Escuela Politécnica Superior de Jaén

Grado en Ingeniería Informática

UNIVERSIDAD DE JAÉN
Escuela Politécnica de Jaén

Trabajo Fin de Grado

DISPENSADOR AUTOMÁTICO DE COMBINADOS DE BEBIDAS

Alumno: Adrián Díaz García

Tutor: Pedro J. Sánchez Sánchez
Dpto: Informática

Septiembre, 2016

Universidad de Jaén

Escuela Politécnica Superior de Jaén
Departamento de Informática

Don Pedro J. Sánchez Sánchez, tutor del Proyecto Fin de Carrera titulado: Dispensador automático de combinados de bebidas, que presenta Adrián Díaz García, autoriza su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, Septiembre de 2016

El alumno: Los tutores:

Adrián Díaz García Pedro J. Sánchez Sánchez

Índice de contenido

Índice de tablas imágenes y gráficas.

CAPÍTULO 1

Introducción

1.1. Introducción al trabajo

Vivimos en un mundo donde la tecnología está a la orden del día, desde la aparición de las primeras máquinas diseñadas para ayudar al hombre en sus tareas cotidianas como pueden ser levantar un gran peso (polea y polipasto), hasta posteriormente las maquinas creadas para satisfacer la necesidad de entretenimiento, véase cajas de música, juegos etc.

El objetivo de las máquinas es realizar un trabajo o tarea de manera automática, en muchos casos sin supervisión continua de una persona, existen máquinas de todo tipo, una de las más populares son las máquinas destinadas a expender productos, estos productos pueden ser desde alimentos, máquinas de refrescos, sándwiches o pizzas, hasta dinero, como cajeros automáticos, estas máquinas son denominadas máquinas expendedoras [1].

Según varios historiadores la máquinas expendedoras tienen su origen en el antiguo Egipto, de la primera máquina de dispensación que se tiene constancia fue diseñada por Herón de Alejandría, esta se usaba para dispensar el agua bendita en los templos del alto Egipto y Tebas [2].

El máximo auge de las máquinas expendedoras se desarrolló con la revolución industrial, Londres [3] fue el lugar donde en la primera década de 1880 se empezaron a instalar las primeras máquinas modernas, cuyo objetivo era vender tarjetas postales de manera automática. Además en Estados unidos en 1888, la compañía Thomas Adams Gum, instaló máquinas expendedoras de chiles en los andenes de metro de Nueva York [4].

Sobre 1897 se añadieron figuras animadas a las máquinas expendedoras para así llamar la atención de los potenciales clientes y favorecer las ventas, en la actualidad las llamamos máquinas tragaperras.

En 1902 abre en Philadelphia se abrió el primer restaurante cuyo funcionamiento se basaba única y exclusivamente en máquinas expendedoras automáticas, este restaurante estuvo abierto de 1902 hasta el 1962. En las máquinas que se encontraban en el local se ofrecían todo tipo de artículos.

En 1920 aparecieron las primeras máquinas expendedoras que vendían y servían bebidas gaseosas en recipientes desechables, por aquel entonces las maquinas no servían bebidas frías, la comercialización de bebidas frías empezó en 1930, siendo estas refrigeradas por hielo.

En 1926 se inventó la hoy en día tan popular máquina de venta de cigarrillos.

En 1946 aparecieron las máquinas que servían café caliente, estas fueron las más populares y las que se extendieron rápidamente por todo el mundo.

En 1960 las maquinas recibieron una importante modernización, era posible pagar en ellas utilizando monedas y billetes, gracias al desarrollo de componentes electrónicas por la década de los 80, las máquinas fueron capaces de aceptar el pago mediante tarjetas de crédito y débito.

1.2. Motivación

El propósito general es desarrollar el concepto de maquina «vending/exposición» partiendo de la idea de crear una máquina capaz de elaborar un combinado de forma automática (previamente elegido por el cliente) seleccionado de una base de datos con diferentes recetas de combinados de un modo llamativo y atractivo, de tal manera que la automatización ahorre trabajo, sea un reclamo y punto de interés para clientes que quieran ver como un sistema automático es capaz de servir un combinado a la vez que se muestra un espectáculo de luces ya que si se quiere llamar la atención de la clientela hacia determinados productos, los colores juegan como aliados a nuestro favor. De forma inconsciente, centramos nuestra atención en los objetos cuyo color nos transmite alguna sensación agradable.

Gracias a las plataformas de desarrollo existentes hoy en día como Arduino [5] es relativamente fácil llevar a cabo proyectos a pequeña y mediana escala para la automatización de tareas con complejidad media/alta.

Arduino es una plataforma de electrónica abierta para la creación de prototipos basada en software y hardware flexibles y fáciles de usar. Se creó para artistas,

diseñadores, aficionados y cualquiera interesado en crear entornos u objetos interactivos.

Arduino puede tomar información del entorno a través de sus conectores de entrada de toda una gama de sensores [6] y puede afectar aquello que le rodea controlando luces, motores y otros actuadores. El microcontrolador [7] en la placa Arduino se programa mediante el lenguaje de programación Arduino (basado en Wiring [8]) y el entorno de desarrollo Arduino (basado en Processing). Los proyectos hechos con Arduino pueden ejecutarse sin necesidad de conectar a un ordenador, si bien tienen la posibilidad de hacerlo y comunicar con diferentes tipos de software (p.ej. Android [9]).

Arduino recibió una Mención Honorífica en la sección Digital Communities de la edición del 2006 del Ars Electrónica Prix [10], uno de los premios más importantes que se otorgan a proyectos relacionados con la electrónica, arte interactivo, animación por ordenador, cultura digital y música.

1.3. Objetivos

Este proyecto tiene los siguientes objetivos:

1 Desarrollar una aplicación para controlar el microcontrolador utilizando tecnología Android.
2 Gestionar una base de datos SQLite [11] donde se almacenarán diversas recetas de combinados, estadísticas y todos aquellos parámetros que serán necesarios e indispensables para el correcto funcionamiento de la máquina.
3 Diseñar un software capaz de interactuar de manera directa y física con el mundo real teniendo en cuenta todos aquellos factores físicos del ambiente y que se encuentran en el entorno de la máquina, tales como: descoordinación del motor, medición precisa de las cantidades dispensadas de cada tipo de fluido, etc.
4 El software diseñado, mostrará el catálogo de bebidas disponibles mostrando así su nivel de dulzor y alcohol de manera aproximada y de forma visual, mediante una interfaz que sea amigable con el usuario.

5 Validará si se disponen de los elementos necesarios para elaborar una bebida determinada.

6 El software proveerá una funcionalidad para ver el estado actual de la máquina (presión de botellas no alcohólicas), número de combinados dispensados y porcentaje restante de cada bebida disponible en la máquina, así como la de sustituir aquella botella que se encuentre vacía.

1.4. Resumen de la memoria

En los siguientes capítulos el lector puede encontrar la siguiente información:

- **Capítulo 2 (Tecnologías necesarias para el proyecto):**
  - Tecnologías para dispositivos móviles: Comparativa entre las actuales tecnologías móviles que ofrece el mercado.
    - Android: Sistema operativo creado por Google basado en Unix.
    - iOS: Sistema propietario creado por Apple Inc.
  - Tecnologías de sistemas empotrados: Comparativa entre las actuales tecnologías de sistemas empotrados que existen en el mercado.
    - Raspberry Pi
    - Arduino: Comparativa entre las diferentes placas que esta familia ofrece.
  - Conclusión y comentarios finales de los sistemas que finalmente han sido elegidos para la realización del proyecto.
- **Capítulo 3 (Comunicación Android y Arduino):**
  - Comparativa de alternativas disponibles para realizar conectividad entre Android y Arduino.
    - OTG (USB)
    - Wifi
    - Bluetooth
  - Desarrollo de la solución que se implementará para el desarrollo del proyecto.
  - Explicación de la estrategia y lenguaje utilizado para llevar a cabo la comunicación entre Android y Arduino, incluyendo las diferentes funciones que Arduino ofrece y como llamarlas.

- **Capítulo 4 (Análisis del proyecto):** Desarrollo de las diferentes etapas de la ingeniería del software
    - Análisis de requerimientos
        - Funcionales
        - No funcionales
    - Análisis del sistema
        - Estudios de casos de uso
        - Desarrollo de escenarios
        - Diagrama de clases
        - Estudio de los modelos de base de datos
        - Diseño de la interfaz
        - Desarrollo de los lenguajes y librerías utilizadas
            - Android (JAVA)
            - C++
        - Explicación de la estructura de ficheros y desarrollo de las funcionalidades en cada fichero.
- **Capítulo 5 (Conclusiones):** Argumentación y exposición de las conclusiones finales.
- **Capítulo 6 (Bibliografía):** Listado de referencias bibliográficas utilizadas para el desarrollo del proyecto.

CAPÍTULO 2

Tecnologías necesarias para el Proyecto

2.1. Tecnologías para dispositivos móviles

La telefonía móvil ha supuesto un gran cambio en la sociedad actual, como pudo serlo internet en su aparición. Esta revolución tiene mucho que evolucionar todavía, los nuevos dispositivos móviles son capaces de ofrecer capacidades de procesamiento y almacenamiento similares a las de un ordenador convencional, esto permite que puedan ser utilizados para muy diversos usos, entre los más populares se encuentran, leer nuestro correo, jugar a juegos, estar conectado a las redes sociales, entre otras.

En la actualidad existen diferentes alternativas en lo referente a las tecnologías móviles, cuando se menciona dicho término en este trabajo se refiere a las diferentes tecnologías que existen en el mercado para dispositivos móviles (teléfonos, móviles, relojes, tabletas, etc.).

A continuación se mencionan las tecnologías móviles más extendidas en el mercado y que son susceptibles de estudio para la realización de este proyecto.

2.1.1 Android

El lanzamiento de Android [12] como nueva plataforma para el desarrollo de aplicaciones móviles ha causado una gran expectación y está teniendo una importante aceptación, tanto por los usuarios como por la industria. En la actualidad se está convirtiendo en la alternativa estándar frente a otras plataformas como IPhone, Windows Phone o BlackBerry.

A diferencia de sus competidores, Microsoft [13], Apple [14], Blackberry [15], Firefox OS [16], Android presenta características que lo diferencian del resto:

- **Sistema abierto**: Android es una plataforma libre basada en Unix [17] siendo esta de código abierto. Una de sus principales ventajas es la posibilidad de ser personalizada sin necesidad de pagar a la empresa creadora, existen diferentes versiones y adaptaciones.
- **Fácil adaptación a diversos hardware**: Android no ha sido diseñado pensando en un hardware específico como móviles o tabletas, es posible

integrarlo en diversos dispositivos como cámaras de fotos, relojes (Smart watch), electrodomésticos inteligentes y en muchos otros dispositivos.

Esta ventaja es muy importante, aunque también puede suponer un gran esfuerzo de los desarrolladores a la hora de crear una aplicación debido a que la aplicación a de funcionar en cualquier tipo de dispositivo (diferentes tamaños de pantalla, diferente microprocesador etc.). La principal competencia de Android, Apple, sigue una estrategia diferente, Apple no diversifica su sistema operativo en diferentes plataformas hardware, Apple ofrece un número muy limitado de dispositivos en los que su plataforma funciona, haciendo más fácil la tarea de desarrollar para el programador.

Portabilidad: En Android se utiliza como lenguaje de programación base Java [18], este lenguaje garantiza que los desarrollos podrán ser ejecutados en cualquier Central processing unit (CPU) [19]. Esto es posible gracias al concepto de máquina virtual [20] que utiliza Java. La máquina virtual Java es un software intermedio que es el encargado de transformar el lenguaje Java, a lenguaje maquina (entendible por el microprocesador), este software "traductor" existen para muchos tipos de sistemas operativos y CPU, por lo que un mismo software escrito en Java puede ser ejecutado en diferentes sistemas sin necesidad de modificación del mismo.

- **Arquitectura basada en estándares**: Un claro ejemplo de la utilización de estándares puede ser el diseño de la interfaz de usuario la cual se realiza en Extensible Markup Language (XML) [21] que posibilita su correcta visualización tanto en pantallas pequeñas como en grandes televisiones, la utilización de estándares confiere al sistema alta calidad, pues estos pasan estrictas revisiones y gran flexibilidad. Los estándares son las referencias que se toman para el desarrollo de las diferentes tecnologías, esto significa que cualquier programador que conozca un estándar es capaz de trabajar con el donde quiera que sea, además si se descubre un problema en un estándar al estar extendido de manera general entre los sistemas que lo siguen, se puede solventar el problema de una manera rápida en los mismos.
- **Servicios incorporados**: Android posee una inmensa cantidad de opciones de desarrollo gracias a las características que incorpora: localización GPS

[22] y localización por redes, bases de datos con SQL (SQLite), reconocimiento de voz, opciones multimedia, diferentes sensores como: giroscopio, acelerómetro, sensor de campo magnético etc.

- **Nivel de seguridad alto**: Esto es posible debido a que cada aplicación se encuentra aislada de las demás, gracias al concepto de ejecución dentro de un sandbox [23], la cual es heredada de Linux. Además, para cada aplicación existen unos permisos de manera que es posible conseguir una configuración con un detalle muy fino, esto significa que se puede establecer de una manera muy precisa aquello que puede llevar a cabo una aplicación y aquello que no.
- **Sistema optimizado para ejecutarse con poca potencia y pocos recursos de memoria**: Android puede ser ejecutado en infinidad de dispositivos de todo tipo, por lo que debe ser capaz de ejecutarse teniendo pocos recursos disponibles. Android utiliza una máquina virtual llamada Dalvik, creada por Google, la cual es una implementación de la máquina virtual de Java, optimizada para dispositivos móviles, por lo que tiene en cuenta la escasa cantidad de recursos, potencia y duración de batería.
- **Contenido multimedia**: Con Android es posible utilizar desde gráficos vectoriales hasta gráficos 3D basados en OpenGL [24]. Además incluye codecs estándar muy comunes en audio y video, como H.264(AVC), MP3, AAC, etc.

Android Inc. en sus orígenes fue adquirida en 2005 por Google. Se trataba de una pequeña empresa dedicada a la creación de aplicaciones para terminales móviles. En ese año empezaron a trabajar en el desarrollo de una máquina virtual Java optimizada para móviles (Dalvik Virtual Machine), aunque en las nuevas versiones se está utilizando ART.

En el 2007, Handset Alliance [25] se creó con el propósito de poder establecer los estándares abiertos para los móviles que conocemos hoy en día. Algunas empresas como son las bien conocidas, Texas Instruments, Motorola, Google, Intel, T-Mobile, Ericsson, Samsung, Toshiba, y varias más fueron las fundadoras de esta alianza, entre uno de los objetivos principales de esta alianza era promover y difundir el diseño de Android. Sus integrantes se comprometieron a publicar una

gran parte de la propiedad intelectual en forma de código abierto y que seguía una licencia Apache v2.0 [26].

La primera versión del Software development kit (SDK) [27] de Android se lanzó a mediados de noviembre del año 2007. En 2008 se presentó el primer teléfono basado en plataforma Android fue llamado TMobile G1. En octubre de 2008 Google lanzó su código de fuente siguiendo una licencia de código abierto Apache V2.0. Durante el año 2010 Android se consolidó como uno de los principales sistemas operativos para teléfonos, con unas cifras muy similares a con las que contaba el iPhone de Apple e incluso las superándolas en Estados Unidos.

En 2011 se lanzaron versiones específicas para tabletas, estas versiones estaban optimizadas para el hardware, pantallas y sensores de las tabletas del momento las versiones específicas para tabletas eran 3.0, 3.1 y 3.2. Durante este año Android se convirtió en la plataforma para móviles más importante alcanzando una cuota de mercado por encima del 50%.

En el año 2012 Google lanzó una tienda de descargas de aplicaciones que llamaron GooglePlay Store, era y es en la actualidad un portal en donde se unen las descargas de aplicaciones y contenidos (libros, películas etc.). Además en 2012 aparecieron las versiones 4.1 y 4.2 con el SDK correspondiente. La plataforma siguió creciendo hasta llegar a una cuota del 75% a finales del 2012.

En 2013 Google lanzó las versiones 4.3 y 4.4 de Android, además alcanzó un 92% en ventas de nuevos teléfonos inteligentes entre diciembre de 2012 y febrero de 2013, frente al 4.4% que consiguió iOS durante ese periodo [28].

En Junio 2014 Google lanzó su versión 5.0 (Versión Application Programming Interface (API) 21) y 5.1 (Versión API 22), comúnmente llamadas Lollipop, actualmente (11 Julio 2016) su cuota de mercado dentro del mundo Android es del 14.3% en su versión 5.0 y 20.8% en su versión 5.1 [29]. En 2014 las ventas de Smartphone con Android 5.0 fueron del 80.2% frente al 14.8% de iOS, el precio medio de un dispositivo Android en 2014 era de 254$ frente a los 657$ en las versiones iOS [30].

En Octubre de 2015 Google lanzó su versión 6.0 (Versión API 23), en esa fecha Apple utilizaba la versión de su sistema operativo iOS 9.0, que fue lanzada por Apple para competir con la versión 6.0 de Google.

Con respecto a las ventas de Smartphone a finales de 2015, Android contaba con un 65.58% de cuota de mercado, Apple con su sistema operativo iOS 9.0 contaba con un 27.24% del total, lo que supuso un gran empujón para las ventas de la compañía [31].

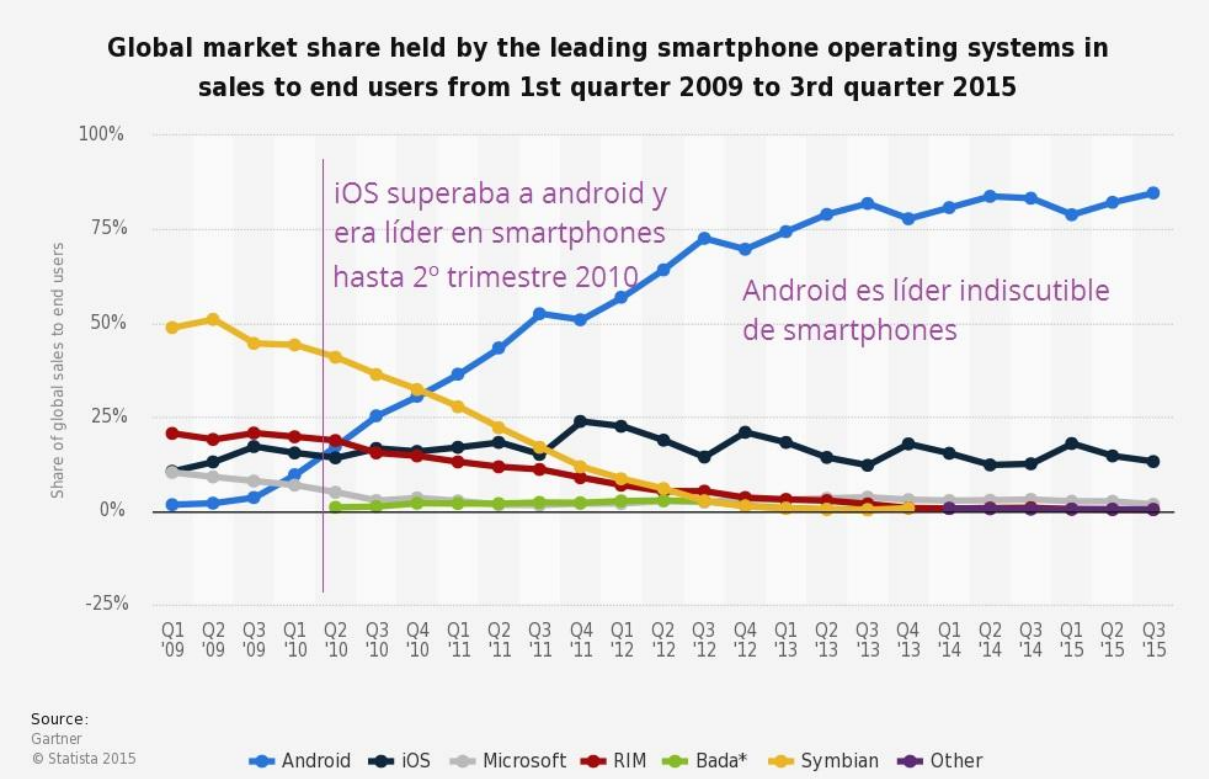
A continuación se muestra una tabla comparativa con las características principales de las principales plataformas utilizadas en 2014.

	Apple iOS 7	**Android 4.3**	**Windows Phone 8**	**BlackBerry OS 7**	**Symbian 9.5**
Compañía	Apple	Open Handset Alliance	Microsoft	RIM	Symbian Foundation
Núcleo del SO	Mac OS X	Linux	Windows NT	Mobile OS	Mobile OS
Licencia de *software*	Propietaria	Software libre y abierto	Propietaria	Propietaria	Software libre
Año de lanzamiento	2007	2008	2010	2003	1997
Fabricante único	Sí	No	No	Sí	No
Variedad de dispositivos	modelo único	muy alta	media	baja	muy alta
Soporte memoria externa	No	Sí	Sí	Sí	Sí
Motor del navegador web	WebKit	WebKit	Pocket Internet Explorer	WebKit	WebKit
Soporte Flash	No	Sí	No	Si	Sí
HTML5	Sí	Sí	Sí	Sí	No
Tienda de aplicaciones	App Store	Google Play	Windows Marketplace	BlackBerry App World	Ovi Store
Número de aplicaciones	825.000	850.000	160.000	100.000	70.000
Coste publicar	$99 / año	$25 una vez	$99 / año	sin coste	$1 una vez
Actualizaciones automáticas del S.O.	Sí	depende del fabricante	depende del fabricante	Sí	Sí
Familia CPU soportada	ARM	ARM, MIPS, Power, x86	ARM	ARM	ARM
Máquina virtual	No	Dalvik	.net	Java	No
Aplicaciones nativas	Siempre	Sí	Sí	No	Siempre
Lenguaje de programación	Objective-C, C++	Java, C++	C#, muchos	Java	C++
Plataforma de desarrollo	Mac	Windows, Mac, Linux	Windows	Windows, Mac	Windows, Mac, Linux

Tabla 2.1: Comparativa de las principales plataformas móviles

http://androideric.blogspot.de/2013/01/12-comparativa-con-otras-plataformas.html

Un aspecto fundamental a la hora de decidir que plataforma utilizar es el indicador de la cuota de mercado. En el siguiente gráfico se muestra un estudio realizado por la empresa independiente Gratner Group [32], en la que se muestra la evolución del mercado de los sistemas operativos disponibles para móviles según el número de terminales que han sido vendidos con ese sistema operativo de fábrica. Debemos destacar el descenso de ventas de Symbian [33] con Nokia (actualmente no se venden teléfonos móviles con este sistema operativo, ya que Nokia fue adquirida por Microsoft), junto con el decremento de BlackBerry, Apple por su parte tiene afianzada una cuota de mercado en torno al 15% bastante estable. Finalmente destacamos el espectacular ascenso de la plataforma Android, que le ha permitido alcanzar en dos años una cuota de mercado superior al 75%.



Gráfica 2.1: Comparativa del crecimiento de las principales plataformas móviles

https://laprestampa.wordpress.com/2016/03/05/apple-el-modelo-de-negocio-de-la-disrupcion/

En lo referente a la arquitectura de Android En las siguientes líneas se dará una visión global por capas de cuál es la arquitectura empleada en Android. Cada una de estas capas utiliza servicios ofrecidos por las anteriores, y ofrece a su vez los suyos propios a las capas de niveles superiores, tal como muestra la siguiente figura realizada por Google:

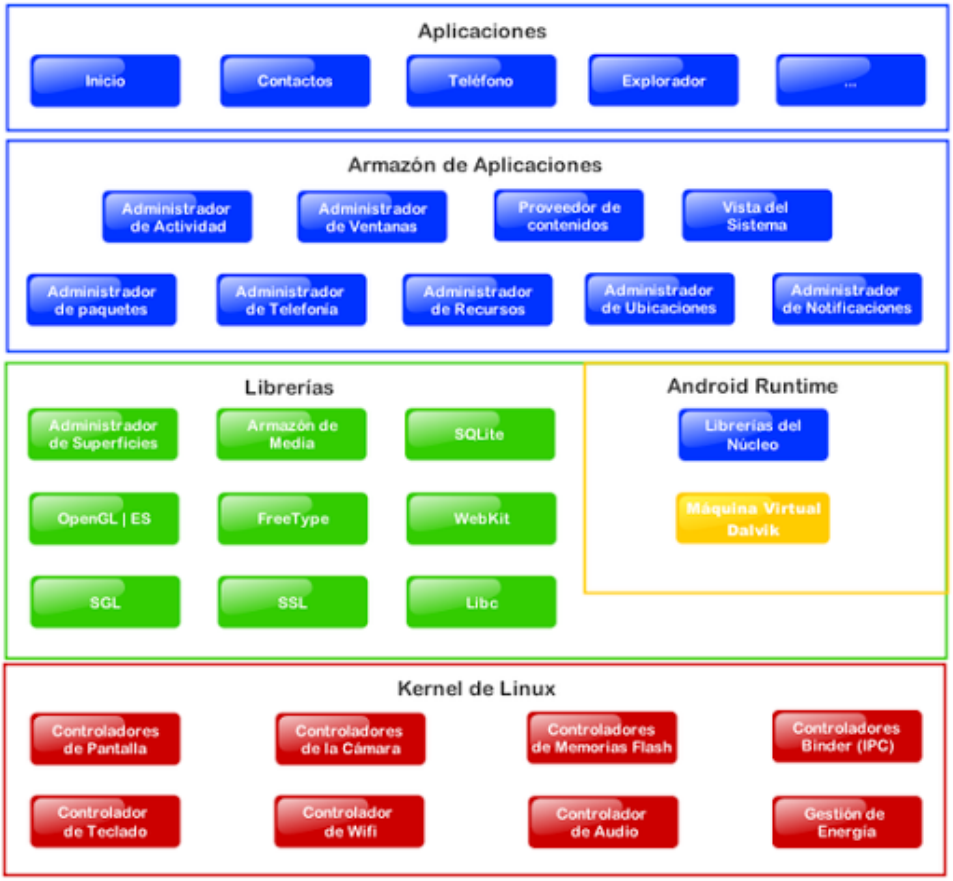


Tabla 2.2: Arquitectura del sistema operativo Android

http://androidos.readthedocs.org/en/latest/data/caracteristicas/

Cito textualmente [34]:

- **"Aplicaciones:** Este nivel contiene, tanto las incluidas por defecto de Android como aquellas que el usuario vaya añadiendo posteriormente, ya sean de terceras empresas o de su propio desarrollo. Todas estas aplicaciones utilizan los servicios, las API [35] y librerías de los niveles anteriores.
- **Framework de Aplicaciones:** Representa fundamentalmente el conjunto de herramientas de desarrollo de cualquier aplicación. Toda aplicación que se desarrolle para Android, ya sean las propias del

dispositivo, las desarrolladas por Google o terceras compañías, o incluso las que el propio usuario cree, utilizan el mismo conjunto de API y el mismo "framework" [36], representado por este nivel.

- **Entorno de Aplicaciones**: Se tratan del conjunto de herramientas de desarrollo con las que cuentan las aplicaciones. Cada aplicación que es desarrollada para la plataforma Android utiliza el mismo conjunto de APIs y el mismo framework, que es representado por este nivel."

Entre esas APIs las más destacadas debido a su frecuencia en el uso son:

Cito textualmente [37]:

- "**Activity Manager:** Es el conjunto de APIs que gestiona el ciclo de vida de las aplicaciones en Android.
- *Window Manager* : Gestiona las ventanas de aplicaciones y usa la librería Surface Manager [38].
- **Telephone Manager:** Aquí se pueden encontrar todas las APIs relacionadas con las funcionalidades propias del teléfono como llamadas, mensajes, etc.
- **Content Provider:** Permite a cualquier aplicación compartir sus propios datos con las demás aplicaciones de Android. Imprescindible ya que si no sería imposible por una aplicación, por ejemplo, ver la Agenda de contactos.
- **View System:** Contiene un gran número de elementos para construir la interfaz de usuario, como listas, mosaicos, botones, control del teclado, etc. Incluye también algunas vistas estándar para los usos más comunes.
- **Location Manager:** Da la posibilidad de utilizar los servicios de localización y posicionamiento GPS a las aplicaciones.
- **Notification Manager:** Con el que las aplicaciones pueden comunicarse con el usuario mediante eventos durante la ejecución de las mismas: una llamada entrante, un SMS, batería baja, etc. Si estas llevan asociada alguna acción, en Android denominada **Intent** ésta se activa mediante un clic.

- **XMPP Service:** Conjunto de APIs para usar el protocolo de intercambio de mensajes basado en XML."

Librerías: La capa correspondiente con las librerías [39] que son empleadas en Android es esta. Han sido creadas utilizando C/C++ [40] y proporcionan a Android la mayor parte de las funcionalidades más características. Sumadas al núcleo basado en Linux, estas librerías constituyen el corazón de Android.

Entre las librerías más importantes, están:

- **Librería libc:** Donde se encuentran las cabeceras y funciones según el estándar del lenguaje C.
- **Librería Surface Manager:** Se encarga de unir los diferentes componentes de navegación en la pantalla. Gestiona las ventanas pertenecientes a las aplicaciones activas en cada momento.
- **OpenGL/SL y SGL:** Aquí se encuentran las librerías de gráficos y, por eso mismo, son sobre las que se apoya la capacidad gráfica de Android. OpenGL/SL controla los gráficos 3D y posibilita el uso del hardware encargado de proporcionar gráficos 3D. Por otro lado tenemos SGL [41], que proporciona gráficos en 2D, por lo cual será la librería que normalmente será más utilizada por las aplicaciones. Con Android puedes crear aplicaciones que empleen ambos tipos de gráficos dando así muchas posibilidades en el desarrollo de aplicaciones gráficas.
- **Librería Media Libraries:** Todos los codecs necesarios para el contenido multimedia soportado en Android están aquí.
- **FreeType:** Permite trabajar de manera rápida y fácil con los diferentes tipos de fuentes.
- **Librería SSL:** Con esta librería se puede utilizar el protocolo para establecer conexiones seguras.
- **Librería SQLite:** La necesaria para la utilización de bases de datos relacionales en nuestras aplicaciones.
- **Librería WebKit:** El motor necesario para las aplicaciones tipo navegador y, núcleo del navegador por defecto instalado en Android." [42].

Runtime de Android: Al mismo nivel que las librerías podemos encontrar el entorno de ejecución. Aquí se encuentran las Core Libraries, que son las necesarias que contienen las clases Java y la máquina virtual Dalvik. Basado en el concepto de la máquina virtual utilizado en Java. Por las limitaciones de los dispositivos móviles (poca memoria y procesador limitado) Google tomó la decisión de crear la máquina virtual Dalvik optimizada para estos dispositivos.

Entre las características de la máquina virtual Dalvik que facilitan esta optimización de recursos se encuentra la ejecución de ficheros Dalvik ejecutables (.dex), un formato optimizado para ahorra memoria, está basada en registros. Cada aplicación corre en su propio proceso Linux con su propia instancia de la máquina virtual Dalvik. Delega al núcleo de Linux algunas funciones como threading (hilos) y el manejo de la memoria a bajo nivel.

Núcleo Linux: Android emplea la versión del núcleo de Linux 2.6 como una capa de abstracción para el hardware disponible en los diferentes dispositivos compatibles con Android. Esta capa contiene los drivers necesarios para que cualquier componente hardware pueda ser utilizado a través de las llamadas necesarias, así como la seguridad, el manejo de la memoria, el multiproceso y la pila de protocolos. Cuando un fabricante añade un nuevo elemento de hardware, lo que debe hacer es incluir los drivers o librerías de control necesarias para su correcto funcionamiento dentro del núcleo de Linux embebido en el propio Android.

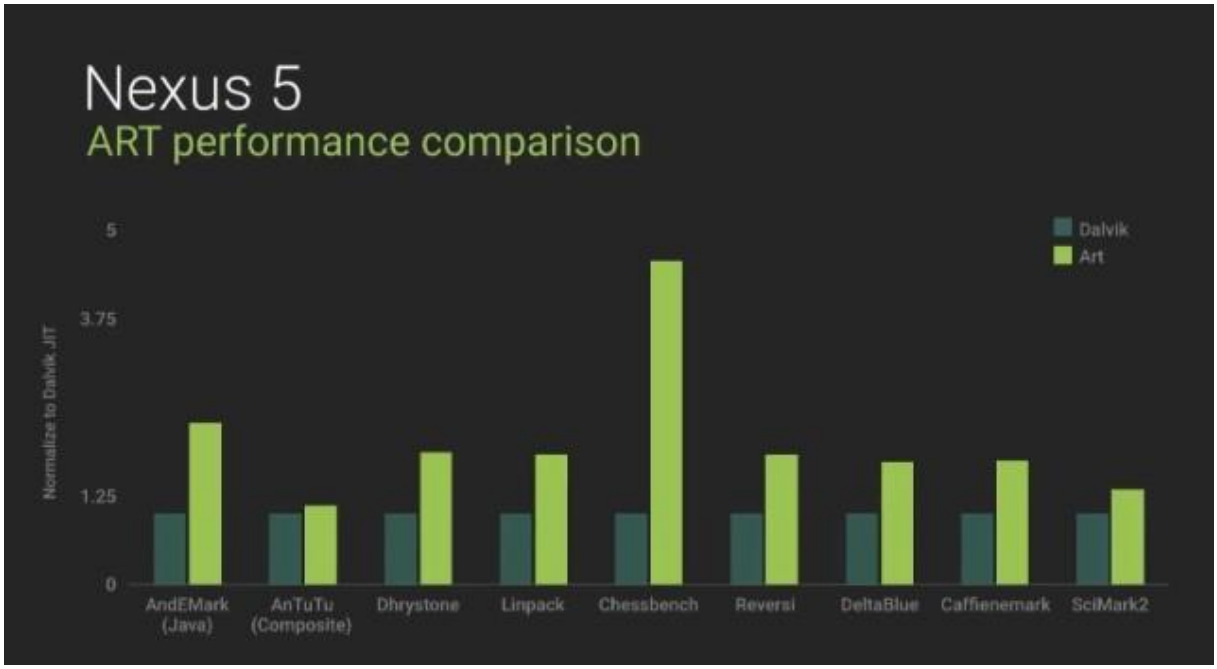
Con la nueva versión de este sistema operativo Android 5.0, entre otras novedades incorpora el nuevo runtime llamado ART, las aplicaciones Android se distribuyen en formato APK, estos apks, contienen clases java compiladas en un formato de bytecode conocido como DEX.

El formato DEX es independiente de la arquitectura de procesador, por lo que para ser ejecutado necesita traducirse a “Código máquina” [43] nativo para el procesador de nuestro dispositivo.

La diferencia fundamental entre Dalvik y ART es cuando hace esta traducción o compilación. Dalvik utiliza lo que se llama compilación Justo a tiempo (Just in time o JIT), mientras que ART utiliza una compilación previa (Ahead-of-time o AOT), por lo

que en el caso de Dalvik, cada vez que se ejecuta una aplicación esta parte se compila para que pueda ser interpretado por el dispositivo, en el caso de ART este código se compila al instalarse la aplicación, por lo que queda listo para la ejecución sin compilación a la hora de ejecutar.

En la siguiente figura se muestra un gráfico comparativo de benchmarking ejecutados en un Nexus 5 en Dalvik y ART.



Gráfica 2.2: Comparativa de rendimiento entre ART y Dalvik

http://hipertextual.com/2014/06/google-android-l

2.1.2. iOS

A continuación analizaré el mayor competidor de Android, iOS de la empresa Apple.

iOS es el sistema operativo utilizado por la compañía Apple Inc., originalmente este sistema operativo fue desarrollado para el teléfono móvil iPhone, posteriormente este sistema operativo fue migrado para ser utilizado también en los iPod [44] e iPad [45], este sistema operativo es muy restringido pues no se permite su instalación en hardware de terceros, en el último cuatrimestre de 2010 contaba con una cuota de mercado del 26%.

En 2007 Apple lanzó el SDK (Kit de desarrollo) de manera pública para desarrolladores terceros, de esta manera los desarrolladores podrían empezar a desarrollar aplicaciones para iPhone, iPod e iPad.

iOS utiliza como lenguaje Objetive-C [46] y para desarrollar aplicaciones es obligatorio utilizar Xcode, solamente compatible con ordenadores MAC.

Las características principales del sistema operativo en sí mismo son las siguientes:

- iOS se deriva del sistema operativo Mac OS X que está basado en Darwin BSD [47] que es un sistema Unix.
- Todo el sistema se encuentra bajo /root y ocupa menos de 500 MegaBytes.

La interfaz de usuario cuentas con las siguientes características:

- Intenta proveer una interfaz amigable:
  - **Manipulación directa:** utiliza gestos multitáctiles que controlan diferentes elementos; elementos deslizantes, interruptores, botones etc.
  - **La interacción:** la interacción es directamente con el sistema operativo, incluyendo todos los gestos deslizantes, pellizcos etc.
- Intenta tener un tiempo de respuesta inmediato con el usuario, las acciones se realizan cuando el usuario lo manda con sus gestos o pulsaciones.
- Da soporte a sensores internos (acelerómetros, giroscopios etc.), esto hace que la experiencia del usuario sea más realista, permitiendo que las aplicaciones modifiquen su comportamiento en base a los datos recibidos por los sensores.

Arquitectura del sistema

La arquitectura del sistema iOS es compartida con el sistema operativo MAC OS X (utilizado en los equipos portátiles y sobremesa), la cual está compuesta por una serie de “capas” o niveles de abstracción:

- **Núcleo del sistema operativo:** Esta capa es la que está basada en Darwin BSD, es la capa más baja de la pila y la más cercana al hardware, por lo que controla la interacción con el mismo.
- **Capa de servicios del núcleo:** Esta capa se encarga de las tareas básicas del sistema operativo como son: Gestión de memoria virtual, gestión de procesos e hilos de ejecución, gestión del sistema de archivos, gestión del acceso a red entre otros. Conocida también como la "capa de servicios principales", muchas aplicaciones están construidas encima de esta. Los servicios son: Servicio de base de datos SQLite, servicio de compra de aplicaciones, soporte para lenguaje XML
- **Capa media:** Conocida también como capa de "medio de comunicaciones", facilita la gestión de ficheros multimedia, contiene lo necesario para poder interactuar con gráficos, audio, video entre otros.
- **Capa "Cocoa Touch":** También conocida como la capa "táctil", esta capa define la infraestructura básica y el soporte para las tecnologías multitarea, entradas táctiles, notificaciones y servicios de alto nivel. Además en la misma se encuentran los recursos principales utilizados por las aplicaciones en sí mismas.

Los niveles más altos actúan como intermediarios entre el hardware y las aplicaciones que se ejecutan en el sistema operativo.

Una de las principales desventajas de este sistema operativo y por lo tanto de los dispositivos que la utilizan son:

- Al ser propietario, el sistema operativo puede ser instalado en dispositivos fabricados por Apple.
- El diseño es cerrado, esto significa que al estar todo centralizado en Apple el diseño de las aplicaciones es más rígido, existiendo un control más rígido con respecto a cómo han de funcionar las mismas [48].

Por todas estas desventajas en iOS y las ventajas mencionadas en la sección Android, junto con la cuota de mercado que tiene cada plataforma, me he decantado por utilizar Android como principal plataforma para desarrollo del proyecto, además

de no disponer de un dispositivo compatible que permita desarrollar y ejecutar una aplicación iOS.

2.2. Tecnologías de sistemas empotrados

Un sistema empotrado o embebido se refiere a los sistemas de computación diseñados para realizar funciones dedicadas, por lo general se tratan de sistemas de computación en tiempo real [49], estos sistemas están diseñados para cubrir unas necesidades específicas y no necesidades generales como son los ordenadores personales.

Por lo general los sistemas empotrados [50] pueden ser programados directamente en el lenguaje ensamblador del microcontrolador o utilizando compiladores específicos en lenguajes C o C++.

Una de las características más importantes de un sistema empotrado es su producción masiva, por lo tanto estos sistemas deben tener un coste muy bajo, suelen tener un procesador relativamente pequeño y una memoria pequeña (tanto Random Memory Access (RAM), como para código), estos sistemas suelen ejecutar pequeñas rutinas de código no demasiado complejas.

Las aplicaciones más comunes de un sistema empotrado son:

- Controlar el proceso de montaje o producción en una fábrica.
- Puntos de venta, como por ejemplo las cajas de los supermercados, pues incorporan lectores de códigos de barras, teclados etc.
- Sistemas de radar de aviones, normalmente se utilizan procesadores de alta potencia para calcular toda la información recibida, estos sistemas deben pesar poco y soportar condiciones extremas.

En el mercado existen diversas soluciones para realizar proyectos de la envergadura de este proyecto, estas soluciones son las siguientes:

- **Raspberry Pi:** Se trata de un ordenador de placa reducida de bajo coste desarrollado en Reino Unido. Este sistema es un ordenador completamente funcional aunque también cuenta con conexiones para

sensores etc. El sistema operativo que ejecuta está basado en UNIX, este dispositivo es ampliamente utilizado para convertir televisiones en Smart TV aunque también cuenta con servidores web integrados, sistemas NAS, servidores ftp etc.

- **Arduino:** Arduino es una plataforma de hardware libre, basada en una placa con un microcontrolador y un entorno de desarrollo, diseñada para facilitar el uso de la electrónica en proyectos multidisciplinares.

El hardware consiste en una placa con un microcontrolador Atmel AVR y puertos de entrada/salida. En este proyecto se utilizarán diferentes versiones del procesador, Atmega2560 y Atmega328.

Arduino se inició en el año 2005 como un proyecto para estudiantes en el Instituto IVREA, en Ivrea (Italia), este instituto tiene como principal foco los campos de diseño de tecnologías y más especialmente el de diseño de interacción. El diseño de interacción es un campo que estudia y define como han de comportarse los sistemas y productos con los que un usuario interactúa. Por aquellos tiempos, los estudiantes usaban el microcontrolador BASIC Stamp [51], cuyo coste era de 100 dólares estadounidenses, lo que se consideraba demasiado costoso para ellos. Por aquella época, uno de los fundadores de Arduino, Massimo Banzi, daba clases en IVREA.

Una vez concluida dicha plataforma, los investigadores trabajaron para hacerlo más ligero, más económico y disponible para la comunidad de código abierto (hardware y código abierto). El instituto finalmente cerró sus puertas, así que los investigadores, entre ellos el español David Cuartielles, promovieron la idea. Banzi afirmaría años más tarde, que el proyecto nunca surgió como una idea de negocio, sino como una necesidad de subsistir ante el inminente cierre del Instituto de diseño Interactivo IVREA. Es decir, que al crear un producto de hardware abierto, éste no podría ser embargado.

Posteriormente, Google colaboró en el desarrollo del Kit Android ADK (Accesory Development Kit), una placa Arduino capaz de comunicarse directamente con teléfonos móviles inteligentes bajo el sistema operativo Android para que el teléfono controle luces, motores y sensores conectados de Arduino de forma nativa.

Para la producción en serie de la primera versión se tomó en cuenta que el coste no fuera mayor de 30 euros, que fuera ensamblado en una placa de color azul, debía ser Plug and Play y que trabajara con todas las plataformas informáticas tales como MacOSX, Windows y GNU/Linux. Las primeras 300 unidades se las dieron a los alumnos del Instituto IVRAE, con el fin de que las probaran y empezaran a diseñar sus primeros prototipos.

2.3. Placas disponibles en el mercado basadas en Arduino

A continuación enumeraré las diferentes placas que existen en el mercado de Arduino.

Las diferencias fundamentales entre cada placa son: capacidad de memoria, velocidad de microprocesador, tamaño y número de entradas y salidas.

Existen diferentes tipos de placas, pues existen infinidad de proyectos, cada proyecto es diferente por naturaleza y tiene unos requisitos diferentes, por lo que una placa puede adaptarse mejor a los requerimientos de un proyecto u otro.

Un ejemplo para poder entender de qué estoy hablando puede ser un proyecto en el que queramos que se encienda una luz al pulsar un botón.

Para este proyecto dada la naturaleza del mismo y su simplicidad no es necesario un gran número de entradas ni salidas. Si nos hacemos una pequeña idea del código (Función encender y apagar), también se llega a la conclusión de que los requerimientos de memoria y procesamiento no son muy altos. Por todo esto tendría más sentido utilizar una placa de menor tamaño con una potencia y precio más reducida.

En cambio si el proyecto que queremos realizar es un osciloscopio necesitaremos una placa que soporte gran cantidad de entradas y salidas, así como que sea capaz de procesar gran cantidad de datos de manera rápida y eficaz.

2.3.1. Atmega2560

La placa Atmega2560 es la placa por excelencia para llevar a cabo proyectos que necesiten bastantes conexiones de entradas y salidas, como por ejemplo un osciloscopio.

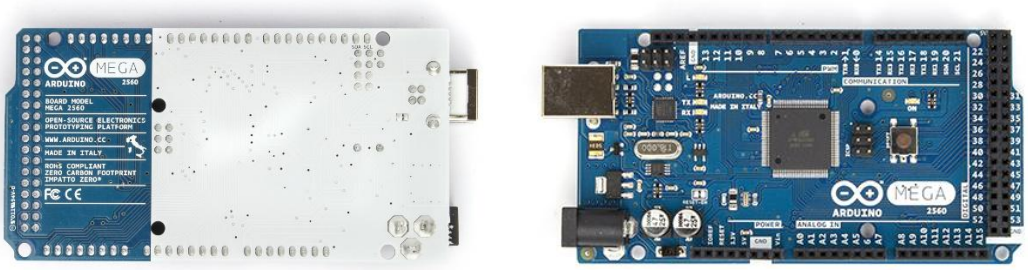


Imagen 2.1: Arduino con Atmega 2560

http://www.arduino.cc/en/Main/ArduinoBoardMega2560

Consta de 54 conexiones de entradas/salidas digitales de los cuales pueden usarse como salidas analógicas 15, también cuenta con 4 UARTs (Puertos serie, para comunicaciones), un cristal oscilador de 16MHz, una conexión USB (Puerto serie) etc.

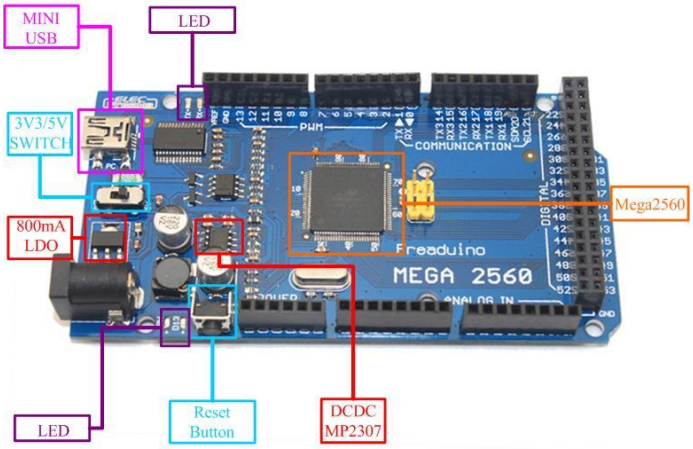


Imagen 2.2: Elementos Arduino 2560

http://kernelreloaded.blog385.com/index.php/archives/fix-arduino-mega-2560-avr-libc-issue/

Ficha técnica

Microcontrolador	ATmega2560
Voltaje de operación	5V
Voltaje de entrada (Recomendado)	7-12V
Voltaje de entrada (Limites)	6-20V
Entradas/Salidas digitales	54
Entradas analógicas	16
Corriente por pin de Entrada/Saluda	40 mA
Corriente para el pin 3.3v	50 mA
Memoria flash	256 KB de los cuales 8 KB es usara por el cargador de inicio.
Memoria RAM	8 KB
EEPROM	4 KB
Velocidad de reloj	16 MHz

2.3.2. Atmega328

Se trata del microcontrolador (pastilla con patillas), este es utilizado en varias versiones de Arduino como son Arduino Uno, Arduino nano, entre otros.

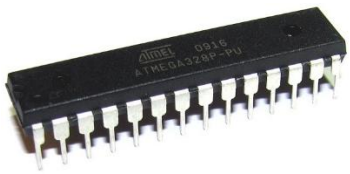


Imagen 2.3: Chip Atmega2560

http://computointegradoritchie.blogspot.de/

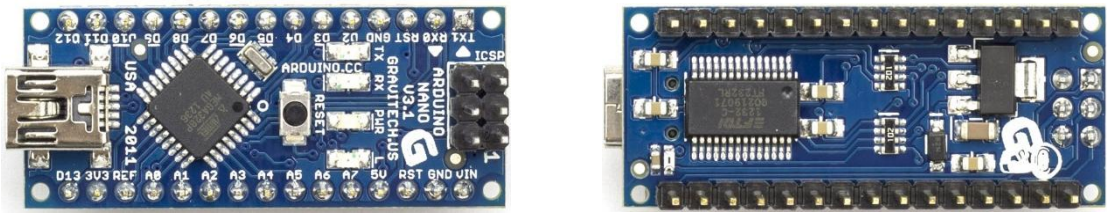


Imagen 2.4: Arduino nano

http://www.arduino.cc/en/pmwiki.php?n=Main/ArduinoBoardNano

Que cuenta con el siguiente diagrama pinout [52].

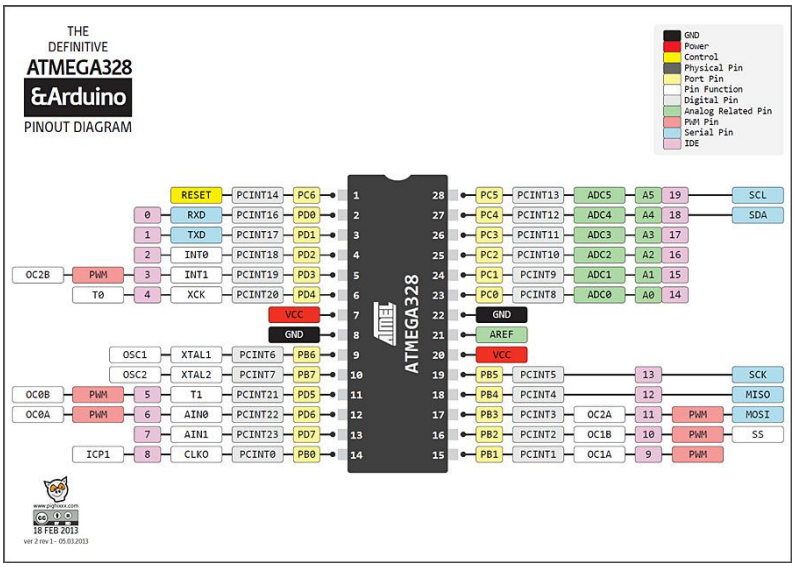


Imagen 2.5: PinOut atmega328

http://imgarcade.com/1/atmega328p-pinout/

En algunas ocasiones se puede optar por utilizar simplemente el controlador, sin la placa debido al tipo de tareas a realizar por el mismo, y diferencia en coste.

Ficha técnica

Microcontrolador	ATmega328
Voltaje de operación	5V
Voltaje de entrada (recomendado)	7-12V
Voltaje de entrada (limite)	6-20V
Pines entrada/salida Digitales	14 (De las cuales 6 pueden usarse como analógicas PWM)
Pines de entrada Analógica	6
Memoria flash	32 KB
SRAM	2 KB
EEPROM	1 KB
Velocidad de reloj	16 MHz

2.3.3. Arduino Galileo

Arduino Galileo es una placa basta en Intel Quark Soc. X1000, cuenta con un procesador de 32 bits. Este Arduino es el único desarrollado por Intel, este es el Arduino más completo, es el único que cuenta con características de un PC, un ejemplo de ellas son:

- Entrada mini-PCI Express
- Puerto Ethernet 100Mb
- Lector MicroSD

Esta placa está recomendada para proyectos que necesiten mucha computación y que puedan utilizar los 32 bits y la velocidad de reloj que ofrece su procesador.

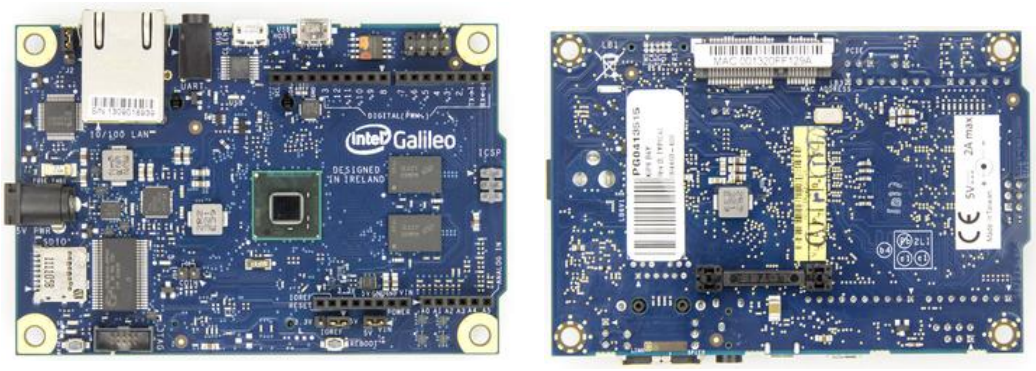


Imagen 2.6: Arduino Galileo

https://www.arduino.cc/en/ArduinoCertified/IntelGalileo

Ficha técnica

Microcontrolador	Arduino Galileo
Voltaje de operación	5V
Voltaje de entrada (recomendado)	7-12V
Voltaje de entrada (limite)	6-20V
Pines entrada/salida Digitales	14 (De las cuales 6 pueden usarse como analógicas PWM)
Pines de entrada Analógica	6
Memoria flash	32 KB
SRAM	512 KB
EEPROM	11 KB
Velocidad de reloj	400 MHz a 32 bits (16Kb de caché)

2.3.4. Arduino Uno

Arduino UNO es la placa por excelencia para empezar en el mundo de la electrónica y los microcontroladores, se trata de la placa más utilizada y documentada que existe de Arduino, aunque esto no significa que sea la que más prestaciones ofrezca.

Esta placa está recomendada para proyectos pequeños o medianos.

Imagen 2.7: Arduino Uno

https://www.arduino.cc/en/Main/ArduinoBoardUno

Ficha técnica

Microcontrolador	Arduino Uno
Voltaje de operación	5V
Voltaje de entrada (recomendado)	7-12V
Voltaje de entrada (limite)	6-20V
Pines entrada/salida Digitales	14 (De las cuales 6 pueden usarse como analógicas PWM)
Pines de entrada Analógica	6
Memoria flash	32 KB
SRAM	2 KB
EEPROM	1 KB
Velocidad de reloj	16 MHz

2.3.5. Arduino Due

Arduino Due fue la primera placa que contaba con un procesador basado en una arquitectura de 32 bits. Esta placa fue anterior a la placa Arduino Galileo, esta ha tomado el lugar de Arduino Due en la actualidad, por lo que el principal segmento de mercado que ocupada es el mismo que el que ocupa hoy en día Arduino Galileo.

Este microcontrolador se utilizaba para proyectos grandes que requirieran una gran potencia de procesamiento, que fueran capaces de aprovechar la alta capacidad de computación con la que cuenta.

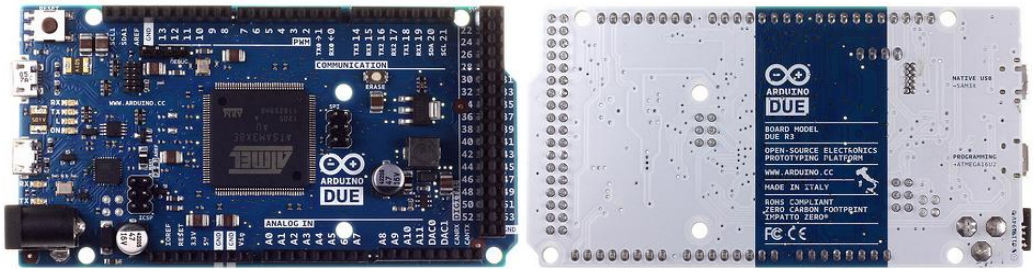


Imagen 2.8: Arduino Due

https://www.arduino.cc/en/Main/ArduinoBoardDue

Ficha técnica

Microcontrolador	Arduino Due
Voltaje de operación	5V
Voltaje de entrada (recomendado)	7-12V
Voltaje de entrada (limite)	6-20V
Pines entrada/salida Digitales	54
Pines de entrada Analógica	16
Memoria flash	512 KB
SRAM	96 KB
EEPROM	1 KB
Velocidad de reloj	84 MHz

CAPÍTULO 3

Comunicación Android y Arduino

3.1. Introducción

Llegados a este punto contamos con un dispositivo móvil, en este caso un Tablet, y una sistema empotrado, en este caso Android, necesitamos encontrar la mejor manera de interconectarlos, para eso se han analizado las siguientes alternativas.

- OTG (USB) [53]
- Wifi [54]
- Bluetooth [55]

Cada una de ellas tiene ventajas e inconvenientes, que mencionaré a continuación.

3.1.1. OTG (USB)

OTG o USB OTG (On-The-Go), es una extensión de USB 2.0 que permite a los dispositivos con puertos USB más flexibilidad en la gestión de la conexión punto a punto, lo que es lo mismo, nuestro dispositivo es capaz de acceder de manera maestra (como si de un PC se tratara) al dispositivo en cuestión, en este caso Arduino utilizando el USB integrado (Puerto serie), enviando instrucciones para que así puedan ser interpretadas por Arduino.

Ventajas

Debido a que se trata de un simple cable, es obvio que una ventaja muy competitiva es el precio, en amazon.es es posible encontrarlos desde 2€.

Imagen 3.1: Cable OTG

http://myfimport.com.co/tienda/convertidor-otg-puerto-v8-para-tablet-o-smartphone/

Desventajas

Se pierde completamente la movilidad y la portabilidad del dispositivo, en el caso de que se quisiera desarrollar en un entorno donde cualquier usuario pudiera emparejar su dispositivo móvil con el sistema, no sería viable usar este sistema.

No todos los dispositivos son compatibles con OTG, no existe un listado oficial, cada dispositivo es diferente, y trata los dispositivos de manera diferente, se requiere un controlador para la comunicación y al no tratarse de un estereotipo propiamente dicho no se tiene la seguridad de cómo va a responder con un determinado dispositivo conectado, en este caso Arduino.

Con cada actualización de sistema el fabricante del Tablet/móvil puede activar, desactivar, o modificar el funcionamiento de esta funcionalidad, véase el ejemplo de Nexus 4 que al principio esta funcionalidad no estaba activa y posteriormente se habilitó.

No existe una funcionalidad nativa de base proporcionada por Google.

Por todos estos motivos no se trata del método más adecuado para implementar en el sistema que se va a desarrollar.

3.1.1. Wifi

El Wifi es una tecnología de comunicación inalámbrica, también conocida como wireless (conexión sin cables), esta tecnología está ampliamente extendida y es el estereotipo de conexión entre todo tipo de dispositivos, como son, ordenadores, televisiones, routers, puntos de acceso, teléfonos, tabletas etc.

Esta tecnología permite crear redes, una red es un conjunto de equipos informáticos conectados entre sí que envían y reciben datos.

Esta tecnología utiliza la frecuencia 2,4 GHz, sigue los estándares IEEE 802.11 [56], actualmente con el estándar 802.11n, es posible alcanzar velocidades de hasta 300Mbps, además se pueden crear redes a muchos metros e incluso kilómetros.

El Wifi, también implementa métodos de seguridad, como son métodos de encriptación, actualmente en el mercado se utilizan WEP [57], WPA2 [58] etc.

Ventajas

Este tipo de comunicación es capaz de conseguir una velocidad muy alta (actualmente el estereotipo 802.11n hasta 300Mbps), implementa una seguridad bastante robusta.

Desventajas

Al tratarse de una tecnología muy avanzada para comunicación entre dispositivos, no es sencilla la infraestructura capaz de proveer una conexión punto a punto entre los dispositivos, ni los dispositivos para gestionar la comunicación entre un microcontrolador y una Tablet.

Además el dispositivo para gestionar conexiones inalámbricas en Arduino tiene un coste más elevado y la implementación del protocolo de comunicación es más complejo, pues no se trata de una comunicación secuencial [59].

Alto consumo de energía, ya que cada paquete de información enviado requiere una codificación, y el recibido una decodificación, esta tecnología no está diseñada para ahorrar energía.

3.1.2. Bluetooth

Es un protocolo de comunicaciones que fue diseñado pensando en dispositivos móviles de bajo consumo, no consigue altas velocidades, ni grandes distancias, pero es ampliamente utilizado en dispositivos que utilizamos cada día, como por ejemplo, auriculares, ratones, teclados, teléfonos móviles (para transmisión de música, video, ficheros, emparejamiento de datos entre Smart watches [60], mandos de consolas (play station), etc.).

La comunicación bluetooth define un canal de comunicación a una velocidad 1 Mbit/s, con un rango óptimo de 10 metros.

El hardware que compone el Bluetooth está compuesto por dos partes:

- **Un dispositivo de radio**: encargado de modular y transmitir la señal.
- **Un controlador digital**: compuesto por una CPU y un procesador de señales.

Ventajas

Al estar este tipo de protocolo especialmente diseñado para dispositivos de bajo consumo, es ideal para este tipo de proyecto, pues el software estará ejecutándose en un Tablet (con batería), que eventualmente puede estar cargándose o no.

Además este protocolo es muy conveniente también para la programación en microcontroladores, pues existen muchas ofertas en el mercado que proveen una comunicación bluetooth transparente para el desarrollador, por lo que la comunicación puede tratarse como si fuera un texto sin formato (palabras simples), para el dispositivo y el programador la comunicación es transparente.

Desventajas

Las mayores desventajas de bluetooth, son la baja velocidad de transferencia de datos, y el poco rango de actuación que tiene.

Para este proyecto esas desventajas no son especialmente problemáticas, pues los datos que se transferirán no son privados ni tampoco se va a hacer un uso de la máquina desde una distancia muy lejana.

3.2. Solución que se implementará en el proyecto

Debido a sus grandes ventajas, y a que las desventajas no son muy relevantes para este proyecto, se ha decidido utilizar una comunicación basada en bluetooth.

A continuación explicaré la topología de la conexión que se establecerá y que tipo de comunicación se implementará.

La comunicación se lleva a cabo mediante comunicación serie, pues es la forma más sencilla y elemental que se puede utilizar, para el microcontrolador este mensaje será recibido como si de un texto plano se tratase, toda la capa de comunicación inalámbrica es transparente.

Para comprobar que un mensaje ha sido recibido correctamente por el receptor, el receptor (microcontrolador) volverá a reenviar un mensaje con el mismo contenido enviado por el emisor, comparándolo se puede saber si ha sido satisfactoria o no.

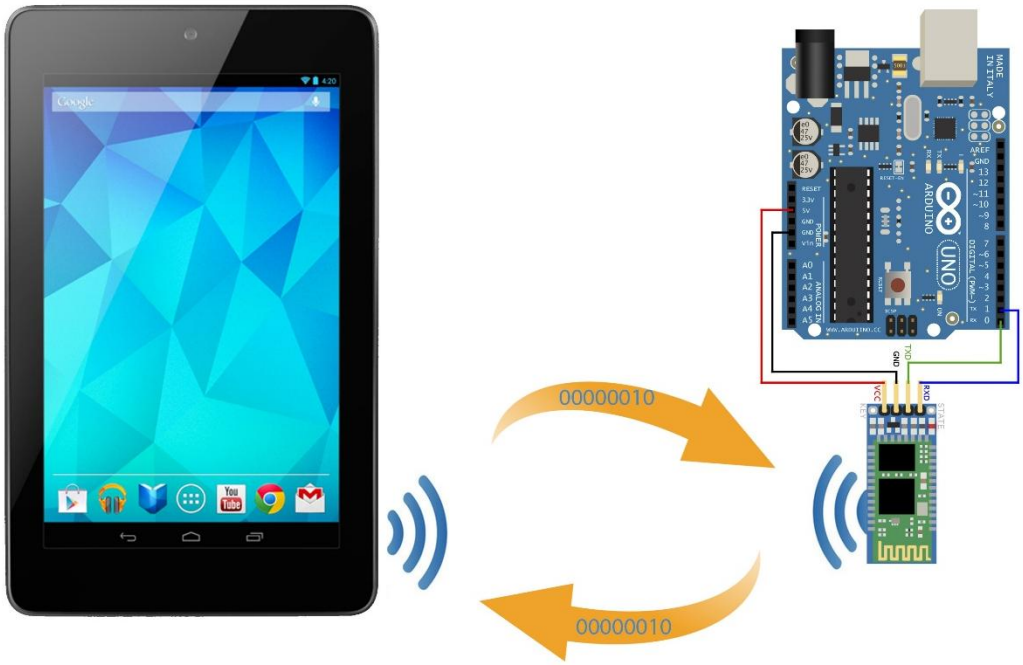


Imagen 3.2: Diagrama comunicación teléfono móvil con Arduino

3.3. Lenguaje de comunicación entre la máquina y el dispositivo móvil

Las funciones que deberán ser capaz de interpretar el sistema para poder realizar todas las acciones necesarias para dispensar el producto son las siguientes:

Funciones del carro [0001XXXX]: La longitud de la máquina se dividió en 256 posiciones, esto significa que el carro podrá situarse en 256 posiciones diferentes, de esta manera se obtienen suficiente resolución para el correcto funcionamiento de la máquina, para representar la posición se utilizara un byte, en un byte es posible representar 256 valores diferentes.

El microcontrolador presupone que todas las instrucciones recibidas son correctas, por lo tanto el que envía la señal de parada de emergencia, en caso de que la retroalimentación [61] sea incorrecta, es el Tablet.

Las instrucciones se envían consecutivamente y Arduino las va procesando e interpretando según el orden de llegada, por lo que es importante esperar la retroalimentación de cada instrucción para asegurarse de que se ha recibido la orden correctamente.

- **Parada de emergencia [00011111]:** Esta instrucción interrumpe cualquier instrucción enviada al motor, si esto se ejecuta la transmisión debe reiniciarse desde el principio, el significado de x en el código significa que no es relevante.
- **Desplazamientos del motor:** Esta rutina se usará para realizar alguna acción referente al motor que genera un desplazamiento horizontal en el carro. Tras la ejecución de estas órdenes se recibirá un byte de confirmación “R” o 0x52.
    - **Acción desplazamiento genérico [**0001**0001]:** Se especifica la posición a la cual el carro debe de desplazarse (Arduino sabe en todo momento en que lugar está situado el carro).
        - **Posición a desplazar:** escalado del deslizador de 0 a 2500 posiciones, se enviará un entero comenzado por el Bit más significativo (MSB)

        - **Velocidad (0-213):** 0 es automático, Arduino lo controla de manera inteligente.
    - **Desplazamiento a la izquierda [**0001**0010]:**
        - **Cantidad de pasos**
            - **Posible error:** Fuera de rango [11111111]
        - **Velocidad (0-213):** 0 es automático, Arduino lo controla de manera inteligente.
    - **Desplazamiento a la izquierda [**0001**0100]:**
        - **Cantidad de pasos**
            - **Posible error:** Fuera de rango [11111111]
        - **Velocidad (0-213):** 0 es automático, Arduino lo controla de manera inteligente.
    - **Calibración de carro [**0001**1000]:** El carro se desplazará hasta la posición de reposo (0) y empezará todo desde el principio (Arduino re-calibrará el carro cada cierto número de movimientos de manera automática)

Funciones de las bebidas [0010XXXX]: Estás rutinas son válidas tanto para las bebidas presurizadas como para las bebidas suspendidas de la barra, con estas funciones se pretende crear la interface para interactuar y llevar a cabo todas las operaciones necesarias para dispensar una bebida.

- **Dispensar [**0010**0001]:** Esta función se encarga de dispensar una determinada bebida en el vaso.
    - **Posición:** Arduino controlará si existe la posición indicada o no, debido a la cantidad de botellas se utilizará 1 byte dividido en 2 Nibble [62]
        - **Posible error:** Fuera de rango [11111111]
        - **Cantidad [entero (2 byte)]:** La cantidad será enviada a Arduino en forma de mililitros, Arduino se encargará de gestionar las cantidades correctamente, el controlador tendrá dos modos de servir, precisión y alta velocidad, el modo será elegido automáticamente por Arduino, el cual controlará dependiendo del valor que se envíe en el parámetro de cantidad.
- **Cambiar botella [**0010**001X]:** Mantendrá parado el compresor si se trata de una botella presurizada, en caso de que sea una botella de barra el Tablet no enviará nada al controlador, puesto que no se requiere de ninguna acción especial.
    - **Activado/Desactivado:** [00100011] **/** [00100010]

- **Botellas de compresor [**0010**01XX]:** Estas órdenes serán enviadas por Arduino.
    - **[**00100**01]:** Para indicar al Tablet que la botella ha sido presurizada correctamente y están listas para servir.
    - **[**00100**10]:** Error en la presurización de las botellas, debe de solicitarse inspección en el sistema para continuar.
- **Finalizar coctel [**0010**1000]:** Esta orden será enviada para llevar a cabo la acción de finalizar **coctel**, Arduino llevará el carro a la posición de finalización y ejecutara la secuencia de luces del vaso programada para ello.
- **Funciones de control de peso [0011XXXX]:** Estas rutinas son utilizadas para interactuar con el peso electrónico instalado en el carro de la máquina.
- **Recalibrar peso a 0 [**0011**0001]:** Esta función recalibrará el peso a 0, sin tener en cuenta el peso que actualmente tenga encima, cuando el coctel comienza se deberá llamar a esta función para descartar el peso del vaso.
- **Preguntar por el peso actual [**0011**0010]:** Esta función sirve para obtener el paso actual que contiene el peso, empezando desde la última vez que se recalibró, se responde con un entero comenzado por el MSB

Funciones de control de luces [010XXXXX]: Estas funciones controlarán la iluminación de la máquina, la cual se compone de 2 grupos de leds, en máquina y en vaso, cada una llevada por un microcontrolador diferente y 3 líneas de color que las controlan: vaso, barra y máquina (luz general).

- **Color de Líneas [**010**0XXXX]:** el byte se compone del código de “Funciones de luces, cambio de color” más 2 bits que determinan la línea más 2 bits que determinan el modo de funcionamiento de esta.
    - **Elección de Línea:**
        - **Vaso [**0100**01**XX**]**
        - **máquina [**0100**00**XX**]**
        - **barra [**0100**10**XX**]**
    - **Modo:**
        - **Color fijo [**0100XX**00]:** Cambia la línea a un color fijo
            - + 3byte de color RGB
        - **Color con transición suave [**0100XX**01]:**
            - + 3byte de color RGB + 1 byte de tiempo
        - **Modo disminución continua [**0100XX**10]:**
            - + 3byte de color RGB + 2 byte de tiempo
        - **Modo estroboscópico continuo [**0100XX**11]:**
            - + 1 byte de tiempo

- **Recoger [**010**011XX]:** Esta función usualmente se llama cuando se termina de servir un coctel, Arduino hace que las luces del vaso parpadeen hasta que el vaso haya sido recogido por el usuario (utilizando sensor de peso).

- **Activado/desactivado:** [01001101] **/** [01001100]

- **Elección de línea máquina o barra [**010**1XXXX]:** el byte se compone del código de “Elección de línea” + 1 bit de elección de Línea Maquina o Barra + 3bite de elección de puesto (1 - 8)
  - **Elección de Línea:**
    - **Máquina [**0101**0**XXX**]**
    - **Barra [**0101**1**XXX**]**
  - **Puesto [**0101X**XXX]:** los 3 LSB determinan el puesto que se desea cambiar

CAPÍTULO 4

Análisis del proyecto

4.1. Descripción

Una de las definiciones de Ingeniería del Software es la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación (funcionamiento) y mantenimiento del software.

El proceso requiere una metodología con 5 etapas:

- **Análisis de requerimientos**: Es la etapa en la que se especifican los requisitos del producto a desarrollar.
- **Especificación**: Es la etapa en la que se procede a desarrollar una descripción exhaustiva y rigurosa del software a desarrollar, también se detalla el comportamiento esperado por el software y como es la intervención del usuario y otros sistemas.
- **Programación**: Se traduce el diseño a código. Es la parte más obvia del trabajo de ingeniería de software y la primera en que se obtienen resultados “tangibles”. No necesariamente es la etapa más larga ni la más compleja aunque una especificación o diseño incompletos/ambiguos pueden exigir que, tareas propias de las etapas anteriores se tengan que realizarse en esta.
- **Prueba**: Consisten en la comprobación del software, se trata de un conjunto de pruebas que sean capaces de determinar si el comportamiento del software y sistema es el esperado para los casos de uso dados.
- **Documentación**: Se trata de la realización de manual de usuario, para los diferentes tipos de usuario que van a utilizar el sistema.
- **Mantenimiento**: En esta etapa se realizan un mantenimiento correctivo (resolver errores) y un mantenimiento evolutivo (mejorar la funcionalidades y/o dar respuesta a nuevos requisitos).

4.2. Análisis de requerimientos.

En esta etapa procederé a desarrollar los requerimientos del sistema a implementar, se detallarán los servicios que ha de prestar el sistema hacia un potencial cliente o usuario del mismo.

Al tratarse de un proyecto cuyo fin es académico su propósito es claro:

“Llevar a cabo Dispensador automático de combinados de bebidas”

Una vez teniendo claro el propósito del proyecto, me dispondré a desarrollar los requerimientos del mismo, las cuales son la descripción del conjunto de propiedades o restricciones definidas con precisión que el sistema debe cumplir, para ello se especificarán dos tipos de requerimientos, requerimientos funcionales y requerimientos no funcionales.

- **Requerimientos funcionales:** Detallan lo que un sistema hace o lo que se espera que se haga, detallan la funcionalidad en sí.
- **Requerimientos no funcionales:** Describen los aspectos del sistema relacionados con el grado de cumplimiento de los requisitos funcionales.

4.2.1. Requerimientos funcionales

Son las declaraciones de los servicios que un sistema debe proveer.

Para comenzar hay que diferenciar entre los diferentes tipos de usuarios que utilizarán el sistema:

- **Consumidor**: Persona que llevará a cabo el uso más simple del sistema, el objetivo principal de este usuario es la de llevar a cabo una dispensación del coctel para su posterior bebida.
    - **Elegir coctel**: Ser capaz de decidir que coctel seleccionar en base a información de ingredientes, alcohol y dulzor del mismo.
    - **Dispensar coctel**: Poder dispensar un coctel previamente seleccionado.

  - **Seleccionar tamaño del vaso**: Configurar el tamaño del vaso que se va a servir (y por lo tanto la cantidad de coctel)
  - **Tomar decisiones en caso de error**: En caso de que no existiera suficiente cantidad de ingredientes para dispensar el coctel seleccionado elegir si desea continuar o no dispensando una menor cantidad.
- **Propietario**: Persona que gestiona la máquina, encargado de hacer mantenimiento de las botellas, configurar recetas, y configuración de la máquina (configuración a nivel básico desde la aplicación)
  - **Conectar con máquina**: Gestionar las conexiones a través de bluetooth con la máquina, llevar a cabo el emparejamiento.
  - **Mantenimiento de botellas**: Gestionar todo lo referente al mantenimiento de botellas
    - **Configurar nuevas botellas**: Configurar nuevas botellas, detallando tamaño de la botella y propiedades de la bebida que contiene, configurando también la posición en la que se encuentra en la máquina.
    - **Editar botellas existentes**: Poder modificar las propiedades de una botella existente, tanto valores como posición en la máquina.
  - **Gestión de recetas**: Crear, editar o eliminar recetas utilizando las botellas que se encuentren en el sistema, seleccionando cantidad (porcentaje del total del coctel) y orden de dispensación.
  - **Identificarse como administrador**: Para llevar todas las tareas de este usuario es necesario identificarse como administrador del sistema, utilizando una contraseña de administrador que es posible configurarla, utilizando también la contraseña anterior de administrador.
  - **Configurar parámetros**: Gestionar y configurar parámetros de configuración del sistema.
    - **Nombre del local**: Poder asignar un nombre para que aparezca en la parte superior de la aplicación
    - **Contraseña de administrador**: Activar el sistema con contraseña y en caso afirmativo configurarla.

        - **Gestionar vaso**: Gestionar tamaño mínimo y máximo del vaso así como la cantidad de incremento y decremento del botón a la hora de configurar el vaso.
        - **Configuración de iluminación**: Gestionar colores de la iluminación de la máquina en los casos cuando la botella está activa e iluminación del carro.
        - **Configuración de colores en gráficas**: Personalizar colores de las gráficas de alcohol y dulzor.
        - **Gestión avanzada de parámetros**: configuración de parámetros de sincronización y tiempos para optimizar la dispensación de cocteles.
        - **Ver historial de dispensaciones**: Consultar un historial de dispensaciones viendo receta, fecha y estado de la dispensación.
        - **Ver historial de errores**: Consultar historial con los errores que se han producido en la máquina, visualizando motivo, fecha de cuando el error ha ocurrido.
    - **Administrador**: Persona/s encargadas del servicio técnico y mantenimiento de segundo nivel del sistema.
        - **Depurar sistema:** El microcontrolador cuenta con un sistema de depuración, para ello es necesario un ordenador con el software necesario, con este software se puede ver que bytes se están enviando desde la máquina en cada momento.

4.2.2. Requerimientos no funcionales

Describen los aspectos del sistema que están relacionados con el grado de cumplimiento de los anteriormente mencionados requerimientos funcionales, estos especifican normalmente las necesidades de software y hardware como por ejemplo pueden ser la interface [63], los dispositivos etc., normalmente son impuestos por un cliente, estándares y leyes.

En este caso se trata de un proyecto académico, por lo que solamente voy a centrarme en las necesidades de software y hardware para asegurar el correcto

funcionamiento del sistema y que la experiencia del usuario sea lo más sencilla y simple posible.

Requerimientos del sistema

En esta sección se va a detallar los requerimientos tanto del equipo que se usará para utilizar la aplicación desarrollada en este proyecto como del microcontrolador encargado de llevar a cabo las acciones asociadas a la comunicación enviada desde la aplicación Android.

Los requisitos mínimos para el sistema son:

- Android 4.0 (o superior)
- Dispositivo con conectividad bluetooth

Hardware

Hay que diferenciar entre varios aspectos, dispositivo encargado de ejecutar la aplicación Android de control y microcontrolador encargado de controlar la máquina.

Este proyecto ha sido realizado en un equipo compuesto por dos personas Adrián Díaz García (Estudiante de Grado en Ingeniería Informática) y José Manuel Contreras Parras (Estudiante de Grado en Ingeniería Electrónica).

En este proyecto se tratan los temas referentes a diseño de software y desarrollo de la aplicación móvil.

Cabe mencionar que los dispositivos hardware elegidos para la maquina son los siguientes:

- **Arduino Atmega2560:** Este microcontrolador se utilizará como controlador principal del sistema y de otros microcontroladores secundarios. Se estuvieron sopesando otras versiones como Arduino Galileo y Arduino Due, pero debido al presupuesto reducido que se contaba para realizar el proyecto determinamos que la utilización de un Arduino Atmega2560 era más que suficiente.
- **Atmega328:** Este microcontrolador se utilizará sin montaje en placa, lo que significará que se utilizará el chip en sí mismo, por razones de

espacio y precio, al tratase de la versión sin placa el precio es mucho más reducido.

En el proyecto de José Manuel Contreras Parras se pueden encontrar los desarrollos y especificaciones de la parte hardware en más detalle.

El dispositivo móvil es necesario que cuente con una plataforma capaz de ejecutar código Android, que disponga de una conexión bluetooth, que cuente con espacio suficiente de almacenamiento para poder almacenar la aplicación (aproximadamente 6 Megabytes), una base de datos que almacene todas las configuraciones y recetas de la misma (aproximadamente 84 Kilobytes), que disponga de una potencia de procesamiento mínima necesaria para poder ejecutar la aplicación y que cuente con suficiente memoria RAM.

Software

Con respecto al software los requerimientos necesarios para que el sistema funcione son:

- Android 4.0 o superior
- Sistema operativo capaz de ejecutar una base de datos SQLite

Requerimientos de la interfaz para la aplicación Android

Cuando se desarrolla una aplicación que va a ser usada por diferentes tipos de personas, se pretende que la utilización de la misma sea lo más sencilla posible sin influir en el número de funciones que esta puede desarrollar, en este punto debemos de centrarnos en la experiencia del usuario [64].

La experiencia de usuario es el proceso que lleva a cabo el usuario cuando interactúa con un producto.

Hay que distinguir entre usabilidad y experiencia del usuario:

- **Usabilidad**: “Es la facilidad con que las personas pueden utilizar una herramienta particular o cualquier otro objeto fabricado por humanos con el fin de alcanzar un objetivo concreto.” [65].

- **Experiencia de usuario**: “Conjunto de factores y elementos relativos a la interacción del usuario, con un entorno o dispositivo concretos, cuyo resultado es la generación de una percepción positiva o negativa de dicho servicio, producto o dispositivo.” [66].

Para que la aplicación sea usable debe seguir los siguientes puntos:

- **Facilidad de aprendizaje**: que se define como el tiempo mínimo necesario desde el primer contacto con la aplicación sin tener ninguna idea de cómo funciona la misma, hasta ser capaz de utilizarla de una manera productiva. En este punto, la aplicación diseñada por mí debe contar con aspectos ya conocidos por el usuario (metáforas), o botones claros que indiquen acciones a realizar, también todos los sucesos que se lleven a cabo deben estar, si no quedan totalmente claros, explicados con texto sencillo.
- **Consistencia**: La aplicación debe ser consistente durante todos las fases de utilización de la misma, por ejemplo si para la selección de un ingrediente se presentan los elementos de una determinada manera, estos elementes deben presentarse de la misma manera durante toda la aplicación (si esto es conveniente), el objetivo de la consistencia es hacer sentir al usuario que se encuentra utilizando la misma aplicación durante todo su uso.
- **Robustez**: En la medida de lo posible la aplicación debe ser capaz de gestionar contratiempos o errores que puedan producirse, bien de manera automática o con la intervención del usuario, detallando claramente que pasos a seguir en cada determinado momento.
- **Tiempo de respuesta**: El tiempo necesario que necesita la aplicación para mostrar los cambios o acciones realizadas por el usuario, en el caso de la aplicación, los cambios se suceden casi en tiempo real, pues no requiere un gran poder de procesamiento.
- **Recuperabilidad**: El sistema debe permitir al usuario recuperarse ante errores inesperados, esta característica va muy unida a la robustez.
- **Disminución de la carga cognitiva**: Para que el usuario se sienta cómodo utilizando la aplicación los elementos de la misma deberán

situarse de la misma manera, y siguiendo una lógica fácilmente recordable por el usuario, así con los usos posteriores el usuario se sentirá cada vez más cómodo.

4.2.2. Análisis del sistema

Una vez clarificados los principales conceptos necesarios para el correcto funcionamiento del sistema, realizaré un análisis del mismo en el que detallare cada una de las etapas por las que ha transcurrido el proceso de creación del mismo. Entre ellas se encuentran la definición de los diferentes perfiles de los usuarios que utilizarán el sistema, los casos de uso que se pueden dar con respecto al uso del sistema, así como el diagrama de Lenguaje Unificado de Modelado (UML). Para completar el análisis, desarrollaré la técnica de los escenarios en la cual probaré el resultado de los casos de uso descritos anteriormente.

Perfiles

Aquí se definirán los diferentes perfiles de usuario que utilizarán el sistema, gracias a estos perfiles será más fácil identificar los requisitos que debe cumplir la aplicación.

Consumidor

El consumidor es la persona que quiere consumir una bebida, utilizando la aplicación puede servirse por sí mismo un combinado.

Conocimientos relativos al programa: el consumidor como usuario, no tiene por qué haber tenido ningún contacto previo con la aplicación.

Habilidad con la tecnología: solamente ha de saber cómo se maneja un dispositivo móvil (teléfono o Tablet) Android, sabiendo que es táctil y ser capaz de entender y seguir instrucciones.

Propietario y Administrador

El propietario es la persona que es propietaria de la máquina y el que la gestiona, se encargará de introducir nuevas botellas en el sistema, cambiar las botellas vacías, llevar a cabo todas las configuraciones del mismo etc.

Conocimientos relativos al programa: el propietario de la máquina debe leer con detalle el manual de la aplicación aparte de recibir un pequeño entrenamiento de cómo gestionar y mantener la máquina.

Habilidad con la tecnología: al igual que el consumidor final, el propietario solo debe tener conocimientos básicos de cómo se maneja un dispositivo móvil Android.

4.2.3. Casos de uso

Un caso de uso es una explicación de cada uno de las actividades o pases que hay que realizarse para llevar a cabo con éxito un proceso. Los personajes o entidades que participarán en un caso de uso se denominan actores, un caso de uso en ingeniería del software es una secuencia de interacciones que se desarrollarán entre un sistema y sus actores en respuesta a un evento que inicia un actor principal sobre el propio sistema.

Tipos de relaciones: existen dos tipos de relaciones.

- **Extiende:** Relación de dependencia entre dos casos de uso que denota que un caso de uso es una especialización de otro. <<extends>>
- **Incluye:** Relación de dependencia entre dos casos de uso que denota la inclusión del comportamiento o de un escenario en otro. <<include

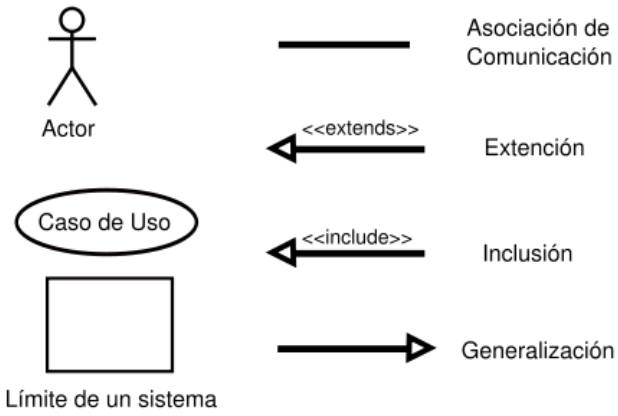


Imagen 4.1: Tabla con notación en un diagrama de casos de uso en UML.
https://es.wikipedia.org/wiki/Caso_de_uso

Diagrama frontera

El siguiente diagrama define cuales son las interacciones que se pueden llevar a cabo con el sistema y que tipo de roles, estas interacciones serán explicadas más profundamente en cada uno de los casos de uso.

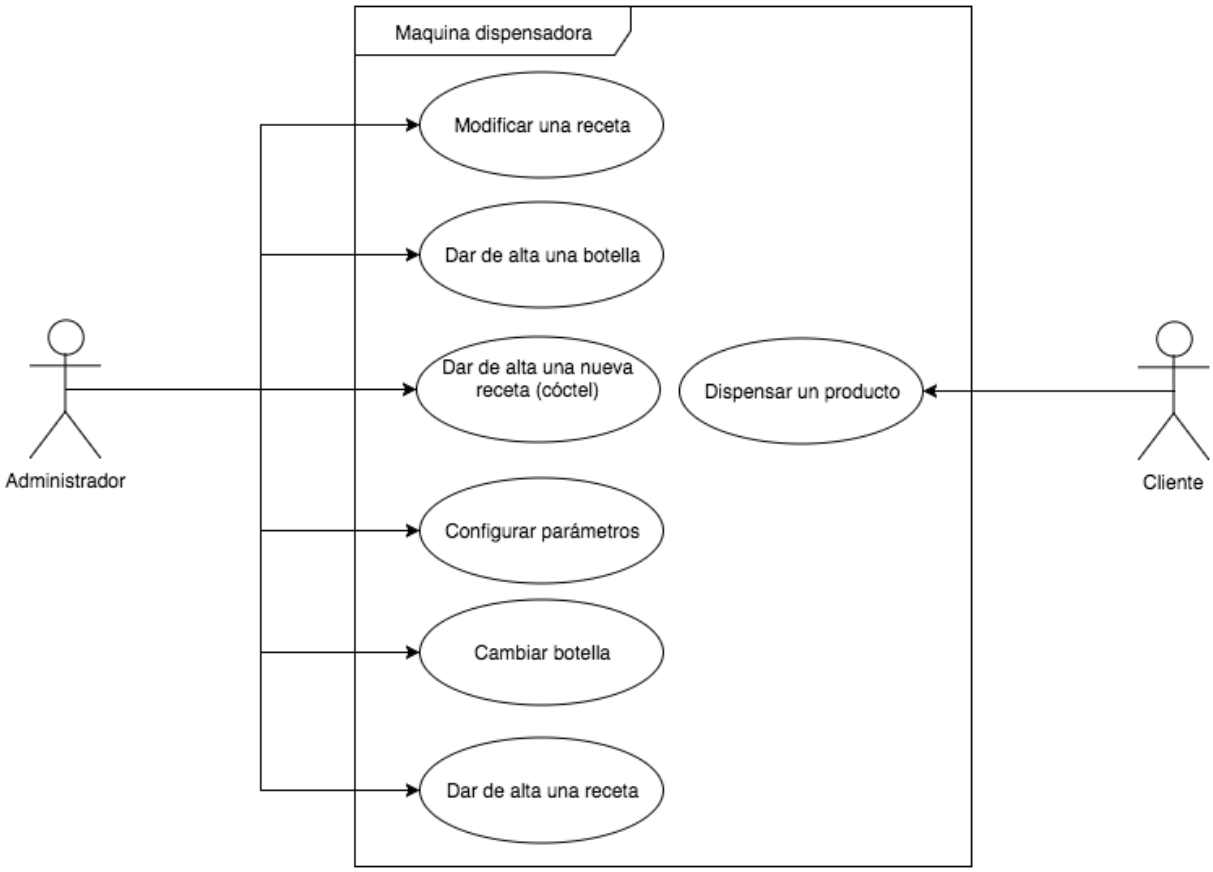


Imagen 4.2: Diagrama frontera.

En los siguientes casos de uso el rol de propietario lo llevará el Administrador, y el de consumidor el cliente.

Caso de uso 1: Dispensación de una bebida

En este caso de uso en potencial cliente quiere dispensar una bebida, llevando a cabo la selección de la misa en la aplicación.

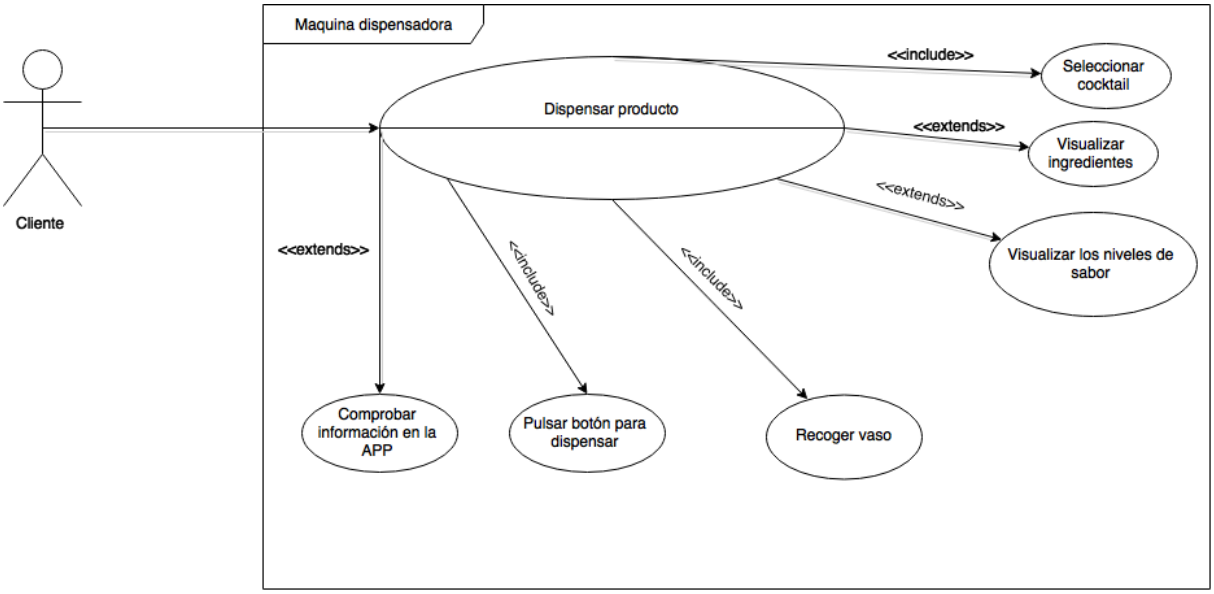


Imagen 4.3: Diagrama UML, caso de uso dispensación de una bebida.

Caso de uso	Dispensación de una bebida
Actor primario	Cliente
Sistema	Maquina dispensadora de bebidas
Participantes	Cliente, Tablet, máquina dispensadora.
Condición previa	La máquina debe estar encendida y el Tablet conectado a la maquina correctamente.
Operaciones básicas	
1	Seleccionar un coctel
2	Visualizar los ingredientes
3	Visualizar los niveles de sabor
4	Pulsar botón para dispensar
5	Comprobar información aplicación
6	Recoger vaso
Alternativas	
4.A	¿Es posible dispensar ese coctel?
4.A.1	Si no, seleccionar otro coctel
4.A.2	Si si, continuar
5.A	¿Se ha producido algún error?
5.A.1	Si no, continuar
5.A.2	Si si, seguir las instrucciones mostradas en la aplicación

Caso de uso 2: Configuración parámetros

En este caso el propietario de la maquina quiere acceder al menú de ajustes, donde se pueden cambiar ajustes tanto de la aplicación software como de la configuración de luces de la propia máquina.

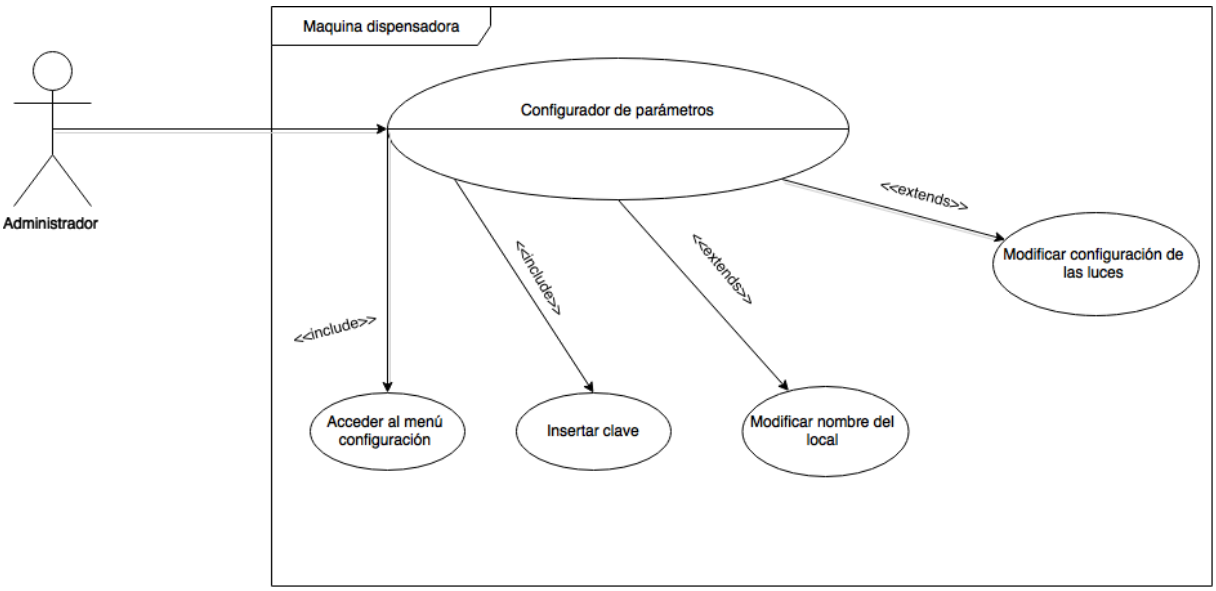


Imagen 4.4: Diagrama UML, caso de uso configuración de parámetros.

Caso de uso	Dispensación de una bebida
Actor primario	Propietario de la maquina
Sistema	Maquina dispensadora de bebidas
Participantes	Propietario, Tablet, máquina dispensadora.
Condición previa	La máquina debe estar encendida y el Tablet conectado a la maquina correctamente.
Operaciones básicas	
1	Acceder a menú configuración
2	Insertar clave
3	Modificar nombre local
4	Visualizar historial

5	Modificar configuración de luces
Alternativas	
2.A	¿Clave correcta?
2.A.1	Si no, intentar de nuevo, o regresar
2.A.2	Si si, continuar

Caso de uso 3: Cambiar botella

En este caso el propietario de la máquina quiere cambiar una botella, y poner otra nueva.

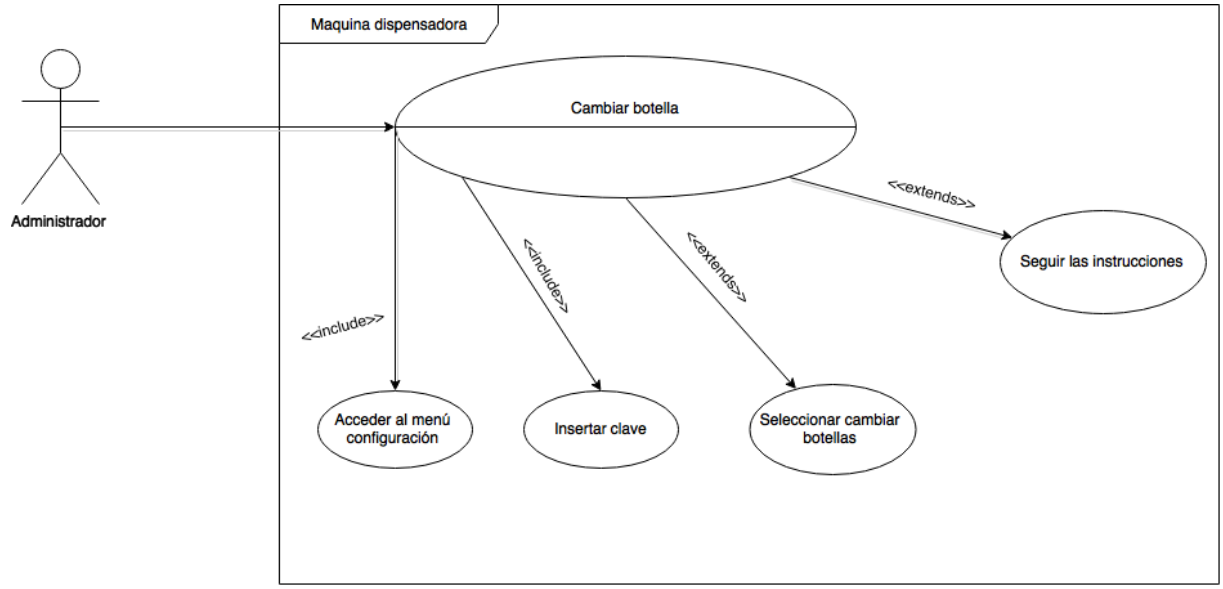


Imagen 4.5: Diagrama UML, caso de uso cambiar botella.

Caso de uso	Dispensación de una bebida
Actor primario	Propietario de la maquina
Sistema	Maquina dispensadora de bebidas
Participantes	Propietario, Tablet, máquina dispensadora.
Condición previa	La máquina debe estar encendida y el Tablet conectado a la maquina correctamente.
Operaciones básicas	
1	Acceder a menú configuración
2	Insertar clave
3	Seleccionar cambiar botellas

4	Seguir las instrucciones
Alternativas	
2.A	¿Clave correcta?
2.A.1	Si no, intentar de nuevo, o regresar
2.A.2	Si si, continuar

Caso de uso 4: Dar de alta una nueva bebida (botella)

En este caso de uso se describe como el propietario de la maquina quiere dar de alta una nueva botella en el sistema.

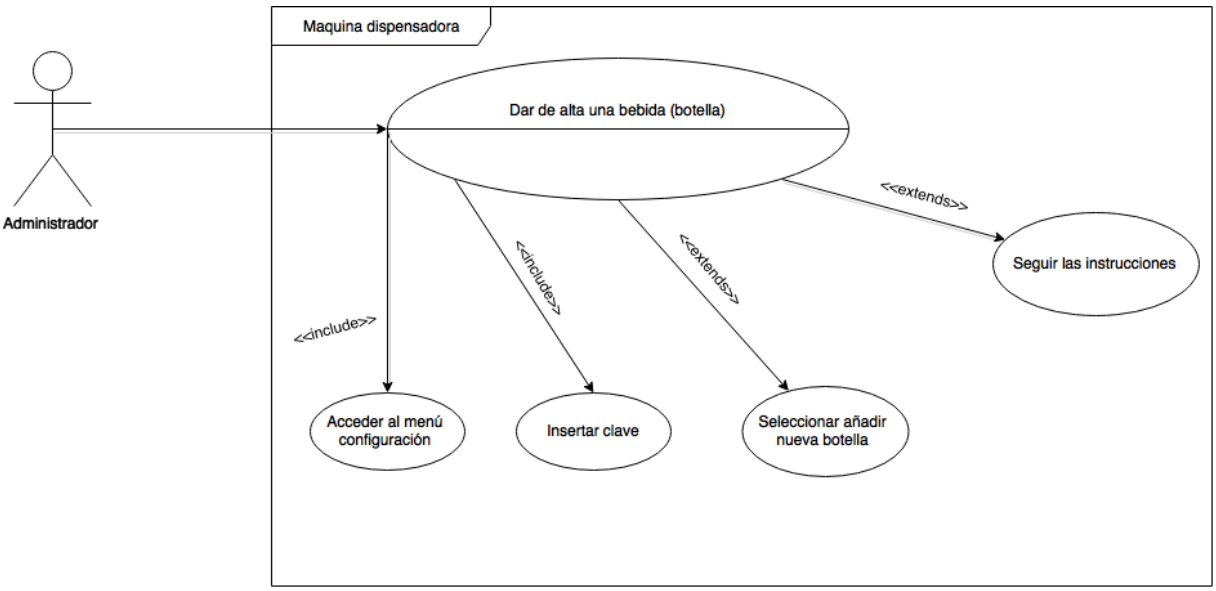


Imagen 4.6: Diagrama UML, caso de uso dar de alta una nueva bebida.

Caso de uso	Dispensación de una bebida
Actor primario	Propietario de la maquina
Sistema	Maquina dispensadora de bebidas
Participantes	Propietario, Tablet, máquina dispensadora.
Condición previa	La máquina debe estar encendida y el Tablet conectado a la maquina correctamente.
Operaciones básicas	
1	Acceder a menú configuración
2	Insertar clave

3	Seleccionar añadir botellas
4	Seguir las instrucciones
Alternativas	
2.A	¿Clave correcta?
2.A.1	Si no, intentar de nuevo, o regresar
2.A.2	Si si, continuar

Caso de uso 5: Insertar una nueva receta coctel

El propietario y administrador de la maquina quiere crear un nuevo coctel, mezclando las bebidas que ya están disponibles en la aplicación.

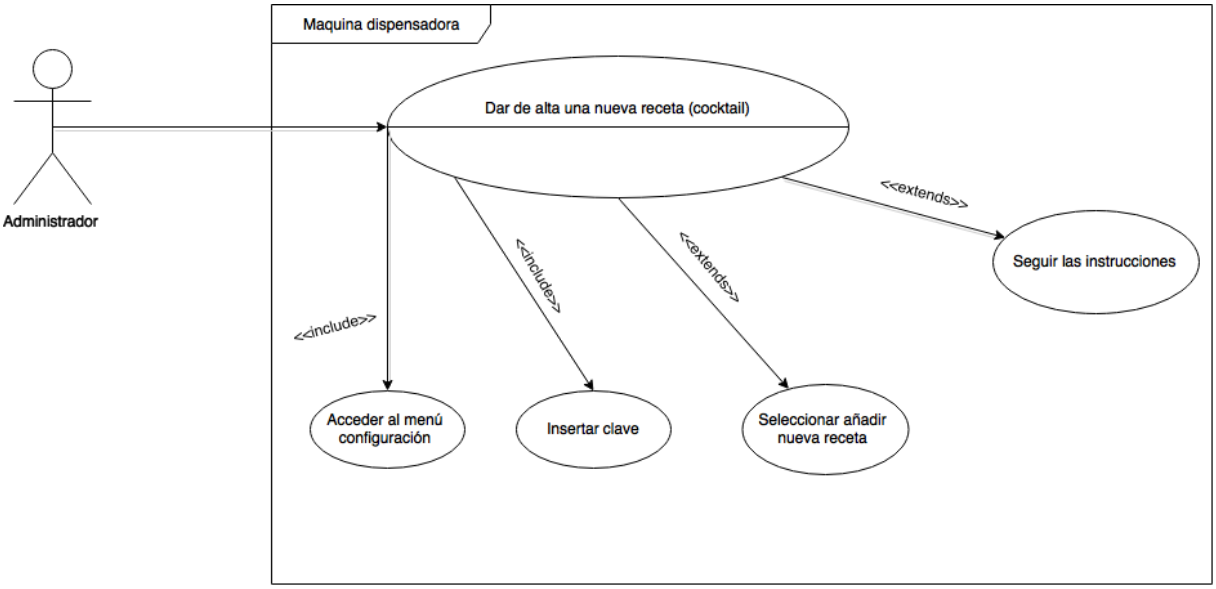


Imagen 4.7: Diagrama UML, caso de insertar una nueva receta.

Caso de uso	Dispensación de una bebida
Actor primario	Propietario de la maquina
Sistema	Maquina dispensadora de bebidas
Participantes	Propietario, Tablet, máquina dispensadora.
Condición previa	La máquina debe estar encendida y el Tablet conectado a la maquina correctamente.
Operaciones básicas	
1	Acceder a menú configuración

2	Insertar clave
3	Seleccionar añadir nueva receta
4	Seguir las instrucciones
Alternativas	
2.A	¿Clave correcta?
2.A.1	Si no, intentar de nuevo, o regresar
2.A.2	Si si, continuar
4.A	¿Existen todas las bebidas necesarias?
4.A.1	Si no, añadir primero las bebidas necesarias
4.A.2	Si si, continuar

4.2.4. Escenarios

Cuando se habla de las funcionalidades principales que he definido en los casos de uso, se hacen sobre representaciones abstractas, para concretar más estas situaciones se utilizan los Escenarios.

Los escenarios es un método de planificación estratégica para predecir el uso de un sistema a largo plazo, para llevar a cabo esta técnica solamente es necesario crear historias ficticias con un nivel adecuado de detalle aplicadas al proyecto y así poder predecir mejor y más acertadamente posibles requerimientos del sistema.

Un Escenario se define como "*una historia de ficción con representación de personajes, sucesos, productor y entornos*".

Para un correcto estudio de todos los casos que se pueden dar, es necesario diseñar varios escenarios para poder reflejas las diferentes posibilidades y puntos de vista que pueden darse.

Existe la posibilidad de llevar a cabo más de un escenario por cada uno de los casos de uso, aunque uno por cada caso de uso debería ser suficiente para ver las cosas más claras.

Los escenarios se componen de los siguientes componentes:

- Nombre del escenario.

- Descripción
- Actores principales
- Flujo de eventos

Nombre del escenario	**Ana Dispensación Mojito**
Descripción	El usuario Ana quiere dispensar un mojito
Actores principales	Ana, es una chica joven de 25 años, ha salido con sus amigas a tomar algo, y le gustaría tomarse un mojito.
Flujo de eventos	1. El usuario desbloquea el teléfono/Tablet que se encuentra junto a la máquina. 2. Se muestra la primera actividad la cual contiene un listado de bebidas, entre ellas el mojito. 3. Selecciona el mojito, comprueba que los ingredientes son correctos y que el grado de dulzor y alcohol es el deseado. 4. Pulsa el botón dispensar 5. Ahora la aplicación pregunta el tamaño del vaso que va a ser utilizado para servir el coctel. 6. Si todo está correcto Ana verá una barra de progreso indicando el progreso de la dispensación de la bebida. 7. Una vez finalizada la receta Ana recogerá el vaso del carro de la máquina.

Nombre del escenario	**Pedro añadir botella**
Descripción	El administrador Pedro quiere añadir una botella al inventario
Actores principales	Pedro es el gerente de un bar y propietario de la máquina dispensadora de cocteles.
Flujo de eventos	1. El usuario desbloquea el teléfono/Tablet que se encuentra junto a la máquina. 2. Se muestra la primera actividad la cual contiene un listado de bebidas, y un menú superior para llevar a cabo varias acciones. 3. Selecciona la acción Gestionar botellas. 4. Una vez pulsado aparecerá (si está activada la opción) un mensaje solicitando la contraseña de administrador (Solo es necesario introducirla una vez). 5. Al pulsar aceptar si la contraseña de administrador es correcta se mostrará la actividad de gestión de botellas. 6. Llegado a este punto Pedro introduce los datos necesarios para crear una nueva botella y pulsa el botón guardar. 7. El usuario comprobará que la recién introducida botella aparece en el listado de la derecha.

Nombre del escenario	**Pedro modificar botella**
Descripción	El administrador Pedro quiere añadir una botella al inventario

Actores principales	Pedro es el gerente de un bar y propietario de la máquina dispensadora de cocteles.
Flujo de eventos	1. El usuario desbloquea el teléfono/Tablet que se encuentra junto a la máquina. 1. Se muestra la primera actividad la cual contiene un listado de bebidas, y un menú superior para llevar a cabo varias acciones. 2. Selecciona la acción Gestionar botellas. 3. Una vez pulsado aparecerá (si está activada la opción) un mensaje solicitando la contraseña de administrador (Solo es necesario introducirla una vez). 4. Al pulsar aceptar si la contraseña de administrador es correcta se mostrará la actividad de gestión de botellas. 5. Pedro seleccionará la botella de la lista de la derecha (lista de botellas disponibles) que desear modificar, al pulsarla la información se cargará para modificarla. 6. Llegado a este punto Pedro introduce los datos necesarios para crear una nueva botella y pulsa el botón guardar. 7. El usuario comprobará que la recién introducida botella aparece en el listado de la derecha.

Nombre del escenario	**Pedro añadir nueva receta**
Descripción	El administrador Pedro quiere añadir una nueva receta al

	inventario
Actores principales	Pedro es el gerente de un bar y propietario de la máquina dispensadora de cocteles.
Flujo de eventos	1. El usuario desbloquea el teléfono/Tablet que se encuentra junto a la máquina. 2. Se muestra la primera actividad la cual contiene un listado de bebidas, y un menú superior para llevar a cabo varias acciones. 3. Selecciona la acción Gestionar recetas. 4. Una vez pulsado aparecerá (si está activada la opción) un mensaje solicitando la contraseña de administrador (Solo es necesario introducirla una vez). 5. Al pulsar aceptar si la contraseña de administrador es correcta se mostrará la actividad de gestión de recetas. 6. Llegado a este punto Pedro pulsara el botón nueva receta, ahí se mostrará una pantalla para preguntar por el nombre de la receta. 7. Una vez introducido y si el nombre de la receta es correcto, se mostrará la actividad en la cual aparecen las botellas que se encuentran disponibles en el sistema. 8. El usuario irá eligiendo botella por botella y añadiendo la cantidad a dispensar (en porcentaje) y el orden en el cual deben ser servidas. 9. En la parte derecha podrá ir comprobando el estado de su receta.

Nombre del escenario	**Pedro modificar receta**

Descripción	El administrador Pedro quiere modificar una receta existente en el inventario
Actores principales	Pedro es el gerente de un bar y propietario de la máquina dispensadora de cocteles.
Flujo de eventos	1. El usuario desbloquea el teléfono/Tablet que se encuentra junto a la máquina. 2. Se muestra la primera actividad la cual contiene un listado de bebidas, y un menú superior para llevar a cabo varias acciones. 3. Selecciona la acción Gestionar recetas. 4. Una vez pulsado aparecerá (si está activada la opción) un mensaje solicitando la contraseña de administrador (Solo es necesario introducirla una vez). 5. Al pulsar aceptar si la contraseña de administrador es correcta se mostrará la actividad de gestión de recetas. 6. Llegado a este punto Pedro pulsara sobre la receta que desea modificar, posteriormente pulsará sobre el botón editar receta. 7. El usuario irá eligiendo botella por botella y añadiendo la cantidad a dispensar (en porcentaje) y el orden en el cual deben ser servidas. 8. En la parte derecha podrá ir comprobando el estado de su receta. 9. Es importante tener en cuenta que el porcentaje no puede superar el 100%

Nombre del escenario	**Pedro modificar configuración**

Descripción	El administrador Pedro quiere modificar la configuración de la aplicación
Actores principales	Pedro es el gerente de un bar y propietario de la máquina dispensadora de cocteles.
Flujo de eventos	1. El usuario desbloquea el teléfono/Tablet que se encuentra junto a la máquina. 2. Se muestra la primera actividad la cual contiene un listado de bebidas, y un menú superior para llevar a cabo varias acciones. 3. Selecciona la acción Ajustes (en última posición). 4. Una vez pulsado aparecerá (si está activada la opción) un mensaje solicitando la contraseña de administrador (Solo es necesario introducirla una vez). 5. Al pulsar aceptar si la contraseña de administrador es correcta se mostrará la actividad de gestión de botellas. 6. En esta actividad Pedro encontrará opciones para configurar tanto software como hardware de la maquina (colores de luces).

4.2.5. Diseño

El diseño del software es la primera fase de desarrollo de cualquier producto o sistema de la ingeniería, este concepto fue definido por Taylor en 1959 como “Proceso de aplicar distintas técnicas y principios con el propósito de definir un dispositivo, proceso o sistema con los suficientes detalles como para permitir su realización física”

El diseño del software es la primera fase de la construcción del sistema, por lo que es muy importante que se siga un procedimiento el cual ayude a construir una base sólida y bien definida de la solución.

El diseño de software al igual que otros métodos de diseño en otras ingenierías es muy cambiante, además es muy complejo hacerlo sin definir unas fases más simples y pequeñas de clarificar y definir, por ello el diseño de software consta de las siguientes fases: Fase de estructura del sistema, Fase de estructura de los datos, Fase de visualización de la aplicación.

4.2.6. Fase de estructura del sistema

En esta fase definiré las clases que intervendrán en el proyecto, para ello utilizaré un diagrama de clases.

4.2.7. Diagrama de clases

Un diagrama de clases es un tipo de diagrama de estructura estática que describe la estructura de un sistema mostrando las clases del sistema, sus atributos, operaciones (o métodos) y las relaciones entre los objetos.

- **Atributo:** un atributo es una especificación que define una propiedad de un objeto, también puede referirse o establecer el valor específico para una instancia determinada.
- **Visibilidad de los atributos:** para especificar la visibilidad de un miembro de la clase, se coloca uno de los siguientes signos delante.

+	**Público**
-	Privado
#	Estático
/	Derivado (se puede combinar con otro)
~	Paquete

Tabla 4.1: Tabla con visibilidades de atributos de una clase en notación UML.

https://es.wikipedia.org/wiki/Diagrama_de_clases

Métodos: es una subrutina cuyo código es definido en una clase, y puede pertenecer tanto a una clase, como es el caso de los métodos de clase o estáticos, como a un objeto, como es el caso de los métodos de una instancia.

El diagrama de clases que a continuación se muestra, muestra las diferentes clases utilizadas en el proyecto en sí mismo, algunas se refieren a objetos del mundo real, como son las clases botellas, recetas (que son un conjunto de botellas, con algunas características propias de las recetas), item_recetas, que es la clase que hace posible la unión de botellas con recetas.

Otras clases como son BotellasMaquina, Carro, Peso, Iluminación etc., son clases encargadas de gestionar la comunicación y la lógica de las funciones que se pueden enviar a la maquina física, para eso todas ellas se apoyan en la clase comunicación la cual es una clase que se encarga de gestionar la comunicación por Bluetooth con cualquier microcontrolador de una manera genérica, la cual acepta valores dobles, y posee la funcionalidad de esperar además a una retroalimentación del lado de la máquina física.

Las clases Receta, botellas e item_receta se encargan básicamente de la gestión de la información, teniendo en cuenta el proceso ETL (en inglés “Extract, Transform, Load”), lo que significa que valida los inputs de los usuarios, y en caso de ser erróneos muestra un mensaje y no guarda datos que sean “incorrectos” (por ejemplo, los ingredientes de un coctel no pueden sumar más del 100% del total, pues eso es una inconsistencia en la lógica de cálculo de la receta), los transforma al formato en el cual se van a almacenar en la base de datos SQLite (mencionada

en el siguiente epígrafe), por ejemplo la tabla de recetas se compone de ids botellas y cantidades, por lo que para una persona es imposible entender que significan los datos que contiene la tabla.

Con respecto a la letra “L” de load, las clases se encargan de recuperar la información de la base de datos y mostrarla en el formato correcto utilizando la lógica correcta, uniendo los diferentes orígenes (botellas, recetas, ítems) cuando es necesario.

El objetivo de las clases Avisos, Error e Historial, es la gestión de intercambio de información y almacenamiento de resultados.

En sí mismo un aviso es una abstracción de un mensaje el cual puede darse para informar de algún suceso que ha ocurrido, al tratarse la comunicación de una comunicación asíncrona, pues la máquina tarda un tiempo en realizar las acciones que se han mandado), cuando la acción termina de parte de la máquina se recibe un aviso.

Las clases Historial e Error, sirven para gestionar diferentes historiales de la máquina, de manera simple, la historial crear una bitácora de las dispensaciones que han ocurrido de manera satisfactoria, teniendo en cuenta la hora de dispensación, y el nombre de la receta que se dispensó.

Por su parte la clase Error, mantiene una bitácora con diferentes errores que se conocen de antemano (por ejemplo, dispensación no posible después de x intentos, porque la válvula de líquido está atascada u otro motivo), además si ocurre algún error desconocido y la máquina lo detecta, esta clase se encarga de registrar el suceso y los detalles que se conocen del mismo (algún tipo de aviso de la máquina).

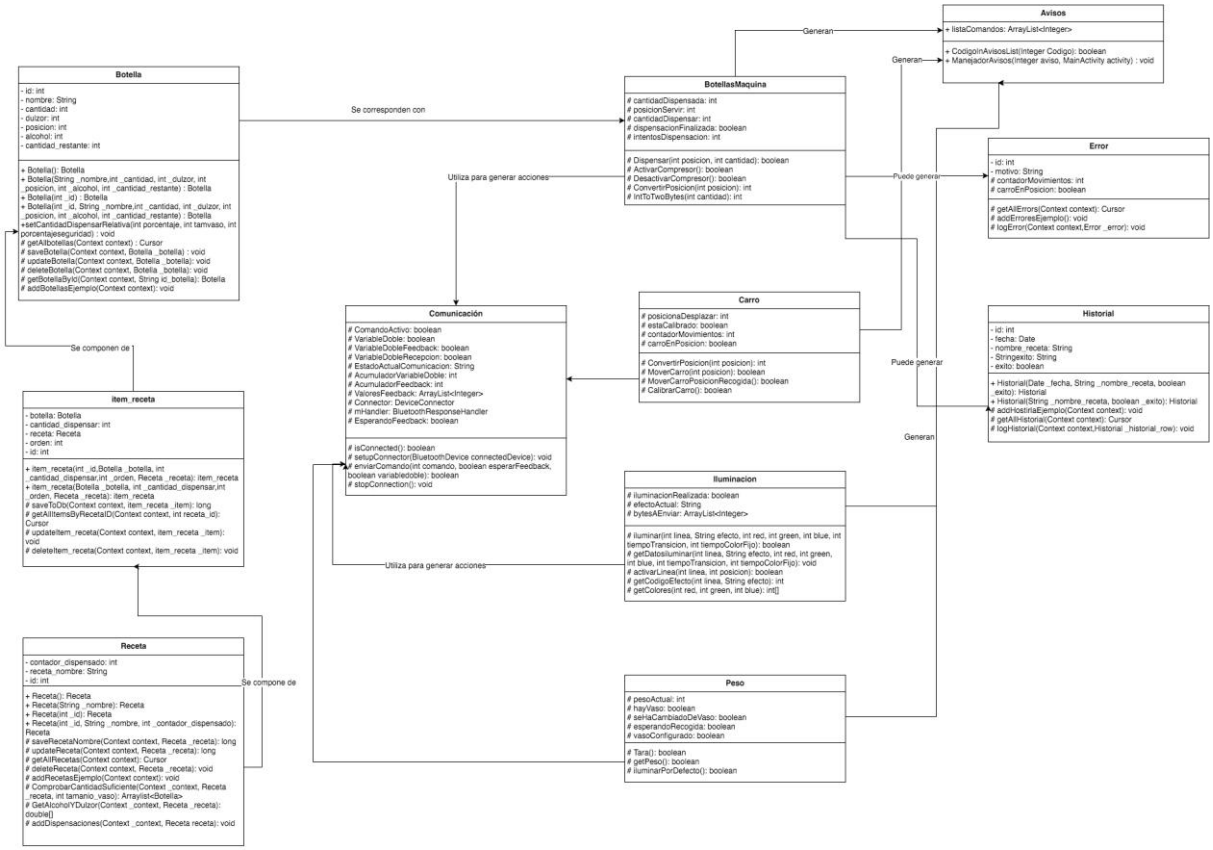


Imagen 4.7: Diagrama de clases.

4.2.8. Diseño de datos

El diseño de datos transforma el modelo del campo de información, que se ha creado durante el análisis, en las estructuras que se van a requerir para el diseño de software, a estas estructuras las llamaré clases, puesto que el software será escrito utilizando la metodología de programación orientada a objetos, la cual se compone de clases y objetos.

4.2.9. Entidad relación

Un modelo entidad relación es una herramienta para el modelado que permite representar las entidades relevantes de un sistema de información así como sus interrelaciones y propiedades.

Entidad: Representa una cosa o un objeto, del mundo real con existencia independiente, es decir, se diferencia únicamente de otro objeto o cosa, incluso siendo del mismo tipo, o una misma entidad.

Una entidad puede sr un objeto con existencia física como una persona, o un objeto con existencia conceptual como puede ser un historial de acciones, una entidad está descrita y se representa por sus características o atributos, por ejemplo la entidad Persona tiene las características: Nombre, Apellido, Genero, Estatura, Peso etc.

Atributos: Los atributos son las características que definen o identifican una identidad. Estas pueden ser muchas y el diseño solo utiliza o implementa las que son necesarios o se va a hacer uso de ellas.

En un conjunto del mismo tipo, cada entidad tiene valores específicos asignados para cada uno de sus atributos, de estar forma es posible su identificación unívoca.

Nótese que dos o más entidades diferentes pueden tener los mismos valores para algunos de sus atributos pero no para todos.

Relación: Describe cierta dependencia entre entidades o permite la asociación de las mismas. Para representarlas en los diagramas de entidad relación, se utiliza la figura geométrica de un rombo, y para unirlas se utilizan líneas hacia las entidades.

Otros elementos muy importantes dentro de los diagramas de entidad relación son las Claves.

Claves: Son un tipo de atributo especial el cual permite identificar de manera inequívoca a cada uno de los objetos que se contendrán en cada entidad, por ejemplo si tenemos la entidad Persona, el DNI podría ser una clave primaria, pues en España el DNI es único.

Las claves deben seguir algunas características para considerarse claves, además existen diferentes tipos de claves:

- **Superclave / Clave primaria:** Es un subconjunto de atributos dentro de la entidad que la distinguen cada una de las entidades de un conjunto de entidades. Para indicarla en el diagrama se utiliza el nombre del atributo subrayado.
- **Clave candidata:** Dada una superclave, si esta desapareciese o dejara de serlo al eliminar uno o más atributos de los que la componen esta pasaría a ser la superclave, pues es una clave igualmente valida, pero que no ha sido elegida en primera instancia para serlo.

En el siguiente diagrama se muestran las diferentes entidades que se dan en el proyecto.

- **Botellas:** La entidad botellas se refiere a cada una de las botellas físicas que pueden ocupar la máquina, ya sean con o sin gas, para cada una de ellas se guardan una serie de características como son:
  - **Nombre:** Nombre de la botella (por ejemplo, Coca Cola)
  - **Cantidad:** Cantidad de la botella, en ml (por ejemplo, 2000ml)
  - **Cantidad_restante:** Cantidad restante en la botella, este dato es muy importante pues la aplicación creara un aviso de que no se puede dispensar el coctel si la cantidad disponible no es suficiente, además esta cantidad variara con las dispensaciones por lo que irá disminuyendo en consecuencia.
  - **Dulzor:** Se refiere al nivel de dulzor de la misma, este dato es utilizado para crear una idea posteriormente al cliente de lo dulce que es un coctel, dependiendo de la cantidad que se dispense de esta botella en una determinada receta y la cantidad de dulzor de la botella, se calcula un dulzor ponderado y se muestra en una gráfica, junto al nivel de alcohol.
  - **Posición:** La máquina se rige por posiciones es necesario que la aplicación mantenga las posiciones en las que se encuentran las botellas, pues a la hora de dispensar tiene que accionar las válvulas de la posición en la cual se encuentra la botella, y además debe desplazar el carro para situar el vaso justo debajo a la posición correcta.

    - **Alcohol:** Nivel de alcohol que aparece en la botella, al igual que el dulzor utilizando la cantidad que se dispensará de la misma y el grado de alcohol de la botella, se mostrará un gráfico del nivel de alcohol en general que tiene la receta
    - **Id:** Identificador auto numérico creado por la base de datos, que se utiliza para crear índices e identificar de manera inequívoca una botella determinada.
- **Recetas:** Es la entidad que se encarga de almacenar las recetas, las recetas no son más que una lista de botellas y una cantidad a dispensar de cada una de ellas, no es necesario nada más puesto que en la propia botella se almacenan todas las características de la misma y la receta en sí misma. Además la entidad recetas llevara un control de cuantas veces ha sido dispensada la receta (por motivos de control).
- **Historial:** Entidad encargada de guardar un historial de las dispensaciones realizadas de cada una de las recetas, su objetivo principal es tener una idea de que recetas se han dispensado, utilizando esta información es posible saber si una receta es más popular que otra o si en un determinado día se han producido picos de uso etc.
- **Error:** Algunas dispensaciones acaban de manera errónea, debido a un problema de hardware o de administración por parte del administrador de la maquina (botella vacía etc.), esta entidad guardará cada uno de los errores teniendo en cuenta fecha y motivo del suceso, para posteriormente poder analizar los resultados y determinar cual pudo ser el origen del problema y si es necesario llevar a cabo acciones para solventarlo.

Normalización

El proceso de normalización en las bases de datos es un proceso que se lleva a cabo para simplificar la manera en la que se van a configurar y representar cada una de las entidades, para llevarla a cabo es necesario aplicar las siguientes reglas:

- Evitar redundancia de datos.
- Reducir los problemas de actualización de los datos en las tablas.
- Proteger la integridad de los datos.

En el modelo relacional se utilizan tablas para representar las relaciones que existen entre entidades, aunque esta debe de cumplir una restricción:

- Cada tabla debe de tener un nombre único.
- No es posible repetir filas completas (cada tabla debe contener una clave única).
- Los datos para una columna siempre serán del mismo tipo (en bases de datos relacionales, existen otro tipo de bases de datos llamadas no SQL que esto no tiene por qué cumplirse).

Lo que se denomina formas normales es aplicar a las tablas de la base de datos.

Una base de datos es un conjunto de tablas normalizadas en su forma normal N.

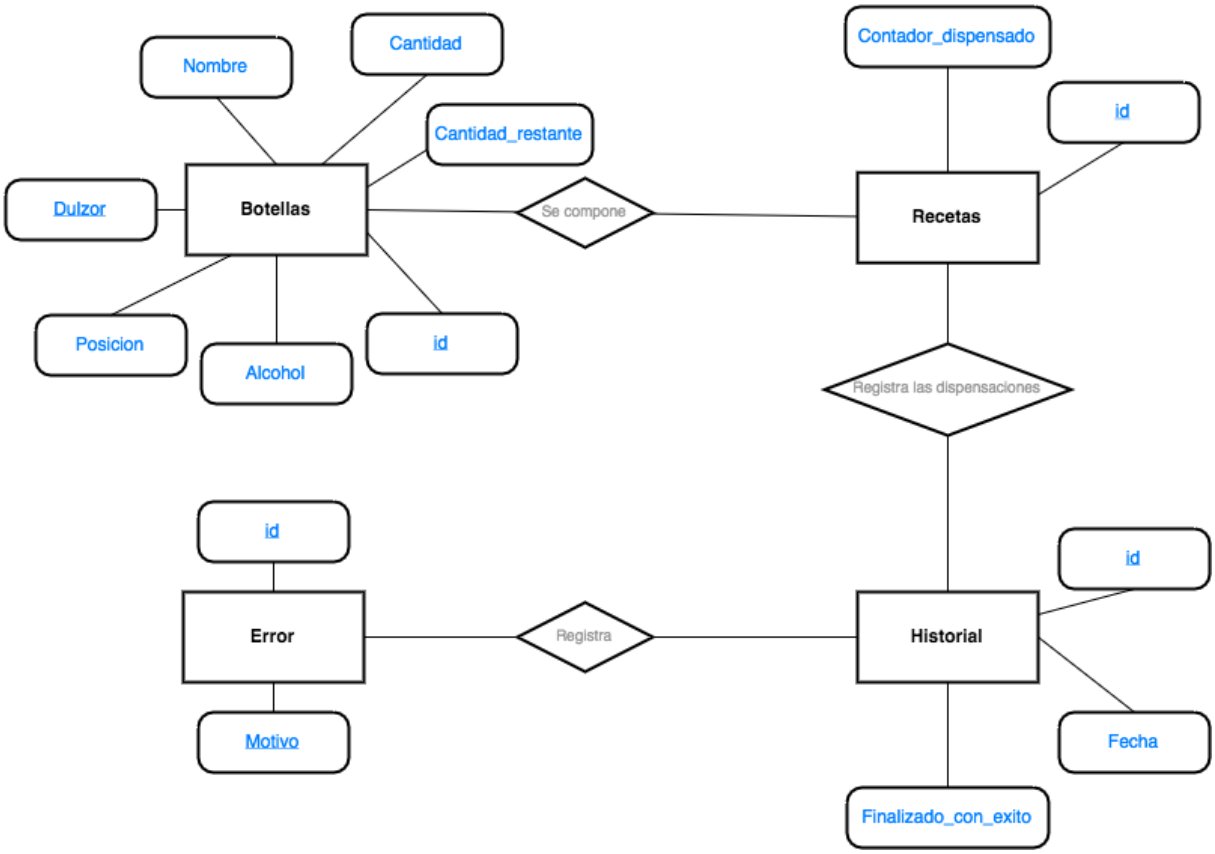
Las tres primeras formas normales son suficientes para cubrir las necesidades de casi todas las aplicaciones, el creador de las formas normales es Edgar F. Codd.

Las formas normales que enunció fueron las siguientes:

- **Primera forma normal (1FN):** Se dice que una tabla está en su forma normal 1FN si:
  - Todos sus atributos son atómicos, un atributo es atómico si los elementos del dominio son simples e indivisibles.
  - La tabla contiene una clave primaria única.
  - La clave primaria no contiene atributos nulos.
  - No existe variación entre el número de columnas.
  - Los campos no clave deben identificarse por la clave (Dependencia funcional)
  - Debe existir una independencia del orden tanto de las filas como de las columnas, si los datos cambian de orden, no deben cambiar su significado.
- **Segunda forma normal (2FN):** Una tabla está en forma normal 2FN si está en forma 1FN y los atributos que no forman parte de ninguna clave dependen completamente de la clave principal, esto significa que todos los atributos que no sean clave principal deben depender solo de una clave principal.
- **Tercera forma normal (3FN):** Una tabla está en forma normal 3FN si es 2FN y además no existen dependencias funcionales transitivas entre los atributos que no son clave.

Esquema conceptual

Una vez transformados los componentes en entidades y relaciones este es el resultado trasladado a un esquema entidad relación.



.

Imagen 4.8: Diagrama entidad relación de la base de datos.

Tablas de la base de datos

Una vez que dispongo de los componentes, del diagrama, sus relaciones y características, el siguiente paso es definir la estructura en la que se traducirá en base de datos.

Además es necesario especificar el tipo de dato que se va a utilizar para cada uno de los atributos, también hay que tener en cuenta que las relaciones dependiendo de la cardinalidad, se convertirán en tablas.

TABLA BOTELLAS			
CAMPO	**TIPO**	**CLAVE**	**DESCRIPCION**
ID	Auto numérico (Integer)	PRIMARY	Identificador único para cada botella, asignado automáticamente
Cantidad	Integer		Cantidad que tiene la botella en mililitros.
Dulzor	Integer		Cantidad de dulzor del 1 al 10 (medida escalar).
Posición	Integer		Posición donde se sitúa la botella en la máquina.
Alcohol	Integer		Nivel de alcohol en % que tiene la botella, este valor se puede encontrar en la botella física.
Cantidad restante	Integer		Cantidad restante que queda en la botella, para poder calcular si hay suficiente liquido como para poder dispensar una receta.

TABLA RECETAS			
CAMPO	**TIPO**	**CLAVE**	**DESCRIPCION**
ID	Auto numérico (Integer)	PRIMARY	Identificador único para cada receta, asignado automáticamente
Nombre	Varchar(250)		Nombre de la receta.
Contador dispensado	Integer		Cantidad de veces que se ha dispensado la receta, a cada dispensación este valor se incrementa en uno, al principio de la creación de la receta está a 0.

TABLA RECETAS_BOTELLAS			
CAMPO	**TIPO**	**CLAVE**	**DESCRIPCION**
ID	Auto numérico (Integer)	PRIMARY	Identificador único para cada relación entre botellas y recetas, asignado automáticamente
ID Botella	Integer	FOREIGN_KEY	Identifica la botella clave externa de la tabla botellas.
ID Receta	Integer	FOREIGN_KEY	Identifica la receta clave externa de la tabla receta.
Cantidad	Integer		Cantidad a dispensar de la botella seleccionada.
Orden	Integer		Orden en el cual la botella debe ser dispensada.

TABLA ERROR			
CAMPO	**TIPO**	**CLAVE**	**DESCRIPCION**
ID	Autonumerico (Integer)	PRIMARY	Identificador único para cada error, asignado automáticamente
Motivo error	Text		Texto identificando el motivo del error.

TABLA HISTORIAL			
CAMPO	**TIPO**	**CLAVE**	**DESCRIPCION**
ID	Autonumerico (integer)	PRIMARY	Identificador único para cada registro en el historial, asignado automáticamente
Fecha	Date		Fecha de cuando se registró el registro del historial.
Nombre Receta	Varchar(250)		Nombre de la receta del historial, no se utiliza una clave externa pues la receta puede dejar de existir y el historial debe guardar incluso datos de recetas que no existan en la actualidad.
Finalizado con éxito	Boolean		Booleano que indica si la dispensación se realizó con éxito o no.

4.2.10. Diseño de la interface

Una vez que se ha definido, explicado y diseñado todo lo referente a la estructura de la que se compone todo el sistema, la estructura de almacenamiento, tratado y recuperación de datos, pasaré a desarrollar el siguiente punto de la Ingeniería del Software, el diseño de la interfaz.

El diseño de la interfaz es un paso clave, puesto que es el punto de unión entre el usuario y el sistema, el usuario interactuará con el sistema a través de la interfaz, por lo que esta debe ser, fácilmente aprendible (utilizando determinados recursos que se mencionaran más adelante), fácilmente recordable e intuitiva.

Para crear la interfaz se utilizaran los widgets (cada uno de los elementos en la pantalla) que provee Android, los cales son customizables, pero hay que ser cuidadoso a la hora de modificarlos, pues puede que el usuario ya tenga algunos conceptos y formas de utilizar las aplicaciones Android aprendidas y al modificarlas se vea afectado el grado de usabilidad o facilidad al interactuar con la misma.

Para la correcta creación de una interfaz en Android, Google, la empresa propietaria del mismo ofrece una guía de estilo que detallare a continuación.

Guía de estilo

A continuación voy a definir la guía de estilo que seguirá mi programa, es importante definir una guía de estilo, puesto que si posteriormente hay que hacer una modificación, por otro programador (o incluso por el que desarrollo la aplicación originariamente) debemos mantener una consistencia en la utilización de la aplicación y en el estilo.

Qué es una Guía de estilo

Es un documento que recoge normativas y patrones básicos relacionados con el "aspecto" de un interfaz para su aplicación en el desarrollo de nuevas "pantallas", en Android llamadas “Actividades” dentro de un entorno concreto.

Es un documento que recoge normativas y patrones básicos relacionados con el "aspecto" de un interfaz para su aplicación en el desarrollo de nuevas "pantallas" dentro de un entorno concreto. (Sitio web de contenidos, nuevas secciones, entorno de aplicaciones de negocio).

Conceptualmente, el interfaz de usuario descansa en 3 puntos:

- **Significado**: es la base del interfaz. Recoge el contenido o información de la pantalla. Textos, campos de formularios, botones, menús...
- **Comportamiento**: trata el funcionamiento del interfaz. Cómo se comporta cuando un usuario envía un formulario (validaciones), hace clic en un enlace...
- **Aspecto**: apariencia final de un sistema: colores, tipografía, disposición de los elementos en pantalla (layout).

Android define una guía de buenas prácticas a seguir para que sus aplicaciones, de esta manera Android se asegura de que todas sus aplicaciones tienen un aspecto similar y que los usuarios automáticamente saben entender cada objeto que haya en la pantalla y como utilizarlo.

Esta aplicación sigue las siguientes sugerencias de la guía de estilo Android:

Android define en su guía de estilo, algunos patrones a seguir, en la aplicación se utilizan los siguientes [67]:

- **Fechas:** Android define la manera correcta de mostrar fechas**,** en la sección donde se pueden visualizar los historiales de dispensación y errores, se siguen las reglas que se definen.

- **Permisos:** Android define como mostrar un mensaje cuando la aplicación necesita algún permiso de un usuario, este patrón se aplica a la hora de pedir permiso para conectarse a la máquina, permiso para continuar si el vaso no está presenta etc.
- **Ajustes:** Android define como deben ser los ajustes de una aplicación, la aplicación sigue los patrones en el apartado ajustes del menú principal.
- **Menú:** El menú se muestra en el panel superior utilizando el icono […], ese icono hace que todo el mundo pueda entender fácilmente donde está en el menú.

Una vez definido que es una hoja de estilos y que información contiene en su interior, pasare a describir los estilos que son utilizados en la aplicación que se ha desarrollado para este Proyecto.

- **Fuentes:** describen el texto que aparece en las actividades (interfaz).
  - **Títulos:** describen los textos en los encabezados de una sección
    - Tipo de letra (fuente): Sans de la familia Droid
    - Tamaño: 22sp
    - Color: Negro
    - Formato: negrita
  - **Texto explicativo:** describen los textos que explican que introducir en un input de texto
    - Tipo de letra (fuente): Sans de la familia Droid
    - Tamaño: 18sp
    - Color: Negro (para descripciones) / Gris para hits (ayudas dentro de los cuadros de texto, inputs)
    - Formato: regular
- **Menús:** describen el menú que se encuentra en la actividad principal, para seleccionar las acciones a llevar a cabo.
  - Opciones: describen cada una de las opciones que muestra el menú.
    - Tipo de letra (fuente): Sans de la familia Droid
    - Tamaño: 14sp
    - Color: Negro
    - Formato: regular
- **Listas:** describen todas las listas que se encuentran en la aplicación, como por ejemplo lista de botellas, lista de recetas etcétera. Cada lista se compone de dos elementos, uno principal y otro secundario, que se encuentra en la parte inferior del elemento.

- Elementos no seleccionados:
  - Elemento principal:
    - Tipo de letra (fuente): Sans de la familia Droid
    - Tamaño: 14sp
    - Color: Gris oscuro
    - Formato: negrita
    - Fondo: transparente
  - Elemento secundario:
    - Tipo de letra (fuente): Sans de la familia Droid
    - Tamaño: 14sp
    - Color: Gris claro
    - Formato: regular
    - Fondo: transparente
- Elementos seleccionados:
  - Elemento principal:
    - Tipo de letra (fuente): Sans de la familia Droid
    - Tamaño: 14sp
    - Color: Gris oscuro
    - Formato: negrita
    - Fondo: color azul (#3366ff)
  - Elemento secundario:
    - Tipo de letra (fuente): Sans de la familia Droid
    - Tamaño: 14sp
    - Color: Gris claro
    - Formato: regular
    - Fondo: transparente (#3366ff)
- **Botones:** describen elementos en forma de botón que al pulsarse realizan una acción.
    - Tipo de letra (fuente): Sans de la familia Droid
    - Tamaño: 14sp
    - Color: Negro
    - Formato: negrita
    - Fondo: Gris
- **Fondos:** se refiere a los fondos de cada una de las actividades
  - Todas las actividades:
    - Imagen:
    - 

- **Logotipo:** Icono o imagen que representa la aplicación en el sistema y en la parte superior de todas las pantallas de la aplicación, además la distingue de manera fácil y rápida del resto de aplicaciones.

Metáforas

La palabra metáfora significa la aplicación de una palabra o de una expresión a un objeto o a un concepto, al cual no lo denota literalmente, con el fin de sugerir una comparación (con otro objeto o concepto) y facilitar su comprensión, por ejemplo cuando hablamos de escritorio del ordenador, se está sugiriendo una metáfora con un escritorio físico, en el cual tenemos documentos, papeles etc.

Aplicándolo al Diseño de Interacción, la metáfora juega una parte primordial a la hora de crear y diseñar la experiencia que el usuario tendrá cuando utilice la aplicación. “Las metáforas en la interfaz son una técnica ampliamente utilizada sin embargo suponen un punto de controversia entre los diseñadores, en cuanto no aseguran el correcto funcionamiento de una interfaz. Incluso, en algunos casos, pueden perjudicar el correcto entendimiento de la interfaz”.

En algunos casos, si es correctamente aplicada, puede favorecer la interacción humano-aplicación, entregándole al usuario el modo de pensar las cosas, para que él pueda deducir el comportamiento de los elementos. Sin embargo, en otros caso, puede llegar incluso a entorpecer el flujo de información entre el usuario y el contenido.

Kay Hofmeester y Dennis Wixon, afirman que el uso de metáforas en sus procesos de diseño se debe a dos razones; para crear un mundo de interfaz de usuario que sea entendible y predecible para nuestros usuarios, y para guiar al equipo en la creación de diseño detallado.

Para estos diseñadores, la metáfora no sólo aparece en el momento en que el usuario interactúa con el objeto, sino que además constituye una herramienta indispensable en el trabajo colaborativo y el diseño de nuevos productos que permitan la correcta comunicación entre el usuario y la plataforma con la que interactúa.

Por otra parte, autores como Donald Norman plantean una visión diametralmente opuesta. En The Invisible Computer, Normar afirma que "las metáforas son un intento para usar una cosa para representar otra, cuando la otra no es lo mismo. Pero si no es lo mismo, ¿cómo puede ayudar la metáfora?".

Una vez que ha sido definido que, como y porque se utilizan metáforas explicare donde y como he utilizado las metáforas dentro de la aplicación:

- **Botones**: Los botones dan la sensación de que son pulsables, al colocar el dedo encima del botón cambia de color y da la sensación de que está siendo pulsada, este recurso viene por defecto con Android, el cual hace que todo el mundo que haya utilizado alguna vez un dispositivo con Android sepa distinguir perfectamente donde se puede pulsar y donde no.
- **Botones con iconos**: Son botones que contienen un icono que a simple vista es entendible, ya sea porque es un símbolo estándar, o porque existe una similitud con la realidad:
    - **Bluetooth**: Este botón utiliza la simbología estándar  , todos los usuarios que tengan conocimientos básicos, conocerán ese símbolo.
    - **Vaso**: Se trata de un icono especialmente diseñado para indicar los diferentes estados que puede tener el vaso en la maquina
        - **No hay vaso**: 
        - **Vaso en posición**: 

Storyboards

Se trata de un conjunto de ilustraciones ordenadas en una secuencia lógica, que sirven como guía y referencia para entender el flujo o funcionamiento de una aplicación.

En el contexto de una aplicación móvil estas sirven para saber la secuencia de pantallas por las cuales tiene que navegar el usuario y donde tiene que interactuar para hacer cada determinada cosa.

Para detallar la transición entre pantallas se utilizarán flechas, las cuales definirán los Caminos de Navegación.

Los storyboards están estrechamente relacionados con los escenarios definidos en el capítulo anterior, se trata de representar gráficamente cada uno de los escenarios.

Los storyboards son ampliamente utilizados porque son muy útiles para que el usuario, o cliente pueda evaluar cómo va a ser la aplicación final, puesto que en este punto es sencillo introducir cambios en el diseño, recordemos que esta fase aun el software no está implementado, pues en este punto el proyecto se encuentra en fase de diseño, en la cual es relativamente barato hacer modificaciones.

Los storyboards que voy a representar en el proyecto son los mismos que se han desarrollado en los escenarios, los cuales son:

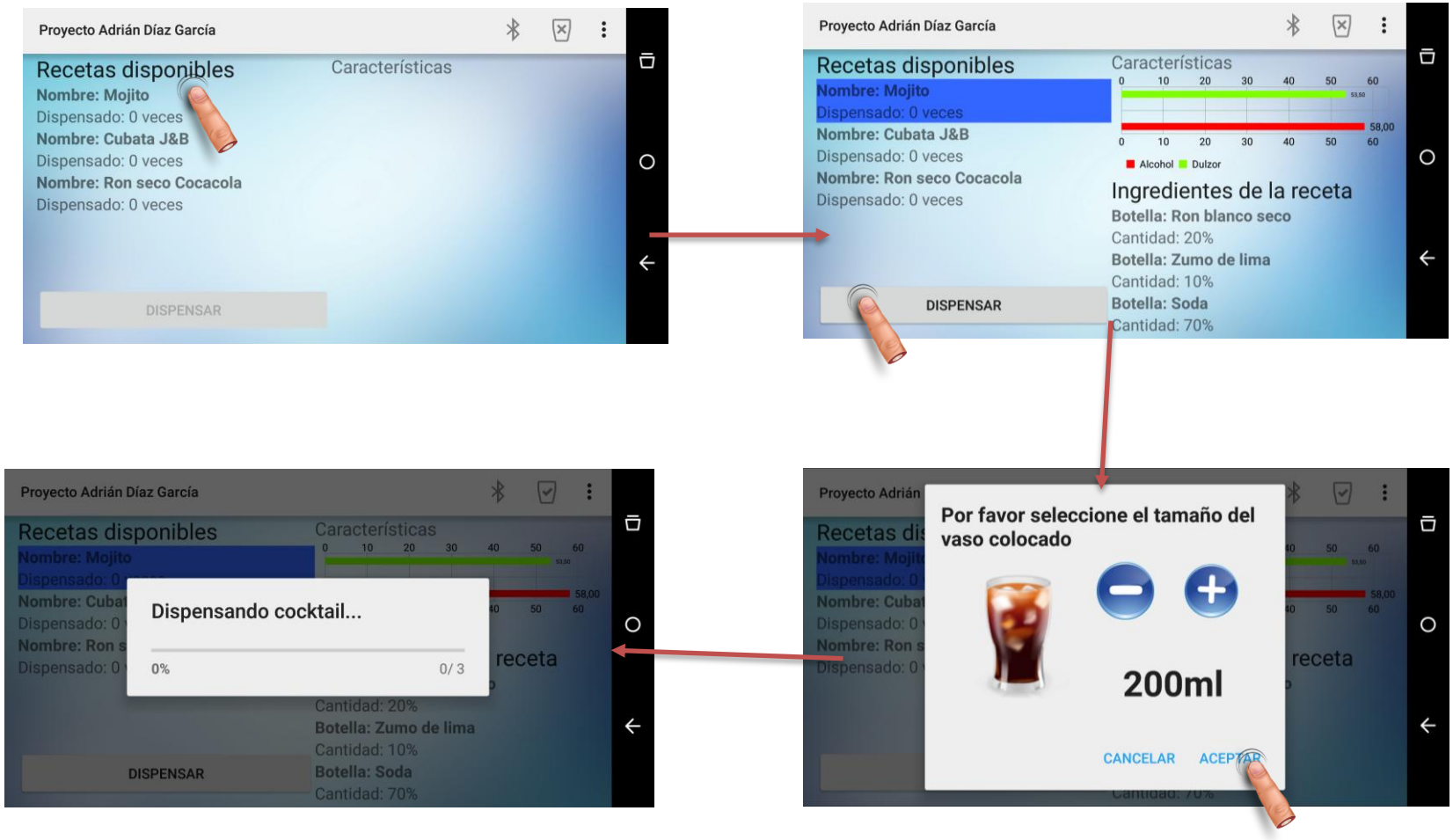
- Dispensación Mojito
- Modificar configuración
- Modificar receta
- Añadir nueva receta
- Modificar botella
- Añadir botella

Además se van a desarrollar las siguientes capturas de pantalla aparte de los que describen los escenarios:

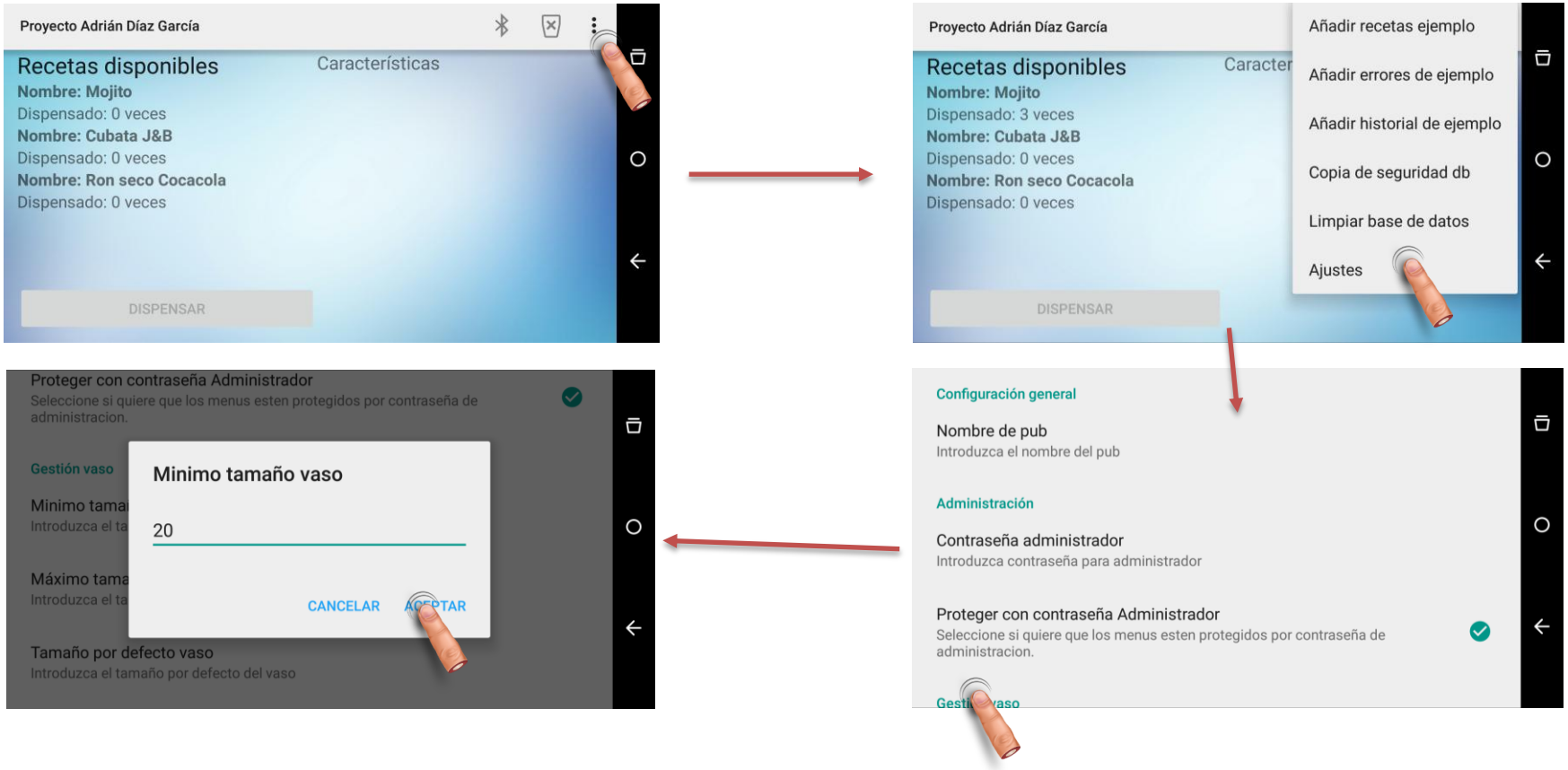
- Conexión bluetooth
- Ver historial
- Identificar como administrador

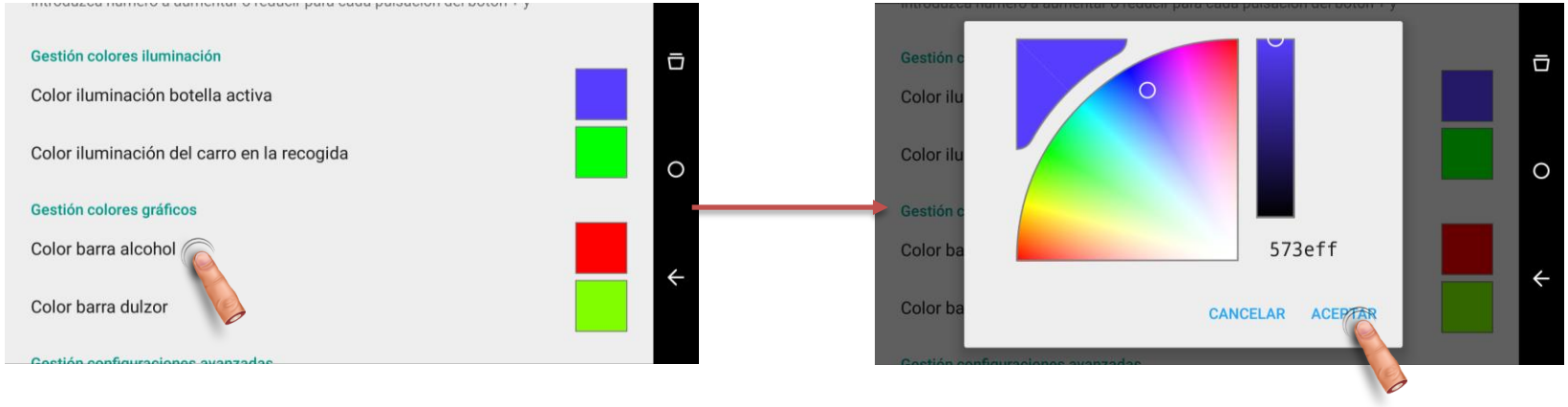
El icono  se utilizará para definir una acción llevada a cabo por el usuario.

Capturas dispensación de mojito.

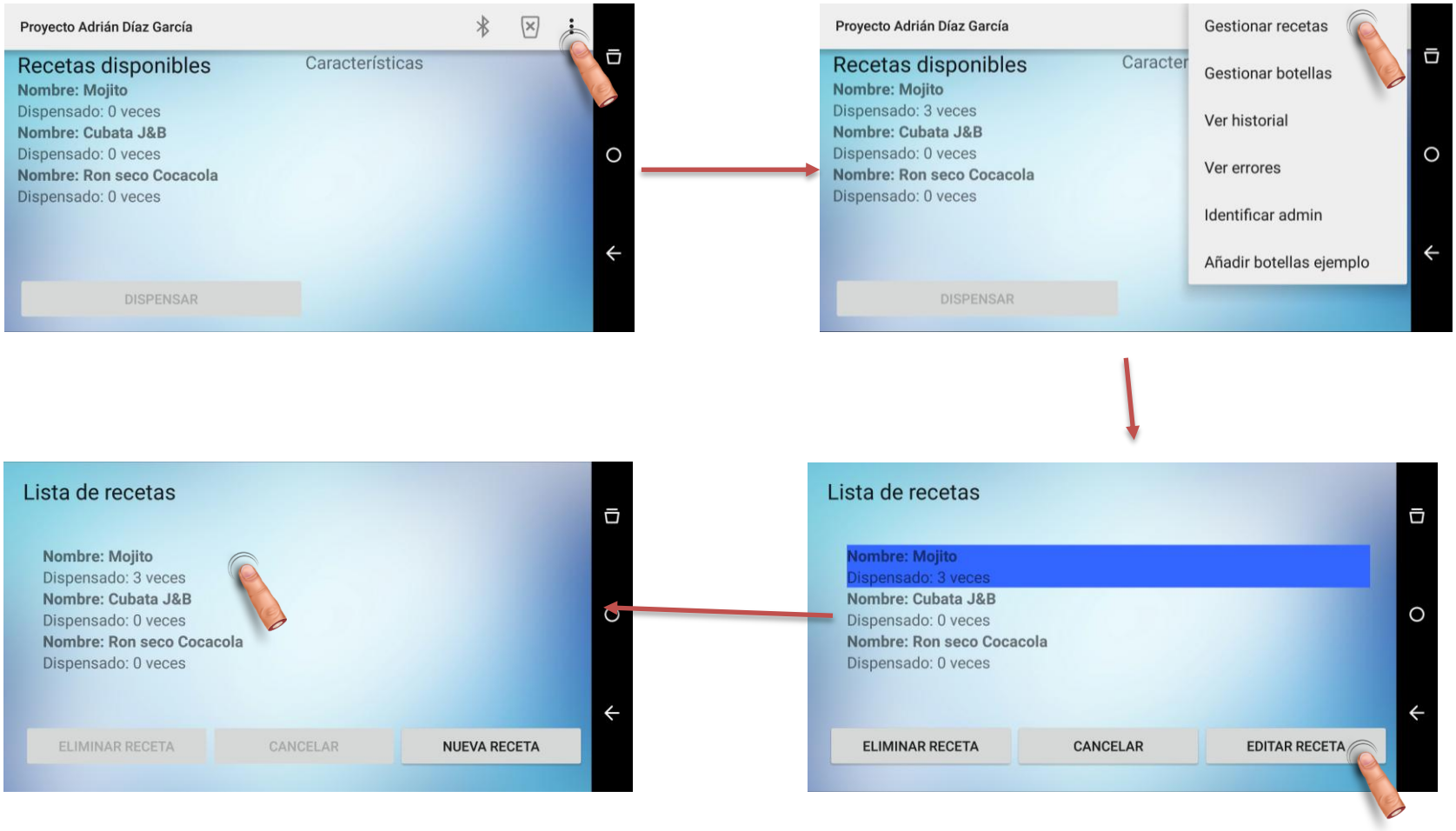


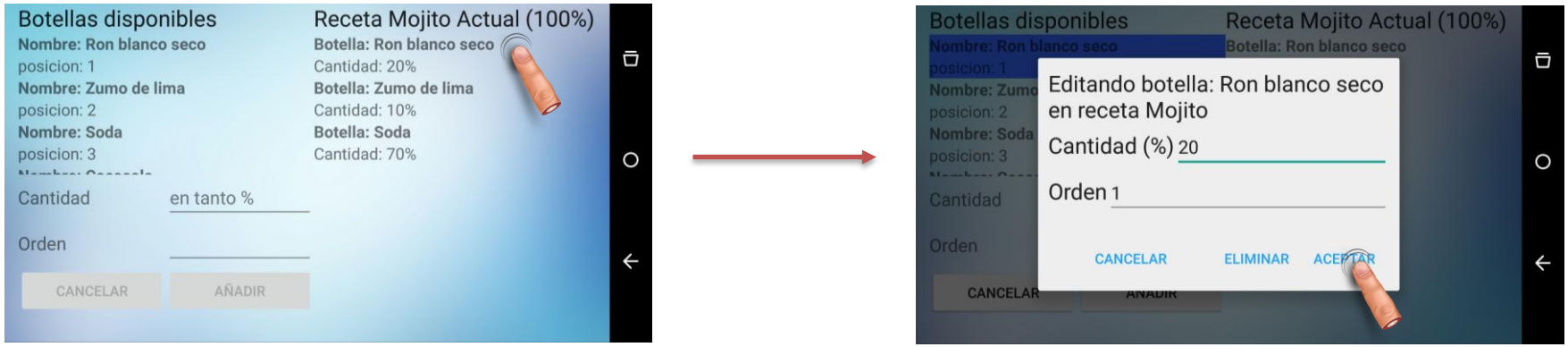
Capturas modificar configuración.



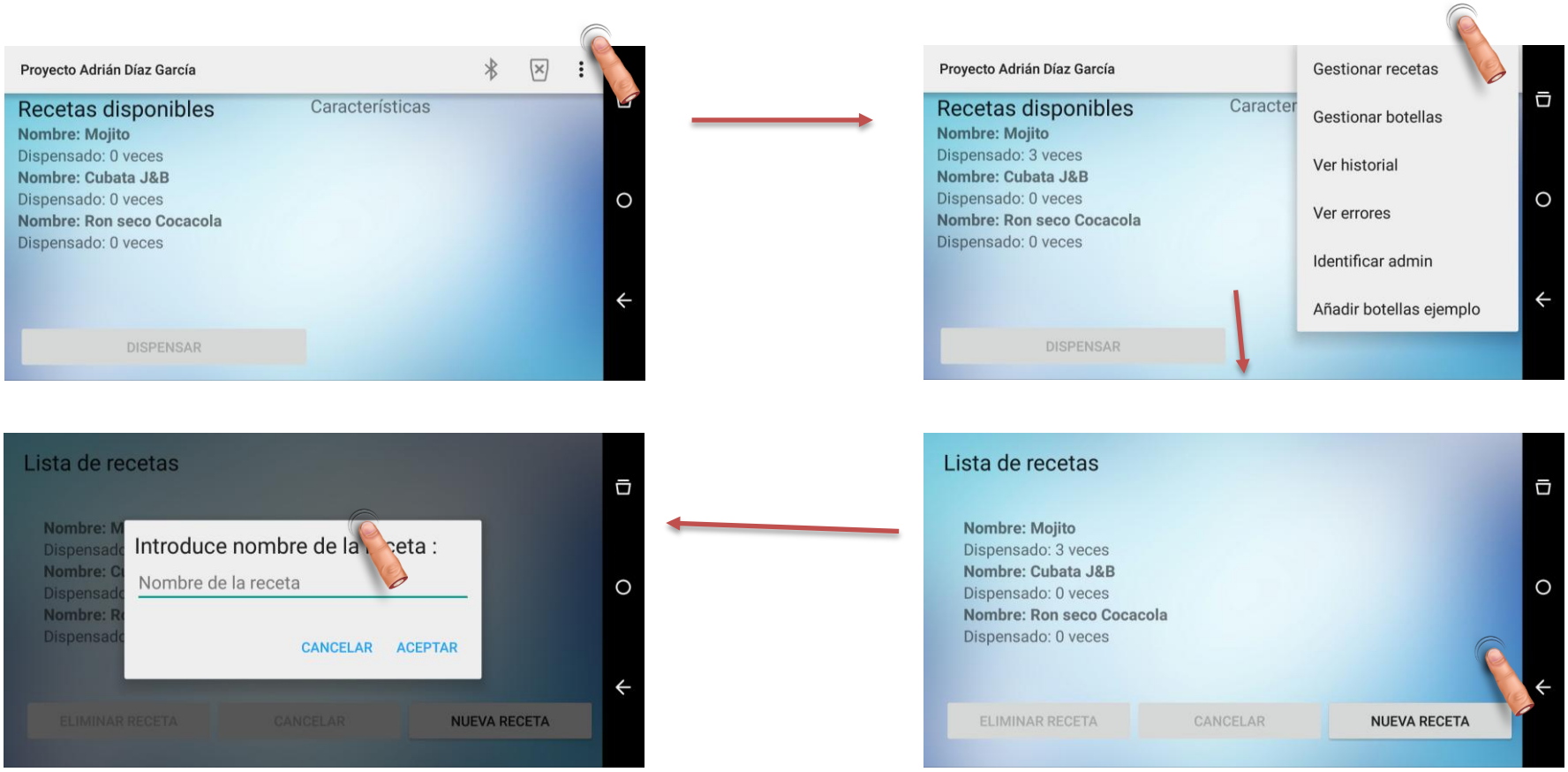

Gestión colores iluminación
Color iluminación botella activa
Color iluminación del carro en la recogida
Gestión colores gráficos
Color barra alcohol
Color barra dulzor
573eff
CANCELAR
ACEPTAR

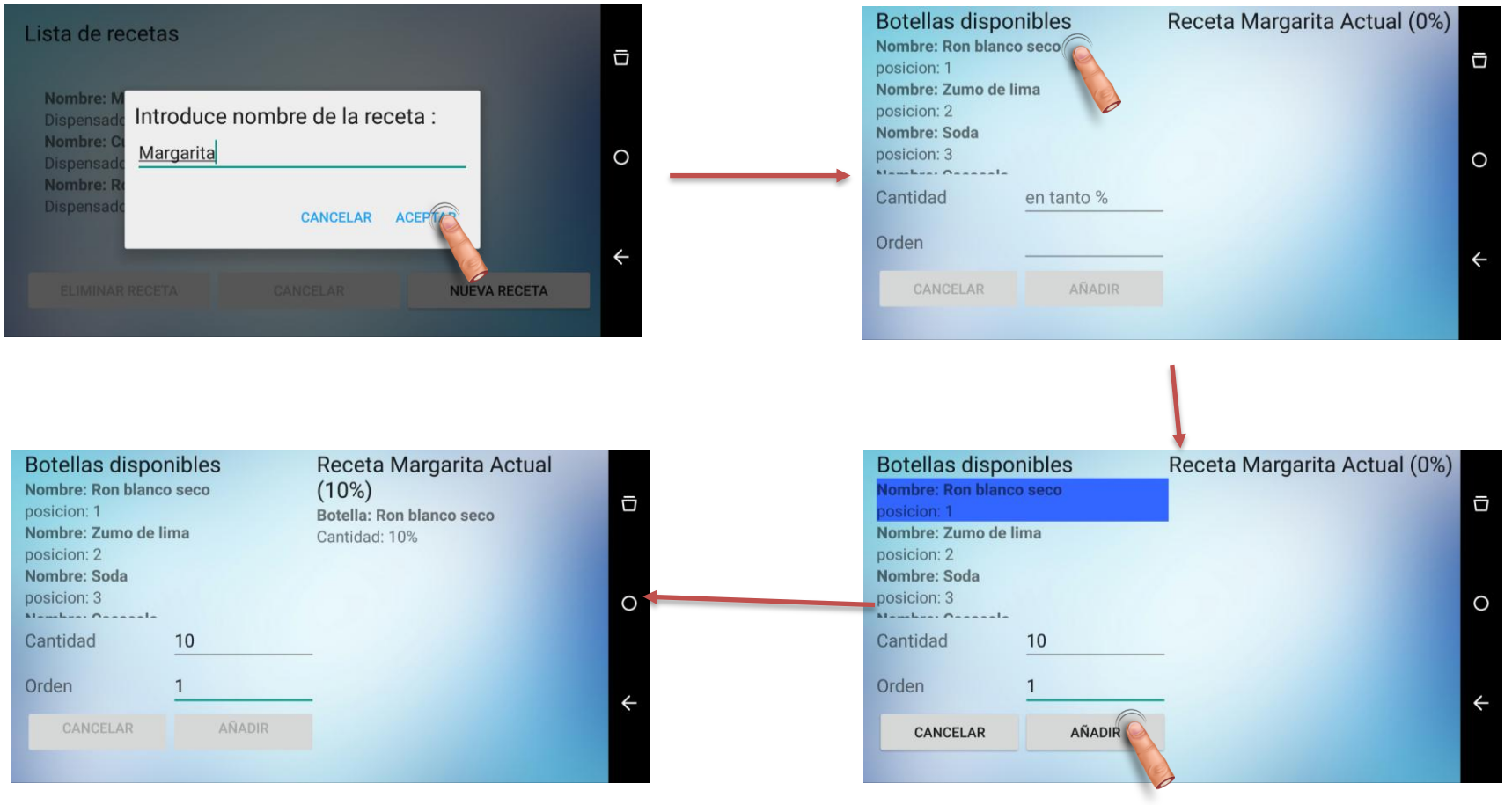
Capturas modificar receta.

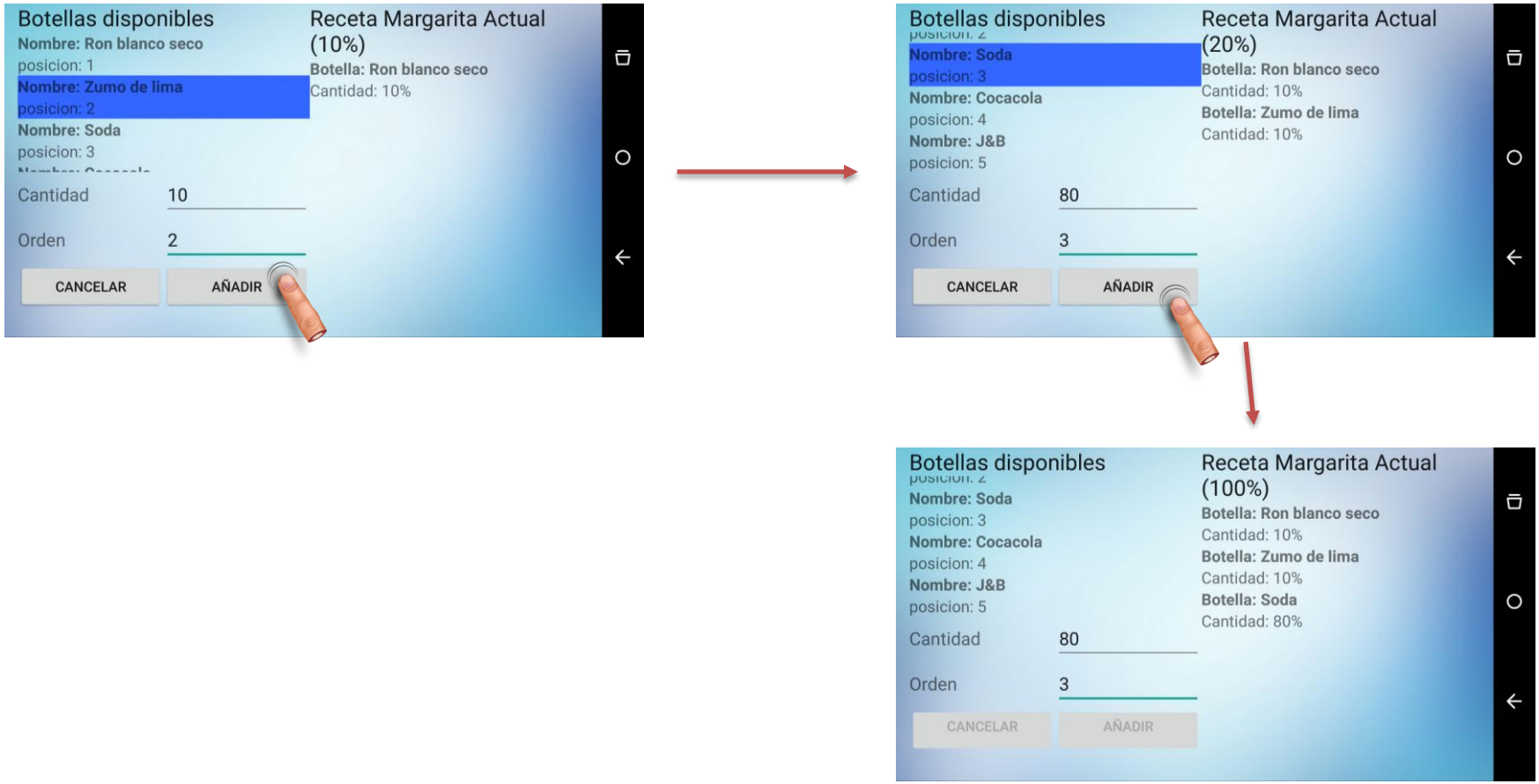



Botellas disponibles
Nombre: Ron blanco seco
posicion: 1
Nombre: Zumo de lima
posicion: 2
Nombre: Soda
posicion: 3
Receta Mojito Actual (100%)
Botella: Ron blanco seco
Cantidad: 20%
Botella: Zumo de lima
Cantidad: 10%
Botella: Soda
Cantidad: 70%
Cantidad
en tanto %
Orden
CANCELAR
AÑADIR
Editando botella: Ron blanco seco en receta Mojito
Cantidad (%) 20
Orden 1
CANCELAR
ELIMINAR
ACEPTAR

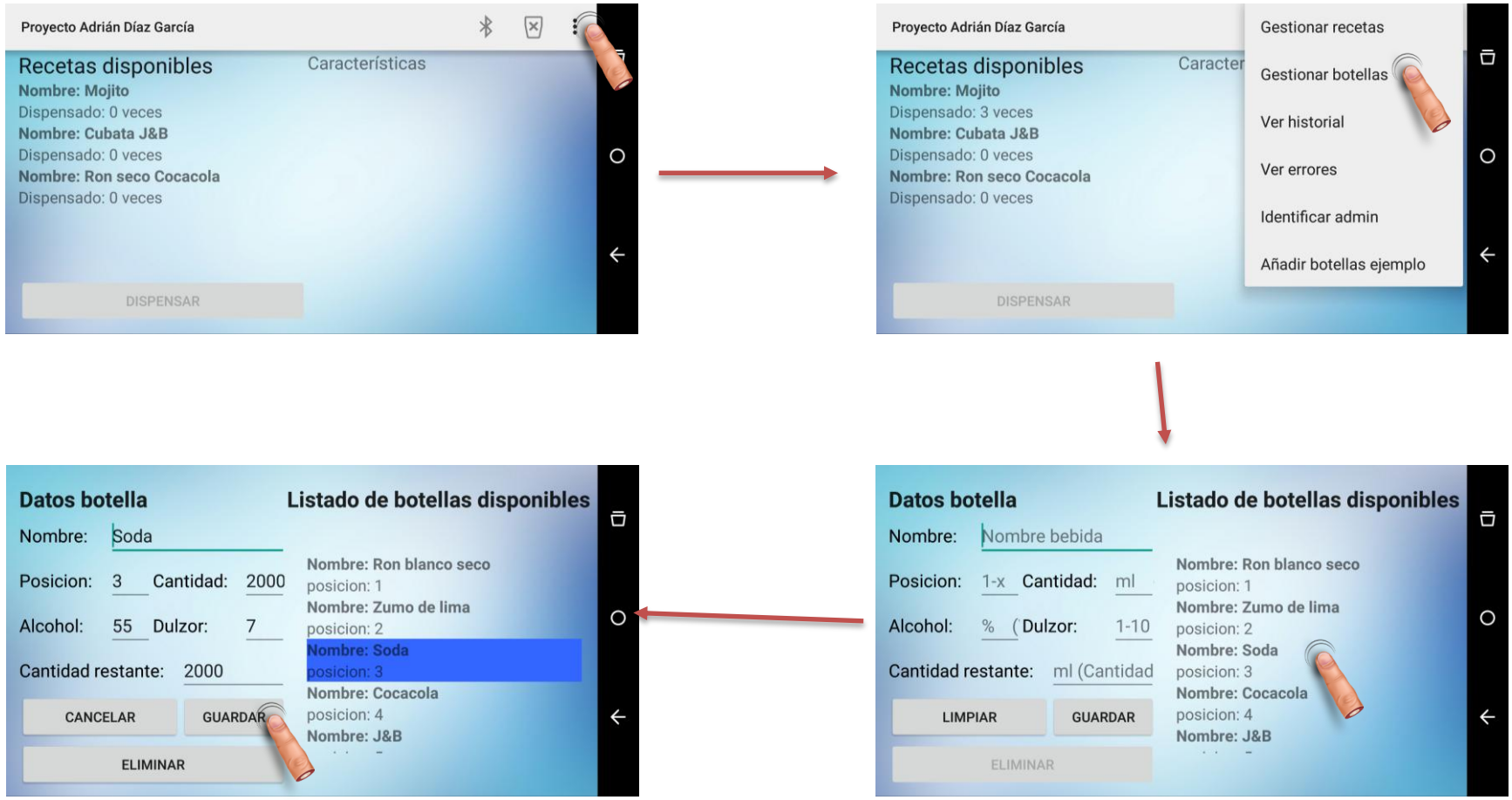
Capturas añadir nueva receta.



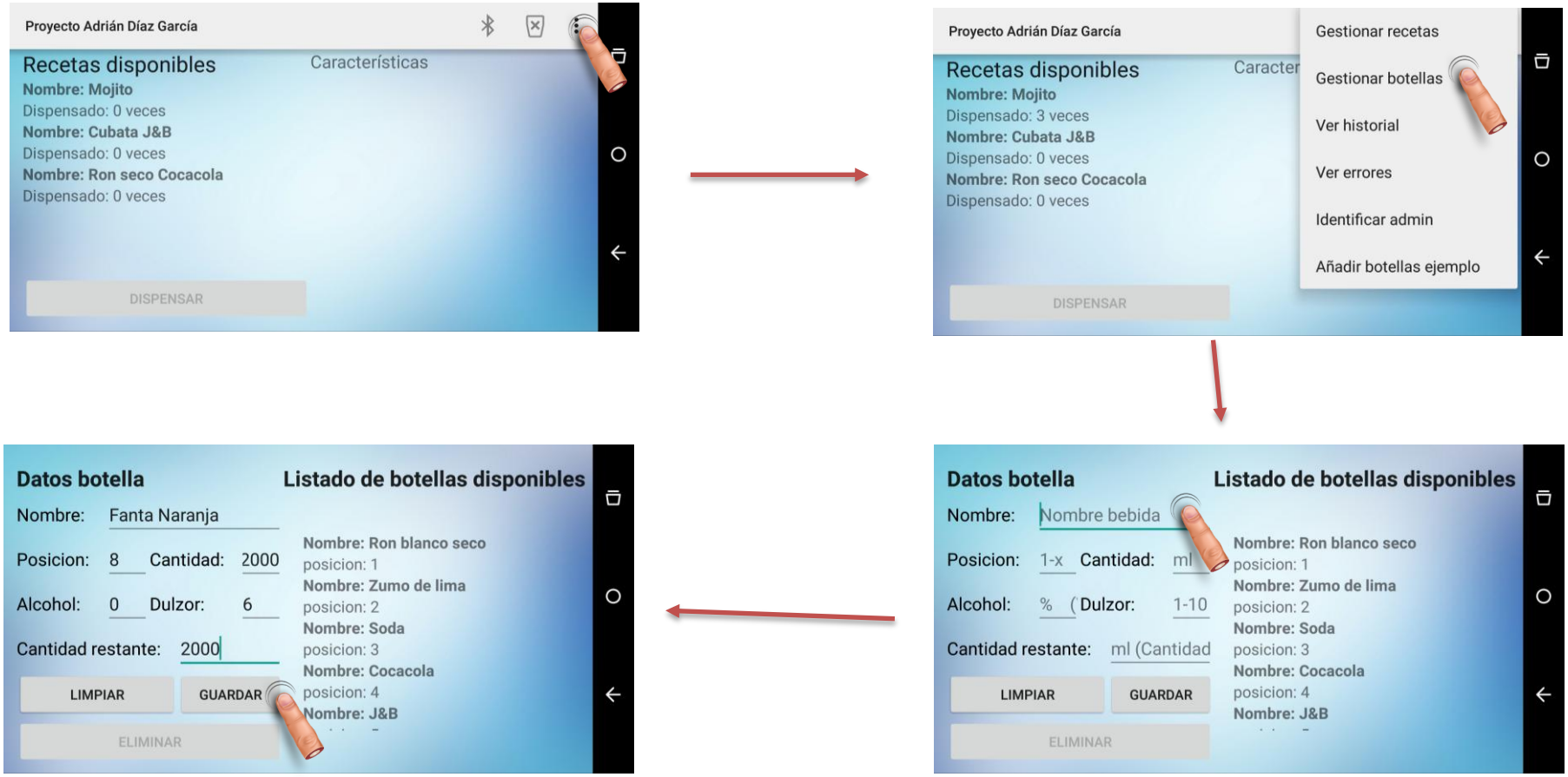

Lista de recetas
Introduce nombre de la receta :
Margarita
CANCELAR
ACEPTAR
ELIMINAR RECETA
CANCELAR
NUEVA RECETA
Botellas disponibles
Nombre: Ron blanco seco
posicion: 1
Nombre: Zumo de lima
posicion: 2
Nombre: Soda
posicion: 3
Receta Margarita Actual (0%)
Cantidad
en tanto %
Orden
CANCELAR
AÑADIR
Botellas disponibles
Nombre: Ron blanco seco
posicion: 1
Nombre: Zumo de lima
posicion: 2
Nombre: Soda
posicion: 3
Receta Margarita Actual (0%)
Cantidad
10
Orden
1
CANCELAR
AÑADIR
Botellas disponibles
Nombre: Ron blanco seco
posicion: 1
Nombre: Zumo de lima
posicion: 2
Nombre: Soda
posicion: 3
Receta Margarita Actual (10%)
Botella: Ron blanco seco
Cantidad: 10%
Cantidad
10
Orden
1
CANCELAR
AÑADIR


Botellas disponibles
Nombre: Ron blanco seco
posicion: 1
Nombre: Zumo de lima
posicion: 2
Nombre: Soda
posicion: 3
Receta Margarita Actual (10%)
Botella: Ron blanco seco
Cantidad: 10%
Cantidad 10
Orden 2
CANCELAR
AÑADIR
Botellas disponibles
Nombre: Soda
posicion: 3
Nombre: Cocacola
posicion: 4
Nombre: J&B
posicion: 5
Receta Margarita Actual (20%)
Botella: Ron blanco seco
Cantidad: 10%
Botella: Zumo de lima
Cantidad: 10%
Cantidad 80
Orden 3
CANCELAR
AÑADIR
Botellas disponibles
Nombre: Soda
posicion: 3
Nombre: Cocacola
posicion: 4
Nombre: J&B
posicion: 5
Receta Margarita Actual (100%)
Botella: Ron blanco seco
Cantidad: 10%
Botella: Zumo de lima
Cantidad: 10%
Botella: Soda
Cantidad: 80%
Cantidad 80
Orden 3
CANCELAR
AÑADIR

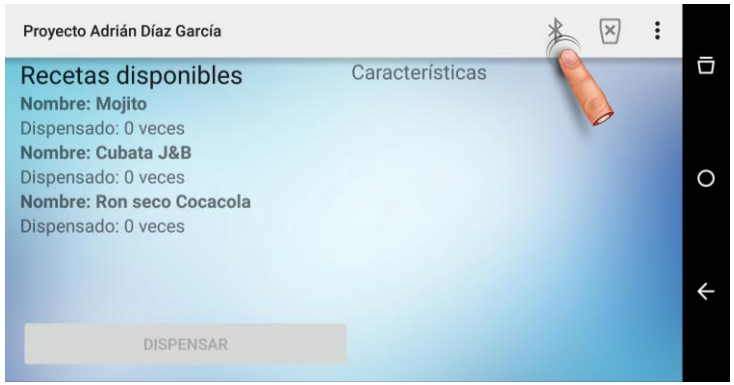
Capturas modificar botella.

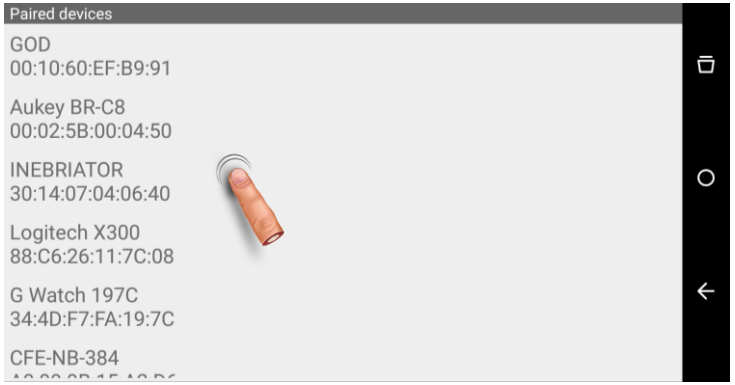


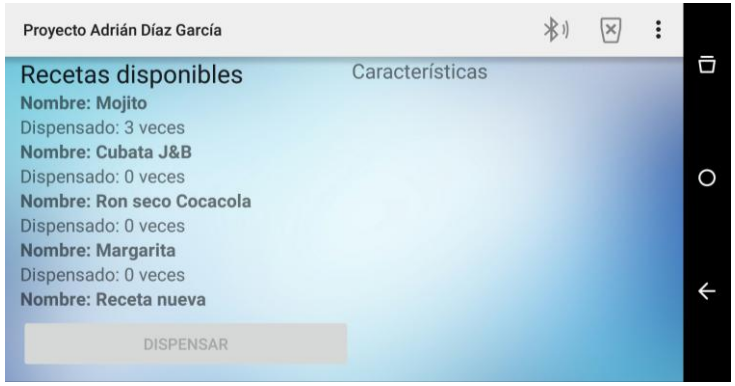
Capturas añadir botella.



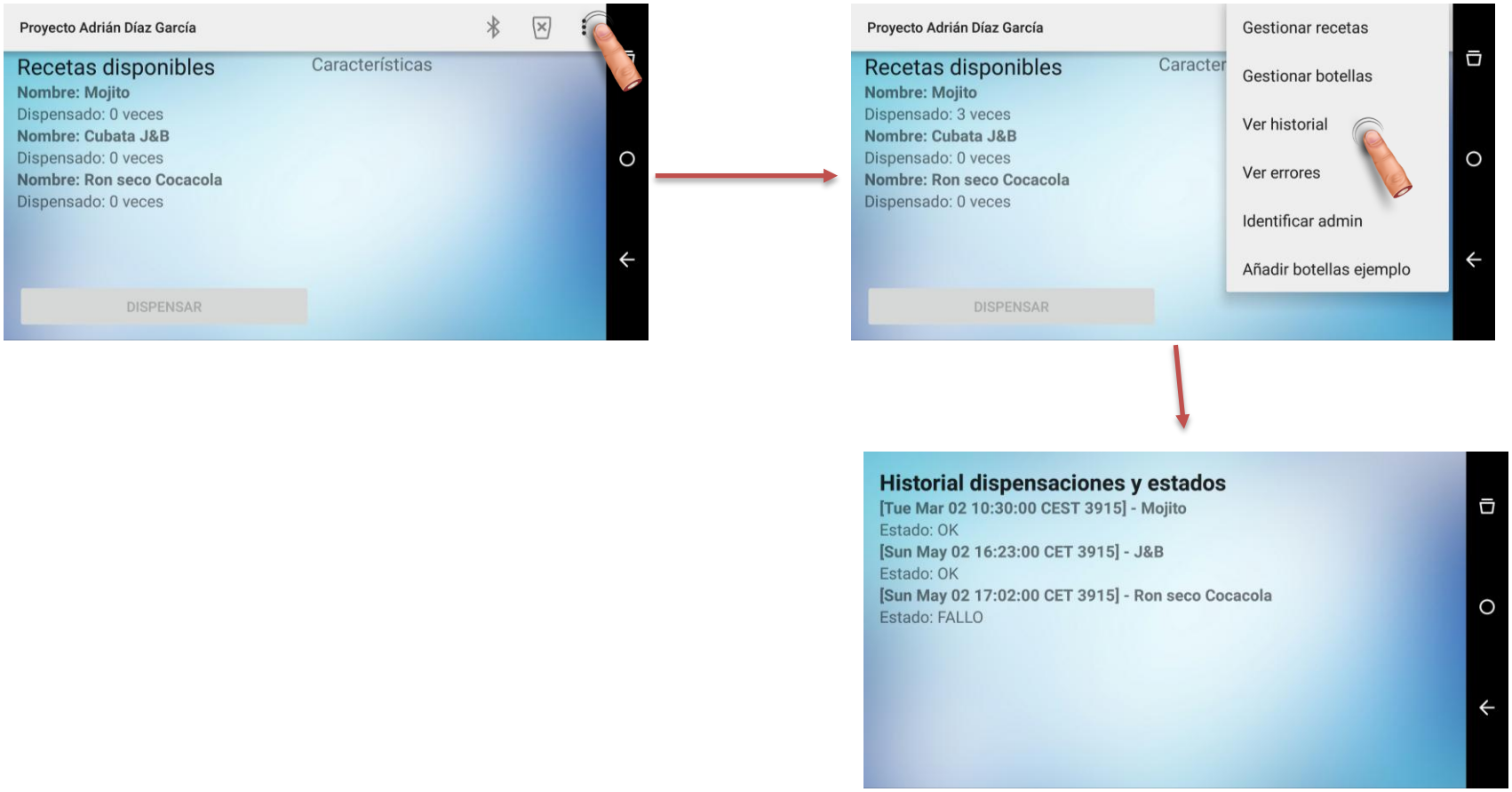
Capturas Conexión bluetooth.



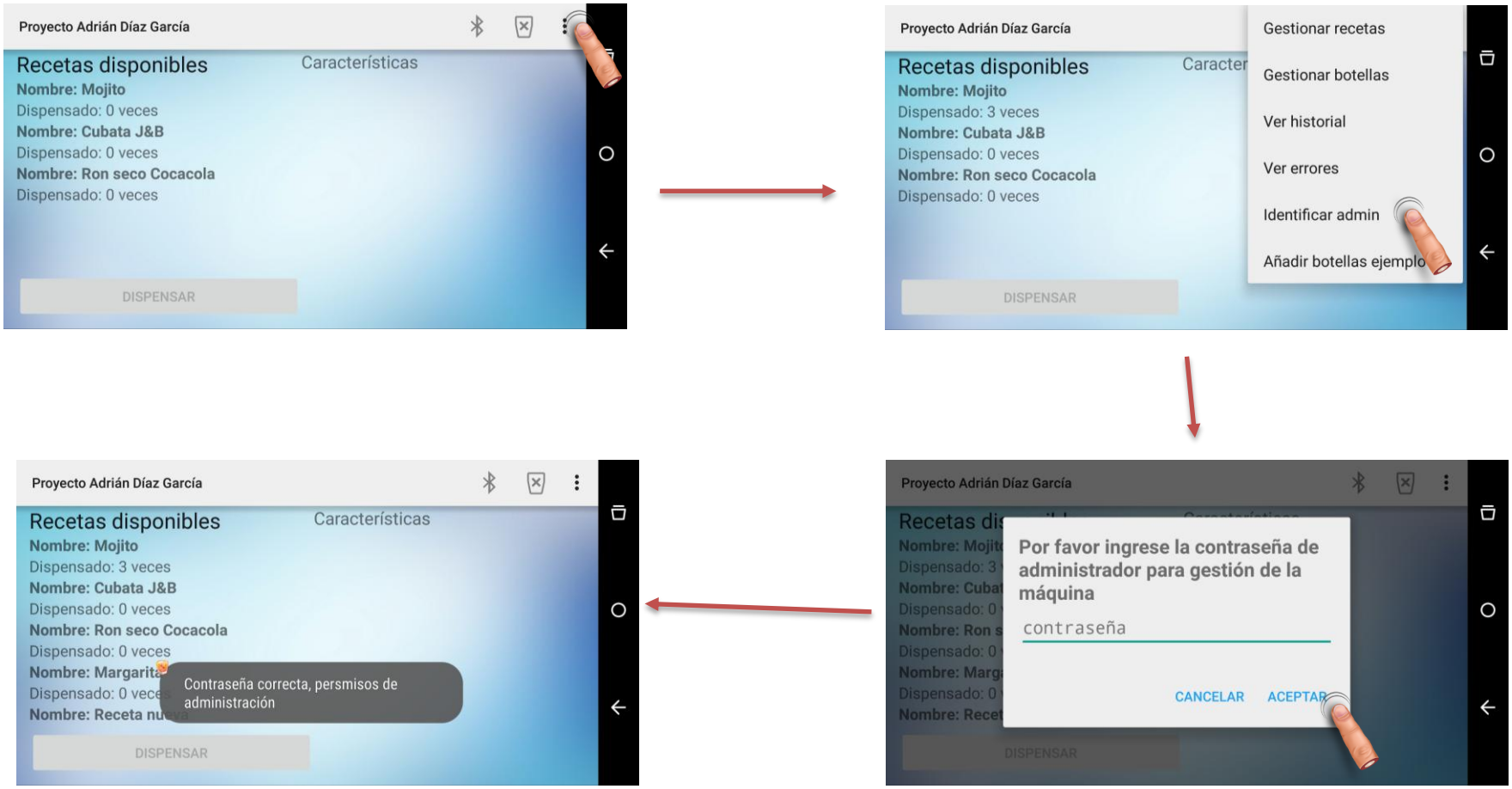




Capturas ver historial.



Capturas identificarse como administrador.



4.2.12. Implementación

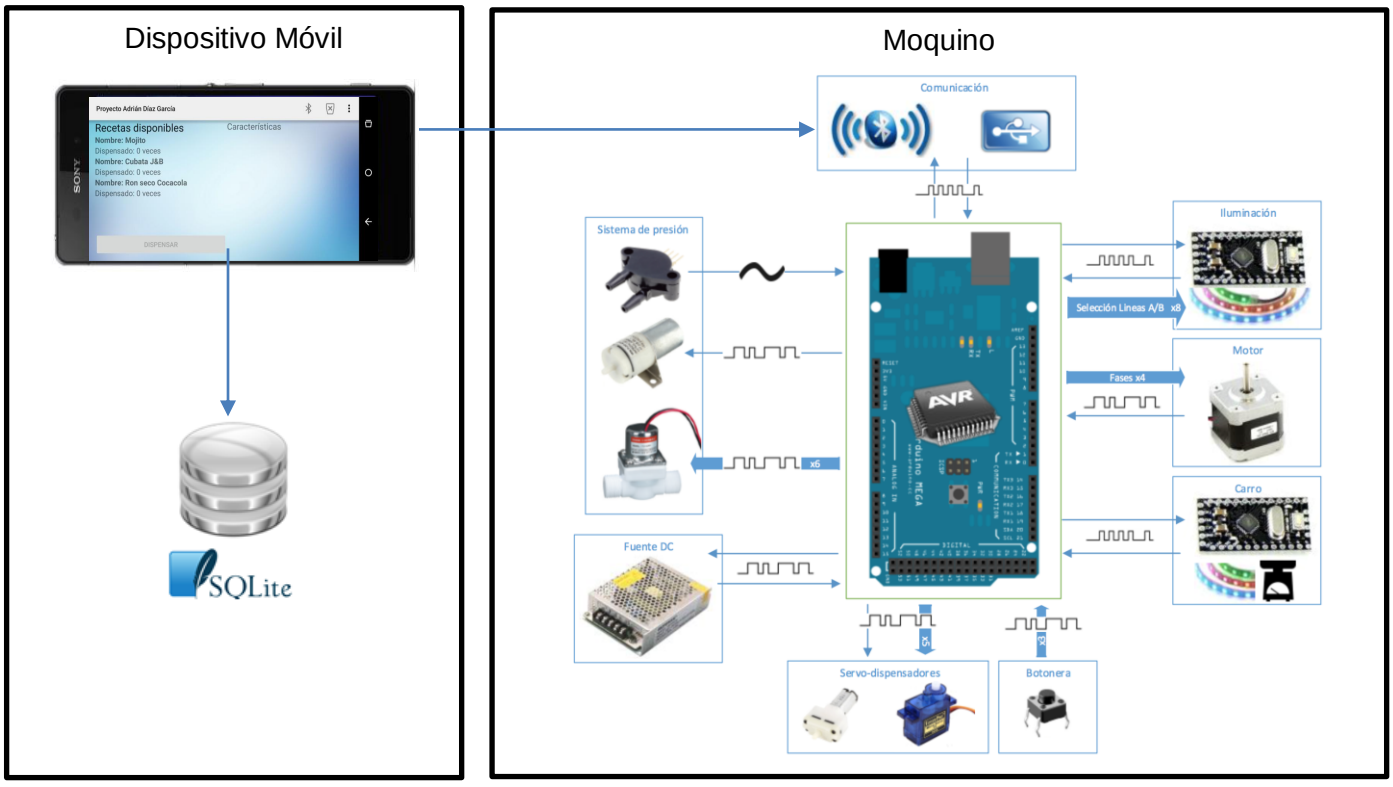
En esta parte de la ingeniería del software se lleva a cabo la implementación de todo el sistema que ha sido diseñado y especificado en las fases anteriores, implementación por lo tanto significa traducir a código, funciones y rutinas en un lenguaje de programación determinado, capaz de ejecutarse donde se supone que ha de hacerlo.

Arquitectura del sistema

En este epígrafe llevare a cabo una pequeña explicación de cómo se estructura todo el sistema y de cómo esta interrelacionado entre sí.

Existen dos dispositivos, conectados (Tablet/teléfono y Arduino), a través de una conexión bluetooth, y para ello utilizan el lenguaje desarrollado en el epígrafe Lenguaje de comunicación entre la máquina y el dispositivo móvil.

Con este tipo de arquitectura todos los datos tanto de configuración de la maquina como de recetas, botellas etc., este tipo de arquitectura propicia una futura actualización de la aplicación con idea de poder monetizar en el futuro la aplicación restringiendo la creación de recetas a un sistema externo y poder vender paquetes de bebidas posteriormente.



Imágen 4.9: Diagráma de arquitectura del sistema

Lenguajes de programación

Al existir dos dispositivos en los que se ejecuta código se utilizan varios lenguajes de programación.

- **Dispositivo móvil:** al estar basada la aplicación en el sistema operativo Android, la aplicación está escrita en JAVA, lenguaje de programación que utiliza Android para la ejecución de sus aplicaciones nativas. Además Android provee un sencillo pero eficaz motor de bases de datos llamado **SQLite**, para interactuar con la base de datos se utiliza SQL.
- **Maquina:** la maquina está basado en una arquitectura de microcontrolador ATMega (en su montaje Arduino), para poderlo programar se utiliza lenguaje C++.

JAVA

El lenguaje de programación java en sus inicios fue creado y desarrollado por Sun Microsystems, la cual fue adquirida por Oracle en 1995.

Su sintaxis proviene en gran parte de C y C++, aunque cuenta con menos utilidades a bajo nivel, JAVA se ejecuta en una máquina virtual llamada máquina virtual JAVA (JVM), con esto Sun Microsystems trataba de crear un lenguaje de programación que pudiera ser ejecutado en diferentes maquinas con diferente arquitecturas y diferente hardware.

Se trata de un lenguaje de programación de propósito general, que se puede ejecutar de manera concurrente (varios hilos), es orientado a objetos y se basa todo en clases. Android decidió utilizar JAVA para el software que se desarrolla en su plataforma, para ello se implementó una máquina virtual llamada Dalvik (DVM), posteriormente sustituido por Art, en las nuevas versiones de Android.

La máquina virtual Dalvik cumple la misma función que la máquina virtual JAVA prestando un especial cuidado a la optimización de rendimiento y consumo de batería, las cuales son extremadamente importantes para dispositivos móviles.

Recolector de basura JAVA

El concepto recolector de basura, no es exclusivo al lenguaje de programación JAVA, este concepto se aplica a diferentes lenguajes de programación con diferentes propósitos.

Este concepto surgió debido a un problema general llamado Fuga de memoria (memory leacks), se trata de un error de software que ocurre cuando un bloque de memoria reservada (por un objeto, un array etc.) no es liberado en un software que se está ejecutando, comúnmente ocurre cuando se pierden las referencias a las direcciones de memoria antes de liberarlas.

El recolector de basura JAVA encuentra referencias huérfanas (zonas de memoria que no son accesibles porque se han perdido todas las referencias que apuntaban a ellas), y las libera para así poder ser utilizada de nueva, evitando fugas de memoria y evitando que el dispositivo se quede sin memoria disponible para poder ejecutar aplicaciones.

Herramientas de desarrollo

En este proyecto se han desarrollado dos partes de software, código Arduino y software para ejecutar en un dispositivo móvil Android.

Para el código C++ que se ejecuta en Arduino se ha utilizado el IDE de desarrollo de Arduino (https://www.arduino.cc/en/Main/Software) el cual se expone con más detalle en el trabajo de fin de grado de José Manuel Contreras Parras (segundo miembro de este proyecto).

Para el software que se ejecutará en el dispositivo Android se ha utilizado Android Studio, este IDE de desarrollo se presentó en el Google IO [68], como reemplazo oficial de Eclipse, que era el anterior entorno de desarrollo que se utilizaba para desarrollar aplicaciones Android.

Está basado en IntelliJ IDEA de Jetbrains [69], y es publicado a través de licencia gratuita Apache 2.0.

Las principales características de Android studio son:

- Renderización en tiempo real.
- Consola de desarrollador: consejos de optimización de código, estadísticas de uso etc.
- Soporte para construcción basada en Gradle.
- Refactorización especifica para Android y arreglos rápidos (refactorización de scope automático).
- Entre otras.

Librerías externas.

Una de las grandes ventajas que tienen los lenguajes de programación estándar, como es JAVA y el desarrollo en Android es la gran y amplia variedad de librerías y añadidos que se pueden utilizar libremente y añadirlos a diversos proyectos.

Para este proyecto he utilizado varias librerías que he encontrado en github [70] de varios desarrolladores que han desarrollado las librerías y las han distribuido de forma gratuita.

MPAndroidChart

Se trata de una librería diseñada para poder crear de manera rápida y fácil gráficos de diferentes tipos, creando incluso animaciones con las mismas.

Esta librería se utiliza para crear los gráficos de nivel de alcohol y dulzor para cada uno de los combinados introducidos por el usuario.

Para crear ese grafico se calcula una media utilizando la cantidad de cada ingrediente, teniendo en cuenta las características de cada uno de ellos.

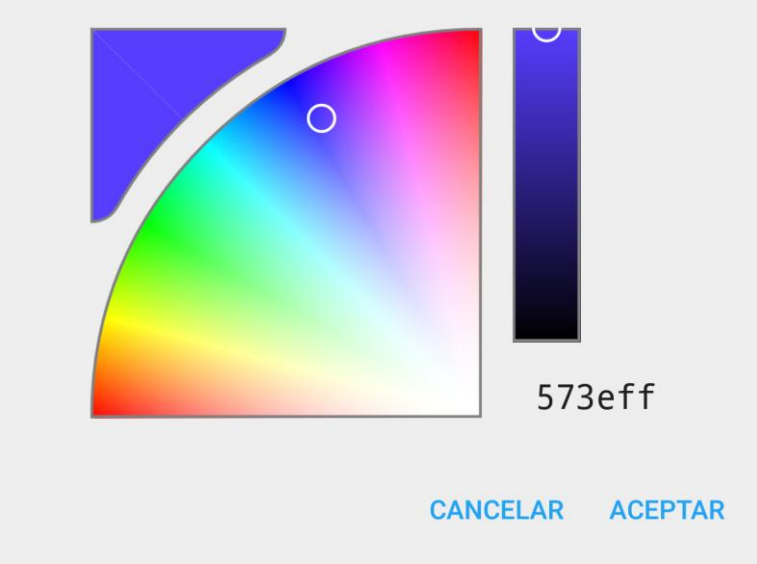
La librería puede encontrase en: https://github.com/PhilJay/MPAndroidChart

Rarepebble:colorpicker

Se trata de una librería diseñada para poder crear de manera rápida y fácil un selector de color RGB [71] enriquecido.

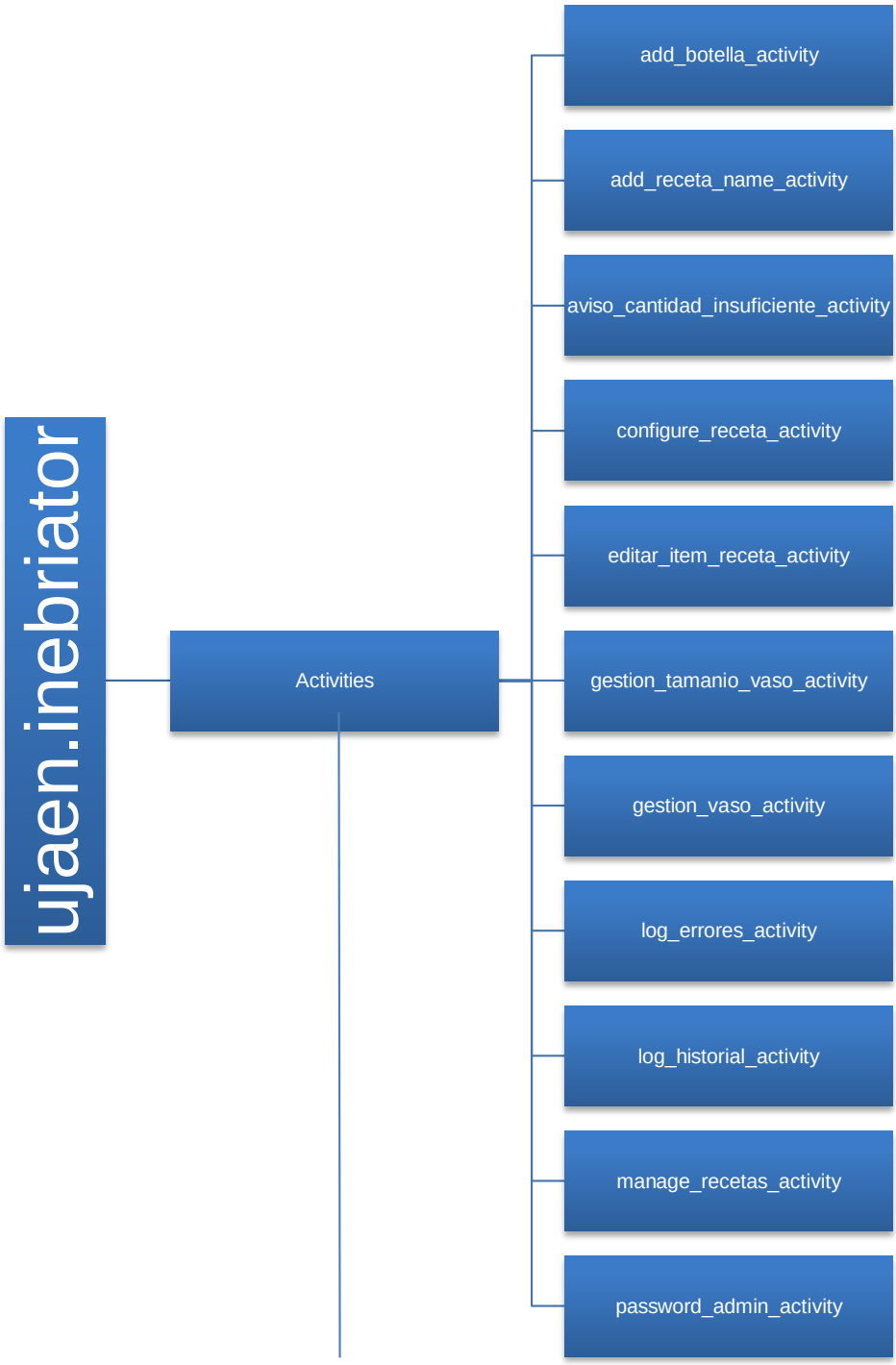
Esta librería es utilizada para crear los selectores de colores en la selección de ajustes, en la sección de colores para los gráficos (anteriormente mencionado) o la selección del color en los ajustes.

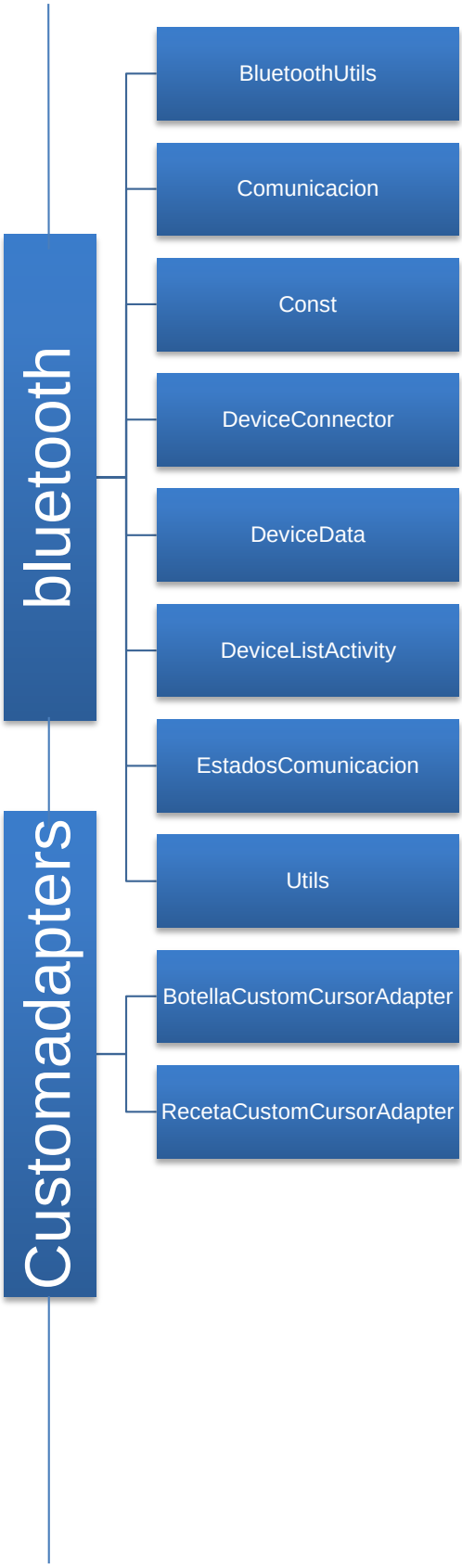
La librería puede encontrase en: https://github.com/martin-stone/hsv-alpha-color-picker-android

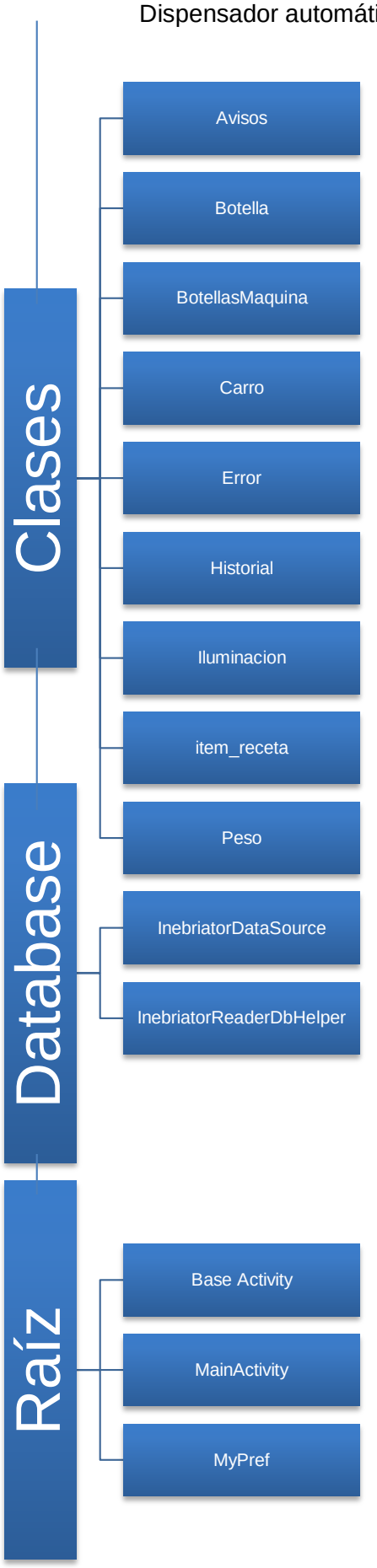


4.2 Ficheros.

En esta sección detallare todos los ficheros que componen el proyecto, detallando que rol juega dentro del proyecto y con qué lógica se ordenan dentro del mismo.




bluetooth
BluetoothUtils
Comunicacion
Const
DeviceConnector
DeviceData
DeviceListActivity
EstadosComunicacion
Utils
Customadapters
BotellaCustomCursorAdapter
RecetaCustomCursorAdapter


Clases
Avisos
Botella
BotellasMaquina
Carro
Error
Historial
Iluminacion
item_receta
Peso
Database
InebriatorDataSource
InebriatorReaderDbHelper
Raíz
Base Activity
MainActivity
MyPref

En la aplicación se pueden encontrar varias carpetas, pero la más importante es la de ujaen.adiaz.inebriator pues es la que contiene el proyecto en sí mismo, las otras carpetas las genera el IDE puesto que Android se sirve de otros ficheros (autogenerados) a partir de estos, para su correcto funcionamiento.

En general las carpetas no cumplen ninguna función más

Entre las diferentes carpetas se encuentran:

- **Activities:** En esta carpeta se encuentran todos los ficheros .java que se utilizan para crear una actividad, una actividad es cada una de las diferentes “pantallas” que se muestran en la aplicación.
- **Bluetooth:** En esta carpeta se encuentran los ficheros necesarios para gestionar la conexión bluetooth entre la máquina y el dispositivo móvil, para ello además de utilizarse las librerías de Android se gestiona todo a bajo nivel, pues el protocolo que implementan las librerías Android por defecto no es el protocolo que utiliza la máquina.
- **Clases:** Cada una de las clases que compone la lógica del programa, cada clase representa una entidad o un contenedor de funciones útiles.
- **Customadapters:** Los adaptadores son “puentes” entre los datos almacenados en una base de datos y una vista (para visualizarlos) Android implementa algunos de serie, pero si se quiere personalizar es necesario crearlos independientemente, en esta carpeta se encuentran los adaptadores personalizados que han sido necesarios para el proyecto.
- **Database:** En esta carpeta se encuentran todas las funciones e utilidades referentes a la gestión, conexión y comunicación con la base de datos, entre ellas se encuentran scripts para generar el esquema propiamente dicho, generar datos de ejemplo etc.
- **Raíz:** En esta carpeta se encuentran los ficheros que son usados justo al abrir la aplicación, por una razón meramente lógica estos no se encuentran dentro de ninguna carpeta.

4.3.1. Ficheros y su contenido

En esta sección se describirán cada uno de los ficheros con cada una de las funciones o variables que contienen y para qué son utilizadas.

Activities

- **Add_botella_activity**: Este fichero contiene la descripción de una actividad encargada de gestionar botellas, contiene los eventos para rellenar los formularios correspondientes cuando se pulsan los botones que contiene y cuando se selecciona una botella que ya existe, pues hay que cargarla, desde esta actividad es posible añadir una nueva botella, eliminar una botella o modificar una existente.
- **Add_receta_name**: Se trata de una actividad de tipo dialogo, esto significa que la manera de mostrarse será en una “ventana emergente”, solamente contiene un campo para añadir un nombre de la receta, a esta actividad se llega desde la actividad “manage_recetas ” el nombre solicitado por la actividad add_receta_name será pasado a la actividad “configure_receta_activity”
- **Aviso_cantidad_no_disponible_activity**: Es una actividad que se muestra cuando se intenta dispensar una receta de la cual no se cuenta suficiente cantidad de todos los elementos necesarios, en esta actividad se detecta exactamente de qué bebida no se tiene suficiente cantidad y se dé la opción al usuario de seguir con la dispensación, la cual será utilizando todas las cantidades disponibles de las que se dispone.
- **Configure_receta_activity**: Contiene la funcionalidad para poder configurar una receta, ya sea nueva o una que ya existía en el sistema, desde esta actividad es posible modificar los ingredientes, orden y cantidades de una receta determinada.
- **Editar_item_receta_activity**: Al igual que otras actividades, esta está diseñada para ser mostrada en forma de “ventana emergente”, a la hora de modificar un ingrediente de una receta la ventana emergente muestra un formulario en el que se puede introducir la cantidad en porcentaje a dispensar (relativa a la receta completa) y el orden en el cual debe dispensarse.

- **Gestion_tamanio_vaso_activity**: También se trata de una actividad diseñada para ser mostrado en forma de “ventana emergente”, en ella se puede modificar el tamaño del vaso en el que se va a dispensar, al configurar cada receta las cantidades se expresan en porcentajes, por lo que la cantidad que se dispensa de cada ingrediente depende del tamaño del vaso, además siempre se deja un margen para que el vaso no rebose, como último comentario nótese que la cantidad por defecto que aparece de vaso es configurable en el menú ajustes de la aplicación.
- **Gestion_vaso_activity**: Al igual que las anteriormente mencionadas actividades se trata de una actividad diseñada para ser visualizada en modo “ventana emergente” esta aplicación es mostrada cuando el vaso no ha sido detectado por la máquina y se está intentando llevar a cabo una dispensación, es posible que la máquina no haya detectado el vaso debido a que el vaso ya estaba colocado en el carro cuando esta se encendió, porque el vaso es demasiado ligero etc.
- **Log_errores_activity**: Se trata de una simple actividad para mostrar el contenido de una tabla de la base de datos que contiene registros de errores, solamente muestra texto que contiene fechas y motivos.
- **Log_historial_activity**: Al igual que la anterior se trata de una actividad encargada de mostrar el contenido de la tabla historial de la base de datos en texto plano, en ella se puede visualizar si la dispensación terminó correctamente o no, que y cuando se intentó dispensar.
- **Manage_recetas_activity**: Se trata de una actividad intermedia en la que se pueden visualizar las recetas que existen, se puede crear una nueva o eliminar una existente, en esta actividad no se realizan configuraciones de los ingredientes de las mismas.
- **Password_admin_activity**: Se trata de una actividad que se muestra en modo “ventana emergente” que recibe como entrada la clave de administrador (que puede ser configurada desde los ajustes), en caso de que la contraseña sea correcta se abrirá la anteriormente solicitada (esta actividad se muestra cuando se intenta abrir una actividad que requiere

contraseña), en caso contrario se cierra y muestra un mensaje diciendo que la contraseña no es correcta.

Bluetooth

Como se mencionó anteriormente en esta carpeta se encuentra todo el código encargado de gestionar las comunicaciones bluetooth.

- **BluetoothUtils:** En este fichero se encuentran las funciones que gestionan la comunicación bluetooth a bajo nivel, es decir crea la conexión y gestiona los tipos de servicios que están disponibles para dicho dispositivo.
- **Comunicación:** Gestiona a nivel de la propia aplicación (gestionando comunicación de la propia maquina), gestiona además los estados en los que se encuentra la comunicación de la máquina, además se encarga de gestionar todo lo referente a la retroalimentación en la comunicación, almacenando un buffer con las respuestas esperadas, en este fichero también es gestionado cuando la comunicación requiere un dato doble. Dentro del mismo se ejecuta un hilo en segundo plano para gestionar los mensajes recibidos por parte de la máquina.
- **DeviceConnector:** El código que contiene este fichero se encarga de gestionar la conexión con el dispositivo y el estado en el que se encuentra. Es la encargada de gestionar excepciones que pueden suceder durante la comunicación.
- **DiviceData:** En este fichero se encuentra la clase que almacena toda la información que existe del dispositivo bluetooth al cual se ha conectado, por ejemplo: nombre dispositivo, dirección física etc.
- **DeviceListActivity:** Este fichero contiene la definición grafica de la actividad encargada de mostrar todos los dispositivos disponibles para poder brindar la posibilidad al usuario de seleccionar a que dispositivo conectarse.
- **EstadosComunicacion:** Fichero que contiene cadenas con los estados de comunicación en los que se puede encontrar la comunicación con la máquina, en este fichero no existe realmente funcionalidad.

- **Utils:** Contiene funciones con las que se puede transformar la información recibida, como por ejemplo, concatenar varios bytes, convertir datos a hexadecimal etc.

Clases

Como anteriormente se ha mencionado, en esta carpeta se encuentran las carpetas que contienen ficheros con las definiciones de las clases necesarias para el correcto funcionamiento de la aplicación.

- **Avisos:** Fichero que contiene todos los avisos que puede enviar la máquina, además contiene las funcionalidades para manejar los avisos que genera la máquina.
- **Botella:** Fichero que contiene la clase que gestiona las botellas, además en esta clase es posible gestionar la creación, modificación y borrado de las botellas desde la base de datos. Entre sus atributos se encuentran: nombre, cantidad, dulzor, posición, alcohol, y cantidad restante de la misma.
- **BotellasMaquina:** Contiene la clase que se refiere a cada uno de los dispensadores en la máquina, esta gestiona de manera “física” los estados de los dispensadores además de poderlos activarlos, desactivarlos etc., además en la misma se puede controlar el compresor (dispositivo que inyecta aire a presión a las botellas que se encuentran colocadas debajo).
- **Carro:** Contiene la clase que contiene toda la lógica y funciones necesarias para poder gestionar el carro de la máquina, véase: calibración del carro, desplazamiento a una posición dada etc., además en la misma se encuentran los mapeos a direcciones físicas de las posiciones que se encuentran justo debajo de los dispensadores.
- **Error:** Fichero que contiene la clase encargada de gestionar el almacenamiento en la tabla de errores que ocurren en la máquina.
- **Historial:** Fichero que contiene la clase encargada de gestionar el almacenamiento en la tabla de historial de los sucesos que ocurren en la máquina.

- **Iluminación:** Contiene la clase que gestiona todo lo relacionado con iluminación, desde ella se puede controlar los efectos de luz, colores, etc. de las diferentes secciones de iluminación con las que cuenta la máquina.
- **Item_receta:** Se trata de una clase que gestiona la conexión que existe entre las botellas y la receta, puesto que cada receta se compone de varias botellas y además de la información que se requiere para dispensarlas, como es la cantidad que se quiere dispensar, y el orden en las cuales se deben dispensar, una receta no es más que un conjunto de items_receta.
- **Peso:** Clase encargada de la gestión del peso que se encuentra en el propio carro de la máquina, entre las funcionalidades que se pueden llevar a cabo se encuentran: tara, obtener el peso actual o iluminar el peso.
- **Receta:** Contiene la información de las recetas, se compone de los items_receta, además en ella se puede controlar los nombres de las recetas, el número de veces que se ha dispensado, entre otras.

CustomAdapters

Como anteriormente se ha mencionado, esta carpeta contiene los adaptadores necesarios para poder visualizar algunas vistas en la aplicación, los ficheros son los siguientes:

- **BotellaCustomCursorAdapter:** Se trata de un adaptador que hereda del base que provee Android pero añade un campo con la posición en la que se encuentra la botella, para poder visualizarlo de manera más fácil de un simple vistazo.
- **RecetaCustomCursorAdapter:** Este adaptador hace que las recetas se visualicen correctamente, mostrando cuantas veces han sido dispensadas.

Database

Como anteriormente se ha mencionado, esta carpeta contiene los ficheros que contienen las funciones necesarias para la creación y gestión de la base de datos.

- **InebriatorDataSource:** En este fichero se encuentra el código necesario para la creación de la base de datos, la mayoría de instrucciones están en lenguaje SQL [72], además en el mismo se encuentran las instrucciones necesarias para actualizar un elemento, eliminarlo o insertarlo, en este fichero se encuentran las referencias a todas las tablas que utiliza la aplicación, que posteriormente las clases hacen uso de ellas.
- **InebriatorReaderDbHelper:** Fichero que hace de interface para el anteriormente mencionado, realmente este fichero es el que es ejecutado que a su vez llama a las funciones de creación de las tablas, si se tiene que hacer algún cambio en la estructura de la base de datos, este es el fichero donde debe hacerse, de esta manera las aplicaciones serán compatibles hacia atrás [73].

Raíz

Esta es la carpeta raíz del proyecto en ella se encuentran algunas de los ficheros más importantes para la aplicación.

- **BaseActivity:** Se trata de una clase que modifica una actividad para incluirle funcionalidad necesaria para la gestión de la librería bluetooth.
- **MainActivity:** Actividad principal en la que se encuentra la mayor parte del código de esta aplicación, en ella se configuran los menús, se gestiona toda la parte encargada de la dispensación y cada vez que se gestiona una acción contra la máquina, pues estas se llevan a cabo en esta actividad, además en ella es donde se hace uso de las librerías necesarias para poder visualizar gráficas.
- **MyPreferencesActivity:** Actividad encargada de gestionar todos los ajustes que se pueden llevar a cabo en la aplicación, en esta actividad se hace uso de la librería externa utilizada para seleccionar un color utilizando un colorpicker, para la iluminación de la máquina.

CAPÍTULO 5

Conclusiones

Este apartado deberá de encontrarse en correlación con los objetivos fijados al principio del proyecto, para comprobar si el mismo cumple lo que pretendía en un primer momento.

1. Desarrollar una aplicación para controlar el microcontrolador utilizando tecnología Android.
2. Gestionar una base de datos SQLite, donde se almacenan diversas recetas de combinados, estadísticas etc.
3. Diseñar un software capaz de interactuar de manera directa y física con el mundo real.
4. El software diseñado, mostrará el catálogo de bebidas disponibles mostrando así su nivel de dulzor y alcohol de manera aproximada y de forma visual.
5. Valida si se disponen de los elementos necesarios para elaborar una bebida determinada.
6. El software proveerá una funcionalidad para ver el estado actual de la máquina.

Después de haber realizado en su totalidad el proyecto, puedo concluir que los objetivos han sido logrados satisfactoriamente, a continuación daré un ejemplo de cada uno de los puntos mencionados anteriormente:

1. Se ha realizado una aplicación basada en Android, que utilizando una comunicación bluetooth es capaz de comunicarse con un microcontrolador, en este caso un Arduino.
2. El programa gestiona una base de datos SQLite, almacenando: recetas, estadísticas (historiales, errores), y las bebidas que están actualmente en la máquina, teniendo en cuenta la cantidad actual restante de las mismas.
3. El software es capaz de ejecutar diversas acciones en motores, luces etc. utilizando un simple botón como puede ser el botón “dispensar”.
4. En el software se puede gestionar un catálogo de bebidas, visualizar su nivel de dulzor y alcohol (de manera gráfica).
5. La aplicación gestiona las cantidades de cada botella, en caso de no disponer de suficiente cantidad de algún líquido, muestra un aviso y modifica las recetas en tiempo real para poder satisfacer las necesidades del cliente.

6. En el software se puede ver si la aplicación se encuentra conectada a la máquina, o si el vaso ha sido detectado entre otras.

Además cabe destacar que para la realización del sistema informático se han seguido las buenas prácticas definidas por las etapas de Ingeniería del Software, las cuales ayudan a poder cumplir con todos los objetivos y requerimientos del sistema.

Este sistema se puede utilizar para atraer más clientela a un determinado pub o restaurante, por el hecho de que la maquina en funcionamiento es llamativa, debido a su automatización y juego de luces.

Además los propietarios de los locales pueden servir de una manera precisa siempre el mismo producto, haciendo así sus recetas de cocteles, siempre consistentes en sabor y cantidad. Es importante tener en cuenta que muchas franquicias gestionan muy estrictamente las cantidades que proporcionan a sus clientes, por lo que la utilización de la maquina desarrollada en el proyecto podría minimizar gran cantidad de costes.

Con respecto a las desventajas de este primer prototipo cabe mencionar que no se ha diseñado de manera que pueda ser comercializada en masa ya que no existe un proceso de creación detalladamente especificado y automatizado.

Además aún está en una fase temprana de desarrollo, lo que significa que existen potenciales casos de uso que no han sido contemplados en este proyecto o que pueden existir inconvenientes que debido a la falta de madurez del mismo no se han hecho patentes.

Bajo mi punto de vista, este trabajo muestra como hoy en día es posible aplicar tecnologías para automatización de tareas que a priori pueden percibirse como no automatizables, pero con ingeniería es posible automatizarlas, resolviendo una necesidad que hasta hoy no existía ninguna manera de solventarla.

CAPÍTULO 6

Bibliografía

[1] Historia del vending: Máquinas expendedoras Fecha de última consulta 07/10/2014

http://www.eraventa.com/historia.htm

[2] Origen e Historia de las máquinas expendedoras Fecha de última consulta 13/07/2016

http://www.oyarzunsl.com/blog/2011/11/vending-el-origen-e-historia-de-las-maquinas-expendedoras/

[3] Vending History Fecha de última consulta 13/07/2016

https://www.varietyfoodservices.com/vending/vending-history/

[4] "Vending Machines" Vending Machines: An American Social History, Kerry Segrave

ASIN: B009G5EGTQ, edición en inglés publicada por McFarland (12 Julio 2012).

[5] ¿Qué es Arduino?: Plataforma OpenSource Fecha de última consulta 24/01/2015

http://arduino.cc/en

[6] Sensores: Máquinas expendedoras Fecha de última consulta 07/10/2014

http://www.profesormolina.com.ar/tecnologia/sens_transduct/que_es.htm

[7] Que es un microcontrolador Fecha de última consulta 24/01/2015

http://sherlin.xbot.es/microcontroladores/introduccion-a-los-microcontroladores/que-es-un-microcontrolador

[8] Wiring y proccessing Fecha de última consulta 24/01/2015

http://projectbot.blogspot.de/2011/01/sobre-arduino-y-su-lenguaje.html

[9] Controlando Arduino con Android Fecha de última consulta 11/07/2016

http://www.prometec.net/android-bt/

[10] Ars Electrónica Prix Fecha de última consulta 24/01/2015

https://en.wikipedia.org/wiki/Prix_Ars_Electronica

[11] SQLite Android Fecha de última consulta 24/01/2015

https://developer.android.com/reference/android/database/sqlite/package-summary.html

[12] "¿Qué es Android?, Características diferenciadoras" El gran libro de Android, Jesús Tomas Girones

ISBN: 978-84-267-1832-7, edición en español publicada por MARCOMBO, S A., Barcelona, España

[13] Microsoft mobile Fecha de última consulta 29/01/2015

https://en.wikipedia.org/wiki/Microsoft_Mobile

[14] Apple mobile Fecha de última consulta 29/01/2015

https://en.wikipedia.org/wiki/Apple_Inc.

[15] Blackberry mobile Fecha de última consulta 29/01/2015

https://es.wikipedia.org/wiki/BlackBerry

[16] Firefox OS Fecha de última consulta 29/01/2015

https://www.mozilla.org/en-US/firefox/os/

[17] Que es Unix Fecha de última consulta 29/01/2015

http://www2.udec.cl/~sscheel/pagina virus/Unix.htm

[18] Java en Android El gran libro de Android, Jesús Tomas Girones

ISBN: 978-84-267-1832-7, edición en español publicada por MARCOMBO, S A., Barcelona, España

[19] Que es una CPU Fecha de última consulta 29/05/2016

http://www.abc.es/tecnologia/consultorio/20150213/abci--201502122200.html

[20] Máquina virtual dalvik El gran libro de Android, Jesús Tomas Girones

ISBN: 978-84-267-1832-7, edición en español publicada por MARCOMBO, S A., Barcelona, España

[21] XML interfaces en Android Fecha de última consulta 29/05/2016

https://developer.android.com/guide/topics/ui/declaring-layout.html

[22] Que es el GPS Fecha de última consulta 31/01/2015

http://stackoverflow.com/questions/33637/how-does-gps-in-a-mobile-phone-work-exactly

[23] Que es un sandbox y para que se usa Fecha de última consulta 29/5/2016

http://inforpaco.blogspot.de/2013/06/que-es-y-para-que-sirve-una-sandbox.html

[24] OpenGL en Android Fecha de última consulta 31/01/2015

https://developer.android.com/training/graphics/opengl/index.html

[25] Hanset Alliance, estándares en los móviles Fecha de última consulta 29/05/2016

https://es.wikipedia.org/wiki/Open_Handset_Alliance

[26] Licencia Apache Fecha de última consulta 13/02/2015

http://www.apache.org/foundation/license-faq.html

[27] Que son los SDKs Fecha de última consulta 13/02/2015

http://www.fandroides.com/que-es-y-para-que-sirve-el-sdk/

[28] Que es iOS Fecha de última consulta 29/05/2016

http://www.apple.com/es/ios/what-is/

[29] Comparativa plataformas móviles Fecha de última consulta 07/10/2014

http://www.androidcurso.com/index.php/tutoriales-android/31-unidad-1-vision-general-y-entorno-de-desarrollo/98-comparativa-con-otras-plataformas

[30] Introducción a las aplicaciones móviles Fecha de última consulta 27/02/2015

http://www.academia.edu/11144573/Introducción_a_las_Aplicaciones_Moviles

[31] Android vs Apple market share in 2015 Fecha de última consulta 15/07/2016

http://bgr.com/2016/06/02/apples-mobile-market-share-sees-big-drop-in-may-as-android-skyrockets/

[32] Gratner Group Fecha de última consulta 27/07/2016

http://www.gartner.com/technology/about.jsp

[33] Symbian Fecha de última consulta 29/05/2016

https://en.wikipedia.org/wiki/Symbian

[34] Arquitectura del sistema operativo Android Fecha de última consulta 07/10/2014

https://sites.google.com/site/swcuc3m/home/android/generalidades/2-2-arquitectura-de-android

[35] Que es una API y los tipos que existen Fecha de última consulta 29/05/2016

https://developer.android.com/reference/packages.html

http://www.androidcurso.com/index.php/tutoriales-android/31-unidad-1-vision-general-y-entorno-de-desarrollo/146-las-versiones-de-android-y-niveles-de-api

[36] Que es un framework Fecha de última consulta 27/02/2015

http://jordisan.net/blog/2006/que-es-un-framework/

[37] APIs más destacadas de Android Fecha de última consulta 07/10/2014

https://sites.google.com/site/swcuc3m/home/android/generalidades/2-2-arquitectura-de-android

[38] Surface Manager Android Fecha de última consulta 27/02/2015

https://silextech.wordpress.com/tag/surface-manager/

[39] Creación de una librería en Android Fecha de última consulta 29/05/2016

http://code.tutsplus.com/es/tutorials/creating-and-publishing-an-android-library--cms-24582

[40] Librerías Android basadas en C/C++ Fecha de última consulta 29/05/2016

https://developer.android.com/ndk/guides/cpp-support.html?hl=es

[41] Graphic acceleration technologies in Android Fecha de última consulta 13/03/2015

http://es.slideshare.net/peterbuck/opengles-and-sgl-2d3d-graphics-accelerating-technologies

[43] What Webkit is Fecha de última consulta 13/03/2015

https://en.wikipedia.org/wiki/WebKit

[43] Que es el código máquina Fecha de última consulta 13/03/2015

https://es.wikipedia.org/wiki/Lenguaje_de_máquina

[44] Producto Apple iPod Fecha de última consulta 29/05/2016

https://es.wikipedia.org/wiki/IPod

[45] Producto Apple iPad Fecha de última consulta 29/05/2016

https://es.wikipedia.org/wiki/IPad

[46] Objetive-C Fecha de última consulta 27/03/2015

http://www.tutorialspoint.com/ios/ios_tutorial.pdf

[47] Comparativa Darwin BSD y Mac OSx Fecha de última consulta 24/07/2017

http://enclavedemac.blogspot.de/2013/01/darwin-vs-mac-os-x.html

[48] Apple, un sistema cerrado Fecha de última consulta 24/07/2017

https://hipertextual.com/archivo/2010/11/porque-android-es-peor-opcion-que-ios-actualmente/?utm_source=dlvr.it&utm_medium=twitter&utm_campaign=appleweblog

[49] Que es un sistema en tiempo real Fecha de última consulta 27/03/2015

http://laurel.datsi.fi.upm.es/~ssoo/STR/Introduccion.pdf

[49] Que son los sistemas empotrados Fecha de última consulta 24/07/2017

https://www.exabyteinformatica.com/uoc/Informatica/Sistemas_empotrados/Sistemas_empotrados_(Modulo_1).pdf

[51] Introducción a basic stamp Fecha de última consulta 15/04/2015

http://www.aprenderobotica.com/group/eslaprimeravez/page/principiantes-basic-stamp

[52] Que es un PinOut Fecha de última consulta 15/04/2015

https://es.wikipedia.org/wiki/Pinout

[53] Que es y cómo se utiliza OTG en Android Fecha de última consulta 21/07/2016

http://www.fandroides.com/que-es-y-como-utilizar-el-otg-en-android/

[54] Que es el Wifi Fecha de última consulta 21/07/2016

http://www.quees.info/que-es-wifi.html

[55] Que es el Bluetooth Fecha de última consulta 21/07/2016

http://www.informatica-hoy.com.ar/telefonos-celulares/Que-es-Bluetooth.php

[56] Que es la IEEE Fecha de última consulta 15/04/2015

http://whatis.techtarget.com/definition/IEEE-Institute-of-Electrical-and-Electronics-Engineers

[57] Que es la seguridad WEP Fecha de última consulta 08/05/2015

https://en.wikipedia.org/wiki/Wired_Equivalent_Privacy

[58] Que es la seguridad WAP2 y comparativa con otras Fecha de última consulta 08/05/2015

http://www.howtogeek.com/167783/htg-explains-the-difference-between-wep-wpa-and-wpa2-wireless-encryption-and-why-it-matters/

[59] Comunicación serie, que es y usos Fecha de última consulta 21/05/2015

http://digital.ni.com/public.nsf/allkb/039001258CEF8FB686256E0F005888D1

[60] Que es y para qué sirve un Smart watch Fecha de última consulta 29/05/2016

http://www.androidexperto.com/dispositivos-con-android/que-es-sirve-smartwatch/

[61] Feedback en comunicaciones Fecha de última consulta 29/05/2016

http://www.significados.com/feedback/

[62] Nibble Fecha de última consulta 10/06/2015

http://www.ehu.eus/mungi/unidades.htm

[63] Concepto de interfaz Fecha de última consulta 22/08/2015

http://definicion.de/interfaz/

[64] Experiencia del usuario UX Fecha de última consulta 04/09/2015

https://en.wikipedia.org/wiki/User_experience_design

[65] Que es la usabilidad Fecha de última consulta 17/07/2016

http://rd.udb.edu.sv:8080/jspui/bitstream/11715/519/1/2.%20La%20usabilidad%20en%20Ingenieria%20de%20Software-%20definicion%20y%20caracteristicas.pdf

[66] Diferencia entre experiencia del usuario y usabilidad Fecha de última consulta 17/07/2016

http://www.nosolousabilidad.com/articulos/uxd.htm

[67] Android desing Fecha de última consulta 29/05/2016

https://developer.android.com/design/index.html

[68] Google IO Fecha de última consulta 29/05/2016

https://events.google.com/io2016/

[69] Jetbrains Fecha de última consulta 29/05/2016

https://www.jetbrains.com/

[70] Github Fecha de última consulta 29/05/2016

https://github.com/

[71] RGB Fecha de última consulta 05/03/2016

https://en.wikipedia.org/wiki/RGB_color_space

[72] SQL Fecha de última consulta 10/03/2016

http://definicion.de/sql/

[73] Concepto de compatibilidad hacia atrás Fecha de última consulta 02/04/2016

https://en.wikipedia.org/wiki/Backward_compatibility