I.E.S LAS SALINAS

PROYECTO FINAL FIN DE GRADO

CURSO 22-23

Weather App

CICLO FORMATIVO GRADO SUPERIOR

DESARROLLO DE APLICACIONES MULTIPLATAFORMA

Tutor: Óscar Lillo

Autor: David Manzanero Ocaña

Tabla de contenido

1º Introducción

1.1º Español

En la actualidad, la meteorología es una parte importante de nuestras vidas, ya sea para planificar nuestras actividades diarias o para tomar decisiones a largo plazo. Weather App es una aplicación para consultar la información meteorológica de una ciudad de España en tiempo real.

La aplicación se enfocará en brindar una experiencia fácil y accesible para el usuario, presentando los datos meteorológicos de forma clara y concisa. Además, la aplicación también incluirá información histórica y previsiones para los próximos días, lo que permitirá al usuario planificar de manera efectiva sus actividades al aire libre.

El objetivo principal de este proyecto es crear una herramienta útil y práctica que brinde información confiable y actualizada sobre el clima en una ciudad específica de España.

Hay muchos posibles usos para una aplicación que consulte la información meteorológica de una ciudad de España. Algunos de ellos incluyen:

1. Planificación de actividades al aire libre: Por ejemplo, si se espera que llueva durante todo el día, el usuario puede decidir planificar una actividad bajo techo.

2. Planificación de viajes: Esto les ayudará a empacar la ropa y otros artículos necesarios para el viaje.

3. Agricultura y jardinería: para conocer las condiciones meteorológicas para su área, lo que les permitirá planificar sus actividades de cultivo y jardinería.

4. Seguridad vial: Útil para los conductores que quieren saber si hay condiciones meteorológicas adversas en su ruta.

5. Deportes: Los aficionados a los deportes pueden utilizar la aplicación para conocer las condiciones meteorológicas actuales, como el senderismo o el ciclismo.

1.2º English

Nowadays, meteorology is an important part of our lives, whether it's for planning our daily activities or making long-term decisions. Weather App is an application for checking the real-time weather information of a city in Spain.

The application will focus on providing an easy and accessible experience for the user, presenting meteorological data in a clear and concise manner. Additionally, the application will also include historical information and forecasts for the coming days, allowing users to effectively plan their outdoor activities.

The main objective of this project is to create a useful and practical tool that provides reliable and up-to-date information about the weather in a specific city in Spain.

There are many possible uses for an application that checks the weather information of a city in Spain. Some of them include:

1. Planning outdoor activities: For example, if it's expected to rain all day, the user can decide to plan an indoor activity.
2. Travel planning: This will help them pack the appropriate clothing and other necessary items for the trip.
3. Agriculture and gardening: To know the weather conditions for their area, allowing them to plan their activities.
4. Road safety: Useful for drivers who want to know if there are adverse weather conditions on their route.
5. Sports: Sports enthusiasts can use the application to know the current weather conditions, such as for hiking or cycling.

1.3º Justificación del proyecto

Actualmente, estamos viendo las consecuencias del cambio climático (sequias más prolongadas, tormentas más fuertes y una mayor frecuencia de DANA´s en España, entre otros) tenemos la necesidad de mantener informados a las personas.

Gracias al acceso rápido de la información y a la avanzada tecnología de la que disponemos actualmente, podemos crear una aplicación gratuita para todos los dispositivos Android y, además, obtener los datos de la Agencia Estatal de Meteorología actualizados.

Este es el principal motivo por el que la empresa Iberoestática S.A nos contactó directamente para realizar un proyecto que cubra está necesidad.

Se ha contado con un presupuesto de 500€ mensuales y un plazo de 3 meses para realizar el proyecto.

*En el Anexo 1 consta el equipo de trabajo.

La justificación de este proyecto radica en la importancia de contar con información meteorológica precisa y confiable para la toma de decisiones informadas en diferentes sectores.

Llevando esta idea a desarrollo, una aplicación de estas características podría ayudar a diferentes ámbitos en la vida de las personas. Por ejemplo:

1. Para planificar actividades al aire libre, viajes, y para tomar decisiones a largo plazo.
2. Sectores como: la agricultura, transporte y energía, entre otros.
3. En la investigación científica, para descubrir nuevos conocimientos y tecnologías.

1.4º Ley Orgánica 3/2018 de 5 de diciembre, de Protección de Datos

La aplicación recopila y trata datos personales de los usuarios, como el correo electrónico y número de teléfono, por lo que hay que mencionar los siguientes artículos:

- Artículo 82. Derecho a la seguridad digital.

  Los usuarios tienen derecho a la seguridad de las comunicaciones que transmitan y reciban a través de Internet. Los proveedores de servicios de Internet informarán a los usuarios de sus derechos.

- Artículo 85. Derecho de rectificación en Internet.

  2. Los responsables de redes sociales y servicios equivalentes adoptarán protocolos adecuados para posibilitar el ejercicio del derecho de rectificación ante los usuarios que difundan contenidos que atenten contra el derecho al honor, la intimidad personal y familiar en Internet y el derecho a comunicar o recibir libremente información veraz, atendiendo a los requisitos y procedimientos previstos en la Ley Orgánica 2/1984, de 26 de marzo, reguladora del derecho de rectificación.

*En el Anexo 2 se menciona el Reglamento Europeo porque trata sobre la protección de las personas físicas en lo que respecta al tratamiento de datos personales y a la libre circulación de estos datos.

2º Metodología empleada

<u>Modelo en cascada con realimentación</u>

- Análisis

  En esta fase, se define y redacta los requisitos del software que se van a desarrollar.

  Se detallan concretamente todo aquello que se espera del software.

- Diseño

Se diseña la arquitectura y el diseño del software.

Se definen los componentes y sus funciones.

El diseño está planificado para garantizar su funcionalidad, eficiencia y facilidad de mantenimiento.

- Implementación

Se escribe el código, usando el diseño ya planificado.

Se lleva a cabo la programación, la prueba y la depuración del código para asegurarse que el código cumpla con los requisitos especificados en la fase de análisis.

- Pruebas

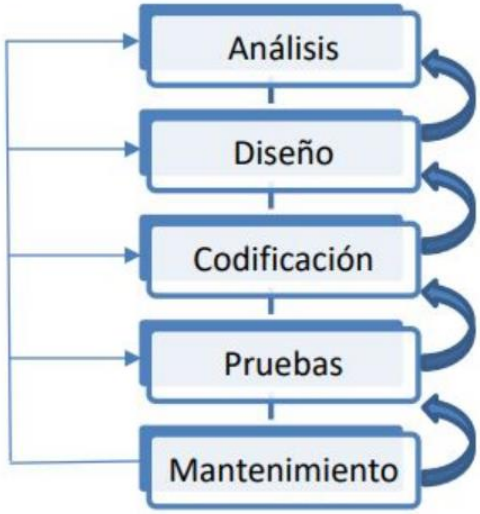
Se realizan pruebas en el software para garantizar que cumple con los requisitos y el diseño especificado.

Se realizan pruebas de unidad, pruebas de integración y pruebas de sistema para garantizar que el software funciona según lo previsto.

- Mantenimiento

El mantenimiento del software se realiza después de su implementación.

Se corrigen los posibles errores y se realizan mejoras para garantizar que el software siga funcionando correctamente y cumpla con los requisitos.



Modelo en cascada

3º Tecnología y herramientas

- API -> AEMET (Interfaz Programación Aplicaciones)
- IDE -> Android Studio (Entorno Desarrollo Integrado)
- Lenguaje -> Java
- Base de datos -> FireBase
- Repositorio -> GitHub
- Diagrama de Flujo -> Diagrams
- Logo de la aplicación -> Desing.AI Logomaker

4º Estimación de recursos y planificación

4.1º Planificación

1. Definir los objetivos del proyecto: Se definen los objetivos del proyecto y cómo se alinean con las necesidades del negocio o del usuario final.
2. Identificar los requisitos del proyecto: Es necesario identificar los requisitos del proyecto en términos de funcionalidad, usabilidad y rendimiento. Esto incluye el análisis de los requisitos de hardware, software y los recursos necesarios para el proyecto.
3. Establecer un cronograma de trabajo: Una vez que se han definido los objetivos y los requisitos del proyecto, es importante establecer un cronograma de

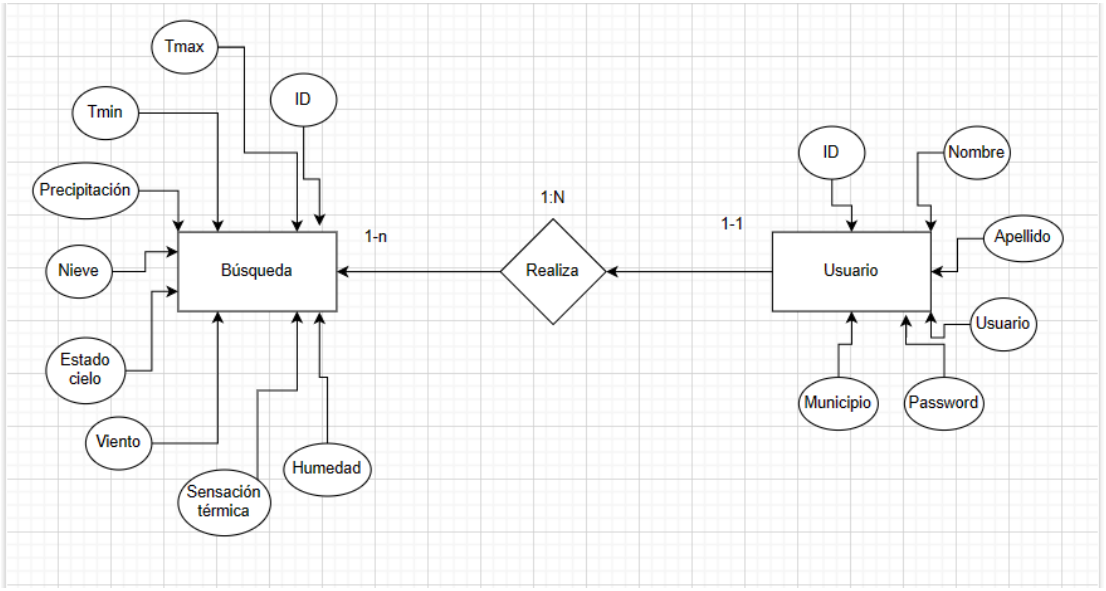
trabajo que incluya los plazos para las diferentes etapas del proyecto, tales como diseño, desarrollo, pruebas y despliegue.

4. Asignar recursos: Es necesario asignar recursos al proyecto, tales como personal, presupuesto, equipo y herramientas. Esto ayudará a garantizar que se cuenta con los recursos necesarios para completar el proyecto en el plazo establecido y con la calidad esperada.
5. Diseñar la arquitectura de la aplicación: La arquitectura de la aplicación es fundamental para garantizar que la aplicación se desarrolla de manera eficiente y escalable. Se debe definir la estructura de la aplicación, la selección de tecnologías y herramientas, y la distribución de tareas entre los miembros del equipo.
6. Desarrollar la aplicación: Una vez que se ha establecido la arquitectura de la aplicación, es importante seguir el cronograma de trabajo y desarrollar la aplicación de acuerdo con los requisitos y objetivos del proyecto.
7. Realizar pruebas y control de calidad: Es fundamental realizar pruebas y control de calidad para asegurarse de que la aplicación funciona correctamente, se ajusta a los requisitos y cumple con las expectativas del usuario final.
8. Desplegar la aplicación: Una vez que se han completado las pruebas y el control de calidad, se debe desplegar la aplicación en la plataforma o tienda de aplicaciones correspondiente.

*En el Anexo 3 se recoge un diagrama de Gantt, mostrando el desarrollo del proyecto con los recursos y tiempo empleado.

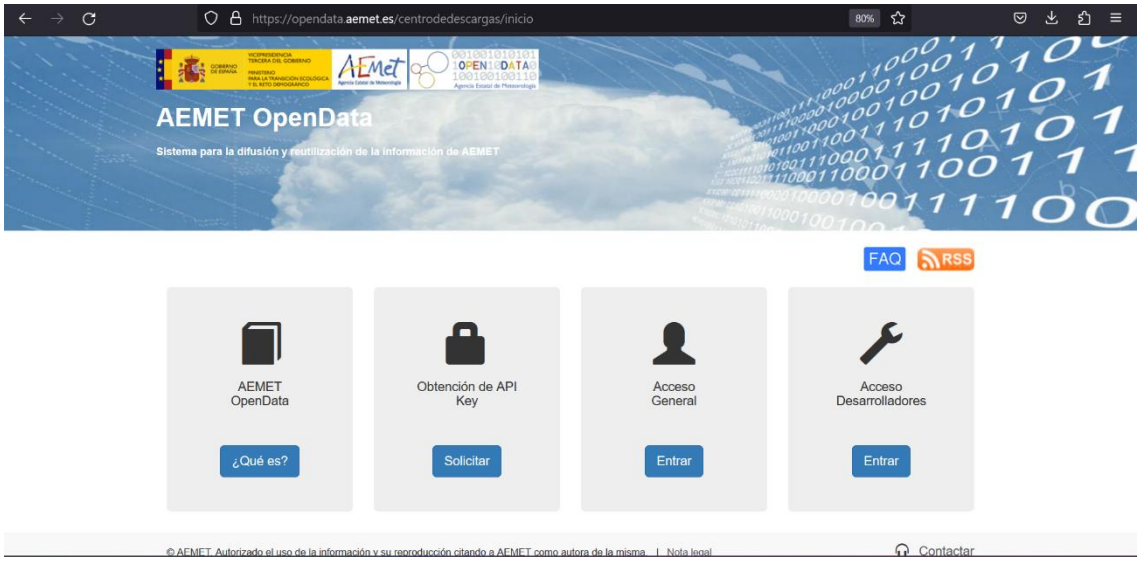
5º Desarrollo del proyecto

5.1º Modelo Entorno-Relacional de la Base de Datos



*En el Anexo 4 mostramos la organización del proyecto, junto con la maqueta con la idea de desarrollo que tenía la empresa.

5.2º Obtención de los datos de la AEMET



AEMET Open Data

Se explica que es AEMET Open Data, una API, los requerimientos para obtener la API Key y las formas de acceso para usuarios y desarrolladores.

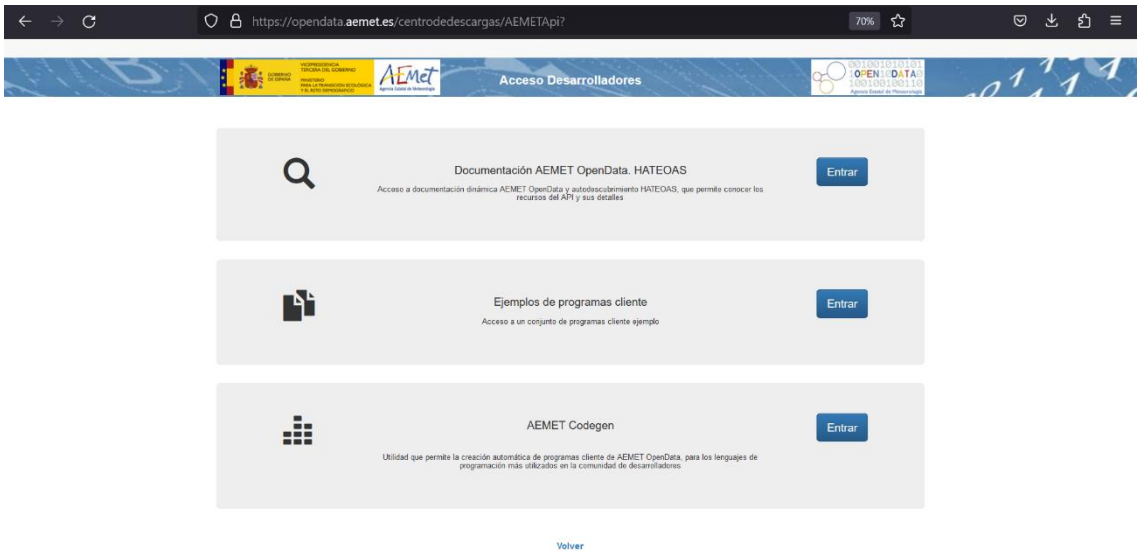
Obtención de API Key

Hay que introducir el email, el cual recibirá la API Key, y pulsar el botón “No soy un robot”.

Acceso General

Introduciendo la API Key recibida, se pueden hacer consultas en los campos que se muestran para verificar que funciona y después, usar la URL en el proyecto.

Acceso Desarrolladores



Documentación AEMET Open Data HATEOSAS

Es una API REST desarrollado por AEMET que permite hacer peticiones de la información meteorológica y climatológica de la Agencia, usando la API Key.

Ejemplos de programas clientes

Se muestra en diferentes lenguajes de programación, como acceder a los datos que ofrece la API.

AEMET Codegen

Swagger Codegen es una herramienta para generar código cliente en diferentes lenguajes de programación a partir del documento Swagger de una API REST.

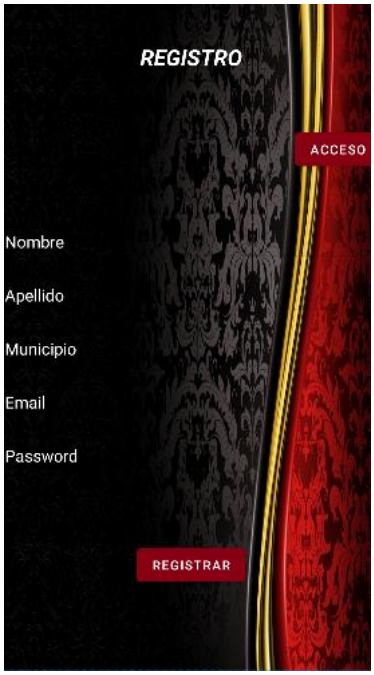
5.3º Diseño de la aplicación

5.3. 1º Diseño de pestañas y acciones

Pantalla de inicio


```java
package com.example.aemetaplicacion.vistas;

import androidx.appcompat.app.AppCompatActivity;

public class HomeActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_home);
    }

    public void iniciar_sesion(View view) {
        Intent intent = new Intent(this, loginActivity.class);
        startActivity(intent);
    }

    public void registrarse(View view) {
        Intent intent = new Intent(this, registroActivity.class);
        startActivity(intent);
    }
}
```

Pantalla de registro

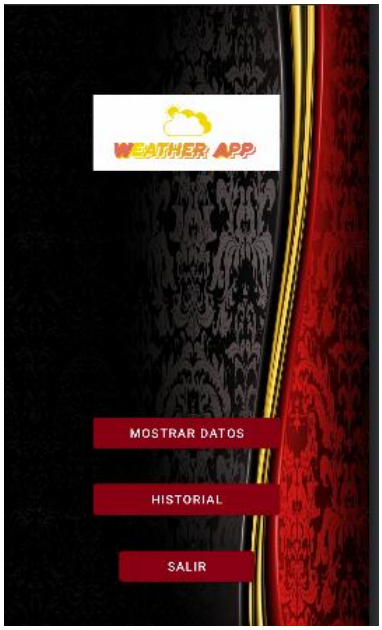



```java
package com.example.aemetaplicacion.vistas;

import androidx.annotation.NonNull;

public class registroActivity extends AppCompatActivity {
    //----------------------------------------
    private EditText edt_nombre;
    private EditText edt_apellidos;
    private EditText edt_municipio;
    private EditText edt_email;
    private EditText edt_password;
    //----------------------------------------
    private FirebaseAuth mAuth;

    @Override
    protected void onCreate(Bundle savedInstanceState) {...
    //-------------------------
    public void onStart(){...
    //-------------------------
    public void registrar(View view) {...
//---------------------------------------------------------------------------
    public void registro_A_login(View view) {...
}
```

Pantalla de inicio de sesión


```java
package com.example.aemetaplicacion.vistas;

import androidx.annotation.NonNull;

public class loginActivity extends AppCompatActivity {

    EditText edt_usuario_login = null;
    EditText edt_password_login = null;
    Button btn_iniciar_sesion = null;
    //-----------------------------------------
    ProgressBar pb_ejecutandose = null;
    int count = 0;
    //-----------------------------------------
    private FirebaseAuth mAuth;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
    //------------METODO PARA PROGRESS BAR----------------
    public void prog(){
    //---------------------------
    public void onStart(){
    //-------------------------
    boolean errores = false;
    public void iniciar_sesion_login(View view) {
    public void login_A_registro(View view) {

}
```

Pantalla principal

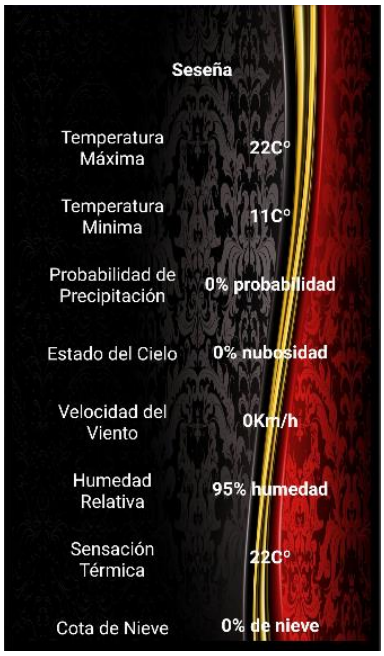



```java
package com.example.aemetaplicacion.vistas;

import androidx.appcompat.app.AlertDialog;

public class pantallaUsuario extends AppCompatActivity {

    TextView txt_nombre_usuario;

    String nombre = "";
    //---------------------------

    @Override
    protected void onCreate(Bundle savedInstanceState) {

    @Override
    public boolean onKeyDown(int keyCode, KeyEvent event) {
    //------------------------------------------------------------------
    public void salir(View view) {
    public void mostrar_historial(View view) {
    public void mostrar_datos(View view) {

}
```

Pantalla para filtrado de datos


```java
package com.example.aemetaplicacion.vistas;

import static com.example.aemetaplicacion.constantes.Peticiones.MUNICIPIOS_API_URL;...

public class elegirDatosActivity extends AppCompatActivity implements AdapterView.OnItemSelectedListener {


    public static final String EXTRA_CODIGO = "dhfvjhbejkvhjwehvl";
    private Spinner sp_comunidad = null;
    private Spinner sp_provincia = null;
    private Spinner sp_municipio = null;
    //-------------------
    private Comunidad comunidadSeleccionada;
    private Provincia provinciaSeleccionada;
    private Municipio municipioSeleccionado;
    //-------------------
    private ArrayList<Comunidad> comunidades;
    private ArrayList<Provincia> provincias;
    private ArrayList<Municipio> municipios;
    //-------------------
    String URL_PETICION1 = MUNICIPIOS_API_URL;
    //-----------------------------------------------------------
    @Override
    protected void onCreate(Bundle savedInstanceState) {...
    //-------------------------------------------------------------------------------
    //-------------------------------------------------------------------------------
    private ArrayList<Municipio> leerPueblosDesdeCSV() {...
    //-------------------------------------------------------------------------------
    //-------------------------------------------------------------------------------
    private ArrayList<Provincia> leerProvinciasDesdeCSV() {...
    //-------------------------------------------------------------------------------
    //-------------------------------------------------------------------------------
    private ArrayList<Comunidad> leerComunidadesDesdeCSV() {...
    //-------------------------------------------------------------------------------
    //-------------------------------------------------------------------------------
    @Override
    public void onItemSelected(AdapterView<?> adapterView, View view, int i, long l) {...
    //-------------------------------------------------------------------------------
    //-------------------------------------------------------------------------------
    private ArrayList<Provincia> ObtenerProvinciasDeComunidad(Comunidad comunidadSeleccionada) {...

    //-------------------------------------------------------------------------------
    //-------------------------------------------------------------------------------
    private ArrayList<Municipio> ObtenerPueblosDeProvincia(Provincia provinciaSeleccionada) {...
    //-------------------------------------------------------------------------------
    //-------------------------------------------------------------------------------
    @Override
    public void onNothingSelected(AdapterView<?> adapterView) {...
    //-------------------------------------------------------------------------------
    //-------------------------------------------------------------------------------
    public void buscar_datos(View view) {...
}
```

Pantalla visualización de la información




```java
public class mostrar_Datos_Activity extends AppCompatActivity {

    String tipo_codigo = "";
    public static String URL_PETICION = "";
    //-----------------------------------------------------
    TextView txt_mostrar_datos_municipio = null;
    TextView txt_temp_maxima = null;
    TextView txt_temp_minima = null;
    TextView txt_precipitacion = null;
    TextView txt_cielo = null;
    TextView txt_viento = null;
    TextView txt_humedad = null;
    TextView txt_sens_termica = null;
    TextView txt_nieve = null;

    @Override
    protected void onCreate(Bundle savedInstanceState) {…
    //-----------------------------------------------------
    private void hacer_llamada_al_aemet(String urlPeticion) throws IOException{…
    //-------------------------------------------------------------------------
    void obtener_datos_aemet(String URL_PETICION) throws IOException {
        OkHttpClient client = new OkHttpClient();

        Request request = new Request.Builder()
                .url(URL_PETICION)
                .build();

        client.newCall(request).enqueue(new Callback() {

            @Override
            public void onResponse(@NonNull Call call, @NonNull Response response) throws IOException {
                Gson gson = new Gson();

                final String myResponse2 = response.body().string();

                ArrayList<Datos_Principales> principales = new ArrayList<Datos_Principales>();

                Datos_Principales dp1 = new Datos_Principales();

                try{…
                }catch (JSONException e){ throw new RuntimeException(e); }

                mostrar_Datos_Activity.this.runOnUiThread(new Runnable() {
                    @Override
                    public void run() {…
                });
            }
            @Override
            public void onFailure(@NonNull Call call, @NonNull IOException e) { call.cancel(); }
        });
    }
}
```

*En el Anexo 5 mostramos las imágenes los pasos a seguir para acceder a la información que proporciona la AEMET.

5.3.2º Estética del proyecto

Fondo de la aplicación

Gama de colores

- Botones -> Rojo oscuro “#880011”
- Texto -> Blanco “#FFFFFFFF”
- Uso de la transparencia en diferentes zonas para asegurar una buena estética.

Logo

5.4º Implementación

- Uso de archivos .csv para la almacenamiento y lectura de las comunidades, provincias y municipios.
- Uso y configuración de spinner con datos de municipios, provincias y comunidades autónomas.
- Uso de librerías okhttp3 y Gson para la gestión de peticiones HTTP a API AEMET y conversión JSON a objetos.

*En el Anexo 6 explicamos brevemente la función de cada uno de los métodos del proyecto.

6º Despliegue y pruebas

DESPLIEGUE

- En el marco de la metodología en cascada con retroalimentación, la mayoría de las funcionalidades se han probado conforme se iban implantando.
- Se ha contado con la colaboración de conocidos para probar la aplicación en diferentes dispositivos Android.
- Distribución privada: Si el proyecto es una aplicación personalizada para un cliente específico, se puede desplegar en la empresa del cliente o en la tienda de aplicaciones interna de la organización. La distribución privada ofrece un mayor control sobre el acceso a la aplicación y se puede personalizar para satisfacer las necesidades específicas de la organización.

PRUEBAS

- De funcionalidad: Estas pruebas verifican que todas las funcionalidades de la aplicación funcionen correctamente. Se pueden realizar pruebas en cada pantalla de la aplicación para asegurarse de que las características clave, como la navegación, el registro de usuario, la búsqueda y la compra de productos, funcionen como se espera.
- De compatibilidad: Estas pruebas se realizan para verificar que la aplicación funcione correctamente en diferentes versiones de Android y en diferentes tipos de dispositivos móviles. Es importante asegurarse de que la aplicación se vea y funcione bien en diferentes tamaños de pantalla y resoluciones.
- De usabilidad: Estas pruebas se centran en la facilidad de uso y la experiencia del usuario. Se pueden utilizar encuestas o entrevistas para recopilar comentarios de los usuarios sobre la interfaz de usuario, la navegación y la funcionalidad.
- de rendimiento: Estas pruebas se realizan para evaluar el rendimiento de la aplicación, como la velocidad de carga, la capacidad de respuesta y la eficiencia de la memoria. Se pueden utilizar herramientas de prueba automatizadas para evaluar el rendimiento en diferentes escenarios.

7º Conclusiones

- Objetivos principales alcanzados.
- Algunas funcionalidades se implantaron con algunas limitaciones.
- El proyecto se completó dentro del plazo establecido, logrando una buena gestión de recursos y planificación.
- Se ha contado con la ayuda de conocidos para recibir un *feedback* de los usuarios y así, evaluar las opiniones y sugerencias para futuras mejoras.

- Se proponen algunas mejoras a futuro…
    - Mejoras de la interfaz gráfica.
    - Aumentar el ámbito geográfico, como Portugal y Francia.
    - Técnicas de monetización de la aplicación.
    - Definir un sector concreto de actuación.
    - Compatibilidad con diferentes dispositivos y plataformas.

8º Bibliografía

Repositorio GitHub donde está almacenada el trabajo

https://github.com/DavidManOc/Wheather_App

Imagen cascada con retroalimentación
https://www.clasesdeinformaticaweb.com/entornos-de-desarrollo-de-software/desarrollo-de-software-fases-y-ciclo-de-vida/

Documentación para desarrolladores de Apps en Android Studio

https://developer.android.com/docs?hl=es-419

Obtener la información de los archivos .csv

http://www.programandoapasitos.com/2017/04/como-leer-fichero-csv-con-java.html

Creación diagrama de flujo

https://www.diagrams.net/

Swagger de la AEMET

https://opendata.aemet.es/dist/index.html?#/predicciones-normalizadas-texto/Predicci%C3%B3n%20provincial%20e%20insular%20hoy.%20Archivo.

Consulta códigos postales

https://www.ayuware.es/blog/base-datos-codigos-postales-espana/

Ley Orgánica 3/2018 de 5 de diciembre, de Protección de Datos

https://www.boe.es/buscar/pdf/2018/BOE-A-2018-16673-consolidado.pdf

Reglamento General Protección de Datos

https://www.boe.es/doue/2016/119/L00001-00088.pdf

Logo de la aplicación

Create unique logos easily with Logomaker from Designs.ai

Descarga municipios de España

https://datos.gob.es/es/catalogo/a09002970-municipios-de-espana

9º Anexos

1º Anexo

ESPECIFICACIONES DE LOS DISPOSITIVOS UTILIZADOS

Propiedades del ordenador

- Dispositivo -> Lenovo
- Memoria RAM -> 8,00 GB (7,80 usables)
- Procesador -> 11th Intel Core i3 3.00 GHz
- Almacenamiento -> 450 GB

Propiedades del teléfono

- Dispositivo -> Xiaomi Redmi 7
- Memoria RAM -> 4.00 GB
- Procesador -> Octa-core Max 2.20 GHz
- Almacenamiento -> 64.00 GB

2º Anexo

REGLAMENTO (UE) 2016/679 DEL PARLAMENTO EUROPEO Y DEL CONSEJO
de 27 de abril de 2016

Reglamento General Protección de Datos

Artículo 67

Entre los métodos para limitar el tratamiento de datos personales cabría incluir los consistentes en trasladar temporalmente los datos seleccionados a otro sistema de tratamiento, en impedir el acceso de usuarios a los datos personales seleccionados o en retirar temporalmente los datos publicados de un sitio internet.

3º Anexo

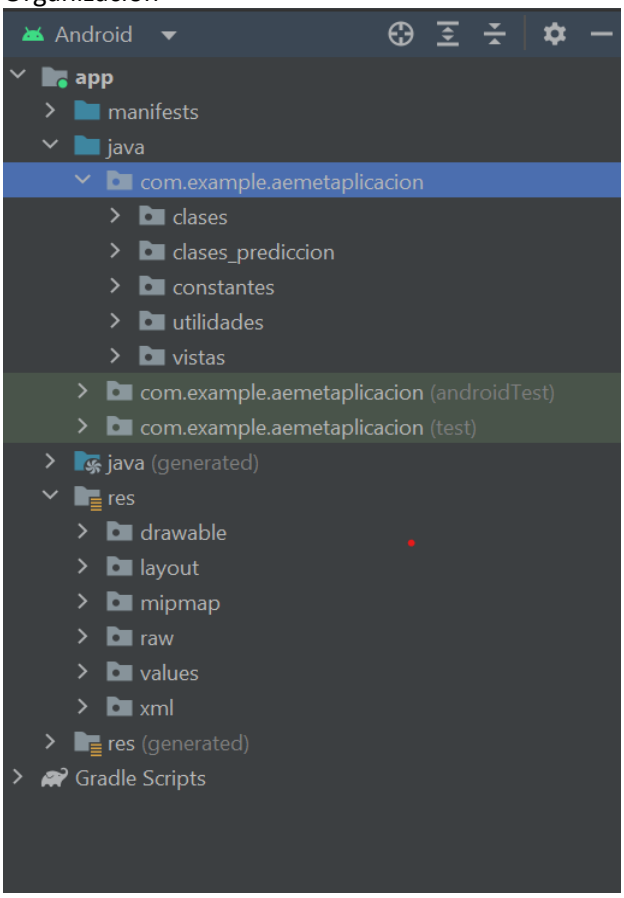
Diagrama de Gantt

			MARZO 2022									JUNIO 2022		
TASK ID	TASK	% DONE	WEEK 1	WEEK 2	WEEK 3	WEEK 4	WEEK 5	WEEK 6	WEEK 7	WEEK 8	WEEK 9	WEEK 10	WEEK 11	WEEK 12
1	**Definición y análisis de requisitos**													
1.1	Definición de objetivos generales	100%												
1.2	Definición de metodología a utilizar	100%												
1.3	Estimación de recursos y planificación	100%												
1.4	Definición de herramientas a utilizar	100%												
1.5	Definición de requisitos específicos	100%												
2	**UX / UI de la aplicación**													
2.1	Diseño del logo	100%												
2.2	Diseño paleta de colores	100%												
2.3	Diseño elementos de la App	100%												
3	**Diseño lógico de la App**													
3.1	Diseño de la base de datos	100%												
3.2	Diseño WireFrames	100%												
4	**Implementación del Software**													
4.1	Registro de usuarios	100%												
4.2	Logueo de usuarios	100%												
4.3	Filtrar por provincias y municipios	100%												
4.4	Revisión del historial de usuarios	0%												
5	**Testing**													
5.1	Definición plan de test	0%												
5.2	Ejecución test de Registro	100%												
5.3	Ejecución test de Logueo	100%												
5.4	Ejecución test de filtrado por municipios	90%												
6	**Documentación**													
6.1	Desarrollo de la memoria	80%												

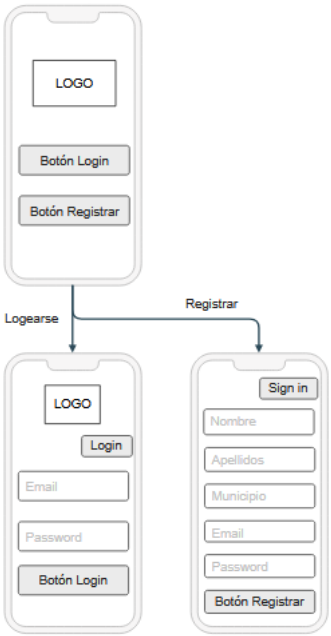
4º Anexo

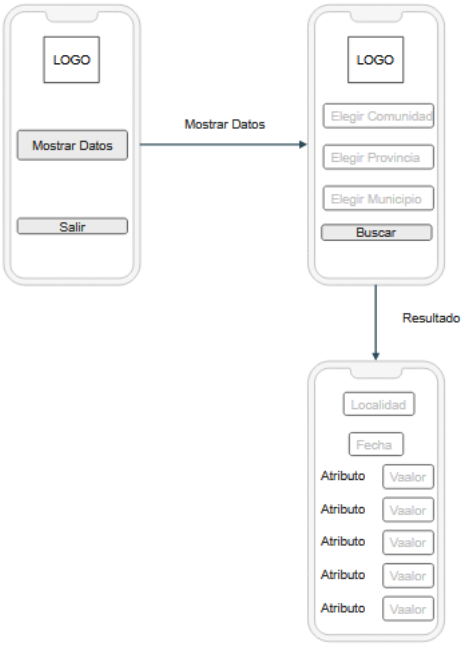
COMPOSICIÓN DEL PROYECTO

Organización



Creación de la maqueta

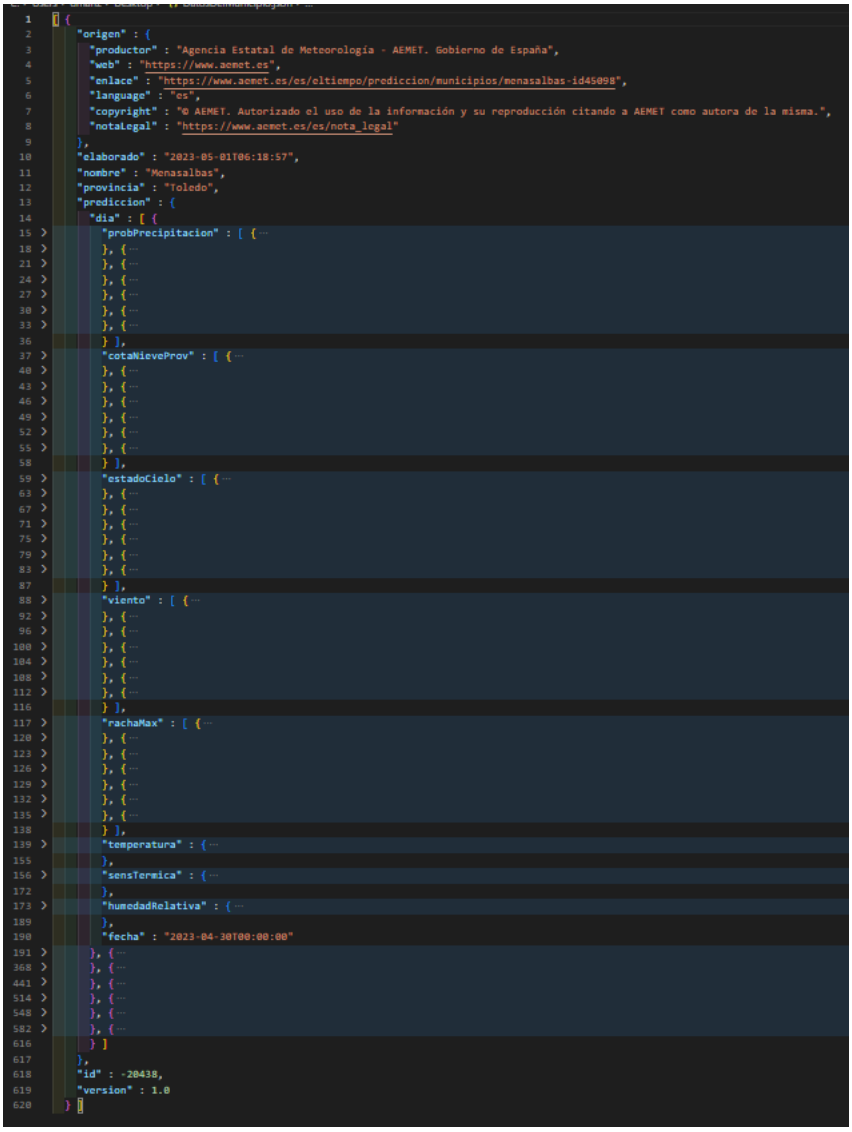

LOGO
Botón Login
Botón Registrar
Logearse
Registrar
LOGO
Login
Email
Password
Botón Login
Sign in
Nombre
Apellidos
Municipio
Email
Password
Botón Registrar


LOGO
Mostrar Datos
Salir
Mostrar Datos
LOGO
Elegir Comunidad
Elegir Provincia
Elegir Municipio
Buscar
Resultado
Localidad
Fecha
Atributo
Vaalor
Atributo
Vaalor
Atributo
Vaalor
Atributo
Vaalor
Atributo
Vaalor

5º Anexo

Mostramos la información que aparece al introducir en el buscador la petición

Para mostrar la información, hay que acceder al enlace “datos”



Enlace para acceder a la petición

https://opendata.aemet.es/opendata/api/prediccion/especifica/municipio/diaria/45161/?api_key=eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJkLm1hbnphLm9AZ21haWwuY29tIiwianRpIjoiMzI5NTQzZDQtOWZlOC00ZWI3LTlmNGYtNWY2NDBhOGQ4MDMxIiwiaXNzIjoiQUVNRVQiLCJpYXQiOjE2Nzk5OTQwMDUsInVzZXJJZCI6IjMyOTU0M2Q0LTlmZTgtNGViNy05ZjRmLTVmNjQwYThkODAzMSIsInJvbGUiOiIifQ.iS-pyI2rDx8luybpjTqXn5p2T-ICFNxX184BxRGN9tM

6º Anexo

Explicación de los métodos de la aplicación

- Pantalla de inicio

  - ➔ OnCreate: inicialización de la actividad
  - ➔ Iniciar_sesion: Botón que te lleva a la vista de Iniciar Sesión
  - ➔ Registrarse: Botón que te lleva a la vista de Registro

- Pantalla de registro

  - ➔ Definimos las variables que se deberán rellenar
  - ➔ Definimos una variable necesaria para la conexión del registro en FireBase
  - ➔ OnCreate:
    - Inicializamos la vista
    - Inicializamos la aplicación de FireBase
    - Asociamos las variables previamente definidas a su correspondiente ID
  - ➔ onStart: método que el sistema invoca una devolución de llamada.
  - ➔ registrar: método para registrar los datos introducidos en la base de datos y se realizan comprobaciones para evitar que accedan sin introducir datos
  - ➔ registra_a_login: botón para acceder a la ventana de iniciar sesión

- Pantalla de inicio de sesión

  - ➔ Primero definimos las casillas que debemos rellenar
  - ➔ OnCreate:
  - ➔ onStart: se realiza una devolución de llamada
  - ➔ iniciar_sesion_login: método para acceder a la aplicación si, previamente, está registrado
  - ➔ login_a_registro: botón para acceder a la vista de registro

- Pantalla principal

  - ➔ OnCreate: inicialización de la actividad
  - ➔ onKeyDown: dando al botón de atrás, puedes salir de la aplicación
  - ➔ salir: botón para salir de la aplicación
  - ➔ mostrar_historial: botón que te lleva a la vista del historial (futura implementación)

➔ mostrar_datos: botón que te lleva a la vista del filtrado de datos

- Pantalla para filtrado de datos

➔ Definimos una constante EXTRA_CODIGO que almacenará el código del municipio ya filtrado
➔ Definimos los spinners que almacenará la Comunidad Autónoma, Provincia y Municipio en ese orden.
➔ Almacenamos las clases de Comunidad, Provincia y Municipio en una variable.
➔ Almacenamos las clases de Comunidad, Provincia y Municipio en un ArrayList.
➔ OnCreate:
➔ leerPueblosDesdeCSV: método para leer el fichero pueblos_por_provincias.csv y visualizar el nombre del pueblo en el spinner correspondiente
➔ leerProvinciasDesdeCSV: método para leer el fichero provincias_por_comunidad.csv y visualizar el nombre del pueblo en el spinner correspondiente
➔ leerComunidadesDesdeCSV: método para leer el fichero comunidades.csv y visualizar el nombre del pueblo en el spinner correspondiente
➔ onItemSelected:
➔ ObtenerProvinciaDeComunidad: A través de un código ya definido en los .csv, se obtiene la provincia correspondiente a esa comunidad
➔ ObtenerMunicipioDeProvincia: A través de un código ya definido en los .csv, se obtiene el municipio correspondiente a esa provincia
➔ onNothingSelected: Cuando no haya nada seleccionado, se mostrará una opción por defecto
➔ buscar_datos: método asociado al botón que lleva a la vista donde se visualizarán los datos y, además, almacenará el código del municipio para la petición a la API

- Pantalla visualización de la información

➔ Definimos la variable “tipo_codigo” que recoge el valor del código del municipio
➔ Definimos una constante URL_PETICION que almacenará la petición de la API
➔ Definimos las variables que se van a rellenar en la vista
➔ OnCreate:
  - Inicializamos la vista
  - Asociamos las variables previamente definidas a su correspondiente ID
  - Recibimos el código de la ventana anterior
  - Invocamos al método “Hacer_llamada_al_aemet”
➔ Hacer_llamada_al_aemet: método que usa la constante URL_PETICION para acceder al archivo JSON
➔ Obtener_datos_aemet: método que recupera los datos que necesitamos y los volcamos en la vista