UNIVERSIDAD EUROPEA DE MADRID

ESCUELA ARQUITECTURA INGENIERÍA Y DISEÑO

CICLO FORMATIVO DE GRADO SUPERIOR
DESARROLLO DE APLICACIONES MULTIPLATAFORMA

PROYECTO FIN DE CICLO

Jose Maria de Federico Arnillas, Eloy Fernández Gallut, Adrián Marcos García.
CURSO 2017-18

TÍTULO: AuxiNet

AUTORES: Jose Maria de Federico Arnillas

Eloy Fernández Gallut

Adrián Marcos García

TUTOR DEL PROYECTO: Ernesto Ramiro Córdoba

FECHA DE LECTURA: 13 de junio de 2018

CALIFICACIÓN:

Fdo:

Ernesto Ramiro Córdoba
Tutor/a del Proyecto

RESUMEN:

Auxinet es una aplicación que se ha diseñado para facilitar la comunicación entre una persona dependiente o de tercera edad, con otra persona voluntaria que quiera ayudar a esa persona. Se utilizará la ubicación de la persona que necesita la ayuda, para mostrarle al voluntario donde se encuentra, permitiendo la facilidad de si se encuentra cerca o tiene tiempo suficiente, llamarla y acercarse en caso de que sea necesario, según el tipo de ayuda que haya solicitado.

Permite saber desde el momento que el dependiente hace clic en un tipo de ayuda, donde se encuentra, ya que se refleja en un mapa que le aparecerá al voluntario, indicando su distancia y el tipo de necesidad que necesita.

Una vez que aparece en el mapa, el voluntario puede clicar en él, obteniendo la información del tipo de ayuda y el nombre del dependiente, permitiéndole realizar una llamada o ver cuánto tiempo puede tardar en llegar hasta él.

También dispone de una información que haya podido incluir la persona dependiente, ya sea su edad, peso… o alguna dolencia, medicación o notas médicas.

ABSTRACT:

Auxinet is an application that has been designed to facilitate communication between a dependent or elderly person with another volunteer who wants to help that person.

The location of the person who needs help will be used to show the volunteer where he is. This will facilitate if they are close or if they have enough time to call if necessary depending on the type of request for help.

Allows you to know from the moment that the dependent clicks on a type of help, where it is located, since it is shown on a map that will appear to the volunteer, indicating their distance and the type of help they need.

Once it appears on the map, the volunteer can click on it, getting information on the type of help and the name of the dependent, allowing him to make a call or see how far is it to get there.

It also has information that may have included the dependent person, whether their age, weight or some disease, medication or medical notes.

AGRADECIMIENTOS

El equipo de desarrollo que lleva a cabo el proyecto AuxiNet a todas las personas que se han visto involucradas no solo en su desarrollo, si no en la ayuda ofrecida para llevar este proyecto hasta su finalización.

A los familiares y amigos que han querido ayudar, permitiendo el tiempo libre a cada uno de nosotros y dándonos todo su apoyo.

A varias asociaciones de vecinos, que nos han ayudado con los tipos de alertas e ideas sobre la información que sería interesante que otra persona puede ver de tí para que te pueda dar una ayuda más personalizada.

Por supuesto y no menos importantes a nuestros guías en este duro pero a la vez gratificante camino:

- Al gran maestro de las técnicas ninjas de Android, el Sensei Luis: que ha tenido la paciencia suficiente para aguantarnos y ayudarnos con el uso de Firebase, tratamientos de Json, uso de Android Studio y un largo etc.

- Raúl Rodríguez, el consigliere: con sus buenos consejos y su guía cuando nos podíamos encontrar perdidos con nuestras dudas o problemas.

- Jose Antonio Perez, el gran maestro Albus Dumbledore: con sus clases magistrales y su aguante enseñándonos Python.

- Nuestro maestro Yoda, Pedro Jose Camacho: el cuál es el culpable de que hoy no podamos vivir sin ver nuestro precioso código bien tabulado y que sea nuestro guía espiritual de las BBDD relacionales y amante de picar código como locos.

- Y, por último, pero no menos importante, a nuestro Obi-Wan Kenobi Ernesto Ramiro: que os demostró que hay que querer igual a Linux, que es muy importante darle a una persona la facilidad de uso de una aplicación y que no por ello tenga que ser desagradable a la vista.

INDICE

1. Introducción 7
1.1. Objetivos 7
1.2. Motivación 8
1.3. Antecedentes 8
2. DESARROLLO DEL Proyecto 10
2.1. Herramientas tecnológicas 10
2.2. Planificación 18
2.3. Descripción del trabajo realizado 18
2.4. Resultados y validación 29
3. CONCLUSIONES 30
3.1. Innovación 31
3.2. Trabajo futuro 31
4. BIBLIOGRAFÍA Y WEBGRAFÍA 32
5.Anexos 33

1. Introducción

Nuestro proyecto se trata de la realización de una aplicación para ayudar a las personas que tengan algún tipo de discapacidad, abandono por parte de familiares o que se encuentren solas y puedan requerir algún tipo de ayuda.

Hemos decantado la aplicación hacia el sector de los móviles, ya que es la plataforma que más gente comparte y tiene un fácil manejo para cualquier persona.

Al igual que el sistema operativo elegido es Android, ya que también es la plataforma más usada y más fácil de personalizar y desarrollar aplicaciones.

La aplicación cuenta con 2 funcionalidades distintas: una persona que requiera algún tipo de ayuda y un voluntario.

Para la persona que requiere la ayuda, dispone de una serie de botones con ayudas predefinidas, que una vez le dé, notificará a los voluntarios a través de un mapa, que alguien necesita ayuda.

1.1. OBJETIVOS

Al tratarse de un proyecto de fin de ciclo, ponemos en práctica todo lo aprendido durante estos dos años y también unas bases para poder incorporar otras tecnologías que no hayamos utilizado durante el mismo.

Como ya se ha comentado anteriormente, la aplicación consta de dos apartados:

- La parte para la persona que necesita ayuda: se trata de una interfaz más clara y fácil de usar, con distintos eventos que luego aparecerá en la interfaz del voluntario para poder recibir la ayuda solicitada.

- La parte del asistente o voluntario: donde recibirá las notificaciones de las personas que puedan necesitar algún tipo de ayuda, ver su información personal y ponerse en contacto con ella.

Los hitos más importantes que hemos conseguido son:

- Utilización de una base de datos MySQL: Queríamos trabajar con una base de datos diferente a Firebase, ya que en un anterior trabajo la utilizamos y teníamos la necesidad de aprender algo nuevo, con las dificultades que esto puede traer en un trabajo en tan poco tiempo.

- Aprender un lenguaje nuevo, PHP: Para la comunicación entre la Base de Datos y la aplicación Android, hemos utilizado como pasarela, documentos PHP para cada método.

- Uso de Google Maps: Parte de nuestra aplicación se basa en la ubicación y poder mostrarlo en un mapa, para que quede claro donde se encuentra la persona que necesita ayuda, su distancia y las herramientas que ofrece Google, como la ruta, marcadores, etc.

1.2. Motivación

No teníamos claro en un principio que proyecto realizar, nos pusimos en contacto con varias ONG (ONCE, La CAIXA) para ver si tenían necesidad de algún proyecto de ayuda. No nos contestaron de forma positiva, pero si nos dieron alguna ayuda al respecto sobre la parte del voluntariado y los problemas que hay en los tiempos de espera cuando quieren relacionar al voluntario con una persona necesitada.

Por lo que nos metimos de lleno en ese apartado, intentando hacer más rápida la forma en que se comunica una persona que tiene una necesidad puntual (ya sea en forma de ayuda o compañía), con los voluntarios.

También realizamos encuestas a diferentes vecinos (aprovechando reuniones vecinales, familiares, etc.) para conocer qué tipo de ayudas serían las que más se suelen necesitar o que problemas se podrían encontrar a la hora de manejar la aplicación.

Esto nos dio una motivación extra, ya que disponíamos de unos datos reales de cómo hacer llegar a esas personas necesitadas una aplicación fácil, sencilla e intuitiva.

1.3. Antecedentes

Para el cuidado y la ayuda de personas mayores, únicamente podemos encontrar aplicaciones de pago, como por ejemplo Joyners. En el apartado gratuitas todo lo que podemos encontrar son aplicaciones para recordar tomar pastillas o que cambien la interfaz por una más sencilla.

La aplicación Joyners tiene bastantes elementos en común con la nuestra, como la solicitud de asistencia en la ayuda a la compra, ayuda en el aseo, etc. El problema fundamental de esta aplicación es que es de pago y dependen también de que te puedan asignar a una persona para que te ayude, ya que tu envías una solicitud de ayuda y ellos te contestan por medio de una centralita cuando podrían atenderle.

Nuestros conocimientos previos eran de Android Studio, con el desarrollo de varias aplicaciones durante el curso, con manejo de diferentes Activities, Fragments así como diferentes elementos en ellas.

Las bases de datos de Firebase y las relacionales (terminamos utilizando estas). Durante la totalidad del segundo año del curso hemos utilizado la herramienta de Firebase (la base de datos que te ofrece de manera gratuita Google), fácil de usar y con muchos tutoriales y documentación oficial. Durante nuestro primer año de curso se puso mucho más hincapié en las bases de datos relacionales, teniendo una estructura muy diferente a la de Firebase (ya que no es relacional).

A la hora de elegir una base de datos diferente a Firebase, tuvimos que realizar una gran investigación para poder realizar llamadas a los datos que queríamos guardar y recoger de la base de datos, ya que no teníamos conocimientos que relacionen Android con una base de datos que no fuera Firebase.

Esto supuso bastante tiempo de investigación, terminando utilizando PHP, como pasarela entre Android y la base de datos.

También tuvimos que investigar para incorporar Maps a nuestra aplicación, ya que al igual que lo anterior, no habíamos trabajado con ello durante el curso.

Disponíamos de conocimientos de Java, que, al tratarse de un idioma de programación bastante común, nos posibilitó el poder realizar búsquedas e investigaciones más avanzadas respecto al código que queríamos utilizar.

2. Desarrollo del proyecto

Para la realización del proyecto hemos utilizado la plataforma Android Studio, ya que lo hemos estado utilizando durante todo el curso y tenemos la familiaridad de su uso. Aparte de que el sistema Android es la plataforma para la que queríamos destinar esta aplicación.

Para la base de datos hemos utilizado MySQL, ya que nos parece mucho más potente y nos da un carácter diferenciador respecto a Firebase.

Para gestionar la Base de datos, hemos utilizado el programa que aporta MySQL: MySQL Workbench.

Para el control de versiones hemos utilizado GitHub, ya que estamos muy familiarizados con él y se integra perfectamente con la plataforma Android Studio.

Como lenguajes de programación, hemos utilizado tanto Java para la parte de desarrollo de la aplicación y PHP, que lo encontramos como pasarela entre la aplicación y la base de datos.

2.1. Herramientas tecnológicas

- **Android Studio:**

Para el desarrollo de la aplicación, hemos utilizado el entorno de desarrollo oficial para Android. Se trata de un entorno gratuito que está disponible para las diferentes plataformas (Windows, Linux, MacOS).

La herramienta pone a nuestra disposición una interfaz dividida por zonas, donde nos muestra las diferentes vistas de un proyecto:

- Donde se encuentran las clases con el código de la aplicación y el código de cada una de las vistas.

- La zona de programación, que es donde insertamos el código en cada una de las clases que puedan tener el programa.

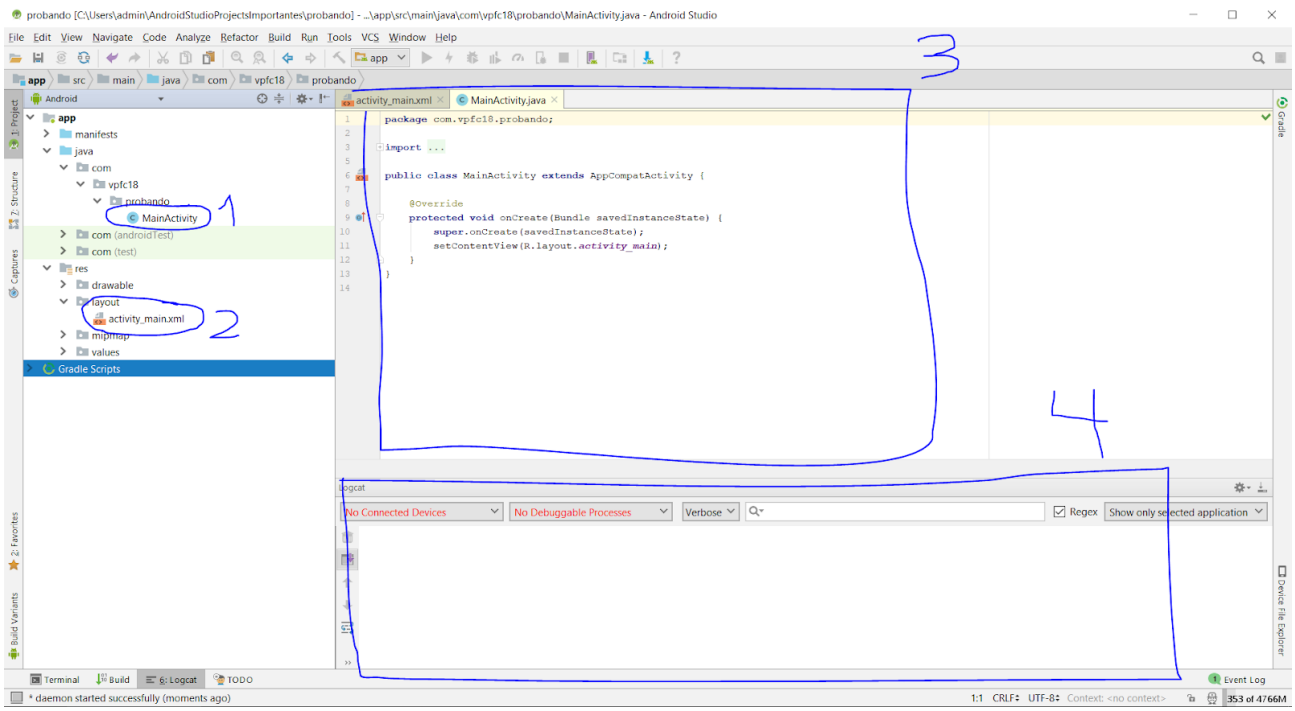
- En la parte inferior, podemos tener la consola, donde se muestra las peticiones que va realizando el programa o los errores que pueda ir dando una vez que lanzamos la aplicación.

- También dispone de un emulador, para poder lanzar la aplicación si no disponemos de un dispositivo Android. varias bibliotecas o recursos, que ayudan a la hora de escribir el código, Este emulador puede ser configurado en cualquiera de las versiones existentes de android, en una gran variedad de tamaños y con una serie de características iniciales que configuremos.

Al tratarse de un programa para Android, los desarrolladores crean herramientas que luego se pueden añadir a tu programa, haciendo que ciertas características no las tengas que crear tu a mano, si no que puedes importarlas y añadirlas a tu proyecto, siendo de gran ayuda a la hora de desarrollar una nueva aplicación.

Permite su integración con GitHub, para el control de versiones, con esto puedes guardar el proyecto en la nube y descargarlo desde cualquier otro dispositivo, a la vez que a un compañero puede bajar el código que tu escribes.

Por último, permite generar el APK, que no es más que la aplicación tal cual, lo que nos permite subirla a la Play Store para poder compartirla y que sea descargada por otras personas.



1 Interfaz de Android Studio

1- Clases con el código.
2- Clases con las vistas
3- Donde se escribe el código de la aplicación
4- Consola

Página oficial: https://developer.android.com/studio/

- **MySQL**

Como base de datos, tenemos la plataforma MySQL, siendo una de las bases de datos más popular del mundo (tratándose de una plataforma de código abierto, aunque no totalmente).

Su creación se remonta a 1995, creada por la empresa MySQL AB, después en 2010 fue comprada por Oracle. El idioma en el que está desarrollada esta base de datos es C y C++.

Dispone de varias versiones, una versión para empresas, permitiendo la monitorización y una asistencia técnica y otra versión pública.

Se trata de una base de datos relacional, en la que las tablas están relacionadas entre sí por medio de una clave o valor común entre ellas.
Es una base de datos fácil de crear y de extender, ya que se pueden añadir nuevas categorías de datos sin tener que modificar los anteriores.

2Logotipo MySQL

Página oficial: https://www.phpmyadmin.net/

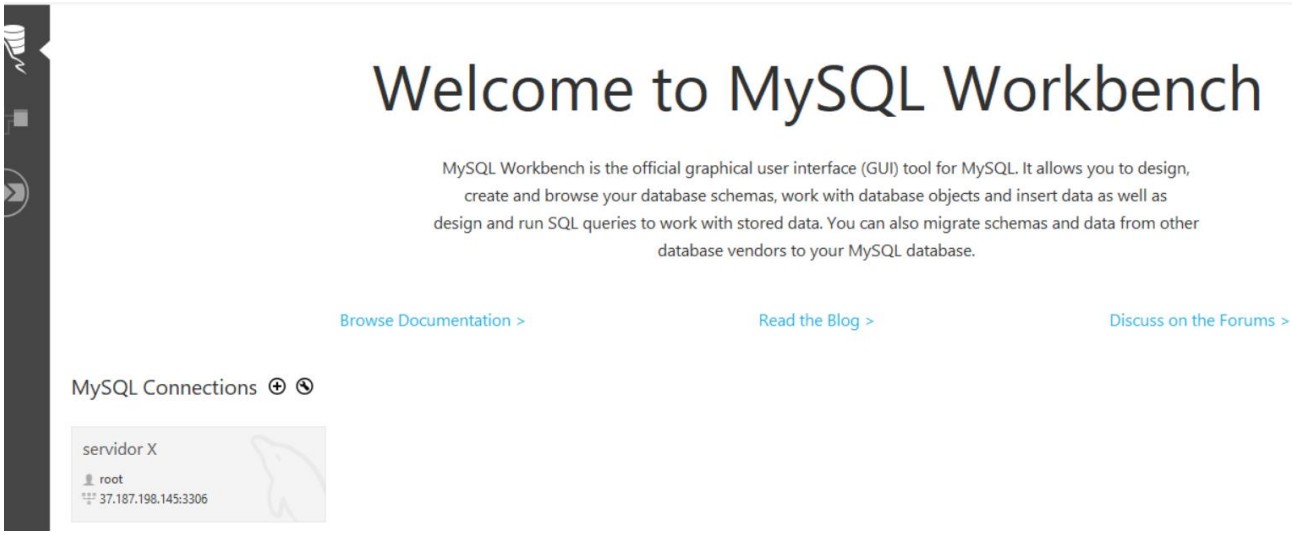
- **MySQL Workbench**

Para la gestión de la base de datos, utilizamos el entorno MySQL Workbench, se trata de una herramienta más potente que la incluida en el servidor, que permite realizar una gestión de ella más rápida y fácil.

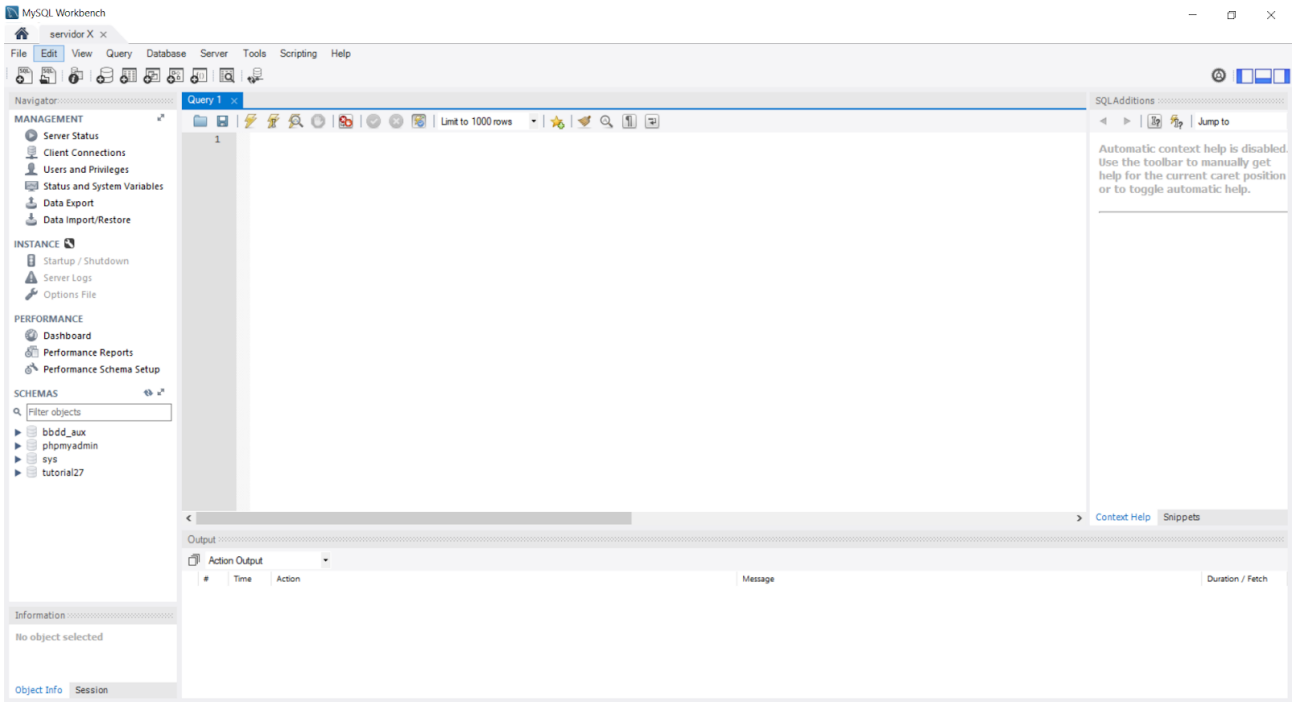
Esta herramienta que se instala en nuestro ordenador, nos ofrece la oportunidad de conectarnos a cualquier servidor al que tengamos acceso y gestionar cada base de datos que tenga disponible, al igual que cada una de las tablas donde se guarda la información.

Puedes crear, modificar o borrar tanto las bases de datos como las tablas desde una interfaz sencilla.

A la hora de crear las tablas, ofrece una interfaz que te ayuda a la hora de seleccionar unas tablas con otras, permitiendo crear las referencias sobre las columnas que son del mismo tipo (puedes tener columnas de números, palabras, fechas…).



3Pantalla principal, donde nos conectamos a los diferentes servidores



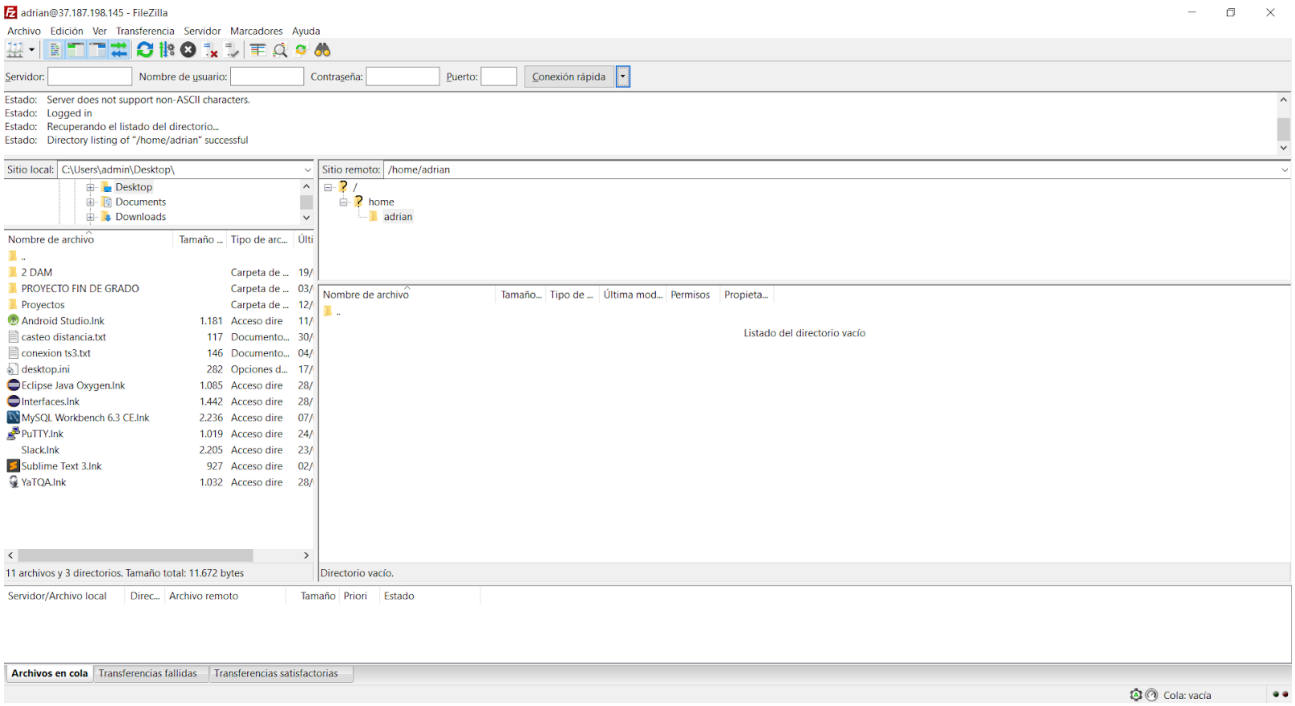
4Dentro de un servidor, con sus diferentes bases de datos en la parte inferior izquierda.

Página oficial: https://www.mysql.com/products/workbench/

- **FileZilla**

Para conectarnos con el servidor, hemos utilizado esta aplicación, que nos permite establecer conexiones y gestión de los archivos que puedan hacer falta dentro de este. Por ejemplo, para insertar los diferentes documentos PHP que hemos utilizado.

Se trata de una aplicación que se instala en nuestro dispositivo y que muestra diferentes ventanas dentro de ella, con las carpetas que se encuentran en el servidor al que nos conectamos, las carpetas de nuestro ordenador actual, así como una consola que nos muestra los mensajes de error o la información sobre el paso de archivos que estemos realizando.



5Aplicación con sus diferentes ventanas

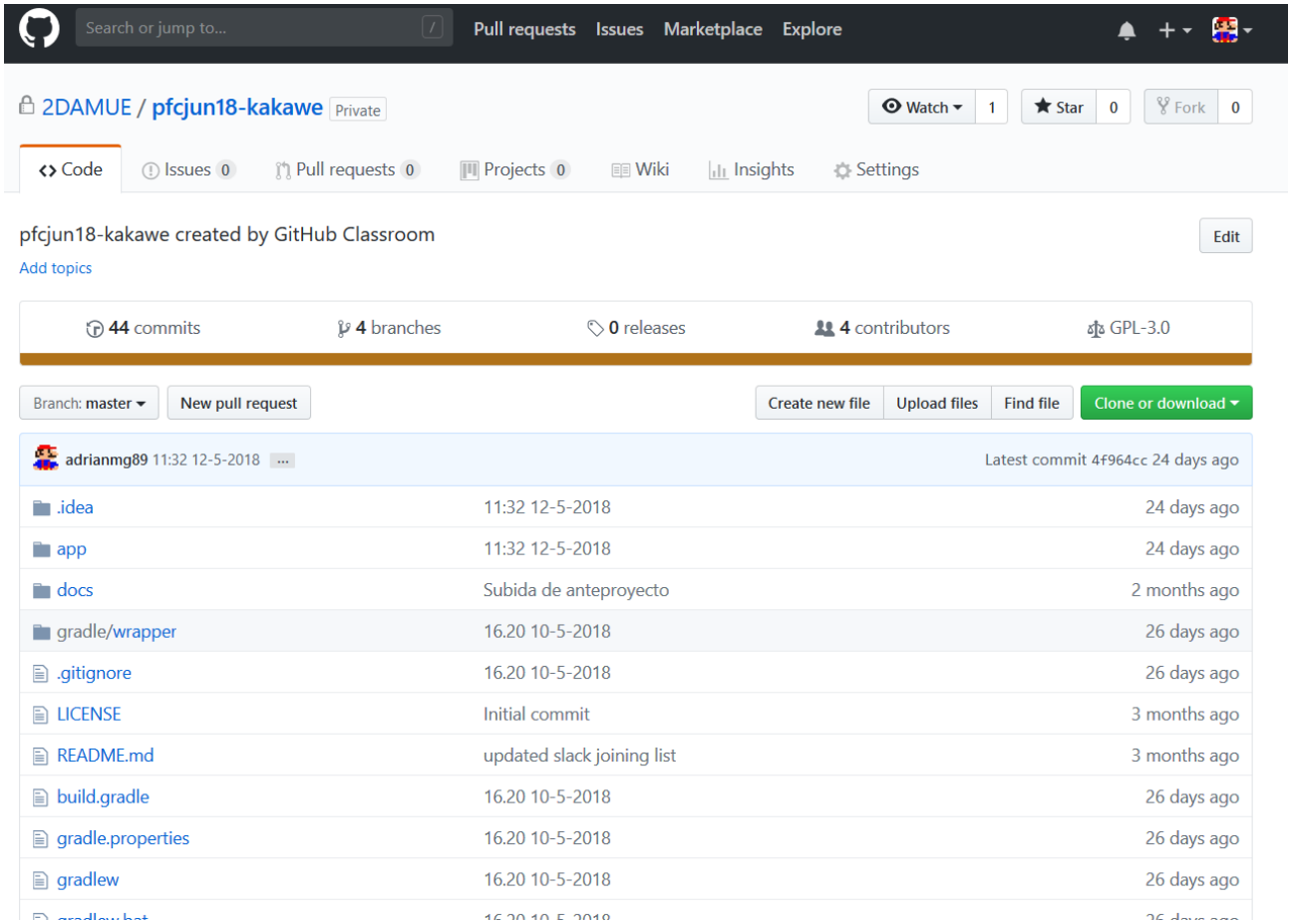
Página oficial: https://filezilla-project.org/

- **GitHub**

Para poder compartir y guardar el código, no solo en nuestro ordenador, hemos utilizado la plataforma de GitHub, ya que nos permite rápidamente subir el código y que nuestros compañeros puedan bajar la última versión y agregar los cambios a su proyecto, esto permite una mayor rapidez a la hora de trabajar, ya que se puede repartir el trabajo mucho mejor.

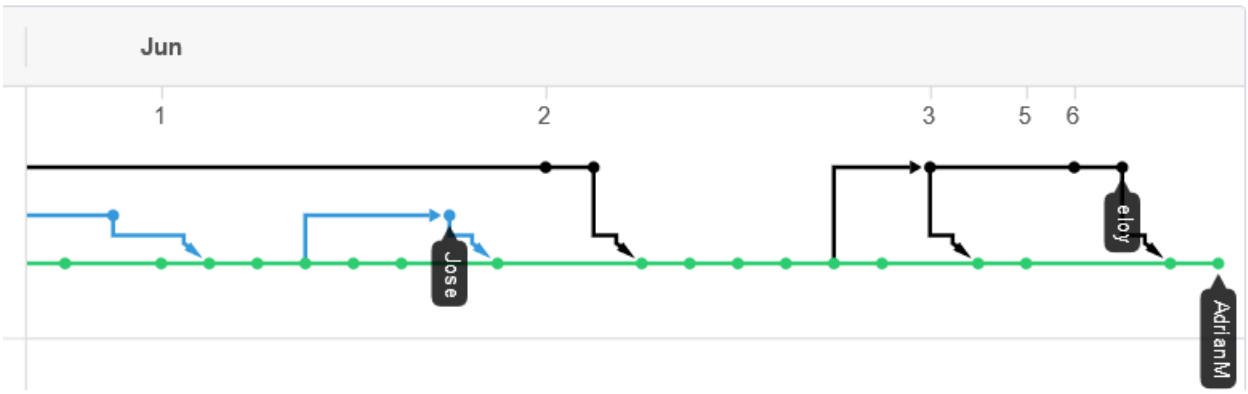
Se trata de una plataforma online donde se trabaja por ramas, permitiendo que varias personas estén al mismo tiempo desde ordenadores distintos, trabajando en el mismo proyecto.

Una vez que se quiere unir el código, te ofrece herramientas para poder "fusionar" ese código con las partes que ha realizado cada persona.



6 Repositorio en GitHub

Para nuestro proyecto utilizamos una rama por cada participante: 3 ramas más la Master.



a Progreso ramas GitHub

Página oficial: https://github.com/

- **Lenguaje de programación Java**

Se trata de un lenguaje de programación orientado a objetos y que puede ser ejecutado en una gran variedad de dispositivos (desde ordenadores, móviles, relojes…), deriva de los lenguajes C y C++.

Este lenguaje En nuestro caso lo utilizamos dentro de la plataforma Android Studio.

```
public class MainActivity extends AppCompatActivity {

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

7 Clase java

- **Lenguaje Php**

Se trata de un lenguaje diseñado para el desarrollo web, siendo unos de los primeros lenguajes de programación que se incorporan en documentos HTML.

Fue diseñado en 1995 y se trata de un lenguaje open Source. Su uso sobre todo se da en servidores Web aunque actualmente está siendo sustituido por otros lenguajes más nuevos y potentes.

```
$correo=$_GET['correo'];
$peso=$_GET['peso'];
$altura=$_GET['altura'];
$grSanguineo=$_GET['grSanguineo'];
$alergias=$_GET['alergias'];
$medicacion=$_GET['medicacion'];
$enfermedades=$_GET['enfermedades'];
$notasMedicas=$_GET['notasMedicas'];


$consulta ="SELECT id_usuario FROM `dependientes` WHERE id_usuario ='$correo'";
        //Si viene un resultado,podemos actualizar los datos medicos
        $implementar = $mysqli->query($consulta);
    if ($resultado = $implementar->fetch_array(MYSQLI_NUM)) {
        //actualizamos el usuario
        $consulta2 = "UPDATE `dependientes` SET `peso` = '$peso', `altura` = '$altura
             '$medicacion', `enfermedades` = '$enfermedades', `notasMedicas` = '$nota
        $implementar2 = $mysqli->query($consulta2);

        $consulta3 ="SELECT * FROM `dependientes` WHERE id_usuario ='$correo'";
        $implementar3 = $mysqli->query($consulta3);
        if ($resultado3 = $implementar3->fetch_array(MYSQLI_NUM)) {
            echo json_encode($resultado3);
        }
    }else{
        echo " no se encontro al usuario ";
    }
?>
```

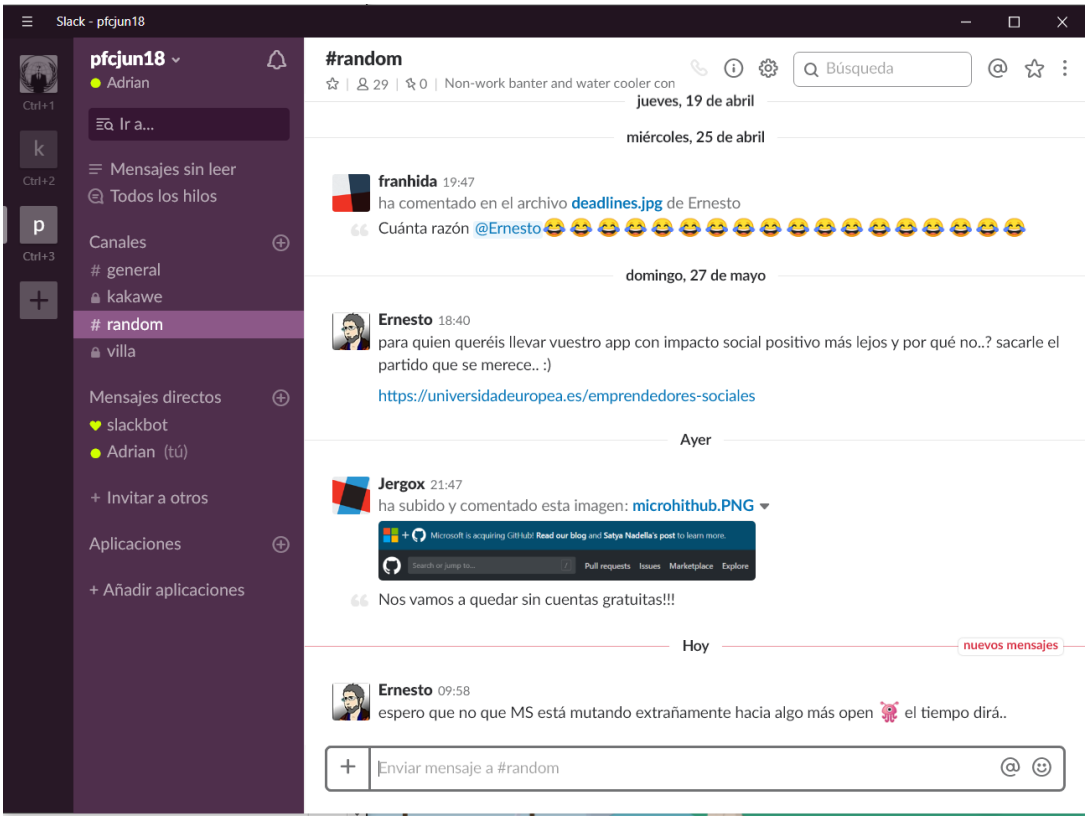
8 Ejemplo de una de nuestras clases PHP

- **Slack**

Como uno de los medios de comunicación, hemos utilizado Slack, un programa organizado por salas, que ofrece conversaciones en grupos o individuales

Permite realizar búsquedas sobre todo el contenido de la aplicación. Puedes tener diferentes espacios de trabajo, con diferentes personas y también permite pasar cualquier tipo de archivo, quedando alojado en sus servidores, permitiendo su descarga más tarde.

Dispone de multiplataforma, tanto en página web, aplicación de escritorio y para dispositivos móviles.



9 Interfaz de escritorio

Página oficial: https://slack.com

2.2. Planificación

El equipo de trabajo está constituido por 3 personas, las cuales han estado usando para su comunicación Slack, Whatsapp y Skype.

La utilización de Skype ha sido fundamental, ya que se realizaban videollamadas al menos una vez a la semana y así suplir el vacío de uno de los miembros al encontrarse en Londres (Reino Unido).

La organización del trabajo se distribuido en tres partes fundamentales:

- Adrian Marcos se ha encargado de la creación de la BBDD (MySql) y backEnd mediante PHP.

- Eloy Fernández se ha encargado del diseño de la app, programación del DashButton (Amazon) y aportando la creación de una clase Modelo de negocio para la interacción de datos.

- Jose M de Federico se ha encargado de enlazar la API de GoogleMaps y los diferentes métodos que se han realizado a la hora de crear los mapas y su funcionalidad.

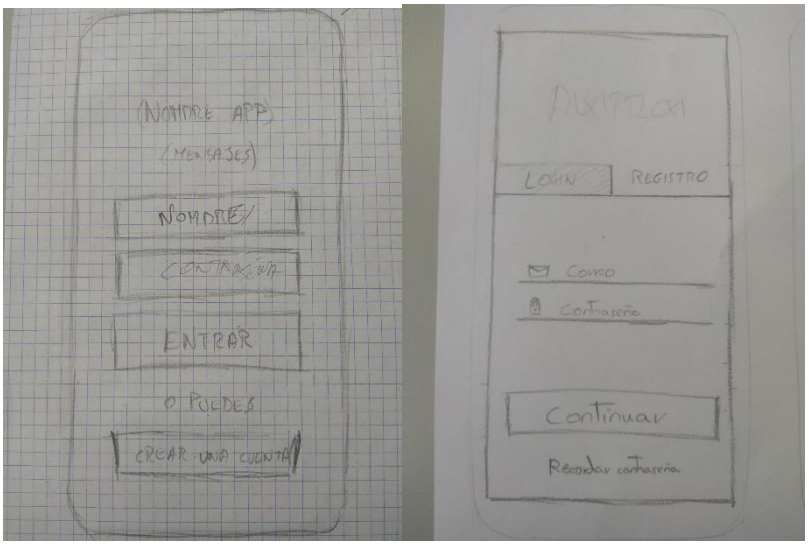
2.3. Descripción del trabajo realizado

El arranque de la app comienza después de un par de reuniones para decidir la dirección que queríamos que tomara la app, tras decidir que sería una app dirigida a personas las cuales por algún tipo de problema necesitará ayuda de manera altruista.

- Login:

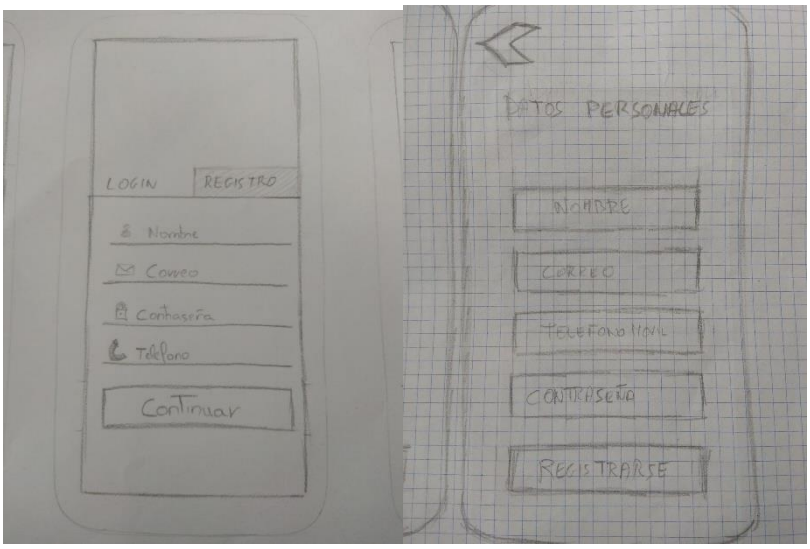
Comienza con los primeros diseños de la app a modo de boceto.

Las primeras ideas fueron la creación de dos Apps una para el asistido y otra para el asistente.



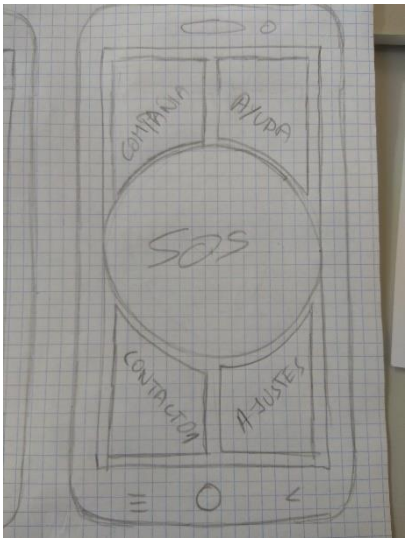
- Registro:

El registro muestra dependiendo de qué rol elijas tendrás que rellenar una vista u otra.



- Vista principal Asistido:

Se crea un diseño de fácil usabilidad e intuitivo para la persona que vaya a utilizar la app en el cual creamos 5 botones con diversas funcionalidades poniendo el foco en el botón de SOS.



Una vez realizado el primer diseño de la app se procede a trasladar la idea a un moqups.

- Login:

Se sigue con la idea inicial de crear dos app, donde el asistido se le proporciona un Login simple con su número de teléfono, el asistente con el correo y contraseña.

10 Asistido *11 Voluntario*

- Registro:

En el registro el asistido solo tendrá que rellenar 3 campos, mientras el el asistente tendrá que proporcionar algunos datos más.

12 Asistido *13 Voluntario*

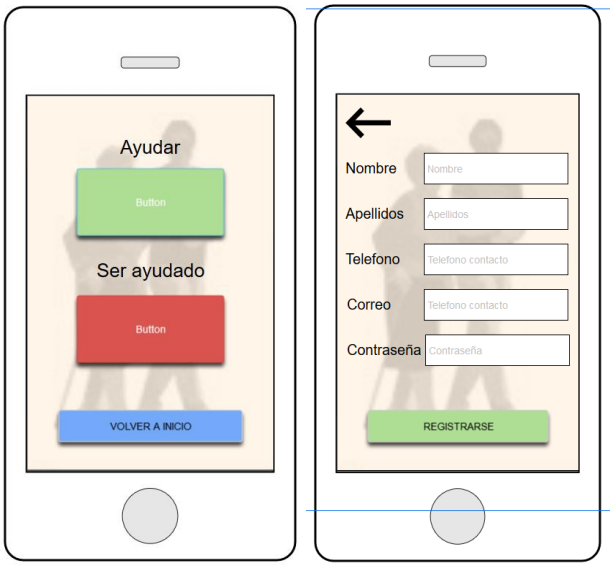
Y antes de comenzar con la creación de la app culminamos con el diseño final(hi-res).

- Login:

Tras cotejar diferentes puntos de vista al final se llegó a la conclusión que se realizaría una única app.

- Registro:

En el registro, primero eliges si eres asistido o asistente (voluntario), después tienes que rellenar una serie de campos obligatorios como el nombre, teléfono, correo y contraseña, más un campo apellido que no es obligatorio para el registro.



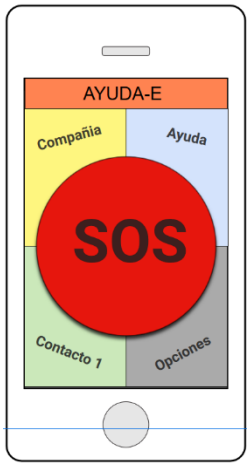
- App Asistido en la que se incluyen 5 botones

Compañía: Manda una alerta a los asistentes pidiendo compañía para el asistido.

Ayuda: Aparecerá un desplegable con diferentes tipos de ayuda pudiendo elegir una de ellas.

Contacto 1 y 2: El asistente podrá personalizar estos botones para que realicen una llama a los contactos previamente asignados.

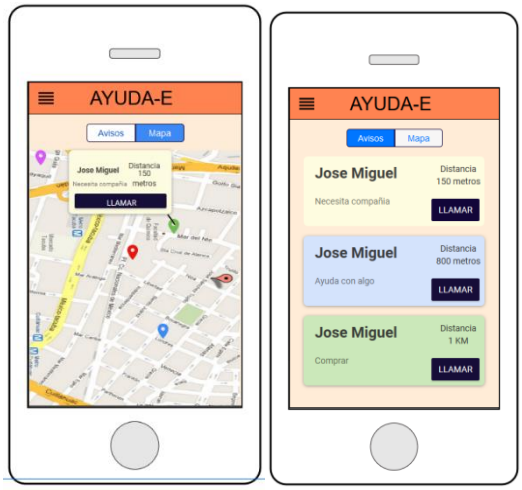
Botón SOS: El foco principal de la app del asistido esta en este botón ya que, si hubiera un tipo de urgencia de nivel alto, realizara una llamada a los servicios de urgencias (112)



- App Asistente:

El foco de la app del asistente está en el mapa, donde el asistido podrá localizar las diferentes alertas que se vayan generando y en consecuencia actuar como vea pertinente.

Podrá observar dos vistas diferentes de las alertas en modo mapa o modo lista.



- El icono de nuestra aplicación también ha sufrido varios cambios a lo largo de todo el desarrollo de la App.

- Quedando finalmente el siguiente:

La primera parte del proyecto se centró en crear las diferentes pantallas que iba a tener la aplicación, ya fueran Activities, Fragments, Dialogs, etc, y la navegabilidad entre ellas, dejando un poco más claro que funcionabilidad queríamos tener.

A continuación, creamos el Login y Registro, ya que son las partes más fáciles de programar en la aplicación.

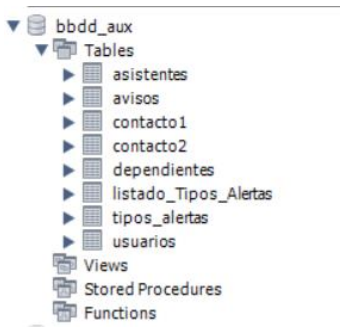
Seguidamente empezamos con las partes más cruciales del proyecto, la implementación del Maps y los diferentes códigos PHP que hacen de pasarela entre la aplicación Android y la base de datos.

```
<?php
//Realizamos la conexion
$mysqli = new mysqli(DATOS DE CONEXION);

$correo=$_GET['correo'];

$consulta ="SELECT id_Email,telefono,nombre,apellidos FROM `usuarios` WHERE
id_Email ='$correo'";
        //Si viene un resultado,son los datos
        $implementar = $mysqli->query($consulta);
    if ($resultado = $implementar->fetch_array(MYSQLI_NUM)) {
        echo json_encode($resultado);
        //si no, insertamos un nuevo usuario
    }else{
        echo " error ";
    }
?>
```

- Respecto a nuestra base de datos, queda con un total de 8 tablas, todas ellas referenciadas con sus claves primarias y foreign Keys.

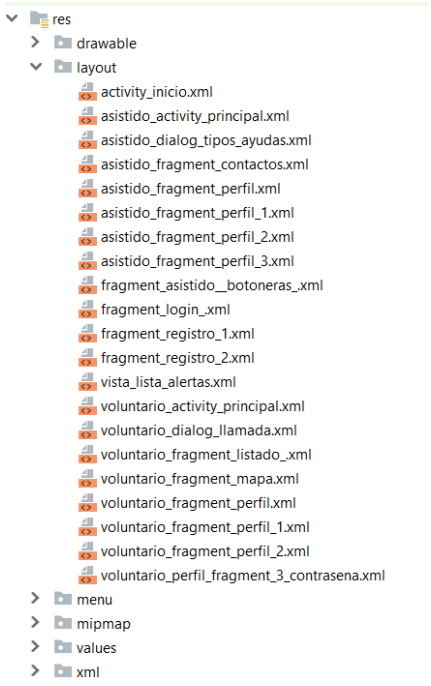
	Tabla	Acción	Filas	Tipo	Cotejamiento	Tamaño	Residuo a depurar
☐	asistentes	Examinar Estructura Buscar Insertar Vaciar Eliminar	2	InnoDB	utf8_spanish2_ci	16 KB	-
☐	avisos	Examinar Estructura Buscar Insertar Vaciar Eliminar	3	InnoDB	utf8mb4_spanish2_ci	64 KB	-
☐	contacto1	Examinar Estructura Buscar Insertar Vaciar Eliminar	2	InnoDB	utf8_spanish2_ci	32 KB	-
☐	contacto2	Examinar Estructura Buscar Insertar Vaciar Eliminar	2	InnoDB	utf8_spanish2_ci	32 KB	-
☐	dependientes	Examinar Estructura Buscar Insertar Vaciar Eliminar	3	InnoDB	utf8_spanish2_ci	16 KB	-
☐	listado_Tipos_Alertas	Examinar Estructura Buscar Insertar Vaciar Eliminar	5	InnoDB	utf8_spanish2_ci	32 KB	-
☐	tipos_alertas	Examinar Estructura Buscar Insertar Vaciar Eliminar	5	InnoDB	utf8_spanish2_ci	16 KB	-
☐	usuarios	Examinar Estructura Buscar Insertar Vaciar Eliminar	5	InnoDB	utf8_spanish2_ci	16 KB	-
	8 tablas	Número de filas	27	InnoDB	latin1_swedish_ci	224 KB	0 B

b Base de datos Vista PhpMyAdmin



c Vista base de datos desde MySQL Workbench

- En cuanto a las distribuciones de nuestras vistas, quedan repartidas y organizadas de la siguiente manera:



d Distribucion de las vistas XML

- La vista de las clases se encuentra en el Anexo

- Aspectos interesantes del código:

- Modelo (Clase AuxinetAPI): Clase encargada de realizar la conexión con la base de datos y devolver la respuesta a cada método invocado.

```
public class AuxinetAPI extends AsyncTask<String, Void, String> {

   String APIUrl = "http://37.187.198.145/llamas/App/";

   private OnResponseListener callBack;
   public Exception exception;

   public AuxinetAPI(OnResponseListener<JSONArray> callBack) {
      this.callBack = callBack;
   }

 @Override
   protected String doInBackground(String... strings) {
      try {
         return downloadUrl(strings[0]);
      } catch (IOException e) {
         exception = e;
      }
      return null;
   }

   @Override
   protected void onPostExecute(String result) {
      if (callBack != null) {
         if (exception == null) {
            try {
               JSONArray response = new JSONArray(result);
               callBack.onSuccess(response);
            } catch (JSONException e) {
               callBack.onFailure(e);
            }
         } else {
            callBack.onFailure(exception);
         }
      }
   }
```

- Interfaz OnResponseListener: Localizamos dos métodos abstractos fundamentales, los cuales serán los encargados de describir el comportamiento de los métodos que interactúan entre el modelo y las clases.

```
public interface OnResponseListener<T> {

    public void onSuccess(T response);
    public void onFailure(Exception e);
}
```

- Dentro del modelo podemos ver cómo interactúan.

```
@Override
protected void onPostExecute(String result) {
    if (callBack != null) {
        if (exception == null) {
            try {
                JSONArray response = new JSONArray(result);
                callBack.onSuccess(response);
            } catch (JSONException e) {
```

```
                callBack.onFailure(e);
            }
        } else {
            callBack.onFailure(exception);
        }
    }
}
```

- Y en la Clase Asistido _Perfil_Fragment_1.java podemos observar cómo se crea un método que interactúa con El modelo y el Interfaz para tratar los datos.

```
public void cargarDatosPerfil() {
        AuxinetAPI auxinetAPI = new AuxinetAPI(new OnResponseListener<JSONArray>() {
            @Override
            public void onSuccess(JSONArray response) {
                try {
                    String apellido = response.getString(3);
                    if (apellido.equals("null")) {
                        et_perfil_apellido.setText("");
                    } else {
                        et_perfil_apellido.setText(response.getString(3));
                    }
                    emailViejo = response.getString(0);
                    et_perfil_email.setText(response.getString(0));
                    et_perfil_telefono.setText(response.getString(1));
                    et_perfil_nombre.setText(response.getString(2));
                } catch (JSONException e) {
                    Toast.makeText(getContext(), "ERROR: " + e.getMessage(),
Toast.LENGTH_SHORT).show();
                }
            }
            @Override
            public void onFailure(Exception e) {
                Toast.makeText(getContext(), "ERROR: " + e.getMessage(),
Toast.LENGTH_SHORT).show();
            }
        });
        auxinetAPI.cargarPerfil(correoUser);
    }
```

Una parte importante de la aplicación es la implementación de la API de Google donde es interesa resaltar los pasos realizados para enlazarla.

- Permisos: son necesarios para poder interactuar con las diferentes funcionalidades que te proporciona la API de GoogleMaps. Ejemplo de permisos de localización:

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.INTERNET" />
```

- Huella digital y Clave: Se necesitará la huella digital para poder utilizar la API de GoogleMaps.

```
<meta-data
   android:name="com.google.android.geo.API_KEY"
   android:value="AIzaSyAxMnx40xmRy26QBYjRQoDwCoM9A2EC0Wk" />
```

- Creación de clase encargada de interactuar con la API de GoogleMaps:

```
public class Voluntario_Mapa_Fragment extends Fragment implements OnMapReadyCallback {

@Override
```

```
public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
    super.onViewCreated(view, savedInstanceState);

    //cargarAlertas();
    mMapView = (MapView) mView.findViewById(R.id.map);
    mMapView.setVisibility(View.INVISIBLE);
    if (mMapView != null) {
        mMapView.onCreate(null);
        mMapView.onResume();
        mMapView.getMapAsync(this);
    }
    btn_voluntarioMapa_cerrar.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            ll_mapa_detalle.setVisibility(View.INVISIBLE);
        }
    });

}
```

- Algunos de los métodos que te permiten crear gracias a la API.
- Posicionamiento de los marcadores a tiempo real.

```
private void posicionAsistidos() {
    Datos_Alertas eAlertas;
    for (int i = 0; i < datos_alertas.size(); i++) {

        eAlertas = datos_alertas.get(i);
        double latitudAsistido1 = eAlertas.getLatitud();
        double longitudAsistido1 = eAlertas.getLongitud();
        String tipoAlerta = eAlertas.getNombreAlerta();
        LatLng posicionAsistido = new LatLng(latitudAsistido1, longitudAsistido1);
        int id_alert = eAlertas.getId_alerta();

        if (tipoAlerta.equals("Aseo")) {

            mGoogleMaps.addMarker(new
MarkerOptions().position(posicionAsistido).icon(BitmapDescriptorFactory.fromResource(R.drawable.botiquin_de_primeros_auxilios))).setTag(i);
        }
```

Se pueden observar las partes más importantes de las llamadas a la base de datos en el anexo

2.4. Resultados y validación

La aplicación contiene un registro fácil y rápido, donde se necesita la inclusión del número de teléfono (ya que es el medio de contacto entre las personas de la aplicación).

Una vez registrado y logueado, la aplicación internamente se divide en 2 partes: una para el asistido y otra para el asistente o voluntario (la interfaz principal cambia dependiendo de quien realice el inicio).

La interfaz principal del asistido muestra una serie de botones de ayuda, dos contactos rápidos (a los que llamar simplemente pulsando en ellos) y un botón deslizante de ayuda que le pone en contacto con el 112.

Después tiene su perfil, donde puede realizar cambios en su correo y demás datos de inicio y por último 2 apartados: uno donde configura los teléfonos de los contactos rápidos y otro donde se encuentra una información no obligatoria sobre una serie de datos médicos (que pueden servir de ayuda

cuando los vea un asistente o voluntario) que puede rellenar o modificar si quiere.

En la parte del asistente o voluntario, su interfaz principal es un mapa, donde aparecerán las alertas que hayan podido solicitar los asistidos (esta carga es un poco más lenta que otras, ya que tiene que cargar el mapa con los diferentes marcadores, si hay mucho se nota un poco de diferencia a la hora de cargar). Si clica sobre alguno de los marcadores, le dará el nombre del asistido, así como el tipo de ayuda que necesita y la distancia a la que se encuentra de él y 2 botones, uno para la navegación y otro para poder llamarle.

También tiene otro apartado con una lista donde aparecerán las alertas y si clica en ellas le ofrece los datos que tenga puesto la persona dependiente.

Dispone de un perfil para gestionar el inicio de sesión y sus datos personales.

Los problemas que nos encontramos en la aplicación son debidos a los métodos que utilizamos para poder conectarnos a la base de datos. Hemos tenido que utilizar Hilos, ya que, en algunas ocasiones, la aplicación se cerraba porque hacíamos un cambio de fragmento y no había terminado de devolver datos, por lo que, al realizar otra nueva petición, se encontraba ocupado el OnPostExecute.

También con el manejo del Maps tenemos algunos problemas, debido que al meter los marcadores, estos dependen de los datos que devuelva la base de datos, si tarda mucho la conexión puede provocar que la aplicación se cierre.

Hemos realizado un test completo a la aplicación (problemas de credenciales al iniciar sesión, fallos de registro, campos vacíos, devolución de valores de la base de datos, concordancia de datos…) y hemos solventado la gran mayoría a día de hoy, faltando por dar las revisiones siguientes.

Algunos errores que nos hemos encontrado han sido con los campos nulos, ya que android no es capaz de manejarlos de forma predeterminada como si hace Java de Eclipe(.null), por tanto hemos tenido que cambiar la forma que inserta campos vacíos y cómo los trata cuando regresa a la aplicación.

3. Conclusiones

El trabajo ha supuesto un gran esfuerzo por parte de los 3 integrantes, ya que compaginarlo con las prácticas, que un compañero se encuentre fuera de España y el poco tiempo que se disponía para realizar el proyecto, nos ha llevado a tener que quedar todas las tardes y todos los sábados para poder llevar el proyecto a buen puerto.

Nuestros hitos más importantes han sido, aparte del ya mencionado problema de distancia entre los participantes, la inclusión de un nuevo lenguaje de programación (PHP) y la utilización del Maps y sus diferentes métodos.

Los problemas más grandes que nos encontramos fueron 2, la utilización de PHP como pasarela entre la base de datos y la aplicación, y el Maps.

Con PHP, porque no conocíamos cómo funcionaba para realizar peticiones a un servidor, ni cómo devolver después esos valores a la aplicación Android. Solo el estudio de este lenguaje nos llevo unas semanas sin parar, ya que tampoco encontrábamos la forma de que Android se comunicara con el documento PHP.

Respecto al Maps, se consiguió alcanzar lo propuesto en un principio: localización de la posición, guardarla en la base de datos, colocar marcadores personalizados, calcular la distancia entre los puntos y realizar la ruta entre un punto y la ubicación del asistente.

Tanto para el PHP como para el Maps, se debe investigar mucho más los métodos utilizados, ya que no son los más óptimos (tardan mucho en cargar o si viene algún dato vacío falla la aplicación) aunque en un principio funcionen. También tenemos problemas con la distancia entre los puntos, ya que se utilizan métodos deprecados que no funcionan adecuadamente según la versión del dispositivo móvil que se utilice.

La incorporación de estos nuevos métodos de trabajo nos supuso un traspiés al inicio del proyecto, ya que nos ralentizaba mucho el poder avanzar hasta que no tuviésemos los conocimientos necesarios para poder continuar.

La aplicación al final solo consta de una única aplicación, ya que en un principio queríamos desarrollar 2 aplicaciones diferenciadas, una para el asistido y otra para el asistente o voluntario, quedando finalmente en una única aplicación, con el Login y Registro en común, pero que luego difiere en el contenido de la misma.

3.1. Innovación

La aplicación presenta un carácter diferenciador del resto de aplicaciones que se aportan como trabajos de final de curso ya que se ha interactuado con diferentes sistemas y lenguajes de programación propuesto para el proyecto.
Ha sido interesante a la vez que complicado tener que ir formándose a la vez que se realizaba la aplicación.

Uno de los puntos más complejos fue enlazar la BBDD con la App, en la cual se utilizó una pasarela (BackEnd) con "PHP" para poder manejar los datos provenientes de la BBDD hasta la app ya que se usaba un servidor remoto donde estaba integrada la BBDD relacional MySql.

La implementación de la Api de GoogleMaps es algo que a nuestro parecer le da un toque diferente a la app al integrar un mapa donde poder observar las diferentes localizaciones de los usuarios y poder realizar rutas de navegación entre diferentes puntos.

3.2. Trabajo futuro

Queremos seguir indagando en el apartado de Google Maps, ya que vemos que no hemos conseguido explotar tanto como quisiéramos ni de la forma que quisiéramos esta API de Google. Ya que implementamos métodos que están deprecados dentro de la aplicación y que podrían llevar a problemas en un futuro.

En el apartado de Base de datos, queremos dedicar tiempo a la investigación entre una conexión a una base de datos externa que no sea Firebase, ya que utilizamos unos métodos que no nos terminan de convencer, porque no nos parecen apropiados y nos han dado más problemas de lo que esperábamos.

Nuestra intención es seguir desarrollando aplicaciones Android y utilizar las diversas tecnologías que tiene, puesto que, en el mundo laboral, no utilizar algo que has aprendido al especializarte en otras cosas, supone el olvido de estas tecnologías.

4. BIBLIOGRAFÍA Y WEBGRAFÍA

Mucha de la documentación proviene de las mismas páginas web, como por ejemplo:

- GITHUB. (2008) https://github.com. Fecha de consulta: 17:33, abril 7, 2018 de https://github.com/FaisalUmair/udemy-downloader-gui/blob/master/README.md
- GOOGLE CLOUD. https://cloud.google.com. Fecha de consulta: 12:02, abril 14, 2018 de Solicitud de api para google maps
- ANDROID DEVELOPER. https://developer.android.com Fecha de consulta: 19:53, abril 11, 2018 de Permisos de la aplicación
- StackOverflow. (2008) https://stackoverflow.com Fecha de consulta: 13:20, abril 7, 2018 de GetRequest desde android a php
- ACADEMIAANDROID (2005) https://academiaandroid.com. Fecha de consulta: 18:45, abril 18, 2018 de Base de datos externa
- CODEPATH, https://guides.codepath.com/android. Fecha de consulta: 11:55, abril 21, 2018 de Libreria
- CODE. https://code.tutsplus.com. Fecha de consulta: 19:22, mayo 30, 2018 de tutoriales

- MySQL. https://dev.mysql.com. Fecha de consulta: 10:46, abril 9, 2018 de manuales
- PHP. http://php.net. Fecha de consulta: 13:21, mayo 12, 2018 de manuales de uso
- GOOGLE MAPS. https://support.google.com/maps. Fecha de consulta: 20:08, abril 25, 2018 de Maps
- MATERIAL IO. https://material.io/. Fecha de consulta: 18:12, mayo 20, 2018 de Diseño

5. ANEXOS

- Como parte importante, agregamos todos los métodos de acceso a la base de datos, que los tenemos en una única clase, que implementa una interfaz que devuelve los resultados al método que la llame.

```
import android.os.AsyncTask;
import org.json.JSONArray;
import org.json.JSONException;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.Reader;
import java.io.UnsupportedEncodingException;
import java.net.HttpURLConnection;
import java.net.URL;

public class AuxinetAPI extends AsyncTask<String, Void, String> {

    String APIUrl = "direccion";

    private OnResponseListener callBack;
    public Exception exception;

    public AuxinetAPI(OnResponseListener<JSONArray> callBack) {
        this.callBack = callBack;
    }

    @Override
    protected String doInBackground(String... strings) {
        try {
            return downloadUrl(strings[0]);
        } catch (IOException e) {
            exception = e;
        }
        return null;
    }

    @Override
    protected void onPostExecute(String result) {
        if (callBack != null) {
            if (exception == null) {
                try {
                    JSONArray response = new JSONArray(result);
                    callBack.onSuccess(response);
                } catch (JSONException e) {
                    callBack.onFailure(e);
                }
            } else {
                callBack.onFailure(exception);
            }
        }
    }

    private String downloadUrl(String myurl) throws IOException {
        myurl = myurl.replace(" ", "%20");
        InputStream is = null;
        int len = 500;

        try {
            URL url = new URL(myurl);
            HttpURLConnection conn = (HttpURLConnection) url.openConnection();
            conn.setReadTimeout(10000 /* milliseconds */);
            conn.setConnectTimeout(15000 /* milliseconds */);
            conn.setRequestMethod("GET");
            conn.setDoInput(true);
            conn.connect();
            int response = conn.getResponseCode();
            is = conn.getInputStream();

            // Convert the InputStream into a string
            String contentAsString = readIt(is, len);
            return contentAsString;
        } finally {
            if (is != null) {
```

```
            is.close();
        }
    }
}

public String readIt(InputStream stream, int len) throws IOException, UnsupportedEncodingException {
    Reader reader = null;
    reader = new InputStreamReader(stream, "UTF-8");
    char[] buffer = new char[len];
    reader.read(buffer);
    return new String(buffer);
}

public void nuevaAlerta(String usuario, String tipoAlerta, String latitud, String longitud) {
    String metodo = "GenerarAlertasApp.php?";
    String parametros = "correo=" + usuario + "&nombreAlerta=" + tipoAlerta + "&latitud=" + latitud + "&longitud=" + longitud;
    this.execute(APIUrl + metodo + parametros);
}

public void cargarPerfil(String usuario) {
    String metodo = "DatosPerfilApp.php?";
    String parametros = "correo=" + usuario;
    this.execute(APIUrl + metodo + parametros);
}

public void actualizarPerfil(String emailViejo, String nombre, String telefono, String email, String apellido) {
    String metodo = "ActualizarPerfilApp.php?";
    String parametros = "correoV=" + emailViejo + "&nombre=" + nombre + "&telefono=" + telefono + "&correoN=" + email + "&apellido=" + apellido;
    this.execute(APIUrl + metodo + parametros);
}

public void cargarContactos(String usuario,String contacto){
    String parametros = "correo=" + usuario;
    if (contacto.equals("contacto1")){
        String metodo = "CargarContacto1App.php?";
        this.execute(APIUrl + metodo + parametros);
    }else{
        String metodo = "CargarContacto2App.php?";
        this.execute(APIUrl + metodo + parametros);
    }
}

public void guardarContactos(String usuario, String contacto,String nombre,String telefono){
    String parametros = "correo=" + usuario+"&nombre="+nombre
            +"&telefono="+telefono;
    if (contacto.equals("contacto1")){
        String metodo = "ActualizarContacto1App.php?";
        this.execute(APIUrl + metodo + parametros);
    }else{
        String metodo = "ActualizarContacto2App.php?";
        this.execute(APIUrl + metodo + parametros);
    }
}

public void loguearUsuario(String usuario,String password){
    String metodo = "LoginApp.php?";
    String parametros = "correo=" + usuario + "&password=" + password;
    this.execute(APIUrl + metodo + parametros);
}

public void registrarUsuario(String usuario,String password,String nombre,String telefono,String tipoUsuario){
    String metodo = "RegistroApp.php?";
    String parametros = "correo="+usuario+"&password="+password+"&nombre="+nombre+"&telefono="+telefono+"&usuario="+tipoUsuario;
    this.execute(APIUrl + metodo + parametros);
}

public void cargarDatosMedicos(String usuario){
    String metodo = "DatosMedicosDependienteApp.php?";
    String parametros = "correo="+usuario;
    this.execute(APIUrl + metodo + parametros);
}

public void actualizarDatosMedicos(String correo,String peso,String altura,String gpSanguineo,String alergias,String medicacion,String nMedicas,String
enfermedades){
    String metodo = "ActualizarDatosMedicosApp.php?";
    String parametros = "correo="+correo+"&peso="+peso+"&altura="+altura+"&grSanguineo="
            +gpSanguineo+"&alergias="+alergias+"&medicacion="+medicacion+"&notasMedicas="
            +nMedicas+"&enfermedades="+enfermedades;
    this.execute(APIUrl + metodo + parametros);
}
```

```
    public void cargarContrasena(String usuario){
        String metodo = "CargarContrasenaApp.php?";
        String parametros = "correo="+usuario;
        this.execute(APIUrl + metodo + parametros);
    }

    public void modificarContrasena(String usuario, String password){
        String metodo = "ModificarContrasenaApp.php?";
        String parametros = "correo="+usuario+"&password="+password;
    }

}
```

- También incluimos el ejemplo de una clase que llama a uno de estos métodos.

```
package com.vpfc18.vpfc18.Principal.Asistido.Perfil;

import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentTransaction;
import android.util.Patterns;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.CompoundButton;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.ToggleButton;
import com.vpfc18.vpfc18.Base_de_datos.AuxinetAPI;
import com.vpfc18.vpfc18.Base_de_datos.OnResponseListener;
import com.vpfc18.vpfc18.R;

import org.json.JSONArray;
import org.json.JSONException;

/**
 * A simple {@link Fragment} subclass.
 */
public class Asistido_Perfil_Fragment_1 extends Fragment {


    EditText et_perfil_email, et_perfil_telefono, et_perfil_nombre, et_perfil_apellido, et_perfil_fnacimiento, et_perfil_sexo;
    Button btn_perfil_cerrarSesion;
    TextView tv_perfil_modContrasena, tv_perfil_datosMedicos, tv_perfil_contactos;
    ToggleButton btn_perfil_modificar_datos;

    String email, emailViejo, nombre, apellido, telefono, correoUser, sexo, fNacimiento;

    public Asistido_Perfil_Fragment_1() {
        // Required empty public constructor
    }


    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {

        View vista = inflater.inflate(R.layout.asistido_fragment_perfil_1, container, false);

        et_perfil_email = (EditText) vista.findViewById(R.id.et_perfil_email);
        et_perfil_telefono = (EditText) vista.findViewById(R.id.et_perfil_telefono);
        et_perfil_nombre = (EditText) vista.findViewById(R.id.et_perfil_nombre);
        et_perfil_apellido = (EditText) vista.findViewById(R.id.et_perfil_apellido);
        et_perfil_sexo = (EditText) vista.findViewById(R.id.et_perfil_sexo);
        et_perfil_fnacimiento = (EditText) vista.findViewById(R.id.et_perfil_fnacimiento);
        tv_perfil_modContrasena = (TextView) vista.findViewById(R.id.tv_perfil_modContrasena);
        tv_perfil_datosMedicos = (TextView) vista.findViewById(R.id.tv_perfil_datosMedicos);
        tv_perfil_contactos = (TextView) vista.findViewById(R.id.tv_perfil_contactos);
        btn_perfil_modificar_datos = (ToggleButton) vista.findViewById(R.id.btn_perfil_modificar_datos);
        btn_perfil_cerrarSesion = (Button) vista.findViewById(R.id.btn_perfil_cerrarSesion);
        correoUser = getArguments().getString("correoUser");

        btn_perfil_modificar_datos.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
```

```
            if (isChecked) {
                habilitarCampos(true);
            } else if (comprobarCampos()) {
                habilitarCampos(false);
                actualizarDatosPerfil();
            }
        }
    });
    tv_perfil_modContrasena.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            modificarContrasena();
        }
    });
    tv_perfil_datosMedicos.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            vistaDatosMedicos();
        }
    });
    tv_perfil_contactos.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            vistaModificarContactos();
        }
    });
    cargarDatosPerfil();
    return vista;

}

private void habilitarCampos(Boolean habilitado) {
    et_perfil_nombre.setEnabled(habilitado);
    et_perfil_apellido.setEnabled(habilitado);
    et_perfil_fnacimiento.setEnabled(habilitado);
    et_perfil_sexo.setEnabled(habilitado);
    et_perfil_telefono.setEnabled(habilitado);
    et_perfil_email.setEnabled(habilitado);

}

private boolean comprobarCampos() {
    email = et_perfil_email.getText().toString().trim();
    telefono = et_perfil_telefono.getText().toString();
    nombre = et_perfil_nombre.getText().toString();
    apellido = et_perfil_apellido.getText().toString();
    sexo = et_perfil_sexo.getText().toString();
    fNacimiento = et_perfil_fnacimiento.getText().toString();

    if (!Patterns.EMAIL_ADDRESS.matcher(email).matches()) {
        String a = "Escribe un correo válido";
        Toast.makeText(getContext(), a, Toast.LENGTH_LONG).show();
        et_perfil_email.requestFocus();
        return false;
    }
    if (telefono.isEmpty()) {
        String a = "No puedes dejar el campo teléfono vacio";
        Toast.makeText(getContext(), a, Toast.LENGTH_LONG).show();
        et_perfil_telefono.requestFocus();
        return false;
    }
    if (nombre.isEmpty()) {
        String a = "No puedes dejar el campo nombre vacio";
        Toast.makeText(getContext(), a, Toast.LENGTH_LONG).show();
        et_perfil_nombre.requestFocus();
        return false;
    }
    if (apellido.isEmpty()) {
        apellido = "null";
    }
    return true;
}

private String devolverCorreo() {
    if (emailViejo.equals(email)) {
        correoUser = email;
    } else {
        correoUser = emailViejo;
    }
    return correoUser;
```

```
}

public void cargarDatosPerfil() {
    AuxinetAPI auxinetAPI = new AuxinetAPI(new OnResponseListener<JSONArray>() {
        @Override
        public void onSuccess(JSONArray response) {
            try {
                String apellido = response.getString(3);
                if (apellido.equals("null")) {
                    et_perfil_apellido.setText("");
                } else {
                    et_perfil_apellido.setText(response.getString(3));
                }
                emailViejo = response.getString(0);
                et_perfil_email.setText(response.getString(0));
                et_perfil_telefono.setText(response.getString(1));
                et_perfil_nombre.setText(response.getString(2));
            } catch (JSONException e) {
                Toast.makeText(getContext(), "ERROR: " + e.getMessage(), Toast.LENGTH_SHORT).show();
            }
        }
        @Override
        public void onFailure(Exception e) {
            Toast.makeText(getContext(), "ERROR: " + e.getMessage(), Toast.LENGTH_SHORT).show();
        }
    });
    auxinetAPI.cargarPerfil(correoUser);
}

public void actualizarDatosPerfil() {
    AuxinetAPI auxinetAPI = new AuxinetAPI(new OnResponseListener<JSONArray>() {
        @Override
        public void onSuccess(JSONArray response) {
            try {
                String apellido = response.getString(3);
                if (apellido.equals("null")) {
                    et_perfil_apellido.setText("");
                } else {
                    et_perfil_apellido.setText(response.getString(3));
                }
                emailViejo = response.getString(0);
                et_perfil_email.setText(response.getString(0));
                et_perfil_telefono.setText(response.getString(1));
                et_perfil_nombre.setText(response.getString(2));
            } catch (JSONException e) {
                Toast.makeText(getContext(), "ERROR: " + e.getMessage(), Toast.LENGTH_SHORT).show();
            }
        }
        @Override
        public void onFailure(Exception e) {
            Toast.makeText(getContext(), "ERROR: " + e.getMessage(), Toast.LENGTH_SHORT).show();
        }
    });
    auxinetAPI.actualizarPerfil(emailViejo, nombre, telefono, email, apellido);
}

private void vistaDatosMedicos() {
    Fragment fragmentoSeleccionado = new Asistido_Perfil_Fragment_2_datosMedicos();
    FragmentTransaction t = getFragmentManager().beginTransaction();
    t.replace(R.id.contenedor_perfil_asistido, fragmentoSeleccionado);
    t.commit();
    correoUser = devolverCorreo();
    Bundle datos = new Bundle();
    datos.putString("correoUser", correoUser);
    fragmentoSeleccionado.setArguments(datos);
}

private void modificarContrasena() {
    Fragment fragmentoSeleccionado = new Asistido_Perfil_Fragment_3_contrasena();
    FragmentTransaction t = getFragmentManager().beginTransaction();
    t.replace(R.id.contenedor_perfil_asistido, fragmentoSeleccionado);
    t.commit();
    correoUser = devolverCorreo();
    Bundle datos = new Bundle();
    datos.putString("correoUser", correoUser);
    fragmentoSeleccionado.setArguments(datos);
}

private void vistaModificarContactos() {
    Fragment fragmentoSeleccionado = new Asistido_Perfil_Fragment_4_contactos();
```

```
        FragmentTransaction t = getFragmentManager().beginTransaction();
        t.replace(R.id.contenedor_perfil_asistido, fragmentoSeleccionado);
        t.commit();
        correoUser = devolverCorreo();
        Bundle datos = new Bundle();
        datos.putString("correoUser", correoUser);
        fragmentoSeleccionado.setArguments(datos);
    }
}
```

- Por último, pero no menos importante, agregamos nuestra distribución de las clases java.