PROYECTO FINAL

PSP, PMDM, ACDD

2021 – 2022

URBIL MELIN

Contenido

Objetivo

El objetivo de esta práctica es crear una aplicación móvil para el sistema operativo Android desde la cual veremos la información en tiempo real de las balizas de Euskalmet.

Las balizas estarán guardadas en una base de datos SQLite la cual la usaremos mediante el ORM Room que trae el mismo Android.

Estas balizas podrán ser activadas y desactivadas, siendo las activadas las cuales podremos visualizar sus datos desde la pestaña de datos.

Por cada baliza activada hay que hacer una petición a la API para obtener sus datos, estos datos serán introducidos en nuestra base de datos en una tabla de lecturas (Readings) para poder pintar gráficos con un histórico de la aplicación. Las API actualiza la información cada 10 minutos, por lo que para tener todos los datos de las balizas tendremos que hacer un scheduler que cada 10 minutos consulte a la API todas las balizas que tenemos activadas y actualice sus datos en la base de datos.

Base de datos

La base de datos es una base de datos SQLite la cual usamos a través del ORM room de Android.

Esta base de datos cuenta con dos tablas, una para las lecturas y otra para las balizas, la de las balizas simplemente cuenta con todos los datos que nos trae el json, tales como la ubicación, el municipio, el tipo de estación, el nombre de la baliza tanto en castellano como en euskera y también cuenta con un campo que no viene del json, el cual guarda si la baliza esta activada o no.

La tabla de los readings cuenta con dos primary keys(balizaId y datetime) y su única función es la de actualizar la información de los readings en la interfaz gráfica, ya que esta tabla nunca tiene mas de un reading por cada baliza ya que cuando se lee un nuevo reading se borra el anterior, pero como la interfaz gráfica se actualiza haciendo uso de observers hacia falta meter la información de la api en la base de datos para poder actualizar en tiempo real la información en la GUI a través del observer.

Diagrama de la base de datos

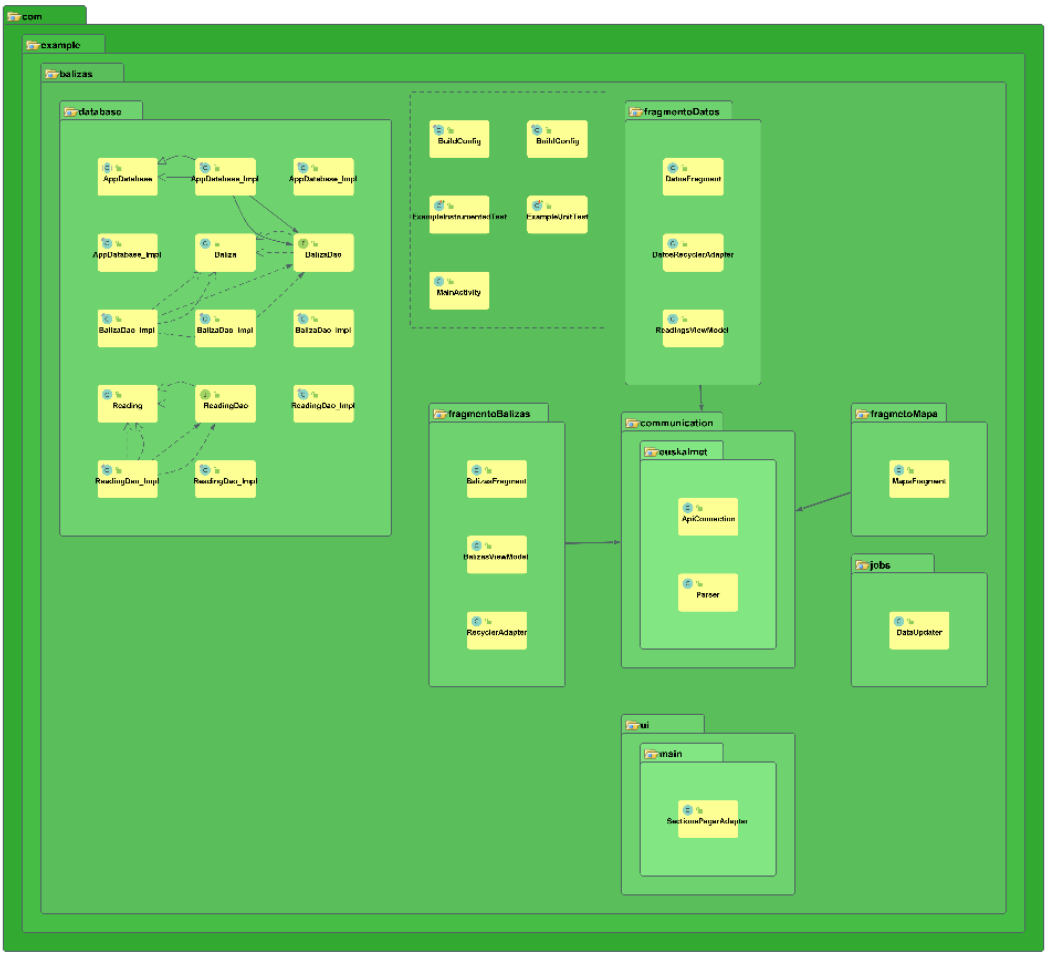
Balizas	
PK	ID
	balizaName char nameEus char municipality char altitude char x char y char stationType char activated char

Readings	
PK	balizaId string NOT NULL datetime string NOT NULL
	temperature double humidity double precipitation double irradiance double

Arquitectura de la aplicación

En cuanto a la arquitectura, esta vez no he seguido un tipo de arquitectura estándar como podría ser la arquitectura MVC, en este caso si que tengo los modelos y las vistas separadas, pero no tengo unos controladores claros ya que en todas las clases hay algo de lógica.

Diagrama de clases



Como podemos ver en el diagrama tenemos 13 paquetes, los tres primeros son los que genera Android Studio por defecto, y los que hay dentro de esos tres son los paquetes principales y en los que está todo el código de la aplicación.

Paquete database

En el paquete database básicamente he puesto las interfaces DAO, los modelos y la clase abstracta AppDatabase.

En los modelos básicamente definimos los atributos del objeto y las columnas de la base de datos y en las interfaces DAO lo que tenemos son los métodos con las queries para poder hacer uso de la tabla de la base de datos a la cual pertenece el DAO.

Por otra parte, la clase AppDatabase es la clase desde la cual accederemos a los DAOs y desde la cual estableceremos la versión de la base de datos.

Paquete communication

El paquete communication es el paquete en el que dentro estarán los paquetes que sirvan para comunicarse con las APIs de los diferentes servicios, como vemos en este caso dentro del paquete communication tenemos únicamente el paquete euskalmet, el cual es el único que nos da servicio actualmente, pero en el caso de querer introducir otro proveedor nuevo habría que hacer en este paquete.

Paquete euskalmet

Este paquete es el que se usa para la comunicación con el servicio de Euskalmet, el cual cuenta con dos clases, ApiConnection y Parser, la clase ApiConnection se encarga de coger los datos de la API y enviárselos a la clase Parser para que los parsee y pueda trabajar con ellos.

Paquete fragmentoBalizas

El paquete fragmentoBalizas como su propio nombre indica es el paquete en el que esta la clase BalizasFragment y sus dependencias como BalizasViewModel y RecyclerAdapter, las cuales sirven para acceder obtener información de la base de datos y para definir el adaptador de la lista reciclable respectivamente.

Paquete fragmentoDatos

Al igual que el paquete anterior este paquete se llama así porque dentro tenemos la clase DatosFragment y en este paquete simplemente esta la propia clase y sus dependencias como ReadingsViewModel, DatosRecyclerAdapter.

Paquete ui.main

El contenido de este paquete realmente son dos solo que uno está dentro del otro, estos paquetes los genera Android Studio de manera automática al iniciar un proyecto con un SectionPagerAdapter, de hecho la única clase que tiene este paquete es una clase que también se llama SectionPagerAdapter.

Paquete jobs

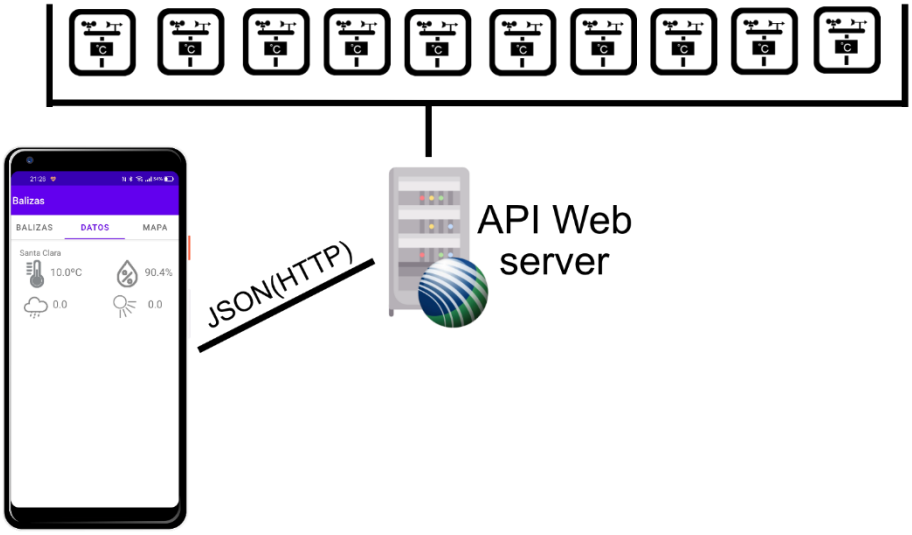
El paquete jobs básicamente consta de una clase la cual se encarga de cada cinco minutos hacer una llamada a la api y traer los datos.

Se llama jobs porque está pensado para guardar en su interior clases que tengan métodos que se ejecuten periódicamente.

Paquete fragmentoMapa

Este paquete únicamente tiene la clase MapsFragment la cual se encarga de mostrar un mapa con las balizas y activar y desactivar las balizas según vamos tocándolas.

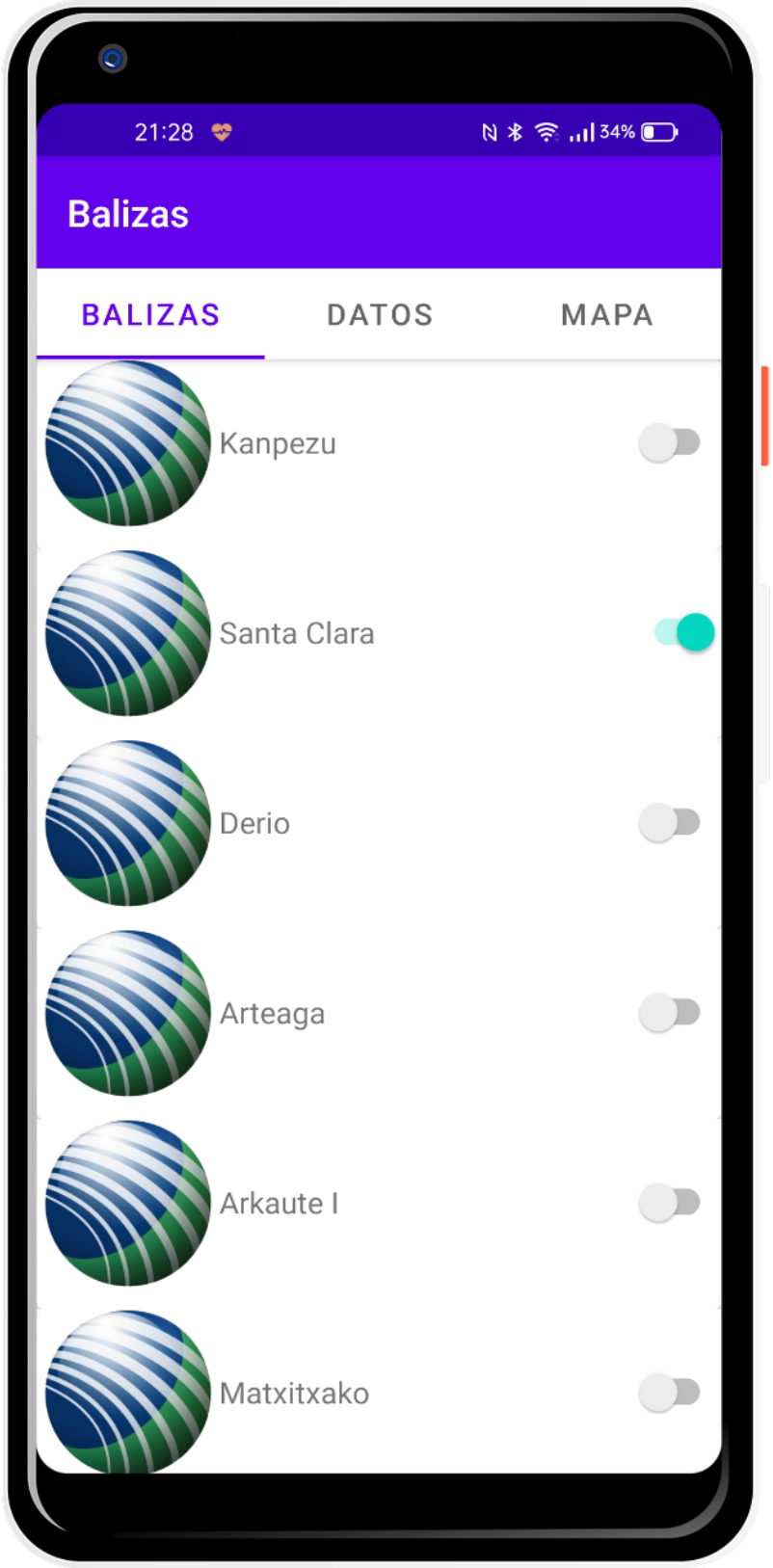
Arquitectura general del sistema



Como podemos ver en la imagen tenemos una serie de balizas meteorológicas conectadas a un servidor web, el servidor genera un archivo JSON con los datos de las balizas y a ese archivo JSON es al que accedemos desde el dispositivo Android a través del protocolo HTTP.

Diseño de la aplicación

La aplicación en general cuenta en la parte superior con un paginador de secciones el cual tiene tres posiciones, Balizas, Datos y Mapa.



Descripción pestaña 1

El diseño de la primera pestaña es simplemente una lista reciclable en la cual mostramos todas las balizas meteorológicas de Euskalmet.

En cada apartado de la lista como se puede apreciar en la imagen tenemos el logo de Euskalmet, el nombre de la baliza en castellano y un switch en el que marcaremos como activada o desactivada la baliza.

Al activar aquí una baliza en la pestaña de datos podremos ver sus datos y en la de mapa nos saldrá de color azul indicando que esta está activada.

Descripción pestaña 2

Para el diseño de la segunda pestaña he escogido un diseño bastante gráfico, usando iconos en lugar de palabras.

En esta pestaña nos encontraremos las balizas que hemos activado ya sea desde el mapa o desde la lista de balizas anteriormente mostrada.

De cada baliza como podemos ver en la imagen sacamos cuatro datos, temperatura, humedad, precipitaciones y radiación solar.

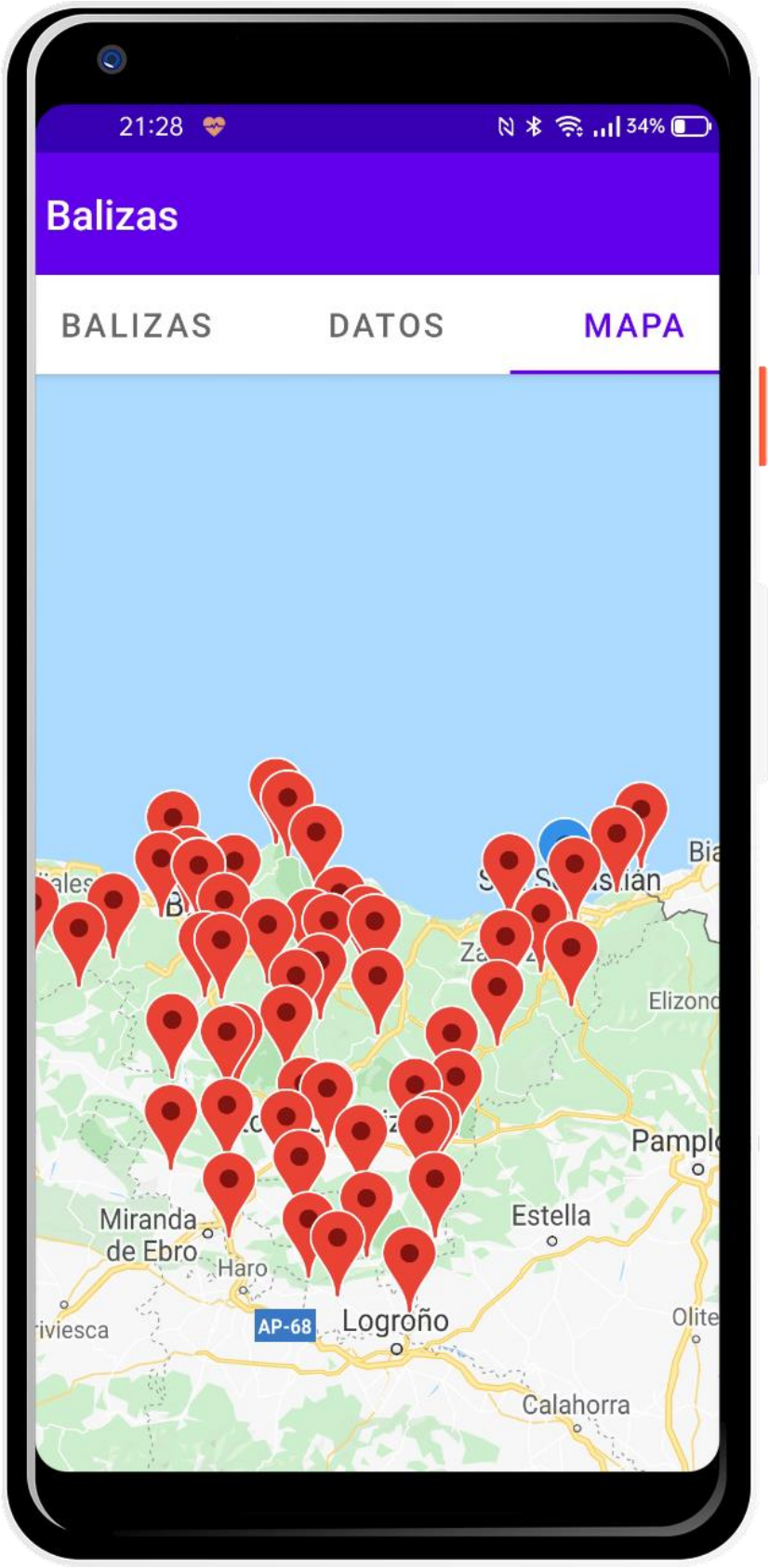
Explicación de los iconos


Temperatura


Humedad


Precipitaciones


Radiación solar



Descripción pestaña 3

La tercera y última pestaña es la pestaña del mapa, en este mapa podremos visualizar todas las balizas del país vasco y haciendo clic sobre ellas las activaremos y podremos ver sus datos desde la pestaña de datos.

Las balizas activadas son marcadas en azul y al hacer clic sobre ellas estas quedarán desactivadas.

Validación y pruebas

Las pruebas que he hecho básicamente consistían en imprimir logs donde he visto posibles puntos de fallo para comprobar que todo funcionara correctamente.

Otra prueba que he hecho es la de descargarme la base de datos room del AVD e inspeccionarla con un visor de bases de datos SQLite.

Código más representativo

El código que creo que merece más la pena analizar es la función que se encarga de parsear los datos del JSON.

Librerías externas

He hecho uso de la librería Joda time para la gestión de las horas y las fechas ya que era mucho mas cómoda de usar que la nativa que trae java.

Recomendaciones para la mejora

Una mejora muy útil seria poner un gráfico que muestre el histórico de cada una de las medidas.

Otra mejora podría ser poner otro proveedor de datos meteorológicos para poder obtener datos de zonas de las que actualmente no tenemos.

Por otra parte, otra mejora podría ser hacer que funcione bien la app al poner el dispositivo en modo oscuro ya que actualmente si bien es verdad que la aplicación funciona en modo oscuro correctamente, al cambiar de modo oscuro a modo claro la aplicación se cierra y hay que volver a abrirla.

Conclusiones finales

Creo que ha sido un proyecto que debido a como está hecho el json que hemos usado para sacar los valores nos ha hecho aprender mucho de algorítmica, ordenación de arrays y gestión de fechas. A mi por ejemplo la gestión de fechas en Java nativo se me ha hecho muy difícil y he tenido que hacer uso de una librería llamada Joda time.

Por otra parte no me ha dado tiempo ha hacerlo todo lo bien que lo hubiera querido hacer debido a problemas en la gestión del tiempo y del proyecto, pero al final ha sido un proyecto que me ha gustado como ha quedado y que creo que ha valido mucho la pena hacer.