Escuela Politécnica Superior de Linares

Grado en Ingeniería Telemática

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Linares

Trabajo Fin de Grado

CREACIÓN DE BASES DE DATOS DESDE UN DISPOSITIVO MÓVIL

Alumno: **Daniel Soria Martínez**

Tutor : Prof. D. Ángel Inocencio Aguilera García
Departamento : Informática

Junio, 2016

Universidad de Jaén
Escuela Politécnica Superior de Linares

Trabajo Fin de Grado

CREACIÓN DE BASES DE DATOS DESDE UN DISPOSITIVO MÓVIL

Alumno: Daniel Soria Martínez

Tutor: Prof. D. Ángel Inocencio Aguilera García

Departamento: Informática

Firma Alumno

Firma Tutor

NOMBRE AGUILERA GARCIA ANGEL INOCENCIO - NIF 52523298S

Firmado digitalmente por NOMBRE AGUILERA GARCIA ANGEL INOCENCIO - NIF 52523298S
Nombre de reconocimiento (DN): c=ES, o=FNMT, ou=FNMT Clase 2 CA, ou=501072765, cn=NOMBRE AGUILERA GARCIA ANGEL INOCENCIO - NIF 52523298S
Fecha: 2016.06.09 11:12:33 +02'00'

Junio, 2016

RESUMEN

El siguiente proyecto trata sobre el desarrollo de una aplicación para dispositivos móviles con sistema operativo Android cuyo nombre será 'MySQL Lite Mobile'. Debido al auge de este sistema en los últimos años han surgido multitud de aplicaciones por la necesidad y comodidad de realizar una tarea determinada desde un medio móvil en lugar de uno estático, es de hecho también así en este caso en el que la aplicación a desarrollar pretende abordar la posibilidad de gestionar bases de datos desde un dispositivo móvil, junto con las tablas que compongan a estas bases de datos, ubicadas tanto en servidores remotos como locales como las que se almacenen dentro del mismo dispositivo. Se entenderá de aquí en adelante por gestionar bases de datos como la capacidad de crear y borrar bases de datos y tablas dentro de estas así como insertar, borrar o modificar registros y propiedades dentro de estas tablas, entre otras funcionalidades. Además de discutir diferentes aspectos sobre la aplicación, también se contemplará la configuración de un servidor con capacidades de almacenamiento de bases de datos y acceso remoto y local seguro. Se incluirá más adelante un análisis de todas las tareas realizadas para el desarrollo y diseño de este proyecto.

ABSTRACT

The project is about developing an application for mobile devices with Android operating system whose name is 'MySQL Lite Mobile'. Due to the rise of this system in recent years there have been many applications the need and convenience to perform a certain task from a mobile device rather than a static one, in fact, the application to develop aims to address the ability to manage databases from a mobile device, along with tables that make up these databases, located in both local and remote servers as they are stored within the same device. It will be understood hereinafter to manage databases as the ability to create and delete databases and tables within these as well as insert, delete or modify records and properties within these tables, among other features. Besides discussing different aspects of the application, server configuration is also contemplated with storage capacities of databases and access secure remote location. An analysis of all work done for the development and design of this project is later included.

ÍNDICE DE CONTENIDOS

Índice de Figuras 7

Índice de Tablas 10

Índice de Siglas y Acrónimos 10

1. INTRODUCCIÓN 12

1.1 Actualidad 12

1.1.1 Smartphones y Android 13

1.1.2 Bases de datos 15

1.2 Motivación 19

1.3 Idea general 19

1.4 Arquitectura del sistema 20

1.5 Estructura de la memoria 21

2. OBJETIVOS 23

2.1 Objetivos de proyecto 23

2.2 Objetivos docentes 24

3. MATERIALES Y MÉTODOS. 25

3.1 Antecedentes 25

3.1.1 Evolución de la telefonía móvil 25

3.1.2 Evolución de las plataformas para dispositivos móviles 27

3.1.3 Historia de las bases de datos 30

3.1.4 Aplicaciones similares en el mercado 32

3.2 Estado del arte tecnológico 33

3.2.1 Plataforma Android 33

3.2.2 Sistemas de Gestión de Bases de Datos 51

3.2.3 IDE Eclipse 56

3.2.4 Servidor Apache Tomcat 57

3.3 Estado del arte legislativo 58

3.4 Análisis del sistema 58
3.4.1 Definición del sistema 58
3.4.1 Definición de requisitos funcionales 61
3.4.2 Definición de requisitos no funcionales 62
3.5 Diseño del sistema 67
3.5.1 Descripción de la arquitectura del sistema 67
3.5.2 Diseño de la lógica de negocio 71
3.5.3 Diseño de la lógica de datos 146
3.5.4 Diseño de la interfaz 150
4. RESULTADOS Y DISCUSIÓN 170
4.1 Simulación de la aplicación 170
4.2 Batería de pruebas 177
4.3 Encuesta a usuarios 179
5. CONCLUSIONES 181
5.1 Conclusiones generales 181
5.2 Trabajo futuro 182
6. ANEXO 1: INSTALACIÓN DEL SOFTWARE 183
6.1 Instalación de servidor 183
6.2 Instalación del cliente 185
7. ANEXO 2: MANUAL DE USUARIO 186
7.1 Inicio de la aplicación 186
7.2 Formulario de registro 188
7.3 Inicio login alternativo 188
7.4 Listar bases de datos 189
7.5 Formulario para crear bases de datos 192
7.6 Listar tablas en dispositivo y en servidor 193
7.7 Log de sentencias SQL 196

7.8 Crear tablas 197
7.9 Mostrar tabla 199
7.10 Insertar registros 200
7.11 Agregar campos 201
7.12 Edición de campos y registros 202
7.13 Claves foráneas 205
8. ANEXO 3: ESTUDIO ECONÓMICO Y PLANIFICACIÓN 207
8.1 Estudio económico 207
8.1.1 Costes materiales 207
8.1.2 Costes de personal 208
8.1.3 Costes totales 208
8.2 Planificación temporal 209
9. BIBLIOGRAFÍA 213

Índice de Figuras

Figura 1.1 Cuota de mercado de sistemas operativos en España [2] 14
Figura 1.2 Principales usos de smartphones [3] 15
Figura 1.3 Ejemplo de tabla 16
Figura 1.4 Sistema de Gestión de Bases de Datos 16
Figura 1.5 Arquitectura del sistema [6] [7] [8] [9] [43] 21
Figura 3.1 Cuota de mercado móvil en España [18] 30
Figura 3.2 Arquitectura Android [25] 39
Figura 3.3 Versiones de Android [28] 44
Figura 3.4 Carpeta /app/src/main/java [30] 46
Figura 3.5 Carpeta /app/src/main/res/ [30] 47
Figura 3.6 Carpeta /app/src/main/res/ [30] 48
Figura 3.7 Estructura general de un proyecto Android [30] 48
Figura 3.8 Arquitectura servidor alternativo 69
Figura 3.9 Arquitectura servidor www.servidormysql.no-ip.org 70

Figura 3.10 Arquitectura servidor móvil 70
Figura 3.11 Diagrama de clases de uso parte 1 72
Figura 3.12 Diagrama de clases de uso parte 2 73
Figura 3.13 Diagrama de clases UML parte 1 93
Figura 3.14 Diagrama de clases UML parte 2 94
Figura 3.15 Diagrama de clases UML parte 3 95
Figura 3.16 Interfaz de *FormularioRegistro.jsp* 96
Figura 3.17 Interfaz de *BasesDeDatosMYSQL.jsp* 99
Figura 3.18 Generación de claves RSA 101
Figura 3.19 Configuración de puertos en router 102
Figura 3.20 Generación de certificado SSL 103
Figura 3.21 Modelo Entidad/Relación 147
Figura 3.22 Interfaz de *Login.java* 151
Figura 3.23 Interfaz de *FormularioRegistro.jsp* 151
Figura 3.24 Interfaz de *LoginAlternativo.java* 153
Figura 3.25 Interfaz de *ListarBasesDeDatos.java* 155
Figura 3.26 Interfaz de *BasesDeDatos.jsp* 156
Figura 3.27 Interfaz de *ListarTablasMovil.java* 159
Figura 3.28 Interfaz de *ListarTablas.java* y *ListarTablasMovil.java* 160
Figura 3.29 Interfaz de *LogSQL.java y LogSQLMovil.java* 161
Figura 3.30 Interfaz de *CrearTablas.java* 162
Figura 3.31 Interfaz de *CrearTablasMovil.java* 163
Figura 3.32 Interfaz de *MostrarTabla.java y MostrarTablaMovil.java* 164
Figura 3.33 Interfaz de *InsertarRegistros.java y InsertarRegistrosMovil.java* 165
Figura 3.34 Interfaz de *AgregarCampos.java* y *AgregarCamposMovil.java* 166
Figura 3.35 Interfaz de *Editar.java* (Edición de campos) 167
Figura 3.36 Interfaz de *EditarMovil.java* (Edición de campos) 167
Figura 3.37 Interfaz de *Editar.java* (Edición de claves primarias) 167
Figura 3.38 Interfaz de *Editar.java* y *EditarMovil.java* (Edición de registros) 168
Figura 3.39 Interfaz de *Editar.java* y *EditarMovil.java* (Edición de valor) 168
Figura 3.40 Interfaz de *ClavesForaneas.java* 169
Figura 4.1 Login lista de bases de datos 171
Figura 4.2 Creación y eliminación de bases de datos 171
Figura 4.3 Lista de tablas 172

Figura 4.4 Renombrado y eliminación de tablas 173
Figura 4.5 Crear tabla y agregar campo 173
Figura 4.6 Insertar registros y visualizar tabla 174
Figura 4.7 Ordenar y seleccionar registros 175
Figura 4.8 Eliminación de registros 175
Figura 4.9 Modificación de valores 176
Figura 4.10 Modificación y eliminación de campos 176
Figura 4.11 Errores de conexión e inicio de sesión 177
Figura 4.12 Error acceso denegado 178
Figura 4.13 Errores de insertar registros 179
Figura 6.1 Instalación de un servidor MySQL [45] 184
Figura 6.2 Interfaz MySQL WorkBench [46] 184
Figura 6.3 Instalación aplicación cliente ProyectoTFG o MySQL Lite Mobile 185
Figura 7.1 Icono MySQL Lite Mobile 186
Figura 7.2 Inicio de aplicación 187
Figura 7.3 Formulario de registro 188
Figura 7.4 Inicio de login alternativo 189
Figura 7.5 Lista bases de datos 191
Figura 7.6 Interfaz de *BasesDeDatos.jsp* 192
Figura 7.7 Lista tablas dispositivo móvil 195
Figura 7.8 Lista tablas servidor MySQL 196
Figura 7.9 Sentencias SQL 197
Figura 7.10 Crear tablas en servidor 198
Figura 7.11 Crear tablas en dispositivo 199
Figura 7.12 Mostrar tabla 200
Figura 7.13 Insertar Registros 201
Figura 7.14 Agregar campos 202
Figura 7.15 Editar servidor MySQL (Edición de campos) 203
Figura 7.16 Editar dispositivo móvil (Edición de campos) 203
Figura 7.17 Editar (Edición de claves primarias) 204
Figura 7.18 Editar (Edición de registros) 204
Figura 7.19 Editar (Edición de valores) 205
Figura 7.20 Claves foraneás 206
Figura 8.1 Tareas diagrama de Gantt 211

Figura 8.2 Diagrama de Gantt 212

Índice de Tablas

Tabla 3.1 Características de Android [23] 35
Tabla 3.2 SGBD más importantes [37] 55
Tabla 3.3 Resumen clases java 104
Tabla 3.4 Asociación con interfaces 145
Tabla 8.1 Costes materiales 207
Tabla 8.2 Costes de personal 208
Tabla 8.3 Costes totales sin IVA 208
Tabla 8.4 Costes totales con IVA 209

Índice de Siglas y Acrónimos

1G: First Generation (Primera Generación de telefonía móvil)

2G: Second Generation (Segunda Generación de telefonía móvil)

3G: Third Generation (Tercera Generación de telefonía móvil)

4G: Fourth Generation (Cuarta Generación de telefonía móvil)

ADT: Android Development Tools (Herramientas para Desarrollo de Android)

API: Application Programming Interface (Interfaz de Programación de Aplicaciones)

CDMA: Code Division Multiple Access (Acceso Múltiple por División de Código)

CPU: Central Processing Unit (Unidad de Procesamiento Central)

ECJ: Eclipse Compiler for Java (Compilador de Java para Eclipse)

EMS: Enhanced Messaging Services (Servicio de Mensajería Mejorado)

ETCS: European Credit Transfer And Accumulation System (Sistema Europeo de Transferencia y Acumulación de Créditos)

GNU: GNU Not Unix (GNU No es Unix)

GPL: General Public License (Licencia Pública General)

GSM: Global System Mobile (Sistema Global Para Las Comunicaciones Móviles)

HTML: HyperText Markup Language (Lenguaje de Marcas de Hipertexto)

HTTP: Hypertext Transfer Protocol (Protocolo para la Transferencia de Hipertexto)

HTTPS: Hypertext Transfer Protocol Secure (Protocolo para la Transferencia de Hipertexto Seguro)

IDE: Integrated Development Environment (Entorno de Desarrollo Integrado)

IP: Internet Protocol (Protocolo de Internet)

JDBC: Java Database Connectivity (Conectividad Base de Datos y Java)

JDT: Java Developments Tools (Herramientas para el Desarrollo de Java)

JSP: Java Server Pages (Páginas de Servidor de Java)

LTE: Long Term Evolution (Evolución a Largo Plazo)

MIMO: Multiple Input Multiple Output (Múltiple Entrada Múltiple Salida)

MMS: Multimedia Messaging Service (Servicio de Mensajería Multimedia)

OFDM: Orthogonal Frequency Division Multiplexing (Multiplexación por División de Frecuencias Ortogonales)

OTI: Object Technology International (Tecnología Internacional de Objetos)

RCP: Rich Client Platform (Plataforma para Cliente Enriquecido)

RIM: Research In Motion (Investigación En Marcha)

SGBD: Database Management System (Sistema de Gestión de Bases de Datos)

SMS: Short Message Service (Servicio de Mensajes Cortos)

SSH: Secure SHell (Intérprete de Órdenes Seguro)

SSL: Transport Layer Security (Seguridad de la Capa de Transporte)

SWT: Standard Widget Toolkit (

SO: Operating System (Sistema Operativo)

SQL: Structured Query Language (Lenguaje de Consultas Estructurado)

TDMA: Time Division Multiple Access (Acceso Múltiple por División de Tiempo)

UML: Unified Modeling Language (Lenguaje Unificado de Modelado)

UMTS: Universal Mobile Telecommunications System (Sistema Universal de Telecomunicaciones Móviles)

USB: Universal Serial Bus (Bus Universal en Serie)

Wi-Fi: Wireless Fidelity (Fidelidad Inalámbrica)

1. INTRODUCCIÓN

La presente memoria contemplará todos los aspectos del proyecto de fin de grado necesarios para poder completar la enseñanza del grado en Ingeniería Telemática. El proyecto a desarrollar trata sobre la construcción de una herramienta destinada a la correcta gestión de bases de datos desde un dispositivo móvil, almacenadas en un servidor remoto o en el mismo dispositivo, de una forma eficaz, intuitiva y segura. Se entiende en este contexto gestionar como la capacidad de crear y borrar bases de datos y tablas dentro de estas así como insertar, borrar o modificar registros y propiedades dentro de estas tablas, entre otras funcionalidades. Además de la aplicación móvil se presentará una posible configuración de un servidor con capacidades de almacenamiento de bases de datos y acceso remoto y local, útil para poder comprobar el correcto funcionamiento de la misma.

Teniendo en cuenta que en la actualidad el sistema operativo Android es el más utilizado por los dispositivos móviles y el más aceptado en el mercado, la aplicación a desarrollar irá destinada al uso en esta plataforma ya que, además permite mediante sencillas interfaces la creación de todas las funcionalidades necesarias para su correcto funcionamiento.

1.1 Actualidad

A continuación se presentará detalladamente cuál es la situación en la que se encuentran las tecnologías que se utilizan en este proyecto. Más concretamente se estudiará el uso de los smartphones en la actualidad, centrado en los smartphone que utilizan la plataforma Android, y el uso de bases de datos en las aplicaciones que son utilizadas por los usuarios diariamente.

1.1.1 Smartphones y Android

El avance tecnológico surgido en los últimos años ha permitido la creación y utilización de dispositivos móviles inteligentes o smartphones, dando lugar a una sociedad en la que casi cualquier persona en casi cualquier momento y lugar puede acceder a la red de redes también conocida como Internet y a los servicios que esta ofrece. El mercado actual no para de abastecerse en cuanto a la venta de smartphone, tal es así este hecho que llega a ser complicado encontrar una sola persona en nuestra sociedad que no haya escuchado acerca de ellos. Según datos recogidos por Ditrendia en España un 87% de la población española cuenta con algún tipo de smartphone actualmente [1]

Entre las tecnologías que se desarrollan actualmente para dispositivos móviles merece destacarse el uso de la plataforma Android como principal competidor de mercado en cuanto a sistemas operativos destinados a estos dispositivos. Recientes estudios de mercado sitúan a Android como el sistema operativo de dispositivos móviles más utilizado en España con un aplastante 86,3%**,** dato que puede apreciarse en la **Figura 1.1**

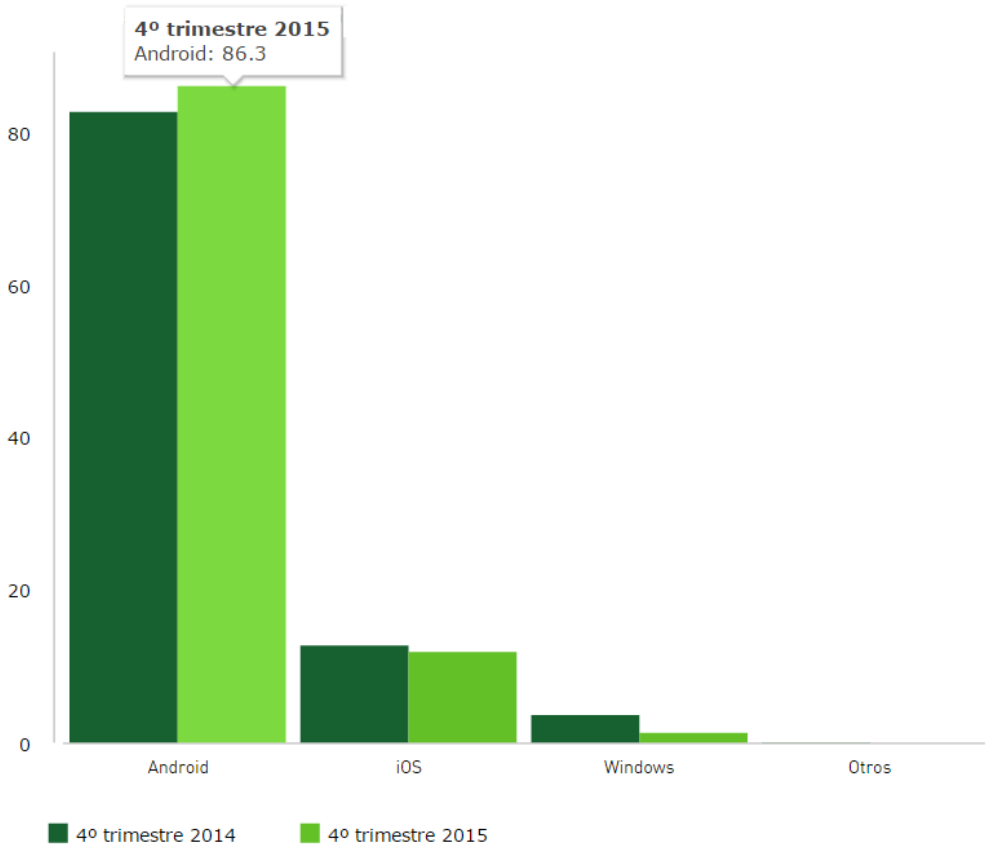


Figura 1.1 Cuota de mercado de sistemas operativos en España [2]

Son muchas las ventajas y utilidades que ofrecen los smartphones a sus usuarios, de forma resumida se mostrarán a continuación algunas de las más importantes y utilizadas a continuación y que aparecen resumidas en la **Figura 1.2**.

1. Los usuarios poseen una herramienta capaz con capacidad de conexión a Internet en casi cualquier lugar, esto desemboca en resumen en una capacidad de utilizar un gran número de servicios se encuentre donde se encuentre el usuario tales como el fácil acceso a una información deseada o la realización de trámites y formularios electrónicos.

2. Las empresas pueden aprovechar la gran cuota de mercado y la accesibilidad que proporcionan estos dispositivos a cualquier servicio deseado para publicitar sus productos y establecer una mejor conexión con sus clientes.

3. El uso de Interfaces de Programación de Aplicaciones o API, entendiendo estas como un conjunto de funciones que facilitan el intercambio de datos entre diferentes aplicaciones, ayudan en gran medida a facilitar el aprendizaje a los desarrolladores interesados en ello.

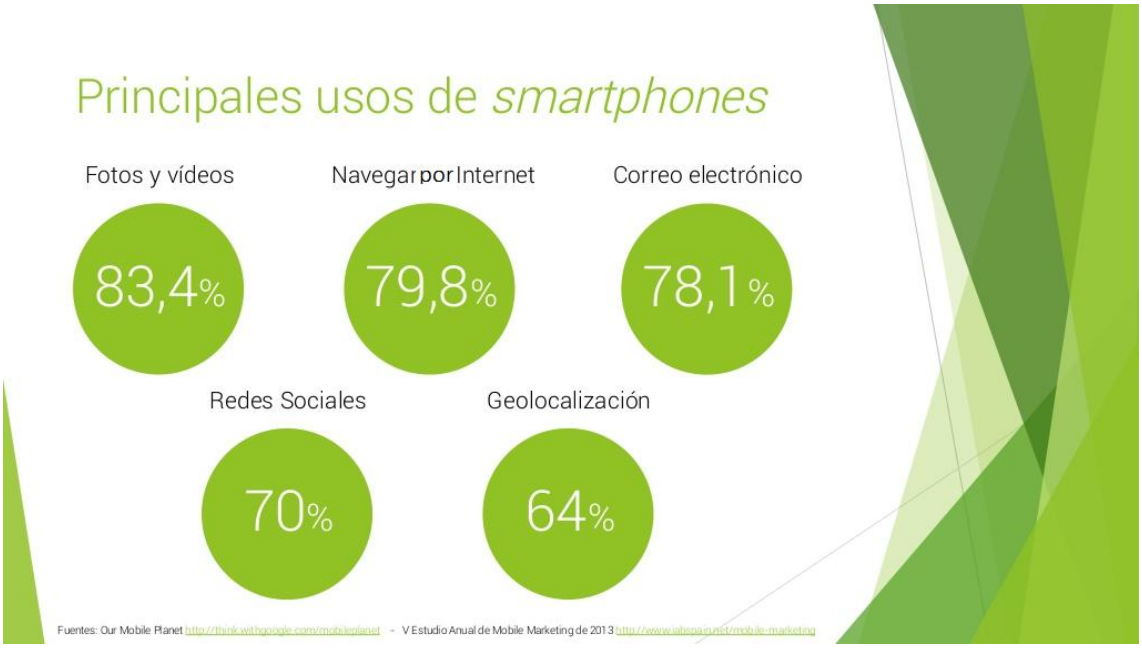


Figura 1.2 Principales usos de smartphones [3]

1.1.2 Bases de datos

Una base de datos puede definirse como un conjunto de información asociada de temática similar. Concretamente esta información es en realidad una o varias tablas que pertenecientes a una única base de datos, las cuales estarán formadas por una cabecera, las cuales a su vez contendrán campos, y unos registros asignados a estos campos, los cuales formarán filas y columnas de registros. Uno o varios de los campos a la vez formarán la clave primaria de la tabla, propiedad por la cual los registros pertenecientes a esa o esas columnas no pueden ser idénticos, por ejemplo el DNI de una persona o un código de identificación. En la **Figura 1.3** puede apreciarse un ejemplo de una tabla con información diversa sobre los empleados de una empresa.

C_Empleado	N_Empleado	Puesto	Sueldo (€)
000001A	Daniel Mudarra Navarro	Administrativo	1500,00
000002A	Martín Martínez Banderas	Administrativo	1500,00
000001B	Santiago Sanz López	Secretario	1200,00
000002B	Marta Rojas Sánchez	Secretaria	1200,00

Cabecera
Filas o Registros
Valores
Campos
Columnas
Clave Primaria

Figura 1.3 Ejemplo de tabla

Entre las principales características de los sistemas de base de datos pueden encontrarse: [5]

- Independencia lógica y física de los datos.
- Redundancia mínima.
- Acceso concurrente por parte de múltiples usuarios.
- Integridad de los datos.
- Consultas complejas optimizadas.
- Seguridad de acceso y auditoría.
- Respaldo y recuperación.
- Acceso a través de lenguajes de programación estándar.

A destacar en cuanto a gestión de base de datos cabe hablar de los llamados Sistemas de Gestión de Base de Datos o SGBD, los cuales son un tipo de software cuyo objetivo es el de servir de interfaz entre la bases de datos y las aplicaciones utilizadas por el usuario.

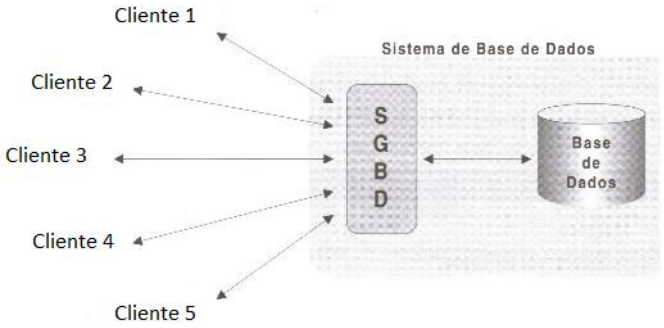


Figura 1.4 Sistema de Gestión de Bases de Datos

Entre las ventajas más importantes en el uso de las bases de datos como medio de almacenamiento de datos pueden encontrarse las siguientes: [5]

1. Control de la redundancia de datos

Mientras que los sistemas de ficheros habituales almacenan varias copias de los mismos datos en diferentes ficheros motivando un desperdicio de espacio de almacenamiento, los sistemas de bases de datos integran estos ficheros de manera que impida la llamada redundancia de datos, no obstante, en ocasiones es necesaria para modelar las relaciones entre los datos.

2. Consistencia de datos

Si un dato está duplicado y el sistema conoce lo reconoce como tal, él mismo puede garantizar la consistencia de todas las copias.

3. Compartir datos

Los sistemas de bases de datos están enfocados a la posibilidad de poder compartir el acceso a bases de datos y tablas entre los usuarios.

4. Asegura integridad de datos

La integridad en una base de datos se refiere a la consistencia y validez de la información almacenada. El método para asegurar dicha integridad se basa en el uso de restricciones y reglas que el mismo SGBD debe ser capaz de mantenerlas.

5. Seguridad aplicada

A través del uso de credenciales los sistemas de bases de datos impiden el acceso no autorizado a la información.

6. Buena accesibilidad a los datos

Muchos SGBD proporcionan lenguajes de consultas que permiten al usuario hacer cualquier tipo de consulta sobre los datos de una manera cómoda

7. Mejora en la productividad

El SGBD proporciona diversas funciones que el programador necesita para escribir en un sistema de ficheros. El hecho de disponer de estas funciones permite al programador centrarse mejor en la tarea que esté llevando a cabo, sin tener que preocuparse de los detalles de implementación de bajo nivel.

8. Mejora en el mantenimiento

Los SGBD separan las descripciones de los datos de las aplicaciones al contrario de lo que sucede con los sistemas de ficheros normales. Esto es lo que se conoce como independencia de datos, tarea que simplifica el mantenimiento de las aplicaciones que acceden a la base de datos.

9. Concurrencia sin problemas

En algunos sistemas de ficheros, si hay varios usuarios que pueden acceder simultáneamente a un mismo fichero, es posible que el acceso interfiera entre ellos de modo que se pierda información o se pierda la integridad. La mayoría de los SGBD gestionan el acceso concurrente a la base de datos y garantizan que no ocurran problemas de este tipo.

Existen diferentes tipos de SGBD en el mercado tecnológico tales como Access, Oracle, MySQL y en menor medida SQLite, siendo estos dos últimos los se utilizarán en este proyecto.

Algunos ejemplos del uso y aplicaciones de bases de datos que se dan en la actualidad son los siguientes:

- Bibliotecas que almacenan información sobre sus libros y clientes.
- Instituciones y empresas a menudo almacenan datos de clientes y personal.
- Centros comerciales que necesitan garantizar la privacidad entorno a sus productos, locales, clientes y personal.
- El llamado database marketing, referido al uso de bases de datos (información) enfocados al cliente.
- Comercio online donde se disponen de bases de datos donde se almacenan los productos a la venta.

1.2 Motivación

La principal motivación por parte del alumno que desarrolla este proyecto consiste en la adquisición de conocimientos en el desarrollo de aplicaciones para terminales móviles en una plataforma Android, ya que, como se ha mencionado anteriormente, es la plataforma para dispositivos móviles que mas penetración tiene en el mercado actualmente y para la que, además, en las últimas fechas se han desarrollado y mejorado los entornos de programación ya existentes lo que facilita en gran medida el desarrollo del proyecto. Otros intereses aparte de la adquisición de conocimientos se encuentran en el empleo de tecnologías conocidas por el alumno durante el transcurso del grado tales como el uso del sistema de gestión de bases de datos MySQL.

El hecho de poder desarrollar una tarea o aprovechar un servicio en cualquier lugar gracias a un dispositivo móvil es una ventaja que será aprovechada en este proyecto mediante la posibilidad de poder gestionar una base de datos ubicada en un servidor remoto o local utilizando dicho terminal móvil. Es de destacar la utilidad de esta aplicación en el uso de servicios que utilicen bases de datos y en situaciones en la que el acceso local al servidor MySQL sea imposible temporalmente.

Por último cabe destacar el hecho de que las aplicaciones del mismo tipo no se encuentran en mercado consolidado, de tal manera que esta aplicación tendría posibilidades de hacerse un hueco en el mercado.

1.3 Idea general

La idea que se tiene a grandes rasgos de este proyecto es la de ofrecer a un usuario la capacidad de gestionar la información que tenga almacenada en las bases de datos de un servidor MySQL de manera cómoda y segura desde su dispositivo móvil en cualquier momento y en cualquier lugar, de manera que, por ejemplo, un usuario que mantenga un servicio de compra online y cuyos productos estén almacenados en una base de datos de MySQL pueda eliminar en cualquier momento un producto anteriormente insertado, insertar unos nuevos o bien modificar parámetros de estos productos tales como pudieran ser el precio o la marca del mismo.

La aplicación desarrollada en este proyecto también contempla la posibilidad de almacenar tablas en una única base de datos ubicada en el mismo dispositivo móvil mediante el sistema de gestión de bases de datos SQLite propio de Android y la de hacer posible la transferencia de registros directamente desde estas tablas a unas con el mismo nombre ubicadas en un servidor MySQL remoto y viceversa.

1.4 Arquitectura del sistema

La arquitectura básica del sistema, la cual puede apreciarse en la **Figura 1.5** plantea tres escenarios diferentes aunque similares, es decir, la conexión entre el dispositivo móvil y su propia base de datos, gestionada utilizando SQLite, la conexión realizada entre el dispositivo y un servidor MySQL cualquiera y la efectuada entre el mismo terminal móvil y el servidor privado MySQL con dirección www.servidormysql.no-ip.org.

Las conexiones realizadas con los servidores externos al terminal precisarán de conexión a Internet por parte de estos, mientras que la realizada con la base de datos ubicada en el dispositivo no la precisa ya que se encuentra dentro del mismo terminal.

El modo de operación en las comunicaciones entre el terminal móvil y el servidor privado será casi idéntico al de mismo terminal con un servidor alternativo diferente salvo por los siguientes hechos:

1. La conexión con el servidor casero es siempre segura, esto es debido al hecho de que se han instalado tecnologías de seguridad en el servidor para que las credenciales de usuario y sentencias que él mismo ejecute en las bases de datos del servidor queden cifradas en el trayecto, no obstante, esto no significa que un servidor diferente no pueda realizar las mismas operaciones si tiene la configuración adecuada.

2. Este servidor privado posee servicios de registros de usuario y creación de bases de datos propios que se diferencian del resto de servidores. Estos servicios existen para ofrecer a los usuarios la capacidad de registrarse y de crear bases de datos en el servidor ya que MySQL no aporta un servicio adecuado para el uso que se le da en este caso.

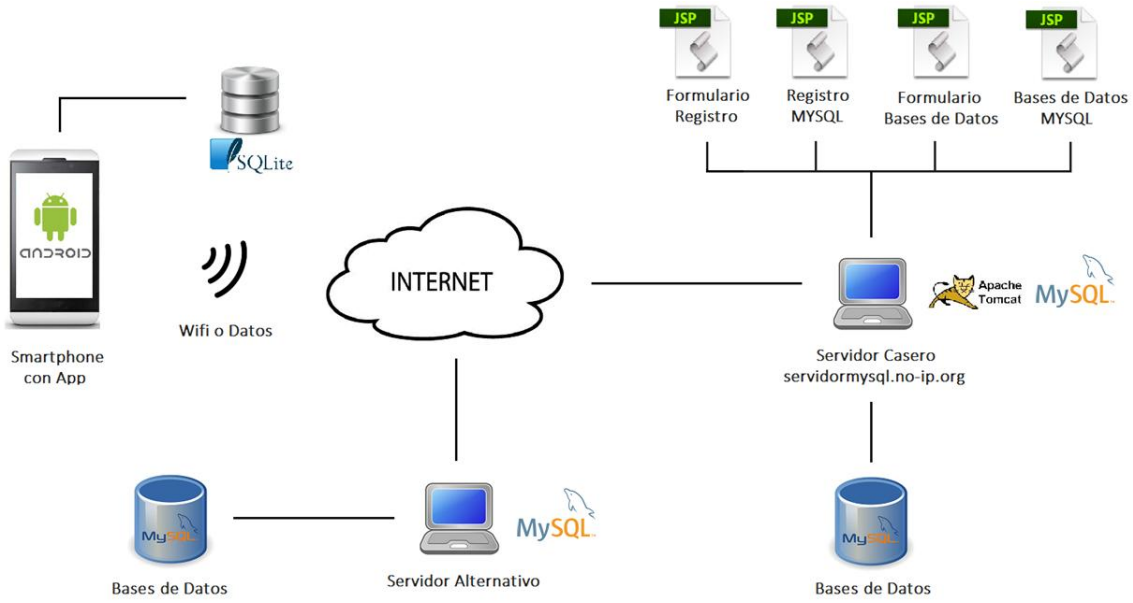


Figura 1.5 Arquitectura del sistema [6] [7] [8] [9] [43]

1.5 Estructura de la memoria

El presente proyecto se encuentra dividido en diferentes estructuras, de diferente contenido temático, que permiten una correcta comprensión del mismo.

La primera de las partes, la cual ya ha aparecido anteriormente, aporta un pequeño resumen dando una visión general del proyecto. A continuación la memoria se centra en la fase de análisis de trabajo donde aparecerán detalles sobre la situación actual de las tecnologías que se emplean en el mismo en la actualidad, las motivaciones por parte del alumno que le han llevado a la realización de este proyecto, una descripción de la idea general del mismo y una descripción de la arquitectura del sistema empleado.

Tras analizar los objetivos docentes y los propios de la aplicación se planteará un análisis de la situación contemporánea en la que se encuentra el proyecto, es decir, sus antecedentes, estado del arte tecnológico y del legislativo.

A continuación aparece información relativa a la fase de desarrollo e implementación del software que tratará en gran medida el sistema de la aplicación, junto con sus requisitos funcionales y no funcionales, un estudio sobre la viabilidad del mismo, teniendo en cuenta el tiempo y costo que ha llevado desarrollarlo.

Una vez acabada la fase de desarrollo aparecerán detalles relativos a la fase de diseño donde se mostrará una descripción profunda del diseño de la aplicación y donde se analizará la arquitectura del sistema a fondo, las tecnologías utilizadas, la lógica de negocio y de datos y el diseño del interfaz para el usuario.

Seguidamente aparecerá una estructura dedicada al uso de la misma aplicación, la cual conlleva ejemplos de uso de la misma y de pruebas realizadas a lo largo del desarrollo del proyecto, contemplado así la fase de pruebas del software.

La memoria concluirá con unos apartados finales propios de cualquier memoria relacionada con la programación, es decir, un apartado de encuestas de usuarios que han probado la aplicación y conclusiones acerca de la misma, con una reflexión añadida sobre cómo podría mejorarse y extenderse la aplicación, unos anexos, en este caso tres en los que aparecerán un manual de usuario y otro sobre como instalar el software necesario para el uso de la aplicación y un estudio económico y temporal que incluirá un diagrama de Gant y un análisis de los costes que ha supuesto el trabajo realizado. Finalmente aparecerá la bibliografía que contendrá las diferentes referencias que han sido necesarias para el desarrollo del proyecto.

2. OBJETIVOS

En el desarrollo del presente proyecto se pretenden conseguir los siguientes objetivos, agrupados según estén relacionados directamente con el uso aplicación en si misma o bien los que sean puramente docentes.

2.1 Objetivos de proyecto

Los objetivos relacionados con el uso de la aplicación por parte de un usuario son los siguientes:

1. Realizar una aplicación para dispositivos móviles que permita al usuario gestionar bases de datos con tecnología MySQL desde un dispositivo móvil, junto con las tablas que compongan a estas bases de datos, ubicadas tanto en servidores remotos como locales como las que se almacenen dentro del mismo dispositivo.

2. Llevando a un caso más práctico el objetivo anterior, se pretende conseguir que un usuario que utilice una aplicación basada en bases de datos, tal como podría ser una tienda online o un registro de biblioteca, pueda gestionar los datos de esta aplicación desde un lugar remoto a través de su dispositivo móvil.

3. Desarrollar una interfaz que sea funcional, eficaz e intuitiva para el usuario que utiliza la aplicación, evitando así la introducción manual de sentencias SQL.

4. Ofrecer un servicio de almacenamiento de bases de datos y de recuperación de los datos de las mismas que sea seguro para el usuario.

5. Permitir al usuario la capacidad de almacenar tablas en su dispositivo móvil ubicadas en servidores externos.

2.2 Objetivos docentes

Los objetivos relacionados con los conocimientos que el alumno espera obtener con la realización de este proyecto son los siguientes:

1. Adquirir conocimientos relativos a la programación para la plataforma Android y java, ya que es el sistema operativo para dispositivos móviles con más demanda en la actualidad y por tanto útil para introducirse en el mundo laboral.

2. Adquirir conocimientos relativos a la elaboración de proyectos relacionados con la programación.

3. Realizar una aplicación propia con fines funcionales para poner a prueba los conocimientos adquiridos durante el transcurso del grado.

4. Aprobar la asignatura Trabajo Fin del Grado de Ingeniería de Telemática, la cual consta de 12 créditos ETCS, traducido en 300 horas de trabajo divididas en 225 de trabajo autónomo y 75 horas presenciales.

3. MATERIALES Y MÉTODOS

En este apartado presenta diversa información de gran importancia acerca del proyecto, concretamente informa de los hechos históricos relacionados con las tecnologías que se utilizan en el mismo así como la situación actual de las mismas. Además se analizará aquí también el marco legal en el que se desarrolla el proyecto. Por último se realizará un análisis del sistema desarrollado, atendiendo a la viabilidad y a los requisitos impuestos así como una descripción de la arquitectura del mismo.

3.1 Antecedentes

A continuación se mostrarán diferentes datos históricos sobre las tecnologías utilizadas, dicha información contendrá la historia, por un lado, de la telefonía móvil hasta la actualidad pasando por el desarrollo de plataformas para dispositivos móviles y culminando en las aplicaciones similares a las de este proyecto que se han fabricado, y por otro de los orígenes de las bases de datos hasta su situación en la actualidad.

3.1.1 Evolución de la telefonía móvil

Un teléfono móvil puede definirse como un dispositivo de pequeño tamaño, portátil, sin hilos ni cables externos que permite mantener conversaciones con otras personas que tengan también otro teléfono móvil, siempre que ambos estén dentro del área de cobertura del servicio que lo facilita. [10]

Los orígenes de los teléfonos móviles se achacan a la Segunda Guerra Mundial donde se vio la necesidad de comunicarse a distancia, hecho que dio lugar a que Motorola crease un equipo militar llamado Handie Talkie H12-16 para comunicaciones vía ondas de radio con banda de frecuencias por debajo de los 600 kHz [11]

Durante los años cincuenta, sesenta y setenta surgieron diferentes prototipos y modelos similares a un teléfono móvil, como el llamado Mobile Telephone System A (MTA) phone en 1955, un artilugio que pesaba 40 kilogramos y que se instalaba en automóviles o el famoso zapatófono, inventado por Martin Cooper en 1973, instrumento con el cual se realizo lo que se considera la primera llamada telefónica de la historia. No fue hasta 1983 la fecha en la que Motorola culminó el proyecto considerado como el primer teléfono móvil de la historia, el Motorola Dynatac 8000x [11], hecho que dio lugar a la fabricación posterior por parte de estas y otras empresas de dispositivos analógicos cada vez más capaces que darían lugar a lo que hoy día se conoce como primera generación de la telefonía móvil o 1G.

Durante la década de los 90 nace la segunda generación de móviles o 2G, cuyo desarrollo tiene como objetivo principal la digitalización de las comunicaciones, ya que estas ofrecen una mejor calidad de voz que las analógicas además de un aumento en el nivel de seguridad simplificando además la fabricación de terminales. Hechos de esta generación a destacar fueron la estandarización del Global System Mobile o GSM, que incluiría el Servicio de Mensajes Cortos o SMS y el llamado roaming, el cual ofrecía la capacidad a un dispositivo de moverse de una zona de cobertura a otra, la implementación por parte de las operadoras telefónicas móviles del Acceso Múltiple por División de Tiempo o TDMA y del Acceso Múltiple por División de Código o CDMA y la posibilidad de utilizar la multiplexación de llamadas, de tal manera que en un canal antes destinado a transmitir una sola conversación a la vez se hizo posible transmitir varias conversaciones de manera simultánea, incrementando así la capacidad operativa y el número de usuarios que podían hacer uso de la red en una misma celda en un momento dado.

La evolución de la tecnología 2G dio lugar al llamado 2.5G. A destacar en esta generación se encuentra el llamado Servicio de Correo Expreso o EMS, el cual es un servicio de mensajería mejorado que permite la inclusión de melodías e iconos dentro del mensaje y el conocido Sistema de Mensajería Multimedia o MMS el cual permitía la inserción de imágenes, sonidos, videos y texto. Para poder prestar estos nuevos servicios se hizo necesaria una mayor velocidad de transferencia de datos lo que dio lugar al surgimiento de las tecnologías GPRS y EDGE.

- GPRS o Servicio General de Paquetes Vía Radio (General Packet Radio Service) que permite velocidades de datos desde 59 kbit/s hasta 120 kbit/s.
- EDGE o Tasas de Datos Mejoradas para la evolución de GSM (Enhanced Data rates for GSM Evolution) permite velocidades de datos de hasta 384 kbit/s.

La tercera generación de telefonía móvil o 3G nace de la necesidad de aumentar la capacidad de transmisión de datos para poder ofrecer servicios como la conexión a Internet desde el móvil, la videoconferencia, la televisión y la descarga de archivos. En este contexto surge el llamado Sistema Universal de Telecomunicaciones Móviles o UMTS, el cual utiliza tecnología CDMA, que permite alcanzar velocidades de entre 144 kbit/s hasta 7.2 Mbit/s.

A raíz de esta última surge la última generación de telefonía móvil conocida hasta el momento, la generación 4, o 4G, la cual ofrece al usuario capacidades de conexión a Internet con un mayor ancho de banda, superiores a los 100 Mbit/s, gracias a la aparición de la tecnología Evolución a Largo Plazo o LTE, que permite, entre muchas otras cosas, la recepción de televisión en alta definición. Dicho sistema está basado completamente en el protocolo IP o Protocolo de Internet e incluye mejoras tales como el uso de MIMO (Múltiples Entradas Múltiples Salidas) y OFDM (Multiplexación por División de Frecuencias Ortogonales). Este servicio lleva ofreciéndose en España desde 2013 y, aunque empezó siendo compatible únicamente para dispositivos de gama alta, actualmente casi todos los dispositivos de media y baja gama pueden disfrutarlo. [12]

3.1.2 Evolución de las plataformas para dispositivos móviles

Las plataformas de dispositivos móviles fue un concepto que empezó a investigarse a finales de la década de los 80, pero no fue hasta 1996 cuando pudo hacerse realidad con la aparición del dispositivo móvil Palm OS 1.0, creado por la compañía Palm que integraba aplicaciones de RIM (traducido como Investigación en Marcha) tales como correo, agenda y memo pad [13]. Algunos de los sistemas operativos más importantes, organizados por la compañía que los creo surgidos a lo largo de la historia se muestran a continuación:

1. Microsoft y Windows Phone

Microsoft lanza en el año 2000 el Pocket PC 2000 (WinCE 3.0) y un año después, este S.O. ya soportaba Messenger y Media Player 8. Tres años después Windows Mobile fue lanzado al mercado, con mejoras y nuevas aplicaciones tales como Microsoft Outlook, Internet Explorer, Word, Excel, Windows Media Player entre otras características). En 2009 dicho sistema fue renombrado como Windows Phone. [13] Actualmente compite en el mercado móvil pero con un éxito muy por debajo de sus competidores Android y iPhone.

2. BlackBerry desarrollada por BlackBerry

El primer dispositivo de la familia fue la BlackBerry 850 comercializado en 1999, dicho dispositivo tenía un teclado completo lo que era inusual en ese momento. Podía enviar mensajes, acceder al correo electrónico, recibir páginas de Internet completas e implementaba una agenda para organizar tareas, con tan solo una pequeña pantalla que podía mostrar ocho líneas de texto. Dicho sistema fue evolucionando a lo largo de los años, los acontecimientos más importantes en su evolución se dieron en 2003 cuando apareció el RIM 850 y 857, dispositivo con mejora capacidad de conectarse a una red inalámbrica, mensajería, fax por Internet y que, a diferencia del anterior, podía utilizarse como teléfono móvil. No fue hasta 2008 cuando la compañía logró una alta penetración en el mercado gracias a sus dispositivos móviles con capacidad de acceso a Internet a través de satélite, aunque en la actualidad su éxito haya decaído debido al dominio de Android y iPhone en el mercado. [14]

3. Symbian

Symbian es un sistema operativo propiedad de Nokia desde 2008 y que en el pasado fue producto de la alianza de varias empresas de telefonía móvil entre las que se encontraban Nokia, Sony Ericsson, Samsung, Siemens y Motorola entre muchas otras, surgido en 1999 con el objetivo de competir con los SO de Palm, Windows Mobile y Android. Aunque tuvo bastante éxito en el mercado entre los años 2006 y 2009 actualmente no cuentan con el que anteriormente tuvo y este mismo año dejará de ofrecerse soporte a los dispositivos que cuenten con esta plataforma. [15]

4. iPhone y Apple

iPhone es una línea de teléfonos inteligentes diseñada y comercializada por Apple Inc. Ejecuta el sistema operativo móvil iOS antes conocido como iPhone OS. Los orígenes del teléfono iPhone se sitúan en 1983, Apple estaba diseñando el Apple Phone, un teléfono inteligente con pantalla táctil, algo que ningún dispositivo móvil de aquella época llevaba pero Apple decidió no lanzarlo al mercado por razones desconocidas. No fue hasta 2005 cuando Apple junto con Motorola sacaron el Motorola ROKR E1 al mercado, el cual fue el primer teléfono capaz de usar iTunes. Desde 2007 Apple ha fabricado e introducido al mercado un teléfono móvil cada año hasta 2015, cada uno con capacidades de memoria RAM y acceso a velocidad de datos más elevadas, aplicaciones, calidad de pantalla, sensores y procesador mejores que sus antecesores. [16] Actualmente compite con Android en el mercado de teléfonos móviles, aunque es superado ampliamente en cuanto a número de usuarios, sobre todo en España.

5. Android

Android es un sistema operativo basado en Linux diseñado principalmente para dispositivos móviles con pantalla táctil, como teléfonos inteligentes y tablets y en menor medida para relojes inteligentes, televisores y automóviles. Sus inicios tienen lugar en 2003 con la fundación de Android Inc, empresa que Google respaldó económicamente y más tarde en 2005 compró. El primer dispositivo móvil con sistema operativo Android, con nombre Android 1.0 Apple Pie, fue el HTC Dream y se vendió en 2008. [17] Hasta la fecha se han puesto en el mercado unas 15 versiones diferentes de este sistema operativo, cada una con diferentes subversiones surgidas a menudo para solucionar diferentes errores de sistema entre a otros cambios menos importantes. En la actualidad y desde hace algunos años los dispositivos con plataforma Android se venden en mayor cantidad que los dispositivos con Windows Phone y iPhone juntos. [17]

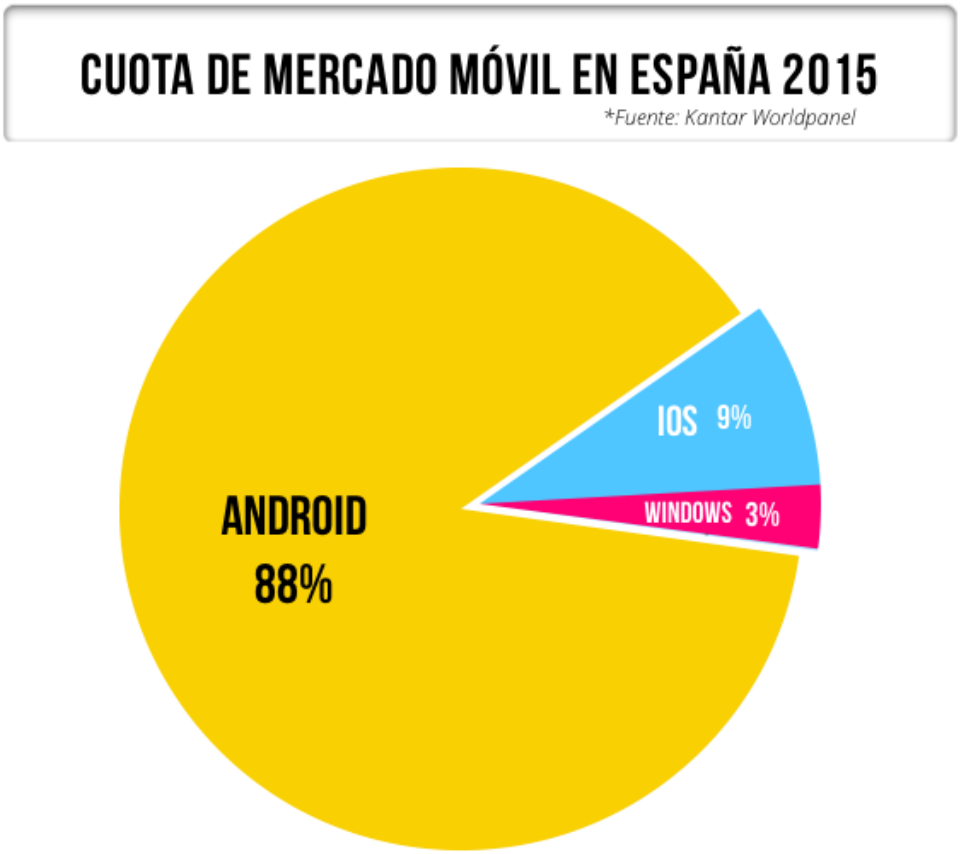


Figura 3.1 Cuota de mercado móvil en España [18]

3.1.3 Historia de las bases de datos

Las base de datos son una herramienta indispensable en la actual sociedad de la información, su utilidad no sólo se debe a que es un conjunto de datos almacenados de alguna forma determinada ya que en una BD también existen una cantidad de elementos que ayudan a organizar sistemáticamente, relacionar, proteger, y administrar de manera eficiente los datos. [19][20]

Desde la antigüedad los diferentes tipos datos han sido registrados por el ser humano en algún tipo de soporte, tales como piedra, papel o en las llamadas tarjetas perforadas. Algunas de las contribuciones más destacables ordenadas cronológicamente, en el campo de almacenamiento de información fueron las siguientes:

1. Las tarjetas perforadas, utilizadas por primera vez en 1884 en una máquina automática y tabuladora muy rudimentaria.

2. A principios de los 50 destaca la aparición de las cintas magnéticas, las cuales son un tipo de soporte de almacenamiento de información donde esta se grababa en pistas sobre una banda plástica con un material magnetizado.

3. En la década de los 60 la bajada de precios que tuvieron los ordenadores dio lugar al uso de discos como medio para almacenar datos. A destacar es el hecho de la aparición del término bases de datos en 1963 y de ahí a la aparición de las primeras generaciones de las bases de datos de red y de las bases de datos jerárquicas con estructura de árbol.

4. En los años 70, un científico informático inglés, llamado Edgar Frank Codd, publica el concepto de base de datos relacionales basadas en establecer relaciones entre los registros pertenecientes a diferentes tablas, además de una serie de reglas para los sistemas de datos relacionales. A raíz de esto nació la segunda generación de los SGBD y el famoso sistema de datos relacional Relational Software System, ahora también conocido como Oracle.

5. En la siguiente década, los 80, se creó un lenguaje de consultas de acceso a bases de datos que permite realizar consultas para recuperar información de interés de una base de datos y realizar cambios de manera sencilla. Oracle, entre otras compañías, comercializaron productos basados en SQL, el cual se acabó convirtiendo en el estándar industrial de las bases de datos relacionales.

6. En los años 90, se investigaron las bases de datos orientadas en objetos las cuales han tenido bastante éxito a la hora de ejecutar datos complejos en los terrenos donde las bases de datos relacionales no han podido desenvolverse de manera eficaz, lo que dio lugar a la aparición de la tercera generación de SGBD. A destacar de las consecuencias que estos hechos tuvieron fueron la creación de herramientas como el Excel y Access, la capacidad de SQL de trabajar en relación a XML y, gracias al nacimiento del World Wide Web o WWW, una forma de consultar las bases de datos remotas mucho más sencilla y estandarizada.

   Actualmente IBM, Microsoft y Oracle dominan el negocio de las bases de datos.

3.1.4 Aplicaciones similares en el mercado

En la actualidad existen cierta cantidad de aplicaciones para dispositivos móviles similares a la desarrollada en este proyecto. Ya que esta posee capacidades tanto para la gestión de bases de datos remotas basadas en MySQL como para una base de datos única y personal en el dispositivo basada en SQLite, se propondrán de ejemplo a continuación algunas de cada tecnología, teniendo más consideración por las basadas en MySQL ya que se presuponen más similares en arquitectura y finalidad a la aplicación desarrollada en este proyecto. [21]

3.1.4.1 Aplicaciones basadas en MySQL

1. Connect2SQL: Aplicación de pago y una de las más completas del mercado con capacidad de conectarse a servidores de bases de datos basados en MySQL, SQL Server (Microsoft SQL), Sybase y PostgreSQL. Funciona con introducción manual de sentencias.

2. MySQL manager: Aplicación de pago, destaca por su interfaz sencilla, completa e intuitiva. Funciona con introducción manual de sentencias.

3. Mobile MySQL Manager: Aplicación con versiones de pago y gratuita, destaca por ofrecer una gran cantidad de funcionalidades a pesar de no utilizar introducción manual de las sentencias, es decir, toda acción se realiza desde la interfaz. Aporta seguridad SSH (Secure SHell) a las conexiones. La única diferencia entre la versión de pago y la gratuita consiste en un método de acceso a la aplicación por contraseña al que solo es posible acceder en la versión de pago.

4. MySQL Client: Aplicación gratuita basado utilizable a través de interfaz y no mediante introducción de código de forman manual. A pesar de no contar con un interfaz llamativo ni con demasiadas funcionalidades cuenta con las más básicas y con la capacidad de realizar conexiones seguras mediante SSL (Secure Sockets Layer).

3.1.4.2 Aplicaciones basadas en SQLite

1. SQLite Editor: Aplicación de pago que permite editar y eliminar cualquier base de datos en el dispositivo. Para los usuarios root permite la edición todas las bases de datos asociadas a aplicaciones instaladas. Funciona mediante uso directo del interfaz sin necesidad de insertar código SQLite manualmente.

2. SQLite Master: Aplicación con versiones de pago y gratuita similar a la anterior que permite no solo editar sino también crear y borrar bases de datos del dispositivo. Para los usuarios root permite la edición todas las bases de datos asociadas a aplicaciones instaladas. La versión de pago aporta servicios destinados a usuarios root y la exportación de tablas a archivos.

3.2 Estado del arte tecnológico

En este apartado se pondrá de manifiesto información diversa acerca de las tecnologías utilizadas en este proyecto con el objetivo principal de dar a entender el motivo por el que se han utilizado tales tecnologías. Se describirán a continuación distintos tipos de información asociados a la aplicación, concretamente acerca de la plataforma Android tales como características, arquitectura, versiones y el lenguaje utilizado para el desarrollo de aplicaciones. Más tarde aparecerán datos utilizados en la aplicación servidora como el software, el SGBD y el IDE con el que se trabaja.

3.2.1 Plataforma Android

Android es una plataforma de software destinada a dispositivos móviles como tablets, teléfonos o relojes inteligentes entre otros. Incluye su propio sistema operativo y aplicaciones por defecto. Se caracteriza por estar basado en Linux, ser desarrollado desde la ideología OpenSource, es decir código abierto, y ser gratuito sin necesidad de pago de licencias. Fue desarrollada por Open Handset Alliance y pertenece a Google desde 2005. Desarrolladores o cualquier persona interesada pueden crear aplicaciones usando el SDK que Android proporciona [22]. A continuación se estudiará a fondo dicha plataforma con los detalles anteriormente mencionados.

3.2.1.1 Características

A continuación se muestra una tabla que contiene las características más importantes de la plataforma Android.

Diseño de dispositivo	Provee adaptación a pantallas de mayor resolución, VGA, biblioteca de gráficos 2D y 3D basada en las especificaciones de la OpenGL ES 2.0 y diseño de teléfonos tradicionales.
Almacenamiento	Herramienta SQLite, una base de datos liviana que es usada para propósitos de almacenamiento de los datos de las aplicaciones.
Conectividad	Android soporta las siguientes tecnologías relacionadas con la conectividad: GSM/EDGE, IDEN, CDMA, EVDO, UMTS, Bluetooth,WiFi, LTE, HSDPA, HSPA+, NFC y WiMAX, GPRS,UMTS y HSDPA+.
Mensajería	SMS y MMS son formas de mensajería tradicionales, y ahora la Android Cloud to Device Messaging Framework (C2DM) es parte del servicio de Push Messaging de Android.
Navegador web	El navegador web incluido en Android está basado en el motor de renderizado de código abierto WebKit, emparejado con el motor JavaScript V8 de Google Chrome.
Soporte de Java	Dalvik es una máquina virtual Java especializada diseñada específicamente para Android y optimizada para dispositivos móviles, los cuales tienen memoria y procesador limitados.
Soporte multimedia	Android soporta los siguientes formatos multimedia: AAC, WebM, H.263, H.264 (en 3GP o MP4), MPEG-4 SP, AMR, AMR-WB (en un contenedor 3GP), HE-AAC (en contenedores MP4 o 3GP), MP3, MIDI, Ogg, Vorbis, WAV, JPEG, PNG, GIF y BMP.
Soporte para streaming	Streaming RTP/RTSP (3GPP PSS, ISMA), descarga progresiva de HTML (HTML5). Adobe Flash Streaming (RTMP) es soportado mediante el Adobe Flash Player.

Soporte para hardware adicional	Android soporta cámaras de fotos, de vídeo, pantallas táctiles, GPS, acelerómetros, giroscopios, magnetómetros, sensores de proximidad y de presión, sensores de luz, gamepad, termómetro y aceleración por GPU 2D y 3D.
Entorno de desarrollo	Inicialmente el entorno de desarrollo integrado utilizado era Eclipse con el plugin de Herramientas de Desarrollo de Android o ADT. Ahora se considera como entorno oficial Android Studio. Este último incluye un emulador de dispositivos, herramientas para depuración de memoria y análisis del rendimiento del software.
Google Play	Google Play es un catálogo de aplicaciones gratuitas y de pago en el que pueden ser descargadas e instaladas en dispositivos Android sin la necesidad de un PC.
Multi-táctil	Android tiene soporte nativo para pantallas capacitivas con soporte multi-táctil.
Bluetooth	El soporte para A2DF y AVRCP fue agregado en la versión 1.5 el envío de archivos (OPP) y la exploración del directorio telefónico fueron agregados en la versión 2.0 y el marcado por voz junto con el envío de contactos entre teléfonos lo fueron en la versión 2.2.
Videollamada	Android soporta videollamada a través de Hangouts.
Multitarea	Multitarea real de aplicaciones está disponible, es decir, las aplicaciones que no estén ejecutándose en primer plano reciben ciclos de reloj.
Características basadas en voz	La búsqueda en Google a través de voz está disponible desde la versión inicial del sistema.
Tethering	Android soporta tethering, una tecnología que permite al teléfono ser usado como un punto de acceso por cable o inalámbrico.

Tabla 3.1 Características de Android [23]

3.2.1.2 Arquitectura

Android es una plataforma para dispositivos móviles que contiene una pila de software donde se incluye un sistema operativo, un middleware y aplicaciones básicas para el usuario. A continuación se desglosarán las funciones de cada una de las capas que forman la arquitectura de Android, la cual podrá apreciarse de manera gráfica en la **Figura 3.2**. Cada una de las siguientes capas utiliza servicios ofrecidos por las anteriores y a su vez los suyos propios a las capas de niveles superiores. [24]

1. Aplicaciones: En este nivel se encuentran tanto las incluidas por defecto de Android como aquellas que el usuario vaya añadiendo posteriormente ya sean de terceras empresas o de su propio desarrollo. Todas estas aplicaciones utilizan los servicios, las API y librerías de los demás niveles.

2. Marco de Aplicación: Representa fundamentalmente el conjunto de herramientas de desarrollo de cualquier aplicación. Toda aplicación que se desarrolle para Android, ya sean las propias del dispositivo, las desarrolladas por Google o terceras compañías o incluso las que el propio usuario cree utilizan el mismo conjunto de API y el mismo marco de trabajo representado por este nivel.

   Entre las API más importantes ubicadas aquí pueden encontrarse las siguientes:

   - Activity Manager: Conjunto de APIs que gestionan el ciclo de vida de las aplicaciones en Android.

   - Window Manager: Gestiona las ventanas de las aplicaciones y utiliza la librería Surface Manager.

   - Telephone Manager: Incluye todas las APIs vinculadas a las funcionalidades propias de un teléfono (llamadas, mensajes, etc.).

   - Content Provider: Permite a cualquier aplicación la capacidad de compartir sus datos con las demás aplicaciones de Android. Por ejemplo, gracias a esta API la información de contactos, agenda, mensajes, etc es accesible para otras aplicaciones.

- View System: Proporciona un gran número de elementos para poder construir interfaces de usuario (GUI), como listas, mosaicos, botones, casillas de verificación, ventanas, control de las interfaces mediante teclado, etc. Incluye también algunas vistas estándar para las funcionalidades más frecuentes.

- Location Manager: Posibilita a las aplicaciones la obtención de información de localización y posicionamiento.

- Notification Manager: Mediante el cual las aplicaciones, usando un mismo formato, comunican al usuario eventos que ocurran durante su ejecución tales como una llamada entrante, un mensaje recibido, un aviso de conexión Wi-Fi disponible, la ubicación en un punto determinado, etc. Si llevan asociada alguna acción, en Android denominada Intent, (por ejemplo, atender una llamada recibida) ésta se activa mediante un simple clic.

- XMPP Service: Colección de APIs para utilizar este protocolo de intercambio de mensajes basado en XML.

3. Librerías o bibliotecas: Esta capa se corresponde con las librerías utilizadas por Android. Éstas han sido escritas utilizando C/C++ y proporcionan a Android la mayor parte de sus capacidades más características. Junto al núcleo basado en Linux, estas librerías constituyen el corazón de Android.

Entre las librerías más importantes ubicadas aquí pueden encontrarse las siguientes:

- Librería libc: Incluye todas las cabeceras y funciones según el estándar de lenguaje C. Todas las demás librerías se definen en este lenguaje.

- Librería Surface Manager: Es la encargada de componer los diferentes elementos de navegación de pantalla. Gestiona también las ventanas pertenecientes a las distintas aplicaciones activas en cada momento.

- OpenGL/SL y SGL: Representan las librerías gráficas y por tanto sustentan la capacidad gráfica de Android. OpenGL/SL maneja gráficos en 3D y permite utilizar, en caso de que esté disponible en el propio dispositivo móvil, el hardware encargado de proporcionar gráficos 3D. Por otro lado SGL proporciona gráficos en 2D, por lo que es la librería más habitualmente utilizada por la mayoría de las aplicaciones. Una característica importante de la capacidad gráfica de Android es que es posible desarrollar aplicaciones que combinen gráficos en 3D y 2D.

- Librería Media Libraries: Proporciona todos los códecs necesarios para soportar el contenido multimedia utilizable en Android (vídeo, audio, imágenes estáticas y animadas, etc.)

- FreeType: Permite trabajar de forma rápida y sencilla con distintos tipos de fuentes.

- Librería SSL: Posibilita la utilización de dicho protocolo para establecer comunicaciones seguras.

- Librería SQLite: Permite la creación y gestión de bases de datos relacionales.

- Librería WebKit: Proporciona un motor para las aplicaciones de tipo navegador y forma el núcleo del actual navegador incluido por defecto en la plataforma Android.

4. Entorno de ejecución de Android: Al mismo nivel que las librerías de Android se sitúa el entorno de ejecución. Éste lo constituyen las Core Libraries las cuales son librerías con multitud de clases Java y la máquina visual Dalvik.

5. Núcleo Linux: Android utiliza el núcleo de Linux 2.6 como una capa de abstracción para el hardware disponible en los dispositivos móviles. Esta capa contiene los drivers necesarios para que cualquier componente hardware pueda ser utilizado mediante las llamadas correspondientes.

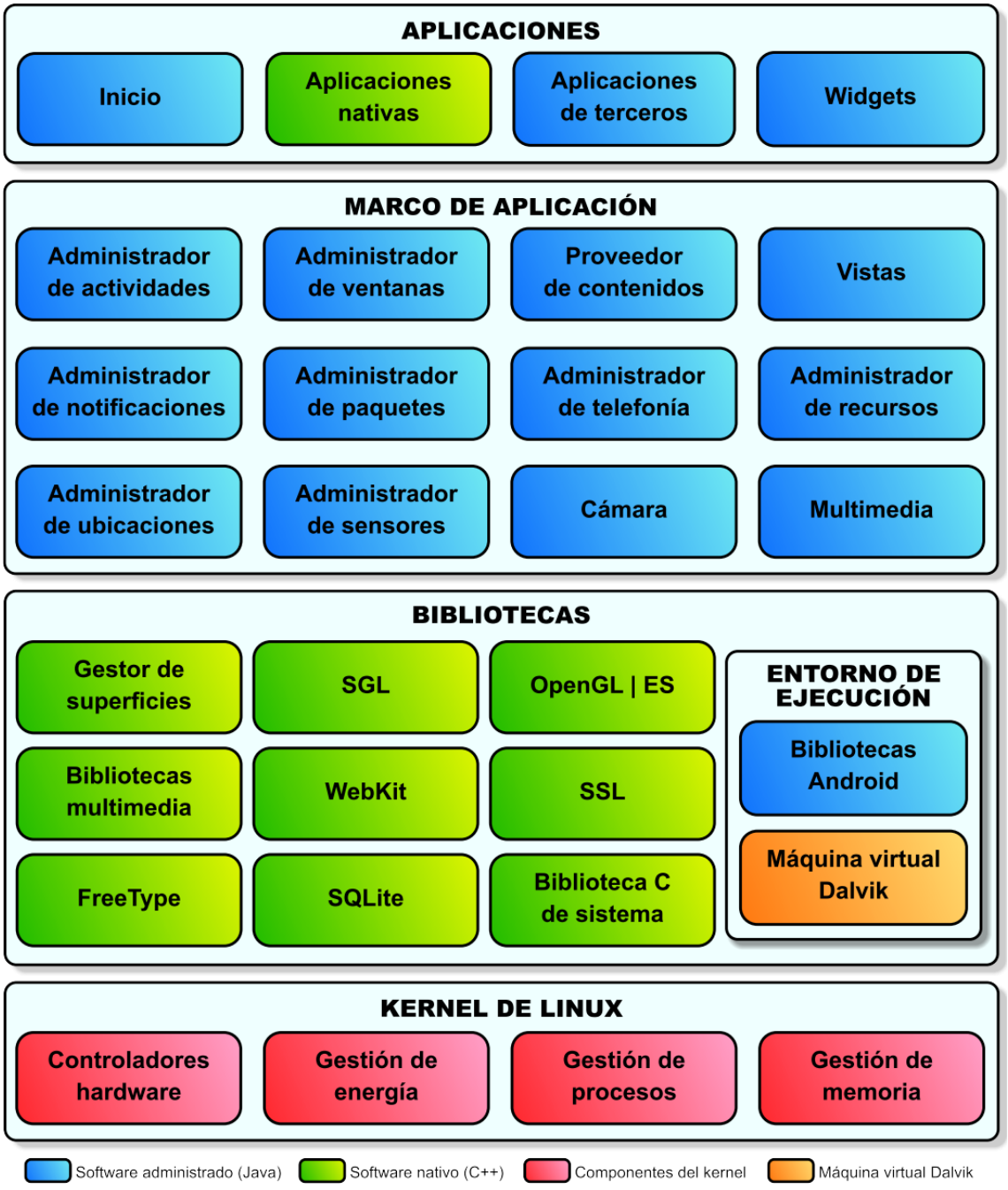


Figura 3.2 Arquitectura Android [25]

3.2.1.3 Evolución de la plataforma

Como ya se ha mencionado anteriormente la historia de Android se remonta al año 2003, fecha en la que el sistema fue desarrollado a modo particular por la compañía Android Inc, una pequeña empresa que en 2005 fue comprada por Google quien continuó el desarrollo del sistema junto con la Open Handset Alliance, un conglomerado de diferentes compañías colaboradoras. En el momento de compra de Android Inc en 2007 Google liberó las versiones de Android conocidas como Alpha y Beta. No eran versiones finales, de hecho ni siquiera había dispositivos comerciales que las utilizasen aunque sí se hicieron públicos los SDK para que la comunidad de desarrolladores empezase a probarlos y a programar aplicaciones para ellos. Por entonces no tenían nombres de dulces asociados a ellas y Google continuó liberando actualizaciones del código hasta aproximadamente septiembre de 2008. [26][27]

A continuación se mostrará un listado de las todas las versiones oficiales de Android, resaltando algunas de sus características más significativas, y sus logos representativos en la **Figura 3.3**.

1. Android 1.0 Apple Pie

Fue la primera versión oficial y definitiva, y encargada de estrenar la plataforma en el mercado y llegó en el HTC Dream en septiembre de 2008, el que es considerado el primer dispositivo comercial con Android incorporado, y que durante su ciclo de vida llegó a ser actualizado en varias ocasiones. Ofrecía un conjunto de características y funcionalidades básicas, tales como una tienda de aplicaciones llamada Android Market, así como compatibilidad y sincronización con los múltiples servicios de aplicaciones de Google (Maps, Contactos, Calendar, mensajería instantánea, Youtube y notificaciones).
En febrero de 2009 llegó una actualización en forma de Android 1.1 Banana Bread que introdujo mejoras y añadió ciertas funcionalidades, si bien mantuvo casi intacta la interfaz del sistema.

2. Android 1.5 Cupcake

Apareció en el mercado el 30 de abril de 2009 e introdujo dos cambios importantes a nivel de interfaz, la posibilidad de insertar widgets en el escritorio y un teclado en pantalla para los teléfonos que no tuvieran teclado físico.

3. Android 1.6 Donut

Apareció en el mercado en Octubre de 2009 y se añadieron múltiples mejoras en cuanto a compatibilidad de hardware (soporte para redes CDMA/EVDO, VPN) o a reconocimiento de voz. Su cuota en el mercado fue mínima debido al dominio de BlackBerry en aquella época.

4. Android 2.0 Eclair

Fue lanzado solo un mes después de Android 1.6 y vino acompañado de una amplia lista de mejoras y novedades entre las que se incluyen un mejor sistema de sincronización, soporte para ciertas tecnologías, cambios en la interfaz de algunas aplicaciones y nuevas funcionalidades. Uno y dos meses después respectivamente se liberaron otras dos subversiones como fueron Android 2.0.1 y Android 2.1 con modificaciones sobre la versión original, principalmente solución a bugs y problemas menores y la capacidad de ser un sistema multitáctil.

5. Android 2.2 Froyo

Es una de las versiones más conocidas de la historia de Android y fue liberada a en mayo de 2010, con la principal virtud de ofrecer un importante lavado de cara así como un amplio listado de novedades. Entre ellas encontramos las notificaciones push, un nuevo motor para el navegador, soporte para pantallas de alta resolución y capacidad de instalación de apps en la memoria externa. Froyo fue una de las versiones con mayor éxito y aceptación por parte del público, y vio tres actualizaciones menores que solucionaron algunos errores: 2.2.1 y 2.2.2 en enero de 2011, y la definitiva 2.2.3 en el mes de noviembre del mismo año.

6. Android 2.3 Gingerbread

Fue lanzada en diciembre de 2010 y se añadieron modificaciones y mejoras que principalmente aludían a la velocidad de respuesta y a la estética. Trajo compatibilidad con nuevas tecnologías como NFC, pantallas más grandes y con mejor resolución y a nivel interno gozó de una cierta optimización para sacarle un mejor partido al hardware. Durante diez meses se introdujeron siete actualizaciones de Gingerbread, desde la versión 2.3.1 hasta la 2.3.7, que en la mayoría de los casos solucionarían bugs menores, pero que también permitieron conocer nuevas funcionalidades como Google Wallet, una de las tecnologías precursoras del actual pago en el móvil.

7. Android 3.0 Honeycomb

Fue lanzada en mayo de 2011 y fue una versión exclusiva para tablets que luego se amplió a otros dispositivos como televisores Google TV que nunca terminaron de cuajar en el mercado. A destacar son los cambios relacionados con la interfaz, ya que ésta era muy diferente en un tablet respecto de un smartphone, que dieron lugar a nuevas funcionalidades pensadas precisamente el nuevo formato. Durante ocho meses se introdujeron seis actualizaciones de Honeycomb, desde la versión 2.3.1 hasta la 2.3.6.

8. Android 4.0 Ice Cream Sandwich

Apareció en los mercados en octubre de 2014 y la interfaz fue renovada casi al completo gracias a la introducción de nuevos botones, iconos, barras de notificaciones e incluso a la tipografía utilizada llamada Roboto. La interfaz y la interacción del usuario cambió de forma tremendamente notable, a la vez que algunas funciones integradas (cámara, visor de fotografías, contactos) cambiaron para adoptar nuevas posibilidades. Igualmente tampoco se olvidaron de nuevas tecnologías como WiFi Direct, aceleración de vídeo vía hardware o grabación de vídeo 1080p. Unos pocos días después del lanzamiento llegó Android 4.0.1, también en el mes de octubre, al que seguiría Android 4.0.2 en noviembre, en ambos casos arreglando pequeños bugs y problemas. Sí fueron más importantes las versiones Android 4.0.3 (diciembre de 2011) y Android 4.0.4 (marzo de 2012), que introdujeron principalmente mejoras en la optimización del sistema.

9. Android 4.1 Jelly Bean

Apareció en julio de 2012 con una estética similar a la anterior versión con ciertos cambios que proporcionaron una mayor fluidez general del sistema y aún mayor compatibilidad con el hardware. A destacar de esta versión fue el hecho de ser compatible tanto para tablets como para smartphones, corrigiendo así el error de la versión 3.0. Fue una de las más longevas y actualizadas como Android 4.2 y 4.3 y cada una actualizada con una o dos subversiones.

10. Android 4.4 KitKat

Google sorprendió al mundo entero llegando a un acuerdo con la famosa marca de chocolatinas en octubre de 2013. Esta es una de las versiones que más cambios ha introducido tanto a nivel estético como, sobre todo, a nivel interno y con vistas al futuro y al mismo tiempo es a día de hoy la versión más utilizada de Android en todo el mundo, algo motivado por sus cuatro actualizaciones que han logrado llevar estabilidad al sistema (4.4.1 y 4.4.2 en diciembre de 2013 y 4.4.3 y 4.4.4 en junio de 2014).

11. Android 5.0 Lollipop

Apareció en noviembre de 2014 y trajo cambios relativos a la interfaz la cual ha sido redefinida casi al completo para incluir el denominado Material Design. Además de la parte gráfica Google implementó múltiples nuevas funcionalidades como una nueva máquina virtual (ART), un nuevo sistema de notificaciones, una mejor gestión de la energía, un nuevo teclado y la definitiva unificación entre los sistemas de smartphone, tablet y también smartwatch y TV. Este importante paquete de mejoras continuó creciendo con las sucesivas subversiones (Android 5.0.1 y Android 5.0.2, en diciembre de 2014) pero sobre todo creció con la versión Android 5.1 que introdujo en marzo de 2015 nuevas posibilidades tales como soporte oficial para dual-SIM, protección frente a pérdidas o robos y llamadas en alta definición. Google ofreció la versión 5.1.1 en el mes abril, solucionando algunos problemas menores.

12. Android 6.0 Marshmallow

Aparece por primera vez el 5 de octubre de 2015 y su listado de novedades es muy amplio. La interfaz se mantiene con pocos cambios respecto de Lollipop y destacan novedades como la introducción de la plataforma de pagos de Google Android Pay, el soporte nativo para los lectores de huellas, elemento fundamental para añadir una dosis de seguridad al proceso, y tecnología USB Type-C.

Figura 3.3 Versiones de Android [28]

3.2.1.4 Estructura y componentes de una aplicación Android

En primer lugar y con el propósito de poder describir de una buena manera cómo están formadas las aplicaciones Android van a estudiarse cuáles son los componentes esenciales con los que se forman estas aplicaciones. Cada componente desempeña un papel específico en la construcción de una aplicación. Entre todos los componentes caben destacar las activities, intents, layouts, services, content providers, views y el documento AndroidManifest.xml. A continuación se mostrarán los directorios que forman la estructura de una aplicación Android y se explicará cuál es la función de cada uno. Dicha estructura puede apreciarse en la **Figura 3.7**. [29][30] [31]

3.2.1.4.1 Componentes de una aplicación Android

1. Activities

Desempeñan el papel que permite construir la interfaz de usuario, es decir, son las pantallas que tiene una aplicación. Tienen como función mostrar los elementos visuales y responder a las acciones del usuario. Una APP suele necesitar varias Activities para tener una interfaz atractiva y cada una es independiente de la otra. Toda actividad hereda de la clase Activity.

2. Intents

Es el elemento básico de comunicación entre los distintos componentes Android. Representa la intención de hacer una determinada acción como puede ser lanzar una Activity, lanzar un servicio, traspasar información entre componentes o realizar una llamada. Las acciones ejecutadas pueden ser internas o externas a la aplicación.

3. Services

Los servicios son componentes que se ejecutan en segundo plano. Son parecidos a los servicios de cualquier otro sistema operativo y están diseñados para seguir ejecutándose si es necesario de manera independiente de cualquier actividad. Un ejemplo de estos servicios es que el reproductor de música esta ejecutándose mientras enviamos un SMS.

4. Contents Providers

Es el componente que se encarga de compartir datos entre aplicaciones de manera que no sea necesario mostrar información acerca de la estructura, almacenamiento interno o la implementación de la aplicación en sí misma.

5. AndroidManifest.xml

Es un archivo de configuración donde se aplican aspectos relativos a la identificación, componentes y permisos necesarios para la ejecución de la aplicación.

6. Views o vistas

Son los componentes básicos que forman parte de la interfaz gráfica. Android dispone de una gran cantidad de estos tales como TextView (etiquetas de texto), EditText (texto editable), Button (botones), ListView (listas) o ImageView (imágenes) entre muchos otros. Estos elementos heredan de la clase View y están definidos mediante código XML. Pueden definirse mediante código o mediante interfaz gráfica.

7. Layout

Es un conjunto de views o vistas que forman una estructura determinada. Existen de diferentes tipos tales como LinearLayout (organiza las vistas de forma lineal), TableLayout (organiza las vistas en forma de tabla) o RelativeLayout (organiza las vistas en cuadrícula) entre otros. Los Layouts son objetos que heredan la clase View y son definidos mediante código XML. Pueden definirse mediante código o mediante interfaz gráfica.

3.2.1.4.2 Estructura de una aplicación Android

1. Carpeta /app/src/main/java

Esta carpeta contiene el código fuente de la aplicación, clases auxiliares, etc. Inicialmente Android Studio creará el código básico de la pantalla (actividad o activity) principal de la aplicación, que en este caso es MainActivity y siempre bajo la estructura del paquete java definido durante la creación del proyecto.

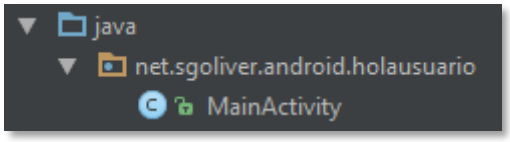


Figura 3.4 Carpeta /app/src/main/java [30]

2. Carpeta /app/src/main/res/

Esta carpeta contiene todos los recursos necesarios para un proyecto, es decir, imágenes, layouts, cadenas de texto, etc. Los diferentes tipos de recursos se pueden distribuir entre las siguientes subcarpetas:

- /res/drawable/ : Contiene las imágenes de la aplicación. Se divide en /drawable-ldpi, /drawable-mdpi y /drawable-hdpi para utilizar diferentes recursos dependiendo de la resolución del dispositivo.
- /res/layout/ : Contiene los ficheros de definición para las diferentes pantallas de la interfaz gráfica. Se puede dividir en /layout y /layout-land dependiendo de la orientación del dispositivo.
- /res/anim/ : Contiene la definición de las animaciones utilizadas por la aplicación.
- /res/menu/ : Contiene la definición de los menús de la aplicación.
- /res/values/ : Contiene recursos de la aplicación como por ejemplo cadenas de texto (strings.xml), estilos (styles.xml), colores (colors.xml), etc.
- /res/xml/ : Contiene los ficheros XML utilizados por la aplicación.
- /res/raw/ : Contiene recursos adicionales, normalmente en formato distinto a XML, que no se incluyan en el resto de carpetas de recursos.

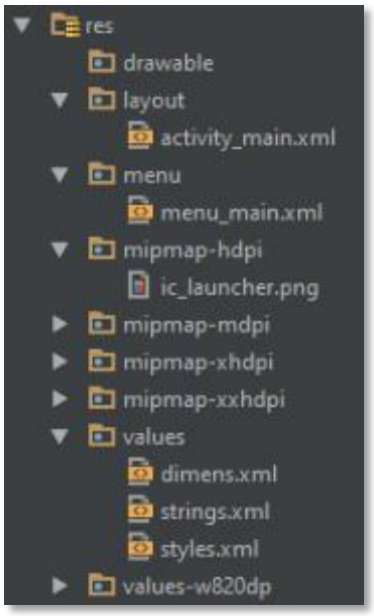


Figura 3.5 Carpeta /app/src/main/res/ [30]

3. Fichero /app/build.gradle

Contiene diferentes tipos de información necesaria para la compilación del proyecto tales como la versión del SDK que Android utilizada para compilar, la mínima versión de Android que soporta la aplicación o las referencias a las librerías externas utilizadas entre otros. En un proyecto existen varios ficheros build.gradle para definir determinados parámetros a distintos niveles, por ejemplo existe un fichero build.gradle a nivel de proyecto y otro a nivel de módulo dentro de la carpeta /app.

4. Carpeta /app/libs

Contiene las librerías java externas (ficheros .jar) que utilice una aplicación.

5. Carpeta /app/build/

Contiene una serie de elementos de código generados automáticamente al compilar el proyecto dirigidos, entre otras muchas casos, al control de los recursos de la aplicación. Un fichero a destacar aquí es el llamado R.java el cual define la clase R la cual contiene una serie de identificadores de cada uno de los recursos de la aplicación incluidos en la carpeta /app/src/main/res/ para que puedan ser accesibles código java.

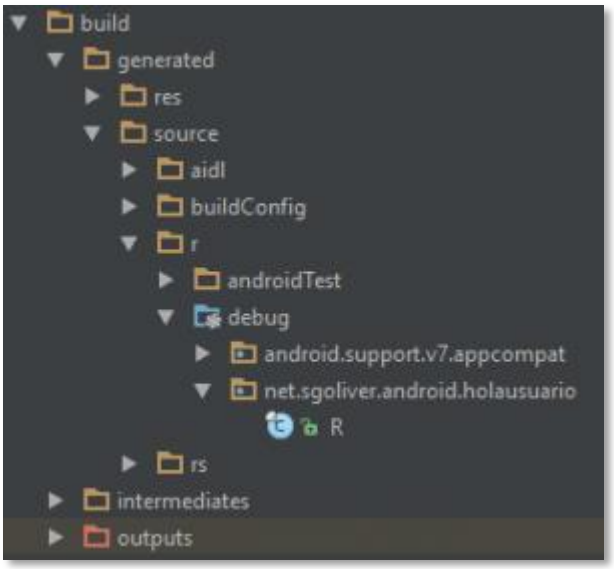


Figura 3.6 Carpeta /app/src/main/res/ [30]

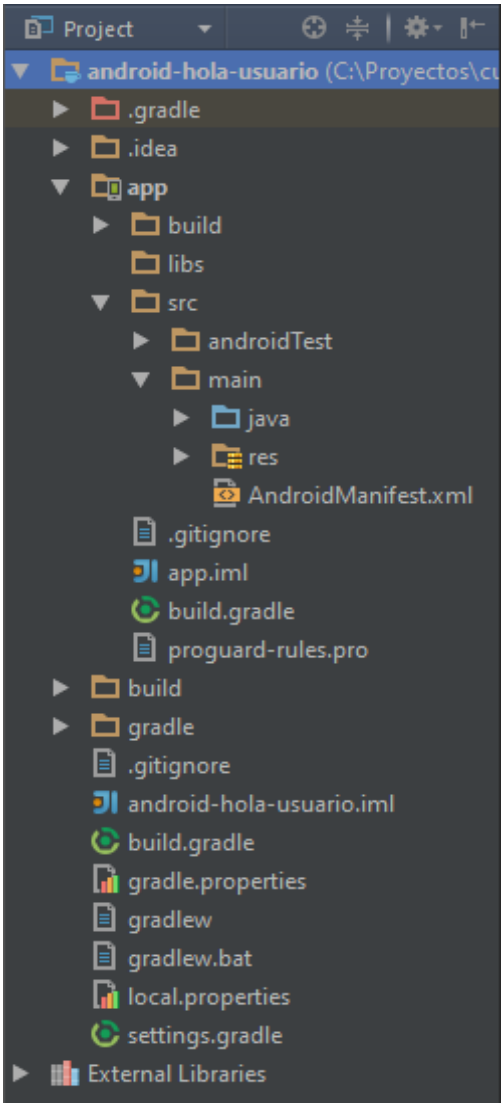


Figura 3.7 Estructura general de un proyecto Android [30]

3.2.1.5 Ventajas y desventajas Android

A continuación se estudiarán cuáles son algunas de las ventajas y desventajas de la plataforma Android. [31]

Ventajas de la plataforma Android

- Plataforma de código abierto: Android está basado en el kernel de Linux y como consecuencia cualquier persona puede realizar una aplicación Android.

- Debido a que muchas compañías relacionadas con la tecnología móvil se encuentran dentro del grupo Open Handset Alliance y que este se encuentra entre los benefactores de Android, se promueven las ventas con el mismo sistema operativo dentro de una gran cantidad de terminales, no solo móviles sino también tablets, GPS, relojes inteligentes o televisores entre otros.

- El sistema operativo pertenece a Google, y contiene muchas aplicaciones por defecto relacionados con cada uno de los servicios que provee la empresa multinacional, como Hangouts y Gmail entre otros.

- La facilidad para crear aplicaciones Android ha favorecido la popularidad y el crecimiento del mismo, dando lugar a las más de 650.000 aplicaciones disponibles en el mercado.

- La existencia de muchas versiones permite a un smartphone tener el mejor desempeño posible si utiliza la última versión soportada para el dispositivo.

- Es un sistema multitarea, es decir, es capaz de hacer funcionar a la vez varias aplicaciones y gestionarlas al mismo tiempo, dejarlas en modo suspensión si no se utilizan e incluso cerrarlas si llevan un periodo determinado de inactividad.

Desventajas de la plataforma Android

- Es muy vulnerable debido a que es de código abierto.
- Un sistema multitarea da lugar a tener varias aplicaciones abiertas hace que el consumo de batería aumente considerablemente.
- No tiene un soporte de actualización como el caso de Apple, el cual permite actualizar a la última versión el software brindado por la empresa de una forma sencilla sin importar el hardware.

3.2.1.6 IDE Android Studio

Android Studio es un entorno de desarrollo integrado basado en IntelliJ IDEA de la compañía JetBrains que utiliza una licencia de software libre Apache 2.0 y se caracteriza por estar programado en Java y ser multiplataforma. Fue presentado por Google el 16 de mayo del 2013 en el congreso de desarrolladores Google I/O con el objetivo de crear un entorno dedicado en exclusiva a la programación de aplicaciones para dispositivos Android y el 8 de diciembre de 2014, fecha en la que se liberó la versión estable de Android Studio 1.0, Google pasó a recomendarlo como IDE oficial para desarrollar aplicaciones para su sistema operativo ya que proporciona numerosas mejoras con respecto al antiguo plugin ADT para Eclipse y a día de hoy ha ido actualizándose y mejorándose hasta la actual versión 1.5. [33][34]

Android Studio está disponible para Windows 2003, Vista, 7, 8 y GNU/Linux, tanto plataformas de 32 como de 64 bits, Linux con GNOME o KDE y Mac OS X.

Los requisitos del sistema para las tres plataformas son:

- 2 GB de RAM (4 GB recomendados)
- 400 MB de espacio en disco
- 1 GB para Android SDK
- Java Development Kit 7 o superior

Android Studio destaca por las siguientes capacidades:

- Renderización en tiempo real
- Consola de desarrollador, consejos de optimización, ayuda para la traducción, estadísticas de uso.
- Soporte para construcción basada en Gradle.
- Refactorización especifica de Android y arreglos rápidos.
- Herramientas Lint para detectar problemas de rendimiento, usabilidad, compatibilidad de versiones y otros problemas.
- Plantillas para crear diseños comunes de Android y otros componentes.
- Soporte para programar aplicaciones para Android Wear.

3.2.2 Sistemas de Gestión de Bases de Datos

Existe una cantidad moderada de SGBD en el mercado actual dominado principalmente por MYSQL, Oracle y SQLServer. A continuación se presentarán los SGBD que se han utilizado en este proyecto y tras ello una tabla comparativa (**Tabla 3.2**) con las características principales de los SGBD que existen en el mercado actual.

3.2.2.1 MySQL

MySQL es un SGBD de datos relacional, multihilo y multiusuario desarrollado por la empresa MySQL AB (más tarde conocida como Sun Microsystems y finalmente adquirida por Oracle Corporation) como un software libre en un esquema de licenciamiento dual, es decir, por un lado se ofrece bajo la GNU GPL para cualquier uso compatible con esta licencia, pero para aquellas empresas que quieran incorporarlo en productos privativos deben comprar a la empresa una licencia específica. MySQL es usado por muchos sitios web como Wikipedia, Google (no para búsquedas), Facebook, Twitter, Flickr, y YouTube. Según las cifras del fabricante, existen más de seis millones de copias de MySQL funcionando en la actualidad, lo que supera la cantidad de cualquier otra herramienta de bases de datos. [35]

Existen varias interfaces de programación de aplicaciones que permiten, a aplicaciones escritas en diversos lenguajes de programación, acceder a las bases de datos MySQL incluyendo C, C++, C#, Pascal, Delphi, Eiffel, Smalltalk, Java (con una implementación nativa del driver de Java), Lisp, Perl, PHP, Python, Ruby, FreeBASIC, y Tel. Cada uno de estos utiliza una interfaz de programación de aplicaciones específica. Su popularidad como aplicación web está muy ligada a PHP que a menudo aparece en combinación con MySQL.

MySQL es ideal para aplicaciones web donde hay baja concurrencia en la modificación de datos y donde es intensiva la lectura de datos ya que está caracterizada por una rápida en la lectura, aunque puede provocar problemas de integridad en entornos de alta concurrencia en la modificación.

MySQL funciona sobre múltiples plataformas entre las que se encuentran AIX, BSD, GNU/Linux, Mac OS X, Solaris, SunOS y todas las versiones de Windows existentes a partir de Windows 95 entre muchas otras.

Entre las características más significativas se encuentran las siguientes:

- Disponibilidad en gran cantidad de plataformas y sistemas.
- Posibilidad de selección de mecanismos de almacenamiento que ofrecen diferentes velocidades de operación, soporte físico, capacidad, distribución geográfica, transacciones etc.
- Transacciones y claves foráneas.
- Conectividad segura.
- Replicación.
- Búsqueda e indexación de campos de texto.
- Uso de multihilos mediante hilos del kernel.
- Completo soporte para operadores y funciones en cláusulas select y where.
- Completo soporte de funciones de agrupación group by y order by.
- Seguridad, ya que ofrece un sistema de contraseñas y privilegios seguro mediante verificación basada en el host y el tráfico de contraseñas está cifrado al conectarse a un servidor.

- Soporta gran cantidad de datos. MySQL Server tiene bases de datos de hasta 50 millones de registros.
- Se permiten hasta 64 índices por tabla (32 antes de MySQL 4.1.2). Cada índice puede consistir desde 1 hasta 16 columnas o partes de columnas. El máximo ancho de límite son 1000 bytes (500 antes de MySQL 4.1.2).
- Los clientes se conectan al servidor MySQL usando sockets TCP/IP en cualquier plataforma. En sistemas Windows se pueden conectar usando named pipes y en sistemas Unix usando ficheros socket Unix.

La versión de MySQL más reciente se sitúa en la 5.7.9 publicada el 22 de octubre de 2015

3.2.2.2 SQLite

SQLite es un sistema de gestión de bases de datos relacional compatible con ACID contenida en una relativamente pequeña (275 kB) biblioteca escrita en C y caracterizada por ser de dominio público. A diferencia de los sistemas de gestión de bases de datos cliente-servidor habituales el motor de SQLite no es un proceso independiente con el que el programa principal se comunica, es decir, la biblioteca SQLite se enlaza con el programa pasando a ser parte integral del mismo. El programa utiliza la funcionalidad de SQLite a través de llamadas simples a subrutinas y funciones. Esto reduce la latencia en el acceso a la base de datos, debido a que las llamadas a funciones son más eficientes que la comunicación entre procesos. El conjunto de las bases de datos (definiciones, tablas, índices, y los propios datos), son guardados como un sólo fichero estándar.

A continuación se describirán algunas de las características de SQLite.

La biblioteca implementa la mayor parte del estándar SQL-92 incluyendo transacciones de base de datos atómicas, consistencia de base de datos, aislamiento, triggers y la mayor parte de las consultas complejas. SQLite usa un sistema de tipos inusual, en lugar de asignar un tipo a una columna como en la mayor parte de los sistemas de bases de datos SQL, los tipos se asignan a los valores individuales por lo que, por ejemplo, se puede insertar un string en una columna de tipo entero.

Varios procesos o hilos pueden acceder a la misma base de datos sin problemas. Varios accesos de lectura pueden ser servidos en paralelo. Un acceso de escritura sólo puede ser servido si no se está sirviendo ningún otro acceso concurrentemente.

SQLite es utilizado en un gran variedad de aplicaciones tales como Adobe Photoshop Elements (como motor de base de datos), Mozilla Firefox (para cookies, favoritos, el historial y las direcciones de red válidas, Opera (WebSQL) o Skype.

Debido a su pequeño tamaño, SQLite es muy adecuado para los sistemas integrados como Firefox OS, iOS5 y Google Chrome y móviles como Android, BlackBerry, Windows Phone 8 y Symbian.

SGBD	Requisitos para instalación	Características
Oracle	•512 RAM •1 GB Memoria virtual •1.5 GB Disco Duro •Tamaño máximo de la BD: 4GB •Arquitectura del sistema: 32/64 bit	•Es un SGBD relacional •Se puede implementar en diferentes plataformas: Microsoft, Unix, Linux y Vms. •Cuenta con un interface de última tecnología basado en Java y Xml. •Un servidor adjunto de aplicaciones para internet, email, seguridad de datos, etc. •Metalink
MySQL	•512 RAM •1 GB Memoria virtual •1 GB Disco Duro •Tamaño máximo de la BD: Sin limite •Arquitectura del sistema: 32/64 bit	•Escrito en C y en C++ •Funciona en diferentes plataformas. •Un sistema de reserva de memoria muy rápido basado en threads. •Un sistema de privilegios y contraseñas que es muy flexible y seguro, que permite verificación basada en el host.
INFORMIX	•256 MB RAM •700 MB Disco Duro •Arquitectura del sistema: 32/64 bit	•Dispone de herramientas graficas •Gestiona múltiples bases de datos remotas de una única consola. •Capacidad de relación de datos de múltiples lugares físicos •Ocupa menos memoria y recursos que el oracle. •Ofrece herramientas para crear menús, formularios de entrada de datos y generadores de listados
SQLServer	•Ediciones Express: 512 MB RAM •Todas las demás ediciones: 1 GB RAM •Procesador x86: 1,0 GHz •Procesador x64: 1,4 GHz •Disco Duro 1 GB	•Facilidad de instalación , distribución y utilización. •Posee una gran variedad de herramientas administrativas y de desarrollo que permite mejorar la capacidad de instalar ,distribuir, administrar y utilizar SQL Server. •Incluye herramientas para extraer y analizar datos resumidos para el proceso analítico en línea. •Incluye herramientas para diseñar gráficamente las base de datos y analizar los datos mediante preguntas en lenguaje normal.
DB2	•256 MB RAM •2 GB de Disco Duro •Arquitectura del sistema: 32/64 bit	•Es un SGBD relacionales multiplataforma. •Tablas de resumen •Tablas replicadas •Agiliza el tiempo de respuesta de una consulta •Recuperación utilizando accesos de solo índices. •Guarda sus datos contra la pérdida, acceso no autorizado o entradas inválidas.
PostgreSQL	•250 MB de espacio en Disco Duro •256 MB RAM •Arquitectura del sistema: 32/64 bit	•Implementación del estándar SQL92/SQL99. •Soporta distintos tipos de datos •Permite la creación de tipos propios. •Incorpora una estructura de datos array. •Permite la declaración de funciones propias, así como la definición de disparadores. •Soporta el uso de índices, reglas y vistas. •Incluye herencia entre tablas (aunque no entre objetos, ya que no existen), por lo que a este gestor de bases de datos se le incluye entre los gestores objeto-relacionales. •Permite la gestión de diferentes usuarios, como también los permisos asignados a cada uno de ellos.

Tabla 3.2 SGBD más importantes [37]

3.2.3 IDE Eclipse

Eclipse es una plataforma de software compuesto por un conjunto de herramientas de programación de código abierto multiplataforma mayormente utilizadas para desarrollar lo que se conoce como aplicaciones de cliente enriquecido, opuesto a las aplicaciones de cliente ligero basadas en navegadores. Fue originado como un proyecto de IBM Canadá y fue desarrollado por OTI (Object Technology International) y en 2003 fue creada la fundación independiente de IBM. Desde 2004 hasta 2015 Eclipse ha ido actualizado trece veces desde la versión 3.0 hasta la actual 4.6 y tiene planeado una nueva versión a mediados de 2016. [38]

En cuanto a licencias Eclipse fue liberado originalmente bajo la Common Public License, pero después fue re-licenciado bajo la Eclipse Public License. Ambas licencias se consideran de software libre pero son incompatibles con Licencia pública general de GNU (GNU GPL).

Eclipse está caracterizado por disponer de un editor de texto con un analizador sintáctico, por utilizar compilación es en tiempo real y tener pruebas unitarias con JUnit, control de versiones con CVS, integración con Ant, asistentes (wizards) para creación de proyectos, clases, tests y refactorización. Asimismo a través de plugins libremente disponibles es posible añadir más funcionalidades.

A continuación se describirá brevemente la arquitectura de Eclipse.

La base de Eclipse es la Plataforma de Cliente Enriquecido o RCP y está constituida por los siguientes componentes:

- Pantalla de carga de Eclipse.
- Plataforma principal, es decir, inicio de Eclipse y ejecución de plugins.
- OSGi: Una plataforma para bundling estándar.
- El Standard Widget Toolkit (SWT).
- JFace para manejo de archivos, de texto y editores de texto.
- El Workbench de Eclipse: Vistas, editores, perspectivas, asistentes etc.

Los widgets de Eclipse están implementados por una herramienta de widget para Java llamada Standard Widget Toolkit. La interfaz de usuario de Eclipse también tiene una capa GUI intermedia llamada JFace.

El IDE Eclipse emplea módulos o plugins para proporcionar toda su funcionalidad. Este mecanismo de módulos da lugar a una plataforma ligera para componentes de software. Adicionalmente permite a Eclipse extenderse usando otros lenguajes de programación como son C/C++ y Python, trabajar con lenguajes para procesado de texto como LaTeX, y utilizar aplicaciones en red como Telnet y SGBD.

Esta plataforma ha sido usada para desarrollar otros IDE como el de Java llamado Java Development Toolkit (JDT) y el compilador Eclipse Compiler for Java (ECJ).

3.2.4 Servidor Apache Tomcat

Apache Tomcat puede definirse como un contenedor de servlets desarrollado bajo el proyecto Jakarta en la Apache Software Foundation. Tomcat implementa las especificaciones de servlets y JavaServer Pages (JSP) de Oracle Corporation por lo que no es un servidor de aplicaciones como JBoss o JOnAS. Incluye el compilador Jasper que compila JSPs convirtiéndolas en servlets y puede funcionar como servidor web por sí mismo e incluso es usado como servidor web autónomo en entornos con alto nivel de tráfico y alta disponibilidad. [39]

Tomcat es desarrollado y actualizado por miembros de la Apache Software Foundation y voluntarios independientes. Los usuarios disponen de libre acceso a su código fuente y a su forma binaria en los términos establecidos en la Apache Software License. Las primeras distribuciones de Tomcat, las versiones 3.0.x, fueron desarrolladas en 1999 y las más recientes las 8.x en 2015, las cuales implementan las especificaciones de Servlet 3.0 y de JSP 2.2. A partir de la versión 4.0 Jakarta Tomcat utiliza el contenedor de servlets Catalina.

Dado que Tomcat fue escrito en Java funciona en cualquier sistema operativo que disponga de la máquina virtual Java.

3.3 Estado del arte legislativo

El sujeto que de uso a esta aplicación debe estar informado de la Ley Orgánica 15/1999 del 13 de Diciembre de Protección de Datos de Carácter Personal, concretamente del apartado 6.1 el cual dice así: El tratamiento de los datos de carácter personal requerirá el consentimiento inequívoco del afectado. El incumplimiento de esta ley puede conllevar tres tipos de sanciones: leves, graves y multas entre 600€ y 600.000€. [42]

De acuerdo a la ley anteriormente mencionada el administrador del servidor no podrá hacer uso comercial de los datos almacenados por los usuarios sin un consentimiento previo de los mismos.

3.4 Análisis del sistema

3.4.1 Definición del sistema

El objetivo del presente proyecto es el de realizar una aplicación para dispositivos móviles con plataforma Android que permita al usuario gestionar bases de datos con tecnología MySQL, junto con las tablas que compongan a estas bases de datos, ubicadas en servidores externos al dispositivo y, por otro lado, ofrecer una herramienta de apoyo a dicho sistema permitiendo la capacidad de traspasar los datos de las tablas almacenadas en servidores externos al dispositivo móvil, y también permitiendo el traspaso de tablas desde el dispositivo a un servidor mediante uso de SQLite, un SGBD implementado en Android dedicado a gestionar las bases de datos asociadas a las aplicaciones del dispositivo.

Teniendo en cuenta lo anterior la aplicación ofrece la posibilidad de trabajar en tres entornos diferentes según el lugar donde se encuentren las bases de datos a gestionar. A continuación se expondrán las características de cada uno de los entornos.

Entorno 1: Servidor privado www.servidormysql.no-ip.org

Es un servidor de bases de datos MySQL gratuito realizado por el mismo desarrollador de la aplicación que permite a cualquier usuario almacenar cualquier número de bases de datos y tablas dentro de estas bases que desee de manera gratuita. En este punto se diferenciarán tres tipos de servicios determinados:

1. Registros de usuarios: En primer lugar los usuarios deberán registrarse con su usuario y contraseña para poder utilizar el servicio de almacenamiento de bases de datos. Con este fin se ha puesto a disposición, mediante un servidor Apache Tomcat, un formulario de registro con dirección https://servidormysql.no-ip.org:8443/Servicios/ FormularioRegistro.jsp.

2. Creación de bases de datos: MySQL no ofrece la posibilidad de dar la capacidad a un usuario de crear cuantas bases de datos desee sin, al mismo tiempo, poder tener acceso al resto de bases de datos de modo que mediante el formulario con dirección https://servidormysql.no-ip.org:8443/Servicios/FormularioBasesDeDatos.jsp podrán crearse bases de datos con acceso único al usuario que las crea.

3. Gestión de bases de datos: Aquí se dan todas las funcionalidades necesarias para que un usuario pueda gestionar sus bases de datos ubicadas en el servidor tales como borrar bases de datos y tablas dentro de estas así como insertar, borrar o modificar registros y propiedades dentro de estas tablas entre otras funcionalidades. Funciona en el puerto por defecto de MySQL 3306.

Entorno 2: Servidor alternativo

La aplicación permite al usuario la capacidad de conectarse a cualquier otro servidor MySQL siempre y cuando se encuentre operativo en el puerto 3306 mediante la introducción de la dirección IP del servidor y sus credenciales de usuario. El usuario podrá gestionar desde la aplicación las bases de datos almacenadas en el servidor a las que tenga acceso de la misma manera que en el caso anterior, salvo en el caso de la creación de bases de datos que aquí se hará de manera directa si es que el usuario tiene ese privilegio permitido en el servidor.

Entorno 3: Servidor móvil

En este caso se pasará a gestionar las tablas de la base de datos con nombre 'BaseMySQLite' ubicada en el terminal, lugar donde se almacenan los registros que hayan sido copiados desde servidores externos o bien introducidos manualmente. En este entorno se contarán con menos capacidades de gestión que en los casos anteriores ya que SQLite ofrece menos funcionalidades en este sentido que MySQL.

Para el desarrollo de la aplicación se han seguido como guía las fases de la ingeniería software, las cuales se mostrarán a continuación. [40]

1. Plan operativo: Etapa donde se define el problema a resolver, las metas del proyecto, las metas de calidad y se identifica cualquier restricción aplicable al proyecto.

2. Especificación de requisitos: Permite entregar una visión de alto nivel sobre el proyecto, poniendo énfasis en la descripción del problema desde el punto de vista de los clientes y desarrolladores. También se considera la posibilidad de una planificación de los recursos sobre una escala de tiempos.

3. Especificación funcional: Especifica la información sobre la cual el software a desarrollar trabajará.

4. Diseño: Permite describir como el sistema va a satisfacer los requisitos. Esta etapa a menudo tiene diferentes niveles de detalle. Los niveles más altos de detalle generalmente describen los componentes o módulos que formarán el software a ser producido. Los niveles más bajos describen con mucho detalle cada módulo que contendrá el sistema.

5. Implementación: Aquí es donde el software a ser desarrollado se codifica. Dependiendo del tamaño del proyecto la programación puede ser distribuida entre distintos programadores o grupos de programadores. Cada uno se concentrará en la construcción y prueba de una parte del software, a menudo un subsistema. Las pruebas, en general, tiene por objetivo asegurar que todas las funciones están correctamente implementadas dentro del sistema.

6. Integración: Es la fase donde todos los subsistemas codificados independientemente se juntan. Cada sección es enlazada con otra y entonces probada. Este proceso se repite hasta que se han agregado todos los módulos y el sistema se prueba como un todo.

7. Validación y verificación: Una vez que el sistema ha sido integrado, comienza esta etapa. Es donde es probado para verificar que el sistema es consistente con la definición de requisitos y la especificación funcional. Por otro lado, la verificación consiste en una serie de actividades que aseguran que el software implementa correctamente una función específica. Al finalizar esta etapa el sistema ya puede ser instalado en ambiente de explotación.

8. Mantenimiento: El mantenimiento ocurre cuando existe algún problema dentro de un sistema existente e involucraría la corrección de errores que no fueron descubiertos en las fases de pruebas, mejoras en la implementación de las unidades del sistema y cambios para que responda a los nuevos requisitos. Las mantenciones se puede clasificar en correctiva, adaptativa, perfectiva y preventiva.

3.4.1 Definición de requisitos funcionales

A continuación se mostrará una lista de los requisitos funcionales necesarios para el correcto funcionamiento de la aplicación:

- El usuario, independientemente del servidor a utilizar, deberá en primer lugar estar registrado para posteriormente identificarse con el servidor mediante el uso de credenciales de contraseña y usuario, este último único para cada uno.

- Las conexiones del cliente móvil con el servidor con deben ser cerradas una vez se ha ejecutado cualquier sentencia para no sobrecargar el servidor.

- El abanico de funcionalidades de gestión tales como crear o borrar bases de datos o tablas, insertar, borrar, visualizar o modificar registros entre otras debe ser lo suficientemente amplio para ofrecer al usuario las suficientes capacidades para que este pueda considerar a la aplicación como útil.

- Las sentencias MySQL que se envíen desde el cliente deben llegar al servidor y que este las ejecute, tras esto, el servidor debe devolver una respuesta según el resultado de la ejecución sentencia.

- El administrador del servidor podrá conocer en todo momento los datos que le lleguen a este desde los clientes.

3.4.2 Definición de requisitos no funcionales

En este apartado se discutirán los requisitos no funcionales de la aplicación los cuales corresponden a las características de funcionalidad, rendimiento, usabilidad, seguridad, fiabilidad, compatibilidad y apariencia, además de requerimientos de software y hardware necesarios para el correcto funcionamiento de la misma.

3.4.2.1 Requerimientos de funcionalidad

Una vez que la aplicación esté totalmente desarrollada se realizarán las pruebas pertinentes para detectar los posibles errores que puedan ocurrir, recalcando aquí la fase de pruebas de todo software. Dichas pruebas se centrarán de forma más activa en las acciones relacionadas con el envío y ejecución de sentencias MySQL y SQLite. Tras solucionar los posibles errores que puedan existir se presentará la aplicación a diversos conocidos y se les preguntará su opinión acerca de la funcionalidad que presenta la aplicación y de las posibles mejoras que pudieran implantarse.

3.4.2.2 Requerimientos de usabilidad

La interfaz de la aplicación debe ser intuitiva y capaz de ofrecer al usuario un modo de gestionar bases de datos evitando la introducción manual de sentencias SQL. Para el uso de la aplicación se precisará acceso a Internet en el dispositivo móvil (WiFi o datos) para poder conectar con un servidor MySQL externo.

3.4.2.3 Requerimientos de seguridad

Un servidor MySQL al que se conecte la aplicación cliente deberá proteger el acceso a las bases de datos de los usuarios mediante credenciales de usuario y contraseña.

Los clientes podrán tener acceso únicamente a sus propias tablas y a las permitidas pertenecientes a otros usuarios, utilizando únicamente las acciones asociadas a los privilegios que el usuario les haya ofrecido. Por otro lado el administrador podrá tener acceso a todas las tablas y registros de todos los usuarios y realizar los cambios que considere oportunos.

Las comunicaciones entre cliente móvil y servidor deberán ser seguras, es decir, las credenciales o al menos la contraseña deberán ir cifradas. Preferiblemente deberán cifrarse también las sentencias MySQL enviadas.

3.4.2.4 Requerimientos de fiabilidad

Se debe contemplar la posibilidad de que exista un error en la sentencia o en la conexión, situación ante la cual la aplicación deberá actuar y en consecuencia el usuario deberá ser informado. MySQL es capaz de sobrellevar los posibles problemas concurrencia, de modo que los servidores MySQL pueden enfrentar situaciones donde varios usuarios accedan a la misma información al mismo tiempo.

3.4.2.5 Requerimientos de apariencia

La interfaz que proporciona la aplicación debe ser intuitiva y sencilla. Los elementos visuales tales como botones, textos editables o seleccionables deberán ser lo suficientemente grandes y deberán estar lo suficientemente separados de manera que un usuario pueda estar seguro del elemento que selecciona al pulsar la pantalla.

La barra de acciones deberá estar siempre visible en cada actividad, excepto en la que se encarga de mostrar los registros de las tablas, en donde podrá desactivarse si así se desea para conseguir una mejor visualización de la tabla.

Cada actividad debe llevar por defecto un elemento scroll vertical que permita al usuario visualizar todo el contenido de la pantalla mostrada cuando se da el caso de que la pantalla del dispositivo móvil es más pequeña que la mostrada por la actividad. Se añadirán también elementos scroll horizontales cuando se precise.

Cuando se deba realizar una cierta acción que suponga una cantidad de tiempo considerable se deberá mostrar un mensaje de carga que indique al usuario que esa acción se está llevando a cabo.

Todo mensaje mostrado por la aplicación estará en castellano, incluido los mensajes de error o avisos, excepto los errores de tipo SQL que mande el servidor al cliente surgidos de la ejecución de sentencias y algunos términos relativos a las tecnologías SQL que estarán en inglés, ya que la traducción de estos errores por parte de MySQL y SQLite al castellano no está adecuadamente realizada.

3.4.2.6 Requerimientos de compatibilidad

La aplicación cliente será compatible con cualquier dispositivo con plataforma Android cuya versión sea superior a la mínima exigida (Android 4.0). Los formularios con extensión jsp deben ser compatibles con el navegador

3.4.2.7 Requerimientos software

Los requerimientos de software se fundamentan principalmente en el uso por parte del terminal móvil de la plataforma Android con una versión superior a la mínima exigida (Android 4.0), posibilitando así que alrededor del 94% de los dispositivos con plataforma Android puedan utilizar la aplicación, y un espacio de almacenamiento disponible de 5.49 MB. El dispositivo de pruebas para tal fase hace uso de la versión de Android 4.4 KitKat.

Los formularios con extensión jsp serán accesibles desde diferentes navegadores. Se han utilizado algunos navegadores gratuitos tanto en el dispositivo móvil como en ordenador como son Google Chrome y Mozilla Firefox.

El equipo de desarrollo utilizado contiene los siguientes elementos software:

- Sistema Operativo Windows 7 Ultimate Service Pack 1 de 64 bits
- Librerías de desarrollo Java SDK 8 y Android SDK 22: Son un conjunto herramientas de desarrollo de software que como su propio nombre sugiere han sido necesarias para la creación del software de la aplicación Android.
- Entorno de desarrollo Android Studio 1.3: IDE desarrollado por Google que se ha designado como oficial para la creación de aplicaciones Android.
- Entorno de desarrollo Eclipse Mars 4.5: IDE que aporta un conjunto de herramientas para desarrollo de software tales como un editor de texto o plugins diversos que serán necesarias para crear los formularios con extensión jsp.
- Servidor Tomcat Versión 7.0.56: Servidor que funciona como un contenedor de servlets y, por tanto, hace posible el acceso a los formularios con extensión jsp.
- Servidor MySQL Versión 5.6.21 en puerto 3306: Servidor privado MySQL al que la aplicación móvil cliente podrá acceder.
- Conector/librería JDBC 5.1.36: Permite la conexión desde una aplicación desarrollada en java con un servidor MySQL.

- Herramienta online para adaptación de imágenes e iconos [40].
- Herramienta online para servicio de DNS [47].
- Plantilla con extensión css para el desarrollo del aspecto de los formularios.
- Software OpenSSL para la creación de certificados.
- Software DIA para la realización de diagramas de los casos de uso y de los modelos entidad/relación de la aplicación.
- Microsoft Office 2007 Profesional: Utilizado para la creación de la documentación del proyecto.

3.4.2.8 Requerimientos hardware

Los requerimientos de hardware para el uso de la aplicación se basan en la utilización de un terminal móvil con plataforma Android y con acceso a Internet donde se instalará la aplicación cliente. Los formularios que proporciona el servidor privado de la aplicación podrán ser accesibles desde un terminal fijo o desde el mismo terminal móvil.

Un usuario podría querer utilizar la aplicación móvil para conectarse a un servidor MySQL local propio, de modo que necesitaría lógicamente tener y configurar dicho servidor.

Para la elaboración de este proyecto se ha contado con los siguientes medios:

1. Un ordenador de sobremesa para el desarrollo de la aplicación con las siguientes características:

    - Procesador Intel(R) Core(TM) i-4130 CPU @ 3.40GHz
    - Memoria instalada (RAM): 8.00 GB
    - Sistema operativo de 64 bits
    - Disco duro de 500 GB.

2. Un dispositivo móvil ZTE Blade L2 para la fase de verificación y utilización con las siguientes características:

- Procesador quad-core 1.3GHz
- 4GB memoria interna, 1GB RAM
- Tamaño de pantalla 480 x 854 pixels, 5.0 pulgadas

3.4.2.9 Estudio de viabilidad

En el Anexo 3: Estudio económico y planificación puede encontrarse un estudio detallando la viabilidad del proyecto realizado. Para tal estudio se ha realizado, en primer lugar, un análisis temporal y económico donde se han tenido en cuenta los recursos materiales y las horas necesarias para cada actividad que ha sido necesaria para la elaboración del proyecto.

Tras valorar los resultados obtenidos del estudio se puede concluir que el proyecto es viable teniendo en cuenta los recursos materiales, temporales y de personal utilizados y la tecnología que actualmente existe necesaria para la elaboración de la aplicación.

3.5 Diseño del sistema

3.5.1 Descripción de la arquitectura del sistema

La arquitectura del sistema a desarrollar contará con dos componentes principales que serán el dispositivo móvil, con plataforma Android y con la aplicación cliente instalada, y el servidor MySQL con el que vaya a conectarse. La aplicación cliente ofrecerá un interfaz adecuado al usuario con las suficientes funcionalidades para poder gestionar de manera cómoda las bases de datos ubicadas en el servidor mientras que, por otro lado, el servidor ejecutará las sentencias enviadas por el cliente.

Se distinguirán tres arquitecturas similares según el servidor elegido para establecer la conexión, es decir, conexión con el servidor MySQL privado con dirección www.servidormysql.no-ip.org, conexión con servidor MySQL alternativo o bien con la base de datos ubicada en el mismo dispositivo mediante SQLite, que no es un servidor propiamente dicho.

Todas las estructuras están representadas en la **Figura 1.5**, las cuales se analizarán a continuación de forma detallada.

3.5.1.1 Arquitectura servidor alternativo

En este caso se estudiará el funcionamiento del sistema cuando el cliente se conecta con un servidor MySQL alternativo al que proporciona la aplicación por defecto.

En primer lugar, si un usuario quisiera realizar una conexión con un servidor MySQL debe estar registrado previamente en el mismo y conocer sus credenciales de usuario y contraseña que lo identifiquen. Dicho esto a continuación se mostrará el funcionamiento de la arquitectura actual, la cual puede encontrarse en la **Figura 3.8**.

La aplicación cliente, instalada en un dispositivo móvil con plataforma Android, podrá realizar una conexión mediante el uso de Internet, bien por WiFi o por datos, con un servidor MySQL que el usuario elija, siempre y cuando se encuentre operativo en el puerto 3306, para ello, el usuario deberá introducir la dirección IP del servidor con el que quiera conectarse y sus credenciales de usuario para identificarse ante él, si este se equivocase durante el inicio de sesión el servidor se lo indicaría. Una vez identificado el usuario, este podrá acceder a las funcionalidades de gestión de bases de datos que proporciona la aplicación, las cuales funcionan gracias al envío de sentencias MySQL desde el terminal móvil al servidor. Algunas de estas funcionalidades son crear y borrar bases de datos y tablas dentro de estas, insertar, borrar o modificar registros y propiedades dentro de estas tablas y otorgar permisos al resto de usuarios sobre las tablas propietarias.

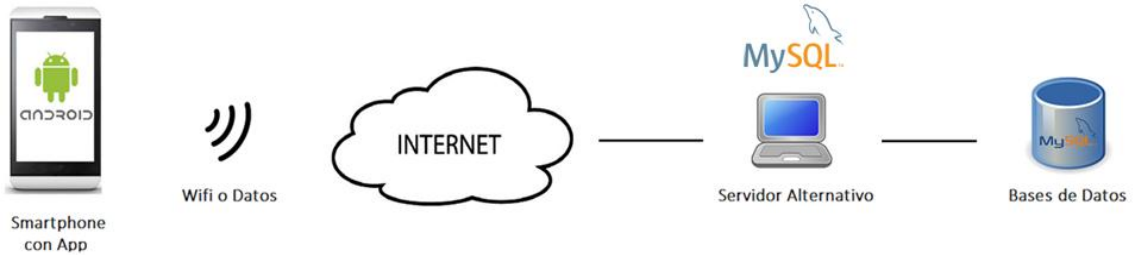


Figura 3.8 Arquitectura servidor alternativo

3.5.1.2 Arquitectura servidor www.servidormysql.no-ip.org

La arquitectura actual funciona de manera similar a la anterior, es decir, un usuario podrá conectarse e identificarse ante el servidor mediante el uso de credenciales y posteriormente, si ha tenido éxito en la identificación, podrá hacer uso de las funcionalidades de la aplicación para gestionar las bases de datos sobre las que tenga control almacenadas en el servidor. Sin embargo el servidor que ahora se utiliza ofrece dos servicios de manera diferente al anterior mediante el uso de formularios que serán accesibles desde el navegador. Los servicios descritos son los siguientes:

1. Registros de usuarios: Los usuarios podrán registrarse en el servidor MySQL mediante el uso de un formulario de registro con dirección https://servidormysql.no-ip.org:8443/Servicios/FormularioRegistro.jsp, ubicado en un servidor Apache Tomcat en el mismo ordenador donde se encuentra el servidor MySQL. Dicho formulario enviará los datos de registro hacia un documento llamado *RegistroMYSQL.jsp*, desde el cual se ejecutará la sentencia MySQL necesaria para la creación del usuario.

2. Creación de bases de datos: MySQL no ofrece la posibilidad de dar la capacidad a un usuario de crear cuantas bases de datos desee sin, al mismo tiempo, poder tener acceso al resto de bases de datos de modo que, mediante el formulario con dirección https://servidormysql.noip.org:8443/Servicios/FormularioBasesDeDatos.jsp, podrán crearse bases de datos con acceso único al usuario que las crea. Dicho formulario enviará los datos de usuario y el nombre de la base de datos hacia un documento llamado *BasesdeDatosMYSQL.jsp*, desde el cual se ejecutará la sentencia MySQL necesaria para la creación del usuario. Estos formularios también se encuentran ubicados en un servidor Apache Tomcat en el mismo ordenador donde se encuentra el servidor MySQL. Dicha arquitectura se muestra a continuación en la **Figura 3.9**

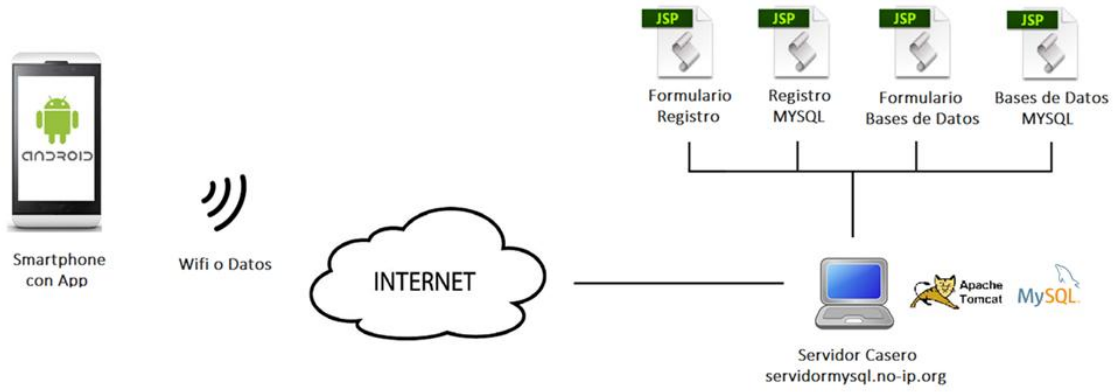


Figura 3.9 Arquitectura servidor www.servidormysql.no-ip.org

3.5.1.3 Arquitectura servidor móvil

En este caso cliente y servidor formarán parte del mismo dispositivo, es decir, se pasará a gestionar las tablas de la base de datos con nombre 'BaseMySQLite' ubicada en el terminal móvil y las sentencias, en este caso de tipo SQLite, se ejecutarán en el mismo terminal. En este entorno se contarán con menos capacidades de gestión que en los casos anteriores ya que SQLite ofrece menos funcionalidades en este sentido que MySQL.

Figura 3.10 Arquitectura servidor móvil

Aunque no se muestre en ninguna arquitectura, la aplicación móvil contiene una función que permite traspasar registros entre tablas SQLite ubicadas en el dispositivo a las tablas de servidores MySQL externos y también en sentido contrario.

3.5.2 Diseño de la lógica de negocio

En este apartado se especificarán los diferentes casos de uso que podrán darse en el uso de la aplicación. Como en apartados anteriores se tendrá en cuenta el servidor con el que esté trabajando la aplicación cliente ya que se podrán dar diferentes casos de uso según el servidor. De forma adicional se mostrarán unos diagramas de clases que ayudarán a entender mejor el funcionamiento de la aplicación.

3.5.2.1 Casos de uso

Se mostrarán a continuación cada una de las diferentes funcionalidades que la aplicación cliente ofrece para conseguir un servicio adecuado de gestión de bases de datos ubicadas en un servidor. En primer lugar cada función será clasificada según el servidor donde se ejecuten y más tarde, cada función será enmarcada en una tabla junto con una descripción de la misma, la actividad del proyecto a la que pertenece, los parámetros de entrada y salida que pudiera tener y las posibles excepciones que pudiesen ocurrir en la ejecución de la misma. Las funciones mencionadas mayoritariamente incluirán sentencias MySQL o SQLite que se enviarán desde la aplicación cliente hasta el servidor que las ejecute.

Durante el funcionamiento de la aplicación se distinguirán tres tipos de actores, el usuario no identificado con un servidor MySQL, el usuario haciendo uso de la base de datos 'BaseMySQLite.sqlite' ubicada dentro del dispositivo móvil y el usuario correctamente identificado con un servidor MySQL. De este último mencionado se diferenciarán dos casos, los usuarios normales y el administrador del servidor. La diferencia entre usuarios radicará en la cantidad de privilegios MySQL que posea, es decir, los usuarios tendrán todos los privilegios existentes sobre sus propias bases de datos y tablas, además de los compartidos por otros usuarios para otras tablas, mientras que por otro lado, el administrador tendrá todos los privilegios posibles sobre todas las bases de datos y tablas de todos los usuarios.

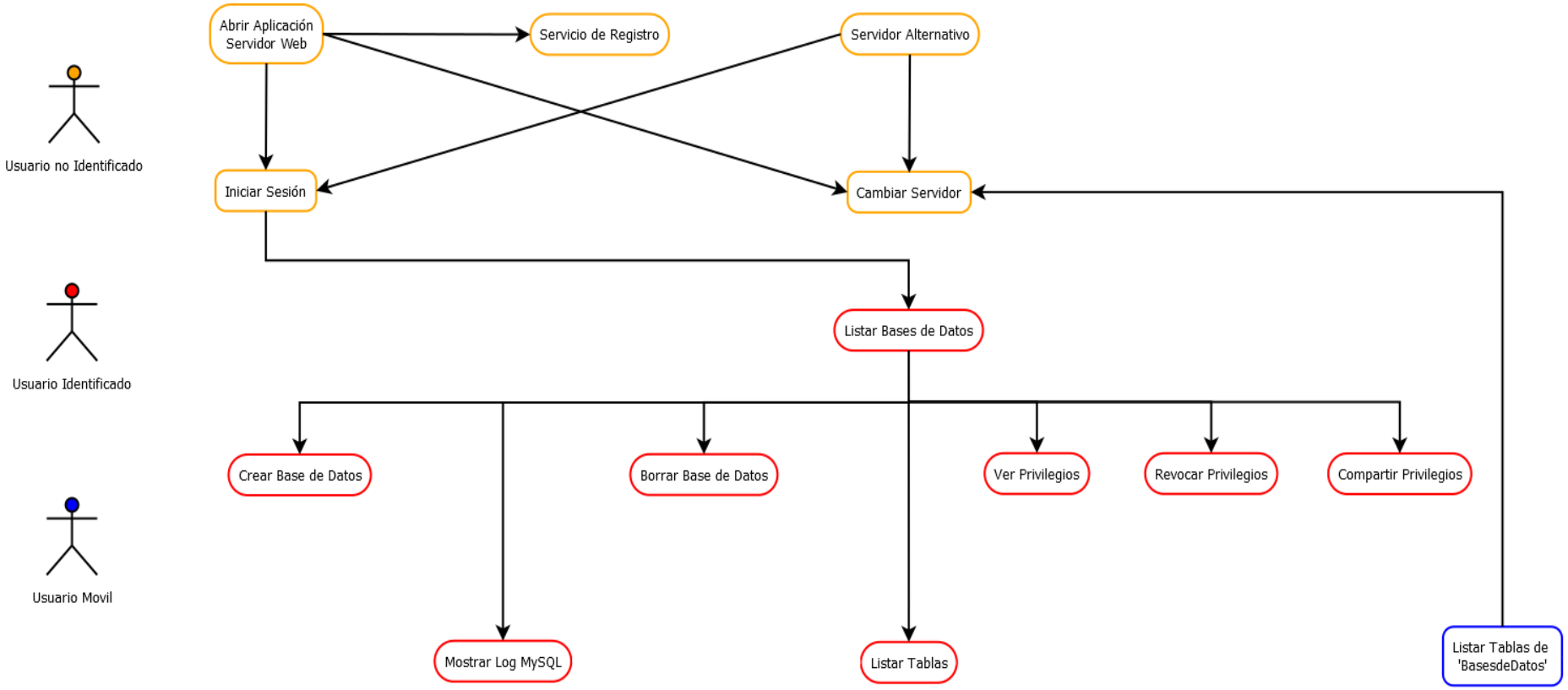


Figura 3.11 Diagrama de clases de uso parte 1

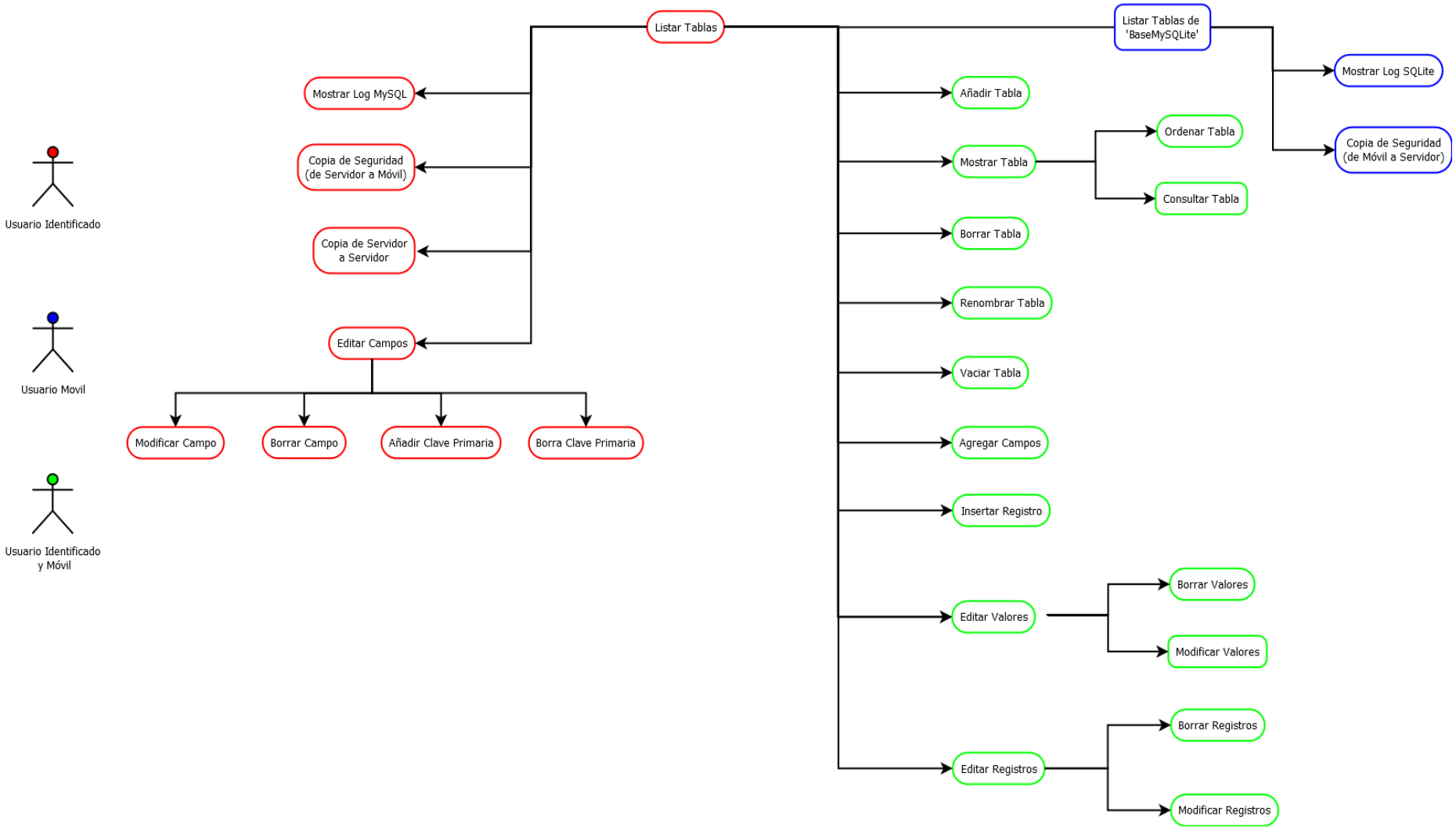


Figura 3.12 Diagrama de clases de uso parte 2

1. Casos de uso únicos en servidor www.servidormysql.no-ip.org.

<table>
<tr><th>Nombre</th><th>Inicio De Aplicación</th></tr>
<tr><td>Actividad</td><td>Login.java.</td></tr>
<tr><td>Actores</td><td>Usuario sin identificar.</td></tr>
<tr><td>Precondición</td><td>Ninguna.</td></tr>
<tr><td>Descripción</td><td>El usuario desde la aplicación accederá a esta actividad al abrir la aplicación.</td></tr>
<tr><td>Secuencia normal</td><td>
<table>
<tr><th>Pasos</th><th>Acción</th></tr>
<tr><td>1</td><td>El usuario abre la aplicación y accede directamente a la actividad Login.java o bien accede desde LoginAlternativo.java o desde ListarTablasMovil.java.</td></tr>
</table>
</td></tr>
<tr><td>Excepciones</td><td>Ninguna.</td></tr>
</table>

<table>
<tr><th>Nombre</th><th>Registro De Usuarios</th></tr>
<tr><td>Actividad</td><td>FormularioRegistro.jsp y RegistroMYSQL.jsp.</td></tr>
<tr><td>Actores</td><td>Usuario sin identificar.</td></tr>
<tr><td>Precondición</td><td>Ninguna.</td></tr>
<tr><td>Descripción</td><td>El usuario a través de un navegador cualquiera podrá registrarse en el servidor mediante la introducción de credenciales de usuario y contraseña, además de un nombre para la creación de una base de datos en el servidor a la que solo tenga acceso (privilegios MySQL) el usuario y el administrador del servidor.</td></tr>
<tr><td>Secuencia normal</td><td>
<table>
<tr><th>Pasos</th><th>Acción</th></tr>
<tr><td>1</td><td>Usuario introduce los datos en FormularioRegistro.jsp y se envían a RegistroMYSQL.jsp.</td></tr>
<tr><td>2</td><td>Se ejecutan sentencias MySQL para la creación del usuario, de la base de datos y de la asignación de privilegios</td></tr>
</table>
</td></tr>
<tr><td>Excepciones</td><td>Excepción 1. Ya existe un usuario con el nombre introducido.
Excepción 2. Ya existe una base de datos con el nombre introducido.
Excepción 3. La contraseña introducida es menor de seis caracteres.
Excepción 4. Java SQLException.
Excepción 5. Java Exception.</td></tr>
</table>

<table>
<tr><th>Nombre</th><th>Inicio De Sesión</th></tr>
<tr><td>Actividad</td><td>Login.java.</td></tr>
<tr><td>Actores</td><td>Usuario sin identificar.</td></tr>
<tr><td>Precondición</td><td>Ninguna.</td></tr>
<tr><td>Descripción</td><td>El usuario desde la aplicación se identifica ante el servidor con dirección www.servidormysql.no-ip.org mediante el uso de credenciales usuario y contraseña.</td></tr>
<tr><td>Secuencia normal</td><td>
<table>
<tr><th>Pasos</th><th>Acción</th></tr>
<tr><td>1</td><td>El usuario introduce sus credenciales de usuario y contraseña y las envía al servidor para identificarse.</td></tr>
<tr><td>2</td><td>El servidor comprueba las credenciales y el usuario queda autenticado.</td></tr>
<tr><td>3</td><td>La aplicación avanza a la actividad ListarBasesDeDatos.java.</td></tr>
</table>
</td></tr>
<tr><td>Excepciones</td><td>Excepción 1. Usuario o contraseña incorrectos.
Excepción 2. Java SQLException.
Excepción 3. Java Exception.</td></tr>
</table>

<table>
<tr><th>Nombre</th><th>Creación De Bases De Datos</th></tr>
<tr><td>Actividad</td><td>FormularioBasesDeDatos.jsp y BasesDeDatosMYSQL.jsp.</td></tr>
<tr><td>Actores</td><td>Usuario identificado.</td></tr>
<tr><td>Precondición</td><td>Ninguna.</td></tr>
<tr><td>Descripción</td><td>El usuario a través de un navegador cualquiera podrá crear una base de datos en el servidor. El usuario necesitará introducir sus credenciales de usuario y contraseña y el nombre de la base de datos que desee crear. Sobre dicha base de datos solo tendrán privilegios MySQL el usuario y el administrador del servidor.</td></tr>
<tr><td>Secuencia normal</td><td>
<table>
<tr><th>Pasos</th><th>Acción</th></tr>
<tr><td>1</td><td>Usuario introduce en FormularioBasesDeDatos.jsp los datos requeridos y se envían a BasesDeDatosMYSQL.jsp</td></tr>
<tr><td>2</td><td>Se identifica correctamente al usuario.</td></tr>
<tr><td>3</td><td>Se ejecutan sentencias MySQL para la creación de la base de datos y asignación de privilegios al usuario.</td></tr>
</table>
</td></tr>
<tr><td>Excepciones</td><td>Excepción 1. Usuario o contraseña incorrectos.
Excepción 2. Ya existe una base de datos con el nombre introducido.
Excepción 3. Java SQLException.
Excepción 4. Java Exception.</td></tr>
</table>

2. Casos de uso únicos en servidor alternativo.

<table>
<tr><th>Nombre</th><th>Inicio De Sesión</th></tr>
<tr><td>Actividad</td><td>LoginAlternativo.java.</td></tr>
<tr><td>Actores</td><td>Usuario identificado.</td></tr>
<tr><td>Precondición</td><td>Ninguna.</td></tr>
<tr><td>Descripción</td><td>El usuario desde la aplicación se identifica mediante la introducción de credenciales de usuario y contraseña y de la dirección IP del servidor MySQL ante el cual desea autenticarse.</td></tr>
<tr><td>Secuencia normal</td><td>
<table>
<tr><th>Pasos</th><th>Acción</th></tr>
<tr><td>1</td><td>El usuario introduce sus credenciales de usuario y contraseña y la dirección IP del servidor las envía al servidor para identificarse.</td></tr>
<tr><td>2</td><td>El servidor comprueba las credenciales y el usuario queda autenticado.</td></tr>
<tr><td>3</td><td>La aplicación avanza a la actividad ListarBasesDeDatos.java</td></tr>
</table>
</td></tr>
<tr><td>Excepciones</td><td>Excepción 1. Usuario o contraseña incorrectos.
Excepción 2. Java SQLException.
Excepción 3. Java Exception.</td></tr>
</table>

<table>
<tr><th>Nombre</th><th>Crear Base De Datos</th></tr>
<tr><td>Actividad</td><td>ListarBasesDeDatos.java.</td></tr>
<tr><td>Actores</td><td>Usuario identificado.</td></tr>
<tr><td>Precondición</td><td>El usuario debe estar identificado frente al servidor.</td></tr>
<tr><td>Descripción</td><td>El usuario desde la aplicación podrá crear una base de datos en el servidor si tiene el permiso requerido.</td></tr>
<tr><td>Secuencia normal</td><td>
<table>
<tr><th>Pasos</th><th>Acción</th></tr>
<tr><td>1</td><td>El usuario introduce el nombre de la base de datos a crear.</td></tr>
<tr><td>2</td><td>La sentencia MySQL es enviada al servidor y ejecutada en el mismo</td></tr>
</table>
</td></tr>
<tr><td>Excepciones</td><td>Excepción 1. Java SQLException.
Excepción 2. Java Exception.</td></tr>
</table>

3. Casos de uso dados en servidor web y servidor alternativo.

<table>
<tr><th>Nombre</th><th>Obtener Lista Bases De Datos</th></tr>
<tr><td>Actividad</td><td>ListarBasesDeDatos.java.</td></tr>
<tr><td>Actores</td><td>Usuario identificado.</td></tr>
<tr><td>Precondición</td><td>El usuario debe estar identificado frente al servidor.</td></tr>
<tr><td>Descripción</td><td>El usuario desde la aplicación accederá a la lista de las bases de datos de las que tenga permiso ubicadas en un servidor MySQL.</td></tr>
<tr><td>Secuencia normal</td><td>
<table>
<tr><th>Pasos</th><th>Acción</th></tr>
<tr><td>1</td><td>El usuario desde Login.java o LoginAlternativo.java pulsa el botón 'Iniciar Sesión'.</td></tr>
<tr><td>2</td><td>La actividad ListarBasesDeDatos.java es abierta y automáticamente la sentencia MySQL que permite obtener las bases de datos se envía al servidor.</td></tr>
<tr><td>3</td><td>La sentencia MySQL es ejecutada en el servidor y la aplicación recibe la respuesta con la lista de bases de datos que finalmente muestra al usuario.</td></tr>
</table>
</td></tr>
<tr><td>Excepciones</td><td>Excepción 1. Java SQLException.
Excepción 2. Java Exception.</td></tr>
</table>

<table>
<tr><th>Nombre</th><th>Borrar Base De Datos</th></tr>
<tr><td>Actividad</td><td>ListarBasesDeDatos.java.</td></tr>
<tr><td>Actores</td><td>Usuario identificado.</td></tr>
<tr><td>Precondición</td><td>El usuario debe estar identificado frente al servidor.</td></tr>
<tr><td>Descripción</td><td>El usuario desde la aplicación podrá eliminar las bases de datos de las que tenga acceso ubicadas en un servidor MySQL.</td></tr>
<tr><td>Secuencia normal</td><td>
<table>
<tr><th>Pasos</th><th>Acción</th></tr>
<tr><td>1</td><td>El usuario selecciona la base de datos a eliminar y pulsa la opción 'Borrar' del menú.</td></tr>
<tr><td>2</td><td>La sentencia MySQL para eliminar la base de datos es enviada y al servidor y ejecutada en el mismo.</td></tr>
</table>
</td></tr>
<tr><td>Excepciones</td><td>Excepción 1. Java SQLException.
Excepción 2. Java Exception.</td></tr>
</table>

<table>
<tr><th>Nombre</th><th>Ver Privilegios</th></tr>
<tr><td>Actividad</td><td>ListarBasesDeDatos.java.</td></tr>
<tr><td>Actores</td><td>Usuario identificado.</td></tr>
<tr><td>Precondición</td><td>El usuario debe estar identificado frente al servidor.</td></tr>
<tr><td>Descripción</td><td>El usuario desde la aplicación podrá obtener todos los privilegios MySQL que tiene en un servidor MySQL.</td></tr>
<tr><td>Secuencia normal</td><td>
<table>
<tr><th>Pasos</th><th>Acción</th></tr>
<tr><td>1</td><td>El usuario selecciona la opción 'Ver Privilegios' del menú.</td></tr>
<tr><td>2</td><td>El usuario es redirigido a la actividad MostrarTabla.java</td></tr>
<tr><td>3</td><td>La sentencia MySQL para obtener los privilegios es enviada al servidor y ejecutada en el mismo.</td></tr>
<tr><td>4</td><td>El servidor devuelve los privilegios del usuario a la aplicación y esta los muestra al usuario.</td></tr>
</table>
</td></tr>
<tr><td>Excepciones</td><td>Excepción 1. Java SQLException.
Excepción 2. Java Exception.</td></tr>
</table>

<table>
<tr><th>Nombre</th><th>Proporcionar Privilegios</th></tr>
<tr><td>Actividad</td><td>ListarBasesDeDatos.java.</td></tr>
<tr><td>Actores</td><td>Usuario identificado.</td></tr>
<tr><td>Precondición</td><td>El usuario debe estar identificado frente al servidor.</td></tr>
<tr><td>Descripción</td><td>El usuario desde la aplicación podrá otorgar diferentes privilegios a otros usuarios sobre una o todas las tablas pertenecientes a una base de datos de las que tenga acceso ubicadas en un servidor MySQL.</td></tr>
<tr><td>Secuencia normal</td><td>
<table>
<tr><th>Pasos</th><th>Acción</th></tr>
<tr><td>1</td><td>El usuario selecciona la opción 'Compartir Privilegios' del menú para abrir un menú de privilegios.</td></tr>
<tr><td>2</td><td>El usuario introducirá el nombre del usuario a privilegiar, la base de datos sobre la que otorgar privilegios, la tabla o todas las tablas de la base de datos a privilegiar y señalará los privilegios a otorgar y la capacidad de otorgar o no los privilegios señalados a otros usuarios.</td></tr>
<tr><td>3</td><td>El usuario pulsara sobre el botón 'Otorgar' para compartir los privilegios seleccionados o sobre el botón 'Otorgar Todos' para otorgar todos los privilegios MySQL que el usuario posea sobre las tablas seleccionadas.</td></tr>
<tr><td>4</td><td>La sentencia MySQL para otorgar privilegios es enviada al servidor y posteriormente ejecutada.</td></tr>
</table>
</td></tr>
<tr><td>Excepciones</td><td>Excepción 1. Java SQLException.
Excepción 2. Java Exception.</td></tr>
</table>

<table>
<tr><th>Nombre</th><th>Revocar Privilegios</th></tr>
<tr><td>Actividad</td><td>ListarBasesDeDatos.java.</td></tr>
<tr><td>Actores</td><td>Usuario identificado.</td></tr>
<tr><td>Precondición</td><td>El usuario debe estar identificado frente al servidor.</td></tr>
<tr><td>Descripción</td><td>El usuario desde la aplicación podrá revocar diferentes privilegios a otros usuarios sobre una o todas las tablas pertenecientes a una base de datos de las que tenga acceso ubicadas en un servidor MySQL.</td></tr>
<tr><td>Secuencia normal</td><td>
<table>
<tr><th>Pasos</th><th>Acción</th></tr>
<tr><td>1</td><td>El usuario selecciona la opción 'Compartir Privilegios' del menú para abrir un menú de privilegios.</td></tr>
<tr><td>2</td><td>El usuario introducirá el nombre del usuario a quien revocar los privilegios, la base de datos y la tabla o todas las tablas de la base de datos de las que revocar privilegios, señalará los privilegios a revocar y la capacidad de otorgar o no privilegios a otros usuarios de estas tablas.</td></tr>
<tr><td>3</td><td>El usuario pulsara sobre el botón 'Revocar' para revocar los privilegios seleccionados o sobre el botón 'Revocar Todos' para revocar todos los privilegios MySQL que el usuario posea sobre las tablas seleccionadas.</td></tr>
<tr><td>4</td><td>La sentencia MySQL para revocar privilegios es enviada al servidor y posteriormente ejecutada.</td></tr>
</table>
</td></tr>
<tr><td>Excepciones</td><td>Excepción 1. Java SQLException.
Excepción 2. Java Exception.</td></tr>
</table>

<table>
<tr><th>Nombre</th><th>Obtener Lista De Tablas</th></tr>
<tr><td>Actividad</td><td>ListarTablas.java.</td></tr>
<tr><td>Actores</td><td>Usuario identificado.</td></tr>
<tr><td>Precondición</td><td>El usuario debe estar identificado frente al servidor.</td></tr>
<tr><td>Descripción</td><td>El usuario desde la aplicación accederá a una lista con todas las tablas pertenecientes a una base de datos anteriormente elegida.</td></tr>
<tr><td>Secuencia normal</td><td>
<table>
<tr><th>Pasos</th><th>Acción</th></tr>
<tr><td>1</td><td>El usuario desde ListarBasesDeDatos.java seleccionará una base de datos y pulsará el botón 'Cargar'.</td></tr>
<tr><td>2</td><td>La aplicación se dirige a la actividad ListarTablas.java donde automáticamente la sentencia MySQL que permite obtener las tablas pertenecientes a una base de datos se envía al servidor.</td></tr>
<tr><td>3</td><td>La sentencia MySQL es ejecutada en el servidor y la aplicación recibe la respuesta con las tablas que finalmente muestra al usuario.</td></tr>
</table>
</td></tr>
<tr><td>Excepciones</td><td>Excepción 1. Java SQLException.
Excepción 2. Java Exception.</td></tr>
</table>

<table>
<tr><th>Nombre</th><th>Obtener Log De MySQL</th></tr>
<tr><td>Actividad</td><td>LogSQL.java.</td></tr>
<tr><td>Actores</td><td>Usuario identificado.</td></tr>
<tr><td>Precondición</td><td>El usuario debe estar identificado frente al servidor.</td></tr>
<tr><td>Descripción</td><td>El usuario desde la aplicación accederá a una lista con todas las sentencias enviadas al servidor MySQL al que se está actualmente conectado.</td></tr>
<tr><td>Secuencia normal</td><td>
<table>
<tr><th>Pasos</th><th>Acción</th></tr>
<tr><td>1</td><td>El usuario desde ListarBasesDeDatos.java o ListarTablas.java pulsará la opción 'Log Comandos SQL'</td></tr>
<tr><td>2</td><td>La aplicación se dirige a la actividad LogSQL.java donde automáticamente se muestra una lista con todas las sentencias enviadas al servidor MySQL con el que se está actualmente autenticado.</td></tr>
</table>
</td></tr>
<tr><td>Excepciones</td><td>Ninguna</td></tr>
</table>

<table>
<tr><th>Nombre</th><th>Editar Campos</th></tr>
<tr><td>Actividad</td><td>Editar.java.</td></tr>
<tr><td>Actores</td><td>Usuario identificado.</td></tr>
<tr><td>Precondición</td><td>El usuario debe estar identificado frente al servidor.</td></tr>
<tr><td>Descripción</td><td>El usuario desde la aplicación podrá modificar diferentes parámetros de los campos pertenecientes a una tabla.</td></tr>
<tr><td>Secuencia normal</td><td>
<table>
<tr><th>Pasos</th><th>Acción</th></tr>
<tr><td>1</td><td>El usuario selecciona una de la lista de tablas pertenecientes a una base de datos en ListarTablas.java y luego pulsa sobre la opción 'Editar Campos'.</td></tr>
<tr><td>2</td><td>La aplicación se dirige a la actividad Editar.java donde aparecerá un menú con todos los parámetros que caracterizan a un campo el cual será seleccionado por el usuario.</td></tr>
<tr><td>3</td><td>El usuario selecciona el campo del cual quiere alterar algunos de sus parámetros, tras esto aparecerá un menú con los parámetros actuales del campo elegido y con las diferentes opciones de las que dispondrá el usuario para alterar dichos parámetros. El usuario pulsará sobre el botón 'Modificar' para alterar los parámetros indicados en los campos o bien sobre 'Borrar' para eliminar el campo elegido.</td></tr>
<tr><td>4</td><td>La sentencia MySQL que permite la modificación de los parámetros del campo o bien, la que permite la eliminación del mismo, es enviada al servidor y ejecutada en el mismo.</td></tr>
</table>
</td></tr>
<tr><td>Excepciones</td><td>Excepción 1. Java SQLException
Excepción 2. Java Exception.</td></tr>
</table>

<table>
<tr><th>Nombre</th><th>Añadir Y Borrar Claves Primarias</th></tr>
<tr><td>Actividad</td><td>Editar.java.</td></tr>
<tr><td>Actores</td><td>Usuario identificado.</td></tr>
<tr><td>Precondición</td><td>El usuario debe estar identificado frente al servidor.</td></tr>
<tr><td>Descripción</td><td>El usuario desde la aplicación podrá añadir la propiedad de clave primaria a los campos de una tabla o bien borrar la propiedad de clave primaria de todos los campos de una tabla.</td></tr>
<tr><td>Secuencia normal</td><td>
<table>
<tr><th>Pasos</th><th>Acción</th></tr>
<tr><td>1</td><td>El usuario selecciona una de la lista de tablas pertenecientes a una base de datos en ListarTablas.java y luego pulsa sobre la opción 'Editar Campos'.</td></tr>
<tr><td>2</td><td>La aplicación se dirige a la actividad Editar.java donde aparecerá un menú con un botón 'Pri.Key' el cual pulsará el usuario.</td></tr>
<tr><td>3</td><td>El usuario podrá pulsar el botón 'Borrar' para eliminar la propiedad de clave primaria de la tabla o bien elegir un campo de la lista y pulsar el botón 'Añadir' para añadir el campo elegido como parte de la clave primaria de la tabla</td></tr>
<tr><td>4</td><td>La sentencia MySQL que permite la eliminación de la propiedad de clave primaria en la tabla o bien la que permite añadir el campo como parte de la clave primaria de la tabla es enviada al servidor y ejecutada en el mismo.</td></tr>
</table>
</td></tr>
<tr><td>Excepciones</td><td>Excepción 1. Java SQLException
Excepción 2. Java Exception.</td></tr>
</table>

Nombre	Añadir Claves Foráneas
Actividad	ClavesForaneas.java.
Actores	Usuario identificado.
Precondición	El usuario debe estar identificado frente al servidor.
Descripción	El usuario desde la aplicación podrá aplicar una propiedad de clave foránea a un campo de una tabla de acuerdo a un campo perteneciente a otra tabla, el cual debe tener propiedad de clave primaria de esta.

<table>
<tr><td>Secuencia normal</td><td>
<table>
<tr><th>Pasos</th><th>Acción</th></tr>
<tr><td>1</td><td>El usuario selecciona desde la actividad ListarTablas.java la opción 'Claves Foráneas' en el menú.</td></tr>
<tr><td>2</td><td>La aplicación se dirige a la actividad ClavesForaneas.java donde aparece un menú adecuado para la creación de claves foráneas.</td></tr>
<tr><td>3</td><td>El usuario selecciona, por un lado, la tabla y el campo de esa tabla al que se le asignará la clave foránea y por otro, la tabla y el campo de referencia diferentes a los anteriores. Por último el usuario selecciona el modo en el que la tabla actuará en función de si los registros de la tabla de referencia son borrados o modificados y pulsará sobre el botón 'Añadir'.</td></tr>
<tr><td>4</td><td>La sentencia MySQL que permite la asignación de la clave foránea según los parámetros introducidos es enviada al servidor y ejecutada en el mismo.</td></tr>
</table>
</td></tr>
<tr><td>Excepciones</td><td>Excepción 1. Java SQLException.
Excepción 2. Java Exception.</td></tr>
</table>

<table>
<tr><th>Nombre</th><th>Borrar Claves Foráneas</th></tr>
<tr><td>Actividad</td><td>ClavesForaneas.java.</td></tr>
<tr><td>Actores</td><td>Usuario identificado.</td></tr>
<tr><td>Precondición</td><td>El usuario debe estar identificado frente al servidor.</td></tr>
<tr><td>Descripción</td><td>El usuario desde la aplicación podrá eliminar la propiedad de clave foránea de un campo perteneciente a una tabla.</td></tr>
<tr><td>Secuencia normal</td><td>
<table>
<tr><th>Pasos</th><th>Acción</th></tr>
<tr><td>1</td><td>El usuario selecciona desde la actividad ListarTablas.java la opción 'Claves Foráneas' en el menú.</td></tr>
<tr><td>2</td><td>La aplicación se dirige a la actividad ClavesForaneas.java donde aparecerá un menú adecuado para la eliminación de claves foráneas.</td></tr>
<tr><td>3</td><td>El usuario selecciona la tabla, el campo de esa tabla el cual contiene una propiedad de clave foránea y la restricción que actúa como identificador de la clave foránea y finalmente pulsa sobre el botón 'Borrar'.</td></tr>
<tr><td>4</td><td>La sentencia MySQL que permite la eliminación de propiedad de la clave foránea según los parámetros introducidos es enviada al servidor y ejecutada en el mismo.</td></tr>
</table>
</td></tr>
<tr><td>Excepciones</td><td>Excepción 1. Java SQLException.
Excepción 2. Java Exception.</td></tr>
</table>

<table>
<tr><th>Nombre</th><th>Copia De Seguridad De Servidor A Terminal Móvil</th></tr>
<tr><td>Actividad</td><td>ListarTablas.java.</td></tr>
<tr><td>Actores</td><td>Usuario identificado.</td></tr>
<tr><td>Precondición</td><td>Ninguna.</td></tr>
<tr><td>Descripción</td><td>El usuario desde la aplicación podrá copiar todos los registros de una tabla ubicada en un servidor MySQL a una tabla con el mismo nombre situada en la base de datos con nombre 'BaseMySQLite' encontrada en el dispositivo móvil.</td></tr>
<tr><td>Secuencia normal</td><td>
<table>
<tr><th>Pasos</th><th>Acción</th></tr>
<tr><td>1</td><td>El usuario desde ListarTablas.java seleccionará una de la lista de tablas y luego pulsará la opción 'Copia de Seguridad'.</td></tr>
<tr><td>2</td><td>La sentencia MySQL que permite la obtención de los campos y registros de la tabla seleccionada es enviada al servidor y ejecutada en el mismo.</td></tr>
<tr><td>3</td><td>La sentencia MySQL que permite la obtención de los parámetros de cada uno de los campos de la tabla seleccionada es enviada al servidor y ejecutada en el mismo.</td></tr>
<tr><td>4</td><td>La sentencia SQLite que permite la creación de la tabla en la base de datos con nombre 'BaseMySQLite' ubicada en el terminal móvil de acuerdo a los parámetros antes obtenidos es ejecutada en el terminal. Si ya existiese una tabla con el mismo nombre en la base de datos no se crearía dicha tabla y se pasará al siguiente paso de cualquier manera.</td></tr>
<tr><td>5</td><td>Las sentencias SQLite que permiten la inserción de cada uno de los registros anteriormente obtenidos en la tabla anteriormente creada en el dispositivo móvil (o bien en la que ya existiese con el mismo nombre) son ejecutadas.</td></tr>
</table>
</td></tr>
<tr><td>Excepciones</td><td>Excepción 1. Java SQLException.
Excepción 2. Java Exception.</td></tr>
</table>

<table>
<tr><th>Nombre</th><th>Copia De Servidor A Servidor</th></tr>
<tr><td>Actividad</td><td>ListarTablas.java.</td></tr>
<tr><td>Actores</td><td>Usuario identificado.</td></tr>
<tr><td>Precondición</td><td>Ninguna.</td></tr>
<tr><td>Descripción</td><td>El usuario desde la aplicación podrá copiar una tabla ubicada en un servidor MySQL a otro servidor junto con todos sus registros y propiedades.</td></tr>
<tr><td>Secuencia normal</td><td>
<table>
<tr><th>Pasos</th><th>Acción</th></tr>
<tr><td>1</td><td>El usuario desde ListarTablas.java seleccionará una de la lista de tablas y luego pulsará la opción 'Copia a Servidor'.</td></tr>
<tr><td>2</td><td>La aplicación mostrará un menú para que el usuario introduzca la dirección IP del servidor MySQL destino, el nombre de la base de datos donde se ubicará la tabla y las credenciales de usuario y contraseña para la identificación del usuario.</td></tr>
<tr><td>3</td><td>La sentencia MySQL que permite la obtención de los campos y registros de la tabla seleccionada es enviada al servidor y ejecutada en el mismo.</td></tr>
<tr><td>4</td><td>La sentencia MySQL que permite la obtención de los parámetros de cada uno de los campos de la tabla seleccionada es enviada al servidor y ejecutada en el mismo.</td></tr>
<tr><td>5</td><td>La sentencia MySQL que permite la creación de la tabla en el servidor MySQL elegido de acuerdo a los parámetros antes obtenidos es enviada a y ejecutada en dicho servidor. Si ya existiese una tabla con el mismo nombre en la base de datos destinataria no se crearía dicha tabla y se pasará al siguiente paso de cualquier manera.</td></tr>
<tr><td>6</td><td>Las sentencias MySQL que permiten la inserción de cada uno de los registros anteriormente obtenidos en la tabla anteriormente creada en el servidor MySQL elegido (o bien en la que ya existiese con el mismo nombre), son enviadas a y ejecutada en dicho servidor.</td></tr>
</table>
</td></tr>
<tr><td>Excepciones</td><td>Excepción 1. Java SQLException.
Excepción 2. Java Exception.</td></tr>
</table>

4. Casos de uso únicos en servidor móvil

<table>
<tr><th>Nombre</th><th>Obtener Lista De Tablas Móvil</th></tr>
<tr><td>Actividad</td><td>ListarTablasMovil.java.</td></tr>
<tr><td>Actores</td><td>Usuario móvil.</td></tr>
<tr><td>Precondición</td><td>Ninguna.</td></tr>
<tr><td>Descripción</td><td>El usuario desde la aplicación accederá a una lista con todas las tablas pertenecientes a la base de datos con nombre 'BaseMySQLite' ubicada en el dispositivo. Si no existiese tal base de datos la aplicación la crearía.</td></tr>
<tr><td>Secuencia normal</td><td>
<table>
<tr><th>Pasos</th><th>Acción</th></tr>
<tr><td>1</td><td>El usuario podrá acceder a la actividad ListarTablasMovil.java desde Login.java o desde LoginAlternativo.java.</td></tr>
<tr><td>2</td><td>La aplicación automáticamente ejecuta la sentencia SQLite que permite obtener las tablas pertenecientes a la base de datos llamada 'BaseMySQLite' ubicada en el dispositivo.</td></tr>
<tr><td>3</td><td>Finalmente dichas tablas son mostradas por la aplicación al usuario.</td></tr>
</table>
</td></tr>
<tr><td>Excepciones</td><td>Excepción 1. Java SQLiteException.
Excepción 2. Java Exception.</td></tr>
</table>

<table>
<tr><th>Nombre</th><th>Obtener Log De SQLite</th></tr>
<tr><td>Actividad</td><td>LogSQLMovil.java.</td></tr>
<tr><td>Actores</td><td>Usuario móvil.</td></tr>
<tr><td>Precondición</td><td>El usuario debe estar identificado frente al servidor.</td></tr>
<tr><td>Descripción</td><td>El usuario desde la aplicación accederá a una lista con todas las tablas pertenecientes a la base de datos con nombre 'BaseMySQLite' ubicada en el dispositivo.</td></tr>
<tr><td>Secuencia normal</td><td>
<table>
<tr><th>Pasos</th><th>Acción</th></tr>
<tr><td>1</td><td>El usuario desde la actividad ListarTablasMovil.java pulsará sobre la opción 'Log Comandos SQL'.</td></tr>
<tr><td>2</td><td>La aplicación se dirige a la actividad LogSQLMovil.java donde automáticamente se muestra una lista con todas las sentencias SQLite realizadas por la aplicación.</td></tr>
</table>
</td></tr>
<tr><td>Excepciones</td><td>Ninguna</td></tr>
</table>

<table>
<tr><th>Nombre</th><th>Copia De Seguridad De Terminal Móvil A Servidor</th></tr>
<tr><td>Actividad</td><td>ListarTablasMovil.java.</td></tr>
<tr><td>Actores</td><td>Usuario móvil.</td></tr>
<tr><td>Precondición</td><td>Ninguna.</td></tr>
<tr><td>Descripción</td><td>El usuario desde la aplicación podrá copiar todos los registros de una tabla encontrada en la base de datos con nombre 'BaseMySQLite' ubicada en el dispositivo móvil, a una tabla con el mismo nombre que la anterior que se encuentre en un servidor MySQL externo.</td></tr>
<tr><td>Secuencia normal</td><td>
<table>
<tr><th>Pasos</th><th>Acción</th></tr>
<tr><td>1</td><td>El usuario desde ListarTablasMovil.java seleccionará una de la lista de tablas y luego pulsará la opción 'Copia de Seguridad'.</td></tr>
<tr><td>2</td><td>La aplicación mostrará un menú para que el usuario introduzca la dirección IP del servidor MySQL destino, el nombre de la base de datos donde se ubica la tabla destinataria y las credenciales de usuario y contraseña para la identificación del usuario.</td></tr>
<tr><td>3</td><td>La sentencia SQLite que permite la obtención de los campos y registros de la tabla seleccionada es ejecutada en el dispositivo.</td></tr>
<tr><td>4</td><td>La sentencia SQLite que permite la obtención de los parámetros de cada uno de los campos de la tabla seleccionada es ejecutada en el dispositivo.</td></tr>
<tr><td>5</td><td>La sentencia MySQL que permite la creación de la tabla en el servidor MySQL elegido de acuerdo a los parámetros antes obtenidos es enviada a y ejecutada en dicho servidor. Si ya existiese una tabla con el mismo nombre en la base de datos destinataria no se crearía dicha tabla y se pasará al siguiente paso de cualquier manera.</td></tr>
<tr><td>6</td><td>Las sentencias MySQL que permiten la inserción de cada uno de los registros anteriormente obtenidos en la tabla anteriormente creada en el servidor MySQL elegido (o bien en la que ya existiese con el mismo nombre), son enviadas a y ejecutada en dicho servidor.</td></tr>
</table>
</td></tr>
<tr><td>Excepciones</td><td>Excepción 1. Java SQLiteException.
Excepción 2. Java Exception.</td></tr>
</table>

5. Casos de uso dados en todos los servidores

<table>
<tr><th>Nombre</th><th>Mostrar Tabla</th></tr>
<tr><td>Actividad</td><td>MostrarTabla.java y MostrarTablaMovil.java.</td></tr>
<tr><td>Actores</td><td>Usuario identificado y usuario móvil.</td></tr>
<tr><td>Precondición</td><td>El usuario debe estar identificado frente al servidor.</td></tr>
<tr><td>Descripción</td><td>El usuario desde la aplicación selecciona observa todos los registros almacenados en una tabla determinada.</td></tr>
<tr><td>Secuencia normal</td><td>
<table>
<tr><th>Pasos</th><th>Acción</th></tr>
<tr><td>1</td><td>El usuario selecciona una tabla de la lista en ListarTablas.java o ListarTablasMovil.java y pulsa sobre el botón 'Ver'.</td></tr>
<tr><td>2</td><td>La aplicación se dirige a la actividad MostrarTabla.java o MostrarTablaMovil.java donde la sentencia MySQL o SQLite que permite la obtención de los registros de la tabla es enviada al servidor y ejecutada en el mismo.</td></tr>
<tr><td>3</td><td>Finalmente la tabla con sus registros es expuesta al usuario.</td></tr>
</table>
</td></tr>
<tr><td>Excepciones</td><td>Excepción 1. Java SQLException o SQLiteException.
Excepción 2. Java Exception.</td></tr>
</table>

<table>
<tr><th>Nombre</th><th>Mostrar Tabla Ordenada</th></tr>
<tr><td>Actividad</td><td>MostrarTabla.java y MostrarTablaMovil.java.</td></tr>
<tr><td>Actores</td><td>Usuario identificado y usuario móvil.</td></tr>
<tr><td>Precondición</td><td>El usuario debe estar identificado frente al servidor.</td></tr>
<tr><td>Descripción</td><td>El usuario desde la aplicación selecciona observa todos los registros almacenados en una tabla determinada en un orden preseleccionado.</td></tr>
<tr><td>Secuencia normal</td><td>
<table>
<tr><th>Pasos</th><th>Acción</th></tr>
<tr><td>1</td><td>El usuario desde MostrarTabla.java o MostrarTablaMovil.java pulsa sobre la opción 'Ordenar Resultados'.</td></tr>
<tr><td>2</td><td>Se mostrará un pequeño menú que permite elegir el campo y el orden según los cuales los registros de la tabla serán ordenados y un botón 'Mostrar' que el usuario tendrá que pulsar.</td></tr>
<tr><td>3</td><td>La sentencia MySQL o SQLite que permite la obtención de los registros de la tabla en el orden elegido es enviada al servidor y ejecutada en el mismo, finalmente la tabla con sus registros ordenados son expuestos al usuario.</td></tr>
</table>
</td></tr>
<tr><td>Excepciones</td><td>Excepción 1. Java SQLException o SQLiteException.
Excepción 2. Java Exception.</td></tr>
</table>

<table>
<tr><th>Nombre</th><th>Consultar Tabla</th></tr>
<tr><td>Actividad</td><td>MostrarTabla.java y MostrarTablaMovil.java.</td></tr>
<tr><td>Actores</td><td>Usuario identificado y usuario móvil.</td></tr>
<tr><td>Precondición</td><td>El usuario debe estar identificado frente al servidor.</td></tr>
<tr><td>Descripción</td><td>El usuario desde la aplicación puede observar todos los registros almacenados en una tabla que concuerden con un campo y un valor introducidos.</td></tr>
<tr><td>Secuencia normal</td><td>
<table>
<tr><th>Pasos</th><th>Acción</th></tr>
<tr><td>1</td><td>El usuario desde MostrarTabla.java o MostrarTablaMovil.java pulsa sobre la opción 'Seleccionar Registros'.</td></tr>
<tr><td>2</td><td>Se mostrará un pequeño menú que permite elegir un campo de la tabla e introducir un valor y finalmente un botón 'Mostrar' que el usuario tendrá que pulsar.</td></tr>
<tr><td>3</td><td>La sentencia MySQL o SQLite que permite la obtención de los registros de la tabla según los datos introducidos es enviada al servidor y ejecutada en el mismo y los registros son finalmente mostrados al usuario.</td></tr>
</table>
</td></tr>
<tr><td>Excepciones</td><td>Excepción 1. Java SQLException o SQLiteException.
Excepción 2. Java Exception.</td></tr>
</table>

<table>
<tr><th>Nombre</th><th>Añadir Tabla</th></tr>
<tr><td>Actividad</td><td>CrearTablas.java y CrearTablasMovil.java.</td></tr>
<tr><td>Actores</td><td>Usuario identificado y usuario móvil.</td></tr>
<tr><td>Precondición</td><td>El usuario debe estar identificado frente al servidor.</td></tr>
<tr><td>Descripción</td><td>El usuario desde la aplicación podrá crear una tabla de acuerdo a un conjunto de campos introducidos y a un conjunto de parámetros dentro de estos campos.</td></tr>
<tr><td>Secuencia normal</td><td>
<table>
<tr><th>Pasos</th><th>Acción</th></tr>
<tr><td>1</td><td>El usuario desde ListarTablas.java o ListarTablasMovil.java pulsa el botón 'Añadir'.</td></tr>
<tr><td>2</td><td>Una ventana aparece frente al usuario para introducir el nombre de la nueva tabla.</td></tr>
<tr><td>3</td><td>La aplicación se dirige a la actividad CrearTablas.java o CrearTablasMovil.java donde aparecerá un menú adecuado para la creación de tablas.</td></tr>
<tr><td>4</td><td>El usuario introduce los campos que precise y los parámetros necesarios pertenecientes a estos campos y pulsa el botón 'Crear'.</td></tr>
<tr><td>5</td><td>La sentencia MySQL o SQLite que permite la creación de la tabla es enviada al servidor y ejecutada en el mismo.</td></tr>
</table>
</td></tr>
<tr><td>Excepciones</td><td>Excepción 1. Java SQLException o SQLiteException.
Excepción 2. Java Exception.</td></tr>
</table>

<table>
<tr><th>Nombre</th><th>Borrar Tabla</th></tr>
<tr><td>Actividad</td><td>ListarTablas.java y ListarTablasMovil.java.</td></tr>
<tr><td>Actores</td><td>Usuario identificado y usuario móvil.</td></tr>
<tr><td>Precondición</td><td>El usuario debe estar identificado frente al servidor.</td></tr>
<tr><td>Descripción</td><td>El usuario desde la aplicación selecciona una tabla de la lista de tablas pertenecientes a una base de datos y la elimina del servidor.</td></tr>
<tr><td>Secuencia normal</td><td>
<table>
<tr><th>Pasos</th><th>Acción</th></tr>
<tr><td>1</td><td>El usuario selecciona una tabla de la lista y pulsa la opción 'Borrar' del menú.</td></tr>
<tr><td>2</td><td>La sentencia MySQL o SQLite que permite el borrado de la tabla es enviada al servidor y ejecutada en el mismo.</td></tr>
</table>
</td></tr>
<tr><td>Excepciones</td><td>Excepción 1. Java SQLException o SQLiteException.
Excepción 2. Java Exception.</td></tr>
</table>

<table>
<tr><th>Nombre</th><th>Insertar Registro</th></tr>
<tr><td>Actividad</td><td>InsertarRegistros.java e InsertarRegistrosMovil.java.</td></tr>
<tr><td>Actores</td><td>Usuario identificado y usuario móvil.</td></tr>
<tr><td>Precondición</td><td>El usuario debe estar identificado frente al servidor.</td></tr>
<tr><td>Descripción</td><td>El usuario desde la aplicación podrá insertar nuevo registros a una tabla ya existente.</td></tr>
<tr><td>Secuencia normal</td><td>
<table>
<tr><th>Pasos</th><th>Acción</th></tr>
<tr><td>1</td><td>El usuario selecciona una de la lista de tablas pertenecientes a una base de datos en ListarTablas.java o ListarTablasMovil.java y luego pulsa sobre la opción 'Insertar'.</td></tr>
<tr><td>2</td><td>La aplicación se dirige a la actividad InsertarRegistros.java o InsertarRegistrosMovil.java donde aparecerá un menú con todos los campos de la tabla.</td></tr>
<tr><td>3</td><td>El usuario introduce los datos que quiere insertar en cada campo y pulsa sobre el botón 'Registrar'.</td></tr>
<tr><td>4</td><td>La sentencia MySQL o SQLite que permite la inserción de los registros a una tabla es enviada al servidor y ejecutada en el mismo.</td></tr>
</table>
</td></tr>
<tr><td>Excepciones</td><td>Excepción 1. Java SQLException o SQLiteException.
Excepción 2. Java Exception.</td></tr>
</table>

<table>
<tr><th>Nombre</th><th>Renombrar Tabla</th></tr>
<tr><td>Actividad</td><td>ListarTablas.java y ListarTablasMovil.java.</td></tr>
<tr><td>Actores</td><td>Usuario identificado y usuario móvil.</td></tr>
<tr><td>Precondición</td><td>El usuario debe estar identificado frente al servidor.</td></tr>
<tr><td>Descripción</td><td>El usuario desde la aplicación selecciona una tabla de la lista de tablas pertenecientes a una base de datos y cambia el nombre de dicha tabla por otro que posteriormente introduzca.</td></tr>
<tr><td>Secuencia normal</td><td>
<table>
<tr><th>Pasos</th><th>Acción</th></tr>
<tr><td>1</td><td>El usuario selecciona una tabla de la lista y pulsa la opción 'Renombrar' del menú.</td></tr>
<tr><td>2</td><td>El usuario introduce el nuevo nombre que desea para la tabla.</td></tr>
<tr><td>3</td><td>La sentencia MySQL o SQLite que permite el renombrado de tabla es enviada y ejecutada en el servidor.</td></tr>
</table>
</td></tr>
<tr><td>Excepciones</td><td>Excepción 1. Java SQLException o SQLiteException.
Excepción 2. Java Exception.</td></tr>
</table>

<table>
<tr><th>Nombre</th><th>Agregar Campo</th></tr>
<tr><td>Actividad</td><td>AgregarCampos.java y AgregarCamposMovil.java.</td></tr>
<tr><td>Actores</td><td>Usuario identificado y usuario móvil.</td></tr>
<tr><td>Precondición</td><td>El usuario debe estar identificado frente al servidor.</td></tr>
<tr><td>Descripción</td><td>El usuario desde la aplicación podrá añadir campos extras a una tabla anteriormente creada y definida.</td></tr>
<tr><td>Secuencia normal</td><td>
<table>
<tr><th>Pasos</th><th>Acción</th></tr>
<tr><td>1</td><td>El usuario selecciona una de la lista de tablas pertenecientes a una base de datos en ListarTablas.java o ListarTablasMovil.java y luego pulsa sobre la opción 'Agregar Campos'.</td></tr>
<tr><td>2</td><td>La aplicación se dirige a la actividad AgregarCampos.java o AgregarCamposMovil.java donde aparecerá un menú adecuado para la agregación de campos.</td></tr>
<tr><td>3</td><td>El usuario indica la cantidad de campos a introducir en la tabla, los parámetros que caracterizan a estos y finalmente donde van a ir situados respecto a los campos que ya existen en la tabla (con SQLite siempre irán al final). Tras esto el usuario pulsará el botón 'Agregar'.</td></tr>
<tr><td>4</td><td>La sentencia MySQL o SQLite que permite la agregación de campos a una tabla es enviada al servidor y ejecutada en el mismo.</td></tr>
</table>
</td></tr>
<tr><td>Excepciones</td><td>Excepción 1. Java SQLException o SQLiteException.
Excepción 2. Java Exception.</td></tr>
</table>

<table>
<tr><th>Nombre</th><th>Vaciar Tabla</th></tr>
<tr><td>Actividad</td><td>ListarTablas.java y ListarTablasMovil.java.</td></tr>
<tr><td>Actores</td><td>Usuario identificado y usuario móvil.</td></tr>
<tr><td>Precondición</td><td>El usuario debe estar identificado frente al servidor.</td></tr>
<tr><td>Descripción</td><td>El usuario desde la aplicación selecciona una tabla de la lista de tablas pertenecientes a una base de datos y elimina todos los registros introducidos en la misma anteriormente.</td></tr>
<tr><td>Secuencia normal</td><td>
<table>
<tr><th>Pasos</th><th>Acción</th></tr>
<tr><td>1</td><td>El usuario selecciona una tabla de la lista y pulsa la opción 'Vaciar' del menú.</td></tr>
<tr><td>2</td><td>La sentencia MySQL o SQLite que permite la eliminación de todos los registros de la tabla es enviada y ejecutada en el servidor.</td></tr>
</table>
</td></tr>
<tr><td>Excepciones</td><td>Excepción 1. Java SQLException o SQLiteException.
Excepción 2. Java Exception.</td></tr>
</table>

<table>
<tr><th>Nombre</th><th>Editar Valores</th></tr>
<tr><td>Actividad</td><td>Editar.java y Editar Movil.java.</td></tr>
<tr><td>Actores</td><td>Usuario identificado y usuario móvil.</td></tr>
<tr><td>Precondición</td><td>El usuario debe estar identificado frente al servidor.</td></tr>
<tr><td>Descripción</td><td>El usuario desde la aplicación podrá modificar el valor de un valor determinado perteneciente a un registro.</td></tr>
<tr><td>Secuencia normal</td><td>
<table>
<tr><th>Pasos</th><th>Acción</th></tr>
<tr><td>1</td><td>El usuario selecciona una de la lista de tablas pertenecientes a una base de datos en ListarTablas.java o ListarTablasMovil.java y luego pulsa sobre la opción 'Editar Valores'.</td></tr>
<tr><td>2</td><td>La aplicación se dirige a la actividad Editar.java o EditarMovil.java donde aparecerá un menú adecuado para la modificación de valores.</td></tr>
<tr><td>3</td><td>El usuario selecciona en primer lugar el campo donde se encuentra el registro a modificar y más tarde la clave primaria asociada al valor. Tras esto el usuario introduce el nuevo valor y pulsa el botón 'Modificar', o bien pulsa 'Borrar' y le asigna un valor nulo.</td></tr>
<tr><td>4</td><td>La sentencia MySQL o SQLite que permite la modificación del valor es enviada al servidor y ejecutada en el mismo.</td></tr>
</table>
</td></tr>
<tr><td>Excepciones</td><td>Excepción 1. Java SQLException o SQLiteException.
Excepción 2. Java Exception.</td></tr>
</table>

<table>
<tr><th>Nombre</th><th>Editar Registros</th></tr>
<tr><td>Actividad</td><td>Editar.java y Editar Movil.java.</td></tr>
<tr><td>Actores</td><td>Usuario identificado y usuario móvil.</td></tr>
<tr><td>Precondición</td><td>El usuario debe estar identificado frente al servidor.</td></tr>
<tr><td>Descripción</td><td>El usuario desde la aplicación podrá modificar los valores que formen parte de un registro.</td></tr>
<tr><td>Secuencia normal</td><td>
<table>
<tr><th>Pasos</th><th>Acción</th></tr>
<tr><td>1</td><td>El usuario selecciona una de la lista de tablas pertenecientes a una base de datos en ListarTablas.java o ListarTablasMovil.java y luego pulsa sobre la opción 'Editar Registros'.</td></tr>
<tr><td>2</td><td>La aplicación se dirige a la actividad Editar.java o EditarMovil.java donde aparecerá un menú adecuado para la modificación de registros.</td></tr>
<tr><td>3</td><td>El usuario selecciona la clave primaria asociada al registro que quiere modificar, tras esto el usuario introduce los nuevos valores que debe tomar y pulsa sobre el botón 'Modificar' o bien pulsa 'Borrar' y elimina todo el registro.</td></tr>
<tr><td>4</td><td>La sentencia MySQL o SQLite que permite la modificación o la que permite la eliminación del registro enviada al servidor y ejecutada en el mismo.</td></tr>
</table>
</td></tr>
<tr><td>Excepciones</td><td>Excepción 1. Java SQLException o SQLiteException.
Excepción 2. Java Exception.</td></tr>
</table>

3.5.2.2 Diagrama de clases UML

A continuación se expondrá un diagrama de clases en UML (Unified Modeling Language) el cual permite observar de forma general la estructura de la aplicación cliente. Debido a la considerable cantidad de clases y relaciones que pueden encontrarse se ha decidido dividir dicho diagrama en tres partes. En la primera de las partes aparecen las relacionadas con las funcionalidades dedicadas a un servidor MySQL externo, en la segunda aparecen las clases relacionadas con la funcionalidad SQLite en el dispositivo y en la tercera de las partes las que relacionan ambos ámbitos. Dichos diagramas pueden visualizarse a continuación en las **Figuras 3.13, 3.14 y 3.15** respectivamente.

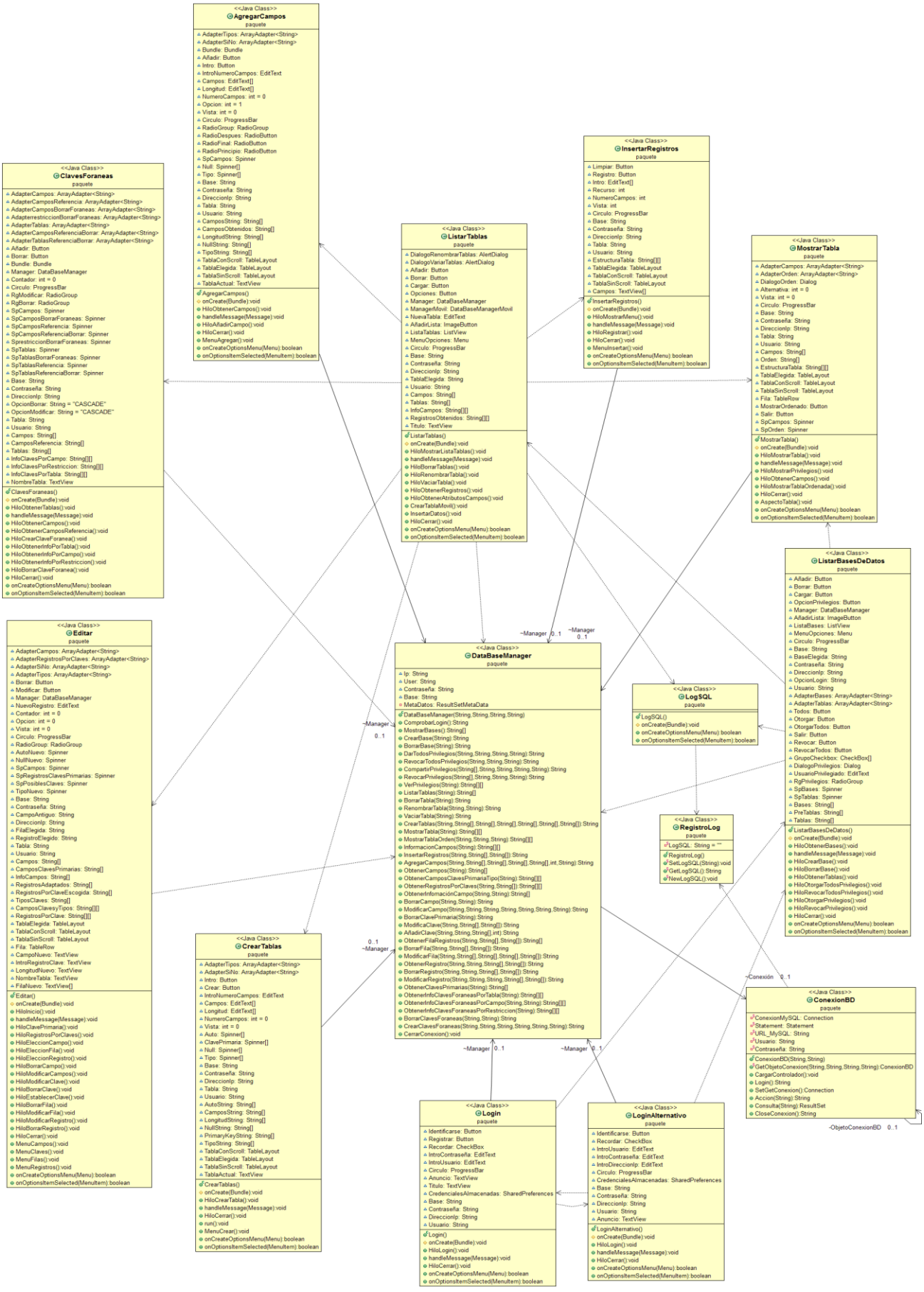


Figura 3.13 Diagrama de clases UML parte 1

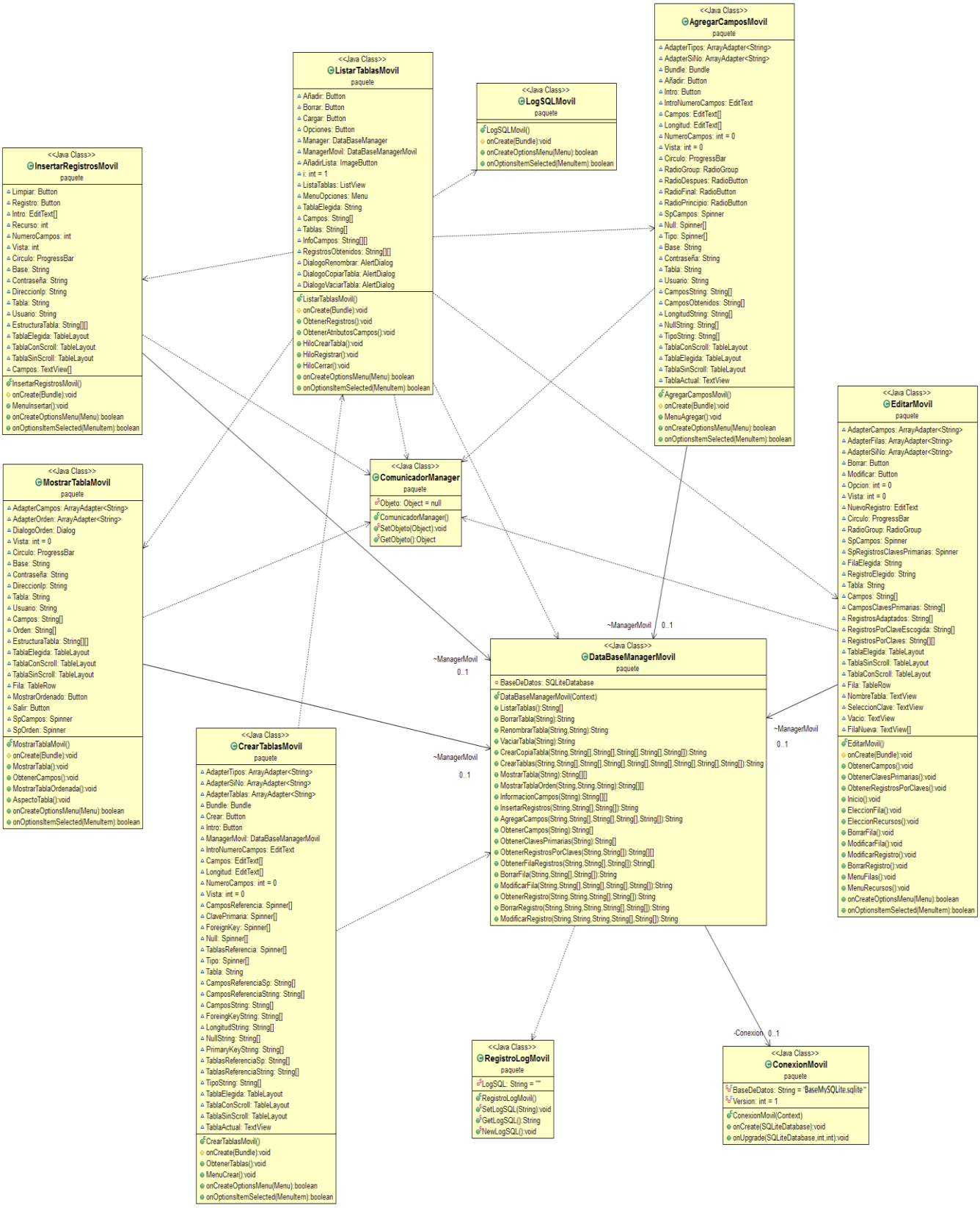


Figura 3.14 Diagrama de clases UML parte 2

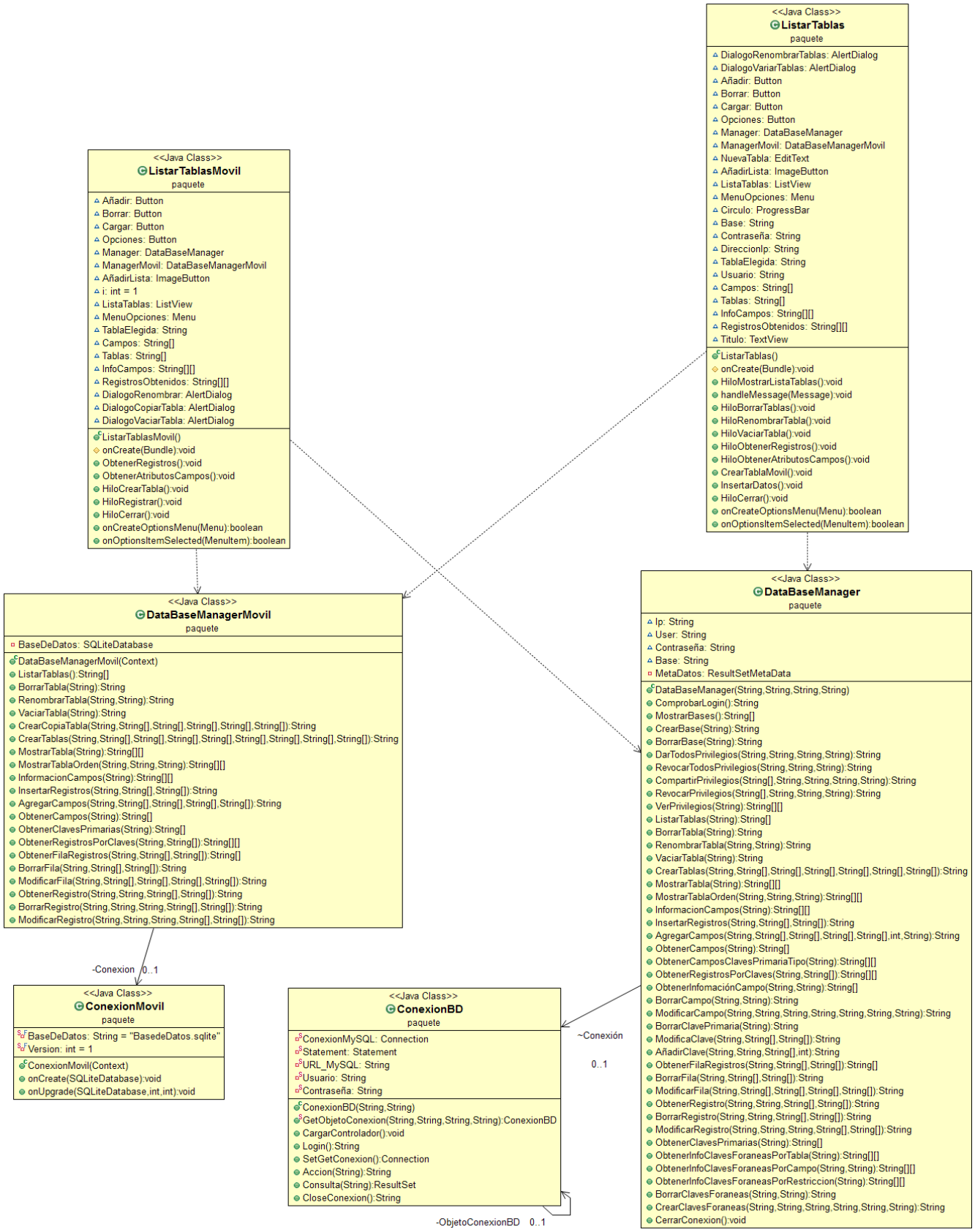


Figura 3.15 Diagrama de clases UML parte 3

3.5.2.3 Implementación de software

Para la elaboración del software que se ha desarrollado en el presente proyecto se ha dividido el trabajo en diferentes fases de desarrollo las cuales se comentarán a continuación. Dichas fases contemplan desde el desarrollo del servidor hasta el de la aplicación cliente y, dentro de este, la implementación del servicio de conectividad con el servidor, las funciones MySQL y SQLite que utilizará la aplicación y las actividades con sus respectivas interfaces.

3.5.2.3.1 Desarrollo del servidor www.servidormysql.no-ip.org

Tras la instalación del servidor MySQL, cuyos detalles pueden encontrarse en el Anexo 1: Instalación del software, se desarrollarán diferentes funcionalidades relativas a la creación de bases de datos y registros de usuarios en el servidor que MySQL por defecto no ofrece, así mismo se implementarán tecnologías de seguridad basadas en SSL para que los datos compartidos por cliente y servidor queden cifrados. Dichos formularios podrán ser accedidos gracias al uso de un servidor Tomcat con puerto 8443.

- Formulario registro de usuarios

La aplicación ofrecerá para este servidor un servicio de registro al usuario, para tal fin dicho servidor dispondrá de un formulario con nombre *FormularioRegistro.jsp* para que el usuario introduzca sus credenciales de usuario y contraseña y un nombre para una base de datos a la que únicamente tendrán acceso el administrador del sitio y el propio usuario. En la **Figura 3.16** puede apreciarse la interfaz gráfica del formulario donde se desarrolla la actividad y a continuación el código HTML del formulario.

Figura 3.16 Interfaz de *FormularioRegistro.jsp*

```
<form action="RegistroMYSQL.jsp" method="post" class="FormularioRegistro" >
        <div>
                <ul>
                        <li>
                                <h2 >Servicio de Registro </h2>
                        </li>
                        <li>
                                <label for="Usuario">Usuario:</label>
                                <input type="text" name="Usuario"/>
                        </li>
                        <li>
                                <label for="Contraseña">Contraseña:</label>
                                <input type="password" name="Contraseña" />
                        </li>
                        <li>
                                <label for="Base">Base De Datos:</label>
                                <input type="text" name="Base" />
                        </li>
                        <li>
                        <button class="Boton" type="submit">Registrarme</button>
                        </li>
                </ul>
        </div>
</form>
```

Una vez que los datos son introducidos y el botón 'Registrarme' es pulsado estos son enviados a un documento con nombre *FormularioMYSQL.jsp* donde se ejecutarán las sentencias MySQL para la creación del usuario, de la bases de datos y de la asignación de privilegios. Para dicha tarea se deberá comprobar si la contraseña es lo suficientemente larga (mayor de 5 caracteres), si ya existe un usuario con el nombre introducido, si existe una base de datos con el nombre introducido o bien si ocurre algún tipo de error. Finalmente la conexión con el servidor es cerrada. A continuación se muestra el código java y HTML que permite ejecutar las sentencias MySQL necesarias.

```
<%
// Construimos las variables que necesitemos de acuerdo a los valores recogidos del formulario
String Usuario = request.getParameter("Usuario");
String Contraseña = request.getParameter("Contraseña");
String Base = request.getParameter("Base");

// Construimos variables que almacenan los comandos de MySQL que precisamos
String CadenaCantidadUsuarios = "SELECT count(User) FROM mysql.user WHERE User = '" + Usuario +
"'";

String CadenaCantidadBases = "SELECT count(schema_name) FROM information_schema.SCHEMATA where
schema_name = '" + Base + "'";

String CadenaCrearUsuario = "CREATE USER '" + Usuario + "'@'%' IDENTIFIED BY '" + Contraseña +
"'";

String CadenaCrearBase = "CREATE DATABASE " + Base;

String CadenaPrivilegios = "GRANT ALL PRIVILEGES ON " + Base + ".* TO '" + Usuario + "'@'%'";

Connection Conexion = null;
Statement Sentencia;
ResultSet Resultado;
String SQLError;
```

```
// Iniciamos aquí el proceso de creación del usuario y de la base de datos
if(Contraseña.length() > 5){
try{
        Class.forName ("com.mysql.jdbc.Driver");
        Conexion =
DriverManager.getConnection("jdbc:mysql://localhost:3306/information_schema","root","");
        Sentencia = Conexion.createStatement();

        Resultado = Sentencia.executeQuery(CadenaCantidadUsuarios);
        Resultado.next();
        int CantidadUsuarios = Resultado.getInt(1);

        Resultado = Sentencia.executeQuery(CadenaCantidadBases);
        Resultado.next();
        int CantidadBases = Resultado.getInt(1);

        if ( CantidadUsuarios > 0){
                Sentencia.close();
                Conexion.close();
                %>
        <script>alert("Lo sentimos, ya existe un usuario con ese nombre")</script>
        <script>window.history.go(-1)</script>
                <%
        } else if( CantidadBases > 0){
                Sentencia.close();
                Conexion.close();
                %>
        <script>alert("Lo sentimos, ya existe una base de datos con ese nombre")</script>
        <script>window.history.go(-1)</script>
                <%
        } else {
                Sentencia.execute(CadenaCrearUsuario);
                Sentencia.execute(CadenaCrearBase);
                Sentencia.execute(CadenaPrivilegios);
        }

        %>
        <script>alert("Enhorabuena su cuenta ha sido creada")</script>
    <%
}catch (SQLException e){
        SQLError = e.getMessage();
         %>
        <script> var Error = "<%= SQLError %>"; alert(Error); </script>
        <%
} catch (Exception e){
        SQLError = e.getMessage();
         %>
        <script> var Error = "<%= SQLError %>"; alert(Error); </script>
         <%

}  finally{
        try{
        if(Conexion!=null){
        Conexion.close();
        }
     }catch(SQLException e){
         SQLError = e.getMessage();
         %>
        <script> var Error = "<%= SQLError %>"; alert(Error); </script>
         <%
    }
  }
} else {
        %>
<script>alert("La contraseña es demasiado corta, utilice una de al menos 6 caracteres")</script>
        <%
        }
        %>
        <script>window.history.go(-1)</script>
```

- Formulario creación de bases de datos

La aplicación ofrecerá para este servidor un servicio de creación de bases de datos al usuario, para tal fin dicho servidor dispondrá de un formulario con nombre *ForumlarioBasesDeDatos.jsp* para que el usuario introduzca sus credenciales de usuario y contraseña y un nombre para una base de datos a la que únicamente tendrán acceso el administrador del sitio y el propio usuario. En la **Figura 3.17** puede apreciarse la interfaz gráfica del formulario donde se desarrolla la actividad y a continuación el código HTML del formulario.

Figura 3.17 Interfaz de *BasesDeDatosMYSQL.jsp*

Una vez que los datos son introducidos y el botón 'Crear Base' es pulsado estos son enviados a un documento con nombre *BasesDeDatosMYSQL.jsp* donde se ejecutarán las sentencias MySQL para la creación de la base de datos y la asignación de privilegios. Para dicha tarea se deberá comprobar si las credenciales del usuario son correctas, si existe una base de datos con el nombre introducido o bien si ocurre algún tipo de error. Finalmente la conexión con el servidor es cerrada. A continuación se muestra el código java y HTML que permite ejecutar las sentencias MySQL necesarias.

```
<%
// Construimos las variables que necesitemos de acuerdo a los valores recogidos del formulario
String Usuario = request.getParameter("Usuario");
String Contraseña = request.getParameter("Contraseña");
String Base = request.getParameter("Base");

String CadenaCantidadBases  = "SELECT count(schema_name) FROM information_schema.SCHEMATA where
schema_name = '" + Base + "'";
String CadenaCrearBase = "CREATE DATABASE " + Base;
String CadenaPrivilegios = "GRANT ALL PRIVILEGES ON " + Base + ".* TO '" + Usuario + "'@'%'";

Connection Conexion = null;
Statement Sentencia;
ResultSet Resultado;
String SQLError;
```

```
// Iniciamos aquí el proceso para la comprobación de credenciales de usuario
try{
        Class.forName ("com.mysql.jdbc.Driver");
        Conexion =
DriverManager.getConnection("jdbc:mysql://localhost:3306/information_schema",Usuario,Contraseña;
} catch(SQLException e){
        SQLError = e.getMessage();
        %>
        <script> var Error = "<%= SQLError %>"; alert(Error); </script>
        <script>window.history.go(-1)</script>
        <%
} catch (Exception e){
        SQLError = e.getMessage();
        %>
        <script> var Error = "<%= SQLError %>"; alert(Error); </script>
        <script>window.history.go(-1)</script>
        <%
}
finally{
        try{
        if(Conexion!=null){
        Conexion.close();
        }
     }catch(SQLException e){
         SQLError = e.getMessage();
         %>
        <script> var Error = "<%= SQLError %>"; alert(Error); </script>
         <%
     }
  }
// Iniciamos aquí el proceso para la creación de la base de datos
try{
        Conexion =
DriverManager.getConnection("jdbc:mysql://localhost:3306/information_schema","root","");
        Sentencia = Conexion.createStatement();
        Resultado = Sentencia.executeQuery(CadenaCantidadBases);
        Resultado.next();
        int CantidadBases = Resultado.getInt(1);
    if(CantidadBases > 0){
                %>
        <script>alert("Lo sentimos, ya existe una base de datos con ese nombre")</script>
        <script>window.history.go(-1)</script>
                <%
        } else {
                Sentencia.execute(CadenaCrearBase);
                Sentencia.execute(CadenaPrivilegios);
        }
        %>
        <script>alert("Enhorabuena su nueva base de datos ha sido creada")</script>
    <%
} catch(SQLException e){
         SQLError = e.getMessage();
         %>
        <script> var Error = "<%= SQLError %>"; alert(Error); </script>
         <%
} catch (Exception e){
        SQLError = e.getMessage();
        %>
        <script> var Error = "<%= SQLError %>"; alert(Error); </script>
        <%
} finally{
        try{
        if(Conexion!=null){
        Conexion.close();
        }
     }catch(SQLException e){
         SQLError = e.getMessage();
         %>
        <script> var Error = "<%= SQLError %>"; alert(Error); </script>
         <%
     }
  }
%>
<script>window.history.go(-1)</script>
```

- Conexiones cifradas

Para conseguir que los datos entre cliente y servidor sean cifrados utilizando tecnología SSL es necesario en primer lugar hacerse con un certificado, en este caso el certificado será realizado y firmado por el propio desarrollador por lo que no se garantizará un servicio de autenticación del servidor. El cifrado de los datos debe contemplarse tanto en el caso de los formularios, donde el usuario puede registrarse y crear bases de datos utilizando para ello conexiones con el servidor Tomcat, como en el de las conexiones realizadas con el servidor MySQL.

Para las conexiones con el servidor Tomcat se generarán claves con el logaritmo RSA, dicho proceso se ejecutará gracias al JDK de java que ofrece dicha posibilidad y está descrito en la **Figura 3.18.** Simplemente bastará con introducir el comando para la generación de claves e introducir los datos que vaya pidiendo la ejecución.

```
C:\PROGRA~1\Java\JDK18~1.0_3\bin>keytool -genkey -alias tomcat -keyalg RSA
Introduzca la contraseña del almacén de claves:
Volver a escribir la contraseña nueva:
¿Cuáles son su nombre y su apellido?
  [Unknown]:  Daniel Soria Martinez
¿Cuál es el nombre de su unidad de organización?
  [Unknown]:  support
¿Cuál es el nombre de su organización?
  [Unknown]:  support
¿Cuál es el nombre de su ciudad o localidad?
  [Unknown]:  Bailen
¿Cuál es el nombre de su estado o provincia?
  [Unknown]:  Jaen
¿Cuál es el código de país de dos letras de la unidad?
  [Unknown]:  es
¿Es correcto CN=Daniel Soria Martinez, OU=support, O=support, L=Bailen, ST=Jaen,
 C=es?
  [no]:  si

Introduzca la contraseña de clave para <tomcat>
        (INTRO si es la misma contraseña que la del almacén de claves):
Volver a escribir la contraseña nueva:
```

Figura 3.18 Generación de claves RSA

Una vez generada la clave deberá configurarse Tomcat para que pueda usar dicha clave y asignar un puerto, en este caso el 8443, que utilizará el protocolo HTTPS, para ello habrá que dirigirse al archivo server.xml del servidor Tomcat e insertar el código que aparece a continuación el cual activa el conector en el puerto 8443 con el protocolo SSL activado y haciendo uso del protocolo HTTPS.

```
<Connector SSLEnabled="true" acceptCount="100" clientAuth="false"
    disableUploadTimeout="true" enableLookups="false" maxThreads="25"
    port="8443" keystoreFile="/.keystore" keystorePass="123456"
    protocol="org.apache.coyote.http11.Http11NioProtocol"scheme="https"
    secure="true" sslProtocol="TLS" />
```

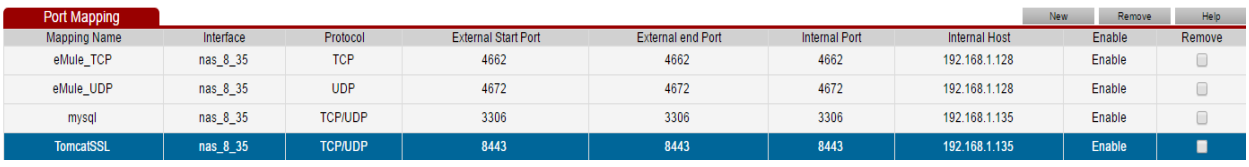
Tras ello deberán borrarse los demás conectores para que solo pueda accederse al servidor a través del puerto que proporciona el servicio de cifrado.

Para las conexiones con el servidor MySQL se utilizará el software OpenSSL para la generación del certificado, dicho proceso es mostrado en la **Figura 3.20** a continuación. Aunque en este caso las instrucciones están en inglés los datos a introducir son muy similares a los del caso anterior tales como la dirección o el e-mail.

Para aplicar el certificado generado a la configuración del servidor MySQL bastará con indicarlo en el archivo my.cnf del servidor, para ello deberá buscarse la etiqueta '[mysqld]' en el mismo e introducir las siguientes líneas que hacen referencia al lugar de la entidad de certificación, al certificado en sí y a la clave del certificado.

ssl-ca= "C:/newcerts/ca.pem"
ssl-cert= "C:/newcerts/server-cert.pem"
ssl-key= "C:/newcerts/server-key.pem"

Una vez configurados todos los aspectos del servidor deberán abrirse los puertos que se utilicen en el router si se desea que el servidor pueda operar de forma remota y direccionarlos al terminal donde reside la aplicación servidora. A continuación se muestra la configuración necesaria en la **Figura 3.19**



Mapping Name	Interface	Protocol	External Start Port	External end Port	Internal Port	Internal Host	Enable	Remove
eMule_TCP	nas_8_35	TCP	4662	4662	4662	192.168.1.128	Enable	
eMule_UDP	nas_8_35	UDP	4672	4672	4672	192.168.1.128	Enable	
mysql	nas_8_35	TCP/UDP	3306	3306	3306	192.168.1.135	Enable	
TomcatSSL	nas_8_35	TCP/UDP	8443	8443	8443	192.168.1.135	Enable	

Figura 3.19 Configuración de puertos en router

Es recomendable también utilizar un servicio de DNS para las conexiones remotas con el servidor, como en este caso para la dirección www.servidormysql.no-ip.org que el servidor ofrece. [47]

Figura 3.20 Generación de certificado SSL

3.5.2.3.2 Desarrollo de la aplicación cliente

El objetivo de la aplicación cliente será la de poder gestionar de manera remota las bases de datos ubicadas en un servidor MySQL externo al dispositivo de una forma segura. Para dicha tarea se ha dividido el proceso de desarrollo del software en tres subprocesos diferenciados según funcionalidad. En primer lugar se ha desarrollado una clase java la cual conlleva las funcionalidades y parámetros necesarios para realiza la conexión con el servidor MySQL, tras esto se construirá otra clase la cual almacenará todas las funciones con las sentencias MySQL que utilizará la aplicación y finalmente se crearán las actividades, o clases java con sus respectivas interfaces que harán uso de las funciones de la clase anteriormente descrita. Para el desarrollo en cuanto a la gestión de tablas almacenadas en el dispositivo móvil se trabajará de manera similar, primero se creará una clase encargada de la conexión con la base de datos creada en SQLite, después otra que albergará las diferentes sentencias SQLite ubicadas en funciones y finalmente las actividades con sus respectivas interfaces que harán uso de estas clases. En la **Tabla 3.3** puede encontrarse un resumen de todas las clases java incluidas para el desarrollo de la aplicación.

Tabla Implementación Cliente			
	Clases para conexión	Clases almacén	Actividades
Conexión MySQL	ConexiónBD.java	DataBaseManager.java	• Login.java • LoginAlternativo.java • ListarBasesDeDatos.java • ListarTablas.java • CrearTablas.java • MostrarTabla.java • InsertarRegistros.java • AgregarCampos.java • Editar.java • ClavesForaneas.java • LogSQL.java
Conexión SQLite	ConexiónMovil.java	DataBaseManagerMovil.java	• ListarTablasMovil.java • CrearTablasMovil.java • MostrarTablaMovil.java • InsertarRegistrosMovil.java • AgregarCamposMovil.java • EditarMovil.java • LogSQLMovil.java

Tabla 3.3 Resumen clases java

- Clases para la conectividad

Para las funcionalidades y parámetros necesarios para realiza la conexión con el servidor MySQL se ha implementado la clase llamada *ConexionMySQL.java* cuyo código aparece a continuación y que será desglosado y explicado posteriormente.

```
public class ConexionMySQL {

   private static Connection Conexion;
   private static Statement Statement;
   private static String URL_MySQL;
   private static String Usuario;
   private static String Contraseña;
   private static ConexionMySQL ObjetoConexionMySQL;


   public ConexionMySQL(String DireccionIp, String Base) {
      URL_MySQL = "jdbc:mysql://" + DireccionIp + ":3306/" + Base +
"?useSSL=true&verifyServerCertificate=false&requireSSL=false";
      CargarControlador();
   }


   public static ConexionMySQL GetObjetoConexion (String DireccionIp, String Usuario,
String Contraseña, String Base) {
      ConexionMySQL.Usuario = Usuario;
      ConexionMySQL.Contraseña = Contraseña;
      ObjetoConexionMySQL = new ConexionMySQL(DireccionIp, Base);
      return ObjetoConexionMySQL;
   }


   public void CargarControlador() {
      try {
         String controlador_MySQL = "com.mysql.jdbc.Driver";
         Class.forName(controlador_MySQL);
      } catch (ClassNotFoundException e) {
         e.printStackTrace();
      }
   }


   public Connection SetGetConexion() throws SQLException {
      Conexion = DriverManager.getConnection(URL_MySQL, Usuario, Contraseña);
      return Conexion;
   }


   public String Login() {
      try {
         Conexion = DriverManager.getConnection(URL_MySQL, Usuario, Contraseña);
         return "1";
      } catch (SQLException e) {
         return e.getMessage();
      } catch (Exception e) {
         return "0";
      }
   }
```

```
    public String Accion(String SentenciaSQL) {
      try {
         RegistroLog.SetLogSQL(SentenciaSQL + "\n\n");
         Statement = SetGetConexion().createStatement();
         Statement.executeUpdate(SentenciaSQL);
         return "1";
      } catch (SQLException e) {
         return e.getMessage();
      } catch (Exception e) {
         return "0";
      }
   }


   public ResultSet Consulta(String SentenciaSQL) throws SQLException {
      RegistroLog.SetLogSQL(SentenciaSQL + "\n\n");
      Statement = SetGetConexion().createStatement();
      ResultSet Resultado = Statement.executeQuery(SentenciaSQL);
      return Resultado;
   }

    public String CloseConexion() {
      try {
         if (Conexion != null) {
            Conexion.close();
         }
         return "1";
      } catch (SQLException e) {
         return e.getMessage();
      } catch (Exception e) {
         return "0";
      }
   }
}
```

El constructor *ConexionMySQL* se encarga de construir la URL necesaria teniendo en cuenta la dirección IP, el puerto y la base de datos a los que conectarse utilizando el conector JDBC, el cual se cargará al momento con la función *CargarControlador*. La URL conlleva además las opciones necesarias para el uso no requerido de SSL en las conexiones realizadas con el servidor. La función *SetGetConexion* crea y devuelve el objeto de conexión teniendo en cuenta los parámetros de usuario y contraseña y de la URL anteriormente construida y la función *GetObjetoConexion* devuelve un objeto construido con la actual clase. Finalmente aparecen las funciones de *Login* para comprobar las credenciales del usuario, realizando para ello una simple conexión con el servidor MySQL, *Acción* para la ejecución de sentencias MySQL en el servidor, *Consulta* para la ejecución de consultas MySQL en el servidor y *CloseConexion* para cerrar la conexión con el servidor una vez realizadas las operaciones pertinentes.

Para realizar la creación de la base de datos donde se almacenan las tablas en el dispositivo y para permitir la conexión con esta se ha implementado la clase llamada *ConexionMovil.java* cuyo código aparece y que será desglosado y explicado a continuación.

```
public class ConexionMovil extends SQLiteOpenHelper {

    private static final String BaseDeDatos = "BaseMySQLite.sqlite";
    private static final int Version = 1;

    public ConexionMovil(Context context) {
        super(context, BaseDeDatos, null, Version);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    }
}
```

Esta clase empieza siendo extendida de la clase *SQLiteOpenHelper*, la cual hace posible la gestión de bases de datos en el dispositivo. Esta será llamada introduciendo los parámetros del contexto en el cual se encuentre la actividad, el nombre de la base de datos a crear y con la que se trabajará, en este caso con nombre 'BaseMySQLite' , y la versión del protocolo. Finalmente las clases *onCreate* y *onUpgrade* son creadas automáticamente para la creación de la base de datos y la actualización de la versión (aunque esta última en realidad no es utilizada la clase exige introducirla).

- Clases almacenadoras de funciones MySQL y SQLite

Para almacenar y organizar todas las funciones que guardan sentencias MySQL y SQLite de las que hace uso la aplicación se hará uso de dos clases según el ámbito en el que se trabaje, es decir, se llamará *DataBaseManager.java* a la clase que almacenará las funciones con sentencias MySQL que se ejecutarán en el servidor externo y por otro lado, se llamará *DataBaseManagerMovil.java* a la clase que almacenará las funciones con sentencias SQLite que se ejecutarán en el mismo dispositivo. Ya que ambos lenguajes parten de SQL tienen una estructura muy parecida. A continuación se mostrará una lista de todas las estructuras de las expresiones MySQL y SQLite que la aplicación puede utilizar. [48]

1. SELECT select_expr FROM table_references WHERE where_definition

Se usa para recuperar registros de una o más tablas.

- Cada select_expr indica las columnas que se quieren recuperar.
- table_references indica la tabla o tablas de las que se recuperan los registros.
- where_definition consiste en la utilización de la palabra clave WHERE seguida por una expresión que indica la condición o condiciones que los registros deben satisfacer para ser seleccionadas. WHERE campo = 'valor'.

2. SHOW DATABASES / SHOW TABLES FROM name_db / SHOW GRANTS FOR CURRENT_USER()

Todas estas expresiones se utilizan para recuperar información del servidor MySQL. La primera recupera las bases de datos almacenadas en el mismo, la segunda recupera las tablas de una base de datos indicada (name_db) y la tercera permite recuperar los privilegios del usuario MySQL que utiliza dicha sentencia.

3. DROP db_name / DROP TABLE tbl_name

Ambas expresiones permiten la eliminación de bases de datos y de tablas respectivamente según las introducidas en db_name y en tbl_name.

4. CREATE DATABASE db_name / CREATE TABLE tbl_name (create_definition)

Ambas sentencias permiten la creación de bases de datos y de tablas respectivamente según las introducidas en db_name y en tbl_name.

create_definition contiene la siguiente estructura:

columns_definition CONSTRAINT symbol PRIMARY KEY CONSTRAINT symbol (col_name,...) FOREIGN KEY (col_name,...) REFERENCES tbl_name_ref (col_name_ref,...) ON DELETE reference_option ON UPDATE reference_option.
reference_option tiene tres opciones: RESTRICT | CASCADE | SET NULL

columns_definition contiene la siguiente estructura:

col_name type NOT NULL | NULL AUTO_INCREMENT, donde col_name es el nombre del campo en la cabecera, donde type contiene puede ser Char, Date, Float, Int, Text, Time o Varchar (longitud).

5. GRANT priv_type ON tbl_name.bd_name TO user WITH GRANT OPTION / REVOKE priv_type GRANT OPTION ON tbl_name.bd_name FROM user

Ambas sentencias permiten la asignación y la eliminación de privilegios respectivamente según la base de datos y tablas introducidas en db_name y en tbl_name. priv_type está referido a los privilegios los cuales se asignarán o se revocarán del usuario. La aplicación ofrece los siguientes privilegios y opciones: CREATE, DROP, SELECT, INSERT, UPDATE, DELETE, ALTER o bien la opción ALL PRIVILEGES que ofrece o revoca todos los privilegios posibles con los que cuente el usuario en la base de datos y tablas seleccionadas.

6. RENAME TABLE tbl_name TO new_tbl_name

   Permite el renombrado de tablas.

7. INSERT INTO tbl_name (col_name,...) VALUES ('value',...)

   Permite le inserción de registros en las columnas de una tabla.

8. UPDATE tbl_name SET col_name = 'value',... WHERE where_definition

   Permite la modificación de valores en los registros de una tabla.

9. DELETE FROM table_name WHERE where_definition / TRUNCATE TABLE table_name

Ambas sentencias permiten la eliminación de registros en una tabla. La primera permite eliminar registros escogidos mientras que la segunda permite eliminar todos los registros de una tabla.

10. ALTER TABLE tbl_name alter_specification

Permite modificar la estructura de una tabla existente. Por ejemplo, se pueden añadir o eliminar campos, cambiar el tipo de un campo existente o renombrar campos.

alter_specification contiene las siguientes opciones para su estructura:

- ADD COLUMN create_definition FIRST | AFTER col_name.
- ADD CONSTRAINT symbol PRIMARY KEY (col_name,...).
- ADD CONSTRAINT symbol FOREIGN KEY tbl_name_ref (col_name_ref,...).
- DROP COLUMN col_name.
- DROP PRIMARY KEY.
- DROP FOREIGN KEY.
- CHANGE COLUMN old_col_name create_definition.
- MODIFY COLUMN create_definition.

Además de las funciones anteriores la actividad *DataBaseManager.java* contará con una función *ComprobarLogin* la cual contiene una llamada a la función *Login* en la actividad *ConexionMySQL.java*.

La clase *DataBaseManager.java* creará un objeto de conexión de la clase ConexiónMySQL con los parámetros de conexión.

```
static ConexionMySQL Conexión;
String Ip;
String User;
String Contraseña;
String Base;

public DataBaseManager(String Ip, String User, String Contraseña, String Base) {
    this.Ip = Ip;
    this.User = User;
    this.Contraseña = Contraseña;
    this.Base = Base;
    Conexión = Conexión.GetObjetoConexion(Ip, User, Contraseña, Base);
}
```

La clase *DataBaseManagerMovil.java* creará un objeto de conexión de la clase ConexiónMovil.

```
private ConexionMovil Conexion;
private SQLiteDatabase BaseDeDatos;
public DataBaseManagerMovil(Context context) {
    Conexion = new ConexionMovil(context);
    BaseDeDatos = Conexion.getWritableDatabase();
}
```

- Actividades e interfaces

A continuación se estudiará el desarrollo de la implementación de todas las actividades que hacen uso de las sentencias MySQL y SQLite anteriormente descritas.

En primer lugar se deberán crear los objetos de las clases de conexión.

```
Manager = new DataBaseManager(DireccionIp, Usuario, Contraseña, Base);
ManagerMovil = new DataBaseManagerMovil(this);
```

Las funciones MySQL utilizadas de *DataBaseManager.java* deberán ser lanzadas en diferentes hilos separados del principal y las conexiones cortadas mediante la función *CerrarConexión* mientras que las funciones SQLite utilizadas de *DataBaseManagerMovil.java* podrán ser utilizadas directamente.

1. Login.java y LoginAlternativo.java

Ambas actividades persiguen hacer posible la identificación del usuario mediante sus credenciales frente a un servidor MySQL externo al dispositivo, el primero frente al servidor con dirección www.servidormysql.no-ip.org y el segundo frente a uno escogido por el usuario. Para ello en primer lugar se construirán los datos requeridos de conexión, es decir, credenciales de usuario y contraseña y la dirección IP del servidor y la base de datos al que conectarse y una variable de estado para diferenciar entre actividades Login.java o LoginAlternativo.java, tras esto se realizará una simple conexión con el servidor y se valorará la respuesta que este ofrezca y si esta es positiva avanzar la aplicación hacia la actividad ListarBasesDeDatos.java junto con los datos mencionados. Se contará también con un sistema para almacenar las últimas credenciales de usuario y contraseña en el caso de que la respuesta sea positiva. Para dicha tarea se hará uso de la función *ComprobarLogin* de la clase *DataBaseManager.java*.

```
public class HiloLogin extends Thread {
    public void run() {
        // Creamos el objeto DataBaseManager de acuerdo a los datos de conexión
        Manager = new DataBaseManager(DireccionIp, Usuario, Contraseña, Base);
        String Respuesta = Manager.ComprobarLogin();
        Message msg = Message.obtain();
        msg.obj = Respuesta;
        msg.setTarget(puente);
        msg.sendToTarget();
        HiloCerrar Hilo = new HiloCerrar();
```

```
        Hilo.start();
    }

    Handler puente = new Handler() {
        public void handleMessage(Message col) {
            Intent Intent = new Intent(Login.this, ListarBasesDeDatos.class);
            String Respuesta = (String) col.obj;
            SharedPreferences.Editor EditorCredencialesAlmacenadas =
CredencialesAlmacenadas.edit();
            if (!Recordar.isChecked()) {
                EditorCredencialesAlmacenadas.putString("Pass", "");
                EditorCredencialesAlmacenadas.putString("User", "");
                EditorCredencialesAlmacenadas.apply();
            }

            if (Respuesta.equals("1")) {
                if (Recordar.isChecked()) {
                    EditorCredencialesAlmacenadas.putString("Pass", Contraseña);
                    EditorCredencialesAlmacenadas.putString("User", Usuario);
                    EditorCredencialesAlmacenadas.apply();
                }

                Anuncio.setText("");

                // Si la identificación es correcta cargamos los datos de conexión en
un Bundle para llevarlos a la siguiente actividad
                Bundle Bundle = new Bundle();
                Bundle.putString("Ip", DireccionIp);
                Bundle.putString("User", Usuario);
                Bundle.putString("Pass", Contraseña);
                Bundle.putString("Base", Base);
                Bundle.putString("Login", "Login");
                Intent.putExtras(Bundle);
                startActivity(Intent);

            } else if (Respuesta.contains("denied")) {
                Anuncio.setTextColor(Color.RED);
                Anuncio.setText("Error usuario o contraseña incorrectos");
            } else if (Respuesta.equals("0")) {
                Anuncio.setTextColor(Color.RED);
                Anuncio.setText("Error inesperado contacte con el administrador");
            } else {
                Anuncio.setText("");
                Toast.makeText(Login.this, Respuesta, Toast.LENGTH_SHORT).show();
            }
            Circulo.setVisibility(View.GONE);
        }
    };
}
```

Login.java además proporciona un botón que permite abrir el navegador en el dispositivo con el formulario *RegistroUsuario.jsp* anteriormente comentado.

Dichas actividades junto con *ListarTablasMovil.java* contendrán un menú para desplazarse entre ellas similar al que se muestra a continuación.

```
public boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId();
        Intent Intent;
        switch (id) {
```

```
            case R.id.MenuWebLogin:
                break;

            // Nos trasladamos a la actividad LoginAlternativo
            case R.id.MenuAlternativoLogin:
                Intent = new Intent(Login.this, LoginAlternativo.class);
                startActivity(Intent);
                break;
            // Nos trasladamos a la actividad ListarTablasMovil
            case R.id.MenuMovilLogin:
                Intent = new Intent(Login.this, ListarTablasMovil.class);
                startActivity(Intent);
                break;
        }
        return super.onOptionsItemSelected(item);
    }
}
```

2. ListarBasesDeDatos.java

Esta actividad tiene como función principal la de permitir al usuario gestionar las bases de datos ubicadas en un servidor MySQL externo al dispositivo. Para ello se necesitarán funciones para visualizar, crear y eliminar dichas bases de datos, las cuales se discutirán a continuación. Para esto en primer lugar se deben recoger los datos de conexión venideros.

- Visualización de bases de datos

En primer lugar, a través de la función *MostrarBases* de *DataBaseManager.java* se enviará una sentencia MySQL al servidor para recoger las bases de datos a las que el usuario tiene acceso y estas serán cargadas en una lista.

```
public class HiloObtenerBases extends Thread {
    public void run() {
        String Respuesta;
        Bases = Manager.MostrarBases();
        Respuesta = "1";
        Message msg = Message.obtain();
        msg.obj = Respuesta;
        msg.setTarget(Puente);
        msg.sendToTarget();
        HiloCerrar Hilo = new HiloCerrar();
        Hilo.start();
    }

    Handler Puente = new Handler() {
        public void handleMessage(Message col) {
            String Respuesta = (String) col.obj;
            switch (Respuesta) {
                case "1":
                ArrayAdapter<String> adapter = new
ArrayAdapter<>(ListarBasesDeDatos.this, R.layout.estiloelistview, Bases);
                ListaBases.setAdapter(adapter);
                ListaBases.addFooterView(AñadirLista);
```

```
                Toast.makeText(ListarBasesDeDatos.this, "Bases de datos cargadas",
Toast.LENGTH_SHORT).show();
                    break;
            }
            Circulo.setVisibility(View.GONE);
        }
    };
}
```

Para visualizar las tablas que se encuentran dentro de cada base de datos bastará con seleccionarla y pulsar sobre el botón 'Cargar' lo que permitirá enviar los datos de conexión a la actividad *ListarTablas.java*, la cual será estudiada posteriormente.

- Creación de bases de datos

En este punto de distinguirán dos casos diferentes según la actividad anterior *Login.java* o *LoginAlternativo.java*. En el primero de los casos la creación de la base de datos se deja en manos del formulario *FormularioBasesDeDatos.jsp* mientras que en el segundo la creación de la misma se realiza desde el mismo cliente, es decir, el usuario deberá introducir el nombre de la base de datos que desea crear desde la aplicación, tras esto se procederá utilizar la función *CrearBase* de *DataBaseManager.java.*

```
public class HiloCrearBase extends Thread {
    public void run() {
        String Respuesta;
        Respuesta = Manager.CrearBase(BaseElegida);
        Message msg = Message.obtain();
        msg.obj = Respuesta;
        msg.setTarget(Puente);
        msg.sendToTarget();
        HiloCerrar Hilo = new HiloCerrar();
        Hilo.start();
    }

    Handler Puente = new Handler() {
        public void handleMessage(Message col) {
            String Respuesta = (String) col.obj;
            Circulo.setVisibility(View.GONE);
            switch (Respuesta) {
                case "1":
                    Toast.makeText(ListarBasesDeDatos.this, "Base de datos creada",
Toast.LENGTH_SHORT).show();
                    finish();
                    startActivity(getIntent());
                    break;
            }
        }
    };
}
```

- Eliminación de bases de datos

El usuario a través de la aplicación podrá seleccionar una base de datos de la lista y borrar dicha base de datos tras confirmar dicha acción gracias al uso de la función *BorrarBase* de *DataBaseManager.java.*

```
public class HiloBorrarBase extends Thread {
    public void run() {
        String Respuesta = Manager.BorrarBase(BaseElegida);
        Message msg = Message.obtain();
        msg.obj = Respuesta;
        msg.setTarget(Puente);
        msg.sendToTarget();
        HiloCerrar Hilo = new HiloCerrar();
        Hilo.start();
    }

    Handler Puente = new Handler() {
        public void handleMessage(Message col) {
            String Respuesta = (String) col.obj;
            switch (Respuesta) {
                case "1":
                    Toast.makeText(ListarBasesDeDatos.this, "Base de datos borrada",
Toast.LENGTH_SHORT).show();
                    finish();
                    startActivity(getIntent());
                    break;
            }
        }
    };
}
```

Por otro lado la actual actividad también contará con funciones para compartir privilegios MySQL entre los usuarios, para ello se mostrará al usuario un menú con diferentes opciones (el cual podrá ser visualizado de mejor manera en la parte donde se discute el diseño de la interfaz) donde el usuario podrá otorgar y revocar diferentes privilegios sobre las bases de datos y tablas a las que tenga acceso.

- Visualizar privilegios

El usuario podrá visualizar sus propios privilegios MySQL gracias a la función *VerPrivilegios* de *DataBaseManager.java* la cual se ejecutará en la actividad *MostrarTabla.java* accediendo a ella desde esta.

```
public class HiloMostrarPrivilegios extends Thread {
    public void run() {
        String Respuesta;
        EstructuraTabla = Manager.VerPrivilegios();
        Respuesta = "1";
        Message msg = Message.obtain();
```

```
        msg.obj = Respuesta;
        msg.setTarget(Puente);
        msg.sendToTarget();
        HiloCerrar Hilo = new HiloCerrar();
        Hilo.start();
    }

    Handler Puente = new Handler() {
        public void handleMessage(Message col) {
            String Respuesta = (String) col.obj;
            switch (Respuesta) {
                case "1":
                    AspectoTabla();
                    break;
            }
         }
    };
}
```

La función *AspectoTabla* permite mostrar los privilegios en una tabla con un aspecto adecuado y atractivo.

- Otorgar privilegios

El usuario tiene la capacidad de otorgar diferentes privilegios escogidos o bien todos y cuantos posea de la base de datos y tablas de esa base seleccionados.

Mediante la función *DarTodosPrivilegios* de *DataBaseManager.java* el usuario podrá otorgar a otro usuario todos los privilegios que disponga de una o todas las tablas pertenecientes a una base de datos.

```
public class HiloOtorgarTodosPrivilegios extends Thread {
    public void run() {
        String CapacidadOtorgar = " ";
        RadioButton selectRadio = (RadioButton)
DialogoPrivilegios.findViewById(RgPrivilegios.getCheckedRadioButtonId());
        if (selectRadio.getText().toString().equals("Si")) {
            CapacidadOtorgar = "WITH GRANT OPTION";
        }
        String Respuesta =
Manager.DarTodosPrivilegios(SpBases.getSelectedItem().toString(),
SpTablas.getSelectedItem().toString(), UsuarioPrivilegiado.getText().toString(),
CapacidadOtorgar);
        Message msg = Message.obtain();
        msg.obj = Respuesta;
        msg.setTarget(Puente);
        msg.sendToTarget();
        HiloCerrar Hilo = new HiloCerrar();
        Hilo.start();
    }

    Handler Puente = new Handler() {
        public void handleMessage(Message col) {
            String Respuesta = (String) col.obj;
            Circulo.setVisibility(View.GONE);
```

```
            switch (Respuesta) {
                case "1":
                    Toast.makeText(ListarBasesDeDatos.this, "Privilegios otorgados
satisfactoriamente", Toast.LENGTH_SHORT).show();
                    break;
            }
    };
}
```

Mediante la función *CompartirPrivilegios* de *DataBaseManager.java* el usuario podrá otorgar los privilegios seleccionados y de los que disponga de una o todas las tablas pertenecientes a una base de datos.

```
public class HiloOtorgarPrivilegios extends Thread {
    public void run() {
                String Respuesta = Manager.CompartirPrivilegios(PrivilegiosEscogidos,
SpBases.getSelectedItem().toString(), SpTablas.getSelectedItem().toString(),
UsuarioPrivilegiado.getText().toString(), CapacidadOtorgar);
        Message msg = Message.obtain();
        msg.obj = Respuesta;
        msg.setTarget(Puente);
        msg.sendToTarget();
        HiloCerrar Hilo = new HiloCerrar();
        Hilo.start();
    }

    Handler Puente = new Handler() {
        public void handleMessage(Message col) {
            String Respuesta = (String) col.obj;
            switch (Respuesta) {
                case "1":
                    Toast.makeText(ListarBasesDeDatos.this, "Privilegios otorgados
satisfactoriamente", Toast.LENGTH_SHORT).show();
                    break;

        }
    };
}
```

- Revocar privilegios

El usuario tiene la capacidad de revocar diferentes privilegios escogidos o bien todos y cuantos posea de la base de datos y tablas de esa base seleccionados.

Mediante la función *RevocarTodosPrivilegios* de *DataBaseManager.java* el usuario podrá revocar a otro usuario todos los privilegios que disponga de una o todas las tablas pertenecientes a una base de datos.

```
public class HiloRevocarTodosPrivilegios extends Thread {
    public void run() {
        String Respuesta =
Manager.RevocarTodosPrivilegios(SpBases.getSelectedItem().toString(),
SpTablas.getSelectedItem().toString(), UsuarioPrivilegiado.getText().toString());
        Message msg = Message.obtain();
```

```
        msg.obj = Respuesta;
        msg.setTarget(Puente);
        msg.sendToTarget();
        HiloCerrar Hilo = new HiloCerrar();
        Hilo.start();
    }

    Handler Puente = new Handler() {
        public void handleMessage(Message col) {
            String Respuesta = (String) col.obj;
            Circulo.setVisibility(View.GONE);
            switch (Respuesta) {
                case "1":
                    Toast.makeText(ListarBasesDeDatos.this, "Privilegios revocados
satisfactoriamente", Toast.LENGTH_SHORT).show();
                    break;
            }
    };
}
```

Mediante la función *RevocarPrivilegios* de *DataBaseManager.java* el usuario podrá revocar a otro usuario los privilegios seleccionados y de los que disponga de una o todas las tablas pertenecientes a una base de datos.

```
public class HiloRevocarPrivilegios extends Thread {
    public void run() {
        String Respuesta = Manager.RevocarPrivilegios(PrivilegiosEscogidos,
SpBases.getSelectedItem().toString(), SpTablas.getSelectedItem().toString(),
UsuarioPrivilegiado.getText().toString());
        Message msg = Message.obtain();
        msg.obj = Respuesta;
        msg.setTarget(Puente);
        msg.sendToTarget();
        HiloCerrar Hilo = new HiloCerrar();
        Hilo.start();
    }

    Handler Puente = new Handler() {
        public void handleMessage(Message col) {
            String Respuesta = (String) col.obj;
            switch (Respuesta) {
                case "1":
                    Toast.makeText(ListarBasesDeDatos.this, "Privilegios Revocados
Satisfactoriamente", Toast.LENGTH_SHORT).show();
                    break;
                }
        }
    };
}
```

3. ListarTablas.java

Esta actividad tiene como función principal la de permitir al usuario gestionar las tablas pertenecientes a una base de datos ubicada en un servidor MySQL externo al dispositivo. Para ello se necesitarán funciones tales como visualizar, crear y eliminar dichas tablas las cuales se estudiarán a continuación. Para ello en primer lugar se deben recoger los datos de conexión venideros.

- Visualizar Tablas

En primer lugar a través de la función *ListarTablas* de *DataBaseManager.java* se enviará una sentencia MySQL al servidor para recoger las tablas pertenecientes a una base de datos a las que el usuario tiene acceso y estas serán cargadas en una lista.

```
public class HiloMostrarListaTablas extends Thread {
    public void run() {
        String Respuesta;
        Tablas = Manager.ListarTablas(Base);
        Respuesta = "1";
        Message msg = Message.obtain();
        msg.obj = Respuesta;
        msg.setTarget(puente);
        msg.sendToTarget();
        HiloCerrar Hilo = new HiloCerrar();
        Hilo.start();
    }

    Handler puente = new Handler() {
        public void handleMessage(Message col) {
            String Respuesta = (String) col.obj;
            switch (Respuesta) {
                case "1":
                    ArrayAdapter<String> adapter = new
ArrayAdapter<>(ListarTablas.this, R.layout.estiloelistview, Tablas);
                    ListaTablas.setAdapter(adapter);
                    ListaTablas.addFooterView(AñadirLista);
                    break;
            }
            Circulo.setVisibility(View.GONE);
        }
    };
}
```

- Eliminar Tablas

El usuario a través de la aplicación podrá seleccionar una tabla de la lista y borrar dicha tabla tras confirmar dicha acción gracias al uso de la función *BorrarTabla* de *DataBaseManager.java.*

```
public class HiloBorrarTablas extends Thread {
    public void run() {
        String Respuesta = Manager.BorrarTabla(TablaElegida);
        Message msg = Message.obtain();
        msg.obj = Respuesta;
        msg.setTarget(puente);
        msg.sendToTarget();
        HiloCerrar Hilo = new HiloCerrar();
        Hilo.start();
    }

    Handler puente = new Handler() {
        public void handleMessage(Message col) {
            String Respuesta = (String) col.obj;
            switch (Respuesta) {
                case "1":
                    Toast.makeText(ListarTablas.this, "Borrado existoso",
Toast.LENGTH_SHORT).show();
                    finish();
                    startActivity(getIntent());
                    break;
            }
        }
    };
}
```

- Renombrar Tablas

El usuario a través de la aplicación podrá seleccionar una tabla de la lista y renombrar dicha tabla tras introducir un nuevo nombre gracias al uso de la función *RenombrarTabla* de *DataBaseManager.java.*

```
public class HiloRenombrarTabla extends Thread {
    public void run() {
        String Respuesta = Manager.RenombrarTabla(TablaElegida,
NuevaTabla.getText().toString());
        Message msg = Message.obtain();
        msg.obj = Respuesta;
        msg.setTarget(puente);
        msg.sendToTarget();
        HiloCerrar Hilo = new HiloCerrar();
        Hilo.start();
    }

    Handler puente = new Handler() {
        public void handleMessage(Message col) {
            String Respuesta = (String) col.obj;
            switch (Respuesta) {
                case "1":
                    Toast.makeText(ListarTablas.this, "Modificación existosa",
Toast.LENGTH_SHORT).show();
                    finish();
                    startActivity(getIntent());
                    break;
            }
        }
    };
}
```

- Vaciar Tablas

El usuario a través de la aplicación podrá seleccionar una tabla de la lista y eliminar todos los registros de dicha tabla tras confirmar dicha acción gracias al uso de la función *VaciarTabla* de *DataBaseManager.java.*

```
public class HiloVaciarTabla extends Thread {
    public void run() {
        String Respuesta = Manager.VaciarTabla(TablaElegida);
        Message msg = Message.obtain();
        msg.obj = Respuesta;
        msg.setTarget(puente);
        msg.sendToTarget();
        HiloCerrar Hilo = new HiloCerrar();
        Hilo.start();
    }

    Handler puente = new Handler() {
        public void handleMessage(Message col) {
            String Respuesta = (String) col.obj;
            switch (Respuesta) {
                case "1":
                    Toast.makeText(ListarTablas.this, "Modificación existosa",
Toast.LENGTH_SHORT).show();
                    finish();
                    startActivity(getIntent());
                    break;
            }
        }
    };
}
```

- Copiar Tablas (hacia dispositivo)

El usuario a través de la aplicación podrá copiar todos los registros de una tabla almacenada en un servidor MySQL externo y enviarlos al dispositivo apoyándose en el uso de tecnología SQLite. Para dicho proceso deberán ejecutarse cuatro sentencias SQL diferentes almacenadas en las funciones *MostrarTabla*, la cual permite recoger los registros de la tabla seleccionada, *InformacionCampos*, para obtener información acerca de los campos de la tabla ambas pertenecientes a la clase *DataBaseManager.java*, *CrearCopiaMovil*¸ para crear la tabla, con el mismo nombre que la del servidor, si es que esta no existiese, e *InsertarRegistros* para introducir los registros regidos a la tabla con el mismo nombre, estas últimas pertenecientes a la clase *DataBaseManagerMovil.java.*

```
public class HiloObtenerRegistros extends Thread {
    public void run() {
        String Respuesta;
```

```
        RegistrosObtenidos = Manager.MostrarTabla(TablaElegida);
        Respuesta = "1";

        Message msg = Message.obtain();
        msg.obj = Respuesta;
        msg.setTarget(puente);
        msg.sendToTarget();
        HiloCerrar Hilo = new HiloCerrar();
        Hilo.start();
    }

    Handler puente = new Handler() {
        public void handleMessage(Message col) {
            String Respuesta = (String) col.obj;
            switch (Respuesta) {
                case "1":
                    HiloObtenerAtributosCampos Hilo = new
HiloObtenerAtributosCampos();
                    Hilo.start();
                    break;
            }
        }

    };
}


public class HiloObtenerAtributosCampos extends Thread {
    public void run() {
        String Respuesta;
        InfoCampos = Manager.InformacionCampos(TablaElegida);
        Respuesta = "1";
        Message msg = Message.obtain();
        msg.obj = Respuesta;
        msg.setTarget(puente);
        msg.sendToTarget();
        HiloCerrar Hilo = new HiloCerrar();
        Hilo.start();
    }

    Handler puente = new Handler() {
        public void handleMessage(Message col) {
            String Respuesta = (String) col.obj;
            switch (Respuesta) {
                case "1":
                    CrearTablaMovil();
                    break;
            }
        }
    };
}


public void CrearTablaMovil() {
        String Respuesta = ManagerMovil.CrearCopiaTabla(TablaElegida, Campos, Tipos,
Longitud, Null, PrimaryKey);

    switch (Respuesta) {
        case "1":
            InsertarRegistros();
            break;
    }
}
```

```
public void InsertarRegistros() {
    String RespuestaFinal = "1";
    for (int i = 1; i < RegistrosObtenidos.length; i++) { // Empieza en 1 para no
contar la cabecera
        String Respuesta = ManagerMovil.InsertarRegistros(TablaElegida, Campos,
RegistrosObtenidos[i]);
        switch (Respuesta) {
            case "1":
                break;
        }
    }
```

- Copiar Tablas (hacia otro servidor MySQL)

El usuario a través de la aplicación podrá seleccionar una tabla almacenada en un servidor MySQL externo crear la misma tabla en otro servidor MySQL diferente. Para dicho proceso deberán ejecutarse cuatro sentencias MySQL diferentes almacenadas en las funciones *MostrarTabla*, la cual permite recoger los registros de la tabla seleccionada, *InformacionCampos*, para obtener información acerca de los campos de la tabla, *CrearTablas*¸ para crear la tabla con el mismo nombre que la ubicada en el dispositivo, si es que esta no existiese, e *InsertarDatos* para introducir los registros recogidos en la tabla con el mismo nombre. Todas estas funciones pertenecen a la clase *DataBaseManager.java*.

```
public class HiloObtenerRegistros extends Thread {
    public void run() {
        String Respuesta;
        RegistrosObtenidos = Manager.MostrarTabla(TablaElegida);
        Respuesta = "1";

        Message msg = Message.obtain();
        msg.obj = Respuesta;
        msg.setTarget(puente);
        msg.sendToTarget();
        HiloCerrar Hilo = new HiloCerrar();
        Hilo.start();
    }

    Handler puente = new Handler() {
        public void handleMessage(Message col) {
            String Respuesta = (String) col.obj;
            switch (Respuesta) {
                case "1":
                    HiloObtenerAtributosCampos Hilo = new
HiloObtenerAtributosCampos();
                    Hilo.start();
                    break;
            }
        }

    };
}
```

```
public class HiloObtenerAtributosCampos extends Thread {
    public void run() {
        String Respuesta;
        InfoCampos = Manager.InformacionCampos(TablaElegida);
        Respuesta = "1";
        Message msg = Message.obtain();
        msg.obj = Respuesta;
        msg.setTarget(puente);
        msg.sendToTarget();
        HiloCerrar Hilo = new HiloCerrar();
        Hilo.start();
    }

    Handler puente = new Handler() {
        public void handleMessage(Message col) {
            String Respuesta = (String) col.obj;
            switch (Respuesta) {
                case "1":
                    CrearTablaMovil();
                    break;
            }
        }
    };
}


public class HiloCrearTabla extends Thread {
    public void run() {
        ManagerNuevo = new DataBaseManager(DireccionIpNuevo, UsuarioNuevo,
ContraseñaNuevo, BaseNuevo);
        String Respuesta = ManagerNuevo.CrearTablas(TablaElegida, Campos, Tipos,
Longitud, Null, PrimaryKey, Auto);
        Message msg = Message.obtain();
        msg.obj = Respuesta;
        msg.setTarget(puente);
        msg.sendToTarget();
        HiloCerrar Hilo = new HiloCerrar();
        Hilo.start();
    }

    Handler puente = new Handler() {
        public void handleMessage(Message col) {
            String Respuesta = (String) col.obj;
            switch (Respuesta) {
                case "1":
                    HiloRegistrar Hilo = new HiloRegistrar();
                    Hilo.start();
                    break;
                default:
                    if (Respuesta.contains("already exist")) {
                        HiloRegistrar Hilo2 = new HiloRegistrar();
                        Hilo2.start();
                    } else {
 Manager = new DataBaseManager(DireccionIp, Usuario, Contraseña, Base);
 Toast.makeText(ListarTablas.this, Respuesta, Toast.LENGTH_LONG).show();
                    }
                    break;
            }
        }
    };
}
```

```
public class HiloRegistrar extends Thread {
    public void run() {
        String Respuesta = "1";
        if (RegistrosObtenidos.length > 1) {
            Respuesta = Manager.InsertarRegistros(TablaElegida, Campos,
RegistrosObtenidos[i]);
        }
        Message msg = Message.obtain();
        msg.obj = Respuesta;
        msg.setTarget(puente);
        msg.sendToTarget();
        HiloCerrar Hilo = new HiloCerrar();
        Hilo.start();
    }

    Handler puente = new Handler() {
        public void handleMessage(Message col) {
            String Respuesta = (String) col.obj;
            switch (Respuesta) {
                case "1":
                    if (i < RegistrosObtenidos.length - 1) {
                        i++;
                        HiloRegistrar Hilo = new HiloRegistrar();
                        Hilo.start();
                    } else {
                      Toast.makeText(ListarTablasMovil.this, "Registro existoso",
Toast.LENGTH_SHORT).show();
                    }
                    break;
            }
        }
    };
}
```

4. ListarTablasMovil.java

Esta actividad tiene como función principal la de permitir al usuario gestionar las tablas pertenecientes la base de datos con nombre 'BaseMySQLite' ubicada en el dispositivo, para ello se necesitarán funciones tales como visualizar, crear y eliminar dichas tablas, las cuales se discutirán a continuación. Para esto en primer lugar se debe recoger el objeto de conexión con la base de datos.

- Visualizar Tablas

Para dicho propósito a través de la función *ListarTablas* de *DataBaseManagerMovil.java* se ejecutará una sentencia SQLite para recoger las tablas y estas serán cargadas en una lista.

```
String Tablas = ManagerMovil.ListarTablas();
ArrayAdapter<String> adapter = new ArrayAdapter<>(ListarTablasMovil.this,
R.layout.estiloelistview, Tablas);
ListView ListaTablas.setAdapter(adapter);
```

- Eliminar Tablas

El usuario a través de la aplicación podrá seleccionar una tabla de la lista y borrar dicha tabla tras confirmar dicha acción gracias al uso de la función *BorrarTabla* de *DataBaseManagerMovil.java.*

```
String Respuesta = ManagerMovil.BorrarTabla(TablaElegida);
switch (Respuesta) {
    case "1":
        Toast.makeText(ListarTablasMovil.this, "Borrado existoso",
Toast.LENGTH_SHORT).show();
        finish();
        startActivity(getIntent());
        break;
}
```

- Vaciar Tablas

El usuario a través de la aplicación podrá seleccionar una tabla de la lista y eliminar todos los registros de dicha tabla tras confirmar dicha acción gracias al uso de la función *VaciarTabla* de *DataBaseManagerMovil.java.*

```
String Respuesta = ManagerMovil.VaciarTabla(TablaElegida);
switch (Respuesta) {
    case "1":
        Toast.makeText(ListarTablasMovil.this, "Borrado existoso",
Toast.LENGTH_SHORT).show();
        finish();
        startActivity(getIntent());
        break;
       }
```

- Copiar Tablas (hacia servidor)

El usuario podrá enviar todos los registros de una tabla almacenada en el dispositivo móvil a un servidor MySQL. Para dicho proceso deberán ejecutarse cuatro sentencias SQL diferentes almacenadas en las funciones *MostrarTabla*, que permite recoger los registros de la tabla seleccionada, *InformacionCampos*, para obtener información acerca de los campos de la tabla ambas pertenecientes a la clase *DataBaseManagerMovil.java*, *CrearTablas*¸ para crear la tabla con el mismo nombre que la ubicada en el dispositivo, si es que esta no existiese, e *InsertarDatos* para introducir los registros recogidos en la tabla con el mismo nombre, estas últimas pertenecientes a la clase *DataBaseManager.java*.

```
public void ObtenerRegistros() {
     String[][] RegistrosObtenidos = ManagerMovil.MostrarTabla(TablaElegida);
}


public void ObtenerAtributosCampos() {
     String[][ InfoCampos = ManagerMovil.InformacionCampos(TablaElegida);
}


public class HiloCrearTabla extends Thread {
    public void run() {
        String Respuesta = Manager.CrearTablas(TablaElegida, Campos, Tipos, Longitud,
Null, PrimaryKey, Auto);
        Message msg = Message.obtain();
        msg.obj = Respuesta;
        msg.setTarget(puente);
        msg.sendToTarget();
        HiloCerrar Hilo = new HiloCerrar();
        Hilo.start();
    }

    Handler puente = new Handler() {
        public void handleMessage(Message col) {
            String Respuesta = (String) col.obj;
            switch (Respuesta) {
                case "1":
                    HiloRegistrar Hilo = new HiloRegistrar();
                    Hilo.start();
                    break;
                default:
                    if (Respuesta.contains("already exist")) {
                        HiloRegistrar Hilo2 = new HiloRegistrar();
                        Hilo2.start();
                    } else {
 Manager = new DataBaseManager(DireccionIp, Usuario, Contraseña, Base);
 Toast.makeText(ListarTablas.this, Respuesta, Toast.LENGTH_LONG).show();
                    }
                    break;
            }
        }
    };
}


public class HiloRegistrar extends Thread {
    public void run() {
        String Respuesta = "1";
        if (RegistrosObtenidos.length > 1) {
Respuesta = Manager.InsertarRegistros(TablaElegida, Campos, RegistrosObtenidos[i]);
        }
        Message msg = Message.obtain();
        msg.obj = Respuesta;
        msg.setTarget(puente);
        msg.sendToTarget();
        HiloCerrar Hilo = new HiloCerrar();
        Hilo.start();
    }

    Handler puente = new Handler() {
        public void handleMessage(Message col) {
            String Respuesta = (String) col.obj;
```

```
            switch (Respuesta) {
                case "1":
                    if (i < RegistrosObtenidos.length - 1) {
                        i++;
                        HiloRegistrar Hilo = new HiloRegistrar();
                        Hilo.start();
                    } else {
                      Toast.makeText(ListarTablasMovil.this, "Registro existoso",
Toast.LENGTH_SHORT).show();
                    }
                    break;
            }
        }
    };
}
```

Tanto *ListarTablas.java como ListarTablasMovil.java* contendrán un menú que permitirá a la aplicación desplazarse a las actividades que a continuación se mostrarán cuyas funciones están relacionadas directamente con la gestión de tablas.

5. CrearTablas.java y CrearTablasMovil.java

La función principal de estas actividades es la de proporcionar al usuario un menú adecuado para la creación de tablas. La actividad *CrearTabla.java* permitirá crear una tabla en un servidor MySQL externo al dispositivo mientras que *CrearTablaMovil.java* lo permitirá en el mismo dispositivo.

- Menú de creación

Ambas actividades mostrarán en sus respectivas interfaces un menú adecuado para la creación de tablas muy similar, para ello en primer lugar el usuario indicará la cantidad de campos que incluirá en la tabla y tras esto aparecerá un menú más extenso para introducir los parámetros propios de cada campo, es decir, nombre del campo, tipo, longitud, capacidad de valores nulos, clave primaria y, solo en la actividad *CrearTablaMovil.java*, opciones referentes a la posibilidad de establecer claves foráneas, es decir, la tabla de referencia y el campo de referencia.

- CrearTablas.java

Después de que el usuario introduzca los datos referentes a los campos de la tabla esta podrá ser creada recogiendo los datos introducidos en el menú gracias al uso de la función *CrearTablas* de *DataBaseManager.java*.

```
public class HiloCrearTabla extends Thread {
    public void run() {
        String Respuesta = Manager.CrearTablas(Tabla, CamposString, TipoString,
LongitudString, NullString, PrimaryKeyString, AutoString);
        Message msg = Message.obtain();
        msg.obj = Respuesta;
        msg.setTarget(Puente);
        msg.sendToTarget();
        HiloCerrar Hilo = new HiloCerrar();
        Hilo.start();
    }

    Handler Puente = new Handler() {
        public void handleMessage(Message col) {
            String Respuesta = (String) col.obj;
            switch (Respuesta) {
                case "1":
                    Toast.makeText(CrearTablas.this, "Registro existoso",
Toast.LENGTH_SHORT).show();
                    break;
           }
        }
   };
}
```

- CrearTablasMovil.java

Después de que el usuario introduzca los datos referentes a los campos de la tabla esta podrá ser creada recogiendo los datos introducidos en el menú gracias al uso de la función *CrearTablas* de *DataBaseManagerMovil.java*.

```
String Respuesta = ManagerMovil.CrearTablas(Tabla, CamposString, TipoString,
LongitudString, NullString, PrimaryKeyString, ForeingKeyString,
TablasReferenciaString, CamposReferenciaString);

switch (Respuesta) {
    case "1":
        Toast.makeText(CrearTablasMovil.this, "Registro existoso",
Toast.LENGTH_SHORT).show();
        Intent Intent = new Intent(CrearTablasMovil.this, ListarTablasMovil.class);
        startActivity(Intent);
        break;
```

6. MostrarTabla.java y MostrarTablaMovil.java

La función principal de estas actividades es la de mostrar todos los registros de una tabla que el usuario haya seleccionado previamente de forma ordenada si este así lo requiere. La actividad *MostrarTabla.java* permitirá mostrar los registros de una tabla almacenada en un servidor MySQL externo al dispositivo mientras que *MostrarTablaMovil.java* lo permitirá con una tabla ubicada en el mismo dispositivo.

- MostrarTabla.java

A través de la función *MostrarTabla* y *MostrarTablaOrden* de *DataBaseManager.java* el usuario podrá visualizar la tabla ordenada o no gracias al uso de un menú con el cual podrá decidir el campo y el orden por el que los registros son mostrados de forma ordenada.

```
public class HiloMostrarTabla extends Thread {
    public void run() {
        String Respuesta;
        EstructuraTabla = Manager.MostrarTabla(Tabla);
        Respuesta = "1";
        Message msg = Message.obtain();
        msg.obj = Respuesta;
        msg.setTarget(Puente);
        msg.sendToTarget();
        HiloCerrar Hilo = new HiloCerrar();
        Hilo.start();
    }

    Handler Puente = new Handler() {
        public void handleMessage(Message col) {
            String Respuesta = (String) col.obj;
            switch (Respuesta) {
                case "1":
                    AspectoTabla();
                    break;
            }
            Circulo.setVisibility(View.GONE);
        }
    };
}


public class HiloMostrarTablaOrdenada extends Thread {
    public void run() {
        String Respuesta;
        EstructuraTabla = Manager.MostrarTablaOrden(Tabla,
SpCampos.getSelectedItem().toString(), SpOrden.getSelectedItem().toString());
        Respuesta = "1";
        Message msg = Message.obtain();
        msg.obj = Respuesta;
        msg.setTarget(Puente);
        msg.sendToTarget();
        HiloCerrar Hilo = new HiloCerrar();
        Hilo.start();
    }

    Handler Puente = new Handler() {
        public void handleMessage(Message col) {
            String Respuesta = (String) col.obj;
            switch (Respuesta) {
                case "1":
                    AspectoTabla();
                    break;
            }
            Circulo.setVisibility(View.GONE);
        }
    };
}
```

Esta actividad cuenta además con una función que le permitirá al usuario realizar consultas de la tabla según un campo elegido y un valor de ese campo introducido gracias al uso de la función *MostrarTablaSelección* de *DataBaseManager.java*.

```
public class HiloMostrarTablaSelección extends Thread {
    public void run() {
            EstructuraTabla = Manager.MostrarTablaSelección(Tabla,
SpCamposObtener.getSelectedItem().toString(), ValorObtener.getText().toString());
            Respuesta = "1";
        Message msg = Message.obtain();
        msg.obj = Respuesta;
        msg.setTarget(Puente);
        msg.sendToTarget();
        HiloCerrar Hilo = new HiloCerrar();
        Hilo.start();
    }

    Handler Puente = new Handler() {
        public void handleMessage(Message col) {
            String Respuesta = (String) col.obj;
            switch (Respuesta) {
                case "1":
                    ValorObtener.setText("");
                    AspectoTabla();
                    break;
            }
            Circulo.setVisibility(View.GONE);
        }
    };
}
```

- MostrarTablaMovil.java

A través de la función *MostrarTabla* y *MostrarTablaOrden* de *DataBaseManagerMovil.java* el usuario podrá visualizar la tabla ordenada o no gracias al uso de un menú con el cual podrá decidir el campo y el orden por el que los registros son mostrados ordenadamente.

```
public void MostrarTabla() {
    EstructuraTabla = ManagerMovil.MostrarTabla(Tabla);
    AspectoTabla();
}

public void MostrarTablaOrdenada() {
        EstructuraTabla = ManagerMovil.MostrarTablaOrden(Tabla,
SpCampos.getSelectedItem().toString(), SpOrden.getSelectedItem().toString());
        AspectoTabla();
}
```

Esta actividad cuenta además con una función que le permitirá al usuario realizar consultas de la tabla según un campo elegido y un valor de ese campo introducido gracias al uso de la función *MostrarTablaSelección* de *DataBaseManager.java*.

```
public void MostrarTablaSelección() {
        EstructuraTabla = ManagerMovil.MostrarTablaSelección(Tabla,
SpCamposObtener.getSelectedItem().toString(), ValorObtener.getText().toString());
        AspectoTabla();
}
```

Como puede apreciarse ambas actividades recurren a otra función llamada *AspectoTabla* la cual, después de recuperar los registros de la tabla, permite que estos sean mostrados de una forma estética en el dispositivo.

7. InsertarRegistros.java e InsertarRegistroMovil.java

Ambas actividades ofrecen funcionalidades adecuadas para la inserción de registros en una tabla a través de un menú adecuado para ello. La actividad *InsertarRegistros.java* permite insertar registros a una tabla ubicada en un servidor MySQL externo al dispositivo mientras que *InsertarRegistroMovil.java* lo permitirá con una tabla ubicada en el mismo dispositivo.

- InsertarRegistros.java

El menú adecuado para la introducción de datos contendrá información diversa sobre los parámetros de los campo que forman la tabla, es decir, el nombre del campo, el tipo de dato que admite el campo, la posibilidad o no que los registros del campo tomen valores nulos, de ser clave primaria de la tabla o no y de ser autoincrementable. Dicho menú puede ser utilizado gracias al uso de la función *InformaciónCampos* de *DataBaseManager.java* la cual contiene la sentencia MySQL que permite obtener la información de los campos pertinente.

```
public class HiloMostrarMenu extends Thread {
    public void run() {
        String Respuesta;
        EstructuraTabla = Manager.InformacionCampos(Tabla);
        Respuesta = "1";
        Message msg = Message.obtain();
        msg.obj = Respuesta;
        msg.setTarget(Puente);
        msg.sendToTarget();
        HiloCerrar Hilo = new HiloCerrar();
        Hilo.start();
    }

    Handler Puente = new Handler() {
        public void handleMessage(Message col) {
            String Respuesta = (String) col.obj;
```

```
            switch (Respuesta) {
                case "1":
                    MenuInsertar();
                    break;
            }
           Circulo.setVisibility(View.GONE);
        }
    };
}
```

Una vez que el usuario introduzca los datos estos podrán ser introducidos en la tabla gracias al uso de la función *InsertarRegistros* de *DataBaseManager.java.*

```
public class HiloRegistrar extends Thread {
    public void run() {
        String[] Cabecera = new String[Campos.length];
        String[] Datos = new String[Intro.length];

        for (int i = 0; i < Intro.length; i++) {
            Datos[i] = Intro[i].getText().toString();
        }

        for (int i = 0; i < Campos.length; i++) {
            Cabecera[i] = Campos[i].getText().toString();
        }

        String Respuesta = Manager.InsertarRegistros(Tabla, Cabecera, Datos);

        Message msg = Message.obtain();
        msg.obj = Respuesta;
        msg.setTarget(Puente);
        msg.sendToTarget();
        HiloCerrar Hilo = new HiloCerrar();
        Hilo.start();
    }

    Handler Puente = new Handler() {
        public void handleMessage(Message col) {
            String Respuesta = (String) col.obj;
            switch (Respuesta) {
                case "1":
                    Toast.makeText(InsertarRegistros.this, "Registro existoso",
Toast.LENGTH_SHORT).show();
                    break;
            }
        }
    };
}
```

- InsertarRegistrosMovil.java

El menú adecuado para la introducción de datos contendrá información diversa sobre los parámetros de los campo que forman la tabla, es decir, el nombre del campo, el tipo de dato que admite el campo, la posibilidad o no que los registros del campo tomen valores nulos, de ser clave primaria de la tabla o no y de ser autoincrementable.

Dicho menú puede ser utilizado gracias al uso de la función *InformaciónCampos* de *DataBaseManagerMovil.java* la cual contiene la sentencia MySQL que permite obtener la información de los campos pertinente.

```
String [][] EstructuraTabla = ManagerMovil.InformacionCampos(Tabla);
MenuInsertar();
```

Una vez que el usuario introduzca los datos estos podrán ser introducidos en la tabla gracias al uso de la función *InsertarRegistros* de *DataBaseManagerMovil.java.*

```
String Respuesta = ManagerMovil.InsertarRegistros(Tabla, Cabecera, Datos);
switch (Respuesta) {
    case "1":
        Toast.makeText(InsertarRegistrosMovil.this, "Registro existoso",
Toast.LENGTH_SHORT).show();
    break;
}
```

La función *MenuInsertar* presente e idéntica en ambas actividades permite mostrar el menú con los parámetros de los campos de una forma estética.

8. AgregarCampos.java y AgregarCampoMovil.java

Ambas actividades ofrecen funcionalidades adecuadas para la agregación de nuevos campos a una tabla ya existente a través de un menú adecuado para ello. La actividad *AgregarCampos.java* permite agregar campos a una tabla ubicada en un servidor MySQL externo al dispositivo mientras que *AgregarCampoMovil.java* lo permitirá con una tabla ubicada en el mismo dispositivo.

- Menú de agregación

Ambas actividades mostrarán en sus respectivas interfaces un menú adecuado para la agregación de campos, para ello en primer lugar el usuario indicará la cantidad de campos que añadirá a la tabla, tras esto aparecerá un menú más extenso para introducir los parámetros propios de cada campo, es decir, nombre del campo, tipo, longitud y capacidad de valores nulos, en adición y solo en la actividad *AgregarCampos.java* se podrá elegir el lugar donde se incluirá el nuevo campo referente a los campos ya existentes. En *AgregarCamposMovil.java* deberá ir siempre al final

- AgregarCampos.java

Después de que el usuario introduzca los datos referentes a los nuevos campos estos podrán ser agregados recogiendo los parámetros introducidos en el menú gracias al uso de la función *AgregarCampos* de *DataBaseManager.java*.

```
public class HiloAñadirCampo extends Thread {
    public void run() {
        String Respuesta = Manager.AgregarCampos(Tabla, CamposString, TipoString,
LongitudString, NullString, Opcion, SpCampos.getSelectedItem().toString());
        Message msg = Message.obtain();
        msg.obj = Respuesta;
        msg.setTarget(Puente);
        msg.sendToTarget();
        HiloCerrar Hilo = new HiloCerrar();
        Hilo.start();
    }

    Handler Puente = new Handler() {
        @TargetApi(Build.VERSION_CODES.ICE_CREAM_SANDWICH_MR1)
        public void handleMessage(Message col) {
            String Respuesta = (String) col.obj;
            switch (Respuesta) {
                case "1":
                    Toast.makeText(AgregarCampos.this, "Registro existoso",
Toast.LENGTH_SHORT).show();
                    Intro.callOnClick();
                    break;
            }
        }
    };
}
```

- AgregarCamposMovil.java

Después de que el usuario introduzca los datos referentes a los nuevos campos estos podrán ser agregados recogiendo los parámetros introducidos en el menú gracias al uso de la función *AgregarCampos* de *DataBaseManagerMovil.java*.

```
String Respuesta = ManagerMovil.AgregarCampos(Tabla, CamposString, TipoString,
LongitudString, NullString);

        switch (Respuesta) {
            case "1":
                Toast.makeText(AgregarCamposMovil.this, "Registro existoso",
Toast.LENGTH_SHORT).show();
                Intro.callOnClick();
                break;
        }
    }

});
```

9. Editar.java y EditarMovil.java

Ambas actividades ofrecen funcionalidades adecuadas para la edición de registros y valores de estos pertenecientes a una tabla a través de un menú adecuado para ello, entendiendo por edición como modificación y eliminación.

La actividad *Editar.java* permite dichas acciones en una tabla ubicada en un servidor MySQL externo al dispositivo mientras que *EditarMovil.java* lo permitirá con una tabla ubicada en el mismo dispositivo. La actividad *Editar.java* además permite la edición de campos.

- Edición de campos

Solo disponible en *Editar.java* permite la edición de los campos de una tabla, es decir, dispondrá de las funcionalidades suficientes para borrar cualquier campo, y los consecuentes registros pertenecientes a la columna de la que forma parte dicho campo, y para modificar los parámetros pertenecientes a los campos anteriormente mencionados.

Eliminación de campos:

Posible gracias al uso de la función *BorrarCampo* de *DataBaseManager.java*

```
public class HiloBorrarCampo extends Thread {
    public void run() {
        CampoAntiguo = SpCampos.getSelectedItem().toString();
        String Respuesta = Manager.BorrarCampo(Tabla, CampoAntiguo);
        Message msg = Message.obtain();
        msg.obj = Respuesta;
        msg.setTarget(Puente);
        msg.sendToTarget();
        HiloCerrar Hilo = new HiloCerrar();
        Hilo.start();
    }

    Handler Puente = new Handler() {
        public void handleMessage(Message col) {
            String Respuesta = (String) col.obj;
            switch (Respuesta) {
                case "1":
                    Toast.makeText(Editar.this, "Borrado existoso",
Toast.LENGTH_SHORT).show();
                    break;
            }
        }
    };
}
```

Modificación de campos:

Posible gracias al uso de la función *ModificarCampo* de *DataBaseManager.java*

```
public class HiloModificarCampos extends Thread {
    public void run() {
        String Respuesta = Manager.ModificarCampo(Tabla, CampoAntiguo, CampoNuevo,
TipoString, LongitudString, NullString, AutoString);
        Message msg = Message.obtain();
        msg.obj = Respuesta;
        msg.setTarget(Puente);
        msg.sendToTarget();
        HiloCerrar Hilo = new HiloCerrar();
        Hilo.start();
    }

    Handler Puente = new Handler() {
        public void handleMessage(Message col) {
            String Respuesta = (String) col.obj;
            switch (Respuesta) {
                case "1":
                    Toast.makeText(Editar.this, "Modificación existosa",
Toast.LENGTH_SHORT).show();
                    break;
                }
        }
    };
}
```

Asignación de clave primaria:

Posible gracias al uso de la función *AñadirClave* de *DataBaseManager.java*

```
public class HiloEstablecerClave extends Thread {
    public void run() {
        if (CamposClavesyTipos.length == 0) {
            int ExisteClave = 0;
        } else {
            int ExisteClave = 1;
        }
        String Respuesta = Manager.AñadirClave(Tabla,
SpPosiblesClaves.getSelectedItem().toString(), CamposClavesPrimarias, ExisteClave);
        Message msg = Message.obtain();
        msg.obj = Respuesta;
        msg.setTarget(Puente);
        msg.sendToTarget();
        HiloCerrar Hilo = new HiloCerrar();
        Hilo.start();
    }

    Handler Puente = new Handler() {
        public void handleMessage(Message col) {
            String Respuesta = (String) col.obj;
            switch (Respuesta) {
                case "1":
                    Toast.makeText(Editar.this, "Registro existoso",
Toast.LENGTH_SHORT).show();
                    break;
            }
        }
    };
}
```

Eliminación de clave primaria:

Posible gracias al uso de la función *BorrarClave* de *DataBaseManager.java*

```
public class HiloBorrarClave extends Thread {
    public void run() {
        String Respuesta = Manager.BorrarClavePrimaria(Tabla);
        Message msg = Message.obtain();
        msg.obj = Respuesta;
        msg.setTarget(Puente);
        msg.sendToTarget();
        HiloCerrar Hilo = new HiloCerrar();
        Hilo.start();
    }

    Handler Puente = new Handler() {
        public void handleMessage(Message col) {
            String Respuesta = (String) col.obj;
            switch (Respuesta) {
                case "1":
                    Toast.makeText(Editar.this, "Borrado existoso",
Toast.LENGTH_SHORT).show();
                    finish();
                    startActivity(getIntent());
                    break;
            }
        }
    };
}
```

- Edición de registros

Se permite la edición de los registros de una tabla, es decir, se dispondrá de las funcionalidades suficientes para borrar cualquier registro y para modificar los valores de pertenecientes a dicho registro.

Eliminación de registros en *Editar.java*:

Posible gracias al uso de la función *BorrarRegistro* de *DataBaseManager.java*

```
public class HiloBorrarRegistro extends Thread {
    public void run() {
        FilaElegida = SpRegistrosClavesPrimarias.getSelectedItem().toString();
        String[] CamposClaves = FilaElegida.split(" - ");
        String Respuesta = Manager.BorrarRegistro(Tabla, CamposClavesPrimarias,
CamposClaves);
        Message msg = Message.obtain();
        msg.obj = Respuesta;
        msg.setTarget(Puente);
        msg.sendToTarget();
        HiloCerrar Hilo = new HiloCerrar();
        Hilo.start();
    }


    Handler Puente = new Handler() {
        public void handleMessage(Message col) {
```

```
            String Respuesta = (String) col.obj;
            switch (Respuesta) {
                case "1":
                    Toast.makeText(Editar.this, "Borrado existoso",
Toast.LENGTH_SHORT).show();
                    break;
            }
        }
    };
}
```

Eliminación de registros en *EditarMovil.java*:

Posible gracias al uso de la función *BorrarRegistro* de *DataBaseManageMovil.java*

```
public void BorrarRegistro() {
    FilaElegida = SpRegistrosClavesPrimarias.getSelectedItem().toString();
    String[] CamposClaves = FilaElegida.split(" - ");
    String Respuesta = ManagerMovil.BorrarRegistro(Tabla, CamposClavesPrimarias,
CamposClaves);
    switch (Respuesta) {
        case "1":
            Toast.makeText(EditarMovil.this, "Borrado existoso",
Toast.LENGTH_SHORT).show();
            break;
    }
}
```

Modificación de registros en *Editar.java*:

Posible gracias al uso de la función *ModificarRegistro* de *DataBaseManager.java*

```
public class HiloModificarRegistro extends Thread {
    public void run() {
        String[] DatosFilaNuevos = new String[FilaNueva.length - 2];
        for (int i = 0; i < FilaNueva.length - 2; i++) {
            DatosFilaNuevos[i] = FilaNueva[i + 2].getText().toString(); }
        FilaElegida = SpRegistrosClavesPrimarias.getSelectedItem().toString();
        String[] CamposClaves = FilaElegida.split(" - ");
        String Respuesta = Manager.ModificarRegistro(Tabla, Campos, DatosFilaNuevos,
CamposClavesPrimarias, CamposClaves);
        Message msg = Message.obtain();
        msg.obj = Respuesta;
        msg.setTarget(Puente);
        msg.sendToTarget();
        HiloCerrar Hilo = new HiloCerrar();
        Hilo.start();
    }

    Handler Puente = new Handler() {
        public void handleMessage(Message col) {
            String Respuesta = (String) col.obj;
            switch (Respuesta) {
                case "1":
Toast.makeText(Editar.this, "Modificación existosa", Toast.LENGTH_SHORT).show();
                    break;
            }
        }
    };
}
```

Modificación de registros en *EditarMovil.java*:

Posible gracias al uso de la función *ModificarRegistro* de *DataBaseManageMovil.java*

```
public void ModificarRegistro() {
    String[] DatosFilaNuevos = new String[FilaNueva.length - 2];
    for (int i = 0; i < FilaNueva.length - 2; i++) {
        DatosFilaNuevos[i] = FilaNueva[i + 2].getText().toString();
    }
    FilaElegida = SpRegistrosClavesPrimarias.getSelectedItem().toString();
    String[] CamposClaves = FilaElegida.split(" - ");
    String Respuesta = ManagerMovil.ModificarRegistro(Tabla, Campos, DatosFilaNuevos,
CamposClavesPrimarias, CamposClaves);
    switch (Respuesta) {
        case "1":
            Toast.makeText(EditarMovil.this, "Modificación existosa",
Toast.LENGTH_SHORT).show();
            break;
    }
}
```

- Edición de valores

Se permite la edición de los valores individuales de una tabla, es decir, se dispondrá de las funcionalidades suficientes para borrar o modificar cualquier valor perteneciente a cualquier registro de la tabla.

Eliminación de valores en *Editar.java*:

Posible gracias al uso de la función *BorrarValores* de *DataBaseManager.java*

```
public class HiloBorrarValores extends Thread {
    public void run() {
        FilaElegida = SpRegistrosClavesPrimarias.getSelectedItem().toString();
        String[] CamposClaves = FilaElegida.split(" - ");
        String Respuesta = Manager.BorrarValor(Tabla,
SpCampos.getSelectedItem().toString(), CamposClavesPrimarias, CamposClaves);
        Message msg = Message.obtain();
        msg.obj = Respuesta;
        msg.setTarget(Puente);
        msg.sendToTarget();
        HiloCerrar Hilo = new HiloCerrar();
        Hilo.start();
    }

    Handler Puente = new Handler() {
        public void handleMessage(Message col) {
            String Respuesta = (String) col.obj;
            switch (Respuesta) {
                case "1":
                    Toast.makeText(Editar.this, "Modificación existosa",
Toast.LENGTH_SHORT).show();
                    break;
            }
        }
    };
}
```

Eliminación de valores en *EditarMovil.java*:

Posible gracias al uso de la función *BorrarValor* de *DataBaseManageMovil.java*

```
public void BorrarRegistro() {
    FilaElegida = SpRegistrosClavesPrimarias.getSelectedItem().toString();
    String[] CamposClaves = FilaElegida.split(" - ");
    String NuevoRecursoString = NuevoRegistro.getText().toString();
    String Respuesta = ManagerMovil.BorrarValor(Tabla,
SpCampos.getSelectedItem().toString(), NuevoRecursoString, CamposClavesPrimarias,
CamposClaves);
    switch (Respuesta) {
        case "1":
            Toast.makeText(EditarMovil.this, "Borrado existoso",
Toast.LENGTH_SHORT).show();
            break;
        case "0":
            Toast.makeText(EditarMovil.this, "Error inesperado contacte con el
administrador", Toast.LENGTH_SHORT).show();
            break;
        default:
            Toast.makeText(EditarMovil.this, Respuesta, Toast.LENGTH_LONG).show();
            break;
    }
}
```

Modificación de valores en *Editar.java*:

Posible gracias al uso de la función *ModificarValor* de *DataBaseManager.java*

```
public class HiloModificarRegistro extends Thread {
    public void run() {
        FilaElegida = SpRegistrosClavesPrimarias.getSelectedItem().toString();
        String[] CamposClaves = FilaElegida.split(" - ");
        String NuevoRegistrostring = NuevoRegistro.getText().toString();
        String Respuesta = Manager.ModificarValor(Tabla,
SpCampos.getSelectedItem().toString(), NuevoRegistrostring, CamposClavesPrimarias,
CamposClaves);
        Message msg = Message.obtain();
        msg.obj = Respuesta;
        msg.setTarget(Puente);
        msg.sendToTarget();
        HiloCerrar Hilo = new HiloCerrar();
        Hilo.start();
    }

    Handler Puente = new Handler() {
        public void handleMessage(Message col) {
            String Respuesta = (String) col.obj;
            switch (Respuesta) {
                case "1":
                    Toast.makeText(Editar.this, "Modificación existosa",
Toast.LENGTH_SHORT).show();
                    break;
            }
        }
    };
}
```

Modificación de valores en *EditarMovil.java*:

Posible gracias al uso de la función *ModificarValor* de *DataBaseManageMovil.java*

```
public void ModificarRegistro() {
    FilaElegida = SpRegistrosClavesPrimarias.getSelectedItem().toString();
    String[] CamposClaves = FilaElegida.split(" - ");
    String NuevoRecursoString = NuevoRegistro.getText().toString();
    String Respuesta = ManagerMovil.ModificarValor(Tabla,
SpCampos.getSelectedItem().toString(), NuevoRecursoString, CamposClavesPrimarias,
CamposClaves);
    switch (Respuesta) {
        case "1":
            Toast.makeText(EditarMovil.this, "Modificación existosa",
Toast.LENGTH_SHORT).show();
            break;
    }
}
```

10. ClavesForaneas.java

Esta actividad ofrece diferentes funcionalidades adecuadas para el establecimiento y eliminación de propiedades de clave foránea en campos de tablas ubicadas en un servidor MySQL externo a través de un menú adecuado para ello. Dicho menú contendrá parámetros referentes a la tabla sobre la que establecer la propiedad y a sus campos y a la tabla de referencia y a sus propios campos.

- Establecimiento de clave foránea

Después de que el usuario seleccione la tabla y el campo de esta tabla sobre el que establecer la propiedad y el campo referente a otra tabla deberá introducir además las opciones de actuación frente a situaciones de modificación y borrado de registros referenciados, tras esto entrará en acción la función *CrearClavesForaneas* perteneciente a la clase *DataBaseManager.java* la cual permite el establecimiento de la propiedad de clave foránea en un campo según los parámetros anteriormente mencionados.

```
public class HiloCrearClaveForanea extends Thread {
    public void run() {
        String Respuesta =
Manager.CrearClavesForaneas(SpTablas.getSelectedItem().toString(),
SpCampos.getSelectedItem().toString(),
SpTablasReferencia.getSelectedItem().toString(),
SpCamposReferencia.getSelectedItem().toString(), OpcionBorrar, OpcionModificar);
        Message msg = Message.obtain();
        msg.obj = Respuesta;
        msg.setTarget(puente);
        msg.sendToTarget();
        HiloCerrar Hilo = new HiloCerrar();
        Hilo.start();
    }
```

```
    Handler puente = new Handler() {
        public void handleMessage(Message col) {
            String Respuesta = (String) col.obj;
            switch (Respuesta) {
                case "1":
                    Toast.makeText(ClavesForaneas.this, "Clave Foránea Creada",
Toast.LENGTH_SHORT).show();
                    break;
            }
        }
    };
}
```

- Eliminación de clave foránea

Después de que el usuario seleccione la tabla y el campo de esta tabla que contenga la propiedad de clave foránea deberá seleccionar, si es que tiene más de una de estas propiedades, cuál de estas desea eliminar seleccionando para ello la referencia de la misma la cual actúa como identificador de ella, tras esto entrará en acción la función *BorrarClavesForaneas* perteneciente a la clase *DataBaseManager.java* la cual permite la eliminación de la propiedad de clave foránea en un campo según los parámetros anteriormente mencionados.

```
public class HiloBorrarClaveForanea extends Thread {
    public void run() {
        String Respuesta =
Manager.BorrarClavesForaneas(SpTablasBorrarForaneas.getSelectedItem().toString(),
SprestriccionBorrarForaneas.getSelectedItem().toString());
        Message msg = Message.obtain();
        msg.obj = Respuesta;
        msg.setTarget(puente);
        msg.sendToTarget();
        HiloCerrar Hilo = new HiloCerrar();
        Hilo.start();
    }

    Handler puente = new Handler() {
        public void handleMessage(Message col) {
            String Respuesta = (String) col.obj;
            switch (Respuesta) {
                case "1":
                    Toast.makeText(ClavesForaneas.this, "Borrado exitoso",
Toast.LENGTH_SHORT).show();
                    break;
            }
        }
    };
}
```

11. LogSQL.java y LogSQLMovil.java

Ambas actividades tienen como función la de mostrar las últimas sentencias MySQL y SQLite ejecutadas por la aplicación respectivamente.

- LogSQL.java

En este caso las sentencias MySQL se almacenarán en la clase estática *RegistroLog.java* desde la clase *ConexiónBD.java* a la cual se le envían todas las sentencias introducidas en las funciones a la clase *DataBaseManager.java*. Finalmente la clase *LogSQL.java* recoge las sentencias almacenadas en *RegistroLog.java* y las muestra por pantalla.

```
TextView LogActual = (TextView) findViewById(R.id.TvLog);
String Log = RegistroLog.GetLogSQL();
LogActual.setText(Log);
```

- LogSQLMovil.java

En este caso las sentencias SQLite se almacenarán en la clase estática *RegistroLogMovil.java* directamente las sentencias introducidas en las funciones a la clase *DataBaseManagerMovil.java*. Finalmente la clase *LogSQMovilL.java* recoge las sentencias almacenadas en *RegistroLogMovil.java* y las muestra por pantalla.

```
TextView LogActual = (TextView) findViewById(R.id.TvLog);
String Log = RegistroLogMovil.GetLogSQL();
LogActual.setText(Log);
```

- Control de errores

Toda ejecución de una sentencia MySQL o SQLite puede dar lugar a un error, de modo que se ha implementado un sistema de control de errores que se ejecutará nada más analizar la respuesta del servidor después de haberse enviado la sentencia. Dicho control de errores tendrá en cuenta tres situaciones posibles, que la respuesta sea positiva y por tanto la sentencia haya sido correctamente ejecutada, que se haya producido un error en la construcción de la sentencia SQL, controlado gracias al uso de la excepción *SQLException* o *SQLiteException* de java y por último un error no previsto de cualquier otro tipo, controlado gracias al uso de la excepción *Exception* de java.

A continuación se muestra un ejemplo del código para el control de errores en una sentencia de consulta para obtener las tablas almacenadas en una base de datos.

```
try {
    Toast.makeText(ListarTablasMovil.this, "Lista de tablas cargadas",
    Toast.LENGTH_SHORT).show(); // Ejecución aceptada
} catch (SQLiteException e) {
    Toast.makeText(ListarTablasMovil.this, e.getMessage(),
    Toast.LENGTH_LONG).show(); // Muestra mensaje de error SQL
} catch (Exception e) {
    Toast.makeText(ListarTablasMovil.this, "Error inesperado contacte
    con el administrador", Toast.LENGTH_SHORT).show(); // Erros inesperado
}
```

- Relación actividades con interfaces

Cada actividad anteriormente mencionada irá asociada a un documento XML que determinará la interfaz gráfica en la que se desarrolle. Algunas actividades compartirán la misma interfaz por su cierto parecido. En la **Tabla 3.4** puede apreciarse la relación entre actividades e interfaces

Tabla Implementación Interfaces	
Actividades	**Interfaces**
Login.java	activity_login.xml
LoginAlternativo.java	activity_login_alternativo.xml
ListarBasesDeDatos.java	activity_listar_bases_de_datos.xml
ListarTablas.java	activity_listar_tablas.xml
ListarTablasMovil.java	activity_listar_tablas_movil.xml
CrearTablas.java y CrearTablasMovil.java	activity_crear_tablas.xml
MostrarTabla.java y MostrarTablaMovil.java	activity_mostrar_tablas.xml
InsertarRegistros.java y InsertarRegistrosMovil.java	activity_insertar_registros.xml
AgregarCampos.java y AgregarCamposMovil.java	activity_agregar_campos.xml
Editar.java y EditarMovil.java	activity_editar.xml
ClavesForaneas.java	activity_claves_foraneas.xml
LogSQL.java y LogSQLMovil.java	activity_log_sql.xml

Tabla 3.4 Asociación con interfaces

- Otros detalles de implementación

- Las interfaces contarán con situaciones en las que se precise un indicador de carga (con nombre 'Circulo').

- En el documento AndroidManifest.xml se han introducido la línea de texto '<uses-permission android:name="android.permission.INTERNET" />' que permite el uso de conexión a internet en la aplicación y la línea 'android:configChanges="orientation|keyboardHidden|screenSize" ' en cada llamada a cada actividad la cual permite que la actividad no sea recargada al cambiar la posición del dispositivo.

- En el documento build.gradle se ha introducido la línea de texto ' compileOptions.encoding "ISO-8859-1" ' que permite el uso de la codificación de caracteres españoles.

- Desarrollo de diferentes documentos XML para estilos de botones y listas.

3.5.3 Diseño de la lógica de datos

A continuación se presentará un modelo que define los diferentes tipos de información que la aplicación ha de tratar y al mismo tempo los modos que tiene de tratar dicha información. El modelo escogido para ello es el modelo Entidad-Relación con el cual se pretende representar las entidades del sistema así como las relaciones entre estas y sus parámetros.

3.5.3.1 Modelo Entidad/Relación

El uso habitual de la aplicación se basa en la gestión de bases de datos y tablas por parte de los usuarios, es por esto que las entidades que deben ser analizadas girarán entorno a los usuarios que realizarán la gestión, tanto con las ubicadas en el dispositivo móvil como en el servidor externo, las entidades que se gestionan, es decir, privilegios, bases de datos, tablas, campos y registros y las capacidades que tienen dichos usuarios de poder tratarlas tales como crearlas, eliminarlas, modificarlas o visualizarlas.

Todas las entidades y relaciones anteriormente mencionadas junto con sus propiedades aparecen mostradas en la **Figura 3.21**

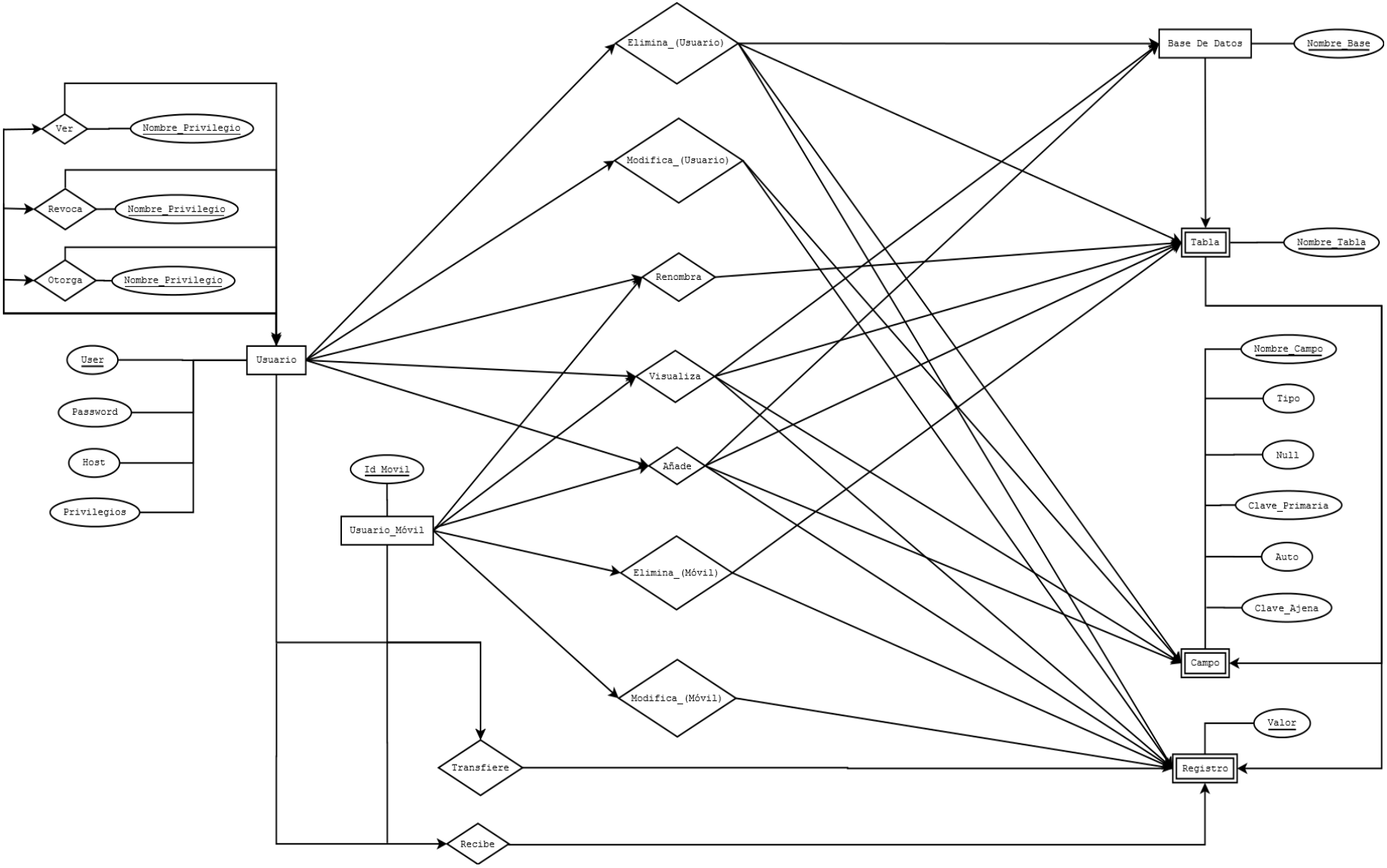


Figura 3.21 Modelo Entidad/Relación

3.5.3.2 Modelo Relacional

Entidades:

Usuario (User, Password, Host, Privilegios)

Usuario_Movil (Id_Movil)

Base De Datos (Nombre_Base)

Tabla (Nombre_Tabla)

Campo (Nombre_Campo, Tipo, Null, Clave_Primaria, Auto, Clave_Ajena)

Registro (Valor)

Relaciones:

Ver (User, Nombre_Privilegio) N:N

Revoca (User1, User2, Nombre_Privilegio) N:N

Otorga (User1, User2, Nombre_Privilegio) N:N

Elimina_(Usuario) (User, Nombre_Base, Nombre_Tabla, Nombre_Campo,Valor) N:N

Modifica_(Usuario) (User, Nombre_Campo,Valor) N:N

Visualiza (User, Nombre_Base, Nombre_Tabla, Nombre_Campo,Valor)) N:N

Renombra (User, Nombre_Tabla) N:N

Añade (User, Nombre_Base, Nombre_Tabla, Nombre_Campo,Valor)) N:N

Elimina_(Móvil) (User, Nombre_Tabla, Valor)) N:N

Modifica_(Móvil) (User, Valor) N:N

Descripción de parámetros:

Usuario			
Atributo	Descripción	Tipo de Dato	Clave Primaria
User	Nombre del usuario	Text	Sí
Password	Contraseña del usuario	Text	No
Host	Dirección IP desde donde puede acceder el usuario (por defecto todas)	Text	No
Privilegios	Privilegios MySQL del usuario	Text	No

Usuario_Móvil			
Atributo	Descripción	Tipo de Dato	Clave Primaria
Id_Móvil	Identificador del usuario móvil	Text	Sí

Base De Datos			
Atributo	Descripción	Tipo de Dato	Clave Primaria
Nombre_Base	Nombre de la base de datos	Text	Sí

Tabla			
Atributo	Descripción	Tipo de Dato	Clave Primaria
Nombre_Tabla	Nombre de la tabla	Text	Sí

Campo			
Atributo	Descripción	Tipo de Dato	Clave Primaria
Nombre_Campo	Nombre del campo	Text	Sí
Tipo	Tipo de dato que admite el campo	Text	No
Null	Permite o no que los registros del campo posean un valor nulo	Text	No
Clave_Primaria	Indican la propiedad de clave primaria en el campo	Text	No
Auto	Indica la propiedad de autoincrementable en el campo	Text	No
Clave_Ajena	Indican la propiedad de clave ajena en el campo	Text	No

Registro			
Atributo	Descripción	Tipo de Dato	Clave Primaria
Valor	Valor del registro	Text	Sí

Ver, Revoca y Otorga			
Atributo	Descripción	Tipo de Dato	Clave Primaria
Nombre_Privilegio	Privilegio MySQL	Text	Sí

3.5.4 Diseño de la interfaz

En el siguiente apartado se procederá a describir cada una de las interfaces con las que cuenta la aplicación móvil. Cada interfaz contiene en la cabecera el nombre de la aplicación, el icono representativo de la misma y un subtítulo dependiente de la interfaz. Cada una será descrita a continuación con un título que identifica el documento java donde se desarrolla, una descripción de las tareas que se ejecutan en cada una y un conjunto de capturas representativas realizada durante el funcionamiento de la aplicación.

3.5.4.1 Inicio de la aplicación: Interfaz de Login.java.

Esta interfaz será mostrada al abrir la aplicación. En ella se destacan tres tareas principales:

1. Cambio de actividad: Desde las opciones situadas en la barra de acción el usuario puede trasladarse a la actividad *LoginAlternativo.java* pulsando en la opción 'Servidor Alternativo' o bien a *ListarTablasMovil.java* pulsando 'Servidor Móvil'.

2. Inicio de sesión: Tras introducir sus credenciales de usuario y contraseña en los campos de texto el usuario podrá pulsar el botón 'Iniciar Sesión' para identificarse ante el servidor con dirección www.servidormysql.no-ip.org. Si la opción de recordar está activada y el usuario logra identificarse exitosamente con el servidor sus credenciales de usuario y contraseña permanecerán ya escritas directamente en los campos de texto. Si el usuario es correctamente autenticado la aplicación avanzará a la actividad *ListarBasesDeDatos.java*

3. Registro de usuario: Al pulsar sobre el botón 'No Tengo Cuenta, Registrarme Ahora', la aplicación abrirá el navegador automáticamente con un formulario *FormularioRegistro.jsp* el cual se describirá a continuación.

Una imagen representativa de la interfaz será mostrada en la **Figura 3.22** a continuación.

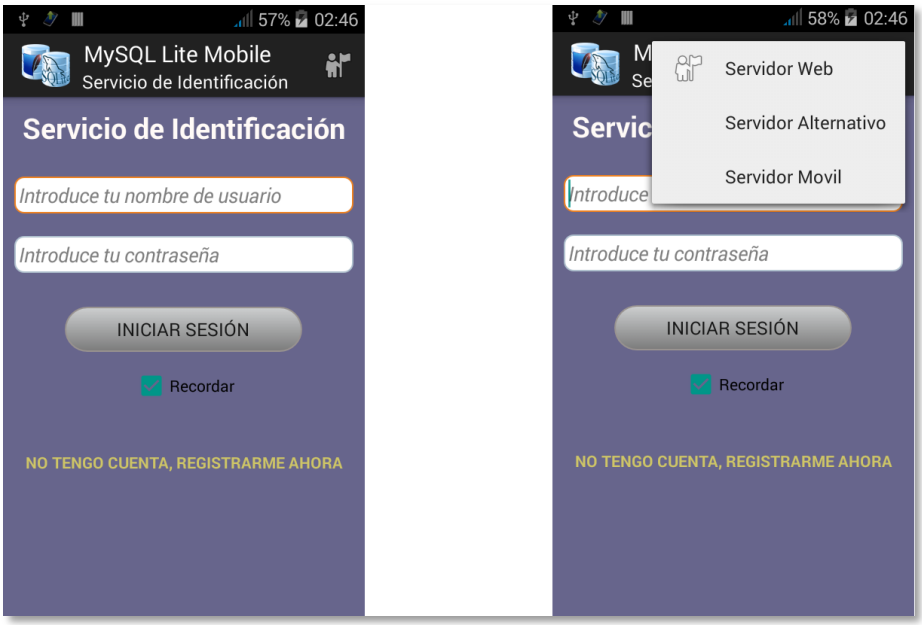


Figura 3.22 Interfaz de *Login.java*

3.5.4.2 Interfaz de FormularioRegistro.jsp

Desde esta interfaz el usuario podrá introducir sus credenciales de usuario y contraseña y un nombre para una base de datos. Tras pulsar el botón 'Registrarme' el usuario será efectivamente registrado en el servidor y una base de datos será creada bajo su propiedad con el nombre anteriormente introducido. Dicho formulario puede apreciarse en la **Figura 3.23**.

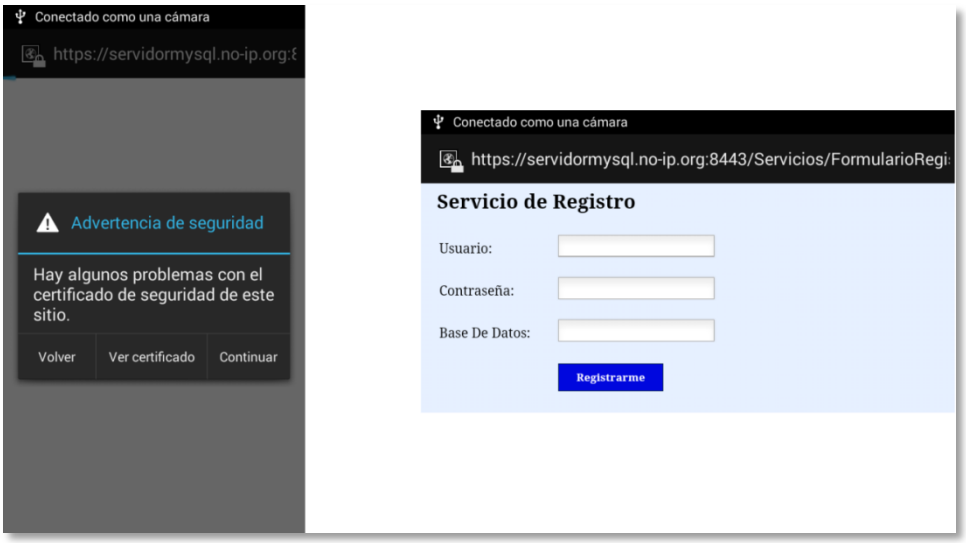


Figura 3.23 Interfaz de *FormularioRegistro.jsp*

Nota: Como puede apreciarse en la **Figura 3.23** el navegador mostrará al principio un advertencia de seguridad, esto es debido a que el certificado utilizado para conseguir un servicio de confidencialidad, servicio por el cual se consigue que la información intercambiada entre el usuario y el servidor sea cifrada, no está firmado por ninguna entidad de certificación oficial.

3.5.4.3 Interfaz de LoginAlternativo.java

La presente interfaz dispondrá al usuario de las funcionalidades suficientes para poder identificarse frente a un servidor MySQL que este elija.

En ella se destacan dos tareas principales:

1. Cambio de actividad: Desde las opciones situadas en la barra de acción el usuario puede trasladarse a la actividad *Login.java* pulsando en la opción 'Servidor Web' o bien a *ListarTablasMovil.java* pulsando 'Servidor Móvil'.

2. Inicio de sesión: La ya mencionada anteriormente que permite al usuario identificarse ante un servidor MySQL. Para ello el usuario deberá introducir en los campos de texto indicados sus credenciales de usuario y contraseña y la dirección IP del servidor objetivo. Si la opción de recordar está activada y el usuario logra identificarse exitosamente con el servidor sus credenciales de usuario y contraseña permanecerán ya escritas directamente en los campos de texto. Si el usuario es correctamente autenticado la aplicación avanzará a la actividad *ListarBasesDeDatos.java*

Una imagen representativa de la interfaz será mostrada en la **Figura 3.24** la cual aparece a continuación.

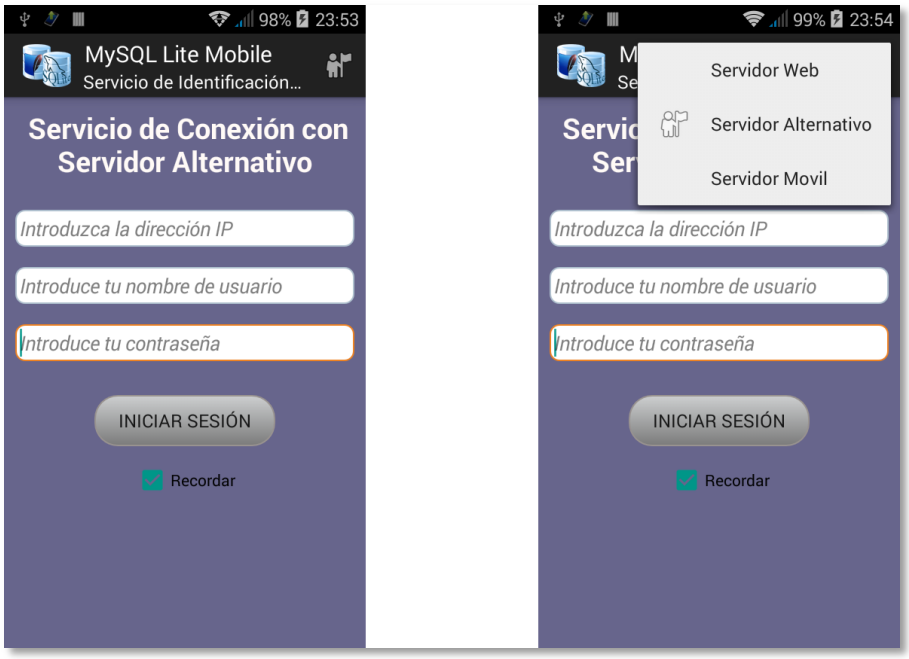


Figura 3.24 Interfaz de *LoginAlternativo.java*

3.5.4.4 Interfaz de ListarBasesDeDatos.java

La presente interfaz dispondrá al usuario de diferentes funcionalidades relacionadas con el tratamiento de las bases de datos ubicadas en el servidor MySQL en el que el usuario se haya identificado. También proporcionará diferentes funciones relacionadas con la capacidad de compartir y revocar privilegios sobre las tablas a las que el usuario tenga acceso.

La interfaz, que puede ser visualizada en la **Figura 3.25**, muestra una lista de las bases de datos a las que el usuario tiene acceso y un pequeño menú que aportará diferentes opciones. El usuario a través de esta interfaz podrá desarrollar una cantidad considerable de tareas las cuales se muestran a continuación.

1. Visualización de bases de datos: La interfaz muestra una lista con las bases de datos a las que tiene acceso el usuario identificado.

2. Creación de bases de datos: Tras pulsar el botón 'Añadir' el usuario podrá crear nuevas bases de datos en el servidor. Según la actividad anterior procedente se distinguirán dos tipos de situaciones, es decir, si la aplicación anteriormente mostraba la actividad *Login.java* al pulsar el botón 'Añadir' se abrirá el navegador automáticamente con un formulario con nombre *FormularioBasesDeDatos.jsp,* no obstante si la actividad anterior era la conocida como *ServidorAlternativo.java* se mostrará una pequeña ventana para que el usuario introduzca el nombre de la nueva base de datos.

3. Borrado de bases de datos: Después de que el usuario seleccione una base de datos de la lista y de que pulse el botón 'Borrar', aparecerá una ventana para confirma la eliminación completa de la base de datos seleccionada.

4. Visualización de log de sentencias MySQL: Tras pulsar sobre el botón 'Permisos', la opción 'Log Comandos SQL' puede ser elegida, acción que permite a la aplicación trasladarse a la actividad *LogSQL.java*.

5. Visualización de privilegios: Tras pulsar sobre el botón 'Permisos' el usuario accederá a las opciones relacionadas con la disposición de privilegios MySQL en el servidor. La opción 'Ver Privilegios' posibilitará que la aplicación se traslada a la actividad *MostrarTabla.java*desde donde el usuario podrá visualizar todos los privilegios que tiene en el servidor MySQL.

6. Compartir y revocar privilegios: Con la opción de 'Permisos' ya pulsada aparecerá la opción de 'Compartir Privilegios' la cual, después de ser pulsada, mostrará un menú con diferentes opciones entre las que se encuentran un campo de texto para indicar el usuario al que va dirigida la acción, un seleccionador de bases de datos y otro de las tablas pertenecientes a la base de datos seleccionada, un conjunto de seleccionables de privilegios y cinco botones cuyas funciones se expondrán ahora. Un botón 'Salir' permite cerrar el menú, el botón 'Otorgar' permite otorgar los privilegios señalados al usuario indicado de la tabla seleccionada pertenecientes a la base de datos introducida, mientras que el botón 'Otorgar Todos' permitirá otorgar todos los privilegios MySQL posibles, de forma opuesta los botones 'Revocar' y 'Revocar Todos' permiten revocar los privilegios indicados, o bien, todos los privilegios.

7. Visualizar tablas de una base de datos: Tras seleccionar una base de datos y pulsar el botón 'Cargar' la aplicación se trasladará a la actividad *ListarTablas.java.*

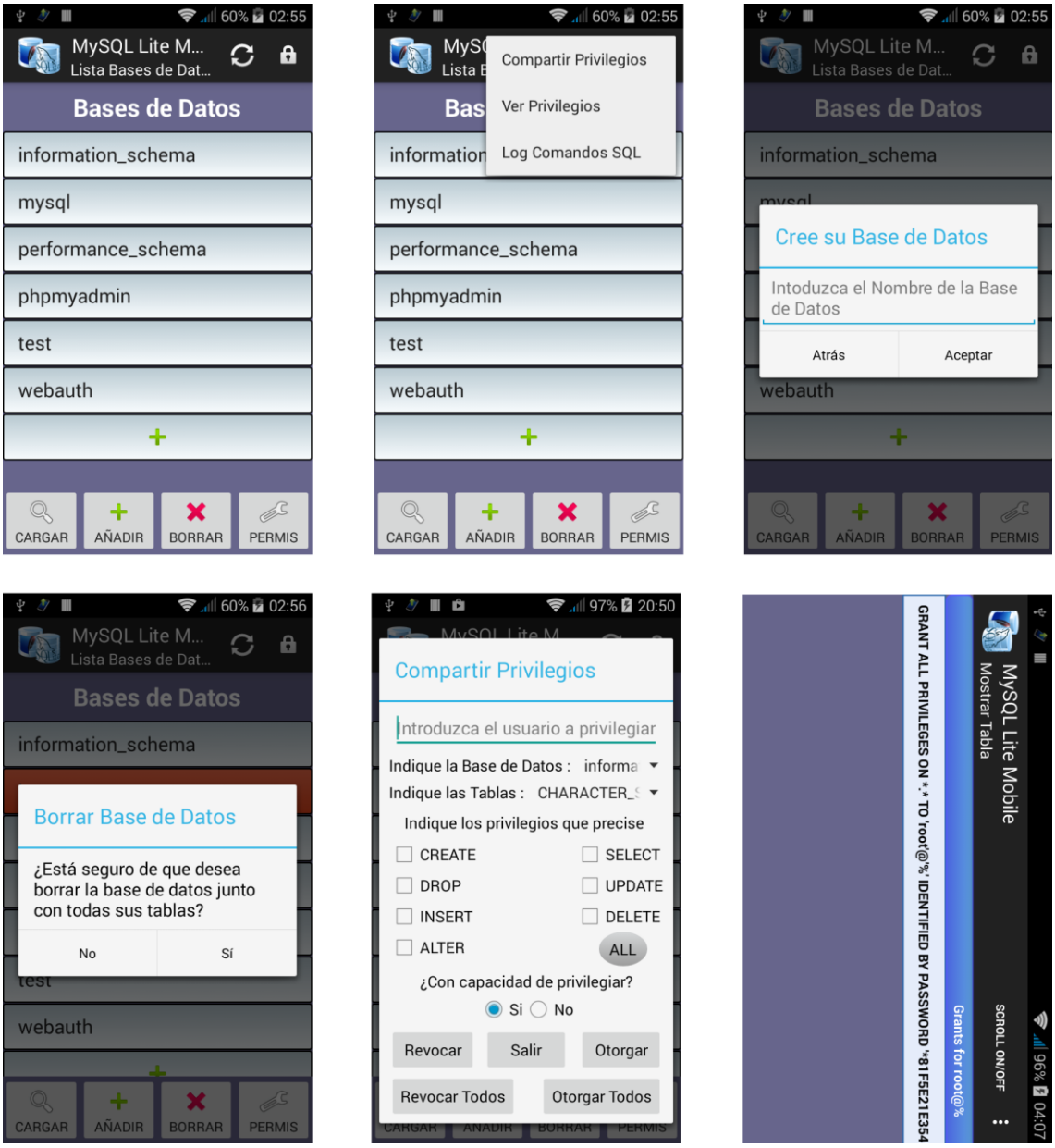


Figura 3.25 Interfaz de *ListarBasesDeDatos.java*

3.5.4.5 Interfaz de BasesDeDatos.jsp

El formulario mostrado en la **Figura 3.26** permite al usuario la creación de bases de datos en el servidor con dirección www.servidormysql.no-ip.org. Los campos que forman el formulario son las credenciales de usuario y contraseña, necesarias para identificar al poseedor de la nueva base de datos, y el nombre que tendrá la base de datos a crear. Finalmente aparece un botón 'Crear Base' cuya función es la de enviar la información anteriormente mencionada al servidor para que finalmente la base de datos pueda ser creada.

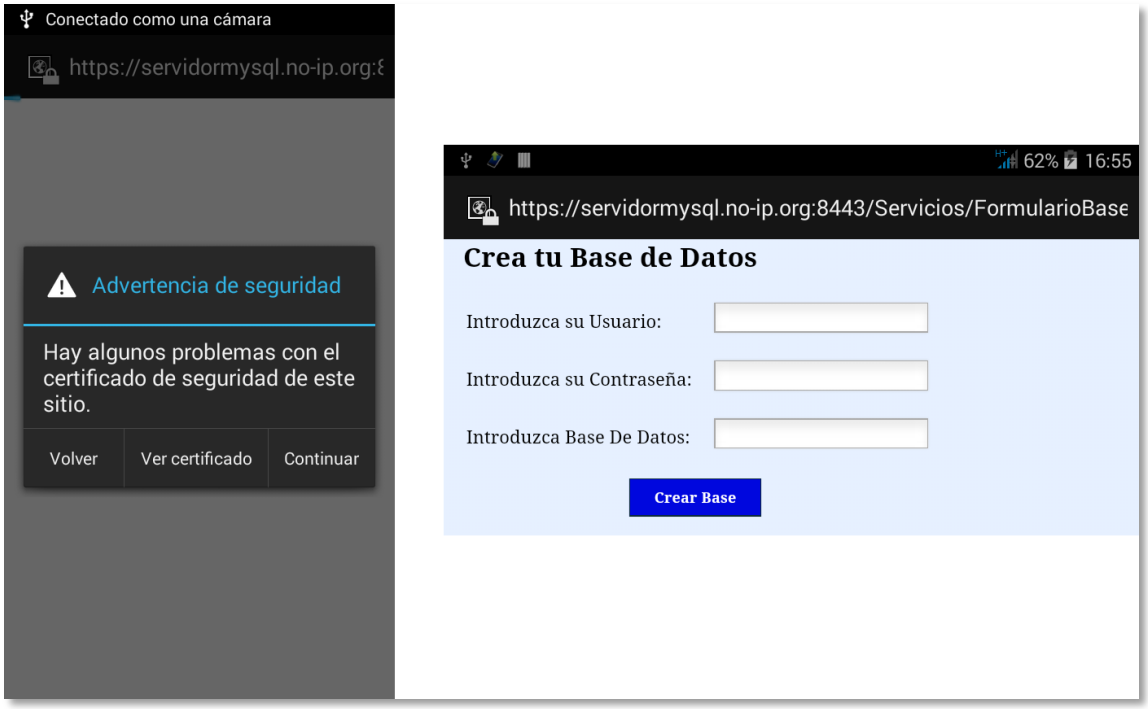


Figura 3.26 Interfaz de *BasesDeDatos.jsp*

Nota: Como puede apreciarse en la **Figura 3.26** el navegador mostrará al principio un advertencia de seguridad, esto es debido a que el certificado utilizado para conseguir un servicio de confidencialidad, servicio por el cual se consigue que la información intercambiada entre el usuario y el servidor sea cifrada, no está firmado por ninguna entidad de certificación oficial.

3.5.4.6 Interfaz de ListarTablas.java y ListarTablasMovil.java

Como el título mismo sugiere ambas actividades contarán con casi la misma interfaz visual, aunque contarán con funcionalidades un tanto diferentes que serán explicadas a continuación. Desde la actividad *ListarTablas.java* se dispondrá al usuario de diferentes funcionalidades relacionadas con el tratamiento de tablas y registros pertenecientes a una base de datos ubicadas en un servidor MySQL en el que el usuario se haya identificado previamente, mientras que en el caso de la actividad *ListarTablasMovil.java* las tablas a las que se aplicarán las funcionalidades serán las contenidas en una base de datos ubicadas en el mismo dispositivo móvil.

La interfaz, que puede ser visualizada en las **Figura 3.27** y **3.28**, muestra una lista de las tablas pertenecientes a una base de datos a las que el usuario tiene acceso y un pequeño menú que aportará diferentes opciones. El usuario a través de esta interfaz podrá desarrollar una cantidad considerable de tareas las cuales se muestran a continuación.

1. Visualización de tablas: La interfaz muestra una lista con las tablas pertenecientes a una base de datos anteriormente seleccionada por el usuario.

2. Creación de tablas: Tras pulsar el botón 'Añadir' la aplicación se desplazará a la actividad *CrearTablas.java* o bien a *CrearTablasMovil.java.*

3. Borrado de tablas: Después de que el usuario seleccione una tabla de la lista y de que pulse el botón 'Borrar' aparecerá una ventana para confirma la eliminación completa de la tabla seleccionada.

4. Visualizar registros de una tabla: Tras seleccionar una tabla y pulsar el botón 'Cargar' la aplicación se trasladará a la actividad *MostrarTablas.java* o bien a *MostrarTablasMovil.java*

Tras pulsar sobre el botón 'Opciones' la aplicación mostrará un menú con diferentes opciones las cuales se discutirán a continuación.

1. Insertar registros: Después de pulsar sobre la opción 'Insertar' la aplicación se trasladará a la actividad *InsertarRegistros.java* o a *InsertarRegistrosMovil.java.*

2. Agregar Campos: Después de pulsar sobre la opción 'Agregar Campos' la aplicación se trasladará a la actividad *AgregarCampos.java* o a *AgregarCamposMovil.java.*

3. Editar campos, editar registros y editar valores: Después de pulsar sobre la opción 'Editar Campos', 'Editar Registros' o 'Editar Valores' la aplicación se trasladará a la actividad *Editar.java* o a *EditarMovil.java.*

4. Editar claves foráneas: Después de pulsar sobre la opción 'Claves Foráneas' la aplicación se trasladará a la actividad *ClavesForáneas.java*. La actividad *ListarTablasMovil.java* en este caso no cuenta con esta funcionalidad y dispondrá de un mensaje de advertencia al usuario.

5. Renombrar tabla: Después de que el usuario seleccione una tabla de la lista y de que pulse pulsa la opción 'Renombrar Tabla' aparecerá una ventana para que el usuario introduzca un nuevo nombre para la tabla seleccionada.

6. Vaciar tabla: Después de que el usuario seleccione una tabla de la lista y de que pulse pulsa la opción 'Vaciar Tabla' aparecerá una ventana para que el usuario confirme si desea que todos los registros de la tabla sean eliminados.

7. Visualización de log de sentencias MySQL o SQLite: Después de pulsar sobre la opción 'Log Comandos SQL', la aplicación se trasladará a la actividad *LogSQL.java*, en el caso de que la actividad anterior fuese *ListarTablas.java*, o a la actividad *LogSQLMovil.java* en el caso de que fuese *ListarTablasMovil.java.*

8. Traspaso de tablas: Después de que el usuario seleccione una tabla de la lista y de que pulse pulsa la opción 'Copia de Seguridad' podrán darse dos situaciones diferentes según la actividad en la que este se encuentre, es decir, si dicha opción se ejecuta desde *ListarTablas.java* la tabla junto con sus registros serán copiados al dispositivo mientras que, por otro lado, si se ejecutase dicha acción desde *ListarTablasMovil.java* un pequeño menú sería abierto para que el usuario introdujese sus credenciales de usuario y contraseña, la base de datos y la dirección IP destino del servidor donde se copiará la tabla seleccionada.

9. Copia de tabla a otro servidor: Después de que el usuario seleccione una tabla de la lista y de que pulse pulsa la opción 'Copia a Servidor' un pequeño menú sería abierto para que el usuario introduzca sus credenciales de usuario y contraseña, la base de datos y la dirección IP destino del servidor donde se copiará la tabla seleccionada.

10. Cambio de actividad: Esta posibilidad solo se contempla en la actividad *ListarTablasMovil.java*. Desde las opciones situadas en la barra de acción el usuario puede trasladarse a la actividad *Login.java* pulsando en la opción 'Servidor Web' o bien a *LoginAlternativo.java* pulsando 'Servidor Alternativo'. En la **Figura 3.27** puede apreciarse dicha alternativa.

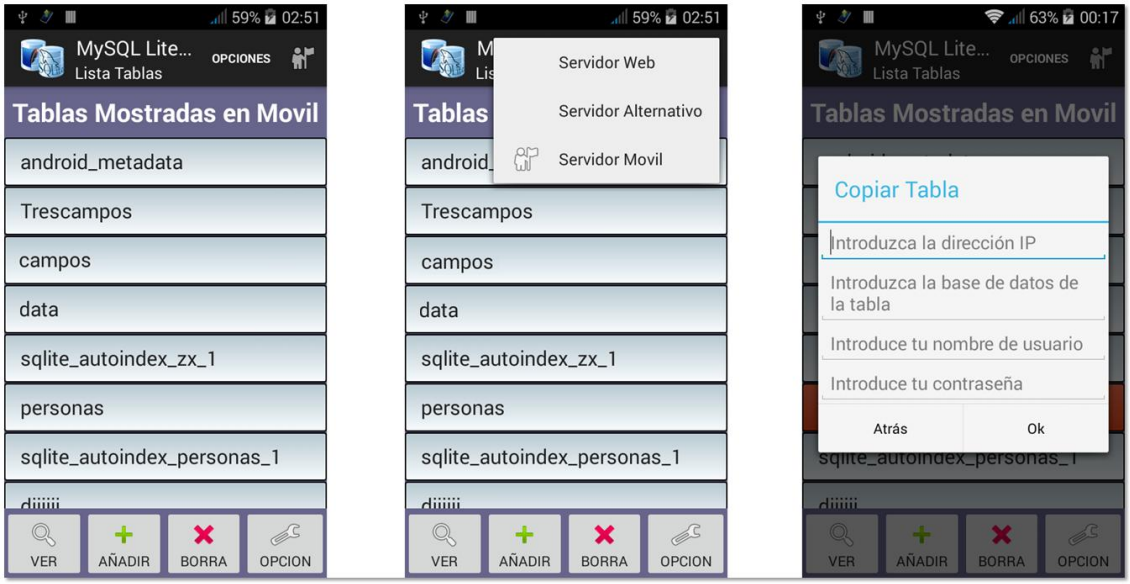


Figura 3.27 Interfaz de *ListarTablasMovil.java*

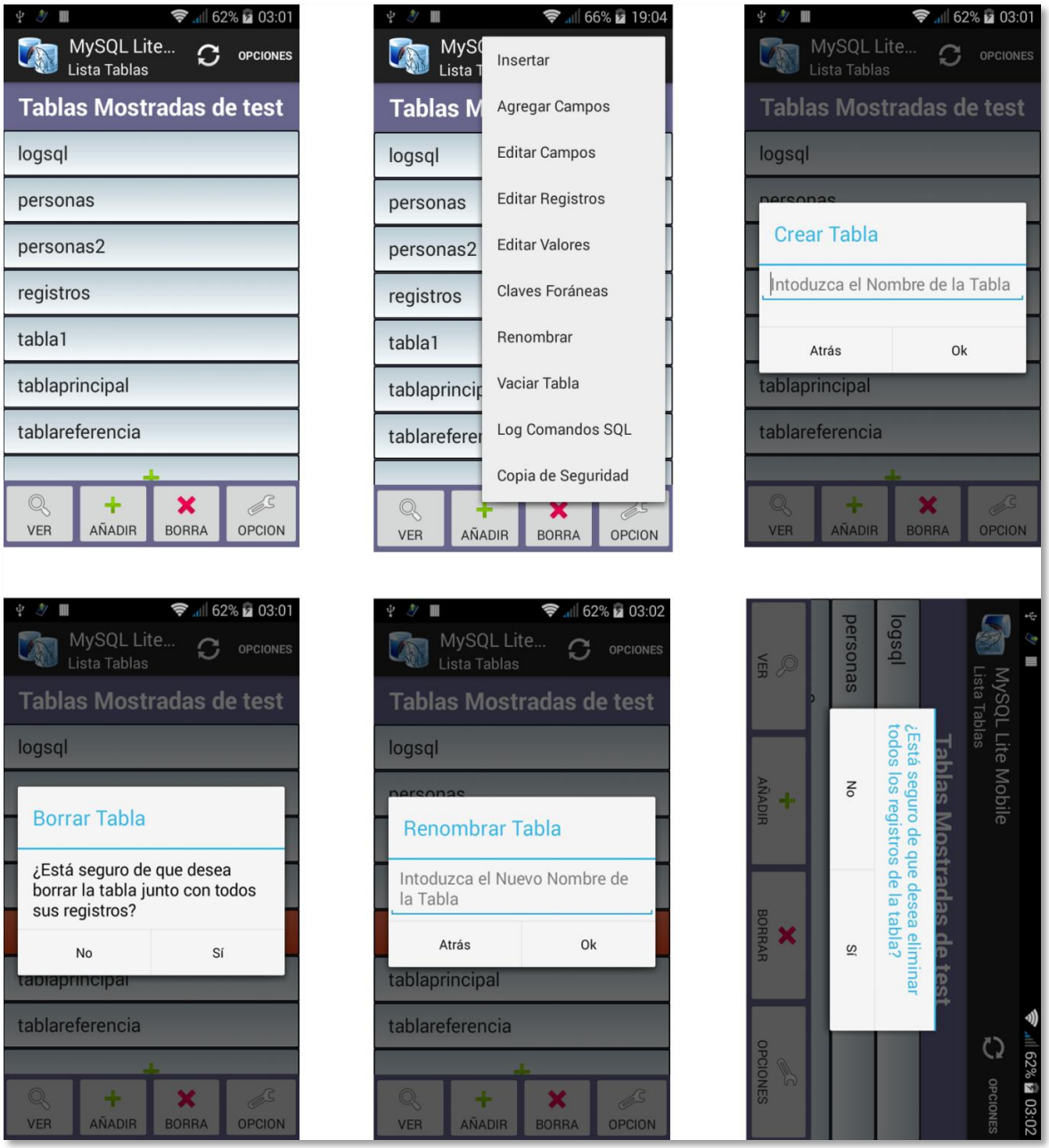


Figura 3.28 Interfaz de *ListarTablas.java* y *ListarTablasMovil.java*

3.5.4.7 Interfaz de LogSQL.java y LogSQLMovil.java

La presente interfaz contiene una funcionalidad muy sencilla, mostrar al usuario las sentencias MySQL enviadas al servidor o bien las sentencias SQLite ejecutadas en el dispositivo por la aplicación. Ambas interfaces pueden apreciarse en la **Figura 3.29**.

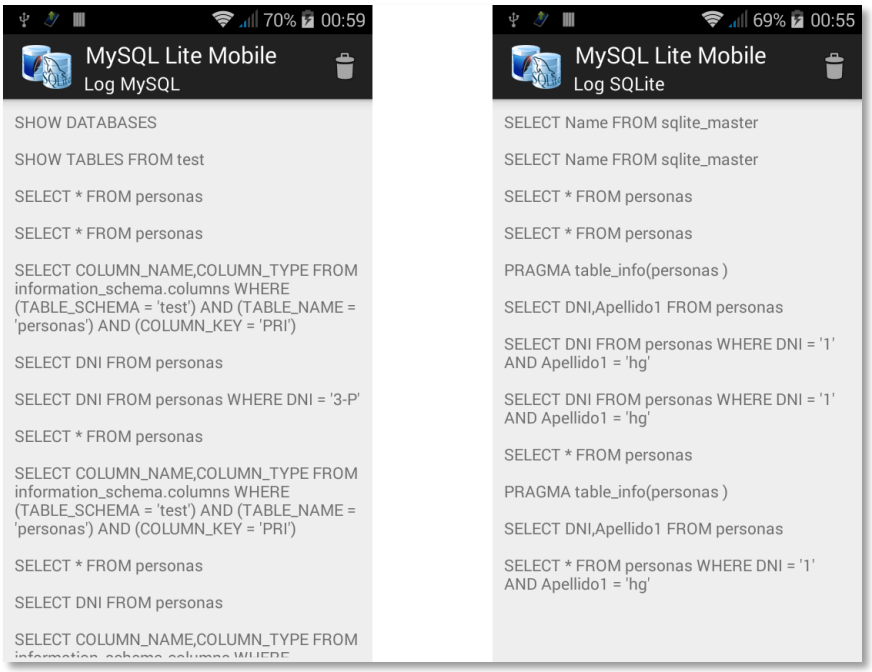


Figura 3.29 Interfaz de *LogSQL.java y LogSQLMovil.java*

3.5.4.8 Interfaz de CrearTablas.java y CrearTablasMovil.java

Esta interfaz ofrece al usuario un menú diseñado para la creación de tablas de acuerdo a unos parámetros los cuales se estudiarán a continuación. Dichos parámetros diferirán en pequeña medida tal y como se aprecia en las **Figuras 3.30 y 3.31** entre las actividades mencionadas en las mismas.

1. Número de campos: Aquí el usuario introducirá el número de campos que desee que tenga la tabla a crear, tras ello deberá pulsar el botón 'Antro'.

2. Campo: El nombre que tendrá el campo que se está editando.

3. Tipo: El tipo de dato que almacenará el campo. Aunque en MySQL y SQLite existe una gran cantidad de tipos la aplicación permite utilizar los más importantes, es decir, Char, Varchar y Text para introducir un texto cualquiera, Int y Float ambos formatos numéricos, el primero para enteros y el segundo para decimales, Date para introducir fechas con formato yyyy-mm-dd por defecto y Time para introducir tiempo con formato hh:mm:ss por defecto.

4. Long: Longitud máxima de caracteres del valor introducido en el campo.

5. Null: Permite o no registros con valor nulo.

6. P.Key: Permite especificar si el campo se considerará parte de la clave primaria de la tabla.

7. Auto: Permite establecer en el campo la propiedad de ser o no autoincrementable. Solo está disponible para *CrearTablas.java.*

8. F.Key, Tabla Ref. y Campo Ref: Solo disponibles en *CrearTablasMovil.java.* Permiten definir establecer la propiedad de clave foránea en los campos de la tabla a crear. El primero permite o no dicha posibilidad, el segundo indica la tabla de referencia y el tercer indica el campo de esa tabla que será de referencia para el campo que se está editando.

Una vez que todos los parámetros hayan sido especificados por el usuario este pulsará sobre el botón 'Crear' para que la tabla sea finalmente creada.

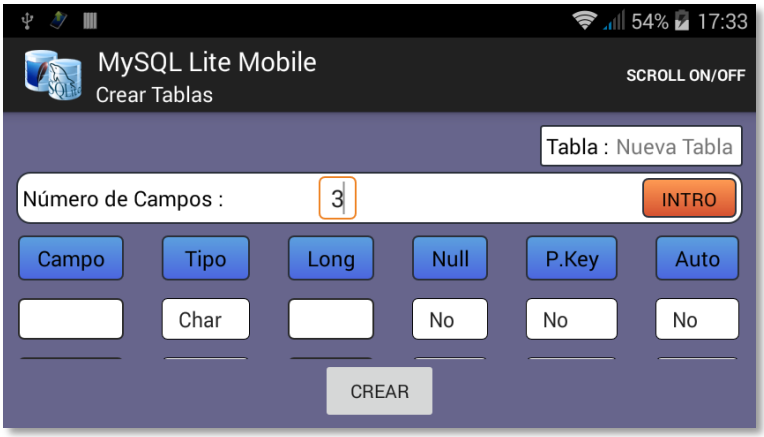


Figura 3.30 Interfaz de *CrearTablas.java*

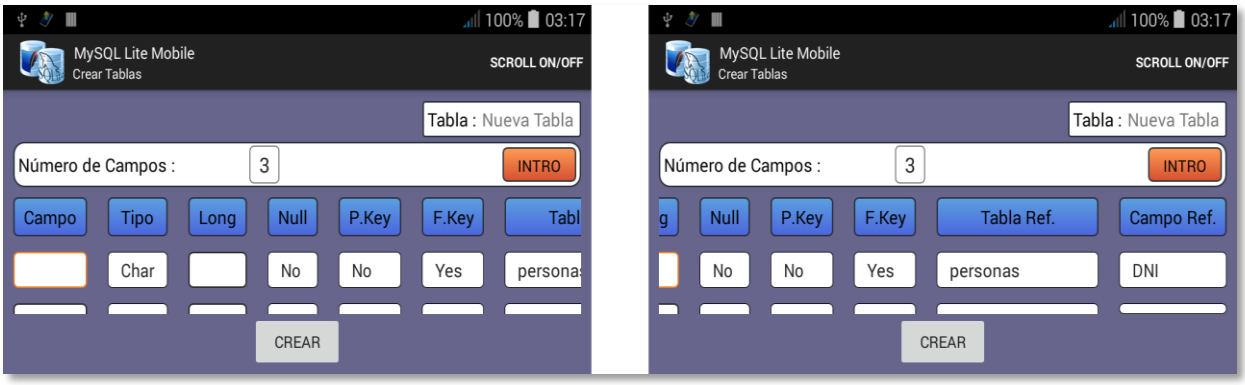


Figura 3.31 Interfaz de *CrearTablasMovil.java*

La opción 'Scroll ON/OFF' que aparece por primera vez en esta actividad y que seguirá apareciendo en algunas interfaces venideras permite activar un scroll o desplazable horizontal en los elementos que sean necesarios para su correcta visualización en pantalla.

3.5.4.9 Interfaz de MostrarTabla.java y MostrarTablaMovil.java

Como bien sugiere el título esta interfaz tiene como función principal la de mostrar todos los campos y registros pertenecientes a una tabla escogida previamente por el usuario, además de tres funcionalidades extras ubicadas en la barra de acción las cuales se explican a continuación y que pueden apreciarse en la **Figura 3.32**

- La primera de ellas permite esconder la barra de acción para poder aprovechar al máximo el tamaño de la pantalla y poder visualizar así mejor tablas que tengan una cantidad considerable de registros.

- La segunda permite ordenar los registros de la tabla según unos parámetros especificados por el usuario, es decir, el campo por el que ordenar los registros y el orden que podrá ser ascendente o descendente. Tras introducir dichos parámetros el usuario deberá pulsar sobre el botón 'Mostrar'.

- La tercera permite obtener registros de la tabla según un campo y el valor de ese campo introducidos. Tras introducir dichos parámetros el usuario deberá pulsar sobre el botón 'Mostrar'.

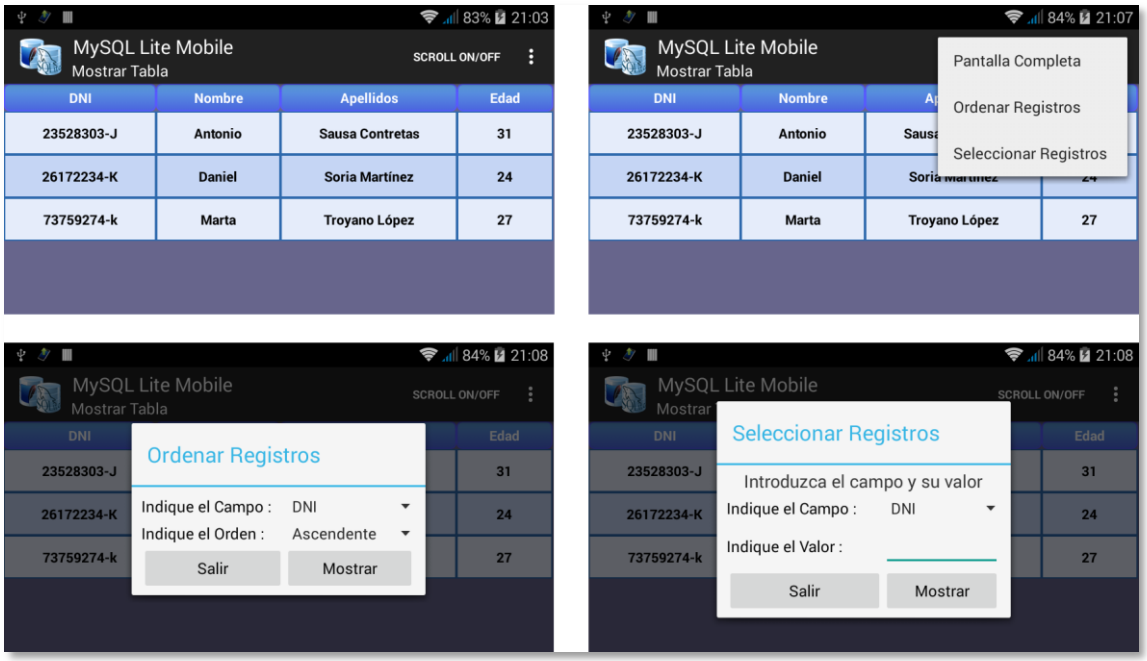


Figura 3.32 Interfaz de *MostrarTabla.java y MostrarTablaMovil.java*

3.5.4.10 Interfaz de InsertarRegistros.java e InsertarRegistrosMovil.java

Esta interfaz provee al usuario de la capacidad de insertar nuevos registros a una tabla de una forma sencilla e intuitiva a través de un menú adecuado para ello. Dicho menú contendrá los siguientes elementos:

1. Elemento Campos: Indica al usuario los nombres de los campos sobre los que insertar los registros y si el campo es o no clave primaria de la tabla mediante la muestra de un icono de una llave.

2. Elemento Tipo: Indica el tipo de información que puede insertarse y la longitud máxima del registro a introducir.

3. Elementos Tipo y Null: Indican, respectivamente, si el campo puede contener un valor nulo y si tiene la propiedad de ser autoincrementable.

4. Elemento Valores: Indican al usuario unos campos de texto donde deberá insertar los valores que querrá introducir a la tabla.

Tras introducir los valores insertados el usuario deberá pulsar el botón 'Registrar' para que se produzca tal acción. El botón 'Limpiar' tiene como función dejar en blanco los campos de texto para la inserción de registros. Esta interfaz puede verse representada en la **Figura 3.33**.

Figura 3.33 Interfaz de *InsertarRegistros.java y InsertarRegistrosMovil.java*

3.5.4.11 Interfaz de AgregarCampos.java y AgregarCamposMovil.java

Esta interfaz ofrece al usuario un menú diseñado para la agregación cómoda de campos a tablas de acuerdo a unos parámetros los cuales se estudiarán a continuación. Dichos parámetros diferirán en pequeña medida tal y como se aprecia en la **Figura 3.34** entre las actividades mencionadas en las mismas.

1. Número de Campos: Indica la cantidad de campos que se agregarán a la tabla. El usuario tras introducir dicho cantidad deberá pulsar sobre el botón 'Intro'.

2. Posición de los campos: Indica el lugar donde se introducirán los campos respecto a los que ya existían, es decir, al final de la tabla, al principio o después de un campo especificado. En el caso de la actividad *AgregarCamposMovil.java* solo podrá situarse al final.

3. Parámetros del campo: Anteriormente mencionados indican parámetros relativos al campo, es decir, el nombre del campo, el tipo y la longitud máxima de los valores asociados a ese campo y la posibilidad o no de introducir valores nulos en ese campo.

Figura 3.34 Interfaz de *AgregarCampos.java* y *AgregarCamposMovil.java*

3.5.4.12 Interfaz de Editar.java e EditarMovil.java

Desde esta interfaz la aplicación aportará tres tipos de servicios diferenciados que son la edición (borrado y modificación) de los campos, registros y valores de una tabla. Dichos servicios se explicarán con más detalle a continuación. La interfaz contará con diferentes elementos gráficos para ofrecer estos servicios los cuales son un grupo de botones, para elegir el servicio de edición que se precise, dos seleccionadores, uno de campos de la tabla a editar y otro con el valor de los campos que forman la clave primaria de la tabla y un menú que será diferente según el servicio seleccionado.

1. Edición de campos: Descrito en las **Figuras 3.35 y 3.36** y solo disponible para *Editar.java*. Después de que el usuario seleccione un campo la aplicación permitirá la eliminación del elegido pulsando sobre el botón 'Borrar' o bien, la modificación de parámetros de estos mediante los elementos Campo, Tipo, Long, Null y Auto cuyas funcionalidades han sido ya explicadas anteriormente con el uso del botón 'Modificar'. El botón 'Pri.key' pulsado mostrará un menú para la edición de claves primarias.

Figura 3.35 Interfaz de *Editar.java* (Edición de campos)

Figura 3.36 Interfaz de *EditarMovil.java* (Edición de campos)

2. Edición de claves primarias: Descrito en la **Figura 3.37** y solo disponible para *Editar.java*. Después de que el usuario seleccione un campo la aplicación permitirá la eliminación de la propiedad de clave primaria en el campo elegido pulsando sobre el nuevo botón 'Borrar' con fondo anaranjado, o bien, la agregación del campo como parte de la clave primaria de la tabla pulsando sobre el botón 'Añadir'.

Figura 3.37 Interfaz de *Editar.java* (Edición de claves primarias)

3. Edición de registros: Descrito en la **Figura 3.38.** Después de que el usuario seleccione uno o varios valores que formen parte de la clave primaria de la tabla la aplicación mostrará un registro cuyos valores podrán ser modificados mediante el uso de campos de texto y del botón 'Modificar', o bien dicho registro puede ser eliminada gracias al uso del botón 'Borrar'.

Figura 3.38 Interfaz de *Editar.java* y *EditarMovil.java* (Edición de registros)

4. Edición de valores: Descrito en la **Figura 3.39.** Después de que el usuario seleccione un campo de la tabla y uno o varios valores que formen parte de la clave primaria de la tabla, un único valor le será mostrado al usuario, el cual podrá ser modificado gracias al uso de un campo de texto y del botón 'Modificar', o bien, eliminado pulsando el botón 'Borrar'.

Figura 3.39 Interfaz de *Editar.java* y *EditarMovil.java* (Edición de valor)

3.5.4.13 Interfaz de ClavesForaneas.java

Esta interfaz tiene como propósito ofrecer un conjunto de elementos suficientes al usuario como para permitir la capacidad de asignar propiedades de clave foránea a los campos de una tabla o bien de eliminar dicha propiedad. Una imagen representativa de esta interfaz aparece representada en la **Figura 3.40.** A continuación se estudiarán cada uno de los elementos que conforman la interfaz divididos en los tres submenús diferentes que aparecen en la misma.

1. Menú selección: El usuario seleccionará aquí la tabla y el campo de esa tabla al que se le aplicará la propiedad de clave foránea y la tabla y el campo de esa tabla que serán de referencia o principal en la propiedad de clave foránea.

2. Menú opciones: El usuario seleccionará aquí el comportamiento que tendrá la tabla con la clave foránea al ser eliminado o modificado un registro en la tabla principal. Una vez seleccionados los elementos anteriormente mencionados el usuario pulsará el botón 'Añadir' para aplicar la propiedad de clave foránea.

3. Menú borrar: El usuario seleccionará aquí la tabla y el campo de esa tabla que contiene una o varias propiedades de clave foránea, después seleccionará la 'restricción' que actuará como identificador de las claves foráneas aplicadas a la tabla y finalmente pulsara el botón 'Borrar' para eliminar la propiedad de clave foránea indicada si así lo desea.

Figura 3.40 Interfaz de *ClavesForaneas.java*

4. RESULTADOS Y DISCUSIÓN

En este apartado se procederá a realizarse una simulación de algunas de las funcionalidades básicas con las que el programa cuenta, tras esto, se realizará una batería de pruebas para desmostrar que la aplicación responde correctamente ante cualquier tipo de error. Una vez comprobado el correcto funcionamiento de la aplicación se les pedirá su opinión acerca de diferentes aspectos de la misma a diferentes usuarios con tal de comprobar si esta es atractiva y entendible para diferentes públicos.

4.1 Simulación de la aplicación

A continuación se llevará a cabo una simulación del uso normal de la aplicación en la que se incluirán las funcionalidades más habituales en el uso de la misma. Dicha simulación se realizará a través de un terminal móvil ZTE Blade L2, con procesador quad-core 1.3GHz, 4GB memoria interna, 1GB de RAM, con un tamaño de pantalla 480 x 854 pixels, 5.0 pulgadas y con la versión de Android 4.4 Kitkat. La conexión con el servidor MySQL será utilizando la misma red local, por lo que el dispositivo móvil estará conectado al mismo router que el servidor mediante una conexión WiFi.

1. Login y listar bases de datos del servidor

En primer lugar el usuario deberá identificarse frente al servidor, para ello bastará con que introduzca la dirección IP del servidor y sus credenciales de usuario y contraseña y que pulse sobre el botón de 'Iniciar Sesión' que aparece en la **Figura 4.1,** tras esto, y si los datos introducidos con correctos se mostrarán las bases de datos accesibles al usuario ubicadas en el servidor.

Figura 4.1 Login lista de bases de datos

2. Crear y Borrar bases de datos

Si el usuario quisiera crear una base de datos deberá pulsar sobre el botón 'Añadir' e introducir el nombre que tendrá la misma, en este caso se creará la base de datos con nombre 'prueba'. Finalmente si tiene los permisos necesarios dicha acción podrá ser resuelta, en cambio si desease borrar una base de datos ya existente deberá seleccionar una de la lista y pulsar sobre el botón 'Borrar' y tras esto deberá pulsar 'Sí' en la ventana emergente. En este caso se eliminará la base de datos creada anteriormente. Ambas acciones pueden contemplarse en la **Figura 4.2.**

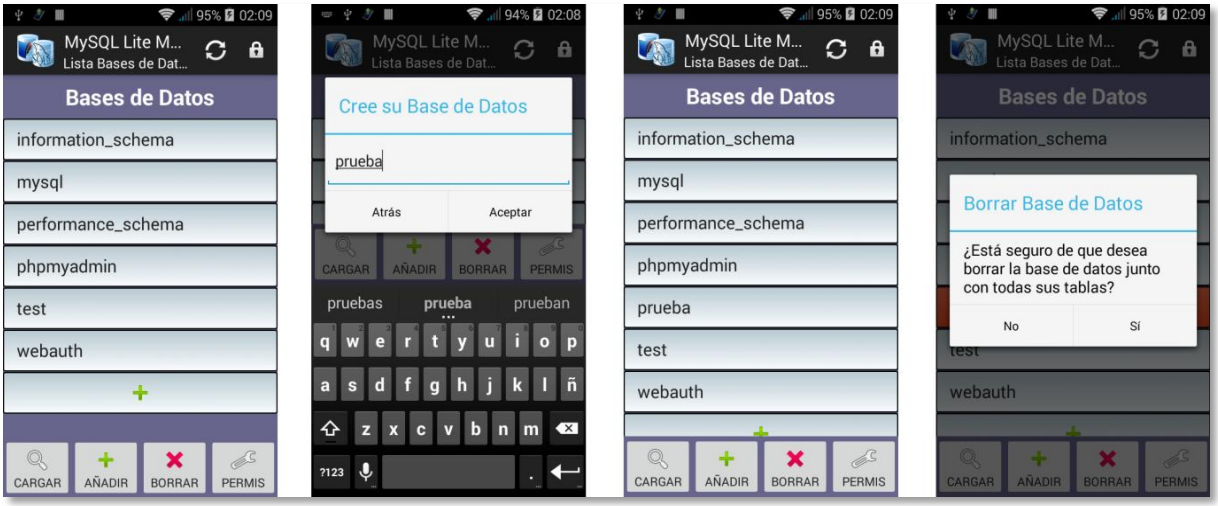


Figura 4.2 Creación y eliminación de bases de datos

3. Listar tablas de una base de datos, renombrar y eliminar una de ellas

Cuando el usuario presione el botón 'Cargar' mostrado en la **Figura 4.2** tras haber seleccionado una base de datos de la lista, la aplicación mostrará una lista con las tablas pertenecientes a la base de datos seleccionada. Dicha lista irá acompañada con un menú que ofrecerá al usuario diferentes opciones que podrían ser ampliadas si este pulsase sobre el botón 'Opciones'. Esto puede apreciarse en la **Figura 4.3.**

Figura 4.3 Lista de tablas

Entre las opciones que nos aporta la aplicación se encuentra la de borrar una tabla de la lista, para ello el usuario deberá seleccionar una, pulsar sobre el botón 'Borrar' y después pulsar en la ventana emergente sobre 'Sí'. De manera similar una tabla puede ser renombrada si se pulsa la opción 'Renombrar' y se introduce el nuevo nombre que desea que contenga la tabla. A continuación aparecerá en la **Figura 4.4** como la tabla con nombre 'tabla1' puede ser renombrada a 'tablarenombrada' y después eliminada.

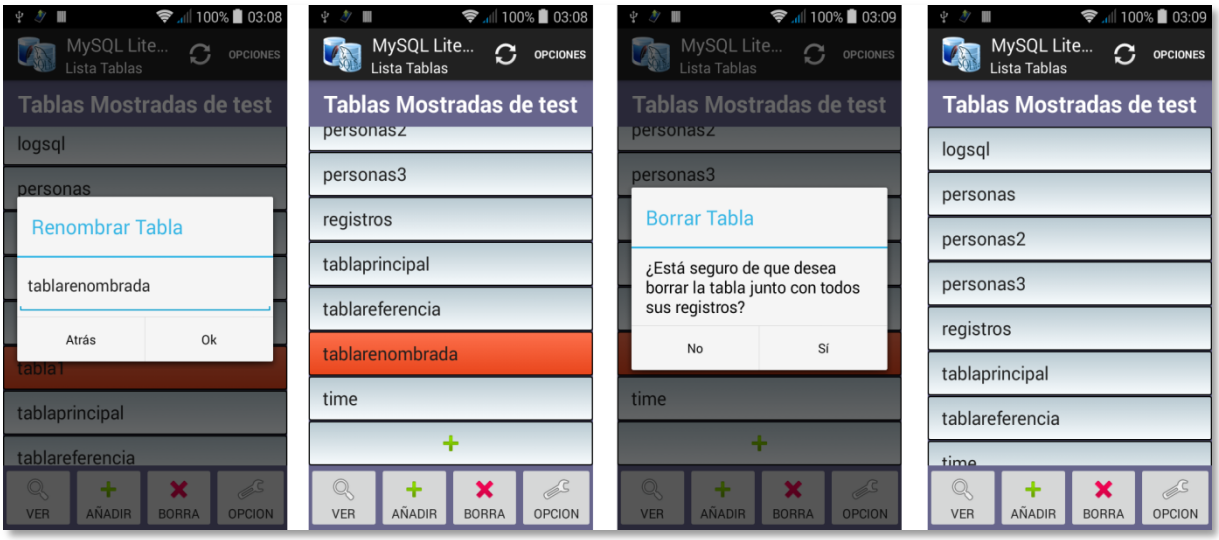


Figura 4.4 Renombrado y eliminación de tablas

4. Crear tabla y agregar nuevos campos

En este caso se va a proceder a crear una tabla denominada 'Alumnos' la cual estará formada por los campos 'DNI', 'Nombre' y 'Apellidos', tras haber creado la tabla posteriormente se le añadirá otro campo llamado 'Edad'. Los menús serán muy similares para ambas acciones, el usuario simplemente introducirá el número de campos a crear y pulsará sobre el botón 'Intro', tras esto aparecerá un menú de campos donde serán caracterizados por los parámetros que el usuario introduzca. Para proceder a estos pasos deberá pulsarse bien el botón 'Añadir' para la creacion de tablas e introducir el nombre que la tabla tendrá o bien sobre la opción 'Agregar Campos' en el otro caso. Este ejemplo puede encontrarse gráficamente en la **Figura 4.5**.

Figura 4.5 Crear tabla y agregar campo

5. Visualizar tabla e insertar registros

Cuando el usuario presione sobre el botón 'Ver' mostrado en la **Figura 4.4** tras haber seleccionado una tabla de la lista la aplicación mostrará una tabla con todos los registros pertenecientes a la escogida. Si el usuario quisiese introducir nuevos registros a la tabla deberá pulsar sobre la opción 'Insertar', entonces un menú adecuado para ello se mostrará frente a él. Ambas opciones pueden apreciarse en la **Figura 4.6**.

Figura 4.6 Insertar registros y visualizar tabla

6. Ordenar y consultar registros

Un usuario que posea tablas con grandes cantidades de registros tendrá difícil localizar alguno en particular, es por eso que la aplicación cuenta con las opciones 'Ordenar Resultados' y 'Seleccionar Registros' en la actividad que permite la visualización de estos. La primera opción permite ordenar los registros por orden alfabético o de numeración según un campo seleccionado y la segunda permite obtener los registros que concuerden con un campo y un valor para ese campo introducidos. En la **Figura 4.7** puede encontrarse un ejemplos sobre cómo puede ser alterado el orden de los registros según el campo 'Nombre' y otro sobre cómo puede ser encontrado un alumno en la tabla con el nombre 'Daniel'.

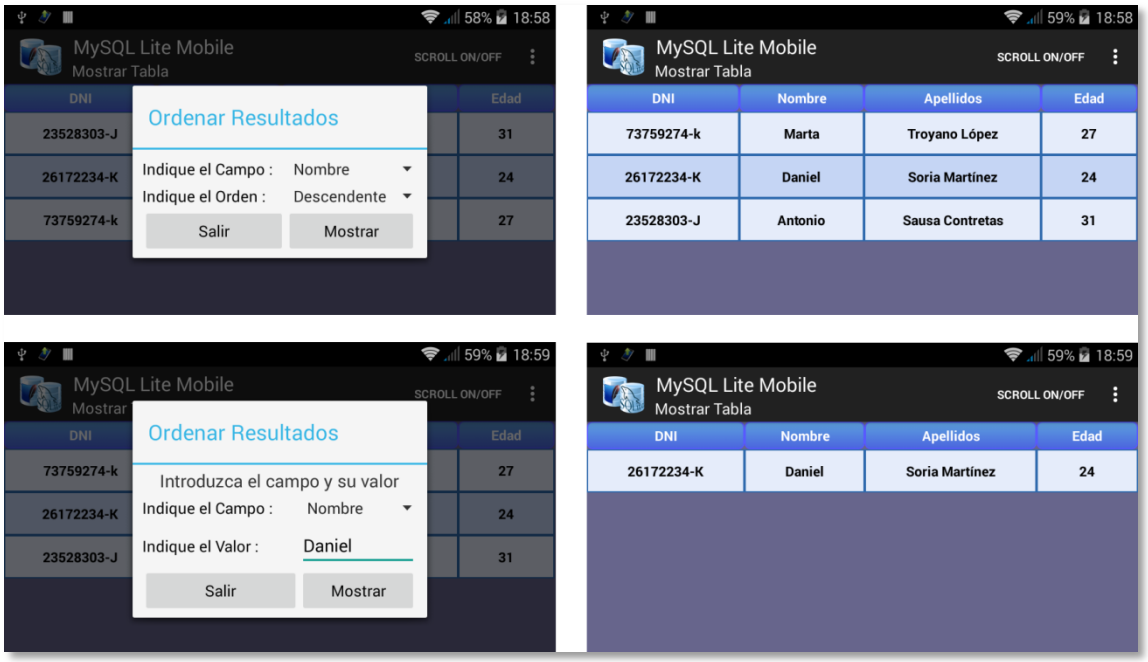


Figura 4.7 Ordenar y seleccionar registros

7. Eliminar y modificar registros

La aplicación ofrece tres formatos diferentes para la edición de registros, es decir, la eliminación completa de todos los registros de la tabla, mediante la opción 'Vaciar Tabla', la eliminación o modificación de un registro individual de la tabla, mediante la opción 'Editar Registros' o bien la eliminación o modificación de un único valor pertenciente a un registro con la opción 'Editar Valor'. En la **Figura 4.8** puede encontrarse un ejemplo sobre la eliminación de un registro en donde se eliminará el primero de la tabla y en la **Figura 4.9** sobre la modificación de un valor escogido, donde se modificará el nombre de Daniel a Juan y en donde pueden apreciarse los cambios realizados.

Figura 4.8 Eliminación de registros

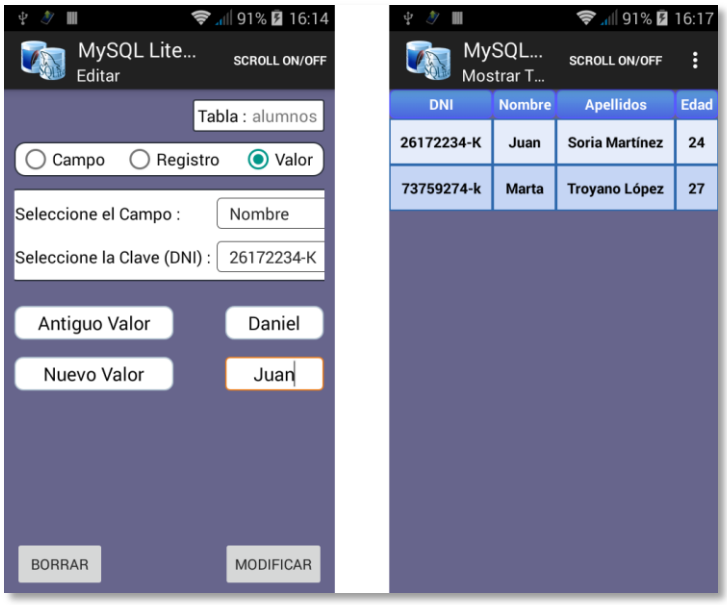


Figura 4.9 Modificación de valores

8. Eliminar y modificar campos

En este caso se mostrará un ejemplo sobre cómo puede ser modificado o eliminado el campo de una tabla, para ello deberá pulsarse sobre la opción 'Editar Campos' para acceder al menú que aparece en la **Figura 4.10**, tras esto podrá ser seleccionado el campo para ser eliminado, junto con los registros pertenecientes a su misma columna, o bien, cambiar distintos parámetros del mismo, en este caso se cambiará el nombre del campo 'Edad' por 'Años' y se eliminará el campo 'Nombre'.

Figura 4.10 Modificación y eliminación de campos

4.2 Batería de pruebas

En este apartado se procederá a realizarse una serie de situaciones en las que se intentará provocar diversos errores en la aplicación para comprobar si es capaz de responder ante estas. Cabe destacar aquí el método básico de control de errores anteriormente mencionado que la aplicación posee, el cual se basa en que cada vez que debe ejecutarse una sentencia SQL se tendrá en cuenta tres situaciones posibles, que la respuesta sea positiva y la sentencia haya sido correctamente ejecutada, que se haya producido un error en la construcción de la sentencia SQL, controlado gracias al uso de la excepción *SQLException* o *SQLiteException* de java y por último un error no previsto, controlado gracias al uso de la excepción *Exception* de java.

A continuación se mostrarán situaciones típicas que darán lugar a error en el uso habitual de la aplicación.

1. Error de conexión y error de inicio de sesión

Un error habitual al inicio de la aplicación sería el de intentar conectarse con el servidor MySQL sin tener ningún tipo de conexión a Internet, ya sea WiFi o datos, en cuyo caso aparecería un mensaje advirtiendo de que no ha sido posible la conexión. Otro error habitual sería el de introducir mal las credenciales de usuario y contraseña lo que provocaría que en la aplicación apareciese un mensaje de 'Error usuario o contraseña incorrectos'. Ambos errores pueden contemplarse en la **Figura 4.11**.

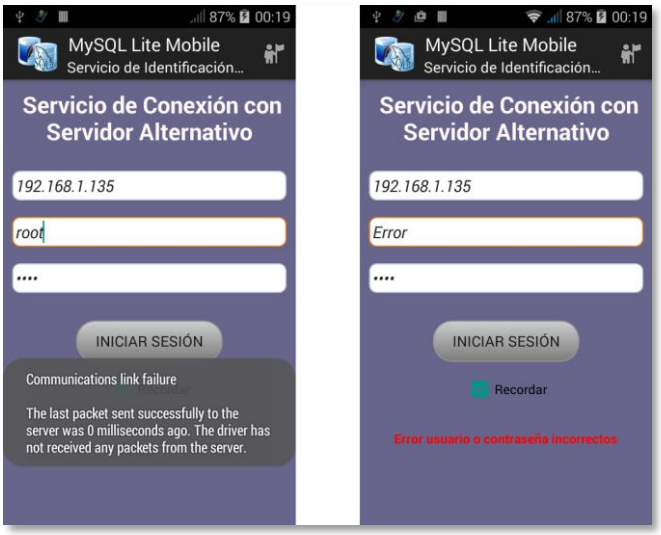


Figura 4.11 Errores de conexión e inicio de sesión

2. Error MySQL de acceso denegado

Existe una situación en la que el usuario podría intentar realizar una acción en un servidor MySQL de la que no tiene permiso, por ejemplo es habitual que en un servidor MySQL el único con capacidad de crear bases de datos sea el administrador del mismo, por ello si un usuario sin los privilegios adecuados intentase crear una desde la aplicación se encontraría con un mensaje de 'acceso denegado'. Un ejemplo gráfico de esta situación puede contemplarse en la **Figura 4.12**.

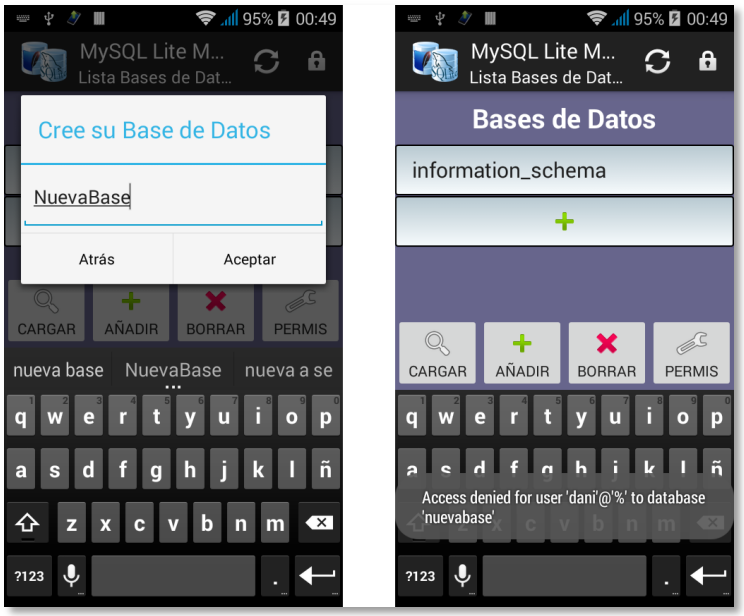


Figura 4.12 Error acceso denegado

3. Error MySQL de insertar registros

Cuando un usuario intenta insertar registros en una tabla pueden ocurrir diversos tipos de errores MySQL, por ejemplo un usuario podría intentar introducir un dato que fuese una palabra en una columna que solo admita valores numéricos o bien no insertar ningún valor cuando la propiedad que impide valores nulos está activa. En la **Figura 4.13** pueden apreciarse ambas situaciones. En la primera pantalla se intentará introducir un registro con valor nulo cuando la propiedad que permite emplear valores nulos no está activa y en la segunda se pretenderá introducir un registro con un valor 'quince' en una columna que solo admite valores numéricos.

Figura 4.13 Errores de insertar registros

4. Otros tipos de errores

Existe una multitud enorme de errores de tipo SQL que el uso de la aplicación podría acarrear, no obstante esta respondería a estos mostrándolos por pantalla de la misma manera que los anteriores. Por ejemplo se podrían producir errores similares si se intentase cambiar el valor de un registro, o si se cambiase el parámetro 'Null' de un campo o bien si se intentase establecer una clave foránea entre campos de diferente tipo. Cabe destacar también que la aplicación respondería ante un error no previsto mostrando un mensaje de 'Error inesperado contacte con el administrador'.

4.3 Encuesta a usuarios

En primer lugar se ha escogido a tres personas diferentes a las que se les ha proporcionado la aplicación móvil para que hagan uso de la misma y den su opinión acerca de ella. Antes de probarlo se les ha explicado su utilidad y la forma en la que se utiliza y funciona de forma breve. Una vez probadas diferentes funcionalidades de la aplicación han contestado a las siguientes preguntas:

1. ¿Consideras útil el uso de la aplicación?
2. ¿Has encontrado la interfaz intuitiva y manejable?
3. ¿Qué es lo que más te ha llamado la atención?
4. ¿Hay algo que no te haya gustado o hay algún aspecto que pudiese mejorarse?

A continuación se mostrarán los datos de los usuarios encuestados y cuáles han sido sus respuestas a las preguntas anteriores.

Usuario 1: Ana María Rojas Martínez estudiante de máster de topografía de 23 años de edad con conocimientos básicos de programación, bases de datos y Android.

1. Sí puesto que ofrece una manera cómoda de gestionar bases de datos desde cualquier lugar ya que todo el mundo suele llevar su móvil consigo.
2. Sí aunque es más cómodo de utilizar en pantallas más grandes.
3. Las numerosas capacidades de edición.
4. Quizás la interfaz podría ser más profesional.

Usuario 2: Jesús Martín Banderas estudiante de ingeniera telemática de 23 años de edad con conocimientos básicos de programación, bases de datos y Android.

1. Sí ya que cuenta con las capacidades de edición más habituales de SQL.
2. Sí bastante una vez que la has usado un poco es muy intuitiva.
3. La capacidad de trasladar tablas desde el servidor al móvil.
4. No se me ocurre ninguna destacable.

Usuario 3: Nicolás Ángel Serrano Linares de 23 años de edad con conocimientos de programación, bases de datos y Android.

1. Sí, me parece muy curioso poder gestionar bases de datos desde el móvil.
2. Sí de hecho no se me ocurría en este momento cómo poder mejorarla.
3. El poder trasladar tablas entre servidor y móvil y poder seguir gestionándolas.
5. El administrador podría tener funcionalidades diferenciadas del resto de usuarios como poder crear o borrar usuarios.

Teniendo en cuenta la opinión de los usuarios que han probado la aplicación podemos afirmar que cumple los requisitos funcionales que el alumno había planteado para la misma, que aporta un sistema útil y cómodo de utilizar para el usuario y que es un programa abierto a mejoras tanto gráficas como funcionales. Conclusivamente puede afirmarse con seguridad que los usuarios han quedados satisfechos con la aplicación.

5. CONCLUSIONES

A continuación se expondrán brevemente las conclusiones a las que ha llevado al alumno el desarrollo de este proyecto de fin de grado relacionado tanto en el ámbito del desarrollo y uso de la aplicación como en el de la adquisición de conocimientos y habilidades. Finalmente aparecerán una serie de opciones y extensiones que podrían ser desarrolladas para la aplicación en un futuro.

5.1 Conclusiones generales

Este proyecto ha sido realizado con el fin de crear una aplicación para dispositivos móviles capaz de gestionar de manera remota una base de datos MySQL mediante uso de un interfaz adecuado evitando así la introducción manual de sentencias. Dichas bases de datos podrían ser utilizadas por otras aplicaciones, tales como páginas web, o bien ser simplemente utilizadas como simples almacenes de información.

Para el desarrollo de la aplicación se ha decidido utilizar como plataforma el sistema operativo Android, debido a su alta tasa de mercado en España y las facilidades que aporta al desarrollador y como IDE Android Studio, ya que es el oficial elegido por Google para el desarrollo de aplicaciones Android.

Aunque la creación de la aplicación ha supuesto grandes difucultades al alumno debido a la poca experiencia de este en el desarrollo de interfaces gráficas, puede afirmarse con seguridad ha sido provechoso para él puesto que gracias a esto ha conseguido habilidades competentes en el ámbito de programación con una tecnología actualmente en auge.

Finalmente puede decirse que el desarrollo de este proyecto de final de grado ha concluido, tras numerosas horas de trabajo, con los objetivos docentes y funcionales propuestos dentro del tiempo estipulado por la asignatura y que el alumno ha adquirido una cantidad considerable de conocimientos en el desarrollo de aplicaciones SQL, Android y de comunicaciones seguras a través de la red.

5.2 Trabajo futuro

La aplicación desarrollada en este proyecto incorpora una gran cantidad de funcionalidades básicas para la gestión de bases de datos no obstante, esta cantidad queda limitada en gran medida debido al tiempo asignado para su elaboración. Dicho esto último y teniendo en cuenta la cantidad de funcionalidades SQL que existen puede considerarse esta aplicación como abierta a nuevas actualizaciones. Algunos ejemplos de posibles actualizaciones son los siguientes:

1. Un sistema dedicado para administrador para la creación y eliminación de usuarios entre otras funcionalidades.
2. Capacidad de realizar consultas complejas pudiendo seleccionar varias tablas, campos y agrupación de registros.
3. Funciones para caracterizar aun más las columnas de registros tales como operaciones matemáticas o concatenación de registros.
4. Funcionalidades para la creación, modificación y eliminación de disparadores e índices.
5. Una consola de comandos para la introducción manual de sentencias.
6. Capacidad de conectar con otros sistemas SQL.

6. ANEXO 1: INSTALACIÓN DEL SOFTWARE

El objetivo de la aplicación que aquí se ha desarrollado es la de permitir al usuario poder gestionar bases de datos con tecnología MySQL ubicadas en un servidor externo desde un dispositivo móvil. Para tal fin la aplicación ofrece la posibilidad de poder conectarse directamente a un servidor particular, o bien uno que sea escogido por el usuario introduciendo para ello los parámetros de conexión necesarios. Dicho esto a continuación se procederá a explicar por un lado, los pasos a seguir para la correcta instalación de un servidor MySQL en una computadora Windows y por otro cómo se instalará la aplicación cliente en un terminal Android.

6.1 Instalación de servidor

El instalador del servidor MySQL puede encontrarse en la página oficial de MySQL [44] y puede ser descargado gratuitamente por cualquier usuario. Una vez que se ha descargado el instalador este podrá ser abierto, tras lo cual aparecerán una serie de ventanas que permitirán al usuario caracterizar diferentes parámetros del servidor tales como la carpeta destino donde se instalarán los archivos del servidor, el puerto donde escuchará el servidor (por defecto 3306), el espacio en memoria dedicado a las bases de datos, el número de conexiones simultáneas permitidas a cada usuario y las credenciales de administrador del servidor. Todo el proceso de instalación resulta muy intuitivo como puede apreciarse en la **Figura 6.1** y bastará con pulsar sobre el botón 'Siguiente' para realizar una instalación efectiva hasta su finalización. Se instalará además automáticamente un cliente llamado MySQL WorkBench que permite la gestión directa de las bases de datos del servidor desde el computador donde reside el servidor. En la **Figura 6.2** puede encontrarse una muestra de la interfaz del software MySQL WorkBench

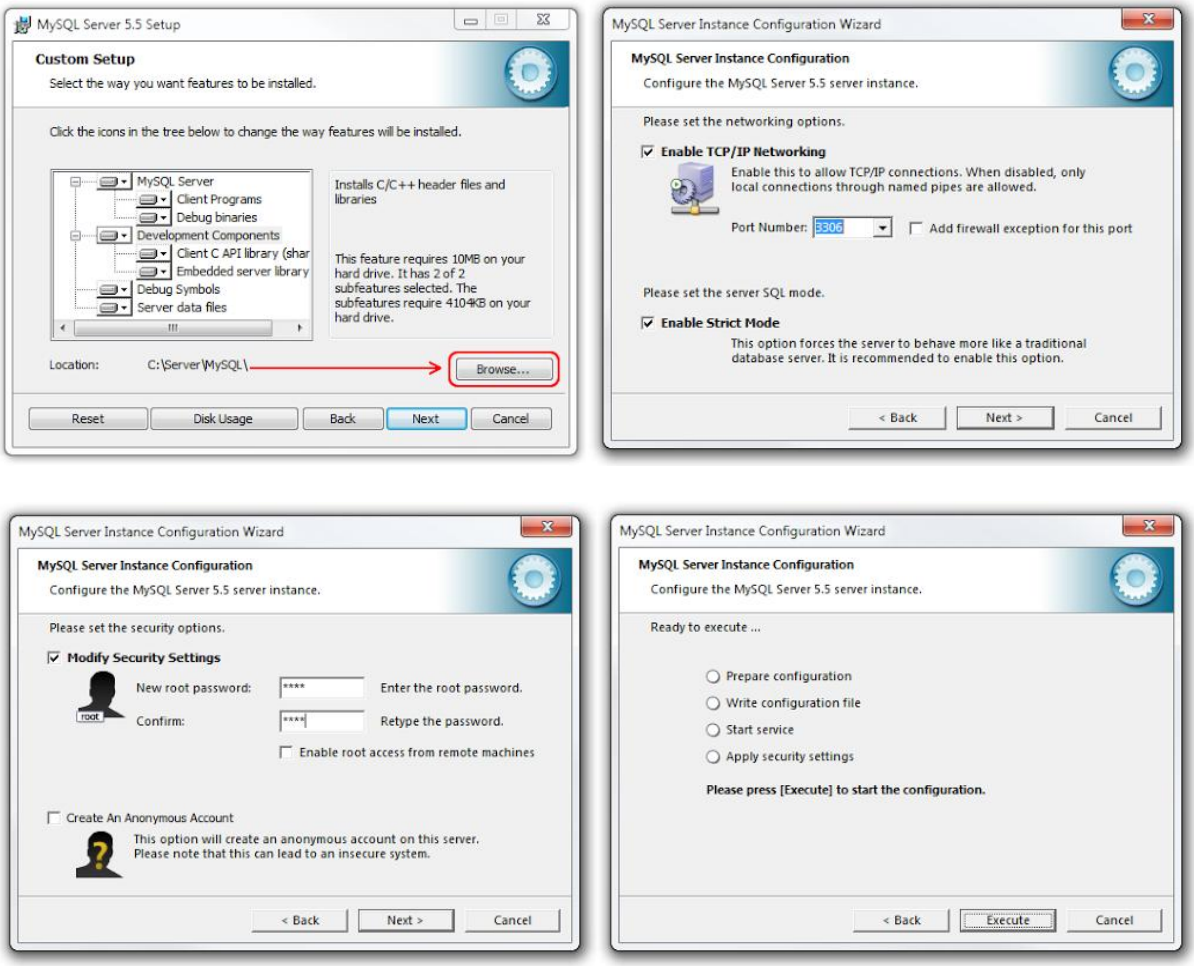


Figura 6.1 Instalación de un servidor MySQL [45]

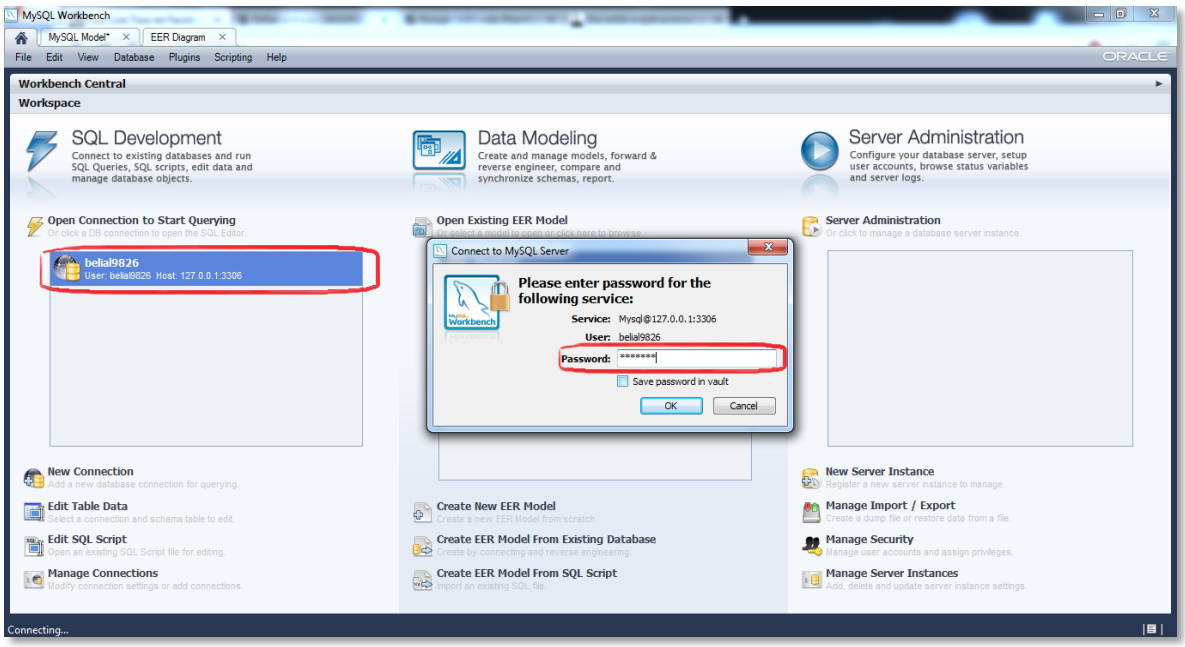


Figura 6.2 Interfaz MySQL WorkBench [46]

6.2 Instalación del cliente

El instalador de la aplicación móvil cliente podrá descargarse desde la tienda oficial de Google [47] de forma gratuita. La instalación de esta aplicación es extremadamente sencilla, simplemente tras descargar el instalador con formato apk típico de los dispositivos Android este debe ser pulsado, tras esto se mostrará una ventana informando al usuario de los permisos necesarios para el uso de la aplicación tal y como se muestra en la **Figura 6.3** y deberá pulsarse sobre el botón 'Instalar'.

Figura 6.3 Instalación aplicación cliente ProyectoTFG o MySQL Lite Mobile

7. ANEXO 2: MANUAL DE USUARIO

El uso de la aplicación 'MySQL Lite Mobile' tiene como objetivo el poder ofrecer al usuario la capacidad de poder gestionar de manera remota las bases de datos ubicadas en un servidor MySQL externo al dispositivo. Para poder realizar dicha tarea esta cuenta aplicación con diferentes funcionalidades a disposición del usuario e interfaces sobre las que este pueda trabajar de manera cómoda. A continuación procederá a describirse cómo el usuario podrá hacer uso de estas funcionalidades.

7.1 Inicio de la aplicación

Una vez que el usuario haya instalado la aplicación en su dispositivo se encontrará con un nuevo icono tal y como aparece en la **Figura 7.1.**

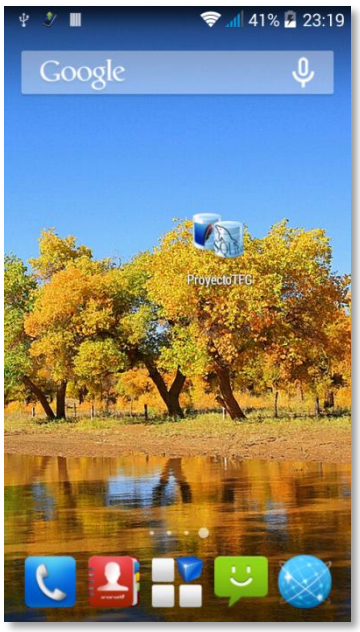


Figura 7.1 Icono MySQL Lite Mobile

Una vez que la aplicación haya sido abierta se encontrará la interfaz de la imagen de la **Figura 7.2** desde la cual el usuario podrá realizar las siguientes tareas:

1. Cambio de actividad: Desde las opciones situadas en la barra de acción el usuario puede trasladarse a la interfaz que aparece en el apartado 7.4 de este manual pulsando sobre la opción 'Servidor Alternativo' o bien a la interfaz del apartado 7.6 pulsando 'Servidor Móvil'.

2. Inicio de sesión: El usuario introducirá sus credenciales de usuario y contraseña en los campos habilitados para ello, tras ello podrá pulsar el botón 'Iniciar Sesión' para identificarse ante el servidor MySQL con dirección www.servidormysql.no-ip.org. Si la opción de recordar está activada y el usuario logra identificarse exitosamente con el servidor sus credenciales de usuario y contraseña permanecerán ya escritas directamente en los campos de texto. Si el usuario es correctamente autenticado la aplicación avanzará a la interfaz del apartado 7.4.

3. Registro de usuario: Al pulsar sobre el botón 'No Tengo Cuenta, Registrarme Ahora' la aplicación abrirá el navegador automáticamente con un formulario el cual será descrito en el siguiente apartado.

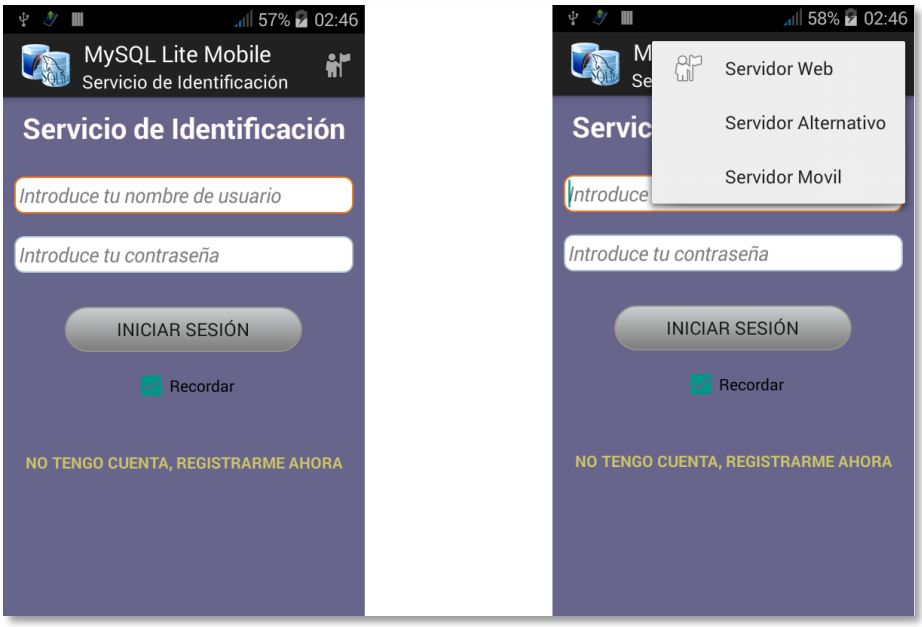


Figura 7.2 Inicio de aplicación

7.2 Formulario de registro

Con el uso de este formulario el usuario podrá registrarse en el servidor anteriormente mencionado, para ello el usuario deberá introducir sus credenciales de usuario y contraseña y un nombre para una base de datos en los campos indicados para ello. Tras pulsar el botón 'Registrarme' el usuario será efectivamente registrado en el servidor y una base de datos será creada bajo su propiedad con el nombre anteriormente introducido. Dicho formulario puede apreciarse en la **Figura 7.3**.

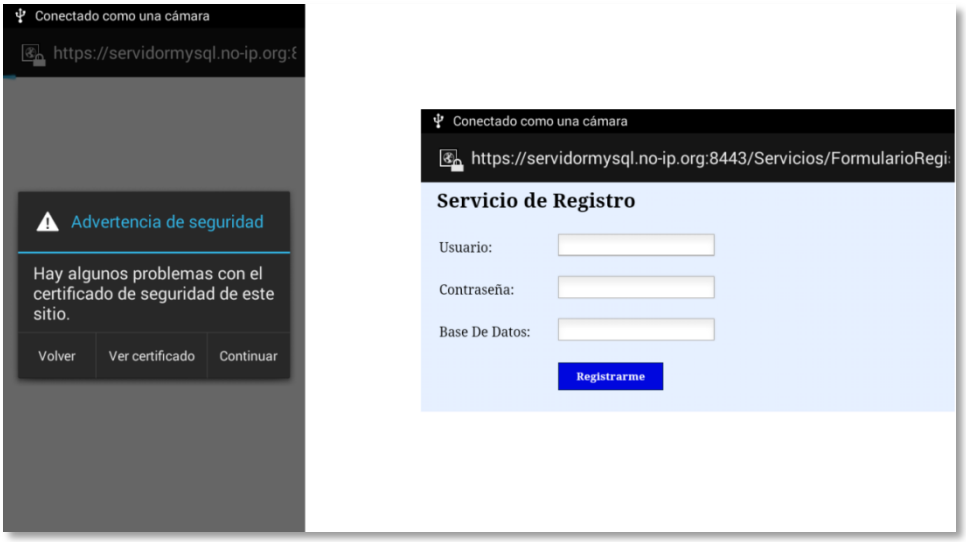


Figura 7.3 Formulario de registro

Nota: Como puede apreciarse en la **Figura 7.3** el navegador mostrará al principio un advertencia de seguridad, esto es debido a que el certificado utilizado para conseguir un servicio de confidencialidad, servicio por el cual se consigue que la información intercambiada entre el usuario y el servidor sea cifrada, no está firmado por ninguna entidad de certificación oficial.

7.3 Inicio login alternativo

La interfaz que se presenta en la **Figura 7.4** dispondrá al usuario de las funcionalidades suficientes para poder identificarse frente a un servidor MySQL que este elija. En ella se destacan dos tareas principales:

1. Cambio de actividad: Desde las opciones situadas en la barra de acción el usuario puede trasladarse a la interfaz del apartado 7.1 pulsando en la opción 'Servidor Web' o bien a la interfaz del apartado 7.6 pulsando 'Servidor Móvil'.

2. Inicio de sesión: El usuario deberá deberá introducir en los campos de texto indicados sus credenciales de usuario y contraseña y la dirección IP del servidor objetivo, tras ello deberá pulsar el botón 'Iniciar Sesión' para identificarse ante el servidor. Si la opción de recordar está activada y el usuario logra identificarse exitosamente con el servidor sus credenciales de usuario y contraseña permanecerán ya escritas directamente en los campos de texto. Si el usuario es correctamente autenticado la aplicación avanzará a la interfaz del apartado 7.4.

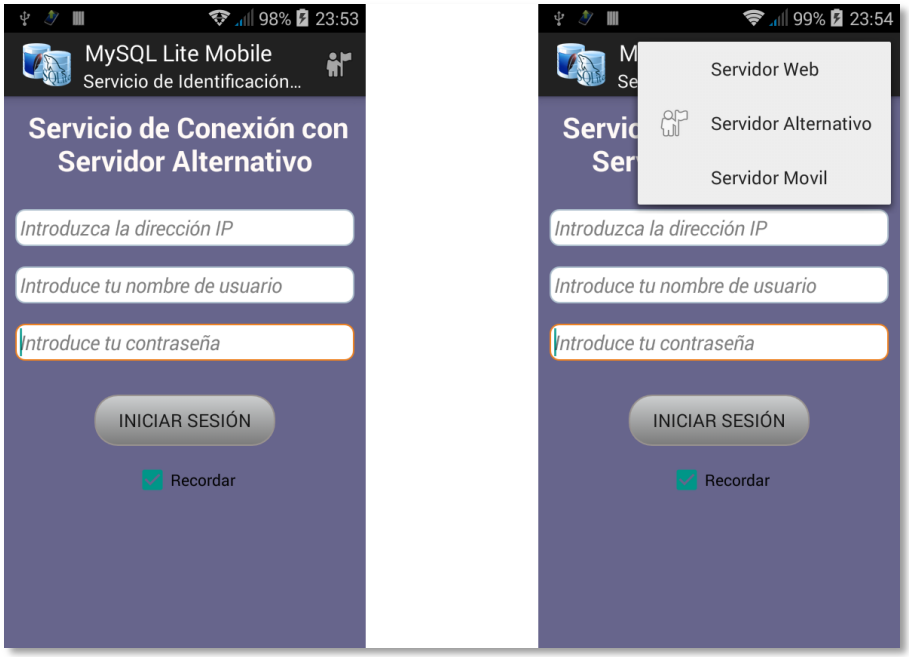


Figura 7.4 Inicio de login alternativo

7.4 Listar bases de datos

La interfaz que es presentada en la **Figura 7.5** dispondrá al usuario de diferentes funcionalidades relacionadas con el tratamiento de las bases de datos ubicadas en el servidor MySQL en el que el usuario se haya identificado. También proporcionará diferentes funciones relacionadas con la capacidad de compartir y revocar privilegios sobre las tablas a las que el usuario tenga acceso.

En primer lugar la interfaz muestra una lista de las bases de datos a las que el usuario tiene acceso y un pequeño menú que aportará diferentes opciones. El usuario a través de esta interfaz podrá desarrollar una cantidad considerable de tareas las cuales se muestran a continuación.

1. Visualización de bases de datos: La interfaz muestra una lista con las bases de datos a las que tiene acceso el usuario identificado.

2. Creación de bases de datos: Tras pulsar el botón 'Añadir' el usuario podrá crear nuevas bases de datos en el servidor. Según la actividad anterior procedente se distinguirán dos tipos de situaciones, es decir, si la aplicación anteriormente mostraba la interfaz del apartado 7.1 al pulsar el botón 'Añadir' se abrirá el navegador automáticamente el formulario del apartado 7.2, no obstante, si la interfaz anterior era la del apartado 7.3 se mostrará una ventana con un campo de texto donde el usuario deberá introducir el nombre de la nueva base de datos, y dos botones 'No' y 'Sí', para confirmar o no la creación de la base de datos

3. Borrado de bases de datos: Después de que el usuario seleccione una base de datos de la lista y de que pulse el botón 'Borrar' aparecerá una ventana con los botones 'No' y 'Sí' para confirmar o no la eliminación completa de la base de datos seleccionada.

4. Visualización de log de sentencias MySQL: Tras pulsar sobre el botón 'Permisos', la opción 'Log Comandos SQL' puede ser elegida, acción que permite a la aplicación trasladarse a la interfaz mostrada en el apartado 7.7.

5. Visualización de privilegios: Tras pulsar sobre el botón 'Permisos' el usuario accederá a las opciones relacionadas con la disposición de privilegios MySQL en el servidor. La opción 'Ver Privilegios' posibilitará que la aplicación permita al usuario visualizar todos los privilegios que tiene en el servidor MySQL.

6. Compartir y revocar privilegios: Con la opción de 'Permisos' ya pulsada aparecerá la opción de 'Compartir Privilegios' la cual después de ser pulsada mostrará un menú con diferentes opciones entre las que se encuentran:

Un campo de texto para indicar el usuario al que va dirigida la acción, un seleccionador de bases de datos y otro de las tablas pertenecientes a la base de datos seleccionada, un conjunto de seleccionables de privilegios y cinco botones cuyas funciones se expondrán ahora. Un botón 'Salir' permite cerrar el menú, el botón 'Otorgar' permite otorgar los privilegios señalados al usuario indicado de la tabla seleccionada pertenecientes a la base de datos introducida, mientras que el botón 'Otorgar Todos' permitirá otorgar todos los privilegios MySQL posibles, de forma opuesta los botones 'Revocar' y 'Revocar Todos' permiten revocar los privilegios indicados, o bien, todos los privilegios.

7. Visualizar tablas de una base de datos: Tras seleccionar una base de datos y pulsar el botón 'Cargar' la aplicación mostrará la interfaz del apartado 7.6.

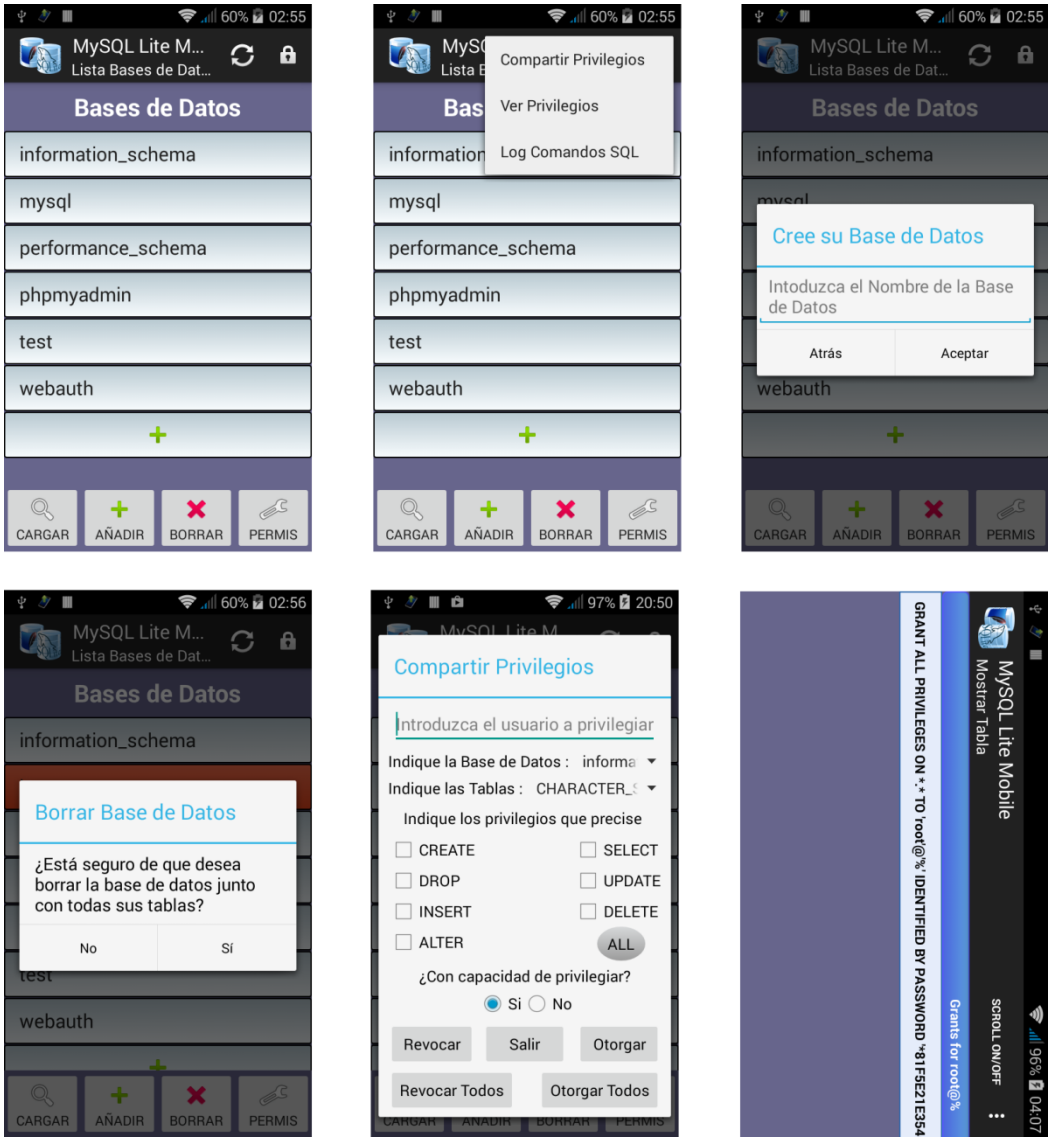


Figura 7.5 Lista bases de datos

7.5 Formulario para crear bases de datos

El formulario mostrado en la **Figura 7.6** permite al usuario la creación de bases de datos en el servidor con dirección www.servidormysql.no-ip.org. Los campos que forman el formulario son las credenciales de usuario y contraseña, necesarias para identificar al poseedor de la nueva base de datos, y el nombre que tendrá la base de datos a crear. Finalmente aparece un botón 'Crear Base' que el usuario deberá pulsar para que la información anteriormente mencionada sea enviada al servidor para que finalmente la base de datos pueda ser creada.

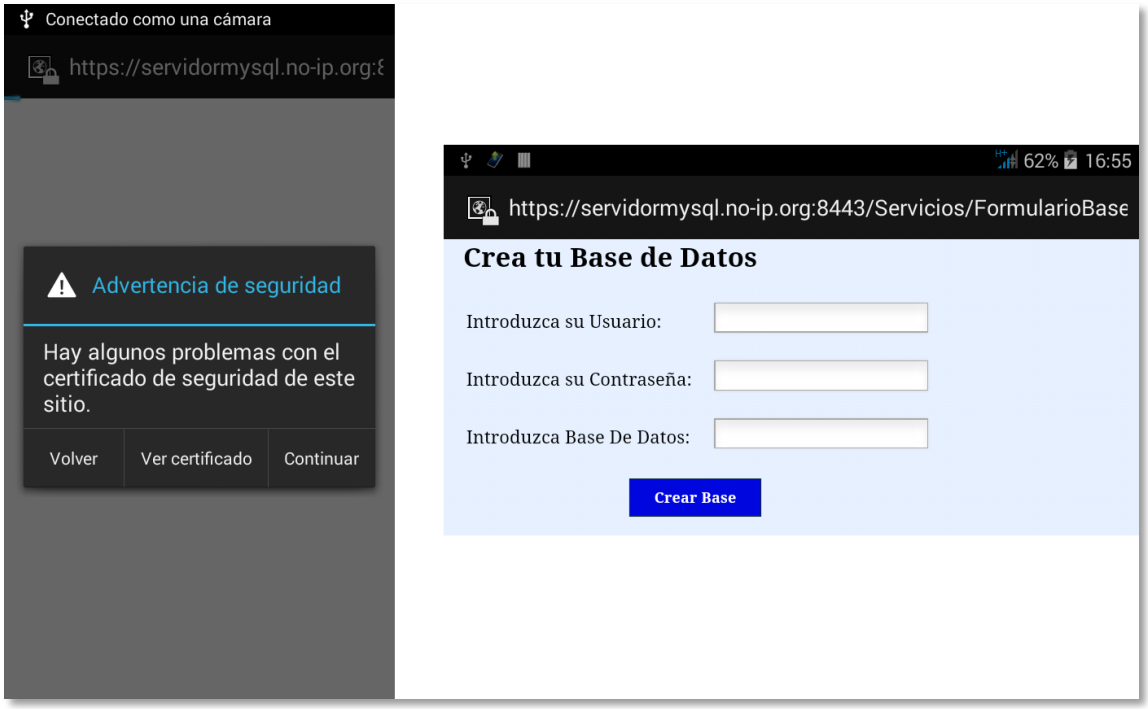


Figura 7.6 Interfaz de *BasesDeDatos.jsp*

Nota: Como puede apreciarse en la **Figura 7.6** el navegador mostrará al principio un advertencia de seguridad, esto es debido a que el certificado utilizado para conseguir un servicio de confidencialidad, servicio por el cual se consigue que la información intercambiada entre el usuario y el servidor sea cifrada, no está firmado por ninguna entidad de certificación oficial.

7.6 Listar tablas en dispositivo y en servidor

Para la visualización y gestión de tablas ubicadas tanto en el dispositivo móvil como en el servidor MySQL se trabajará con casi la misma interfaz visual, aunque contarán con funcionalidades un tanto diferentes que serán explicadas a continuación. Por un lado se dispondrá al usuario de diferentes funcionalidades relacionadas con el tratamiento de tablas y registros pertenecientes a una base de datos ubicadas en un servidor MySQL en el que el usuario se haya identificado previamente, mientras que por otro, las tablas a las que se aplicarán las funcionalidades serán las contenidas en una base de datos ubicadas en el mismo dispositivo móvil.

La interfaz que puede ser visualizada en las **Figuras 7.7 y 7.8** muestra una lista de las tablas pertenecientes a una base de datos a las que el usuario tiene acceso y un pequeño menú que aportará diferentes opciones. El usuario a través de esta interfaz podrá desarrollar una cantidad considerable de tareas las cuales se muestran a continuación.

1. Visualización de tablas: La interfaz muestra una lista con las tablas pertenecientes a una base de datos anteriormente seleccionada por el usuario.

2. Creación de tablas: Tras pulsar el botón 'Añadir' la aplicación se desplazará a la interfaz mostrada en el apartado 7.8.

3. Borrado de tablas: Después de que el usuario seleccione una tabla de la lista y de que pulse el botón 'Borrar' aparecerá una ventana con los botones 'No' y 'Sí' para confirmar o no la eliminación de la tabla.

4. Visualizar registros de una tabla: Tras seleccionar una tabla y pulsar el botón 'Cargar' la aplicación se trasladará a la interfaz que se muestra en el apartado 7.9.

Tras pulsar sobre el botón 'Opciones' la aplicación mostrará un menú con diferentes opciones las cuales se discutirán a continuación.

1. Insertar registros: Después de pulsar sobre la opción 'Insertar' la aplicación se trasladará al interfaz que se muestra en el apartado 7.10.

2. Agregar Campos: Después de pulsar sobre la opción 'Agregar Campos' la aplicación se trasladará a la actividad que se muestra en el apartado 7.11.

3. Editar campos, editar registros y editar valores: Después de pulsar sobre la opción 'Editar Campos', 'Editar Registros' o 'Editar Valores' la aplicación se trasladará a la interfaz del apartado 7.12

4. Editar claves foráneas: Después de pulsar sobre la opción 'Claves Foráneas' la aplicación se trasladará a la interfaz del apartado 7.13. La actividad que muestra la lista de tablas ubicadas en el dispositivo móvil en este caso no cuenta con esta funcionalidad y dispondrá de un mensaje de advertencia al usuario.

5. Renombrar tabla: Después de que el usuario seleccione una tabla de la lista y de que pulse pulsa la opción 'Renombrar Tabla' aparecerá una ventana con un campo de texto para que el usuario introduzca un nuevo nombre para la tabla seleccionada y dos botones 'Atrás' y 'Ok' para confirmar o no el renombrado de la tabla.

6. Vaciar tabla: Después de que el usuario seleccione una tabla de la lista y de que pulse pulsa la opción 'Vaciar Tabla' aparecerá una ventana con los botones 'No' y 'Sí' para que el usuario confirme si desea que todos los registros de la lista sean eliminados.

7. Visualización de log de sentencias MySQL o SQLite: Después de pulsar sobre la opción 'Log Comandos SQL' la aplicación se trasladará a la interfaz del apartado 7.7.

8. Traspaso de tablas: Después de que el usuario seleccione una tabla de la lista y de que pulse la opción 'Copia de Seguridad' podrán darse dos situaciones diferentes según la actividad en la que este se encuentre, es decir, si dicha opción se ejecuta desde la actividad que muestra la lista de tablas ubicadas en un servidor MySQL la tabla junto con sus registros serán copiados al dispositivo mientras que, por otro lado, si se ejecutase dicha acción desde la actividad que muestra las tablas ubicadas en el dispositivo un pequeño menú sería abierto para que el usuario introdujese sus credenciales de usuario y contraseña y la base de datos y la dirección IP destino del servidor donde se copiará la tabla seleccionada en unos campos de texto.

9. Copia de tabla a otro servidor: Después de que el usuario seleccione una tabla de la lista y de que pulse pulsa la opción 'Copia a Servidor' un pequeño menú sería abierto para que el usuario introduzca sus credenciales de usuario y contraseña, la base de datos y la dirección IP destino del servidor donde se copiará la tabla seleccionada.

10. Cambio de actividad: Esta posibilidad solo se contempla en la actividad que permite la visualización de tablas ubicadas en el dispositivo móvil. Desde las opciones situadas en la barra de acción el usuario puede trasladarse a la interfaz del apartado 7.1 pulsando sobre la opción 'Servidor Web' o bien a la interfaz del apartado 7.3 pulsando 'Servidor Alternativo'.

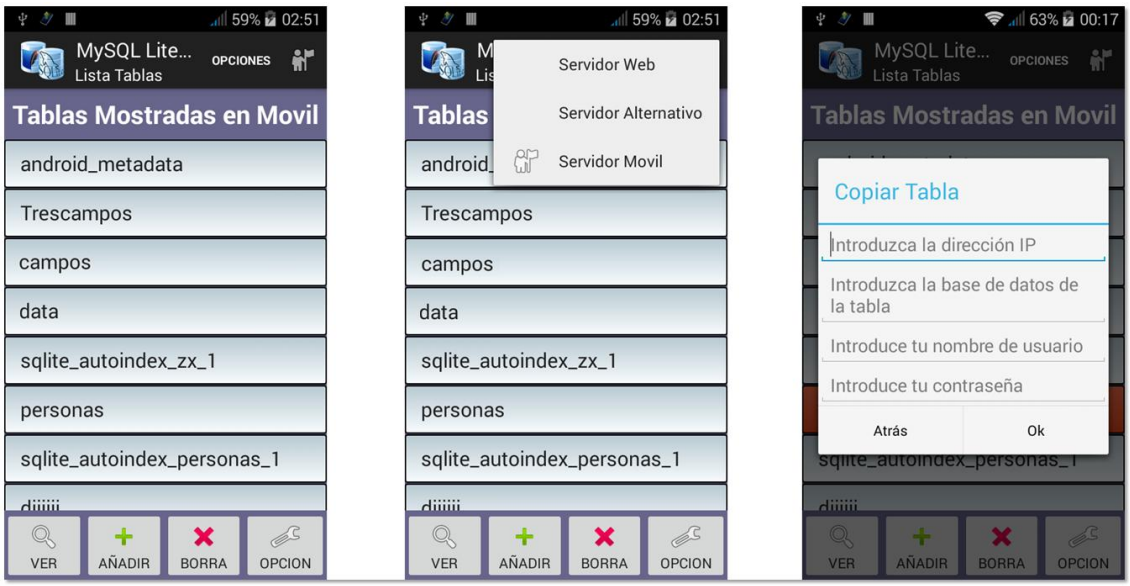


Figura 7.7 Lista tablas dispositivo móvil

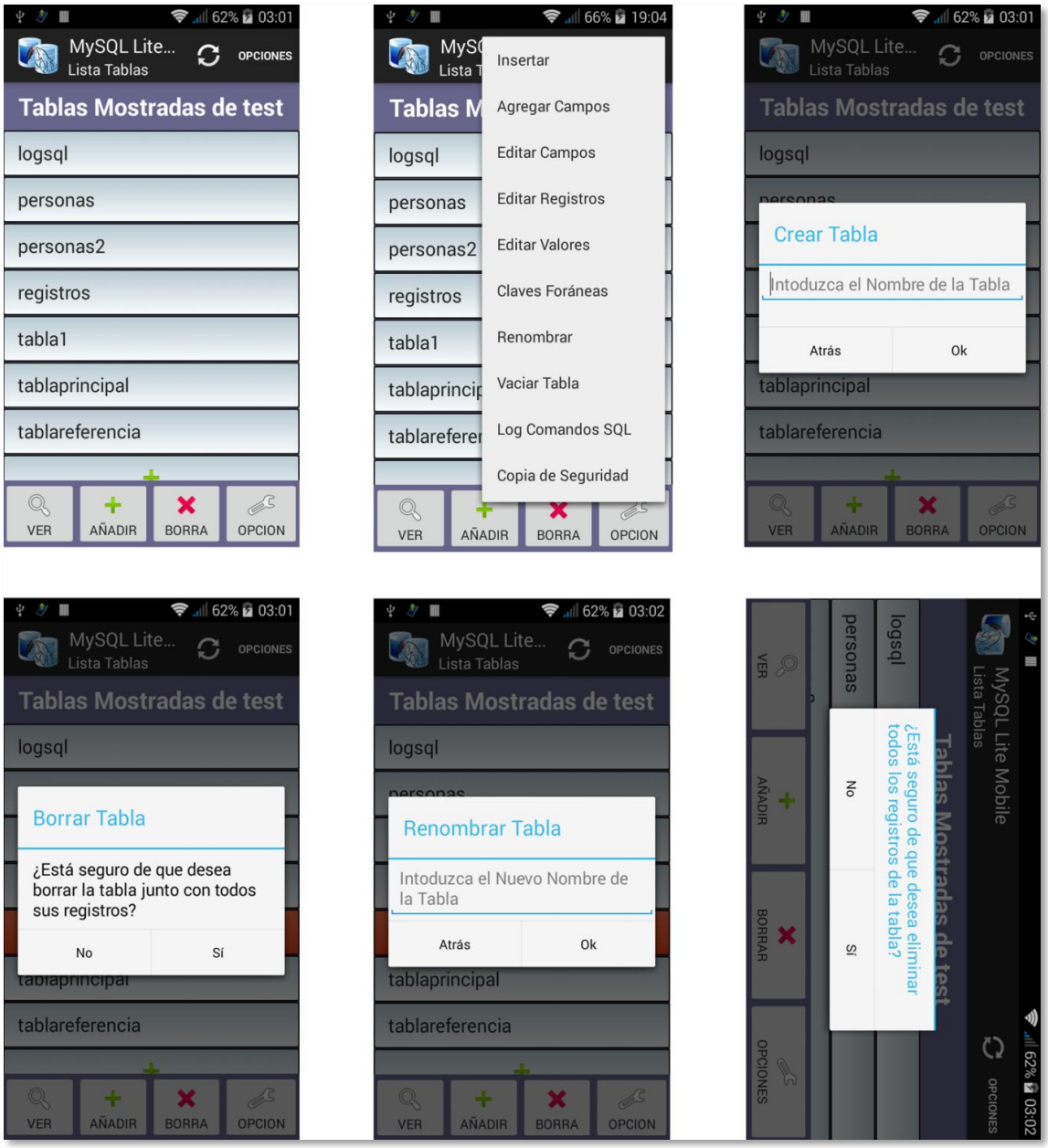


Figura 7.8 Lista tablas servidor MySQL

7.7 Log de sentencias SQL

El usuario puede contemplar las sentencias MySQL enviadas al servidor o bien las sentencias SQLite ejecutadas en el dispositivo por la aplicación. Ambas interfaces pueden apreciarse en la **Figura 7.9**.

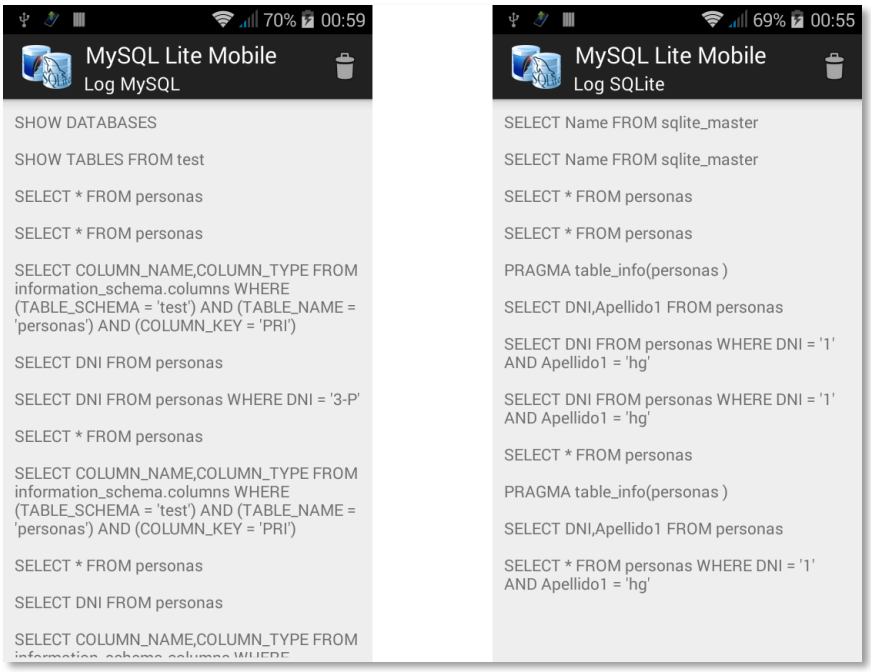


Figura 7.9 Sentencias SQL

7.8 Crear tablas

La aplicación ofrece en la interfaz de las **Figuras 7.10 y 7.11** al usuario un menú diseñado para la creación de tablas de acuerdo a unos parámetros los cuales se presentarán a continuación. Dichos parámetros diferirán en pequeña medida según las tablas sean creadas en el servidor externo o en el dispositivo móvil.

1. Número de campos: Aquí el usuario introducirá el número de campos que desee que tenga la tabla a crear, tras ello deberá pulsar el botón 'Intro'.

2. Campo: El nombre que tendrá el campo que se está editando.

3. Tipo: El tipo de dato que almacenará el campo. Aunque en SQL existe una gran cantidad de tipos la aplicación permite utilizar los más importantes, es decir, Char, Varchar y Text para introducir un texto cualquiera, Int y Float ambos formatos numéricos, el primero para enteros y el segundo para decimales, Date para introducir fechas con formato yyyy-mm-dd por defecto y Time para introducir tiempo con formato hh:mm:ss por defecto.

4. Long: Longitud máxima de caracteres del valor introducido en el campo.

5. Null: Permite o no registros con valor nulo.

6. P.Key: Permite especificar si el campo se considerará parte de la clave primaria de la tabla.

7. Auto: Permite establecer en el campo la propiedad de ser o no autoincrementable. Solo está disponible parra tablas ubicadas en un servidor MyQSL

8. F.Key, Tabla Ref. y Campo Ref: Solo disponibles para la creación de tablas en el dispositivo. Permiten definir establecer la propiedad de clave foránea en los campos de la tabla a crear. El primero permite o no dicha posibilidad, el segundo indica la tabla de referencia y el tercer indica el campo de esa tabla que será de referencia para el campo que se está editando.

Una vez que todos los parámetros hayan sido especificados por el usuario este pulsará sobre el botón 'Crear' para que la tabla sea finalmente creada.

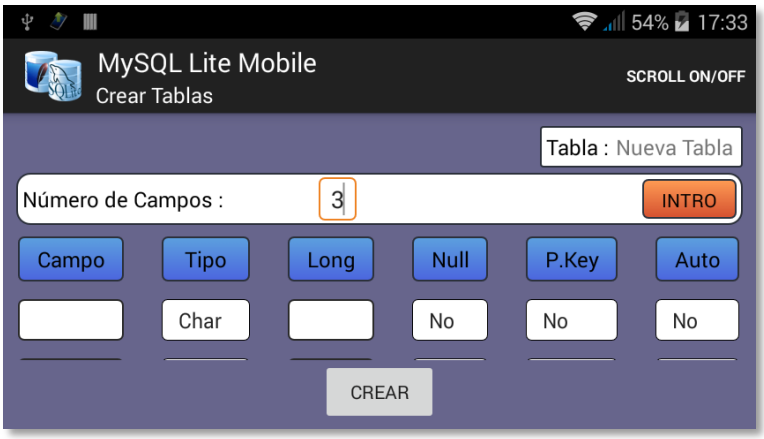


Figura 7.10 Crear tablas en servidor

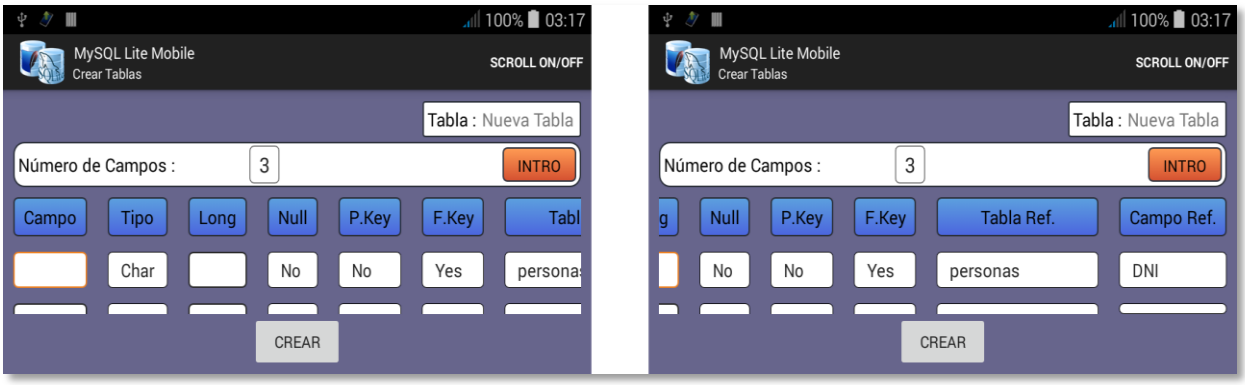


Figura 7.11 Crear tablas en dispositivo

La opción 'Scroll ON/OFF' que aparece por primera vez en esta actividad y que seguirá apareciendo en algunas interfaces venideras permite activar un scroll o desplazable horizontal en los elementos que sean necesarios para su correcta visualización en pantalla.

7.9 Mostrar tabla

La aplicación permite mostrar todos los campos y registros pertenecientes a una tabla escogida previamente por el usuario, además de tres funcionalidades extras ubicadas en la barra de acción las cuales se explican a continuación y que pueden apreciarse en la **Figura 7.12.**

- La primera de ellas 'Pantalla Completa' permite esconder la barra de acción para así aprovechar al máximo el tamaño de la pantalla y poder visualizar mejor tablas que tengan una cantidad considerable de registros.

- La segunda 'Ordenar Resultados' permite ordenar los registros de la tabla según unos parámetros especificados por el usuario, es decir, tras pulsar dicho botón aparecerán dos indicadores uno para introducir el campo por el que ordenar los registros y otro para indicar el orden que podrá ser ascendente o descendente y finalmente dos botones 'Salir', que permite cerrar el menú y 'Mostrar' que permite mostrar los registros ordenados según lo indicado anteriormente.

- La tercera permite obtener registros de la tabla según un campo y el valor de ese campo introducidos. El usuario tras pulsar sobre 'Seleccionar Registros' podrá contemplar dos indicadores para introducir el campo y el valor y dos botones, 'Salir', para cerrar el menú y 'Mostrar' para contemplar los registros coincidentes con los parámetros introducidos.

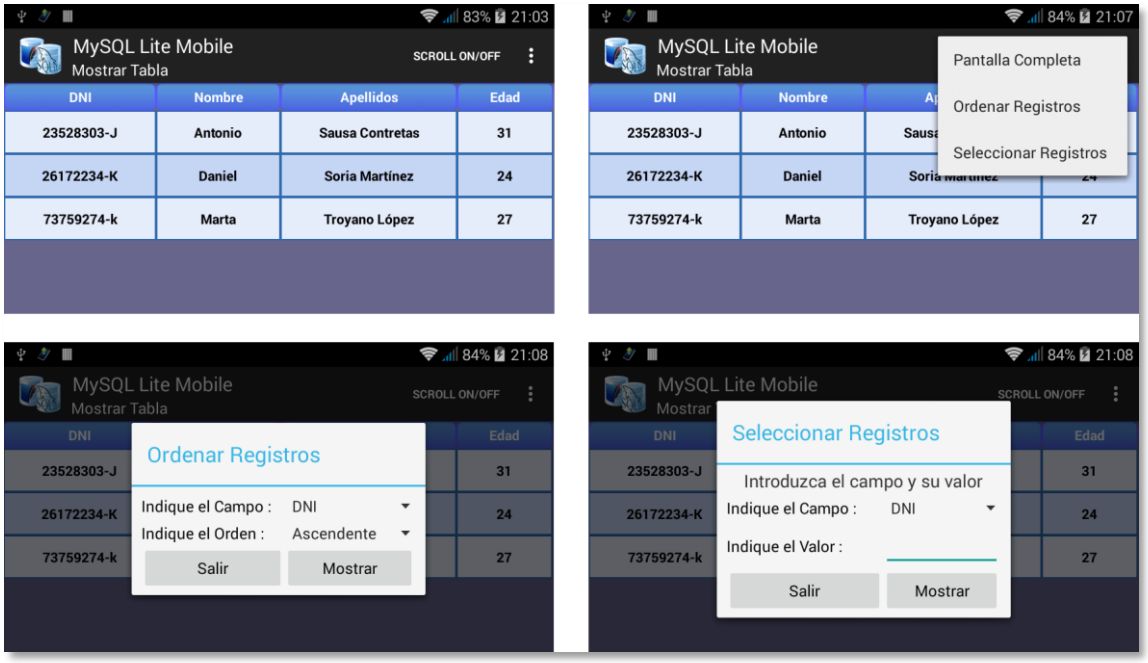


Figura 7.12 Mostrar tabla

7.10 Insertar registros

Esta interfaz provee al usuario de la capacidad de insertar nuevos registros a una tabla de una forma sencilla e intuitiva a través de un menú adecuado para ello. Dicho menú contendrá los siguientes elementos:

1. Elemento Campos: Indica al usuario los nombres de los campos sobre los que insertar los registros y si el campo es o no clave primaria de la tabla mediante la muestra de un icono de una llave.

2. Elemento Tipo: Indica el tipo de información que puede insertarse y la longitud máxima del registro a introducir.

3. Elementos Tipo y Null: Indican respectivamente si el campo puede contener un valor nulo y si tiene la propiedad de ser autoincrementable.

4. Elemento Valores: Indica al usuario unos campos de texto donde deberá insertar los valores que querrá introducir a la tabla.

Tras introducir los valores insertados el usuario deberá pulsar el botón 'Registrar' para que se produzca tal acción. El botón 'Limpiar' tiene como función dejar en blanco los campos de texto para la inserción de registros. Esta interfaz puede verse representada en la **Figura 7.13**.

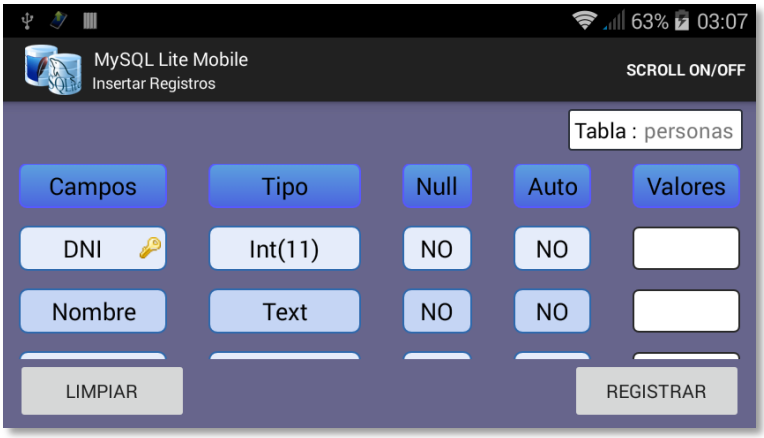


Figura 7.13 Insertar Registros

7.11 Agregar campos

En este apartado se contemplará la posibilidad de agregar nuevos campos a tablas ya creadas, para ello la aplicación ofrece al usuario un menú diseñado para la agregación cómoda de campos a tablas de acuerdo a unos parámetros los cuales se estudiarán a continuación. Dichos parámetros diferirán en pequeña medida según vaya la tabla a ser creada en el dispositivo o en el servidor MySQL tal y como se aprecia en la interfaz de la **Figura 7.15**, la cual contiene los siguientes elementos:

1. Numero de Campos: El usuario, en primer lugar deberá introducir en el campo de texto indicado para ello la cantidad de campos que se agregarán a la tabla en formato numérico, tras ello deberá pulsar sobre el botón 'Intro' y aparecerá un menú para caracterizar cada uno de ellos.

2. Posición de los campos: Se deberá indica el lugar donde se introducirán los campos respecto a los que ya existían, es decir, el usuario a través de la interfaz indicará si se posicionarán al final de la tabla, al principio o después de un campo especificado. Para el caso de tablas en el dispositivo solo podrán situarse al final.

3. Parámetros del campo: Finalmente se deberán caracterizar los parámetros relativos a los campos, es decir, el nombre del campo, el tipo y la longitud máxima de los valores asociados a ese campo y la posibilidad o no de introducir valores nulos en ese campo.

Figura 7.14 Agregar campos

7.12 Edición de campos y registros

La aplicación aportará tres tipos de servicios diferenciados en la interfaz mostrada en las **Figuras 7.15, 7.16**, **7.17**, **7.18 y 7.19** como son la edición (borrado y modificación) de los campos, registros y valores de una tabla.

La interfaz contará con diferentes elementos gráficos para ofrecer estos servicios, los cuales son un grupo de botones, para elegir el servicio de edición que se precise, dos seleccionadores, uno de campos de la tabla a editar y otro con el valor de los campos que forman la clave primaria de la tabla y un menú que será diferente según el servicio seleccionado. Dichos servicios se explican con más detalle a continuación.

1. Edición de campos: Cuya interfaz se muestra en las **Figuras 7.15 y 7.16** y solo disponible para tablas ubicadas en servidores MySQL. Después de que el usuario seleccione un campo la aplicación permitirá la eliminación del elegido pulsando sobre el botón 'Borrar', o bien, la modificación de parámetros de estos mediante los elementos Campo, Tipo, Long, Null y Auto cuyas funcionalidades han sido ya explicadas anteriormente y el uso del botón 'Modificar'. El botón 'Pri.key' tras ser pulsado mostrará un menú para la edición de claves primarias.

Figura 7.15 Editar servidor MySQL (Edición de campos)

Figura 7.16 Editar dispositivo móvil (Edición de campos)

2. Edición de claves primarias: La interfaz para este servicio aparece en la **Figura 7.17** y solo está disponible para tablas ubicadas en un servidor MySQL. Después de que el usuario seleccione un campo la aplicación permitirá la eliminación de la propiedad de clave primaria en el campo elegido pulsando sobre el nuevo botón 'Borrar' con fondo anaranjado, o bien, la agregación del campo como parte de la clave primaria de la tabla pulsando sobre el botón 'Añadir'.

Figura 7.17 Editar (Edición de claves primarias)

3. Edición de registros: La interfaz para este servicio aparece en la **Figura 7.18.** Después de que el usuario seleccione uno o varios valores que formen parte de la clave primaria de la tabla la aplicación mostrará un registro cuyos valores podrán ser modificados mediante el uso de campos de texto y del botón 'Modificar', o bien dicho registro puede ser eliminada gracias al uso del botón 'Borrar'.

Figura 7.18 Editar (Edición de registros)

4. Edición de valores: La interfaz para este servicio aparece en la **Figura 7.19**. Después de que el usuario seleccione un campo de la tabla y uno o varios valores que formen parte de la clave primaria de la tabla un único valor le será mostrado al usuario, el cual podrá ser modificado introduciendo un dato en el campo de texto al lado del indicador 'Nuevo Valor' y pulsando el botón 'Modificar', o bien, eliminado pulsando el botón 'Borrar'.

Figura 7.19 Editar (Edición de valores)

7.13 Claves foráneas

La interfaz mostrada en la **Figura 7.20** ofrece un conjunto de elementos suficientes al usuario como para permitir la capacidad de asignar propiedades de clave foránea a los campos de una tabla o bien de eliminar dicha propiedad**.** A continuación se explicarán cada uno de los elementos que conforman la interfaz divididos en los tres menús diferentes que aparecen en la misma.

1. Menú selección: El usuario seleccionará aquí la tabla y el campo de esa tabla al que se le aplicará la propiedad de clave foránea y la tabla y el campo de esa tabla que serán de referencia o principal en la propiedad de clave foránea.

2. Menú opciones: El usuario seleccionará aquí el comportamiento que tendrá la tabla con la clave foránea al ser eliminado o modificado un registro en la tabla principal. Una vez seleccionados los elementos anteriormente mencionados el usuario pulsará el botón 'Añadir' para aplicar la propiedad de clave foránea.

3. Menú borrar: El usuario seleccionará aquí la tabla y el campo de esa tabla que contiene una o varias propiedades de clave foránea, después seleccionará la 'restricción' que actuará como identificador de las claves foráneas aplicadas a la tabla y pulsará el botón 'Borrar' para eliminar dicha propiedad de clave foránea escogida si así lo desea.

Figura 7.20 Claves foraneás

8. ANEXO 3: ESTUDIO ECONÓMICO Y PLANIFICACIÓN

8.1 Estudio económico

El presente apartado mostrará un estudio económico del actual proyecto fin de grado donde aparecerán los costes que han sido necesarios para su elaboración. Dichos costes se clasificarán en costes materiales, es decir, los atribuidos al software y al hardware necesario para el desarrollo del proyecto y costes de personal referidos a las personas que han intervenido en su elaboración.

8.1.1 Costes materiales

El hardware utilizado para el desarrollo de este proyecto ha sido un ordenador de altas prestaciones y un dispositivo smartphone de gama media. Se ha considerado que el uso de ambos dispositivos para el proyecto ha sido de entorno a unos cuatro meses y que la vida media de un ordenador es de tres años y la de un dispositivo móvil de dos años. Todo el software utilizado para el desarrollo del proyecto es gratuito y por tanto no presenta ningún tipo de coste.

La fórmula para conseguir el coste atribuible al proyecto sería por tanto
Coste Atribuible = (Coste Unitario / Vida Media) * Tiempo De Uso

En la **Tabla 8.1** puede apreciarse un resumen de los costes materiales.

Costes Materiales		
Concepto	Coste Unitario (€)	Coste Atribuible (€)
Ordenador de altas prestaciones	499	54,44
Dispositivo Smartphone	119	19,83
Total	**618**	**75,28**

Tabla 8.1 Costes materiales

8.1.2 Costes de personal

A continuación en la **Tabla 8.2** se muestra una lista con todos los costes asociados a los distintos papeles profesionales que el alumno ha tomado para el desarrollo del proyecto.

Costes De Personal			
Profesional	Horas	Coste Horario (€)	Sueldo (€)
Jefe de Proyecto (Ingeniero)	50	60	3.000
Analista	45	35	1.575
Arquitecto de Software	30	35	1.050
Desarrollador de Software	100	50	5.000
Diseñador Gráfico	50	45	2.250
Tester	25	25	625
Total	**300**		**10.800**

Tabla 8.2 Costes de personal

8.1.3 Costes totales

A continuación se calcularán cuáles han sido los costes totales necesarios para el desarrollo del proyecto. Dichos costes serán la suma de los costes materiales y de personal además de otros costes indirectos tales como la electricidad o el agua los cuales supondrán el diez por ciento de los costes totales. Para finalizar se calculará el IVA actual del proyecto para completar la suma de los costes.

Costes Totales (Sin IVA)	
Concepto	Presupuesto (€)
Costes Materiales	75,28
Costes De Personal	10.800,00
Costes Indirectos (10%)	1.087,53
Total (Sin IVA)	**11.962,81**

Tabla 8.3 Costes totales sin IVA

Costes Totales (Con IVA)	
Concepto	Presupuesto (€)
Costes IVA (21%)	2.512,20
Coste Total	**14.475,01**

Tabla 8.4 Costes totales con IVA

El presente presupuesto del Proyecto Fin de Grado Creación de bases de datos desde un dispositivo móvil asciende a la cantidad de **catorce mil cuatrocientos setenta y cinco euros con un céntimo.**

Linares, Junio 2016.

Firmado: Daniel Soria Martínez

Graduado en Ingeniería Telemática

8.2 Planificación temporal

La estimación temporal para la realización de este proyecto ha sido de trescientas horas de trabajo repartidas en cuatro meses. Debido a la complejidad que presenta la elaboración de este proyecto de fin de grado y para que fuese factible su realización se ha optado por dividir el trabajo en diferentes fases de desarrollo formadas cada una por diferentes tareas. Dichas fases son mostradas a continuación.

Fase 1: Establecimiento de objetivos y documentación inicial

- Análisis de requerimientos y especificación de objetivos (3 días)
- Búsqueda de la información bibliográfica (4 días)
- Estudio de las tecnologías a utilizar (4 días)
- Instalación del software requerido (1 día)
- Búsqueda y realización de tutoriales y aplicaciones sencillas (5 días)

Fase 2: Elaboración del servidor

- Configuración de los servidores MySQL y HTTPS (4 días)
- Creación de formularios (2 días)

Fase 3: Elaboración de la aplicación cliente

- Realización de las actividades java de la aplicación (30 días)
- Realización de las interfaces xml de la aplicación (15 días)
- Conexión entre actividades (2 días)

Fase 4: Realización de pruebas de software en un dispositivo real

- Pruebas en dispositivo ZTE Blade L2 (5 días)
- Corrección y depuración (7 días)

Fase 5: Documentación del proyecto

- Manual de instalación de la aplicación y manual de usuario (4 días)
- Realización de encuestas a usuarios y conclusiones (2 días)
- Realización de capturas de la aplicación (2 días)
- Redacción de la memoria (26 días)
- Corrección y maquetación (4 días)

A continuación será mostrado un diagrama de Gantt el cual albergará todas las tareas mencionadas anteriormente. La **Figura 8.1** muestra las tareas ordenadas cronológicamente con sus duraciones correspondientes mientras que la **Figura 8.2** muestra el diagrama de Gantt con las tareas mencionadas.

Id	Nombre de tarea	Duración	Comienzo	Fin
1	Análisis de requerimientos y especificación de objetivos	3 días	vie 01/01/16	dom 03/01/16
2	Búsqueda de la información bibliográfica	4 días	lun 04/01/16	jue 07/01/16
3	Estudio de las tecnologías a utilizar	4 días	vie 08/01/16	lun 11/01/16
4	Instalación del software requerido	1 día	mar 12/01/16	mar 12/01/16
5	Búsqueda y realización de tutoriales y aplicaciones sencillas	5 días	mié 13/01/16	dom 17/01/16
6	Configuración de los servidores MySQL y HTTPS	4 días	lun 18/01/16	jue 21/01/16
7	Creación de formularios	2 días	vie 22/01/16	sáb 23/01/16
8	Realización de las actividades java de la aplicación	30 días	dom 24/01/16	lun 22/02/16
9	Realización de las interfaces xml de la aplicación	15 días	mar 23/02/16	mar 08/03/16
10	Conexión entre actividade	2 días	mié 09/03/16	jue 10/03/16
11	Pruebas en dispositivo ZTE Blade L2	5 días	vie 11/03/16	mar 15/03/16
12	Corrección y depuración	7 días	mié 16/03/16	mar 22/03/16
13	Manual de instalación de la aplicación y manual de usuario	4 días	mié 23/03/16	sáb 26/03/16
14	Realización de encuestas a usuarios y conclusiones	2 días	dom 27/03/16	lun 28/03/16
15	Realización de capturas de la aplicación	2 días	mar 29/03/16	mié 30/03/16
16	Redacción de la memoria	26 días	jue 31/03/16	lun 25/04/16
17	Corrección y maquetación	4 días	mar 26/04/16	vie 29/04/16

Figura 8.1 Tareas diagrama de Gantt

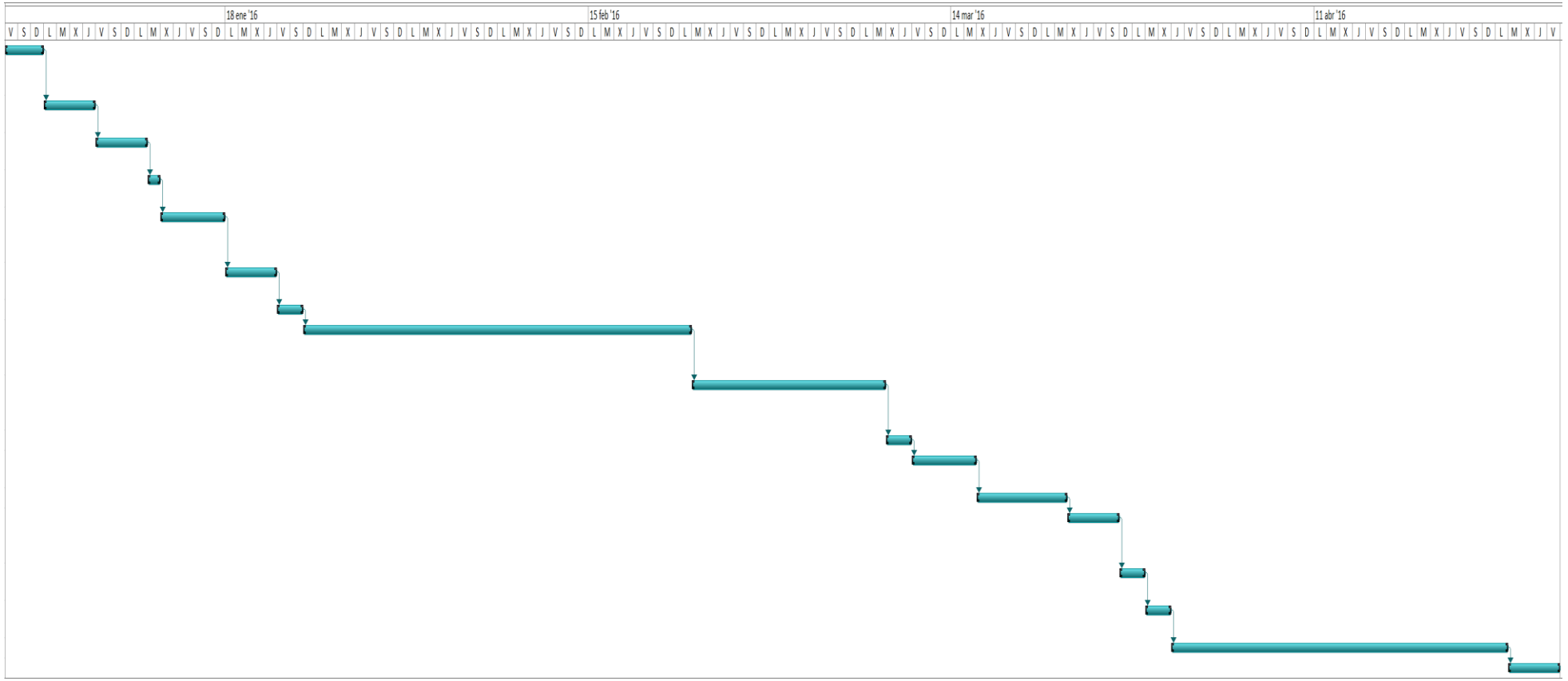


Figura 8.2 Diagrama de Gantt

9. BIBLIOGRAFÍA

[1] En España un 87% de la población española cuenta con algún tipo de smartphone actualmente http://www.xatakamovil.com/movil-y-sociedad/espana-territorio-smartphone

[2] **Figura 1.1** Cuota de mercado de sistemas operativos en España http://es.kantar.com/tech/m%C3%B3vil/2016/enero-2016-cuota-de-mercado-de-smartphones-en-espa%C3%B1a-diciembre-2015/

[3] **Figura 1.2** Principales usos de smartphones http://es.slideshare.net/liburutegiak/liburutegiak-press-esp-32755578/3

[4] **Figura 1.3** Ejemplo de tabla http://2.bp.blogspot.com/-6IMfESsT7nc/T4w0yiD_m9I/AAAAAAAAAEc/BeY2W5oQ99E/s1600/Ejemplo+-5Tabla+2.bmp

[5] Bases de datos http://www.maestrosdelweb.com/que-son-las-bases-de-datos/

[6] Imagen base de datos MySQL http://findicons.com/files/icons/977/rrze/720/database_MySQL.png

[7] Imagen PC servidor https://www.google.es/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&ved=&url=https%3A%2F%2Fsvn.apache.org%2Frepos%2Fasf%2Fincubator%2Fooo%2Fsymphony%2Ftrunk%2Fmain%2Fextras%2Fsource%2Fgallery%2Fsymbols%2FIcon-Computer01White.png&bvm=bv.115339255,d.d24&psig=AFQjCNFzHuDtLvYlS88B_63s9eSg09D2VA&ust=1456969621593918

[8] Imagen Base de datos https://www.google.es/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&ved=0ahUKEwiRmPKDwqTLAhXC0hoKHZ7zC7QQjBwIBA&url=http%3A%2F%2Fwww.fancyicons.com%2Fdownload%2F%3Fid%3D4843%26t%3Dpng%26s%3D256&psig=AFQjCNHm18a_tofnp0fhsX9to7XIbXnmMg&ust=1457094468697252

[9] Logo SQLite http://ichnolite.com/html/assets/images/sqllite.png

[10] Definición de teléfono móvil https://www.google.es/webhp?sourceid=chrome-instant&ion=1&espv=2&ie=UTF-8#q=que%20es%20un%20telefono%20movil

[11] Antecedentes del teléfono móvil http://www.muycanal.com/2014/01/31/futuro-del-telefono-movil

[12] Antecedentes del teléfono móvil
https://es.wikipedia.org/wiki/Historia_del_tel%C3%A9fono_m%C3%B3vil
[13] Antecedentes Sistemas Operativos http://jjuanhdez.es/?p=726
[14] Evolución BlackBerry https://es.wikipedia.org/wiki/BlackBerry
[15] Evolución Symbian https://es.wikipedia.org/wiki/Symbian
[16] Evolución iPhone https://es.wikipedia.org/wiki/IPhone#Historia
[17] Evolución Android https://es.wikipedia.org/wiki/Android
[18] **Figura 3.1** Cuota de mercado móvil en España [18]
http://www.reasonwhy.es/sites/default/files/styles/body_620_width/public/grafica-mercados-cuota-moviles-espana-reasonwhy.es__1.png?itok=li4JJlcP
[19] Historia Bases de Datos http://dryvalleycomputer.com/index.php/bases-de-datos/introduccion/45-historia-de-las-bases-de-datos
[20] Historia Bases de Datos http://www.monografias.com/trabajos72/base-datos/base-datos.shtml
[21] Aplicaciones SQLite y MySQL https://play.google.com/store
[22] Introducción a Android http://www.monografias.com/trabajos101/sistema-operativo-android/sistema-operativo-android.shtml
[23] **Tabla 3.1** Características de Android
https://es.wikipedia.org/wiki/Android#Caracter.C3.ADsticas
[24] Arquitectura Android
https://sites.google.com/site/swcuc3m/home/android/generalidades/2-2-arquitectura-de-android
[25] **Figura 3.2** Arquitectura Android
https://columna80.files.wordpress.com/2011/02/0013-01-pila-software-android.png
[26] Evolución de Android http://www.xatakamovil.com/sistemas-operativos/de-cupcake-a-marshmallow-asi-han-sido-las-versiones-de-android-a-lo-largo-de-su-historia
[28] Evolución de Android http://es.gizmodo.com/7-anos-de-historia-la-evolucion-de-la-homescreen-de-an-1734716500
[28] **Figura 3.3** Versiones de Android http://www.infinitecurl.net/wordpress/wp-content/uploads/2016/02/android_versiones.jpg
[29] Componentes de una aplicación Android
http://blog.agencialanave.com/introduccion-explicacion-de-los-componentes-principales-de-android-tutorial-2/

[30] Estructura de una aplicación Android http://www.sgoliver.net/blog/estructura-de-un-proyecto-android-android-studio/

[31] Estructura de una aplicación Android http://programando-android.blogspot.com.es/p/estructura-de-un-proyecto-android.html

[32] Ventajas y desventajas de Android http://gigatecno.blogspot.com.es/2014/05/ventajas-y-desventajas-de-android.html

[33] Android Studio http://academiaandroid.com/android-studio-v1-caracteristicas-comparativa-eclipse/

[34] Android Studio https://es.wikipedia.org/wiki/Android_Studio

[35] MYSQL https://es.wikipedia.org/wiki/MySQL

[36] SQLite https://es.wikipedia.org/wiki/SQLite

[37] **Tabla 3.2** SGBD más importantes http://2.bp.blogspot.com/-ml45jAQbewc/UDEt37Uv57I/AAAAAAAAAAM/8VlAJpFEI3M/s1600/Comparacion+SGBD.jpg

[38] Eclipse https://es.wikipedia.org/wiki/Eclipse_(software)

[39] Apache Tomcat https://es.wikipedia.org/wiki/Tomcat

[40] Fases de desarrollo software https://es.wikipedia.org/wiki/Desarrollo_por_etapas

[41] Herramienta de adaptación de imágenes http://romannurik.github.io/AndroidAssetStudio/icons-launcher.html#foreground.type=image&foreground.space.trim=1&foreground.space.pad=0&foreColor=607d8b%2C0&crop=0&backgroundShape=square&backColor=170404%2C100&effects=shadow

[42] Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal http://www.boe.es/boe/dias/1999/12/14/pdfs/A43088-43099.pdf

[43] Imagen Logo MySQL http://www.quantacell.com/wp-content/uploads/2014/06/logo-MySQL.png

[44] Servidor MySQL https://dev.mysql.com/downloads/mysql/

[45] **Figura 6.1** http://menteprincipiante.com/2011/06/instalar-y-configurar-mysql-en-windows-7-paso-a-paso/

[46] **Figura 6.2** http://jagonzalez.org/wp-content/uploads/2013/09/workbench13.png

[47] Servicio DNS https://www.noip.com/

[48] Expresiones MySQL y SQLite http://mysql.conclase.net/