Universitat de les Illes Balears

Escuela Politécnica Superior

Memoria del Trabajo de Fin de Grado

Aplicación Gestor de Actividades para una Unidad de Seguridad en Android "TaskU.S."

Diego Fernando Rivera Ávila

Grado de Ingeniería en Informática

Año académico 2018-19

DNI del alumno: 45695688R

Trabajo tutelado por Ramon Mas Sansò
Departamento de Ciencias Matemáticas e Informática.

Índice

Índice de Figuras

Índice de Tablas

Lista de Acrónimos

U.S.	Unidad de Seguridad
SQL	Structured Query Language
PHP	Hypertext Preprocessor
JSON	JavaScript Object Notation
PDF	Portable Document Format
IDE	Integrated Development Environment
GNU	GNU no es Unix
XML	eXtensible Markup Language
W3C	Word Wide Web Consortium
APACHE	a patchy server
HTTP	Hypertext Transfer Protocol
BSD	Berkeley Software Distribution
GPL	General Public License
LAMP	Linux, Apache, MySQL, PHP
GUI	Graphical User Interface
API	Application Programming interface
XHR	XMLHttpRequest
FCM	Firebase Cloud Messaging

Resumen

Al pertenecer al ejército de tierra durante casi 12 años, pude comprobar que la gestión administrativa interna sigue largos procesos protocolarios, que para poder informar de algún incidente se debía hacer de una manera siempre muy formal, aunque fueran incidentes sin mayor importancia o muy cotidianos.

Pero fue al entrar en la Unidad de Seguridad que pude corroborar que simples peticiones de cambios de guardia (donde solo era necesario que tu compañero estuviera de acuerdo), solicitudes de vacaciones y peticiones de información a tu jefe directo, eran unos procesos tediosos y evitables.

La mejor forma de mitigar estos problemas, era crear una aplicación móvil, que nos permitiera a los mismos integrantes de la unidad poder resolver la mayoría de incidencias que nos pudieran aparecer dentro de la unidad, de esta manera evitaríamos intermediarios (en este caso el jefe de la unidad), pero siempre informados de todos los procesos y por otro lado permitir al jefe de la unidad agilizar sus procesos y de esta manera disminuir el tiempo de respuesta ante las peticiones de los miembros de la unidad.

La aplicación resultante nos permite como tropa de la unidad de seguridad, realizar los cambios de cualquier actividad con otro compañero de manera casi inmediata, estar informado al instante de cualquier novedad que surja por parte del jefe de la unidad o cualquier miembro de la tropa; Como jefe de la unidad, le permite asignar las actividades de manera mucho más rápida y sencilla a cualquier miembro de la tropa, generar los cuadrantes de las guardias y listados del personal de vacaciones de una manera muy simple.

Podemos decir que esta aplicación permite mejorar la relación que existe entre todos los miembros de la unidad, gracias a que nos apoyamos en el uso de la tecnología permitiéndonos convertir largos procesos de solicitudes y peticiones en simples notificaciones al presionar un botón.

Agradecimientos

Quisiera en primer lugar agradecer, a todos los profesores de los que tuve el placer de recibir sus conocimientos, ya que aprendí muchísimo, en especial a mi tutor, ya que fue en una de sus asignaturas en la que me entro el gusanillo por la programación en Android.

También quiero agradecer, a mi familia por todo el apoyo que siempre me han brindado en todo momento.

A Vane por estar siempre y pese a todo ahí, por toda su comprensión y ayuda en los momentos más estresantes de este largo camino y gracias por todos esos ánimos que me dabas para seguir adelante, muchas gracias.

Y, en definitiva, a todos los que me habéis ayudado para que este proyecto fuera posible, compañeros de la universidad, compañeros de la unidad de seguridad "JAIME II", muchísimas gracias.

1. Introducción

En la actualidad nos rodeamos de tecnología constantemente, estamos viviendo una gran revolución de la información que permiten que muchos de nuestros procesos y/o actividades se realicen ya de forma automatizada, lo que optimiza y perfecciona algunas actividades y muchas veces sin ser conscientes de ello, por poner un ejemplo, nuestro dispositivo móvil. Los teléfonos móviles han evolucionado de una forma tan rápida que ya podemos utilizarlos para casi cualquier cosa (aparte de su función principal que es la de llamar), leer el email, usarlo como navegador, ver películas, leer libros, tomar buenas fotografías, redes sociales, escuchar música, pagar en establecimientos.

Está tan arraigado en nosotros que no nos imaginamos el no poder utilizar muchas de las aplicaciones que hoy en día forman parte de nuestra vida cotidiana, ya sea para ver las noticias, el tiempo o comprar las entradas del próximo concierto. El mundo de las aplicaciones ha trasgredido a ámbitos tan complejos como las administraciones públicas que permiten que muchos de nuestros trámites y gestiones, los hemos pasado de realizar presencialmente, rellenando múltiples formularios, haciendo colas interminables, a poder realizarlos cómodamente desde nuestros hogares, oficinas en fin desde cualquier sitio donde podamos tener conexión a la red y además gracias a la tecnología de los teléfonos inteligentes no hace falta en algunas ocasiones el uso de un ordenador.

Pero por desgracia no en todos los sectores se avanza de la misma manera en el uso de las tecnologías para nuestra comodidad; En especial el ejército, ya que al ser un sector que prima bastante la seguridad de la información, es muy cerrado en el hecho de compartirla con el exterior y muy restrictiva en el interior, ya que dentro de una base por ejemplo, se deben llevar a cabo muchos procesos para realizar cualquier trámite interno, por esto muchas de las gestiones que tienen que realizar su personal está todavía muy atrasado con respecto al personal civil.

Una forma de empezar a superar este obstáculo es automatizar algunos procesos y trámites que se realizan en una pequeña parte del ejército, para ser más exactos en una unidad de seguridad de la comandancia general de Baleares; En esta unidad todavía se realizan trámites siguiendo un protocolo de formularios que lo único que consiguen realmente es aumentar el tiempo de respuesta y papeleo que normalmente no tiene mayor utilidad más que mantener tradiciones de antaño.

Ya que tenemos de nuestra parte la tecnología de nuestros teléfonos inteligentes, podemos apoyarnos en su uso para agilizar estos procesos y trámites, queremos permitir que cualquier personal de la unidad, en cualquier lugar sin necesidad de un ordenador pueda realizar los trámites que necesite en ese momento de forma rápida, segura y fiable.

2. Acerca del Proyecto

2.1 Objetivos

Como hemos mencionado en la introducción queremos que una parte importante de las gestiones que el personal de la unidad de seguridad del acuartelamiento JAIME II de la comandancia de las islas Baleares se pueda realizar de manera más cómoda y eficiente, pero en todo momento segura.

Para ello los principales objetivos que queremos conseguir con nuestra aplicación móvil son:

a- Gestionar correctamente los diferentes servicios del personal

Que cualquier miembro de la unidad de seguridad (U.S.) pueda visualizar los diferentes servicios que le han sido asignados por el jefe de los componentes de la unidad como por ejemplo guardias, escoltas, patrullas, etc. Además, pueda solicitar cambios directamente a cualquier otro compañero y que de manera automática se proceda a realizar el cambio.

b- Facilitar la comunicación entre los componentes de la unidad y el jefe del personal

Permitir que trámites como solicitudes de permisos, cancelación de servicios, asignación de vacaciones oficiales, se puedan realizar directamente sin necesidad de acudir a la oficina de la unidad para su trámite.

c- Permitir la comunicación entre todos los componentes de la unidad

Evitar la barrera que existe a la hora de comunicar por motivos de rangos y escalafones dentro del ejército entre tropa y mandos, para ello permitir que toda notificación, anuncio u orden se pueda visualizar por todos los componentes de la unidad, evitando aplicaciones externas.

2.2 Descripción

Frente a los objetivos planteados anteriormente la solución que nos pareció más correcta para ser desarrollada fue crear una aplicación con dos tipos de usuarios (**tropa U.S. - jefe U.S.**) y conectada a un servidor.
La aplicación la desarrollaremos para el sistema Android y permitirá la interacción de los usuarios con los diferentes datos que se generen. Por otro lado, en el servidor se almacenarán los datos que se necesitan, en una base de datos y ejecutara las operaciones necesarias para llevar a cabo los diferentes trámites gracias a servicios web.

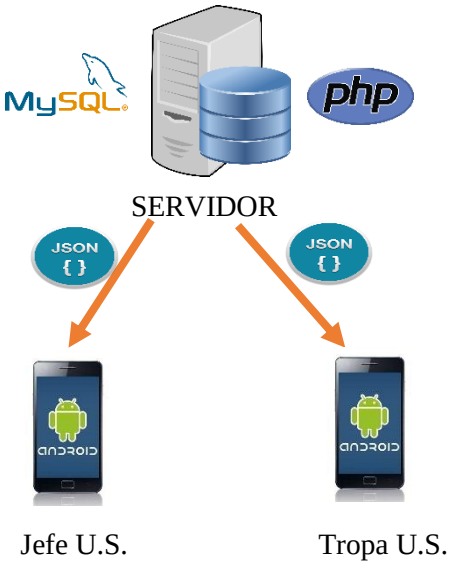

Figura 1. Esquema de funcionamiento del proyecto

Por lo tanto, en el servidor se almacenarán todos los datos necesarios. Utilizaremos el sistema de gestión de base de datos relacional MySQL desarrollado bajo la licencia dual: licencia publica general/ licencia comercial por Oracle Corporation y una de las más populares, sobre todo para entornos de desarrollo web [1].

En el servidor realizaremos las operaciones necesarias para modificar la base de datos, para ello utilizaremos PHP que es un lenguaje de programación de propósito general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico [2], además obtendremos la información preparada desde el servidor para los clientes en formato JSON que es un formato de texto sencillo para el intercambio de datos. Una de sus principales ventajas es que resulta mucho más sencillo el escribir un analizador sintáctico (parser) para él [3].

El objetivo que nos planteamos es el diseño e implementación de un prototipo funcional.

3. Tecnología Utilizada

En este capítulo se explican las tecnologías que hemos utilizado para realizar este proyecto. Todas las tecnologías que se muestran a continuación representan las opciones más importantes que están disponibles actualmente.

Primero analizaremos las herramientas en el sistema Android y luego las herramientas para desarrollar los servicios web.

3.1 Android

Una de las principales razones por la que utilizamos el sistema Android sobre el que vamos a desarrollar nuestra aplicación es que la mayor parte de los integrantes de la unidad de seguridad usan este sistema en sus dispositivos, podemos verlo en la tabla 1.

Militar	Android	IOS
Tropa	18	2
Mando	2	0
	20	2

Tabla 1. Número de dispositivos en la unidad año 2019

También podemos ver una comparativa a nivel mundial en la que vemos que el sistema operativo Android tiene una cuota de mercado muy alta y una previsión para el próximo año, esto nos servirá por si queremos ampliar nuestra aplicación a otras unidades de seguridad en las distintas bases del estado español.

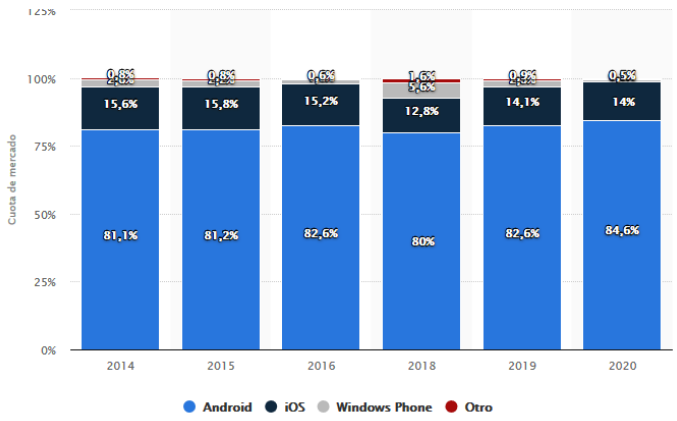

Figura 2.Comparativa de sistemas operativos móviles

Esta estadística muestra en la figura 2 la cuota de mercado de los sistemas operativos para smartphones dentro de los pedidos de smartphones que se realizaron en todo el mundo entre 2014 a 2019 y una previsión para el 2020 [4].

Podemos detallar también en la figura 3 y la tabla 2, que versión del sistema operativo Android es el más utilizado hasta el mes de mayo del 2019 [5].

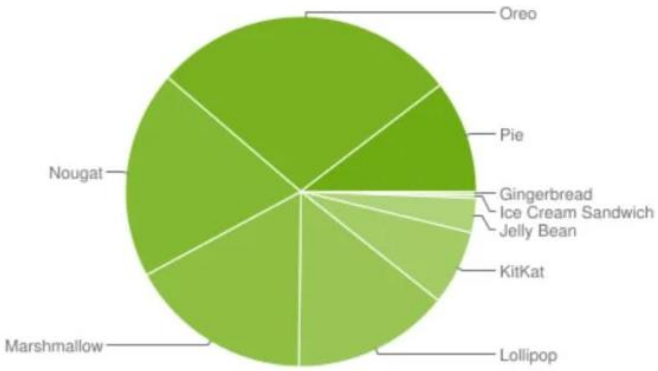

Figura 3. Comparativa de versiones de Android

Version	October 2018 distribution (%)	Current distribution (%)	Change (pp)
2.3	0.2	0.3	+0.1
4.0	0.3	0.3	0
4.1	1.1	1.2	+0.1
4.2	1.5	1.5	0
4.3	0.4	0.5	+0.1
4.4	7.6	6.9	-0.7
5.0	3.5	3	-0.5
5.1	14.4	11.5	-2.9
6.0	21.3	16.9	-4.4
7.0	18.1	11.4	-6.7
7.1	10.1	7.8	-2.3
8.0	14	12.9	-1.1
8.1	7.5	15.4	+7.9
9	N/A	10.4	+10.4

Tabla 2. Porcentaje de uso de diferentes versiones Android

3.1.1 Android Studio

Android studio es el entorno de desarrollo integrado oficial para la plataforma Android. Reemplazó a Eclipse como IDE oficial para el desarrollo de aplicaciones Android.
Está basado en el software IntelliJ IDEA de JetBrains y ha sido publicado de forma gratuita a través de la licencia Apache 2.0 [6].

Está disponible para las plataformas Microsoft Windows, macOS y GNU/Linux. Ha sido diseñado específicamente para el desarrollo de Android.

Las características [6] que se proporcionan en la versión estable actual:

- Integración de ProGuard y funciones de firma de aplicaciones

- Más especificación a la hora de programar
- Renderizado en tiempo real
- Consola de desarrollador: consejo de optimización, ayuda para la traducción, estadísticas de uso.
- Soporte para construcción basada en Gradle.
- Refactorización especifica de Android y arreglos rápidos.
- Editor de diseño enriquecido que permite a los usuarios arrastrar y soltar componentes de la interfaz de usuario
- Herramientas Lint para detectar problemas de rendimiento, usabilidad, compatibilidad de versiones y otros problemas.
- Plantillas para crear diseños comunes de Android y otros componentes.
- Soporte para programar aplicaciones para Android Wear
- Soporte integrado para Google Cloud Pattform, que permite la integración con Google Cloud Messaging y App Engine.
- Un dispositivo virtual de Android que se utiliza para ejecutar y probar aplicaciones

Para desarrollar aplicaciones con Android studio podemos utilizar los lenguajes de Java, Kotlin y XML. Los dos primeros son lenguajes de programación que se pueden utilizar para desarrollar las funcionalidades de la aplicación. El lenguaje de programación Kotlin [7] es un lenguaje de tipado estático que corre sobre la máquina virtual de java y que también puede ser compilado a código fuente de JavaScript. Kotlin está diseñado para interoperar con código java y es dependiente del código java de su biblioteca de clases, tal como pueda ser el entorno de colecciones de java.

Pero para la realización de nuestro proyecto utilizaremos Java porque es un lenguaje mucho más trabajado a lo largo del grado.

El XML se trata de un metalenguaje extensible de etiquetas que fue desarrollado por el W3C, una sociedad mercantil internacional que elabora recomendaciones para la Word Wide Web [8].
El XML lo utilizamos para dar formato a la interfaz que permite controlar la aplicación y permite la interacción del usuario con la aplicación.

Gracias a XML Android studio nos ofrece un entorno para el desarrollo de aplicaciones muy potente y accesible para el usuario. La interfaz que nos ofrece Android studio nos permite desarrollar el diseño de nuestra aplicación de una forma simple e intuitiva, ya que solo con arrastrar podemos colocar los elementos donde queramos en la aplicación. Si en cambio queremos editar manualmente los ficheros XML podemos cambiar de vista sin ningún problema y veremos las etiquetas creadas en nuestro proyecto. Para añadirle funcionalidades a nuestro proyecto escribimos código Java que no es tan simple como el crear ficheros XML pues no podemos arrastrar. Pero disponemos de ayudas como el autocompletar cuando escribimos algún método, podemos ver los errores de sintaxis entre otros.

Por último, tenemos como ayuda toda la documentación que nos ofrece Android en su página web.

3.2 Servidor

Como hemos visto, necesitamos almacenar datos y gestionar servicios web, para ello necesitamos un servidor que los aloje. Lo que hemos elegido para utilizar en nuestro proyecto es un servidor que nos proporciona la empresa Hostalia, mostramos en la tabla 3 las características que tiene nuestro servidor.

Sistema Operativo	Linux CentOS
Espacio en disco	25 GB
Memoria RAM	1 GB
Transferencia	1000 GB
Copia de Seguridad	Si
PHP y MySQL	Si

Tabla 3. Características del servidor

Utilizamos un servidor en esta compañía por las características y precio que nos ofrece, de esta manera podemos utilizar nuestra aplicación fuera de nuestra red local.
Podemos ver que utiliza Linux CentOS el cual es un sistema operativo de código abierto, basado en la distribución Red Hat Enterprise Linux, cuyo objetivo es ofrecer al usuario un software de clase empresarial gratuito [9].

Además, este servidor nos proporciona Apache, MariaDB y PHP. Apache es un acrónimo de *"a patchy server*". Es un servicio de páginas web HTTP de código abierto que sirve para colocar varias plataformas como Unix, BSD, GNU/Linux, Windows, Macintosh. Tiene una amplia aceptación en la red que alcanza su máxima cuota de mercado siendo el servidor empleado en el 70% de los sitios web [10]; con el podremos levantar el punto de acceso donde la aplicación acudirá para enviar sus datos.

MariaDB es un sistema de base de datos que proviene de MySQL, pero con licencia GPL, este sistema nos permitirá guardar los datos recibidos en nuestro servicio web. Por último, PHP es un lenguaje de código abierto para el desarrollo web, se suele describir como lenguaje de lado del servidor, lo que quiere decir, que el código se ejecuta en el servidor y el usuario no es consciente del contenido del script.

3.2.1 MySQL

Es un sistema de gestión de base de datos relacional de código abierto, basado en lenguaje de consulta estructurado (SQL).
MySQL se ejecuta en prácticamente todas las plataformas, incluyendo Linux, Unix y Windows. Se asocia más con las aplicaciones basadas en la web y la publicación en línea y es un componente importante de una pila empresarial de código abierto llamado LAMP [11].

LAMP se trata de una combinación muy popular en los servidores de Linux, que obedece al acrónimo de Linux, Apache, MySQL y PHP.

Para trabajar el esquema relacional que utilizaremos en nuestro proyecto utilizaremos el programa MySQL Workbench.

3.2.1.1 MySQL Workbench

MySQL Workbench es una herramienta visual de diseño de base de datos que integra desarrollo de software, administración de base de datos, diseño de base de datos, gestión y mantenimiento para el sistema de base de datos MySQL. Es el sucesor de DBDesigner 4 de fabFORCE.net, y reemplaza el anterior conjunto de software, MySQL GUI tools Bundle [12].

Las características generales destacadas de MySQL Workbench 5.2 son:

- Conexión y gestión de instancia de base de datos.
- Asistente de elementos de acción
- Completamente compatible con guiones escritos en Python y Lua.
- Soporte para complementos personalizados

MySQL Workbench es uno de los primeros productos de la familia MySQL que ofrece dos ediciones diferentes – una de código abierto y otra edición comercial.

3.2.2 PHP

PHP es el acrónimo de Hipertext Preprocesor. Es un lenguaje de programación del lado del servidor gratuito e independiente de plataforma, rápido, con una gran librería de funciones y mucha documentación [2].
Un lenguaje del lado del servidor es aquel que se ejecuta en el servidor web, justo antes de que se envíe al cliente.

En nuestro servidor podemos utilizar la versión PHP 7 la cual nos trae las siguientes novedades [13] con respecto a la versión 5.6:

- Tipado en el retorno de funciones
- Declaración de tipos escalares
- Quitar warnings de date.timezone
- Arreglar el comportamiento de “foreach”
- Mayor rapidez al parsear parámetros en la API.
- Sintaxis de escape de punto de código Unicode
- Operador ternario con isset
- Comportamiento de enteros
- Corrige fallo desbordamiento ZPP
- Árbol de sintaxis abstracta

3.2.3 API

Podemos definir que una API es un conjunto de funciones, subrutinas y procedimientos que ofrece cierta librería para ser utilizado por otro software [14].
Una API representa la capacidad de comunicación entre componentes de software. Se trata del conjunto de llamadas a ciertas bibliotecas que ofrecen acceso a determinados servicios.

3.2.3.1 Temperatura

En nuestra aplicación utilizamos una API para obtener la temperatura del acuartelamiento *JAIME II*, que es donde está ubicada la unidad de seguridad para la que desarrollaremos la aplicación. *Dark Sky API* es la empresa que realiza esta API, la cual nos permite buscar el tiempo en cualquier sitio del mundo devolviéndonos en formato JSON:

- Condiciones actuales de tiempo
- Las previsiones minuto a minuto hasta una hora
- Las previsiones de hora a hora y día a día hasta siete días
- Las observaciones hora a hora y día a día hasta décadas.
- Las alertas meteorológicas severas en EEUU, Canadá, UE e Israel

Formato de petición

```
https://api.darksky.net/forecast/[key]/[latitude],[longitude]
```

y en la respuesta podemos obtener datos como, por ejemplo:

- Temperatura
- Icono temperatura
- Intensidad de precipitación
- Probabilidad de precipitación
- Humedad
- Presión
- Velocidad del viento
- Visibilidad

Para poder utilizar esta API, hemos tenido que registrarnos en su página web y obtenemos acceso solo a algunas características en la versión gratuita, pero para nuestro proyecto es suficiente.

3.2.3.2 Chat

En nuestra aplicación necesitaremos crear un chat para la comunicación interna entre todos los componentes de la unidad, por ello utilizamos esta API https://www.scaledrone.com/ el cual tiene las características como mensajes en tiempo real, historial de mensajes.

La API scaleDrone utiliza webSockets mejorados cuando es posible y recurre a tecnologías como streaming XMR (XmlHttpRequest), sondeo JSONP y sondeo XHR cuando es necesario.

Para utilizar esta API tenemos que registrarnos en su página web y utilizamos la versión gratuita, la limitación que tenemos al utilizar la versión gratuita es que el número máximo de usuarios conectados al mismo chat son 20 y el número máximo de mensajes en el chat en un día son 144 mensajes, para nuestra aplicación es más que suficiente.

3.2.3.2 Notificaciones Push

Para poder enviar notificaciones a todos los usuarios conectados con nuestra aplicación utilizamos FCM (Firebase Cloud Messaging), el cual nos permite notificar a una app cliente por ejemplo un correo electrónico nuevo o que hay datos para sincronizar. En nuestra aplicación lo utilizaremos

para enviar mensajes de notificación para informar a los miembros de la unidad los cambios, asignación de actividades, cancelación de actividades rechazadas, etc.

Una implementación de FCM incluye dos componentes principales para enviar y recibir datos [15]:

- Un entorno de confianza como Cloud Functions para Firebase o un servidor de apps para generar, orientar y enviar mensajes
- Una app cliente de iOS, Android o web (JavaScript) que reciba mensajes.

3.2.4 Peticiones Post y Get

El método POST y GET son métodos del protocolo HTTP el cual está compuesto por una petición (request) y una respuesta (response) [16].

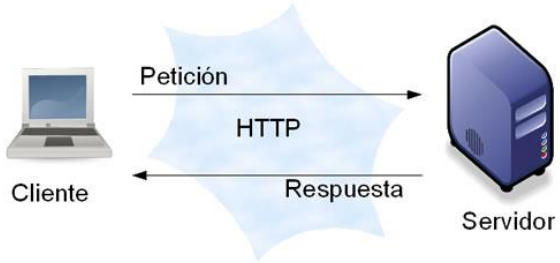

Figura 4. Esquema de funcionamiento de los métodos GET y POST.

En nuestra aplicación utilizamos el método GET para solicitar información del servidor y se nos devuelve en formato JSON.
En cambio, cuando queremos modificar datos de nuestro servidor vía web utilizamos el método POST.

3.2.5 Postman

Este es el programa que utilizaremos para probar la API que vamos a implementar, para realizar todas las consultas de información a nuestra base de datos. Como utilizamos código PHP, sabemos que este necesita recibir peticiones externas para comprobar que funciona correctamente.

Gracias a Postman podemos comprobar cuál es su cabecera y lo que tiene el cuerpo del mensaje ya que tiene una interfaz muy intuitiva, solo necesitamos ingresar la URL que utiliza la API, que tipo de petición queremos usar, POST o GET y el cuerpo de datos que va a transportar.

Utilizaremos la herramienta Postman para corregir posibles errores de funcionamiento de nuestras funciones creadas en el servidor.

4. Aplicación Android

Antes de entrar en el diseño y desarrollo de la aplicación veremos conceptos importantes sobre elementos necesarios para crear nuestra aplicación en Android. Podemos consultar la página web de Android para aclarar mejor los conceptos https://developer.android.com .

4.1 Activity

Este es el componente principal de la interfaz gráfica de una aplicación Android. A cada activity se le asigna una ventana en la cual muestra la interfaz de usuario, con esta el usuario podrá interactuar para llevar a cabo las diferentes acciones que se hayan programado en la aplicación [17].

Para construir esta interfaz gráfica utilizaremos elementos llamados Views (vistas), en las cuales podremos agregar diferentes elementos como botones, listas, cuadros de texto, etc.
Como norma general se suelen utilizar varios activities en una aplicación y están de alguna manera ligados entre sí. Si hay varios activities ejecutándose, estos se almacenan en una pila y cuando le damos al botón atrás en nuestro dispositivo se extrae el activity de la cima y se reanuda su ejecución.

Cada vez que creamos un activity, se define en el fichero AndroidManifest.xml con la etiqueta <activity>.
Para iniciar un activity, podemos llamar a dos métodos

- startActivity ()
- startActivityForResult ()

Podemos ver en la figura 5 el ciclo de vida de un activity [27]

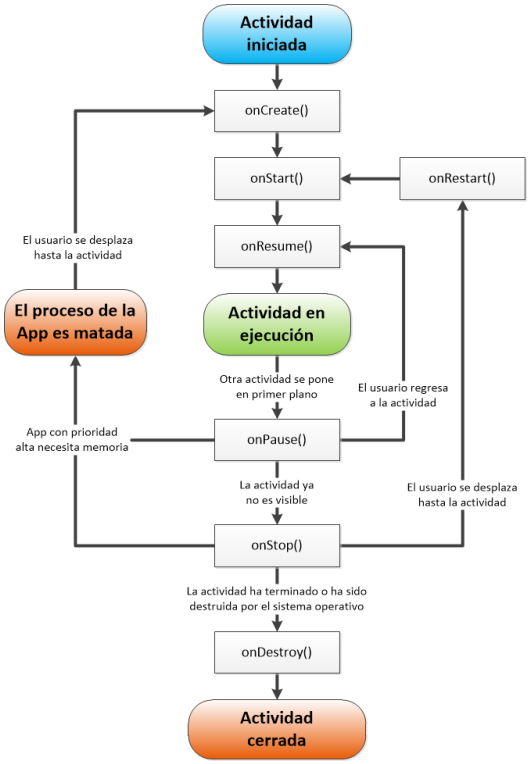

Figura 5.Ciclo de vida de un Activity.

4.1.1 Fragments

Un fragment representa un comportamiento o una parte de la interfaz de usuario en un Activity. Siempre debe estar integrado a una activity y el ciclo de vida del fragment se ve directamente afectado por el ciclo de vida del activity [23].

4.2 Service

Los services (servicios) son componentes que no tienen interfaz gráfica que se ejecutan en segundo plano. Se llaman a través de otro componente, como por ejemplo un activity y aunque el activity finalice su ejecución seguirá el servicio ejecutándose en segundo plano.

También al crear nuevos servicios se declaran en el fichero AndroidManifes.xml utilizando la etiqueta <service> y para iniciar los servicios podemos utilizar estos métodos:

- startService ()
- bindService ()

4.3 Intent

Un intent es un elemento básico de comunicación entre los componentes anteriores, es decir, podremos llamar a un activity, empezar un servicio, iniciar otra aplicación, etc. El uso más importante de los intent es iniciar los activities.

Los objetos intent están formados por un paquete de información. Esta información es importante para el componente que lo recibe, como por ejemplo que acción tiene que realizar y los datos necesarios para realizarla.

4.4 Content Provider

Es un componente destinado a compartir datos entre aplicaciones. Estos datos se pueden almacenar en el sistema de archivos, una base de datos SQLite o en cualquier lugar que nuestra aplicación tenga acceso.

Podemos poner como ejemplo de content provider la forma como Android gestiona la información de los contactos, realizando peticiones al método query () en la interfaz ContentResolver.

4.5 Broadcast Receiver

Es un componente capaz de detectar y luego reaccionar frente a acciones como por ejemplo mensajes globales del sistema, batería baja, SMS recibido, llamada recibida, etc.
En los Broadcast Receiver tampoco muestran ninguna interfaz gráfica.
Al igual que los service y activities también tenemos que registrarlos utilizando alguno de estas dos formas:

- registerReceriver ()
- en el fichero AndroidManifest.xml con la etiqueta <receiver>

Para iniciar el Broadcast Receiver podemos usar uno de estos dos métodos:

- sendBroadCast ()
- sendOrderedBroadcast ()

4.6 Layout

Son la estructura visual que Android utiliza como interfaz de usuario para los activities. Se pueden crear utilizando etiquetas XML o con el editor de Android studio, también podemos crear elementos de diseño en tiempo de ejecución.

Esto nos permite separar la parte lógica de la aplicación de la parte visual y mantener mejor organizado el código que la controla.

4.7 Volley

Es una librería de HTTP que nos permite hacer más fácil y rápidas las conexiones de Android con la red. También cuenta con una programación automática de las peticiones que nos ayudan a evitar escribir sentencias repetitivas.
Podemos ver en la figura 6 el funcionamiento de una petición en volley [18].

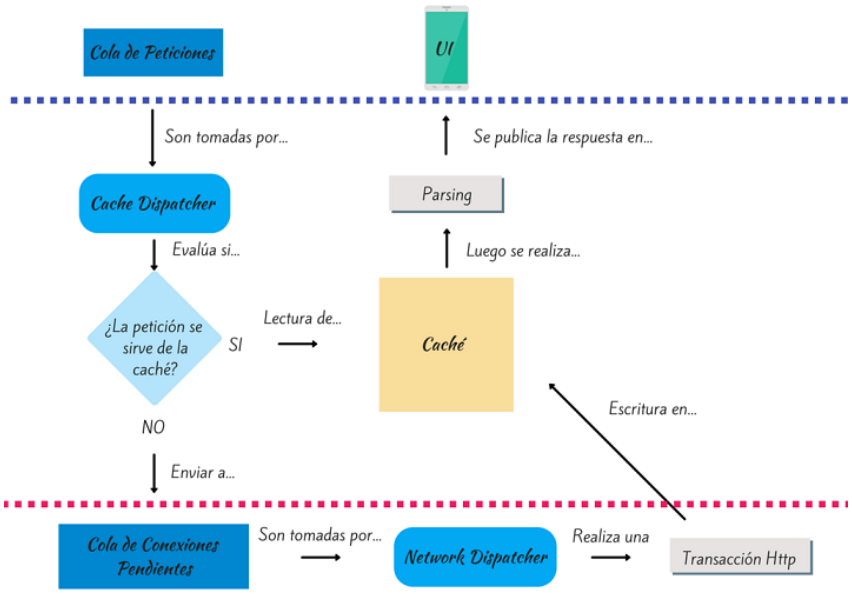

Figura 6. Esquema de una petición en Volley

4.8 Manifest

Es el fichero AndroidManifest.xml que está en la raíz de nuestra aplicación, es un fichero donde se encuentran las configuraciones básicas de la aplicación. Se puede modificar a través de la interfaz gráfica o modificando directamente el propio fichero XML.

En este fichero podemos encontrar:

- Paquete Java de la aplicación
- Descripción de los componentes
- Nombre de las clases que implementan cada uno de los componentes
- Procesos que tienen los componentes
- Define los permisos que se deben dar a la aplicación

4.9 Permisos

Una aplicación básica no tiene permisos asociados, es decir no puede realizar nada que comprometa al sistema u otras aplicaciones. Para añadir permisos debemos modificar el fichero AndroidManifest.xml y al ejecutar la aplicación aparecerán mensajes solicitando al usuario permisos para realizar las acciones marcadas en el manifest.

Algunos ejemplos de permisos en el manifest son:

- “Android.permission.INTERNET”: permitimos abrir conexiones de red
- “Android.permission.ACCESS_FINE_LOCATION”: saber la ubicación, GPS
- “Android.permission.ACCESS_COARSE_LOCATION: saber la ubicación, WIFI

5. Análisis

En este capítulo presentamos la fase de análisis, parte inicial en todo proyecto de software, definiremos los requisitos y veremos una arquitectura global de nuestra aplicación. Después podremos con los requisitos obtenidos entrar en el diseño de la aplicación.
Es en este capítulo donde debemos dejar las bases claras del proyecto para poder construir el sistema correctamente.

Lo primero que tenemos que realizar es definir los requisitos y una metodología de desarrollo acorde con las necesidades de nuestro proyecto.

La arquitectura que tendrá nuestra aplicación será la de cliente-servidor, formado por el servidor (que contiene la base de datos, los servicios web para obtener datos) y el cliente, nuestra aplicación móvil.

A continuación, desglosaremos las funcionalidades y las características del sistema teniendo en cuenta tanto los requisitos funcionales como los no funcionales.

5.1 Requisitos

5.1.1 Requisitos Funcionales

Los requisitos funcionales describen todas las interacciones que nuestros usuarios tendrán con el sistema.

5.1.1.1 jefe de la unidad

Código	Requisitos
RF1	El jefe de la unidad podrá registrar, eliminar o modificar a cualquier militar en el sistema, a través de un formulario.
RF2	El jefe de la unidad podrá asignar y desasignar actividades a miembros de la unidad de seguridad
RF3	El jefe de la unidad podrá crear rangos de fechas de vacaciones oficiales
RF4	El jefe de la unidad podrá asignar o desasignar rango de vacaciones a miembros de la unidad de seguridad
RF5	El jefe de la unidad podrá generar PDF del cuadrante de guardias
RF6	El jefe de la unidad podrá generar PDF de los permisos asignados

Tabla 4. Requisitos funcionales del jefe de unidad

5.1.1.2 tropa de la unidad

Código	Requisitos
RF7	La tropa de la unidad podrá ver las actividades asignadas para el día actual
RF8	La tropa de la unidad podrá ver la información personal relativa a la unidad seguridad.
RF9	La tropa de la unidad podrá ver la próxima actividad asignada
RF10	La tropa de la unidad podrá ver todas las actividades asignadas en el mes actual y el mes siguiente.
RF11	La tropa de la unidad podrá solicitar cambiar actividades con otro miembro de la unidad
RF12	La tropa de la unidad podrá solicitar cancelar actividades al jefe de la unidad

Tabla 5. Requisitos funcionales de la tropa en la unidad

5.1.1.3 jefe y tropa de la unidad

Código	Requisitos
RF13	Cualquier miembro de la unidad de seguridad podrá iniciar sesión en la aplicación introduciendo nombre de usuario y contraseña.
RF14	Cualquier miembro de la unidad de seguridad podrá cerrar sesión en la aplicación
RF15	Cualquier miembro de la unidad de seguridad podrá ver las notificaciones recibidas
RF16	Cualquier miembro de la unidad de seguridad podrá ver los permisos oficiales asignados
RF17	Cualquier miembro de la unidad de seguridad podrá cambiar su contraseña
RF18	Cualquier miembro de la unidad podrá enviar y recibir mensajes a la aplicación

Tabla 6. Requisitos funcionales de los miembros de la unidad

5.1.2 Requisitos No Funcionales

Los requisitos no funcionales son restricciones o condiciones que se imponen al sistema

Código	Requisitos
RNF1	Solo se podrá crear un perfil por militar

RNF2	Se mostrará formulario de inicio de sesión una sola vez hasta que no se cierre la sesión
RNF3	Los datos de usuarios y contraseñas se guardarán en el sistema cifrados
RNF4	Los mensajes recibidos y enviados en la aplicación serán cifrados
RNF5	La interfaz será sencilla, atractiva e intuitiva
RNF6	Se mostrarán notificaciones de errores, alerta y confirmación
RNF7	El tiempo de respuesta de la aplicación no será superior a los 5 segundos, informando de error si se supera este tiempo.

Tabla 7. Requisitos No funcionales

5.2 Metodología de desarrollo

Una metodología de desarrollo es una manera de interpretar la realidad, de hecho, las metodologías de desarrollo se consideran como una estructura utilizada para planificar y controlar el procedimiento de creación de un sistema [19].

Para nuestro proyecto creemos que la metodología de desarrollo más adecuada es la de prototipo la cual permite a los desarrolladores la posibilidad de crear solo la muestra de la resolución para poder validar su funcionalidad ante los clientes y poder hacer los cambios que sean necesarios antes del producto final.

Esta metodología pasa por las siguientes etapas [20] mostradas en la figura 7

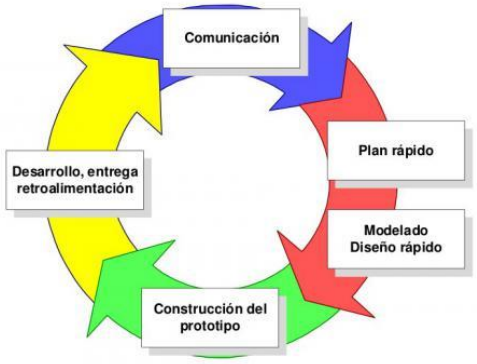

Figura 7. Etapas de metodología de prototipo

Este modelado es muy útil cuando el cliente sabe lo que quiere, pero no detalla los requisitos.
A continuación, podemos ver como desarrollamos esta metodología en nuestro proyecto en la figura 8.

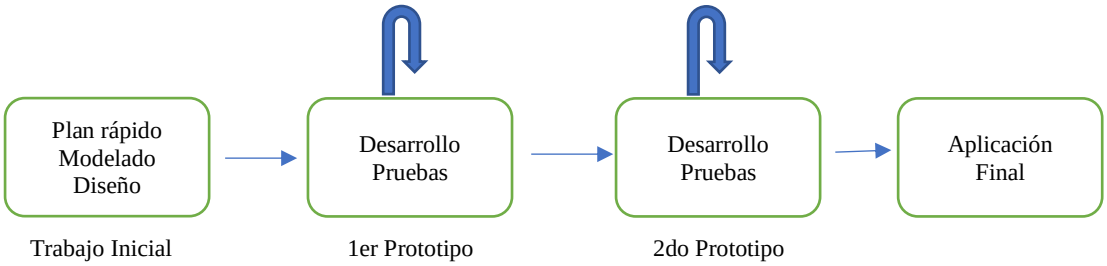

Figura 8. Modelo de prototipo de nuestra aplicación

5.3 Descripción de prototipos

En nuestro primer prototipo creamos la base de datos en el servidor, creamos algunas funciones PHP y la interfaz de usuario de todas las pantallas que se utilizan en la aplicación en modo jefe de la unidad, también desarrollamos las funcionalidades de inicio de sesión, asignar/desasignar actividades, gestión de usuarios, crear/eliminar rango de vacaciones, generar PDF y crear chat de U.S.

En el segundo prototipo creamos la interfaz de todas las pantallas en modo tropa de la unidad, las funcionalidades de obtener temperatura, actividades de día actual, información del soldado, próxima actividad a realizar, visualización de todas las actividades a realizar para el mes actual y el mes siguiente, cambiar o cancelar una actividad, gestión de contraseñas, gestión de notificaciones, gestión de vacaciones oficiales.

En el último prototipo se mejoran funcionalidades en prototipos anteriores, se crean funcionalidades de notificaciones en modo segundo plano, se mejora la interfaz de algunas pantallas.

Cuando se termina cada prototipo se realizan una serie de pruebas para comprobar que cumplen con su finalidad.

5.4 Base de Datos

Como podemos ver en el análisis no podemos crear la base de datos en el mismo dispositivo, lo que nos obligara a implementar una base de datos externa en el servidor y a la que accederemos a través de la API del servidor que crearemos.

En esta base de datos almacenaremos información de los usuarios, actividades, cambios, notificaciones entre otros datos, todo esto lo explicaremos más adelante en el capítulo de diseño.

5.5 Seguridad

Crearemos una pequeña capa de seguridad porque queremos evitar algunos tipos de ataques. Ya que la aplicación se comunica con el servidor a través de peticiones con el protocolo HTTP para enviar y recibir información. Pero esta información puede ser adquirida por personal ajeno con ataques como *Man in the Middle* como contraseñas o nombres de usuario, pero con esta capa, si la información es capturada no le será de utilidad ya que ciframos los datos como contraseñas y nombre de usuarios, también ciframos los mensajes que se envían al chat de la aplicación.

6. Diseño

En este capítulo describiremos todo el proceso de diseño de nuestra aplicación “TaskU.S.”.
Se ha realizado el diseño que cumple todos los requisitos que describimos en el capítulo de análisis. Todo el diseño nos muestra una idea completa de nuestra aplicación.

6.1 Interfaz gráfica de usuario

Para poder desarrollar nuestra aplicación utilizaremos el programa **JustInMind**.
JustinMind es una plataforma de prototipado de alta fidelidad para prototipar aplicaciones web y móvil que son iguales que una aplicación real [21].
JustinMind está llena de funciones intuitivas que ayudan a construir y validar *wireframes* interactivos antes de que empecemos a escribir código.
También se pueden añadir eventos e interacciones y simula los prototipos en dispositivos reales.

6.1.1 Primer prototipo

En este prototipo creamos las pantallas para el usuario jefe de la unidad

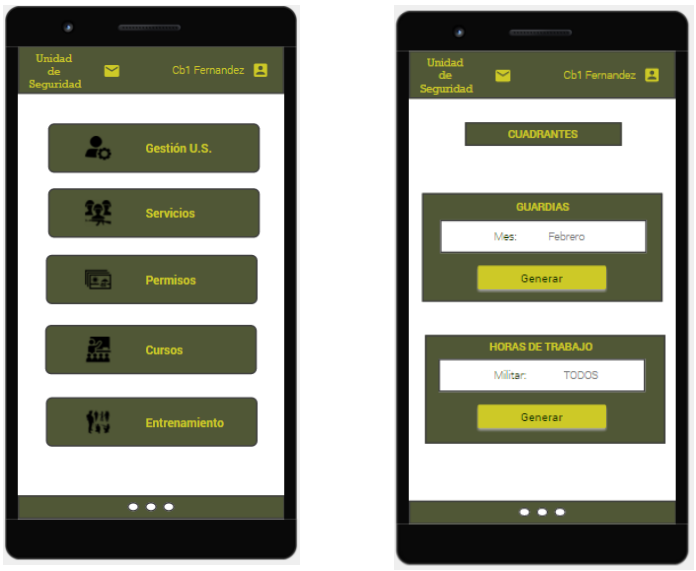

Figura 9. Pantalla inicio y generar PDF

En la figura 9 podemos ver la pantalla de inicio en la que contiene diferentes botones que realizaran las diferentes funcionalidades, gestión de usuarios, gestionar las actividades, gestionar los permisos, también vemos en este prototipo que diferenciamos dos tipos de actividades los cursos y entrenamiento para ser tratados de forma diferente.

También vemos la pantalla en la que podremos crear informes PDF como los cuadrantes de guardias y las horas de trabajo de los militares.

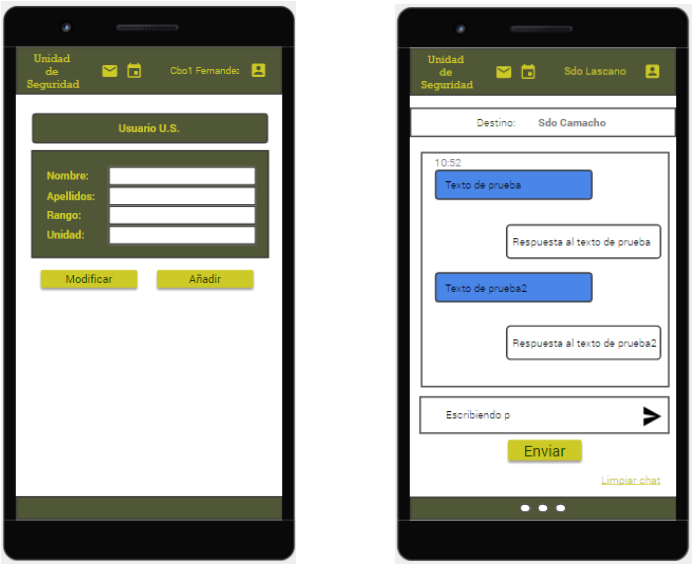

Figura 10. Gestión de militares y chat de la unidad de seguridad

En la figura 10 podemos ver la pantalla en la que el jefe de la unidad puede añadir, modificar a cualquier militar.
También vemos el chat de la unidad, en la que los miembros de la unidad pueden enviar y leer mensajes en la aplicación.

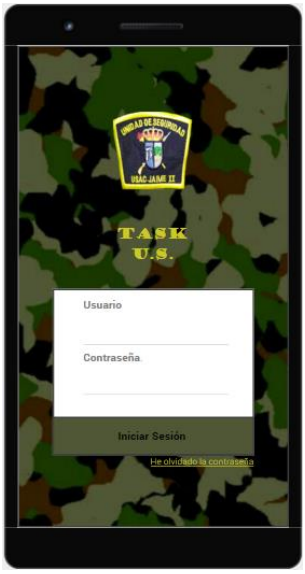

Figura 11. Inicio de sesión

En la figura 11 podemos ver la pantalla de inicio de sesión, que lo podrá utilizar todos los miembros de la unidad de seguridad, vemos el escudo de la unidad y el nombre de la aplicación.

6.1.2 Segundo prototipo

En este prototipo creamos las pantallas para la tropa de la unidad

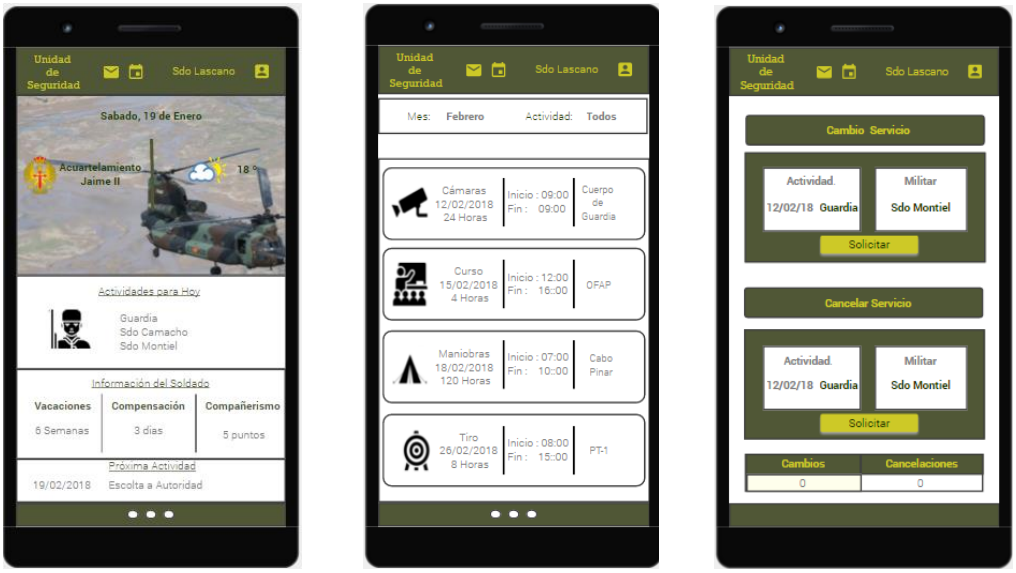

Figura 12. Pagina Inicio, lista de actividades, pagina de cambio de actividades

En la figura 12 podemos ver la página de inicio para la tropa de la unidad, donde podemos ver el tiempo que hace en el acuartelamiento *JAIME II*, también las actividades que se tiene que realizar en el día actual, la información relativa del militar en la unidad, como por ejemplo el número de semanas de vacaciones disponibles, los días libres que tiene asignados y el nivel de compañerismo dentro de la unidad y por ultimo podemos ver la siguiente actividad que tiene que realizar el militar.

Se puede ver la lista de actividades que se tienen que realizar en el mes actual y el mes siguiente dependiendo de la selección en la parte superior y al presionar cualquier actividad saltamos a la siguiente pantalla en la que podremos solicitar cambiar la actividad con otro compañero o solicitar cancelar la actividad al jefe de la unidad.

Figura 13. Página de notificaciones y pagina de cambio de contraseña

En la figura 13 podemos ver la página donde podremos ver las notificaciones que les llegan a los miembros de la unidad de seguridad, como solicitud de cambios de servicio, respuesta de solicitud de cancelación y podemos cambiar la contraseña asignada a cada militar.

6.2 Base de Datos

En este subapartado tratamos el diseño relacionado con la base de datos del servidor. Como hemos mencionado en apartados anteriores utilizaremos MySQL Workbench. Al principio mostramos un diseño modelo entidad-relación y luego mostramos en detalle las tablas creadas.

6.2.1 Modelo Entidad-Relación

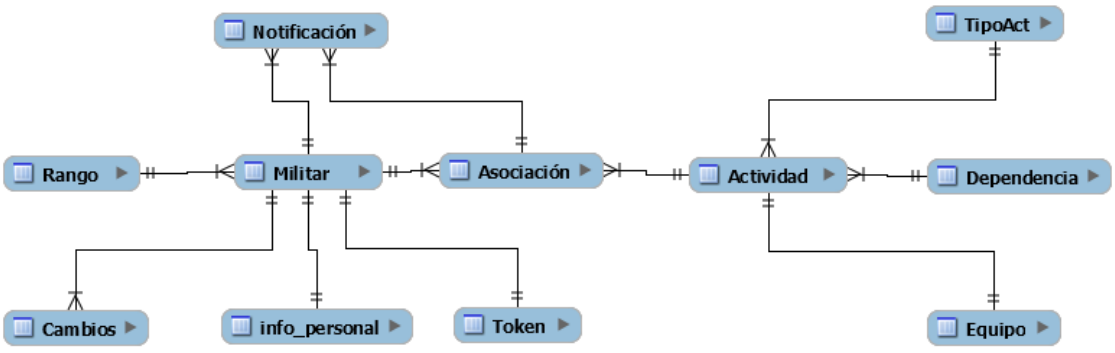

Figura 14. Modelo entidad relación

En la figura 14 podemos ver el esquema entidad relación de la base de datos. Como podemos ver, un militar tiene características a parte de sus datos personales, como un rango ya que existen varias escalas en el ejército hay que definirlas, también tiene una información personal relativa a la unidad de seguridad, como su número de vacaciones disponibles, días de compensación disponibles, etc. También se le define que cualquier militar tiene asignadas solicitud de cambios de actividades, notificaciones de esos cambios y por último guardamos token que son identificadores de los dispositivos físicos donde el usuario inicia sesión.

Podemos definir por otro lado todo lo relacionado con las actividades a realizar en el acuartelamiento como el tipo de actividad, donde se realiza y el equipo necesario para realizarlo.

Por último, podemos relacionar cada militar y las actividades con la tabla asociación.

6.2.2 Tablas de la Base de Datos

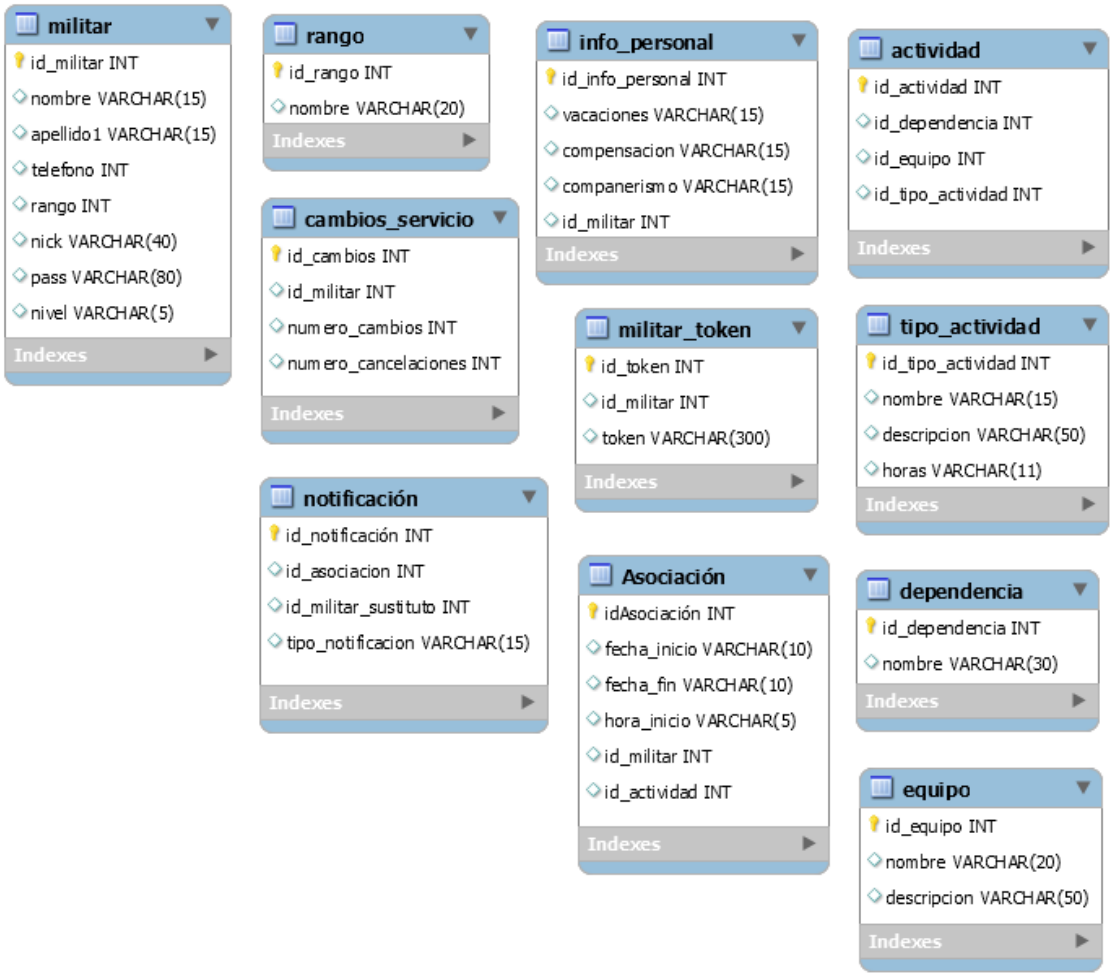

Figura 15. Tablas de la base de datos

- **Militar**
  En esta tabla guardaremos los datos de cada militar, por ejemplo, nombre, apellidos, rango, etc. Y también guardaremos su nick y su contraseña, pero cifrado y además el nivel de acceso para determinar si es Mando o Tropa.
- **Rango**
  Esta tabla nos sirve para almacenar todos los rangos disponibles en el acuartelamiento desde soldado hasta la máxima autoridad en el *JAIME II* que es el coronel. Los rangos generales del ejército no se contemplan en la aplicación por qué no interaccionaran con la aplicación de ninguna forma, ni como actividad, ni como usuario. Si en la actividad "Escolta" se debe realizar a un general, por ejemplo, se utilizará el término siempre "autoridad".
- **Cambios_Servicios**
  Esta tabla nos permite tener siempre a mano el número de solicitudes de cambio de actividad a otro compañero o solicitud de cancelaciones al jefe de la unidad y poder mostrar al usuario y que sea consiente de ese dato para futuras solicitudes.

- **Notificación**
  Esta tabla nos permite almacenar todas las notificaciones de solicitudes de cambios de actividad a un compañero o cancelación al jefe de la unidad, donde se especifica el tipo de solicitud, quien solicita el cambio o cancelación y la fecha de la actividad.
- **Info_Personal**
  Aquí almacenaremos la información relativa con la unidad de seguridad de cada militar por ejemplo las vacaciones, compañerismo, días de compensación.
- **Militar_Token**
  En esta tabla almacenaremos los tokens que generan el servicio de FCM (Firebase Cloud Messaging) al iniciar sesión en cada dispositivo.
- **Asociación**
  Esta tabla nos permite guardar las asociaciones entre militar y las actividades que tienen que realizar
- **Actividad**
  Aquí almacenamos datos necesarios para identificar cada actividad a realizar por los militares dentro del acuartelamiento.
- **Tipo_Actividad**
  Esta tabla nos permite almacenar los diferentes tipos de actividades que se pueden realizar como por ejemplo guardias, escoltas, patrullas, tiro, etc.
- **Dependencia**
  Almacenamos información de los lugares del Acuartelamiento donde se realizan las actividades como cuerpo de guardia, campo de tiro, sala de simulación, oficina de la unidad, etc.
- **Equipo**
  Guardamos todos los equipos que se necesitan para realizar las diferentes actividades por ejemplo para la actividad marcha es necesario casco, mochila, PECO (porta equipo de combate), chambergo, etc.

Además, creamos una tabla auxiliar llamada rango_vacaciones en la cual almacenaremos rango de fechas que definen las vacaciones oficiales que la tropa de la unidad puede escoger.

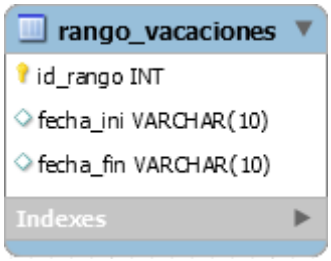

Figura 16. Tabla rango de vacaciones

- **Rango_Vacaciones**
  En esta tabla guardaremos los diferentes rangos de vacaciones disponibles que la tropa puede seleccionar, estos rangos son a decisión de los mandos de la unidad.

6.3 Scripts

Como vimos en el apartado anterior en el análisis, nuestra aplicación se basa en cliente servidor. Ahora en este apartado mostraremos los ficheros que hemos creado para realizar las funcionalidades en el servidor mediante ficheros PHP.
Los scripts php se agrupan por funcionalidades importantes que se necesitan en la aplicación.

Para gestionar todas las funcionalidades de conexión con la base de datos creamos el fichero **BaseDatos.php** con este fichero podemos iniciar la conexión, desconectar, obtener la conexión, etc. Para poder utilizar este fichero hemos creado el fichero **mysql_pw.php** en el cual guardamos datos para realizar la conexión a la base de datos como nombre de usuario, contraseña, nombre de la base de datos, etc. Y este fichero lo guardamos fuera de la carpeta raíz de la aplicación en el servidor en la carpeta “external_includes”.

Para modificar la cantidad de solicitudes de cambios y cancelaciones realizados por los militares se utiliza el fichero **modificarCambiosServicios.php**

Si queremos gestionar los militares como crear, modificar o eliminar militar del sistema utilizamos el fichero **modificarDatosMilitar.php.**
Para cambiar la información personal relativa a la unidad de seguridad como número de vacaciones oficiales disponibles (como norma general los militares tienen 6 semanas de vacaciones a disfrutar desde el 15 de enero a 15 enero del siguiente año), número de días de compensación (son días libres que se otorgan cuando se realizan actividades que superan la jornada normal de trabajo 12 horas) y por ultimo nivel de compañerismo (este dato se incluye en la aplicación para informar a los demás militares la cantidad de solicitudes de cambio de actividades ha aceptado, es decir si es buen compañero) utilizamos el fichero **modificarInfoPersonal.php .**

En cuanto a la gestión de notificaciones de solicitud de cambios a otro compañero de la unidad o la solicitud de cancelación de actividades al jefe de la unidad utilizamos el fichero **modificarNotificaciones.php.** El fichero **modificarRangoVacaciones.php** nos permite añadir, eliminar o modificar el rango de vacaciones oficiales que se puede asignar a cualquier personal de tropa de la unidad.

Si queremos gestionar la relación entre actividades y militar utilizamos el fichero **modificarServiciosMilitar.php** que modifica la tabla de asociaciones.
Cuando se inicia sesión desde un dispositivo móvil se actualiza o si es la primera vez se crea un token que lo identifica y se almacena en la base de datos, para ello utilizamos el fichero **modificarToken.php.**

Por último, tenemos un fichero llamado **Seguridad.php** en el cual hay funciones para encriptar la contraseña y el Nick y un fichero que nos permite controlar el acceso a la aplicación llamado **sesion.php**.

Por otro lado, para obtener toda la información que hay en la base de datos y es necesaria para la aplicación utilizamos los ficheros **webServiceBaseMilitar.php** y **webServiceDatosMilitar.php**.

6.4 Seguridad

Para controlar la seguridad en nuestra aplicación definimos dos puntos a cubrir, en el servidor y en la misma aplicación.

1. Servidor
   Como vimos en el apartado anterior tenemos un fichero que encripta el nick y la contraseña.
   En ambos casos se utiliza la sal, que son bits aleatorios que se usan como una de las entradas en las funciones derivadora de claves [29].
   Para encriptar el nick utilizamos la función hash SHA1 que tiene un bloque 160 bits. SHA1 es más robusto y seguro que MD5.
   Para encriptar la contraseña utilizamos la función hash de php la cual genera un valor cifrado con base a un string.
   De esta manera en la base de datos solo se guardan los hashes.

2. Aplicación
   Dentro de la aplicación creamos una clase llamada Seguridad la cual tiene dos métodos uno para encriptar y el otro desencriptar los mensajes que enviamos al chat de la unidad

Por ahora, la seguridad en nuestra aplicación es básica, se tendría que mejorar al llegar al producto final, pero para conseguir los objetivos propuestos de nuestro proyecto es suficiente.

7. Implementación

En este capítulo desarrollaremos la fase de implementación del diseño que vimos en el capítulo anterior, como vimos en el capítulo de tecnologías utilizadas utilizaremos el IDE de Android Studio y el emulador que trae consigo, además utilizaremos el dispositivo móvil Xiaomi Mi A2 para las ejecuciones de la aplicación y las respectivas pruebas en la parte cliente y en la parte servidor usaremos el servidor asignado por la empresa Hostalia, especificado en el capítulo 3 Tecnologías utilizadas.

7.1 Cliente

Lo primero que vemos es la distribución interna de la aplicación. Mediante el siguiente esquema podemos ver los ficheros java dentro del proyecto Android.

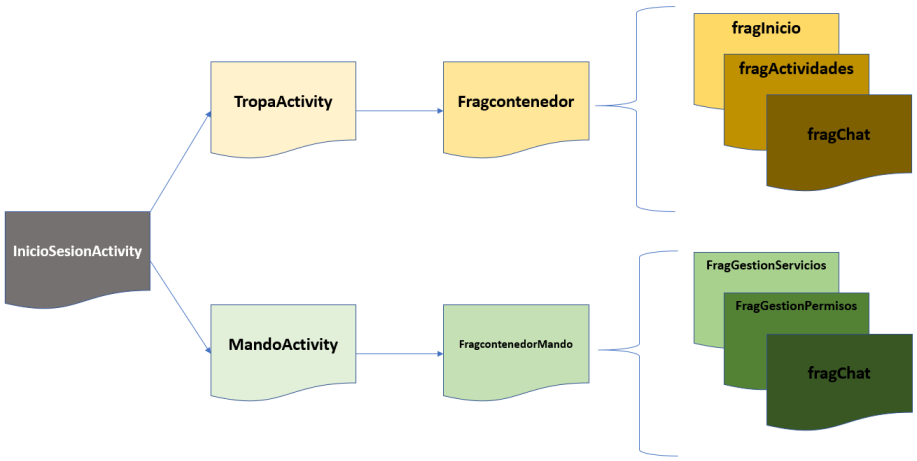

Figura 17. Esquema general de funcionamiento de la aplicación

Como podemos ver en la figura 17 el esquema general de la aplicación es el siguiente:

1- Se inicia el activity *inicioSesionActivity.java* donde se introducen los datos de usuario, nick y contraseña. Se entiende que las credenciales no están guardadas ya en dispositivo y se tiene que mostrar la pantalla, en la figura 18 podemos ver que en Android utilizamos *shared preferences* lo cual nos permite almacenar datos en el dispositivo en forma clave-valor y leerlos después.

```
//comprobamos si ya esta loggueado
Context context = getApplicationContext();
SharedPreferences sharedPref= context.getSharedPreferences(
        "fichero_datos_usuario",Context.MODE_PRIVATE);
String usuarioGuardado= sharedPref.getString("usuario", defValue: null);
String contrasenaGuardada= sharedPref.getString("contrasena", defValue: null);
String idMilitarGuardada= sharedPref.getString( key: "idmilitar", defValue: null);
String nivelGuardado = sharedPref.getString( key: "nivel", defValue: null);
```

Figura 18. Comprobación de datos en el dispositivo

2- Si las credenciales son las de tropa de la unidad, se lanza el activity *tropaActivity.java* en este activity se carga el fragment contenedor y con este los siguientes fragments:

   a. **FragHeader.java**: Es un fragment que se carga como cabecera en la que se muestra el nombre de la unidad, botones para ver las notificaciones y las vacaciones asignadas, también se muestra el nombre del militar y un botón que nos permitirá cambiar la contraseña.
   En la figura 19 podemos ver como creamos los enlaces a los componentes de la interfaz gráfica como los logos que son *ImageView* para poder por ejemplo asignarles listeners.

```
//creamos listener para el logo usuario
ImageView logo = (ImageView)view.findViewById(R.id.logo_usuario);
    logo.setOnClickListener((v) → {

            Intent intent = new Intent (v.getContext(), GestionPerfil.class);
            getContext().startActivity(intent);

    });
    //creamos listener para el logo de gestion de vacaciones
ImageView logoVacaciones = (ImageView)view.findViewById(R.id.imageGestionVacaciones);
logoVacaciones.setOnClickListener((v) → {
        Intent intent = new Intent (v.getContext(), GestionPermisos.class);
        getContext().startActivity(intent);
});

//creamos listener para el logo de notificaciones
ImageView logoNotifi = (ImageView)view.findViewById(R.id.imageGestionNotificaciones);
logoNotifi.setOnClickListener((v) → {
        Intent intent = new Intent (v.getContext(), GestionNotificaciones.class);
        getContext().startActivity(intent);

});
```

Figura 19. Creación de elementos en la cabecera

   b. **FragFooter.java:** Es un fragment que se carga como pie en la que se muestra tres círculos que definen en que pantalla te encuentras cambiando de color al deslizar entre pantallas.

3- Si lanzamos el fragment *FragContenedor.java* lanzamos con este los siguientes fragments:

   a. **FragInicio.java:** En este fragment se cargarán 4 fragments:
      i. FragTiempo.java

En este fragment cargaremos una imagen, la fecha actual, la temperatura y el nombre del acuartelamiento y el escudo del ejército de tierra. En la figura 20 podemos ver como creamos el enlace al icono de la temperatura y obtenemos la imagen que tenemos guardada en la carpeta *"drawable"*, para asignarla al icono.

```
//actualizamos el icono del clima
iconoTiempo= (ImageView)inf.findViewById(R.id.iconoTemperatura) ;
if(DatosFragments.iconoTemperatura!=null){
    DatosFragments.iconoTemperatura=Parser.parserIconoTemperatura(DatosFragments.iconoTemperatura);
    int resourceId = getResources().getIdentifier(DatosFragments.iconoTemperatura,  defType: "drawable", getActivity().getPackageName());
    iconoTiempo.setBackgroundResource(resourceId);
}
```

Figura 20. Pintar la temperatura en el fragment

ii. FragActividadHoy.java
En este fragment se carga las actividades que tiene que realizar el militar en el día actual.

iii. FragInfoSoldado.java
En este fragment cargamos la información personal del militar relativa a la unidad, vacaciones, compensación y compañerismo.

iv. FragProximaActividad.java
En este fragment mostramos el nombre y fecha de la próxima actividad que tiene que realizar el militar.

b. **FragActividades.java:** En este fragment cargamos los siguientes fragments:

i. FragSelector.java
En este fragment podremos elegir el mes del cual queremos ver las actividades en el fragment de las listas de actividades, también podemos filtrar por tipo de actividades. En la figura 21 podemos ver como creamos y rellenamos el spinner que muestra los meses para poder seleccionarlo al querer ver las actividades.

```
//creamos el spinner para dos meses
String[] meses=getMesesActividad();
Spinner sMeses= (Spinner) view.findViewById(R.id.spinnerMes);
sMeses.setSelection(posicion);
sMeses.setAdapter(new ArrayAdapter<String>(getContext(),android.R.layout.simple_dropdown_item_1line,meses));
```

Figura 21. Creación de spinner del mes

ii. FragListaActividades.java
Aquí veremos todas las actividades que se hayan seleccionado con el selector.

c. **FragChat.java:** En este fragment cargamos los siguientes fragments:

i. FragZonaMensajes.java:
En este fragment se mostrarán los mensajes que se escriben en la aplicación por parte de los miembros de la unidad.

ii. FragEnviarMensajes.java:
En este fragment podremos escribir lo que queremos enviar y mostraremos un botón que al pulsar se envía lo que se ha escrito.

4- Si lanzamos el fragment *FragContenedorMando.java* lanzamos con este los siguientes fragments:

a. **FragGestionServicios.java:** En este fragment mostraremos botones y campos para poder asignar/desasignar actividades a la tropa de la unidad también creamos un botón para modificar los perfiles de los militares. En la figura 22 vemos como creamos los enlaces de los componentes de la pantalla para poder trabajar con ellos.

```
//enlazamos el editText a un evento para el datepicker
etFecha = (EditText) view.findViewById(R.id.et_fecha_actividad);
etHoras = (EditText) view.findViewById(R.id.et_horas);
sAct = (Spinner) view.findViewById(R.id.spinner_nom_actividad);
sActividadesCancelar=(Spinner)view.findViewById(R.id.spinner_actividaes_cancelar);
Button btAñadirMilitares = (Button)view.findViewById(R.id.bnAñadirMilitares);
Button btCrearServicio = (Button)view.findViewById(R.id.bnCrearServicio);
Button btGestionMilitar = (Button)view.findViewById(R.id.bnGestionPerfiles);
Button btCancelarServicio = (Button) view.findViewById(R.id.bnCancelarServicio);
DatosProgress.progressBarCrear = (ProgressBar)view.findViewById(R.id.progressBarCrear);
DatosProgress.progressBarCancelar = (ProgressBar)view.findViewById(R.id.progressBarCancelar);
swipeRefreshLayoutGestionServicios=(SwipeRefreshLayout)view.findViewById(R.id.swipeRefreshLayoutGestionServicios);
```

Figura 22. Enlace a componentes de la interfaz gráfica

b. **FragGestionPermisos.java:** En este fragment podemos crear o eliminar rangos de vacaciones, también podremos asignar/desasignar permisos a cualquier miembro de la unidad y por último creamos botones para generar PDF.
Podemos ver en la figura 23 como asignamos los listener necesarios a los componentes de la pantalla.

```
//listener
etFechaRangoIni.setOnClickListener(this);
etFechaRangoFin.setOnClickListener(this);
bnCrearRango.setOnClickListener(this);
bnEliminarRango.setOnClickListener(this);
bnAñadirMilitares.setOnClickListener(this);
bnAsignarPermisos.setOnClickListener(this);
bnMostrarPermisos.setOnClickListener(this);
bnMostrarCuadrante.setOnClickListener(this);
//onRefreshListener
swipeRefreshLayoutGestionPermisos.setOnRefreshListener(this);
//itemListener
spinnerTipoOperacion.setOnItemSelectedListener(this);
spinnerDiaExtra.setOnItemSelectedListener(this);
spinnerListaRangoPermisos.setOnItemSelectedListener(this);
```

Figura 23. Asignación de los listeners a los componentes de la interfaz gráfica

c. **FragChat.java:** Fragment igual que en el *FragContendor.java*. En la figura 24 podemos detallar como notificamos a la aplicación cuando hay cambios en la lista de mensajes que envían los usuarios y la asignación del adapter, para poder transformar correctamente nuestros datos al formato que queremos mostrar.

```
mensajeAdapter.notifyDataSetChanged();
mensajeVista = (ListView) view.findViewById(R.id.lista_mensajes);
mensajeVista.setAdapter(mensajeAdapter);
```

Figura 24. Asignación del adapter de los mensajes

7.1.1 Inicio de sesión

En el activity de inicio de sesión cargamos un fondo, con el nombre de la aplicación "***TaskU.S.***" y el escudo de la unidad. Creamos dos EditText (son componentes que se utilizan en Android para introducir texto) uno para introducir el nombre de usuario y el otro para la contraseña.

También declaramos un Button (componente botón que crea un evento) para lanzar la petición de comprobación de usuario y contraseña en el servidor.

Al lado derecho ponemos un ImageView (componente para cargar imágenes) que al pulsar nos muestra un mensaje de información, indicando que el nombre de usuario y contraseña solo los define el jefe de la unidad.

Por último, podemos ver la versión en la que se encuentra la aplicación.

Figura 25. Pantalla inicio de sesión

7.1.2 Acceso Modo Tropa

Cuando lanzamos el activity de tropa cargamos un *ViewPager* (es un componente que permite desplazarnos entre pantallas deslizando el dedo horizontalmente) con 3 páginas: Inicio, Actividades y Chat que se explicaran en este apartado. En la figura 26 vemos como definimos el ViewPager en la interfaz gráfica.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/fragment_contendor_principal"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">

    <android.support.v4.view.ViewPager
        android:id="@+id/vpPager"
        android:layout_width="match_parent"
        android:layout_height="642dp">

    </android.support.v4.view.ViewPager>
</LinearLayout>
```

Figura 26. Declaración de ViewPager

7.1.2.1 Pantalla Inicio

Esta es la pantalla de inicio, cuando se inicia en modo tropa.
Es un activity al que se le añaden diferentes fragments para diferenciar las diferentes zonas.

Figura 27. Pantalla inicio Tropa

Para crear esta pantalla se implementa en el *activity_tropa.xml* con un *LinearLayout* el cual utiliza los siguientes ficheros para componer la pantalla:

- *fragment_frag_header.xml:*

  Fichero que nos sirve para instanciar la barra superior en la que se muestra la unidad a la que pertenece, podemos ver un icono de carta que nos indicara si hay notificaciones por contestar, un icono de calendario el cual nos permite ver las vacaciones oficiales asignadas, también podemos ver el nombre de militar con el rango y el icono de contacto el cual nos permite cambiar la contraseña.

- *fragment_frag_tiempo.xml:*

  Este fichero nos permite cargar dos ImageView, uno para el icono del clima del tiempo que hace en este momento y otro para el escudo del ejército de tierra, creamos 3 TextView uno para la fecha, otro para el nombre del acuartelamiento y otro para la temperatura.

- *fragment_frag_actividad_hoy.xml:*

  En este fichero creamos un *TextView* donde ponemos el título y un *ListView* donde cargaremos la lista de las actividades a realizar el día actual. En la lista mostraremos las actividades gracias al fichero *fragment_frag_bloque_actividad.xml* el cual contiene un ImageView para poner la imagen de la actividad a realizar, y 3 *TextView* para indicar el nombre de la actividad, y los nombres de dos compañeros con los que tendrá que realizar la actividad.

- *fragment_frag_info_soldado.xml:*

Creamos en este fichero 7 *TextView* para mostrar la información relativa del militar. En la figura 28 podemos ver la implementación de un *TextView.*

```
<TextView
    android:id="@+id/textView5"
    android:layout_width="337dp"
    android:layout_height="28dp"
    android:layout_marginStart="8dp"
    android:layout_marginLeft="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginRight="8dp"
    android:gravity="center"
    android:text="Información del Soldado"
    android:textColor="#000000"
    android:textSize="18sp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

Figura 28. Código TextView del título información del soldado

- *Fragment_frag_proxima_actividad.xml:*

En este fichero creamos 3 *TextView* para mostrar la información de la siguiente actividad.

7.1.2.2 Pantalla Actividades

Esta es la pantalla de muestra de actividades, solo se carga en modo tropa ya que en modo Mando no tiene sentido utilizarla. Para crear esta pantalla implementamos en el fichero *fragment_frag_actividaes.xml* un *LinearLayout* y utilizamos los siguientes ficheros:

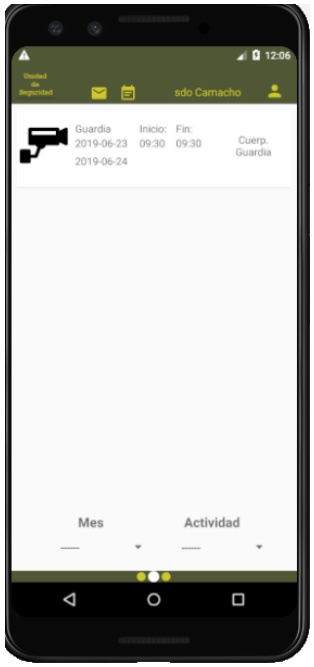

Figura 29. Pantalla Actividades

- *fragment_frag_selector.xml:*

En este fichero creamos dos *TextView* para poner los títulos (mes y actividad) y creamos dos *Spinner* (componente que nos permite mostrar una lista en forma desplegable) para las actividades que se pueden realizar y otro para el mes.

- *fragment_frag_lista_actividades.xml:*

En este fichero creamos un *ListView* para que las actividades aparezcan en forma de lista. Las actividades se muestran gracias al fichero *fragment_frag_bloque_actividad_completa.xml* que nos crea 8 *TextView* para rellenar todos los datos informativos de la actividad y un *ImageView* para el icono de la actividad.

7.1.2.3 Pantalla Chat

En esta pantalla creamos el chat de la unidad de seguridad, se carga en modo tropa y modo mando. Para crear Esta pantalla implementamos en el fichero *fragment_frag_chat.xml* un *LinearLayout* con un *TextView* para el título de la pantalla y cargamos dos fragments que utilizan los siguientes ficheros:

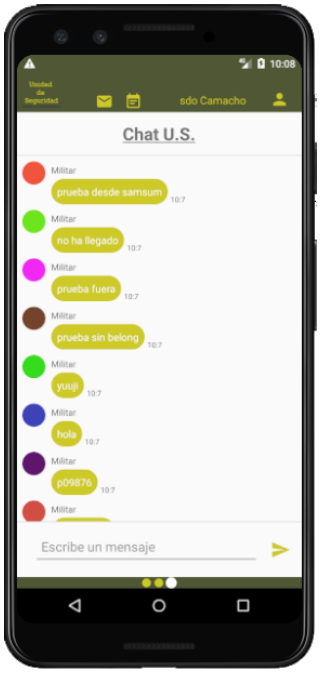

Figura 30. Pantalla Chat U.S.

- *fragment_frag_zona_mensajes.xml:*

  En este fichero creamos un ListView para cargar la lista de mensajes. Para poder ver el formato de los mensajes implementamos dos ficheros, uno para mensaje propio y otro para mensaje de los otros componentes de la unidad.

  - *item_mensaje_enviado.xml*
    En este fichero mostramos el mensaje en una elipse de color gris, mostrando la hora de envío
  - *item_mensaje_recibido.xml*

    En este fichero mostramos el mensaje en una elipse de color amarillo, que pinta circulo y el nombre del militar y la hora de recibido.

- *fragment_frag_enviar_mensajes.xml:*

  Este fichero nos permite escribir y enviar los mensajes al chat de la unidad, implementamos un *EditText* en el cual recibe el mensaje a enviar y creamos un *ImageButton* que al pulsar se enviara el mensaje. En la figura 31 podemos ver como implementamos el *ImageButton*.

```
<ImageButton
    android:id="@+id/boton_enviar_mensaje"
    android:layout_width="28dp"
    android:layout_height="28dp"
    android:layout_marginHorizontal="10dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginRight="8dp"
    android:layout_marginBottom="8dp"
    android:background="@drawable/ic_send_black_24dp"
    android:padding="20dp"
    android:scaleType="fitCenter"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.703"
    app:layout_constraintStart_toEndOf="@+id/et_campo_escribir_mensaje"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.51" />
```

Figura 31. Código del ImageButton para enviar mensaje

7.1.3 Acceso Modo Mando

Cuando lanzamos el activity de mando cargamos un *ViewPager* con 3 páginas: gestión de servicios, gestión de permisos y Chat que se explicaran a continuación.

7.1.3.1 Pantalla Gestión de Servicios

Esta pantalla nos permite asignar/desasignar las actividades a la tropa de la unidad y gestionar perfiles, para el funcionamiento se implementa el fichero *fragment_frag_gestion_servicios.xml* el cual tiene varios componentes *View* (son componentes básicos que ayudan a construir la interfaz gráfica) que dividen la pantalla en zonas con estos *View* se le añaden *TextView* para los títulos y *EditText* para introducir la fecha y hora, *Spinners* para cargar la lista de tipos de actividades y las actividades creadas.

Por último, creamos un botón que nos envía a otro activity que implementamos en el fichero *activity_gestion_militares.xml*, este fichero se explica más adelante.

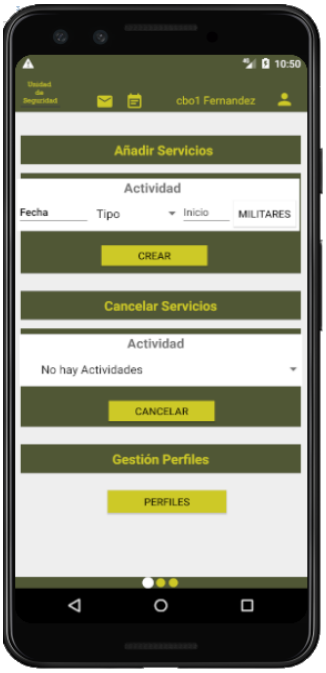

Figura 32. Pantalla gestión de servicio

7.1.3.2 Pantalla Gestión de Permisos

En esta pantalla nos permite crear rangos de vacaciones y luego con esos rangos se pueden asignar/desasignar a los militares, también podemos añadir días extras si se considera necesario y al final de la pantalla podemos generar archivos PDF, el cuadrante de guardias y el listado de permisos asignados. Para implementar esta pantalla lo hacemos en el fichero *fragment_frag_gestion_permisos.xml* en el cual creamos varios *View* para definir las diferentes zonas.

También añadimos *EditText* para añadir las fechas para crear los rangos de vacaciones, añadimos sus *Button* para añadir o eliminar los rangos.
En la zona de asignar permisos creamos 3 *Spinner*, uno para poder seleccionar entre asignar o eliminar rango de vacaciones a los militares, uno para poder seleccionar el rango de vacaciones que se ha creado en la sección anterior y el ultimo *spinner* nos permite añadir días al rango de vacaciones, también le añadimos un *Button* para aplicar las selecciones.

Por último, creamos dos *Button* para generar ficheros PDF, el código de las funciones se explican más adelante.

Figura 33. Pantalla gestión de permisos

7.1.3.3 Pantalla Chat

Esta pantalla es la misma que se explicó en el apartado **7.1.2.3 Pantalla Chat** en el modo Tropa.

7.1.3.4 Pantalla Gestión de Perfiles

Esta pantalla nos permite añadir, eliminar o modificar cualquier militar. Para implementar esta pantalla creamos el fichero *activity_gestion_militares.xml* en el cual creamos un *View* en el que se cargan un *Spinner* para poder seleccionar el tipo de operación que se desea realizar. Añadimos 7 *EditText* que nos permitirán introducir todos los datos del militar necesarios, en el campo teléfono será de tipo numérico, de esta manera solo se podrá introducir números, en el campo de contraseña el tipo de entrada será de password, para que lo que se escriba sea oculto. Por último, agregamos un *Button* para realizar la operación.

Figura 34. Pantalla Gestión de Perfiles

7.1.4 Pantallas Auxiliares

En este apartado veremos las pantallas que sirven para completar alguna funcionalidad y que no se puede insertar en las pantallas principales, ya que no se quería sobrecargar las pantallas principales y de esta forma que estuviesen mejor organizados.

7.1.4.1 Pantalla Notificaciones

En esta pantalla mostramos las notificaciones de solicitud de cambios o cancelaciones de las actividades, para implementar esta pantalla utilizamos el fichero *activity_gestion_notificaciones.xml.*

Figura 35. Notificaciones sin responder

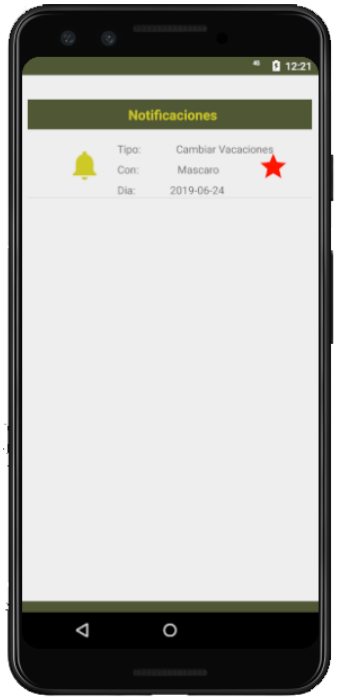

Figura 36 Pantalla Notificaciones

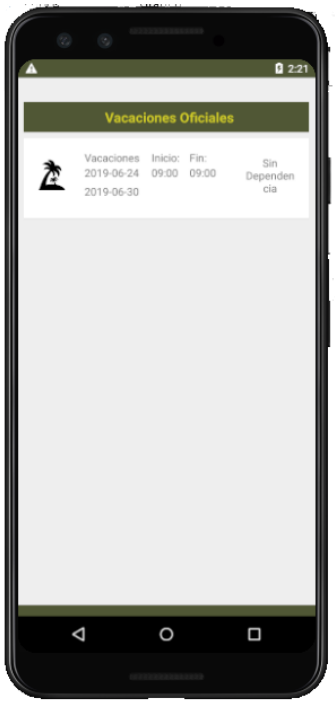

Figura 37. Pantalla de Vacaciones

Como vemos en la figura 35, podemos saber que tenemos notificaciones al ver en la cabecera el icono de sobre con una campana roja.

En el fichero que implementamos para esta pantalla, creamos un *View* y un título con un *TextView* y para mostrar todas las notificaciones creamos un *ListView*. Podemos ver cómo queda la pantalla en la figura 36.

Para poder ver correctamente cada notificación creamos el fichero *fragment_frag_bloque_notificacion.xml* en el cual creamos un *ImageView* de una campana amarilla, luego 6 *TextView* para poder poner la información relativa a la notificación y, por último, un ImageView en forma de estrella que dependiendo del nivel de compañerismo que tenga el militar será de un color u otro (rojo, amarillo, verde).

7.1.4.1 Pantalla Vacaciones

En esta pantalla que podemos ver en la figura 37 se muestran todos los rangos de vacaciones que se le han asignado al militar, creamos el fichero *activity_gestion_permisos.xml* para mostrar la pantalla.

En el fichero utilizamos un *View* para poner dentro el título y con un *TextView* el texto del título. Para mostrar las vacaciones creamos una lista con el *ListView* y para mostrar las vacaciones correctamente utilizamos el fichero *fragment_frag_bloque_actividad_completa.xml* que ya ha sido explicado en el apartado **7.1.2.2 Pantalla Actividades**.

7.1.4.2 Pantalla Cambio Contraseña-Cerrar Sesión

En esta pantalla mostraremos primero que todo, la opción de cambiar la contraseña del usuario y de segundo un botón para cerrar sesión.
También mostraremos en un recuadro información del autor de la aplicación y el motivo para realizarlo.
Para realizarlo implementamos el fichero *activity_gestion_perfil.xml* y lo podemos ver en la figura 38.

Primero creamos componentes *View* para definir las diferentes zonas en la pantalla, para la zona de cambiar contraseña, añadimos *EditText* para la contraseña antigua y la nueva, y un *Button* para solicitar el cambio.

Definimos *Button* para cerrar sesión, el cual borra los datos de usuario y contraseña guardados en el dispositivo.

En la última sección mostramos la información del autor con *TextView* y para poner los escudos de las unidades usamos *ImageView*.

Figura 38. Pantalla cambio de contraseña

7.1.4.3 Pantalla Cambio y Cancelación de Actividad

Esta pantalla que podemos ver en la figura 39 aparecerá cuando se presiona cualquier actividad en la lista de actividades para que se pueda solicitar el cambio o cancelación. Para implementar esta pantalla utilizamos el fichero *fragment_frag_cambios.xml* en el cual utilizamos *View* para definir zonas: solicitar cambios, solicitar cancelación y cantidad de solicitudes enviadas.

En la zona de solicitar cambios utilizamos un *Spinner* para cargar la lista de militares disponibles para realizar el cambio, para los títulos y textos utilizamos *TextView* y un *Button* para iniciar la función de solicitar cambio.

En la zona de solicitar cancelación solo utilizamos *TextView* para mostrar la información y un *Button* para iniciar la función de solicitar la cancelación.

En la última zona utilizamos solo *TextView* para los textos de los títulos y la cantidad de cambios y cancelaciones.

Figura 39. Pantalla de cambios

7.2 Servidor

En este apartado realizaremos la implementación necesaria de la aplicación en el lado del servidor, que consiste en construir la base de datos y crear ficheros php para realizar las funciones descritas en el capítulo de diseño.

7.2.1 Base de Datos

Como hemos descrito en el capítulo de diseño de la aplicación, se crea una base de datos relacional con las tablas que describimos en ese capítulo, teniendo en cuenta las claves primarias y claves foráneas. Se utiliza la herramienta *phpAdmin* que es una herramienta escrita en PHP con idea de administrar MySQL a través de páginas web, utilizando un navegador web [24].

7.2.2 Scripts PHP

En el grupo de ficheros que creamos podemos ver dos tipos de ficheros, el primero de los dos tipos nos sirve para insertar, modificar o eliminar datos de la base de datos, el segundo tipo lo utilizamos para consultar los datos.

Cada funcionalidad de los scripts lo hemos descrito en la parte de diseño, en este apartado mostraremos algunos de ellos.

7.2.2.1 Scripts Conexión

Lo primero que tenemos que realizar es crear funcionalidades para hacer la conexión y poder desconectarnos de la base de datos, para ello creamos los siguientes scripts:

- mysql_pw.php

  Este script nos permite almacenar la información que se puede ver en la figura 40 para hacer la conexión a la base de datos, como nombre del servidor, nombre de la base de datos, usuario y contraseña, guardado en una carpeta externa a la carpeta raíz de la aplicación.

```php
<?php

    $wgDBserver = "PMYSQL107.dns-servicio.com";

    $wgDBname = "6616563_base_militar";

    $wgDBuser = "jerry23";

    $wgDBpassword = [redacted];


?>
```

Figura 40. Datos de conexión a la base de datos

- BaseDatos.php

  Este script nos permite crear funcionalidades de conectar, desconectar y obtener la conexión a la base de datos.
  Para utilizar correctamente este script tenemos que referenciar el script donde están los datos de conexión con la instrucción de la figura 41.

```php
//Realizamos la conexion a la base de datos
require_once($_SERVER["DOCUMENT_ROOT"]."/external_includes/mysql_pw.php");
```

Figura 41. Instrucción para incluir fichero mysql_pw.php

7.2.2.2 Scripts de Gestión de la Seguridad

- Seguridad.php

  En este script nos permite cifrar la contraseña y el nick de los usuarios al darse de alta, por ejemplo. Para poder utilizar la base de datos, tenemos que incluir el script "BaseDatos.php". En la figura 42 podemos ver el contenido del script.

```php
<?php
require_once "BaseDatos.php";
$GLOBALS['salt'] = 'Salt20Jun19';

class Seguridad{

    public static function hashPass($pass){
        return hash('gost', $pass.$GLOBALS['salt']);
    }
    public static function hashNick($nick){
        return sha1($nick.$GLOBALS['salt']);
    }

}
?>
```

Figura 42. Script de funciones de encriptado

7.2.2.3 Scripts de Gestión de Sesión

- Sesion.php

  Este script nos permite comprobar si el usuario ha introducido un nick y un password correcto, para ello primero incluimos los ficheros necesarios con las instrucciones de la figura 43.

```php
require_once "BaseDatos.php";
require_once "Seguridad.php";
```

Figura 43. Ficheros incluidos en el script sesión.php

  Luego comprobamos que hay valores pasados por parámetros, para poder comprobarlos utilizamos la siguiente instrucción:

```php
if(isset($_GET["nick"]) && isset($_GET["pass"])){
```

  Si hay parámetros, nos conectamos a la base de datos, utilizamos el script de seguridad para cifrar el nick y la contraseña para comprobar en la base de datos ya que en ella estos datos están cifrados.

  Si existe el militar con esos datos, el script nos devolverá un JSON con los datos necesarios para la aplicación, se muestran en la figura 44.

```php
if($reg=mysqli_fetch_array($resultado)){
    $temp= array();
    $temp['idMilitar']=$reg['id_militar'];
    $temp['apellidol']=$reg['apellidol'];
    $temp['nivel']=$reg['nivel'];
    array_push($json,$temp);
}
```

Figura 44. Construcción de JSON de retorno

7.2.2.4 Scripts de Gestión de Usuarios

- modificarDatosMilitar.php

  En este script podemos añadir, eliminar y modificar los datos de los militares en la base de datos. También en este script definimos la función para cambiar la contraseña.
  Creamos funciones auxiliares que ayudan a la realización de las funciones principales:
  - correctaContrasena (…) nos indica si la contraseña antigua es correcta
  - existeMilitar (…) nos comprueba si el militar está en el sistema
  - getIdMilitar (…) nos devuelve el id del militar
  - getIdByNick (…) obtenemos el id del militar por su nick

  Podemos ver en la figura 45 un ejemplo de una función del script.

```php
public function anadirMilitar($nombre,$apellido1,$apellido2,$telefono,$rango,$nick,$pass,$nivel){
    //Realizamos la conexion a la base de datos
    $bd= new BaseDatos();
    $bd->conectar();
    $con=$bd->getConexion();
    //obtenemos el indice del rango militar
    $rango = strtolower($rango);
    $cadenaRango = "select id_rango from rango where nombre='$rango'";
    $resultadoRango=mysqli_query($con,$cadenaRango);
    $retRango = mysqli_fetch_array($resultadoRango);
    $idRango = $retRango['id_rango'];
    //ciframos nick
    $seguridad = new Seguridad();
    $nickCifrado=$seguridad->hashNick($nick);
    //obtenemos el indice del militar
    $idMilitar=self::getIdMilitar($con,$nombre,$apellido1,$apellido2);
    //comprobamos si existe el registro
    if (self::existeMilitar($con,$idMilitar,$nickCifrado)==TRUE){
        echo "Ya existe Militar con ese Nick";
    } else {

        //ciframos la contraseña
        $passCifrada=$seguridad->hashPass($pass);
        //creamos la sentencia a la base de datos
        $cadena = "insert into militar (nombre,apellido1,apellido2,telefono,rango,nick,pass,nivel)
        values ('$nombre','$apellido1','$apellido2','$telefono','$idRango','$nickCifrado','$passCifrada','$nivel');";

        if($con->query($cadena)==TRUE){
            echo "Nuevo Militar añadido Correctamente";
        }else{
            echo "Error: $cadena <br> $con->error";
        }
    }
```

Figura 45. Código para añadir nuevo militar

- modificarInfoPersonal.php

  Con este script podemos añadir, eliminar y modificar la información personal de los militares relativos a la unidad.
  Creamos la función auxiliar existeInfoMilitar(…) que comprueba si existe el registro de la información del personal en la base de datos.

7.2.2.5 Scripts de Gestión de Actividades

- modificarCambiosServicios.php

  En este script controlamos poder añadir, eliminar y modificar todos los bloques de cambios de servicios para cada militar.
  Implementamos funcionales auxiliares como:
  - existeCambioById (…) nos dice si existe un registro por el idMilitar
  - existeCambio (…) nos dice si existe un registro por el id

- getIdCambio (…) nos da el id del registro

En la figura 46 vemos un ejemplo de una función del script.

```php
public function eliminarCambio($idCambio){
    //Realizamos la conexion a la base de datos
    $bd= new BaseDatos();
    $bd->conectar();
    $con=$bd->getConexion();
    //comprobamos si existe el registro
    if (self::existeCambio($con,$idCambio)==TRUE){
        //creamos la sentencia a la base de datos
        $cadena = "delete from cambio_servicio where id_cambio='$idCambio';";

        if($con->query($cadena)==TRUE){
            echo "cambio borrado correctamente";
        }else{
            echo "Error: $cadena <br> $con->error";
        }
    } else {
        print("No existe cambio");
    }
    //desconectamos de la base de datos
    $bd->desconectar();
}
```

Figura 46. Código para eliminar cambio

- modificarServiciosMilitar.php

  En este script podemos añadir, eliminar y modificar todas las actividades o servicios que se han asignado a los militares. Implementamos funciones auxiliares:
  - existeAsociacion (…) comprueba si existe registro de la asociación
  - existeAsociacionDia (…) comprueba si existe registro por fecha
  - getIdServicio (…) obtenemos el id del servicio
  - calcularFechaFin (…) calculamos la fecha final de la actividad

7.2.2.6 Scripts de Gestión de Notificaciones

- modificarNotificaciones.php

  Este script nos permite añadir, eliminar y modificar las notificaciones creadas para los militares, las notificaciones son solicitudes de cambios o cancelaciones. Creamos las siguientes funciones auxiliares:
  - eliminarNotificacionById (…) eliminamos notificación por idMilitar
  - existeNotificacionById (…) comprueba si existe registro por idMilitar
  - existeNotificacion (…) comprueba si existe registro
  - getIdNotificacion (…) obtenemos el id del registro

  En la figura 47 podemos ver un ejemplo de función en el script.

```php
public function existeNotificacionByID($con,$idMilitar){
    $consulta="select * from notificacion where id_militar_sustituto='$idMilitar';";
    $resultado=mysqli_query($con,$consulta);
    $row_cnt=mysqli_num_rows($resultado);
    if($row_cnt>0){
        return TRUE; //existe registro
    }else{
        return FALSE; //No existe registro
    }
}
```

Figura 47. Código para comprobar registro por idMilitar

7.2.2.7 Scripts Gestión Auxiliares

- modificarRangoVacaciones.php

  En este script podemos añadir, eliminar y modificar los rangos de vacaciones que crea el jefe de la unidad, Creamos las funciones auxiliares:
  - existeRango (…) comprueba si existe el registro del rango
  - getIdRango (…) obtenemos el id del registro

```php
public function getIdRango($con,$fechaIni,$fechaFin){
    $consulta="select id_rango from rango_vacaciones where fecha_ini='$fechaIni' and fecha_fin='$fechaFin';";
    $resultado=mysqli_query($con,$consulta);
    $ret = mysqli_fetch_array($resultado);
    $segm = $ret['id_rango'];
    $row_cnt=mysqli_num_rows($resultado);
    if($row_cnt>0){
        return $segm;
    }else{
        return -1; //No existe id
    }
}
```

Figura 48. Código que devuelve id de registro de rango

- modificarToken.php

  En este script nos permite añadir, eliminar y modificar los tokens de los dispositivos de los militares. Las funcionalidades auxiliares son:
  - eliminarTokenByIdMilitar (…) eliminamos el token por idMilitar
  - existeIdMilitar (…) comprueba si existe registro por idMilitar
  - existeToken (…) comprueba si existe registro
  - getIdToken (…) nos da el id del registro

7.2.2.8 Scripts Consultas

- webServiceBaseMilitar.php

  En este script creamos todas las funcionalidades que nos da la información que necesitamos para luego mostrarla en nuestra aplicación.
  Algunas funciones que implementamos en este script:
  - getTodasAsociaciones (…) nos da todas las actividades a realizar por el militar
  - getTodasVacaciones (…) obtenemos las vacaciones asignadas por militar
  - getAsociacionFecha (…) nos da todas las actividades por fecha

  En la figura 49 vemos un ejemplo de función del script.

```php
public function getInfoPersonal($idMilitar){

    //Realizamos la conexion a la base de datos
    $bd= new BaseDatos();
    $bd->conectar();
    $con=$bd->getConexion();

    //Creamos la consulta a la base de datos
    $cadena="select * from info_personal where id_militar='".$idMilitar."';";
    $tabla=mysqli_query($con,$cadena);
    $registro=array();

    while($reg=mysqli_fetch_array($tabla)){
        $temp= array();
        $temp['vacaciones']=$reg['vacaciones'];
        $temp['compensacion']=$reg['compensacion'];
        $temp['companerismo']=$reg['companerismo'];
        array_push($registro,$temp);
    }

    //desconectamos de la base de datos
    $bd->desconectar();

    //devolvemos el json
    return $registro;
}
```

Figura 49. Código para obtener la información personal del militar

- webServiceDatosMilitar.php

  En este script creamos funciones que nos permitan obtener datos específicos que se necesitan en la aplicación, por ejemplo, obtener el id de alguna aplicación, los datos para generar los PDF como el cuadrante de guardias, también se han añadido funciones para eliminar actividades y notificaciones ya pasadas de la base de datos de cada militar.
  Algunas funciones implementadas:
  - getIdActividad (…) devuelve el id de una actividad
  - obtenerNumerCambioCancelacionMilitar (…) nos da la cantidad de solicitudes de cancelaciones y cambios
  - getDatosGenerarCuadrante (…) nos da los datos, para crear los cuadrantes

8. Pruebas

Para comprobar que nuestro sistema realiza correctamente las funcionalidades que se han implementado para lograr los objetivos y requisitos que se querían de nuestro sistema, vamos a realizar una batería de pruebas. En primer lugar, la fase de pruebas de la parte cliente (es decir la aplicación Android) y, por otro lado, la parte servidor.

8.1 Pruebas en el cliente

Para realizar las pruebas en el cliente, se realizaban cada vez que se terminaba un prototipo para intentar determinar posibles errores. Se utiliza tanto el tipo de pruebas de caja blanca (que se centran en los detalles procedimentales del software, por lo que su diseño está fuertemente ligado al código fuente [25]) y el tipo de pruebas de caja negra (que se estudia desde el punto de vista de las entradas que recibe y las salidas o respuestas que produce, sin tener en cuenta su funcionamiento interno [26]).

Primero, se han realizado pruebas de caja negra, analizando funciones que podían dar fallos debido a su complejidad, o por que realizaban conexiones al servidor. Se han comprobado uno a uno para solucionar los problemas que surgían, para evitar que volvieran a surgir en el futuro.

Finalmente, hemos realizado pruebas de caja blanca, analizando el código y verificando las partes que podrían ser más problemáticos, que solían coincidir con los errores que nos marcaban con las pruebas de caja negra.

Ahora detallamos las diferentes pruebas que hemos realizado en cada uno de los prototipos.

- Prototipo jefe de unidad: Se prueba el inicio de sesión, asignar/desasignar actividades a militares, gestión de usuarios, gestión de permisos y generación de PDF.
- Prototipo tropa de unidad: Se prueba mostrar actividades, solicitar cambios o cancelaciones.
- Prototipo final: Se prueba gestión de notificaciones, visualización de vacaciones, cambio de contraseña, cerrar sesión, funcionamiento de chat, recepción de notificaciones en segundo plano.

Creamos las siguientes tablas para poder ver la lista de pruebas realizadas y en que prototipo se ha realizado, la gran parte de las pruebas que se realizaron fueron operaciones que en teoría no se tendrían que poder realizar para comprobar como el sistema responde a esos fallos.

8.1.1 Pruebas Prototipo jefe

En la tabla 8 podemos ver todas las pruebas que hemos realizado al prototipo del jefe de la unidad.

Entrada	Salida esperada
nick o pass incorrectos	No permite entrada a la aplicación
Asignar actividades con algún campo en blanco	No asigna la actividad
Cargar actividades con un swipe	Rellenar spinner con actividades
Desasignar actividades	Elimina asignación en la base de datos
Registrar nuevo militar	Crea nuevo registro en la base de datos
Registrar militar más de una vez	No permite el registro
Eliminar registro de militar	Elimina registro de la base de datos
Eliminar registro militar que no existe	No permite eliminar registro
Crear o eliminar rango de vacaciones con alguna fecha en blanco	No permite crear o eliminar el rango
Cargar los rangos de vacaciones con un swipe	El spinner se llena con los rangos de vacaciones
Asignar rango de vacaciones con algún campo vacío	No permite asignar rango de vacaciones
crear el fichero PDF de permisos	Crea el fichero permisos y lo guarda en descargas
Crear el fichero PDF cuadrante de guardias	Crea el fichero cuadrante y lo guarda en descargas

Tabla 8. Pruebas a prototipo jefe de unidad

Se ha detectado en casos muy aislados que las operaciones con swipe tenemos que realizarla dos veces para que carguen los datos y se refresque la página correctamente.

También en casos muy raros al trabajar con los calendarios de Android, aunque aparezca en los *EditText* las fechas, al presionar el botón de aplicar no detecta correctamente las fechas dentro de los *EditText*, a no ser que cambie de página y se vuelva a ella.

8.1.2 Pruebas Prototipo Tropa

En la tabla 9 podemos ver las pruebas que realizamos al prototipo de tropa

Entrada	Salida esperada
Mostrar actividades al seleccionar un mes y tipo	Lista de actividades del mes seleccionado y tipo
Solicitar cambio de actividad	Se envía notificación de solicitud de cambio
Solicitar cancelación	Se envía notificación de solicitud de cancelación

Tabla 9. Pruebas a prototipo tropa

Se ha detectado en algunos casos al intentar cambiar una actividad, al cargar la pantalla de solicitud de cambio, no le da tiempo a cargar los datos de militares disponibles para realizar el cambio y el *Spinner* aparece vacío, pero si se vuelve a atrás se carga correctamente.

8.1.3 Pruebas Prototipo final

En la tabla 10 podemos ver las pruebas que realizamos al prototipo final

Entrada	Salida esperada
Se crea notificación a militar	Se puede ver icono de campana roja
Presionar icono de mensajes	Se muestra pantalla de notificaciones
Presionar notificación	Se muestra mensaje de aceptación de solicitud
Aceptar solicitud de cambio o cancelación	Se cambia o elimina la actividad en la base de datos
Denegar solicitud de cambio o cancelación	Se elimina notificación
Presionar icono calendario	Se muestra pantalla de vacaciones
Presionar icono de contacto	Se muestra pantalla cambio de contraseña/ cerrar sesión
Cambiar contraseña con contraseña antigua incorrecta	No se modifica la base de datos
Presionar botón de cerrar sesión	Se borran los datos de usuario y contraseña del dispositivo
Enviar mensaje al chat	Se muestra el mensaje en el chat
Enviar solicitud de cambio/cancelación	Se muestra notificación en dispositivo en segundo plano

Tabla 10. Pruebas al prototipo final

Se ha podido ver que en algunas ocasiones el icono de notificaciones no se muestra correctamente. También muy poco frecuente que al aceptar la notificación de cambio se elimina de la pantalla, pero al volver a atrás se vuelve a cargar la notificación.

8.2 Pruebas en el servidor

Como describimos en el capítulo 3-tecnologías utilizadas, para realizar las pruebas en el servidor, del funcionamiento de nuestra API, utilizaremos el programa POSTMAN para comprobar que pasados los parámetros adecuados nos retorna lo que esperamos.

Para poder realizar las pruebas lo más organizadamente posible, las agrupamos en categorías, basándonos en sus características y su finalidad.

A continuación, mostraremos las pruebas más importantes de las que hemos realizado:

8.2.1 Pruebas Inicio Sesión

Para realizar esta prueba usaremos la dirección url:
https://38966471.servicio-online.net/TaskUS/sesion.php?

para utilizar el script de sesión, y los parámetros que requiere este script son nick y pass, que para nuestra prueba utilizaremos el nick = “fox” y el pass = “1234” que son los datos del jefe de la unidad. Podemos ver en la figura 50 como tendremos que poner los datos utilizando el programa POSTMAN.

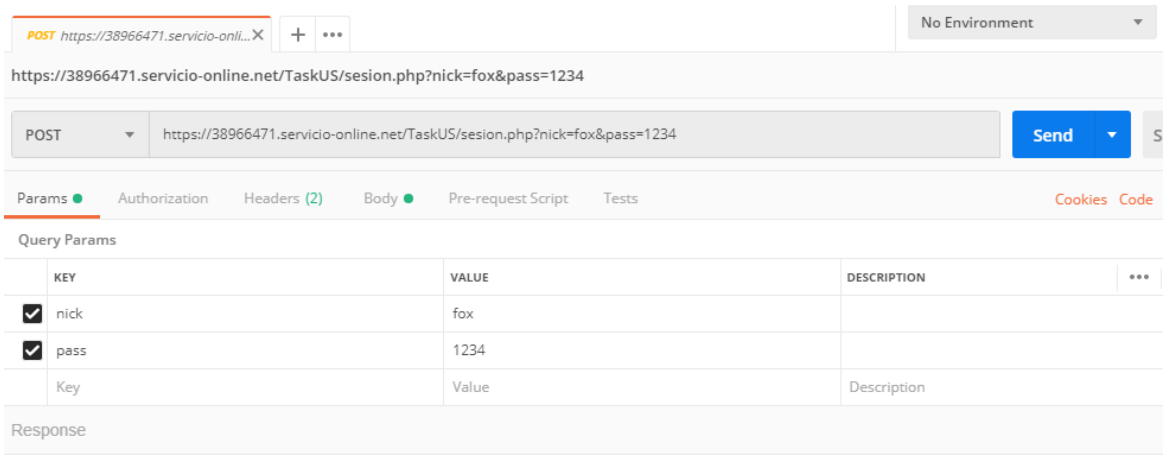

Figura 50. Petición de inicio de sesión

Al enviar la solicitud esperamos como respuesta un JSON en el cual contiene el id del militar, su apellido1 y su nivel de permiso en la aplicación.

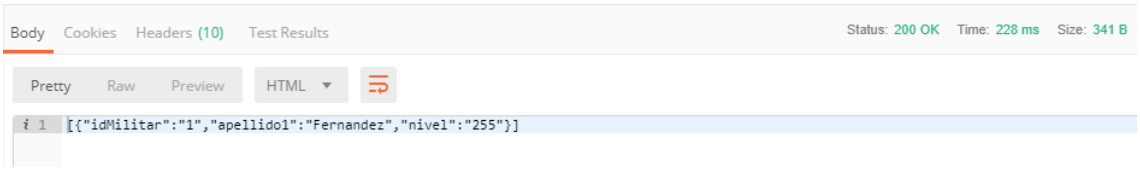

Figura 51. Respuesta de inicio de sesión

En la figura 51 podemos ver la respuesta de la petición y podemos comprobar que es lo que esperábamos.

8.2.2 Pruebas Gestión de Usuarios

En esta categoría vamos a mostrar la prueba a la funcionalidad de añadir un militar nuevo al servidor.
Para hacer la prueba utilizaremos la url:
https://38966471.servicio-online.net/TaskUS/modificarDatosMilitar.php?

Para utilizar el script de gestión de usuarios y tenemos que pasar por parámetro:

- Tipo de operación: definimos que es una modificación con el parámetro modificar y su valor “am” que indica añadir militar.
- Datos personales: aquí pasamos los datos del militar, nombre, apellido1, apellido2, teléfono, rango, nick, pass y por último el nivel de permiso.

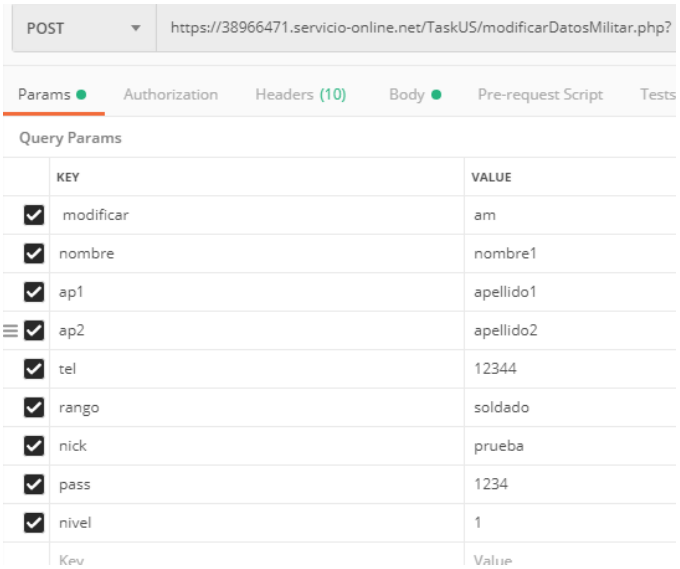

Figura 52. Petición de añadir militar

En la figura 52 podemos ver como introducimos los parámetros en el programa para realizar la prueba. Esperamos como respuesta el mensaje del servidor en el cual se informa que se ha añadido el militar correctamente, o no se ha añadido por que ya existe o fallos en algún dato.

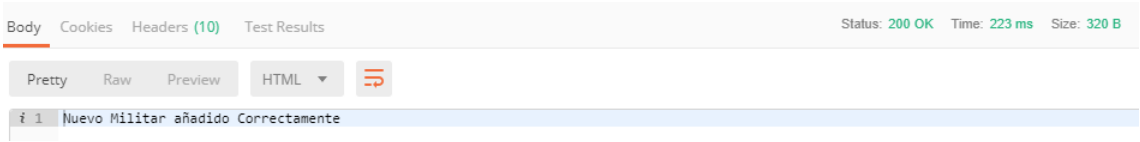

Figura 53. Respuesta al añadir un nuevo militar

Podemos ver en la figura 53 que nuestro militar ha sido añadido correctamente.

8.2.3 Pruebas Gestión de Actividades

Como ejemplo de esta categoría vamos a realizar la prueba a la función eliminar cambio, que son registros que almacenan la cantidad de solicitudes de cambios y cancelaciones de servicios de cada militar.

Para realizar la prueba utilizaremos la url:
https://38966471.servicio-online.net/TaskUS/modificarCambiosServicios.php?

Que utiliza el script de gestión de cambios de servicios, como parámetros debemos definir el tipo de operación (que en este caso es modificar y su valor es “ec” que significa eliminar cambio) y el id del registro a eliminar.

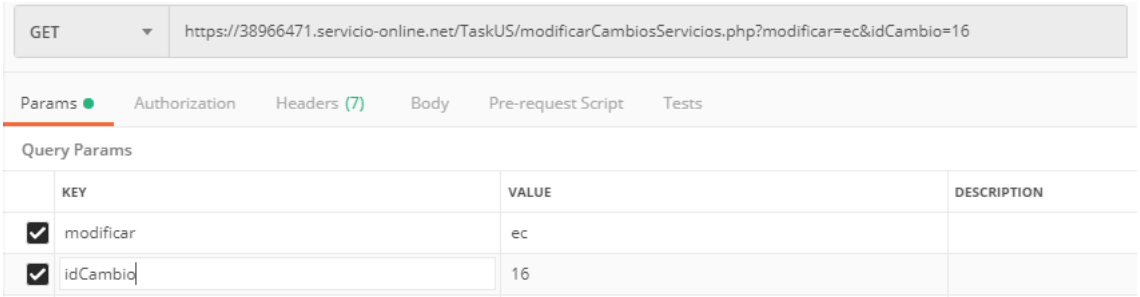

Figura 54. Petición para eliminar registro de cambios

Podemos ver en la figura 54 los parámetros que necesita la petición para realizar la funcionalidad correctamente.

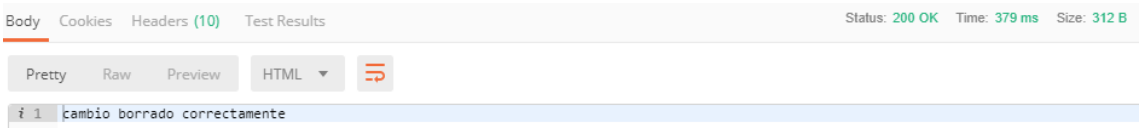

Figura 55. Respuesta a eliminar registro de cambios

Podemos ver en la figura 55 que se ha eliminado correctamente el registro de cambios con el id pasado por parámetro.

8.2.4 Pruebas Gestión de Notificaciones

Como ejemplo de esta categoría vamos a realizar la prueba a la función eliminar Notificación, por idMilitar que son registros que almacenan las notificaciones de solicitudes de cambios y cancelaciones de servicios de cada militar.

Para realizar la prueba utilizaremos la url:
https://38966471.servicio-online.net/TaskUS/modificarNotificaciones.php?

Que utiliza el script de gestión de notificaciones de cambios de servicios, como parámetros debemos definir el tipo de operación (que en este caso es modificar y su valor es "enxid" que significa eliminar notificación por id) y el id del militar a eliminar.

Figura 56. Petición para eliminar notificación por idMilitar

En la figura 56 vemos que parámetros pasamos a la petición.

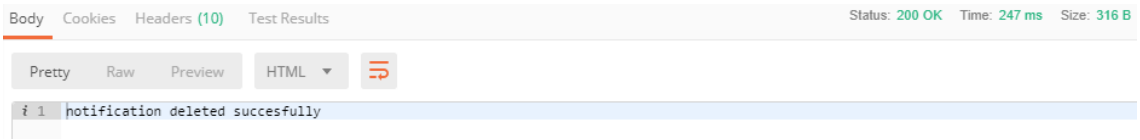

Figura 57. Respuesta al eliminar notificación por idMilitar

Podemos ver el mensaje de respuesta en el que nos informa que se ha eliminado correctamente la notificación.

8.2.5 Pruebas Gestión Auxiliares

Como ejemplo de esta categoría vamos a realizar la prueba a la función añadir rango, que son registros que almacenan la fecha de inicio y la fecha fin en la que los militares pueden disfrutar de sus permisos oficiales.

Para realizar la prueba utilizaremos la url:
https://38966471.servicio-online.net/TaskUS/modificarRangoVacaciones.php?
Ya que vamos a utilizar el script para modificar el rango de vacaciones, pasaremos como parámetros el tipo de operación (que en este caso es modificar y su valor es "ar" que significa añadir rango) y las fechas (fecha inicio y fecha fin).

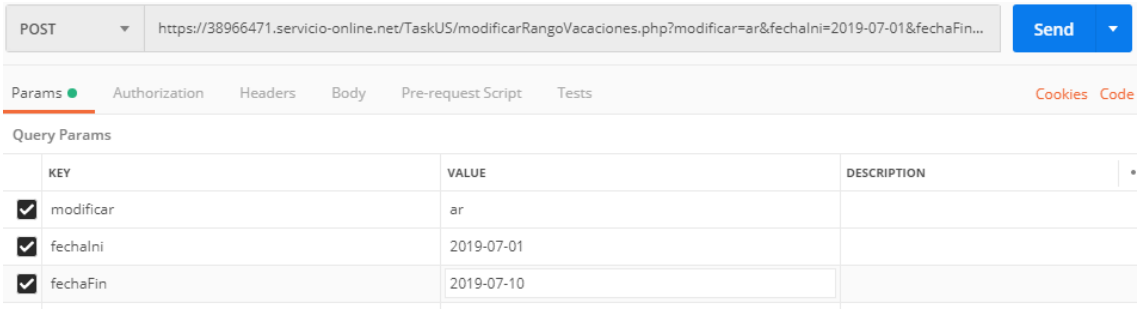

Figura 58. Petición para añadir un nuevo rango de vacaciones

En la figura 58 vemos los parámetros que introducimos para realizar la petición.

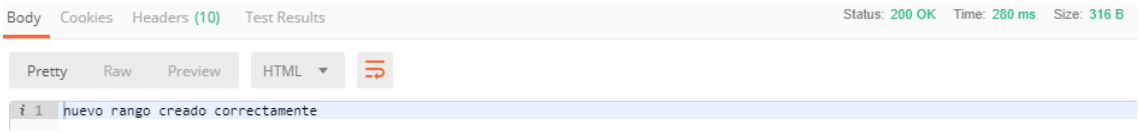

Figura 59. Respuesta de petición añadir rango de vacaciones

En la figura 59 podemos ver que se ha creado un nuevo rango correctamente.

8.2.6 Pruebas Consultas a la Base de Datos

Como ejemplo de esta categoría vamos a realizar la prueba a la función getInfoPersonal ( ), que nos da la información personal, relativa a la unidad.

Para realizar la prueba utilizaremos la url:
https://38966471.servicio-online.net/TaskUS/webServiceBaseMilitar.php?

Ya que vamos a utilizar el script webServiceBaseMilitar.php para obtener la información del militar pasado primero que todo el tipo de operación, que en este caso será una consulta y su valor será “personal” y luego el id del militar.

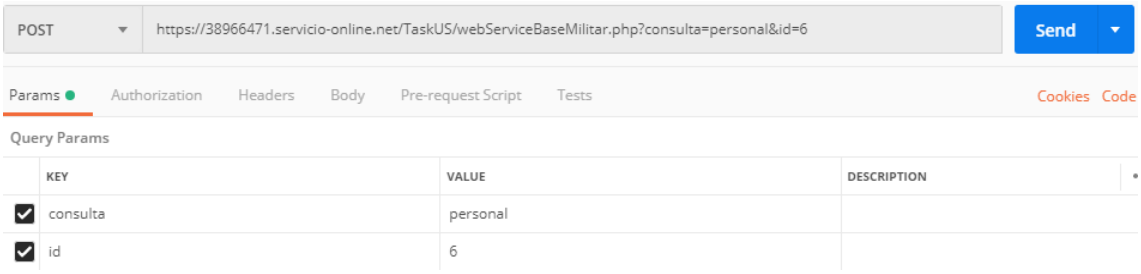

Figura 60. Petición de información del militar

En la figura 60 podemos ver los parámetros que introducimos en el programa para poder realizar la petición.

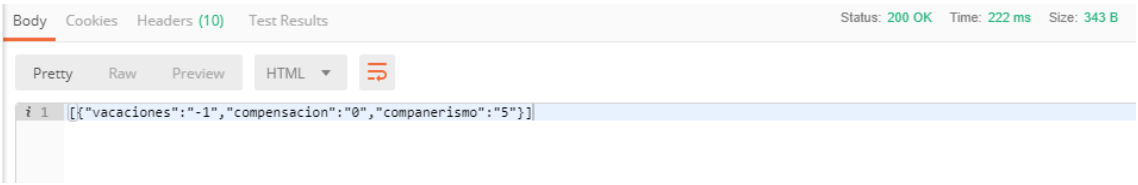

Figura 61. Respuesta de la petición de información del militar

En la figura 61 podemos ver que nos llega un JSON con la información del militar, en este caso vemos que no tiene vacaciones ni días de compensación y que su nivel de compañerismo está en nivel 5.

Otro ejemplo de prueba en esta categoría es realizarla a la función getDatosGenerarCuadrante( ) la cual nos tiene que devolver de todos los militares en el sistema, todas las actividades que sea de tipo “Guardia” o “Vacaciones” para poder crear un fichero PDF con esos datos.

Para realizar la prueba utilizaremos la url:
https://38966471.servicio-online.net/TaskUS/webServiceDatosMilitar.php?

Ya que en este caso utilizaremos el script webServiceDatosMilitar.php para obtener los datos. Tendremos que pasar por parámetro el tipo de operación que en este caso será una consulta y su valor será “cuadrante”, podemos ver cómo queda en la figura 62.

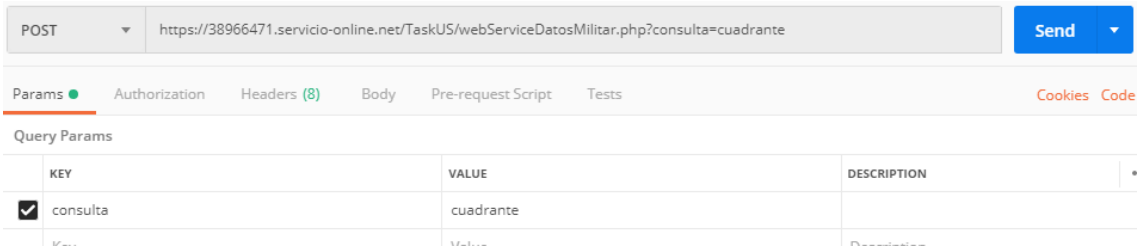

Figura 62. Petición de datos del cuadrante

Figura 63. Respuesta de la petición de datos de cuadrante

Como podemos ver en la figura 63 nos llega en formato JSON todos los militares registrados y todas las actividades de tipo “Guardia” y “Vacaciones”.

9. Conclusiones

Todos los objetivos y requisitos que planteamos en los capítulos iniciales se han cumplido. El principal objetivo era construir una aplicación móvil para el sistema operativo Android, la cual permitía que todos los componentes de la unidad de seguridad del acuartelamiento JAIME II pudieran interactuar de una manera más eficiente con lo relacionado a la asignación de las actividades a desarrollar dentro de la unidad.

La aplicación que hemos desarrollado permite para la tropa, poder consultar en cualquier momento las actividades que se le han sido asignadas, como también el poder solicitar cambios a sus compañeros por razones que le impidan realizar esa actividad. También por otro lado le permite solicitar directamente la cancelación de la actividad al jefe de la unidad, el cual tendrá su criterio para aprobarlo o no. Además, como función auxiliar de la aplicación, permite la comunicación de novedades de la unidad dentro de la misma aplicación, así, cualquier miembro de la unidad será capaz de conocer dicha información.

En cuanto a la parte del jefe de la unidad, la aplicación permite asignar las diversas actividades de forma rápida, segura y fiable a todos los integrantes de tropa, también le simplifica la opción de crear rangos de vacaciones y asignar dichos rangos al personal, por otro lado, el jefe de la unidad ahorra tiempo al crear los cuadrantes de guardia, ya que antes se realizaban de forma manual y con la aplicación se genera de manera automática al presionar tan solo un botón.

A título personal, he aprendido a desarrollar proyectos de principio a fin, pasando por todas las fases de desarrollo de software, como también a desarrollar aplicaciones para dispositivos Android, ampliando conocimientos sobre lenguajes como PHP y Java.

La realización del proyecto me ha permitido mejorar en mi planificación, la administración de mis recursos y en especial mejorar la manera de investigar sobre temas en concreto.

10. Trabajo Futuro

Esta aplicación tiene muchas mejoras que pueden convertirla en una gran aplicación.

La primera de ellas es mejorar la interfaz gráfica, permitiendo que sea más intuitiva y sencilla de utilizar, pudiendo aplicar técnicas para hacerla más accesible a todos los componentes de la unidad de seguridad.

También agregar más funcionalidades importantes que permitan colmar todas las necesidades de un miembro de la unidad de seguridad evitando así protocolos innecesarios.
Se podría también pensar en generalizar la aplicación para que pudiese servir a cualquier unidad de seguridad en todo el territorio español de esa forma convertirla en una aplicación oficial del ejército de tierra, pasando por todos los trámites necesarios para conseguir tal fin.

Otro de los aspectos importantes a mejorar es la seguridad, ya que en este proyecto solo hemos tratado el cifrado de los nick, contraseñas y mensajes enviados por la red. Al ser una aplicación con información de carácter militar, se debe tratar con especial cuidado este tema.

La implementación de la aplicación también podría ser mejorada de tal forma que se optimizara las funcionalidades y el uso de los recursos en los diferentes dispositivos que utilizan la aplicación.

Por otro lado, se podría crear la aplicación para otros sistemas operativos como el de iOS para el personal que utiliza iPhone y sacar la versión web para que los jefes de unidad tengan más funcionalidades y sea más cómodo al utilizarlo.

También debemos tener en cuenta que los sistemas operativos móviles como Android siempre se están actualizando, por lo que se tendría que realizar tareas de mantenimiento para analizar las nuevas versiones y adaptar nuestro proyecto, de esta manera evitaremos posibles comportamientos erróneos para versiones más modernas de Android.

Anexo A

MANUAL DE USUARIO

Usamos para la demostración de este manual de usuario el emulador de Android studio, con las siguientes características:

- Pixel 3 Api 22
- 1080 x 2160: 440 dpi
- Android 5.1
- 5,5 GB

Al iniciar la aplicación por primera vez, se muestra la pantalla de inicio de sesión.

Figura 64. Pantalla inicio de sesión

1- Campo para introducir el nick del usuario
2- Campo para introducir la contraseña del usuario
3- Icono para mostrar información de como registrarse en el sistema
4- Botón para iniciar sesión en la aplicación

El único usuario que está registrado al inicio de la aplicación es el usuario por defecto para el jefe de la unidad con los siguientes datos:

- Nick de usuario: fox
- Contraseña: 1234

ACCESO JEFE DE UNIDAD

1- Icono para ver las notificaciones de solicitudes de cancelaciones
2- Icono para ver las vacaciones oficiales asignadas
3- Icono para cambiar contraseña o cerrar sesión
4- Campo para introducir la fecha de inicio de actividad
5- Campo para introducir tipo de actividad
6- Campo para introducir la hora de inicio de la actividad
7- Botón para seleccionar los militares para asignar la actividad
8- Botón para crear la asociación.
9- Campo para elegir la actividad a eliminar
10- Botón para eliminar la actividad seleccionada
11- Botón para crear, eliminar o modificar perfil del militar

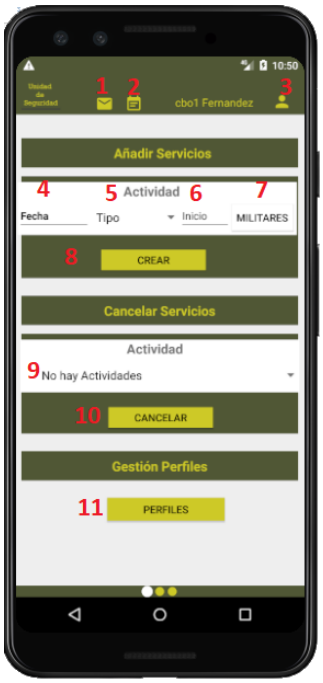

Figura 65. Pantalla Gestión Servicios

1- Campo para introducir la fecha de inicio de vacaciones
2- Campo para introducir la fecha de finalización de vacaciones
3- Botón para crear rango de vacaciones
4- Botón para eliminar rango de vacaciones
5- Selección de tipo de operación
6- Selector de un rango de vacaciones
7- Botón para seleccionar militares para asignar vacaciones
8- Selector de número de días libres
9- Botón para asignar o eliminar vacaciones de al militar
10- Botón para generar documento PDF de los permisos asignados
11- Botón para generar documento PDF de las guardias asignadas

Figura 66. Pantalla Gestión de Permisos

1- Selector de tipo de operación
2- Campo para introducir el nombre
3- Campo para introducir el primer apellido
4- Campo para introducir el segundo apellido
5- Campo para introducir el número de teléfono
6- Selector de rango militar
7- Campo para el nick del militar
8- Campo para la contraseña del militar
9- Botón para realizar la operación seleccionada

Figura 67. Pantalla Gestión de Perfiles

1- Zona de mensajes escritos en el chat
2- Zona para escribir los mensajes
3- Botón para publicar los mensajes en el chat

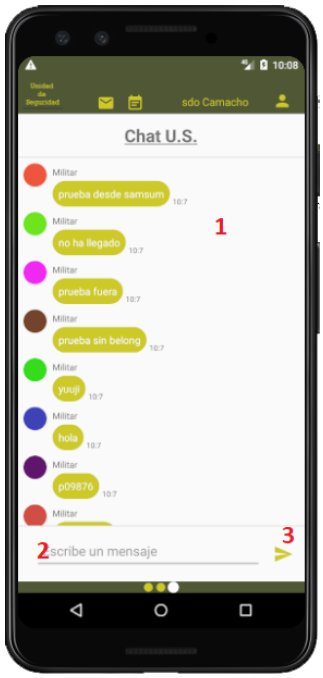

Figura 68. Pantalla del Chat

1- Campo para introducir la contraseña anterior
2- Campo para introducir la contraseña nueva
3- Botón para cambiar la contraseña
4- Botón para cerrar sesión

Figura 69. Pantalla Contraseña-sesión

ACCESO TROPA DE LA UNIDAD

Para el acceso de esta demostración, hemos utilizado los siguientes valores para la pantalla de inicio de sesión:

- Nick de usuario: camacho1
- Contraseña: 1234

Esta es la página inicial de la tropa, podemos ver que muestra 4 zonas, la primera nos informa del tiempo que hace en el acuartelamiento JAIME II, la fecha actual y la temperatura.

En la segunda zona, nos muestra la lista con las actividades que tiene que realizar el militar, para el día actual.

En la tercera zona muestra la información sobre cuantas semanas de vacaciones le queda, cuantos días de compensación tiene y una métrica que define cuanto de compañero es, este valor ira aumentando con cada solicitud de cambio de guardia realice a un compañero en el mes actual, luego cuando se termine el mes, el contador se volverá a reiniciar.

Figura 70. Pantalla inicio de tropa

En esta pantalla podemos ver todas las actividades que tengamos asignadas. En la zona 1 se mostrará la lista de las actividades y en la zona 2, tenemos un selector en el cual podemos elegir el mes que queremos ver y podemos filtrar las
actividades por tipo de actividad.

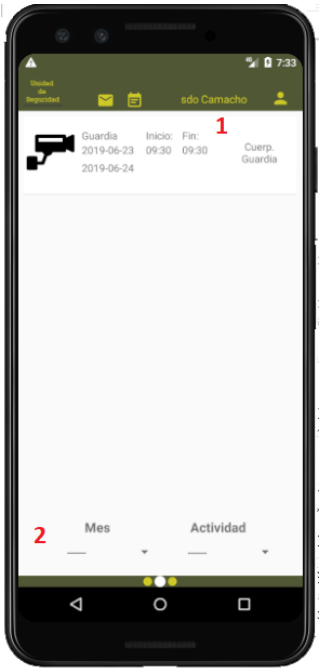

Figura 71. Pantalla de lista de Actividades

Esta es la pantalla para solicitar un cambio de servicio a un compañero de la unidad, aparecerá cuando se presiona encima de la actividad a cambiar.

Tenemos un desplegable (1) que permite seleccionar el militar al que le queremos solicitar el cambio.
Luego presionamos el botón (2) para enviar la solicitud.

Para enviar una solicitud de cancelación presionamos el botón solicitar (3) y se enviara una solicitud al jefe de la unidad.

Al final de la pantalla tenemos dos contadores que nos indican el número de solicitudes que se han enviado en el mes actual.

Figura 72. Pantalla de solicitud de cambios

Bibliografía

[1] MySQL Workbench. (s.f.). En *Wikipedia*. Recuperado el 1 de junio de 2019 de https://es.wikipedia.org/wiki/MySQL

[2] PHP. (s.f.). En *Wikipedia*. Recuperado el 1 de junio de 2019 de https://es.wikipedia.org/wiki/PHP

[3] JSON. (s.f.). En *Wikipedia*. Recuperado el 1 de junio de 2019 de https://es.wikipedia.org/wiki/JSON

[4] statista. (s.f.). "*Cuota de mercado de sistemas operativos para smartphones por pedidos 2014-2020"*. Recuperado de https://es.statista.com/estadisticas/600731/cuota-de-mercado-de-sistemas-operativos-para-smartphones-por-pedidos--2020/

[5] García, D. (2019, 8 mayo). "*6 meses después, vuelven los datos de distribución de Android: la fragmentación sigue siendo un problema"*. Recuperado 1 junio, 2019, de https://andro4all.com/2019/05/distribucion-android-mayo-2019

[6] Android Studio. (s.f.). En *Wikipedia*. Recuperado 1 de junio, 2019 de https://es.wikipedia.org/wiki/Android_Studio

[7] Kotlin (lenguaje de programación). En *Wikipedia*. Recuperado el 1 de junio de 2019 de https://es.wikipedia.org/wiki/Kotlin_(lenguaje_de_programaci%C3%B3n)

[8] Pérez Porto, J., & Gardey, A. (2013). "*Definición de XML*". Recuperado 2 junio, 2019, de https://definicion.de/xml/

[9] CentOS. (s.f.). En *Wikipedia*. Recuperado el 2 de junio de 2019 de https://es.wikipedia.org/wiki/CentOS

[10] Venemedia comunicaciones C.A. (2016, 10 mayo). "*Apache - Qué es y Definición 2019"*. Recuperado 2 junio, 2019, de https://conceptodefinicion.de/apache/

[11] Rouse, M. (2015, enero). *"¿Qué es MySQL?"*. Recuperado 3 junio, 2019, de https://searchdatacenter.techtarget.com/es/definicion/MySQL

[12] MySQL Workbench. (s.f.). En *Wikipedia*. Recuperado el 3 de junio de 2019 de https://es.wikipedia.org/wiki/MySQL_Workbench

[13] Méndez, S. (2015, 6 abril). "*Novedades PHP 7"*. Recuperado 4 junio, 2019, de http://www.sgmendez.com/2015/03/31/novedades-php-7

[14] Interfaz de programación de aplicaciones. (s.f.). En *Wikipedia*. Recuperado el 3 de junio de 2019 de https://es.wikipedia.org/wiki/Interfaz_de_programaci%C3%B3n_de_aplicaciones

[15] Google Developers. (s.f.). *"Firebase Cloud Messaging*". Recuperado 4 junio, 2019, de https://firebase.google.com/docs/cloud-messaging/?hl=es-419

[16] Micayael (2014, 17 julio). "*Métodos GET vs POST del HTTP*". Recuperado 4 junio, 2019, de http://blog.micayael.com/2011/02/09/metodos-get-vs-post-del-http/

[17] Academia Android. (2016, 19 marzo). "*Componentes de una Aplicación Android"*. Recuperado 4 junio, 2019, de https://academiaandroid.com/componentes-aplicacion-android/

[18] Revelo, J. (2016, 9 diciembre). *"Volley: Librería De Android Para Realizar Peticiones Http*". Recuperado 4 junio, 2019, de http://www.hermosaprogramacion.com/2015/02/android-volley-peticiones-http/

[19] Gómez, K. (2017, 27 julio). "*Top 5 Metodologías de Desarrollo de Software*". Recuperado 4 junio, 2019, de https://www.megapractical.com/blog-de-arquitectura-soa-y-desarrollo-de-software/metodologias-de-desarrollo-de-software

[20] Marich (2016, 31 marzo). "*Metodología de prototipos*". Recuperado 4 junio, 2019, de http://marich.blogspot.es/1459397526/metodologia-de-prototipos/

[21] UOC. (s.f.). *"Justinmind Plataforma de prototipado para hacer prototipos de aplicaciones web y móviles que son iguales que los reales | Biblioteca de la UOC"*. Recuperado 4 junio, 2019, de http://biblioteca.uoc.edu/es/recursos/justinmind-plataforma-de-prototipado-para-hacer-prototipos-de-aplicaciones-web-y-moviles

[22] Sal (criptografía). (s.f.). En *Wikipedia*. Recuperado el 5 de junio de 2019 de https://es.wikipedia.org/wiki/Sal_(criptograf%C3%ADa)

[23] Google Developver. (s.f.). "*Fragments | Android Developers*". Recuperado 5 junio, 2019, de https://developer.android.com/guide/components/fragments

[24] phpMyAdmin. (s.f.). En *Wikipedia*. Recuperado el 7 de junio de 2019 de https://es.wikipedia.org/wiki/PhpMyAdmin

[25] Pruebas de caja blanca. (s.f.). En *Wikipedia*. Recuperado el 10 de junio de 2019 de https://es.wikipedia.org/wiki/Pruebas_de_caja_blanca

[26] Caja negras (sistemas). (s.f.).En *Wikipedia*. Recuperado el 13 de junio de 2019 de https://es.wikipedia.org/wiki/Caja_negra_(sistemas)

[27] RamonInvarato (2012, 4 noviembre). "*Activity - entender y usar una Actividad – Jarroba*". Recuperado 3 junio, 2019, de https://jarroba.com/activity-entender-y-usar-una-actividad/