
Escuela Superior de Ingeniería

Programación en Internet

GRADO EN INGENIERÍA INFORMÁTICA

# MILF Android APP

Autores:
Luis Miguel Carpio Gavira
Sergio García Navarro

Cádiz, 10 de mayo de 2020

# Índice de Contenidos

**1. Capítulo 1** **3**
1.1. Introducción . . . . . 3

**2. Capítulo 2** **4**
2.1. Prototipado . . . . . 4
2.1.1. Prototipo de la aplicación . . . . . 4

**3. Capítulo 3** **20**
3.1. Creación de la aplicación . . . . . 20
3.1.1. Primeros pasos . . . . . 20
3.1.2. Get all . . . . . 21
3.1.3. Get one . . . . . 21
3.1.4. Put . . . . . 22
3.1.5. Delete one . . . . . 22
3.1.6. Post . . . . . 23

**Bibliografía** **24**

# Índice de figuras

2.1. Widget integrado en la página principal . . . . . 5
2.2. Pantalla principal de la aplicación . . . . . 6
2.3. Menú en la pantalla principal . . . . . 7
2.4. Vista de los asistentes del evento . . . . . 8
2.5. Resultado de pulsar Añadir Asistente . . . . . 9
2.6. Resultado de añadir un asistente . . . . . 10
2.7. Resultado de pulsar en Detalles sobre un asistente . . . . . 11
2.8. Resultado de modificar un asistente . . . . . 12
2.9. Resultado de eliminar un asistente . . . . . 13
2.10. Pestaña de fechas importantes . . . . . 14
2.11. Resultado de pulsar una fecha pasada . . . . . 15
2.12. Resultado de pulsar una fecha próxima . . . . . 16
2.13. Resultado de pulsar la fecha del evento del día de su celebración . . . . . 17
2.14. Vista de localización . . . . . 18
2.15. Vista de información . . . . . 19

3.1. Dependencia de OkHttp . . . . . 20
3.2. Petición get all . . . . . 21
3.3. Petición get one . . . . . 21
3.4. Petición put . . . . . 22
3.5. Petición delete one . . . . . 22
3.6. Petición post . . . . . 23

# Capítulo 1

## 1.1. Introducción

En este documento se incluye un tutorial que explica paso a paso con capturas de pantalla y pasos detallados la creación de una aplicación android que nos ayude a cumplir los requisitos funcionales del hito 4. Para ello, este documento incluirá un prototipo que ha sido diseñado con el fin de hacernos una idea clara de la estructuración de la información y de la funcionalidad de la aplicación, así como el uso de OkHttp para facilitarnos la implementación de las peticiones HTTP.

# Capítulo 2

## 2.1. Prototipado

### 2.1.1. Prototipo de la aplicación

Previamente a desarrollar algo de la aplicación, hemos decidido diseñar un prototipo (wireframe) para estructurar los requisitos funcionales que el enunciado especifica. Para ello, hemos usado el logotipo que realizamos para la página web del evento, este aparecerá en la pantalla principal, con un menú en la parte superior haciendo uso de la escala de colores que seleccionamos para el sitio web.

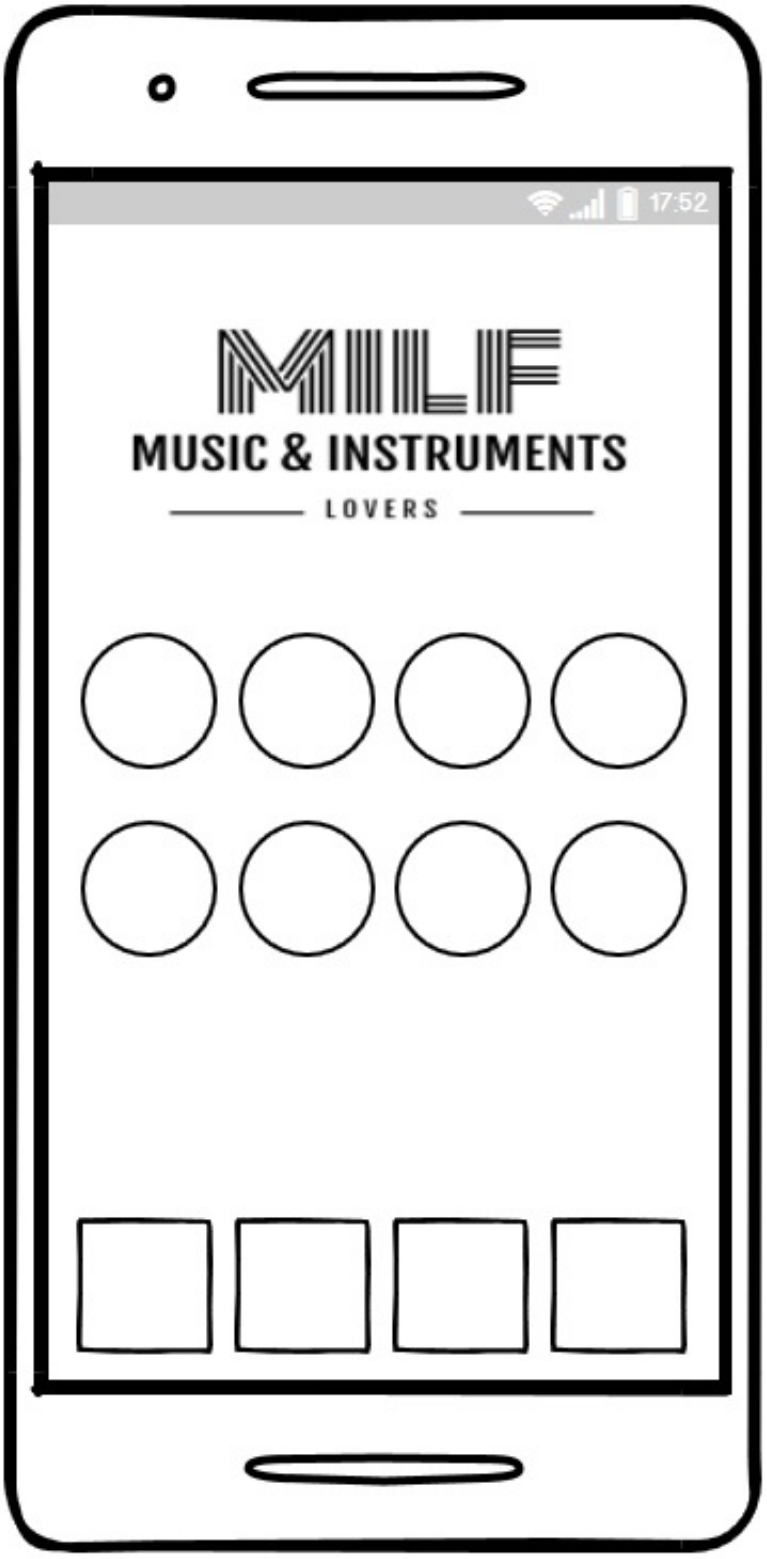


Figura 2.1: Widget integrado en la página principal


Figura 2.2: Pantalla principal de la aplicación


Figura 2.3: Menú en la pantalla principal

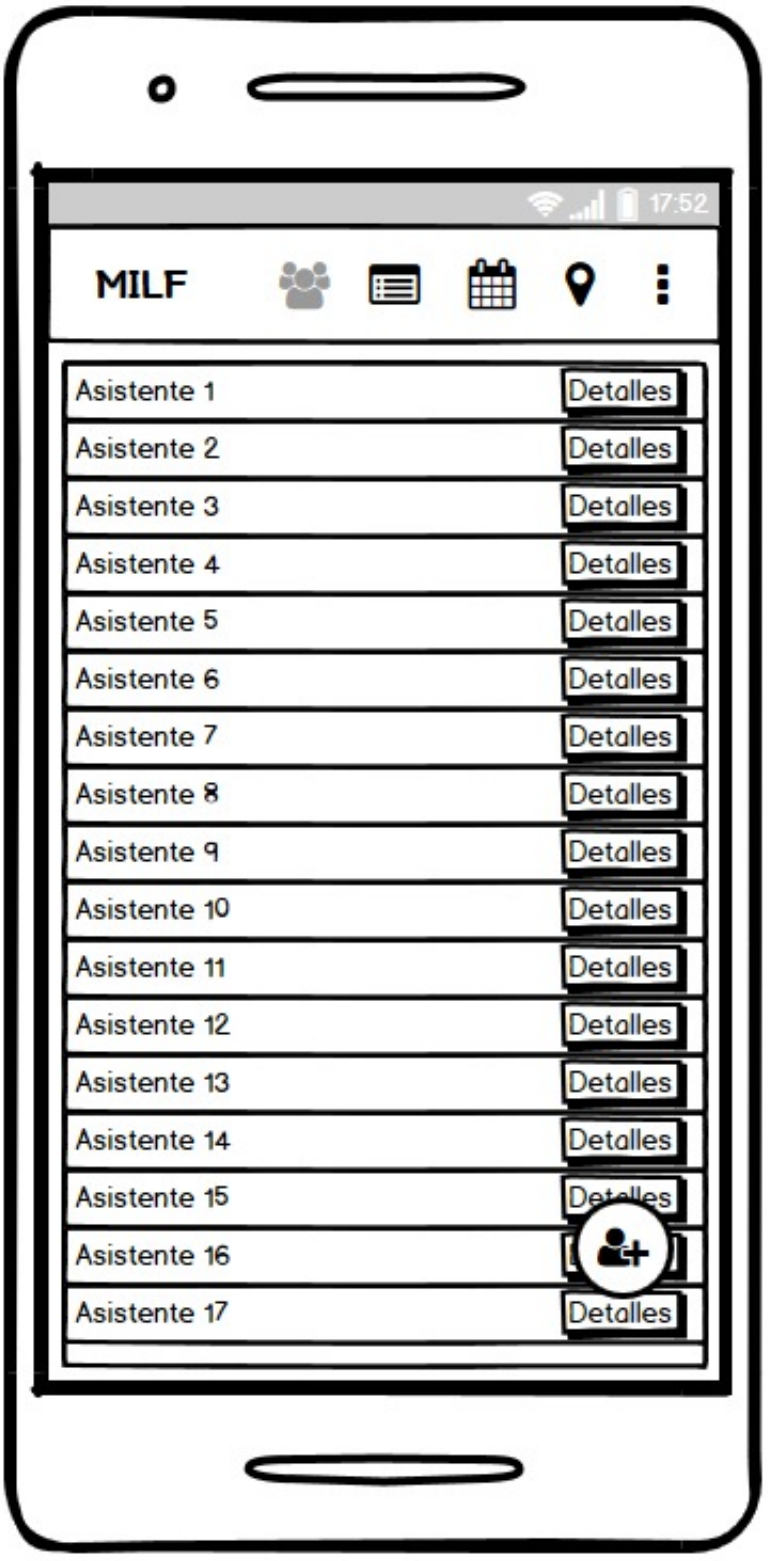


Figura 2.4: Vista de los asistentes del evento

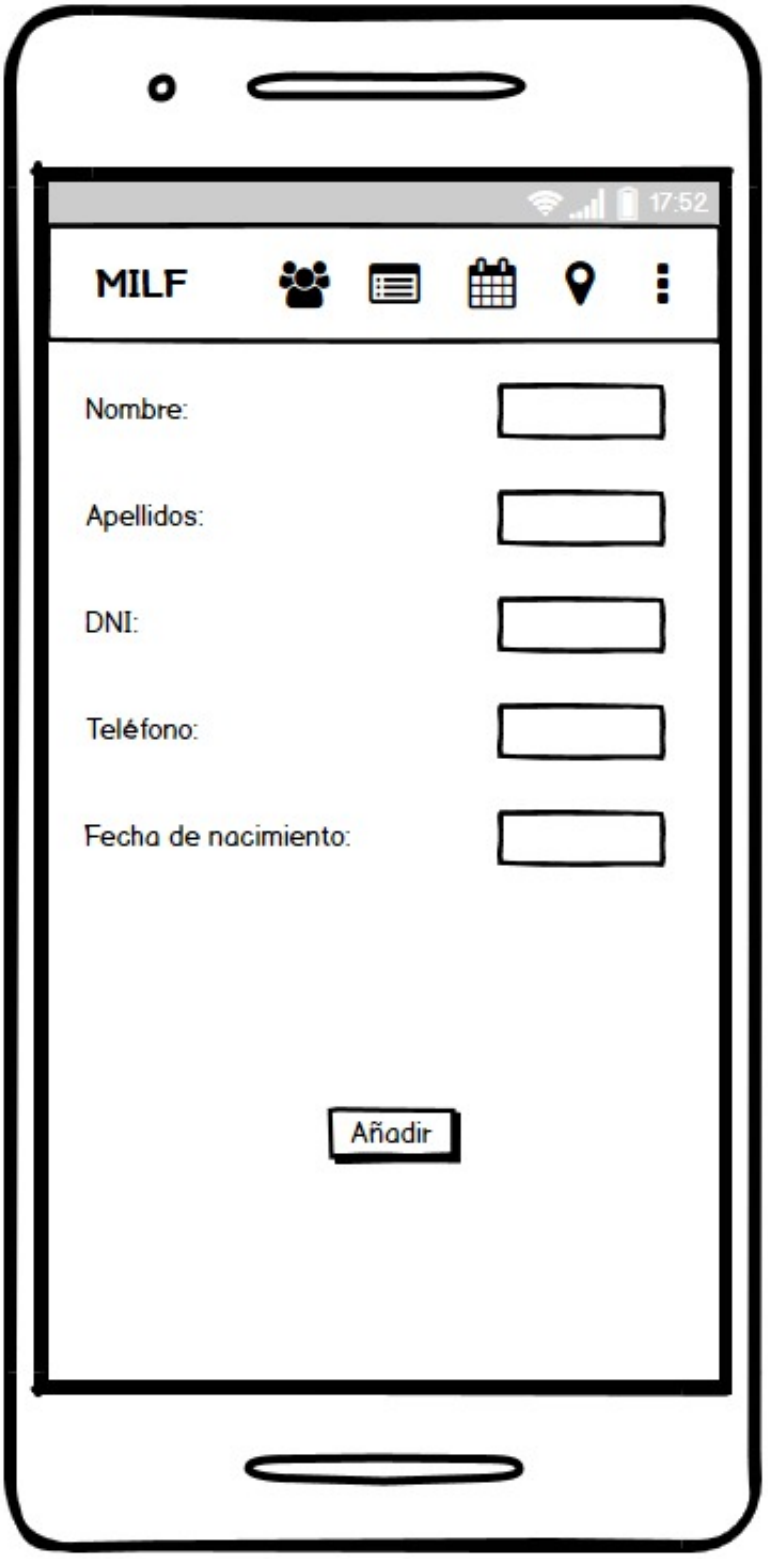


Figura 2.5: Resultado de pulsar Añadir Asistente

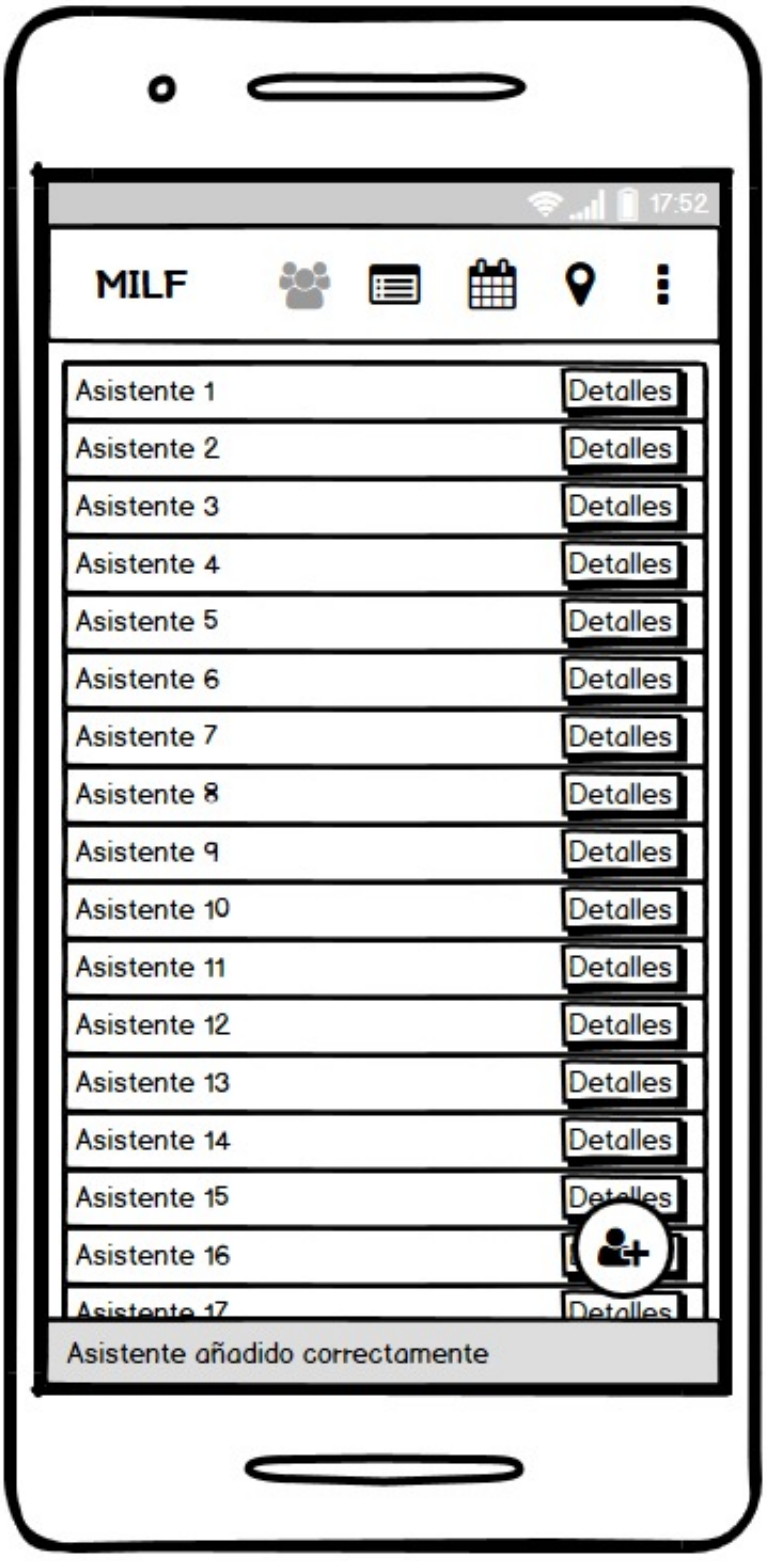


Figura 2.6: Resultado de añadir un asistente


Figura 2.7: Resultado de pulsar en Detalles sobre un asistente

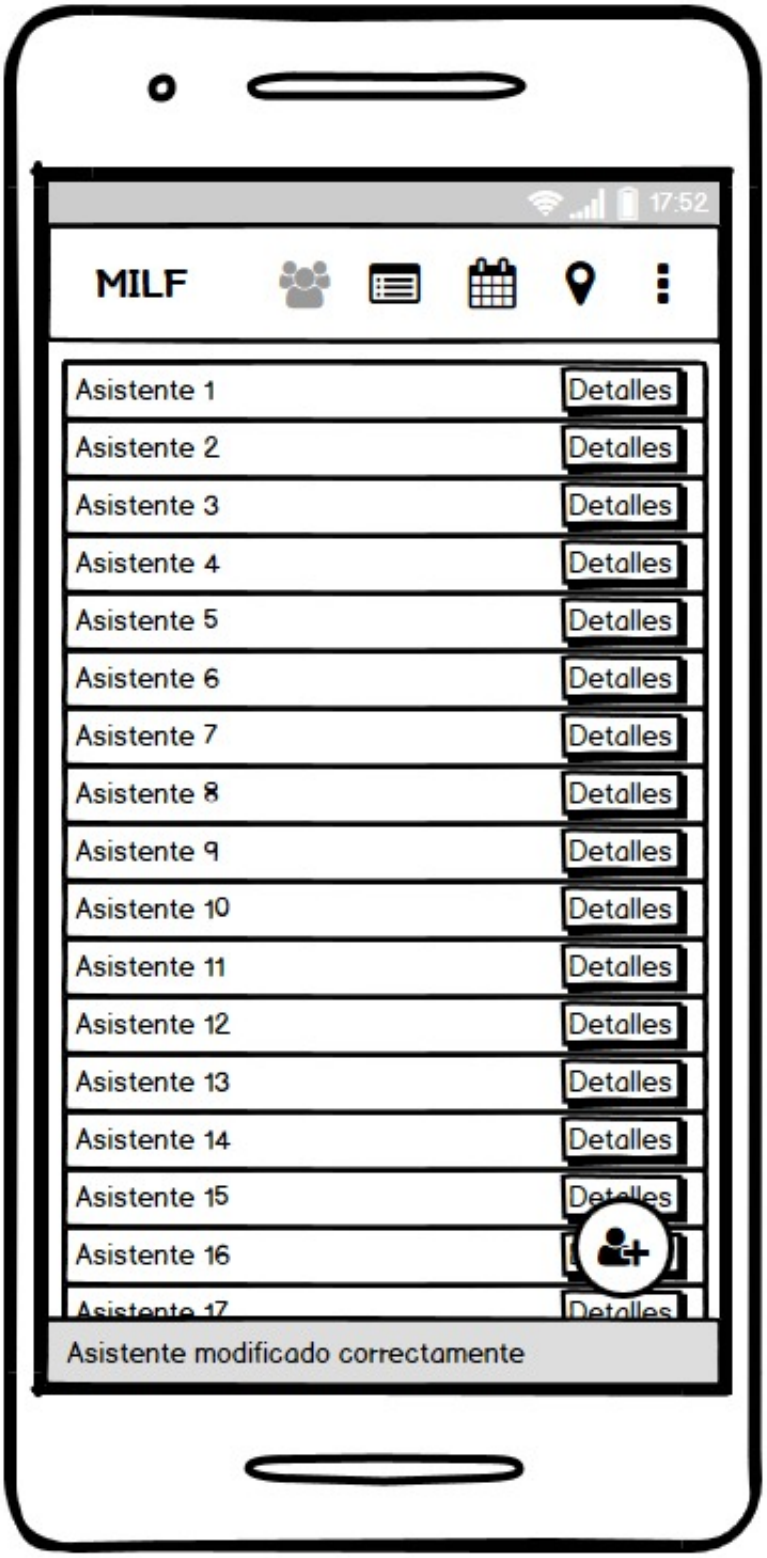


Figura 2.8: Resultado de modificar un asistente

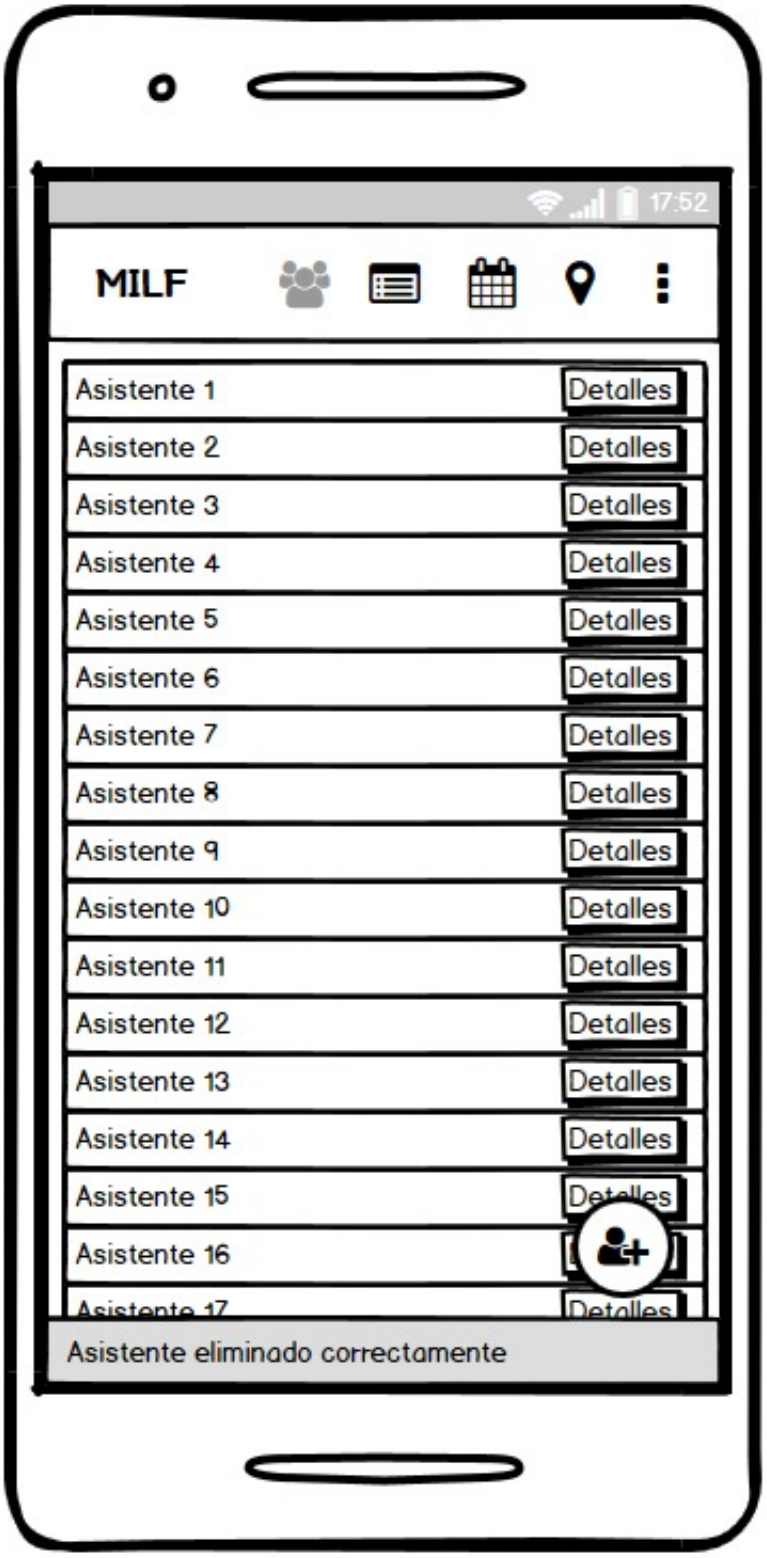


Figura 2.9: Resultado de eliminar un asistente

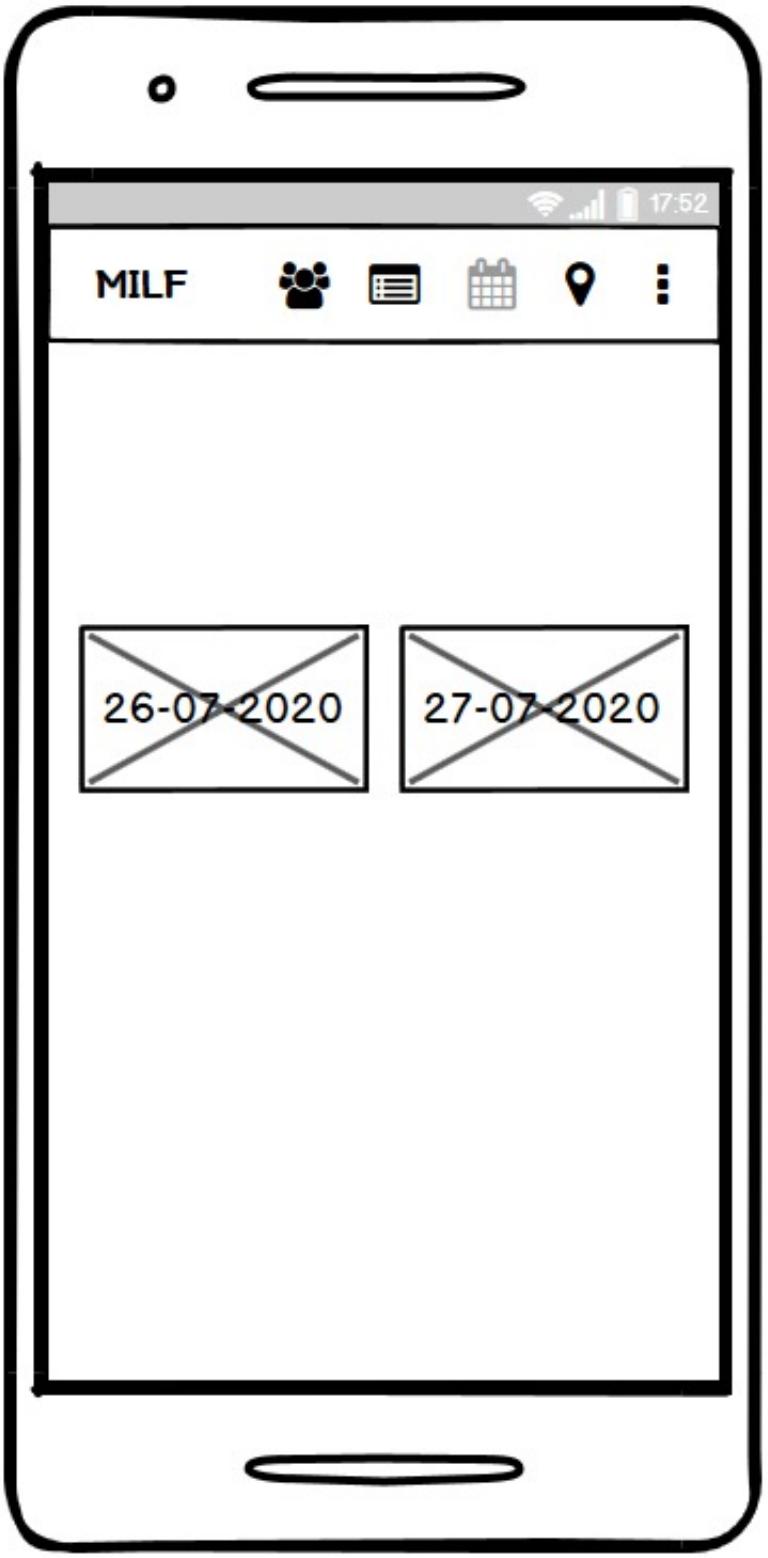


Figura 2.10: Pestaña de fechas importantes

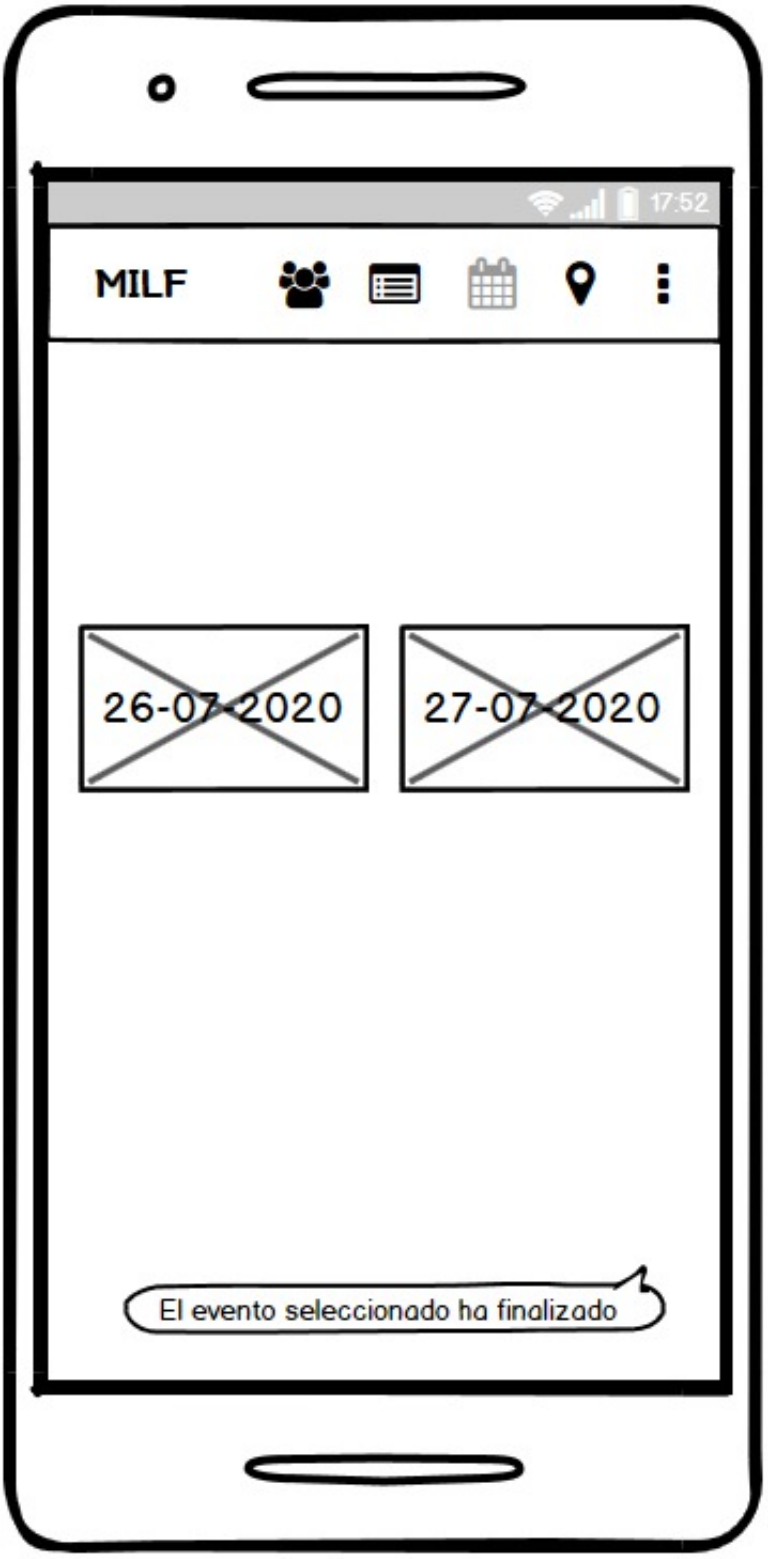


Figura 2.11: Resultado de pulsar una fecha pasada

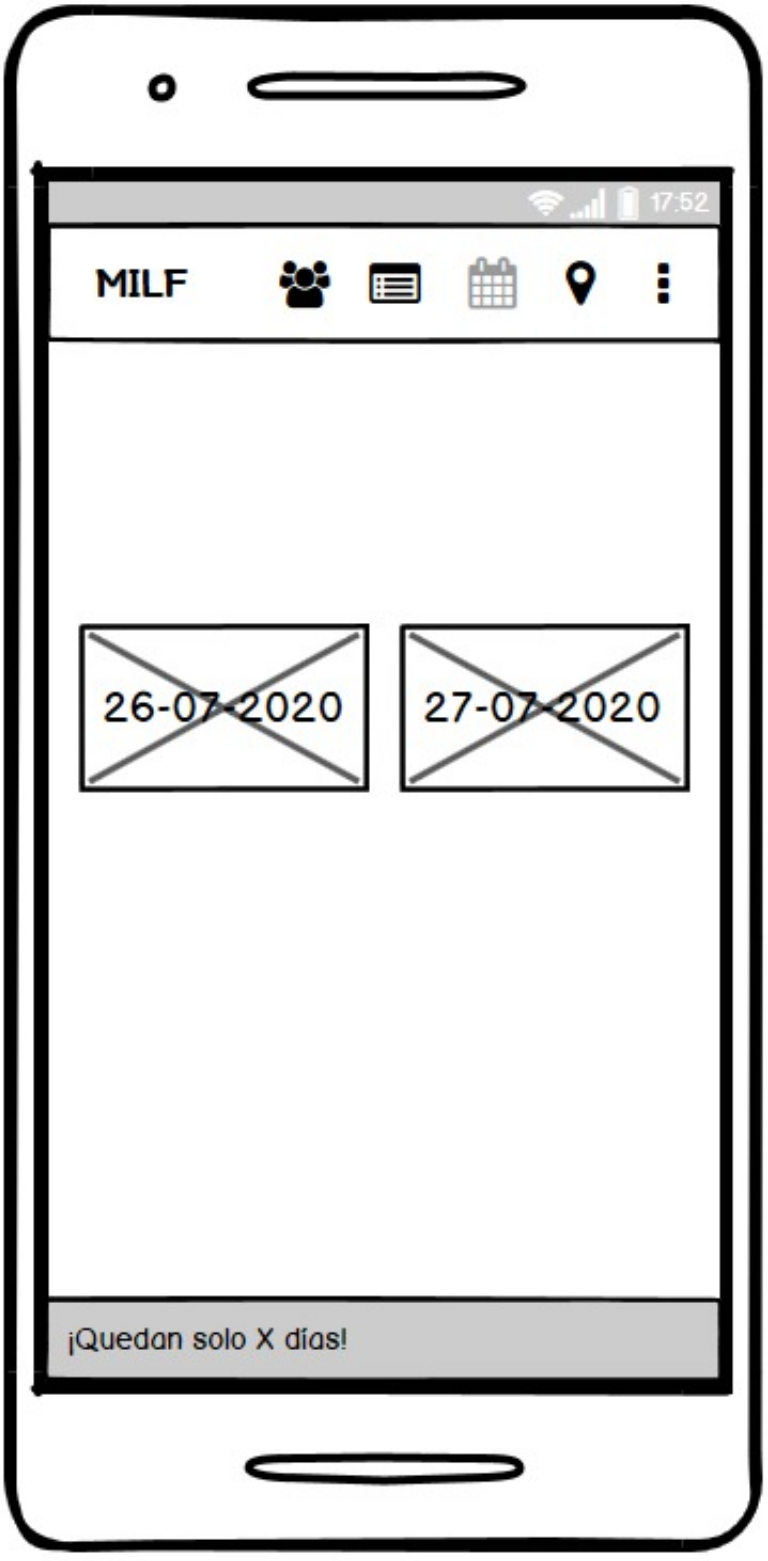


Figura 2.12: Resultado de pulsar una fecha próxima

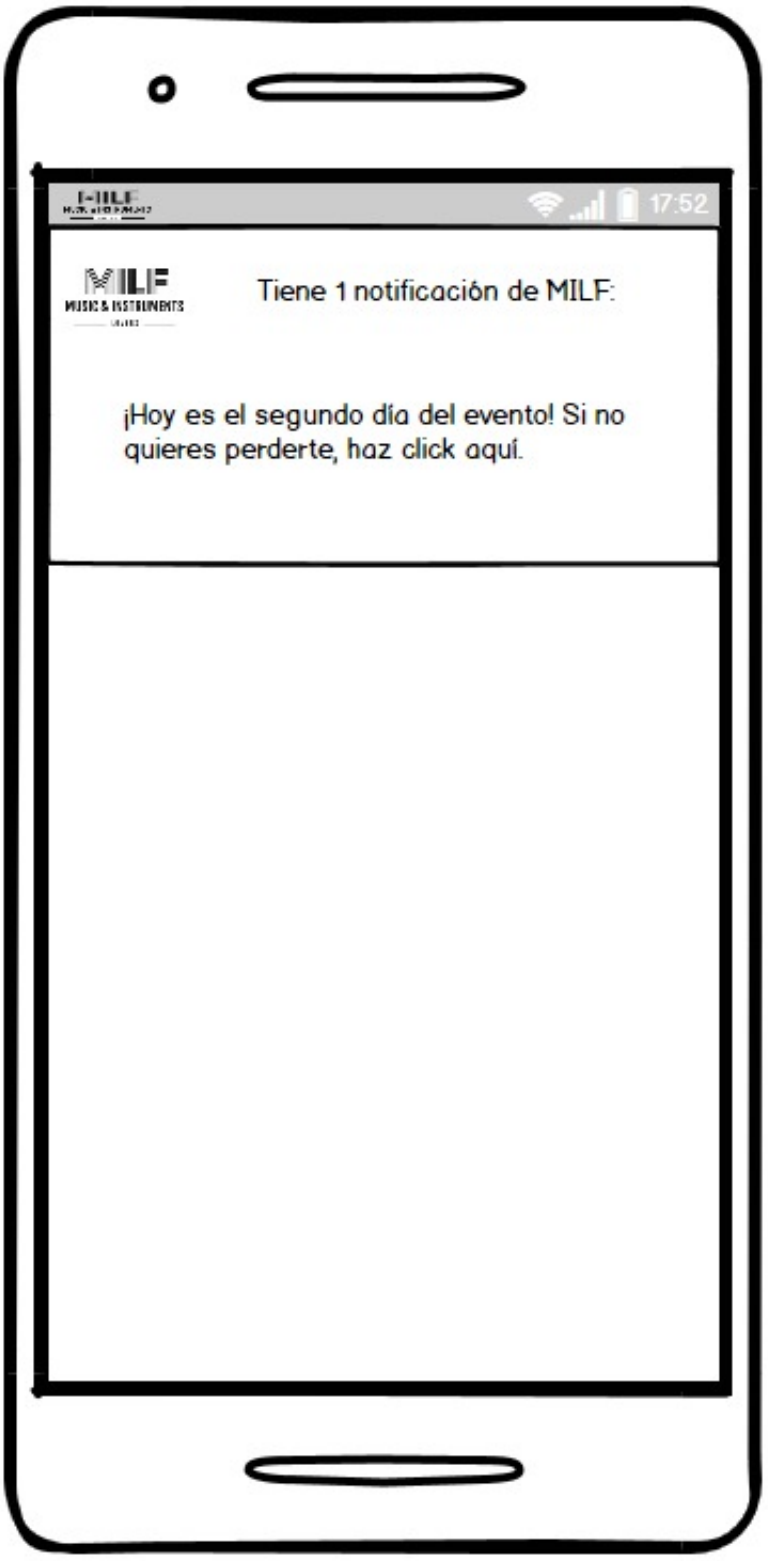


Figura 2.13: Resultado de pulsar la fecha del evento del día de su celebración

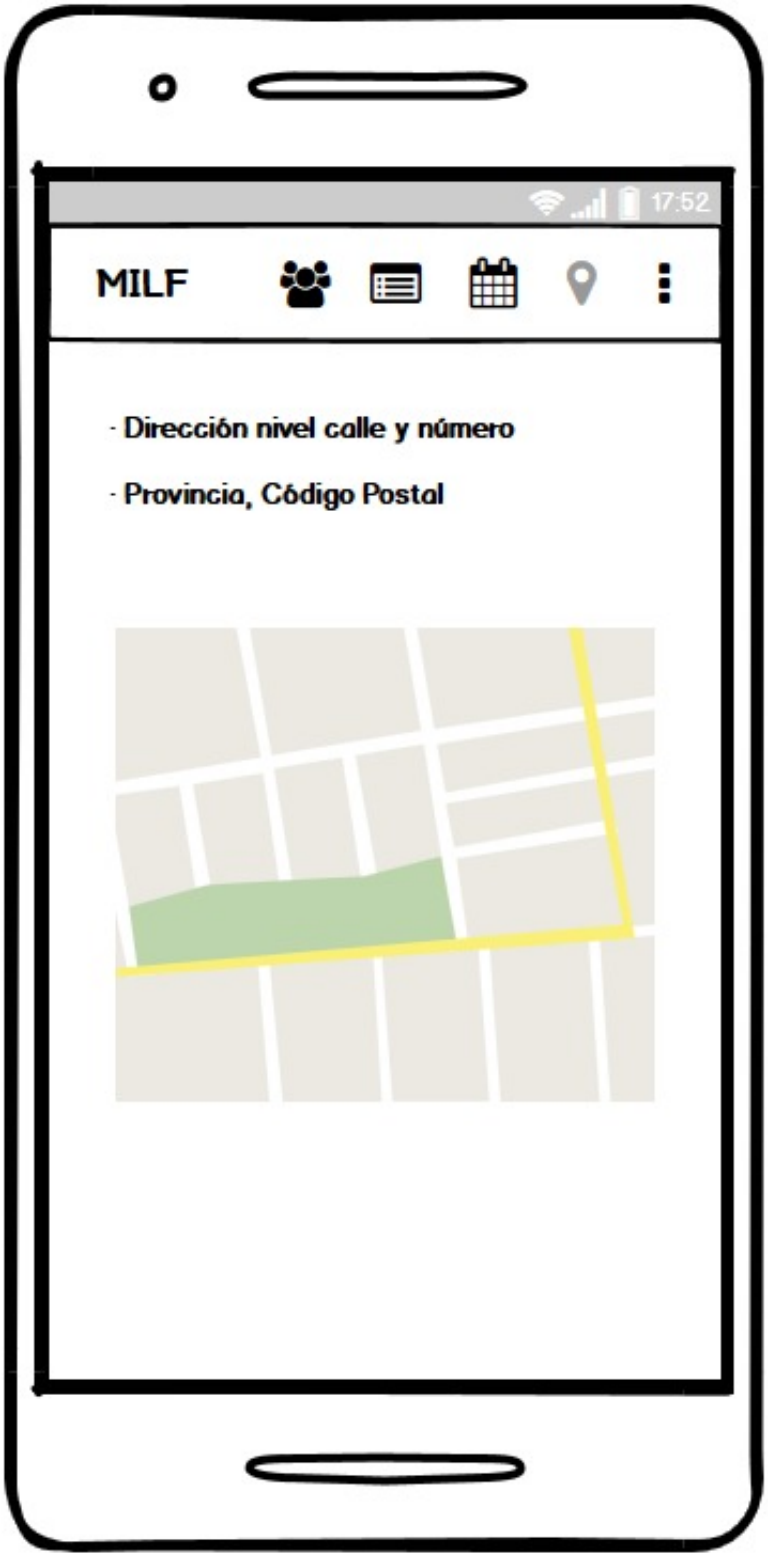


Figura 2.14: Vista de localización

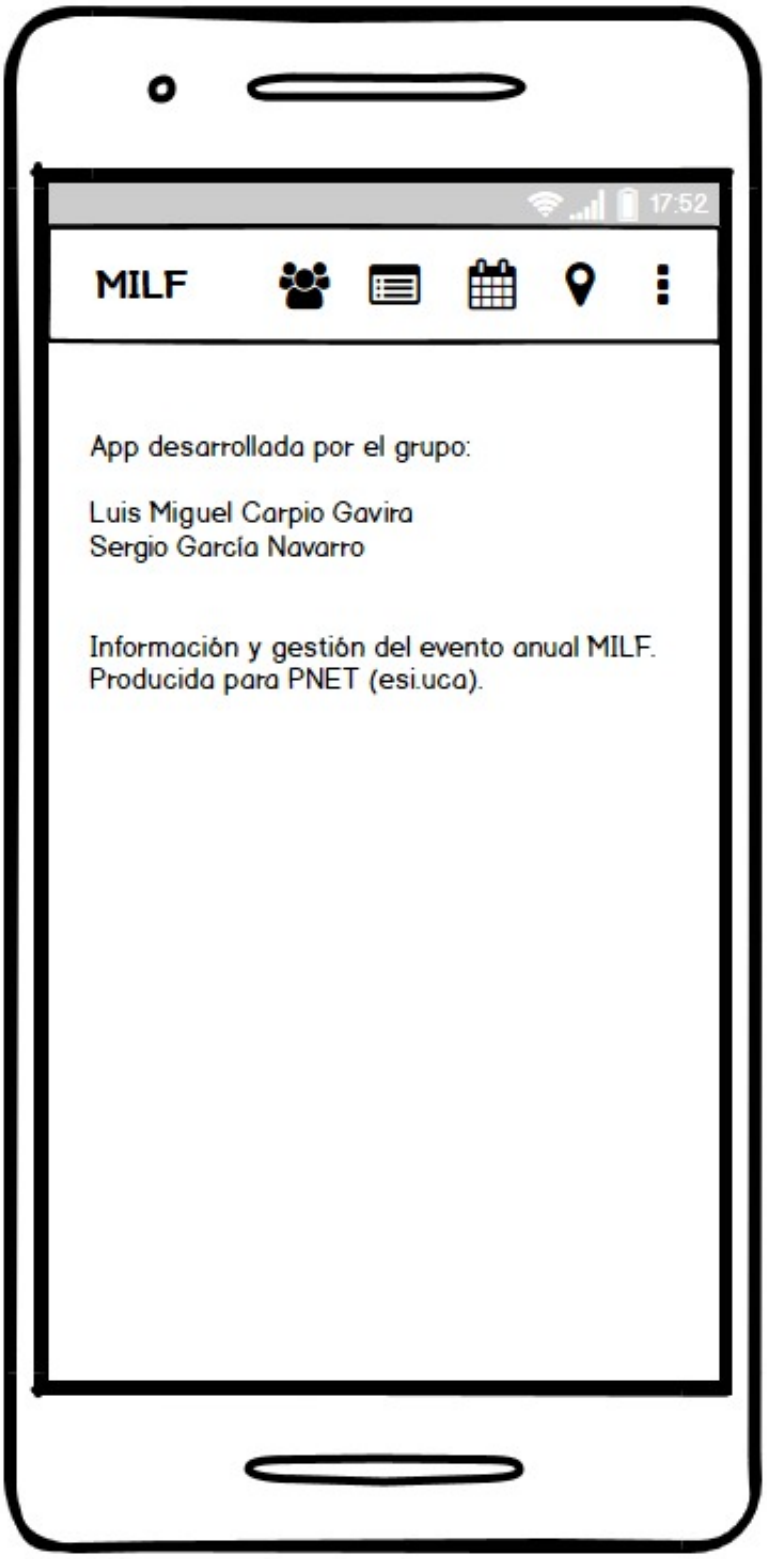


Figura 2.15: Vista de información

# Capítulo 3

## 3.1. Creación de la aplicación

### 3.1.1. Primeros pasos

Para comenzar debemos tener Android Studio [1] instalado y un proyecto creado en dicha aplicación. Empezaremos especificando el cliente que hemos utilizado en la aplicación: OkHttp [2], librería externa que nos facilita la implementación de las peticiones HTTP.

Debemos añadir la dependencia al fichero **build.gradle**.

```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])

    implementation 'androidx.appcompat:appcompat:1.1.0'
    implementation 'com.google.android.material:material:1.0.0'
    implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
    implementation 'androidx.navigation:navigation-fragment:2.0.0'
    implementation 'androidx.navigation:navigation-ui:2.0.0'
    implementation 'com.squareup.okhttp3:okhttp:3.12.0'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'androidx.test.ext:junit:1.1.1'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.2.0'
}
```

Figura 3.1: Dependencia de OkHttp

### 3.1.2. Get all

Usando este código obtendremos todos los asistentes del evento.

```
JSONArray text = null;
OkHttpClient client = new OkHttpClient();
Request request = new Request.Builder().url("http://"+"10.0.2.2"+":8080/audience/").build();
try {
    Response res = client.newCall(request).execute();
    text = new JSONArray(res.body().string());
} catch (Exception e) {
    e.printStackTrace();
}
return text;
```

Figura 3.2: Petición get all

### 3.1.3. Get one

Usando este código obtendremos uno de los asistentes a partir de su id.

```
JSONObject data = null;
OkHttpClient client = new OkHttpClient();
Request request = new Request.Builder().url("http://"+"10.0.2.2"+":8080/audience/"+_id).build();
try {
    Response res = client.newCall(request).execute();
    data = new JSONArray(res.body().string()).getJSONObject( index: 0);
} catch (Exception e) {
    e.printStackTrace();
}
return data;
```

Figura 3.3: Petición get one

### 3.1.4. Put

Usando este código lograremos actualizar el usuario a través de su id.

```
OkHttpClient client = new OkHttpClient();
RequestBody formBody = new FormBody.Builder()
        .add( name: "Nombre", nombre)
        .add( name: "Apellidos", apellidos)
        .add( name: "DNI", dni)
        .add( name: "FechaNac", fechaNac)
        .add( name: "Telefono", telefono)
        .add( name: "FechaIns", fechaIns)
        .build();
Request request = new Request.Builder().url("http://"+"10.0.2.2"+":8080/audience/"+_id).put(formBody).build();
```

Figura 3.4: Petición put

### 3.1.5. Delete one

Usando este código eliminaremos a un usuario a partir de su id.

```
OkHttpClient client = new OkHttpClient();
Request request = new Request.Builder().url("http://"+"10.0.2.2"+":8080/audience/"+_id).delete().build();
try {
    Response res = client.newCall(request).execute();
    return true;
} catch (Exception e) {
    e.printStackTrace();
    return false;
}
```

Figura 3.5: Petición delete one

### 3.1.6. Post

Usando este código añadimos a un usuario a los asistentes actuales con los datos que hayamos insertado.

```
OkHttpClient client = new OkHttpClient();
Instant instant = new Timestamp(System.currentTimeMillis()).toInstant();
ZonedDateTime zdt = instant.atZone(ZoneId.of("Europe/Madrid"));
RequestBody formBody = new FormBody.Builder()
        .add( name: "Nombre", nombre)
        .add( name: "Apellidos", apellidos)
        .add( name: "DNI", dni)
        .add( name: "FechaNac", fechaNac)
        .add( name: "Telefono", telefono)
        .add( name: "FechaIns",  value: zdt.toString().substring(0, 10) + " " + zdt.toString().substring(11, 19))
        .build();
Request request = new Request.Builder().url("http://"+"10.0.2.2"+":8080/audience/").post(formBody).build();

try {
    Response res = client.newCall(request).execute();
    return true;
} catch (Exception e) {
    e.printStackTrace();
    return false;
}
```

Figura 3.6: Petición post

# Bibliografía

[1] Android Studio: `https://developer.android.com/studio` [Accedido el 23-4-2020]

[2] OkHttp: `https://square.github.io/okhttp/` [Accedido el 20-5-2020]