UNIVERSITAT POLITÈCNICA DE VALÈNCIA

_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

Desarrollo de una aplicación móvil en entorno Android que permite ahorrar en consumo a los barcos que atracan en el Puerto de Valencia

Raquel Macián Marí

Tutor: Carlos Alberto Hernández Franco

Trabajo Fin de Grado presentado en la Escuela Técnica Superior de Ingenieros de Telecomunicación de la Universitat Politècnica de València, para la obtención del Título de Graduado en Ingeniería de Tecnologías y Servicios de Telecomunicación

Curso 2017-18

Valencia, 11 de septiembre de 2018

Escuela Técnica Superior de Ingeniería de Telecomunicación
Universitat Politècnica de València
Edificio 4D. Camino de Vera, s/n, 46022 Valencia
Tel. +34 96 387 71 90, ext. 77190
www.etsit.upv.es

VLC/ CAMPUS
VALENCIA, INTERNATIONAL CAMPUS OF EXCELLENCE

EMAS
Gestión ambiental verificada
REG.NO. ES-CV- 000030

Resumen

El presente trabajo se centra en la realización de una aplicación en entorno Android que permita ahorrar en consumo a los barcos que atracan en el Puerto de Valencia.

Actualmente, tan solo las terminales disponen de la hora actualizada en la que se realizará el atraque de los buques y asimismo el inicio de las operaciones. En contraposición gran parte de las embarcaciones únicamente disponen de la hora prevista de atraque, e incluso algunas no disponen de ella.

El principal objetivo de la aplicación será que las embarcaciones puedan consultar en tiempo real la hora de atraque en el Puerto de Valencia. De esta forma los capitanes de dichas embarcaciones establecerán una velocidad apropiada, ahorrando así combustible, y llegarán a una hora apropiada para atracar.

Resum

El present treball se centra en la realització d'una aplicació en entorn Android que permeta estalviar en consum als barcos que atraquen en el Port de València.

Actualment, tan sols les terminals disposen de l'hora actualitzada en la que es realitzarà l'atracada dels barcos i així mateix l'inici de les operacions. En contraposició gran part de les embarcacions únicament disposen de l'hora prevista d'atracada, i inclús algunes no disposen d'ella.

El principal objectiu de l'aplicació serà que les embarcacions puguen consultar en temps real l'hora d'atracada en el Port de València. D'esta manera els capitans de les dites embarcacions establiran una velocitat apropiada, estalviant així combustible, i arribaran a una hora apropiada per a atracar.

Abstract

The present work focuses on the realization of an application in Android environment that allows saving in consumption to the ships that dock in the Port of Valencia.

Currently, only terminals have the updated time in which the berthing of the ships will be carried out and also the start of operations. In contrast, a large part of boats only have the expected time of berthing, and even some do not have it.

The main objective of the application will be that boats can consult in real time the docking time in the Port of Valencia. In this way the captains of these vessels will establish an appropriate speed, thus saving fuel, and they will arrive at an appropriate time to dock.

Tabla de contenido

ÍNDICE DE ILUSTRACIONES

Capítulo 1. Introducción

1.1 Motivación

Con el paso de los años la tecnología se ha convertido en un elemento indispensable en el día a día. Actualmente las empresas punteras disponen de una gran plataforma tecnológica permitiendo grandes mejoras tanto al empleado como en los servicios ofertados por ella. Tal y como se muestra en la siguiente figura casi el 100% de las empresas grandes en 2016 disponían de ordenadores y de conexión a internet [1].

		Empresas con menos de 10 empleados	Empresas con más de 10 empleados
Disponen de ordenadores	1	73,08	99,57
Tiene conexión a internet	1	70,22	98,70
Tiene conexión a internet y página web	2	29,81	77,69
Utilizan medios sociales	2	31,20	49,57
Realizan ventas por comercio electrónico	1	4,45	20,41
Realizan compras por comercio electrónico	1	15,70	31,36

1. Datos medidos en porcentaje sobre el total de empresas de cada tipo
2. Datos medidos en porcentaje sobre el total de empresas con conexión a internet de cada tipo

Figura 1: Estadística uso de tecnologías

La tecnología avanza a pasos agigantados y en especial dentro del sector de transporte marítimo. En la actualidad la innovación tecnológica en este sector se manifiesta principalmente en el aumento de carga y en la creciente tendencia a utilizar buques cada vez más grandes.

Algunos cambios han sido de gran escala como PCS (Port Community System) que ha cambiado en gran medida la gestión de solicitudes y prestación de servicios portuarios. En el Puerto de Valencia, en el que nos centramos en el proyecto, PCS da servicio a 3 puertos de la región con un tráfico de más de 4.4 millones de contenedores anuales.

Otra de las tecnologías que podría destacar dentro del sector sería Marine Traffic, servicio disponible tanto en aplicación web o móvil. Esta aplicación permite saber de manera actualizada la posición de los buques a través de un interfaz sencillo y manejable.

Tiene gran variedad de opciones para los usuarios,desde calcular la ruta del buque,conocer la meteorología o incluso conocer la posición de los otros usuarios.

Figura 2: Icono Marine Traffic

Con este proyecto intentaremos realizar una mejora tecnolgica la cual vamos a ir explicando a lo largo de todo el cuerpo del proyecto.

1.2 Problema planteado

Las autoridades portuarias son ante el consignatario o representante del buque, la ventanilla única de recepción de documentos relativos a una escala prevista conforme al marco normativo regulador de dicha actividad.

Centrándome más en el caso de la Autoridad Portuaria de Valencia (APV), lugar de mis prácticas formativas, para la autorización de una escala de un buque, debe presentarse una solicitud ante la APV. Dicha solicitud, que es presentada por el consignatario o representante del buque debe incluir la información exigida en el Documento Único de Escala (DUE).

La documentación asociada a una escala de un buque en puerto incluye gran cantidad de documentos necesarios para todas las entidades públicas o privadas implicadas en dicha actividad.

Una de las actividades comprendidas en la escala de un buque en puerto es su atraque, y, dentro de estas actividades, el de los buques portacontenedores, proceso que voy a analizar más en profundidad.

La autorización del atraque de un buque corresponde a la Comisaría del Puerto que indica el muelle en el que se realiza el atraque teniendo en cuenta los buques que están operando, el calado del muelle y del buque, el equipamiento portuario necesario para el trabajo, la posición de las zonas de carga y descarga, las demandas futuras de atraques, etc.

El punto de atraque es asignado por la APV después de analizar con cada una de las empresas estibadoras la posición operativa más adecuada para que la actividad del buque se realice de la forma más eficiente y se cumpla con toda la normativa reguladora aplicable a dicha actividad.

En el cuadro siguiente se ofrece una visión cartesiana de una distribución de atraques en una terminal de contenedores siendo el eje de las abscisas el espacio o atraque ocupado por buques trabajando (color naranja) y el eje de ordenadas el tiempo previsto de trabajo. Los buques previstos son los que figuran en color azul con el mismo planteamiento espacio temporal. La forma de distribución del tiempo de trabajo viene dada por las jornadas de trabajo tipo de los estibadores portuarios.

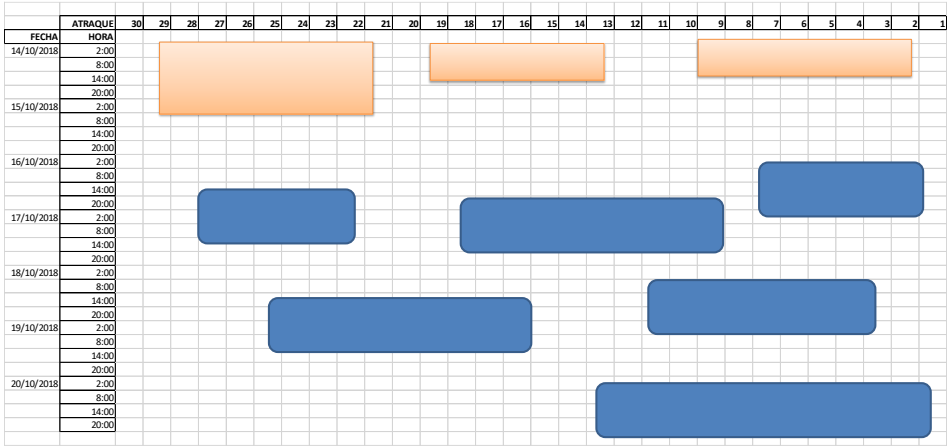


Figura 3: Distribución de atraques

Como he comentado anteriormente existen plataformas que ofrecen una visión gráfica de los buques navegando y en puerto, pero ninguna de ellas da solución a cuándo estará libre el atraque previsto asignado por la APV y acordado con la empresa estibadora que va a operar al buque.

La situación actual es el contacto por voz o datos del consignatario del buque con la terminal y el posterior contacto por los mismos medios con el capitán del buque que se dirige a puerto para informarle de la disponibilidad de atraque.

La solución que se plantea con este proyecto es una aplicación en la que interactúan buque y terminal para optimizar los tiempos de llegada y atraque con el consiguiente beneficio económico y medioambiental al posibilitar el ajuste de la velocidad del buque a la disponibilidad de atraque.

1.3 Estructura del documento

En este apartado se explicará de manera resumida el contenido de cada uno de los apartados del Trabajo de Fin de Grado:

1. **Introducción**

   En este apartado se explica la motivación, el problema planteado y la estructura de todo el cuerpo del proyecto.

2. **Metodología**

   Se explican las diferentes metodologías más utilizadas en el desarrollo de software y mi elección para elaborar el proyecto

3. **Marco teórico**

   Se explica de manera detallada las herramientas así como la tecnología utilizada

4. **Desarrollo**

   Se explicará el desarrollo de cada una de las partes de la aplicación.

5. **Conclusiones**

   En este último apartado se darán a conocer las conclusiones tras la elaboración del proyecto, así como un análisis económico respecto al ahorro de combustible y las diferentes líneas futuras entorno al proyecto.

Capítulo 2. Metodología

A la hora de abordar el desarrollo de una aplicación es imprescindible planificar el proceso, y en consecuencia su metodología, ya que la realización de un proyecto de tipo software es difícil y minucioso.

Cuando hablamos de metodología, en concreto la metodología de desarrollo de software, nos referimos al conjunto de procedimientos utilizados para el diseño de sistemas de información**.** En ingeniería de software cuando se hace referencia al desarrollo, se está hablando del desarrollo de programas, los cuales deben cumplir una serie de etapas o fases, para poder funcionar con otros métodos ya establecidos en otras disciplinas de ingeniería. [2]

Existen gran variedad de metodologías pero se pueden dividir en dos grandes grupos: las metodologías tradicionales y las metodologías ágiles. En los siguientes apartados explicaré algunas de las más utilizadas dentro de la ingeniería de software

2.1 Tipos de metodología

2.1.1 Modelo Cascada

Este modelo pertenece al grupo de metodologías tradicionales. Es una de las metodologías más antiguas y poco a poco está cayendo en desuso. La mayoría de empresas optan por otras metodologías más innovadoras.

Su principal característica es la imposibilidad de pasar a la siguiente etapa sin haber acabado por completo la etapa anterior. Sigue un proceso de trabajo como el que se muestra en la siguiente figura 4:

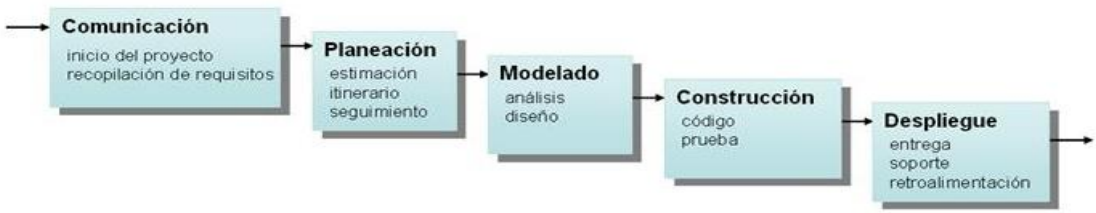


Figura 4: Modelo Cascada

2.1.2 Modelo de prototipo.

Aunque al igual que la metodología en cascada esta metodología tradicional tiene mucha antigüedad sigue siendo muy utilizada actualmente.

Este método sigue un procedimiento de desarrollo que permite hacer un prototipo para posteriormente poder validar su funcionalidad y hacer los cambios necesarios. Sigue un proceso de trabajo como el de la figura 5:

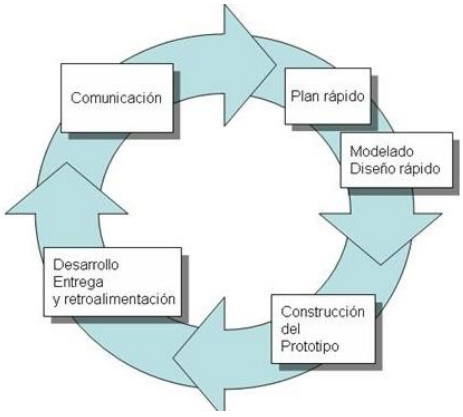


Figura 5: Modelo prototipo

2.1.3 Modelo en espiral

El modelo en espiral fue utilizado diseñado por Barry Boehm en 1986. Se trata de una combinación entre el modelo de cascada y el modelo en prototipos con una etapa añadida que trata la gestión de riesgos.

Consiste en ciertas etapas que se van elaborando en modo de espiral que a su vez utiliza procesos de la misma forma en que en el modelo de cascada pero sin llevar un orden establecido. Básicamente se caracteriza porque conforme va avanzando el proceso se irá incrementando el nivel de código fuente, un aumento en la gestión de riesgos y por supuesto una mejora en los tiempos de ejecución y planificación del sistema. Vemos las diferentes fases que lo forman en la figura 6:

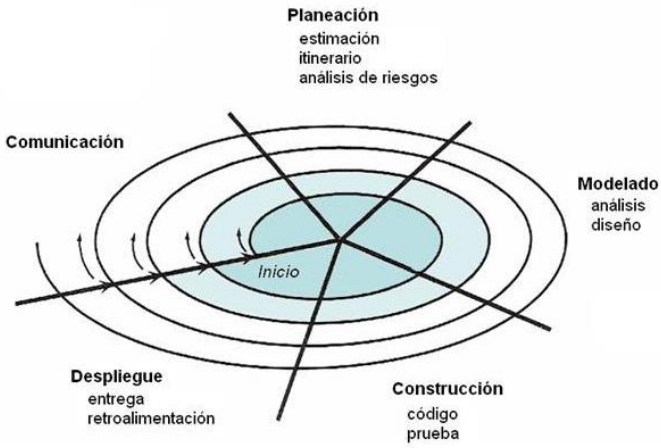


Figura 6: Modelo en espiral

2.1.4 Desarrollo rápido de aplicaciones (RAD)

En este modelo se le da especial importancia a la obtención de un prototipo funcional para posteriormente ir mejorándolo añadiendo una mayor complejidad. Se recomienda el uso de patrones por si es necesario un cambio en los requisitos. Se basa en la estructura mostrada en la figura 7:

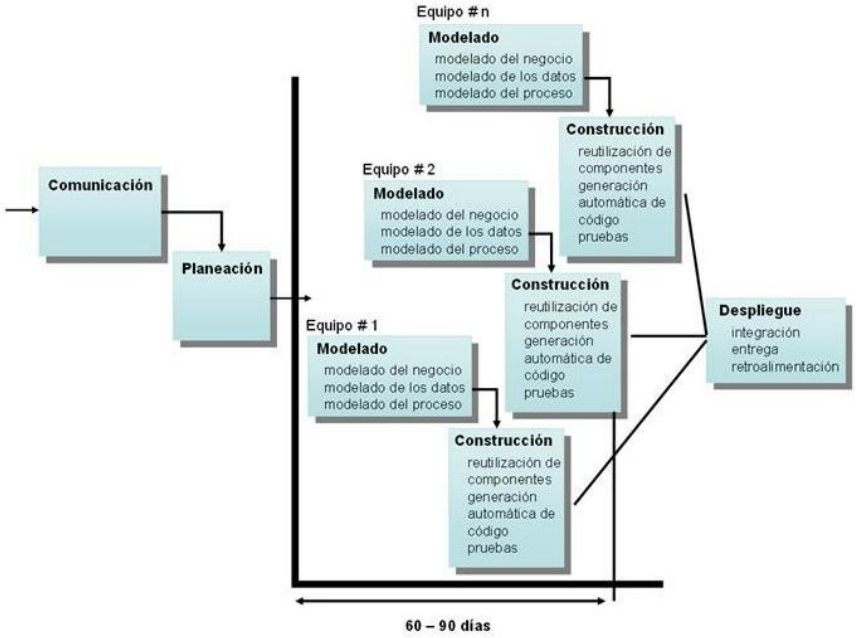


Figura 7: Modelo RAD

2.1.5 Metodología de programación externa (XP)

Por ultimo hablaré de una de las metodologías ágiles más utilizadas en el desarrollo de aplicaciones móviles, la metodología de programación externa. Esta metodología, se utiliza para evitar el desarrollo de funciones que no sean necesarias, pero sobre todo para para llevar a cabo proyectos complicados. Sin embargo el proceso de elaboración puede durar más tiempo.

La programación extrema se basa en 5 valores fundamentales para que se lleve a cabo el proyecto. Estos se muestran en la figura 8.

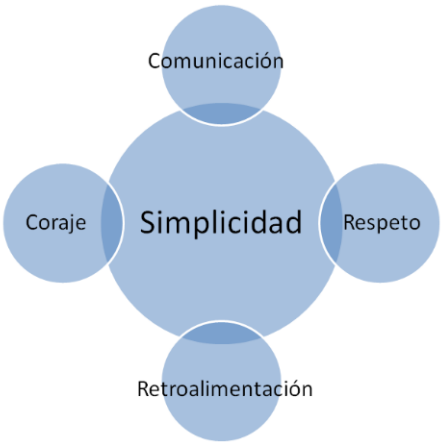


Figura 8: Modelo XP

2.2 Elección de la metodología

Para la elaboración de este proyecto he decidido basarme en la metodología de desarrollo rápido de aplicaciones (RAD).Como hemos comentado en el apartado anterior se caracteriza por la elaboración de un prototipo funcional para posteriormente ir mejorándolo.

En mi caso nunca he elaborado ningún proyecto en programación Android y puesto que no sabía a qué dificultades podía enfrentarme he preferido llevar a cabo esta metodología más conservadora para poder realizar una idea sencilla para luego si es posible mejorarla.

Capítulo 3. Marco teórico

En este capítulo haremos una breve descripción de todos los elementos utilizados tanto hardware como software así como las herramientas utilizadas a lo largo de la aplicación. Se explicarán tantos detalles concretos sobre el funcionamiento así como su historia.

Principalmente abordaremos el sistema operativo en el que hemos trabajado, Android, además de otras herramientas que han sido útiles a lo largo de la creación del proyecto ya sea para la interfaz gráfica o para el programa.

3.1 Java

El lenguaje de programación Java, fue diseñado por la compañía Sun Microsystems Inc, con el objetivo de crear un lenguaje que pudiera funcionar en sistemas de ordenadores y que fuera independiente de la plataforma en la que se vaya a ejecutar. Esto significa que un programa de Java puede ejecutarse en cualquier máquina o plataforma.

En1991 FirstPerson, filial de Sun y en concreto James Gosling creó un lenguaje de programación para desarrollar aplicaciones aplicables a electrodomésticos. Este lenguaje recibió en nombre de Oak.Años más tarde se abandonó este proyecto hasta que en 1995, concretamente en Septiembre, aparece el primer *Kit de Desarrollo de Java* (JDK) destacando la aparición de la primera versión de java a principios de 1997.Posteriormente ha ido mejorándose y han ido apareciendo continuamente nuevas versiones de la plataforma.

Figura 9: Icono java

3.1.1 Características

El lenguaje Java presenta las siguientes características:

- **Sencillo**. No presenta la complejidad de otros lenguajes y emplea una programación orientada a objetos. Aunque la sintaxis de Java es muy similar a otros lenguajes como C y C++, lenguajes conocidos por gran parte de los desarrolladores.

- **Orientado a Objetos**. la programación orientada a objetos es un paradigma de programación que utiliza entidades llamadas objetos como elementos fundamentales a la hora de construir una solución. El objetivo de este tipo de programación es la reutilización de estos objetos genéricos en diferentes proyectos, una reutilización del código que es precisamente una premisa de la Ingeniería del Software, de este modo se pretende reducir el número de proyectos fallidos, así como obtener un menor tiempo de desarrollo

- **Gestión automática dela memoria**: Java utiliza un recolector automático de basura para la gestión de la memoria en el ciclo de vida de los objetos. El programador decide cuando se crean y destruyen los objetos, sin preocuparse de la gestión de la memoria. El entorno en tiempo de ejecución de Java, llamado Java runtime¸ se encarga del ciclo de vida de los objetos con la ayuda del recolector de basura, que libera la memoria que ocupaban estos objetos.

- **Seguro**. Java fue diseñado en una estructura Button Up teniendo en cuenta la seguridad de la plataforma. Además, se implementaron medidas de seguridad en el lenguaje y en el sistema de ejecución en tiempo real

- **Multitarea**. el lenguaje de programación Java soporta la sincronización de múltiples hilos de ejecución, concepto llamado multithreading, por lo que tiene la capacidad de cumplir diferentes funciones al mismo tiempo.

- **Dinámico**. Java no necesita cargar completamente el programa en memoria sino que las clases compiladas pueden ser cargadas bajo demanda en tiempo de ejecución. Esto provoca que la ejecución sea más rápida y en consecuencia nos ahorra tiempo de espera.

- **Independencia de la plataforma**: el software escrito en Java está preparado para ejecutarse en cualquier tipo de hardware, es decir, el software se programa solamente una vez para que pueda ejecutarse en varios dispositivos, lo que se conoce como WORA, como ya hemos mencionado antes.

3.1.2 Funcionamiento

Como hemos comentado en el apartado anterior una de las principales características de java es la capacidad de ser compilado e interpretado al mismo tiempo. Mediante el compilador de Java, el programa fuente es traducido a un lenguaje que llamamos Java bytecodes.Este archivo de extensión .class será interpretado y ejecutado por el intérprete de Java (Máquina Virtual Java).Es por esto que Java es considerado multiplataforma, ya que existe un intérprete para cada equipo diferente. En resumen, la compilación se produce una vez y la interpretación cada vez que el programa se ejecuta. Este proceso se muestra en la imagen siguiente:

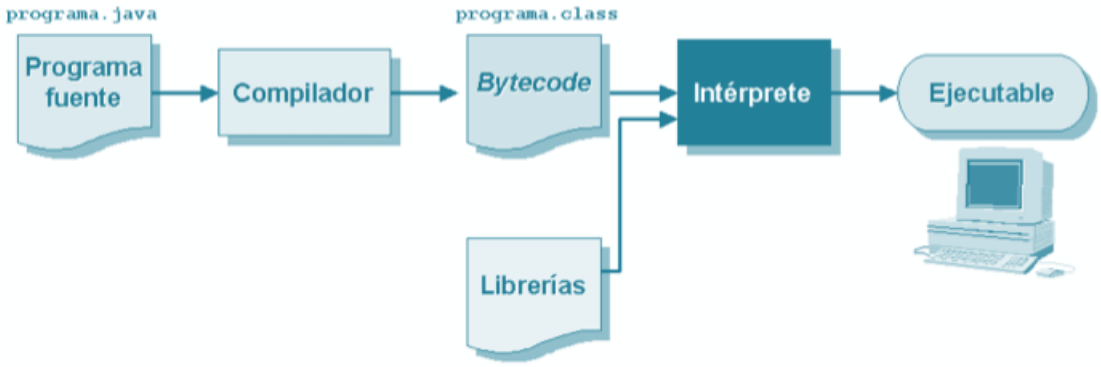


Figura 10: Funcionamiento Java

3.2 XML

XML son las siglas en inglés de "eXtensible Markup Language" y fue creado World Wide Web Consortium.Este lenguaje se caracteriza por el uso de etiquetas.

Se podría decir que XML no es un lenguaje en sí, sino una manera de definir multitud de lenguajes. De hecho no sólo se usa en internet, si no que se ha convertido en un estándar para el intercambio de información estructurada entre diferentes plataformas. [4]

Admite que cualquier aparato adaptado a XML se pueda comunicar con otro para el intercambio de información. Sus características más importantes son que permite separar el código de la presentación del mismo y que es completamente extensible mediante el uso de nuevas etiquetas creadas por el desarrollador.

Este ha sido el lenguaje empleado para definir los layout que formarán la interfaz de las pantallas de la aplicación. Posteriormente se detallará más información de las utilidades de este lenguaje utilizadas en el proyecto.

3.3 Bases de datos.

Las bases de datos permiten almacenar datos de manera estructurada gracias a registros y campos con la menor redundancia posible. Se componen del hardware, constituido por dispositivos físicos de almacenamiento; el software, compuesto por el sistema DBMS; y como último componente, el conjunto de datos, los cuales serán almacenados y procesados para convertirse en información.

Existen gran variedad de bases de datos pero nos centraremos en las de ámbito informático ya que son las que nos interesan para este proyecto. Encontramos bases de datos jerárquicas, de red, las orientadas a objetos pero en mi caso me centraré en las relacionales que son las que voy a utilizar.

Una base de datos relacional es un conjunto de tablas formadas por filas y columnas .Los filas o registros forman los diferentes objetos de la tabla mientras que las columnas las características (variables) de los objetos.

Hay diferentes bases de datos de tipo relacional como por ejemplo MySQL, Oracle o SQL Server. Nos centramos en explicar las características de SQLite ya que es el motor de base de datos que he elegido para este proyecto.

Figura 11: Imagen SQLite y Android

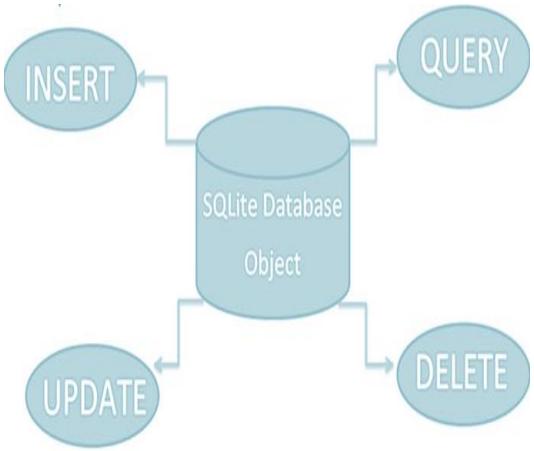
3.3.1 SQLITE

Es un gestor de base de datos SQL transaccional. Gracias a sus características principales, que voy a explicar a continuación, se diferencia de otros gestores de bases de datos proporcionando grandes ventajas sobre ellos.

- **Configuración sencilla:** No requiere configuración de rutas, tamaños, puertos, entre otros puntos que sí que son necesarios para otros gestores de datos.
- **No demanda el soporte de un servidor:** Contiene una serie de librerías que se encargan de la gestión en lugar de un servidor.
- **Es Software Libre:** es de código abierto por lo que todos los archivos de compilación se encuentran disponibles para cualquiera que lo desee.
- **Genera un archivo para el esquema:** SQLite almacena toda la base de datos en un archivo único multiplataforma, siendo este punto una gran ventaja en cuanto a temas de seguridad y migración.
- **Almacena los datos de forma persistente:** Permite que aunque se apague el dispositivo una vez se encienda los se encuentren correctos en la aplicación.

3.3.2 Funcionamiento

Cuando queremos trabajar con bases de datos necesitamos crear, actualizar y otras operaciones, se necesita crear una clase SQLiteOpenHelper. Esta clase sirve para gestionar la creación de bases de datos y la administración de versiones. Proporciona los métodos OnCreate y OnUpgrade



- El método onCreate (SQLiteDatabase SQLiteDatabase) se llama solo una vez durante el ciclo de vida de la aplicación.
  Se llamará siempre que haya una primera llamada a la función getReadableDatabase () o getWritableDatabase () disponible en la clase SQLiteOpenHelper. Así que esta clase llama al método onCreate () después de crear la base de datos y crear una instancia del objeto SQLiteDatabase. El nombre de la base de datos se pasa en llamada de constructor.

Figura 12: Opciones bases de datos

- El método onUpgrade (SQLiteDatabase db, int oldVersion, int newVersion) solo se llama cada vez que hay una actualización en la versión existente. Entonces, para actualizar una versión tenemos que incrementar el valor de la variable de versión pasada en el constructor de la superclase.

3.4 DB Browser for SQLite

Es una herramienta que se emplea para trabajar con SQLite para crear, diseñar y edita bases de datos. Fue creado por Tabuleiro Produções y llamado como Arca Database Browser. En 2014, fue renombrado a DB Browser for SQLite.Este programa no requiere formalidad con comandos SQL. Esta herramienta se utiliza tanto por desarrolladores como usuarios amateurs, de tal modo por lo que su uso debe seguir siendo sencillo

Nos permite crear y compactar archivos de bases de datos; crear, definir, modificar y eliminar tablas; crear, definir y eliminar índices; examinar, editar, agregar, buscar y eliminar registros, entre otras tantas funciones. [5]

Figura 13: Ícono DB Browser for SQLite

3.5 CANVA

Canva es una herramienta de diseño con la que podemos crear y publicar diversidad de diseños bonitos y elegantes sin necesidad de utilizar herramientas con un manejo más complejo como Photoshop o Illustrator. Esta aplicación tiene grandes ventajas una de ellas es que tiene un software libre por lo que es accesible para toda tipo de usuarios.

Figura 14: Ícono Canva

Con esta herramienta he elaborado el icono de aplicación, el fondo de pantalla y algunos de los botones dentro de la aplicación.

3.6 Android

Con el paso de los años ha coexistido gran variedad de sistemas operativos pero en la actualidad son IOS y Android los que disponen de la mayor cuota de mercado, siendo Android la más puntera con diferencia.

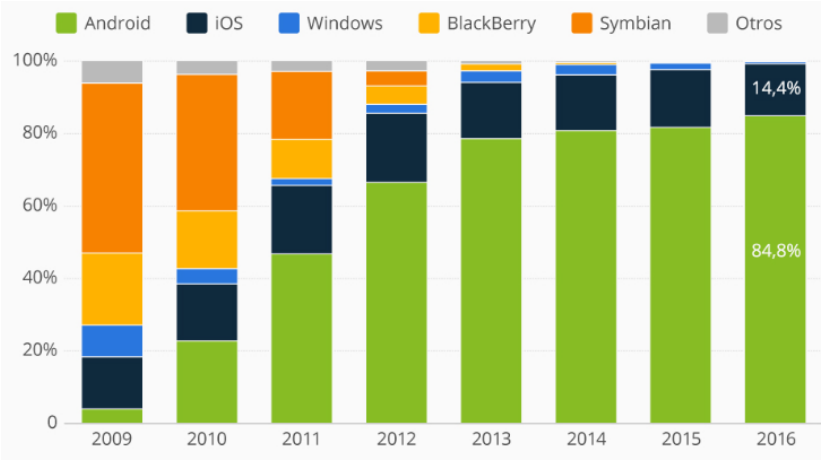


Figura 15: Diagrama de barras uso de S.O

Es por esto que he decidido utilizar Android como sistema operativo ya que por ser el sistema con mayor número de aplicaciones también existen más facilidades para elaborar la aplicación.

3.6.1 Historia

En 2003, concretamente en octubre, Rich Miner, Andy Rubin, Nick Sears y Chris White creaban Android Inc. En sus comienzos, la actividad de la empresa se centraba en "el desarrollo de software para teléfonos móviles". Android Inc. trabajó en la sombra hasta que Google reclutó algunas "startup" del sector móvil.

Tiempo después Google se hizo con Dodgeball, empresa que en 2009 dio paso a Google Latitude. El 5 de noviembre de 2007 Android Inc. fundó la OHA, una alianza de 35 componentes liderada por Google. Desde ese día se crea Android como la plataforma que conocemos hoy en día.

Figura 16: Historia Android

Google ha sido el encargado de publicar la mayor parte del código fuente del sistema operativo, gracias a Apache, una fundación que da soporte a proyectos software de código abierto. Un año más tarde vimos funcionando por primera vez un HTC Dream, un móvil con la primera versión de Android, la 1.0., el modelo G1 de HTC, un móvil deslizable hacia el costado con teclado QWERTY y una gran pantalla sensible al tacto.

Ha pasado mucho tiempo y han ido habiendo un gran número de cambios de los que destacan la plena integración de los servicios de Google, el navegador Web compatible con HTML y XHTML en la primera versión de Android y el estreno en 2008 la tienda de aplicaciones para Android, llamada Android Marquet (actualmente Google Play). [7]

3.6.2 Arquitectura

En este apartado se dará una visión general por capas de la arquitectura Android. Cada una de estas capas utiliza servicios ofrecidos por las anteriores, y a su vez ofrece los suyos a las capas superiores [8].

Esta arquitectura se muestra en la siguiente figura:

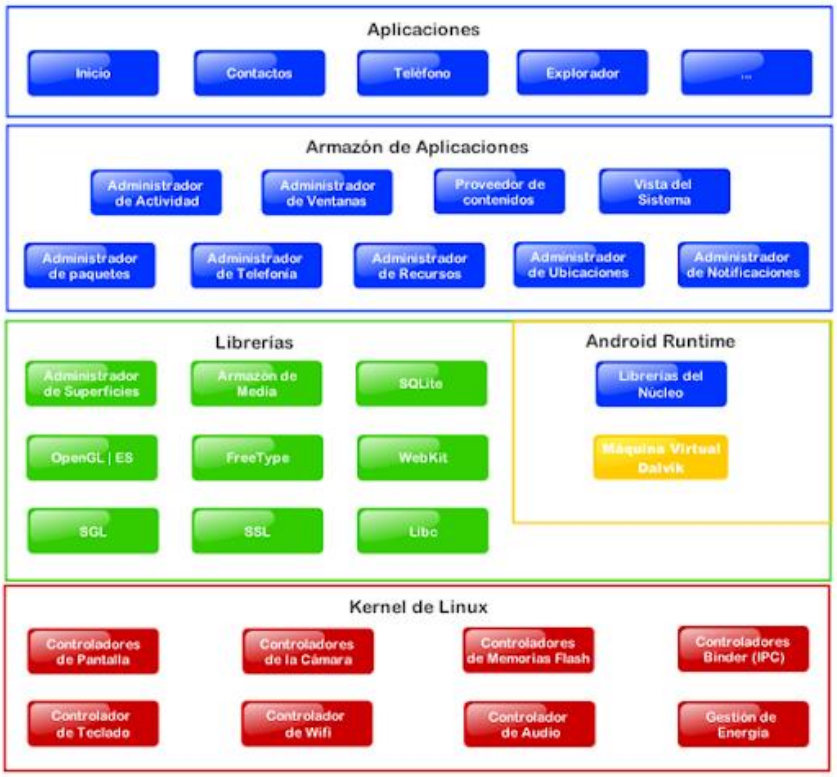


Figura 17: Arquitectura Android

- **Aplicaciones:** Este nivel contiene las aplicaciones incluidas por Android así como aquellas que vaya diseñando los usuarios.

- **Framework de Aplicaciones:** Representa principalmente las herramientas de desarrollo de una aplicación. Una aplicación Android de cualquier tipo utiliza el mismo conjunto de API y el mismo "framework" que conforma este nivel.

- **Librerías:** esta capa la forman todas las librerías utilizadas por Android. Están escritas mediante C/C++ y junto al núcleo Linux constituyen los elementos más importantes de Android.

- **Android runtime:** Al mismo nivel que las librerías de Android se sitúa el entorno de ejecución. Ésta capa está formada por las Core Libraries, librerías de clases Java, y la máquina visual Dalvik.

- **Núcleo Linux**: Esta capa se encarga del hardware disponible en los dispositivos móviles. Destacan en esta capa los drivers

3.7 Android Studio

Android Studio es un entorno de desarrollo para la plataforma Android. Desde el 16 de Mayo de 2013 es considerado el IDE oficial para el desarrollo en entorno Android de aplicaciones, sustituyendo a la plataforma Eclipse.

Figura 18: Icono Android Studio

3.7.1 Estructura

Una vez generamos un nuevo proyecto en la plataforma Android Studio el código quedará organizado como se muestra en la siguiente imagen:

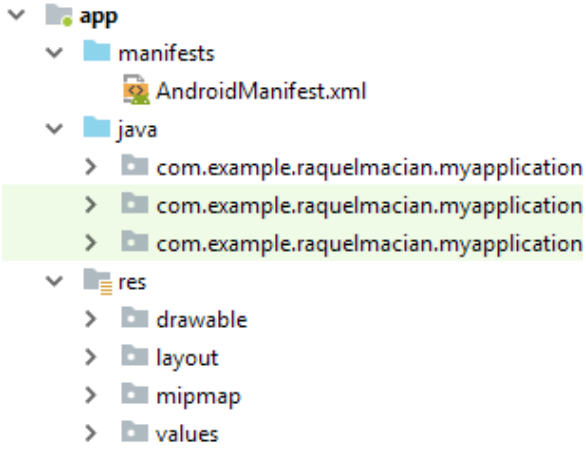


Figura 19: Estructura proyecto

Dentro de la carpeta Manifest se encuentra el archivo AndroidManifest.xml el cual lista todos los elementos de la aplicación. En este archivo se incluyen la API mínima, los componentes de la aplicación, permisos y otros muchos elementos indispensables para el funcionamiento de la aplicación.

Si seguimos desglosando la estructura del proyecto tenemos que tener en cuenta el concepto de actividad. Cuando hablamos de actividad en un entorno Android nos referimos a aquello que se muestra en una única pantalla dentro de nuestra aplicación. Toda actividad estará formado por su interfaz de usuario (archivo XML declarado en la carpeta layout) y su parte lógica (archivo .java almacenado en la carpeta java).

3.7.2 Parte lógica

Una actividad puede estar principalmente en tres estados: en ejecución (se encuentra en el primer plano de la pantalla y tiene la atención del usuario), en pausa (otra actividad está en primer plano, pero esta todavía está visible) o detenida (la actividad está completamente eclipsada por otra actividad).

Cuando una actividad va cambiando entre los distintos estados, se comunica a través de diferentes métodos. En conjunto, estos métodos definen el ciclo de vida completo de una actividad. La siguiente figura muestra el camino que debe tomar una actividad entre estados [10]:

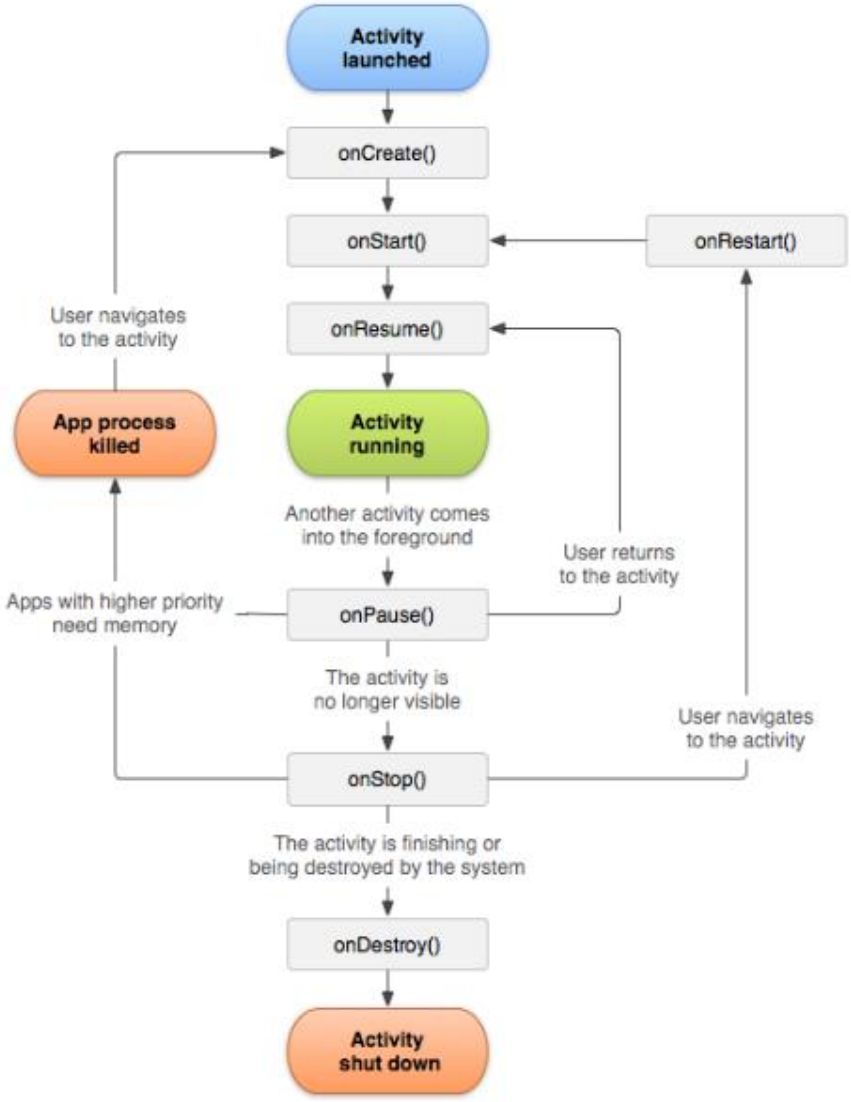


Figura 20: Ciclo de vida de la actividad

3.7.3 Interfaz de usuario

La interfaz de usuario se presenta a través de vistas. Cada vista controla un espacio específico dentro de la ventana de la actividad y responden a la interacción del usuario con el dispositivo móvil.

Android ofrece varias vistas listas, los "widgets", que proporcionan elementos visuales para la pantalla. Hay gran variedad de widgets pero explicaré los detalles de aquellos que he utilizado:

- Button: control con texto o imagen que realiza una acción cuando el usuario lo presiona

- TextView: se utilizan como medio de salida, es decir, para mostrar un determinado texto al usuario

- RecyclerView: permite mostrar en pantalla grandes colecciones de datos

- LinearLayout: es un diseño que organiza otras vistas, ya sea horizontalmente o verticalmente

Estos elementos son los que me han permitido completar el archivo XML correspondiente a cada actividad, además de los archivos que contienen las carpeta drawable y values.Por un lado la carpeta drawlable contiene imágenes externas que podemos incluir en la aplicación mientras que la carpeta values contiene archivos XML con valores predefinidos. Por lo tanto todo lo relacionado con el diseño visual de la aplicación estará almacenado en las siguientes carpetas

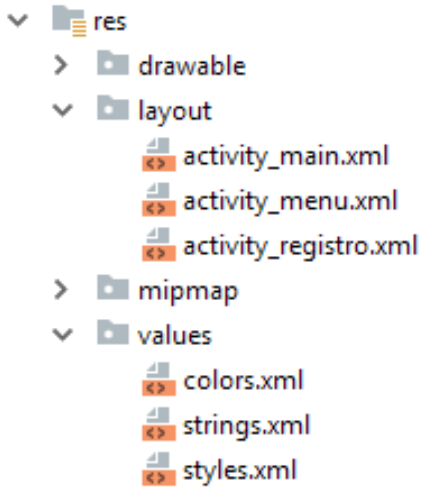


Figura 21: Estructura layout

Capítulo 4. Desarrollo

Dividiremos este capítulo en las dos partes fundamentales que componen este proyecto que son las Bases de datos y las diferentes actividades que la conforman.

4.1 Actividades

Para poder realizar la aplicación he necesitado programar diferentes actividades. Es por esto que en este apartado voy a explicar detalladamente las características fundamentales de cada una de ellas. Todas estas actividades se relacionan siguiendo el siguiente esquema:

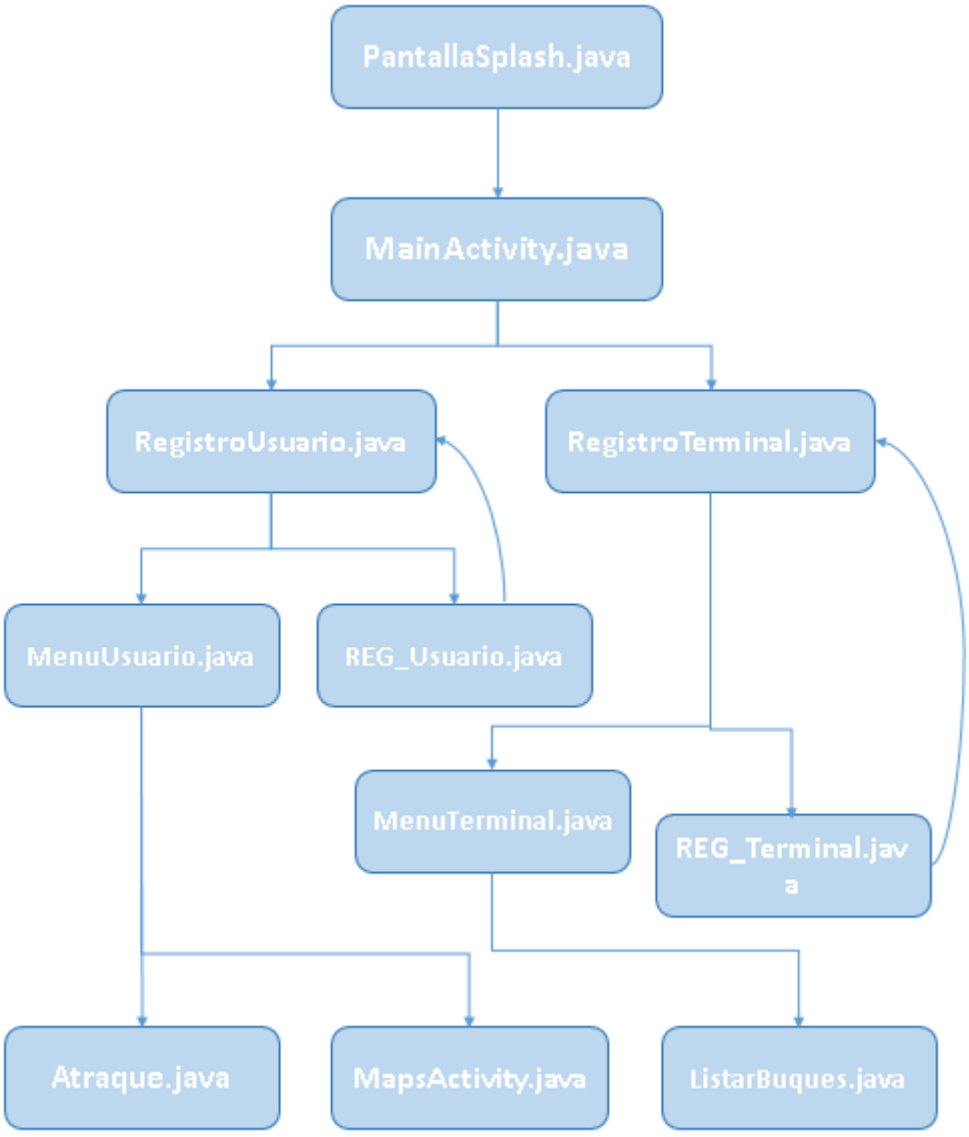


Figura 22: Esquema Actividades

4.1.1 Pantalla de inicio

Para comenzar a utilizar nuestra aplicación deberemos pinchar en el icono .En este caso hemos creado a través del software Canva el siguiente diseño que nos permite acceder a la aplicación:

Figura 23: Ícono aplicación

Posteriormente se mostrara la pantalla de inicio en el que se plasma el icono de la aplicación durante unos segundos. En esta actividad la parte lógica es muy sencilla ya que tan solo hay que mostrar durante unos segundos la pantalla mientras que respecto al layout tan solo definiremos el fondo con el comando background.

```
public class PantallaSplash extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_pantalla_splash);

        new Handler().postDelayed(new Runnable() {
            @Override
            public void run() {
                Intent intent=new Intent(PantallaSplash.this,MainActivity.class);
                startActivity(intent);
                finish();
            }
        },2000);
    }
}
```

Figura 24: Código actividad Splash

4.1.2 Elección del tipo de usuario

Una vez mostrada la actividad anterior el usuario deberá elegir como debe acceder. En caso de ser una terminal pinchara en el "Button" Terminal mientras que si se trata de un buque pinchará en "Button" buque.

Figura 25: Layout main_activity

En este caso la parte lógica también es muy sencilla en la que tan solo habrá que configurar dos botones para poder acceder a la actividad correspondiente. En mi caso yo lo he hecho como se muestra en el siguiente código:

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    terminal = findViewById(R.id.terminal);
    terminal.setOnClickListener((view) → {
            startActivity(new Intent(MainActivity.this,RegistroTerminal.class));
    });
```

Figura 26: Código activity main

Respecto a la parte del diseño hemos modificado el fondo predeterminando a través del comando background y hemos puesto una imagen propia. Por otro lado hemos dado una forma redondeada a los botones también mediante el comando background pero en este caso hemos hecho un código XML propio como el que se muestra en la figura:

```
<corners
android:bottomLeftRadius="40dp"
android:bottomRightRadius="40dp"
android:radius="3dp"
android:topLeftRadius="40dp"
android:topRightRadius="40dp" />

<solid android:color="#ffffff"/>

<stroke
    android:width="5dp"
    android:color="#ff33b5e5"/>


/shape>
```

Figura 27: XMLshape

4.1.3 Registro e inicio de sesión

Para poder entrar al menú pertinente habrá que seguir un proceso de registro. Para hacerlo he creado dos bases de datos que almacenan la información acerca de los usuarios y de las terminales.

En caso de ya estar registrado tan solo deberá introducir las credenciales para entrar al menú. Si nos logueamos como buque deberemos introducir el nombre del usuario y la contraseña mientras que si iniciamos sesión como terminal introduciremos el nombre de la terminal y la contraseña.

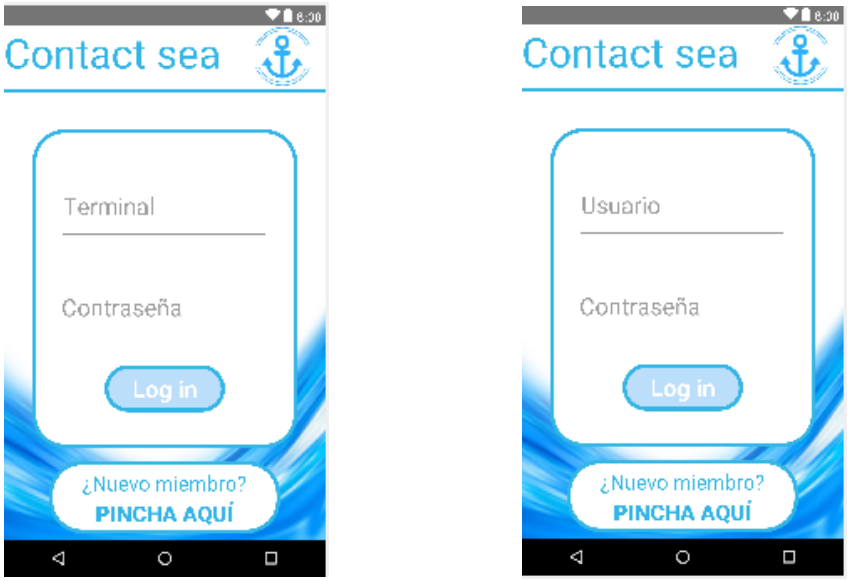


Figura 28: Log in

En caso de que iniciemos sesión entraremos al menú mientras que si todavía no nos hemos registrado deberemos cumplimentar la información (figura 29) para posteriormente iniciar sesión y poder entrar al menú.

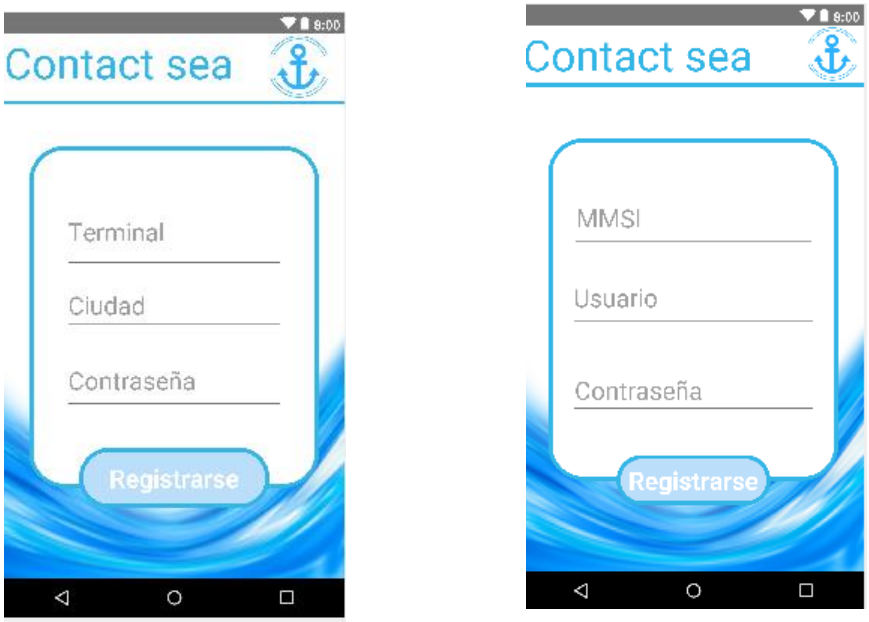


Figura 29: Registro

4.1.4 Menu Usuario

Una vez acabado el proceso de registro estas son las diferentes posibilidades de que tienen los buques

Figura 30: Menu Usuario

Respecto a la parte lógica para las funciones de tiempo y noticias utilizo un intent que nos permite entrar en las respectivas url y así obtener la informacion actualizada sobre el puerto de valencia de la meteorologia y de las noticias portuarias.

Por otro lado los botones de atraque y mapa nos permiten entrar en otras actividades que explicaré más adelante.Parte del código se muestra en la siguiente imagen:

```
atraque=findViewById(R.id.atraque);
atraque.setOnClickListener((view) → {
        startActivity(new Intent(MenuUsuario.this,Atraque.class));
});
```

```
public void tiempo(View view) {
    Intent browserIntent = new Intent(Intent.ACTION_VIEW, Uri.parse("http://www.puertos.es/es-es/oceanografia/Paginas/portus.aspx"));
    startActivity(browserIntent);
}
```

Figura 31: Código Menu Usuario

4.1.5 Atraque

Con esta actividad los usuarios podrán consultar la información actualizada respecto a su atraque. El usuario deberá introducir el MMSI del buque y pulsar en el botón buscar para realizar una consulta a la base de datos. Si se encuentran los datos actualizados almacenados en la base de datos estos se mostraran en los campos, mientras que si no lo están se mostrará un toast informando de que la información no se encuentra disponibles.

Figura 32: Actividad atraque

4.1.6 Maps

Esta actividad tiene una programación distinta ya que tenemos que entrar a la google console para conseguir una clave que nos permita obtener los permisos necesarios para mostrar la posición en que nos encontramos. En el resto de actividades he empleado una actividad vacía mientras que en este caso he utilizado una actividad de tipo mapa ya que viene predefinida por Android Studio

4.1.7 Menú Terminal

Las terminales disponen solo de una funcionalidad. Añadir la información actualizada de los atraques:

Figura 33: Menu terminal

En esta actividad y a través de diferentes botones podemos añadir la información, editarla, borrarla e incluso mostrarla .Para borrar, añadir o editar la información utilizaremos los diferentes TextView y posteriormente daremos a los botones respectivos para realizar la acción. Mientras que para mostrar los datos cambiaremos a la actividad Listarbuques que explicare en el siguiente apartado. Para realizar esta funcionalidad he implementado el siguiente código:

```
public void agregarAtraques(String mmsi, String buque, String fecha, String hora, String atraque){
    SQLiteDatabase bd=getWritableDatabase();
    if(bd!=null){
        bd.execSQL("INSERT INTO ATRAQUES VALUES('"+mmsi+"','"+buque+"','"+fecha+"','"+hora+"','"+atraque+"')");
        bd.close();
    }
}

public void buscarAtraques(BuqueModelo buque, String mmsi){
    SQLiteDatabase bd=getReadableDatabase();

    Cursor cursor = bd.rawQuery( sql: "SELECT * FROM ATRAQUES WHERE MMSI='"+mmsi+"'", selectionArgs: null);
    if (cursor.moveToNext()){
        do {
            buque.setNombre(cursor.getString( columnIndex: 1));
            buque.setFecha(cursor.getString( columnIndex: 2));
            buque.setHora(cursor.getString( columnIndex: 3));
            buque.setAtraque(cursor.getString( columnIndex: 4));
        }while (cursor.moveToNext());
    }
}

public void editarAtraques(String mmsi, String buque, String fecha, String hora, String atraque){
    SQLiteDatabase bd=getWritableDatabase();
    if(bd!=null){
        bd.execSQL("UPDATE ATRAQUES SET BUQUE='"+buque+"',FECHA='"+fecha+"',HORA='"+hora+"',ATRAQUE='"+atraque+"' WHERE MMSI='"+mmsi+"'");
        bd.close();
    }
}

public void borrarAtraques(String mmsi){
    SQLiteDatabase bd=getWritableDatabase();
    if(bd!=null){
        bd.execSQL("DELETE FROM ATRAQUES WHERE MMSI='"+mmsi+"'");
        bd.close();
    }
}
```

Figura 34: Código edición de atraques

4.1.8 ListarBuques

Esta actividad estará formada por un RecyclerView que se irá almacenando por la información que irá rellenando la terminal. Cada buque se irá añadiendo uno tras otro con sus características. La siguiente figura muestra el archivo XML buque que representa el interfaz de cada uno de los buques que se van añadiendo:

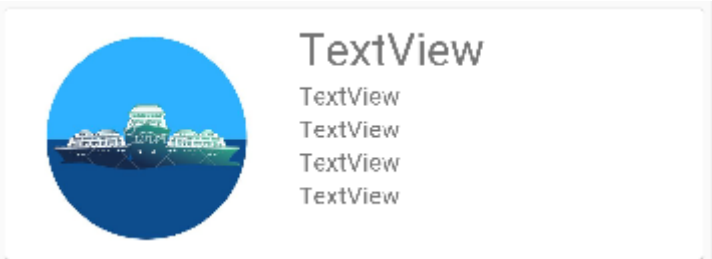


Figura 35: ítem buque

4.2 Bases de datos

Para la realización de la aplicación he necesitado crear tres bases de datos. Do de ellas almacenarán la información acerca de los usuarios y de las terminales mientras que una tercera almacena la información acerca de los atraques.

4.2.1 Buques

Como he mostrado en las actividades de registro los buques deben introducir su MMSI, un usuario y una contraseña; estos dos últimos les permitirán entrar en la aplicación.

Cuando hablamos de MMSI nos referimos a "la serie de nueve dígitos que identifica inequívocamente a cada estación del servicio móvil digital". En nuestro archivo .java quedaran declarada de la siguiente manera:

```
public class DBHelper extends SQLiteOpenHelper {
    public DBHelper(Context context, String name, SQLiteDatabase.CursorFactory factory, int version) {
        super(context, name, factory, version);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL("create table usuarios(codigo integer primary key,usuario text,contrasena text)");
        db.execSQL("insert into usuarios values(01,'admin','admin')");
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        db.execSQL("create table usuarios(codigo integer primary key,usuario text,contrasena text)");
        db.execSQL("insert into usuarios values(01,'admin','admin')");
    }
}
```

Figura 36: Código Base de datos Buques

Hemos declarado la primary key de tipo integer ya que el MMSI son siempre números mientras que el usuario y la contraseña son de tipo texto.

Para comprobar su correcto funcionamiento utilicé el software DB Browser for SQLite de manera que toda la información que los usuarios vayan introduciendo quedarían plasmados de la siguiente manera

	codigo	usuario	contrasena
	Filtro	Filtro	Filtro
1	1	admin	admin
2	193872645	raquel	123456

Figura 37: Base de datos buques

4.2.2 Terminales

De igual manera que los buques esta base de datos presenta la siguiente estructura:

```
public class DBTERMINAL extends SQLiteOpenHelper {
    public DBTERMINAL(Context context, String name, SQLiteDatabase.CursorFactory factory, int version) {
        super(context, name, factory, version);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL("create table usuarios(usuario text primary key,ciudad text,contrasena text)");
        db.execSQL("insert into usuarios values(01,'admin','admin')");
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        db.execSQL("create table usuarios(usuario text primary key,ciudad text,contrasena text)");
        db.execSQL("insert into usuarios values(01,'admin','admin')");
    }
}
```

Figura 38: Código base de datos terminales

En este caso las tres variables son de tipo texto siendo el nombre de la terminal (usuario) la clave primaria de la base de datos.

	usuario	ciudad	contrasena
	Filtro	Filtro	Filtro
1	1	admin	admin
2	MSC	Barcelona	sisi

Figura 39: Tabla base de datos Terminales

4.2.3 Atraques

En este caso la base de datos estará compuesta por el MMSI, el nombre del buque, la fecha de atraque, la hora y el atraque. Posteriormente compruebo el funcionamiento y los valores quedan almacenados de la siguiente manera:

	MMSI	BUQUE	FECHA	HORA	ATRAQUE
	Filtro	Filtro	Filtro	Filtro	Filtro
1	123456789	buque1	12/06/2008	12:00	m21

Figura 40: Tabla base de datos Atraques

Capítulo 5. Conclusiones

Elaborar este proyecto ha sido un proceso largo y complicado. Mis nociones en cuanto a programación Android eran nulas y he tenido que ir adquiriendo estas habilidades de manera autónoma. Me gustaría haber incluido muchas más funcionalidades pero durante la elaboración del proyecto he tenido muchos problemas, sobretodo de tipo hardware. En mi caso no dispongo de un ordenador muy potente y eso ha ralentizado mi trabajo constantemente. Estoy muy contenta con el trabajo final a pesar de las dificultades pero soy consciente que puede tener muchas mejoras y las explicaré en el siguiente punto.

5.1 Líneas futuras

En este punto enumeraré las diferentes mejoras posibles en la aplicación.

- En la funcionalidad de mapas podría crear un servidor externo que almacene la posición de los buques así mostrarlos por pantalla. Podríamos adquirir esa información a través del AIS.
- Mejorar en menú de la terminal mejorando la edición de atraques.
- Insertar algún tipo de notificación para que los usuarios sepan cuando la información está actualizada y así poder entrar en ese mismo instante a la información.
- Dentro de la actividad mapas permitir ir de un sitio a otro incluyendo la ruta.
- Incluir la funcionalidad de brújula.

5.2 Análisis económico

En la actualidad se ha producido un constante aumento del precio del combustible y en consecuencia un aumento de gasto en las embarcaciones. Centrándonos en el ámbito marítimo es muy importante tener en cuenta la velocidad, la cual depende principalmente de la potencia y su desplazamiento. La potencia de los motores depende de su capacidad en HP (Caballos de Potencia) y el desplazamiento del peso del barco. [11]

Existe una regla en ingeniería naval conocida como la velocidad económica, la cual es la velocidad óptima en donde un barco de desplazamiento tiene su mejor eficiencia.

$$VE = 1.34\sqrt{L}$$

$$VE = 1.34\sqrt{L} \quad L = Eslora\_en\_pies$$

Figura 41: Ecuación velocidad óptima

Para poder implementar esta fórmula tenemos que conocer el concepto de eslora. Entendemos como eslora la a longitud de la embarcación y en este caso se expresará en pies (1 metro = 3,28084 pies).

Como ejemplo calcularemos la velocidad optima de un barco que contiene las siguientes especificaciones en su ficha técnica:

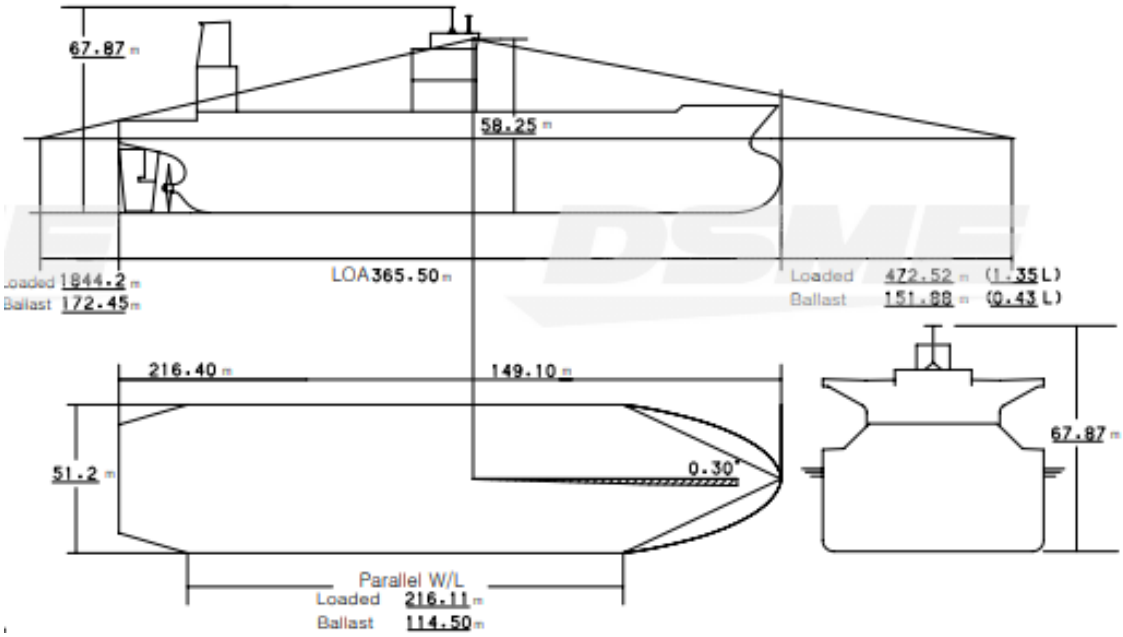


Figura 42: Ficha técnica

En esta imagen podemos ver que la eslora es de 365.50 m por lo que tiene un valor de 1199,147 pies. Aplicando la fórmula de la velocidad eficiente la velocidad para estas especiaciones es de

$$VE = 1{,}34\sqrt{L}\ siendo\ L = 1199{,}147\ pies$$

Para ver de forma visual el ahorro según la velocidad he realizado la siguiente gráfica con valores aproximados ya que el proceso de cálculo de ahorro es más complejo. En este caso he realizado un factor aproximado de 2:1 pero habría que calcularlo en función de dependiendo de su eslora, desplazamiento y potencia de manera más detallada.

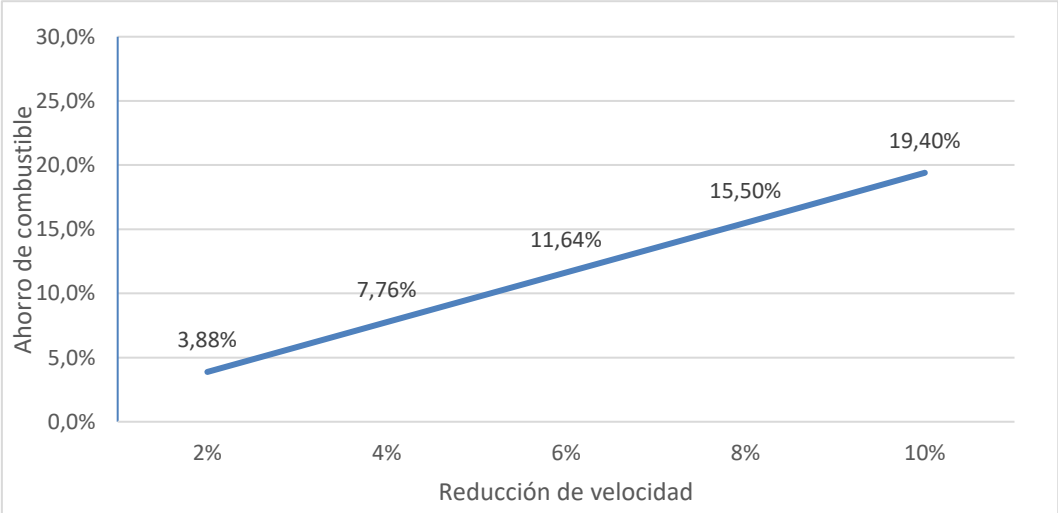


Figura 43: Gráfico de consumo

Anexo

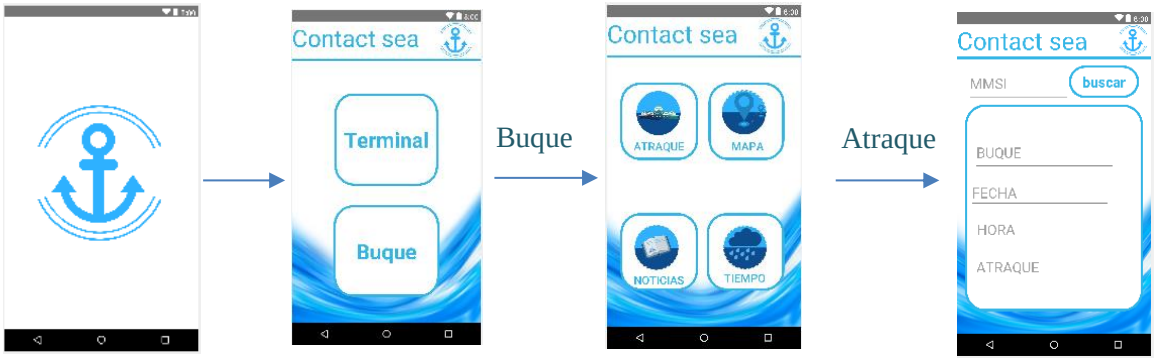
Manual de usuario

En este apartado explicaré el funcionamiento de la aplicación. En primer lugar se mostrará un SplashActivity para posteriormente acceder a los servicios de la aplicación según el tipo de usuario

- **Buque**

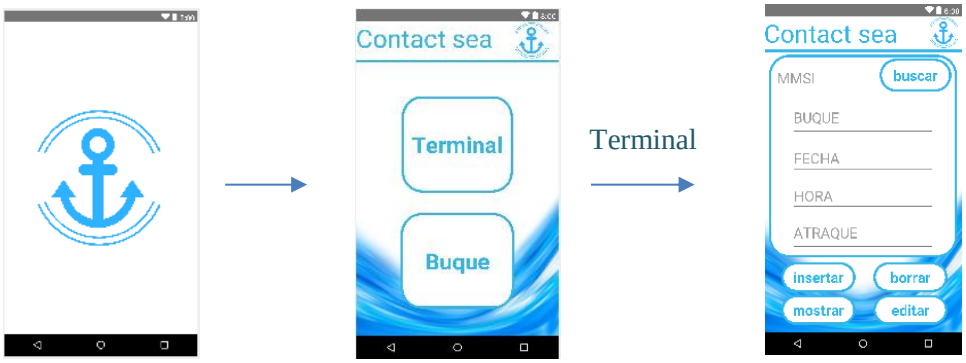
Lo primero que haremos será introducir nuestro usuario y contraseña para poder entrar en el menú principal. En caso de no estar registrados clicaremos en el texto de "Pinche aquí" para posteriormente poder introducir las credenciales y acceder al menú.

Una vez acabado el proceso de iniciado de sesión accedemos al menú que nos ofrecerá cuatro posibilidades. Por un lado tanto el botón de tiempo y noticias nos redireccionarán a dos url correspondientes a la actualidad meteorológica y portuaria respectivamente. Por otro lado a través del botón mapa podremos saber nuestra posición en el mapa mientras que con el botón atraque sabremos la información acerca de nuestro atraque. Esta última actividad permitirá consultar el atraque a través del MMSI llenándose así todos los campos correspondientes a su atraque correspondiente.



- **Terminal**

De igual manera que con los buques deberemos estar registrados para poder acceder posteriormente al menú. Una vez realizado este proceso las terminales ya pueden almacenar la información en la aplicación. En esta actividad se pueden realizar muchas modificaciones en los registros. Para ello lo primero que debemos hacer es buscar a través del MMSI la información acerca del atraque con el que posteriormente podremos trabajar. Podremos ingresar la información acerca del atraque, editarla e incluso borrarla. Además permite a través del botón "mostrar"plasmar los atraques almacenados en la base de datos.



Bibliografía

[1] http://www.ine.es/dyngs/INEbase/es/categoria.htm?c=Estadistica_P&cid=1254735576692

[2] https://es.wikipedia.org/wiki/Metodolog%C3%ADa_de_desarrollo_de_software

[3] https://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n)

[4]https://fsvelectronicainformatica.blogspot.com/2012/12/arquitectura-mvc-android-android-json-o.html

[5] https://sqlitebrowser.org/

[6] https://about.canva.com/es_es/

[7] https://es.wikipedia.org/wiki/Android

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

[9] https://es.wikipedia.org/wiki/Android_Studio

[10] https://developer.android.com/guide/components/activities?hl=es-419

[11] http://www.ricepropulsion.com/TNLS/AhorroComustible.htm