



Universidad de Jaén

Escuela Politécnica Superior de Jaén

APLICACIÓN MÓVIL PARA GESTIONAR EL CUADERNO DE CAMPO Y REALIZAR ASESORAMIENTO EN EL OLIVAR

Autor: Ana Jiménez Navarrete

Grado en Ingeniería Informática

Director: Lidia Ortega Alvarado

Departamento de Informática

Fecha: 31/07/2024

Licencia CC





Universidad de Jaén
Escuela Politécnica Superior de Jaén
Departamento de Informática

Doña Lidia Ortega Alvarado, tutora del Proyecto Fin de Carrera titulado: **Aplicación móvil para gestionar el cuaderno de campo y realizar asesoramiento en el olivar**, que presenta Ana Jiménez Navarrete, autoriza su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, Julio de 2024

El alumno:

JIMENEZ
NAVARRETE
ANA -
21035525R

Firmado
digitalmente por
JIMENEZ NAVARRETE
ANA - 21035525R
Fecha: 2024.07.31
07:01:46 +02'00'

Ana Jiménez Navarrete

El tutor:

ORTEGA
ALVARADO
LIDIA -
24241033E

Firmado digitalmente
por ORTEGA
ALVARADO LIDIA -
24241033E
Fecha: 2024.07.31
09:37:48 +02'00'

Dña. Lidia Ortega Alvarado

Agradecimientos

Me gustaría mostrar mi agradecimiento a todas las personas que han contribuido positivamente en la realización de este Trabajo Fin de Grado. No obstante, quiero hacer algunas menciones especiales.

Al gran equipo docente de la Escuela Politécnica de Jaén, porque en mayor o menor medida cada uno ha contribuido en la construcción de este trabajo.

A mi tutora, Lidia, por darme la oportunidad y confianza necesaria para iniciar y finalizar este proyecto.

A mis abuelos, no han aportado conocimientos teóricos a este trabajo, pero sí han incluido valores básicos como la dedicación y humildad.

A mis padres y hermanas. Han vivido la desesperación, desilusión y agobio de este proyecto, pero me sorprende que en ningún momento hayan dudado de mí y siempre me han ofrecido su apoyo incondicional. Este trabajo también forma parte de vosotros, porque contiene la perfección de mi padre, la constancia de mi madre, la dureza de mi hermana Claudia, la alegría de mi hermana Elena y la picaresca de mi hermana Julia.

A todos, gracias.

Tabla de contenidos

CAPÍTULO 1 DOCUMENTACIÓN.....	21
1.1. Introducción.....	22
1.2. Motivación.....	22
1.3. Objetivos generales.....	23
1.4. Objetivos específicos.....	24
1.5. Contexto socioeconómico.....	25
1.5.1. Origen.....	25
1.5.2. El olivar en España.....	26
1.5.3. Necesidades del olivo.....	30
1.6. Antecedentes y estado del arte.....	32
1.6.1. Agricultura de precisión.....	35
1.6.2. Legislatura reguladora.....	36
1.6.3. Cuaderno de explotaciones agrícolas.....	39
1.7. Metodología de desarrollo.....	43
1.8. Definiciones y abreviaturas.....	46
1.9. Estructura de la memoria.....	47
CAPÍTULO 2 ANÁLISIS INICIAL.....	49
2.1. Introducción.....	50
2.2. Perfiles de usuarios en el sistema.....	50
2.3. Análisis de requisitos.....	51
2.3.1. Requisitos funcionales.....	52
2.3.2. Requisitos no funcionales.....	53
2.4. Documentación de requisitos.....	54
2.4.1. Casos de uso.....	55
2.4.2. Diagramas de secuencia.....	58
2.4.2.1. Diagramas de secuencia de la aplicación móvil.....	60

2.4.2.2. Diagramas de secuencia de la aplicación web.....	72
2.5. Análisis de tecnologías en sistemas operativos para móviles.....	79
2.6. Análisis de tecnologías y herramientas para el desarrollo web.....	81
2.6.1. Visual Studio Code.....	81
2.6.2. HTML.....	81
2.6.3. CSS.....	82
2.6.4. JavaScript.....	82
2.6.5. Vue.js.....	82
2.6.6. Node.js.....	83
2.7. Análisis de tecnologías en sistemas gestores de base de datos.....	83
2.8. Arquitectura final.....	85
2.9. Planificación temporal.....	86
2.10. Presupuesto.....	89
CAPÍTULO 3 DESARROLLO.....	93
3.1. Introducción.....	94
3.2. Primera iteración: Modelo de datos.....	95
3.2.1. Introducción.....	95
3.2.2. Nivel conceptual.....	96
3.2.3. Nivel lógico.....	98
3.2.4. Implementación.....	116
3.2.5. API.....	118
3.2.6. API en Oracle Cloud.....	119
3.2.7. Pruebas.....	122
3.3. Segunda iteración: Diseño de la interfaz de usuario.....	123
3.3.1. Introducción.....	123
3.3.2. Sketch.....	124
3.3.3. Wireframes.....	132
3.3.4. Mockups.....	140
3.4. Tercera iteración: Aplicación móvil.....	150

3.4.1. Android Studio.....	150
3.4.2. Bibliotecas.....	152
3.4.3. Implementación.....	153
3.4.3.1. Base de datos y repositorio.....	153
3.4.3.2. Consultas a la API.....	157
3.4.3.3. Clase general.....	160
3.4.3.4. Comienzo e inicio de sesión.....	163
3.4.3.5. Pantalla principal.....	165
3.4.3.6. Información de la parcela.....	168
3.4.3.7. Menú de operaciones según el rol del usuario.....	171
3.4.3.8. Visualización, creación, modificación y borrado de tratamientos para agricultores.....	172
3.4.3.9. Validación de tratamientos para el técnico.....	177
3.4.3.10. Visualización, inserción y borrado de labores y cosechas.....	180
3.4.3.11. Visualización, creación y borrado de imágenes y validación por parte del técnico.....	183
3.4.3.12. Visualización y creación de análisis de laboratorio.....	187
3.4.3.13. Históricos gráficos.....	188
3.4.3.14. Creación de informes y cuadernos de campo.....	191
3.4.4. Pruebas.....	195
3.5. Cuarta iteración: Aplicación web.....	196
3.5.1. Vue.js.....	196
3.5.2. Bibliotecas.....	197
3.5.3. Implementación.....	198
3.5.3.1. Estructura de vista y componentes.....	198
3.5.3.2. Inicio de sesión y cookies.....	200
3.5.3.3. Menú, pantalla principal y cierre de sesión.....	204
3.5.3.4. Mantenimiento de una entidad.....	207
3.5.3.5. Historial o detalle de un cuaderno.....	214
3.5.3.6. Problemas en forma de imagen.....	215

3.5.4. Pruebas.....	217
CAPÍTULO 4 CONCLUSIONES.....	221
4.1. Introducción.....	222
4.2. Trabajos futuros.....	222
CAPÍTULO 5 ANEXOS.....	225
5.1. Introducción.....	226
5.2. Manual de usuario para obtener la información geométrica de una parcela en QGIS.....	226
CAPÍTULO 6 REFERENCIAS BIBLIOGRÁFICAS.....	235
6.1. Bibliografía.....	236

Lista de imágenes

Imagen 1: Producción de aceite de oliva en el mundo, promedio entre 1964 y 2021 - Faostat [3].....	26
Imagen 2: Mejores 10 países en producción de aceite de oliva en 2021 (FAOSTAT [3]).....	27
Imagen 3 : Superficie y producción del olivar en España [4].....	28
Imagen 4 : Superficie de olivar en la provincia de Jaén [4].....	29
Imagen 5: Diferencia de rendimiento y producción entre regadío y secano [4].....	30
Imagen 6: Parámetros climáticos para el cultivo del olivar [8].....	32
Imagen 7: Proceso de la metodología Scrum.....	45
Imagen 8: Simbología en un caso de uso.....	56
Imagen 9: Diagrama de casos de uso de la aplicación móvil.....	57
Imagen 10: Diagrama de casos de uso de la aplicación web.....	58
Imagen 11: Simbología presente en los diagramas de secuencia.....	60
Imagen 12: Diagrama de secuencia para iniciar sesión en la aplicación móvil.....	61
Imagen 13: Diagrama de secuencia para ver listado de cuadernos en la aplicación móvil.....	62
Imagen 14: Diagrama de secuencia para el crear un cuaderno.....	63
Imagen 15: Diagrama de secuencia para ver el listado de parcelas en la aplicación móvil.....	64
Imagen 16: Diagrama de secuencia para ver la información de la parcela en la aplicación móvil.....	64
Imagen 17: Diagrama de secuencia para descargar y/o actualizar información en la aplicación móvil.....	65
Imagen 18: Diagrama de secuencia para ver históricos en la aplicación móvil.....	66
Imagen 19: Diagrama de secuencia para crear informes en la aplicación móvil...	66
Imagen 20: Diagrama de secuencia para ver, añadir, modificar y/o borrar tratamientos en la aplicación móvil.....	67

Imagen 21: Diagrama de secuencia para ver, añadir y borrar labores en la aplicación móvil.....	68
Imagen 22: Diagrama de secuencia para ver, añadir y borrar cosechas en la aplicación móvil.....	69
Imagen 23: Diagrama de secuencia para ver, añadir y borrar Imágenes en la aplicación móvil.....	70
Imagen 24: Diagrama de secuencia para ver, añadir análisis en la aplicación móvil	71
Imagen 25: Diagrama de secuencia para validar y retroalimentar tratamientos en la aplicación móvil.....	71
Imagen 26: Diagrama de secuencia para iniciar sesión en la aplicación web.....	72
Imagen 27: Diagrama de secuencia para ver, añadir, borrar y editar usuarios en la aplicación web.....	73
Imagen 28: Diagrama de secuencia para ver, añadir, borrar y editar parcelas en la aplicación web.....	74
Imagen 29: Diagrama de secuencia para ver, añadir, borrar y editar cuadernos en la aplicación web.....	75
Imagen 30: Diagrama de secuencia para ver, añadir, borrar y editar equipos en la aplicación web.....	76
Imagen 31: Diagrama de secuencia para ver, añadir, borrar y editar personal en la aplicación web.....	77
Imagen 32: Diagrama de secuencia para ver, añadir y borrar empresas en la aplicación web.....	78
Imagen 33: Diagrama de secuencia para listar problemas en imágenes en la aplicación web.....	79
Imagen 34: Diagrama de secuencia para mostrar un historial de un cuaderno en la aplicación web.....	79
Imagen 35: Arquitectura final del conjunto del sistema.....	86
Imagen 36: diagrama de Gantt con la estimación de duración del proyecto.....	88
Imagen 37: Diagrama Entidad-Relación.....	97
<u>Imagen 38: Módulos de una API REST en Oracle Cloud.....</u>	<u>120</u>

Imagen 39: Plantillas de un módulo en una API REST en Oracle Cloud.....	120
Imagen 40: Módulos de una API REST en Oracle Cloud.....	121
Imagen 41: Lógica de un controlador GET en una API REST en Oracle Cloud..	121
Imagen 42: Sketch del inicio de sesión en la aplicación móvil.....	125
Imagen 43: Sketch del listado de parcelas en la aplicación móvil.....	125
Imagen 44: Sketch de la información de una parcela en la aplicación móvil.....	126
Imagen 45: Sketch del menú de operaciones en la aplicación móvil.....	126
Imagen 46: Sketch del listado de cualquier operación en la aplicación móvil.....	127
Imagen 47: Sketch de la visualización de una operación en la aplicación móvil	127
Imagen 48: Sketch para añadir o editar una operación en la aplicación móvil....	128
Imagen 49: Sketch para ver histórico de los datos en la aplicación móvil.....	128
Imagen 50: Sketch para hacer informes en la aplicación móvil.....	129
Imagen 51: Sketch para listar los agricultores en la aplicación móvil.....	129
Imagen 52: Sketch para iniciar sesión en la aplicación web.....	130
Imagen 53: Sketch del menú de operaciones en la aplicación web.....	130
Imagen 54: Sketch del mantenimiento de una entidad en la aplicación web.....	131
Imagen 55: Sketch del historial relativo a un cuaderno en la aplicación web.....	131
Imagen 56: Sketch de la visualización de las imágenes en la aplicación web....	132
Imagen 57: Wireframe del inicio de sesión en la aplicación móvil.....	133
Imagen 58: Wireframe del listado de parcelas en la aplicación móvil.....	133
Imagen 59: Wireframe de la información de la parcela en la aplicación móvil....	134
Imagen 60: Wireframe del menú de operaciones en la aplicación móvil.....	134
Imagen 61: Wireframe del listado de cualquier operación en la aplicación móvil	135
Imagen 62: Wireframe de visualización de una operación en la aplicación móvil.....	135
Imagen 63: Wireframe para añadir o editar una operación en la aplicación móvil....	136
Imagen 64: Wireframe de histórico en la aplicación móvil.....	136
Imagen 65: Wireframe sobre la creación de informes en la aplicación móvil.....	137
Imagen 66: Wireframe del listado de agricultores en la aplicación móvil.....	137

Imagen 67: Wireframe del inicio sesión en la aplicación web.....	138
Imagen 68: Wireframe del menú de operaciones en la aplicación web.....	138
Imagen 69: Wireframe del mantenimiento de una entidad en la aplicación web.	139
Imagen 70: Wireframe del historial relativo a un cuaderno en la aplicación web	139
Imagen 71: Wireframe de la visualización de las imágenes en la aplicación web.....	140
Imagen 72: Mockups del inicio de sesión en la aplicación móvil.....	141
Imagen 73: Mockups del listado de parcelas en la aplicación móvil.....	141
Imagen 74: Mockups de la información de la parcela en la aplicación móvil.....	142
Imagen 75: Mockups del menú de operaciones en la aplicación móvil.....	143
Imagen 76: Mockups del listado de cualquier operación en la aplicación móvil..	143
Imagen 77: Mockups de visualización de la operación en la aplicación móvil.....	144
Imagen 78: Mockups añadir o editar una operación en la aplicación móvil.....	144
Imagen 79: Mockup de la interacción en la pantalla de añadir o editar una operación, entrada de fechas (izquierda) y guardar cambios(derecha) en la aplicación móvil.....	145
Imagen 80: Mockups de histórico en la aplicación móvil.....	146
Imagen 81: Mockups sobre la creación de informes en la aplicación móvil.....	146
Imagen 82: Mockups de la notificación al finalizar la creación de un informe en la aplicación móvil.....	147
Imagen 83: Mockups del listado de agricultores en la aplicación móvil.....	147
Imagen 84: Mockups del inicio sesión en la aplicación web.....	148
Imagen 85: Mockups del menú de operaciones en la aplicación web.....	148
Imagen 86: Mockups del mantenimiento de una entidad en la aplicación web...	149
Imagen 87: Mockups del historial relativo a un cuaderno en la aplicación web...	149
Imagen 88: Mockups de la visualización de las imágenes en la aplicación web.	150
Imagen 89: Diagrama UML de la base de datos y repositorio en la aplicación móvil	157
Imagen 90: Diagrama UML del servicio API de la aplicación móvil.....	158
Imagen 91: Diagrama UML del esquema general de cada actividad de la	

aplicación.....	161
Imagen 92: Diagrama UML de la primera actividad de la aplicación y el inicio de sesión.....	163
Imagen 93: Resultado de la primera actividad de la aplicación y el inicio de sesión.	165
Imagen 94: Diagrama UML con las actividades principales del agricultor y técnico..	166
Imagen 95: Resultado de la lista de parcelas (izquierda) y cuadernos (derecha) en la aplicación móvil.....	167
Imagen 96: Diagrama UML de la información de una parcela.....	170
Imagen 97: Resultado de la información de una parcela en la aplicación móvil..	171
Imagen 98: Diagrama UML de los menús de operaciones en la aplicación móvil.....	172
Imagen 99: Resultado de los menús de operaciones en ambos roles, agricultor y técnico, izquierda y derecha respectivamente.....	172
Imagen 100: Diagrama UML de la visualización, creación, modificación y borrado de los tratamientos por parte del agricultor.....	173
Imagen 101: Resultado de la visualización de tratamientos, de sus detalles y la modificación de sus parámetros.....	174
Imagen 102: Resultado del borrado de un tratamiento por parte del agricultor y su posterior subida.....	176
Imagen 103: Resultado de la creación de un nuevo tratamientos de tipo fertilizante	177
Imagen 105: Respuesta de la validación de un tratamiento.....	179
Imagen 106: Respuesta de la actualización de la base de datos local.....	180
Imagen 107: Diagrama UML de la visualización, creación y borrado de labores	181
Imagen 108: Resultado de la visualización, creación y borrado de labores.....	182
Imagen 109: Diagrama UML de la visualización, creación y borrado de cosechas...	182
Imagen 110: Resultados de la visualización, creación y borrados de cosechas.	183
Imagen 111: Diagrama UML de la visualización, creación y borrado de los	

problemas en imágenes.....	184
Imagen 112: Resultado de la visualización, creación y borrado de problemas en imágenes.....	186
Imagen 113: Resultado de la validación de una imagen problemática.....	186
Imagen 114: Diagrama UML de la visualización y creación de análisis.....	187
Imagen 115: Resultado de la visualización y creación de análisis de laboratorio	188
Imagen 116: Diagrama UML de los históricos.....	189
Imagen 117: Resultado de la lista de históricos y el gráfico de rendimientos y kilos.	190
Imagen 118: Resultado del gráfico de aplicaciones (izquierda) y calendario de operaciones (derecha).....	191
Imagen 119: Diagrama UML de la creación de informes.....	192
Imagen 120: Resultado del listado de informes, la solicitud de permisos para notificar y creación del fichero.....	193
Imagen 121: Resultado del informe de cosechas y su anterior notificación.....	194
Imagen 122: Resultado de la creación del cuaderno de campo.....	195
Imagen 123: Resultado de un inicio de sesión fallido en la aplicación web.....	202
Imagen 124: Resultado del menú y pantalla principal en la aplicación web.....	205
Imagen 125: Confirmación del cierre de sesión.....	207
Imagen 126: Agrupación y filtrado de cabeceras.....	209
Imagen 127: Resultado del mantenimiento de la entidad 'Usuario'.....	210
Imagen 128: Resultado de la ventana emergente de confirmación de borrado en la aplicación web.....	211
Imagen 129: Ventana emergente para insertar una parcela.....	214
Imagen 130: Historial recopilatorio de la información de un cuaderno.....	215
Imagen 131: Problemas representados mediante imágenes.....	216
Imagen 132: Vista del mantenimiento de la entidad 'Cuaderno'.....	218
Imagen 133: Nuevo proyecto y administrador de fuentes en QGIS.....	227
Imagen 134: Conectar nueva conexión en QGIS.....	228
Imagen 135: Añadir nueva capa en QGIS.....	229

Imagen 136: Parcelas del municipio de Villacarrillo en QGIS.....	229
Imagen 137: Seleccionar parcela en QGIS.....	230
Imagen 138: Exportar geometría de una parcela en QGIS.....	231
Imagen 139: Seleccionar el archivo que contiene la geometría de una parcela en la aplicación web.....	232

Lista de tablas

Tabla 1: Plazos y requisitos para la entrada en vigor progresiva del CUE.....	39
Tabla 2: Costes hardware.....	89
Tabla 3: Costes software.....	90
Tabla 4: Coste humano.....	91
Tabla 5: Amortización de los coste hardware y software.....	92
Tabla 6: Coste total del proyecto.....	92
Tabla 7: Resumen de la entidad Cuaderno.....	99
Tabla 8: Resumen de la entidad Usuario.....	99
Tabla 9: Resumen de entidad Tipo Rol.....	100
Tabla 10: Resumen de la entidad Municipio.....	100
Tabla 11: Resumen de la entidad Provincia.....	100
Tabla 12: Resumen de la entidad Personal.....	101
Tabla 13: Resumen de la entidad Equipo.....	101
Tabla 14: Resumen de la entidad Polígono.....	102
Tabla 15: Resumen de la entidad Parcela.....	103
Tabla 16: Resumen de la entidad Sistema Riego.....	104
Tabla 17: Resumen de la entidad Tipo Asesoramiento.....	104
Tabla 18: Resumen de la entidad Tipo Aire.....	104
Tabla 19: Resumen de la entidad Análisis.....	105
Tabla 17: Resumen de la entidad Tipo Material.....	105
Tabla 21: Resumen de la entidad Cosecha.....	106
Tabla 22: Resumen de la entidad Cliente.....	106
Tabla 23: Resumen de la entidad Operacion.....	107
Tabla 24: Resumen de la entidad Tipo Labor.....	107
Tabla 25: Resumen de la entidad Tratamiento.....	108
Tabla 26: Resumen de la entidad Tipo Tratamiento.....	108

Tabla 27: Resumen de la entidad Fitosanitario.....	109
Tabla 28: Resumen de la entidad FitosanitarioAses.....	110
Tabla 29: Resumen de la entidad Semilla.....	111
Tabla 30: Resumen de la entidad Postcosecha.....	112
Tabla 31: Resumen de la entidad Herramienta.....	112
Tabla 32: Resumen de la entidad Tipo Herramienta.....	113
Tabla 33: Resumen de la entidad Fertilización.....	113
Tabla 34: Resumen de la entidad Tipo Abono.....	114
Tabla 35: Resumen de la entidad Tipo Fertilizante.....	114
Tabla 36: Resumen de la entidad Riego.....	115
Tabla 37: Resumen de la entidad Imagen.....	115
Tabla 38: Tipos de datos en Oracle.....	117

Listado de bloques de código

Código 1: Inicio de las preferencias de la aplicación móvil.....	154
Código 2: Inicialización de la base de datos siguiendo el patrón Singleton.....	155
Código 3: Definición de métodos en la interfaz Dao.....	156
Código 4: Definición del servicio que accede a la API.....	158
Código 5: Peticiones a la API desde la aplicación móvil.....	159
Código 6: Interfaz de peticiones a la API.....	160
Código 7 : XML para definir un menú en Android.....	162
Código 8: Inicio de las preferencias de la aplicación móvil.....	162
Código 9: Iniciar y dirigir a la pantalla principal según el rol del usuario.....	164
Código 10: Implementación de una lista desplegable en la aplicación móvil.....	167
Código 11: Implementación de un cuadro de diálogo personalizado.....	179
Código 12: Implementación de la apertura de una aplicación externa y su recogida de datos.....	185
Código 13: Declaración de un gráfico combinado y circula con MPAndroidChart.....	191
Código 14: Resumen de la creación de un documento con la librería iText.....	193
Código 15: Renderizado del proyecto en Vue.....	199
Código 16: Contenedor de las páginas enrutadas.....	199
Código 17: Estructura de enrutamiento de páginas en Vue.....	200
Código 18: Formulario de inicio de sesión en la aplicación web.....	201
Código 19: Interceptores para comprobación de sesión en la aplicación web....	204
Código 20: Menú y cabecera en la aplicación web.....	205
Código 21: Implementación completa de una tarjeta para la pantalla principal...	206
Código 22: Renderizado de tarjetas en la aplicación web.....	206
Código 23: Definición de un cuadro de diálogo y la eliminación de cookies.....	207
Código 24: Implementación de una tabla con Devextreme.....	209

Código 25: Renderizado de columnas a través de una plantilla v-slot.....	211
Código 26: Implementación de un popup.....	212
Código 27: Plantilla para la creación de una ventana emergente personalizada	213
Código 28: Conversión y visualización de una imagen en base64.....	217
Código 29: Coloreado fondo de un registro de un DxDataGrid.....	218
Código 30: Tratamiento del contenido del archivo que contiene la geometría de una parcela.....	233

CAPÍTULO 1

DOCUMENTACIÓN

1.1. Introducción

Este presente documento constituye toda la documentación del proyecto **Aplicación móvil para gestionar el cuaderno de campo y realizar asesoramiento en el olivar.**

La estructura de capítulos y apartado de este documento se ha realizado siguiendo los criterios y recomendaciones que establece la norma UNE 157801:2007–“Criterios Generales para la elaboración de proyectos de Sistemas de Información”.

En este primer capítulo se presentan las especificaciones del presente Trabajo Fin de Grado, a lo largo del cuál se expondrá toda la información recopilada acerca del contexto que nos engloba, la causa del problema a tratar y el estado actual de la tecnología usada para resolver el mismo.

1.2. Motivación

En los últimos años hemos sufrido un cambio tecnológico enorme. Esto ha hecho que actualmente tengamos los suficientes recursos para almacenar grandes cantidades de datos o capacidad de cómputo suficiente para cálculos muy complejos.

La agricultura es la columna vertebral de cualquier sociedad y no nos cabe duda que es imprescindible, y al mismo tiempo olvidada por algunas administraciones. Nosotros queremos ayudar a su desarrollo y capacitar a sus profesiones de las herramientas que otros ámbitos ya usan.

También, queremos centrarnos en simplificar los procesos burocráticos y administrativos de los agricultores. Estos profesionales son expertos en entender la tierra, cuidar el ganado o explotar un cultivo, no en legislación ni en tareas ofimáticas. Debemos asumir que supone un gran cambio en la forma de trabajar de los productores agrícolas, por lo que tenemos que ser conscientes de facilitar al máximo todo el proceso con eficientes y eficaces sistemas informáticos.

Cabe destacar que debido al contexto que nos rodea queremos centrarnos en el motor de la provincia de Jaén, el olivar. Queremos devolver todo lo que nos ha dado este cultivo, en forma de conocimiento, con la finalidad de ayudar y facilitar el proceso productivo, al mismo tiempo que se aumenta el rendimiento.

1.3. Objetivos generales

En rasgos generales, este Trabajo Fin de Grado plantea un proyecto cuyo objetivo es mantener al agricultor informado en todo momento del estado de sus explotaciones olivareras. En España el olivar es un cultivo tradicional, por lo que se puede llegar a tener tareas muy mecanizadas o arraigadas, pero aun así inciertas o poco contrastadas. Actualmente, nos encontramos con un nivel tecnológico y científico muy elevado y al igual que la tecnología, la olivicultura también ha evolucionado con ella. Por lo tanto nos vamos a aprovechar de los avances tecnológicos y la agricultura de precisión para aumentar el rendimiento y facilitar las tareas dentro del olivar.

Nuestra propuesta es digitalizar las tareas que el agricultor realiza físicamente en el campo. Así, conseguimos la suficiente información como para tomar decisiones en consecuencia real de la explotación. Además, otra de nuestra finalidad es agilizar la comunicación entre el campo y la ciencia, entre el agricultor y el técnico agrónomo. La comunicación continua y fluida entre estas dos figuras en ocasiones puede ser difícil o inexistente. Por lo tanto, con nuestro sistema planteamos una comunicación directa y rápida, donde el técnico deberá evaluar y validar cada una de las actuaciones que el agricultor realice.

Por otro lado, la normativa (debidamente explicada en la Sección [1.6.2](#)) exige el correcto desarrollo del Cuaderno de Explotaciones Agrícolas (en adelante CUE) por parte del titular de la explotación o autorizado. La finalidad del CUE es la de incorporar y mantener las actividades relacionadas con la gestión agraria, es decir, lo referente a las actuaciones fitosanitarias, aplicación de fertilizantes o cosechas recolectadas, junto a información indispensable relativa a la administración de las explotaciones. Por este motivo, parte de la información recolectada por el agricultor será de utilidad para completar y actualizar el CUE.

Más adelante hablaremos de las especificaciones técnicas del proyecto, pero sí podemos pensar que todo sistema informático que trabaja con información deberá tener un almacenaje para ella, llamado base de datos. Esta base de datos será la columna vertebral del sistema y de la comunicación del mismo. Es por ello, que se plantea el desarrollo de una aplicación desplegada en la web para alimentar esta base de datos y ver en todo momento su contenido actualizado.

El desarrollo de los tres aspectos anteriores serán los principales puntos de apoyo del proyecto.

1.4. Objetivos específicos

En base a los tres pilares fundamentales de nuestro proyecto, descritos en la sección anterior, vamos a ser más específicos y concretar sobre nuestros objetivos. El siguiente listado expondrá los objetivos específicos generados a partir de los objetivos generales:

- Digitalizar las tareas del olivar:
 - Desarrollar una aplicación para que el agricultor pueda registrar cada tarea o labor que realiza en el campo.
 - Favorecer la movilidad del agricultor y técnico realizando una aplicación móvil fácil de usar y aprender.
 - Mantener una comunicación fluida entre el agricultor y el técnico para consultar y resolver problemas fácilmente.
 - Tener en cuenta el contexto del agricultor, es decir, limitaciones de cobertura o usabilidad de aplicaciones.
- Generar el CUE digital:
 - A partir de la información generada por el agricultor y técnico, permitir que el CUE se pueda crear desde la aplicación móvil.
 - Cumplir la legislación relacionada con el CUE para despreocupar al agricultor y técnico.
 - Generar informes históricos relacionados con los trabajos realizados por el agricultor.
- Gestionar la información del sistema

- Realizar una aplicación web basada en la nube con el objetivo de facilitar la movilidad y portabilidad de este objetivo.
- Un responsable deberá alimentar la base de datos para que el agricultor y técnico tengan toda la información que se visualiza en sus aplicaciones.
- Visualizar la información registrada para tener constancia de los datos existentes.

1.5. Contexto socioeconómico

La olivicultura goza de una larga tradición en la provincia de Jaén, convirtiéndose en la primera opción de cultivo en la provincia. A pesar de tener muy interiorizada la cultura olivarera nos podemos preguntar algunas cuestiones, ¿por qué Jaén? o ¿por qué el olivo y no otro cultivo?.

1.5.1. Origen

En primer lugar, nos vamos al origen, en concreto, a la etapa del Neolítico. Aunque no se conoce con certeza el verdadero origen geográfico, este se atribuye a la zona conocida como Asia Menor. Si nos centramos en España es en el año 1050 a.C cuando se introdujo el olivo, a través de las colonias fenicias, aunque no alcanzó notable desarrollo. Tras el fin de la tercera guerra púnica (149-146 a.C), la victoria romana aconteció la presencia del olivo en la región mediterránea, hasta entonces controlada por Cartago. Esta expansión se produjo y aceleró gracias a la artimaña de Roma en utilizar el olivo como arma pacificadora, consiguiendo así el asentamiento de poblaciones en tierras conquistadas.

Posteriormente, con la presencia de los árabes en el sur de la antigua Hispania, se introdujeron variedades y los mismos influyeron en la difusión del cultivo. Hasta el punto que vocablos castellanos como aceituna, aceite o acebuche¹ tienen raíz árabe [1]. No obstante, gracias a la arqueobotánica se ha podido demostrar que estas plantas emblemáticas crecían en el litoral

¹ El acebuche es cómo se conoce al olivo en estado silvestre o salvaje. Su forma es de arbusto o matorral, sus tallos son más recios y robustos y su fruto es inferior en dimensión y cantidad que los del olivo. De ahí, el término 'el olivo se está acebuchando' olivo dejado de cultivar o trabajar[].

mediterráneo de la Península Ibérica en estado silvestre mucho antes de su domesticación [2].

El descubrimiento de América en el siglo XV hizo posible que el olivo se expandiera por un nuevo continente. Actualmente, podemos ver cómo países tan distantes en culturas y orígenes como Australia, China o Brasil cultivan el olivo. Debido a la modernización del cultivo, derivado del avance tecnológico y mecánico, aparecen explotaciones solamente dedicadas a la producción intensiva. Se propicia el aumento del rendimiento en la olivicultura tradicional y los agricultores aumentan sus conocimientos técnicos en la profesión. Este hecho se puede ratificar con la siguiente imagen (Imagen 1), donde se muestra la producción de aceite por país utilizando un promedio desde el año 1964 hasta el 2021.

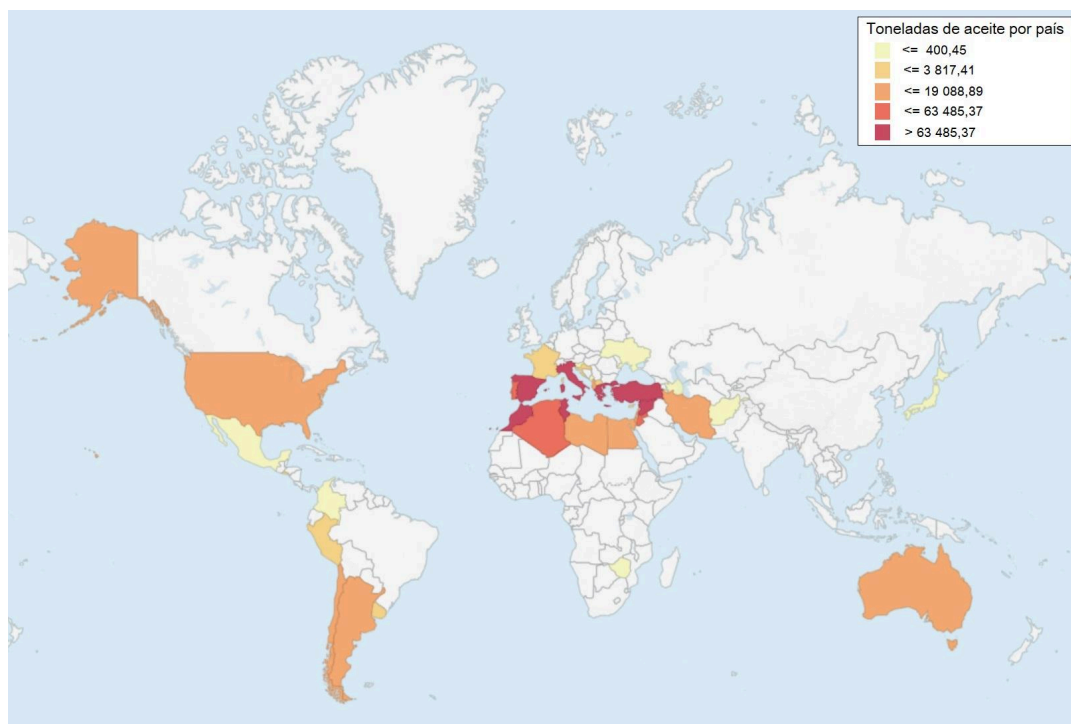


Imagen 1: Producción de aceite de oliva en el mundo, promedio entre 1964 y 2021 - Faostat [3]

1.5.2. El olivar en España

España lidera la lista de los mayores productores de aceite del mundo, así como también lo hace en superficie y comercio exterior. Como consecuencia, el sector de la olivicultura no solo tiene un gran impacto en la economía española,

sino que también tiene una gran repercusión social, ambiental y territorial. En la gráfica que encontramos en la Imagen 2 se refleja visualmente la diferencia entre España y el resto de países del mundo, en concreto, España produce 1.492.069 toneladas de aceite e Italia, inmediatamente después, produce 338.631 toneladas.

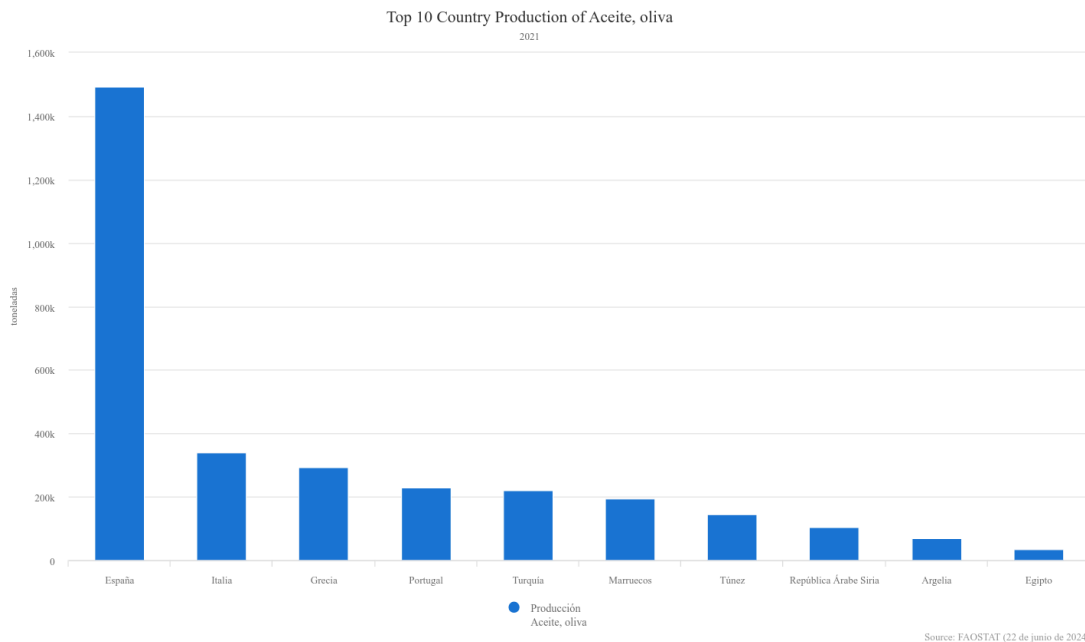


Imagen 2: Mejores 10 países en producción de aceite de oliva en 2021
(FAOSTAT [3])

Si hablamos de números, el sector puede llegar a generar directamente más de 32 millones de jornales por campaña en España. Siendo en muchos municipios rurales su actividad principal. En estos municipios se forma un tejido industrial y comercial alrededor del olivar que lo hace directamente el principal motor económico.

El desigual reparto de este cultivo igualmente se evidencia en el territorio español, tan solo no son productoras las Comunidades Autónomas de Asturias y Cantabria (ver Imagen 3).

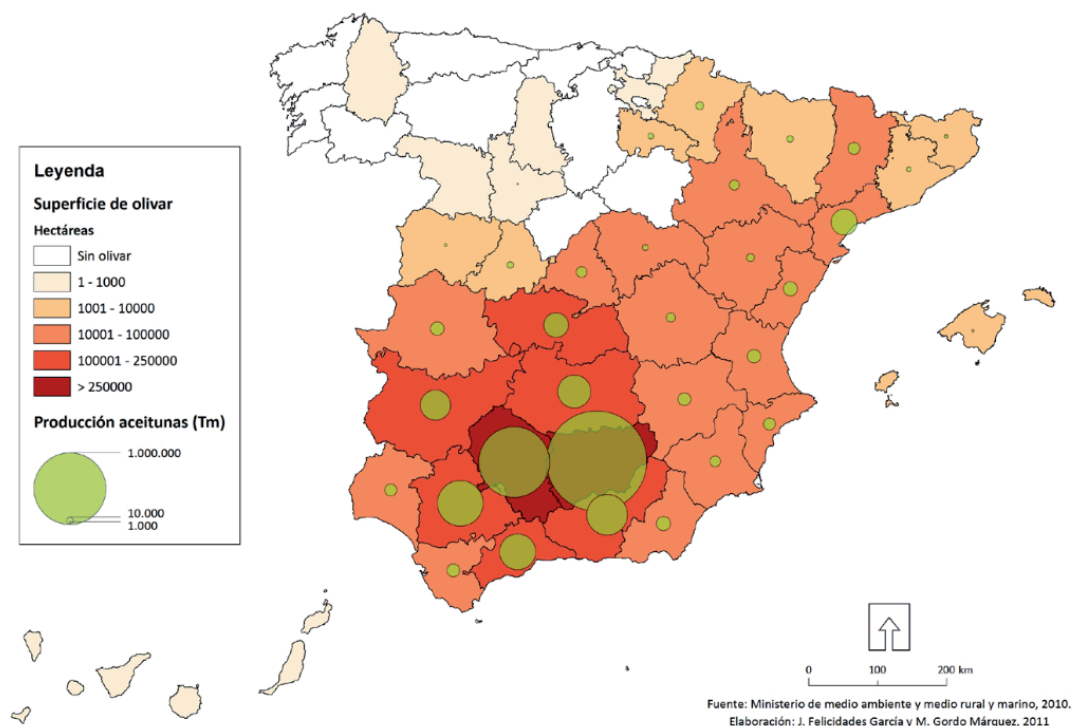
Superficie y producción de olivar para almazara en España, 2008

Imagen 3 : Superficie y producción del olivar en España [4]

Cabe destacar que no solo existe el olivar de almazara, es decir, olivares cuyas aceitunas se exprimen para obtener aceite. Sino que también en España existe lo que se llama olivar de aceituna de mesa y de doble aptitud, esta última válida para aceite y aceituna de mesa. De las 2.768.266,67 hectáreas de superficie dedicadas al olivar, el 88,76% se usa para aceite, el 3,01% para aceituna de mesa y el 8,23% para la aceituna híbrida [5].

Son las provincias de Sevilla, Badajoz y Cáceres con el 51%, 21% y 15%, respectivamente, las punteras en la aceituna de mesa. Por el contrario, Jaén con 568.600 hectáreas dedicadas a la aceituna de almazara, mayoritariamente de la variedad picual, sitúan a la provincia en la primera posición del ranking nacional, acaparando el 25% del total nacional. En el pódium le siguen provincias como Córdoba, Granada y, de nuevo, Badajoz. Si ampliamos la escala geográfica y nos centramos en Jaén (ver Imagen 4), el 92% de la superficie dedicada a cultivo en la provincia está ocupada por olivos. Definiéndolo como un monocultivo exclusivo y excluyente. La mayor concentración se localiza en el eje que conforma Alcaudete y Beas de Segura, destacando municipios como la capital, Martos,

Porcuna, Torredelcampo y Torredonjimeno. Le sigue en importancia la comarca de La Loma, con Úbeda y Baeza. Otros núcleos importantes son Alcaudete y Alcalá la Real en la comarca Sierra Sur.

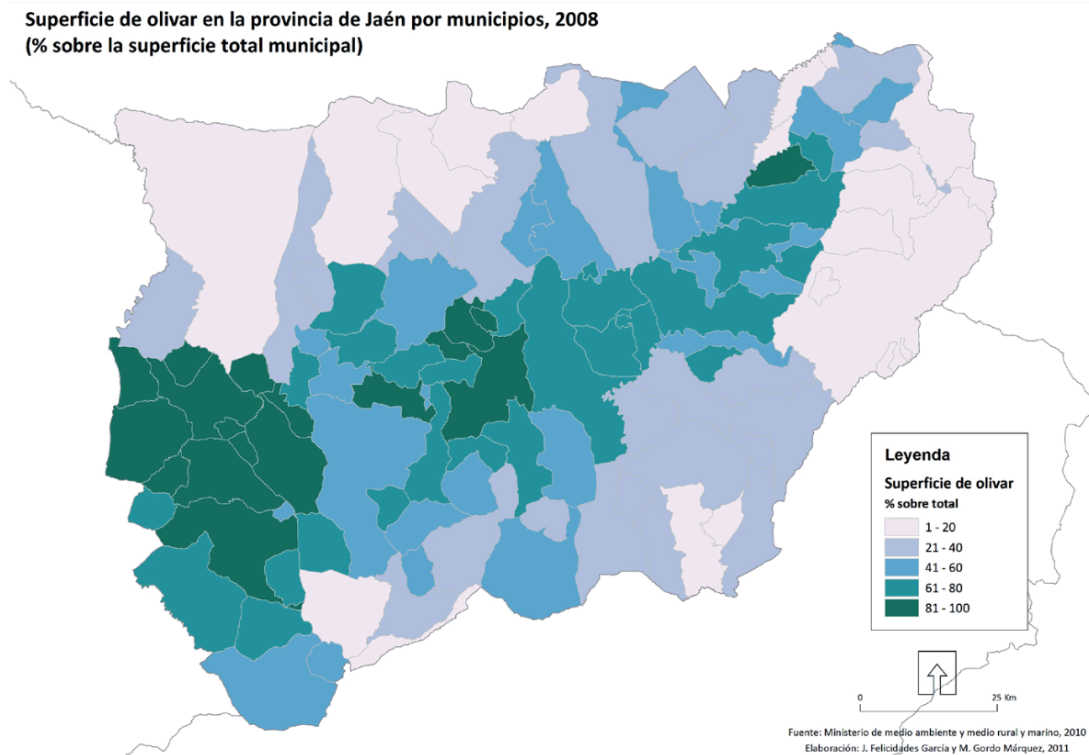


Imagen 4 : Superficie de olivar en la provincia de Jaén [4]

El olivar jienense no ha sido ajeno al avance técnico de los últimos tiempos, haciendo posible una modalidad intermedia o mixta, a medio camino entre la tradicional y la nueva agricultura. Se caracteriza por la permanencia de los cultivos tradicionales, pero incorporando nuevas técnicas de producción y sustituyendo plantaciones por otras más jóvenes. Una de las tendencias crecientes es la apuesta por el regadío, tradicionalmente no era una práctica frecuente pues ha sido un cultivo de secano muy bien adaptado a condiciones de sequía. Sin embargo, esta situación está cambiando ya que la aplicación adicional de agua incrementa notablemente la producción. Investigadores señalan que las necesidades de agua del olivo están entre 1.000 y 2.000 m^3/ha , cantidad muy por debajo de lo que requiere otro cultivo. Sus efectos son significativos, en la Imagen 5 se ve la diferencia de producción y rendimientos entre superficies de regadío y secano en las mismas regiones [4].

AÑO	ÁMBITO	SUPERFICIE (HAS.)				RENDIMIENTO (KG. /HA)		PRODUCCIÓN (T.)
		TOTAL	SECANO	REGADÍO				
				TOTAL	%	SECANO	REGADÍO	
1998 ⁽¹⁾	Jaén	558.140	444.330	113.810	20,39	2.826	4.480	1.680.964
	Andalucía	1.365.119	1.177.168	187.951	13,77	2.428	4.101	3.389.312
	España	2.221.902	1.989.903	231.999	10,44	1.731	3.800	4.020.662
2008 ⁽²⁾	Jaén	568.600	387.631	180.969	31,83	2.910	4.742	1.967.423
	Andalucía	1.406.652	1.114.559	292.093	20,76	2.572	4.477	4.064.004
	España	2.280.579	1.912.668	367.911	16,13	1.949	4.206	5.088.942

Fuente: MAPA, 2001 (1) y MARM, 2010 (2). Elaboración: Gordo, 2011.

Imagen 5: Diferencia de rendimiento y producción entre regadío y secano [4]

1.5.3. Necesidades del olivo

Lógicamente, la distribución geográfica del olivo es parte causada por su propia naturaleza biológica. El olivo es característico de un clima templado-cálido propio de la cuenca mediterránea. El efecto de las condiciones climáticas sobre el cultivo puede ser directo (heladas) o indirecto (déficit hídrico). La comprensión del efecto de los factores ambientales sobre el olivo es fundamental para la planificación de nuevas plantaciones o gestión de las ya existentes.

Si se quiere obtener el máximo rendimiento y calidad en el fruto, hay que entender y conocer su estructura vegetal y reproductiva a la perfección. El olivo necesita un invierno suave con temperaturas próximas a 0 grados, ya que en este periodo se produce un reposo vegetativo, es la etapa donde las yemas brotan y dan lugar las inflorescencias². Sin embargo, también requiere de veranos largos, secos y cálidos para sacar adelante el crecimiento del fruto. En resumen, el olivo es fructífero³ en un intervalo de temperaturas de entre los -7 y 40 grados.

En casos extremos, donde las **temperaturas** mínimas son de -12 grados estaríamos hablando de heladas que pueden causar lesiones y daños severos en el olivo. Cuando se comienza a ver los efectos negativos de la helada se dice que "el olivo está movido". Estos daños son más significativos aún en plantaciones de

² Disposición de las flores en las ramas

³ Que produce fruto

corta edad. La efectividad de la helada se ve reflejada en la muerte de las yemas de flor, frutos con menor tamaño, deformación en las futuras hojas o aparición de hongos. En definitiva, causando una merma de la producción y la calidad del aceite, aunque no llegue a su muerte total. Los daños descritos serán tan menores cuanto la duración y la brusquedad de las bajas temperaturas sea [6].

Otro de los aspectos a conocer y mejorar es el **suelo**, ya que este conocimiento es imprescindible para establecer enriquecedores planes de abonado y/o manejar adecuadamente el riego si existe. Los suelos se componen de una matriz sólida (constituida por partículas minerales y materia orgánica), una fase líquida y aire. Cada suelo, de acuerdo con el tamaño y naturaleza de sus componentes, presentan unas propiedades físicas, químicas y biológicas que determinan su aptitud agrícola.

A parte del efecto positivo de las **precipitaciones** sobre la acumulación de agua en el suelo, la lluvia puede tener efectos negativos. El primero está relacionado con el encharcamiento después de fuertes lluvias en los suelos mal drenados y el segundo es la reducción de la polinización debido al lavado por las gotas de lluvia y al acotamiento de la viabilidad del polen cuando se humedece [7].

Sabemos cuáles son las mejores condiciones del olivar, pero ¿dónde las podemos encontrar?. En el libro "El cultivo del olivar" [8] se indica que el municipio de Torredonjimeno (Jaén) reúne las mejores condiciones para el olivo, invierno suave, sin peligro de heladas, verano largo y caluroso sin exceso y pluviometría alta. En un término medio está la estación de Cordobilla (Córdoba) que tiene un invierno que, aunque es más frío que el de Torredonjimeno, no ofrece limitaciones para el cultivo del olivo, sin embargo, el verano es más caluroso y la pluviometría más baja reduciendo así la producción del olivar en seco. En cuanto a los extremos, tenemos el clima de Mora de Toledo pues es un ejemplo de límite por temperaturas bajas con inviernos muy fríos. En contraposición está la región de Sfax, Túnez, donde se cultiva el olivo en un clima semidesértico con pluviometrías anuales inferiores a los 200 mm. A continuación, se visualiza una tabla donde se reúnen los datos cuantitativos que validan las afirmaciones anteriores (Imagen 6).

<i>Parámetros climáticos de localidades con cultivo de olivar</i>			
	<i>Localidades</i>		
	<i>Mora de Toledo</i>	<i>Torredonjimeno</i>	<i>Cordobilla</i>
Temperatura			
<i>Invierno</i>			
t' anual	-7,6	1,2	-4,9
Mes más frío	Diciembre	Diciembre	Diciembre
t' del mes más frío	-7,4	2,5	-1,6
t del mes más frío	-0,6	6,9	2,0
T del mes más frío	12,1	11,4	14,8
<i>Primavera-Otoño</i>			
e (meses)	4,0	8,0	5,1
E (meses)	5,8	12,0	7,4
Semestre más cálido	May.-Oct.	May.-Oct.	May.-Oct.
M (°C)	31,0	28,3	33,1
Balance de agua			
Pluviometría	353	626	487
Meses húmedos	4	6	5
Meses intermedios	4	3	2
Meses secos	4	3	5

t' = temperatura media de las mínimas absolutas, t = temperatura media de las mínimas.

T = temperatura media de las máximas.

e = duración de la estación con las t', correspondientes a cada mes, superiores a 7°C.

E = duración de la estación con las t', correspondientes a cada mes, superiores a 2°C.

M = Media de las máximas del semestre más cálido.

Mes seco: $I_h < 0,50$; Mes intermedio: $0,50 < I_h < 1,00$; Mes húmedo: $I_h > 1,00$.

$I_h = P/ETP$, siendo P = precipitación media mensual y ETP = evapotranspiración potencial media mensual.

Según Elías y Ruiz (1977).

Imagen 6: Parámetros climáticos para el cultivo del olivar [8]

1.6. Antecedentes y estado del arte

Es fácil reconocer el avance tecnológico en la agricultura, solo hace falta ver el tipo de maquinaria utilizada antes y ahora, la forma de realizar las labores o la manera de obtener el producto final. Además de ello, actualmente se han sumado leyes y decretos que favorecen la sostenibilidad del medio ambiente. Sin embargo, nos encontramos con unas administraciones públicas que disponen de una gran cantidad de datos, proporcionados por los propios agricultores, dispersos en estos momentos.

Una buena práctica es unificar toda la información de manera sistemática, esto produciría indudables sinergias entre administraciones y, al mismo tiempo, se

simplificará la labor de los productores agrarios. Esto es lo que se pretende cumplir con el Real Decreto 1024/2022 [\[9\]](#).

Actualmente constamos del Sistema de Información de Explotaciones Agrícolas y Ganaderas (en adelante SIEX). Herramienta que consta de un conjunto de base de datos y registros administrativos interconectados, en el que se integran el Registro General de Explotaciones Ganaderas, el Registro Nacional de Organizaciones de Productores y Asociaciones de Organizaciones de Productores, el Registro General de Operadores de Producción Ecológica, el Sistema Nacional de Información de Razas o la base de datos PROLAC. Otra de las herramientas existentes es el Registro de Explotaciones Agrarias, registro electrónico establecido y gestionado por las comunidades autónomas. Por último, está el CUE o Cuaderno de Explotación Agraria que pretende aunar la actividad agrícola de las explotaciones con cierta información técnica.

Aquí viene la novedad, la norma exige que el **cumplimiento del CUE** se realice por medios electrónicos, excluyendo la posibilidad de utilizar el papel. Y este es nuestro objetivo, ayudar a digitalizar esta herramienta y aprovechar su función para ofrecer otros servicios. Los contenidos que se quieren en el mismo se especifican en el Apartado [1.6.3](#), pero sí consideramos importante explicar qué trabajos se realizan en el olivo de almazara antes de analizar el contenido del CUE. Podemos clasificar los trabajos en:

- **Manejo del suelo:** el contenido del agua en el suelo dependen de su naturaleza (profundidad y capacidad de retención). En el periodo de noviembre-marzo la actividad vegetativa es escasa por las bajas temperaturas y la pluviometría es superior al consumo del agua por el cultivo. Es en ese momento cuándo se acumula agua en el suelo. El objetivo de las labores en el suelo es conservar el agua y reducir las pérdidas por evaporación directa. Una de las técnicas que se utilizan es el laboreo, en la actualidad, es el cultivador de brazos flexibles. Cuya misión es preparar el suelo para infiltrar el agua de lluvia invernal y eliminar las malas hierbas, alcanzando entre 15-20 cm de profundidad. En primavera cuando las malas hierbas están desarrolladas, se emplea la grada de

discos. Este apero⁴, al voltear el suelo, provoca grandes pérdidas de agua por evaporación, dando lugar a la compactación del suelo en profundidad. Ya en verano, cuando el suelo está totalmente seco, se realizan labores superficiales con gradas de púas o rastras, cuya misión es pulverizar el suelo y tapar las grietas. También en verano cada vez es más común barrer o soplar los suelos para eliminar las hojas caídas, procediéndose a continuación a compactar el suelo utilizándose para ello un rulo compactador liso.

- **Fertilización:** el abonado es una de las prácticas más frecuentes en agricultura, pues tiene por objetivo satisfacer las necesidades nutritivas de las plantas cuando los nutrientes necesarios para su crecimiento no son aportados en cantidades suficiente por el suelo. Las plantas perennes y leñosas, como el olivo, se diferencian de las anuales en que aquellas permanecen vivas durante mucho más tiempo que las últimas, por lo que deben disponer de reserva que les permitan sobrevivir incluso bajo condiciones desfavorables. El abonado no es la única variable que controla la producción del olivar, ya que la falta de criterio al establecer un plan de fertilización afecta negativamente. Un abonado racional debe aportar tan solo los elementos que requieran los árboles en un momento determinado y, únicamente, cuando existan pruebas de que esos elementos son precisos.
- **Riego:** el olivar es un cultivo tradicionalmente de secano, sin embargo, se ha comprobado experimentalmente que el riego aumenta considerablemente el rendimiento del olivar. La finalidad del riego consiste en evitar que el contenido de agua del suelo alcance un nivel umbral por debajo del cual el cultivo sufre déficit hídrico y la producción merme. Existen técnicas de programación de riesgos que permiten calcular cuándo regar y qué dosis aplicar. Una de esas técnicas es la llamada balance de agua, que consiste en calcular las variaciones en el contenido de agua del suelo como la diferencia entre las entradas y las salidas de agua de la parcela.
- **Fertirrigación:** la fertirrigación consiste en aplicar a la planta los abonos disueltos en el agua de riego, generalizado en el uso de riegos localizados.

⁴ Instrumento que se emplea en la labranza

Un olivar de secano tiene un mayor volumen de suelo al tener una distribución más homogénea, por el contrario, las raíces de los árboles regados se concentran principalmente en las zonas afectadas por el aporte del agua. Por este motivo, aportando los abonos disueltos en el agua de riego se consigue localizar los nutrientes directamente en las zonas con mayor densidad y actividad radical. Esta técnica da la posibilidad de aplicar los abonos tantas veces como sea necesario, sin que ello signifique un agravante económico como en otras técnicas.

- **Poda:** se entiende por poda la serie de operaciones por las que se modifica la forma natural de su vegetación, vigorizando o restringiendo el desarrollo de las ramas con el fin de darle forma y conseguir la máxima productividad e, incluso, renovar o restaurar el árbol. La correcta realización de esta técnica exige al agricultor: conseguir un buen equilibrio hoja/raíz y asimismo que la relación hoja/madera sea la mayor posible; es necesario cortar las ramas envejecidas para que sean sustituidas por otras jóvenes y las cortezas tanto del tronco como de las ramas principales deben estar protegidas de la insolación para evitar quemaduras.
- **Recolección:** labor cumbre del proceso de producción, uno de los momentos más importantes del olivar, la recolección de la aceituna. Se realiza entre los meses de octubre y febrero, aunque con diferencias según la región, clima y/o variedad. Las características de la explotación y de sus olivos van a marcar el sistema de recolección, desde el tradicional vareo hasta técnicas modernas de vibración. El vareo consiste en dar con una vara en las ramas del olivo con el objetivo de que la aceituna caiga sobre una manta o mantón. Las técnicas de vibración están más extendidas en la actualidad y consisten en agitar mediante el uso de maquinaria las ramas o troncos del olivo [8].

1.6.1. Agricultura de precisión

Todo el conocimiento que se adquiere del cultivo del olivar, ya sea sobre su anatomía, sobre las labores a realizar o las mejoras técnicas, tienen como único objetivo aumentar la producción y calidad. Pero no solo es una demanda del propio agricultor, sino que también la población hace crecer su demanda hacia la calidad.

Esta evolución se obtuvo mediante la generación de nuevas tecnologías, como por ejemplo, semillas mejoradas, insumos modernos, agroquímicos más eficientes, nuevas maquinaria agrícola para preparar, sembrar, cultivar, cosechar y procesar los productos agrícolas. A medida que las prácticas agrícolas mejoran, surgen nuevos desafíos, principalmente con respecto a la sostenibilidad ambiental y económica del proceso de producción.

Para ello, a partir de la década de los 70, se comenzó a dibujar un nuevo concepto de agricultura debido. A finales de los años 90 y principios de los 80 se liberó el sistema de posicionamiento global por satélite (GPS) para uso civil, permitiendo desarrollar equipos inteligentes con el objetivo de minimizar el coste de la producción. A ese conjunto de procesos y sistemas aplicados se los denomina **Agricultura de Precisión**.

Dentro de la agricultura de precisión existen los siguientes agentes: recolección de datos, procesamiento e interpretación de la información y toma de decisión mediante la aplicación de insumos. Las Tecnologías de la Información y Comunicación (TIC) más comunes son: sistemas de información geográfica (SIG), procesamiento de imágenes, machine learning⁵, GPS, redes de sensores, drones, teledetección, etc.

En definitiva, la agricultura de precisión es un conjunto de técnicas orientadas a optimizar el uso de insumos agrícolas en función de la cuantificación de la variabilidad espacial y temporal. La agricultura de precisión no solamente consiste en medir la variabilidad existente, sino también en la adopción de buenas prácticas administrativas. La diferencia con la agricultura tradicional se ve reflejada en la posibilidad de identificar, cuantificar y mapear esa variabilidad [10].

1.6.2. Legislatura reguladora

En la introducción del Apartado [1.6](#) hemos descrito las distintas herramientas que existen en el ámbito de la agricultura, pero es el CUE el sistema referente en todo nuestro proyecto. El CUE está sujeto a una serie de normativas, leyes de

⁵ Algoritmos y modelos estadísticos que utilizan los sistemas de computación con el fin de llevar a cabo tareas sin instrucciones explícitas, en vez de basarse en patrones e inferencias.

ámbito nacional y autonómico, por la que se rige la información contenida en él. Dicha normativa es la siguiente:

- Real Decreto 1311/2012, de 14 de septiembre por el que se establece el marco de actuación para conseguir un uso sostenible de los productos fitosanitarios [11].
- Orden APA/326/2007 de 9 de febrero, por la que se establecen las obligaciones de los titulares de explotaciones agrícolas y forestales en materia de registro de la información sobre el uso de productos fitosanitarios [12].
- Real Decreto 261/1996, de 16 de febrero, sobre protección de las aguas contra la contaminación producida por los nitratos procedentes de fuentes agrarias [13].
- Decreto 36/2008, de 5 de febrero, la Consejería de Presidencia de la Junta de Andalucía por el que se designan las zonas vulnerables y se establecen medidas contra la contaminación por nitratos de origen agrario [14].

Durante el desarrollo de este presente TFG se presentó una actualización del CUE propuesto por la Junta de Andalucía, elaborado por la Dirección General de la Producción Agrícola y Ganadera de la Consejería de Agricultura, Pesca, Agua y Desarrollo Rural, en coordinación con la Dirección General de Ayudas Directas y Mercados. Esta actualización del CUE es motivado principalmente por tres Reales Decretos:

- Real Decreto 1054/2022, del 27 de diciembre, por el que se establece y regula el Sistema de información de explotaciones agrícolas y ganaderas y de la producción agraria, así como el Registro autonómico de explotaciones agrícolas y el Cuaderno digital de explotación agrícola [9].
- Real Decreto 1048/2022, del 27 de diciembre, sobre la aplicación, a partir de 2023, de las intervenciones en forma de pagos directos y el establecimiento de requisitos comunes en el marco del Plan Estratégico de la Política Agrícola Común, y la regulación de la solicitud única del sistema integrado de gestión y control [15].
- Real Decreto 1051/2022, de 27 de diciembre, por el que se establecen normas para la nutrición sostenible en los suelos agrarios [16].

La entrada en vigor de la nueva Política Agraria Común (en adelante PAC) proveniente de Europa, que se aplicará a partir de 2023 hasta 2027, trae consigo cambios importantes en los sistemas de gestión de información. Requiriendo un adecuado proceso de seguimiento y gestión de explotaciones agrícolas y ganaderas para asegurar que se realiza de manera objetiva y completa. Es decir, la Orden APA/204/2023 publicada en el Boletín Oficial del Estado (en adelante BOE) el 4 de marzo de 2023 [17] dictamina:

1. El titular de la explotación o sus autorizados deberán completar el contenido mínimo necesario del Registro Autonómico de Explotaciones y del Cuaderno Digital de Explotaciones Agrícolas. Esta información deberá actualizarse en el sistema informático central del Ministerio de Agricultura, Pesca y Alimentación mencionado en los artículos 6.5 y 9.2 del Real Decreto 1054/2022 del 27 de diciembre.
2. Establecer el alcance y la cronología para la integración de los conjuntos de datos previstos en los registros de la disposición adicional tercera del Real Decreto 1054/2022 del 27 de diciembre, en el sistema de información de explotaciones agrícolas y ganaderas y de la producción agraria conforme al artículo 4 de dicho real decreto.

La puesta en marcha de la ley fue modificada por el Real Decreto Ley 4/2023 [18], del 11 mayo del 2023. Publicado por motivo del contexto de daños derivados de la sequía y elevadas temperaturas, sumado a la grave situación generada por la guerra de Ucrania. Esta medida intenta dar apoyo a los titulares de las explotaciones agrarias que han visto agravada la seguridad alimentaria y la economía del sector. Como consecuencia se establecen medidas de gestión y apoyo para los abastecimientos y regadíos situados en territorios afectados por la sequía.

Teniendo en cuenta las modificaciones de la entrada en vigor del CUE mediante el Real Decreto Ley 4/2023 mencionado anteriormente, el Ministerio de Agricultura, Pesca y Alimentación decide sacar una nota aclaratoria el 1 de agosto de 2023 [19]. La cual nos desglosa la siguiente información. La implantación del CUE se realizará en 4 fases de forma progresiva y escalonada. Las explotaciones estarán sujetas a una determinada fase dependiendo de su extensión,

intensificación y el tipo de información a declarar. En la Tabla 1 se sintetizan las fases para su mejor comprensión.

Tipo de explotación	Características	Obligatoriedad
Pequeña	Superficie agraria total menor o igual a 5 hectáreas. Siempre y cuando tengan una superficie de regadío menor o igual a 1 ha, excepto invernaderos. Dispongan solamente pastos, siempre que no realicen aplicaciones de fertilizantes en dichas superficies.	Voluntario el CUE digital o mantener los registro en papel que determine la normativa sectorial.
Mediana	Resto de explotaciones, es decir, que no sean catalogadas como "grandes" y superficies de pastos, sólo si son objeto de fertilización.	1 de septiembre de 2025
Grande	Superficie total, sumando tierras de cultivo y cultivos permanentes, superior a 30 hectáreas. Sobre el total de la superficie de tierras de cultivo y de cultivo permanente, tener más de 5 hectáreas de regadío. Cultivos bajo invernadero.	1 de septiembre de 2024

Tabla 1: Plazos y requisitos para la entrada en vigor progresiva del CUE

1.6.3. Cuaderno de explotaciones agrícolas

Todas la legislación mostrada y explicada en la sección anterior justifica el contenido que aparece en el CUE. En concreto, en el Anexo II de la Orden APA/204/2023 [17] aparece explicado el contenido mínimo del mismo. Ampliado por el Real Decreto 1048/2022 [9] y 1051/2022 [15], teniendo en cuenta el nuevo modelo actualizado. Además, la Junta de Andalucía nos proporciona una guía para su adecuado cumplimiento.

Deberá rellenarse un cuaderno por año, si el cultivo se desarrolla dentro de un mismo año, como el algodón, o bien por campaña si se trata de un cultivo que abarque dos años, como la fresa u olivo.

A continuación, se detallarán los distintos epígrafes. Toda la información se recoge en una guía que propone la Junta de Andalucía [20] y de las nuevas actualización nos informamos en una noticia que publica la Junta de Andalucía [21]. El CUE se divide en los siguientes epígrafes:

1. **Información general.** El titular de la explotación agraria es aquel que asume el riesgo total de la explotación. Se entiende por explotación agraria aquella unidad técnico económica de la que se obtienen productos agrarios, mediante procesos mecánicos y el uso de mano de obra humana.
 - a. Datos generales de la explotación. Principalmente se dedica a exponer los datos personales del titular o representante de la explotación. El número del Registro de Explotaciones Autonómico en determinadas comunidades no está implementado, luego no siempre es rellenado (en Andalucía no existe).
 - b. Personas o empresas que intervienen en el tratamiento con productos fitosanitarios. Todas las personas o empresas dedicadas a trabajar con los productos aplicados serán anotados en esta sección, datos identificativos, carné de aplicador y número de inscripción en ROPO, si se trata de una empresa.
 - c. Equipos de aplicación de productos fitosanitarios propios de la explotación. Se rellenarán los datos de los equipos de aplicación propios de la explotación que intervengan en los tratamientos fitosanitarios. En caso de aplicaciones contratadas a empresas aplicadoras, no se rellenará este apartado.
 - d. Asesor, agrupación o entidad de asesoramiento a la que pertenece la explotación. Este apartado no es obligatorio, solo para las explotaciones que se rigen por la Gestión Integrada de Plagas (GIP). En caso de pertenecer a alguna agrupación de producción integrada (API), ATRIA o cultivar bajo el sistema de agricultura ecológica, indicar el nombre de la agrupación y su identificación. Si la explotación tiene parcelas en varias de las circunstancias anteriores o bien que tenga algún asesoramiento se rellenan tantas líneas como casos se presenten.

2. Identificación de las parcelas de la explotación.

- a. Datos identificativos y agronómicos de las parcelas. Se aportarán los datos de todas las parcelas que forman parte de la explotación, con sus referencias SIGPAC, sus datos agrónomos, con indicación del sistema de asesoramiento que sigue dicha parcela, si existiera. A cada una de las parcelas se le asigna un número de orden el cual podrá ser referenciado de aquí en adelante para indicar acciones comunes entre parcelas distintas.
- b. Datos identificativos medioambientales de las parcelas. Se indicarán aquellas parcelas de la explotación que tengan o estén próximas a puntos de captación de aguas procedente de pozos y masas de agua utilizadas para consumo humano y/o estén situadas total o parcialmente en zonas específicas de protección. Según el Real Decreto 140/2003, del 7 de febrero, se entiende por consumo de agua humana a todas aquellas aguas ya sea en estado original o después del tratamiento, utilizadas para beber, cocinar, preparar alimentos, higiene personal y para otros usos domésticos. Exceptuando aquellas que, procediendo de un abastecimiento individual y fuente natural, supongan un volumen medio de menos de 10 m^3 diarios de agua o que abastezca a menos de 50 personas.

3. Información sobre tratamientos fitosanitarios

- a. Registro de actuaciones fitosanitarias de la parcela. Esta hoja muestra las actuaciones fitosanitarias que tengan lugar sobre las parcelas de la explotación, reflejando las parcelas tratadas, fecha, superficie, plaga, aplicador que interviene (previamente anotado en el apartado 1.b), equipo utilizado (previamente anotado en el apartado 1.c, en caso que se utilice maquinaria propia de la explotación). Por último se rellenan los restantes campos con las características del producto fitosanitario utilizado y la eficacia observada.
- b. Registro de actuaciones fitosanitarias de la parcela. Se procederá igual que en el punto anterior, pero para aquellas parcelas objeto de asesoramiento en GIP, añadiendo además información sobre la justificación de la actuación así como detalles de dicha actuación, ya

sea química o no química. Esta parte debe ser validada por el técnico agrónomo.

- c. Registro de uso de semilla tratada. Se completará en caso de utilizar semilla tratada con fitosanitarios, tanto si dicha semilla ya contiene ese tratamiento desde el proveedor como si el tratamiento lo realiza el agricultor en la explotación. Indicando las parcelas en las que se utiliza dicha semilla, así como los datos del fitosanitario empleado.
 - d. Registro de tratamientos postcosecha. Al igual que el apartado anterior, se rellenará en el caso de aplicar algún tratamiento sobre el producto cosechado, siempre y cuando dicho tratamiento se lleve a cabo en la propia explotación.
 - e. Registro de tratamientos de los locales de almacenamiento. Igualmente, se completará en caso de aplicar fitosanitarios en algún local de almacenamiento de la explotación para combatir problemas fitosanitarios.
 - f. Registro de tratamientos de los medios de transporte. De igual modo, registrar todo tratamiento fitosanitario aplicado sobre los medios de transporte utilizados para el traslado del producto vegetal dentro de la explotación o desde la misma hasta los centros de manipulación.
- 4. Registro de análisis de productos fitosanitarios.** En el caso de realizar algún análisis de residuos de productos fitosanitarios para controlar la presencia de los mismos en cultivos o productos recolectados. Pueden ser propuesto de forma voluntaria, dentro de un plan de autocontrol, por indicación del asesor o del sistema de calidad de la producción. Se cumplimentará la información al respecto: resultado, material o laboratorio.
- 5. Registro de cosecha comercializada.** Se anotarán los datos correspondientes a las partidas de cosecha comercializada, con indicación de cantidades, procedencia y destinatarios detallando, de manera voluntaria, el número de Registro General Sanitario y de Empresas Alimentarias y Alimentos del operador al que se ha destinado la mercancía.
- 6. Fertilización o plan de abonado.** Se especifica la descripción del fertilizante, composición del mismo y anotaciones sobre las unidades de fertilizantes aportadas. Por ahora es obligatorio para los solicitantes de eco regímenes.

7. **Riego.** Se anotará la superficie a la que se le aplica el sistema de regadío, así como el volumen de agua aportada.
8. **Ecorregímenes.** Constaría de 3 páginas pertenecientes a distintos ecorregímenes: práctica de pastoreo extensivo (art. 30 RD 1048/2022), práctica de siega sostenible en pastos o islas de biodiversidad (art.31 RD 1048/2022), espacios de biodiversidad (cultivos bajo agua), cubiertas vegetales espontáneas o sembradas en cultivos leñosos (art.42 RD 1048/2022) y cubiertas inertes de restos de poda (art.43 RD 1048/2022).
9. **Ayudas sociales.** Este apartado está implícito en el cuaderno siempre y cuando se completen los apartados 3, 6 y 7.

Para terminar, decir que nuestro sistema está capacitado para rellenar y completar todos los apartados, excepto el octavo. Los ecorregímenes consideramos que son aplicables a cultivos y situaciones muy específicas, quedando fuera del objetivo principal del proyecto. Aunque lo contemplamos como mejora futura.

1.7. Metodología de desarrollo

Las empresas y proyectos actualmente cambian rápidamente, por lo que deben responder frente a nuevas oportunidades y mercados de forma ágil. Este entorno flexible hace que sea imposible obtener un conjunto completo de requerimientos de software estable.

En consecuencia, un proceso en cascada o lineal se prolongan con frecuencia porque no están preparados para los cambios. Su proceso es especificar por completo los requerimientos y, después, diseñar, construir y probar el sistema. Son los métodos ágiles los que interrumpen para sustituir a los métodos en cascada. Estos métodos ágiles se centran, gracias al enfoque **incremental**, en el software en lugar del diseño y la documentación.

La filosofía de este tipo de métodos es entregar con prontitud un software operativo a los clientes, quienes entonces proporcionarán requerimientos nuevos y variados para incluirlos en posteriores iteraciones. Consiguiendo así simplificar el proceso burocrático, evitar trabajar con dudas a largo plazo y eliminar

documentación que quizá nunca se emplee. Existen varios principios que todo método ágil debe cumplir y son:

1. Participación del cliente: deben intervenir estrechamente durante el proceso de desarrollo. Su función consiste en ofrecer y priorizar nuevos requerimientos del sistema y evaluar las iteraciones presentadas.
2. Entrega incremental: el software se desarrolla en incrementos y el cliente es el que especifica los requerimientos de cada iteración.
3. Personas, no procesos: tiene que reconocerse y aprovecharse las habilidades del equipo de desarrollo. Debe permitirse a los miembros del equipo desarrollar sus propias formas de trabajar sin procesos establecidos.
4. Adoptar el cambio: diseñar el sistema para adaptar cambios fácilmente.
5. Simplicidad: enfocarse en la simplicidad tanto en el software a desarrollar como en el proceso de desarrollo.

De entre todas las metodologías ágiles, la **Scrum** es en la que nos centraremos. Al ser una metodología ágil por definición se usa para equipos de trabajo multidisciplinares en ciclos iterativos centrados en el cliente. Los tres pilares básicos sobre los que se basa Scrum son:

1. **Sprint**: son periodos de trabajo cuya duración oscila entre una semana y un mes. No pueden superar este tiempo porque es proporcional al riesgo de un contexto cambiante. Al final de cada sprint se genera un incremento que se suma al resto del producto desarrollado.
2. **Eventos**: dentro de cada sprint habrá diferentes procesos que seguir como:
 - a. Organizar el trabajo pendiente: el líder del equipo identificará qué trabajo pendiente hacer en un sprint, asegurándose que está claramente documentado.
 - b. Planificación: antes de comenzar se realiza una planificación para saber qué enfoque tiene ese sprint.
 - c. Comienzo: normalmente dura dos semanas y durante este tiempo el equipo trabajará en sus tareas asignadas.

- d. Reuniones: organizar reuniones diarias de actualización para dar a conocer cualquier obstáculo inesperado que haya surgido.
 - e. Revisión: una vez que se haya terminado el sprint se presenta el trabajo al equipo.
- 3. Roles:** existen varios puestos importantes en el desarrollo de Scrum:
- a. Responsable de producto: persona a cargo de la lista de tareas pendientes del producto, centrado a transmitir el punto de vista del usuario al equipo.
 - b. Scrum Master: persona que dirige los distintos eventos, considerado el gerente del proyecto.
 - c. Equipo Scrum: todas las personas que están trabajando en el sprint. Los miembros del equipo deben auto-organizarse y ser colaborativos para lograr el objetivo de mejora continua.
- 4. Artefactos:** son herramientas imprescindibles, es algo que se crea para resolver un problema y son: el trabajo pendiente del producto, el trabajo pendiente del sprint y el incremento del producto [22].

En la Imagen 7 se muestra una representación gráfica del proceso seguido con la metodología Scrum.

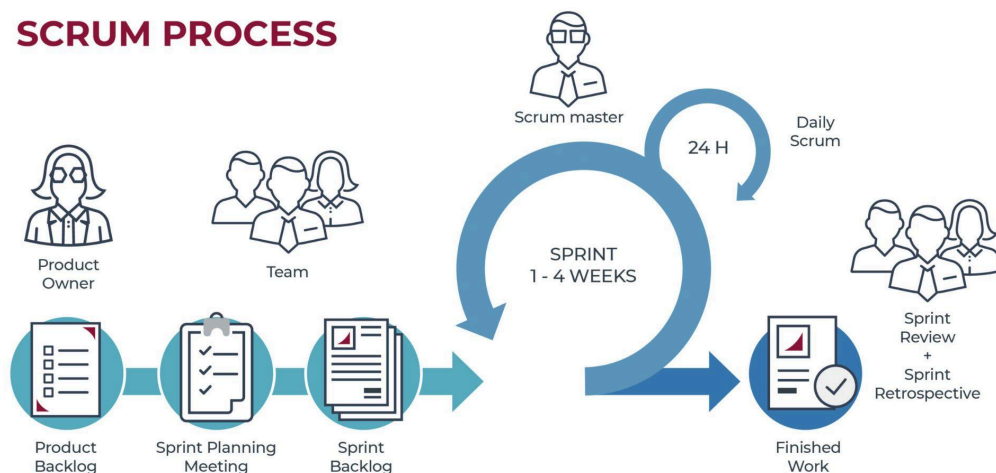


Imagen 7: Proceso de la metodología Scrum ⁶

⁶ <https://ausum.cloud/scrum-metodologia-agil-mas-popular-en-empresas/>

1.8. Definiciones y abreviaturas

Con la realización de este apartado se pretende exponer de forma clara y concreta la definición de diversos conceptos relevantes y redundantes a lo largo del proyecto:

- **Cuaderno de Explotaciones Agrícolas (CUE):** incorpora y mantiene las actividades relacionadas con la gestión agraria, es decir, lo referente a las actuaciones fitosanitarias, aplicación de fertilizantes o cosechas recolectadas, junto a información indispensable relativa a la administración de las explotaciones. Es asignado a una explotación agrícola en el periodo de un ejercicio.
- **Sistema de Información de Explotaciones Agrícolas y Ganaderas (SIEX):** herramienta que consta de un conjunto de bases de datos y registros administrativos interconectados.
- **PROLAC:** base de datos que regula la organizaciones de productos de leche y de las organizaciones interprofesionales en el sector lácteo. En ella las autoridades competentes registran a las organizaciones que lo soliciten.
- **Agrupación de Producción Integrada (API):** grupo de personas o entidades que operan con el objetivo de obtener productos agrarios diferenciados, garantizando al consumidor calidad y seguridad alimentaria y protegiendo al medio ambiente mediante la utilización racional de productos fitosanitarios.
- **Agrupaciones para Tratamientos Integrados de Agricultura (ATRIA):** entidades asociativas creadas desde la administración sin ánimo de lucro, cuya finalidad es colaborar con las administraciones autonómicas contra la lucha de agentes nocivos en los cultivos.
- **Gestión Integrada de Plagas (GIP):** estrategia dinámica de control de plagas. No se trata sólo de eliminar las plagas, sino de lograr un equilibrio en el ecosistema.
- **Sistema de Información Geográfica de Parcelas Agrícolas (SIGPAC):** es un registro oficial cuyo cometido es la gestión administrativa de las ayudas de la Política Agrícola Común (PAC) permitiendo, principalmente, la

identificación de los terrenos agrícolas sobre los que se solicitan ayudas y la determinación de su admisibilidad para las ayudas.

- **CRUD:** CRUD es la abreviatura en inglés de las palabras Create (crear), Read (leer), Update (modificar) y Delete (borrar). Se utiliza para definir el tipo de operación aplicable a un objeto o entidad, sobre todo en el ámbito de las bases de datos.
- **Sistema Gestor de Base de Datos (SGBD):** programa informático compuesto por un conjunto de programas que permiten acceder y gestionar los datos asegurando la integridad, control de concurrencia, seguridad y gestión de usuarios.
- **Registro Oficial de Productores y Operadores (ROPO):** el ministerio establece este registro para establecer el marco de actuación para conseguir un uso sostenible de los productos fitosanitarios. Esta aplicación permite la consulta pública de entidades y operadores registrados.
- **Registro Oficial para la inscripción de la Maquinaria Agrícola (ROMA)** en los supuestos establecidos legalmente.
- **Registro General Sanitario de Empresas Alimentarias y Alimentos (RGSEAA):** base de datos de carácter informativo cuya finalidad es la protección de la salud pública y de los intereses de los consumidores.
- **Uniform Resource Identifier (URI):** identificador de recurso uniforme, contiene el esquema, autoridad, ruta, consulta y fragmento del recurso web.
- **Uniform Resource Locator (URL):** conjunto de letras y símbolos que identifican una página concreta dentro de la web.
- **Unified Modeling Language (UML):** lenguaje de modelado de software, utilizado para comprender los diseños, arquitectura de código y la implementación.

1.9. Estructura de la memoria

Este documento se divide en cinco capítulos diferenciados claramente. En función de la cronología del desarrollo y la elección de una metodología de desarrollo incremental (presentada en el Apartado [1.7](#)) hemos dividido el proyecto en los siguientes capítulos.

La fase de formación e investigación previa al desarrollo de cualquier proyecto es de gran importancia e imprescindible para crear un producto acorde con la oferta actual del mercado. En este sentido, el Capítulo [1](#) describe el contexto que rodea al proyecto, el problema a resolver indicando los objetivos generales y específicos, se reflejan los antecedentes y estado del arte para llevar a cabo la propuesta planteada y la metodología a seguir durante el proyecto.

A medida que pasan los capítulos el nivel de detalle aumenta y es en el Capítulo [2](#) cuando se realiza un análisis más exhaustivo. Comenzamos explicando cuál es la función de cada perfil del sistema, seguimos analizando los requisitos funcionales y no funcionales, cuyo estudio se complementa con los casos de uso y diagramas de secuencia. Después, observamos cuáles son las tecnologías disponibles para cada aplicación a desarrollar y para la base de datos. Finalmente, terminamos con la planificación de las tareas y actividades a cumplir y un presupuesto estimado de los gastos del proyecto.

El Capítulo [3](#) engloba el grueso del proyecto, es decir, la implementación del sistema. Este se divide en cuatro iteraciones: base de datos, diseño de la interfaz, aplicación móvil y aplicación web. Es aquí dónde se verán los resultados.

En el Capítulo [4](#) se reúnen los trabajos futuros, mejoras y conclusiones sobre el proyecto.

Por último, el Capítulo [5](#) desarrolla los manuales o anexos útiles para el usuario.

CAPÍTULO 2

ANÁLISIS INICIAL

2.1. Introducción

A lo largo de este segundo capítulo se explicará el estudio realizado antes de desarrollar el proyecto. En concreto, se especificarán los requerimientos planteados y las tareas de cada perfil de usuario. Además, otro punto importante será evaluar el abanico de tecnologías disponibles y así decidir las más convenientes según sus prestaciones y capacidades.

2.2. Perfiles de usuarios en el sistema

La solución adoptada puede ser aplicable a una empresa o cooperativa con una sección dedicada al asesoramiento técnico y legal en explotaciones olivareras. Por lo tanto está pensado para que existan tres roles dentro del proyecto y son:

- **Administrador:** a través de la aplicación web, este usuario tendrá disponible toda la información que está a disposición de los agricultores y técnicos. Por lo tanto tiene la responsabilidad de recopilar todos los datos necesarios del agricultor con el objetivo de hacer funcional la aplicación móvil. Dicha información es la siguiente:
 - Datos generales de la explotación, así como el número de registro de explotación nacional y autonómico.
 - Datos personales del titular o representante de la explotación.
 - Datos personales de los trabajadores o empresas que intervienen en el tratamiento fitosanitario y en las diferentes labores.
 - Documentación de los equipos utilizados para la aplicación de productos fitosanitarios.
 - Datos identificativos, agrónomos y medioambientales de las parcelas que comprenden la explotación.

Al fin y al cabo se trata de la información inicial que necesitan los técnicos y agricultores para empezar a trabajar.

- **Técnico agrónomo:** tiene asociado un número determinado de agricultores, a los cuales va a tener acceso completo desde la aplicación

móvil a toda la información de su cuaderno activo. El técnico tendrá la posibilidad de añadir informes o análisis de laboratorio realizados, validar actuaciones fitosanitarias propuestas por el agricultor y crear el cuaderno de campo digital en cualquier momento. Además, el técnico puede comunicarse con el agricultor mediante mensajes de retroalimentación a imágenes de problemáticas que el agricultor haya subido al sistema.

- **Agricultor:** tiene la responsabilidad de mantener el cuaderno actualizado con todas las actuaciones, labores y cosechas que conciernen a una explotación. Sin embargo, las actuaciones fitosanitarias tienen otra manera de actuar. Primero el agricultor propone una actuación y la registra en el sistema. A continuación, es tarea del técnico evaluarla o, si es pertinente, modificar algunos de sus características que considere no aptas por motivos técnicos. Por último, una vez la actuación es validada el agricultor ya puede teóricamente aplicarla en el campo. El agricultor puede crear en cualquier momento informes, serán documentos en formato PDF que podrá imprimir o archivar. Por otro lado, el agricultor tiene la capacidad de comunicarle al técnico, mediante imágenes, problemas que puedan aparecer en el campo.

Tanto el agricultor como el técnico van a tener acceso a elementos gráficos que reflejen información histórica en el tiempo con respecto a actuaciones, labores y/o cosechas.

2.3. Análisis de requisitos

La etapa de análisis es una fase imprescindible para realizar un software de calidad. Por regla general los errores más graves y difíciles de corregir provienen de esta etapa. Por lo cual, el análisis de requisitos consiste en determinar lo que el software a desarrollar debe hacer, el servicio que ofrece y sus restricciones. Las funcionalidades que definen al software son reflejo de las necesidades de los usuarios finales del producto.

Este análisis o estudio tiene como resultado un conjunto de propiedades y restricciones expuestas de forma precisa y no demasiado extensa, llamado **requisito o requerimiento**. A menudo, los requisitos del sistema se agrupan

principalmente en funcionales y no funcionales. Se describen en detalle en los siguientes dos apartados.

2.3.1. Requisitos funcionales

Este tipo de requisito es un enunciado de alto nivel que muestra cómo debería reaccionar o comportarse el sistema al cambio de estado producido por el usuario. Dichos requisitos dependen directamente del tipo de software desarrollado, de los usuarios finales para los que va dirigido y el enfoque deseado por la organización.

A continuación, se enumeran los requisitos funcionales hallados para el presente TFG, según el sistema utilizado y el rol del usuario:

1. Aplicación móvil para agricultor y técnico:

- 1.1. Independientemente del tipo de rol que tenga el usuario, deberá introducir sus credenciales, es decir, número del DNI y contraseña.
- 1.2. Ambos usuarios podrán visualizar el listado de parcelas que tengan asociado, ya sea para asesorar, en el caso de técnico, como responsables de su gestión, en el caso de los agricultores.
- 1.3. Todo usuario tendrá acceso a la información de cada parcela que tenga asociada.
- 1.4. Tanto el agricultor como el técnico tendrán la capacidad de actualizar y generar información para mantenerse comunicados.
- 1.5. De forma adicional, el agricultor y técnico podrán visualizar un histórico, mediante representación gráfica, de la información recogida y almacenada en la aplicación.
- 1.6. Los usuarios podrán crear distintos informes en formato PDF como relaciones de cosechas recogidas, labores o tratamientos.

2. Aplicación móvil para el agricultor:

- 2.1. El agricultor tendrá la capacidad de añadir, modificar y/o eliminar las distintas operaciones que aplique en el campo. Ya sea tratamientos, labores, registros de recolección de cosecha o imágenes indicando problemas.

3. Aplicación móvil para el técnico agrónomo:

- 3.1. El técnico agrónomo tendrá que visualizar una lista de agricultores, asociados desde el administrador para gestionar sus explotaciones.
- 3.2. Tendrá la responsabilidad de validar y/o indicar posibles modificaciones o mejoras de los tratamientos propuestos por el agricultor. Así como retroalimentar al agricultor con mensajes en las imágenes que considera una problemática.
- 3.3. De forma exclusiva, el técnico tiene la potestad de añadir y/o eliminar registros de análisis de productos fitosanitarios en caso de realizarse.
- 3.4. Solamente el técnico podrá crear el cuaderno de campo completo.

4. Aplicación web

- 4.1. El administrador será el único tipo de usuario que puede iniciar sesión en la aplicación web.
- 4.2. Tendrá la capacidad de gestionar la información relacionada con las parcelas, usuarios, cuadernos de campo, empresas comercializadoras, equipos y personal de todas las explotaciones registradas. Para todos ellos se podrá listar, insertar, editar y/o eliminar los registros.
- 4.3. Visualizará toda la información detallada que está relacionada con un cuaderno de campo determinado.
- 4.4. La aplicación web mostrará las problemáticas, mediante imágenes, que los agricultores registran en el sistema.

2.3.2. Requisitos no funcionales

Por el contrario, un requisito no funcional describe limitaciones o restricciones temporales, de desarrollo o impuestas por estándares. Es decir, son características a tener en cuenta en la arquitectura global del proyecto, no centrado en una característica individual. Como ejemplos podemos decir la seguridad, rendimiento, disponibilidad o la fiabilidad. Este nivel de requisito es muy importante para el conjunto porque su fracaso haría que el sistema fuera inútil [23]. Posteriormente, se detallan los requisitos no funcionales precisados para el actual TFG:

1. Todo usuario agricultor y técnico podrá acceder a la aplicación móvil siempre y cuando el administrador lo haya dado de alta en el sistema. Para ello el administrador deberá solicitar toda la información pertinente y adjuntarla en la aplicación web.
2. Los objetivos principales de la aplicación móvil deberán de poder realizarse sin conexión a Internet, debido a que son condiciones normales en el campo.
3. La cantidad de información transmitida entre la base de datos y el usuario deberá de ser la mínima posible, debido a una posible baja velocidad de conexión.
4. La aplicación utilizará OpenGL ES 3.0, compatible con Android 4.3 y posteriores. Por tanto, la aplicación debe funcionar en cualquier dispositivo que tenga instalado la versión de Android 4.3 o superior.
5. Es muy importante que las aplicaciones sean fáciles de aprender a utilizar, intuitivas y amigables. Así, evitamos que los usuarios dediquen un sobreesfuerzo en utilizar la aplicación.
6. En cada entrada de datos el sistema deberá ser capaz de comprobar el formato de los datos y avisar con mensajes el motivo del error, si hubiera.
7. Para todas las vistas de la aplicación web se tendrá la posibilidad de filtrar la información a listar. Consiguiendo así un mayor manejo sobre los datos.
8. Además, se deberá explicar con claridad cuál es la función de cada pantalla que se visualice en las aplicaciones.
9. Ambas aplicaciones deberán de tener un tiempo de respuesta asequible, es decir, podemos considerar aceptable entre 3 y 5 segundos desde que el usuario realiza una petición hasta que la aplicación da respuesta.

2.4. Documentación de requisitos

El documento de requisitos de software es un comunicado oficial de lo que deben implementar los desarrolladores del sistema. Incluye tanto los requisitos funcionales como no funcionales. Para ayudar a entender la funcionalidad de un documento de requisitos vemos cuáles son los usuarios que lo forman:

- Clientes del sistema: son los que especifican los requisitos y comprueban que se cumplen sus necesidades, así como los cambios que deseen.
- Administradores: los usan para planear un presupuesto y el proceso de desarrollo.
- Ingenieros: los usan para entender qué sistema deberá desarrollarse, las pruebas determinadas para su validación, así como, la comprensión de los componentes y sus relaciones.

Por lo tanto, dicho documento se expande en los siguientes apartados, los cuales amplían cada una de sus partes.

2.4.1. Casos de uso

Una vez que ya hemos definido qué debemos de cumplir en nuestro proyecto y los distintos perfiles de usuario, vamos a analizar los casos de uso. Un **caso de uso** cuenta una historia acerca de cómo un usuario final interactúa con el sistema bajo un conjunto específico de circunstancias.

El primer paso para escribir un caso de uso es definir un conjunto de **actores** que vayan a participar. Los actores son las distintas personas, roles o dispositivos que se comunican o interactúan con el sistema o producto. Un usuario normal puede tener varios papeles diferentes cuando usa el sistema, mientras un actor representa una clase de entidades externas que solo tiene un papel en el contexto del caso de uso. Debido a que la indagación de los requerimientos es una actividad evolutiva, no todos los actores son identificativos en la primera iteración. Los actores principales interactúan para lograr la función requerida del sistema y obtienen el beneficio previsto de este, trabajan con el software en forma directa y con frecuencia. Los actores secundarios dan apoyo al sistema, de modo que los primeros puedan hacer su trabajo [24].

También, será necesario representar el **sistema** dónde los actores toman las acciones y las **asociaciones** que definirán la comunicación entre actores y casos de uso. En la Imagen 8 se representa la simbología explicada y aplicada en los siguientes diagramas sobre los casos de uso.

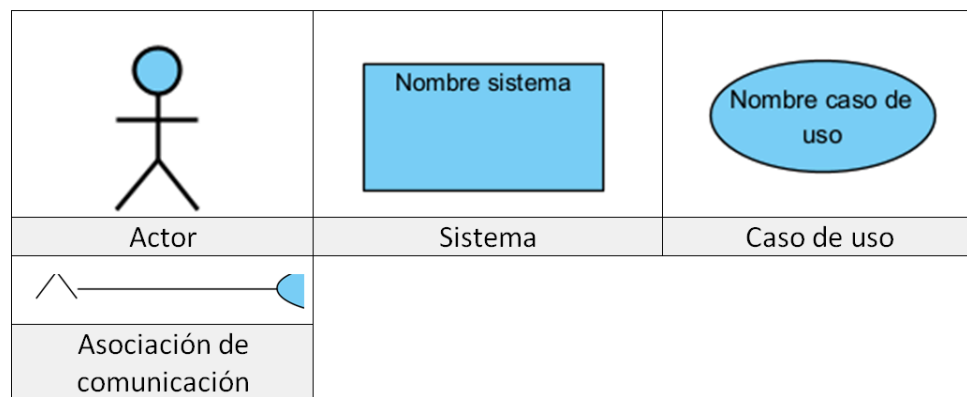


Imagen 8: Simbología en un caso de uso

A continuación, vamos a presentar nuestros casos de uso. En nuestro caso vamos a identificar a tres actores principales que son: agricultor, técnico y administrador. Estos son los mismos roles que habrá en las aplicaciones (ver Apartado [2.2](#)). También, habrá dos tipos de sistemas, la aplicación móvil y la web. En la Imagen [9](#) se hace alusión al diagrama de caso de uso general de la aplicación, se muestran las tareas que los agricultores y técnicos tendrán la posibilidad de realizar en la aplicación móvil. Y, por último, en la Imagen [10](#) se visualiza el diagrama general de los casos de uso relacionados con la aplicación web, solo accesibles desde el administrador.



Imagen 9: Diagrama de casos de uso de la aplicación móvil

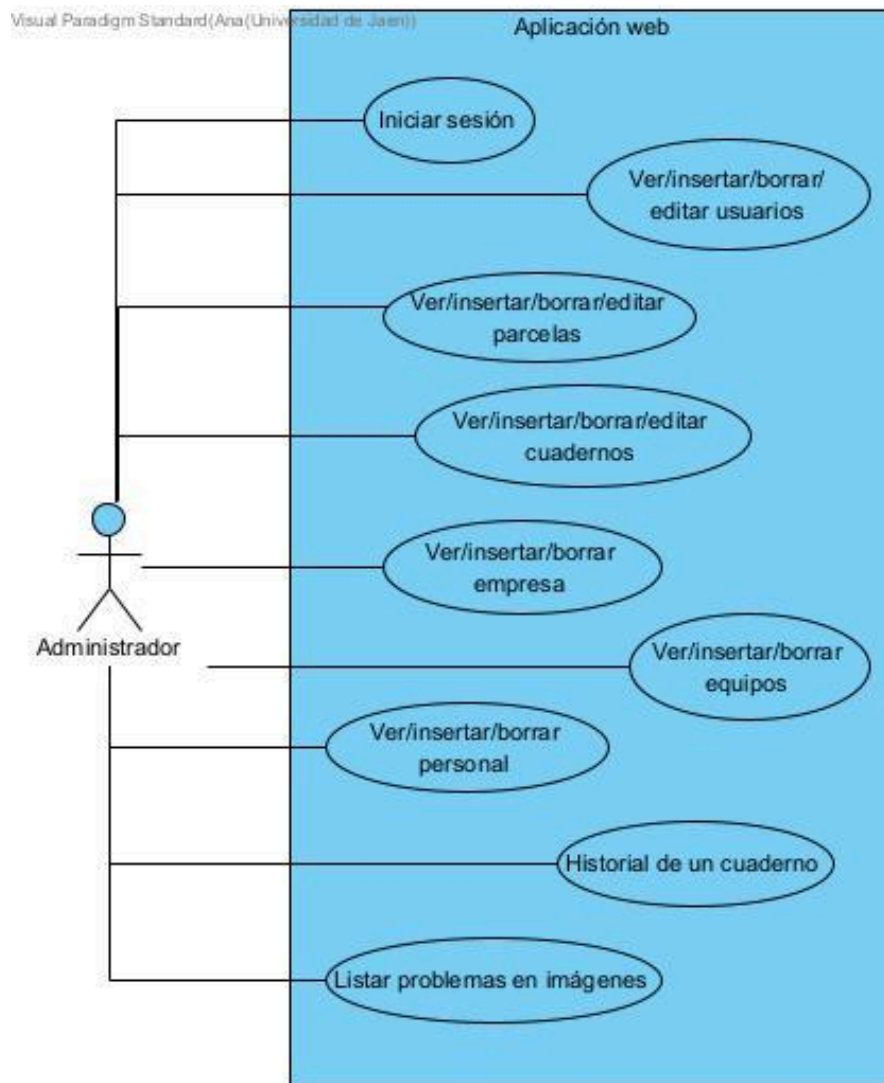


Imagen 10: Diagrama de casos de uso de la aplicación web

2.4.2. Diagramas de secuencia

Los **diagramas de secuencia** modelan la secuencia lógica a través del tiempo de las acciones que definen un caso de uso. Al fin y al cabo un caso de uso es la acción que realiza un actor, pero se enumera de forma general, un diagrama de secuencia es más específico.

Un diagrama de secuencia se estructura mediante **líneas de vida**, que a su vez pueden representar a un actor o cualquier otro elementos que participe en el transcurso del caso de uso. Normalmente, en las líneas de vida, se enuncia el nombre de este elemento junto a su simbología. Por ejemplo, a lo largo de nuestros diagramas se utilizará una **clase límite**, esta será la interfaz o *frontend*

que establece el diálogo entre el actor y el sistema. Otro tipo de objeto a utilizar es el controlador, este hará la función de servidor, *backend* o sistema encargado de ejecutar determinados procesos.

Además, una línea de vida mantiene una cronología que nos permite visualizar de un rápido vistazo las interacciones, mensajes y actores de ese elemento. Cada línea de vida puede contener uno o varios **hilos de ejecución**, estos se tratan de momentos de la vida del objeto en la cuál se ejecutan procesos o funciones. Como ya hemos comentado, existen **mensajes**, estos definen la forma de comunicación entre elementos o líneas de vida y pueden ser:

- **Síncronos:** el objeto que envía un mensaje queda bloqueado hasta que termina la llamada
- **Asíncronos:** terminan inmediatamente después de ser llamado, crean un nuevo hilo de ejecución dentro de la secuencia. Ambos tipos de mensajes pueden tener una respuesta.
- **Auto mensaje:** mensaje que se envía un objeto a sí mismo.

Por otro lado, se pueden incorporar **diagramas operacionales**. Es decir, podemos encapsular bloques de la línea de vida en operaciones *loop*, *alt* o *opt*, entre otros. *Loop* representa un bloque cuyos mensajes e hilos de ejecución se repiten en base a la veracidad de una condición. Sin embargo, *alt* es una operación que se ejecuta una sola vez si se cumple o no una condición. Finalmente, *opt* indica que la ejecución de los mensajes que contiene el bloque es opcional [25]. En la Imagen 11 representamos la distinta simbología que utilizaremos en los diagramas de secuencia.

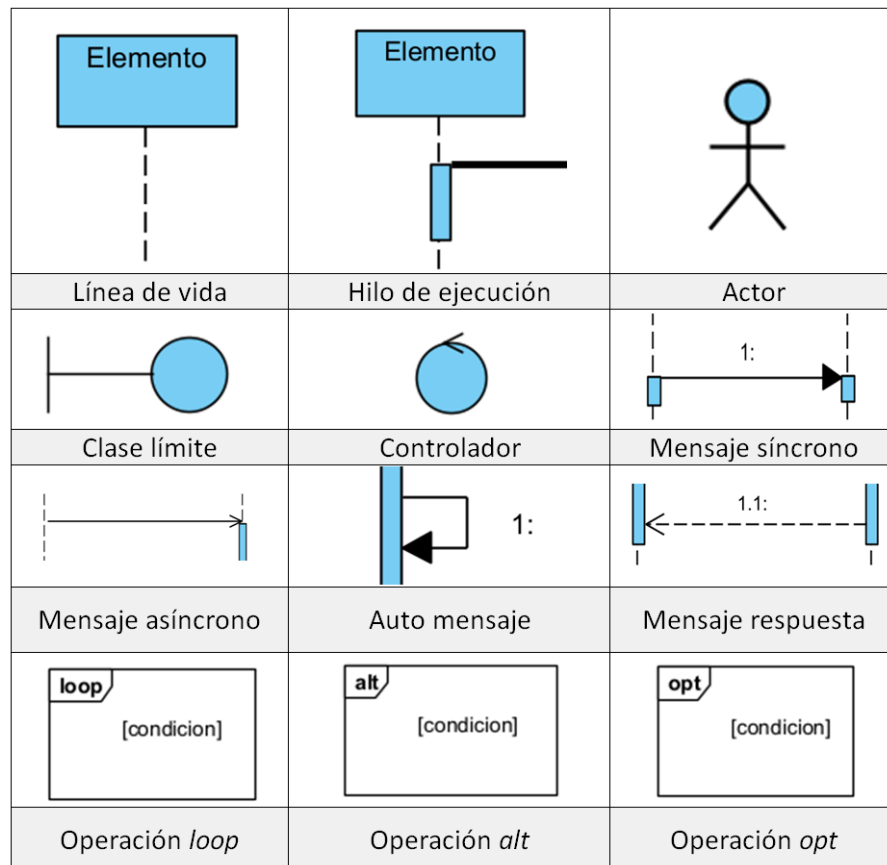


Imagen 11: Simbología presente en los diagramas de secuencia

2.4.2.1. Diagramas de secuencia de la aplicación móvil

Comenzaremos con los diagramas de secuencia que representan los casos de uso de la aplicación móvil. Como ya se ha indicado en el Imagen [9](#), puede haber casos de uso comunes a los dos actores, agricultores y técnicos, estos solo se explicarán una vez. A lo largo de los diagramas se indicarán cinco distintos elementos para las líneas de vida: actor, interfaces (UI), controlador, base de datos local y base de datos global. Los tres primeros objetos sí se explican con anterioridad (ver Apartado [2.4.2](#)) porque depende de su representación, pero los demás elementos son genéricos. Estos se tratan de las bases de datos, más adelante explicaremos con detalle las especificaciones de la arquitectura del sistema (ver Apartado [2.8](#)), no obstante, podemos adelantar a grandes rasgos que la llamada 'base de datos local' será la que dispone el propio dispositivo y la 'base de datos global' la que está desplegada en la nube.

- **Iniciar sesión (Imagen 12):** primero, se comprueba si el usuario ya ha iniciado sesión con anterioridad en la aplicación, si es así ir directamente a su pantalla principal. Sino el usuario introduce sus credenciales, DNI y contraseña, en el formulario de inicio de sesión. Se consulta la base de datos local para obtener los parámetros del usuario o la base de datos global, si no lo tenemos localmente. Cuando ya se tiene el usuario, validar sus credenciales con las introducidas por el formulario y si son correctas entrar a la aplicación, de lo contrario mostrar el mensaje de error pertinente.

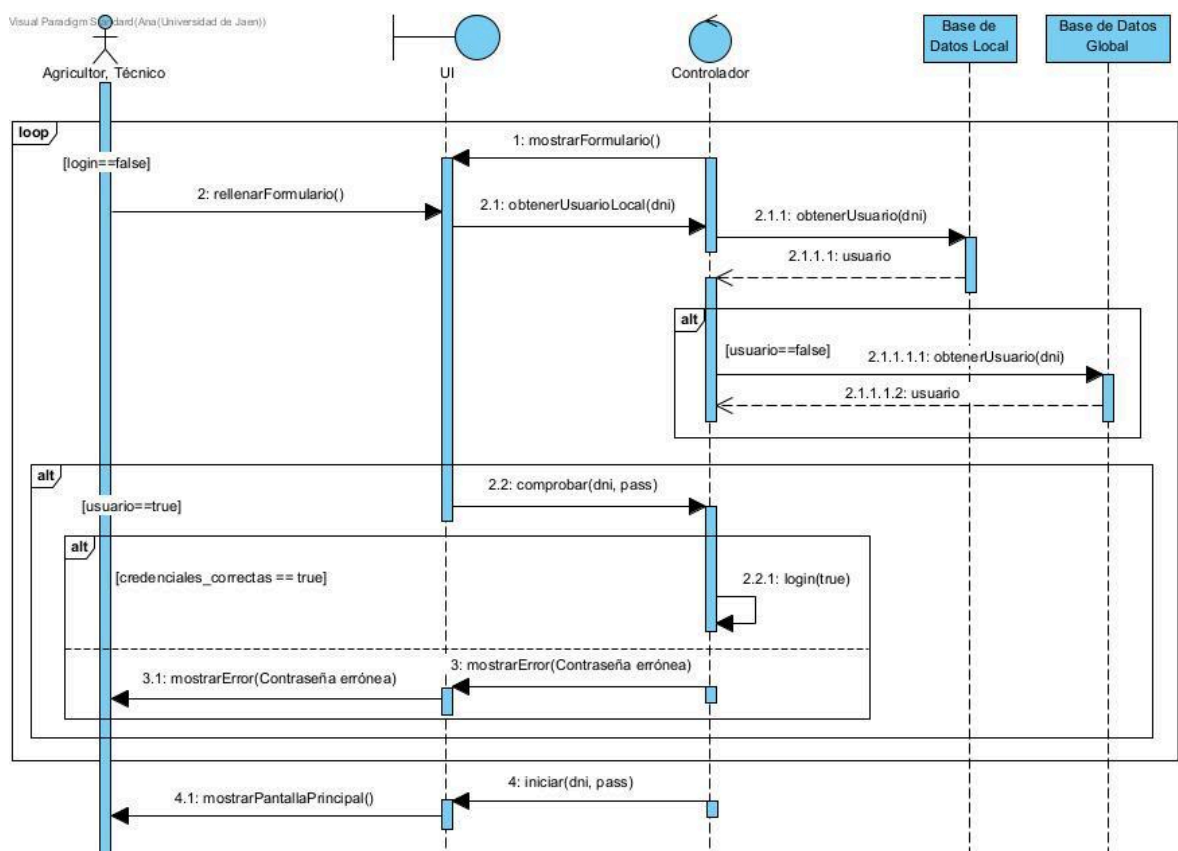


Imagen 12: Diagrama de secuencia para iniciar sesión en la aplicación móvil

- **Ver listado cuadernos (Imagen 13):** esta vista será la pantalla principal para los técnicos, aparecerán todos los agricultores que tiene asociados para asesorar y tienen un cuaderno de campo activo. Se podrá pulsar un cuaderno que aparecerá dispuestos en una lista, con el fin de ofrecer dos funcionalidades. Una será para crear el informe de cuaderno de campo y otra para ver las parcelas que tiene registradas este cuaderno. Estas

últimas funcionalidades aparecerán en otro diagrama debido a su gran tamaño.

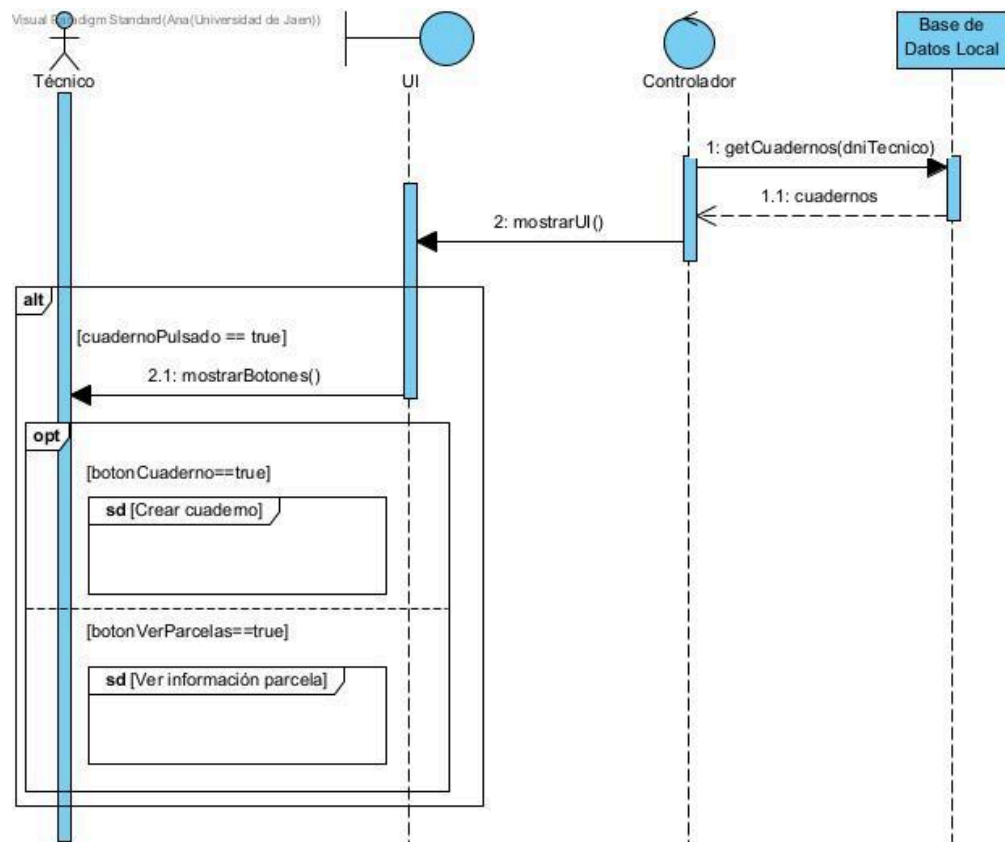


Imagen 13: Diagrama de secuencia para ver listado de cuadernos en la aplicación móvil

- **Crear cuaderno (Imagen 14):** esta opción sólo es accesible desde el listado de cuaderno, por lo tanto solo los técnicos serán los que pueden crear el cuaderno de campo. Lo primero que se realiza después de pulsar el botón es comprobar que se tienen los permisos para lanzar notificaciones. Esto es porque inmediatamente después de crear y guardar el informe en el dispositivo se notifica al usuario.

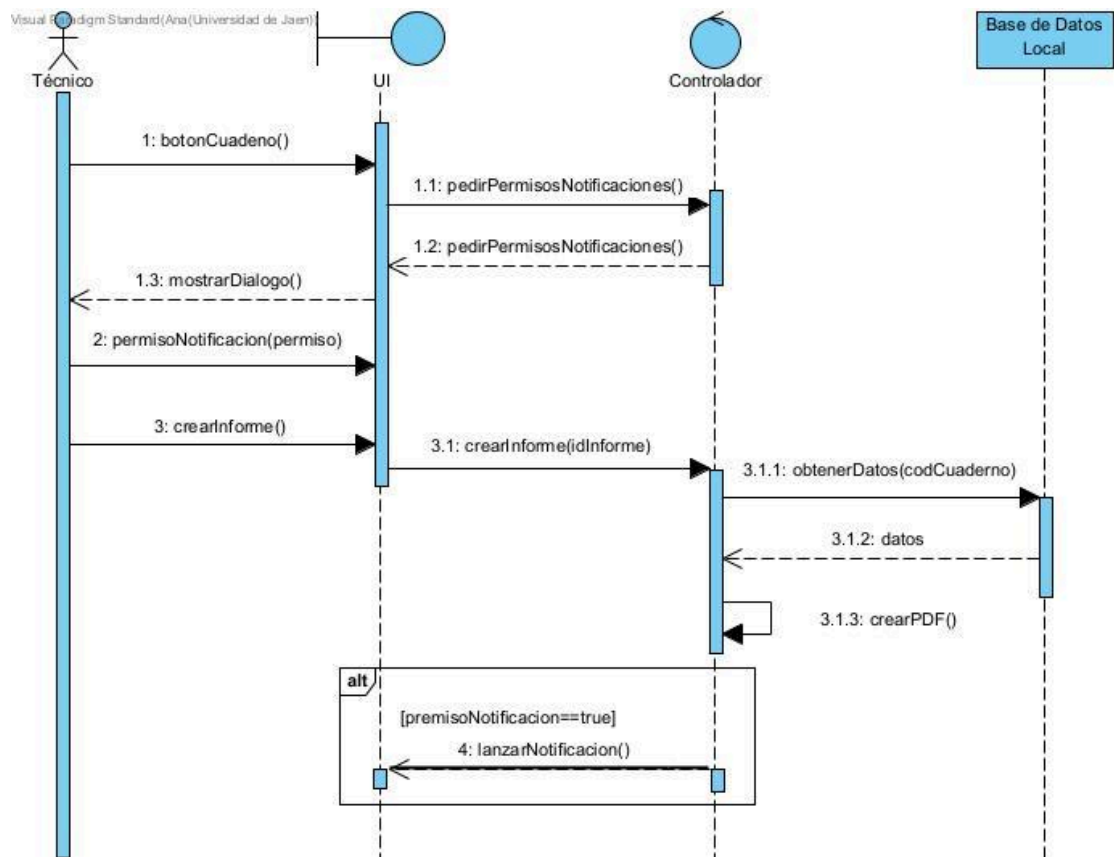


Imagen 14: Diagrama de secuencia para el crear un cuaderno

- **Ver listado parcelas (Imagen 15):** tanto el técnico como el agricultor pueden acceder a esta vista. El técnico puede acceder a ella a través del botón asociado a un cuaderno seleccionado, sin embargo para el agricultor será la pantalla principal que visualizará después de iniciar sesión. En ambos casos se visualizará el listado de parcelas asociadas a un agricultor. Para cada una de las parcelas se desplegarán dos botones, uno para visualizar su información y otro para mostrar las operaciones accesibles.
- **Ver información parcela (Imagen 16):** igualmente, se trata de una vista que puede ser accesible desde el rol de agricultor y de técnico. Siempre tras pulsar el botón que aparece al seleccionar la parcela del listado. Además, se visualizará una imagen de la parcela seleccionada, junto a los datos relativos de la parcela seleccionada.

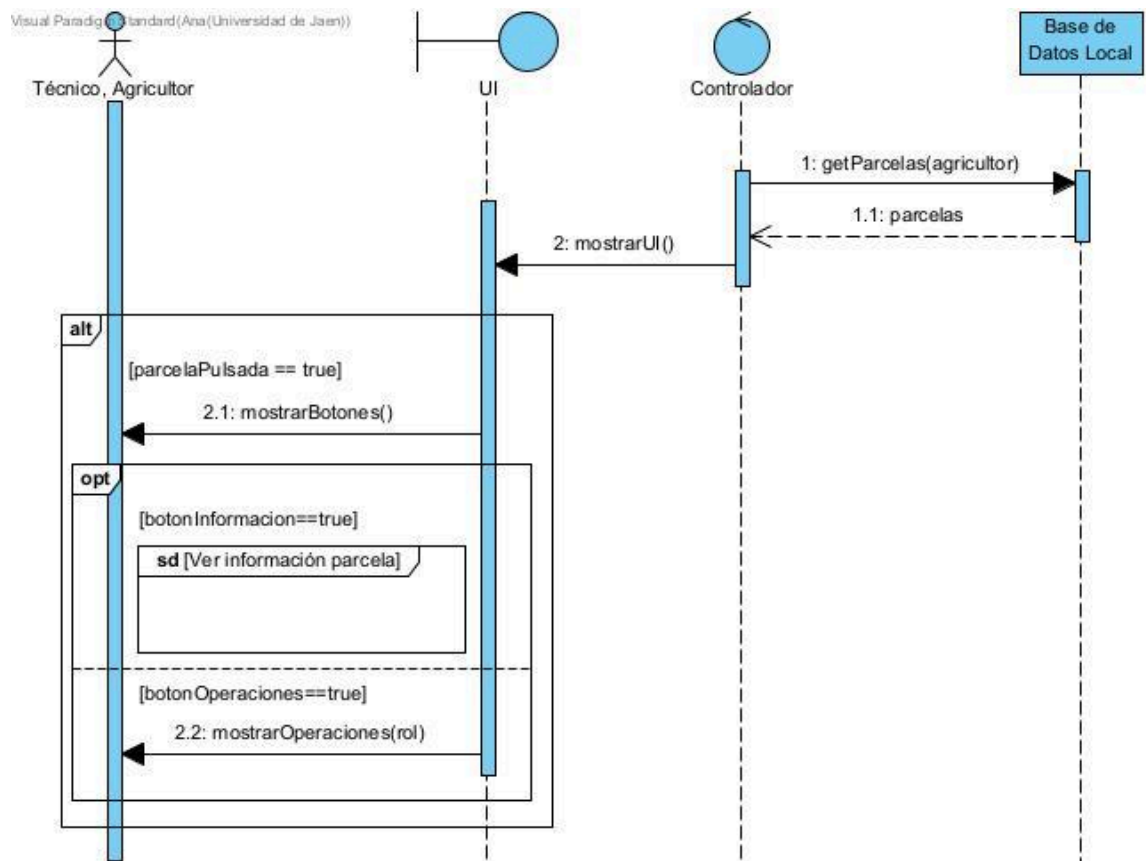


Imagen 15: Diagrama de secuencia para ver el listado de parcelas en la aplicación móvil

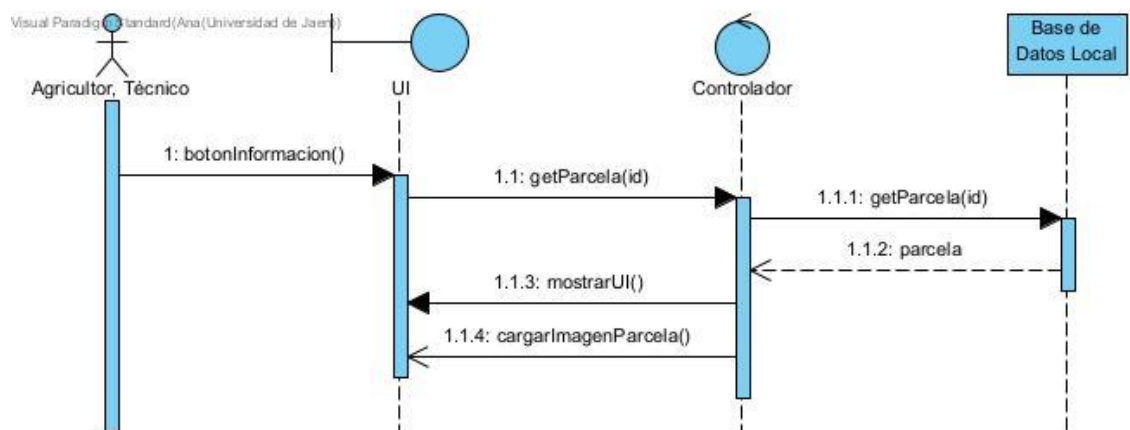


Imagen 16: Diagrama de secuencia para ver la información de la parcela en la aplicación móvil.

- **Descargar/actualizar información (Imagen 17):** esta funcionalidad es esencial en la aplicación móvil, porque será la que permitirá tener la información actualizada entre el técnico y el agricultor. Es decir, la comunicación entre ambos roles se hace gracias a la base de datos global en la nube. Por este motivo, desde el móvil se podrá descargar o actualizar los cambios que haya en la base de datos global que no estén en la base de datos local.

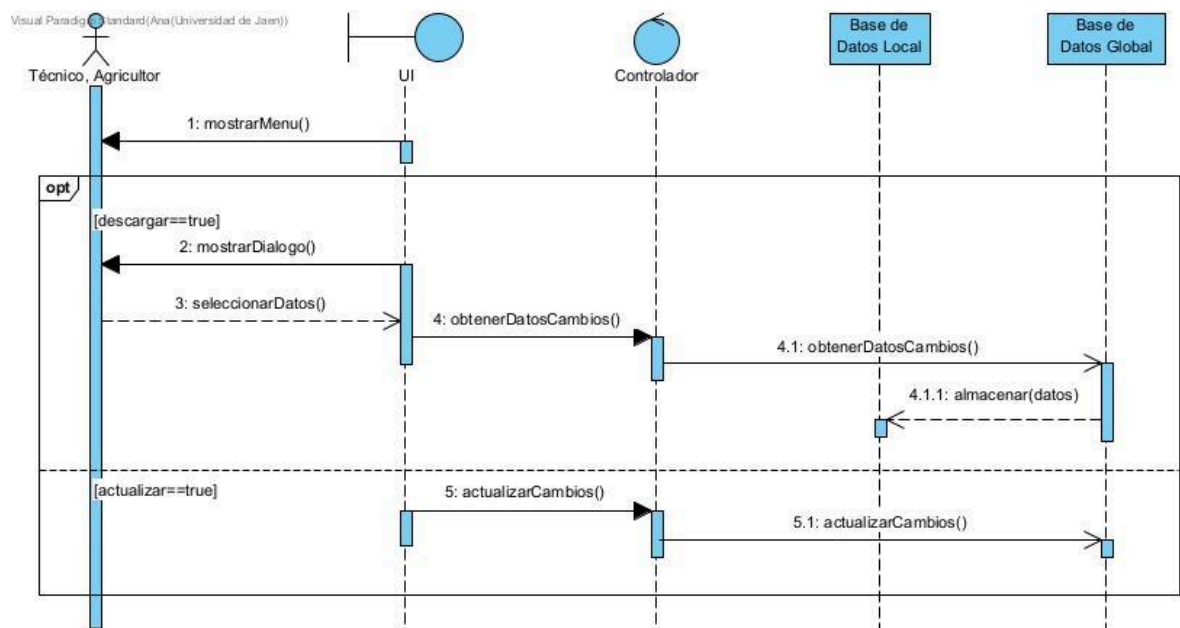


Imagen 17: Diagrama de secuencia para descargar y/o actualizar información en la aplicación móvil

- **Ver históricos (Imagen 18):** igual que las tres últimas vistas, los históricos también son comunes a ambos roles, técnicos y agricultores. Estos históricos mostrarán distintos gráficos relacionados con las cosechas, los tratamientos fitosanitarios o labores registradas por el agricultor.
- **Crear informes (Imagen 19):** será de gran utilidad para los técnicos y agricultores la obtención de documentos o informes en formato pdf, para poder imprimirlos o archivarlos. De nuevo se tendrá la capacidad de lanzar notificaciones, para mejorar la experiencia de usuario, siempre y cuando previamente se hayan aceptado los permisos.

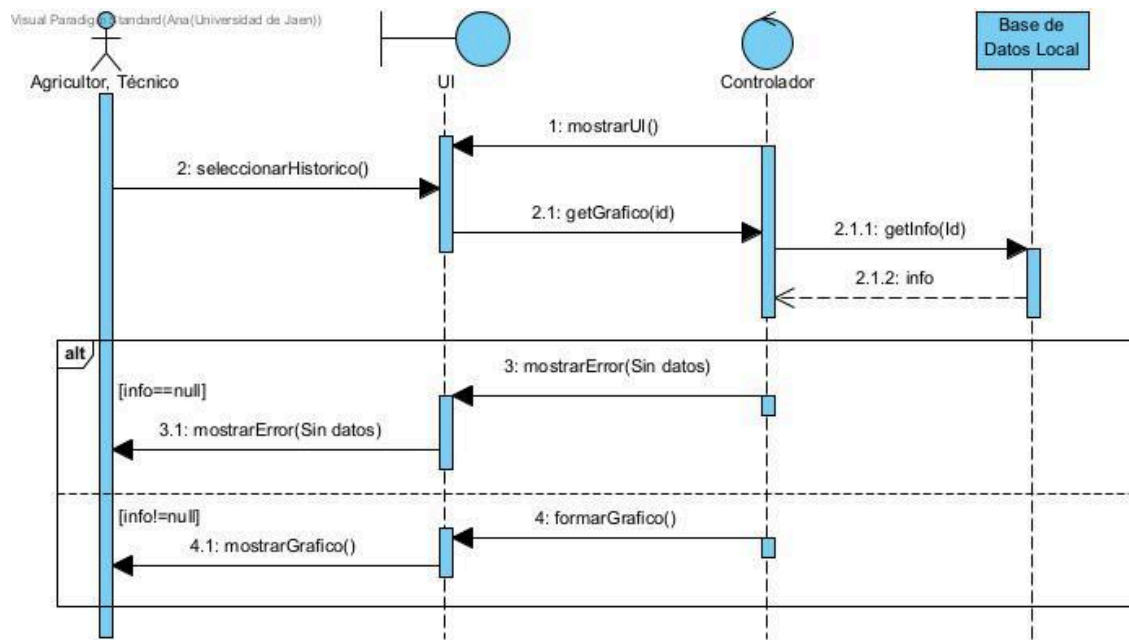


Imagen 18: Diagrama de secuencia para ver históricos en la aplicación móvil

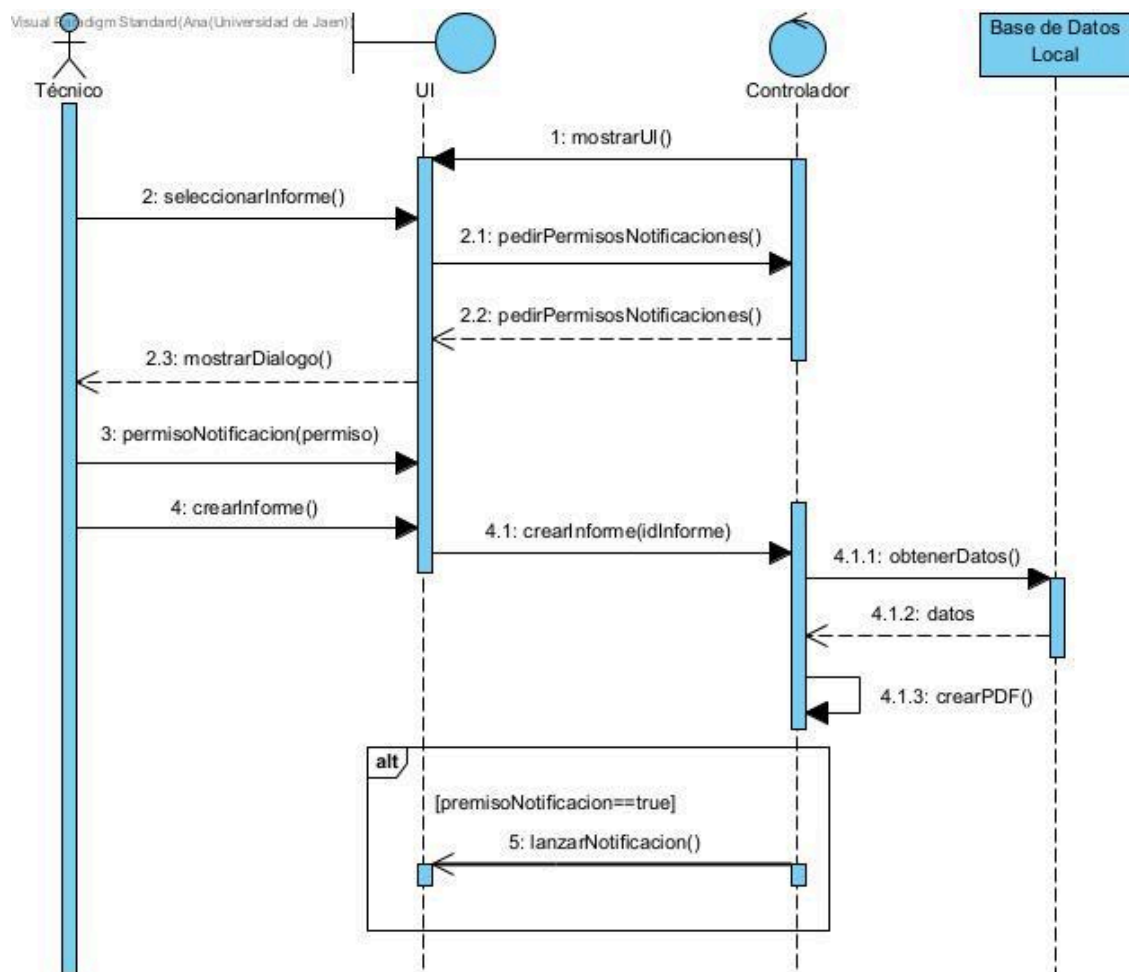


Imagen 19: Diagrama de secuencia para crear informes en la aplicación móvil

- **Ver/añadir/modificar/borrar tratamientos (Imagen 20):** en este diagrama de secuencia se realizan las operaciones CRUD relacionadas con los tratamientos fitosanitarios que el agricultor registra. Solo este tipo de usuario podrá realizar todas las operaciones.

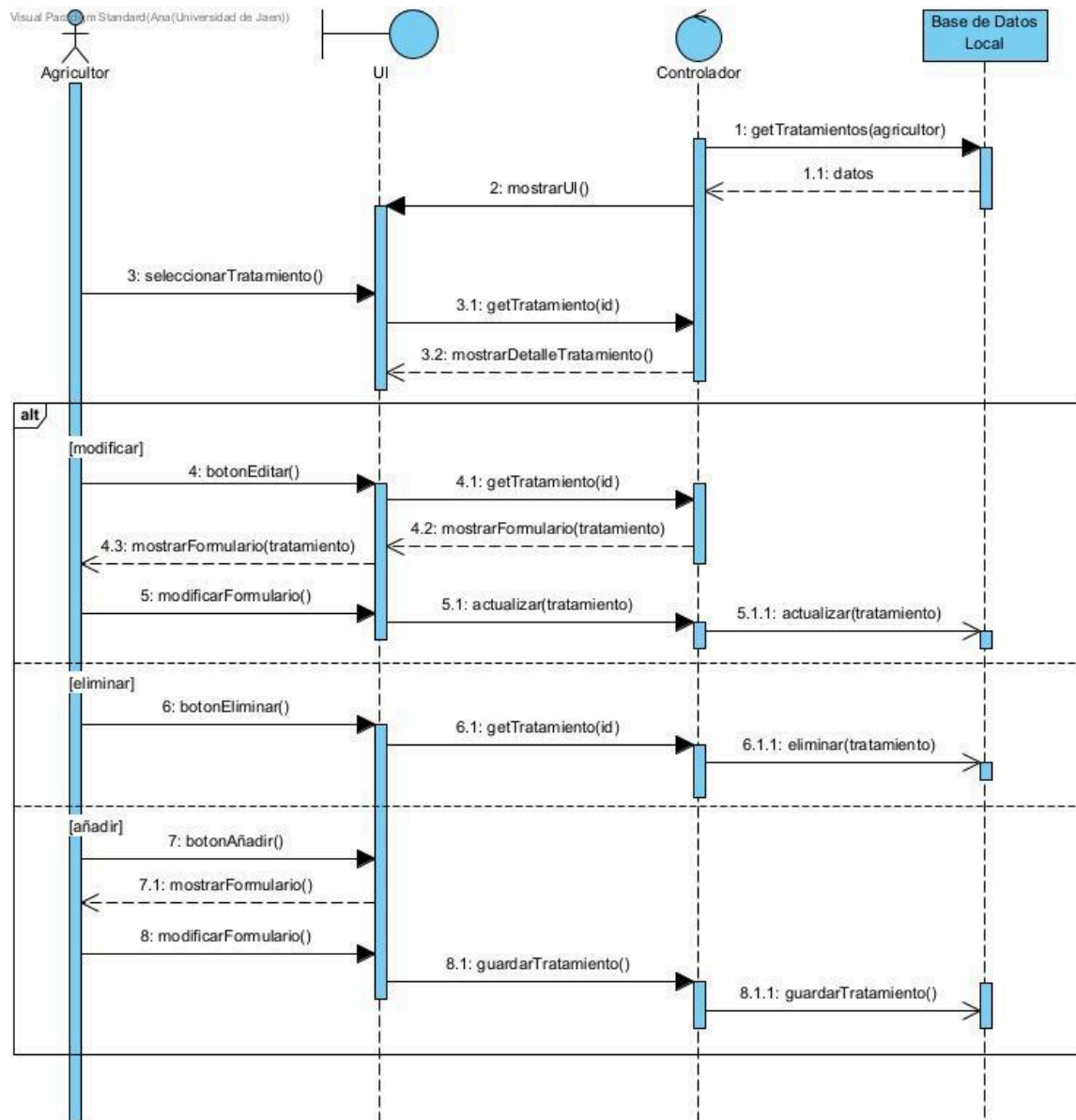


Imagen 20: Diagrama de secuencia para ver, añadir, modificar y/o borrar tratamientos en la aplicación móvil

- **Ver/añadir/borrar labores (Imagen 21):** en este caso se quiere operar sobre las labores que el agricultor realiza en el campo, como pueden ser poda, recogida del fruto, limpia, etc. Las operaciones que se realizan sobre labores son visualizar, añadir y/o borrar de la base de datos.

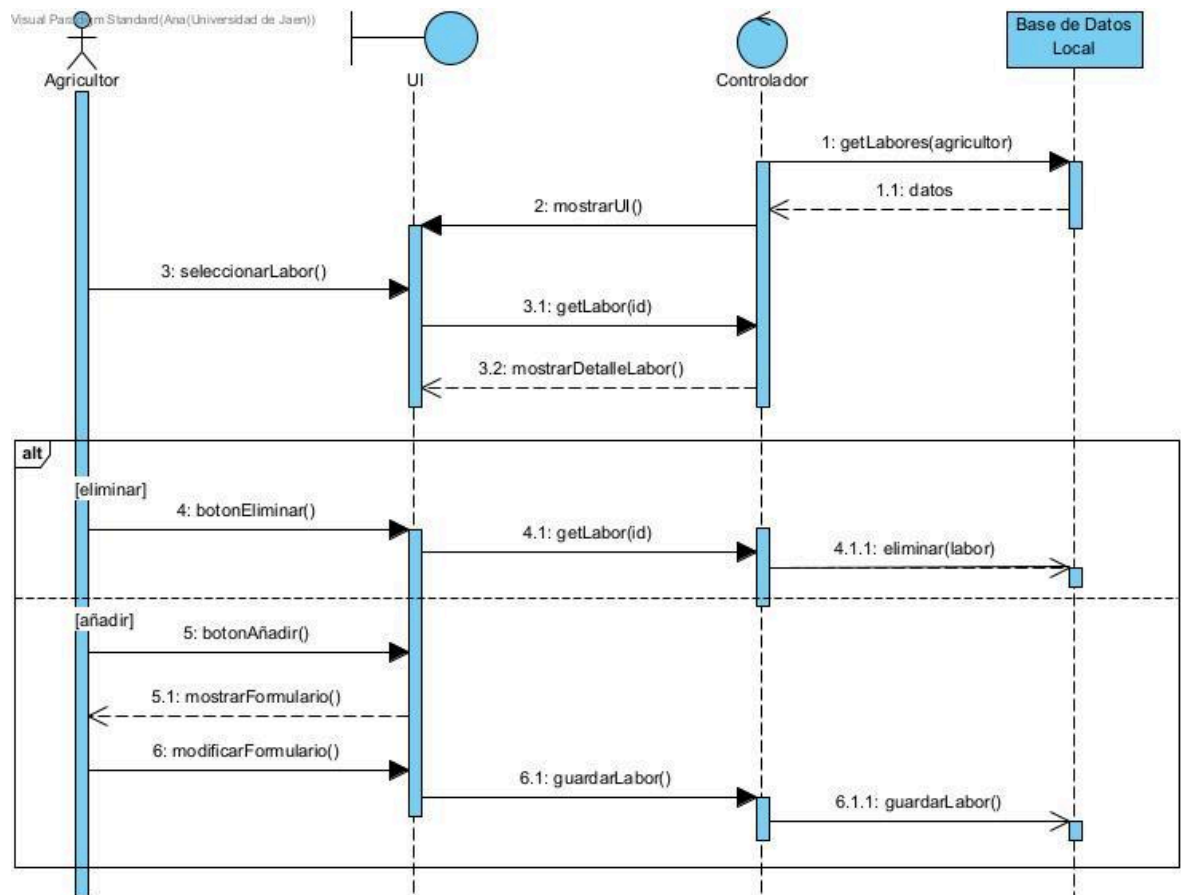


Imagen 21: Diagrama de secuencia para ver, añadir y borrar labores en la aplicación móvil

- **Ver/añadir/borrar cosechas (Imagen 22):** ahora se realizan las mismas operaciones que en el diagrama anterior, pero esta vez sobre cosechas. Es decir, las entregas de fruto realizadas a la empresa encargada de tratar y comercializar el producto.

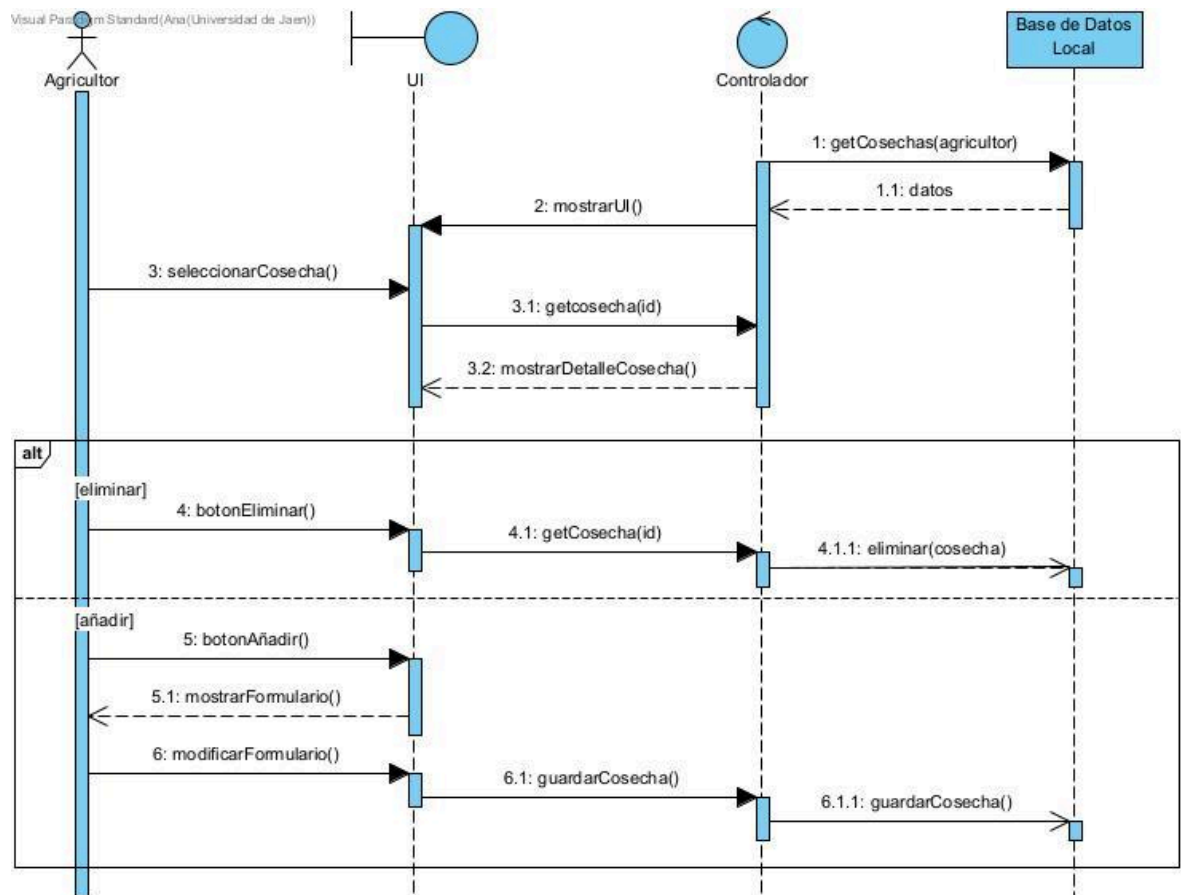


Imagen 22: Diagrama de secuencia para ver, añadir y borrar cosechas en la aplicación móvil

- **Ver/añadir/borrar imágenes (Imagen 23):** finalmente, dentro de los casos de uso del agricultor, este diagrama de secuencia representa las operaciones que se pueden aplicar sobre las imágenes. Recordamos que las imágenes representan problemáticas que el agricultor ve en el campo y quiere trasladar al técnico.

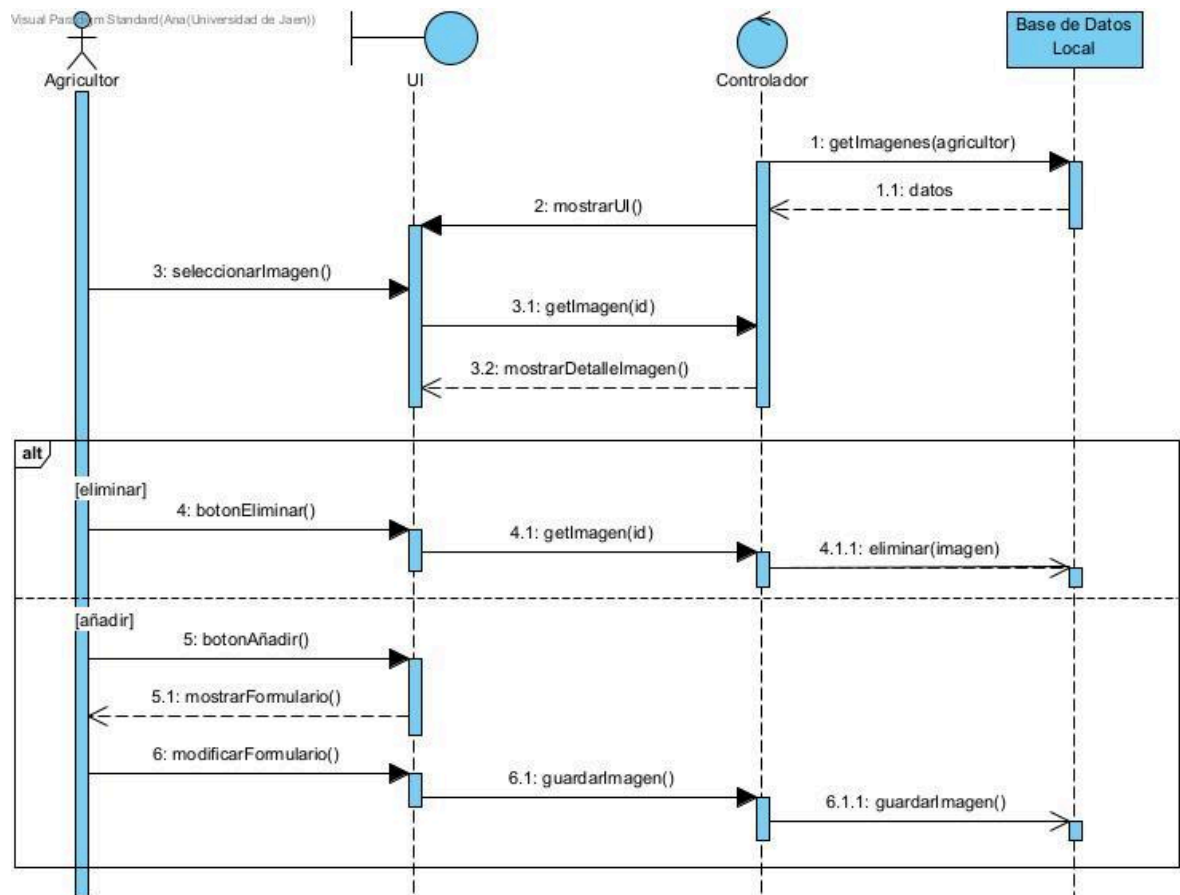


Imagen 23: Diagrama de secuencia para ver, añadir y borrar Imágenes en la aplicación móvil

- **Ver/añadir análisis (Imagen 24):** por otro lado, vamos a desarrollar los diagramas relacionados con las acciones de un técnico. Este diagrama representa las posibles operaciones aplicadas en los análisis, es decir, pruebas sobre el cultivo o fruto realizadas en un laboratorio. Solamente el técnico podrá ver o insertar nuevos registros.
- **Validar y retroalimentar tratamientos (Imagen 25):** este diagrama de secuencia representa otra característica fundamental en la aplicación móvil, es decir, la comprobación por parte del técnico de los distintos tratamientos registrados por el agricultor.

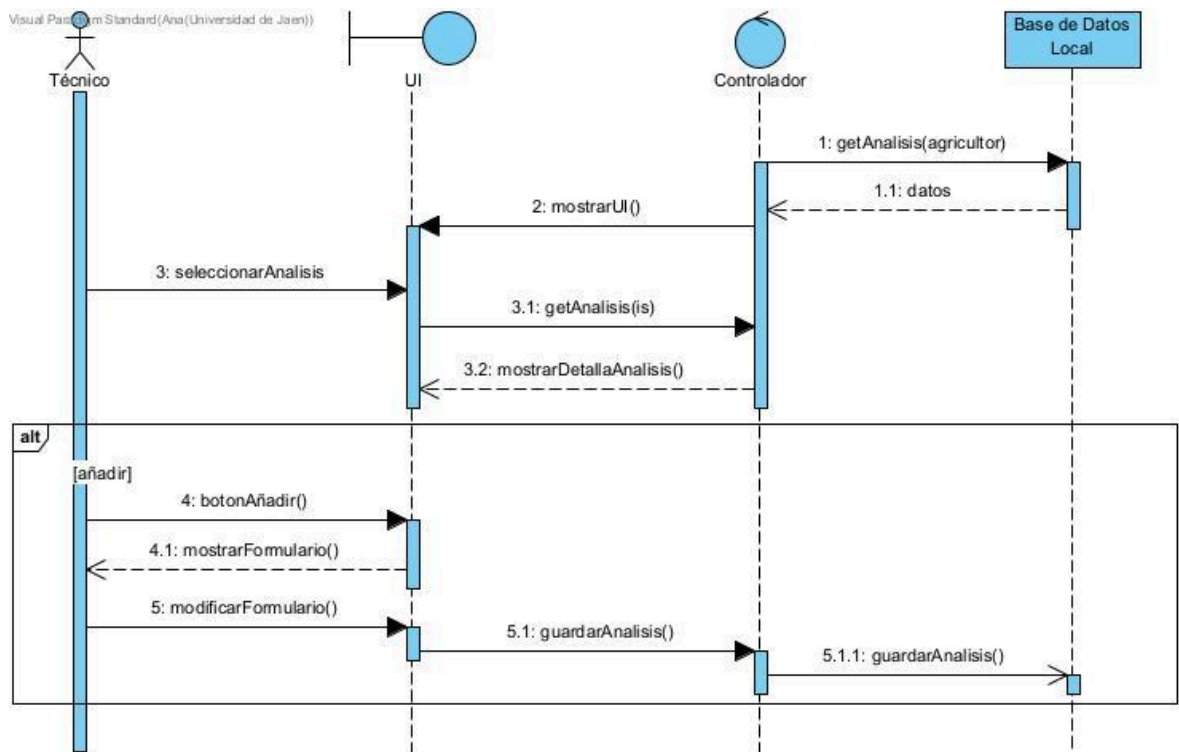


Imagen 24: Diagrama de secuencia para ver, añadir análisis en la aplicación móvil

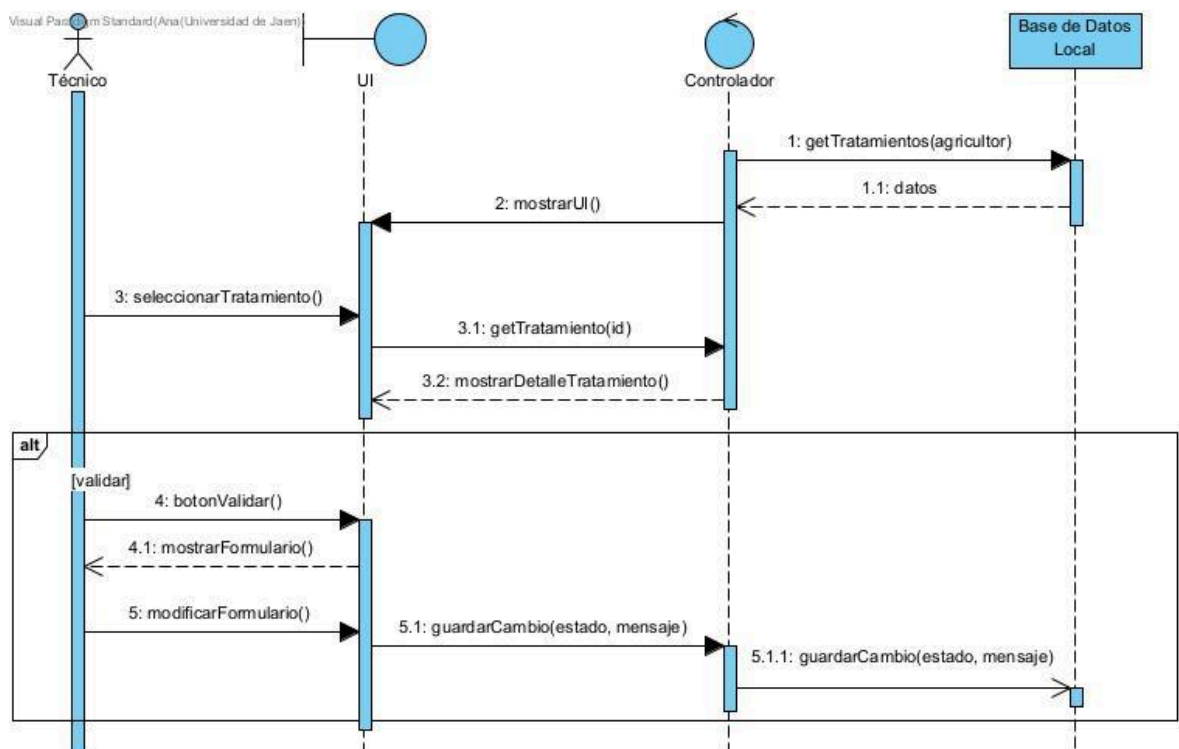


Imagen 25: Diagrama de secuencia para validar y retroalimentar tratamientos en la aplicación móvil

2.4.2.2. Diagramas de secuencia de la aplicación web

Seguimos con los casos de uso de la aplicación web. Para este tipo de aplicación solo tendrá acceso el administrador del sitio, por lo que no habrá distinciones entre diagramas.

- **Iniciar sesión (Imagen 26):** la aplicación web también tiene un inicio de sesión, con el objetivo de que solo puedan beneficiarse los usuarios registrados en el sistema y, en concreto, los administradores. El diagrama es similar al inicio de sesión de la aplicación web, sin embargo, aquí directamente se consulta la base de datos global.

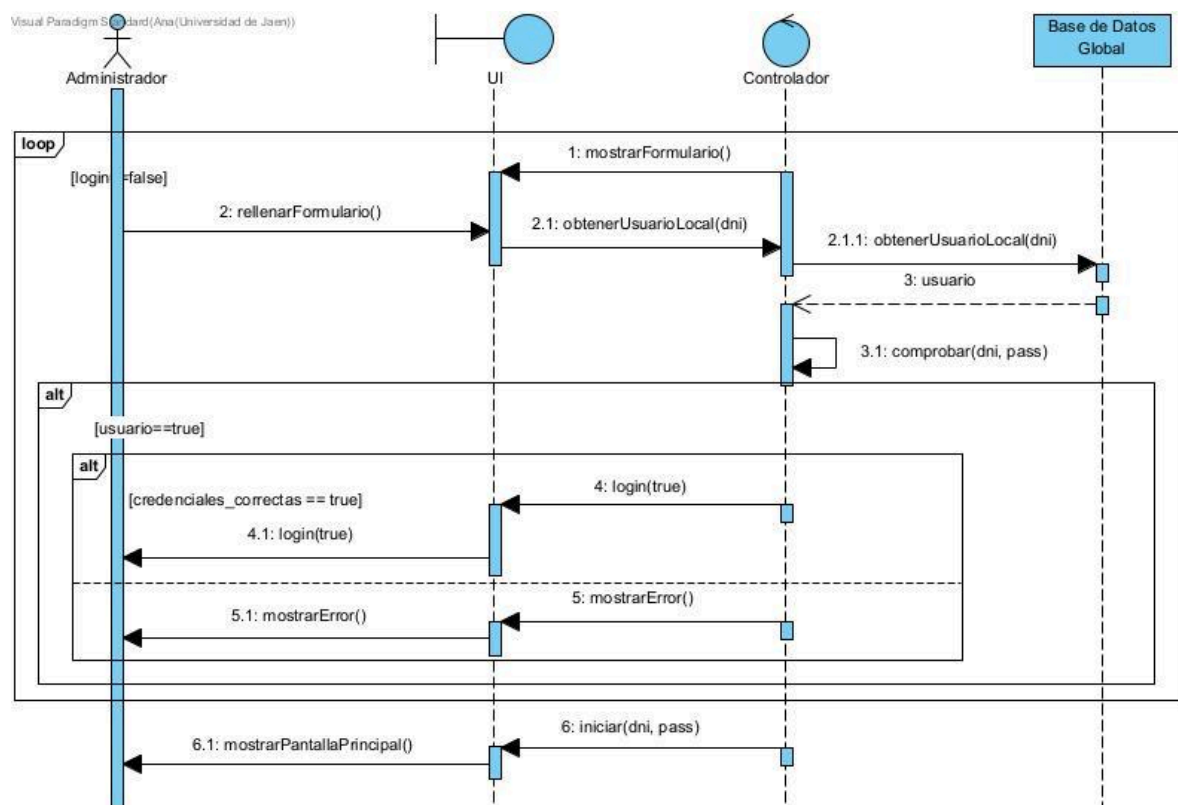


Imagen 26: Diagrama de secuencia para iniciar sesión en la aplicación web

- **Ver/añadir/borrar/editar usuarios (Imagen 27):** unas de las funcionalidad de la aplicación web será realizar operaciones CRUD sobre la entidad usuario, es decir, un administrador podrá insertar, eliminar, ver y/o editar un usuario. La estructura de este tipo de casos de uso en la aplicación web es mostrar las entidades en una tabla, seleccionar cada uno de los registros para eliminarlos o editarlos e insertar nuevos.

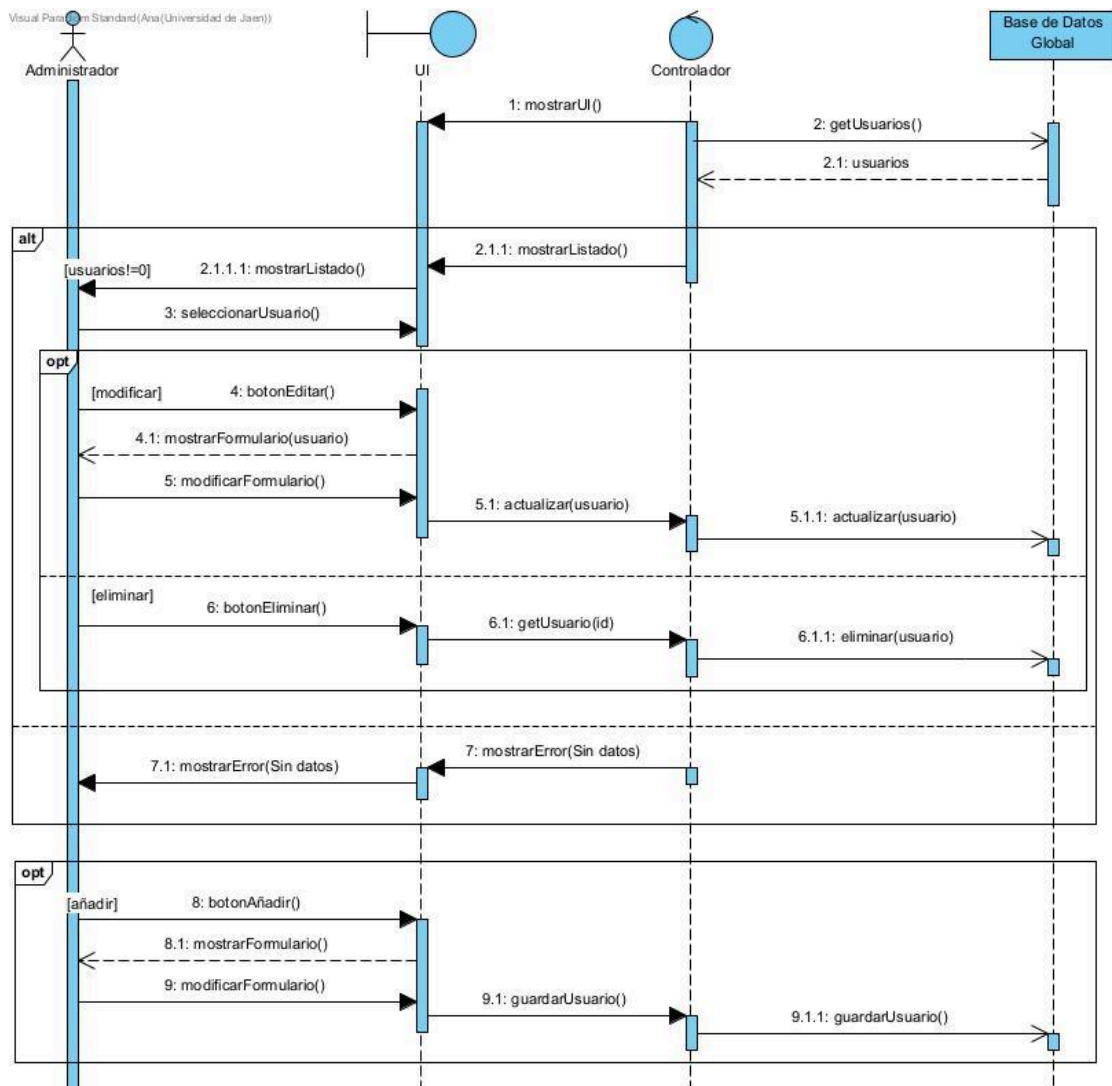


Imagen 27: Diagrama de secuencia para ver, añadir, borrar y editar usuarios en la aplicación web

- **Ver/añadir/borrar/editar parcelas (Imagen 28):** de igual modo que con los usuarios del sistema, se hace lo mismo con las parcelas que conforman las explotaciones de los agricultores. Se sigue la misma estructura, cada función estará operativa mediante un botón y dependiendo de la acción se mostrará el formulario o no.

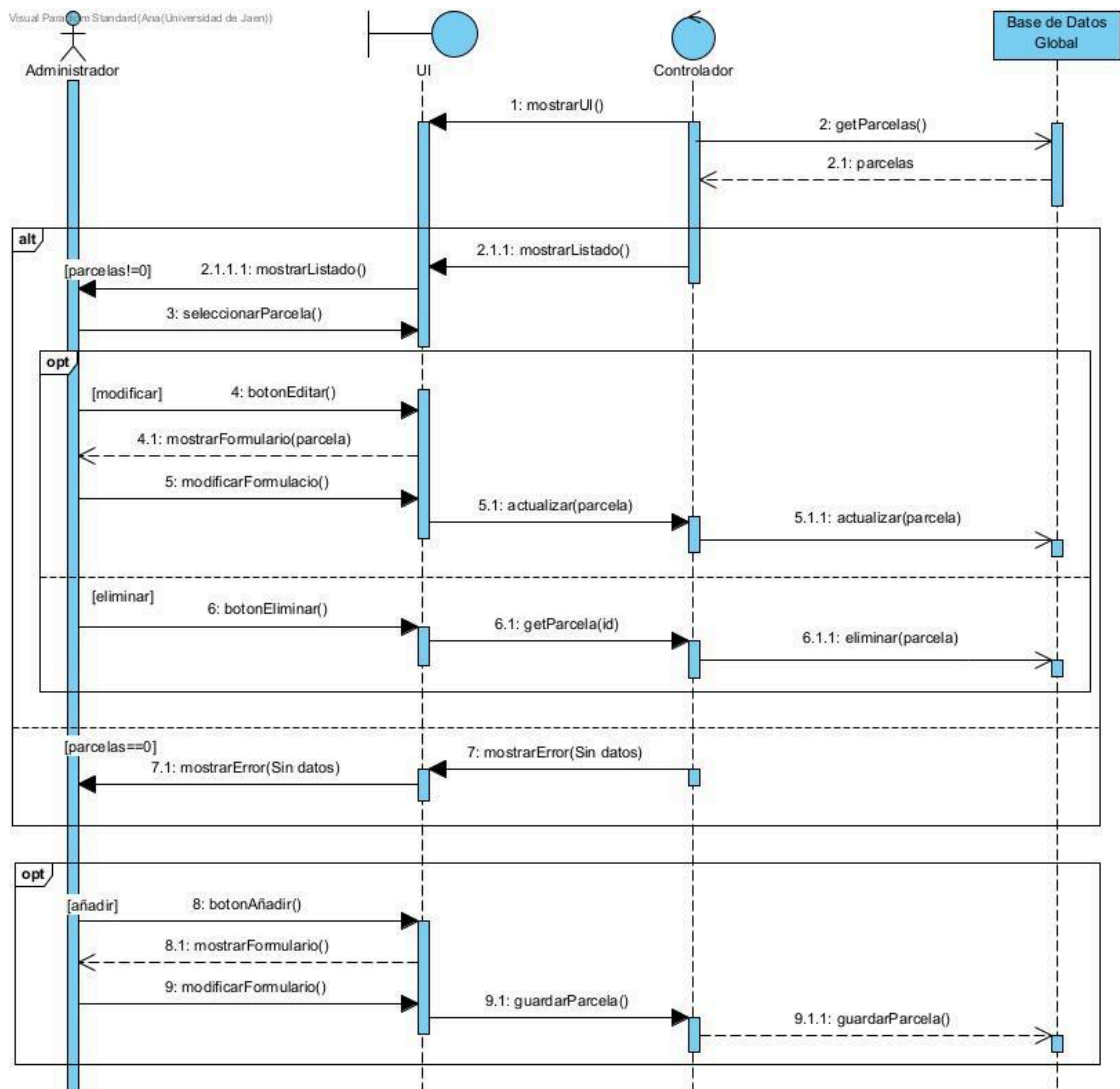


Imagen 28: Diagrama de secuencia para ver, añadir, borrar y editar parcelas en la aplicación web

- **Ver/añadir/borrar/editar cuadernos (Imagen 29):** repetimos el enfoque, pero esta vez para los cuadernos de campo que existen en el sistema. Se permiten las mismas operaciones.

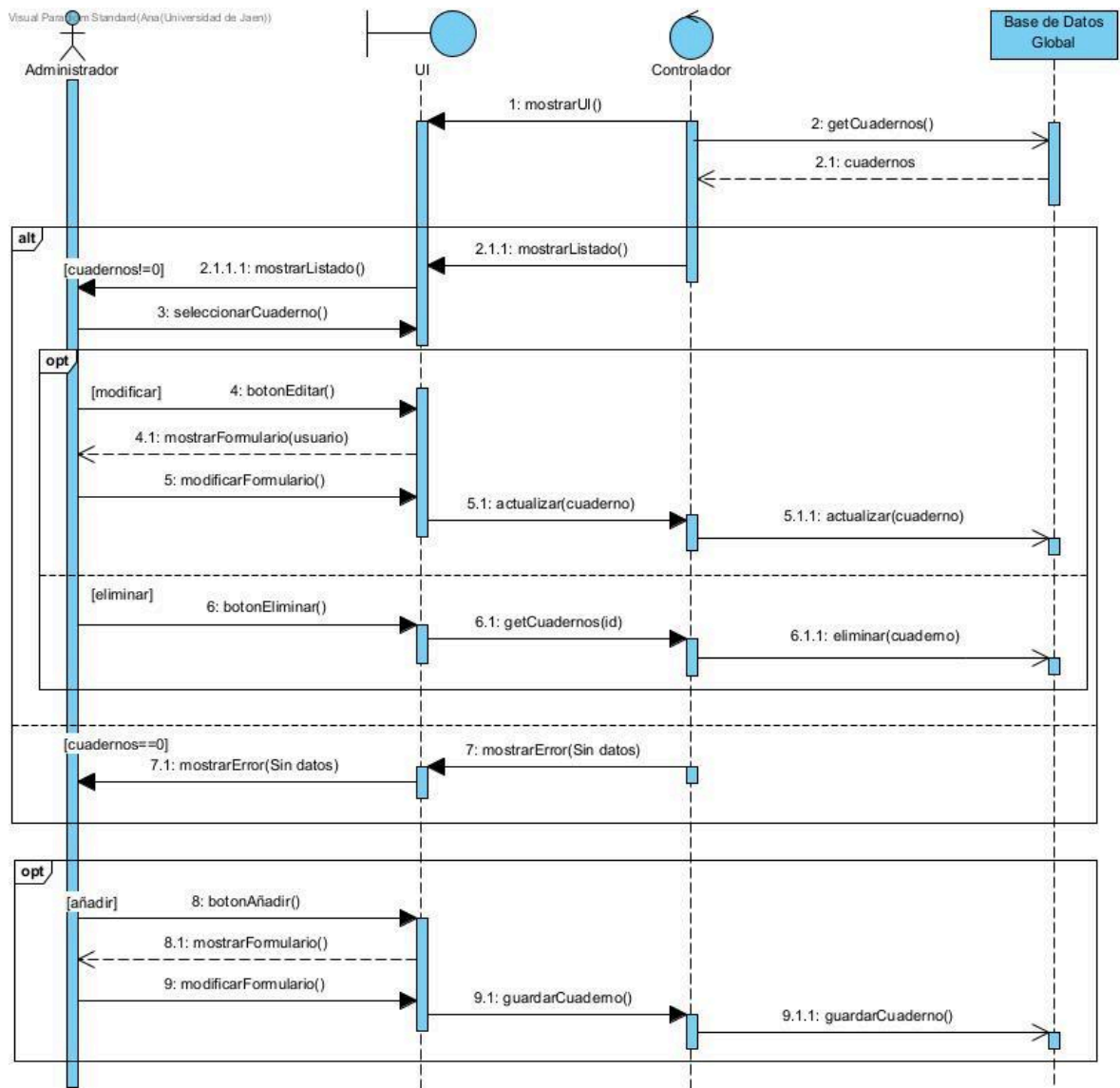


Imagen 29: Diagrama de secuencia para ver, añadir, borrar y editar cuadernos en la aplicación web

- **Ver/añadir/borrar/editar equipos (Imagen 30):** al igual que ocurre con usuario, parcelas y cuadernos, sobre equipos también se puede operar. Recordamos que un equipo representa a una herramienta que utiliza el agricultor para realizar sus labores en el campo.

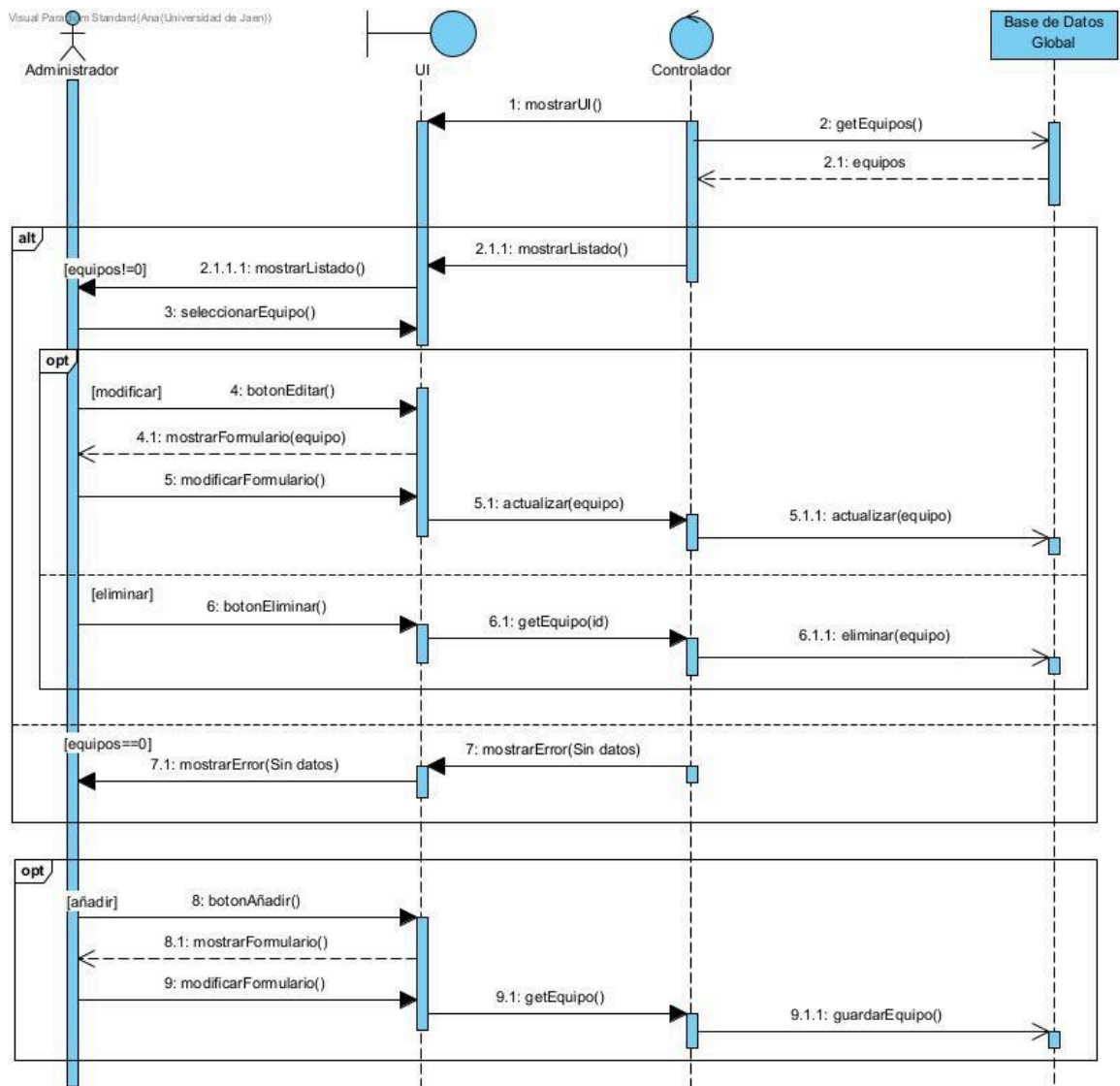


Imagen 30: Diagrama de secuencia para ver, añadir, borrar y editar equipos en la aplicación web

- **Ver/añadir/borrar/editar personal (Imagen 31):** por último, personal será la entidad la cual se le podrá realizar las cuatro operaciones siguiendo la misma estructura comentada hasta el momento.

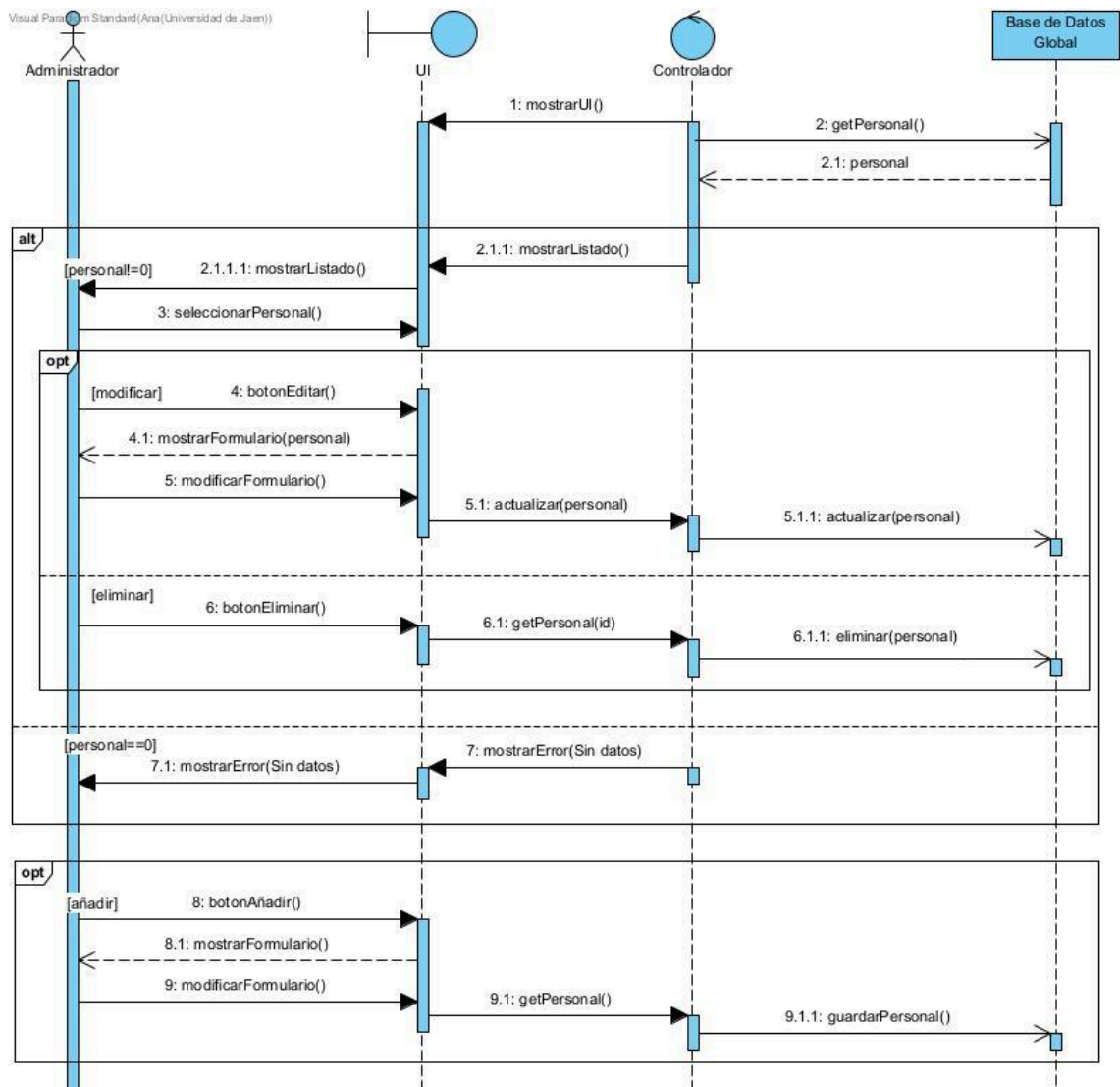


Imagen 31: Diagrama de secuencia para ver, añadir, borrar y editar personal en la aplicación web

- **Ver/añadir/borrar empresas (Imagen 32):** consideramos que las empresas son objetos muy estáticos, por lo tanto, solo se permitirá ver, añadir y/o borrar. Una empresa representa la entidad que comercializa con el producto que el agricultor obtiene de su explotación.

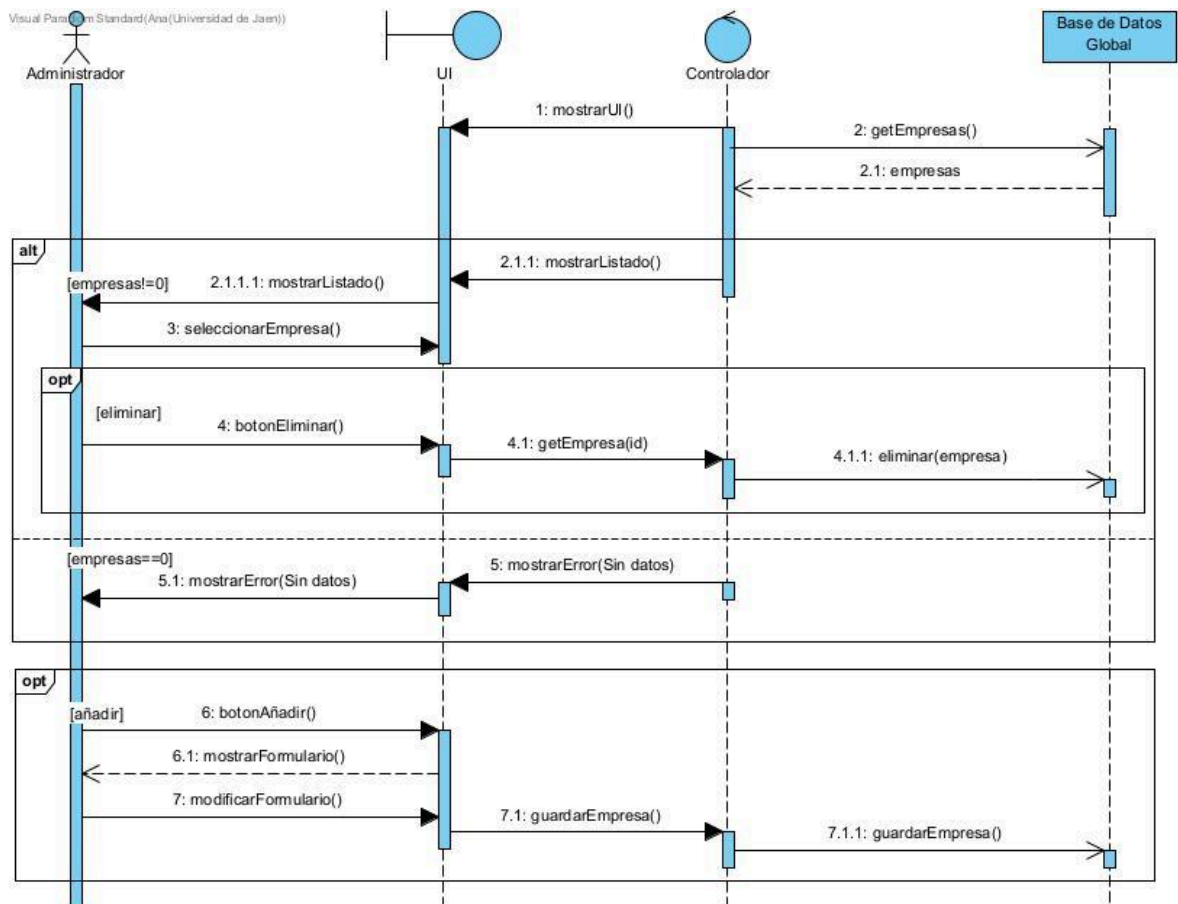


Imagen 32: Diagrama de secuencia para ver, añadir y borrar empresas en la aplicación web

- **Listar problemas en imágenes (Imagen 33):** una imagen será un problema que el agricultor ha detectado en el campo y ha considerado oportuno mostrar al técnico. En este diagrama se sigue el procedimiento básico para listar las imágenes en relación a un agricultor.
- **Historial de un cuaderno (Imagen 34):** de forma adicional, el administrador podrá visualizar un historial de forma resumida con toda la información que el agricultor y el técnico han ido generando a lo largo de un ejercicio concreto, es decir, se mostrará los datos relacionados con un cuaderno de campo desde la aplicación web.

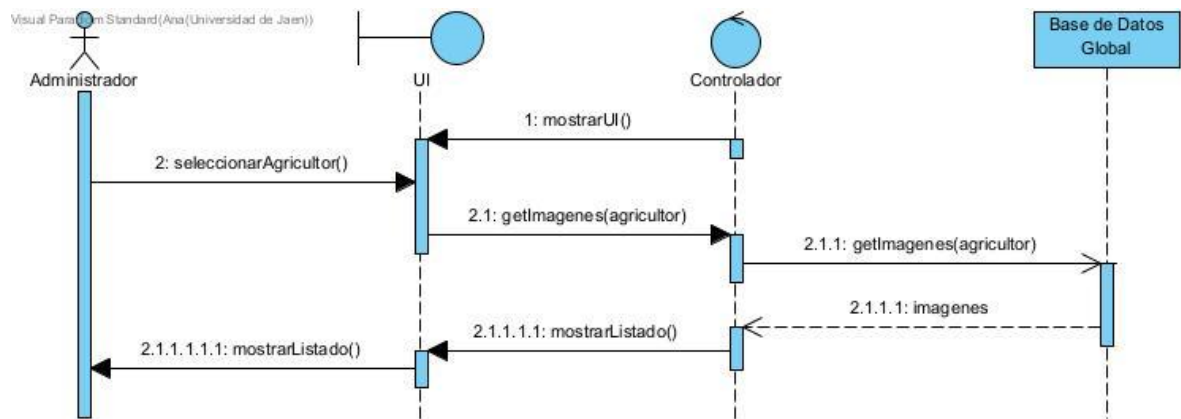


Imagen 33: Diagrama de secuencia para listar problemas en imágenes en la aplicación web

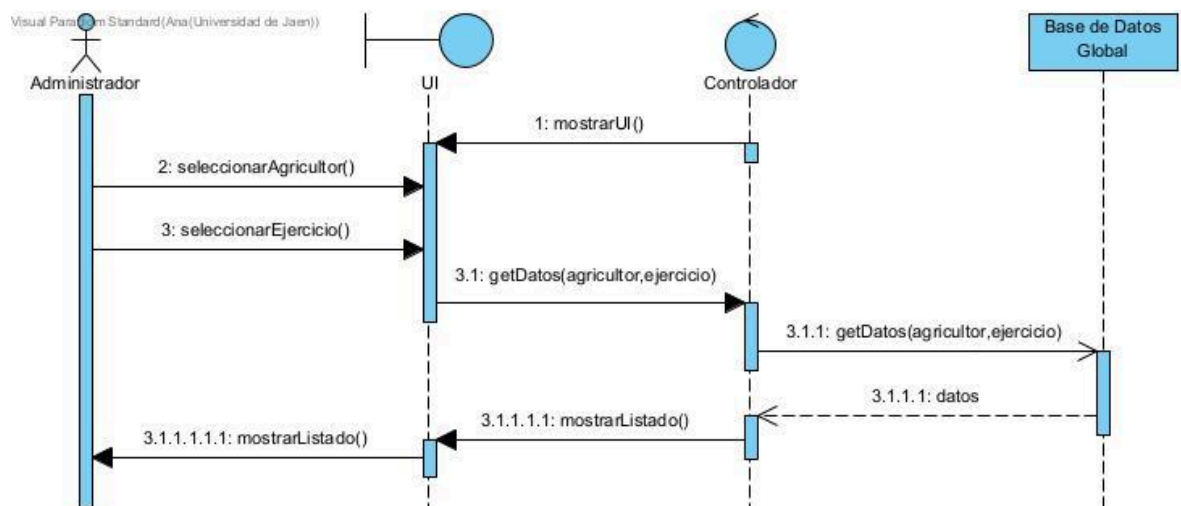


Imagen 34: Diagrama de secuencia para mostrar un historial de un cuaderno en la aplicación web

2.5. Análisis de tecnologías en sistemas operativos para móviles

Aprovechamos que estamos en el capítulo que concierne al análisis inicial del proyecto para discutir sobre las distintas tecnologías y lenguajes de programación disponibles. En primer lugar presentamos las posibilidades que encontramos para desarrollar una aplicación móvil, junto a sus ventajas e inconvenientes.

1. **Android:** Es un sistema operativo móvil basado en el núcleo Linux y otros software de código abierto. Inicialmente, fue desarrollada por la empresa Android Inc, que fue adquirida por Google LLC en 2005. Fue presentado en 2007 por la fundación Open Handset Alliance (OHA) con el objetivo de avanzar en los estándares abiertos de los dispositivos móviles. Ese mismo día se anuncia la primera versión, Android 1.0 Apple Pie y en 2011 se registra una cuota de mercado global del 50.6 %, más del doble que el segundo sistema operativo (iOS de Apple). En 2018 se supera ya los dos millones de aplicaciones disponibles en Google Play Store, la tienda de aplicaciones oficial de Android. Aunque inicialmente era el lenguaje de programación en Java, a día de hoy tenemos otras alternativas como C++, C, Apache Cordova o Kotlin [\[26\]](#).
2. **iOS:** es un sistema operativo de código cerrado desarrollado por Apple Inc, exclusivamente para los propios dispositivos móviles que la empresa fabrica, como son el iPhone, iPad o iPod. Apple reveló la existencia de iOS a principios de 2007, aunque no fue hasta 2008 cuando salió la primera versión beta. El interés del SDK aumentaría en meses siguientes debido al progresivo crecimiento de la plataforma iPhone, incrementado en 2007 por el lanzamiento de iPod y seguidamente en 2010 por el anuncio del iPad. Para desarrollar aplicaciones en iOS, tenemos la herramienta XCode, que sólo está disponible para ordenadores MacOS. Inicialmente, el lenguaje de programación oficial era Objective-C, basado en C/C++, pero fue reemplazado en 2014 por Swift. Lenguaje creado específicamente para iOS, cuya sintaxis es más sencilla que la de Objective-C [\[27\]](#).

La tecnología a utilizar será **Android**. Esta es una plataforma más **extendida**, ya que en el mercado existen muchas marcas de dispositivos que soportan este sistema operativo, lo que produce su abaratamiento. Por el contrario, iOS es más exclusivo porque solo se puede utilizar en dispositivos de la empresa Apple, bastante más caros. Además, a la hora del desarrollo se nos hace más complicado en iOS porque el **entorno de desarrollo** utilizado sólo está disponible para ordenadores Mac OS y no es nuestro caso. Sin embargo, el entorno para Android, Android Studio, sí se puede instalar en los sistemas operativos Windows y Linux. Otro de los factores clave para la lección es el tipo de **usuarios** al que va

dirigido nuestra aplicación, nuestros clientes generalmente tienen una media de edad alta y un nivel de estudio bajo. Consideramos que Android es un sistema operativo más fácil de aprender e intuitivo.

2.6. Análisis de tecnologías y herramientas para el desarrollo web

En este apartado vamos a explicar las distintas tecnologías y herramientas que hemos utilizado para el desarrollo de la aplicación web.

2.6.1. Visual Studio Code

Visual Studio Code es un editor de código fuente gratuito y de código abierto desarrollado por Microsoft para los sistemas operativos Windows, Linux y MacOS. Entre su potencial encontramos el soporte a la depuración, control de código fuente integrado mediante GIT, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código. Una de sus grandes ventajas es su elevado grado de personalización por parte del usuario. Existe una gran cantidad de extensiones mediante las cuales se pueden agregar funciones y características al editor [\[28\]](#).

2.6.2. HTML

HTML o *HyperText Markup Language* es un lenguaje de marcado utilizado en la creación de páginas web. Define una estructura básica y un código para la presentación de contenido en una página web, que incluye texto, imágenes, vídeos, juegos, entre otros elementos. Este estándar es gestionado por el *World Wide Web Consortium (W3C)* o *Consortio WWW*, una organización dedicada a la estandarización de la mayoría de las tecnologías asociadas a la web.

El lenguaje HTML se fundamenta en la diferenciación como filosofía de desarrollo, es decir, la página web sólo contiene texto dejando al navegador web la responsabilidad de unir todos los elementos referenciados a su ubicación y visualizarlos [\[29\]](#).

2.6.3. CSS

CSS o Cascading Style Sheets, en español, hojas de estilo en cascada, es un lenguaje de diseño gráfico para definir y crear la presentación de un documento estructurado escrito en un lenguaje de marcado como HTML. CSS está diseñado principalmente para marcar la separación del contenido del documento y la forma de presentación de este, características tales como las capas o layouts, los colores y las fuentes. Buscando mejorar la accesibilidad del documento, proveer más flexibilidad y control en la especificación visual, permitir que varios documentos HTML compartan un mismo estilo usando una sola hoja de estilos separada en un archivo .css y reducir la complejidad y la repetición de código en la estructura del documento [30].

En términos de estilo hemos utilizado el marco de trabajo Bootstrap. Bootstrap es un conjunto de herramientas de código abierto para diseño de sitios web y aplicaciones web. Contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en HTML y CSS [31].

2.6.4. JavaScript

JavaScript (abreviado comúnmente JS) es un lenguaje de programación interpretado, orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico utilizado en el lado del cliente. Su función es mejorar en la interfaz de usuario y producir páginas web dinámicas [32].

2.6.5. Vue.js

Para el desarrollo de la aplicación web trabajaremos con el entorno de trabajo llamado Vue.js, comúnmente conocido como Vue. Es una herramienta de código abierto basada en JavaScript cuya funcionalidad es la construcción de interfaces de usuario y aplicaciones de una sola página. Se centra en la representación declarativa y la composición de componentes. Las características avanzadas necesarias para aplicaciones complejas como el enrutamiento, la gestión de estados y las herramientas de construcción se ofrecen a través de librerías y paquetes de apoyo mantenidos oficialmente.

Vue.js permite extender el HTML con atributos llamados directivas. Las directivas ofrecen funcionalidad a las aplicaciones HTML, y vienen como directivas incorporadas o definidas por el usuario [33].

2.6.6. Node.js

Node.js es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor. Basado en el lenguaje de programación JavaScript, asíncrono, con entrada y salida de datos en una arquitectura orientada a eventos y basado en el motor V8 de Google. Fue creado en 2009 con el enfoque de ser útil en la creación de programas de red altamente escalables, como por ejemplo, servidores web.

Actualmente, se ha convertido en un pilar fundamental para el desarrollo de aplicaciones web modernas ya que su capacidad para manejar operaciones asíncronas y su versatilidad lo hacen indispensable. Con Node.js JavaScript se convierte en una herramienta universal para el desarrollo tanto del *frontend* como del *backend* [34].

2.7. Análisis de tecnologías en sistemas gestores de base de datos

Una **base de datos** es un modelo de algún aspecto de una organización. Esta definición la dio Kent en 1978. Habitualmente a esta realidad se le llama universo del discurso o dominio del discurso. Este dominio estará compuesto por clases, atributos y relaciones. Una base de datos se considera una colección concreta de datos dispuesta de forma organizada, de lo cuales podemos crear conocimiento.

La unidad mínima dentro de la realidad que abstrae una base de datos es el dato o hecho. Esta unidad carece de significado, los datos por sí solos son inútiles, es decir, necesitamos interpretarlos y situarlos en un contexto con significado, creando así la información. Para trabajar con dicha información es necesario disponer de un **Sistema Gestor de Bases de Datos (SGBD)**. Un SGBD es un programa informático compuesto por un conjunto de programas que

permiten acceder y gestionar los datos asegurando la integridad, control de concurrencia, seguridad y gestión de usuarios.

Como ya está descrito en los requisitos del proyecto, Apartado [2.3](#), existirán dos aplicaciones, una aplicación móvil y otra web, y para ambas será necesario almacenar información. Para la aplicación móvil tendremos una base de datos local en el dispositivo y **SQLite** es soportado por Android. SQLite es una herramienta que permite almacenar información en dispositivos empujados, es decir, de forma eficaz, potente, rápida y en equipos con pocas capacidades hardware, como puede ser un teléfono móvil. Sin embargo para la base de datos global nos podemos encontrar:

1. **Oracle Database:** es un SGBD relacional orientado a objetos creado por Oracle, optimiza la gestión y seguridad de los conjuntos de datos creando esquemas estructurados a los que solo pueden acceder administradores autorizados. Oracle surge en 1977 bajo el nombre de SDL. En 1979 fue creado dicho sistema, motivado por el estudio de George Koch sobre SGBD. Dicho estudio está considerado como uno de los más complejos jamás escrito sobre base de datos, ya que incluía una comparativa de diferentes productos destacando el sistema relacional como el más completo desde el punto de vista técnico. Durante mucho tiempo ha sido el sistema rey en el mercado, aunque su dominio empezó a disminuir cuando aparecieron otros competidores con licencia 'Open Source' [\[35\]](#).
2. **PostgreSQL:** otro SGBD relacional orientado a objetos y de código abierto, nacido en 1996, que utiliza y amplía el lenguaje SQL combinado con muchas funciones que almacenan y escalan de forma segura las cargas de trabajo de datos más complicadas. PostgreSQL se ha ganado una sólida reputación por su arquitectura comprobada, confiabilidad, integridad de datos y la dedicación de la comunidad de código abierto detrás del software. Se ejecuta en todos los principales sistemas operativos y no tiene un gestor de errores, haciendo muy difícil conocer el estado de corrección de los mismos [\[36\]](#).

En conclusión, el SGBD elegido es Oracle Database. El principal motivo es la **experiencia**, este tipo de SGBD ya es conocido para nosotros y nos resultaba

más familiar. A parte la gran ventaja que vemos es la posibilidad de beneficiarnos de una cuenta de forma gratuita, mediante fines académicos, para acceder a toda la ventaja de Oracle en la **nube**. Esto nos permitirá exponer nuestra base de datos y facilitar la comunicación entre aplicaciones. Las características y uso de Oracle Database Cloud se detallará debidamente en el momento que se implemente la base de datos a utilizar, en concreto, en el Apartado [3.2.6](#).

2.8. Arquitectura final

Llegados a este punto donde ya hemos desmembrado y escogido las tecnologías a usar, ya estamos en disposición de definir la arquitectura global del proyecto en cuanto a tecnología.

Lo que intenta explicar la Imagen [35](#) es la comunicación de la aplicación móvil y web a través de internet y mediante la API y base de datos desplegada en la nube. En nuestro caso vamos a mantener una arquitectura cliente-servidor basado en el almacenamiento en la nube y sus funciones son:

- El **servidor** tendrá instalado el entorno Node.js que actuará como *backend* de la aplicación web, cuya parte de interfaz de usuario o *frontend* será responsabilidad de Vue.js, basado en los lenguajes HTML, CSS y JavaScript.
- El **almacenamiento en la nube** es ofrecido por Oracle y permitirá tener acceso y control a una instancia de la base de datos como recurso virtual accesible desde Internet, es decir, no como recurso local. Además, Oracle Cloud nos brinda el servicio API REST, lo que nos facilita la definición de la comunicación entre clientes y base de datos. Este elemento es el que denominamos en el proyecto como base de datos global.
- Tendremos dos tipos de **clientes**, una aplicación móvil desarrollada por Android Studio con una pequeña base de datos local SQLite para el agricultor y técnico y una aplicación web para el administrador accesible desde Internet. De forma auxiliar el administrador hará uso del programa QGIS localmente, con el objetivo de ayudar a la obtención de la información que deberá registrar. Explicaremos con más detenimiento el uso de QGIS en nuestro proyecto, concretamente en el Apartado [5.2](#).

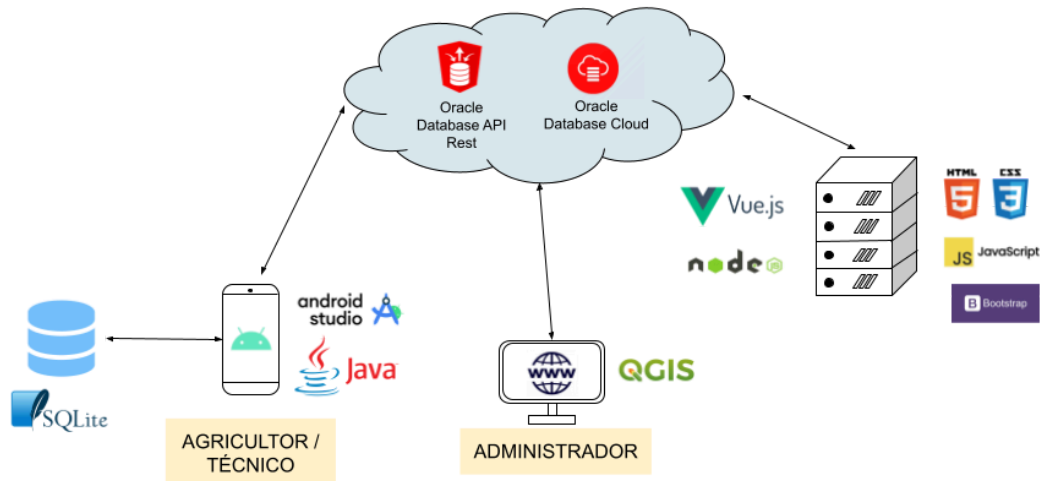


Imagen 35: Arquitectura final del conjunto del sistema

2.9. Planificación temporal

En lo que respecta a la etapa de planificación, una de las más relevantes durante la puesta en marcha de un proyecto y, además, la que acomete mayor incertidumbre. Por ello, se realizará una estimación del tiempo dedicado al desarrollo de este presente proyecto.

Debido a la metodología de trabajo escogida (ver Apartado [1.7](#)) y la estructura clara de capítulos y apartados que establece la norma UNE 157801:2007, podemos clasificar el proyecto en los siguientes incrementos o *scripts*:

1. Documentación.
 - a. Búsqueda de información sobre el objetivo y contexto del proyecto.
 - b. Búsqueda de información relacionada sobre los antecedentes y estado del arte del proyecto.
2. Análisis Inicial.
 - a. Desglose de los perfiles de usuarios y sus requisitos dentro del sistema.
 - b. Análisis de las tecnologías ofrecidas para las distintas aplicaciones.
 - c. Definir la arquitectura final del proyecto.
3. Desarrollo del sistema.
 - a. Primera iteración: modelo y esquema de la base de datos.

- b. Segundo iteración: diseño de las interfaces de usuario.
 - c. Tercera iteración: aplicación móvil.
 - d. Cuarta iteración: aplicación web.
4. Conclusiones y anexos.
- a. Redactar los trabajos futuros que se pueden aplicar y los manuales pertinente útiles para el lector del proyecto.

A continuación se expone toda la estimación de tiempos en un diagrama de Gantt en la Imagen [36](#). Un diagrama de Gantt es un herramienta de planificación y gestión de proyectos que ayuda a esto mismo, visualizar la distribución y duración de las tareas en el tiempo.

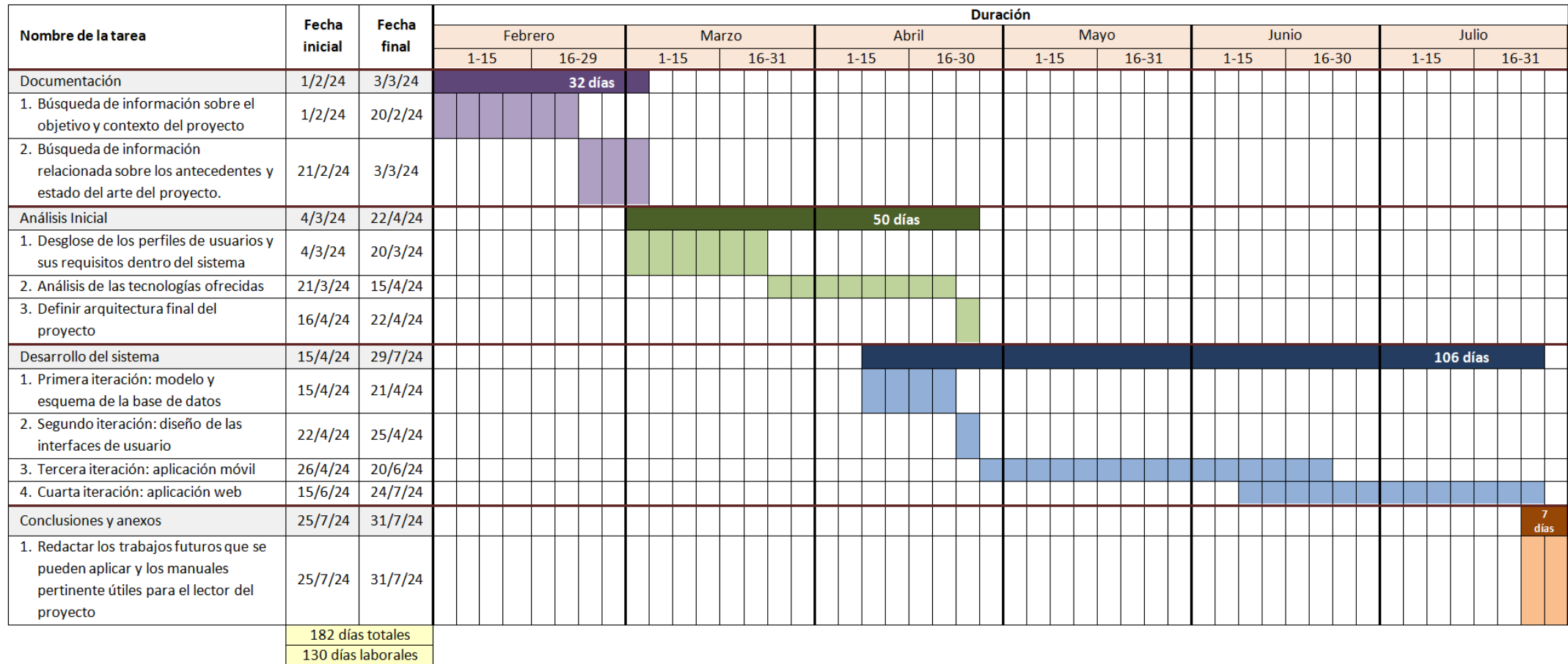


Imagen 36: diagrama de Gantt con la estimación de duración del proyecto

2.10. Presupuesto

En este último apartado del segundo capítulo vamos a calcular el coste total que tendría nuestro proyecto. Este coste será la suma del coste hardware, software y humano o personal utilizado a lo largo del mismo.

El **coste hardware** corresponde al gasto invertido en dispositivos tecnológicos necesarios para realizar los proyectos. En la Tabla 2 englobamos todos los gastos de todos los componentes, indicando su tipo y precio individual.

Componente	Tipo	Precio
Lenovo ThinkPad E15 i7-10510UU/16GB/512GB SSD/15.6"	Ordenador portátil	1.139,00 €
Logitech MX Master 3S	Ratón	83,00 €
AOC 24P2QM 23.8"LED FullHD 75Hz	Pantalla	150,00 €
Xiaomi Redmi Note 12	Teléfono móvil	175,00 €
Total		1.547,00 €

Tabla 2: Costes hardware

El **coste software** representa el precio de las licencias solicitadas, en nuestro caso, casi todas las licencias son académicas y sin coste. En la Tabla 3 resumimos dichos gastos.

Componente	Tipo	Precio
Windows 10	Sistema operativo	168,99 €
Android Studio	Entorno de desarrollo móvil	0 €
Visual Studio Code	Entorno de desarrollo web	0 €
Oracle Cloud	Desarrollo base de datos y API	0 €
Sql Developer	Gestión de base de datos	0 €
Visual Paradigm	Diagramas	0 €
Total		168,99 €

Tabla 3: Costes software

Por último se anotará el **coste humano** o de los distintos perfiles profesionales que formaría el equipo de desarrollo (ver Tabla 4). El equipo de desarrollo para este trabajo está formado por una persona, la autora de este proyecto, encargada de realizar las actividades correspondientes a tres distintos roles, que son:

- Gestor de proyectos: coordina los procesos, las herramientas, los miembros del equipo y las capacidades para entregar proyectos que cumplan los objetivos y satisfagan los requisitos.
- Analista programador: es el encargado de generar y administrar documentación e informes y elaborar estudios de viabilidad financiera relacionados con los proyectos.
- Programador junior: es la persona que desarrolla los sistemas e implica entender los problemas, concebir la solución y traducir estas ideas en programas eficientes y eficaces.

Aunque los salarios base se pueden consultar en el Boletín Oficial del Estado, hemos consultado una página web de la empresa Indeed. Indeed es una plataforma de búsqueda de empleo que también ofrece un buscador de sueldos totalmente actualizados y basados en sueldos reales. Todos los datos se obtienen con referencia en España.

Por otro lado debemos de hallar las **horas trabajadas** en el proyecto. Según la guía docente relativa a la asignatura Trabajo Fin de Grado de la Escuela Politécnica Superior de Jaén, consta de 12 créditos, equivalentes a 300 horas de trabajo. En nuestro caso, planificamos 130 días de trabajo a una media diaria de 3.5 horas, sumando un total de 455 horas trabajadas en el proyecto.

Puesto	Salario anual	Salario por hora	Horas trabajadas	Coste
Gestor de proyectos ⁷	48.051,00 €	23,93 €	75	1.797,00 €
Analista programador ⁸	35.378,00 €	17,64 €	110	1.764,00 €
Programador junior ⁹	23.668,00 €	11,84 €	270	3.196,80 €
Total			455	6.757,80 €

Tabla 4: Coste humano

Las herramientas hardware y software tienen una vida útil mayor que la propia duración del proyecto, por lo que el coste de compra no tiene sentido introducirlo directamente en el presupuesto, sino el importe amortizado según el tiempo que se haya usado.

A continuación, en la Tabla 5, se indican los cálculos correspondientes a las amortizaciones de los costes hardware y software.

⁷ https://es.indeed.com/career/gestor-de-proyectos/salaries?from=top_sb

⁸ https://es.indeed.com/career/analista-programador/salaries?from=top_sb

⁹ https://es.indeed.com/career/programador-junior/salaries?from=top_sb

Gasto	Coste total	Vida útil	Tiempo de uso	Amortización
Ordenador portátil	1.372,00 €	2.190 días (6 años)	130 días	81,44 €
Teléfono móvil	175,00 €	1.460 días (4 años)	106 días	12,71 €
Windows 10	168,99 €	3.650 días (10 años)	130 días	6,02 €
Total				100,17 €

Tabla 5: Amortización de los coste hardware y software

Finalmente, en la Tabla [6](#), resumimos el coste total estimado clasificado en partes.

	Gasto
Coste hardware	94,15 €
Coste software	6,02 €
Coste humano	6.757,80 €
Total	6.857,97 €

Tabla 6: Coste total del proyecto

CAPÍTULO 3

DESARROLLO



3.1. Introducción

Llegados a este punto del proyecto se han estudiado cada uno de los aspectos teóricos con detenimiento. Es momento de comenzar a desarrollar de forma técnica todo lo recopilado. En base a la metodología de desarrollo (ver Sección [1.7](#)), vamos a dividir el trabajo en cuatro iteraciones:

- **Iteración 0 Documentación:** Antes de comenzar con el desarrollo técnico del proyecto es imprescindible un análisis previo. En el Capítulo [1](#) realizamos búsquedas de información sobre los objetivos y contexto que nos engloba y definimos los requisitos a cumplir. Dentro de esta iteración también incluimos el desarrollo del Capítulo [2](#), en él discutiremos la selección de las distintas tecnologías, arquitecturas disponibles para cada tipo de aplicación y documentamos en profundidad cada funcionalidad del sistema en el Capítulo [2](#).
- **Iteración 1 Modelo de datos:** en esta iteración se pretende desarrollar los aspectos relacionados con la base de datos. Es decir, las definiciones de los conceptos a utilizar con el objetivo de diseñar la estructura y relaciones de la base de datos. Por último, se detalla el procedimiento, plataforma y métodos seguidos en su implantación.
- **Iteración 2 Diseño de la interfaz de usuario:** se explicará el procedimiento seguido para diseñar las interfaces que el usuario visualizará en sus pantallas, tanto en la aplicación móvil como en la web.
- **Iteración 3 Aplicación móvil:** explicaremos la implementación de la aplicación móvil.
- **Iteración 4 Aplicación web:** igualmente, se detallará la manera de implementar la aplicación web con las tecnologías escogidas.

Además, de forma implícita, finalizada la cuarta iteración tenemos la integración de todo el sistema.

3.2. Primera iteración: Modelo de datos

Comenzamos con la primera iteración llamada "Modelo de datos". Como su propio nombre indica vamos a crear la infraestructura necesaria para almacenar y referenciar la información de nuestro sistema.

Lógicamente, lo primero será diseñar la base de datos, es decir, qué datos almacenar y cómo se relacionan entre ellos. También, se explicará el concepto de API y cómo se ha utilizado en nuestro proyecto.

3.2.1. Introducción

El análisis de una base de datos consta de varias fases. En primer lugar, un análisis abstracto de aquellas entidades que formarán la base de datos, así como las relaciones que establecen unas con otras y sus consecuentes restricciones. El resultado de esta fase se ve reflejado en el llamado modelo conceptual. Existen diferentes modelos, pero, en general, un modelo de datos es un grupo de herramientas conceptuales que nos sirven para describir los datos, sus relaciones, su semántica y sus límites.

En nuestro caso, trabajaremos con un **modelo relacional**, desarrollado por Edgar Codd, entonces investigador de la IBM a principios de los años 70. Este tipo de modelo precede a los modelos jerárquicos y en red, los cuales presentaban ciertas deficiencias de flexibilidad y lentitud en el acceso a la información.

Gráficamente, un modelo relacional de una base de datos se representa mediante un **diagrama Entidad-Relación**. Entre sus ventajas encontramos la flexibilidad y la claridad a la hora de especificar las limitaciones de los datos. Como su propio nombre indica, se basa en la concepción del mundo real como un conjunto de entidades y de relaciones entre ellas. Cada **entidad** posee ciertos **atributos** que la hacen diferente del resto. La definición de **relaciones** hace que las entidades implicadas tengan una o varias **restricciones**, causada por su propia naturaleza o característica del problema [24].

Como nos indican los objetivos de este presente proyecto el desarrollo consta de dos aplicaciones, móvil y web. Ambas tendrán el mismo esquema de información porque los datos se comparten, luego, con un mismo diseño abordaremos ambas aplicaciones.

3.2.2. Nivel conceptual

El nivel conceptual de una base de datos se conoce como la descripción de todos las entidades y relaciones existentes entre la información almacenada. Como consecuencia, en la Imagen [37](#) se visualiza el diagrama Entidad-Relación del modelo de datos utilizado.

Durante el desarrollo del presente TFG hubo una actualización en la plantilla que ofrece la Junta de Andalucía como CUE. Ante esta actualización vimos conveniente modificar la forma de organizar los datos, principalmente los relacionados con los tratamientos fitosanitarios. Concretamente, es añadido un apartado para la fertilización y un registro del riego proporcionado a cada parcela. En la Sección [1.6.3](#) ya se explica la información que aparece en el cuaderno de campo más exhaustivamente, teniendo en cuenta esta última actualización.

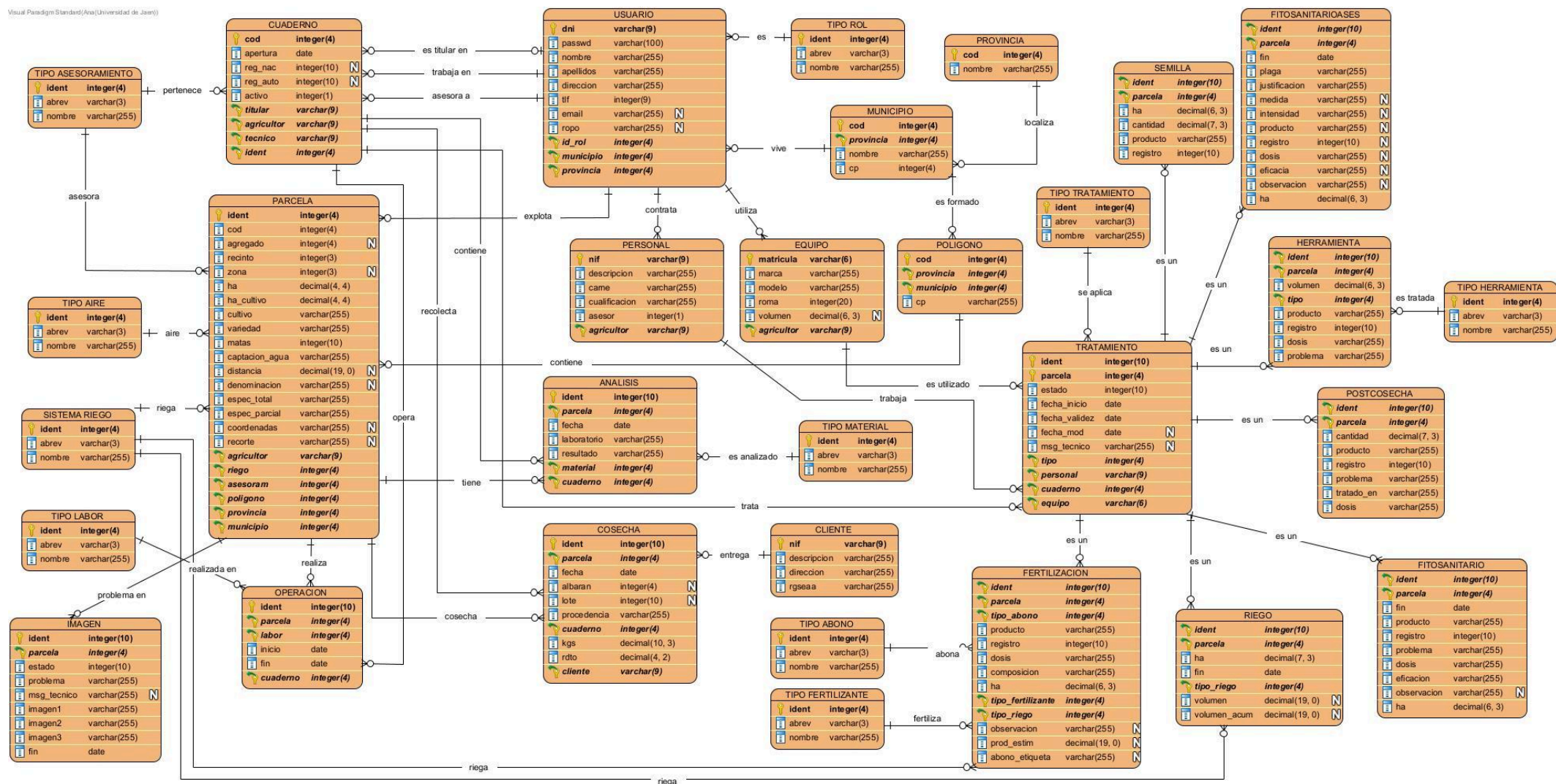


Imagen 37: Diagrama Entidad-Relación

3.2.3. Nivel lógico

El nivel lógico de una base de datos es un nivel intermedio entre el nivel conceptual y físico. La correspondencia directa del nivel conceptual es la información estructurada en **tablas**, perteneciente al nivel lógico. Cada tabla se divide horizontalmente en **filas o registros** y verticalmente en **columnas o campos**, conjuntamente describen una instancia de una entidad determinada.

A continuación, para cada una de las entidades obtenidas del diagrama definitivo Entidad-Relación (ver Imagen [37](#)) se completará una tabla. Con estas tablas pretendemos exponer con claridad el objetivo de cada entidad y sus atributos.

CUADERNO (Tabla 7): entidad que almacena los datos generales de la explotación en un ejercicio completo.

CUADERNO			
Campo	Tipo de dato	Restricción	Significado
cod	Entero	Clave primaria	Identificador único
apertura	Fecha	No nulo	Fecha de apertura
ejercicio	Entero	No nulo	Año que engloba la campaña
reg_nac	Cadena	Nulo	Número del Registro de Explotaciones Nacional
reg_auto	Cadena	Nulo	Número del Registro de Explotaciones Autonómico
titular	Cadena	Clave foránea	Titular de la explotación definida en el cuaderno
agricultor	Cadena	Clave foránea	Agricultor responsable de la explotación definida en el cuaderno
tecnico	Cadena	Clave foránea	Técnico agrónomo que asesora la explotación definida en el cuaderno
asesoram	Entero	Clave foránea	Tipo de asesoramiento que se aplicará en el cuaderno

activo	Entero	No nulo	Indica si el cuaderno está activo, es decir, no se ha cerrado el ejercicio.
--------	--------	---------	---

Tabla 7: Resumen de la entidad Cuaderno

USUARIO (Tabla 8): entidad que representa la información personal de cualquier perfil que trabaja con la aplicación móvil o web.

USUARIO			
Campo	Tipo de dato	Restricción	Significado
dni	Cadena	Clave primaria	Identificador único del usuario
passwd	Cadena	No nulo	Contraseña para iniciar sesión
nombre	Cadena	No nulo	Nombre del usuario
apellidos	Cadena	No nulo	Apellidos del usuario
municipio	Entero	Clave foránea	Código del municipio del domicilio
provincia	Entero	Clave foránea	Código de la provincia a la que pertenece el municipio
direccion	Cadena	No nulo	Dirección del domicilio
tfl	Entero	No nulo	Teléfono de contacto
email	Cadena	Nulo	Correo electrónico de contacto
ropo	Cadena	Nulo	Número de identificación en el ROPO
id_rol	Entero	Clave foránea	Tipo de rol que cumple en el sistema

Tabla 8: Resumen de la entidad Usuario

TIPO_ROL (Tabla 9): roles de usuario que están presentes en el sistema. Esto definirá sus permisos y posibilidades en ambas aplicaciones.

TIPO_ROL			
Campo	Tipo de dato	Restricción	Significado

ident	Entero	Clave primaria	Identificador único
abrev	Cadena	No nulo	Abreviatura del rol
nombre	Cadena	No nulo	Nombre completo del tipo de rol

Tabla 9: Resumen de entidad Tipo Rol

MUNICIPIO (Tabla 10): municipio que identificará el lugar del domicilio del usuario, también se utilizará para localizar cada polígono.

MUNICIPIO			
Campo	Tipo de dato	Restricción	Significado
cod	Entero	Clave primaria	Código del catastro único por provincia
provincia	Entero	Clave primaria y foránea	Provincia a la que pertenece el municipio
nombre	Cadena	No nulo	Nombre del municipio
cp	Entero	No nulo	Código postal

Tabla 10: Resumen de la entidad Municipio

PROVINCIA (Tabla 11): información relativa a las provincias de los municipios. Las provincias que serán almacenadas serán solo andaluzas, por lo tanto se identificará por el código del catastro.

PROVINCIA			
Campo	Tipo de dato	Restricción	Significado
cod	Entero	Clave primaria	Código del catastro único por comunidad autónoma
nombre	Cadena	No nulo	Nombre de la provincia

Tabla 11: Resumen de la entidad Provincia

PERSONAL (Tabla 12): dentro de una explotación se identificarán a las personas que trabajan aplicando los distintos tratamientos o labores.

PERSONAL			
Campo	Tipo de dato	Restricción	Significado
nif	Cadena	Clave primaria	Código NIF de la empresa a la que pertenece o DNI de la persona
descripcion	Cadena	No nulo	Nombre de la empresa o persona
carne	Cadena	No nulo	Número del carné o del registro ROPO, si se trata de una empresa
cualificacion	Cadena	No nulo	Tipo de cualificación fitosanitaria
asesor	Entero	No nulo	Indica si esta persona se trata de un asesor o no
agricultor	Cadena	Clave foránea	Agricultor o jefe del trabajador

Tabla 12: Resumen de la entidad Personal

EQUIPO (Tabla 13): equipos de trabajo y de transporte utilizados en las tareas de la explotación.

EQUIPO			
Campo	Tipo de dato	Restricción	Significado
matricula	Cadena	Clave primaria	Matrícula del vehículo
marca	Cadena	No nulo	Marca del equipo
modelo	Cadena	No nulo	Modelos del equipo
volumen	Decimal	Nulo	Volumen del equipo si tiene
roma	Entero	No nulo	Número de registro ROMA
adquisicion	Date	Nulo	Fecha de adquisición del vehículo
inspección	Date	Nulo	Fecha de la última inspección
agricultor	Cadena	Clave foránea	Agricultor dueño del equipo

Tabla 13: Resumen de la entidad Equipo

POLIGONO (Tabla 14): entidad que localiza el lugar dónde se encuentra una parcela.

POLIGONO			
Campo	Tipo de dato	Restricción	Significado
cod	Entero	Clave primaria	Código del polígono proveniente del catastro
municipio	Entero	Clave primaria y foránea	Municipio donde se localiza el polígono
provincia	Entero	Clave primaria y foránea	Provincia donde se localiza el polígono
paraje	Cadena	No nulo	Nombre que caracteriza a dicho polígono

Tabla 14: Resumen de la entidad Polígono

PARCELA (Tabla 15): representan toda la información de las parcelas que forman una explotación y están asociadas a un agricultor del sistema.

PARCELA			
Campo	Tipo de dato	Restricción	Significado
ident	Entero	Clave primaria	Identificador único
agricultor	Entero	Clave foránea	DNI del agricultor responsable de la parcela
cod	Entero	No nulo	Código de la parcela en los datos SIGPAC
poligono	Entero	Clave foránea	Código del polígono proveniente del catastro
municipio	Entero	Clave foránea	Código del municipio al que pertenece el polígono
provincia	Entero	Clave foránea	Código de la provincia a la que pertenece el polígono
agregado	Entero	Nulo	Agregado de la parcela
recinto	Entero	No nulo	Recinto de la parcela
zona	Entero	Nulo	Zona de la parcela
ha	Decimal	No nulo	Hectáreas de la superficie de la

			parcela
ha_cultivo	Decimal	No nulo	Hectáreas cultivadas
cultivo	Cadena	No nulo	Tipo de cultivo
variedad	Cadena	No nulo	Tipo de variedad del cultivo
matas	Entero	No nulo	Número de matas plantadas y productivas
riego	Entero	Clave foránea	Tipo de riego
aire	Entero	Clave foránea	Tipo de aire
asesoram	Entero	Clave foránea	Tipo de asesoramiento
captacion_a gua	Cadena	Check(SI,NO)	Si existe o no un punto de agua procedente de pozos y masas de agua utilizadas para consumo humano
distancia	Decimal	Nulo	Distancia en metros si el punto de captación de agua está fuera de la parcela
denominaci on	Cadena	Nulo	Identificador, si existe, del punto de captación de agua
espec_total	Cadena	Check(SI,NO)	Parcela en zona específica ¹⁰ totalmente
espec_parci al	Cadena	Check(SI,NO)	Parcela en zona específica parcialmente
coordenada s	Cadena	Nulo	Coordenadas UTM del polígono de la parcela
recorte	Cadena	Nulo	Punto mínimo y máximo del recuadro que envuelve la ortofoto de la parcela

Tabla 15: Resumen de la entidad Parcela

SISTEMA_RIEGO (Tabla 16): sistema de riego que pueden ser aplicados en una parcela o en determinados tratamientos.

¹⁰ Se considerarán zonas específicas, según el artículo 34 del RD 1311/2012, las zonas de extracción de agua para consumo humano, las zonas de protección de hábitats naturales, flora y fauna y zonas de protección de especies acuáticas significativas declaradas protegidas en virtud del Reglamento de la Planificación Hidrológica

SISTEM_RIEGO			
Campo	Tipo de dato	Restricción	Significado
ident	Entero	Clave primaria	Identificador único
abrev	Cadena	No nulo	Abreviatura de nombre completo
abrev	Cadena	No nulo	Nombre completo

Tabla 16: Resumen de la entidad Sistema Riego

TIPO_ASESORAMIENTO (Tabla 17): forma de asesorar una cuaderno completo o aplicable solamente a una aplicación fitosanitaria.

TIPO_ASESORAMIENTO			
Campo	Tipo de dato	Restricción	Significado
ident	Entero	Clave primaria	Identificador único
abrev	Cadena	No nulo	Abreviatura de nombre completo
abrev	Cadena	No nulo	Nombre completo

Tabla 17: Resumen de la entidad Tipo Asesoramiento

TIPO_AIRE (Tabla 18): formas de gestionar el aire y ventilación de los cultivos, especialmente significativo en cultivos de invernadero.

TIPO_AIRE			
Campo	Tipo de dato	Restricción	Significado
ident	Entero	Clave primaria	Identificador único
abrev	Cadena	No nulo	Abreviatura de nombre completo
abrev	Cadena	No nulo	Nombre completo

Tabla 18: Resumen de la entidad Tipo Aire

ANALISIS (Tabla 19): registro de un análisis de laboratorio solamente realizado por el técnico agrónomo.

ANALISIS			
Campo	Tipo de dato	Restricción	Significado
ident	Entero	Clave primaria	Identificador único
parcela	Entero	Clave primaria y foránea	Parcela analizada
cuaderno	Entero	Clave foránea	Cuaderno al que se relaciona el análisis
fecha	Fecha	No nulo	Fecha de realización del análisis
material	Cadena	Clave foránea	Material analizado
laboratorio	Cadena	No nulo	Información del laboratorio donde se realiza el análisis
resultado	Cadena	No nulo	Resultado del análisis

Tabla 19: Resumen de la entidad Análisis

TIPO_MATERIAL (Tabla 20): materiales relacionados con el cultivo que puedan ser objeto de análisis.

TIPO_MATERIAL			
Campo	Tipo de dato	Restricción	Significado
ident	Entero	Clave primaria	Identificador único
abrev	Cadena	No nulo	Abreviatura de nombre completo
abrev	Cadena	No nulo	Nombre completo

Tabla 17: Resumen de la entidad Tipo Material

COSECHA (Tabla 21): registro de una entrega y venta del producto recolectado.

COSECHA			
Campo	Tipo de dato	Restricción	Significado
ident	Entero	Clave primaria	Identificador único
parcela	Entero	Clave primaria y foránea	Parcela a la que pertenece el fruto
cuaderno	Entero	Clave foránea	Cuaderno al que pertenece
fecha	Fecha	No nulo	Fecha de recogida
albaran	Entero	Nulo	Número de albarán
lote	Entero	Nulo	Número de lote
procedencia	Cadena	Check(vuelo, suelo)	Procedencia del producto recolectado
kgs	Decimal	No nulo	Kilos de fruto recogidos
rdto	Decimal	No nulo	Rendimiento del producto
cliente	Cadena	Clave foránea	Cliente que comercializa con el producto

Tabla 21: Resumen de la entidad Cosecha

CLIENTE (Tabla 22): persona o entidad que compra el producto que el agricultor ha recolectado.

CLIENTE			
Campo	Tipo de dato	Restricción	Significado
nif	Cadena	Clave primaria	Identificador único
descripcion	Cadena	No nulo	Nombre o razón social del cliente
direccion	Cadena	No nulo	Dirección de localización
rgseaa	Cadena	No nulo	Número en el RGSEAA

Tabla 22: Resumen de la entidad Cliente

OPERACION (Tabla 23): acción de hacer una labor en una parcela enmarcada en un cuaderno.

OPERACION			
Campo	Tipo de dato	Restricción	Significado
ident	Entero	Clave primaria	Identificador único
parcela	Entero	Clave primaria y foránea	Parcela dónde se realiza la operación
labor	Entero	Clase foránea	Tipo de labor
inicio	Fecha	No nulo	Fecha inicial
fin	Fecha	No nulo	Fecha final
observacion	Cadena	Nulo	Comentario adicional
cuaderno	Entero	Clave foránea	Cuaderno al que se relaciona la operación

Tabla 23: Resumen de la entidad Operacion

TIPO_LABOR (Tabla 24): tipo de labor que se aplica en una operación concreta.

TIPO_LABOR			
Campo	Tipo de dato	Restricción	Significado
ident	Entero	Clave primaria	Identificador único
abrev	Cadena	No nulo	Abreviatura de nombre completo
nombre	Cadena	No nulo	Nombre completo

Tabla 24: Resumen de la entidad Tipo Labor

TRATAMIENTO (Tabla 25): solución que se realiza ante un problema presente en una parcela.

TRATAMIENTO			
Campo	Tipo de dato	Restricción	Significado
ident	Entero	Clave primaria	Identificador único
parcela	Entero	Clave primaria y foránea	Parcela donde se aplicará el tratamiento
tipo	Entero	Clave foránea	Tipo de tratamiento
fecha_inicio	Fecha	No nulo	Fecha de inicio
fecha_validez	Fecha	Nulo	Fecha de validez por parte del técnico
fecha_mod	Fecha	Nulo	Fecha de modificación si tiene
estado	Entero	No nulo	Estado del tratamiento
msg_tecnico	Cadena	Nulo	Mensaje después de modificar o validar el tratamiento
personal	Cadena	Clave foránea	Trabajador que aplicará el tratamiento
equipo	Cadena	Clave foránea	Equipo utilizado
cuaderno	Entero	Clave foránea	Cuaderno asignado

Tabla 25: Resumen de la entidad Tratamiento

TIPO_TRATAMIENTO (Tabla 26): tipo de tratamiento aplicable.

TIPO_TRATAMIENTO			
Campo	Tipo de dato	Restricción	Significado
ident	Entero	Clave primaria	Identificador único
abrev	Cadena	No nulo	Abreviatura de nombre completo
nombre	Cadena	No nulo	Nombre completo

Tabla 26: Resumen de la entidad Tipo Tratamiento

FITOSANITARIO (Tabla 27): características de un tratamiento que aplica una actuación dónde interviene un producto fitosanitario, bien un plaguicida o insecticida.

FITOSANITARIO			
Campo	Tipo de dato	Restricción	Significado
ident	Entero	Clave primaria	Identificador único
parcela	Entero	Clave primaria y foránea	Parcela donde se aplicará el tratamiento
fin	Fecha	No nulo	Fecha de finalización de la actuación
producto	Cadena	No nulo	Nombre del producto que se aplica
registro	Entero	No nulo	Número del registro del producto
problema	Cadena	No nulo	Problema o motivo por el cual se realiza el tratamiento
dosis	Cadena	No nulo	Dosis aplicada del producto indicando la unidad de medida
eficacia	Cadena	No nulo	Eficacia que observa el agricultor
observacion	Cadena	Nulo	Mensaje que aporta el agricultor al tratamiento
ha	Decimal	No nulo	Hectáreas aplicadas por el producto

Tabla 27: Resumen de la entidad Fitosanitario

FITOSANITARIOASES (Tabla 28): la intención de esta entidad es la misma que la anterior, con la diferencia que esta hace referencia a parcelas con cultivos y superficies sujeto de asesoramiento.

FITOSANITARIOASES			
Campo	Tipo de dato	Restricción	Significado

ident	Entero	Clave primaria	Identificador único
parcela	Entero	Clave primaria y foránea	Parcela donde se aplicará el tratamiento
fin	Fecha	No nulo	Fecha de finalización de la actuación
plaga	Cadena	No nulo	Plaga a tratar
justificacion	Cadena	No nulo	Motivo por el cual se aplica el producto
medida	Cadena	Nulo	Tipo de medida utilizada en la alternativa no química de intervención, si existe
intensidad	Cadena	Nulo	Intensidad de medida utilizada en la alternativa no química de intervención, si existe
producto	Cadena	Nulo	Nombre del producto que se aplica en caso de existir alternativa química en la intervención
dosis	Cadena	Nulo	Dosis aplicada en caso de existir alternativa química en la intervención
registro	Entero	Nulo	Número de registro del producto en caso de existir alternativa química en la intervención
eficacia	Cadena	Nulo	Eficacia que observa el agricultor
observacion	Cadena	Nulo	Mensaje que aporta el agricultor al tratamiento
ha	Decimal	No nulo	Hectáreas aplicadas por el producto

Tabla 28: Resumen de la entidad FitosanitarioAses

SEMILLA (Tabla 29): tipo de tratamiento aplicable en una actuación, dónde el agricultor trata la semilla antes de realizar la plantación.

SEMILLA			
Campo	Tipo de dato	Restricción	Significado
ident	Entero	Clave primaria	Identificador único
parcela	Entero	Clave primaria y foránea	Parcela donde se aplicará el tratamiento
ha	Decimal	No nulo	Superficie sembrada
cantidad	Decimal	No nulo	Cantidad, en kilos, de semilla tratada
producto	Cadena	No nulo	Nombre del producto que se aplica
registro	Entero	No nulo	Número del registro del producto

Tabla 29: Resumen de la entidad Semilla

POSTCOSECHA (Tabla 30): aplicación fitosanitaria realizada después de la recolección del fruto.

POSTCOSECHA			
Campo	Tipo de dato	Restricción	Significado
ident	Entero	Clave primaria	Identificador único
parcela	Entero	Clave primaria y foránea	Parcela donde se aplicará el tratamiento
problema	Cadena	No nulo	Problemática
cantidad	Decimal	No nulo	Cantidad, en kilos, de semilla tratada
producto	Cadena	No nulo	Nombre del producto que se aplica
registro	Entero	No nulo	Número del registro del producto
tratado_en	Cadena	No nulo	Parte de la planta donde se efectúa el tratamiento

dosis	Cadena	No nulo	Cantidad de producto utilizado, señalando la unidad de medida
-------	--------	---------	---

Tabla 30: Resumen de la entidad Postcosecha

HERRAMIENTA (Tabla 31): esta entidad hace referencia a los tratamientos que se le hacen a las herramientas que utilice el agricultor para aplicar y tratar los productos fitosanitarios.

HERRAMIENTA			
Campo	Tipo de dato	Restricción	Significado
ident	Entero	Clave primaria	Identificador único
parcela	Entero	Clave primaria y foránea	Parcela donde se aplicará el tratamiento
tipo	Entero	Clave foránea	Tipo de herramienta
descripcion	Cadena	No nulo	Matrícula, si es un local indicará una descripción.
producto	Cadena	No nulo	Nombre del producto que se aplica
registro	Entero	No nulo	Número del registro del producto
problema	Cadena	No nulo	Problemática
dosis	Cadena	No nulo	Cantidad de producto utilizado, señalando la unidad de medida
volumen	Decimal	Nulo	Volumen tratado de la herramienta

Tabla 31: Resumen de la entidad Herramienta

TIPO_HERRAMIENTA (Tabla 32): tipo de herramienta aplicada en un tratamiento de limpieza de herramientas.

TIPO_HERRAMIENTA			
Campo	Tipo de dato	Restricción	Significado

ident	Entero	Clave primaria	Identificador único
abrev	Cadena	No nulo	Abreviatura de nombre completo
nombre	Cadena	No nulo	Nombre completo

Tabla 32: Resumen de la entidad Tipo Herramienta

FERTILIZACION (Tabla 33): acción de fertilizar sobre una parcela.

FERTILIZACION			
Campo	Tipo de dato	Restricción	Significado
ident	Entero	Clave primaria	Identificador único
parcela	Entero	Clave primaria y foránea	Parcela donde se aplicará el tratamiento
fin	Fecha	No nulo	Fecha de fin del tratamiento
tipo_abono	Entero	Clave foránea	Tipo de abono
tipo_fertilizante	Entero	Clave foránea	Tipo de fertilizante
tipo_riego	Entero	Clave foránea	Tipo de riego
producto	Cadena	No nulo	Nombre del producto que se aplica
dosis	Cadena	No nulo	Cantidad de producto utilizado, señalando la unidad de medida
composicion	Cadena	No nulo	Riqueza del fertilizante
ha	Decimal	No nulo	Superficie tratada
observacion	Cadena	Nulo	Observación del aplicador
prod_estim	Decimal	Nulo	Producción estimada, en kg/ha
abono_etiqueta	Cadena	Nulo	Abono aplicado cuando no está el tipo de fertilizante preestablecido

Tabla 33: Resumen de la entidad Fertilización

TIPO_ABONO (Tabla 34): tipo de abono aplicable a una fertilización.

TIPO_ABONO			
Campo	Tipo de dato	Restricción	Significado
ident	Entero	Clave primaria	Identificador único
abrev	Cadena	No nulo	Abreviatura de nombre completo
nombre	Cadena	No nulo	Nombre completo

Tabla 34: Resumen de la entidad Tipo Abono

TIPO_FERTILIZANTE (Tabla 35): tipo de abono aplicable a una fertilización.

TIPO_FERTILIZANTE			
Campo	Tipo de dato	Restricción	Significado
ident	Entero	Clave primaria	Identificador único
abrev	Cadena	No nulo	Abreviatura de nombre completo
nombre	Cadena	No nulo	Nombre completo

Tabla 35: Resumen de la entidad Tipo Fertilizante

RIEGO (Tabla 36): entidad que recoge todos los registros de riego dentro de una parcela.

RIEGO			
Campo	Tipo de dato	Restricción	Significado
ident	Entero	Clave primaria	Identificador único
parcela	Entero	Clave primaria y foránea	Parcela donde se aplicará el tratamiento
fin	Fecha	No nulo	Fecha de fin del tratamiento
tipo_riego	Entero	Clave foránea	Tipo de riego

ha	Decimal	No nulo	Superficie tratada
volumen	Decimal	Nulo	Superficie tratada
volumen_acum	Decimal	Nulo	Superficie tratada

Tabla 36: Resumen de la entidad Riego

IMAGEN (Tabla 37): datos relativos a un hecho específico que el agricultor considera problemático y que necesita asesoramiento técnico.

IMAGEN			
Campo	Tipo de dato	Restricción	Significado
ident	Entero	Clave primaria	Identificador único
parcela	Entero	Clave primaria y foránea	Parcela donde hace referencia la imagen
fin	Fecha	No nulo	Fecha de fin del tratamiento
estado	Entero	No nulo	Estado de la imagen
problema	Cadena	No nulo	Problemática o motivo de la imagen
msg_tecnico	Cadena	Nulo	Respuesta del técnico
imagen1	Cadena	No nulo	Primera parte de la imagen en base 64
imagen2	Cadena	No nulo	Primera parte de la imagen en base 64
imagen3	Cadena	No nulo	Primera parte de la imagen en base 64

Tabla 37: Resumen de la entidad Imagen

3.2.4. Implementación

La tecnología seleccionada para almacenar los datos a tratar por el sistema (detallada en la Sección 2.7) es Oracle Cloud DataBase. Debido a ello, la primera alternativa es utilizar la interfaz gráfica proporcionada por Oracle Cloud. Entre otras cosas nos permite ejecutar consultas y scripts, buscar y gestionar los objetos de la base de datos, cargar datos, crear gráficos y optimizar consultas SQL mediante planes de ejecución gráficos. Aclarar que SQL es un lenguaje específico, diseñado para administrar y recuperar información de un SGBD.

La segunda alternativa para trabajar con la base de datos es utilizar Oracle SQL Developer¹¹. Se trata de un entorno de desarrollo integrado para trabajar con Oracle DataBase en implementaciones tanto tradicionales como en la nube. Oracle permite trabajar con esta herramienta de forma gratuita y local. Para enlazar SQL Developer y Oracle Cloud necesitamos la llamada 'Cartera Cloud'. Los **archivos de billetera** junto con el identificador del usuario y la contraseña de la base de datos brindan acceso a la base de datos de forma autónoma y localmente en el dispositivo. Los archivos de billetera deben ser guardados y compartidos con precaución y seguridad solo a usuarios autorizados. Indistintamente, trabajaremos con ambos entornos.

A lo largo del nivel lógico (Sección 3.2.3) hemos comentado los tipos de datos de cada atributo de forma general, la Tabla 38 muestra los tipos de datos utilizados en SQL y sus características.

Tipo de dato	Tipo de dato Oracle	Características
Entero	NUMBER (n)	n es el número de dígitos máximo
Decimal	NUMBER (n, m)	m indica la cantidad de dígitos dedicados a la parte decimal
Cadena	VARCHAR2 (x)	Cadena de caracteres de longitud variables, x será la cantidad de byte máximos. En contra del tipo

¹¹ <https://www.oracle.com/es/database/sqldeveloper/>

		CHAR (x) , la memoria para almacenar es dinámica, adaptable al dato.
Fecha	DATE	Almacena un valor de fecha y hora. Oracle guarda internamente el siglo, año, mes, día, hora, minuto y segundo

Tabla 38: Tipos de datos en Oracle

Por otro lado, vamos a tener una base de datos con restricciones, ya sea por mantener una relación con otra entidad o bien requerida por su naturaleza, existen las siguientes tipos de reglas:

- **Regla de integridad de entidades.** Esta regla es aplicada a las claves primarias de las relaciones entre entidades. Por definición, una clave primaria es una clave irreducible que se utiliza para identificar el registro de forma única. Este tipo de integridad es violada cuando se intenta introducir una tupla con una clave primaria ya existente o nula.
- **Regla de integridad referencial.** Aplicada al acoplamiento de claves ajenas o foráneas. Todo valor de una clave foránea debe ser valor nulo o valor existente en la tabla referenciada como clave primaria. Esta cualidad se rompe cuando se intenta insertar un valor en la tabla referencia que no está en la tabla referenciada. O, también, en el intento de eliminar una tupla cuya clave primaria es referenciada.
- **Regla de los nulos.** Cuando en un registro un atributo es ausente, se dice que es nulo. Un nulo no representa el valor cero ni la cadena vacía ya que éstos son valores que tienen significado en ciertos contextos.
- **Regla de negocio.** Debido a la estrategia de funcionamiento del sistema global, será necesario imponer ciertas restricciones específicas sobre algunos campos. Como puede ser la restricción CHECK (expresión) . Esta permite incluir una condición, solo si esa condición es verdadera se inserta el registro.

Opcionalmente, una restricción puede ser declarada como **restricción de columna o restricción de tabla**. De columna quiere decir que afecta al atributo

para el cual ha sido declarada y, de tabla, significa que se definen para más de un columnas [47].

La decisión tomada para realizar las creaciones de las tablas es utilizar restricciones de tabla. Aunque a las restricciones de columna también se le pueden poner nombre, nos resulta más legible y cómodo utilizar nombre con restricciones de tabla. En cuanto al nombre de las restricciones creamos una estructura común en todas ellas, con el objetivo de facilitar la depuración e identificación de errores procedentes de restricciones. Consta de las siglas del tipo de restricción que se trata, la entidad en cuestión y el atributo en la que se aplica (`tipo_entidad_atributo`). Las siglas que utilizo para `tipo` son `pk` (clave primaria o *primary key*), `fk` (clave foránea o *foreign key*), `nn` (no nulo o *not null*), `n` (nulo o *null*) y `ck` (condición o *check*).

Decidido los tipos de datos y las restricciones a aplicar, es momento de crear la propia base de datos, o dicho de otra forma, el esquema de la base de datos.

En primer lugar, mediante el lenguaje de definición de datos o **DDL** (*Language of Definition of Data*), se ejecutan las sentencias propias que permiten crear el esquema de la base de datos. El resultado de la ejecución de este tipo de lenguaje es la creación de un conjunto de tablas almacenadas en un archivo especial conocido como diccionario de datos. Este tipo de lenguaje tiene los comandos: `CREATE`, `ALTER` y `DROP`.

En segundo lugar, se utilizará el lenguaje de manejo de datos o **DML**, en inglés, *Data Management Language*, para brindar la posibilidad a los usuarios de consultar, acceder o manipular los datos almacenados en el esquema. Los principales comandos que utilizamos son: `SELECT`, `INSERT`, `UPDATE` y `DELETE`.

3.2.5. API

En este apartado trataremos el eje de nuestra arquitectura (descrita en Apartado 2.8), es decir, el elemento que hace de nexo de unión entre la aplicación móvil y web. Oracle Cloud tiene mucho potencial y prueba de ello es la posibilidad de crear nuestras propias API en la nube.

En primer lugar, hay que aclarar el concepto de API. API son las siglas en inglés de *Application Programming Interfaces*, que en español significa Interfaz de Programación de Aplicaciones. Existen muchos tipos y, en nuestro caso, estas serán las características que definen a nuestra API:

- **Exposición:** El uso de una API u otra depende de la funcionalidad de la misma y el objetivo del creador. En este caso tratamos con una API web ya que tenemos detrás una propiedad intelectual que el usuario no necesita conocer.
- **Interfaz:** como su nombre indica una API es una interfaz. No tiene aspecto visual y por eso su uso está orientado a los datos a manipular. De la parte visual se encargará otra herramienta.
- **Enfoque:** una llamada a un API puede ser con o sin estado. Sin estado significa que las sucesivas llamadas a la API son totalmente independientes unas de otras. Y es de este modo cómo la implementamos.

El tipo de API que se utiliza en Oracle Cloud es **REST**. En concreto REST son las siglas en inglés de *REpresentational Status Transfer* o transferencia de estado representacional, no es ni un protocolo ni un estándar. En otras palabras, define un conjunto de métodos como `GET`, `POST`, `PUT` y `DELETE` que los clientes pueden utilizar para acceder a los datos que ofrece el servidor. Dicho intercambio de datos se realiza mediante el protocolo HTTP¹² [40].

3.2.6. API en Oracle Cloud

Conceptualmente Oracle Cloud estructura la API en módulos, plantillas y manejadores. Los siguientes pasos muestran el procedimiento seguido para crear nuestra API en Oracle Cloud:

1. **Módulos:** es una unidad organizativa que agrupa plantillas de recursos. En la Imagen 38 se muestra la vista de Oracle Cloud que permite gestionar los diferentes módulos existentes.

¹² Protocolo web en la capa de aplicación de transferencia de datos de hipertexto mediante una arquitectura cliente/servidor

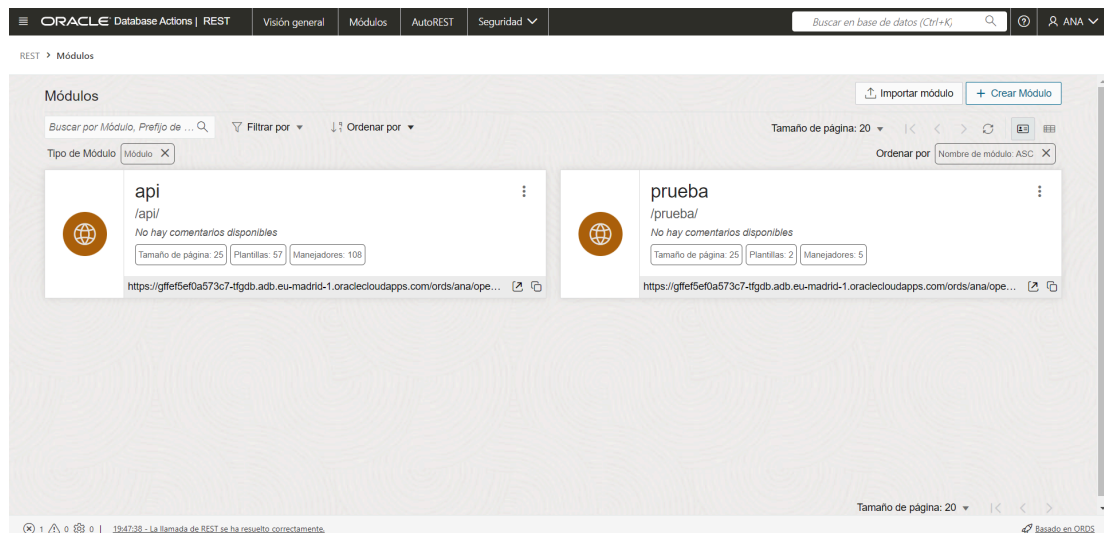


Imagen 38: Módulos de una API REST en Oracle Cloud

- 2. Plantillas:** de nuevo, una plantilla es un 'contenedor' que agrupa distintos controladores. Al final lo que se pretende es definir las distintas URL o solicitudes al servidor relacionadas con una misma tabla de la base de datos (Imagen 39). Uno de los parámetros que se podrá configurar es el nombre asignado dentro de la URI. Cabe la posibilidad de poder añadir parámetros a todo tipo de consulta, esta se puede hacer mediante la plantilla URI (/prueba/:numero) o cómo secuencia en la petición (/prueba?decim=40).

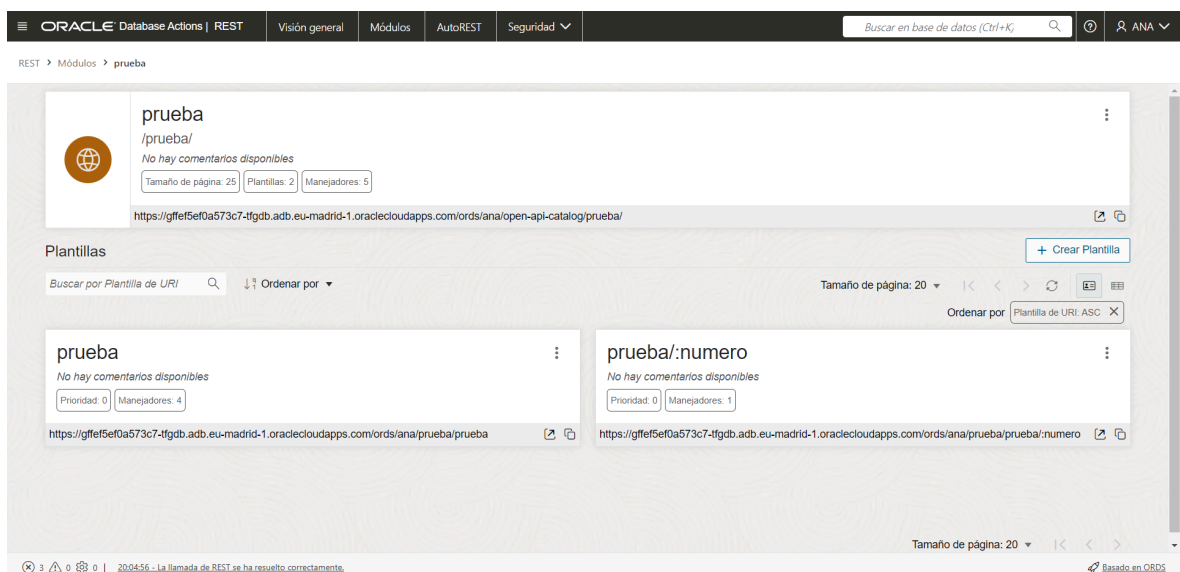


Imagen 39: Plantillas de un módulo en una API REST en Oracle Cloud

- 3. Controladores:** para una misma plantilla se pueden crear hasta cuatro tipos de controladores, cada uno de ellos asociado a un método distinto: GET, POST, PUT y/o DELETE (Imagen 40). Estos controladores tendrán la lógica de la API.

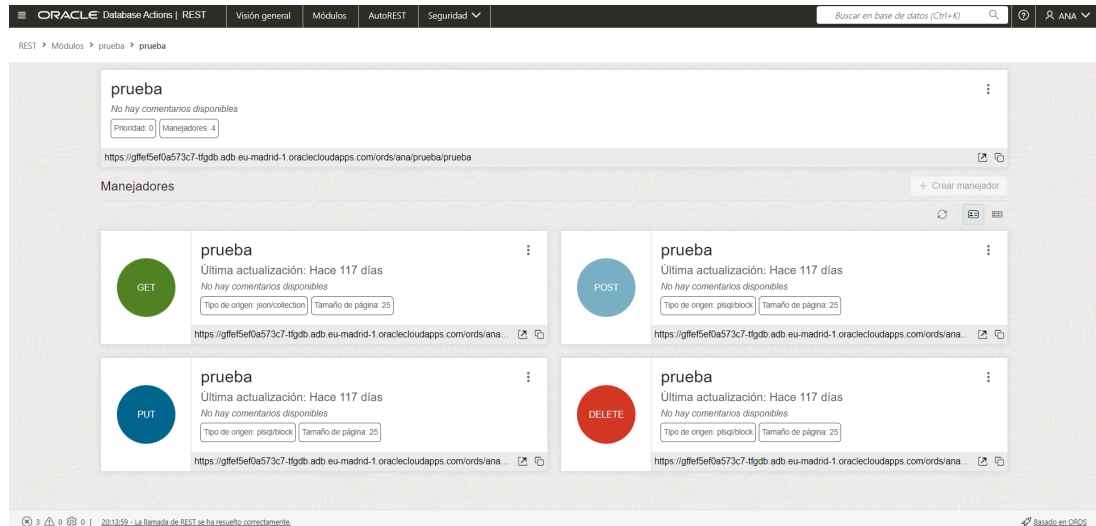


Imagen 40: Módulos de una API REST en Oracle Cloud

En forma de ejemplo mostramos en la Imagen 41 la implementación de un controlador GET. La lógica de cada controlador se codifica solamente mediante el lenguaje SQL.

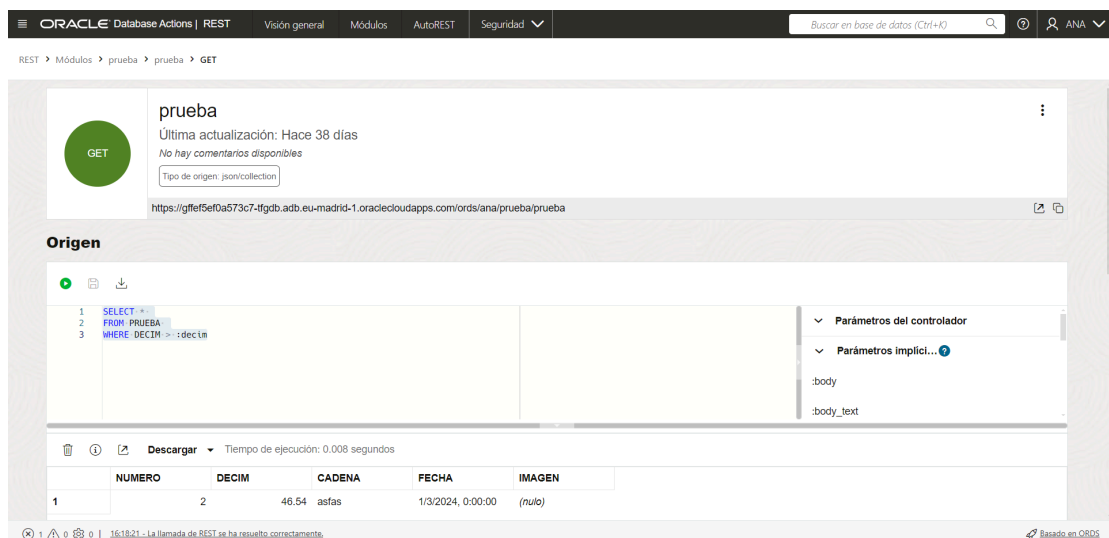


Imagen 41: Lógica de un controlador GET en una API REST en Oracle Cloud

Las imágenes aportadas muestran cómo se estructura una API en Oracle Cloud con un módulo, plantilla y controladores a modo de prueba. Sin embargo,

para nuestro caso planteamos el módulo llamado 'api' (enunciado en Imagen [38](#)). Internamente tendremos una plantilla por entidad de la base de datos. Cada una de ellas estará formada por cuatro controladores, uno por cada método, que definen las operaciones sobre dicha entidad.

3.2.7. Pruebas

A lo largo del desarrollo han surgido nuevas necesidades que en un principio no se tenían en cuenta, en términos de esquema de base de datos. Por ejemplo, en un principio se contempló que un objeto de la entidad 'Tratamiento', 'Análisis', 'Operación' o 'Cosecha' estaría asociado a una parcela en concreto, sin embargo, es más adecuado pensar que este objeto puede cambiar entre diferentes cuadernos que siempre tienen las mismas parcelas asociadas. Por este motivo se actualizó el esquema de la base de datos añadiendo una nueva clave foránea referenciando a la tabla 'Cuaderno'.

Con relación a la API, se realizaron los cuatro tipos de métodos por cada entidad de la base de datos, consultar, insertar, actualizar y borrar. Sin embargo, más adelante nos dimos cuenta que en varias entidades encontramos la necesidad de obtener los datos en otro formato o a partir de otro parámetros. Por ello, se añadieron nuevos controladores con nuevos métodos de consulta.

De forma adicional, se ha utilizado otra herramienta software para el desarrollo llamada **Postman**. Postman es una plataforma para construir y usar una API. Entre sus beneficios está la de acelerar el proceso de desarrollo y aumentar la calidad. [\[41\]](#)

A parte de estos cambios en nuestro esquema, se han realizado cambios menores. Como pueden ser cambios de nombre de atributos o corrección en nombre de restricciones que no seguía la nomenclatura establecida.

En conclusión, al terminar esta primera iteración tenemos finalizada la creación de la base de datos, cuyo esquema podemos visualizar en la Imagen [37](#), y la API. Disponibles para acceder a la información tanto desde la aplicación móvil como web.

3.3. Segunda iteración: Diseño de la interfaz de usuario

Seguimos con la segunda iteración, ‘Diseño de la interfaz de usuario’. A lo largo del presente capítulo explicaremos el proceso de diseño de la interfaz, los distintos métodos utilizados y los esquemas que utilizaremos para apoyarnos en el proceso de desarrollo.

3.3.1. Introducción

El proceso de diseño de la interfaz es invisible de cara al usuario hasta que no se convierte en una interfaz tangible. Este proceso comienza en el momento que se analiza la arquitectura a seguir del sistema y los requerimientos a cumplir.

Aclarar que cuando hablamos de diseño nos referimos a él como recurso estético de la aplicación. Aunque esto conlleva una gran atención, también, se hace énfasis en el diseño de interacción para definir el comportamiento y disposición de los elementos en la interfaz.

No obstante, antes de crear un diseño concreto, debemos identificar los elementos básicos que se pueden incluir en nuestra interfaz final. Las posibilidades son muchas y dependiendo de la categoría en la que se encuadre nuestra aplicación móvil podemos reconocer algunos elementos comunes o típicos. Obviamente los elementos que podemos encontrar en una aplicación móvil han evolucionado con el tiempo. Actualmente la **pantalla táctil** es el medio de interacción más utilizado en los sistemas operativos móviles, por lo que adoptan una tecnología basada en gestos. Entre sus desventajas vemos la precisión del gesto táctil, esta dificultad se vuelve crítica cuando la interfaz incluye botones muy pequeños o no tiene límites apreciables al usuario.

En este proceso identificamos diversas maneras en las que el usuario puede interactuar con una pantalla táctil. En nuestro caso vamos a escoger dos de los más comunes: **toque** y **deslizamiento**. Un toque es el gesto para elegir un elemento de la interfaz equivalente al clic de las interfaces tradicionales con

ventanas, menús y ratones. El desplazamiento será vertical o horizontal y permite revelar contenido o arrastrar otros elementos en la pantalla.

Existen más tipos como toque sostenido, toque doble, muti-touch o sin toques, pero recordamos que los usuarios de nuestra aplicación son agricultores y técnicos agrónomos cuya edad media es alta y nivel tecnológico generalmente bajo. Su entorno de trabajo es el campo y tenemos que facilitar el aprendizaje con una interacción sencilla y común.

Antes de llegar a una interfaz tangible el diseño se apoya de prototipos previos que recogen una visión inicial de la aplicación o simplemente una lluvia de ideas. Generalmente, un prototipo es la unidad básica de prueba de un producto. Por definición, no es el producto final sino que es una elaboración inacabada del mismo. Se clasifican en dos tipos: **prototipos de baja y alta fidelidad**. Es decir, si se presenta un prototipo sin decoración, centrado en la funcionalidad y con los elementos básicos de la interfaz, serán de baja fidelidad. De lo contrario, diseños que se acercan al producto final donde se hace énfasis en lo estético, se hablará de prototipos de alta fidelidad [42].

A continuación se detallan las fases de desarrollo del diseño, incrementando en fidelidad.

3.3.2. Sketch

En primer lugar, empezamos con un prototipo de baja fidelidad, el Sketch. Este nos permite crear una idea inicial de forma muy rápida, realizando una tormenta de ideas. Normalmente se realiza con papel y lápiz ya que se pretende plasmar las primeras ideas. Lo que se pretende es identificar los elementos más característicos como el logo, la zona de navegación, menús o botones.

En el siguiente listado se muestran los diseños realizados en ambas aplicaciones:

- **Aplicación móvil:** en los diagramas que no se indican nada serán comunes al agricultor y técnico:
 - Inicio de sesión (Imagen [42](#)): primera pantalla que ve el usuario si no hay credenciales guardadas en la aplicación previamente.



Imagen 42: Sketch del inicio de sesión en la aplicación móvil

- Listado de parcelas (Imagen 43): listado de las parcelas que un agricultor tiene asignadas, para el perfil del agricultor será su pantalla principal

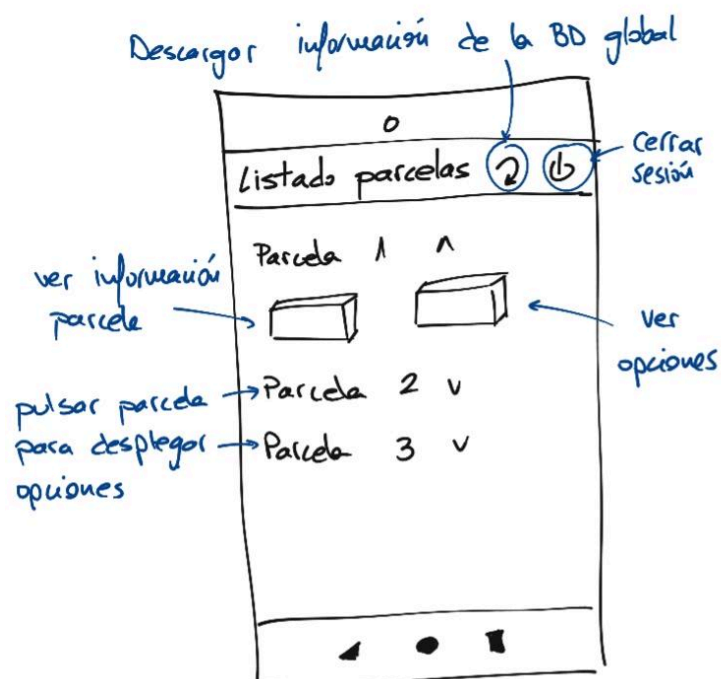


Imagen 43: Sketch del listado de parcelas en la aplicación móvil

- Información de una parcela (Imagen 44): información detallada de la parcela seleccionada en el listado, datos del SIGPAC, datos agrónomos, captación de agua o zonas específicas.



Imagen 44: Sketch de la información de una parcela en la aplicación móvil

- Menú de operaciones (Imagen 45): pantalla dedicada a mostrar las diferentes opciones que tiene el usuario para realizar sobre una parcela, son distintas según el rol del usuario.

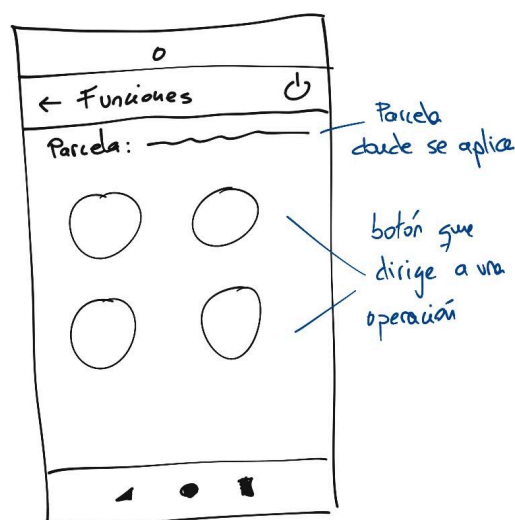


Imagen 45: Sketch del menú de operaciones en la aplicación móvil

- Listado de cualquier operación (Imagen 46): para cualquier entidad de la base de datos a la que se pueda aplicar las operaciones CRUD tendrá esta pantalla.

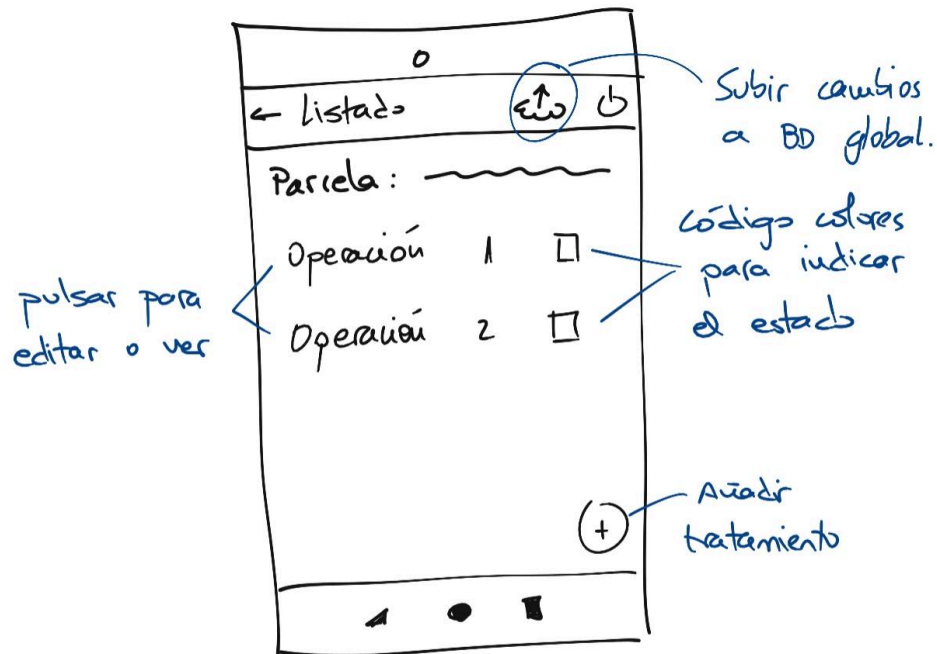


Imagen 46: Sketch del listado de cualquier operación en la aplicación móvil

- Ver operación (Imagen 47): ver las especificaciones de la operación.

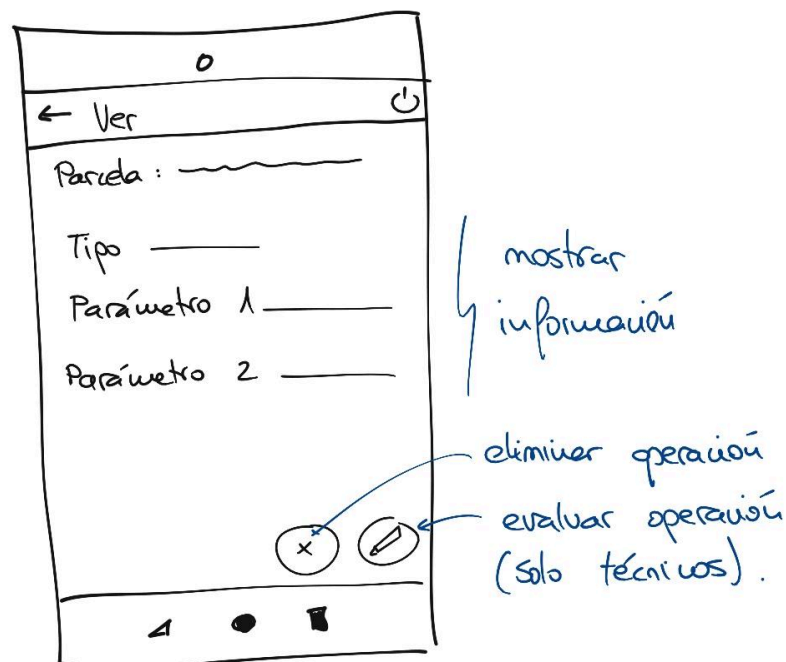


Imagen 47: Sketch de la visualización de una operación en la aplicación móvil

- Añadir o editar una operación (Imagen 48): los parámetros van a depender de la entidad.

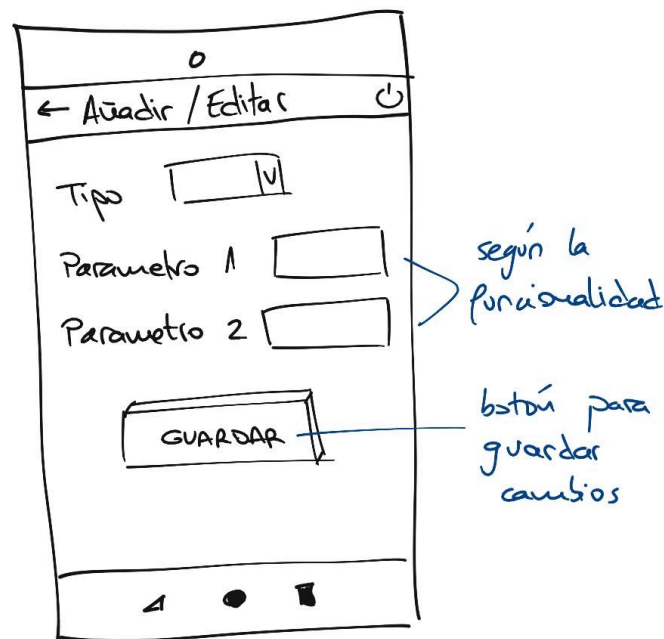


Imagen 48: Sketch para añadir o editar una operación en la aplicación móvil

- Ver histórico (Imagen 49): se podrá visualizar un resumen de los datos gráficamente.



Imagen 49: Sketch par ver histórico de los datos en la aplicación móvil

- Crear informes (Imagen [50](#)): creación de documentos en formato pdf

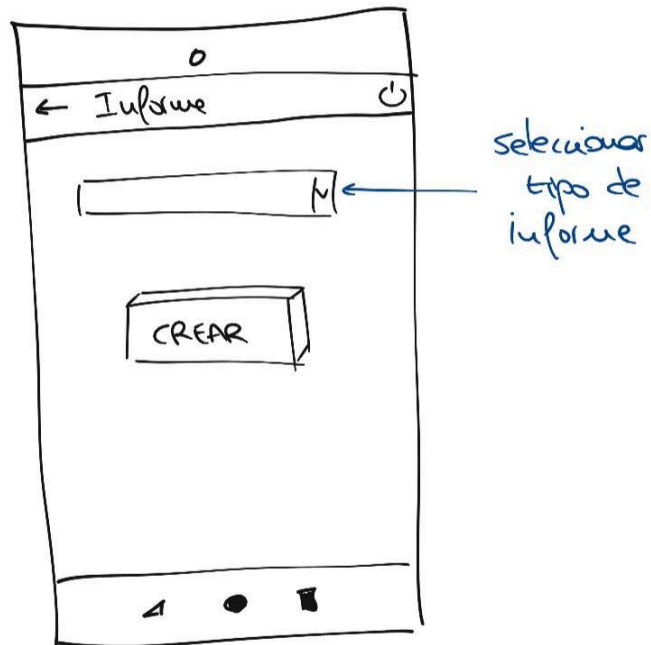


Imagen 50: Sketch para hacer informes en la aplicación móvil

- Listado de agricultores (Imagen [51](#)): pantalla que muestra los agricultores que un técnico tiene que asesorar.

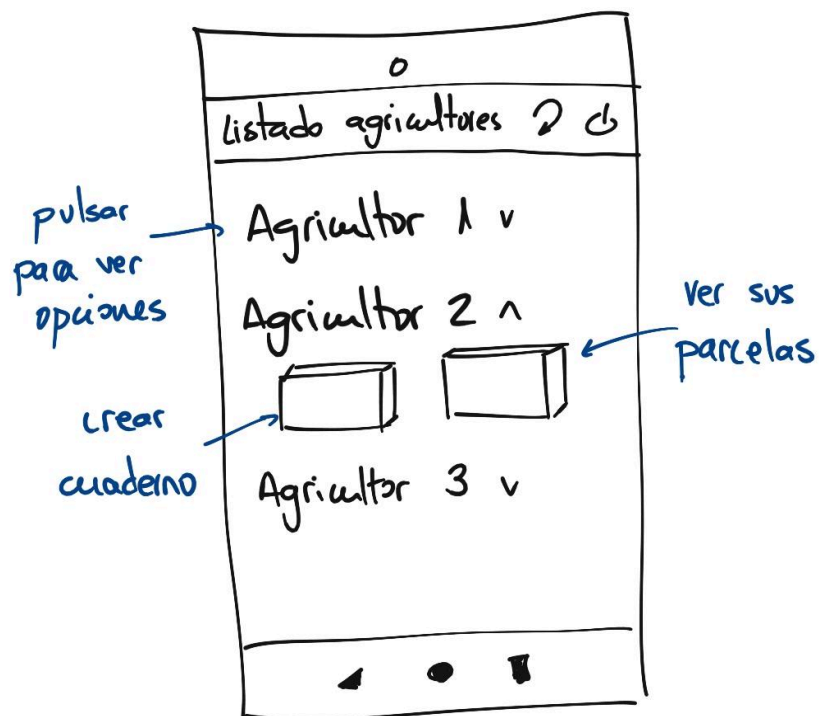


Imagen 51: Sketch para listar los agricultores en la aplicación móvil

- **Aplicación web:** solo el perfil del administrador podrá acceder:
 - Inicio de sesión (Imagen [52](#)):

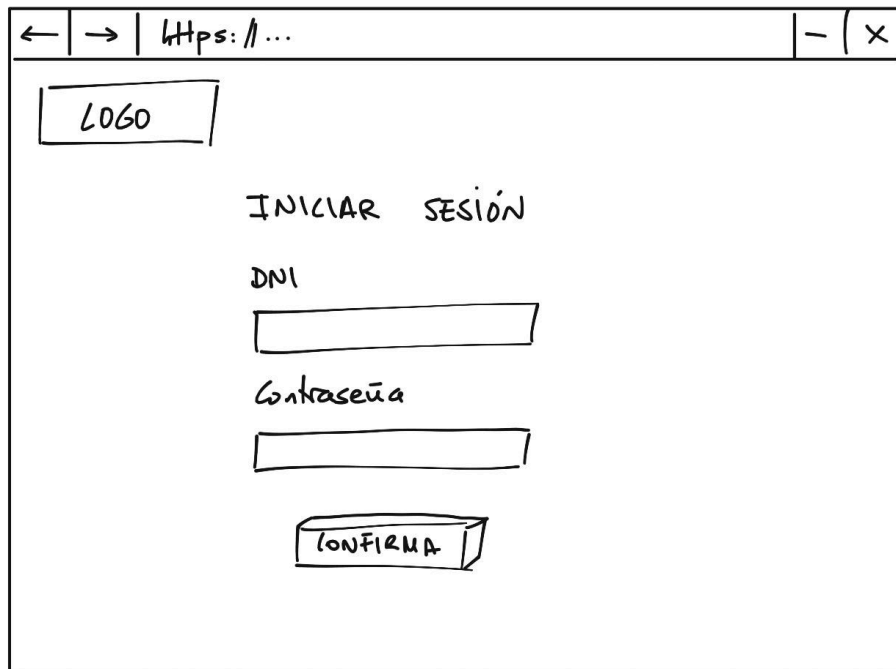


Imagen 52: Sketch para iniciar sesión en la aplicación web

- Menú de operaciones (Imagen [53](#)): tarjetas en formato cuadrícula que dirige a la funcionalidad.

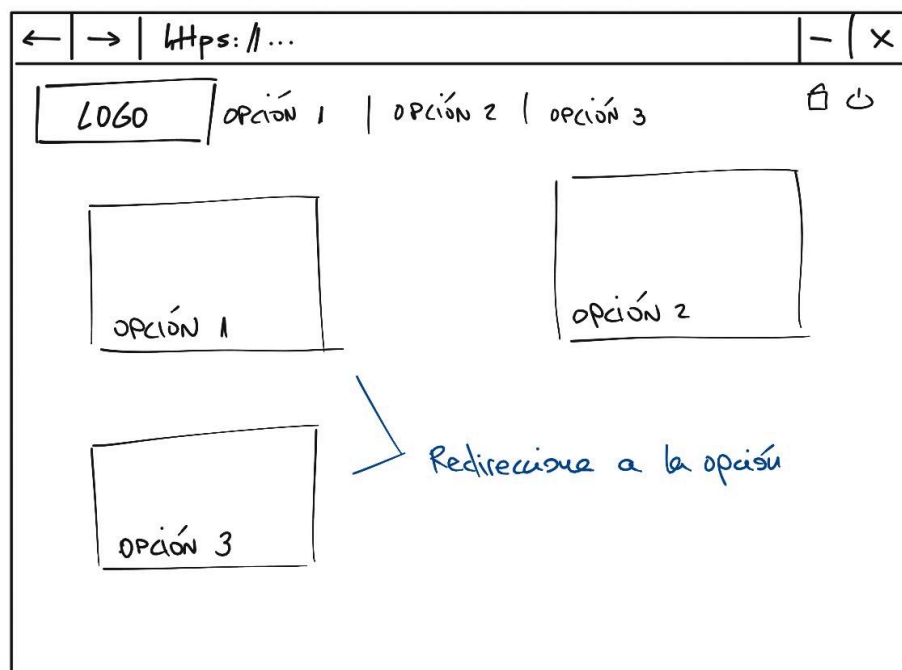


Imagen 53: Sketch del menú de operaciones en la aplicación web

- Mantenimiento de una entidad (Imagen 54): mantenimiento significa pantalla cuya utilidad es ofrecer las operaciones CRUD sobre una tabla de la base de datos.

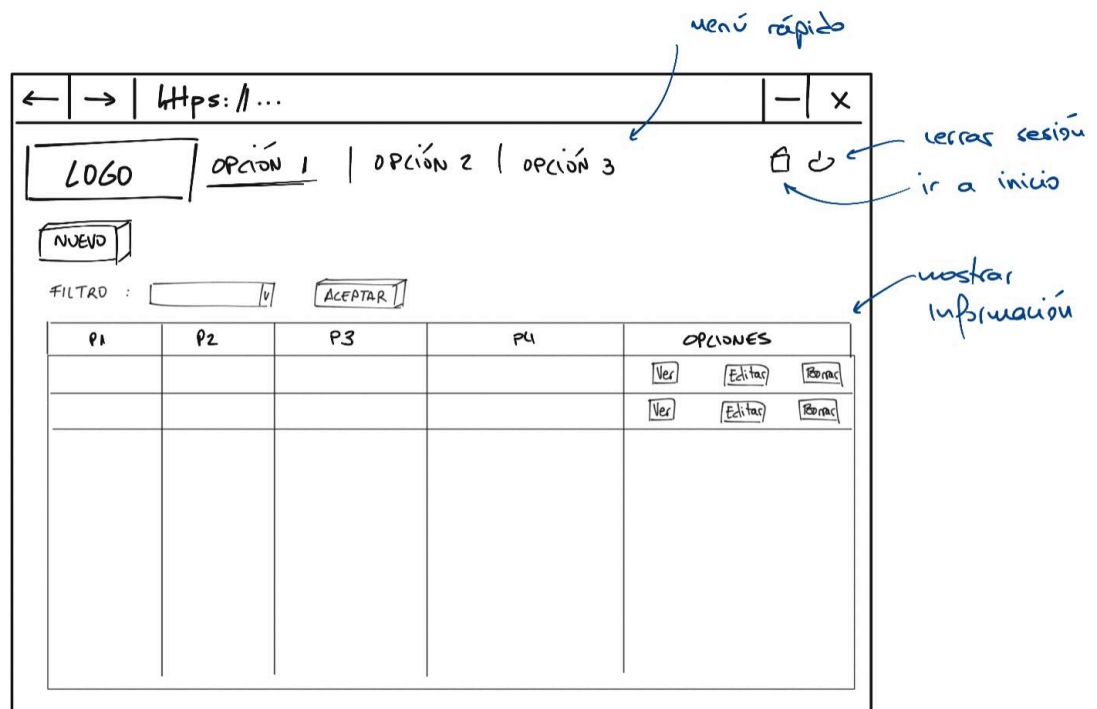


Imagen 54: Sketch del mantenimiento de una entidad en la aplicación web

- Historial de un cuaderno (Imagen 55): información resumida de un cuaderno de campo.

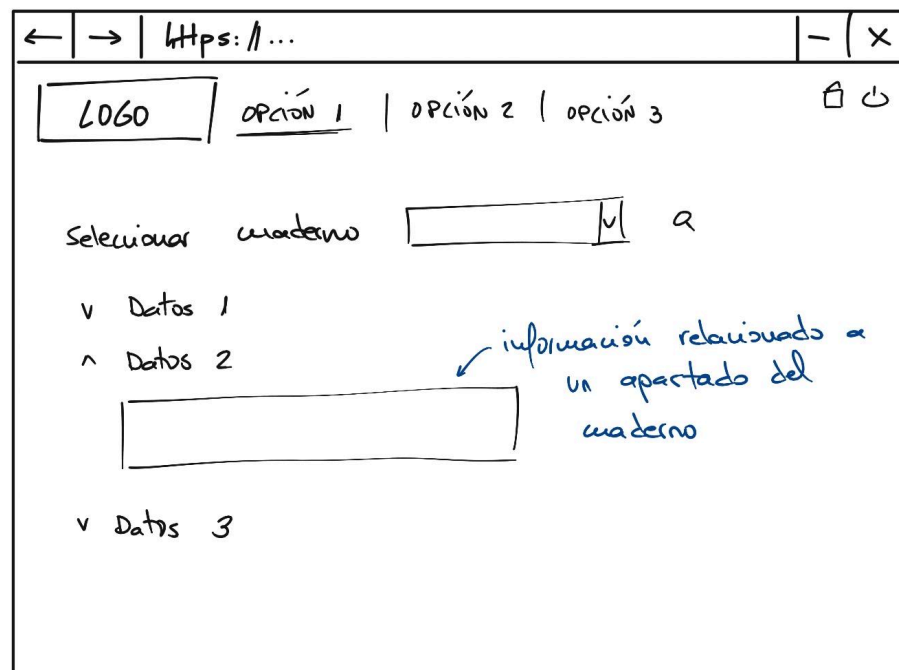


Imagen 55: Sketch del historial relativo a un cuaderno en la aplicación web

- Visualizar imágenes de problemas (Imagen [56](#)): información resumida de un cuaderno de campo.

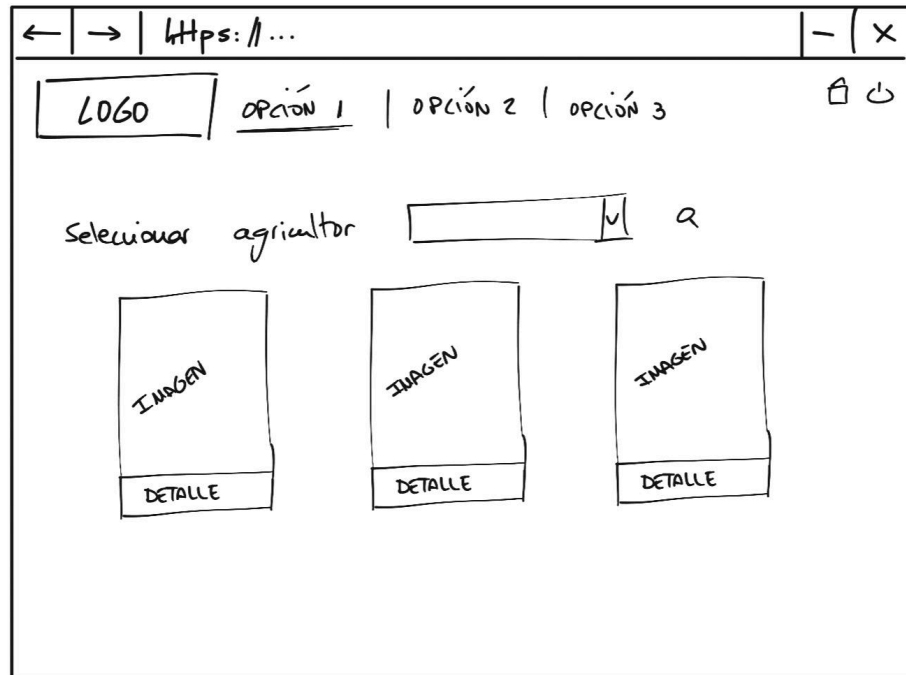


Imagen 56: Sketch de la visualización de las imágenes en la aplicación web

3.3.3. Wireframes

Los Wireframes son los llamados bocetos de alambre. Este tipo de prototipo es de una fidelidad media ya que se centra en mostrar la asignación de espacio, priorización de contenido, funcionalidades, comportamientos y jerarquía deseados. Carecen de estilo tipográfico.

- **Aplicación móvil:** sólo accesible para agricultores y técnicos.
 - Inicio de sesión (Imagen [57](#)): realizado a partir de la Imagen [42](#).

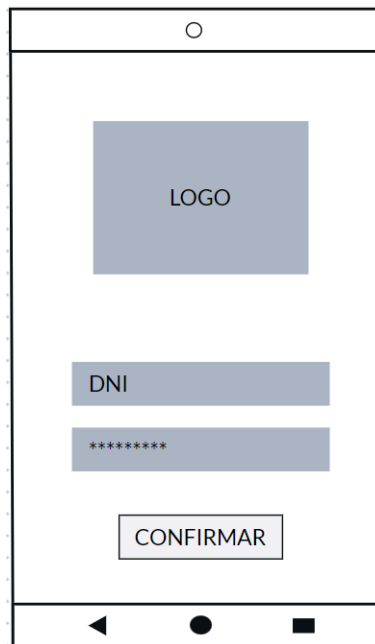


Imagen 57: Wireframe del inicio de sesión en la aplicación móvil

- Listado de parcelas (Imagen [58](#)): realizado a partir de la Imagen [43](#).

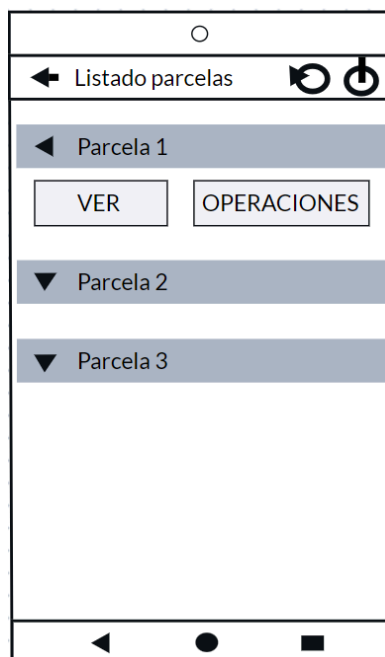


Imagen 58: Wireframe del listado de parcelas en la aplicación móvil

- Información de una parcela (Imagen [59](#)): realizado a partir de la Imagen [44](#).

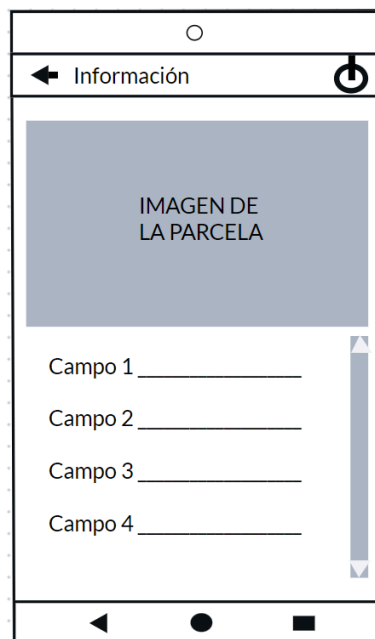


Imagen 59: Wireframe de la información de la parcela en la aplicación móvil

- Menú de operaciones (Imagen [60](#)): realizado a partir de la Imagen [45](#).

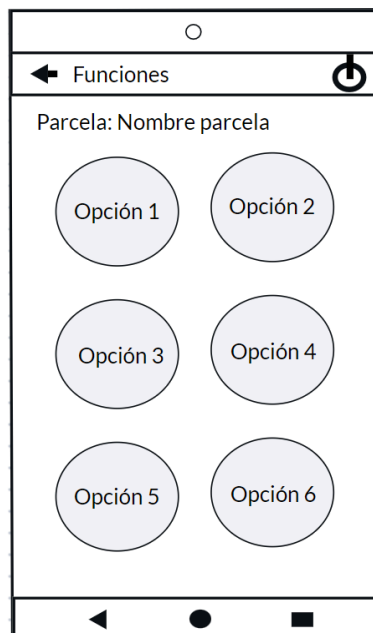


Imagen 60: Wireframe del menú de operaciones en la aplicación móvil

- Listado de cualquier operación (Imagen [61](#)): realizado a partir de la Imagen [46](#).

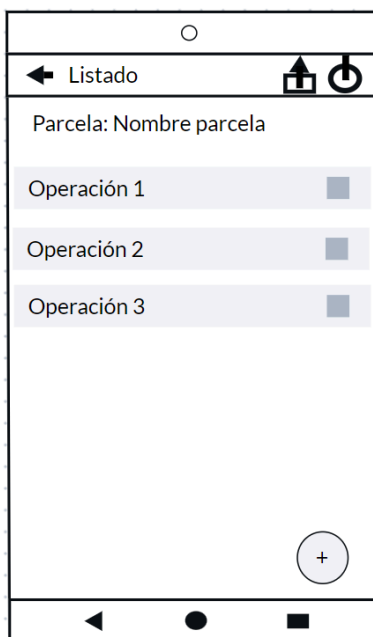


Imagen 61: Wireframe del listado de cualquier operación en la aplicación móvil

- Ver operación (Imagen [62](#)): realizado a partir de la Imagen [47](#).

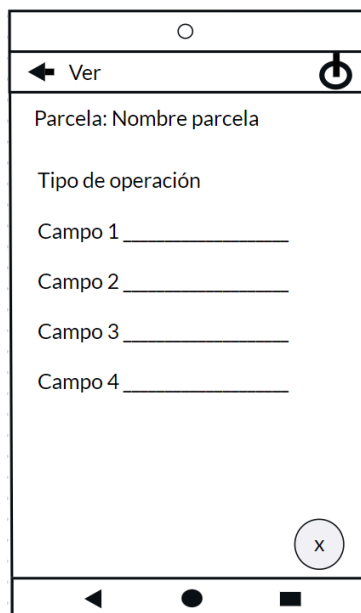


Imagen 62: Wireframe de visualización de una operación en la aplicación móvil

- Añadir o editar una operación (Imagen [63](#)): realizado a partir de la Imagen [48](#).

A mobile application wireframe for the 'Añadir / Editar' screen. At the top is a status bar with a circle icon. Below it is a header bar with a back arrow, the text 'Añadir / Editar', and a power icon. The main content area starts with the label 'Parcela: Nombre parcela'. Below this is a 'Tipo:' label followed by a dropdown menu. Then there are four input fields labeled 'Campo 1', 'Campo 2', 'Campo 3', and 'Campo 4'. At the bottom of the form is a 'CONFIRMAR' button. The screen is framed by a standard Android navigation bar at the bottom.

Imagen 63: Wireframe para añadir o editar una operación en la aplicación móvil

- Ver histórico (Imagen [64](#)): realizado a partir de la Imagen [49](#).

A mobile application wireframe for the 'Históricos' screen. At the top is a status bar with a circle icon. Below it is a header bar with a back arrow, the text 'Históricos', and a power icon. The main content area starts with a dropdown menu labeled 'Selecciona un tipo de histórico'. Below this is a large rectangular area labeled 'GRÁFICA'. At the bottom of the main content area is the text 'Descripción del gráfico'. The screen is framed by a standard Android navigation bar at the bottom.

Imagen 64: Wireframe de histórico en la aplicación móvil

- Crear informes (Imagen [65](#)): realizado a partir de la Imagen [50](#).

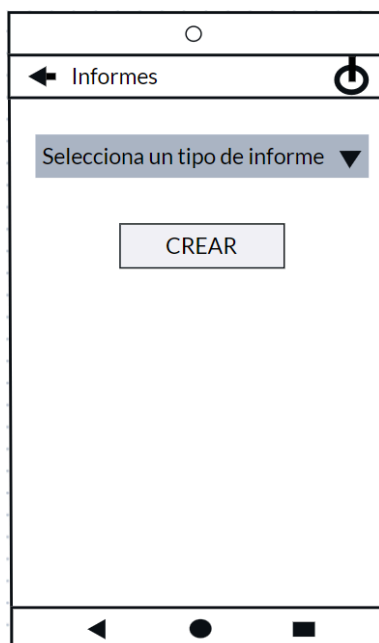


Imagen 65: Wireframe sobre la creación de informes en la aplicación móvil

- Listado de agricultores (Imagen [66](#)): realizado a partir de la Imagen [51](#).

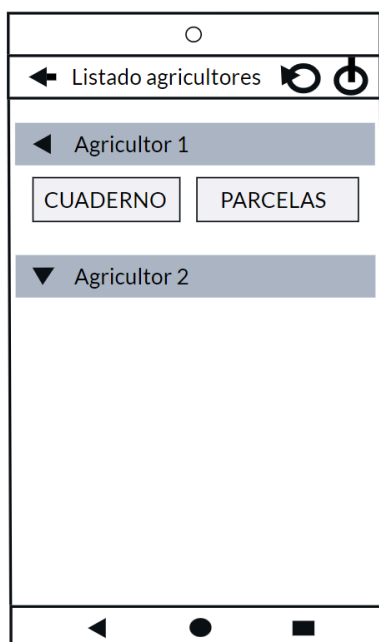


Imagen 66: Wireframe del listado de agricultores en la aplicación móvil

- **Aplicación web:** solo el perfil del administrador podrá acceder:
 - Inicio de sesión (Imagen [67](#)): realizado a partir de la Imagen [52](#).



Imagen 67: Wireframe del inicio sesión en la aplicación web

- Menú de operaciones (Imagen [68](#)): realizado a partir de la Imagen [53](#).

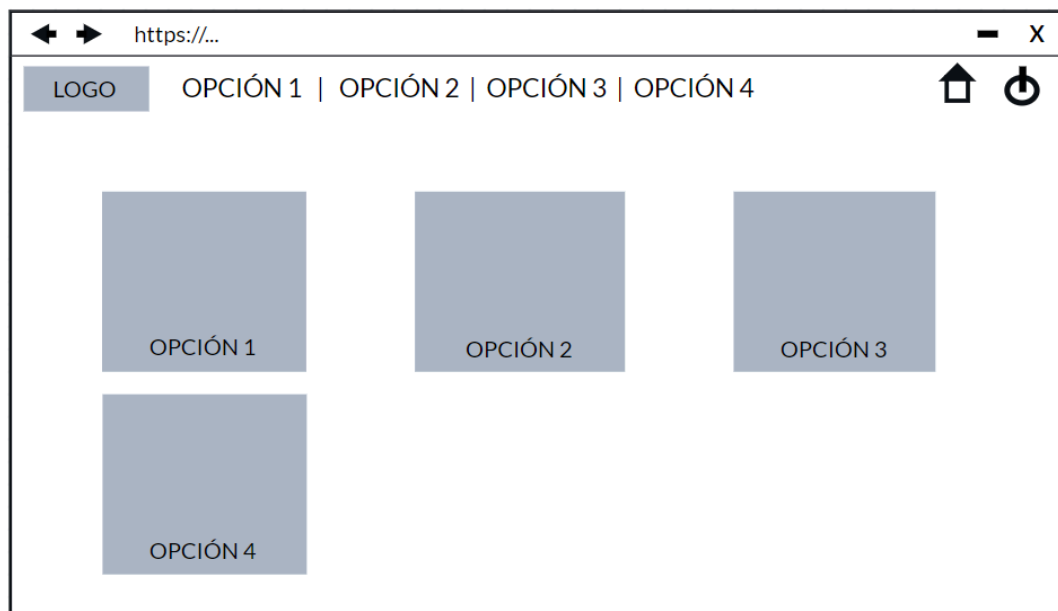


Imagen 68: Wireframe del menú de operaciones en la aplicación web

- Mantenimiento de una entidad (Imagen [69](#)): realizado a partir de la Imagen [54](#).

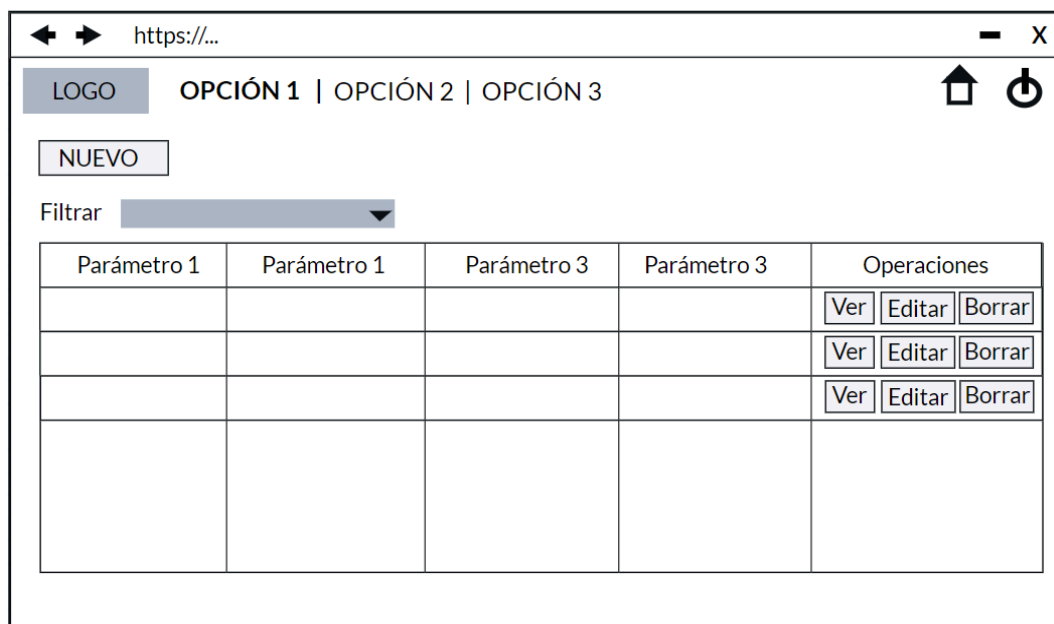


Imagen 69: Wireframe del mantenimiento de una entidad en la aplicación web

- Historial de un cuaderno (Imagen [70](#)): realizado a partir de la Imagen [55](#).



Imagen 70: Wireframe del historial relativo a un cuaderno en la aplicación web

- Visualizar imágenes de problemas (Imagen [71](#)): realizado a partir de la Imagen [56](#).

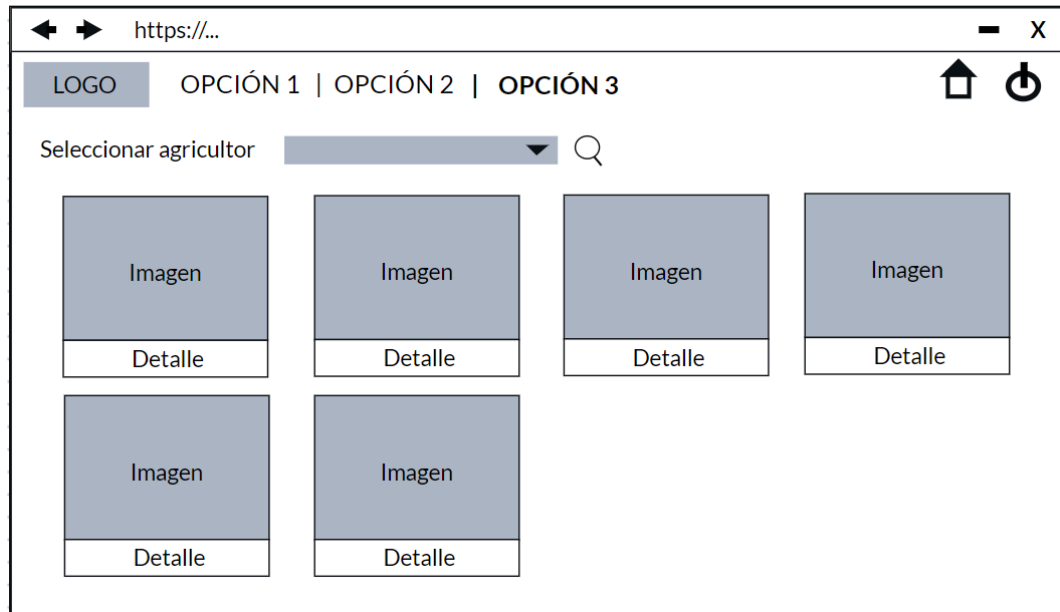


Imagen 71: Wireframe de la visualización de las imágenes en la aplicación web

3.3.4. Mockups

Por último nos centramos en los Mockup o maquetas, son los prototipos de más fidelidad. Se usa el wireframe como plantilla e introduce todo los elementos gráficos y visuales. Nos ayudamos de la aplicación Marvel¹³ para realizar los prototipos.

- **Aplicación móvil:**
 - Inicio de sesión (Imagen [72](#)): realizado a partir de la Imagen [57](#).

¹³ <https://marvelapp.com/>



Imagen 72: Mockups del inicio de sesión en la aplicación móvil.

- Listado de parcelas (Imagen [73](#)): realizado a partir de la Imagen [58](#).

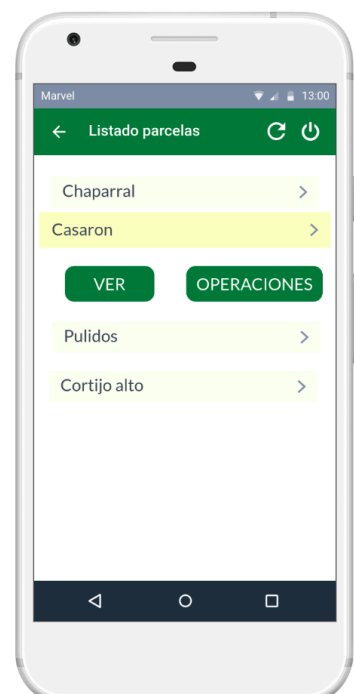


Imagen 73: Mockups del listado de parcelas en la aplicación móvil.

- Información de una parcela (Imagen [74](#)): realizado a partir de la Imagen [59](#).



Imagen 74: Mockups de la información de la parcela en la aplicación móvil.

- Menú de operaciones (Imagen [75](#)): a partir de la Imagen [60](#).
- Listado de cualquier operación (Imagen [76](#)): realizado a partir de la Imagen [61](#).



Imagen 75: Mockups del menú de operaciones en la aplicación móvil.

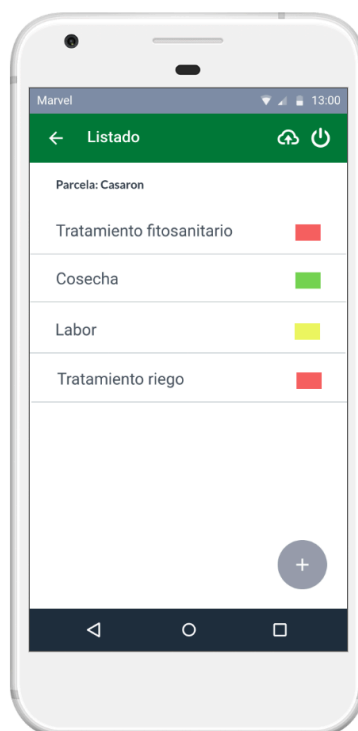


Imagen 76: Mockups del listado de cualquier operación en la aplicación móvil

- Ver operación (Imagen [77](#)): realizado a partir de la Imagen [62](#).



Imagen 77: Mockups de visualización de la operación en la aplicación móvil

- Añadir o editar una operación (Imagen [78](#)): realizado a partir de la Imagen [63](#).



Imagen 78: Mockups añadir o editar una operación en la aplicación móvil

La pantalla para añadir o editar una operación tendrá diferentes entradas según el tipo de operación. Lo más característico de esta interacción serán las entradas de fechas (Imagen [79](#) izquierda) y el botón que guarda los cambios en la base de datos (Imagen [79](#) derecha).

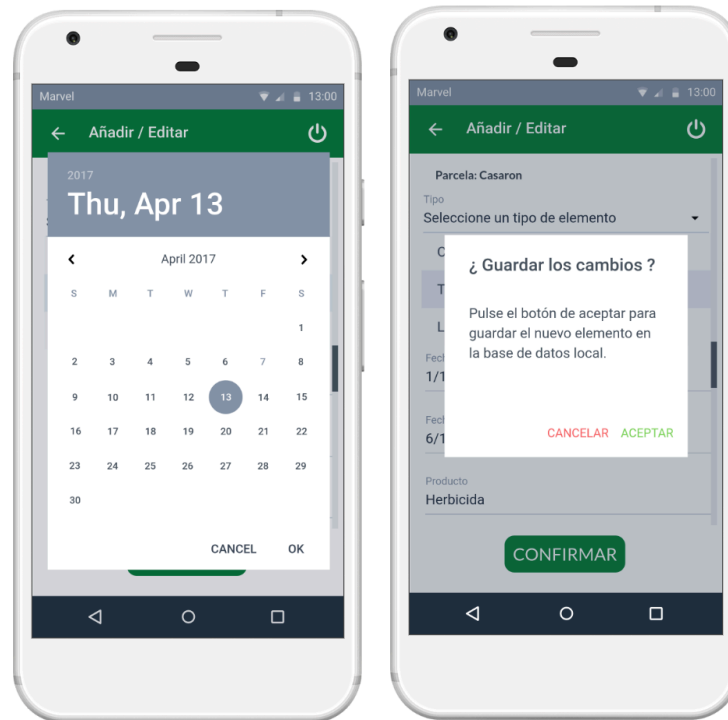


Imagen 79: Mockup de la interacción en la pantalla de añadir o editar una operación, entrada de fechas (izquierda) y guardar cambios(derecha) en la aplicación móvil

- Ver histórico (Imagen [80](#)): realizado a partir de la Imagen [64](#).
- Crear informes (Imagen [81](#)): realizado a partir de la Imagen [65](#). Cuando el informe es creado salta una notificación con el objetivo de redirigir al usuario directamente al documento almacenado en el dispositivo (Imagen [82](#)). Esta funcionalidad también es utilizada para la creación de un cuaderno, botón accesible desde el listado de agricultores (Imagen [83](#)).



Imagen 80: Mockups de histórico en la aplicación móvil.

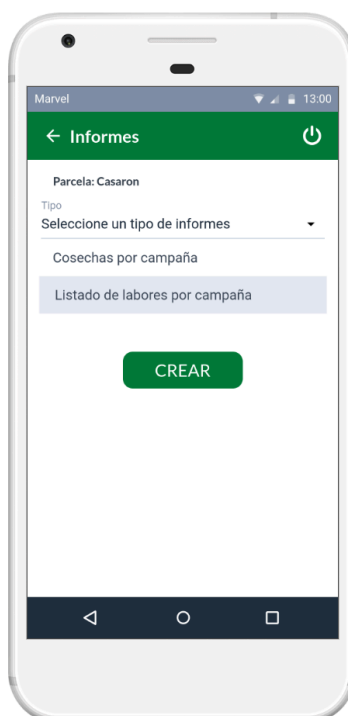


Imagen 81: Mockups sobre la creación de informes en la aplicación móvil.



Imagen 82: Mockups de la notificación al finalizar la creación de un informe en la aplicación móvil

- Listado de agricultores (Imagen [83](#)): a partir de la Imagen [66](#).

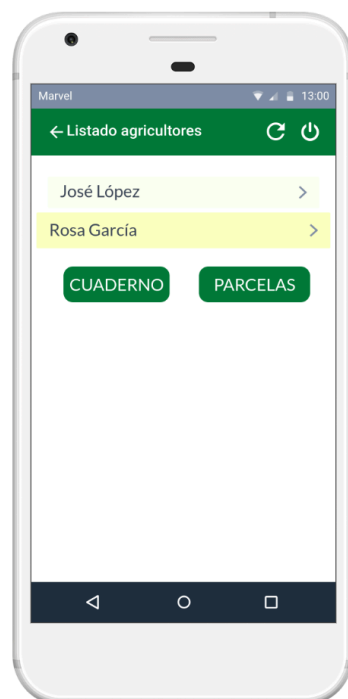


Imagen 83: Mockups del listado de agricultores en la aplicación móvil

- **Aplicación web:** solo el perfil del administrador podrá acceder:
 - Inicio de sesión (Imagen [84](#)): realizado a partir de la Imagen [67](#).

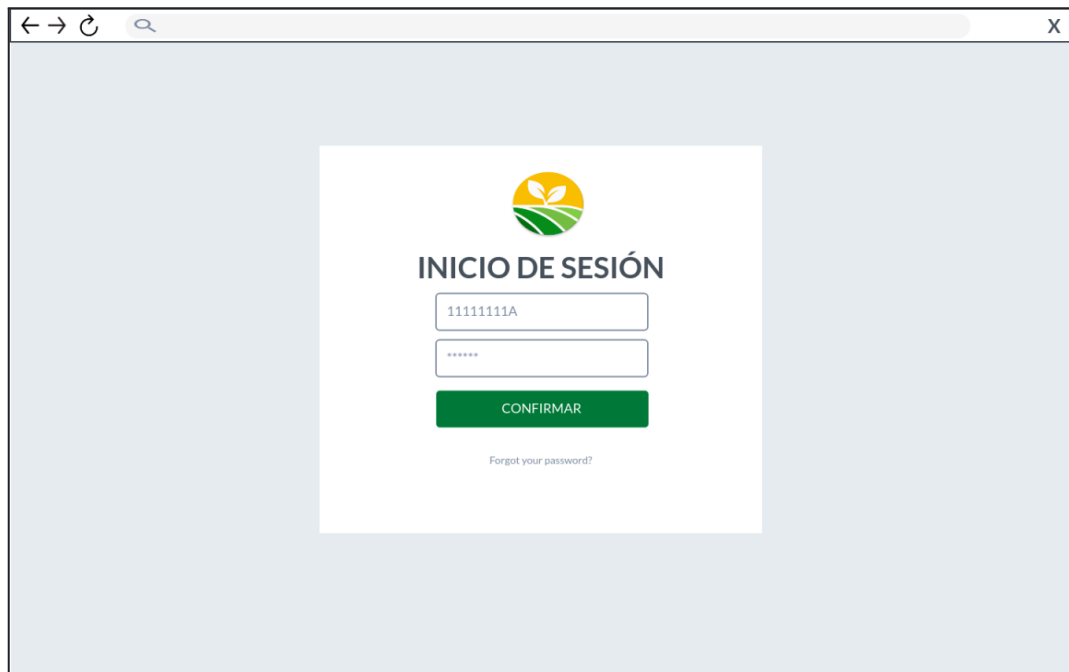


Imagen 84: Mockups del inicio sesión en la aplicación web

- Menú de operaciones (Imagen [85](#)): realizado a partir de la Imagen [68](#).



Imagen 85: Mockups del menú de operaciones en la aplicación web

- Mantenimiento de una entidad (Imagen [86](#)): realizado a partir de la Imagen [69](#).

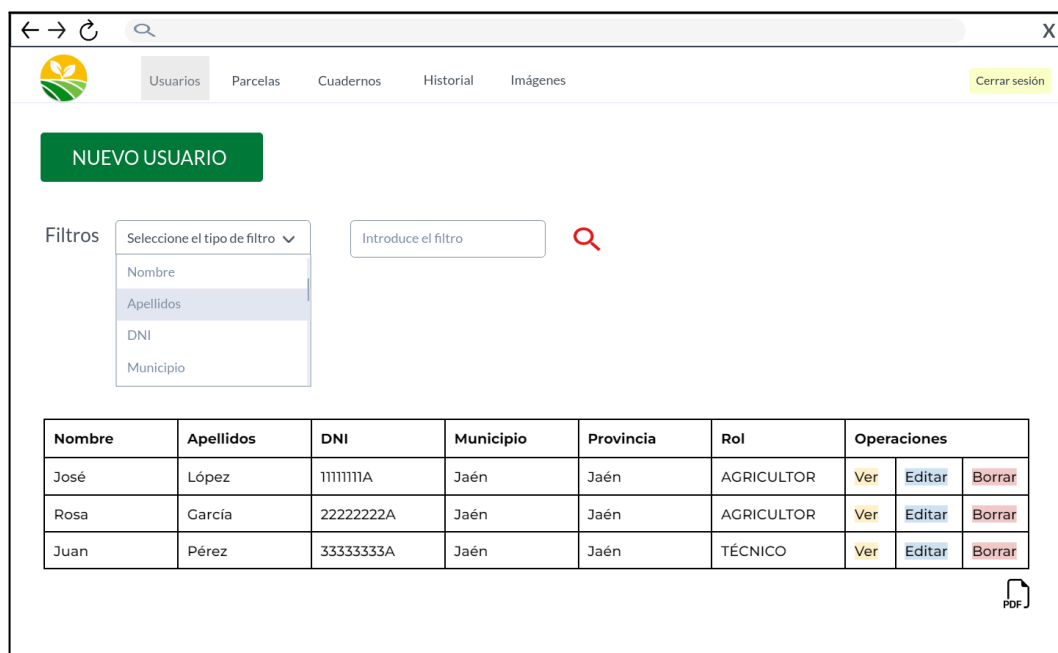


Imagen 86: Mockups del mantenimiento de una entidad en la aplicación web

- Historial de un cuaderno (Imagen [87](#)): realizado a partir de la Imagen [70](#).

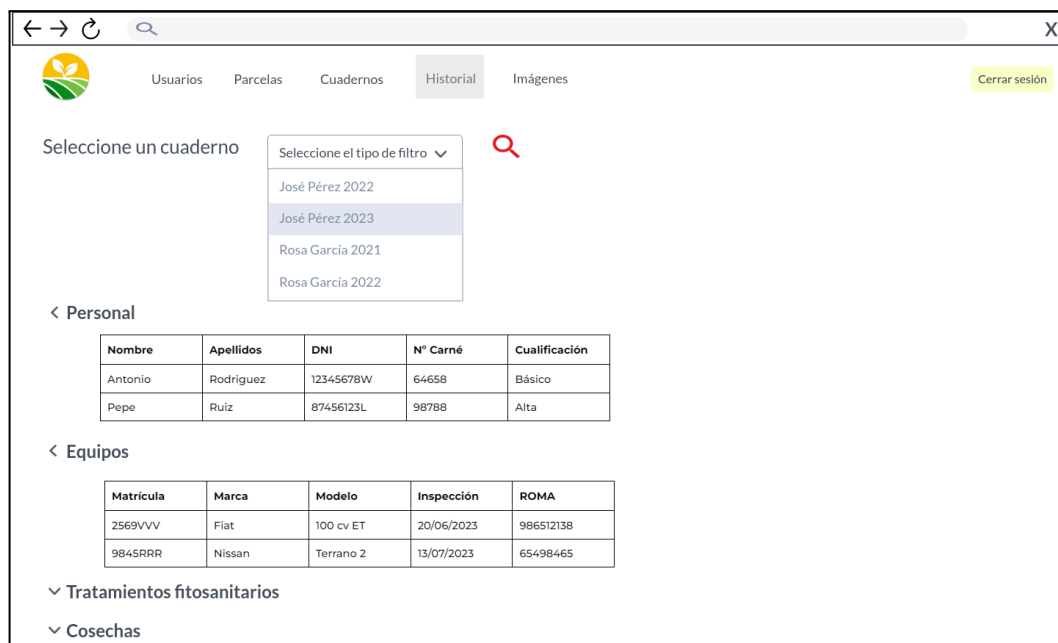


Imagen 87: Mockups del historial relativo a un cuaderno en la aplicación web

- Visualizar imágenes de problemas (Imagen [88](#)): realizado a partir de la Imagen [71](#).

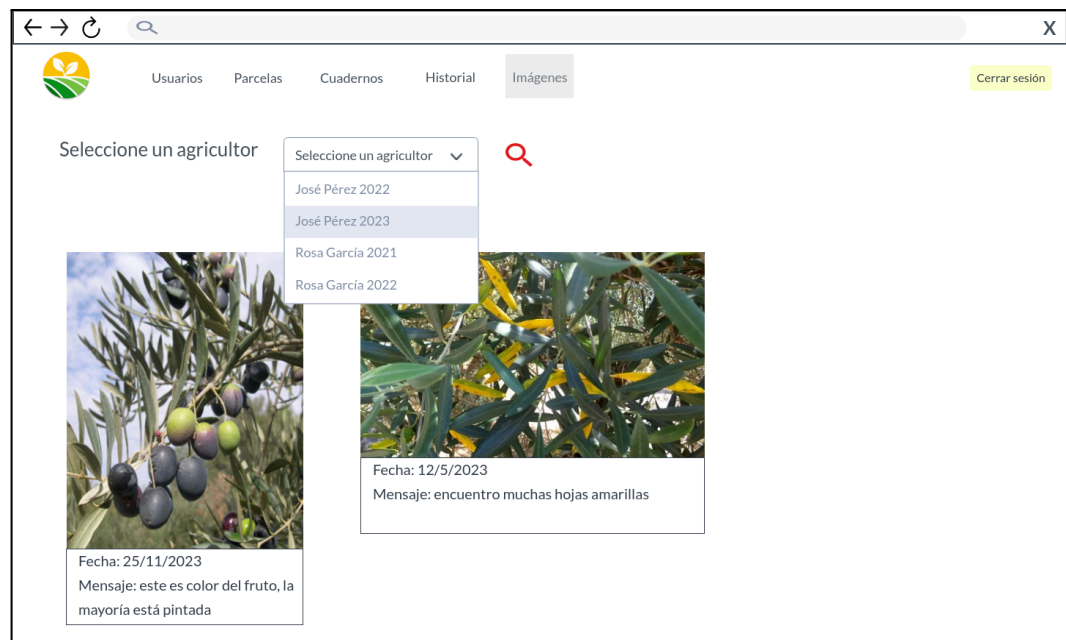


Imagen 88: Mockups de la visualización de las imágenes en la aplicación web

3.4. Tercera iteración: Aplicación móvil

En esta tercera iteración vamos a desarrollar la aplicación móvil. Como hemos explicado anteriormente este proyecto presenta una aplicación móvil para dos tipos de usuarios, ya que sus funcionalidades en algunos aspectos no son iguales, aunque el desarrollo se lleva a cabo conjuntamente.

Principalmente hablaremos de las características del entorno de desarrollo, Android Studio, ya que interpretamos esencial conocer la herramienta con la que trabajaremos. Después, entraremos a explicar cada funcionalidad implementada desde el lenguaje de programación, Java.

3.4.1. Android Studio

En el Apartado [2.5](#) explicamos el por qué de Android Studio y en esta sección desarrollaremos los aspectos básicos a tener en cuenta.

Independientemente del lenguaje de programación que utilicemos, Kotlin, Java o C++, Android Studio compila el código creando un fichero APK. Este tipo de archivo incluye todo el contenido de una aplicación Android y es el archivo que usan los dispositivos, con tecnología Android, para instalar una aplicación. Cada

aplicación de Android implementa el principio mínimo de privilegio. Esto significa que cada aplicación tiene acceso predeterminado solo a los componentes que necesita para llevar a cabo su trabajo y nada más. De esta manera se consigue un entorno muy seguro, en el que una aplicación no puede acceder a partes del sistema para las que no tiene permisos.

Android Studio divide una aplicación en **componentes**. Cada componente es un punto de entrada por el que el sistema o usuario ingresa a una aplicación. Los siguientes componentes son fundamentales en cualquier aplicación Android. Ya que a lo largo del desarrollo se nombran algunos, queremos introducir dichos conceptos de forma breve:

- **Actividades.** Una actividad (clase `Activity`) lo que permite es la interacción con el usuario, al fin y al cabo será una interfaz determinada visible por el usuario. Si bien las actividades se complementan para ofrecer una experiencia de usuario uniforme, cada una es independiente de las demás. De esta manera, una aplicación diferente puede iniciar una actividad de otra aplicación, si se permite. Por ejemplo, cuando queremos compartir una imagen o iniciar la cámara desde nuestra aplicación.
- **Servicios.** Al contrario de una actividad, un servicio permite mantener la ejecución de una aplicación en segundo plano sin interfaz, útil para realizar operaciones de ejecución prolongada o procesos remotos. Por ejemplo, un servicio podría reproducir música en segundo plano mientras el usuario se encuentra en otra aplicación o podría capturar datos de la red sin bloquear la interacción con una actividad. Un servicio se implementa como una subclase heredada de `Service`.
- **Receptores de emisiones.** Este componente posibilita que el sistema entregue eventos a la aplicación fuera de un flujo de usuario habitual. El sistema puede entregar emisiones incluso a las aplicaciones que no estén en ejecución. Por ejemplo, una alarma sobre un evento futuro o el nivel de batería bajo. Se puede implementar como una subclase de `BroadcastReceiver` y cada receptor de emisión se entrega como un objeto `Intent`.

- **Proveedores de contenido.** Un proveedor de contenido administra un conjunto compartido de datos de la aplicación, respaldado por SQLite. En ocasiones es necesario este tipo de almacenamiento persistente para, por ejemplo, guardar la información necesaria para establecer la aplicación en un estado concreto.

Para que el sistema Android pueda iniciar un componente de la aplicación, debe reconocer su existencia leyendo el **archivo de manifiesto** de la aplicación, `AndroidManifest.xml`. Además, puede hacer ciertas cosas: identificar los permisos de usuario que requiera la aplicación, establecer el nivel de API mínimo que requiere la aplicación, declarar características de hardware y software exigido y bibliotecas necesarias [43].

3.4.2. Bibliotecas

Una librería o biblioteca, en nuestro contexto de desarrollo de software, es un componente esencial y conveniente. En definitiva es un conjunto de archivos que contienen código y datos, cuya finalidad es ser utilizada por otros programas de forma autónoma y garantizar el cumplimiento de ciertas tareas y funciones para la que se ha sido creada. Para hacer uso de una biblioteca tenemos que agregar una dependencia en nuestro proyecto, dentro del archivo `build.gradle`. A continuación, voy enumerar las distintas bibliotecas utilizadas y su motivo:

- **Room**¹⁴. Una de nuestras necesidades es almacenar información estructurada de forma local en la aplicación. Causado por la necesidad del usuario de explorar su contenido mientras esté sin conexión a Internet. La biblioteca Room brinda una potente capa de abstracción para SQLite, permite acceso local fácilmente.
- **MPAndroidChart**¹⁵. Su creador es Philipp Jahoda. Esta biblioteca está dedicada a la representación visual de gráficos atractivos y versátiles. Es ligera y está disponible como dependencia de gradle. Entre sus características ofrece la escalabilidad en ambos ejes, el arrastre y desplazamiento, la exaltación de valores, plantillas de colores predefinidas, grabación de gráficos, leyendas, animaciones, etc.

¹⁴ <https://developer.android.com/training/data-storage/room?hl=es-419>

¹⁵ <https://github.com/PhilJay/MPAndroidChart>

- **iText**¹⁶. Es una biblioteca de código abierto desarrollada por Bruno Lowagie y Paulo Soares, entre otros. Sirve para crear PDF, RTF y HTML en Java. Contiene clases muy útiles para generar dinámicamente documentos que contengan texto, imágenes, tablas y/o listas, controlando de forma precisa su aspecto. También, se podrá configurar cualquier característica del documento: márgenes, tipos de letras, marcas de agua, etc.
- **Retrofit**¹⁷. Básicamente es una librería para realizar peticiones a una API cuyas solicitudes están basadas en el protocolo HTTP.

3.4.3. Implementación

Aclarar que para desarrollar este presente apartado, y sus consecuentes subapartados, se ha hecho uso de la documentación oficial disponible en línea sobre desarrollo de aplicaciones Android [43]. Esta es muy amplia y detallada y, por ello, haré alusión a partes de ella a través de enlaces con el objetivo de facilitar la lectura y comprensión del lector.

Veremos más detenidamente cómo se ha realizado la implementación de la aplicación móvil y cómo se cumplen cada uno de los requisitos planteados en el inicio (ver Apartado 2.3). Para facilitar la comprensión de la explicación, así como la propia implementación, vamos a presentar una serie de diagramas UML, creados desde el mismo entorno de desarrollo Android Studio con la ayuda del plugin¹⁸ `PlantUML Integration`¹⁹.

3.4.3.1. Base de datos y repositorio

Desde un primer momento vamos a tener la necesidad de acceder a la base de datos local. Por ello es uno de los primeros aspectos a programar. Para desarrollar una base de datos SQLite en Android vamos a utilizar la biblioteca Room (explicada en la Sección 3.4.2).

¹⁶ <https://itextpdf.com/>

¹⁷ <https://square.github.io/retrofit/>

¹⁸ Un plugin es un complemento informático que permite extender las funciones de la aplicación o programa dónde se use.

¹⁹ <https://plantuml.com/es/>

Cada una de las entidades existentes en la base de datos serán representadas con una clase. Los atributos y nombre que tendrán estas clases serán los mismos que los descritos en el Apartado [3.2.3](#). Sin embargo, no debe ser una clase cualquiera de Java, sino una clase que SQLite reconozca como entidad de la base de datos. Esto se consigue con la anotación de Room `@Entity`. Room debe conocer cuáles son los atributos que conforman la entidad, se indicarán con las anotaciones, `@NonNull`, `@Nullable` o `@Ignore`, atributo no nulo, nulo o ignorado por la base de datos respectivamente. En el Código [1](#) se muestra la entidad 'Análisis' ya que es completa y simple.

```
Java
@Entity(tableName = "analisis",primaryKeys = {"ident","parcela"},
foreignKeys = { @ForeignKey(entity = Parcela.class, parentColumns =
{"ident"},childColumns = {"parcela"}), @ForeignKey(entity =
Cuaderno.class, parentColumns = {"cod"},childColumns =
{"cuaderno"}),@ForeignKey(entity = TipoMaterial.class, parentColumns =
{"ident"},childColumns = {"material"})}))
public class Analisis {
    @NonNull
    private Integer ident, parcela, numero,material,cuaderno;
    @NonNull
    private String fecha;
    @Ignore
    private Long fechaDate;
    @NonNull
    private String laboratorio, resultado; ...
}
```

Código 1: Inicio de las preferencias de la aplicación móvil

Una vez creadas todas las entidades que forman la base de datos, la biblioteca Room requiere crear una clase que gestione la base de datos, llamada `BaseDatos`, heredando de `RoomDatabase`. Esta clase sigue un **patrón Singleton** (ver Imagen [89](#)) y debe reconocerse como tal con la anotación `@DataBase`. Es un patrón de diseño de software creacional que nos permite asegurarnos que la clase que sigue el patrón tenga una única instancia y punto de acceso. Este patrón se ajusta a nuestro código, ya que no tiene sentido mantener varias instancias de la misma base de datos, sería redundante y contraproducente [\[45\]](#). La clave para crear el patrón Singleton está en el método estático

`getBaseDatos()`, dónde se crea la instancia del objeto siempre y cuando no se haya hecho previamente (ver Código 2).

```
Java
@Database(entities = {Usuario.class, Parcela.class, ...}, version=1,
exportSchema = false)
public abstract class BaseDatos extends RoomDatabase {
    private static BaseDatos INSTANCIA;
    public static BaseDatos getBaseDatos(Context...context){
        if ( INSTANCIA==null ) {
            synchronized (BaseDatos.class) {
                INSTANCIA = Room.databaseBuilder(context[0],
                BaseDatos.class, "bd").fallbackToDestructiveMigration().build();
            }
        }
        return INSTANCIA;
    }
}
```

Código 2: Inicialización de la base de datos siguiendo el patrón Singleton.

Además, en este fragmento de código podemos ver que se declara una interfaz abstracta llamada `Dao`. Esta interfaz centrará todo el acceso a la base de datos, con el objetivo de evitar realizar una interfaz por entidad. Room debe saber qué interfaz será la que se ocupe del acceso, de nuevo, la identificamos con una anotación, `@Dao`. En dicha interfaz se definen las distintas operaciones DML de manipulación de datos. Normalmente, tendremos un método para insertar, eliminar, actualizar y varios para consultar, en el Código 4 visualizamos las operaciones sobre la tabla 'Tratamiento'. Room comprueba sintácticamente todas estas sentencias SQL en tiempo de compilación.

```
Java
@Dao
public interface Dao{
    @Insert(onConflict = OnConflictStrategy.FAIL)
    Void insert(Tratamiento tratamiento);
    @Update
    int update(Tratamiento tratamiento);
    @Delete
    Void delete(Tratamiento tratamiento);
    @Query("DELETE FROM tratamiento WHERE :id=ident AND
    :parcela=parcela")
```

```
Void deleteTratamiento(Integer id,Integer parcela);
@Query("SELECT * FROM tratamiento WHERE :id=ident AND
:parcela=parcela")
Tratamiento getTratamiento(Integer id, Integer parcela);
@Query("SELECT * FROM tratamiento WHERE :parcela=parcela AND
cuaderno=:cuaderno")
List<Tratamiento> allTratamientosEjercicio(Integer parcela, Integer
cuaderno);
...
}
```

Código 3: Definición de métodos en la interfaz Dao.

La clase `Dao` permite gestionar el acceso y manipulación a la base de datos, sin embargo, desde la actividad no podemos llamar directamente a sus métodos. Es necesario hacer estas consultas mediante hilos en segundo plano, de lo contrario, la interfaz se quedaría bloqueada. Por este motivo, se crea una nueva clase llamada `Repositorio` para encapsular la manipulación de estos hilos en segundo plano. Al igual que con la clase `BaseDatos`, esta clase sigue el patrón Singleton. Para crear los hilos en segundo plano se utilizará la clase `AsyncTask<Parametros, Progreso, Resultado>`²⁰. `Parametros` se refiere a los datos enviados al hilo antes de su ejecución, `Progreso` será al valor devuelto durante la ejecución del hilo y `Resultado` es el resultado al finalizar el hilo. La clase nos ofrece diferentes métodos a sobrecargar que definen un momento concreto de la ejecución del hilo [46].

Todas las relaciones entre clases se describen en la Imagen [89](#).

²⁰ <https://developer.android.com/reference/android/os/AsyncTask>

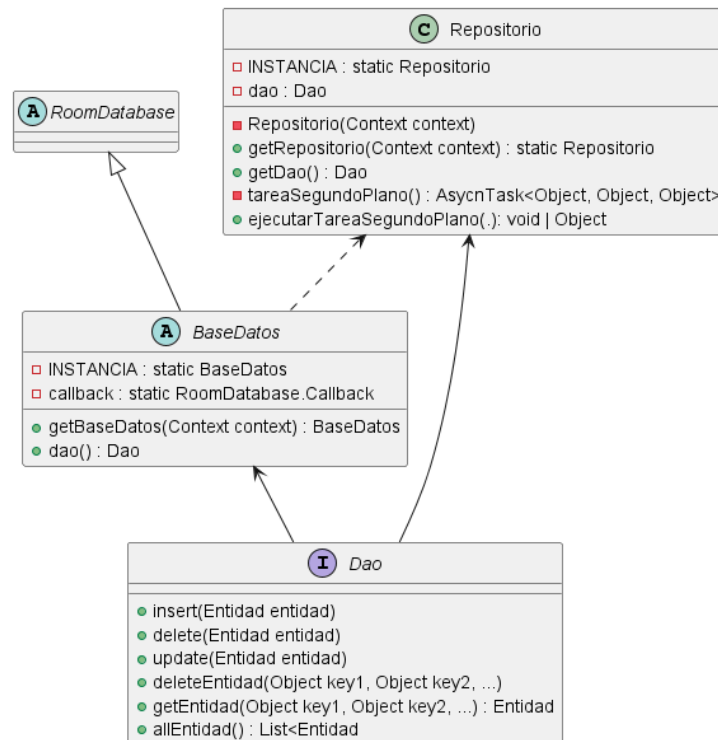


Imagen 89: Diagrama UML de la base de datos y repositorio en la aplicación móvil

3.4.3.2. Consultas a la API

En un primer momento podemos crear registros en la base de datos como ejemplo, pero esto no se acerca al contexto real de la aplicación móvil. Por ello en esta sección se crea la estructura para hacer llamadas a la API (explicada en el Apartado [3.2.6](#)) que nos permitirá acceder a los datos que contiene la base de datos global.

La librería Retrofit, detallada en el Apartado [3.4.2](#), nos ayudará a realizar las peticiones a la API a través del protocolo HTTP. Tomamos la misma filosofía que en la definición de `Repositorio`, cada petición se efectuará sobre un hilo en segundo plano para no bloquear la interfaz. La clase `ServicioApi` contiene la implementación del servicio que accede a la API y de todos los hilos en segundo plano, en el Código [4](#) se muestra cómo se levanta dicho servicio y en la Imagen [90](#) el diagrama entre clases (en la sección [3.4.3.6](#) se comentará la utilidad del atributo `SERVICIO_ORTOFOTO`).

```

Java
public class ServicioApi{
    private static final String RUTA_BASE =
    "https://gffef5ef0a573c7-tfgdb.adb.eu-madrid-1.oraclecloudapps.com/ords/an
a/api/", RUTA_BASE_ORTOFOTO = "https://www.ign.es/wms-inspire/";
    private static IServiciosApi SERVICIO_API;
    private static IServiciosOrtofoto SERVICIO_ORTOFOTO;
    public ServicioApi(){
        try{
            Retrofit retrofit = new Retrofit.Builder()
                .baseUrl(RUTA_BASE)
                .addConverterFactory(GsonConverterFactory.create())
                .build();
            SERVICIO_API = retrofit.create(IServiciosApi.class);
            Retrofit retrofit2 = new Retrofit.Builder()
                .baseUrl(RUTA_BASE_ORTOFOTO)
                .addConverterFactory(GsonConverterFactory.create())
                .build();
            SERVICIO_ORTOFOTO =retrofit2.create(IServiciosOrtofoto.class);
        }catch (Exception ex ){
            System.out.println(ex.getMessage());
        }
    }
    ...
}

```

Código 4: Definición del servicio que accede a la API.

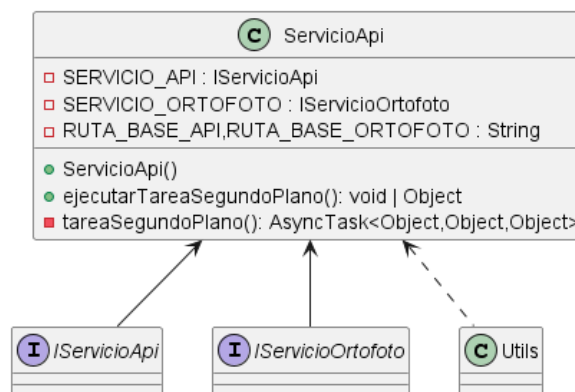


Imagen 90: Diagrama UML del servicio API de la aplicación móvil

La creación de servicio que apunte a una API requiere una interfaz dónde se declaren todas las peticiones posibles, la llamada `IServiciosApi`. En el Código 5 se muestran las operaciones permitidas para la entidad 'Tratamiento' y, en general, todas las casuísticas que se contemplan en la aplicación.

```

Java
public interface IServiciosApi {
    @POST("tratamiento")
    Call<Void> postTratamiento(@Body Tratamiento body);
    @PUT("tratamiento")
    Call<Void> putTratamiento(@Body Tratamiento body);
    @HTTP(method="DELETE",path="tratamiento",hasBody = true)
    Call<Void> deleteTratamiento(@Body Utils.ClaveComun tratamiento);
    @GET("tratamiento/{dni}")
    Call<Utils.TratamientoS> allTratamientos(@Path("dni") String dni);
    @GET("cuadernos")
    Call<Utils.CuadernosS> getCuadernosTecnico(@Query("tecnico") String
        dni);
    ...
}

```

Código 5: Peticiones a la API desde la aplicación móvil.

Cada una de las peticiones tiene asociado un método, GET, POST, PUT o DELETE. Habrá varias formas de pasarle parámetros a la petición; en forma de cuerpo (@Body), en la propia ruta (@Path) o cómo parámetros concatenados en la petición (@Query). La respuesta de una petición será una interfaz parametrizada heredada de Retrofit, Call<T>. Su parámetro será una clase que mapea la información que recibimos en formato JSON en clases de Java.

Finalmente, en el Código 6 se indica cómo se realizan las peticiones (obviando todas las capturas de errores y comprobaciones). Se pueden hacer de forma: síncrona, no se ejecuta la siguiente instrucción hasta que no se tiene la respuesta, y asíncrona, se definen dos eventos, ejecutados cuando se tenga la respuesta.

```

Java
// Síncrono
Call<UsuarioS> call=SERVICIO_API.getUsuario(strings[0]);
Dao dao = Repositorio.getDao();
usuario = call.execute().body().usuario.get(0);
dao.insert(usuario);

// Asíncrono
ClaveComun cosecha = new ClaveComun(ident,parcela);
Call<Void> call2 = SERVICIO_API.deleteCosecha(cosecha);
call2.enqueue(new Callback<Void>() {

```

```
@Override
public void onResponse(Call<Void> call, Response<Void> response) {
    System.out.println(response.code());
}
@Override
public void onFailure(Call<Void> call, Throwable t) {
    System.out.println(t.getMessage());
}
});
```

Código 6: Interfaz de peticiones a la API.

3.4.3.3. Clase general

De manera general, decidimos adoptar una clase base para cada pantalla de la aplicación, es decir, cada actividad. Será gestionada con una clase llamada `ActividadBase` donde incluimos los aspectos comunes que tendrán todas las actividades de la aplicación

A su vez esta clase hereda de la clase `AppCompatActivity`,²¹ sucesora de `Activity`²², clase que abstrae toda la funcionalidad e interacción de una vista. Como podemos ver en la Imagen 91, en `ActividadBase` definiremos el menú de herramientas que aparecerá en la parte superior de la pantalla. `AppCompatActivity` nos ofrece el método `onCreateOptionsMenu(.)` para representar el propio menú y el método `onOptionsItemSelected(.)` para definir la funcionalidad de cada opción del menú. La definición del menú se realiza mediante lenguaje XML, en el Código 7 aparece cómo se implementa nuestro menú.

²¹ <https://developer.android.com/reference/androidx/appcompat/app/AppCompatActivity>

²² <https://developer.android.com/reference/android/app/Activity>

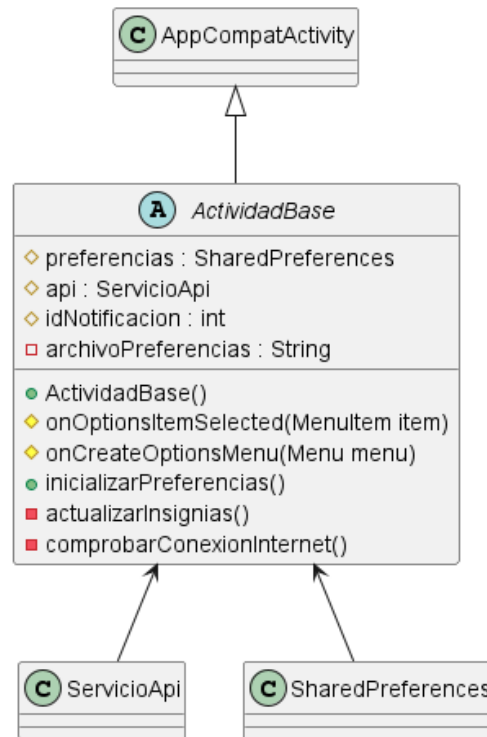


Imagen 91: Diagrama UML del esquema general de cada actividad de la aplicación

Java

```

<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:app="http://schemas.android.com/apk/res-auto">
  <item
    android:id="@+id/servidor"
    android:title="Subir al servidor"
    android:orderInCategory="1"
    android:icon="@drawable/subir"
    app:actionLayout="@layout/badge_subir"
    app:showAsAction="always" />
  <item
    android:id="@+id/actualizar"
    android:title="Actualizar contenido"
    android:orderInCategory="2"
    android:icon="@drawable/actualizar"
    app:showAsAction="always" />
  <item
    android:id="@+id/cerrarSesion"
    android:title="Cerrar sesión"
    android:orderInCategory="3"
    android:icon="@drawable/cerrar"
    app:showAsAction="always" />
</menu>
  
```

```
</menu>
```

Código 7 : XML para definir un menú en Android

El elemento `<menu>` debe ser el nodo raíz del archivo y cada opción del mismo será un `<item>`. Tendrá un identificador (para asignarle una respuesta), un título, un orden de prioridad (establece el orden en la barra de herramientas), un icono y el modo de mostrar dicha opción. En nuestro caso, vamos a mostrar las opciones siempre en la barra, pero puede pasar que se quiera colapsar en un menú vertical (tres puntos), en este caso, el título cobra más sentido.

A lo largo de la aplicación tendremos un menú común con tres opciones visualizadas con un icono característico. Además, se utilizarán insignias, es decir, el punto rojo en el icono del menú, parámetro `actionLayout`. Su única función es alertar al usuario que tiene cambios pendientes en su dispositivo que debe subir a la base de datos global, siempre y cuando, se tenga conexión a Internet.

La clase `ActividadBase` también contiene la instancia del objeto `SharedPreferences`²³. Esta clase es una interfaz de Android que sirve para acceder y modificar los datos de preferencias, datos que forman una colección relativamente pequeña de pares clave-valor en tiempo de ejecución. Usaremos el método `getSharedPreferences(String fichero, int modo)` para abrir el archivo que guarda los datos (ver Código 8). Las preferencias están guardadas en modo privado, sólo accesible desde nuestra aplicación.

```
Java
public void inicializarPreferencias() {
    if ( preferencias==null ) {
        preferencias = getSharedPreferences(archivoPreferencias,
                                           Context.MODE_PRIVATE);
        preferencias.edit().putInt(Utils.CONTADOR,0).apply();
    }
}
```

Código 8: Inicio de las preferencias de la aplicación móvil

²³ <https://developer.android.com/reference/android/content/SharedPreferences>

3.4.3.4. Comienzo e inicio de sesión

Como ya hemos comentado en varias ocasiones la aplicación tiene dos tipos de usuarios, agricultor y técnico, la pantalla de comienzo e inicio de sesión serán comunes. En la Imagen [92](#) se muestra el diagrama UML creado.

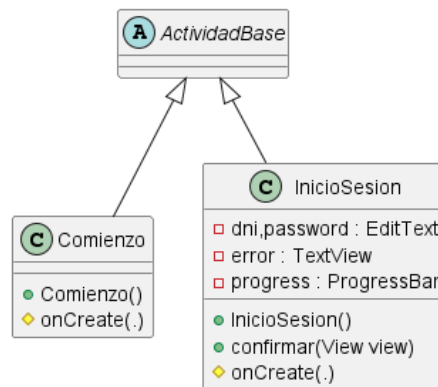


Imagen 92: Diagrama UML de la primera actividad de la aplicación y el inicio de sesión

La primera actividad que verá el usuario al ejecutar la aplicación es la llamada **Comienzo**, visualmente consta de la imagen del logo y una barra de progreso circular sin posible interacción. Internamente se utiliza para comprobar la sesión del usuario, es decir, si el usuario ya ha iniciado sesión previamente, sus credenciales estarán guardadas en el archivo de preferencias y se podrá acceder a la aplicación directamente. De lo contrario, el usuario debe introducir sus credenciales, DNI y contraseña, obligatoriamente. Esta actividad es llamada **InicioSesion**.

Una vez que se ha comprobado las credenciales del usuario la aplicación se redirige a la pantalla principal. La pantalla principal de un agricultor será el listado de parcelas y la del técnico el listado de cuadernos a asesorar. La variación, según el rol, de pantalla principal se implementa mediante un receptor de emisiones, es decir un objeto `Intent`. Este objeto se utilizará bastante a lo largo de la aplicación porque nos permitirá definir la interacción del usuario en la aplicación (ver Código [9](#)). Siempre que queramos añadir una nueva actividad a la pila de navegación utilizaremos el objeto `Intent`, cuyos parámetros serán el

contexto actual y la actividad destino, respectivamente. Después, se llamará al método `finish()`, para eliminar de la pila de navegación la actividad actual.

```
Java
...
if ( rolUsuario==Utils.Rol.AGRICULTOR){
    preferencias.edit().putString(Utils.AGRICULTOR,
        usuario.getDni()).apply();
    startActivity( new Intent(Comienzo.this, ListadoParcelas.class));
}else{
    preferencias.edit().putString(Utils.TECNICO,
        usuario.getDni()).apply();
    startActivity( new Intent(Comienzo.this, ListadoCuadernos.class));
}
finish();
```

Código 9: Iniciar y dirigir a la pantalla principal según el rol del usuario.

Toda actividad de nuestra aplicación heredará de la clase `ActividadBase`, por lo que siempre se sobrecarga el método `onCreate()`, cuya función es inicializar todas las variables utilizadas. También, es ahí dónde se inicia la interfaz, instrucción `setContentView(R.layout.interfaz)`. En algunas clases heredadas se sobrecarga el método `onResume()`, invocado cuando la actividad pasa al primer plano de navegación. En la Imagen 93 vemos el resultado de ambas actividades y los elementos de la interfaz que se corresponden con los siguientes objetos de android:

- `LinearLayout`²⁴: diseño que organiza otras vistas ya sea horizontal o verticalmente, son contenedores, no son visibles en pantalla.
- `ImageView`²⁵: muestra recursos de imágenes, por ejemplo `Bitmap` o `Drawable`. Usado comúnmente para configurar el color y escalado de la imagen.
- `TextView`²⁶: este simplemente muestra texto al usuario, se puede editar su aspecto.

²⁴ <https://developer.android.com/reference/android/widget/LinearLayout>

²⁵ <https://developer.android.com/reference/android/widget/ImageView>

²⁶ <https://developer.android.com/reference/android/widget/TextView>

- `EditText`²⁷: introducir y modificar texto. Es necesario definir el tipo de dato de entrada, esto afectará al tipo de teclado desplegado, los caracteres aceptables y la apariencia del texto.
- `ProgressBar`²⁸: indica el progreso de una operación, existe el modo indeterminado o determinado, en función de si se sabe o no el tiempo de la duración de la operación.
- `Button`²⁹: el usuario puede tocar o hacer clic para realizar una acción.

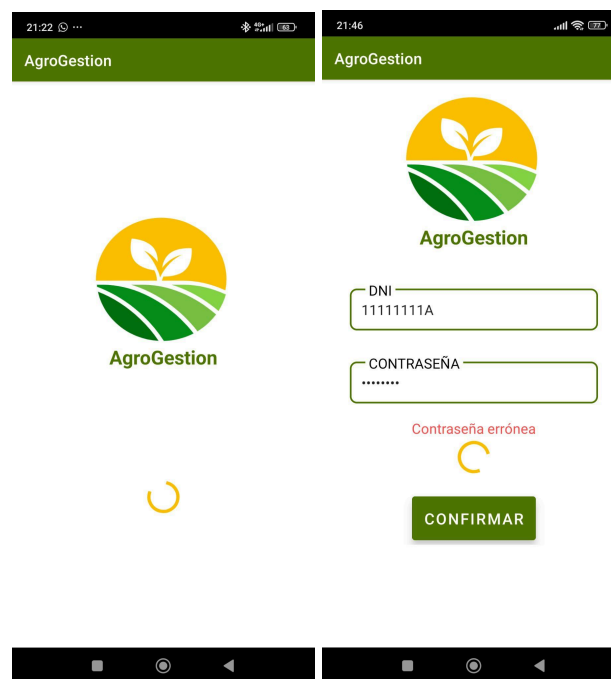


Imagen 93: Resultado de la primera actividad de la aplicación y el inicio de sesión

3.4.3.5. Pantalla principal

Tanto para el agricultor como para el técnico la pantalla principal o pantalla vista después de iniciar sesión se trata de un listado, pero con diferentes datos. Ambas clases no se relacionan entre sí, pero sí mantienen una estructura similar (ver Imagen [94](#)).

²⁷ <https://developer.android.com/reference/android/widget/EditText>

²⁸ <https://developer.android.com/reference/android/widget/ProgressBar>

²⁹ <https://developer.android.com/reference/android/widget/Button>

El elemento utilizado para mostrar los objetivos en la lista será `ListView`³⁰. Permite mostrar una lista vertical desplazable mediante un adaptador. Debemos de crear un adaptador personalizado porque introduciremos dos botones y varios párrafos de texto en cada elemento de la lista. El `AdaptadoParcela` y `AdaptadorCuaderno`, subclases de `BaseAdapter`, contendrán solamente la información que ayudará al usuario identificar rápidamente la parcela o cuaderno.

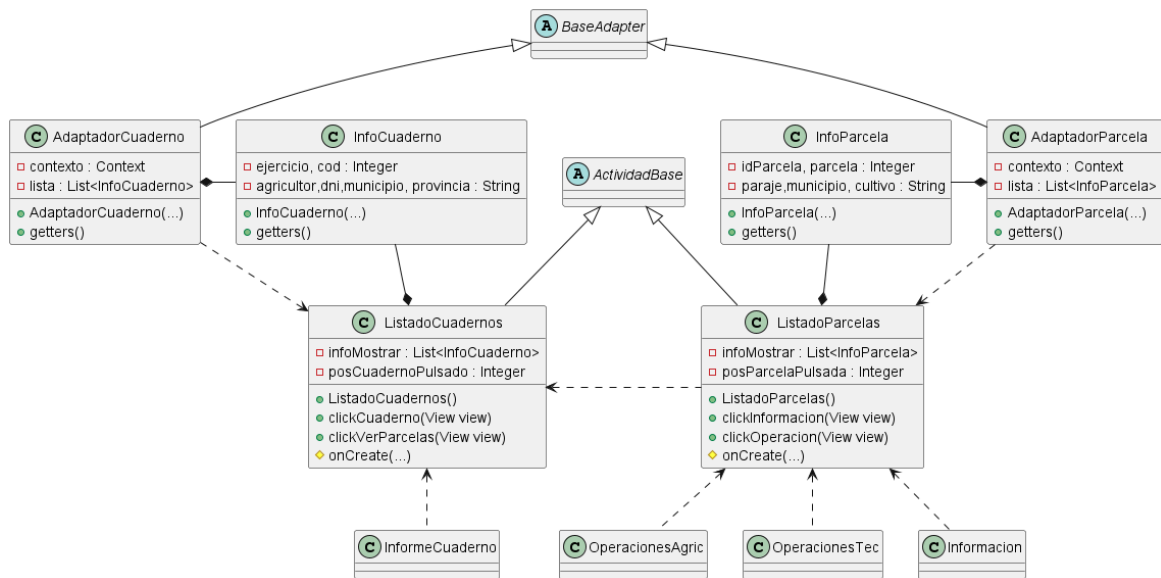


Imagen 94: Diagrama UML con las actividades principales del agricultor y técnico

En la Imagen 95 se visualiza lo explicado anteriormente. Ambos listados permiten pulsar sus elementos y en ambos casos se despliegan dos botones. Para el agricultor se podrá ver la información de la parcela pulsada y las acciones que podrá realizar sobre dicha parcela y el técnico podrá ver las parcelas asignadas al agricultor y realizar el informe del cuaderno de campo del cuaderno pulsado. Todas las dependencias mostradas en el UML de las clases `ListadoCuadernos` y `ListadoParcela` serán las actividades destino mostradas con los diferentes botones desplegados. A lo largo del proyecto se utilizará bastante la lista desplegable y en el Código 10 se detalla cuál es la implementación de la misma, por ejemplo para las parcelas, y de su interacción con el usuario.

³⁰ <https://developer.android.com/reference/android/widget/ListView>

```

Java
ArrayList infoMostrar = new ArrayList<>();
ListView listView = findViewById(R.id.listView);
AdaptadorParcela adaptador=new AdaptadorParcela(this,infoMostrar);
listView.setAdapter(adaptador);
listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> adapterView,
        View view, int i, long l) {
        if ( itemPulsadoAntes!=null ){ // Desaparecen botones
            itemPulsadoAntes.setVisibility(View.GONE);
        } // Aparecer botones del item pulsado
        itemPulsadoAntes = view.findViewById(R.id.listaBotones);
        itemPulsadoAntes.setVisibility(View.VISIBLE);
        posParcelaPulsada = i;
    }
});

```

Código 10: Implementación de una lista desplegable en la aplicación móvil.

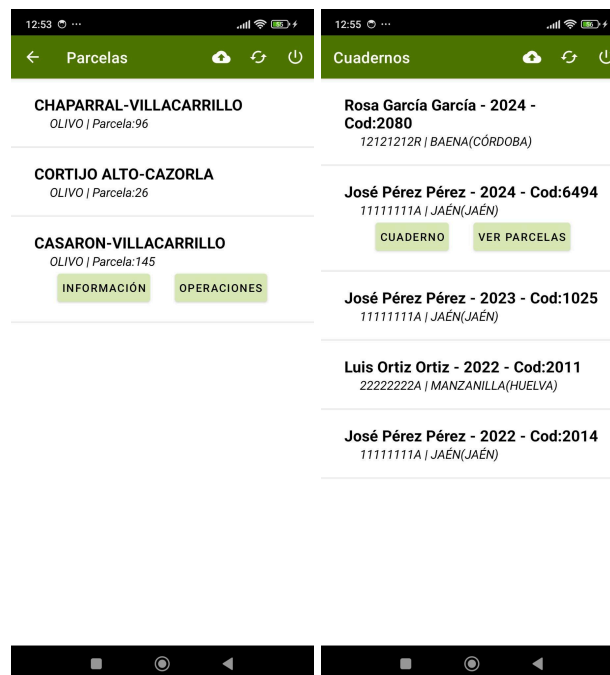


Imagen 95: Resultado de la lista de parcelas (izquierda) y cuadernos (derecha) en la aplicación móvil

3.4.3.6. Información de la parcela

Una parcela tiene muchos datos relacionados, por lo que necesitaremos una pantalla para mostrarlos. Además, añadiremos una ortofoto que represente visualmente dicha parcela. Tanto el agricultor como el técnico podrán acceder a la funcionalidad explicada, es decir, a la actividad `ListaParcelas`. Ya sea como pantalla principal o derivada del listado de cuadernos, respectivamente. Y es esta actividad la encargada de derivarnos a `Informacion`, actividad responsable de visualizar la ortofoto y el listado que reúne todos los datos de la parcela.

Una **ortofoto** es la presentación de una zona de la superficie terrestre en la que todos los elementos están a la misma escala, obtenidas por satélite o por sensores aerotransportados. Desde el Instituto Geográfico Nacional se coordina el Plan Nacional de Ortografía Aérea, cooperativo entre distintas administraciones públicas. Entre los productos que ofrece este plan nos interesa el servicio 'WNS Ortofotos PNOA máxima actualidad'. Este nos ofrece un servicio de descarga de ortofotos a través de Internet.

Por ejemplo si hacemos click en el siguiente enlace: <https://www.ign.es/wms-inspire/pnoa-ma?service=WMS&version=1.3.0&request=GetMap&layers=Ol.OrthoimageCoverage&crs=EPSG:25830&bbox=494985,4215340,495140,4215509&width=700&height=700&format=image/png>, podemos visualizar una parcela. Este enlace está compuesto por diversos parámetros, pero el que define la parcela es el llamado `bbox` (sombreado). Este se basa en una concatenación de coordenadas UTM, los dos primeros números forman el punto inferior izquierdo y los dos últimos números forman el punto superior derecho de la imagen que enmarca la parcela [48].

En términos de implementación, haremos una petición en segundo plano con la biblioteca Retrofit igual que explicamos en el Apartado 3.4.3.2, pero esta vez para hacer peticiones al servicio WNS. La diferencia en la implementación de la petición, con respecto a las peticiones a la API, es el tipo de datos que devuelve la consulta, en este caso, `RespondeBody`. En él viene encapsulada la imagen

como un conjunto de bytes. Sin embargo, para trabajar con imágenes en Android haremos una conversión a `Bitmap`³¹.

No solo vamos a mostrar la imagen, sino que también dibujaremos el contorno de la parcela. Para ello crearemos una escena en OpenGL con el objeto `GLSurfaceView`³², una implementación de `SurfaceView`. También, tenemos que implementar nuestro propio `GLSurfaceView.Renderer`, clase encargada del renderizado de la escena, se basa en estos tres métodos:

- `onSurfaceCreated`: ejecutado cuando la escena se crea por primera vez, es decir, se crean las instancias de los modelos a visualizar.
- `onDrawFrame`: procede al dibujo de la escena inicializada anteriormente. Se carga, compila y enlaza los shaders para definir el renderizado y los datos a utilizar (vértices y colores).
- `onSurfaceChanged`: ejecutado cuando la vista dónde se visualiza la escena ha cambiado de dimensiones, donde se calcula la matriz de proyección de la pantalla disponible.

El elemento a dibujar al fin y al cabo es un polígono, formado por líneas y puntos, sobre un rectángulo cuya textura será la imagen. Trabajamos con los siguientes modelos que hacen analogía a su elemento geométrico:

- **Cuadrado**: el recuadro que va a contener la textura de la ortofoto.
- **Punto**: representa cada vértice del polígono que forma la parcela.
- **Línea**: unión de dos puntos.

Para iniciar nuestra escena, `glView`, a partir de la imagen, le pasamos los puntos límite del cuadrado y las coordenadas UTM del polígono, información que contiene la entidad `Parcela`. Decir que la coordenadas UTM no se pueden representar directamente en la escena OpenGL, hay que hacer una interpolación previa al intervalo $[-1,1]$. Partiendo de las coordenadas UTM y de los puntos que definen el cuadrado utilizamos las siguientes fórmulas:

$$s_x = 2 * \frac{x - x_{min}}{x_{max} - x_{min}} - 1 \quad (1)$$

³¹ <https://developer.android.com/reference/android/graphics/Bitmap>

³² <https://developer.android.com/reference/android/opengl/GLSurfaceView>

$$s_y = 2 * \frac{y - y_{min}}{y_{max} - y_{min}} - 1$$

Donde:

- (s_x, s_y) : punto interpolado en el intervalo $[-1,1]$ dado la coordenadas UTM (x,y) .
- (x_{min}, y_{min}) : coordenadas del punto inferior izquierdo de la ortofoto.
- (x_{max}, y_{max}) : coordenadas del punto superior derecho de la ortofoto.

Todas las clases, método y variables se pueden ver en el diagrama UML en la Imagen [96](#).

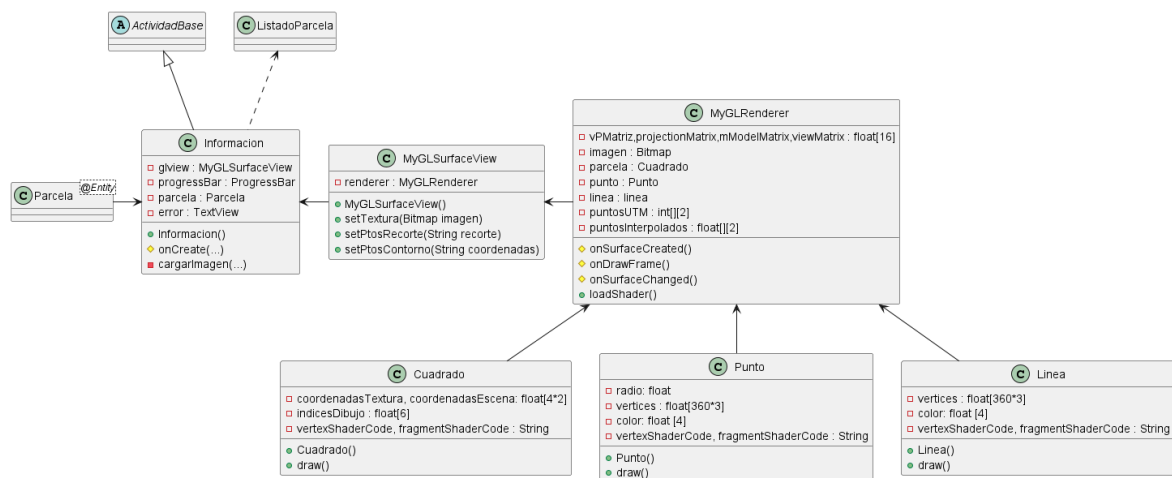


Imagen 96: Diagrama UML de la información de una parcela

En la Imagen [97](#) podemos ver el resultado obtenido para este objetivo. La imagen de la izquierda muestra al usuario que se está produciendo la descarga de la ortofoto, en la imagen central ya se muestra junto al contorno de la parcela y a la derecha se ve la información de la parcela al completo.

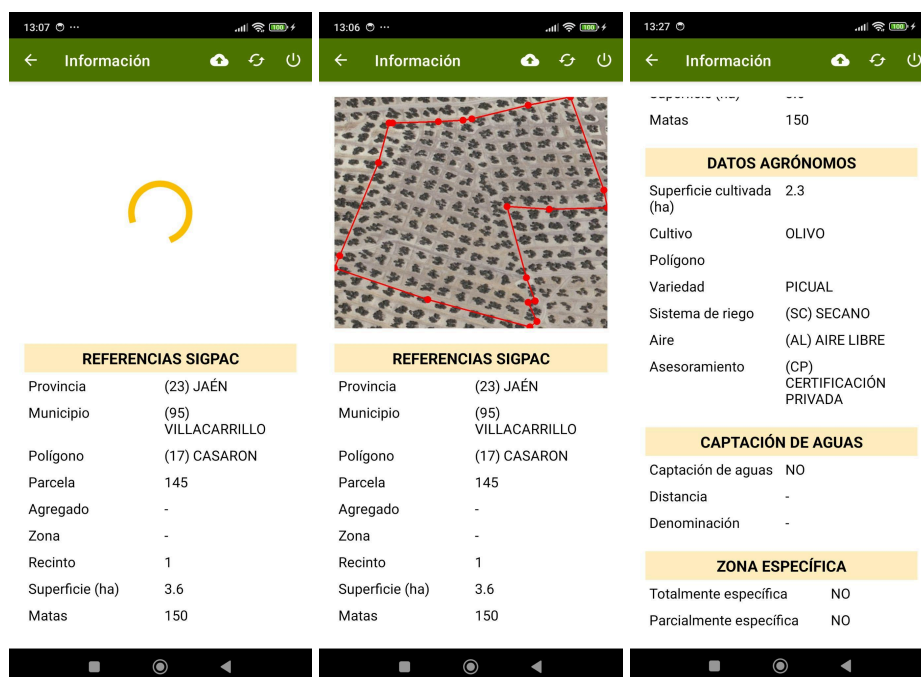


Imagen 97: Resultado de la información de una parcela en la aplicación móvil

3.4.3.7. Menú de operaciones según el rol del usuario

Como hemos dicho en varias ocasiones, el agricultor y técnico tendrá diferentes funcionalidades. Por lo cual tendrán diferentes menús de operaciones.

En esta actividad se visualizará una lista de tarjetas con un icono y texto característico. La tarjeta es creada con el elemento `CardView`³³, lo que nos permitirá disponer las diferentes opciones más ordenadas. Estas actividades, `OperacionesAgric` y `OperacionesTecn`, ya fueron enunciadas en el diagrama dedicado a las pantallas principales (ver Imagen 94), ya que son accesibles desde el botón 'Operaciones' del `ListadoParcelas`. En definitiva es el método `clickOperaciones()` de la actividad `ListadoParcelas` donde se hace la diferenciación entre roles.

Estas actividades no son complejas, simplemente contienen el redireccionamiento de cada opción de los menús. En la Imagen 98 se visualiza su diagrama UML y en la Imagen 99 el resultado de la interfaz. Aclarar que todas las relaciones de dependencias serán explicadas en los posteriores apartados.

³³ <https://developer.android.com/reference/androidx/cardview/widget/CardView>

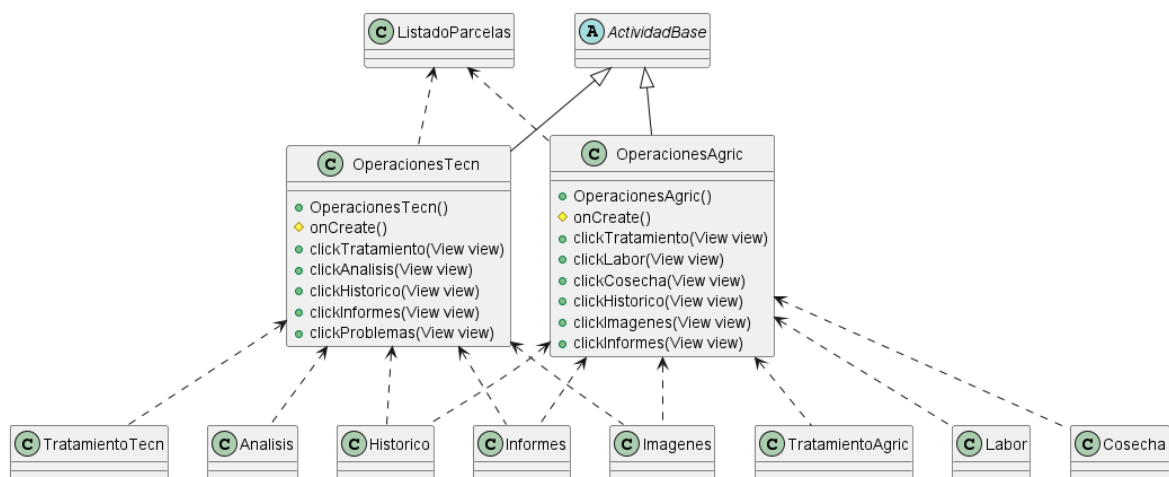


Imagen 98: Diagrama UML de los menús de operaciones en la aplicación móvil



Imagen 99: Resultado de los menús de operaciones en ambos roles, agricultor y técnico, izquierda y derecha respectivamente

3.4.3.8. Visualización, creación, modificación y borrado de tratamientos para agricultores

Dentro de las posibles operaciones disponibles para el agricultor encontramos la llamada 'Tratamientos'. La pantalla que presentaremos en este apartado se ocupa de registrar y mostrar los tratamientos fitosanitarios de una parcela en

concreto. A partir de la Imagen [100](#), desarrollamos los componentes de esta actividad.

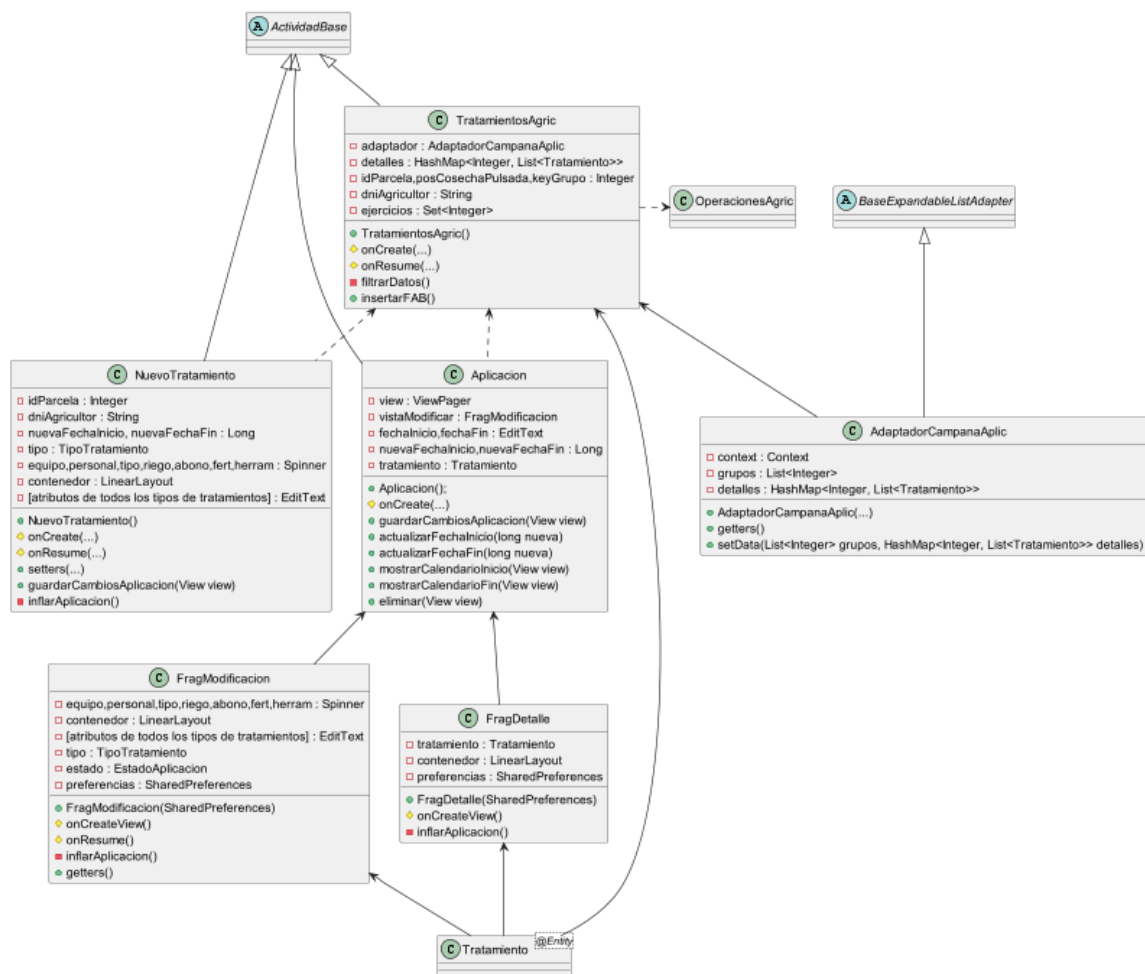


Imagen 100: Diagrama UML de la visualización, creación, modificación y borrado de los tratamientos por parte del agricultor.

En primer lugar, nos encontraremos con una lista expandible, es decir, una lista de dos niveles que se desplaza verticalmente. Para su implementación se utiliza el objeto `ExpandableListView`³⁴. Es muy útil cuando se tiene una lista de elementos muy larga o los elementos tienen una clara distinción entre ellos, ya sea temporal o semántica. En nuestro caso, la clasificación se hará por ejercicios³⁵. En primer nivel se escoge una visualización sencilla donde aparecen los dos años que comprende el ejercicio. Sin embargo, en el segundo nivel mostrará información personalizada del tratamiento. Como consecuencia,

³⁴ <https://developer.android.com/reference/android/widget/ExpandableListView>

³⁵ En la olivicultura un ejercicio empieza con la actividad de la poda durante la primavera y termina con la recolección del fruto en invierno.

sobrecargamos un adaptador de la clase `BaseExpandableListAdapter`, llamado `AdaptadorCampanaAplic`.

Al lado del tipo de tratamiento y de la fecha de inicio de su aplicación, dispuestos verticalmente, tenemos un cuadrado (ver Imagen 101). Dependiendo del estado de la actuación en un momento dado, el cuadrado puede tomar los siguientes colores y sus consecuentes significados:

- Verde: el tratamiento ya ha sido validado por el técnico y todos sus parámetros son correctos.
- Rojo: bien el tratamiento ha sido propuesto por el agricultor, pero el técnico agrónomo no ha validado su aplicación final, o el técnico ha realizado modificaciones.
- Amarillo: el tratamiento ha sido comprobado por el técnico, pero hay algunos parámetros o cuestiones incorrectos, como pueden ser fechas, producto o dosis. El tratamiento no está en condiciones de ser aplicado y el agricultor deberá corregir los aspectos indicados.

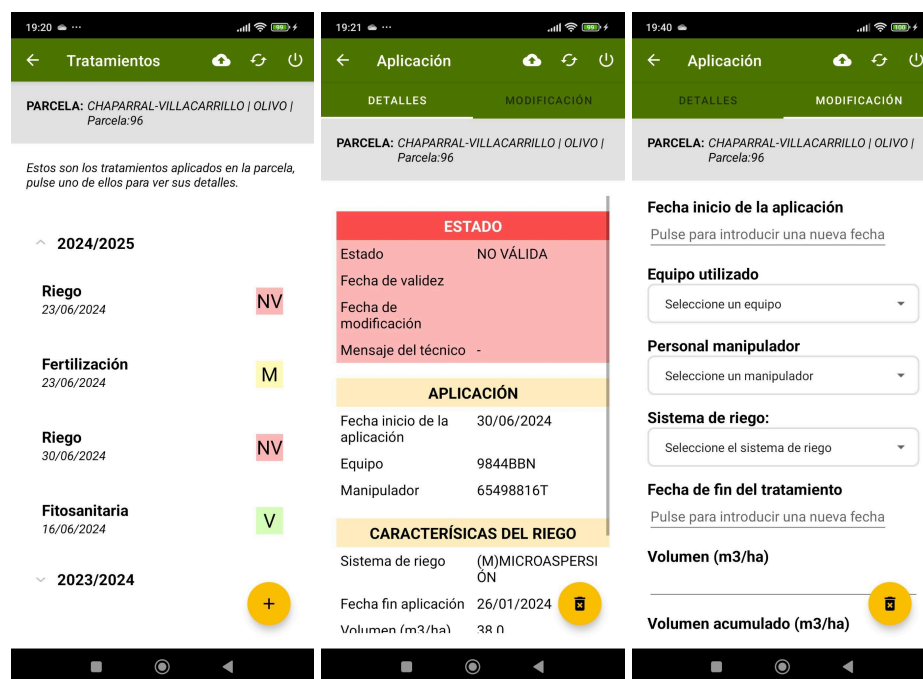


Imagen 101: Resultado de la visualización de tratamientos, de sus detalles y la modificación de sus parámetros

Cuando se pulsa uno de los tratamientos se dirige a la actividad `Aplicacion`. Esta navegación oscila entre una vista que muestra todos los detalles del tratamiento, `FragDetalle`, y otra vista que aporta lo necesario para modificar la aplicación del tratamiento, `FragModificacion`. Estas dos vistas son adaptadas según el tipo de tratamiento, ya que difieren en sus campos. Habrá una excepción, cuando el tratamiento está validado no se permitirá modificar sus parámetros ni eliminarlo por completo, lógicamente en este estado el técnico ya lo ha revisado con unos parámetros concretos y sus cambios ocasionarían incongruencias. Por ejemplo en la Imagen [101](#) se puede ver la lista expandible de dos niveles (izquierda), la visualización de un tratamiento de riego no validado (central) y la posible modificación de cualquiera de sus parámetros (derecha).

Además, al pulsar un tratamiento no validado o modificado, se podrá eliminar. Aparecerá un botón de acción flotante, `FloatingActionButton`³⁶, para dar la opción de eliminarlo. Por ejemplo, en la Imagen [102](#) se muestran los pasos seguidos para eliminar el tratamiento visto en la Imagen [101](#). Antes de eliminarlo se pregunta al usuario una segunda confirmación (izquierda) y mediante un mensaje temporal se notifica (central) si se realiza el borrado con éxito. Finalmente, podemos ver en la imagen central que el tratamiento no aparece en la vista y aparece una alerta en el icono de subida. Esta alerta significa que hay cambios pendientes y tras pulsar el icono se harían las peticiones pertinentes para actualizar la base de datos global (derecha).

³⁶<https://developer.android.com/reference/com/google/android/material/floatingactionbutton/FloatingActionButton>

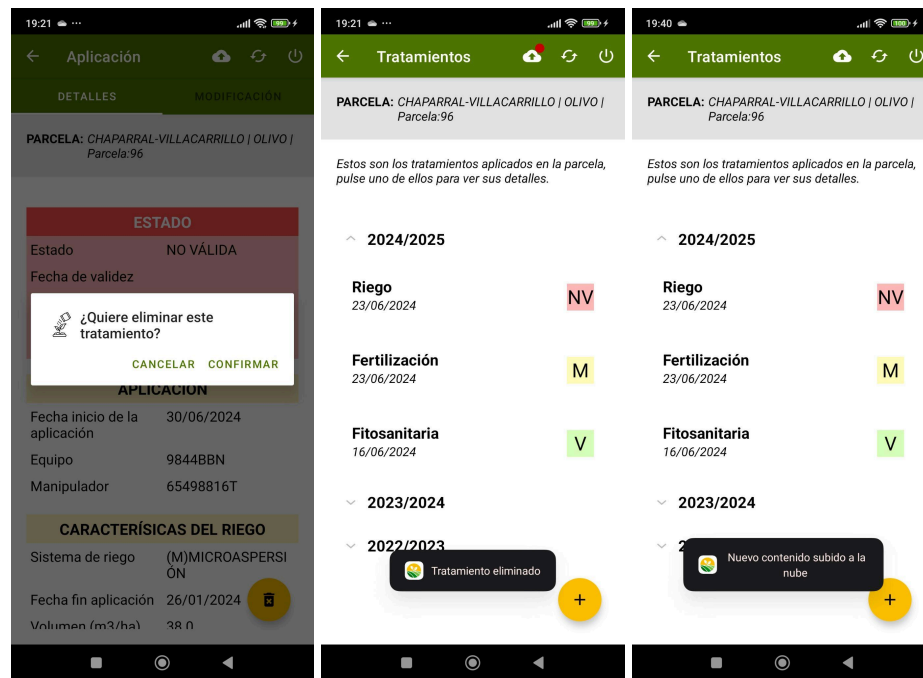


Imagen 102: Resultado del borrado de un tratamiento por parte del agricultor y su posterior subida

Por último, la actividad `Aplicacion` nos permite añadir un nuevo tratamiento. A través de un botón de acción flotante, en la parte inferior derecha, el agricultor podrá acceder a una nueva actividad llamada `NuevoTratamiento`. Nos permitirá rellenar todos los datos correspondientes al nuevo tratamiento, ya que distintos tratamientos tienen características comunes y específicas. Esos parámetros específicos dependen del tipo de tratamiento elegido mediante el `Spinner`³⁷ dispuesto para ello. Un `Spinner` es un desplegable y muestra todos los elementos de una lista permitiendo al usuario elegir solamente uno de ellos.

Como respuesta a la elección de un tipo de tratamiento, se actualizará el contenido de un contenedor, inicialmente vacío, con los elementos correspondientes. Para facilitarle la tarea al agricultor se mostrarán mensajes de error si intenta guardar datos incorrectos o incompatibles con nuestra base de datos. En la Imagen [103](#) representamos la inserción de un nuevo tratamiento de tipo fertilizante, después de elegir el tipo de tratamiento podemos ver en la imagen central cómo se completa toda la información que el usuario debe rellenar.

³⁷ <https://developer.android.com/reference/android/widget/Spinner>

Después de guardar los cambios, se actualiza la lista y el icono notifica de los cambios pendientes.

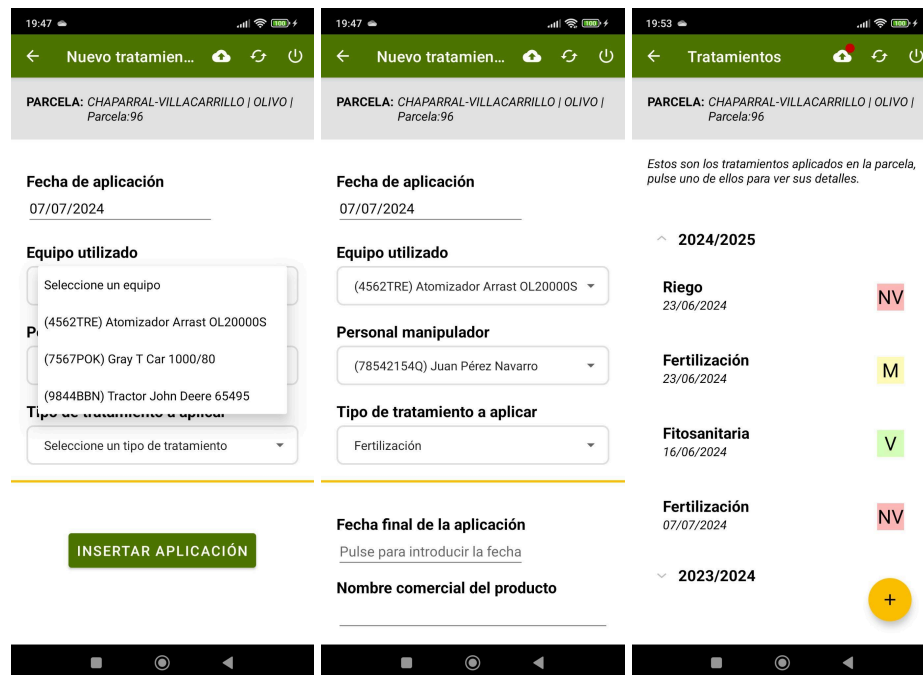


Imagen 103: Resultado de la creación de un nuevo tratamientos de tipo fertilizante

3.4.3.9. Validación de tratamientos para el técnico

La responsabilidad del técnico es validar y comprobar cada tratamiento. A diferencia del agricultor, al técnico se le muestran los tratamientos en una lista simple, `ListView`, ya que son tratamientos relativos a un cuaderno. De nuevo implementamos un adaptador personalizado para visualizar el estado del tratamiento en la lista, llamada `AdaptadorTratamiento`. La actividad a la que dirige cada pulsación de un tratamiento es `EvaluarTratamiento`. En ella se visualiza exactamente la misma interfaz que en la actividad `FragDetalle`, donde se visualiza las especificaciones del tratamiento, pero con el botón flotante que permite evaluar. En la Imagen [104](#) se ve el diagrama de clases UML.

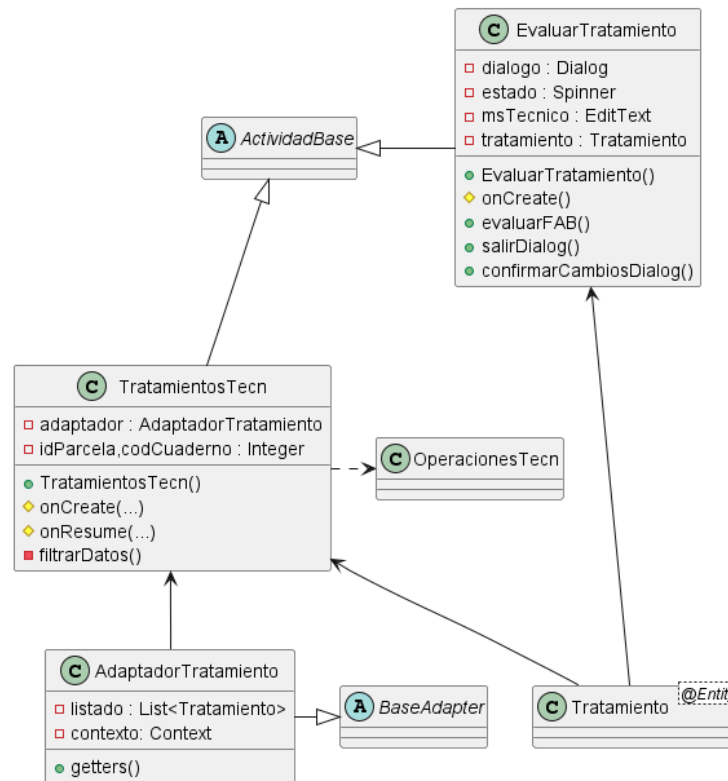


Imagen 104: Diagrama UML de la validación de un tratamiento

En otras ocasiones hemos utilizado lo que se conoce como cuadro de diálogo, implementado en Android mediante el objeto `Dialog`³⁸. Hasta el momento eran diálogos predeterminados con un mensaje, botón de cancelar y confirmar. Sin embargo, en este caso será un diálogo personalizado. Es decir, crearemos un desplegable para que el técnico pueda elegir el estado del tratamiento y una entrada de texto para introducir un mensaje. En el Código 11 se implementa lo dicho anteriormente. En la Imagen 105 tenemos un ejemplo para evaluar un tratamiento, a la izquierda vemos el icono disponible, en la imagen central el cuadro de diálogo con el estado y mensaje de retroalimentación y a la derecha la alerta de cambios pendientes sobre el icono de subida una vez evaluado el tratamiento.

Java

```
public void evaluarFAB(View view) {
    LayoutInflater inflater = getLayoutInflater();
    View dialogView = inflater.inflate(R.layout.dialog_evaluacion, null);
```

³⁸ <https://developer.android.com/develop/ui/views/components/dialogs?hl=es-419>

```

spinnerEstado = dialogView.findViewById(R.id.estado);
ArrayAdapter<Utils.EstadoAplicacion> adaptador = new
    ArrayAdapter<>(this, android.R.layout.simple_spinner_item,
        Utils.EstadoAplicacion.values());
adaptador.setDropDownViewResource(
    android.R.layout.simple_spinner_dropdown_item);
spinnerEstado.setAdapter(adaptador);
msgTecnico = dialogView.findViewById(R.id.msgTecnico);
msgTecnico.setText("");

dialogo = new Dialog(EvaluarTratamiento.this,R.style.Dialog);
dialogo.setCancelable(true);
dialogo.setContentView(dialogView);
dialogo.show();
}

```

Código 11: Implementación de un cuadro de diálogo personalizado.

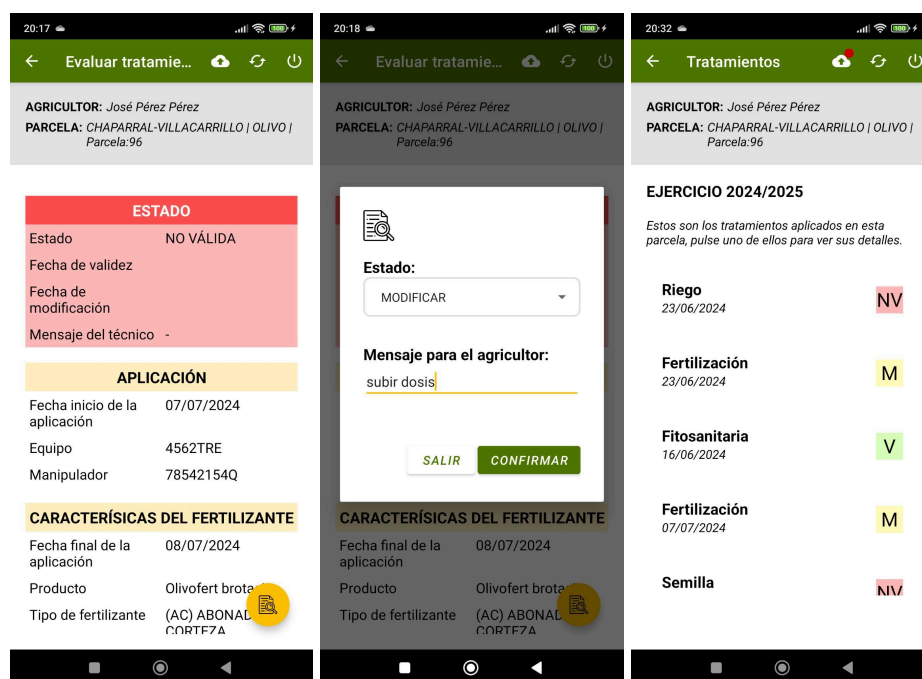


Imagen 105: Respuesta de la validación de un tratamiento

Desde el punto de vista del agricultor, deberá actualizar la información que contiene en la base de datos local si quiere comunicarse con el técnico. Los pasos a seguir están descritos en la Imagen [106](#) y son: primero, pulsar el segundo botón del menú superior (izquierda), después, seleccionar lo que desea actualizar (central) y la base de datos local queda actualizada (derecha). Podemos ver en la Imagen [106](#) cómo el tratamiento borrado en la Imagen [102](#) no aparece en el

listado (izquierda) y el tratamiento validado en la Imagen [105](#) aparece con el estado amarillo en lugar de rojo (derecha).

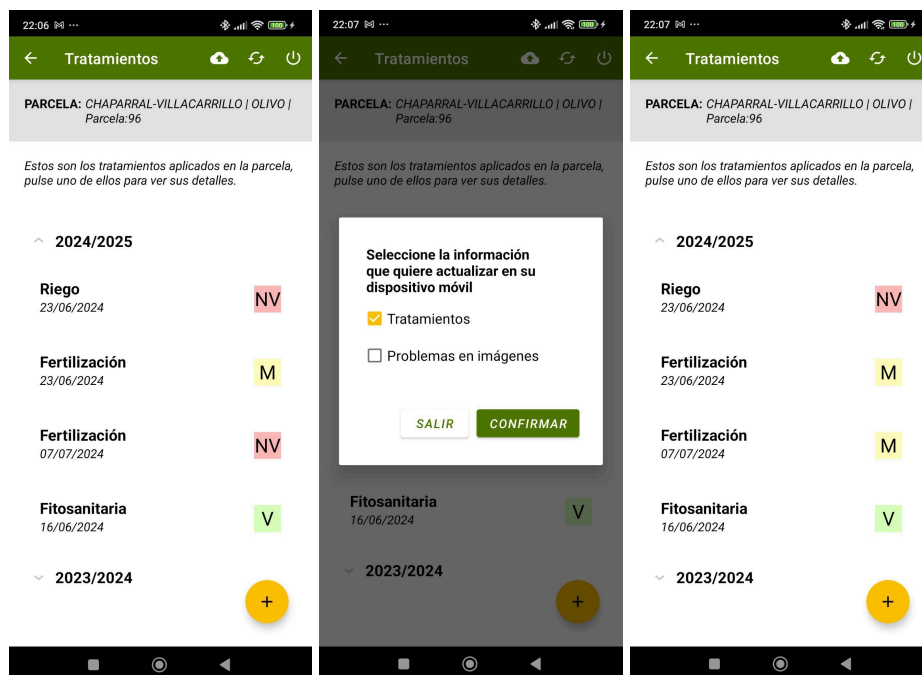


Imagen 106: Respuesta de la actualización de la base de datos local.

3.4.3.10. Visualización, inserción y borrado de labores y cosechas

Otra de las funcionalidades a la que podemos acceder como agricultores es la nombrada 'Labores'. Las labores serán los diferentes trabajos que el agricultor realiza en el campo, como poda, recolección del fruto, entre otros. La actividad principal, descrita en la Imagen [107](#), será `Labor`. En ella las labores estarán clasificadas en ejercicios, en una lista de dos niveles. Al igual que hicimos con la lista de tratamientos (ver Apartado [3.4.3.8](#)), personalizamos la vista del segundo nivel, gracias al adaptador `AdaptadorLaborOperacion`. Esta vez consideramos que la información a presentar será las fechas de inicio y fin del trabajo junto a un pequeño texto que indique las observaciones del agricultor.

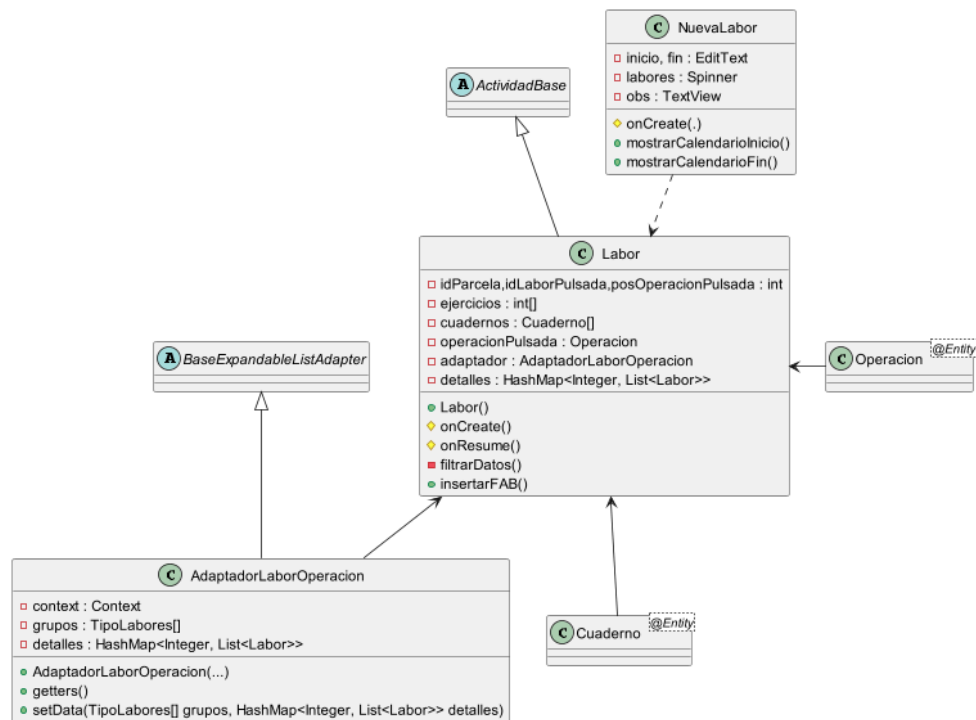


Imagen 107: Diagrama UML de la visualización, creación y borrado de labores

Destacar que cuando se pulsa alguno de los trabajos, saltará un cuadro de diálogo para confirmar o no la eliminación de la labor. Todo esto se puede ver en los resultados de la interfaz en la Imagen [108](#). Intentamos seguir la misma estructura en toda la aplicación, por lo que implementaremos otro botón flotante para redirigir a la creación de una nueva labor, gracias a la clase `NuevaLabor` (ver Imagen [108](#)).

Quizá una de las operaciones más importantes y útiles para el agricultor es la posibilidad de mantener un registro de la cosecha recolectada. La actividad, llamada `Cosecha`, extrae los registros individualizados por parcela de nuestra base de datos local (ver Imagen [109](#)). Se sigue exactamente la misma filosofía, una lista de dos niveles clasificados por ejercicio. No obstante, a diferencia de las labores, cuando se pulsa una de las cosechas se visualizará un diálogo personalizado, como el utilizado en el Código [12](#), pero con diferente interfaz. Este diálogo estará formado por una tabla con los parámetros de la cosecha y la información de la empresa comercializadora del producto. Desde el diálogo también se puede eliminar el mismo registro.

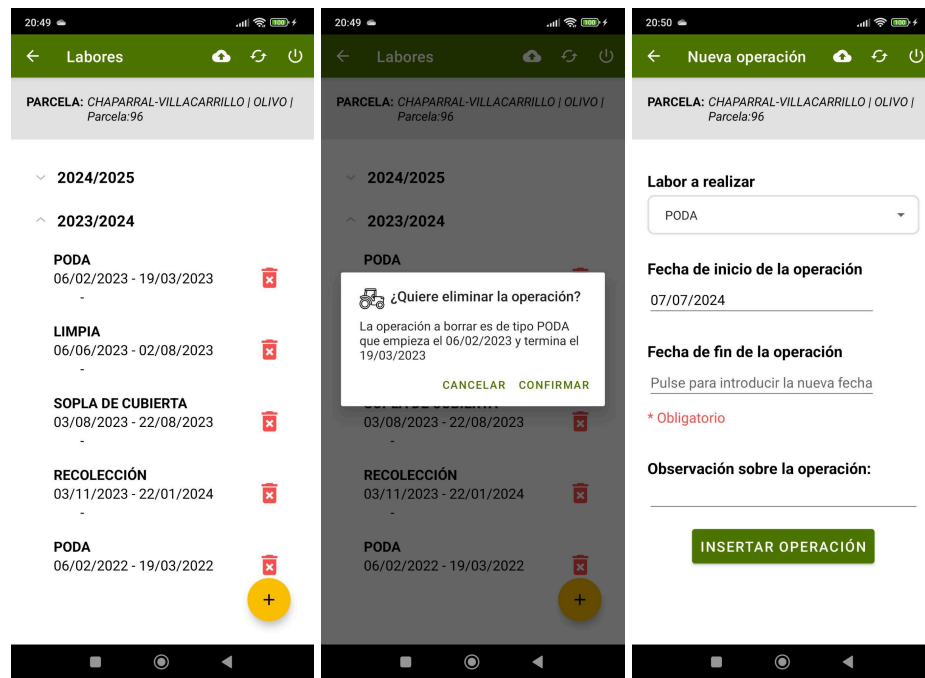


Imagen 108: Resultado de la visualización, creación y borrado de labores

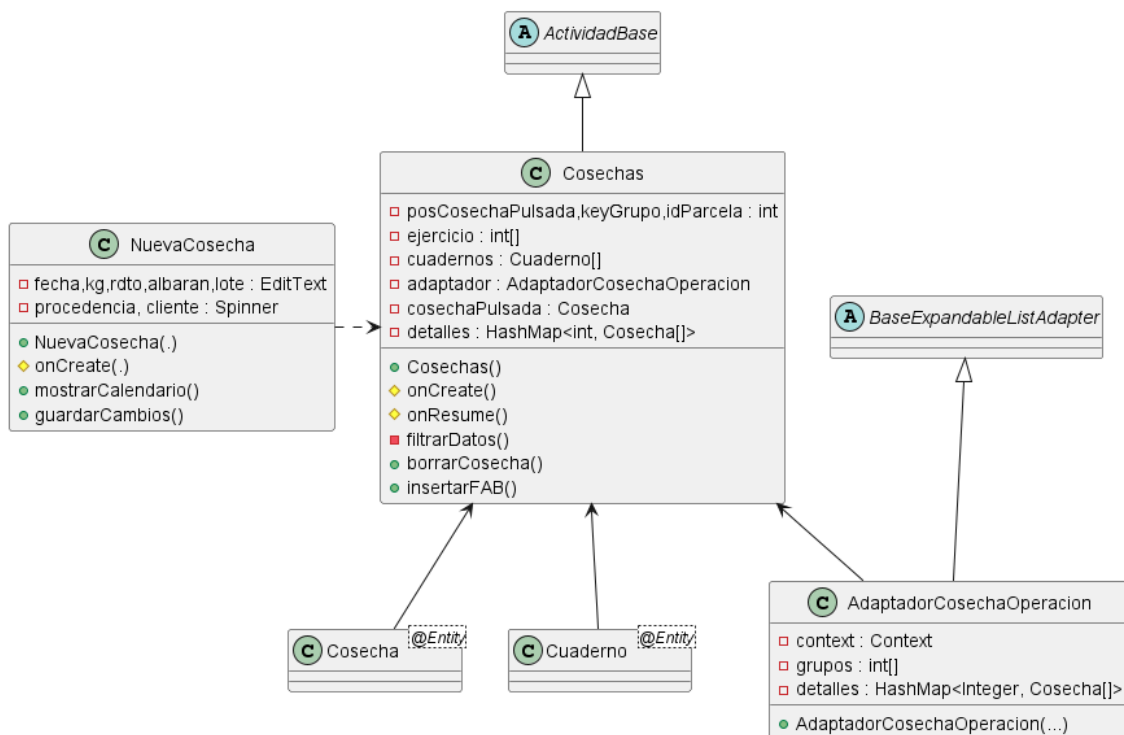


Imagen 109: Diagrama UML de la visualización, creación y borrado de cosechas

Las interfaces resultantes para las cosechas se visualizarán en la Imagen [110](#), a la izquierda se ve el listado de cosechas, al pulsar cualquier registro se vería la imagen central y las características para crear un nuevo registro a la derecha.

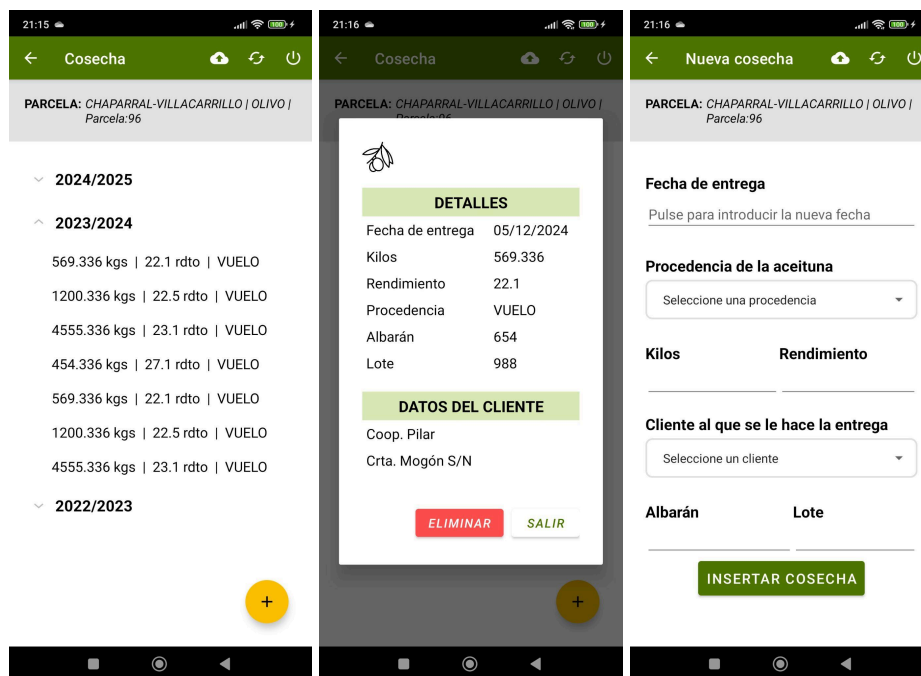


Imagen 110: Resultados de la visualización, creación y borrados de cosechas

3.4.3.11. Visualización, creación y borrado de imágenes y validación por parte del técnico

Esta funcionalidad es muy interesante porque lo que se pretende es mantener una conexión más cercana entre el agricultor y el técnico. Cuando el agricultor detecte algún tipo de anomalía podrá comunicárselo al técnico a través de una imagen. Junto a la imagen se indicará una pequeña descripción de la misma para concretar más aún el problema y un mensaje del técnico como respuesta.

En esta ocasión para listar los problemas almacenados se utiliza una lista simple de tarjetas. Estas tarjetas se implementan gracias a la sobrecarga de un nuevo adaptador, `AdaptadorImágenes`, el cuál agrupa toda la información relevante de la tarjeta: imagen, estado, fecha, problema y respuesta del técnico. También, cuando se pulse la tarjeta aparecerá un cuadro de diálogo personalizado dónde se podrá eliminar la problemática en cuestión, solo permitido

desde el rol del agricultor. El diagrama UML de esta funcionalidad está en la Imagen [111](#).

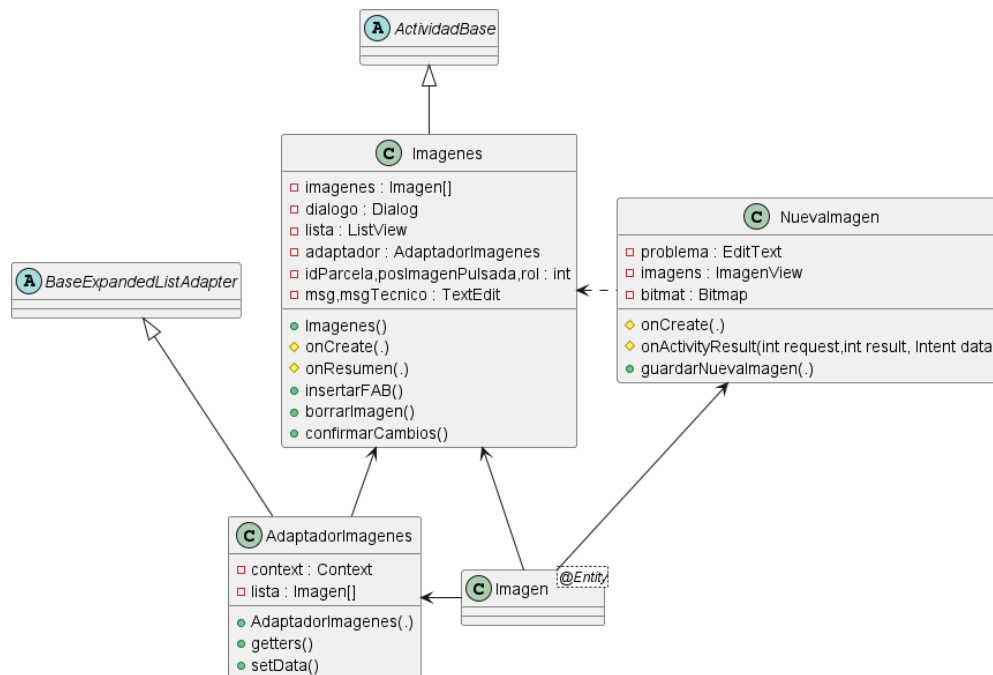


Imagen 111: Diagrama UML de la visualización, creación y borrado de los problemas en imágenes

Obviamente, el agricultor tiene que tener la posibilidad de crear una nueva problemática. La actividad `NuevaImagen` será la encargada de esta función, la mayor dificultad de esta nueva actividad es la selección de una imagen del dispositivo. En ocasiones hemos mencionado el elemento `Intent` de Android, cuyo objetivo era navegar entre actividades de nuestra aplicación. Sin embargo, esta vez se va a utilizar para acceder a una aplicación externa. En este caso, se hará uso de la acción `ACTION_PICK` junto al tipo de dato a seleccionar `MediaStore.Images.Media.EXTERNAL_CONTENT_URI`, permitiendo así mostrar un selector de aplicaciones instaladas en nuestro dispositivo que almacena archivos multimedia, en especial se limitará a imágenes. Una vez que se tiene en la aplicación la imagen se transforma a `Bitmap`. En el Código [12](#) se describe la implementación de la intención, cuya respuesta se captura en el método `onActivityResult()`. Además, en la Imagen [112](#) visualizamos la lista de imágenes antes y después de insertar una nueva imagen junto al selector de aplicaciones para adjuntar una nueva imagen del dispositivo.

```

Java
public void seleccionarFoto(View view) {
    Intent intent = new Intent(Intent.ACTION_PICK,
                               MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
    intent.setType("image/*");
    startActivityForResult(intent.createChooser(intent, "Selecciona
                               aplicación"), COD_SELECCION);
}
@Override
protected void onActivityResult(int requestCode, int resultCode,
                                Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (resultCode == RESULT_OK && requestCode==COD_SELECCION ) {
        mostrar.setVisibility(View.VISIBLE);
        Uri ruta = data.getData();
        try {
            imageBitmap=MediaStore.Images.Media.getBitmap(
                getApplicationContext().getContentResolver(), ruta);
            mostrar.setImageBitmap(imageBitmap);
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
}
}

```

Código 12: Implementación de la apertura de una aplicación externa y su recogida de datos

Es importante tener en cuenta que un mapa de bits ocupa bastante memoria, esto nos dará problemas a la hora de insertar la imagen en la base de datos, tanto en la versión SQLite como en Oracle, en ambos el tamaño de los datos está limitado. Debido a ello, se utilizará una compresión del mapa de bits. La compresión se repetirá hasta que el tamaño del mapa supere el límite, garantizando una calidad de imagen aceptable. La imagen se guardará en una cadena de base 64 repartida en tres columnas de la entidad 'Imagen'.

Para implementar la parte de técnico, la validación de la imagen, la única diferencia es eliminar el botón flotante para añadir y cambiar la interfaz presentada en el cuadro de diálogo accionado cuando se pulse sobre una tarjeta. El técnico podrá introducir un mensaje de texto para responder al problema presentado por el agricultor, visualizado en la parte izquierda de la Imagen [113](#), y la lista actualizada.

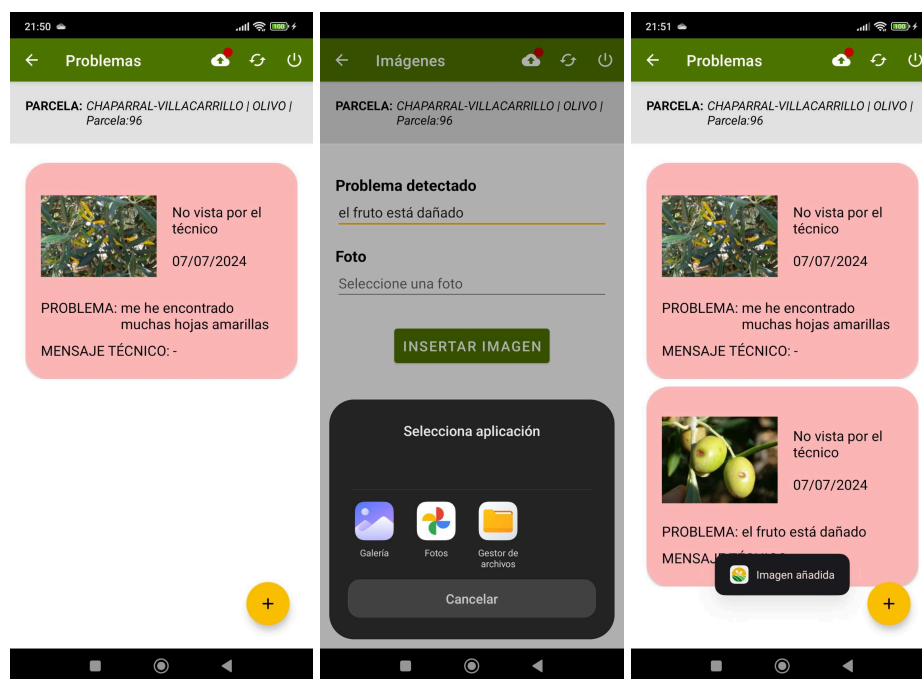


Imagen 112: Resultado de la visualización, creación y borrado de problemas en imágenes

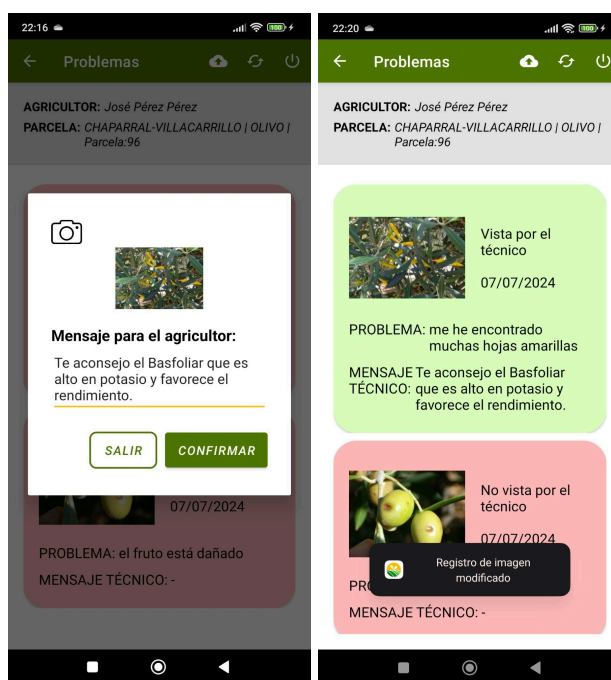


Imagen 113: Resultado de la validación de una imagen problemática

3.4.3.12. Visualización y creación de análisis de laboratorio

Dentro de los epígrafes que comprenden el CUE (ver Apartado [1.6.3](#)) hay uno dedicado a los análisis. Con análisis nos referimos a una prueba de laboratorio realizada sobre el olivo para controlar el estado del fruto o el olivo. Esta funcionalidad solo está disponible para el técnico.

Seguimos con la misma estructura, una lista para mostrar los análisis de una parcela y cuaderno en concreto y un botón flotante para redirigir a la actividad que hará la inserción de un nuevo registro. En la Imagen [114](#) podemos ver el diagrama UML que representa las relaciones entre clases.

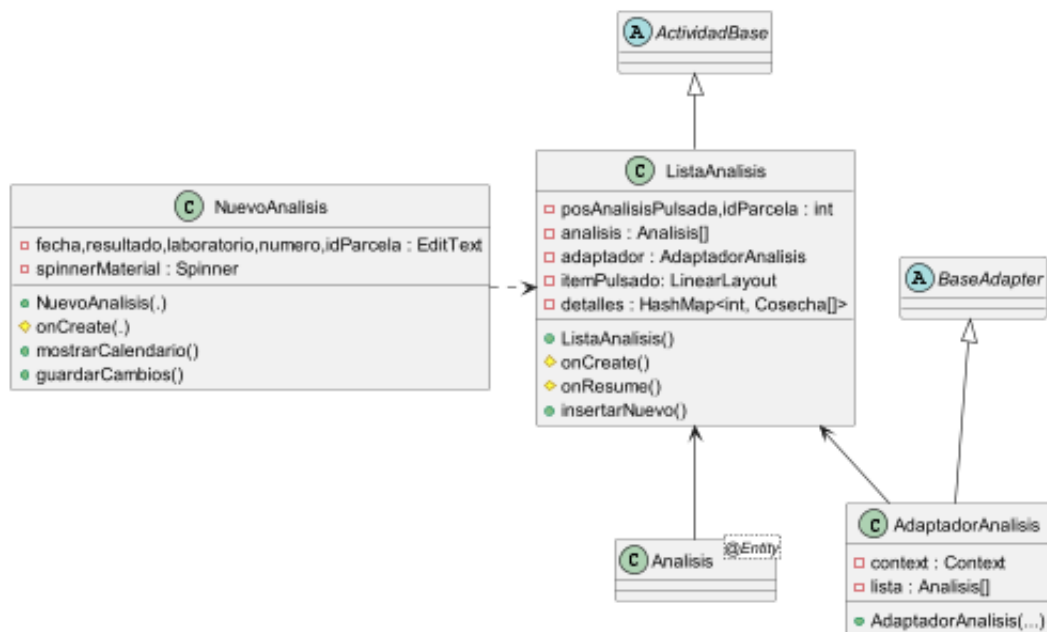


Imagen 114: Diagrama UML de la visualización y creación de análisis

La actividad `ListaAnálisis` será la encargada de mantener la lista de análisis, una lista sencilla creada con el elemento `ListView`. Cada análisis tendrá una vista contraída y expandida, mostrando la fecha de registro y los detalles respectivamente. Personalizamos cada elemento de esa lista con el adaptador `AdaptadorAnálisis`. En la Imagen [115](#) vemos el resultado, en la imagen izquierda tenemos la lista de análisis donde mostramos o no los detalles en cada pulsación.

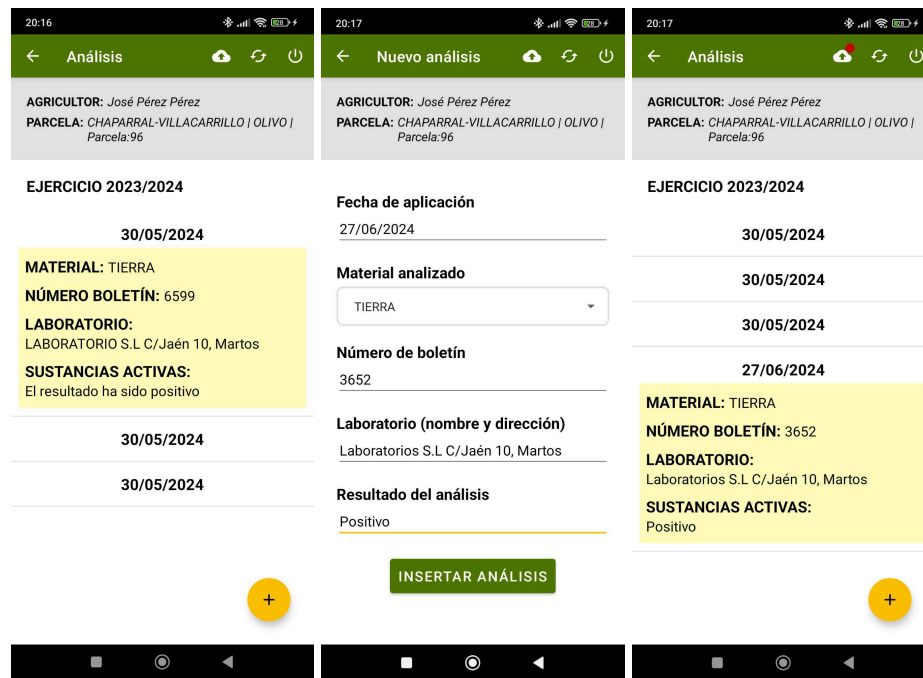


Imagen 115: Resultado de la visualización y creación de análisis de laboratorio

Por otro lado, tras pulsar el botón flotante situado en la parte inferior derecha de la pantalla, podemos insertar un nuevo registro. La actividad `NuevoAnálisis` será la encargada de ello, se inician y comprueban todas las entradas que forman la interfaz del formulario. En la Imagen [115](#) (centro) mostramos la interfaz de la inserción, cuyo botón de confirmar los cambios hace que se almacene el registro en la base de datos local y se active una alerta en el icono de la barra de herramienta superior para actualizar la base de datos global (derecha).

3.4.3.13. Históricos gráficos

Los históricos pretenden representar los datos presentes en la base de datos de forma gráfica, con el único objetivo de ayudar y agilizar la visión de los mismos tanto para el agricultor como para el técnico. Su diagrama UML está en la Imagen [116](#). Se hará uso de la librería `MPAndroidChart`.

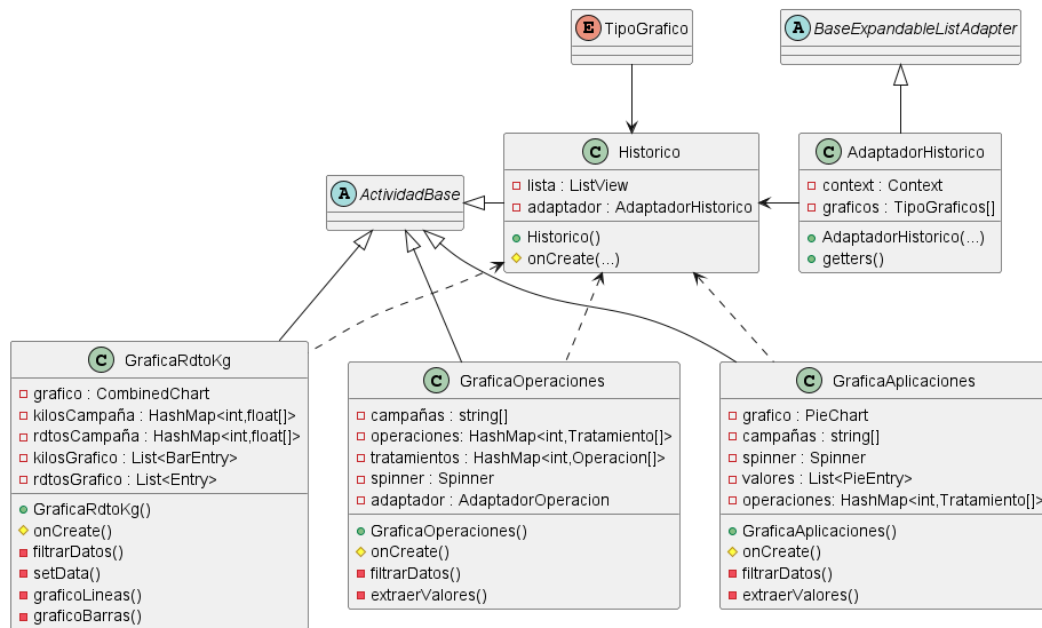


Imagen 116: Diagrama UML de los históricos

De momento se han definido tres tipos de históricos. El primero se trata de la conjunción de dos tipos de gráficos, actividad `GraficaRdtoKg`, uno es un gráfico de barras donde se muestran los kilos de aceituna recolectada durante cada una de las campañas registradas en una parcela. El otro es un gráfico de líneas cuyos puntos son el porcentaje medio de rendimiento asociado a cada campaña. Todo esto se entiende mejor aludiendo a la vista final obtenida en la Imagen [117](#). Además, cuando se pulsa sobre alguno de los ejes de cualquiera de las campañas podemos observar la cantidad de kilos de aceite producidos en función de los kilos de aceituna y su rendimiento.



Imagen 117: Resultado de la lista de históricos y el gráfico de rendimientos y kilos

En segundo lugar, el gráfico circular, actividad `GraficoAplicaciones`. Lo que se pretende es representar el peso de las actuaciones fitosanitarias aplicadas por ejercicio en una parcela en concreto, con la finalidad de mostrar la importancia de algunos tratamientos con respecto a otros. Por último, el gráfico de operaciones, actividad `GraficoOperaciones`, no se utiliza ningún tipo de gráfico sino una lista la cual muestra unos elementos personalizados mediante `AdaptadorHistorico`. La finalidad es unificar las labores y tratamientos en una lista ordenada por fecha (ver Imagen [118](#)).

En el Código [13](#) podemos ver cómo se declaran dichos gráficos en la interfaz. En cuanto a la inserción de los datos, para cada tipo de gráfico crearemos un conjunto de datos, `BarDataSet`, `LineDataSet` y `PieDataSet`, para el gráfico de barras, de líneas o circular respectivamente. Cada uno tendrá una lista de entradas diferentes, `BarEntry`, `Entry` y `PieEntry`, respectivamente. Con el objetivo de crear el propio gráfico con los objetos `BarData`, `LineData` y `PieData`.



Imagen 118: Resultado del gráfico de aplicaciones (izquierda) y calendario de operaciones (derecha)

Java

```
<com.github.mikephil.charting.charts.CombinedChart
    android:id="@+id/barChart"
    android:layout_width="match_parent"
    android:layout_height="600dp"
    android:layout_margin="5dp" />
<com.github.mikephil.charting.charts.PieChart
    android:id="@+id/pieChart"
    android:layout_width="match_parent"
    android:layout_height="400dp"
    android:layout_margin="5dp" />
```

Código 13: Declaración de un gráfico combinado y circula con MPAndroidChart

3.4.3.14. Creación de informes y cuadernos de campo

La creación de informes pretende reunir distinta información sobre una misma parcela para exportarla al formato PDF. Como sabemos PDF es un formato de almacenamiento de documentos digitales independiente de la plataforma tanto software o hardware que lo muestre y almacene. En términos de desarrollo esta actividad es similar a las demás (ver Imagen [119](#)) ya que tendremos un listado

que muestra los tipos de informes a crear y una actividad posterior para crear el propio informe.

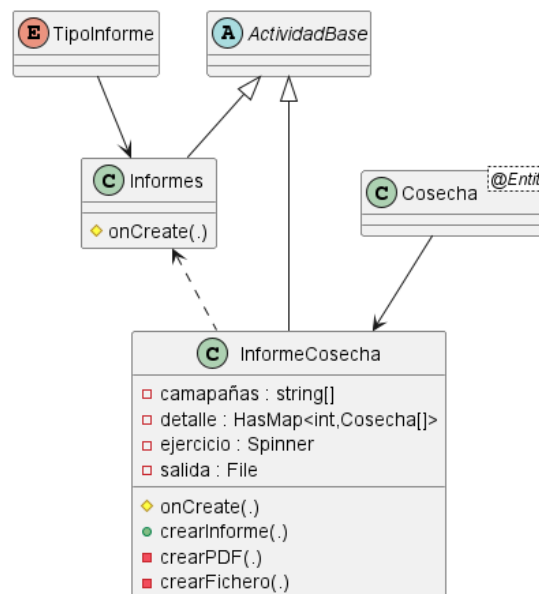


Imagen 119: Diagrama UML de la creación de informes

Por ahora solo hemos creado el informe de cosechas, actividad `InformeCosecha`. Antes de crear el documento se comprobarán los permisos de notificaciones, porque se lanzará una notificación redirigiendo a la apertura del fichero, cuando finalice la creación del documento. En la Imagen [120](#) se muestra la lista de informes (izquierda), la solicitud de permisos (central) y creación del informe (derecha).

En cualquier caso, se acepte o no los permisos, el usuario podrá seleccionar la campaña o campañas de las que quiere el informe. Una vez pulsado el botón, se iniciará la creación del documento en un hilo en segundo plano, indicando la espera con la barra de progreso circular. Para ello nos ayudamos de la librería `iText` y el objeto `File`³⁹ y `FileOutputStream`⁴⁰ de Android. En el Código [14](#) se indica el esquema utilizado para completar el documento. En la función `crearFichero()` se crea un directorio con el nombre de nuestra aplicación, sino existe aún, y se define la ruta de almacenamiento. Posteriormente, el documento estará formado por objetos `PdfPTable` y `PdfPCell` propios de

³⁹ <https://developer.android.com/reference/java/io/File>

⁴⁰ <https://developer.android.com/reference/java/io/FileOutputStream>

iText. Dependiendo de la información se crean y rellenan las tablas pertinentes y que podemos ver en la Imagen [121](#).

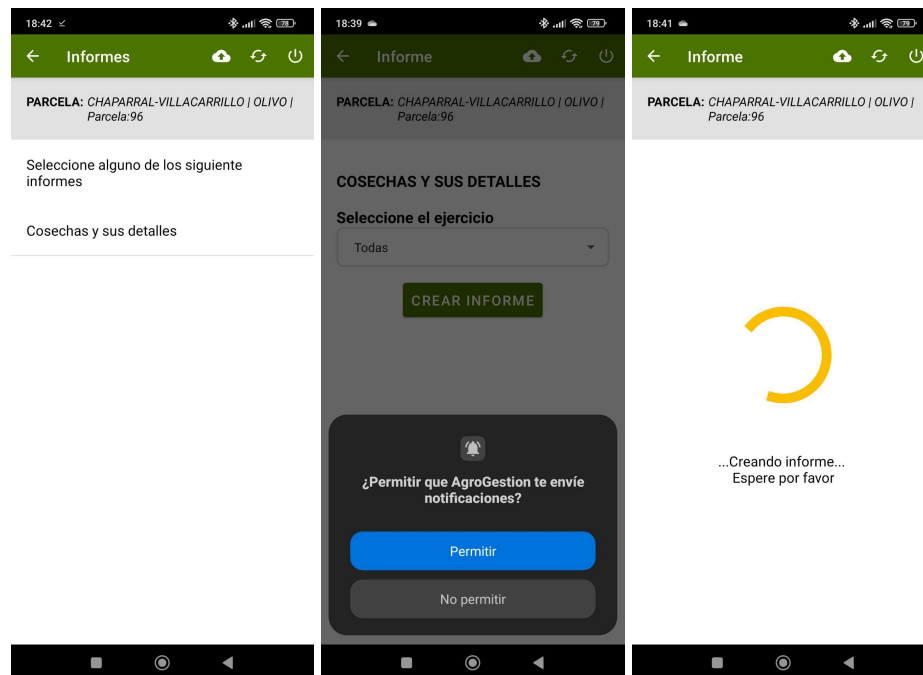


Imagen 120: Resultado del listado de informes, la solicitud de permisos para notificar y creación del fichero

```
Java
// Guardarlo en el almacenamiento interno
File directorio = crearFichero(nombreFichero);
FileOutputStream salida = new FileOutputStream(directorio);
// Crear documento con iText
Document documento = new Document(PageSize.A4, 60f, 60f, 60f, 60f);
PdfWriter.getInstance(documento, salida);
documento.open();
tituloInforme(documento);
tablaCabecera(tabla, yearActual, documento);
tablaResumen(detalles.get(yearActual), tabla, documento);
documento.close();
```

Código 14: Resumen de la creación de un documento con la librería iText

En cuanto a la notificación, se utiliza el objeto `NotificationManager`⁴¹ y `NotificationCompat.Builder`⁴². La metodología que hay que seguir es crear un canal de notificaciones con las características deseadas y, a continuación,

⁴¹ <https://developer.android.com/reference/kotlin/android/app/NotificationManager>

⁴² <https://developer.android.com/reference/androidx/core/app/NotificationCompat.Builder>

definir la propia notificación. Como en nuestro caso se podrá pulsar la notificación para ver el fichero, internamente se creará un `Intent`, cuya acción será `Intent.Action VIEW`. El resultado es representado en la Imagen [121](#).

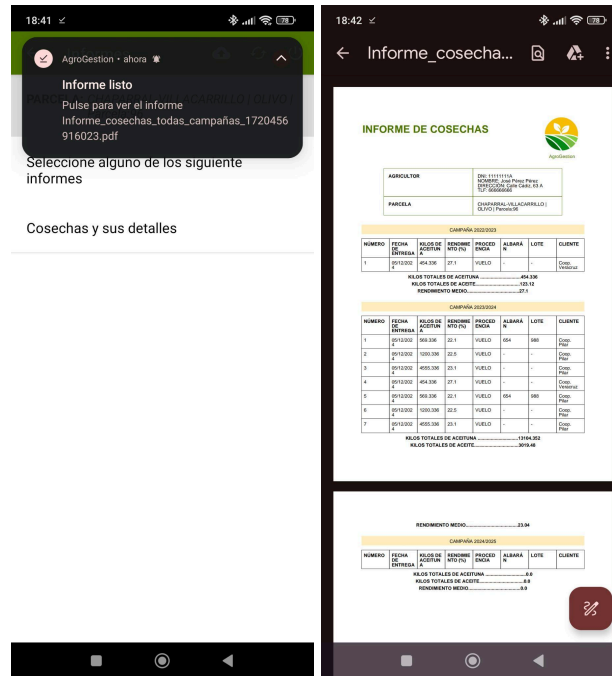


Imagen 121: Resultado del informe de cosechas y su anterior notificación.

Recordar que en Apartado [3.4.3.5](#) comentamos que la pantalla principal que ve el técnico tras iniciar sesión es un listado de cuadernos asignados para asesorar. Pues bien, en la Imagen [122](#) (central) podemos ver el botón desplegado al pulsar un cuaderno, llamado 'Cuaderno', que redirige a la actividad InformeCuaderno (izquierda). Esta clase es idéntica a InformeCosecha, con la diferencia del informe a crear. Para este caso se creará el CUE para un ejercicio concreto, según la normativa y contenido descrito en el Apartado [1.6](#). Tras la finalización del proceso de creación en la imagen central aparece la notificación, si pulsamos sobre ella nos redirige al documento PDF creado (derecha).

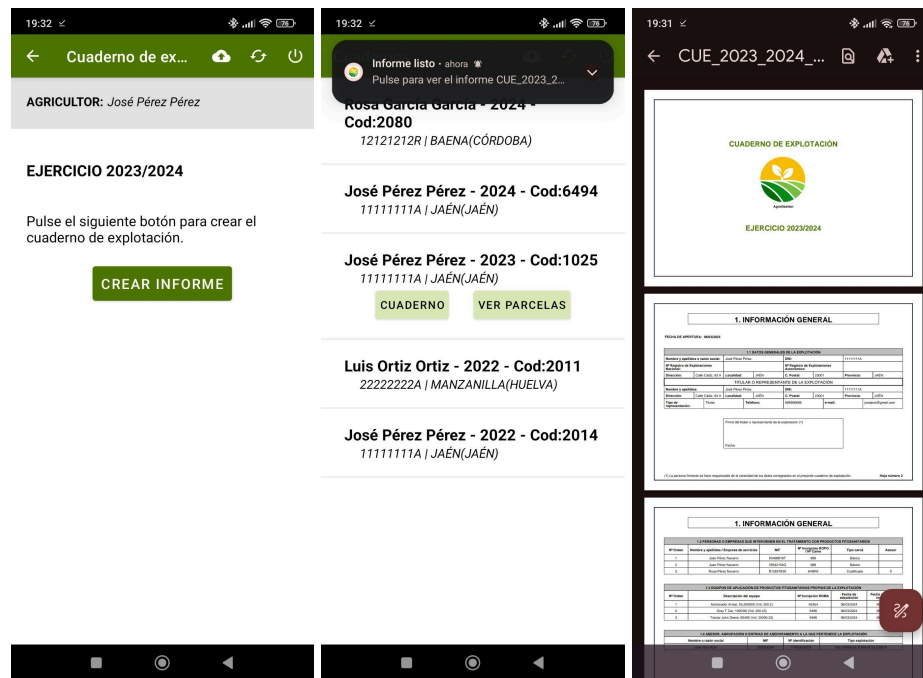


Imagen 122: Resultado de la creación del cuaderno de campo

3.4.4. Pruebas

Durante todo el desarrollo hemos tenido la necesidad de comprobar lo que desarrollamos, así como la validez de las consultas a la base de datos o la buena visualización de la interfaz. Por este motivo a la hora de crear la base de datos la nutrimos con ejemplos inventados, interrumpiendo el funcionamiento normal del sistema, ya que lo ideal sería obtener datos de la base de datos global. Sin embargo, en este momento no se tiene la aplicación web desarrollada para ingresar registros. En la Imagen [89](#) se puede observar la clase `BaseDatos` que contiene la instancia con la que se accede a ella, llamada `INSTANCIA`. Al seguir un patrón Singleton, el método `getBaseDatos()` sólo es llamado una única vez en tiempo de ejecución. Es en esa inicialización relleno la base de datos con datos ficticios, gracias a la definición del evento `addCallback(.)` en `INSTANCIA`. En él se ejecuta en segundo plano todas las inserciones de datos ficticios (ver Código [2](#)).

En conclusión, finalizada está iteración se da por terminada el desarrollo de la aplicación móvil, a la espera de crear la aplicación web para crear datos reales y visualizarlos desde la aplicación móvil.

3.5. Cuarta iteración: Aplicación web

Finalmente, en la presente iteración, ‘Aplicación web’, desarrollaremos la aplicación dedicada al rol de administrador. Este tipo de usuario tiene la responsabilidad de nutrir y gestionar la base de datos. Con el único objetivo de asegurar que el agricultor y técnico puedan realizar sus funciones en la aplicación móvil.

En primer lugar, se contextualiza al lector en el marco de trabajo y lenguajes escogidos para el desarrollo. En segundo lugar, se explicará como hemos cumplidos los requisitos planteados para esta aplicación.

3.5.1. Vue.js

Como explicamos en el análisis de tecnologías para aplicaciones web (ver Apartado [2.6.5](#)), trabajaremos con el marco de trabajo llamado Vue, una herramienta accesible, versátil y potente para construir interfaces de usuario. A diferencia de otros marcos monolíticos, Vue está diseñado desde cero para ser utilizado incrementalmente. La librería central está enfocada solo en la capa de visualización, y es fácil de utilizar e integrar con otras librerías o proyectos existentes. Por otro lado, Vue también es perfectamente capaz de impulsar sofisticadas *single-page applications* o aplicación de página única. Este nuevo enfoque para los sitios web tiene como filosofía reescribir **dinámicamente** una página web actual con datos recibidos del servidor. A diferencia del método predeterminado que es cargar páginas completamente nuevas. Una aplicación de página única tiene como propósito dar una experiencia más fluida a los usuarios, como si fuera una aplicación de escritorio.

Uno de los beneficios que encontramos en Vue es la división que tiene en **componentes**. El sistema de componentes es importante porque nos permite crear aplicaciones a gran escala compuestas de componentes pequeños, independientes y reutilizables. En esencia, un componente es una instancia de un elemento con opciones predefinidas. Llegados a este punto podemos pensar que los componentes de Vue son similares a etiquetas o elementos personalizados, pero proporcionan características que no están disponibles en elementos

sencillos, especialmente el flujo de datos de componentes cruzados, la comunicación de eventos personalizados y las integraciones de herramientas de construcción [49]. En el Apartado 3.5.3 se detalla cual es su implementación y uso en nuestra aplicación.

3.5.2. Bibliotecas

Ya en el Apartado 2.6 enunciamos qué tipos de herramientas y lenguajes íbamos a utilizar en la aplicación web. No obstante, a continuación, listamos las librerías de terceros que hemos usado a lo largo del desarrollo:

- **axios**⁴³: axios es un cliente HTTP basado en promesas para Node.js. Una promesa en JavaScript es un objeto que representa la terminación o el fracaso de una operación asíncrona. Será utilizado para realizar todas las consultas a nuestra API.
- **crypto**⁴⁴: módulo de Node.js que proporciona funcionalidad de criptografía, como funciones HASH, HMAC, cifrado, descifrado, firma y verificación de OpenSSL .
- **devextreme**⁴⁵: conjunto de componentes de interfaz de usuarios con capacidad de respuesta. Todos ellos ofrecen una sintaxis de configuración declarativa y compatibilidad con TypeScript. Podemos encontrar cuadrículas de datos, gráficos interactivos, editores de datos, navegación, etc.
- **vuetify**⁴⁶: una colección de componentes prefabricados combinados con funciones potentes. Su objetivo es proporcionar a los desarrolladores todas las herramientas necesarias para crear experiencias de usuario dinámicas y atractivas.
- **sweetalert2**⁴⁷: librería de JavaScript que ayuda a mostrar cuadros de diálogo al usuario de forma responsiva, personalizable y atractiva.

⁴³ <https://axios-http.com/es/docs/intro>

⁴⁴ <https://github.com/brix/crypto-js>

⁴⁵ https://js.devexpress.com/Vue/Documentation/Guide/Vue_Components/DevExtreme_Vue_Components/

⁴⁶ <https://vuetifyjs.com/en/introduction/why-vuetify/#getting-started>

⁴⁷ <https://sweetalert2.github.io/>

- **vue-cookies**⁴⁸: útil para almacenar datos en la memoria caché del navegador o cookies.
- **vue-router**⁴⁹: es la biblioteca oficial de Vue para realizar el sistema de enrutamiento. Entre sus funcionalidades encontramos: definición, navegación de rutas, rutas anidadas y ganchos de navegación.
- **jspdf**⁵⁰: una biblioteca para generar archivos PDF del lado del cliente desde JavaScript .

3.5.3. Implementación

Es el momento de comenzar a implementar todo lo descrito anteriormente, como consecuencia, se dividirá en epígrafes los aspectos importantes de la aplicación web.

3.5.3.1. Estructura de vista y componentes

Comenzamos describiendo el archivo principal del proyecto, `main.ts`, en él se importan las librerías y variables comunes en todo el proyecto. Además, se encarga de montar o renderizar el proyecto en el navegador (ver Código [15](#)).

```
JavaScript
import Vue from 'vue'
import App from './App.vue'
import router from './router'
import vuetify from './plugins/vuetify'
import VueCookies from 'vue-cookies'
import VueRouter from 'vue-router'
import '@assets/estilos.css'
import utilidades from './herramientas/utilidades'
import 'devextreme/dist/css/dx.light.css';
Vue.config.productionTip = false
Vue.use(VueCookies)
Vue.use(VueRouter);
window.$rutas = utilidades.rutas
new Vue({
  router,
  vuetify,
  render: h => h(App)
```

⁴⁸ <https://github.com/alfhen/vue-cookie>

⁴⁹ <https://router.vuejs.org/>

⁵⁰ <https://artskydj.github.io/jsPDF/docs/index.html>

```
}).$mount('#app')
```

Código 15: Renderizado del proyecto en Vue

Como hemos comentado en el Apartado [3.5.1](#), el tipo de aplicación creada es de página única, por lo que en el Código [16](#) se muestra el código que contiene el archivo `App.vue`, cuya finalidad es definir la estructura de contenedores que almacenan las páginas enrutadas.

```
JavaScript
<template>
  <v-app>
    <v-main>
      <router-view/>
    </v-main>
  </v-app>
</template>
<script lang="ts">
import Vue from 'vue';
export default Vue.extend({
  name: 'App',
  data: () => ({
    //
  }),
});
</script>
```

Código 16: Contenedor de las páginas enrutadas

Otro de los archivos importantes que tiene nuestro proyecto es el `index.ts`. En él se declaran los parámetros y jerarquía de las rutas que componen nuestra aplicación. Por ejemplo, en el Código [17](#) se muestra la estructura de rutas hasta el momento, donde la vista `HomeView` será la ruta raíz y las demás serán sus hijas.

```
JavaScript
import ...
const routes: Array<RouteConfig> = [
  {
    path: '/login',
    name: 'login',
```

```
    props: true,
    component: LoginView
  }, {
    path: '/',
    name: 'home',
    component: HomeView,
    children: [ ... ]
  },
]
const router = new VueRouter({
  mode: 'history',
  base: process.env.BASE_URL,
  routes
})
export default router
```

Código 17: Estructura de enrutamiento de páginas en Vue

Además, vemos necesario modular el código, principalmente, para evitar la redundancia. En concreto, en los archivos `utilidades.ts` y `formularios.ts` tendremos distintas variables y funciones comunes y los objetos que definirán cada uno de los formularios enviados a la API, respectivamente. A lo largo de los siguientes apartados se hará uso de ellos, simplemente habrá que importarlos en el apartado de `script` y hacer uso de sus variables exportadas, `export default`.

3.5.3.2. Inicio de sesión y cookies

Aunque para la aplicación web sólo es accesible el rol de administrador, debemos de controlar el acceso a los datos. Para lo cual, se crea una pantalla dedicada solamente al inicio de sesión.

Obligatoriamente, el usuario deberá de introducir su DNI, incluido la letra, y la contraseña válida. Para crear la interfaz vamos a utilizar la librería `vuetify`, fundamentalmente la etiqueta `v-text-field` y `v-btn`. Nos permite recoger una entrada de texto introducida por el usuario y mostrar un botón. Adicionalmente, introducimos una barra de progreso circular, `v-progress-circular`, para dar una mejor experiencia al usuario. En el Código [18](#) recalcamos el bloque de código que define el formulario, obviando el resto de contenedores.

JavaScript

```

...
<form v-on:submit.prevent="login">
  <h2 style="text-align: center; padding: 1em;"> INICIO DE SESIÓN</h2>
  <v-text-field color="black" label="Usuario" outline
    v-model="formData.Usuario" />
  <v-text-field label="Contraseña" outlined clearable
    :type="showPassword ? 'text' : 'password'" v-model="formData.Passwd"
    :append-icon="showPassword ? 'mdi-eye' : 'mdi-eye-off'"
    @click:append="showPassword = !showPassword" />
  <p class="msgError">{{ msgError }}</p>
  <div class="progress">
    <v-progress-circular v-if="progress" class="bar-template"
      color="primary" indeterminate :height="8" />
  </div>
  <div class="boton-login">
    <v-btn style="padding: 0.75em; font-size: 0.9em;" padding="1%"
      width="100%" type="submit" color="primary2" dark small>
      CONFIRMAR</v-btn>
  </div>
</form>
...

```

Código 18: Formulario de inicio de sesión en la aplicación web

Uno de los atributos más importantes será `v-model`, este acompaña a un componente de Vue para construirlo en base a esa variable. Las variables que el usuario introduzca serán mapeadas en dicho atributo. Es en el evento `login`, ejecutado tras enviar el formulario, donde se utilizarán para obtener los datos introducidos por el usuario. El procedimiento a seguir es solicitar a la API el usuario y comprobar las credenciales, si son inválidas o el rol es inadecuado mostraremos un mensaje de error. De lo contrario se accede al interior de la aplicación.

Cuando se realiza una petición axios se declaran dos eventos, `then()`, ejecutado cuando se tiene una respuesta exitosa de la petición, y `catch()`, lanzado cuando se produce un error durante la ejecución de la petición. En la Imagen [123](#) vemos un ejemplo donde se captura un error, exactamente, cuando el usuario introduce una contraseña errónea.

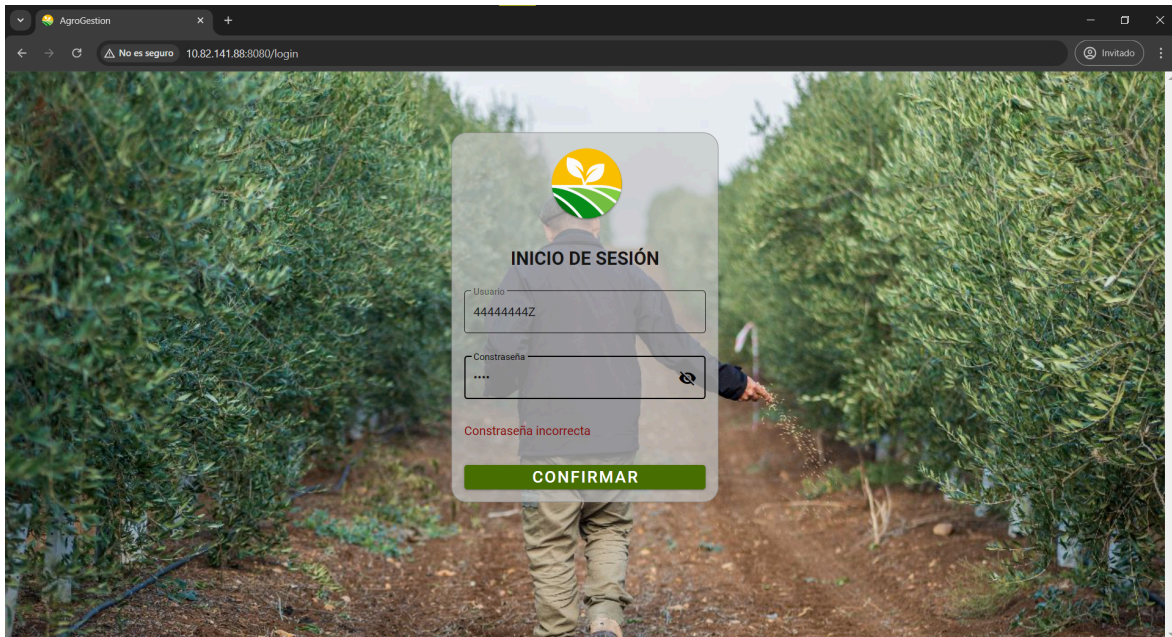


Imagen 123: Resultado de un inicio de sesión fallido en la aplicación web

Siempre que el usuario haya iniciado sesión de forma exitosa se guardarán determinados datos en la memoria del navegador, es decir, cookies. Nosotros almacenamos el nombre y apellidos del administrador, así nos ahorramos peticiones a la API, lo utilizaremos posteriormente para mostrarlos en la cabecera de la aplicación. Podemos acceder a la memoria del navegador gracias a la librería vue-cookies. También, para aumentar la seguridad, antes de guardar la información, codificamos las cookies con el algoritmo AES de la librería cryptojs, aunque se recomienda no guardar datos sensibles.

Para finalizar este apartado vamos a comentar los aspectos de seguridad implementados en nuestra aplicación. Vamos a implementar la autenticación para demostrar la identidad del usuario y evitar la suplantación de identidad. La autenticación tiene lugar en la comprobación de una contraseña, que lo realizamos con el inicio de sesión. La suplantación de identidad la vamos a erradicar con la comprobación de la fecha de expiración de la sesión.

Cuando un usuario se autentifica con éxito son almacenadas en las cookies algunos datos. Luego, la existencia de esos datos nos determinan si el usuario es veraz. Por otro lado, utilizaremos otra cookie en el navegador para almacenar la fecha de caducidad de la sesión. Siempre y cuando la fecha de expiración sea anterior a la fecha actual se elimina la sesión, sino se renueva la fecha de

expiración con la fecha actual. Todas estas comprobaciones se realizan antes de cada petición a la API y antes de enrutar otra nueva página. Tanto la librería axios como vue-router nos permite definir lo que se conoce como interceptores. Un interceptor modifica el comportamiento de un componente de manera transparente. Debido a que es común en toda la aplicación se declara en el archivo `main.ts`. En el Código [19](#) podemos ver cómo implementar un interceptor para cada petición `axios` antes de su ejecución y cómo la librería vue-router nos permite definir el método `beforeEach`. La lógica de los interceptores es la misma, cuando no haya cookies significa que el usuario no ha iniciado sesión y cuando la fecha expiración sea anterior a la actual se elimina sesión.

```
JavaScript
axios.interceptors.request.use(
  function (config) {
    if (!comprobarExpiracion(router.currentRoute.name))
      router.replace("/login")
    return config;
  }
)
Vue.prototype.$axios = axios;
router.beforeEach((to, from, next) => {
  if (!comprobarExpiracion(to.fullPath.replace('/', ''))) { next("/login")
  else next();
  return;
});
function comprobarExpiracion(ruta: string | null | undefined) {
  if (ruta == "login") return true;
  if (Vue.$cookies.get('datosUsuario') == null ||
    Vue.$cookies.get('exp') == null) { // fuera
    Swal.fire("Error", "No ha iniciado sesión en la aplicación", "error")
    return false;
  }
  if (Vue.$cookies.get('exp') != null) {
    var ahora = new Date().getTime();
    var exp = parseInt(Vue.$cookies.get('exp'))
    if (ahora < exp) {
      Vue.$cookies.set('exp',
        new Date().getTime() + 1000 * 60 * utilidades.MIN_EXPIRACION_MAX)
    } else {
      Vue.$cookies.remove('datosUsuario')
      Vue.$cookies.remove('exp')
      Swal.fire("Error", "Se ha detectado una inactividad en la
        aplicación, vuelva a iniciar sesión para verificar su identidad", "error")
      return false;
    }
  }
}
```



```

    }
  }
  return true;
}

```

Código 19: Interceptores para comprobación de sesión en la aplicación web

3.5.3.3. Menú, pantalla principal y cierre de sesión

Nuestra aplicación tendrá más de una página o ruta accesible, por ello, vemos necesario definir un menú que estructure y enlace dichas páginas. Creamos un menú lateral para evitar ocupar demasiado espacio en la pantalla a causa del aumento de opciones. Como hemos dicho anteriormente, la vista `HomeView` será la pantalla principal, de la cual emanan las demás. Por este motivo es en esta vista dónde definimos implementar el menú.

De nuevo, nos ayudamos de la librería `vuetify`. La etiqueta `v-navigation-drawer` tiene unas propiedades interesantes para dar forma a nuestro menú. En concreto `expand-on-hover`, que nos permite alternar entre una versión expandida o contraída cuando se pasa o no el ratón por encima del menú, respectivamente. Junto a la etiqueta `mini-variant.sync` que permite crear el menú contraído. Las diferentes opciones de menú tendrán tres parámetros: un icono, un título y un enlace. Internamente tendremos una lista del tipo `v-list` que contiene los componentes `v-list-item`, `v-list-item-icon` y `v-list-item-title`. También, se añade una cabecera que contiene un texto con el nombre del administrador que ha iniciado sesión y los botones para ir a la pantalla principal o cerrar sesión. En el Código [20](#) se puede apreciar la implementación y en la Imagen [124](#) el resultado.

JavaScript

```

<v-navigation-drawer dark color="primary" v-model="drawer"
  :mini-variant.sync="mini" expand-on-hover permanent>
  <v-list-item class="px-2" style="padding-bottom: 1.5%;">
    <v-list-item-avatar>
      <v-img src="../assets/logo.png"></v-img>
    </v-list-item-avatar>

```



```

    <v-list-item-title>Agrogestion</v-list-item-title>
  </v-list-item>    <v-divider></v-divider>
  <v-list v-for="(item, index) in menu" v-bind:key="index">
    <v-list-item link :to="item.link">
      <v-list-item-icon>
        <v-icon>{{ item.icon }}</v-icon>
      </v-list-item-icon>
      <v-list-item-title>{{ item.title }}</v-list-item-title>
    </v-list-item>
  </v-list>    <v-divider></v-divider>
</v-navigation-drawer>

```

Código 20: Menú y cabecera en la aplicación web

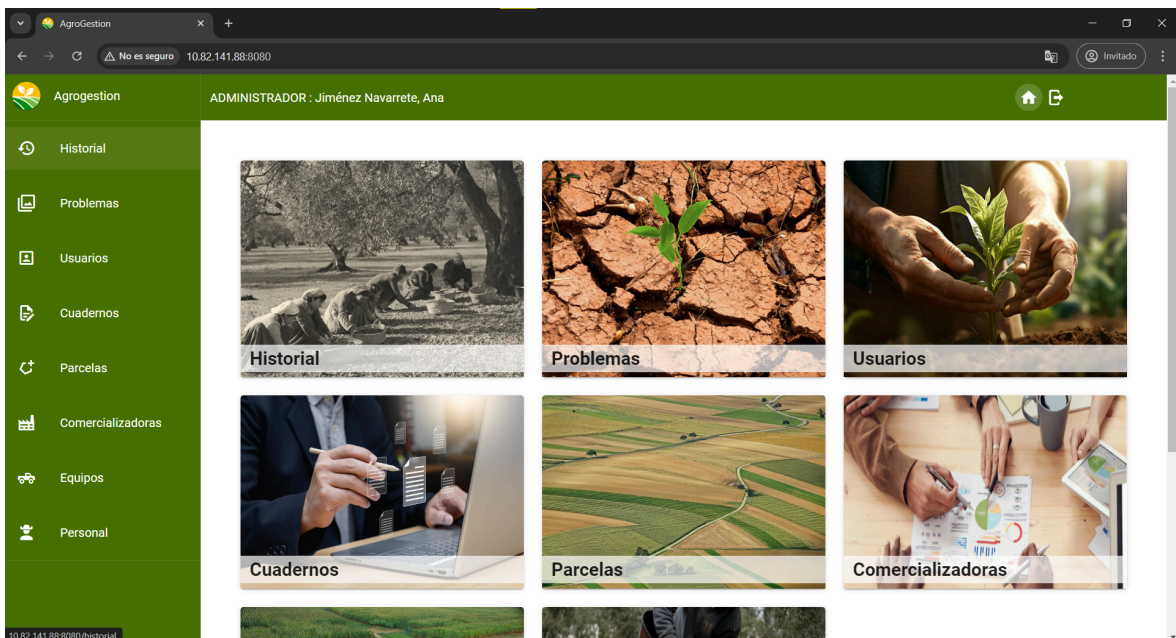


Imagen 124: Resultado del menú y pantalla principal en la aplicación web

La pantalla que se visualiza en la Imagen [124](#) es la llamada pantalla principal. Esta es accesible desde el botón de la cabecera, cuyo icono es una casa y tras iniciar sesión. Lo que se pretende es mostrar el mismo menú que aparece en el lateral pero con tarjetas o `v-card`. Estas tarjetas se ordenan de forma responsiva o adaptable en la pantalla y se encapsulan en el componente `TarjetaTemplate`. Este componente tendrá como propiedad un objeto que contiene: un título, la ruta de su imagen y el enlace. En el Código [21](#) se muestra su implementación al completo.

```

JavaScript
<template>
  <v-card class="tarjeta" hover :to='tarjeta.link' >
    
    <h2 class="titulo-tarjeta">{{tarjeta.title}}</h2>
  </v-card>
</template>
<script lang="ts">
  import Vue from 'vue';
  export default Vue.extend({
    props:{ tarjeta: Object, }
  });
</script>

```

Código 21: Implementación completa de una tarjeta para la pantalla principal

En la vista `PrincipalView` es donde se instancia este componente, gracias al bucle `v-for` se recorre la lista de tarjetas, al mismo tiempo que se renderiza. La adaptabilidad a las dimensiones de la pantalla se debe a las etiquetas `v-row` y `v-col` (ver Código [22](#)).

```

JavaScript
<v-row>
  <v-col cols="12" md="4" sm="6" v-for="nodo in tarjetas" >
    <TarjetaTemplate :tarjeta="nodo"/>
  </v-col>
</v-row>

```

Código 22: Renderizado de tarjetas en la aplicación web

Por último, acabaremos esta sección con el cierre de sesión. El botón habilitado para ello es el que aparece en la parte superior derecha de la cabecera. La lógica de este botón simplemente es eliminar las cookies del navegador, antes de ello se pide al usuario una segunda confirmación. El cuadro de diálogo para que el usuario confirme el cierre de sesión es implementado con la librería `sweetalert2` (ver Imagen [125](#)). En el Código [23](#) se muestra su implementación.

```

JavaScript
logout() {
  Swal.fire({
    title: `¿Desea cerrar sesión?`, icon: 'question',

```

```
showConfirmButton: false, showDenyButton: true,  
showCancelButton: true, denyButtonText: 'Cerrar sesión',  
cancelButtonText: 'Cancelar',  
}).then((res) => {  
  if (res.isDenied) {  
    this.$cookies.remove('datosUsuario')  
    this.$router.replace('/login')  
  }  
})})
```

Código 23: Definición de un cuadro de diálogo y la eliminación de cookies.

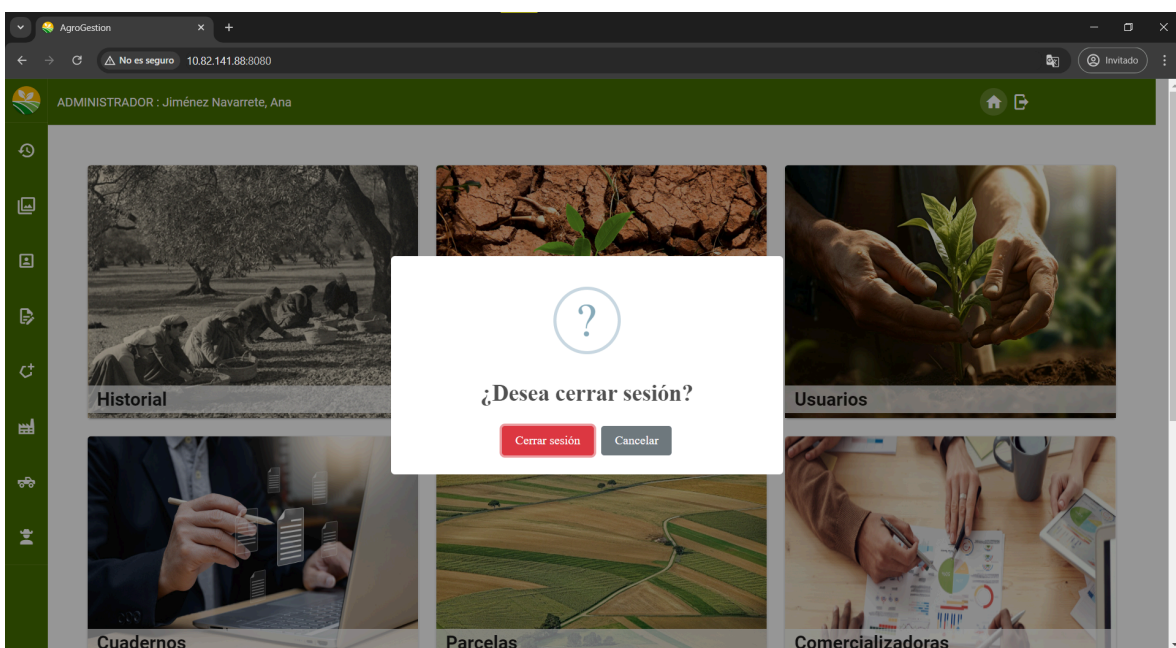


Imagen 125: Confirmación del cierre de sesión

3.5.3.4. Mantenimiento de una entidad

Cuando hablamos de mantenimiento nos referimos a las distintas operaciones aplicables sobre una entidad de la base de datos. En este apartado, vamos a definir las páginas tituladas en el menú y en la pantalla principal como: usuarios, cuadernos, parcelas, comercializadoras, equipos y personal. Todas ellas tienen la misma estructura:

- Título y descripción: al principio de la vista se anunciará la funcionalidad y una pequeña descripción de lo que se visualizará.

- Botón para ver y/o ocultar filtros: para cada entidad se definirán una serie de filtros ejecutados en tiempo real.
- Botón para insertar nuevo registro: se visualizará una ventana emergente superpuesta sobre la vista actual para completar el formulario de la entidad en cuestión.
- Caja de agrupación: las cabezas que se arrastren serán agrupadas en la tabla en tiempo real.
- Tabla de contenido: se listan los registros almacenados en la base de datos global. Dependiendo del tipo de entidad se permitirá ver, editar y/o eliminar cada registro.

Es aquí dónde utilizamos los componentes de la librería Devextreme. En nuestro caso vamos a encapsular la tabla junto a sus operaciones en una plantilla llamada `TablaTemplate`.

En primer lugar, aparecerá el contenedor de filtros, implementados con el componente `DxTextBox`, cuya finalidad es recoger las entradas de texto que el usuario introduce. Para hacer el filtrado reactivo, es decir, que su efecto se realice en tiempo real, vamos a utilizar la función `filter(.)` de `DxDataGrid`. Cada vez que se introduzca o borre un carácter en la caja se utiliza dicha función, cuyos parámetros son tres elementos: el identificador del campo, la operación de comparación y el valor introducido.

Inmediatamente después, se reserva un contenedor dónde se podrá arrastrar las cabezas de las tablas. Esta agrupación se ve reflejada en las filas con fondo grisáceo formadas en la tabla. En la Imagen [126](#) apreciamos la acción del filtrado y la agrupación de contenidos.

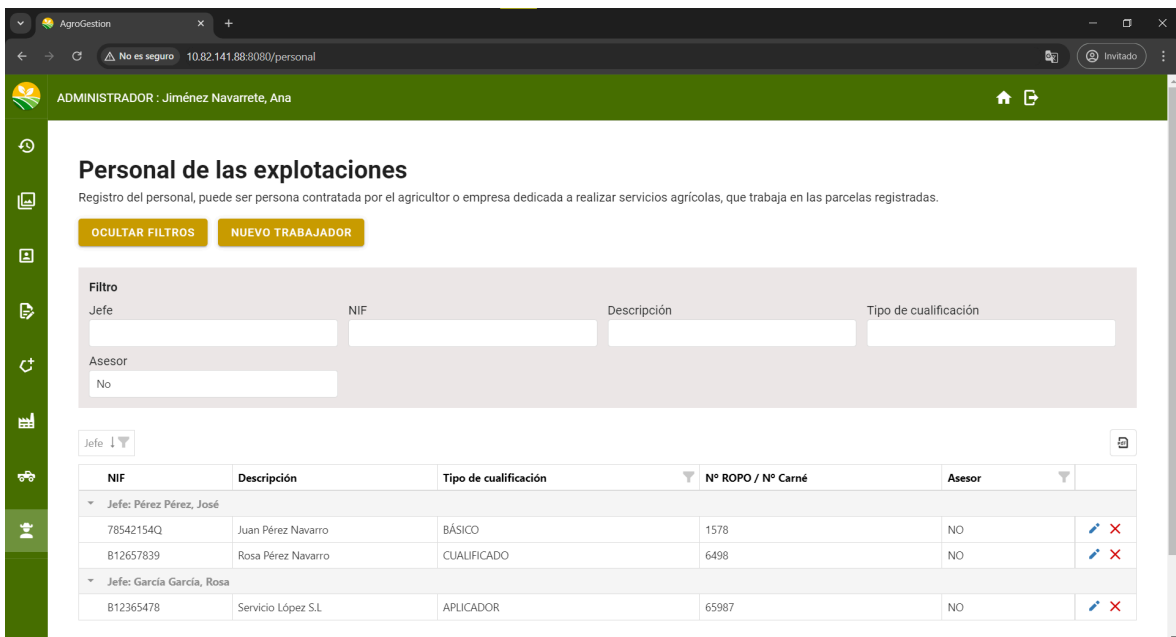


Imagen 126: Agrupación y filtrado de cabeceras

Por último, se define el componente `DxDataGrid`. Es un potente componente con muchas propiedades disponibles, pero nosotros solo utilizamos las que se detallan en el Código 24. Destacar la variable `dataSource`, objeto de la clase `DataSource` e inicializada tras hacer la petición a la base de datos. El objeto `editing` definirá las acciones permitidas para cada una de las filas de la tabla y `row-prepared` que requiere una función, ejecutada cuando se renderiza cada fila de la tabla. Por lo demás son atributos que afectan al estilo de la tabla. En la Imagen 127 se visualiza con claridad todo lo expuesto, para la entidad 'Usuario'.

JavaScript

```
<DxDataGrid ref="tabla" :dataSource="dataSource" :showRowLines=true
  :showColumnLines=true :show-borders=true :column-auto-width=true
  :wordWrapEnabled=true width="100%" :showRowFilter=true
  :editing="{useIcons: true, allowAdding: false, allowUpdating: true,
    allowDeleting: true}" :headerFilter="{visible: true,}"
  @exporting="onExporting" @row-prepared="onRowPrepared">
  <DxGroupPanel visible=true
    empty-panel-text="Arrastre aquí las columnas que desea agrupar" />
  <DxExport enabled=true :formats={['pdf']}" />
  <slot></slot>
</DxDataGrid>
```

Código 24: Implementación de una tabla con Devextreme

Como vemos en el Código [24](#), dentro de `DxDataGrid` se utilizan los componentes `DxGroupPanel` y `DxExport`. `DxGroupPanel` consigue la posibilidad de agrupar y `DxExport` habilita un botón que permite exportar la tabla en un informe en formato PDF de la tabla.

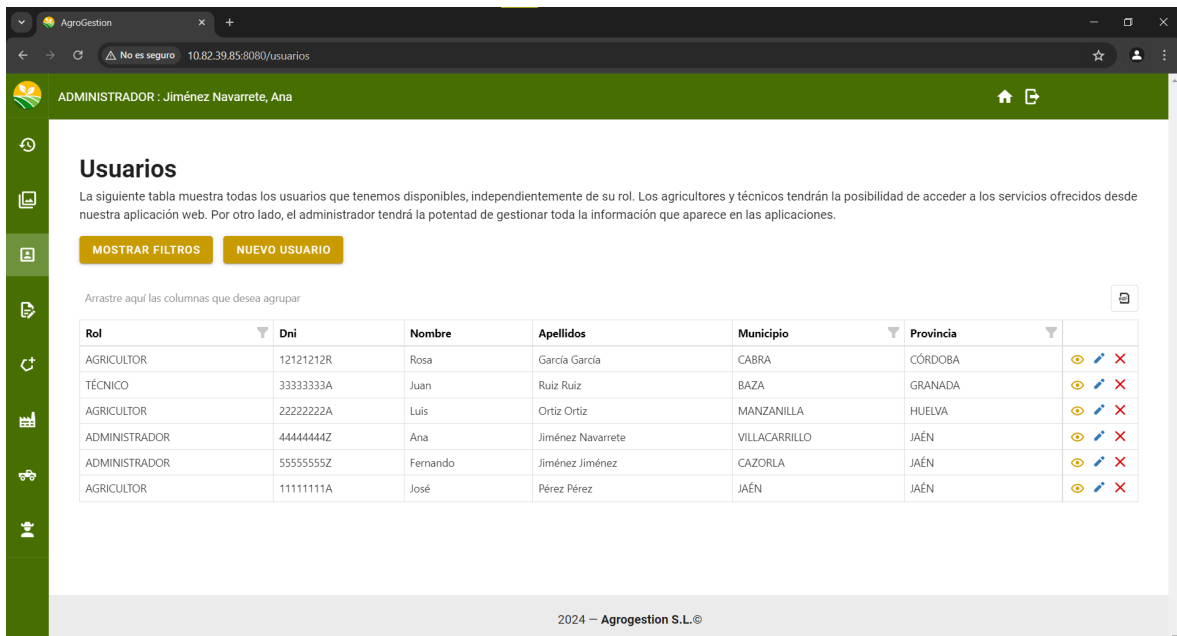


Imagen 127: Resultado del mantenimiento de la entidad 'Usuario'

Como hemos dicho al principio de este apartado, hemos creado una plantilla, `TablaTemplate`, para mantener todas las entidades uniformemente. Sin embargo, no todas las entidades tendrán las mismas columnas. Como las columnas son parte diferenciadora de cada entidad se pasarán a la plantilla en forma de `slot`. El elemento `slot` sirve como punto de distribución de contenido, esto permite a Vue componer componentes a partir de `template`. Por ejemplo, la entidad 'Usuario', se implementa en la vista llamada `UsuarioView`. Como vemos en el Código [25](#) dentro del componente `TablaTemplate` se declara un `template v-slot` que se renderiza en el elemento `<slot></slot>` de la plantilla `TablaTemplate`. El componente `template v-slot` se implementa fuera de la plantilla porque la lógica que conlleva dependerá de la entidad.

JavaScript

```
<TablaTemplate :peticion="window.$rutas.allUsuarios" :filtros="filtros"
  tituloBotonNuevo="Nuevo usuario" :clickNuevo="clickNuevo"
```

```

    ref="template" tituloPDF="Listado de usuarios">
<template v-slot>
  <DxColumn v-for="(nodo, index) in filtros" :key="index"
    :dataField="nodo.dataField" :caption="nodo.titulo"
    :allow-header-filtering="nodo.filtroCabecera"
    :allow-grouping="nodo.esFiltro" />
  <DxColumn type="buttons" :fixed="true" fixed-position="right">
    <DxButton name="view" cssClass="mdi mdi-eye-outline eye"
      :onClick="ver" />
    <DxButton name="edit" icon="edit" :onClick="edit" />
    <DxButton name="delete" icon="close" :onClick="remove" />
  </DxColumn>
</template>
</TablaTemplate>

```

Código 25: Renderizado de columnas a través de una plantilla v-slot

En la Imagen [127](#) vemos que la última columna contiene las distintas opciones aplicables sobre la fila a través de iconos. Son tres botones que ofrecen la posibilidad de ver, editar y eliminar el registro, respectivamente.

En el caso de eliminar, para confirmar la decisión del usuario se lanzará un cuadro de diálogo (ver Imagen [128](#)).

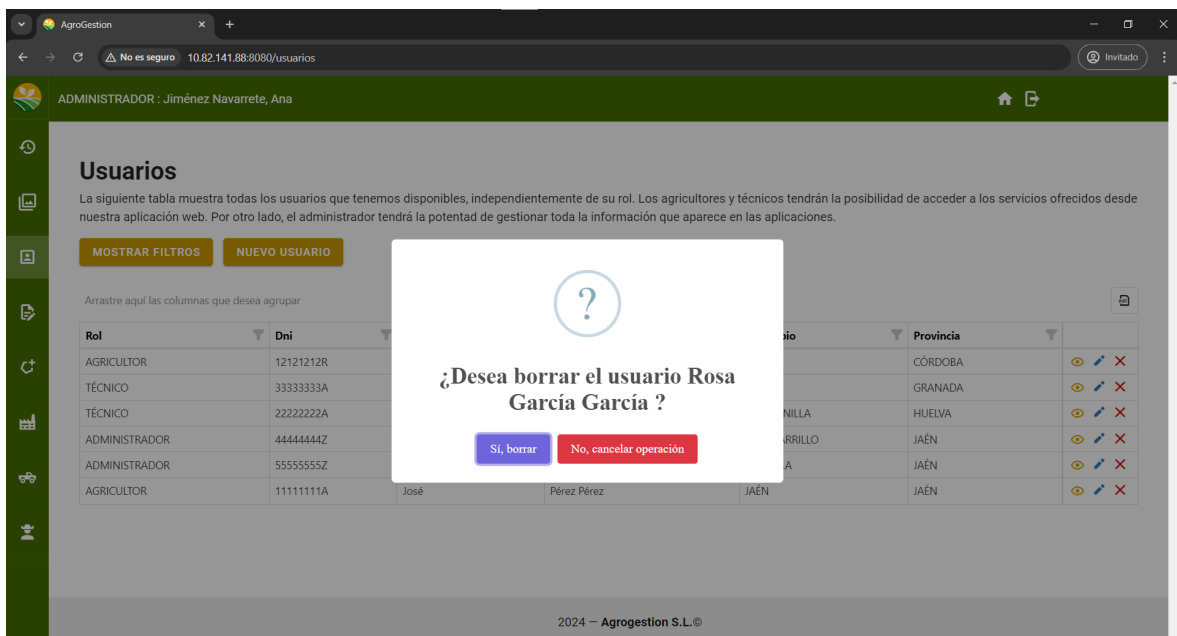


Imagen 128: Resultado de la ventana emergente de confirmación de borrado en la aplicación web

Por otro lado, para editar y ver se creará una ventana emergente personalizada. Este componente se llama `DxPopup` y su contenido será un formulario, encapsulado en la plantilla `FormTemplate`. En el Código [26](#) se describe una implementación general para dicha ventana emergente o también llamado 'popup'.

JavaScript

```
<DxPopup height="auto" max-height="90%" :visible.sync="popup"
  :drag-enabled=false :hide-on-outside-click=true :show-close-button=false
  :show-title=true :title="titulos[tipopopup]">
  <FormTemplate :columns="columnsForm" :formData="formData"
    :clickItem="clickItem" :popupVer="popupVer" ref="formulario">
    <template v-slot>
      <!-- Item dependientes -->
    </template>
  </FormTemplate>
  <div style="color:red; font-size: 16px">{{ msgError }}</div>
  <!--Botones-->
  <DxToolbarItem widget="dxButton" location="after" toolbar="bottom"
    :options="{type:'danger',text:'Cancelar',
      onClick:()=>{popup = false}}"/>
  <DxToolbarItem widget="dxButton" location="after" toolbar="bottom"
    :visible="!popupVer" :options="{ type: 'success',text: 'Confirmar',
      onClick: guardarCambios,}" />
</DxPopup>
```

Código 26: Implementación de un popup

A parte de las propiedades de los componentes involucradas en el aspecto, destacar el elemento `template v-slot` del `FormTemplate`. Ese contenido está reservado para columnas del formulario que mantiene algún tipo de relación o columnas especiales (entradas de archivos). Las demás serán entradas de texto plano, desplegables o caja de verdadero o falso, se consideran entradas normales, cuyos parámetros se definen en el objeto `columnsForm`. En el Código [27](#) se puede ver cómo se implementarán dichas entradas. Devextreme nos ofrece los componentes `DxForm`, `DxGroupItem` y `DxSimpleItem`, para crear la lógica de un formulario, ordenar sus elementos y definir las entradas de datos, respectivamente.

JavaScript

```

<DxForm ref="form" label-location="top" :form-data="formData"
  :col-count="3":show-colon-after-label=true :scrollingEnabled=true>
  <DxGroupItem v-for="nodo in columns">
    <DxSimpleItem v-if="nodo.check==true" :dataField="nodo.dataField"
      v-model="formData[`${nodo.dataField}`]"
      :label="{ text: nodo.label }" editor-type="dxCheckBox"
      :editor-options="{ disabled: popupVer || nodo.disabled }" />
    <DxSimpleItem v-else :key="nodo.dataField" :dataField="nodo.dataField"
      :label="{ text: nodo.label }" :validation-rules="nodo.reglas"
      :editor-options="{ disabled: popupVer || nodo.disabled }">
      <template #default v-if="nodo.autocomplete != null">
        <AutocompletadoTemplate :disabled="popupVer"
          :formData="formData" :dataField="nodo.dataField"
          :valueExpr="nodo.autocomplete.value" :clickItem="clickItem"
          :displayExpr="nodo.autocomplete.display"
          :peticionDataSource="nodo.autocomplete.peticionDataSource" />
      </template>
    </DxSimpleItem>
  </DxGroupItem>
</slot></slot>
</DxForm>

```

Código 27: Plantilla para la creación de una ventana emergente personalizada

Uno de los componentes propios más utilizados es el editor de texto autocompletable y desplegable, `AutocompleteTemplate`, que al fin y al cabo está creado sobre `DxSelectBox` de Devextreme. Este componente será útil cuando tengamos una entrada cuya información es seleccionada entre un conjunto de datos determinado. En la Imagen [129](#) podemos apreciar la pantalla que permite crear una nueva parcela. Decir que la misma ventana emergente instanciada para una entidad es utilizada tanto para añadir, ver o editar. La diferencia recae en la propiedad `disabled` de las entradas de texto y en los datos que el formulario utiliza para construirse. Al insertar el formulario estará vacío con las entradas habilitadas, al editar el formulario estará completo con las entradas habilitadas y al ver el formulario estará completo con las entradas deshabilitadas.

ADMINISTRADOR: Jiménez Navarrete, Ana

Añadir una nueva parcela

Agricultor: *
Selecciona...
García García,Rosa
Pérez Pérez,José
Ortiz Ortiz,Luis

Hectáreas: *
Código de la parcela: *
Recinto: *
Agregado: *
Matas: *
Hectáreas cultivadas: *
Cultivo: *
Variedad: *
Sistema de riego: *
Tipo de aire: *
Tipo asesoramiento: *
Captación de agua: *
Distancia a la fuente de agua, si existe: *
Denominación: *
Zona totalmente específica: *
Zona parcialmente específica: *
Provincia: *
Municipio: *
Código del polígono: *

Cancelar Confirmar

2024 - Agrogestion S.L.®

Imagen 129: Ventana emergente para insertar una parcela

3.5.3.5. Historial o detalle de un cuaderno

Este requisito pretende reunir toda la información de un cuaderno que aparece por separado en la aplicación web, junto a las operaciones y tratamientos que el agricultor haya creado para dicho cuaderno.

Se podrá filtrar por el agricultor y el ejercicio de los cuadernos a su nombre, esto se implementa con la plantilla `AutocompleteTemplate`. Sin embargo, lo más destacable en cuanto a desarrollo es la creación de un nuevo componente llamado `InfoTemplate`. Se ha decidido crear un nuevo componente, a pesar de utilizar también `DxDataGrid`, ya que se podrá expandir o contraer en pantalla y no tendrá ni filtros ni agrupación ni operaciones por registro (ver Imagen [130](#)).

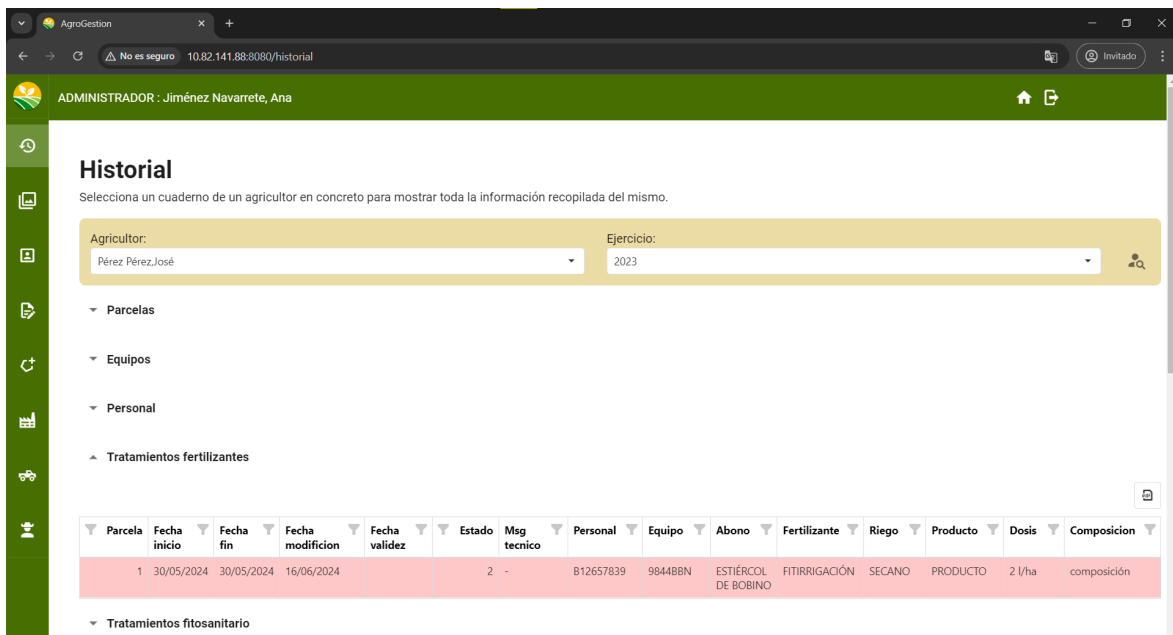


Imagen 130: Historial recopilatorio de la información de un cuaderno

3.5.3.6. Problemas en forma de imagen

Finalmente, acabamos este capítulo con la presentación de los problemas en forma de imagen. El agricultor tiene la capacidad de almacenar imágenes en la base de datos que representan algún tipo de anomalía o incidencia que encuentra en el campo. Además, el técnico tiene la responsabilidad de visualizar y evaluar las imágenes. Ambas funciones realizadas desde la aplicación móvil. Como administrador se habilitará una pantalla para visualizar todas las imágenes que haya almacenadas en la base de datos para un agricultor en concreto (ver Imagen [131](#)).

De nuevo, se implementa un pequeño contenedor de filtros para introducir el agricultor y una fecha límite de registro, con la plantilla `AutocompleteTemplate` y una entrada de tipo fecha. Al pulsar el icono de buscar, aparecerá un conjunto de tarjetas. Esta tarjeta no es la misma que la que aparece en la pantalla principal (ver Imagen [124](#)), por ello se crea un nuevo componente. `ImagenTemplate` será su nombre y consta de una imagen y varios textos. Las imágenes las vamos a mostrar con la etiqueta `v-img` de `vueify`.

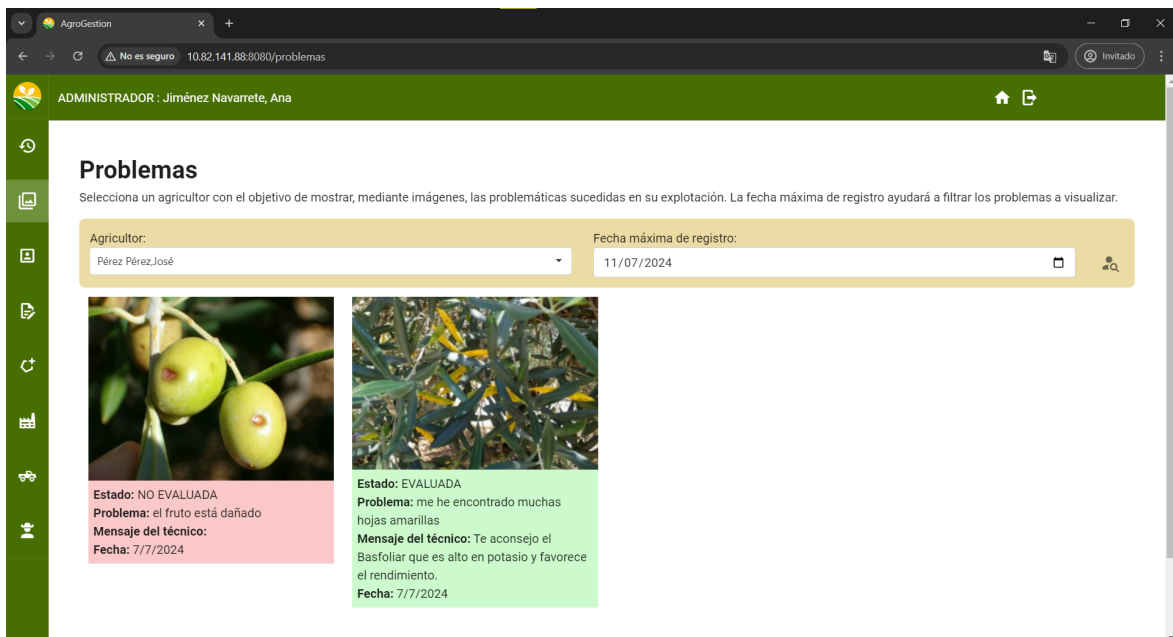


Imagen 131: Problemas representados mediante imágenes

Recordamos que la imagen se almacena en base 64⁵¹. En la entidad 'Imagen' de la base de datos tenemos tres atributos, `imagen1`, `imagen2` e `imagen3`. La longitud de una cadena de texto en Oracle es limitada, y ese límite determina la división de la cadena en tres partes. Lo único que hay que hacer para formar la imagen será concatenar dichos atributos junto al tipo de dato. En el Código [28](#) vemos de forma explícita su implementación. Se llevará a cabo en el método `created()`, ejecutado antes del renderizado del componente `ImagenTemplate`.

JavaScript

```
<template>
  <div class="tarjeta">
    <v-img :src="src" />
    <div class="tarjeta-img" :style="{ backgroundColor: colorEstado }">
      <strong>Estado:</strong> {{ tituloEstado }} </br>
      <strong>Problema:</strong> {{ imagen["problema"] }} </br>
      <strong>Mensaje del técnico:</strong> {{ imagen["msg_tecnico"] }}
      </br> <strong>Fecha:</strong>
      {{ new Date(imagen["fin"]).toLocaleDateString() }}
    </div>
  </div>
</template>
<script lang="ts">
```

⁵¹ Esquema de codificación para convertir datos binarios en un formato de texto ASCII, conjunto de letras latinas, números y dos caracteres: + y /.

```
...
created() {
  const estado = utilidades.estadoProblema[this.imagen["estado"]] as any
  this.tituloEstado = estado.nombre;
  this.colorEstado = estado.color

  let img = new Image();
  img.src = `data:image/jpeg;base64,${this.imagen["imagen1"]}
            ${this.imagen["imagen2"]} ${this.imagen["imagen3"]}`;
  this.src = img as any;
},
...
</script>
```

Código 28: Conversión y visualización de una imagen en base64

3.5.4. Pruebas

Para terminar este capítulo sobre el desarrollo de la aplicación web vamos a comentar algunos aspectos que no contemplamos en la lista de requisitos inicial, pero consideramos necesarios.

En rasgos generales en el mantenimiento de todas las entidades se sigue el mismo patrón: enunciado, filtros, agrupación y tabla. Sin embargo, hemos notado que para la entidad 'Cuaderno' debíamos añadir nuevas funcionalidades. Por lógica, un cuaderno de campo tiene un periodo de duración que se llama ejercicio y no sería correcto que un cuaderno pueda ser modificado o eliminado transcurrido dicho ejercicio. Además, los campos que definen el año del ejercicio y la fecha de apertura van a tener una validación más. Antes de guardar o editar cualquier cuaderno se comprobará que tanto el ejercicio como la fecha de apertura no sean anteriores al actual. Ya que no tiene sentido crear un cuaderno con un ejercicio o fecha pasada.

Por otro lado, queremos dar la posibilidad de dar por terminado un cuaderno, pero sin borrarlo. Un cuaderno cuyo ejercicio ha sido cumplido carece de asesoramiento y seguimiento por parte del técnico. Debido a ello consideramos necesario añadir un nuevo atributo a la entidad 'Cuaderno' (Tabla [7](#)) llamado *activo*, cuyos valores serán cero o uno. En base a este nuevo atributo vamos a

pintar cada registro de la tabla verde si está activo y rojo de lo contrario. Es más explicativo la Imagen [132](#) que muestra el resultado.

Código	Ejercicio	Fecha de apertura	Registro nacional de la explotación	Registro autonómico de la explotación	Titular de la explotación	Agricultor de la explotación	Técnico encargado de la explotación	Tipo de asesoramiento	Activo
1026	2024	20/03/2024			Ortiz Ortiz, Luis	Ortiz Ortiz, Luis	Ruiz Ruiz, Juan	AGRICULTURA ECOLÓGICA	SI
6494	2024	06/03/2024			Pérez Pérez, José	Pérez Pérez, José	Ruiz Ruiz, Juan	PRODUCCIÓN INTEGRADA	SI
2080	2024	15/06/2024			García García, Rosa	García García, Rosa	Ruiz Ruiz, Juan		NO
1025	2023	06/03/2023			Pérez Pérez, José	Pérez Pérez, José	Ruiz Ruiz, Juan		SI
2011	2022	02/04/2024	656546546	98998979	Pérez Pérez, José	Ortiz Ortiz, Luis	Ruiz Ruiz, Juan		NO
2014	2022	29/06/2024			Ortiz Ortiz, Luis	Pérez Pérez, José	Ruiz Ruiz, Juan		NO

Imagen 132: Vista del mantenimiento de la entidad 'Cuaderno'

En términos de implementación solamente tendremos que sobrecargar la función `row-prepared` del componente `DxDataGrid` (ver Código [24](#)), encargado de crear la tabla. La función es ejecutada por el componente antes de renderizar en pantalla cada fila. Por ello, para cada fila comprobamos el valor del atributo activo y, en consecuencia, añadimos un estilo u otro. En el Código [29](#) mostramos el contenido de la función sobrecargada.

```
JavaScript
onRowPrepared(e: any) {
  if (e.rowType === "data")
    if (e.data.activo == "SI")
      e.rowElement.classList.add("validado");
    else
      e.rowElement.classList.add("novalido");
},
```

Código 29: Coloreado fondo de un registro de un `DxDataGrid`

De igual forma, hacemos uso de la propiedad `row-prepared` en la tabla que muestra la información de un cuaderno. Mantenemos el mismo código de colores y pintamos un tratamiento verde, amarillo o rojo según su estado válido,

modificado o inválido. En la Imagen [130](#) se ve un ejemplo de ello y en el Apartado [3.4.3.8](#) se recuerda el significado del código de colores.

En resumen, concluida esta última iteración queda finalizada la aplicación web. La aplicación web permite visualizar y gestionar la información de la base de datos global. Gracias a ello, la aplicación móvil es alimentada con datos reales y queda disponible a agricultores y técnicos. De forma implícita, se completa el proyecto cumpliendo al mismo tiempo los requisitos iniciales.

CAPÍTULO 4

CONCLUSIONES

4.1. Introducción

Es momento de ser críticos y en este capítulo comentaremos nuestras conclusiones acerca del trabajo realizado y también hablaremos de algunas posibles mejoras que podríamos añadir en un futuro.

4.2. Trabajos futuros

Todo es mejorable y este Trabajo Fin de Grado no va a ser menos. A pesar de los objetivos conseguidos a lo largo del proyecto nos hemos dado cuenta que hay aspectos mejorables y/o funcionalidades para añadir. A continuación se enuncian algunos de las líneas de progreso abiertas referentes a la aplicación móvil:

- La interfaz de usuario se ha intentado hacer lo más intuitiva posible ya que conocemos el tipo de usuario final de nuestras aplicaciones. Sin embargo, consideramos una mejora ofrecer una guía de uso para cada funcionalidad. En un dispositivo móvil el espacio es limitado y, en ocasiones, una o dos palabras no son suficientes para entender la información a introducir en un determinado campo. Por eso, sería posible diseñar una guía detallada para cada pantalla de la aplicación y que sea accesible desde el menú de herramientas superior.
- Como sabemos, para cada parcela es posible visualizar su información geográfica mediante la ortofoto y sobre ella es dibujado el polígono que la forma. Sin embargo, debemos de asumir que esta funcionalidad es potencialmente mejorable. Una gran ventaja sería añadir interacción a la geometría. Por ejemplo, la capacidad de subdividir la parcela en regiones más pequeñas, con el objetivo último de realizar un seguimiento más controlado y exacto sobre las parcelas.
- Investigación de la implicación de los ecorregímenes en el olivar y su posible adición al CUE.
- Actualmente se ofrecen tres tipos de gráficos, pero también sería beneficioso incluir otros tipos de gráficos que comparen la información de distinta forma.

- Igualmente para los informes. Existe gran variedad de información en el sistema y ello conlleva la posibilidad de combinarla y mostrar de muchas formas.
- Hemos conseguido que la aplicación móvil sea igual de funcional independientemente de la conexión a Internet, aunque, se podría hacer más reactiva. Por ejemplo, eliminar el hecho de que el usuario tenga que actualizar y descargar la información manualmente, realizando dichas acciones de forma transparente.

También, incluimos las conclusiones y mejoras apreciadas en la aplicación web:

- Con la aplicación móvil nuestro objetivo principal era mantener y gestionar la información de la base de datos. Pero, además añadimos dos funcionalidades extras que son el historial de un cuaderno concreto y la visualización de problemáticas mediante imágenes. La visualización detallada de la aplicación de tratamientos puede ser otra mejora.
- Por lógica, un cuaderno debe ser generado cada campaña, podemos pensar en agilizar este proceso mediante la generación de plantillas. Así podríamos hacer una creación masiva de cuadernos donde solamente cambie el ejercicio y la fecha de apertura.
- Actualmente, se restringe el acceso al administrador, sin embargo, podríamos ampliar el acceso a agricultores y técnicos. Se debería incluir permisos dependientes del rol, cuyos parámetros restringirían y darían acceso a ciertas partes de la aplicación.
- En este tipo de aplicación no se han hecho uso de representaciones visuales o gráficas de los datos, la librería `Devextreme` ofrece gran versatilidad en ese sentido y podría ser de gran utilidad implementar gráficos de rendimientos de cosechas, aplicaciones fitosanitarias o actividad del usuario.

Además, cómo mejora general cabe la posibilidad de aumentar la seguridad en la autenticación de ambas aplicaciones.

En conclusión, se ha pretendido cumplir todos los objetivos y requisitos planteados al principio con la mayor eficacia posible, pero, aún así el proyecto puede seguir creciendo y mejorando. Si el proyecto es implementado en un

contexto real los usuarios pueden aportar ideas que nosotros como analistas y desarrolladores no tengamos en cuenta. Esto conseguiría adaptar el sistema más aún al entorno real, orientado al campo, a los agricultores y a la calidad del producto.

CAPÍTULO 5

ANEXOS

5.1. Introducción

De forma adicional, añadimos este último capítulo al proyecto para desarrollar un manual para ayudar al usuario. En especial se añade un manual que describe el procedimiento para obtener la geometría de una parcela. Al final se obtendrá un fichero que el administrador deberá de adjuntar en la aplicación móvil cuando inserte o modifique una parcela.

5.2. Manual de usuario para obtener la información geométrica de una parcela en QGIS

La cuarta iteración trata sobre el desarrollo de la aplicación web (ver Apartado [3.5](#)) y entre los puntos que se discuten se habla de la posibilidad de ejecutar distintas operaciones sobre entidades de la base de datos. En concreto, este tema es discutido en el Apartado [3.5.3.4](#) que explica cómo se puede insertar, editar, ver y/o eliminar registros de una entidad. Nos centraremos en la entidad llamada 'Parcela', ya que a la hora de insertar una nueva parcela necesitamos su información geográfica, con el objetivo último de poder visualizar la ortofoto y contorno de la parcela en la aplicación móvil. A continuación, explicaremos diversas herramientas que nos permitirán obtener dicha información.

En la descripción de sus atributos, Tabla [15](#), podemos ver que existen dos atributos llamados `coordenadas` y `recorte` en la entidad, dedicados a almacenar las coordenadas UTM de la geometría y límites de la ortofoto, respectivamente. Nos referimos con geometría al conjunto de puntos que forman el polígono que define una parcela. Y, en cuanto a ortofoto, se refiere a la imagen satélite de la parcela. Como explicamos en el Apartado [3.4.3.6](#) nos beneficiamos del servicio 'WNS Ortofotos PNOA máxima actualidad' que ofrece el Instituto Geográfico Nacional. Gracias a este servicio podemos obtener una ortofoto de cada parcela registrada en las administraciones públicas.

Utilizaremos la herramienta **QGIS**. QGIS es un Sistema de Información Geográfica (SIG) de software libre y de código abierto, opera bajo la licencia GNU GPL, compatible con plataformas GNU/Linux, Mac OS, Microsoft Windows y Android. Algunas de sus características más destacadas son: facilidad de interconexión con muchas bases de datos geoespaciales tanto autocontenidas como en arquitectura cliente-servidor, manejo de archivos vectoriales y soporte de una gran número de tipos de archivos ráster [50].

Lo primero que vamos a hacer es conectar el servidor de mapas nacional a QGIS, con el objetivo de visualizar una imagen satélite de todo el territorio español. Nos vamos al menú superior y hacemos click sobre la opción ‘Capas’ y ‘Administrador de fuentes de datos’. En la Imagen 133 podemos ver la ventana que aparece.

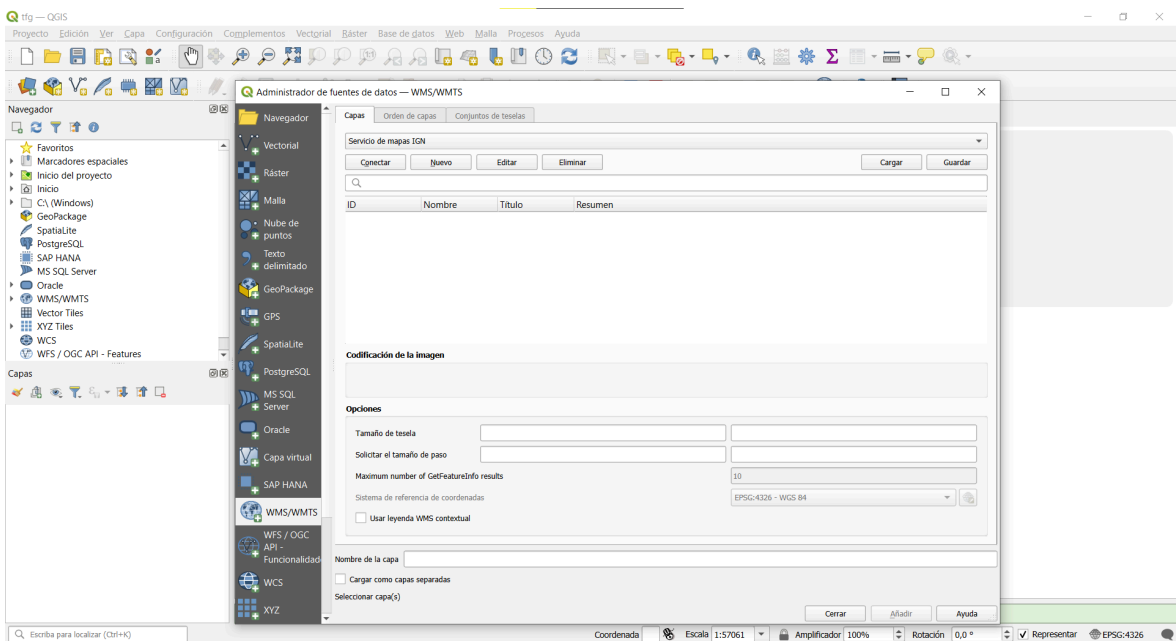


Imagen 133: Nuevo proyecto y administrador de fuentes en QGIS

A continuación, pulsamos el botón ‘Nuevo’ e introducimos los parámetros de conexión al servicio: un nombre para identificar la fuente de datos y la URL del servidor⁵². Después, pulsamos el botón ‘Aceptar’ para guardar los cambios. Una vez completado aparecerá la conexión en el administrador, vemos cómo el nombre de la conexión realizada está disponible para ser seleccionada. Lo siguiente será pulsar el botón ‘Conectar’, seleccionar la opción cuyo identificador

⁵² <http://www.ign.es/wms-inspire/pnoa-ma?Request=GetCapabilities&Service=WMS>

es el tres (Ortoimagen), elegir el sistema de referencia de coordenadas adecuado (UTM zona 30) y pulsar el botón ‘Añadir’. En la Imagen [134](#) observamos cual es el resultado del proceso.

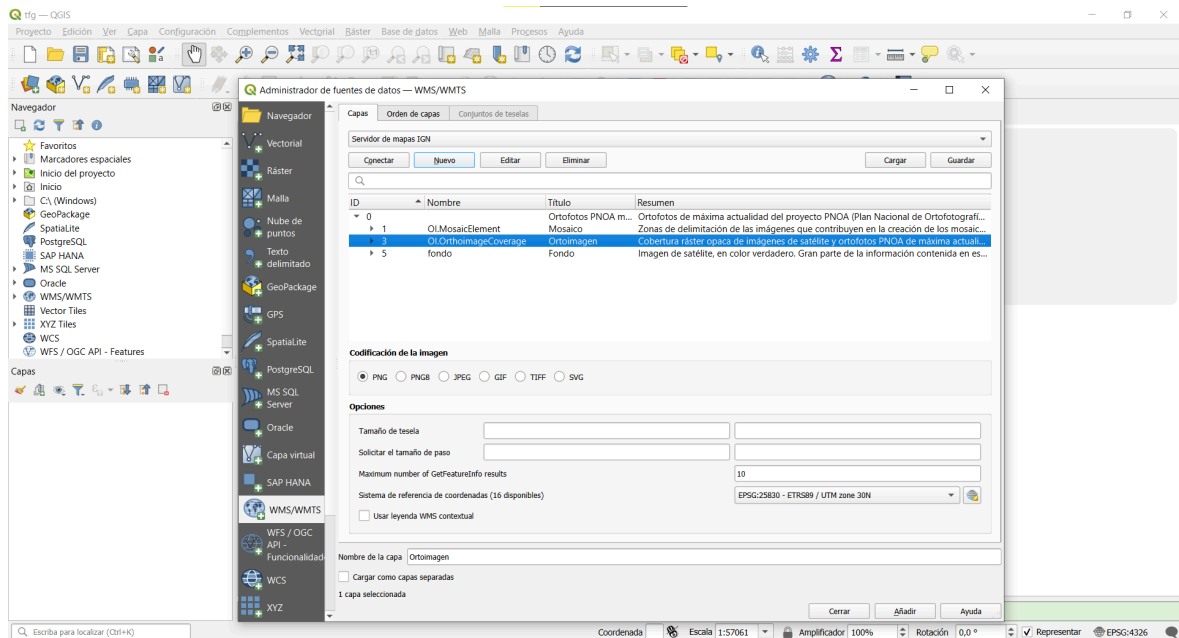


Imagen 134: Conectar nueva conexión en QGIS

Si todo ha salido correctamente debe de aparecer en el lienzo una imagen satélite del territorio español. Ahora ya estamos en disposición de añadir las geometrías de las parcelas. Desde la Junta de Andalucía se pone a disposición de la ciudadanía la información geográfica de SIGPAC⁵³, agrupadas por municipios y provincias. En nuestro caso nos interesa descargar los datos agrupados por municipios. El archivo `zip` descargado contiene varios tipos de archivos por municipio, pero a nosotros nos interesa el formato `shapefile`. Cada uno de ellos está nombrado con un código numérico, el código de la provincia seguido del código del municipio, ambos provenientes del catastro.

Desde el QGIS se tendrá que añadir una capa por cada municipio, para ello desde el menú superior seleccionamos ‘Capa’, ‘Añadir Capa’ y ‘Añadir capa vectorial’. Esto nos lleva a una ventana emergente que nos pide adjuntar la dirección del archivo, cuya extensión es `.shp` (ver Imagen [135](#)). Al pulsar el botón ‘Aceptar’ vemos que la capa es añadida en el listado izquierdo.

⁵³ <https://www.juntadeandalucia.es/organismos/agriculturapescaaguaydesarrollorural/servicios/sigpac/visor/paginas/sigpac-descarga-informacion-geografica-shapes-provincias.html>

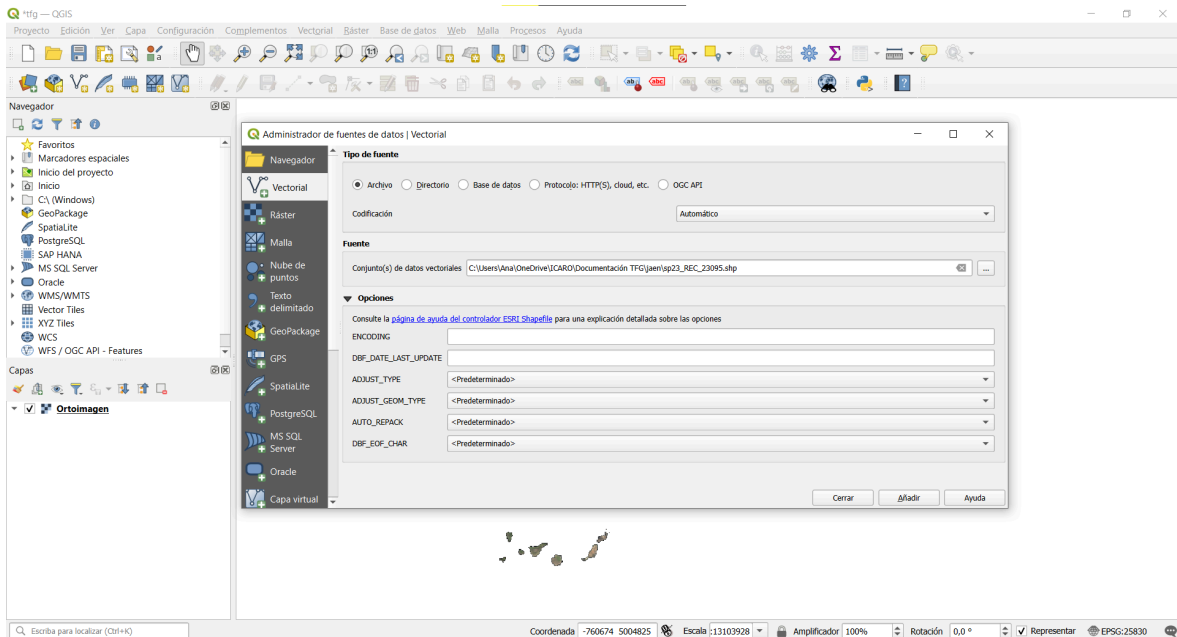


Imagen 135: Añadir nueva capa en QGIS

Después de insertar las capas deseadas podemos hacer zoom sobre el municipio y ver la limitación de cada parcela registrada. Por ejemplo, en la Imagen [136](#) se observa todo el terreno de Villacarrillo, municipio de Jaén, aunque hay más capas cargadas en el proyecto.

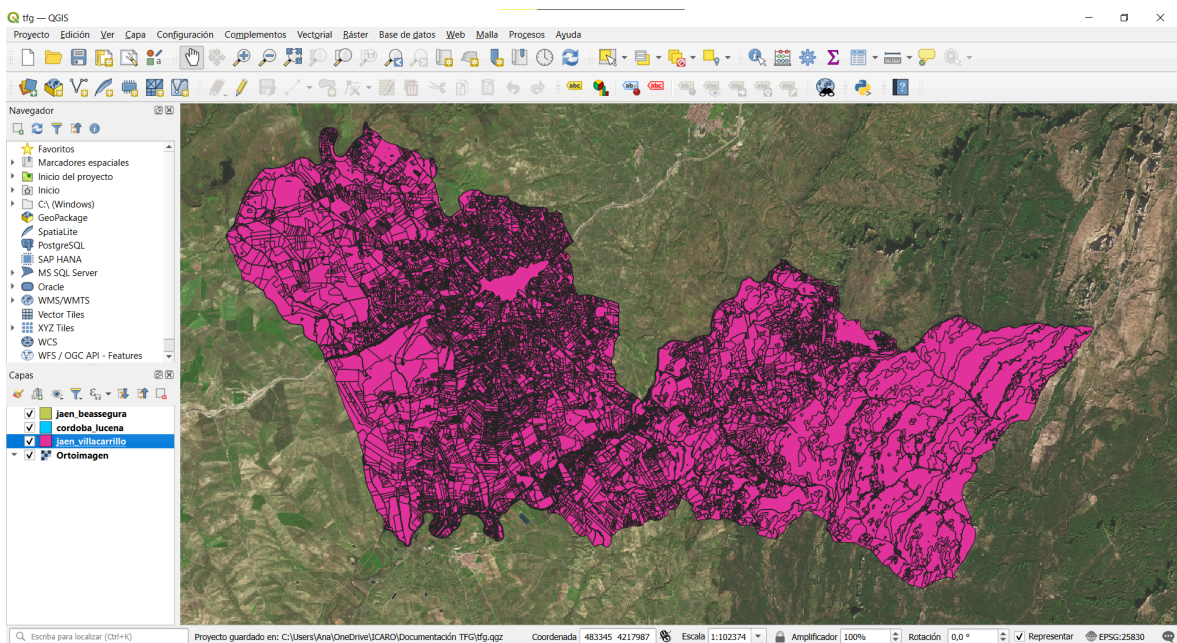


Imagen 136: Parcelas del municipio de Villacarrillo en QGIS

Ya estamos en disposición de buscar una parcela determinada. En el menú de la parte superior hay un icono nombrado ‘Seleccionar objetos por valor’. Una vez que seleccionamos la capa del municipio, pulsamos dicho botón, inmediatamente después aparecerá una ventana para introducir todos los parámetros necesarios para localizar una parcela. El último paso es pulsar el botón ‘Seleccionar objetos’ y la parcela es sombreada con un color amarillo (ver Imagen [137](#)).

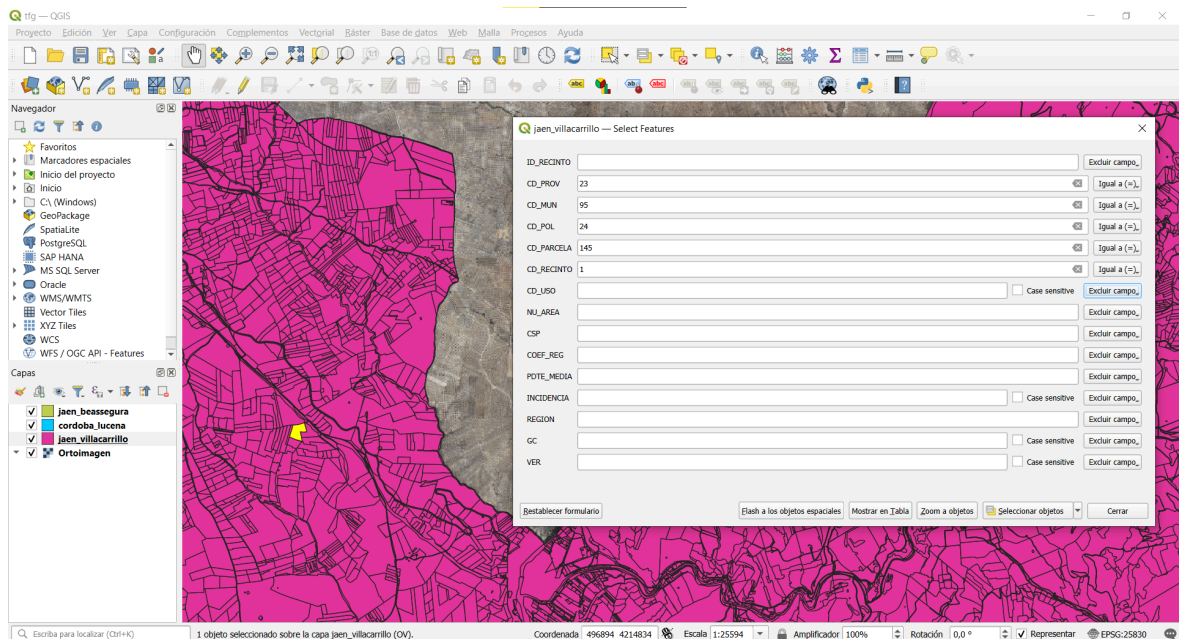


Imagen 137: Seleccionar parcela en QGIS

El siguiente paso es exportar su geometría. El sombreado amarillo significa que la parcela se ha encontrado y seleccionado en el paso anterior, por eso podemos hacer click derecho en la capa del municipio y seleccionar ‘Exportar’ y ‘Guardar objetos seleccionados como...’. Después, se abre la ventana que aparece en la Imagen [138](#), únicamente tendremos que mantener el formato del archivo a exportar (GeoJSON) y el sistema de referencia de coordenadas (UTM zona 30). Inmediatamente después se genera una nueva capa con la parcela exportada, en nuestro caso, es insignificante y podemos eliminarla ya que lo que nos interesa es el archivo generado.

Recordamos que al principio de este presente apartado nuestro objetivo era encontrar la geometría de una parcela para almacenarla en nuestra base de datos. Como ya sabemos, es el administrador el responsable de registrar toda la

información relativa a una exportación desde la aplicación web y será el encargado de adjuntar el archivo exportado en el paso anterior.

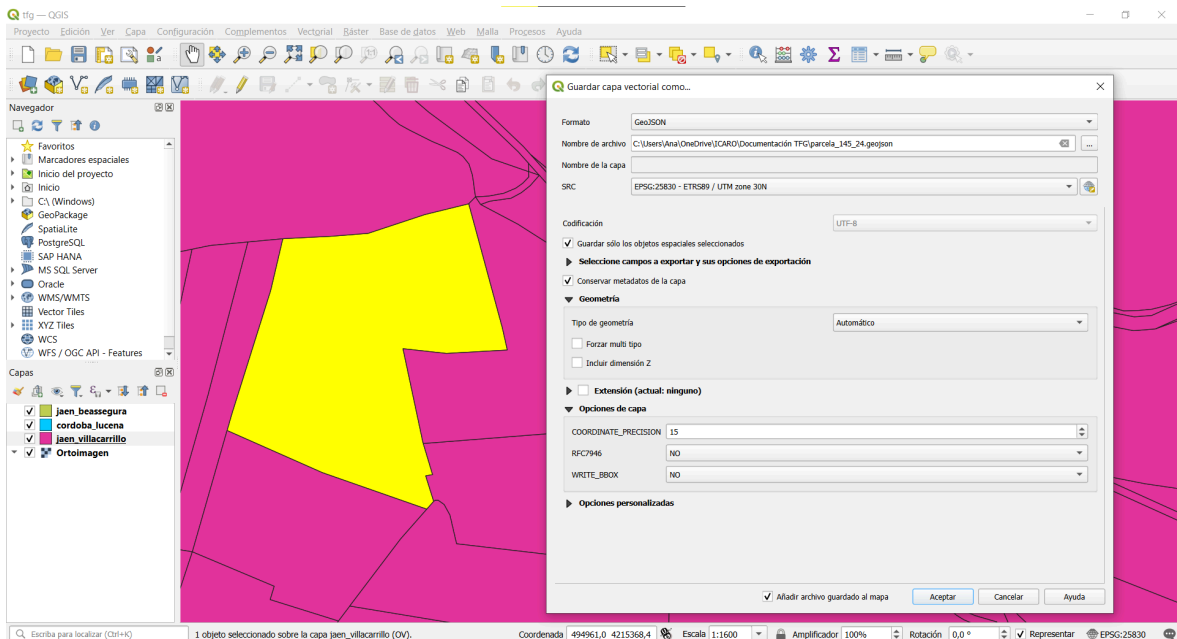


Imagen 138: Exportar geometría de una parcela en QGIS

Volvemos a recordar cuál era la pantalla de la aplicación que nos permite añadir una nueva parcela. En concreto nos interesa la entrada habilitada para subir a nuestra aplicación archivos. En la Imagen [139](#) observamos el selector de archivos.

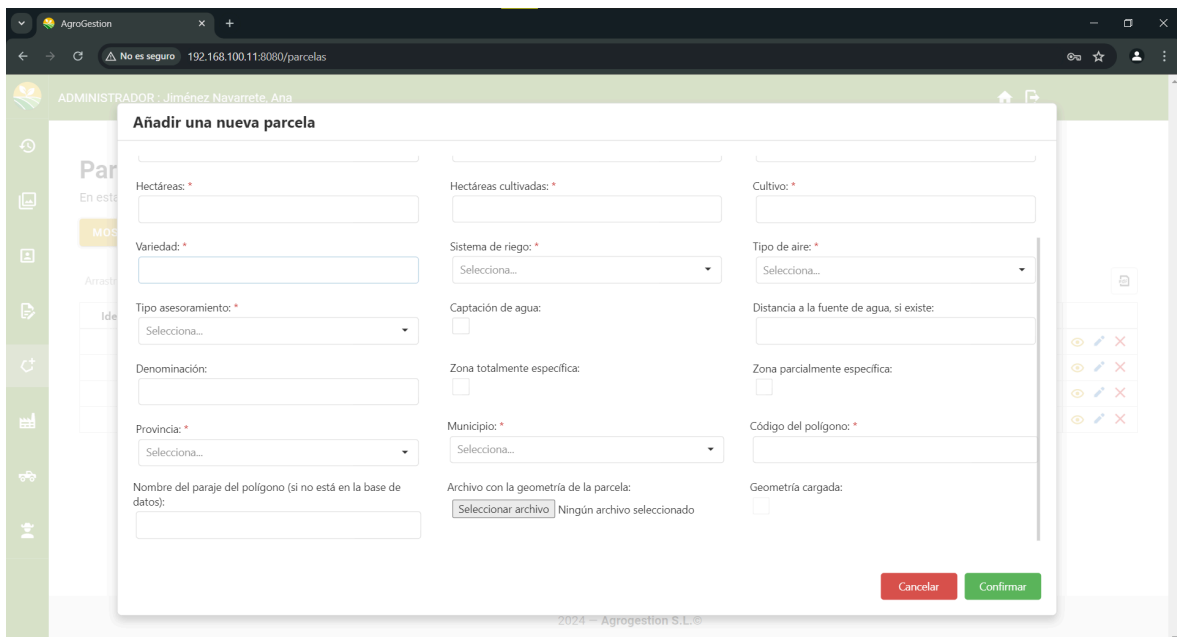


Imagen 139: Seleccionar el archivo que contiene la geometría de una parcela en la aplicación web

En términos de implementación utilizamos una etiqueta HTML, `<input>`, con el tipo de dato `file`. Lo más importante es cómo se realiza el tratamiento de los datos. La información del archivo exportado tiene la estructura JSON, conveniente para nosotros porque desde Vue es muy sencillo deserializar dicho contenido. Después recorremos el vector y obtenemos las coordenadas, al mismo tiempo que calculamos cuáles serán las máximas y mínimas que limitan la parcela. En el Código [30](#) se presenta la función que trata los datos, ejecutada cuando se confirman los cambios del formulario.

JavaScript

```
archivoSeleccionado(event: any) {
  const file = event.target.files[0];
  const reader = new FileReader();
  reader.onload = (e) => {
    this.contenidoArchivo = new String(e.target?.result?.toString())
    let coordenadas = (JSON.parse(this.contenidoArchivo as string))
      .features[0].geometry.coordinates[0][0]
    // Buscar el recuadro y transformas las coodenadas de la geometría
    var xmin = 999999999999999, xmax = -999999999999999,
      ymin = 999999999999999, ymax = -999999999999999
    this.formData.coordenadas = "";
    for (let i = 0; i < coordenadas.length; i++) {
      this.formData.coordenadas += `,${coordenadas[i][0]}`
    }
  }
}
```

```
                                `${coordenadas[i][1]}`  
    let c = { x : parseFloat(coordenadas[i][0]),  
              y : parseFloat(coordenadas[i][1]) }  
    if (c.x < xmin) xmin = c.x;  
    if (c.x > xmax) xmax = c.x;  
    if (c.y < ymin) ymin = c.y;  
    if (c.y > ymax) ymax = c.y;  
  }  
  this.formData.coordenadas = this.formData.coordenadas.substring(1);  
  this.formData.recorte = `${Math.trunc(xmin)} ${Math.trunc(ymin)}  
                           ${Math.trunc(xmax)} ${Math.trunc(ymax)}`  
};  
reader.readAsText(file);  
}
```

Código 30: Tratamiento del contenido del archivo que contiene la geometría de una parcela

Finalmente, hecho esto ya tendríamos todos los datos disponibles para mostrar la ortofoto de una parcela y dibujar su contorno en la aplicación móvil.

CAPÍTULO 6

REFERENCIAS

BIBLIOGRÁFICAS

6.1. Bibliografía

- [1] 9 Oliveres. El Acebuche, ¿qué es y cuáles son sus características? 9 Oliveres. URL: <https://9oliveres.com/blogs/noticias/el-acebuche-que-es-y-caracteristicas> (visitado 30-09-2023).
- [2] Sebastián Álamo Romero. «Efectos económicos de la aplicación de la agricultura de precisión en una explotación de olivar en la provincia de Jaén». En: (2004). Publisher: Universidad de Córdoba, Servicio de Publicaciones.
- [3] DOCUMENTOS ICONOGRÁFICOS. «DE LO REAL A LO IMAGINARIO: EL ACEBUCHES Y EL OLIVO EN EL MUNDO IBÉRICO».
- [4] FAOSTAT. URL: https://www.fao.org/faostat/es/#rankings/countries_by_commodity (visitado 05-10-2023).
- [5] Mercedes Gordo Márquez, Jesús Felicidades García y José Menor Toribio. «44. Temporeros extranjeros en la campaña de la aceituna de Jaén: el actual contexto de crisis y sus repercusiones en la red de albergues1-2».
- [6] Aceite de oliva. URL: <https://www.mapa.gob.es/es/agricultura/temas/producciones-agricolas/aceite-oliva-y-aceituna-mesa/aceite.aspx> (visitado 09-10-2023).
- [7] Los olivos en las heladas: preparación y daños - Oleopalma. Section: Curiosidades. 15 de ene. de 2019. URL: <https://oleopalma.com/los-olivos-en-lasheladas-preparacion-y-danos/> (visitado 30-09-2023).
- [8] BARRANCO NAVERO DIEGO, FERNANDEZ ESCOBAR RICARDO y RALLO ROMERO LUIS. El cultivo del olivo 7º ed. 7.a ed. Google-Books-ID: hWFIDgAAQBAJ. Ediciones Mundi-Prensa, 1 de ene. de 2017. 1012 págs. ISBN: 978-84- 8476-714-5.
- [9] R. Fernández Escobar, Diego Barranco y Luis Rallo. El cultivo del olivo. GoogleBooks-ID: OD6_8HdOsA8C. Mundi-Prensa Libros, 2004. 808 págs. ISBN: 978- 84-8474-128-2.

[10] Ministerio de la Presidencia, Relaciones con las Cortes y Memoria Democrática. Real Decreto 1054/2022, de 27 de diciembre, por el que se establece y regula el Sistema de información de explotaciones agrícolas y ganaderas y de la producción agraria, así como el Registro autonómico de explotaciones agrícolas y el Cuaderno digital de explotación agrícola. 29 de dic. de 2022. URL: <https://www.boe.es/eli/es/rd/2022/12/27/1054> (visitado 18-10-2023).

[11] Agricultura de precisión. IICA. 244 págs.

[12] BOE-A-2012-11605 Real Decreto 1311/2012, de 14 de septiembre, por el que se establece el marco de actuación para conseguir un uso sostenible de los productos fitosanitarios. URL: <https://www.boe.es/buscar/act.php?id=BOE-A2012-11605> (visitado 23-06-2024).

[13] Ministerio de Agricultura, Pesca y Alimentación. Orden APA/326/2007, de 9 de febrero, por la que se establecen las obligaciones de los titulares de explotaciones agrícolas y forestales en materia de registro de la información sobre el uso de productos fitosanitarios. 19 de feb. de 2007. URL: <https://www.boe.es/eli/es/o/2007/02/09/apa326> (visitado 23-06-2024).

[14] BOE-A-1996-5618 Real Decreto 261/1996, de 16 de febrero, sobre protección de las aguas contra la contaminación producida por los nitratos procedentes de fuentes agrarias. URL: <https://www.boe.es/buscar/act.php?id=BOE-A-1996-5618> (visitado 23-06-2024).

[15] DECRETO 36/2008, de 5 de febrero, por el que se designan las zonas vulnerables y se establecen medidas contra la contaminación por nitratos de origen agrario. URL: <https://www.juntadeandalucia.es/boja/2008/36/1> (visitado 23-06-2024).

[16] Ministerio de Agricultura, Pesca y Alimentación. Real Decreto 1048/2022, de 27 de diciembre, sobre la aplicación, a partir de 2023, de las intervenciones en forma de pagos directos y el establecimiento de requisitos comunes en el marco del Plan Estratégico de la Política Agrícola Común, y la regulación de la solicitud única del sistema integrado de gestión y control. 29 de dic. de 2022. URL: <https://www.boe.es/eli/es/rd/2022/12/27/1048> (visitado 23-06-2024).

[17] Ministerio de la Presidencia, Relaciones con las Cortes y Memoria Democrática. Real Decreto 1051/2022, de 27 de diciembre, por el que se establecen normas para la nutrición sostenible en los suelos agrarios. 29 de dic. de 2022. URL: <https://www.boe.es/eli/es/rd/2022/12/27/1051> (visitado 08-02-2024).

[18] Ministerio de Agricultura, Pesca y Alimentación. Orden APA/204/2023, de 28 de febrero, por la que se establece y regula el contenido mínimo del Registro autonómico de explotaciones agrícolas y del Cuaderno digital de explotación agrícola y la cronología de incorporación de fuentes de información en el Sistema de información de explotaciones agrícolas, ganaderas y de la producción agraria. 4 de mar. de 2023. URL: <https://www.boe.es/eli/es/o/2023/02/28/apa204> (visitado 10-10-2023).

[19] Jefatura del Estado. Real Decreto-ley 4/2023, de 11 de mayo, por el que se adoptan medidas urgentes en materia agraria y de aguas en respuesta a la sequía y al agravamiento de las condiciones del sector primario derivado del conflicto bélico en Ucrania y de las condiciones climatológicas, así como de promoción del uso del transporte público colectivo terrestre por parte de los jóvenes y prevención de riesgos laborales en episodios de elevadas temperaturas. 12 de mayo de 2023. URL: <https://www.boe.es/eli/es/rdl/2023/05/11/4> (visitado 18-10-2023).

[20] Pesca y Alimentación Ministerio de Agricultura. Nota aclaratoria sobre la entrada en vigor del cuaderno digital de explotación agrícola. RAIF. URL: <https://www.mapa.gob.es/es/agricultura/participacion-publica/notaexplicativacambioescuadernodtcm30-658532.pdf> (visitado 10-10-2023).

[21] Guía para la cumplimentación del cuaderno de explotación. URL: https://www.juntadeandalucia.es/export/drupaljda/Guia_para_la_cumplimentacion_del_cuaderno_de_explotacion.pdf (visitado 23-10-2023).

[22] admin@raif. Nueva versión del Cuaderno de Explotación. RAIF. 23 de oct. de 2023. URL: <https://www.juntadeandalucia.es/agriculturapescaaguaydesarrollorural/raif/nueva-VERSION-del-cuaderno-de-explotacion/> (visitado 02-12-2023).

- [23] Scrum: conceptos clave y cómo se aplica en la gestión de proyectos [2024] Asana. URL: <https://asana.com/es/resources/what-is-scrum> (visitado 23-06-2024).
- [24] Ian Sommerville. Ingeniería de software. 9ª ed. México [etc: Pearson Educación, 2011. xv+773. ISBN: 978-607-32-0603-7.
- [25] Roger S. Pressman. Ingeniería de software: un enfoque práctico. Col. de Luis Antonio Medellín Serna y Bruce R. Maxim. 9ª ed. México: McGraw-Hill, 2010. 671 págs. ISBN: 978-1-4562-8772-6.
- [26] Android. En: Wikipedia, la enciclopedia libre. Page Version ID: 160847328. URL: <https://es.wikipedia.org/w/index.php?title=Android&oldid=160847328> (visitado 27-06-2024).
- [27] iOS. En: Wikipedia, la enciclopedia libre. Page Version ID: 160932217. URL: https://es.wikipedia.org/w/index.php?title=IOS&oldid=160932217#Tecnolog%C3%83%C2%ADas_no_admitidas (visitado 27-06-2024).
- [28] Visual Studio Code. En: Wikipedia, la enciclopedia libre. Page Version ID: 160901360. URL: https://es.wikipedia.org/w/index.php?title=Visual_Studio_Code&oldid=160901360#cite_note-ars-opensource-1 (visitado 27-06-2024).
- [29] HTML. En: Wikipedia, la enciclopedia libre. Page Version ID: 160479578. URL: <https://es.wikipedia.org/w/index.php?title=HTML&oldid=160479578> (visitado 27-06-2024).
- [30] CSS. En: Wikipedia, la enciclopedia libre. Page Version ID: 159796485. URL: <https://es.wikipedia.org/w/.php?title=CSS&oldid=159796485> (visitado 27-06-2024).
- [31] Bootstrap (framework). En: Wikipedia, la enciclopedia libre. Page Version ID: 160017323. URL: [https://es.wikipedia.org/w/index.php?title=Bootstrap_\(framework\)&oldid=160017323](https://es.wikipedia.org/w/index.php?title=Bootstrap_(framework)&oldid=160017323) (visitado 27-06-2024).
- [32] JavaScript. En: Wikipedia, la enciclopedia libre. Page Version ID: 160780005. URL: <https://es.wikipedia.org/w/index.php?title=JavaScript&oldid=160780005> (visitado 27-06-2024).

- [33] Vue.js. En: Wikipedia, la enciclopedia libre. Page Version ID: 157941652. URL: <https://es.wikipedia.org/w/index.php?title=Vue.js&oldid=157941652> (visitado 27-06-2024).
- [34] Node.js. En: Wikipedia, la enciclopedia libre. Page Version ID: 159853075. URL: <https://es.wikipedia.org/w/index.php?title=Node.js&oldid=159853075> (visitado 28-06-2024).
- [35] Oracle Database. En: Wikipedia, la enciclopedia libre. Page Version ID: 155456661. URL: https://es.wikipedia.org/w/index.php?title=Oracle_Database&oldid=155456661 (visitado 27-06-2024).
- [36] PostgreSQL: About. URL: <https://www.postgresql.org/about/> (visitado 29-06-2024).
- [37] Fray León Osorio Rivera. Base de datos relacionales. ITM, 2008. 253 págs. ISBN: 978-958-8351-42-1.
- [38] José Manuel Navarro. «Iniciación a Oracle8». En: La Web del programador (2003).
- [39] Mercedes Marqués-Andrés y col. Bases de datos. Universitat Jaume I, 2011.
- [40] 1 Introduction to APIs. ISBN: 978-1-61729-585-0. URL: <https://learning.oreilly.com/library/view/api-design-patterns/9781617295850/Text/01.htm> (visitado 25-02-2024).
- [41] What is Postman? Postman API Platform. Postman API Platform. URL: <https://www.postman.com/product/what-is-postman/> (visitado 18-06-2024).
- [42] Sebastián Serna. Diseño de interfaces en aplicaciones móviles. Grupo Editorial RA-MA. 234 págs.
- [43] Documentation | Desarrolladores de Android | Android Developers. URL: <https://developer.android.com/docs?hl=es-419> (visitado 05-12-2023).
- [44] Laurent Debrauwer y Fien Van der Heyde. UML 2.5: iniciación, ejemplos y ejercicios corregidos. Ediciones ENI, 2016.

- [45] Singleton. URL: <https://refactoring.guru/es/design-patterns/singleton> (visitado 18-06-2024).
- [46] AsyncTask en Android - Jarroba. URL: <https://jarroba.com/asynctask-enandroid/> (visitado 14-12-2023).
- [47] Mercedes Marqués-Andrés et al. Bases de datos. Universitat Jaume I, 2011.
- [48] Instituto Geográfico Nacional, Geoportal oficial del Instituto Geográfico Nacional de España, URL: <http://www.ign.es> (visitado 7-7-2024)
- [49] Introducción - Vue.js, Vue.js - The Progressive {JavaScript} Framework URL: <https://es.vuejs.org/v2/guide/> (visitado 09/07/2024)
- [50] QGIS. En: Wikipedia, la enciclopedia libre. Page Version ID: 160956357. URL: <https://es.wikipedia.org/w/index.php?title=QGIS&oldid=160956357> (visitado 14-04-2024).