Institut Puig Castellar
Santa Coloma de Gramenet

imagin

Diseño e implementación de una aplicación para la configuración y presentación de un Smart Mirror

CFGS Administració de Sistemes Informàtics i Xarxes
CFGS Desenvolupament d'Aplicacions Multiplataforma

Pablo Patiño Caraballo
Pol Sevillano González
Manel Aguayo Jiménez

Segundo de DAM

01/06/2017

Resumen del proyecto:

Este proyecto pretende comunicar dos aplicaciones basadas en el lenguaje Android desarrolladas con el fin de crear un espacio personalizado para cualquier usuario de SmartMirror, ofreciéndole la posibilidad de crear, mediante los *widgets* correspondientes, un entorno a medida.

Para ello utilizaremos la tecnología de Firebase para tener un control de las configuraciones de los usuarios a tiempo real, que tendrán instalada nuestra app en sus SM y en sus smartphones (siendo estos últimos una suerte de mando a distancia con el cual modificar y crear el entorno que se mostrará en el escenario del espejo, mediante drag & drop).

Abstract:

This project pretends to communicate two Android Apps developed with Android Java Programming Language in order to allow for our users to make her customized space that will be displayed in her SM. Also, we would to know some programming languages and consistency platforms (like Firebase).

Palabras clave: Java, Android, Firebase, POO, SmartMirror, RaspberryPi, JSON, Android Studio, Widget.

Índice

1. Introducción

1.1 Contexto y justificación del proyecto

Tras una exhaustiva investigación en el campo de los SM descubrimos que hasta el momento estos dispositivos solo estaban pensados para, mediante una aplicación que corre sobre Linux, mostrar una información limitada y estática, de modo que si un usuario quería crearse su propio SM en casa se le mostraría exactamente el mismo contenido que a cualquiera que ya tuviera uno, siendo este un escenario de poca utilidad.

Nosotros queremos crear un escenario ambientado por nuestros usuarios, donde cada uno pueda modificar su espacio, personalizandolo y haciéndolo único. Para ello queremos dar un empujón a un sector tecnológico (los SM) que está de capa caída debido a que todavía nadie ha intentado ir más allá, ni desarrollar en firme un software que interactúe con este hardware.

En resumen, este proyecto no intenta cubrir una necesidad existente sino crear una nueva necesidad en los usuarios; que no puedan estar sin su propio SM personalizado.

1.2 Objetivos del proyecto

- Formación sobre la plataforma Firebase.
- Formación sobre Raspberry Pi 2.
- Preparación e instalación de Android en la Raspberry Pi.
- Diseño del SM.
- Implementación de la aplicación que permita comunicar al sujeto configurador con el SM.
- Comunicación entre aplicaciones.
- Creación del producto final.

1.3 Enfoque y método a seguir

La estrategia a seguir es crear un producto nuevo mediante uno existente, es decir, la creación de una app totalmente nueva con funcionalidades personalizadas en un soporte ya existente como son los SM, dando así una nueva utilidad y enfoque a estos dispositivos.

1.4 Planificación del proyecto

1.4.1 Planificación del proyecto y tareas

1. Formación y aprendizaje
    a. Firebase
    b. Raspberry Pi
    c. Fabric
    d. APIS
    e. Tecnología SmartMirror

2. Toma de requisitos

3. Diseño de la aplicación y estado del arte
    a. Diseño de la interfaz gráfica
        i. Paleta de colores
        ii. Menús
        iii. Gestión de espacios
    b. Diagramas UML
        i. Casos de uso
        ii. Clase
        iii. Secuencia
    c. Diseño de logotipo

4. Diseño de la base de datos
    a. Estructura en JSON
    b. Sistema de sincronización/identificación

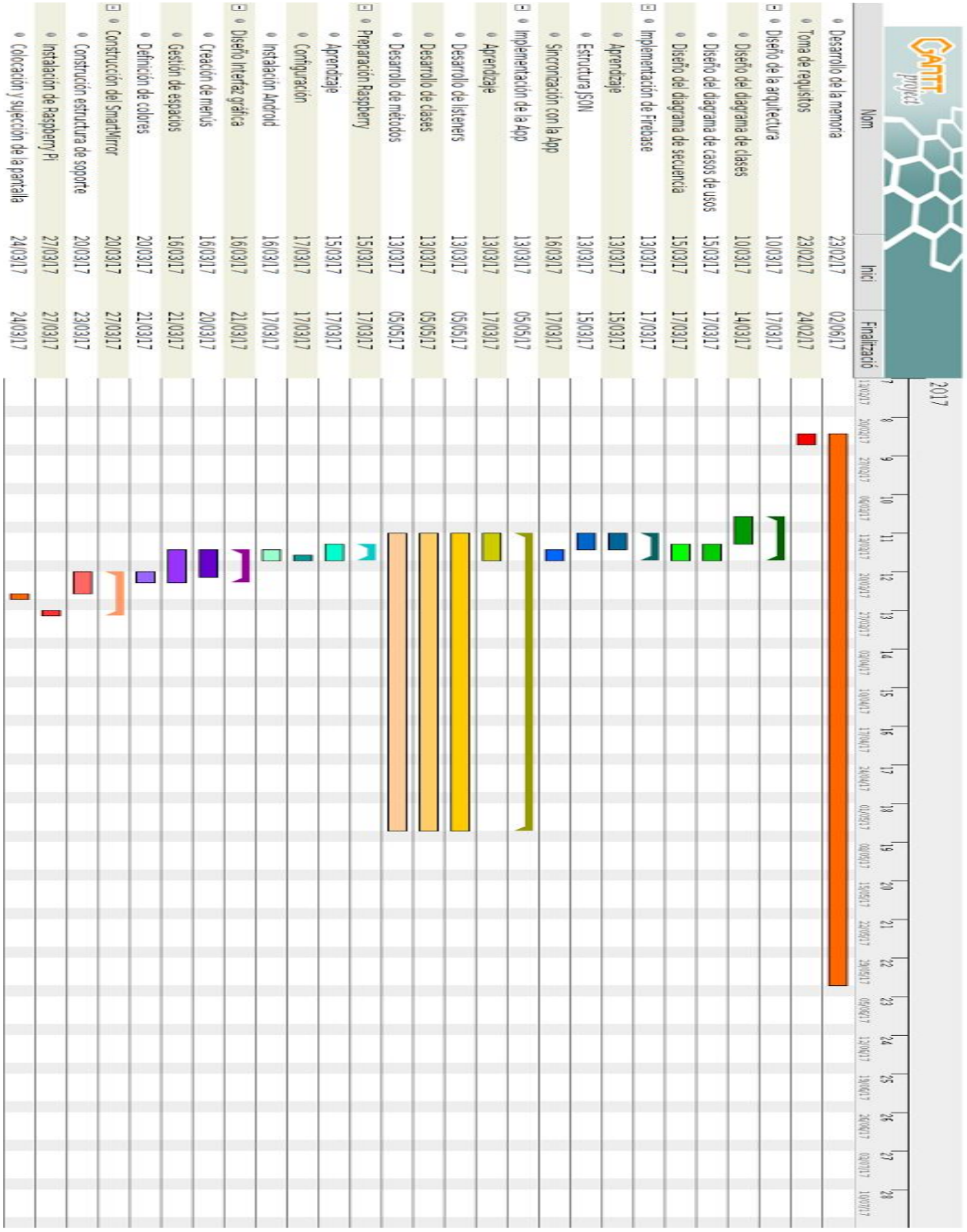
5. Implementación de la aplicación
    a. Creación de clases
    b. Métodos
    c. Listeners
    d. Conexión con Firebase
    e. I/O Data con Firebase

6. Realización de la memoria del proyecto

7. Construcción del SM
    a. Construcción de estructura de soporte.
    b. Colocación y sujección de pantalla.
        i. Creación de sujeciones de madera.
        ii. Instalación de cristal.
    c. Instalación de Raspberry Pi
        i. Instalación del sistema Android
        ii. Construcción de soportes.
        iii. Colocación de Rasperry Pi y cableado.
        iv. Organización de cables y conexiones.

1.4.2 Diagrama de Gantt

En este diagrama podemos observar el planteamiento que tenemos respecto a las fases de nuestro proyecto desde la toma de requisitos hasta la documentación del proyecto.



1.5 Breve sumario de productos obtenidos

Para la producción del proyecto se precisaría:

- Raspberry Pi 2.
- Tarjeta microSD de 8 GB.
- Pincho WiFi compatible con Raspberry Pi 2.
- Cable HDMI.
- Espejo de vidrio Reflectasol.
- Estructura de sujección y escenario de madera de pino.
- Monitor HDMI

1.5.1 Presupuesto final

Raspberry Pi 3 Starter Kit ---------- 64.99 €
Acrylic See-Through Mirror ---------- 26.99 €
Otros materiales (maderas, adaptadores..) ---------- 40.99 €

El presupuesto finalmente no fue aprobado por el departamento de informática del Instituto Puig Castellar, por lo que finalmente todos los recursos necesarios para la realización de este proyecto han salido de nuestros bolsillos.

1.6 Breve descripción de los demás capítulos del proyecto

En los siguientes capítulos del proyecto se reflejarán los distintos apartados del proceso de diseño y estructuración del proyecto así como la implementación de nuestra aplicación, Imagin.

2. Diseño y toma de requisitos

2.1 Toma de requisitos, criterio y decisiones

El proyecto Imagin transfiere un acabado visual potente tras una trabajada configuración tanto de hardware como de software.

En primer lugar, la aplicación Imagin (disponible para dispositivos Android) requerirá un sistema operativo igual o mayor a Android KitKat 4.4 para poder tener un buen funcionamiento. De la misma manera, se precisarán dos dispositivos, pues uno realizará la acción de espejo y su compañero, de configurador.

Por otro lado y para disfrutar de una mejor experiencia visual, recomendamos que el dispositivo que hace la función de espejo esté vinculado a un espejo acrílico. Así podrá tener resultados parecidos a los de la siguiente imagen.

2.2 Diseño de la aplicación y estado del arte

Tipografía

La tipografía escogida es Futura Helvética, aunque se usará la fuente Muli como alternativa gratuita, dado que proviene de Google Fonts.

Características que demuestran su idoneidad

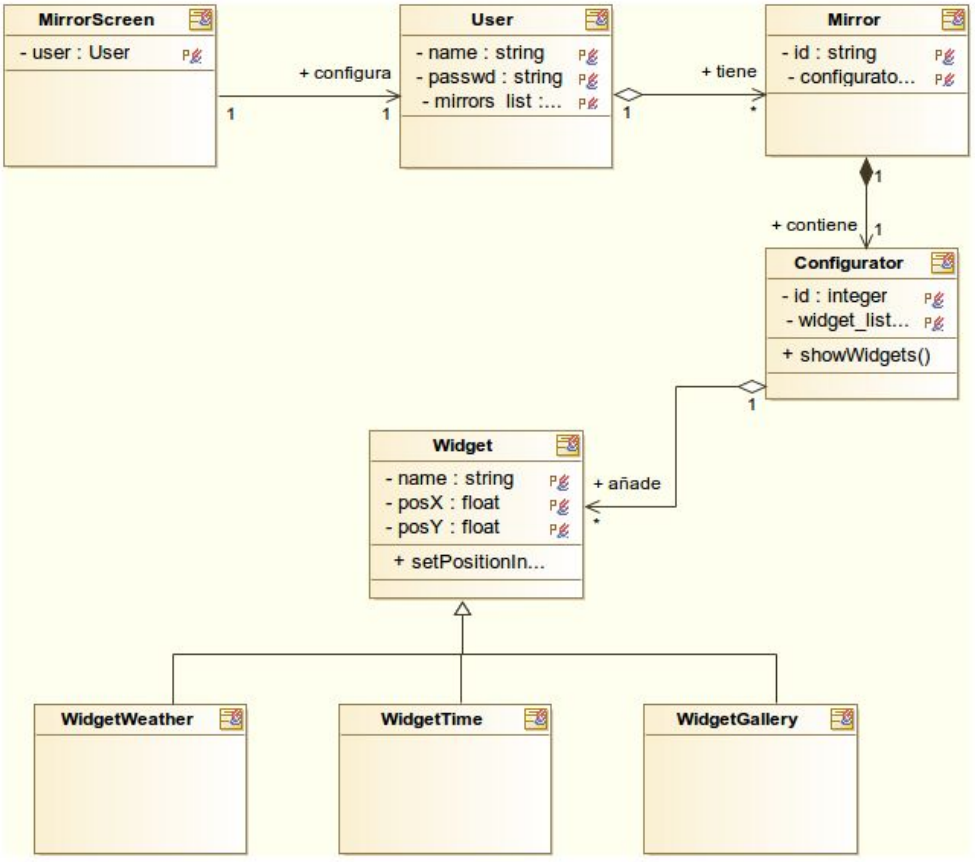
- Un proyecto de código abierto como Imagin necesita una tipografía Sans serif pura racionalista. Por su carácter anodino e internacional.

- Se enfatiza el ritmo natural de la lectura occidental, de izquierda a derecha. Especialmente en la palabra “imagin” que contiene la letra m y n, las delicadas inclinaciones que se muestran en los ejemplos adjuntos facilitan la lectura sobre la interfaz. Que en nuestro caso se limita a dos dispositivos: móvil y smart mirror.

Cómo curiosidad la placa conmemorativa que Neil Armstrong y Edwin Aldrin dejaron en la luna estaba escrita con Futura por su alta legibilidad y su voluntad internacional. Para futuras visitas.

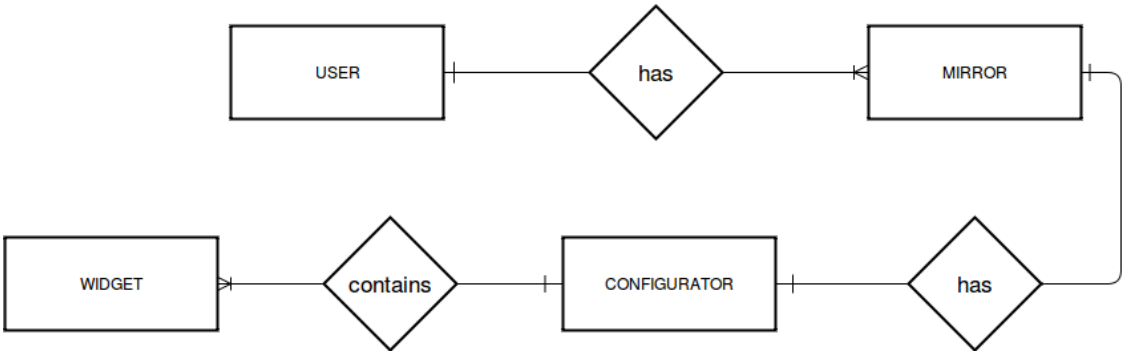
2.2.1 Diagramas de clases (aproximación real)

El presente diagrama es una representación abstracta de lo que será nuestra aplicación, ya que esta está conformada por *activities* y no por clases definidas. Por lo que esto sirve para hacerse una ligera idea de cómo está diseñada nuestra aplicación.



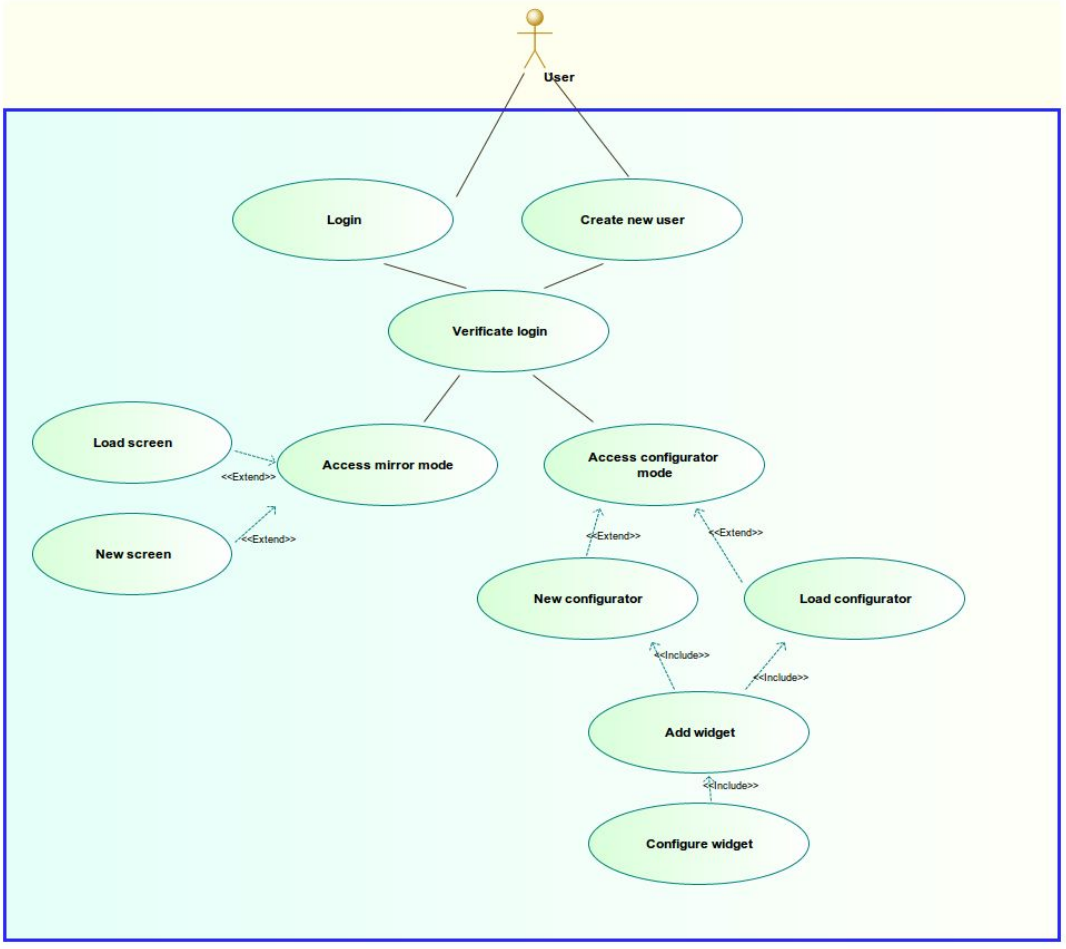
2.2.2 *Diagramas de entidad relación*

En el siguiente diagrama se refleja cómo se estructuraría en un sistema gestor de base de datos relacional nuestra información. Aún así, como más adelante se verá, éste proyecto estructura los datos con la tecnología JSON.



2.2.3 *Diagramas de casos de uso*

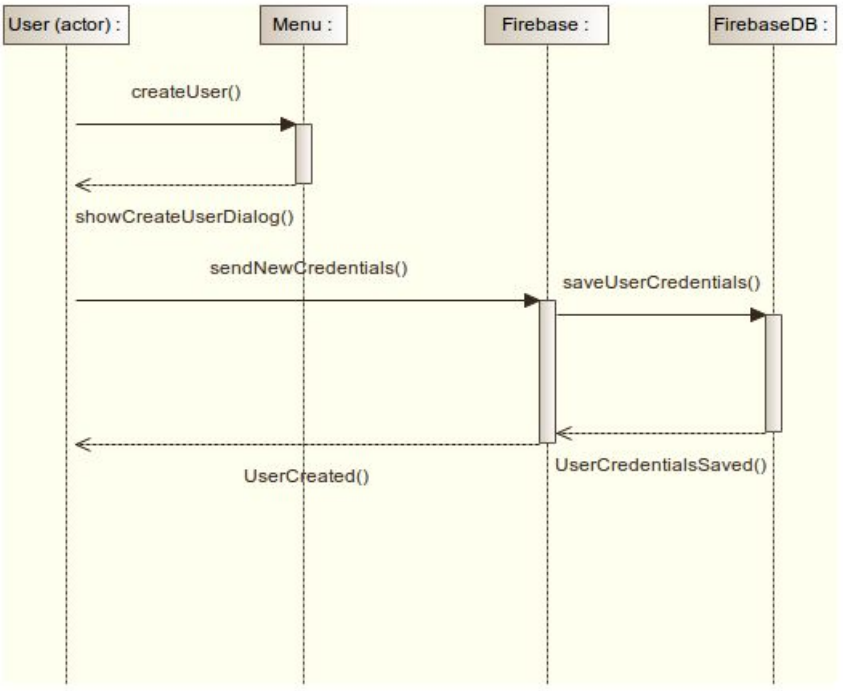
En el siguiente diagrama se muestra a grandes rasgos la interacción del usuario con la aplicación.



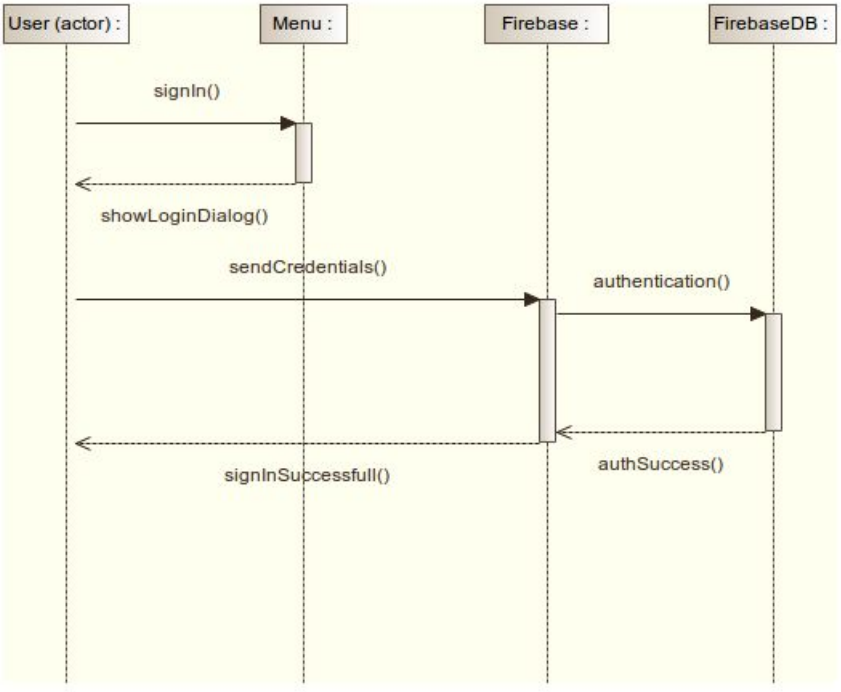
2.2.4 Diagramas de secuencia

Los siguientes diagramas muestran pequeños procesos que ocurren durante la ejecución de la App.

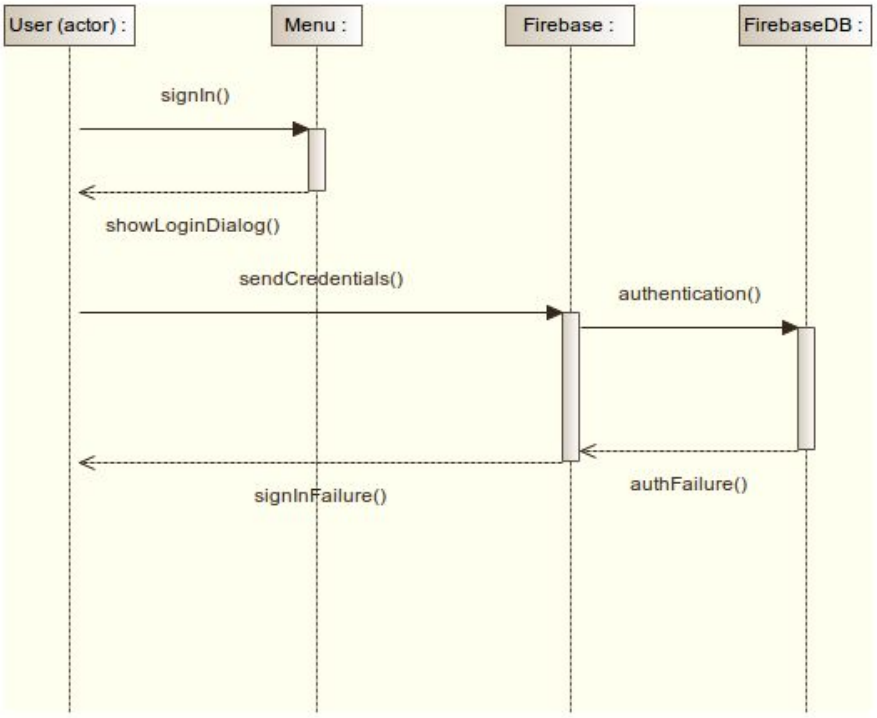
Register New User



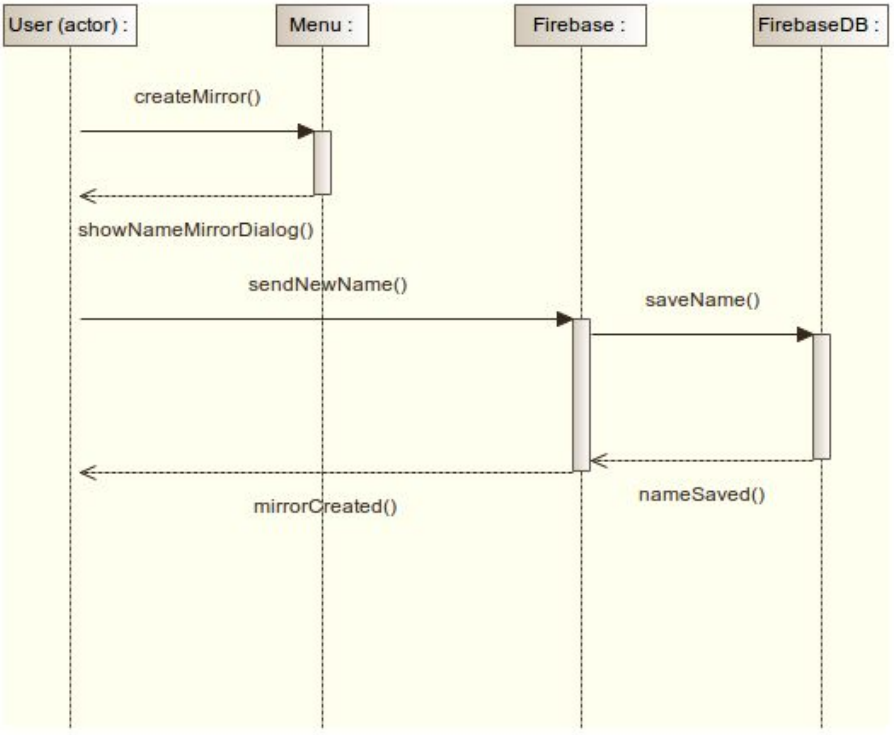
SignIn User Successfully



SignIn User Failure

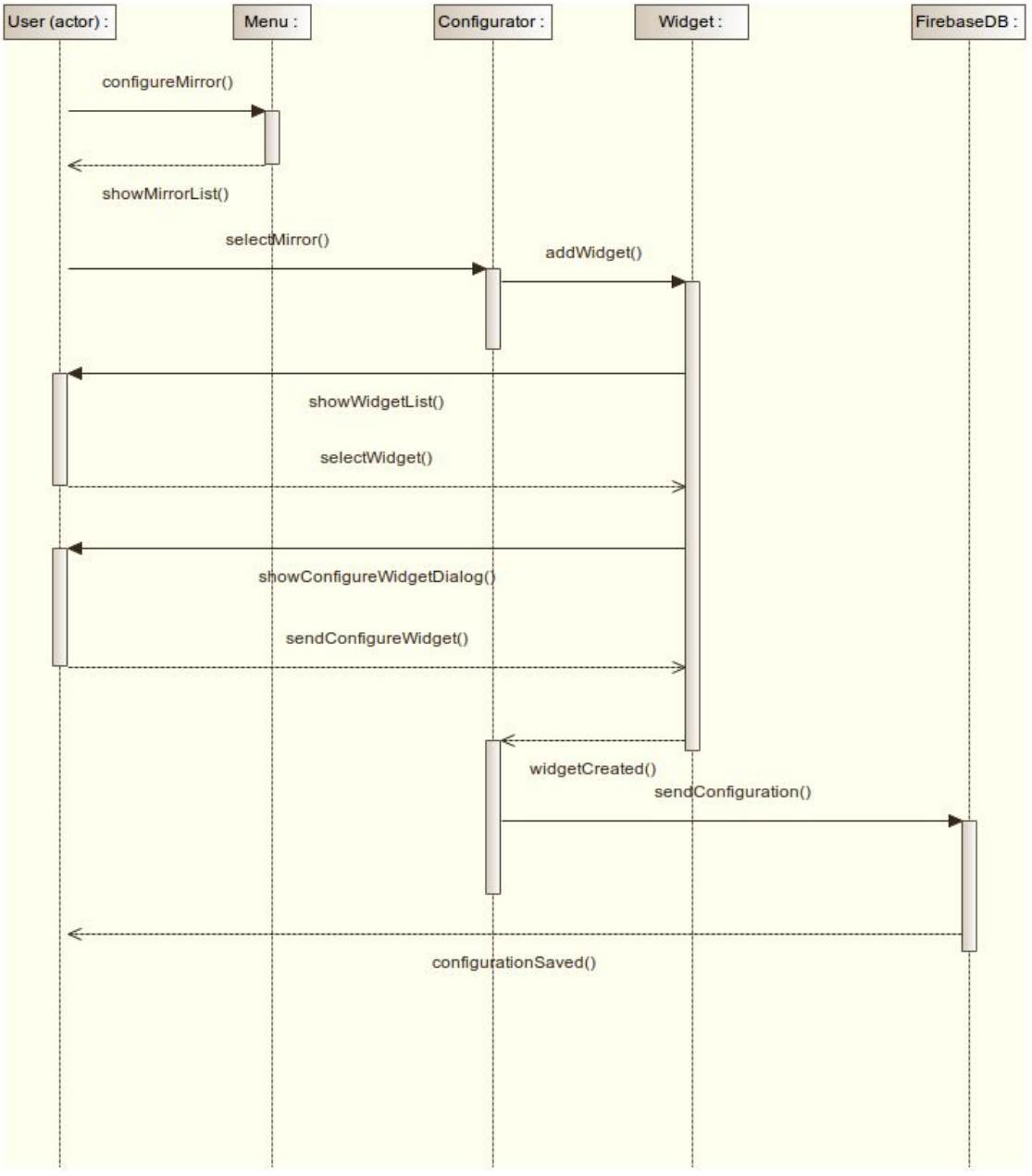


Create New Mirror



Create New Configuration (with widgets)

El siguiente diagrama muestra el proceso de creación de la configuración de un espejo. El configurador añade y configura *widgets* y todo el conjunto es guardado en la base de datos Firebase posteriormente.

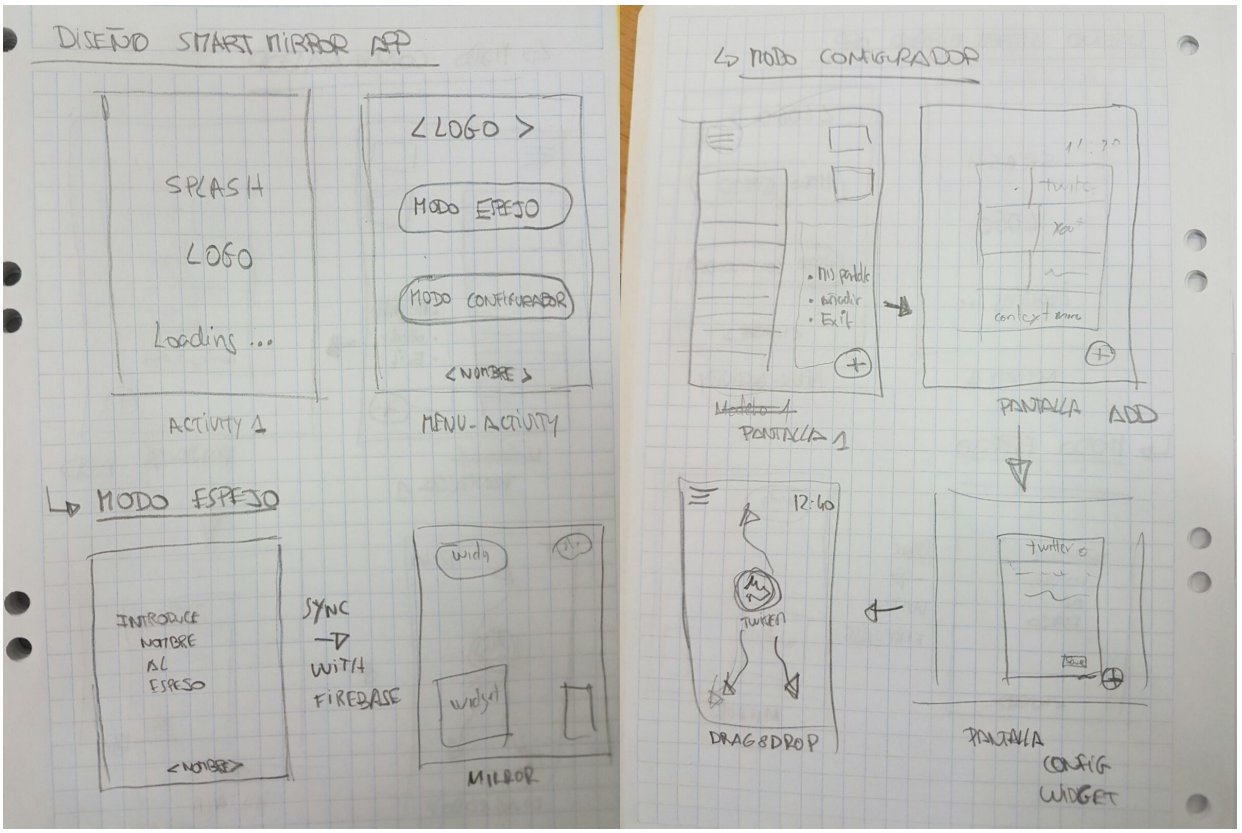


2.2.5 Diagrama de la estructura BBDD JSON


```
"imagine-e34e4":{
   "users":[{
         "FJOfLeZxBacAvxmZrK4bvP058Px1":{
               "KlPXBCt1mgfDYRTUirR":{
                     "configurator":[{
                           "widgetTime":{
                                 "horaActual":"Europe/Madrid",
                                 "name":"widgetTime",
                                 "posXinMirror":0,
                                 "posYinMirror":0
                           }
                     },
                     {
                           "widgetTwitter":{
                                 "active":true,
                                 "hashtag":" ",
                                 "name":"widgetTitter",
                                 "posXinMirror":0,
                                 "posYinMirror":2,
                                 "userName":"Manel Aguayo"
                           }
                     },
                     {
                           "widgetWeather":{
                                 "city":"Barcelona",
                                 "name":"widgetWeather",
                                 "pathImagen":"nubladob",
                                 "posXinMirror":0,
                                 "posYinMirror":5,
                                 "temp":"23.2"
                           }
                     }],
                     "id":"KlPXBCt1mgfDYRTUirR",
                     "name": "Clase2"
               }
         }
   }]
}
```

2.3 Bocetos generales de las activities de la aplicación

Las siguientes imágenes muestran los primeros pasos de Imagin. Los bocetos representan una fase muy temprana del diseño pero que ayuda a tener una primera idea de cómo acabará visualizandose la app.



2.4 Diseño de logotipo e imágenes de la aplicación

En los bocetos adjuntos se muestra la intención de crear una identidad corporativa con el nombre completo del proyecto Imagin. A partir del cual se creará el icono de la aplicación tomando como punto de referencia las dos letras 'i' presentes en el nombre del proyecto por su forma antropomorfa relacionada con el uso cotidiano del espejo.

Esta fue la primera idea que se manejó para el proyecto, con una cierta inclinación para darle un toque de sabor y sabrosura.

En la siguiente imagen se muestra una de las pruebas cuando se decidió juguetear con la idea del 3D ya que la app trabaja con un espejo que muestra la realidad.

imagin

imagin

El proceso creativo cambió radicalmente al concebir la idea de que al haber dos letras ies que tienen cierto parecido a la fisionomía humana, podríamos usarlo para diseñar algo más divertido y cercano. Aquí una de las ideas en la que el segundo punto de la i es una sonrisa.

Intentamos darle un giro un poco más serio al diseño, sin perder la premisa de jugar con las letras ies.

Finalmente encontramos una línea de diseño con la que nos sentíamos cómodos, siguiendo la idea de jugar con las ies. Se trata de un diseño limpio y conciso en la que se busca mostrar el símil de una persona reflejada en el espejo. El icono es la representación minimalista de esta idea, donde las letras actúan como una persona viéndose reflejada.

No se trata del diseño final de la imagen corporativa, pero sí será la línea a seguir para encontrar nuestro diseño final.

Diseño final

A continuación mostramos el diseño final de la aplicación Imagin. Elegido como color principal el azul y como secundario el negro. La fuente Future Helvética protagoniza la imagen corporativa de la aplicación. El icono de ejecución, sin embargo, juega con la visión del usuario, dejándole la libertad para intuir que se representa a una persona junto a su reflejo en su espejo.

3. Implementación de la aplicación

3.1 Clases Java (consistencia)

La siguiente estructura de clases permiten y definen la consistencia de este proyecto, permitiendo, a partir de las estructuras HashMap, introducir directamente el User que conllevará todo lo de su interior.

User

La clase User es una de las clases de consistencia con más importancia. Pues es la que controla que espejos tiene cada usuario.

Mirror

La clase Mirror define al espejo en sí. Cada espejo tendrá un nombre, un identificador (creado por Firebase) y lo más importante, su configurador. Es importante decir que cada espejo puede tener únicamente una configuración.

Configurator

El Configurator contiene todos los tipos de *widgets* que están implementados en nuestra aplicación. También, por otro lado, es la clase encargada de controlar cuál es el espejo con el que está trabajando a tiempo real la aplicación (el espejo el cual configura).

Widget

La clase Widget es una clase general en la que encontramos las variables generales que compartirán todos los *widgets*. El nombre, si está activo en la aplicación y su posición en la pantalla, son las variables que esta clase contiene.

WidgetTime

La clase WidgetTime es configurada sobre la activity WidgetTimeConfiguratorActivity que veremos más adelante. La clase contiene la lista de *timezones* que existen en Android y una variable que almacena el *timezone* que el usuario quiere mostrar en el espejo.

WidgetTwitter

La clase WidgetTwitter es configurada sobre la *activity* WidgetTwitterConfiguratorActivity que veremos más adelante. Esta clase guarda el usuario con el cual se ha autenticado en Twitter, el *hashtag* y/o el username que el usuario decide buscar en la clase configuradora mediante filtros.

WidgetWeather

La clase WidgetWeather es donde se hacen todas las transacciones para descargar la información del tiempo de la pagina web www.openweathermap.org en forma de JSON. Hablaremos más adelante de lo que concierne a la parte de código .

DefaultMirror

Esta clase representa la configuración por defecto que tiene cada espejo creado. Cuando el usuario crea un espejo, se le asigna esta configuración inicial, que luego en el modo configurador, el usuario podrá modificar a su gusto.

3.2 Activities

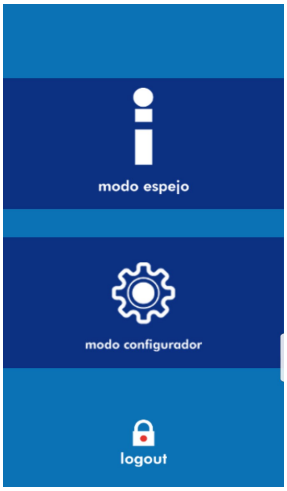
SplashActivity

La SplashActivity es aquella que se lanza cuando se ejecuta la aplicación. Se muestra en *background* la imagen corporativa de la aplicación esperando a tener todos los preparativos para iniciar la aplicación.

FirebaseConnection y GoogleApiActivity

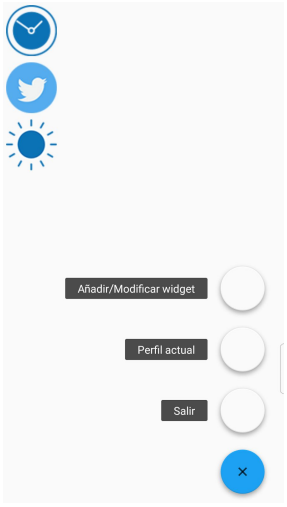
Ligado al inicio de sesión y en segundo plano, también se inicia sesión en los servicios de Google, que nos permite la autenticación mediante una cuenta existente de Google en Firebase, automáticamente al iniciar sesión se guarda en la base de datos dicho usuario.

StartMenuActivity



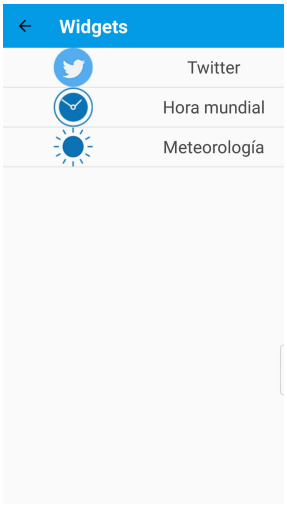
Esta *activity* ofrece dos modos con los que podremos ejecutar la app. En primer lugar el modo espejo será aquel que el usuario debe escoger para ejecutarse en el dispositivo que tenga conectado la pantalla que hace las veces de espejo. El otro modo que conforma nuestra app es el configurador. Este modo puede abrirse desde cualquier dispositivo para configurar un espejo seleccionado de tu lista de espejos. Cabe destacar que un espejo tiene un único configurador.

ConfiguratorView



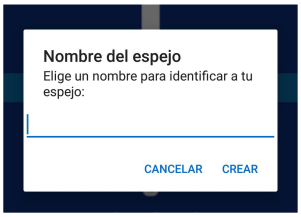
Se trata de la *activity* principal del Modo Configurador. El usuario interactúa, configura y posiciona los *widgets* que desea que sean mostrados en el espejo elegido. El posicionamiento en el espejo se realiza mediante el sistema *Drag&Drop* en el que el usuario puede ubicar un *widget* en el espejo simplemente arrastrándolo por la pantalla, a modo previsualización. En esta *activity* reina el botón flotante que despliega el menú con tres opciones: Añadir *widgets*, perfil actual(Información del espejo en el que estás), y salir a la *activity* de elección de modos.

WidgetSelectConfiguratorList



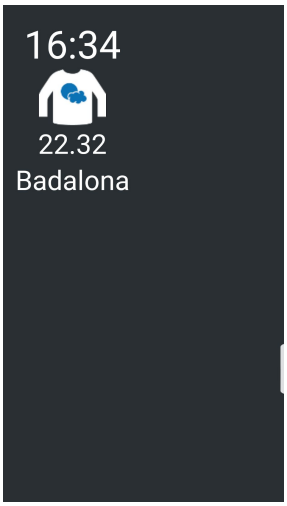
En esta *activity* encontraremos la lista de *widgets* disponibles de la aplicación Imagin. Al seleccionar un *widget* de la lista iremos directamente a configurar dicho *widget*.

MirrorCreateFragment



En esta clase creamos el DialogFragment donde le ponemos el nombre al espejo y lo definimos como espejo actual en el que mostraremos la información de la base de datos en Firebase.

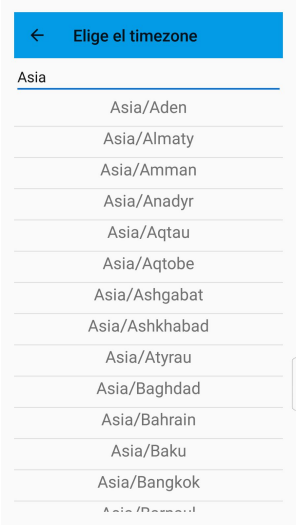
MirrorActivity



La MirrorActivity, como bien define su nombre, es la propia visualización del SM. La información se muestra y se posiciona gracias a su configurador. El usuario propietario de este espejo ha elegido anteriormente qué y dónde mostrar los *widgets* desde el 'Modo configurador'.

En el caso que el propietario elija no configurar el espejo, éste tendrá unos *widgets* por defecto configurados.

WidgetTimeConfiguratorActivity

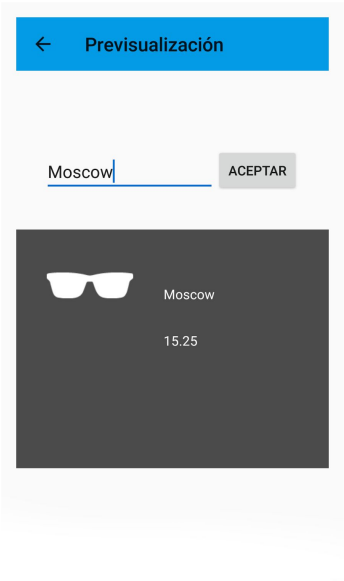


Ésta *activity* permite elegir al usuario la zona horaria a la que desea adaptar la hora que aparece en el espejo. Para facilitar la búsqueda en la extensa lista de zonas horarias, se ha habilitado un buscador que va actualizando a tiempo real el *listview*. También es conocido como el configurador del *widget* del tiempo.

MainActivity

Esta activity es la principal, donde se inicia Fabric y se ejecuta la primera activity de Imagin, la splashActivity. También aplicamos las políticas de permisos.

WidgetWeatherConfiguratorActivity

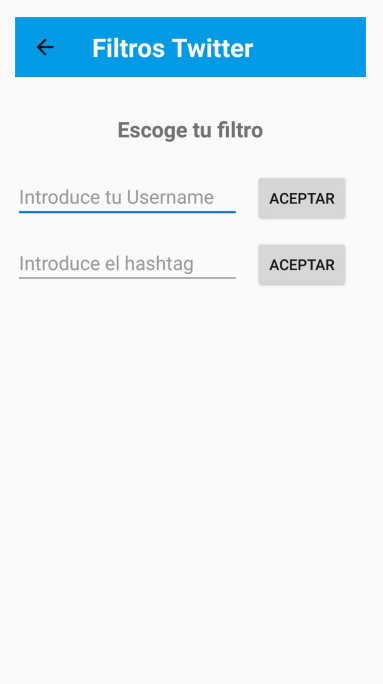


WidgetWeatherConfiguratorActivity es el configurador del *widget* meteorológico. Éste configurador permite, a partir de una búsqueda, saber el estado meteorológico de una ciudad específica del mundo.

WidgetTwitterLoginActivity

Desde aquí es posible hacer *login* en una cuenta de Twitter además otorgamos permisos a Imagin, mediante Fabric, para utilizar Twitter y todas sus utilidades. Solo es necesario hacerlo una vez, al hacerlo quedarán las credenciales cacheadas.

WidgetTwitterConfiguratorActivity



Esta *activity* es la que configura lo que se mostrará en el espejo, por defecto se muestra el *timeline* de la cuenta con la que se haya hecho *login* el usuario, sin embargo Imagin permite búsquedas filtradas para mostrar un hashtag o tus interacciones con un usuario en concreto.

3.3 Widgets y futuras implementaciones

El siguiente punto explica los aspectos, en cuanto a código se refiere, más importantes de nuestros *widgets*. Además, también se añaden posibles futuras implementaciones.

3.3.1 WidgetTime

El widget del tiempo nos permite, a partir de una zona horaria, retornar la hora a tiempo real en esa zona. El siguiente código representa el método que se encarga de realizar esa función.

```
public static String timeNow(String timezone){

    TimeZone tz = TimeZone.getTimeZone(timezone);
    Calendar c = Calendar.getInstance(tz);
    String time = String.format("%02d" , c.get(Calendar.HOUR_OF_DAY))+":"+
            String.format("%02d" , c.get(Calendar.MINUTE));

    return time;

}
```

Una de las cosas más importantes del proyecto Imagin es la lectura y escritura de los datos en Firebase, la base de datos JSON que da soporte a Imagin.

3.3.2 WidgetWeather

```
public String getJSON(String url, int timeout) {
    HttpURLConnection c = null;
    try {
        URL u = new URL(url);
        c = (HttpURLConnection) u.openConnection();
        c.setRequestMethod("GET");
        c.setRequestProperty("Content-length", "0");
        c.setUseCaches(false);
        c.setAllowUserInteraction(false);
        c.setConnectTimeout(timeout);
        c.setReadTimeout(timeout);
        c.connect();
```

Cabe destacar la manera en como se descarga el JSON de la ciudad elegida, para ello utilizaremos la clase HttpURLConnection que nos permitirá hacer el trabajo tan solo rellenando ciertos parámetros de la descarga, entre ellos le decimos que el método que utilizamos es *GET*, le pasaremos un *timeout* para cortar la conexión en caso de que la conexión perdure demasiado tiempo y para finalizar, lanzamos el método *connect* para hacer la conexión y poder descargar el fichero.

3.3.3 WidgetTwitter

Este Widget utiliza el *framework* Fabric para la autenticación en Fabric, permite buscar y mostrar, mediante una query que se realiza a la API de Twitter, el *timeline* que más interese. Los datos del usuario, y de sus búsquedas se guardan en la base de datos de Firebase.

```
if (wtt.getActive()) {

    if (wtt.getUserName().isEmpty() ) {
        searchTimeline = new SearchTimeline.Builder().query(wtt.getHashtag()).build();
    }else if(wtt.getHashtag().isEmpty()) {
        searchTimeline = new SearchTimeline.Builder().query(wtt.getUserName()).build();
    }

    TweetTimelineListAdapter timelineAdapter = new TweetTimelineListAdapter(MirrorActivity.this, searchTimeline);
    ListView lv = new ListView(MirrorActivity.this);
    lv.setAdapter(timelineAdapter);

    if (lv.getParent() != null) {
        ((ViewGroup) lv.getParent()).removeView(lv);
    }

    currentListview=lv;
```

3.3.4 WidgetInstagram

En el *widget* de Instagram podremos ver las fotos de nuestros contactos y podremos hacer búsquedas a través de un buscador donde podremos ver las fotos de una persona en concreto,

3.3.5 WidgetYoutube

Este widget lo que pretende es, poder visualizar videos en el SM, teniendo en el apartado del configurador un buscador para poder elegir que contenido multimedia visualizar en el espejo.

3.3.6 WidgetMagicDraw

El WidgetMagicDraw permite al usuario poner el espejo en 'modo dibujo' en la que mediante su dispositivo configurador podrá realizar dibujos en el espejo, a tiempo real.

3.4 Análisis de la producción

El siguiente punto busca analizar la capacidad productiva de los integrantes del proyecto con el objetivo de mejorar para un futuro próximo. La información extraída y mostrada a continuación es mostrada bajo el concepto de Github Analytics y la analítica personal de cada desarrollador.

Se han invertido en Imagin un total de unas **120 horas** aproximadamente, entre diseño, estructuración e implementación. El proyecto ha tenido **69 *commits*** y hasta **11 ramificaciones** laterales.

Según Github, la primera versión de la aplicación Imagin ha sido desarrollada durante **62 días** con un total de **más de 6000 línea**s de código. La actividad en el proyecto ha ido aumentando al paso del tiempo.

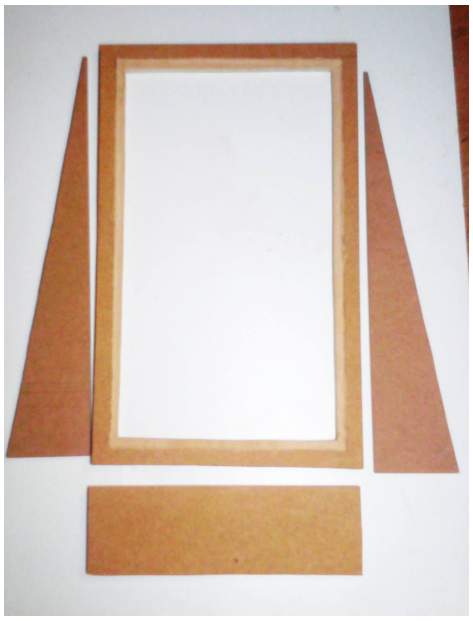
3.5 Construcción y diseño del espejo

Este espejo está pensado para ser construido con una lámina de acrílico de doble cara para tener un efecto óptimo, lamentablemente este material es complicado de conseguir en España por lo que se ha utilizado otra solución igual de válida que nos permite tener ese efecto mágico que buscábamos.

Nuestro espejo es una lámina de vidrio reflectasol que permite el paso de la luz en una dirección (de dentro a fuera).

Tras la lámina de vidrio se encuentra un panel LED de 20” que hará las veces de escenario mágico.

A dicha lámina la recubre un marco de madera de pino para proteger y esconder toda la parte de *hardware*, y darle un toque más hogareño.



Tras todos estos elementos se encuentra la Raspberry con el sistema Android, y un ladrón para alimentar la pantalla y la Raspberry, de este modo del armazón del espejo solo sale un único cable que deberemos enchufar para que todo funcione.

4. Conclusiones

La coordinación entre compañeros es esencial para poder realizar un proyecto cooperativo como es Imagin. La toma de decisiones y la resolución de los problemas precisan de una comunicación y una coordinación entre los integrantes del proyecto.

La formación adquirida y documentada en los anexos de este proyecto ha permitido conocer otras tecnologías y ampliar así el campo formativo. Los conocimientos en Android, y en tecnologías como Firebase, Fabric, y API's han subido un escalón, siendo, como no podía ser de otra forma, un proyecto muy enriquecedor.

Los objetivos mínimos establecidos para este proyecto han sido completados correctamente, tras un ajuste de planificación.

Conforme ha avanzado el proyecto se ha ido mejorando la metodología. Se ha adquirido nuevas técnicas a la hora de programar, así como una mejora de la autonomía.

A lo largo del desarrollo de este proyecto han habido cambios en la estructura del proyecto tanto en *software* como en *hardware* que han hecho necesario improvisar e ingeniar soluciones óptimas de las que a día de hoy sería posible sentirse satisfechos.

Se ha establecido una línea de trabajo futuro buscando la mejora y la adaptabilidad del usuario a la aplicación, así como el uso de muchas más tecnologías.

Imagin nació bajo la idea de ser un soplo de aire fresco para todos aquellos que quieren tener un SM más personalizable en sus casas, y ser un punto de encuentro para todos aquellos desarrolladores que quieran agregar sus ideas y sus conocimientos para hacer Imagin la app definitiva para SM en el mundo Android.

Agradecer los diseños y las imágenes de Imagin a Rosana Patiño Caraballo, por su trabajo, esfuerzo y dedicación al proyecto.

5. Glosario

- SM : Smart Mirror.
- SDCard: Tarjeta de memoria.
- HDMI: Conector de imagen y sonido para conectar la Raspberry con la pantalla.
- UML: Diagramas de diseño de la aplicación.
- App: Aplicación.
- Android: Sistema operativo donde va a correr nuestra aplicación.
- JSON: JavaScript Object Notation
- API: Application Programming Interface
- Widget: Se trata de micro aplicaciones que se despliegan en la pantalla de la aplicación.
- DB: (Data Base) acrónimo de base de datos en inglés.

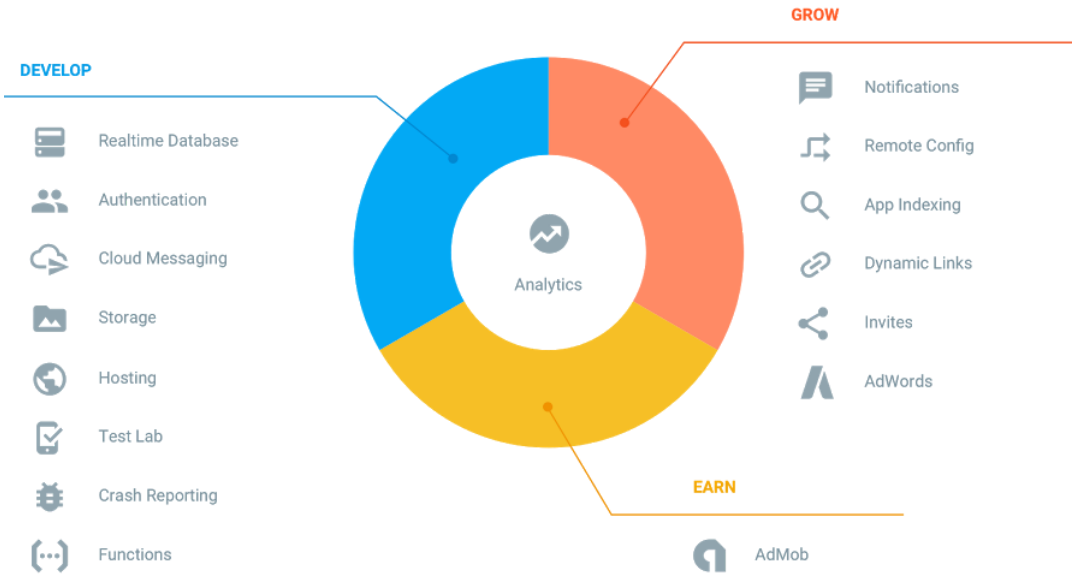
6. Bibliografía

1. Firebase Google Documentation - https://firebase.google.com/docs/
2. Raspberry Pi Documentation - https://www.raspberrypi.org/
3. Android Documentation - https://developer.android.com/
4. StackOverflow - https://stackoverflow.com
5. Documentación RP2 Google Play Services - https://groups.google.com/forum/#!topic/android-rpi/vH_xwFFjOVM https://www.orduh.com/gapps-5-1-lollipop-download/ https://www.raspberrypi.org/forums/viewtopic.php?f=46&t=47152
6. http://www.hermosaprogramacion.com/2015/08/tutorial-layouts-en-android/
7. GridLayout Documentation - http://www.androidhive.info/2012/09/android-adding-search-functionality-to-listview/
8. OpenWeahter - https://openweathermap.org/current
9. Drag&Drop documentation - https://www.tutorialspoint.com/android/android_drag_and_drop.htm
10. Fabric & Twitter https://get.fabric.io/
11. Android in Raspberry Pi 2 https://www.youtube.com/watch?v=aSgQDhM84Ko

7. Anexos

7.1 Firebase

Qué es Firebase?



Google define a Firebase como “Firebase es una plataforma móvil que le ayuda a desarrollar rápidamente aplicaciones de alta calidad, aumentar su base de usuarios y ganar más dinero. Firebase se compone de características complementarias que se pueden mezclar y combinar para satisfacer necesidades.”

Estructura de la base de datos

Todos los datos de base de datos en tiempo real de Firebase se almacenan como objetos JSON. Puede pensar en la base de datos como un árbol JSON alojado en la nube. A diferencia de una base de datos SQL, no hay tablas ni registros. Cuando agrega datos al árbol JSON, se convierte en un nodo en la estructura JSON existente con una clave asociada. Puede proporcionar sus propias claves, como ID de usuario o nombres semánticos..

Acciones de lectura y escritura en Android

Para operaciones básicas de escritura, puede usar setValue () para guardar datos en una referencia especificada, reemplazando cualquier dato existente en esa ruta.

```
private void writeNewUser(String userId, String name, String email) {
    User user = new User(name, email);

    mDatabase.child("users").child(userId).setValue(user);
}
```

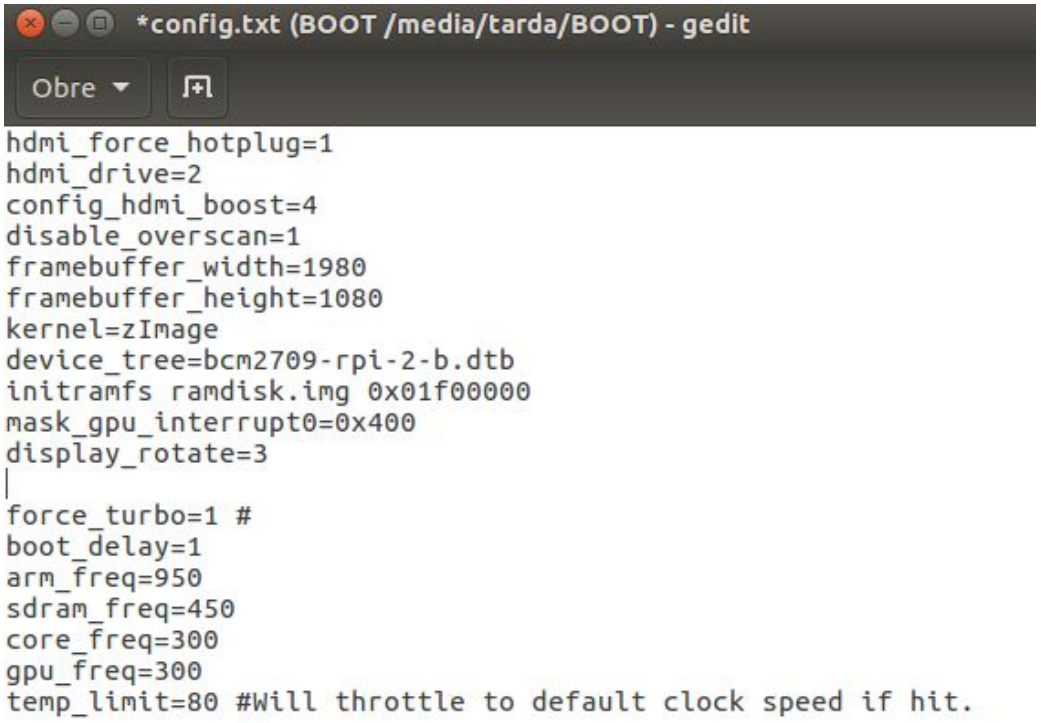
7.2 Raspberry Pi (con Android Lollipop 5)

Introducir Android en una Raspberry Pi no es tarea sencilla, ya que las Raspberry's no están optimizadas para dicho S.O. Existen softwares como BerryRoot que incorporan un Android KitKat, lamentablemente estas opciones no funcionaron. Por lo que se realizaron las particiones de la SD a mano para permitir que la Raspberry Pi pudiese correr un sistema Android.

Aún así el sistema Android funciona de manera poco eficiente en la Respberry PI 2 por lo que fue necesario realizar un ligero overclock de la misma, aumentando su capacidad para poder correr de manera más fluida el Android 5.0 Lollipop.

En la siguiente foto podemos observar el contenido del archivo config.txt y por consiguiente la configuración de la Raspberry. La primera parte muestra la configuración estándar, la resolución de la pantalla (1980x1080), y otros parámetros para crear el ambiente perfecto para Imagin.

La segunda parte es el overclock del dispositivo aumentando todas sus capacidades un escalón más.



7.3 ADB (Android Debug Bridge)

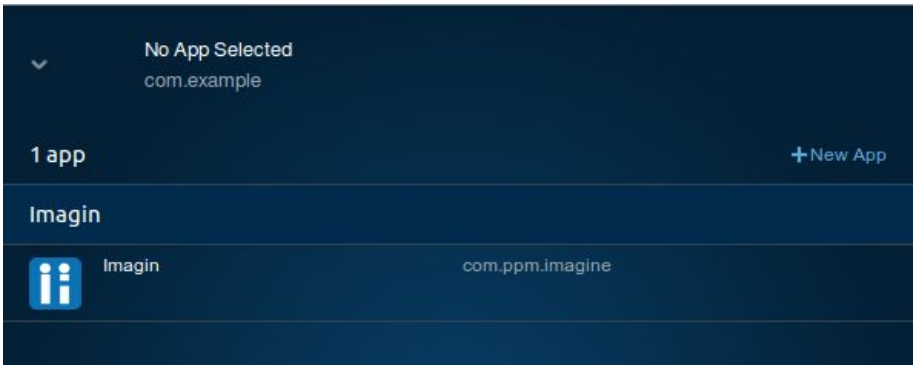
Es una herramienta software que nos permite interactuar con un sistema Android desde un ordenador. Así, por ejemplo, a través de ADB (Android Debug Bridge) podemos ejecutar comandos para copiar archivos desde el ordenador al teléfono, del teléfono al ordenador o reiniciar el dispositivo en el modo recovery.

El proceso llevado a cabo con Android Debug Bridge para el proyecto Imagin ha sido el siguiente:

1. Conexión con el dispositivo Android mediante el comando **adb connect ip_dispositivo**.
2. Acceder a ADB como root con **adb root**
3. Montaje del dispositivo en el sistema
4. Instalación remota del .apk mediante el comando **adb install nombre_app.apk**

7.4 Fabric

Fabric es una suite para apps móviles que creó Twitter, y que más tarde compró Google, el cual integró este Framework en su IDE Android Studio. Fabric permite utilizar distintas herramientas con distintas funcionalidades.. Fabric ha facilitado tanto la obtención de la Secret_Key y de la API_key, como del Login en Twitter desde la app Imagin.



Fabric nos permite interactuar con sus herramientas desde nuestro IDE facilitando la integración de funciones y de clases que se encargan de conectar las herramientas con el código.