eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Trabajo fin de grado

Ingeniería del software

EHU Zurekin,

aplicación móvil para la agrupación de servicios EHU para el alumnado.

David Ruiz Alunda

Junio 2021

Dirección: Mª Begoña Losada Pereda

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

Actualmente para que un alumno esté completamente informado de lo que pasa en la universidad necesita de varias aplicaciones:

- **G.A.U.R.** , para saber los horarios de las asignaturas, las tutorías que ofrecen los docentes de esas asignaturas, las notas finales, etc.
- **Correow** , para ver los correos, tanto los informativos como los personales.
- **Página EHU**, generalmente para ver el calendario, los horarios de las asignaturas, y los horarios de los exámenes.
- **eGela,** para acceder al material, para ver las entregas, leer mensajes.

La información que un alumno necesita está dispersa entre estas aplicaciones y muchas veces no está actualizada, no es correcta o no coincide.

EHU Zurekin propone ser una aplicación que ayude a los alumnos a encontrar la información que necesitan rápidamente y que siempre esté actualizada.

EHU Zurekin actúa como una vista previa actualizada, rápida, y con accesos directos a toda aquella información que un alumno necesite. Tendrá conexión con el servidor webPosta para ver los correos. Obtendrá las noticias (RSS) de la EHU para poder estar informado de todo lo que ocurre. Se nutrirá de Gaur o de una base de datos para poder ver el calendario, los horarios, las asignaturas y la información de los docentes (como sus tutorías y despacho) . EHU zurekin estará disponible en inglés, español y euskera.

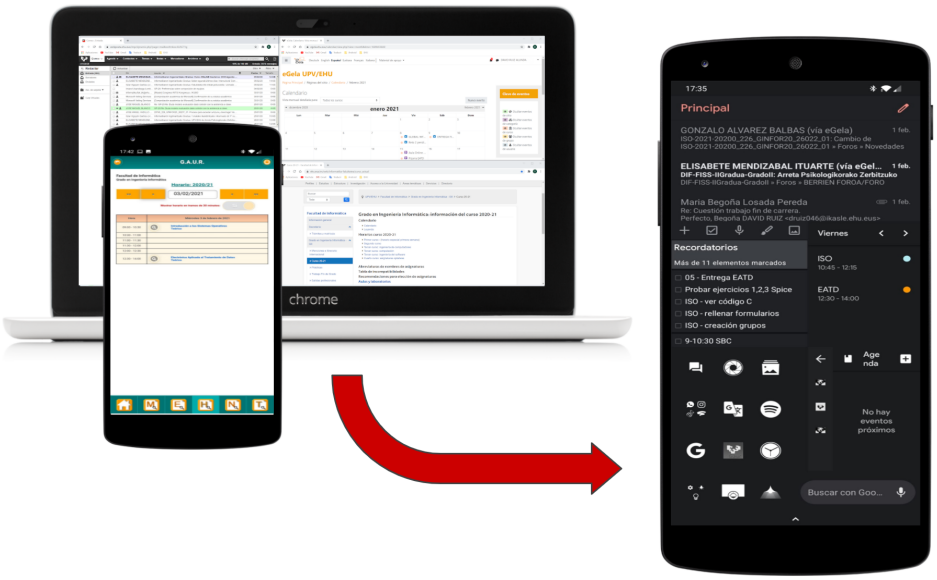


Figura 1. Objetivo de la aplicación[1]

[1] Aplicación demo realizada haciendo uso de Nova Launcher y otras aplicaciones

CONTEXTO Y ANTECEDENTES

La actualidad de los alumnos

Un grupo de alumnos, nada más empezar la carrera, crea un grupo de Whatsapp donde se pone de foto de perfil el horario. Generalmente extraído de las página de la EHU.

A medida que pasan los cursos; aparecen las especialidades, las asignaturas opcionales, empiezan a quedar asignaturas de otros años, y es entonces cuando esta foto de perfil pierde el sentido y se hace complicado saber qué asignaturas tienes y donde.

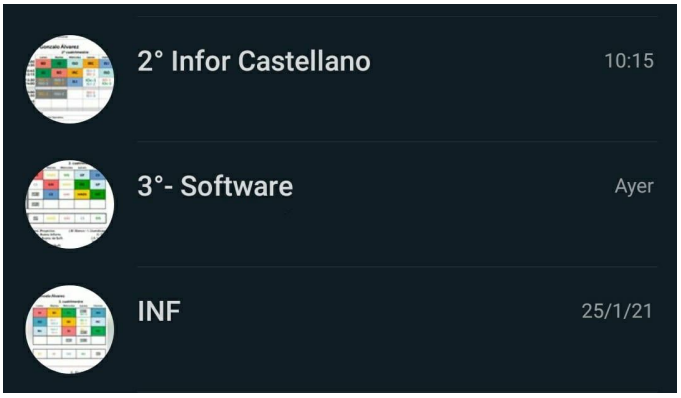


Figura 2. Prueba 1: grupos de Whatsapp

Pocos alumnos saben donde es la siguiente clase. Preguntas como estas se repiten constantemente.

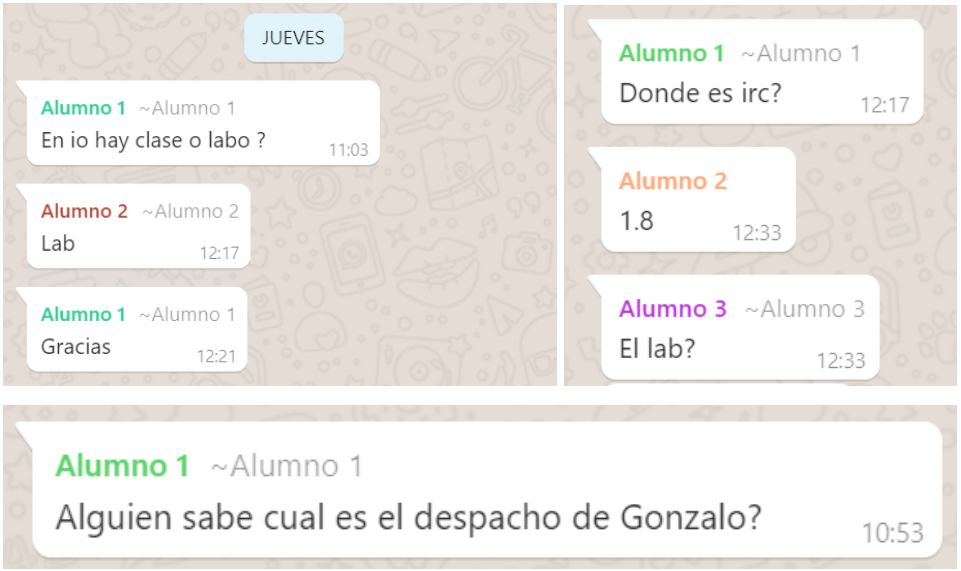
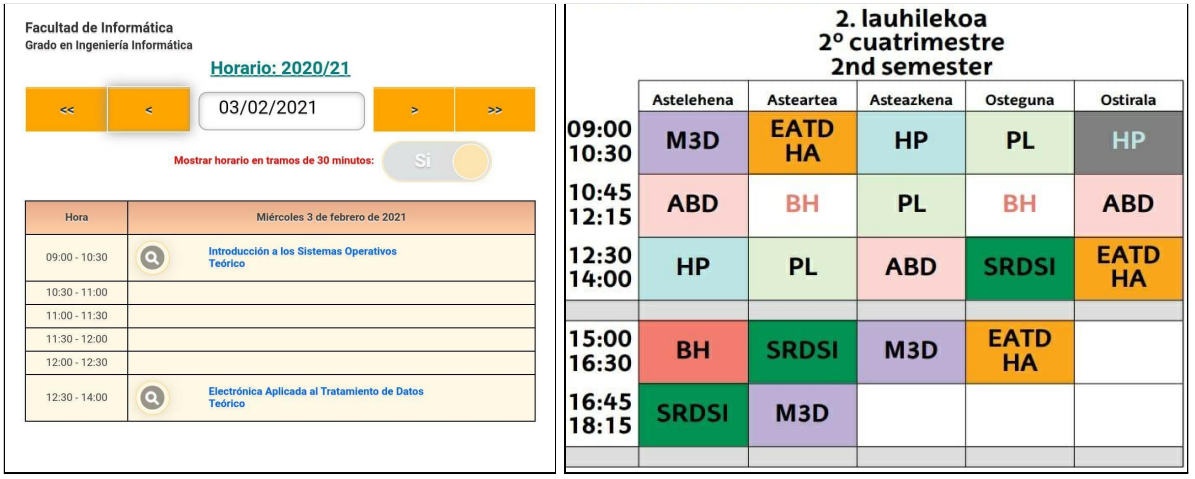


Figura 3. Prueba 2: dudas sobre despachos y lugar de la clase[2]

[2] Los nombres de los alumnos han sido ocultados por su privacidad

A veces, incluso aunque consultemos la aplicación G.A.U.R, la información no está actualizada o no es correcta y por tanto nos vemos obligados a descargar el PDF de horarios de EHU o preguntar por el grupo.



Facultad de Informática
Grado en Ingeniería Informática

Horario: 2020/21

<< < 03/02/2021 > >>

Mostrar horario en tramos de 30 minutos: Si

Hora	Miércoles 3 de febrero de 2021
09:00 - 10:30	Introducción a los Sistemas Operativos Teórico
10:30 - 11:00	
11:00 - 11:30	
11:30 - 12:00	
12:00 - 12:30	
12:30 - 14:00	Electrónica Aplicada al Tratamiento de Datos Teórico

2. lauhilekoa
2º cuatrimestre
2nd semester

	Astelehena	Asteartea	Asteazkena	Osteguna	Ostirala
09:00 10:30	M3D	EATD HA	HP	PL	HP
10:45 12:15	ABD	BH	PL	BH	ABD
12:30 14:00	HP	PL	ABD	SRDSI	EATD HA
15:00 16:30	BH	SRDSI	M3D	EATD HA	
16:45 18:15	SRDSI	M3D			

Figura 4. Prueba 3: errores en G.A.U.R (no coincide)

En la imagen de la izquierda vemos que el miércoles 3 de febrero tengo clase de ISO (asignatura de segundo) de 9:00 - 10:30 y EATD de 12:30 - 14:00. Esto no es cierto. Como vemos en la imagen de la derecha; el PDF con el horario muestra que el miércoles no tengo EATD y no es un día especial (horario de jueves, ni viernes, etc).

El despacho del profesor casi nunca está en G.A.U.R. Si el profesor no lo tiene puesto en eGela, buscar cual es su despacho es tarea complicada y hay que recurrir al grupo de WhatsApp o enviarle un correo al profesor.

Un alumno pocas veces usa G.A.U.R y no quiere andar descargando los pdfs con los horarios. Muchos tienden a crear su propio horario con aplicaciones de terceros en su móvil.

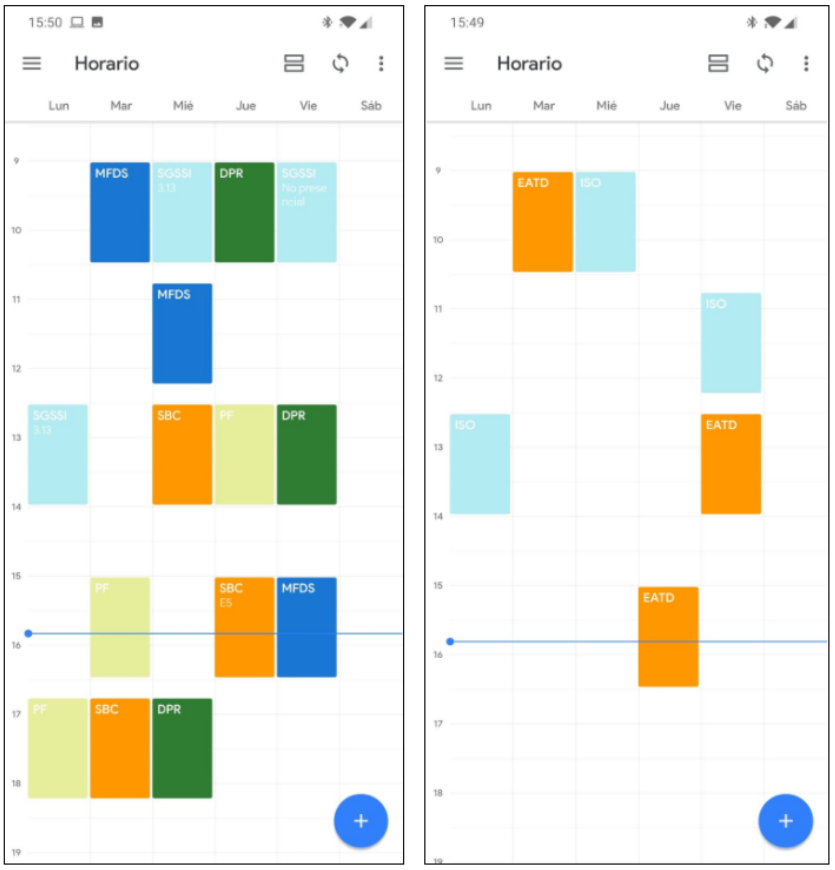


Figura 5. Prueba 4: mi aplicación de horario[3]

Por ejemplo, en mi caso, gracias a una aplicación hice los horarios para aclararme ya que hicieron cambios de última hora.

[3] Para realizar el horario se ha usado la aplicación Agenda Escolar

Servicios actuales de la EHU

Correo vía web (webposta, correow)

Es el servicio web que permite a los profesores, alumnos y demás miembros de la universidad enviar y recibir mensajes.

Solemos usar esta aplicación para enviar preguntas a los profesores, para consultar temas de matrícula, etc.

A este mail llegan los correos/respuestas de los profesores, ofertas de empleo, publicidad que envían desde la universidad y otros tipos de mensajes.

Es obligatorio para un alumno redirigir este correo de la ehu a otra aplicación como gmail para enterarse de los correos que llegan o, si no, entrar cada poco en la web.

eGela

Es el servicio, basado en moodle, que más utilizan los alumnos. Con esta aplicación podemos acceder a las asignaturas; ver las entregas a realizar, ver el material de la asignatura, etc. También es otro método de envío de mensajes.

Algunos profesores no usan esta plataforma y, a veces, nos encontramos con que prefieren usar otra página web y tenemos que apuntar el link de esta.

También, a fecha de hoy, los enlaces para videoreuniones (generalmente mediante BBC), se ponen en eGela.

Se puede descargar la aplicación móvil Moodle y conectarla con el eGela de la universidad.

G.A.U.R

Es una aplicación móvil donde podemos ver nuestro horario, las asignaturas, matrícula, las notas, etc. También tiene una página web con más información.

EHU

Es el servicio web donde se publican los horarios, el calendario, las aulas de las asignaturas y las fechas de los exámenes. (Generalmente en formato PDF).

A parte de esto, la página web EHU tiene muchas más funciones, pero un alumno, generalmente solo usa estas. También se suele usar para temas de matrícula, información acerca de Erasmus, créditos, TFG, etc.

OBJETIVOS DEL PROYECTO

Motivación

En tercero realicé la asignatura Interacción persona computador, IPC, con Begoña. En esta asignatura hubo una entrega opcional que proponía crear la interfaz de un móvil para la universidad. Fue un proyecto que me gustó mucho realizar. El proyecto que realicé se puede ver en Youtube mediante este link:

https://www.youtube.com/watch?v=9S75xWBu1XQ

En dicho proyecto proponía una aplicación tipo launcher que contaba con dos modos: una launcher normal sin limitaciones (como el de tu propio dispositivo) y otro con limitaciones (sin distracciones, para que el alumno se centre) donde se podía ver el calendario, los correos, las asignaturas para cada día, etc. Este launcher contaba con acceso rápido a eGela y con accesos a las aplicaciones básicas del dispositivo. Esto, para minimizar las distracciones.

Esta "aplicación" fue realizada usando Adobe XD. Realmente, era un simple diseño no funcional. No se podía usar en un dispositivo real.

Durante la carrera fui probando distintas aplicaciones para gestionar la información. Creaba launchers mezclando muchas aplicaciones para intentar tener toda la información siempre a la vista.

Tener siempre la información delante en un launcher supone una gran diferencia frente a tener que entrar a una aplicación o a varias para ver dicha información. Es más cómodo, rápido y accesible.

Esto, junto a los distintos problemas relacionados con las aplicaciones de la EHU mencionados anteriormente, me motivó a llevar a cabo este proyecto.

Planificación inicial

Propuesta y objetivo inicial

Agrupar en una misma aplicación, a modo de vista previa, el contenido de las diferentes plataformas.

Figura 6. Desglose de tareas

Últimos correos

Crear una pequeña ventana donde se muestren los últimos correos recibidos en Correow.

Debe ser una visualización simple que contenga el nombre del emisor del correo y el asunto de los últimos 5-10 correos. (También se podría incluir la imagen de perfil del emisor, parte del cuerpo del correo, etc)

Horario

Crear una pequeña ventana donde se muestren las asignaturas que toca cada día junto con el aula/lab donde se impartirá.

Tareas

Muestras las últimas tareas y entregas para cada asignatura. También podrían añadirse enlaces.

Otros Noticias EHU, enlaces, entregas, etc

La propuesta inicial es realizar una aplicación tipo launcher donde podamos tener toda la información que necesitamos a primera vista y con accesos rápidos a estas.

Estimado de tiempos inicial

Comencé realizando los diferentes módulos de la aplicación; parte correo, parte horario, noticias, etc ya que el realizar el launcher iba a suponer mucho tiempo y era mejor centrarse primero en la funcionalidad de la aplicación.

Previamente a esta planificación estudié e intente tener una idea clara de las tareas que debería realizar y el tiempo que me llevaría implementarlas. Por esta razón, la primera iteración, que veremos más tarde, es simplemente una primera visión del proyecto.

Estimado de tiempos

	Tarea	Estimado
Instalación	• Android Studio y conf. Dispositivos • Librerías.	5h 1h
Seguridad	• Copias de seguridad	1h
Formación	• Android básico • Correow • RSS • Moodle • G.A.U.R • Launcher	40h 5h 5h 5h 5h 10h
Implementación	• Login • Correow • RSS • Horario • Launcher	1h 30h 30h 30h 60h
Diseño	• Diseño aplicación	30h
Documentación	• Memoria • Presentación	60h 20h
Reuniones	Reuniones presenciales/BBC y correos	10h

Tabla 1. Estimación tiempos inicial

Como se puede apreciar, crear el launcher me suponía demasiado tiempo. Barajé la idea de prescindir de los RSS pero esta era una tarea que sabía que podía realizar y de complejidad baja. El launcher, en cambio, seguramente no pudiese acabarlo. Esto supondría una mala inversión del tiempo para el trabajo.

Decidí dejar apartada la idea del launcher y dejarlo como una mejora a futuro.

Desglose de tiempos

Instalación

Instalar el entorno android studio y configurarlo para usar dispositivos físicos. Instalar sobre él las librerías necesarias.

Seguridad

Realización de copias de seguridad periódicas de la aplicación. Tanto localmente como en la nube.

Formación

Adquirir los conocimientos básicos de Android y Android Studio.

Conocer los distintos servicios de la EHU e investigar formas de implementarlos en Android. Estudiar RSS, protocolos de envío y recepción de emails, investigar los servicios web de moodle, etc.

Implementación

Llevar a cabo las distintas tareas de la propuesta inicial. (BACKEND)

En este apartado también se incluye el comentar el código.

Diseño

LLevar a cabo el diseño de la aplicación (FRONTEND)

En este apartado también se incluye la traducción.

Documentación

Documentar la memoria y la presentación.

Reuniones

Reuniones periódicas con Begoña. Estimados unos 30-45 minutos por reunión.

Iteraciones

Cada iteración durará aproximadamente 2 semanas donde se realizará una reunión al inicio de cada una.

Gestión de riesgos

Las asignaturas requieran de mucha dedicación

Tengo 3 asignaturas en este segundo cuatrimestre. Estas asignaturas requieren de un tiempo de dedicación que hasta la fecha desconozco. No obstante, al ser solo 3 asignaturas hay tiempo para realizar el TFG. El **nivel de riesgo** es **bajo**.

Para intentar prevenir un cuello de botella de trabajos y exámenes habrá que llevar todo al día. En el caso de que este desastre ocurra no quedará otra que aplazar el proyecto.

No ser capaz de realizar la aplicación

Como he comentado, al partir de cero en Android hay muchos problemas que pueden surgir. Puede que tarde más de lo previsto en entender el entorno y puede que se me complique más de lo previsto la implementación. Considero que tiene un **nivel de riesgo medio** ya que parece un entorno similar a windows builder de eclipse usado en IS1.

Tener problemas con los dispositivos

Puede que los dispositivos que uso para realizar las pruebas fallen y no pueda usarlos.

El nivel de **riesgo es bajo** sin embargo la **repercusión es bastante grande**. Es muy poco probable que teniendo dos dispositivos fallen los dos . Aún así, si estos fallan, será complicado llevar a cabo el proyecto dado que mi ordenador no es lo suficientemente potente para lanzar simulacones de dispositivos y adquirir nuevos smartphones es complicado.

Al tener que conectar varias veces el smartphone al ordenador, vía mini USB, un potencial problema que puede acarrear esta acción es que la entrada USB se estropee. Es un problema que supondría buscar otro método para instalar la aplicación en el dispositivo y sería mucho más lento. Para prevenir el problema intentaré hacer un uso delicado de los dispositivos.

Contagio por COVID-19

En mi caso, el nivel de **riesgo es bajo.** Si el contagio no llega a nada grave, seguiré realizando el trabajo manteniendo reuniones vía BBC.

En el caso de que sea grave no quedará otra que aplazar el proyecto.

Estudio de viabilidad

Al ser un proyecto propuesto por mi y partir desde 0 es complicado hacerse una idea inicial de la viabilidad del proyecto.

Android Studio tiene mucha documentación y una gran comunidad detrás que ayuda a que la herramienta crezca. Esto es una gran ventaja a la hora de implementar la aplicación.

Se concluye que este proyecto es viable ya que consiste en utilizar todos los conceptos aprendidos durante la carrera en un entorno parecido. Requerirá de mucha investigación pero con los conocimientos que he adquirido durante la carrera se puede realizar.

HERRAMIENTAS, LENGUAJE Y DISPOSITIVOS

Android Studio

¿Qué es?

Es un entorno de desarrollo para aplicaciones android.

Se pueden realizar virtualizaciones de smartphones y probar las aplicaciones ahí. Los dispositivos virtuales se usan para hacer pruebas en dispositivos antiguos, nuevos, con pantallas diferentes, etc sin necesidad de tenerlos físicamente. No obstante, esto requiere un ordenador potente y por tanto, yo solo podía trabajar con dispositivos físicos.

Es un entorno muy potente, muy bien hecho e intuitivo.

¿Por qué lo elegí?

Elegí android studio principalmente porque mi idea inicial era la de realizar un launcher.

Otra razón fue porque yo no tenía dispositivos IOS con los que realizar pruebas.

La última razón fue porque Android Studio tiene un gran grupo de usuarios detrás, hay mucha información en internet y su página de documentación es muy buena.

000webhost

¿Qué es?

Es un servicio de hosting web donde puedes crear tu página web, tu base de datos, almacenar información, etc.

Cuenta con el servicio phpmyadmin para manejar la administración de MySQL.

Tiene una interfaz muy intuitiva con muchas opciones.

¿Por qué lo elegí?

Lo elegí principalmente por ser el hosting que más hemos usado durante la carrera. Ya tenía el conocimiento para trabajar con él. Además, esta página web, ofrece un servicio de hosting gratis, eso sí, muy lento y con mucha publicidad.

Lenguaje Java

¿Qué es?

Es un lenguaje de programación.

¿Por qué lo elegí?

Lo elegí principalmente por haber trabajado durante toda la carrera con este lenguaje. La otra opción era kotlin, pero al no tener conocimiento sobre este lenguaje y que la información era más escasa decidí usar java.

Otros lenguajes

PHP: Es un lenguaje de código abierto adecuado para el desarrollo web. Lo usaré principalmente para enlazar el código de Android Studio con la base de datos.

XML: Gran parte del código de una aplicación en Android Studio ha de estar escrito en este lenguaje:

- Lo que vemos por pantalla: los campos para introducir texto, los botones, las imágenes, etc.
- La traducción, los temas de la aplicación y colores, etc.

SQL: Un lenguaje usado para la administración de bases de datos relacionales.

Oneplus 6T y Huawei P9

Dispositivos donde realicé las pruebas.

OnePlus 6T

Principalmente trabajé con este smartphone ya que es un dispositivo que trabaja estrechamente con google a la hora de implementar el software. Este dispositivo cuenta, a fecha de hoy, con Android 10. Este dispositivo cuenta con una pantalla de ratio 19,5:9.

Huawei P9

Usé este dispositivo principalmente para verificar el funcionamiento en otra versión de android; android 5.1 y en otro tipo de pantallas; ratio 16:9

GitHub y Google Drive

Almacenamiento en la nube:

Son dos herramientas las que he utilizado para el almacenamiento en la nube: GitHub y Backup and sync from google.

El servicio de google permite tener una carpeta o varias enlazadas a su nube. Cada cambio que se hace en el proyecto se guarda directamente en la nube constantemente. Lo usé principalmente para tener el proyecto guardado continuamente en la nube. Esto es un buen seguro para, por ejemplo, cuando haya un corte de luz.

Github, en cambio, lo usé para el control de versiones. Cada vez que finaliza una iteración o una parte importante hacia commit y push. Encima, todo el código es visible desde su plataforma de forma más cómoda.

Procesador de texto

Utilicé Google Docs para realizar la memoria. Google Docs también dispone de guardados automáticos en la nube pero tiene muy pocas opciones. Los índices de imágenes y tablas por ejemplo hay que realizarlos manualmente, el pié de imagen también, etc...

METODOLOGÍA

Al no tener objetivos bien definidos se optó por utilizar una metodología ágil y centrada en el usuario. Al principio no sabíamos que se iba a poder realizar y que no.

Al principio de cada reunión proponía las nuevas ideas a Begoña. Después yo realizaba un pequeño análisis de la tarea a realizar. Si la veía factible, comenzaba a diseñar. Una vez tenida la idea clara comenzaba con la implementación y construcción de la tarea en Android Studio. Al finalizar la implementación realizaba unas pequeñas pruebas y en la próxima reunión lo enseñaba. Estas iteraciones tenían un periodo de 2 semanas.

Durante este proceso, se realizaban pruebas con los compañeros. Tras ellos probar la aplicación, me comentaban sus experiencias de uso. De esto obtuve la opinión de cada alumno; lo que les gustaría ver, lo que no, otras ideas, etc. Tras esto, realizaba mejoras a la aplicación.

ANÁLISIS Y DISEÑO

CASOS DE USO

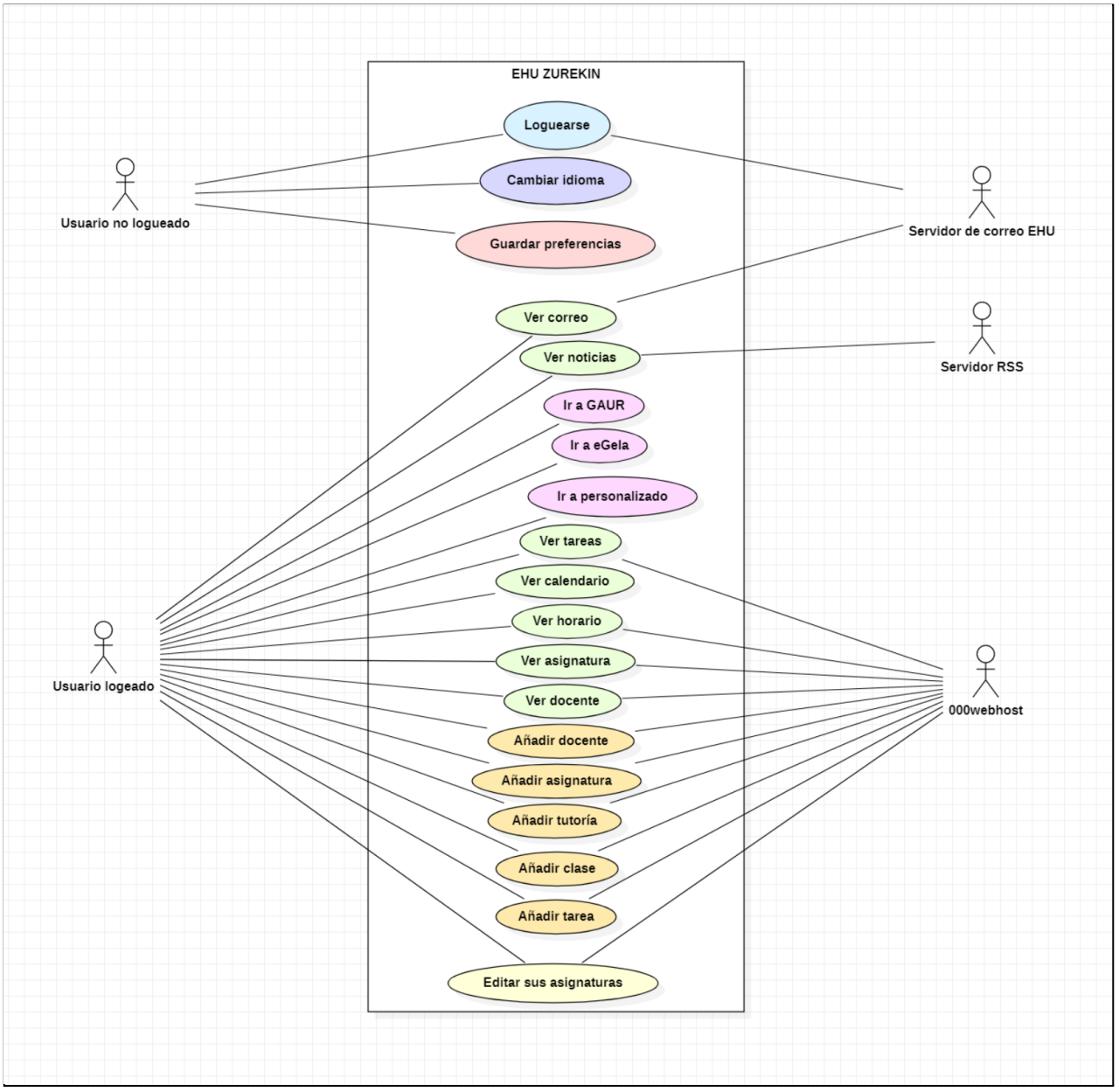


Figura 7. Diagrama de casos de uso (StarUML)[4]

[4] StarUML es un herramienta de modelado que sigue los estándares UML y MDA

REQUISITOS FUNCIONALES Y NO FUNCIONALES

Loguearse

Un alumno debe poder loguearse en la aplicación. Tendrá un espacio para el LDAP y otro para su contraseña.

Cambiar de idioma

Un alumno debe poder escoger su idioma. Por ahora se implementará la aplicación en Español, Inglés y Euskera. Habrá una opción extra para seleccionar automáticamente el idioma (entre esos tres) según el idioma del smartphone.

Guardar preferencias

Se guardará en memoria del smartphone (caché) los datos introducidos para que sea más fácil loguearse la próxima vez.

Ver correo

Se mostrarán los 10 últimos correos y al clicar sobre uno se verá una vista amplificada de ese correo. Se tendrá que actualizar automáticamente.

Ver noticias

Se mostrarán las últimas noticias de las EHU obtenidas de los RSS. Al clicar sobre una notícias deberá enviar al navegador y ver la noticia completa.

Ver tareas

Se mostrarán las últimas tareas añadidas por los otros alumnos. Solo se deberán mostrar las tareas que correspondan con las asignaturas del alumno logueado.

Estas tareas podrán ser enlaces, por ejemplo a un grupo de whatsapp para que todos los alumnos tengan acceso a ese grupo.

Ver calendario

Se mostrará el calendario de informática. (El pdf de calendario que está en la web EHU)

Ver horario

Se mostrarán las asignaturas que tocan cada día, su hora de inicio/fin y su aula/laboratorio.

Ver asignatura

Se mostrará información ampliada de una clase/asignatura. Por ejemplo: links a BBC, el docente que la imparte, etc.

Ver docente

Se mostrará su correo (con la posibilidad de copiarlo), su despacho y sus tutorías.

Ir a eGela

Enlace directo a eGela mediante un botón.

Ir a GAUR

Enlace directo a EHU mediante un botón.

Ir a personalizado

Posibilidad de editar botones de acceso directo.

Esto permitirá tener accesos rápidos a los enlaces que queramos. Por ejemplo, a nuestra página de 000webHost.

Editar asignaturas

Poder añadir o eliminar asignaturas para ver su información.

Añadir docente

Añadir información acerca de un docente.

Añadir asignatura

Añadir información acerca de una asignatura.

Añadir tutoría

Añadir información acerca de una tutoría.

Añadir clase

Añadir información acerca de una clase.

Añadir tarea

Añadir una tarea o enlace para que los demás alumnos de esa asignatura la puedan ver.

DISEÑO BASE DE DATOS

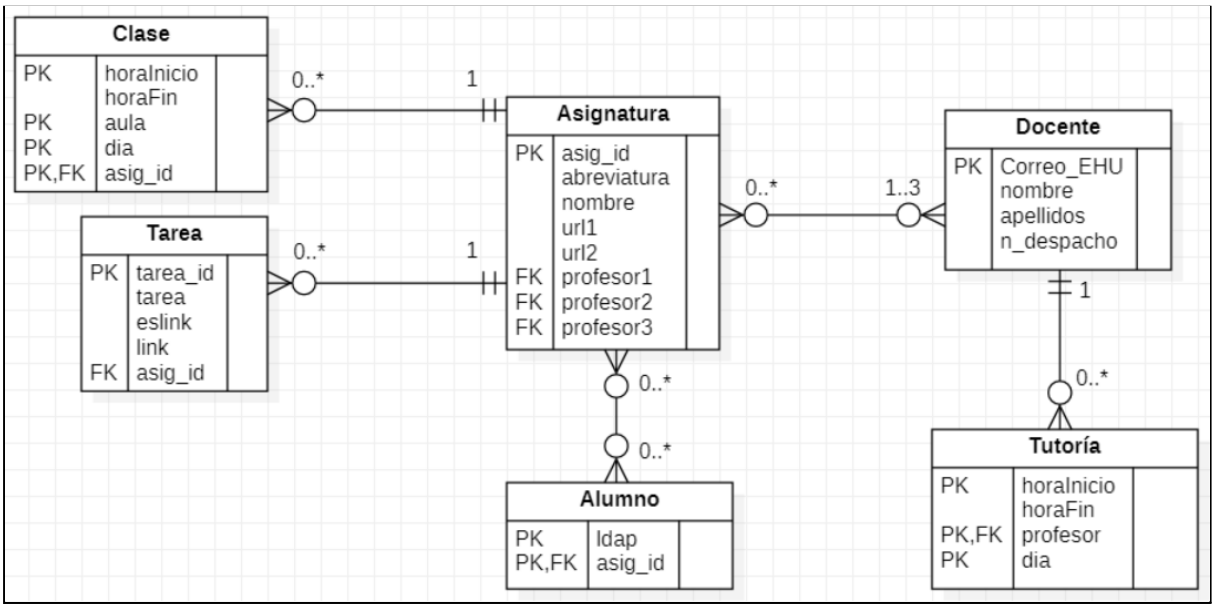


Figura 8. Modelo entidad relación (StarUML)

Clase: Una clase, en este contexto, es un margen de tiempo donde se imparte una materia. Tiene una hora de inicio (horaInicio) y una hora de fin (horaFin) en un determinado día (dia). Se imparte una asignatura en ese tiempo (asig_id). Esta clase se realiza en un aula o laboratorio (aula).

Tarea: Las tareas son añadidas por los usuarios. Pueden ser links (eslink), por ejemplo para entrar en grupos de Whatsapp. En este caso se debe añadir la url (link). El texto a introducir (tarea) es el ”quehacer” y en el caso de que sea un link, el titulo del link. El id es autoincremental.

Asignatura: Se guarda el título de la asignatura (nombre) y la abreviatura de este (abreviatura), además se guarda el profesor o profesores que la imparten (profesor1). También se pueden añadir hasta 2 links (por ejemplo para BBC).

Docente: Del docente/profesor se almacena su correo electrónico (Correo_EHU), su nombres (nombre) y sus apellidos (apellidos). Además, también se guarda su número de despacho (n_despacho).

Tutoría: Tutorías impartidas por los profesores (profesor). Se guarda su hora de inicio (horaInicio), Hora fin (horaFin) y el dia (dia).

Alumno: Se almacena su número de LDAP (ldap) y las asignaturas que elige (asig_id).

IMPLEMENTACIÓN

Todo el código es accesible mediante el enlace de github:

https://github.com/davidruizalunda/EHU-UPV-AndroidApp.git

CONTEXTO

Para entender los siguientes apartados explicaré brevemente cosas a tener en cuenta de Android y Android Studio.

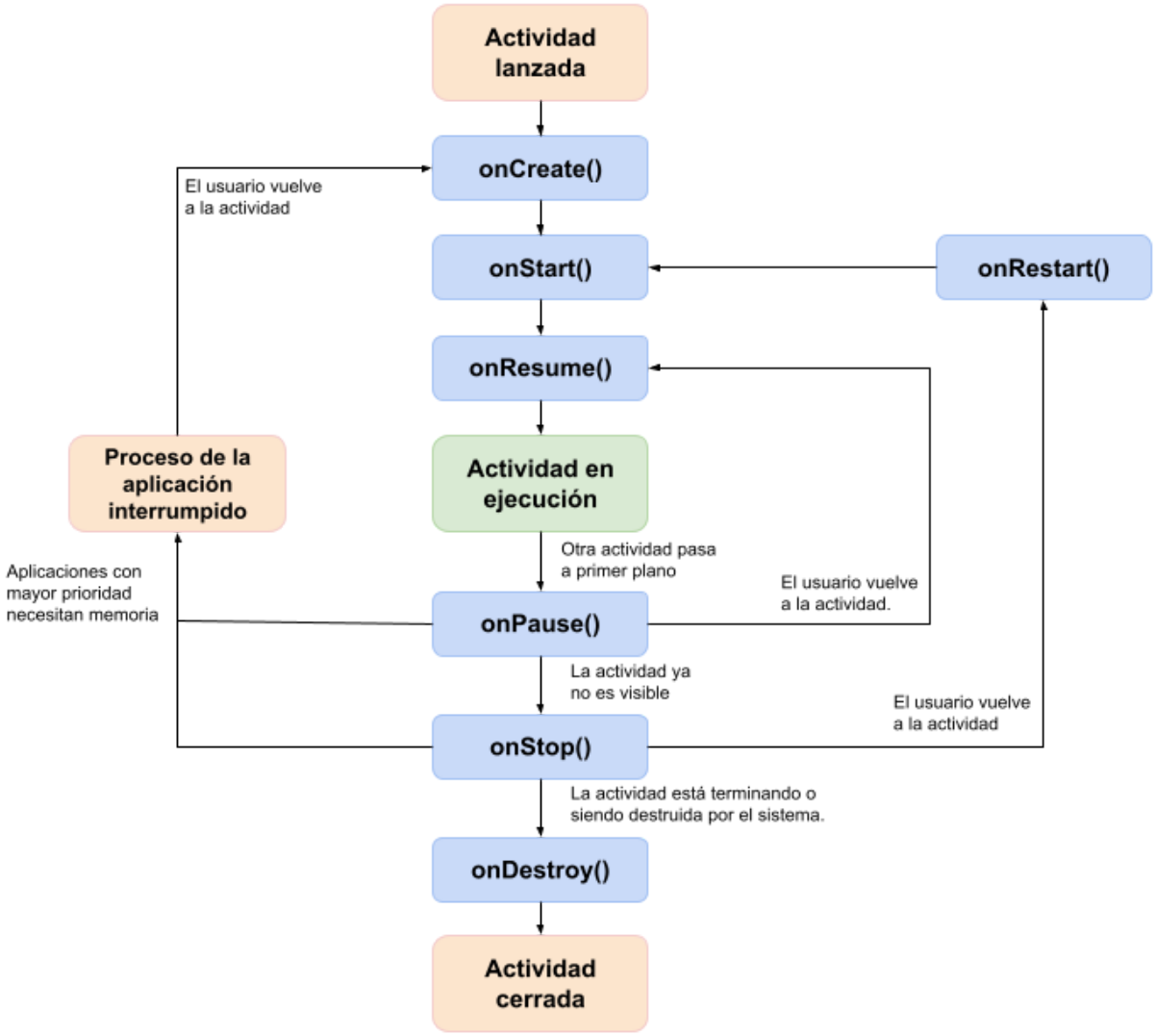


Figura 9. Ciclo de vida de un activity

¿Qué es una actividad?

Un activity (o actividad) es una pantalla o ventana de Android. Ahí podemos ver elementos, interactuar con ellos, etc.

Puede leer el siguiente apartado si quiere entrar en profundidad:

Introducción a las actividades (de https://developer.android.com/)

https://developer.android.com/guide/components/activities/intro-activities?hl=es

Ciclo de vida de un activity

Cuando un activity (o actividad) se lanza:

1. Se crea el activity (onCreate)
2. Se inicia el activity (onStart)
3. Se hace visible el activity (onResume)

En ese momento la actividad está en ejecución y podemos interactuar con ella. A veces otra actividad pasa a primer plano y entonces:

4. Se pausa el activity (onPause)
5. Se para y se hace invisible (onStop)

También hay que tener en cuenta lo siguiente:

- Cuando minimizamos la aplicación estamos en: onStop
- Cuando cerramos la aplicación (tras minimizar o de otra forma) : onDestroy
  6. onDestroy destruye la actividad y cierra el ciclo de vida del activity.
- Cuando pasamos de la actividad A a la B (en la misma aplicación) y después volvemos a la A no se ejecuta el método onCreate nuevamente. El activity ya está creado y el método que se ejecuta es onStart. (El activity A se minimiza al pasar a B)

En el código que he escrito, no aparecen tantos métodos. El único obligatorio a implementar es onCreate. Los demás se obtienen a partir de la clase padre. No obstante, hay una excepción en el código en donde si hacemos uso de otros métodos y de ahí que explique el ciclo de vida de un activity.

En nuestro proyecto nos basta con esto y con entender por encima el diagrama. Pero, si desea leer más acerca del ciclo de vida de un activity (actividad), puede leer lo siguiente:

Ciclo de vida de una actividad (de https://developer.android.com/)

https://developer.android.com/guide/components/activities/activity-lifecycle

¿Cómo se crea un activity?

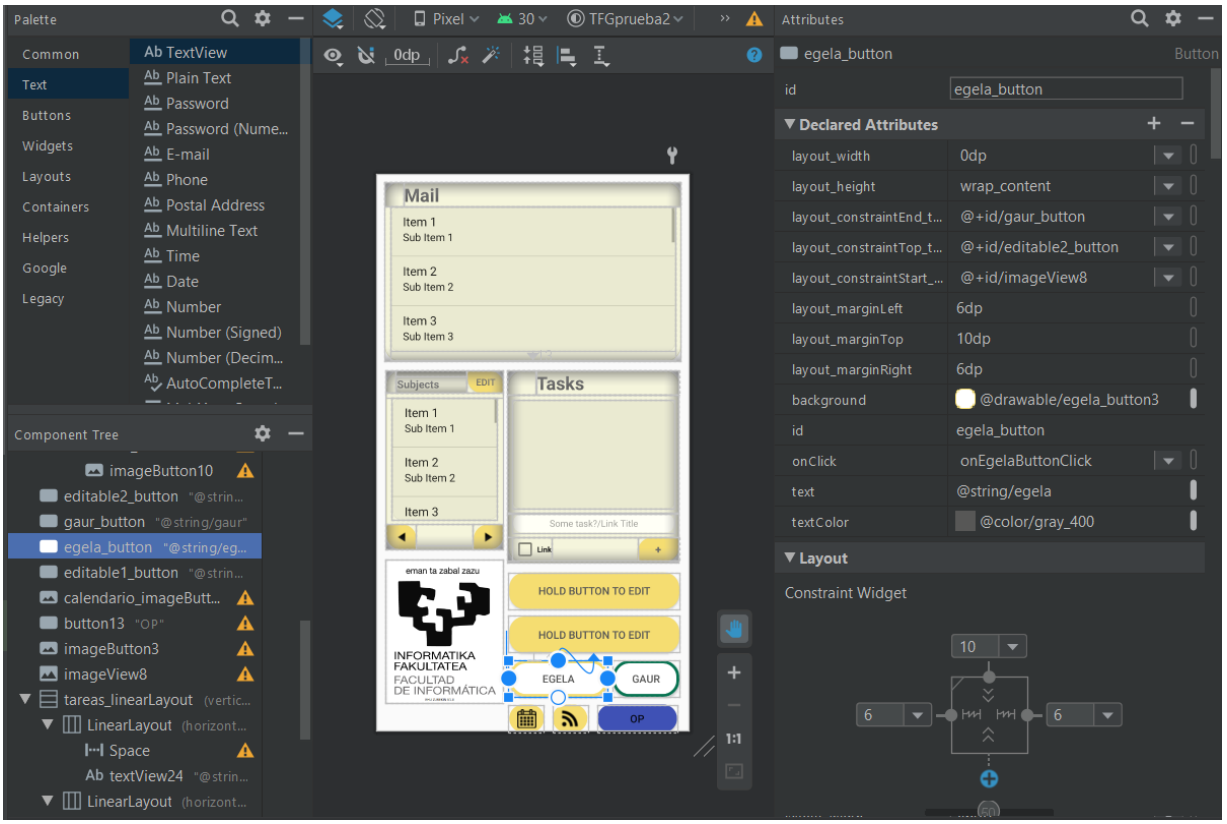


Figura 10. Herramienta de diseño en Android Studio

Normalmente se hace uso del editor (vista diseño) que nos proporciona android studio similar al que vimos en IS1 con windows builder. Tenemos una paleta donde seleccionar elementos, un menú con atributos para cada elemento, una vista previa, etc.

Todo lo que hacemos con esa herramienta también se puede realizar a mano en xml. Una vez te haces con el formato es más sencillo realizar los cambios aquí que en la vista de diseño.

```
<Button
    android:id="@+id/egela_button"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="6dp"
    android:layout_marginLeft="6dp"
    android:layout_marginTop="10dp"
```

Figura 11. Diseño en xml

Estos son algunos de los atributos que he usado:

- android:layout_margin...="6dp"
  Para dejar márgenes en cada elemento.
- android:background="@drawable/egela_button3"
  Para darle diseño a algunos elementos.
- android:textColor="@color/gray_400"
  Para elegir un color.

Figura 12. Vista Blueprint

Vista BluePrint

Esta vista se usa para ver la relación de cada elemento dentro de un activity. A veces no se puede ver realmente lo que hay en el activity si no se usa esta vista.

Por ejemplo cuando un elemento no aparece hasta que ocurre algún evento. En la vista de diseño no lo veremos.

Para colocar todo correctamente es una buena herramienta.

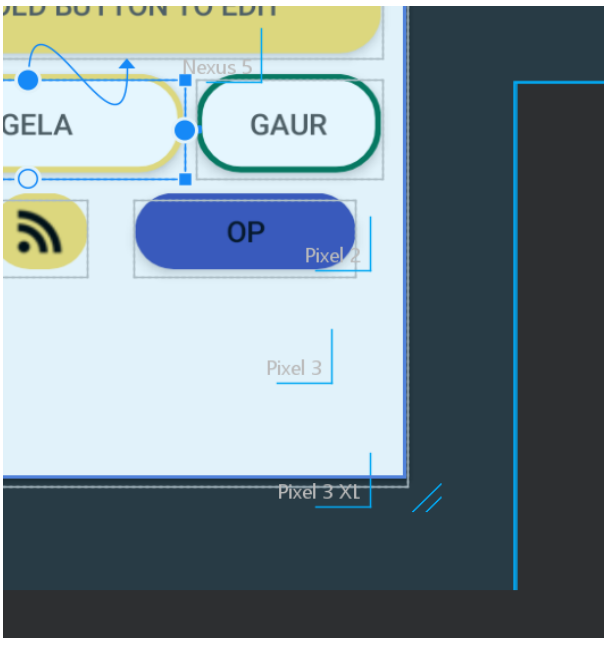


Figura 13. Cambio de tamaño de pantalla

Distintos tamaños de pantalla

Un truco en android studio, tanto en la vista diseño como blueprint, es cambiar el tamaño de la pantalla para ver como queda usando esas dos rayitas.

Como podemos ver en la imagen de la izquierda, apreciamos que ponel: Nexus 5, Pixel 2, Pixel3, Pixel 3XL. Dispositivo de la marca de Google con distintos tipos de pantallas.

Es complicado conseguir que se vea correctamente la aplicación para todo tipo de pantallas.

Temas a tener en cuenta en la implementación

Hilos

Es importante trabajar usando hilos de ejecución. Las aplicaciones Android en su hilo principal cargan todo el Activity (dibujan la pantalla) y otros procesos pesados. Si escribimos todo el código en el hilo principal se satura y deja de responder la pantalla.

A veces podemos darnos cuenta de esto viendo la consola, pero muchas otras veces no.

```
I/Choreographer: Skipped 53 frames!  The application may be doing too much work on its main thread.
I/OpenGLRenderer: Davey! duration=978ms; Flags=0, IntendedVsync=9247998959222, Vsync=9248882292520, O
```

Figura 14. Aviso de hilos

Es necesario crear varios hilos. Podemos usar clases del tipo AsyncTask o instanciar la clase Thread.

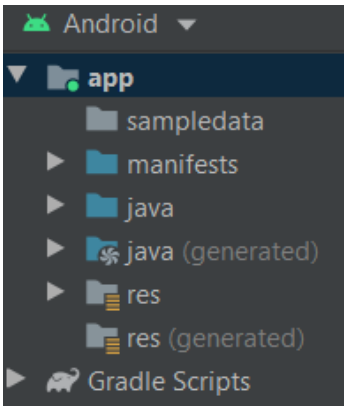
Si se instancia la clase Thread las funciones pesadas deben llamarse en el método run y después volver al hilo principal. Desde un hilo que no sea el principal no se pueden hacer cambios en la pantalla.

Guardar datos persistentes en memoria

Para poder recordar el ldap, la contraseña y la información de los editables.

Para esto se ha usado la clase “SharedPreferences”. Se pueden guardar varios tipos de datos como Strings, int, Booleanos, etc. Si se borra el almacenamiento y la caché de la aplicación se borran estos datos.

ESTRUCTURA PROYECTO EHU ZUREKIN



En la imagen de la izquierda podemos ver cómo se dividen las carpetas internas en Android Studio.

Dentro de **manifests** se escriben los permisos que va a necesitar la aplicación; el de internet, por ejemplo y las preferencias básicas de la aplicación; como su nombre, el estilo a usar, el icono a usar, etc.

En el apartado **Gradle Scripts** principalmente lo uso para importar las librerías.

Figura 15. Estructura Android

Dentro de **Gradle Scripts > build.gradle** se han añadido las librerías necesarias.

Para javamail:

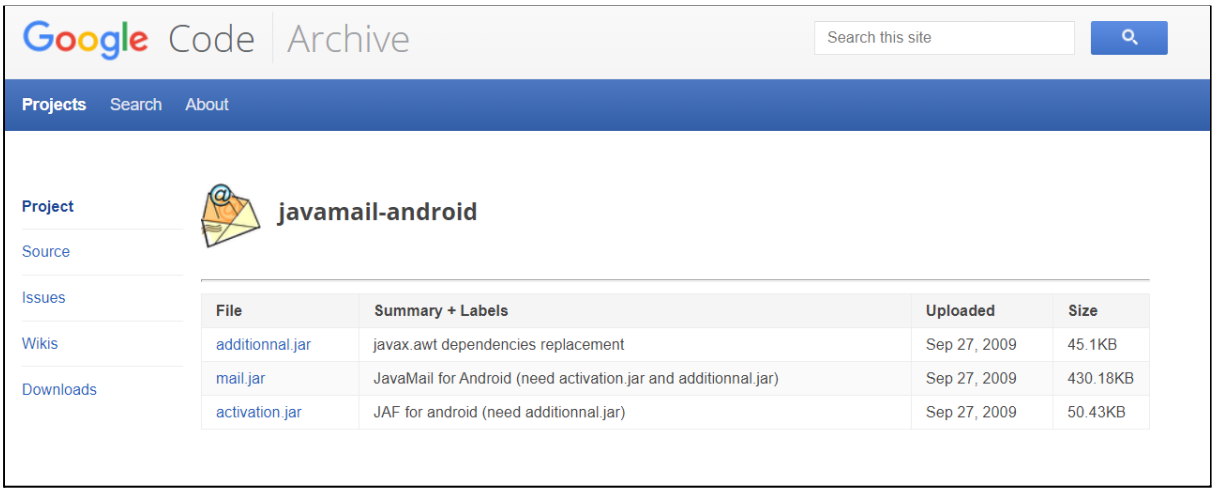


Figura 16. Librerías javaMail

También se ha hecho uso de la librería jsoup (java HTML parser).

Las librerías que necesitan tener el .jar en el proyecto se añaden en la carpeta libs de project. Después se vuelve a la vista android, click derecho sobre app y click en open module settings. En el apartado “dependencies” podremos ir añadiendo las librerías.

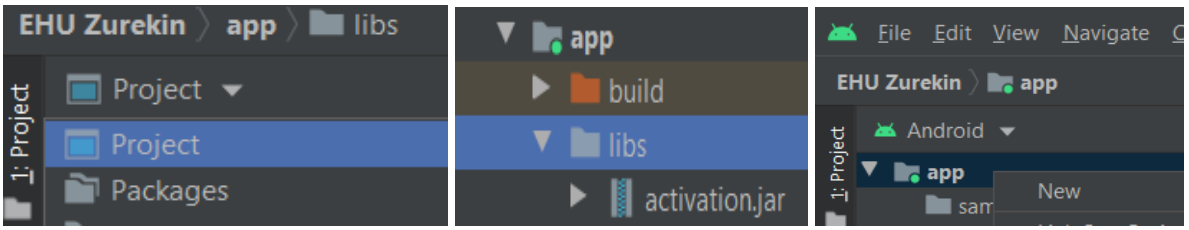
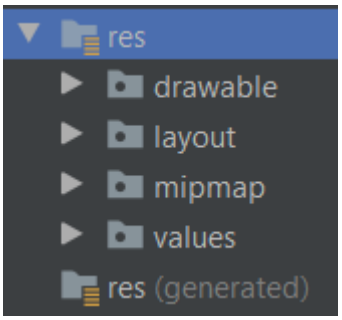


Figura 17. Importación de librerías

ESTRUCTURA FRONTEND EHU ZUREKIN



Carpeta res:

Dentro de esta carpeta se encuentra el frontend de la aplicación, todo lo que el usuario ve y con lo que puede interactuar, las traducciones y demás.

Figura 18. Carpeta RES

drawable:

En esta carpeta se encuentran:

- Todas las imágenes e iconos de la aplicación: Los iconos de gaur y de la ehu, las fotos de las banderas, las fechas del horario, la foto del calendario, etc.
- Todas las imágenes necesarias para la animación del círculo de progreso.
- Algunos .xml que he creado para poder editar el diseño de los elementos de la aplicación (los botones, los campos de texto, etc).

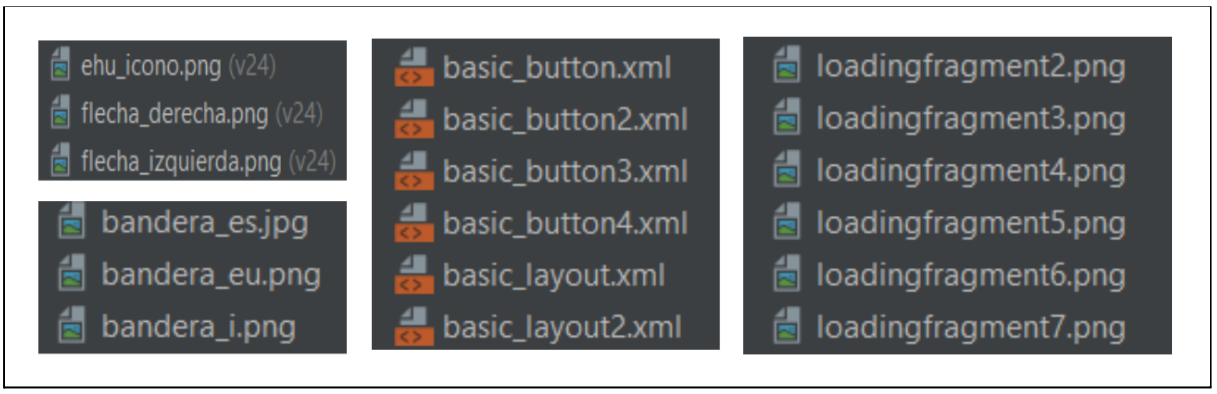


Figura 19. Archivos drawable

layout:

En esta carpeta se encuentran:

- **..._activity.xml:** El diseño de los activitys.
- **my...listview.xml:** Diseño de listViews personalizados.
- **popup_....xml:** Diseño de popups.

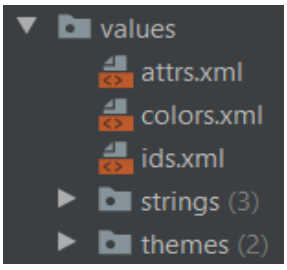
Para realizar estos archivos se pueden realizar a mano (escribiendo xml) o usando las vistas de diseño y blueprint.

Figura 20. Archivos layout

mimap:

Se encuentra el icono de la aplicación y todas sus variantes. (Con forma cuadrada, redonda, etc)

values:



En esta carpeta se incluyen varias subcarpetas y archivos. Los que nos interesan son:

colors.xml: que permite declarar los colores que se van a usar durante la aplicación

strings: aquí se pueden ir añadiendo las traducciones.

Figura 21. Archivos values

En futuras versiones sería aconsejable hacer uso de los demás archivos y crear temas (themes) personalizados.

strings:

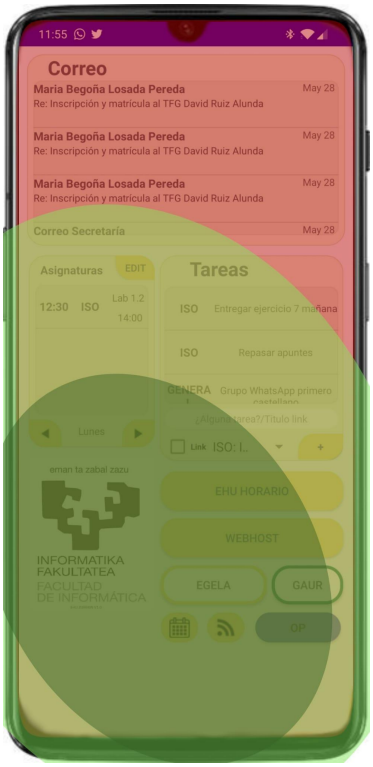
```
<!-- Generales -->
<string name="seleccion_actual">Selección actual:</string>
<string name="empty_button">Vaciar campos</string>
<string name="remover_button">Eliminar</string>
<string name="add_button">Añadir</string>

<!-- Docente -->
<string name="nombreDocente">Nombre</string>
<string name="nombreDocenteHint">Nombre del profesor</string>
```

Figura 22. Ejemplo xml strings

Diseño

Se ha optado por un diseño minimalista que intenta seguir las nuevas tendencias en diseño. Esquinas curvadas, uso de popups y tener todo cerca del pulgar[5].



Actualmente es tendencia que las pantallas vayan creciendo. Desde 2017 que han ido cambiado las resoluciones de las pantallas y sus formatos. Desde el conocido 16:9 y 4.7 pulgadas hasta otros menos comunes como el 21:9 y 6.9 pulgadas.

Es complicado el uso a una sola mano del dispositivo, el pulgar no llega a todos lados. En EHU Zurekin se ha intentado que todos los elementos con los que interactúa el alumno estén lo más cerca posible del pulgar. Como vemos en la imagen, la parte de arriba es inaccesible, la parte media es incómoda y la parte baja es la zona de confort para el usuario.

Esta forma está pensada para que tanto los zurdos como los diestros tengan una complejidad similar de actuación sobre la pantalla.

Figura 23. Al alcance del dedo

Logo

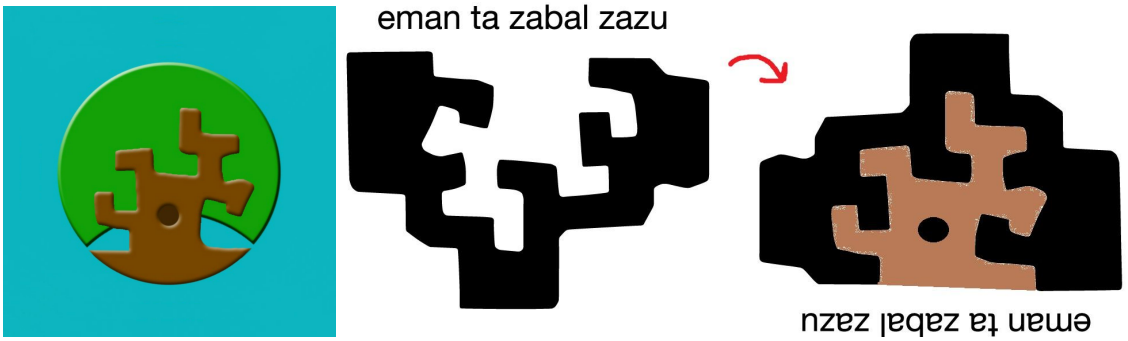


Figura 24. Logo EHU Zurekin

Dentro del recuadro azul podemos ver el logo de EHU Zurekin.

El logo de EHU Zurekin es el interior del logo de la EHU al revés en forma de árbol. Simboliza que las ramas son las aplicaciones (Correow, eGela, Gaur,...) y que EHU Zurekin las agrupa en una sola aplicación.

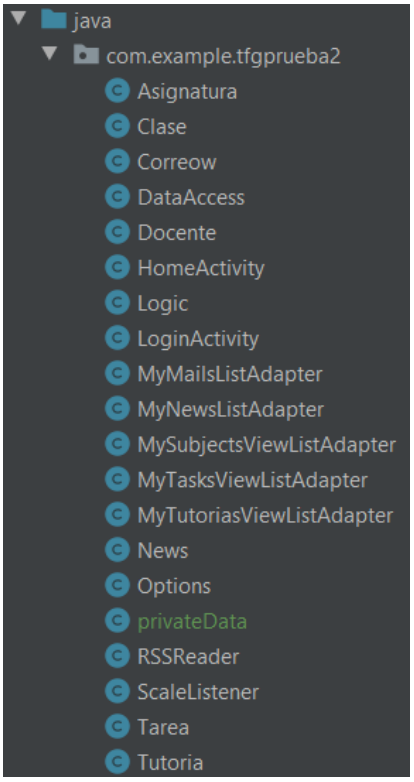
[5] Esta idea la obtuve de la interfaz [One UI de samsung]

ESTRUCTURA BACKEND EHU ZUREKIN

Toda la lógica que hace que EHU Zurekin funcione; tanto la lógica de la propia aplicación como la del servidor.

Android Studio (backend Android)

En la carpeta **java** está la mayoría de la lógica de la aplicación.



LoginActivity: La lógica correspondiente a la visualización del activity Login.

HomeActivity: La lógica correspondiente a la visualización del activity de Home.

Options: La lógica correspondiente a la visualización del activity de Options.

Logic: Funciones básicas que pueden no estar enlazadas con un activity y funciones de paso a la obtención de datos.

DataAccess: Funciones de acceso a los datos.

RSSReader: Clase para la lectura de RSS y sus funciones.

Asignatura, Clase, Docente, Tarea y Tutoria: Son los distintos objetos relacionados con las entidades de la base de datos.

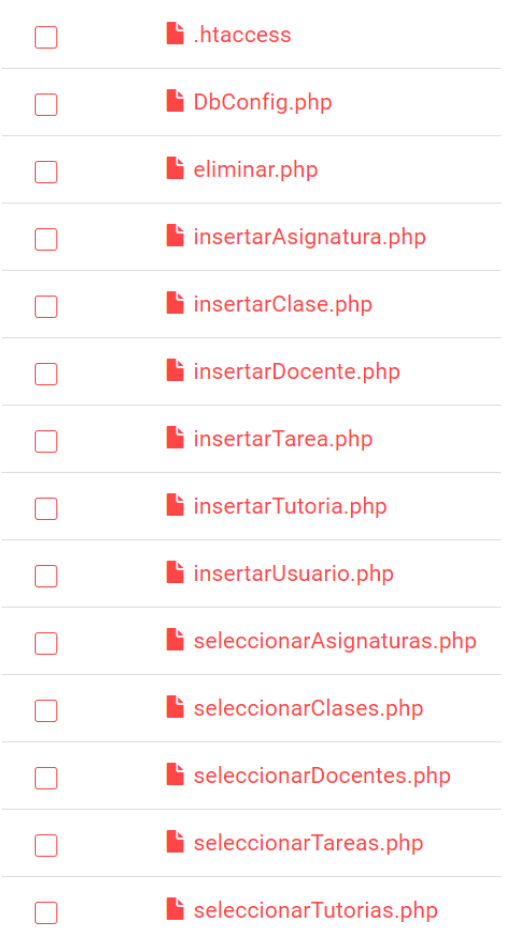
Figura 25. Archivos .java

Correow y News: Son los objetos relacionados con los emails y las noticias.

My..ListAdapter: Es la lógica que permite editar el funcionamiento de los listViews personalizados de Android Studio.

privateData: Es una clase donde, en formato String, se guardan los enlaces necesarios y privados para acceder a la información. Se ha creado esta clase para que estos datos permanezcan privados al hacer commit en gitHub. (Esta clase no aparece en gitHub)

000webhost (backend servidor)



DbConfig.php: Configuración para el acceso a la base de datos, las credenciales para poder acceder; contraseña, nombre de usuario, etc.

eliminar.php: Desde android se manda una sentencia sql de eliminación y este php la ejecuta.

insertar....php: Métodos para insertar cada objeto en su respectiva entidad.

Sería posible cambiar la estructura y crear un insert general donde se puedan mandar sentencias sql de inserción distintas.

seleccionar...php: Método para obtener información de la base de datos.

Figura 26. PHPs del servidor

Si alguno da error devuelve un mensaje que es leído en Android y lanzado en un Toast[6].

Los métodos insertar se podrían compactar en un único php que haga una sentencia de inserte en vez de varios por cada entidad. (Al igual que eliminar.php)

Los phps "seleccionar..." guardan la información que se consulta en un array que se codifica con json y este json llega a Android Studio. Cada array es distinto para según qué entidad.

[6] Un toast es un pequeño campo de texto que aparece durante un periodo corto de tiempo en la pantalla.

IMPLEMENTACIÓN BACKEND

Login

La forma para loguearse es comprobar que se puede acceder al servidor de correo de webposta (correow).

En la imagen 27 podemos ver los parámetros de conexión al servidor de correo.

```
Tipo de servidor: IMAP o POP  (Mejor utiliza IMAP)
Nombre de servidor: ikasle.ehu.eus
Puerto: imap 993, pop 995
Protocolo SSL
Usuario: Tu login LDAP
Contraseña: Tu contraseña
```

Figura 27. Parámetros de conexión correow

```
Properties properties = new Properties();
properties.put("mail.imap.host", "ikasle.ehu.eus");
properties.put("mail.imap.port", "993");
properties.put("mail.imap.starttls.enable", "true");
Session emailSession = Session.getDefaultInstance(properties);

store = emailSession.getStore("imaps");
store.connect("ikasle.ehu.eus", ldap, password);
```

Figura 28. Implementación en java conexion correow

Comprobación

Para loguearse no se introduce la dirección de correo si no el LDAP. Por defecto, solo se puede introducir un LDAP numérico.

IMAP y POP3 son protocolos para poder recibir mensajes de correo desde un servidor.

IMAP trabaja directamente en el servidor mientras que POP3 descarga los mensajes en el dispositivo local.

En la aplicación se ha optado por el uso del protocolo IMAP por recomendación de Begoña y por las mejoras que presenta IMAP frente a POP3. Es decir, para conectar con el servidor y obtener los mensajes usamos el protocolo **IMAP y javaMail** Android.

Login intenta establecer una conexión con el servidor usando este protocolo y si ese usuario existe y tiene esa contraseña se descargan todos los mensajes que tiene en su buzón. Se **descargan todos los mensajes de la bandeja de entrada**. Puede que otros, como los de SPAM, no aparezcan. Si se comprueba que el usuario y la contraseña coinciden se va al activity home.

Círculo de progreso

Durante esta comprobación se muestra un pequeño círculo de progreso. El círculo de progreso son varias imágenes del tipo PNG que generan una animación gracias a la clase AnimationDrawable. Se ha usado una imagen GIF de licencia libre y con el uso de la página web ezgif.com se ha dividido este GIF en 12 imágenes. Cuantas más imágenes, la animación es más fluida.

Idioma

La aplicación, actualmente, está disponible en 3 idiomas; Español, Inglés y Euskera.

Se puede seleccionar entre esos tres idiomas o dejar que automáticamente se escoja, de entre esos tres, un idioma según el idioma del dispositivo. Cada vez que se cambia el idioma hay que refrescar la pantalla, cambiar el idioma y dejar seleccionado el idioma que se ha elegido. Esto supone algunos problemas que podremos apreciar en el apartado de problemas de implementación.

Recordar los datos

Al marcar la casilla de “recordar” y presionar el botón de loguearse, se almacenan los datos introducidos en el dispositivo con la ayuda de la clase SharedPreferences.

Se guarda tanto el ldap como la contraseña pero además el idioma que se ha seleccionado como preferido y la propia casilla checkbox.

Manejo de hilos dentro de login.

Tanto la comprobación como la animación del círculo de progreso se realizan en otro hilo (no en el principal). Esto es así porque las dos tareas son muy pesadas para ser lanzadas en el hilo principal, la aplicación podría congelarse.

Home

Actualización

Tanto los correos como las tareas se actualizan cada minuto.

Correow

Como he comentado anteriormente al logearse con éxito se descargan todos los mensajes de la carpeta de entrada del correow. Únicamente se muestran los últimos 10 en la aplicación. Esto también se ejecuta en otro hilo.

Noticias

Se ha creado una clase RSSReader (tipo AsyncTask) donde se obtiene el xml de los RSS disponibles de la ehu y se han ido recorriendo las diferentes etiquetas y almacenando en el objeto News las cosas que interesaban como el título (title) el enlace (link) el autor (author) y la fecha <updated>. Cada vez que se pulsa el botón de RSS y se abre el popup con las noticias estas se actualizan.

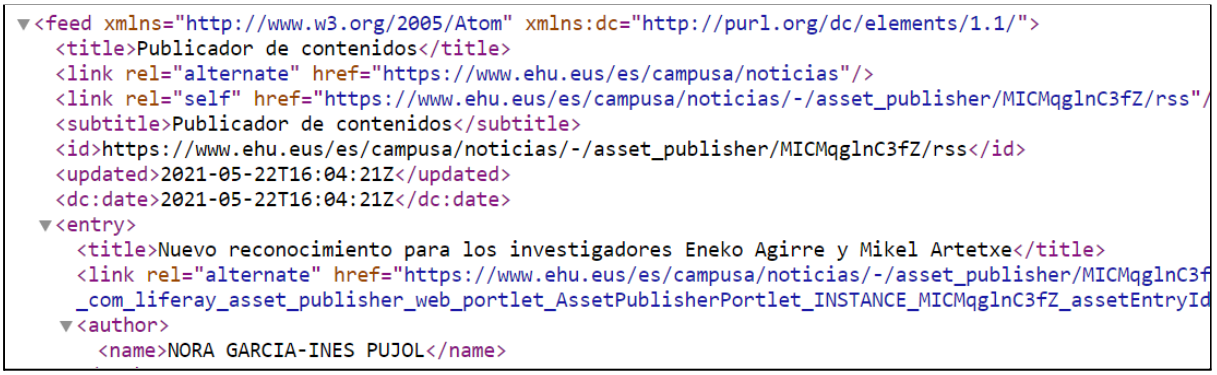


Figura 29. Formato xml noticias EHU

```
Node actual = entryChilds.item(j);
if(actual.getNodeName().equalsIgnoreCase("title")){
    noticia.setTitle(actual.getTextContent());
}
if(actual.getNodeName().equalsIgnoreCase("link")){
    noticia.setLink(actual.getAttributes().item(1).getTextContent()); //href
}
if(actual.getNodeName().equalsIgnoreCase("author")){
    if (actual.getChildNodes().item(1).getNodeName().equalsIgnoreCase("name")){
        noticia.setAuthor(actual.getChildNodes().item(1).getTextContent());
    }

}
if(actual.getNodeName().equalsIgnoreCase("published")){
```

Figura 30. Obtención de información del RSS de la EHU

Botones personalizables

Su implementación se basa en guardar en memoria la información que el usuario ha añadido gracias a la clase sharedPreferences. Esta información no se guarda en la base de datos por seguridad. Se guarda en el smartphone. Si otra persona coge tu móvil y se loguea con su cuenta verá los botones tal cual los has editado tu.

Calendario

Abre un popup con la imagen del calendario de informática.

Enlaces eGela y GAUR

Al pulsarlos se intenta enviar al usuario al enlace que por defecto es https://egela.ehu.eus/ para eGela y https://gestion.ehu.es/gaur GAUR.

Horario

Este horario se nutre de la base de datos. Se guarda la información y se muestra en un lisview. Podemos pinchar en la clase que queramos y se abre el popup de la asignatura con su información y la información del docente. Esto es porque las asignaturas, las clases y los profesores están enlazados. Una clase está enlazada a una asignatura por su id, una asignatura a un profesor por su correo y las tutorías a un profesor también por su correo.

Tareas

Las tareas también hacen uso de la base de datos y se guardan en un lisview. Si la tarea es un enlace al pulsarla actúa como un lanzador y envía al usuario a este enlace. Hay una asignatura que todos los alumnos tienen que se llama “GENERAL”. No tiene clases ya que no es una asignatura como tal y sirve para enviar mensajes a todo el mundo.

Tanto los horarios, asignaturas, docentes y tareas se añaden con el activity options.

Días

Al ser una aplicación multilenguaje, hay que traducir todos los días de la semana.

En la base de datos se guardaban los días en español. Al implementar el multilenguaje se optó por guardar los días con un número de 0 al 6. 0 = Lunes, 1 = Martes, etc. Después se obtiene el día (traducido) correspondiente con dicho número gracias al archivo strings.xml. Realizarlo con números facilita bastante la implementación.

Entidades BD a Android

Al obtener las entidad de la base de datos se guardaban en un array con sus respectivos objetos java. Para hacerlo de forma más eficaz, se decidió guardar en una estructura de datos tipo mapa (Map) los distintos objetos para un acceso más rápido.

Por ejemplo, un Map donde la clave es el correo del profesor y los valores: sus tutorías.

PROBLEMAS IMPLEMENTACIÓN

Durante la realización del proyecto y vistos los fallos que surgían categorizé los fallos en dos grupos ”problemas normales”, “problemas de mayor complejidad” y “fallos simples”.

Muchos de estos errores se solucionan gracias a la consola, el depurador o a la revisión del código, pero otros son mucho más difíciles de encontrar.

- **Problemas normales**

  Todo tipo de problema que tiene fácil solución.

- **Problemas de mayor complejidad**

  Aquellos cuya complejidad es mayor a la que esperaba en un principio.

- **Fallos simples**

  Problemas que generalmente aparecen por mi culpa (por la automatización de las acciones). Estos también pueden aparecer por otras razones. Generalmente ves el código bien pero no entiendes por qué falla.

Puse en funcionamiento una serie de pasos a realizar cada vez que me encontraba con un problema.

1. Aparece un error
2. Ver consola e intentar arreglar el error (30 minutos máximo)
    a. Puede que sea necesario buscar información del problema.
3. Si la consola no da ningún error o no es claro:
    a. Revisar el código detenidamente. (6 revisiones máximo, tomar descanso)
        i. Seguir todas las ramas y verificarlo todo.
    b. Utilizar depurador o mensajes por pantalla.
4. Si todavía no se ha encontrado el error:
    a. Preguntar en StackOverflow
5. Si todavía el error persiste y el código no es muy grande
    a. Rehacer el código (función) desde 0
6. Si todavía el error persiste
    a. Dejar unos días de descanso y volver a realizar estos pasos o buscar una alternativa

Muchas veces se soluciona el problema siguiendo estos últimos pasos y no se llega a comprender lo que fallaba. Por cada paso es recomendable dejar un tiempo de descanso para no saturarse y acabar perdiendo más tiempo del necesario.

Ahora comentaré algunos de los errores que han ido surgiendo durante el proyecto.

Error al actualizar los correos 10 veces (1 hora)

```
W/System.err: javax.mail.AuthenticationFailedException: [UNAVAILABLE] Maximum number of connections from user+IP exceeded (mail_max_userip_connections=10)
W/System.err:     at com.sun.mail.imap.IMAPStore.protocolConnect(IMAPStore.java:566)
        at javax.mail.Service.connect(Service.java:288)
W/System.err:     at javax.mail.Service.connect(Service.java:169)
        at com.example.tfgprueba2.DataAccess.login(DataAccess.java:34)
        at com.example.tfgprueba2.DataAccess.getMessages(DataAccess.java:55)
        at com.example.tfgprueba2.BusinessLogic.getCorreows(Businesslogic.java:18)
        at com.example.tfgprueba2.HomeActivity$2.run(HomeActivity.java:74)
W/System.err:     at java.lang.Thread.run(Thread.java:919)
    java.lang.NullPointerException: Attempt to get length of null array
        at com.example.tfgprueba2.BusinessLogic.getCorreows(Businesslogic.java:24)
        at com.example.tfgprueba2.HomeActivity$2.run(HomeActivity.java:74)
W/System.err:     at java.lang.Thread.run(Thread.java:919)
```

Figura 31. Error actualizar 10 veces los correos

Esto sería un ejemplo de problema normal, al actualizarse 10 veces los correos la aplicación lanzaba este error. El error se soluciona **cerrando la bandeja de entrada** ya que cada vez que se obtenían los correos se abría una nueva conexión con esa bandeja. La confusión que tuve fue debido a que la conexión con el servidor y la conexión con la bandeja no era lo mismo. Se soluciona tras informarme de como cerrar correctamente la bandeja de entrada.

Error por espacio en la librería jsoup (40 minutos)

```
implementation files('libs/mail.jar')
implementation files('libs/jsoup-1.13.1.jar ')
implementation files('libs/jsoup-1.13.1-sources.jar')
implementation files('libs/jsoup-1.13.1-javadoc.jar')
```

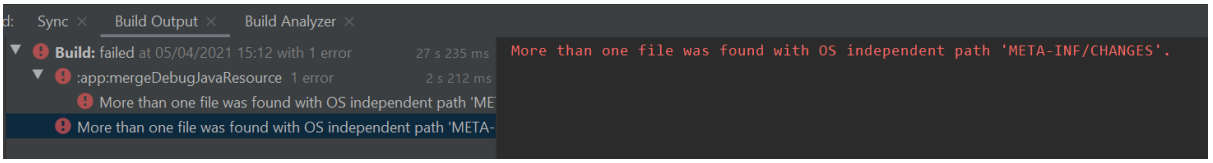


Figura 32. Error librería jsoup

Esto sería un ejemplo de ”fallo simple”.

Como se puede ver en esta imagen, hay **un espacio en la segunda línea al final**

“`implementation files ('lib/jsoup-1.13.1.jar_')`”

Como se puede apreciar, la consola lanza un error que me llevó a realizar muchas pruebas y muchos cambios que no eran necesarios. No realicé ninguna pregunta en stackOverflow y directamente reescribí esa parte del código. Empecé reescribiendo las librerías y por suerte ahí estaba el problema.

Los docentes se repiten (3 horas)

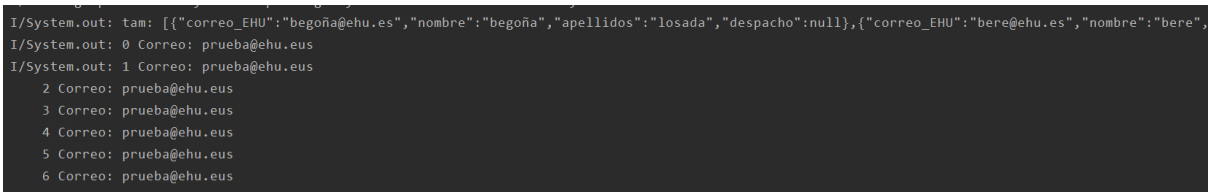


Figura 33. Error docentes repetidos

Otro ejemplo de fallo simple. El array que se almacena tenía siempre el mismo docente (el último) en vez de tener todos ellos. Seguí los pasos mencionados anteriormente incluso pregunté en stackOverflow. Nadie entendía por qué ocurría. Finalmente, no reescribí el código ya que decidí volver a comprobarlo todo y siguiendo las ramas me dí cuenta que las **variables de la clase docente eran estáticas**.

El problema de los lunes (1 hora)

En la base de datos guardaba el día de la clase y el día de la tutoría. Pero cuando implementé el multilenguaje en la aplicación dejó de tener sentido que todo estuviera en español. Decidí, como he explicado anteriormente, guardar en la base de datos los días con un número de 0 al 6. 0 = Lunes, 1 = Martes, etc.

El problema fue que por alguna extraña razón las clases de los lunes y las tutorías de los lunes no se guardaban en la base de datos.

Con este problema llegue al paso 3 donde llené todo con líneas para imprimir por pantalla ya que el depurador no me arrojaba información. Finalmente llegué al php de conexión y me puse a investigar hasta llegar al manual de php.

Manual php

https://www.php.net/manual/en/types.comparisons.php

Resulta que yo hacía uso de la función **empty** para verificar si se habían introducido todos los datos. Si no estaba vacío se suponía que estaba lleno. Tras ver la tabla me di cuenta que **el string "0" lo interpretaba como que esa variable estaba vacía.** Al final, decidí usar **isset** en vez de sumar un número a cada día.

Problema en webhost. Desaparece el servidor.

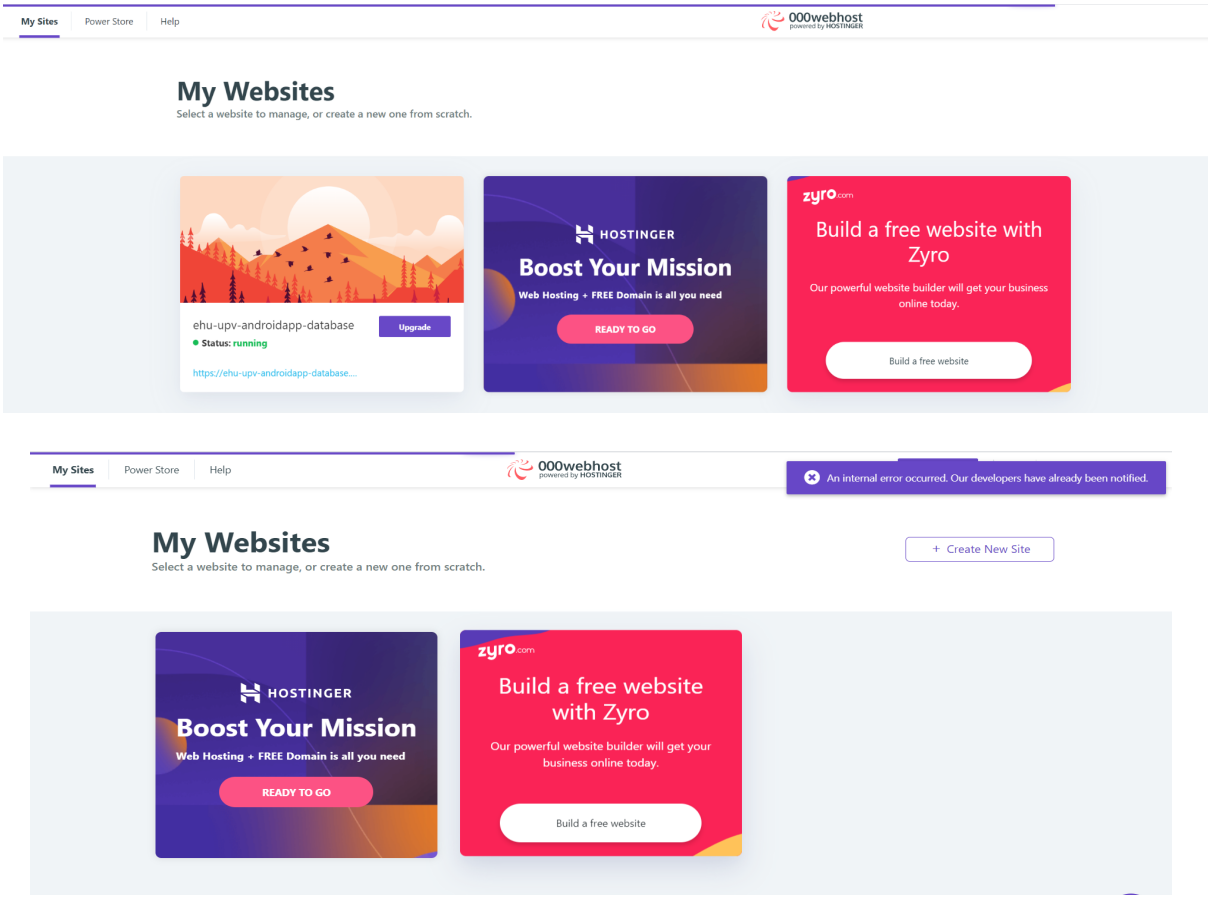


Figura 34. Desaparece el servidor webhost

Durante toda una mañana el servidor estaba inaccesible. Al ser esto uno de los posibles riesgos, ya estaba preparado y con todas las copias de seguridad hechas. Si el servidor hubiera seguido sin funcionar hubiera migrado todos mis archivos a otro servidor. En la imagen de abajo se puede apreciar un mensaje de error: “An internal error occurred...”

El calendario no se puede ampliar

Tras hacer toda la lógica para que se pudiese ampliar una imagen (hacer zoom en ella) e intentar añadirla a la imagen surgió el problema de estar trabajando en un popup en vez de en un activity. Había posibles soluciones para este problema pero al llevar más tiempo del planificado se omitió y se invirtió ese tiempo en otras tareas.

La aplicación no se ajusta a todos los dispositivos

Es muy complicado realizar una aplicación y que se ajuste a todas las pantallas. Si la pantalla es muy pequeña puede que no entre toda la información. Pero el tener una pantalla grande con poca información tampoco es del todo adecuado. La solución óptima sería implementar distintas vistas personalizadas como hacen las aplicaciones profesionales. No obstante, esto me iba a llevar demasiado tiempo así que se puso como mejora a realizar. Actualmente la aplicación se adapta a las pantallas pero no de la forma más óptima para distintos móviles.

Cambio de nombre en la aplicación

La aplicación tenía el título de “TFGPrueba2” hasta que decidí cambiarlo. Esto supuso que Git se desconectará de Android y tuve que dar unas cuantas vueltas para entender cómo volver a conectarlo desde el punto en el que estaba.

Lentitud

Para aumentar la rapidez se decidió guardar las entidades en android en estructuras de tipo mapa en java. Por ejemplo, una clave que sea el correo del profesor y sus valores las tutorías que tiene.

Mantener seleccionado el idioma

El login de idiomas tiene varios problemas.

- Un alumno debe poder dejar recordado el idioma que ha elegido.

  Solución:

  Esto supone que nada más iniciar la aplicación esta debe buscar si hay algún idioma que se ha marcado para recordarlo (haciendo uso de la clase shared preferences).

- Un alumno debe poder cambiar entre idiomas y que el activity se refresque.

  Solución:

  Para cambiar el idioma y que este cambie en el activity, hay que refrescar. Al refrescar y abrir nuevamente el activity, este no recuerda el nuevo idioma seleccionado. Para mantenerlo seleccionado, opté por enviarlo como un parámetro al mismo activity.

GAUR Seguridad SSO y Moodle en eGela

Debido a los permisos que requieren tanto GAUR como eGela se buscó otra forma de implementar las funcionalidades relacionadas con estos servicios. Como comenté al principio, GAUR no tiene siempre la información correcta y a veces no coincide con la de la página web de la EHU. Una forma de solucionar este problema es que sean los propios alumnos los que añadan esta información. Si se hacen cambios durante el curso, estos se podrán actualizar fácilmente en EHU Zurekin y de forma más rápida.

Muchos alumnos crean su propio horario usando aplicaciones de terceros y por tanto es una buena idea que sean ellos los que introduzcan la información.

PRUEBAS CON ALUMNOS

A medida que el proyecto fue desarrollándose se realizaron pruebas con distintos compañeros. 6 fueron los alumnos que probaron mi aplicación. Todos desde mi dispositivo. Se utilizó la técnica “Cognitive walkthrough”.[7]

Se les dijo que hicieran una clase de pruebas:

- Ver sus correos, las noticias.
- Seleccionar asignaturas.
- Añadir algún docente, clase, tutoría o asignatura.
- Ir a eGela/ Gaur.
- Añadir una tarea.
- Intentar editar un botón editable.
- Entrar en el enlace BBC de una asignatura.
- Decirme qué día es la última tutoría de un docente.

De estas pruebas saqué algunas conclusiones:

- La tarea en la que tardaban más era en la edición de un botón. Algunos descubren a la primera como editarlos (manteniéndolo presionado) pero otros buscaban por la pantalla algún otro botón para editarlo. Tras comentarlo con estos últimos me comentaron que no lo cambiase ya que mantener el botón era una mejor opción. Decidí cambiar el texto del botón y en vez de poner de primeras: “botón editable” decidí poner “mantén pulsado para editar”.
- Otro problema fue que algunos alumnos no saben cual es el icono de los RSS y por tanto no encontraron la noticias directamente. Tuvieron que ir descartando hasta encontrarlo. No es muy difícil ya que hay pocos botones.
- El ver la información de una asignatura tampoco es del todo intuitivo. Hay que clicar en el horario la asignatura y entonces se abre un popup con la información. Esto según me comentaron tiene sus pros y contras. Se puede llegar en un solo clic a la asignatura para ver su BBC pero también puede que tardes más al tener que recorrer el horario para encontrar la asignatura.

En las demás tareas no tuvieron demasiado problema y se realizaron correctamente.

[7] El método de recorrido cognitivo es un método de inspección de usabilidad utilizado para identificar problemas de usabilidad en sistemas interactivos, centrándose en lo fácil que es para los nuevos usuarios realizar tareas con el sistema. (Obtenido de wikipedia) https://en.wikipedia.org/wiki/Cognitive_walkthrough

GESTIÓN DEL PROYECTO

En las tablas vemos el nombre de la iteración, la tarea a realizar, el estimado de tiempos y el empleado acumulado. El tiempo estimado es el tiempo que se planificó que se iba a tardar en realizar dicha tarea. El tiempo empleado acumulado es el tiempo que se va acumulando por cada iteración en esa tarea.

Si una tarea lleva más tiempo del estimado se añaden esta horas extras con un símbolo de suma. Por ejemplo: 20h estimadas + 4h imprevistas.

Si una tarea lleva menos tiempo del previsto o se acaba antes de llegar al estimado se pone un punto al final. ”20h.”.

Iteración 0 (25/05/2020 - 7/06/2020 // 10/08/2020 - 16/08/2020)

Con vistas al TFG, comencé en mayo del 2020 a realizar el curso “Android desde 0”.

Decidí seguir este curso porque mi conocimiento en Android era nulo y como bien dice su nombre, el curso enseña Android desde 0. Además, trabaja con lenguaje java.

En esta iteración 0 realicé un tercio de las diferentes prácticas que se explican y realizan en estos vídeos. También realicé pequeñas Apps similares para practicar.

Tiempos iteración 0

IT 0	Tarea	Estimado	Empleado Acumulado
Formación	• Android básico	40h	15h

Tabla 2. Estimación iteración 0

El 9/11/2020 envié un correo a Begoña preguntando acerca del trabajo de fin de grado.

Iteración 0.1 (16/11/2020 - 27/11/2020)

Reuniones iteración 0.1

16 noviembre 2020 Primera reunión inicial

Temas tratados

- Propuse mi idea de TFG y sus diferentes características.
- Se discutió acerca de las funciones que debería tener la aplicación para ser viable como TFG.

Acuerdos adoptados

- Investigar las posibilidades que ofrece android para realizar dichas características.
- Investigar las posibilidades que ofrecen los distintos servicios de la EHU.

27 noviembre 2020 Segunda reunión inicial

Temas tratados

- Se habló acerca de la investigación realizada y se realizó la primera planificación.

Acuerdos adoptados

- Seguir investigando.

Resumen iteración 0.1

Esta iteración se centró principalmente en la investigación, por encima, de las tareas a realizar y de las posibilidades del proyecto.

Se investigó el funcionamiento de la aplicación K-9 mail[8] y las diferentes librerías de Java/Android que realizan esta tarea.

Investigué brevemente sobre los RSS de la EHU. Al ya haber trabajado con RSSs en las prácticas, no supuso mucho trabajo hacerme una idea de como plantearlo.

Tiempos iteración 0.1

IT 0.1	Tarea	Estimado	Empleado Acumulado
Formación	• Android básico • Correow • RSS • Moodle • G.A.U.R	40h 5h 5h 5h 5h	15h 2h 10min 2h 2h
Reuniones	Reuniones presenciales/BBC y correos	10h	1,5h

Tabla 3. Estimación iteración 0.1

[8] https://k9mail.app/

Iteración 1 (19/01/2021 - 24/01/2020)

Resumen iteración 1

Tras las vacaciones de navidad y acabar los exámenes retomé el proyecto.

Reanudé el curso de Android y concluí con él. Paralelamente seguí investigando. Obtuve la información de acceso al correow y realicé pruebas de conexión con apps tipo cliente de correo para verificar dicha información. K-9 mail.

Imágenes

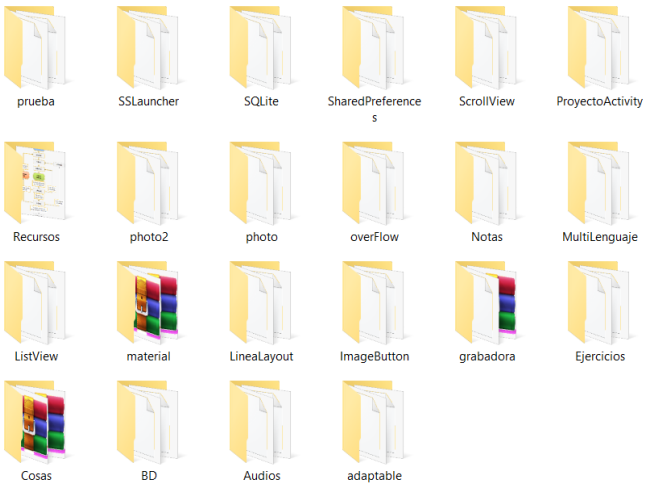


Figura 35. Proyectos de formación en Android.

Tiempos iteración 1

IT 1	Tarea	Estimado	Empleado Acumulado
Formación	• Android básico • Correow • RSS • Moodle • G.A.U.R	40h 5h 5h 5h 5h	40h. 3h 10min 3h 3h
Reuniones	Reuniones presenciales/BBC y correos	10h	1,5h

Tabla 4. Estimación iteración 1

Respecto a Moodle y G.A.U.R la obtención de datos iba a ser complicada por el tema de los permisos. Leí acerca de los servicios web que ofrece Moodle y vi que iba a requerir demasiado tiempo y que no era tan necesario.

Iteración 2 (4/02/2021 - 4/03/2021)

Reuniones iteración 2

4 febrero 2021 reunión de inicio de la iteración

Temas tratados

- Se habló sobre la información obtenida acerca de las diferentes aplicaciones de la EHU.

Acuerdos adoptados

- Realizar una pequeña aplicación tipo cliente de correo para conectar con correow. En esta aplicación deberían aparecer los últimos 10 correos de correow.
- Seguir investigando.

Resumen iteración 2

En el transcurso de esta iteración investigué acerca las diferentes formas de acceso a los correos que Android Studio ofrece, y finalmente decidí hacer uso de la librería JavaMail para Android.

Hice un diseño básico de esta aplicación, la implementé y conseguí mostrar los 10 últimos correos.

La implementación de esta tarea la planifiqué en dos partes.

La primera; crear la lógica de comprobación y conexión con el servidor IMAP ikasle.ehu.eus para obtener los últimos 10 mensajes y almacenarlos en un arreglo.

La segunda; crear en un activity una lista editable para poder incluir ahí el asunto, el remitente, una pequeña previsualización del correo, la fecha, etc.

Las tareas se realizaron satisfactoriamente.

No obstante, la aplicación tenía estos fallos:

- No mostraba el contenido de los correos (error MimeMultipart)
- El nombre del remitente a veces aparecía codificado (error =?utf-8)
- Solo permitía actualizar 10 veces los correos, después lanzaba una excepción.

En esta iteración empecé a realizar copias de seguridad en propio ordenador. Seguía un formato de carpetas nombradas por versiones: ”TFG_Prueba1”, ”TFG_Prueba2”, etc.

Imágenes

Primera versión

En esta primera versión de la parte de Correow conseguí obtener el número total de mensajes y mostrar el primero de ellos.

Hice pequeñas pruebas para ver el formato que seguían estos correos.

Aquí ya empezaron las complicaciones ya que no sabía a que hacía referencia el texto MimeMultipart.

Esto se asignó como una nueva tarea de investigación.

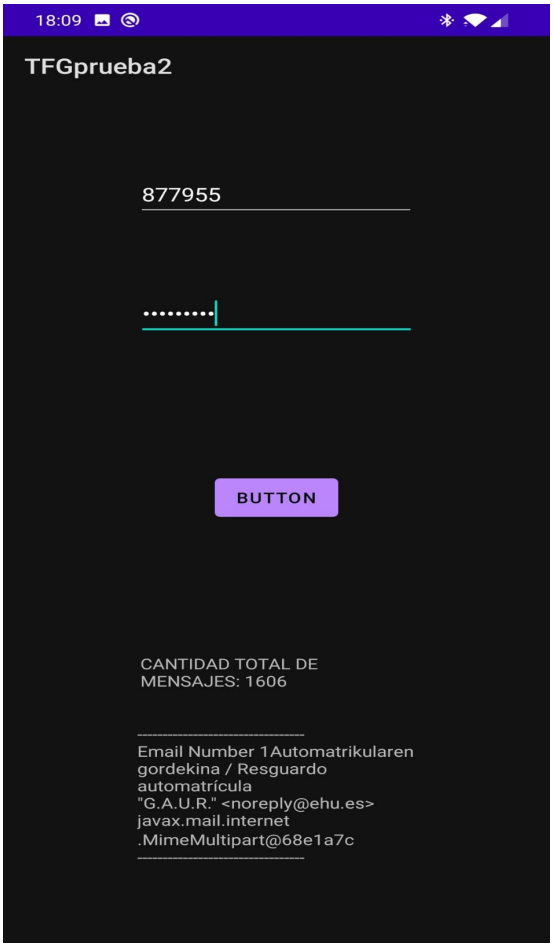


Figura 36. Login y correo iteración 2. Primera versión

Segunda versión

En esta segunda versión se consiguieron obtener los últimos mensajes pero faltaba seguir investigando el formato. Únicamente se mostraba la dirección de correo y el asunto del mensaje.

Mostraba los 10 mensajes pero solo los últimos 5 eran visibles.

No se actualizaban los correos automáticamente.

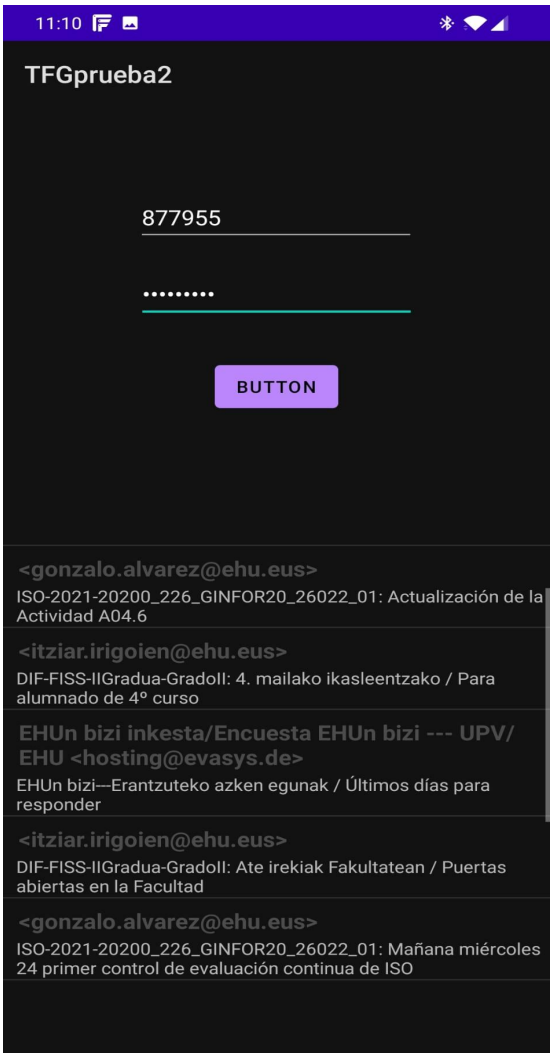


Figura 37. Login y correo iteración 2. Segunda versión

Más tarde implementé el código para obtener los 10 últimos correos. Aquí apareció el segundo problema, el nombre del remitente a veces aparecía codificado.

Como se puede apreciar en la imagen de la izquierda, algunos nombres estaban codificados siguiendo el formato "=?utf-8...". Esto ya me daba una pista, el problema era debido a las tildes y otros caracteres que contiene el español.

En la imagen de la izquierda aparece el resultado una vez solucionado este problema.

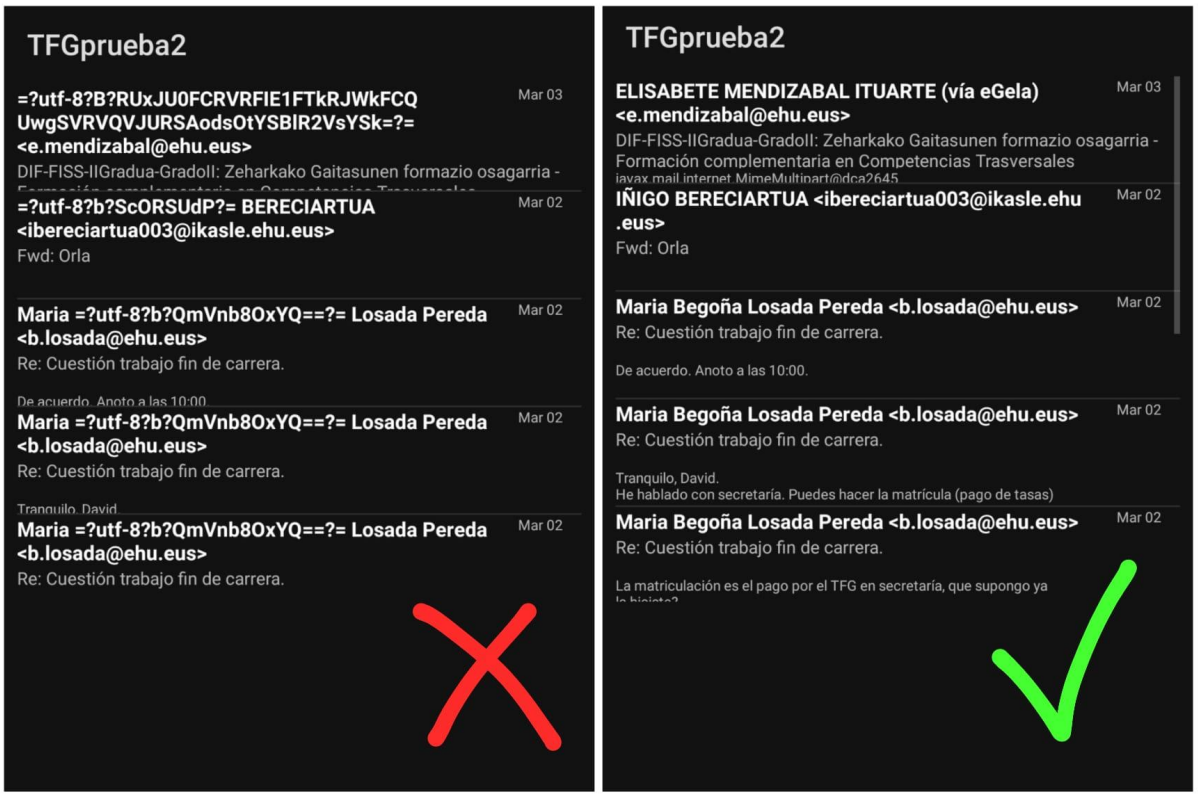


Figura 38. Solución error utf-8

Llegado a este punto seguía teniendo dos problemas.

- No mostraba siempre el contenido de los correos (error MimeMultipart)
- Solo permitía actualizar 10 veces los correos, después lanzaba una excepción.

Tiempos iteración 2

	Tarea	Estimado	Empleado Acumulado
Instalación	• Android Studio y conf. Dispositivos • Librerías.	5h 1h	4h. 30min
Seguridad	• Copia App Android • Nube App Android (GitHub) • Copia Documentación • Nube Documentación (Google Drive)	20min 1h 20min 20min	5min 0h 0h 0h
Formación	• Android básico • Correow • RSS • Moodle • G.A.U.R	40h 5h 5h 5h 5h	40h. 3.5h 10min 3.5h 3.5h
Implementación	• Login • Correow	1h 30	1h 18h
Reuniones	Reuniones presenciales/BBC y correos	10h	2h

Tabla 5. Estimación iteración 2

En esta iteración trabajaba sin copias en la nube. Realizaba únicamente las copias de seguridad del proyecto en el propio ordenador. Más tarde me dí cuenta que iba a necesitar, por si acaso, subir el proyecto a la nube y una forma de controlar las versiones.

Iteración 3 (4/03/2021 - 18/03/2021)

Reuniones iteración 3

4 marzo 2021 reunión de inicio de la iteración

Temas tratados

- Enseñé mi primera versión de la parte de correow

Acuerdos adoptados

- Solucionar problemas MimeMultipart y actualización 10 veces.
- Poner el login en otra ventana previa. (Hasta ahora se mostraba todo en una misma ventana como se ve en la imagen)
- Implementar actualización automática cada X tiempo.
- Realizar una pequeña aplicación tipo RSS y conectarlo a las noticias EHU.
- Seguir investigando.

Resumen iteración 3

La primera tarea que realicé fue **poner el login separado de los correos**. El usuario primero introduciría sus datos y si son correctos le enviaría al home donde se muestran sus correos. Esto parecía una tarea sencilla pero surgieron problemas.

Al momento de poner el login en otra ventana previa al home, vi que era necesario dar un **feedback al usuario** para que este sepa que se está tramitando su solicitud de conexión. Al este introducir sus datos y dar al botón de entrar, la aplicación tardaba un poco en comprobar sus datos y enviarlo al home. Esto podía producir que el usuario, ansioso, pulsara el botón muchas veces.

Para solventar este problema se me ocurrió poner un **círculo de progreso** para que este hiciera una pequeña animación al lado del botón para entrar. Pensaba que era tan fácil como agarrar un gif y hacerlo aparecer y desaparecer durante el transcurso de la comprobación. Al final fue un poco más complicado, había que usar animaciones.

Esto dio pie a un problema que hasta entonces desconocía. Al pulsar el botón de entrar, la aplicación no hacía nada, dejaba de funcionar y no entendía el por qué.

Tuve que investigar bastante ya que no sabía por dónde coger este error ya que el código aparentemente funcionaba bien. La consola no lanzaba ningún error.

Tras hacer muchas pruebas separando el código e informarme sobre como funciona android llegué a la conclusión que podía ser tema de hilos.

En su hilo principal (UI thread), android dibuja la pantalla y realiza otros procesos que tardan un tiempo. Hasta ahora, tanto la animación de progreso como la comprobación de datos y obtención de correos trabajaba en el hilo principal ocupando mucho tiempo. Esto, junto a la propia animación de pulsación del botón, hacía que la aplicación dejará de funcionar. El hilo principal estaba ocupado y la pantalla dejaba de responder.

Una vez entendido el problema me puse a investigar el manejo de hilos en android para intentar llevar estas tareas pesadas a otro hilo.

Una vez entendido el manejo de hilos y solucionado este problema, realicé la parte de **actualizar cada X tiempo** automáticamente. No fue muy complicada esta tarea pero el **error de actualizar más de 10 veces los correos seguía vigente**.

Investigué un poco más acerca de los RSS e hice una primera implementación. Fue bastante sencilla la implementación debido a mi conocimiento y el trabajo que realicé en las prácticas.

Durante el transcurso de esta iteración había realizado commits en git pero no push.

Imágenes

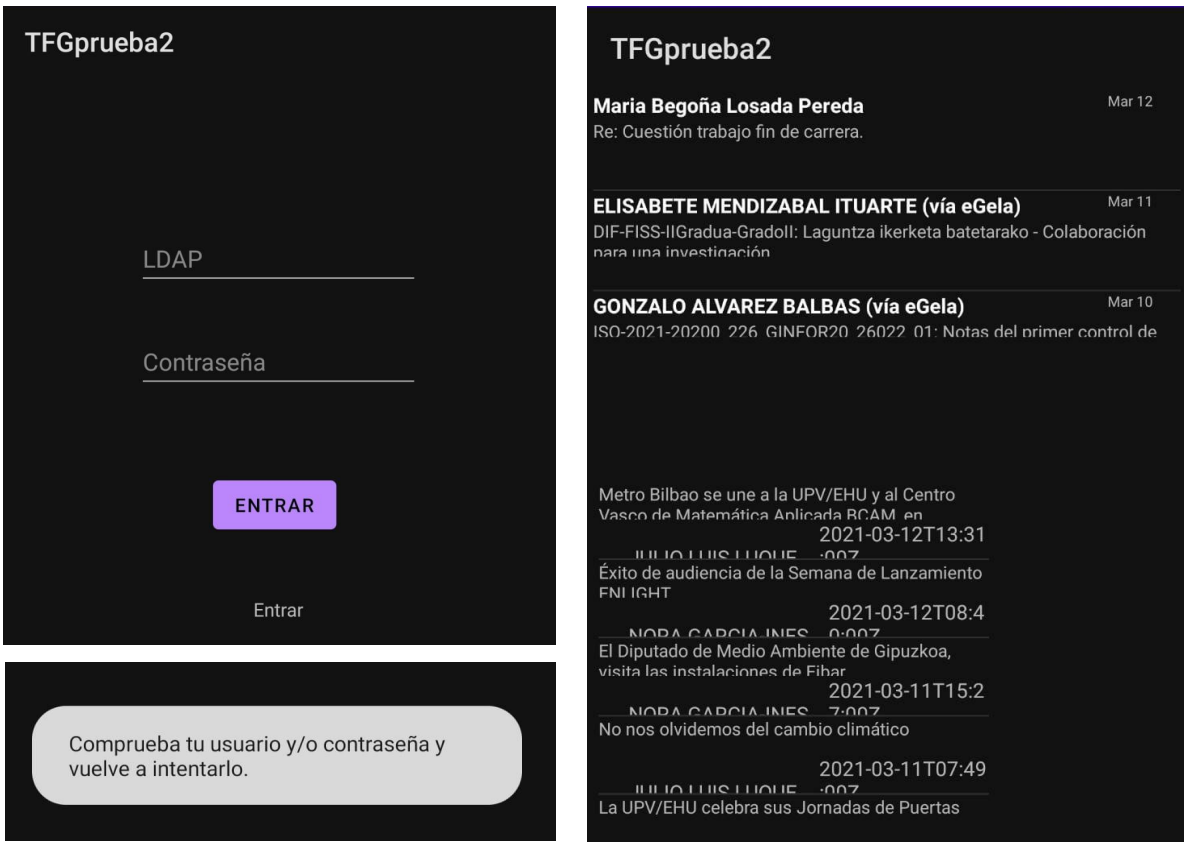


Figura 39. Login, Correos y Noticias iteración 3

Se puede apreciar como al fallar muestra un mensaje de error. Si los datos coinciden, va al Home de la aplicación donde muestra una lista de correos y una lista de noticias debajo.

La aplicación en este punto carece de diseño y usa un tema propio de android.

Tiempos iteración 3

	Tarea	Estimado	Empleado Acumulado
Instalación	• Android Studio y conf. Dispositivos • Librerías.	5h 1h	4h 30min
Seguridad	• Copia App Android • Nube App Android (GitHub) • Copia Documentación • Nube Documentación (Google Drive)	20min 1h 20min 20min	8min 30min 3min 3min
Formación	• Android básico ○ Android imprevistos • Correow • RSS • Moodle • G.A.U.R	40h +12h 5h 5h 5h 5h	40h. +12h 4h 30min. 3.5h 3.5h
Implementación	• Login • Correow • RSS	1h 30h 30h	1h + 6h 23h 10h
Reuniones	Reuniones presenciales/BBC y correos	10h	2.5h

Tabla 6. Estimación iteración 3

Preparé el entorno gitHub en android studio y en la propia plataforma y realicé los commits. No recordaba que era necesario hacer push después.

Iteración 4 (18/03/2021 - 31/03/2021)

Reuniones iteración 4

18 marzo 2021 reunión de inicio de la iteración

Temas tratados

- Enseñé la nueva versión de la parte de correow
- Enseñé mi primera versión de la parte de RSS noticias
- Comente los errores que había solucionado y las ideas para el tema del horario.

Acuerdos adoptados

- Solucionar problema actualización 10 veces.
- Mejorar el código.
- Mejorar RSSs.
- Investigar el horario.

Resumen iteración 4

Fue una iteración donde me centré en mejorar el código de la aplicación y solucionar errores. Tanto el error de actualizar los correos 10 veces como otros errores fueron solucionados. Se realizó una estructura más correcta del código, un mejor diseño para los correos/RSS y se pusieron accesos rápidos a eGela y GAUR.

No conseguí que el botón de acceso rápido a la aplicación GAUR de android funcionara, por tanto hice que redirigiera a la web.

En esta iteración tuve que tomar la decisión de si seguir investigando con Moodle y G.A.U.R o buscar otra forma. Investigué sobre estos servicios y busqué información sobre otras posibles formas de realizar la tarea del horario.

Imágenes

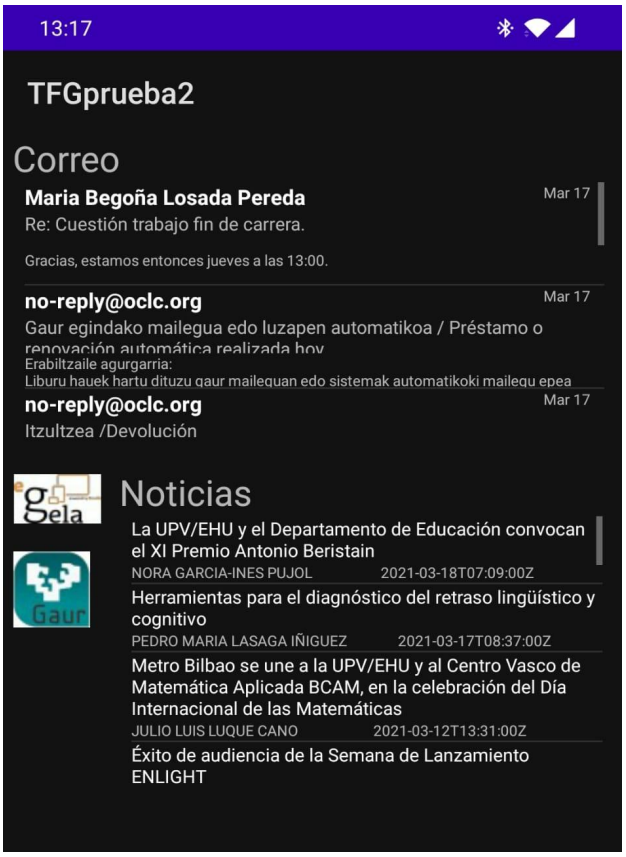


Figura 40. Correos, Noticias y botones iteración 4

Como se aprecia en esta imagen el diseño base ahora está más limpio y se puede ver de forma más clara los diferentes apartados de los correos y de las noticias.

Hay dos botones con los logos de eGela y Gaur y estos, al pulsarlos, redirigen a dichas páginas.

Tiempos iteración 4

	Tarea	Estimado	Empleado Acumulado
Instalación	• Android Studio y conf. Dispositivos • Librerías.	5h 1h	4h. 30min
Seguridad	• Copia App Android • Nube App Android (GitHub y G.D) • Copia Documentación • Nube Documentación (Google Drive)	20min 1h 20min 20min	10min 35min 10min 10min
Formación	• Android básico ○ Android imprevistos • Correow • RSS • Moodle • G.A.U.R	40h +20h 5h 5h 5h 5h	40h. +12h 4h. 30min. 4h. 4h.
Implementación	• Login • Correow • RSS • Otros	1h 30h 30h 5h	1h + 6h 25h. 18h. 20min
Reuniones	Reuniones presenciales/BBC y correos	10h	3h

Tabla 7. Estimación iteración 4

G.D, Google Drive, es otra plataforma que empecé a usar complementariamente. Google Drive en versión PC. Cada edición que hiciera en el proyecto se guardaba en la nube de Google Drive directamente esto conjuntamente a Git.

Iteración 5 (31/03/2021 - 16/04/21)

Reuniones iteración 5

31 de marzo 2021 reunión de inicio de la iteración

Temas tratados

- Enseñé lo que llevaba hasta ese momento.
- Comente lo que pensaba acerca del tema G.A.U.R y eGela.

Acuerdos adoptados

- Plantear e intentar empezar a implementar la parte de los horarios.

Decisión tomada

La decisión era si seguir intentando acceder a los datos de eGela y G.A.U.R o buscar otra alternativa. Tras investigar y ver que obtener dichos datos iba a suponer mucho tiempo, dedicación e investigación, decidí buscar otra alternativa.

Mi idea fue usar una base de datos. Esto suponía una gran ventaja frente a obtener los datos de estas plataformas; estar siempre actualizada.

A veces, estas plataformas no están actualizadas y la información está repartida entre ellas y no coincide. La mejor opción es que sean los propios alumnos los que rellenen estos datos. Hay muchos alumnos que ya lo hacen por su cuenta con aplicaciones de terceros. Los alumnos, después de los profesores, son los primeros en enterarse de los cambios. Y que GAUR y EHU cambien la información es algo que demoraría mucho.

Resumen iteración 5

Me centré en diseñar y crear la base de datos e implementar en Android una forma para introducir estos datos. Si un usuario ve que algo no está actualizado, puede cambiarlo o añadir nuevas cosas que no están en G.A.U.R o EHU.

Para ello use scripts PHP junto a 000webhost y phpmyadmin (MySQL). A la hora de implementar en android, al añadir las librerías, tuve el problema con la librería Jsoup. La investigación sobre PHP, 000webhost y MySQL no supuso gran tarea debido a que ya se trabajó con estas herramientas durante la carrera.

Imágenes

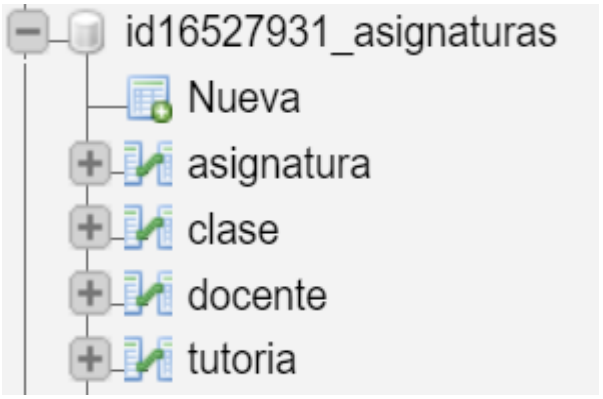


Figura 41. Entidades base de datos inicial

A la izquierda, la base de datos. Se pueden apreciar las distintas entidades, esto siguiendo el modelo anteriormente enseñado.

En la imagen de la derecha, vemos la primera prueba que realicé en android para comprar si podía añadir docentes. Más tarde implementé las demás partes, asignatura, clase, tutoría y les di un diseño básico.

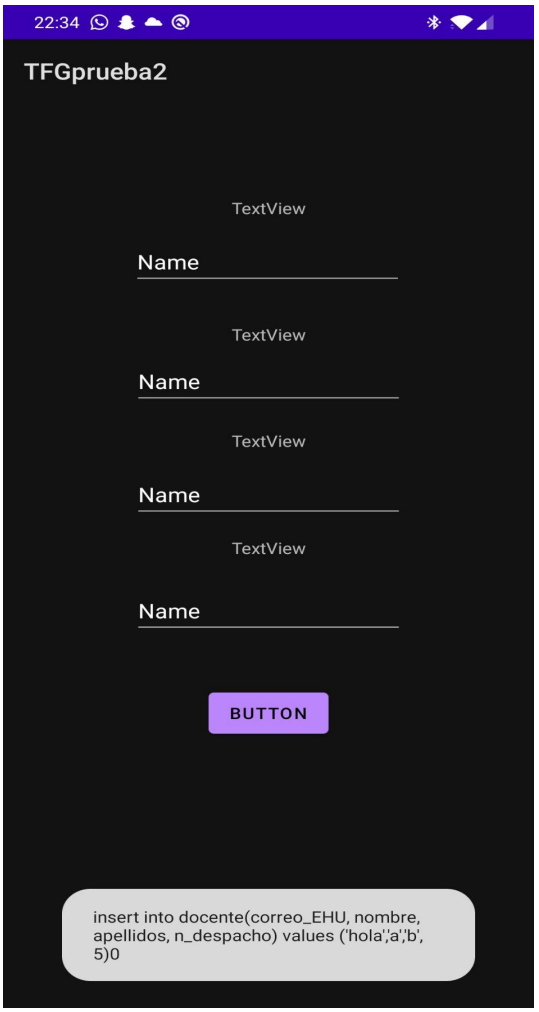


Figura 42. Primera prueba inserción

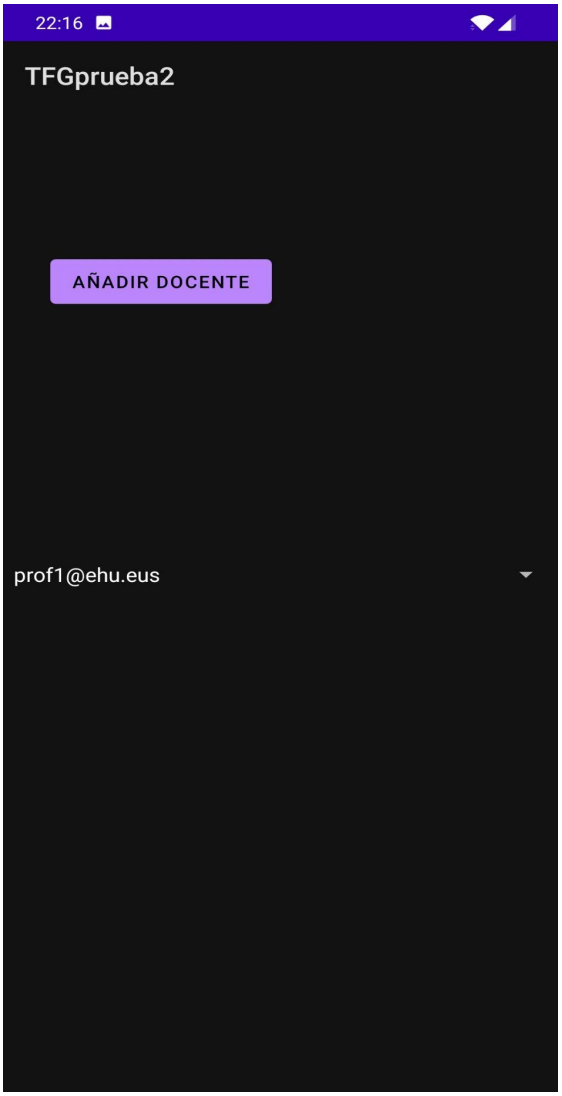


Figura 43. Primera prueba selección

Primero realicé una forma para ver los docentes en una lista desplegable, tras realizar diferentes pruebas y ver que todo funcionaba correctamente, implementé las demás clases.

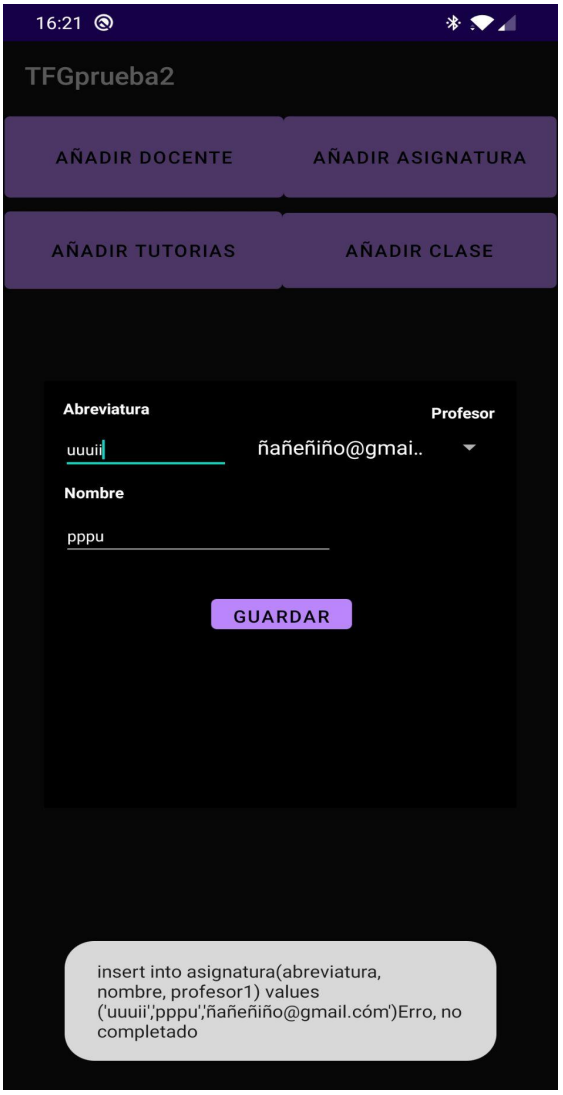


Figura 44. Primera prueba popups

Decidí que para introducir datos se mostrará un popup ya que da una mejor imagen a la aplicación. Implementé las demás clases y ya era posible rellenar toda la base de datos.

Tiempos iteración 5

	Tarea	Estimado	Real
Instalación	• Android Studio y conf. Dispositivos • Librerías. • Servidor	5h 1h 1h	4h 1h 1h
Seguridad	• Copia App Android • Nube App Android (GitHub) • Copia Documentación • Nube Documentación (Google Docs)	20min 1h 20min 20min	13min 40min 10min 10min
Formación	• Android básico ○ Android imprevistos • Correow • RSS • Moodle • G.A.U.R • BD	40h 20h 5h 5h 5h 5h 5h	40h. 15h 4h. 30min. 4h. 4h. 3h

Implementación	• Login
	• Correow
	• RSS
	• Horario
	  ○ Inserción
	  ○ Selección
	  ○ Visualización
	  ○ BD
	• Otros
	• Pruebas
		1h
		30
		30
		
		30
		15
		20
		10
		5
		5
			1h + 6h
			25h.
			18h.
			
			18h
			6h
			3h
			5h
			20min
			10min

Diseño	• Home • Correow • RSS • Horario • Otros • Pruebas	2 2 2 3 2 5	0h 0h 0h 0h 0h 0h
Documentación	• Memoria • Presentación	70 20	2h 0h
Reuniones	Reuniones presenciales/BBC y correos	10h	3.5h

Tabla 8. Estimación iteración 5

A pesar de que tardé poco en realizar el diseño de la base de datos e implementarla en phpmyadmin, la implementación en Android fue complicada, realicé varias pruebas con distintas librerías.

Iteración 6 (16/04/21 - 29/04/21)

Reuniones iteración 6

16 de abril 2021 reunión de inicio de la iteración

Temas tratados

- Enseñé lo que llevaba hasta ese momento.
- Comenté las ventajas del uso de una base de datos.
- Hablamos sobre cómo implementar la agenda de correow. (Para añadir tareas)

Acuerdos adoptados

- Mejorar y finalizar con la parte de los horarios.
- Solucionar algunos problemas que se vieron durante la reunión.
- Añadir multilingüe.
- Investigar agenda correow.

Resumen iteración 6

Realicé la traducción completa para la aplicación. Añadí a la base de datos otra tabla para las asignaturas de los usuarios e implementé la forma de poder elegir qué asignaturas tiene un usuario y cómo ver su información.

En esta iteración se dio solución al problema de los ”lunes”. Como ya hemos comentado, esto era debido a que en los PHPs se hacía uso de la función Empty en vez de Isset. Empty devuelve true si se le pasa un el número 0. Se decidió, para nombrar los días de la semana, el uso de números en vez de nombres dado que la aplicación era multilingüe.

Imágenes

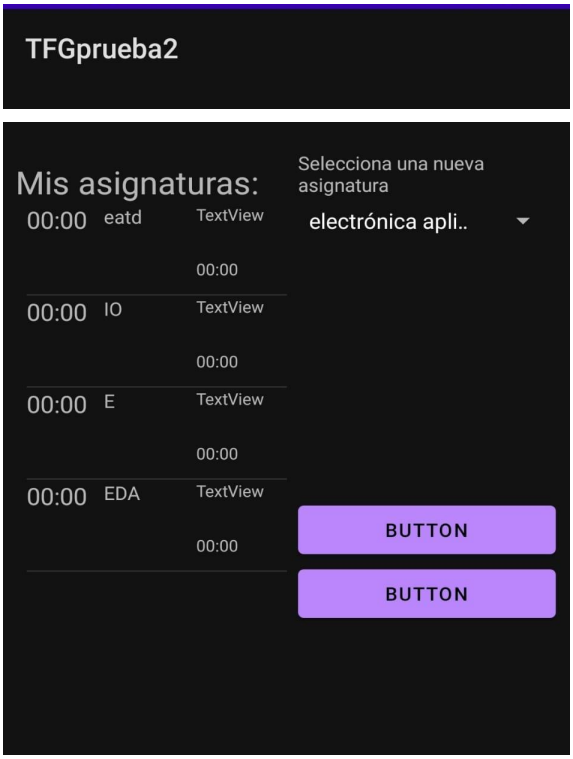


Figura 45. Prueba horario

Primera versión visualización de asignaturas para el usuario.

Como se puede apreciar en la imagen, la información no es correcta, solo aparecen las abreviaturas de las asignaturas.

Aún así, la estructura ya estaba realizada y solo me faltaba rellenar estos campos y hacer esta parte de una forma más intuitiva y bonita para añadir y eliminar las asignaturas de un usuario.

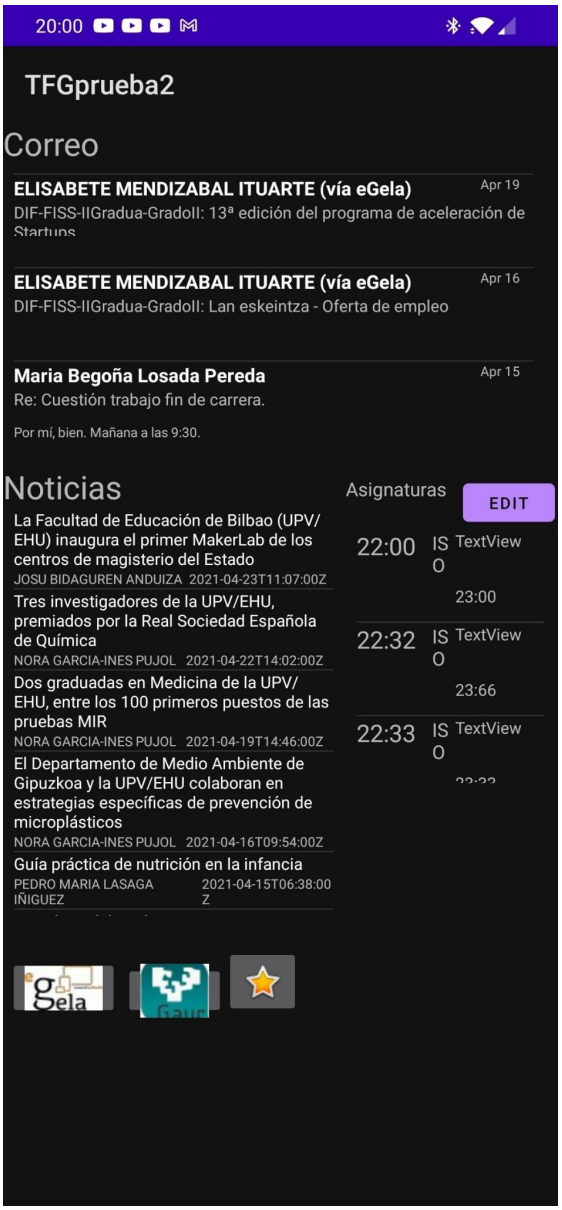


Figura 46. Primera versión (Modo oscuro)

Se puede apreciar una estrella que al pulsarla envía al usuario a la zona para añadir asignaturas que hemos visto previamente. En este Home, el usuario ya puede ver los correos, las noticias y las asignaturas que tiene.

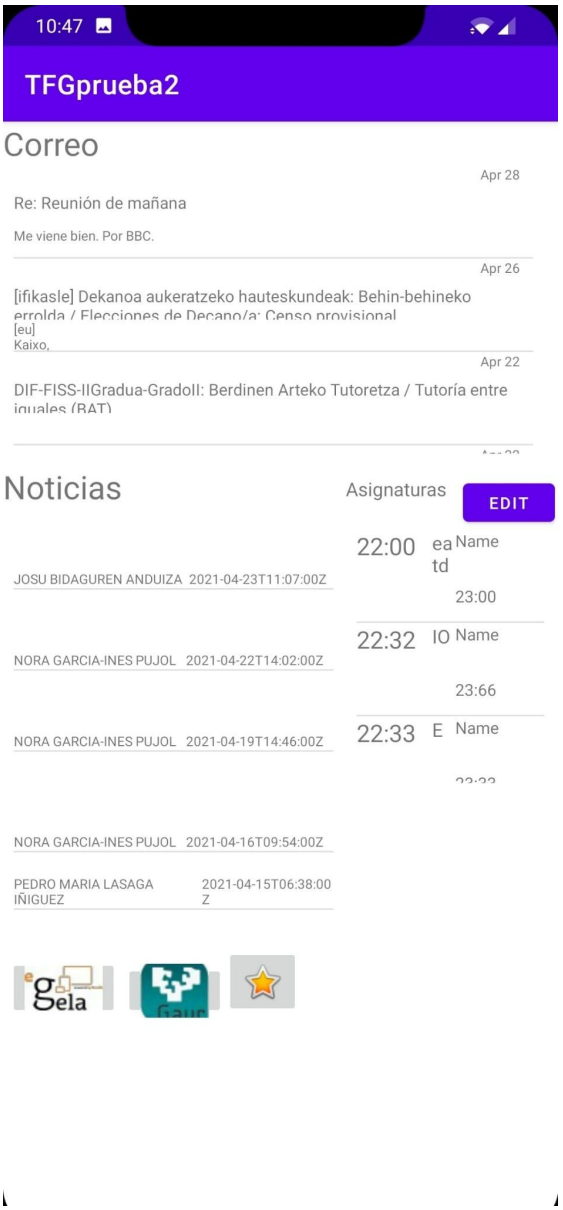


Figura 47. Primera versión (Modo claro)

La aplicación activando el modo claro en Android impedía ver algunos de los textos ya que se superpone el color.

Había que entrar ya en la parte de diseño.

Tiempos iteración 6

	Tarea	Estimado	Real
Instalación	• Android Studio y conf. Dispositivos • Librerías. • Servidor	5h 1h 1h	4h. 1h. 1h.
Seguridad	• Copia App Android • Nube App Android (GitHub) • Copia Documentación • Nube Documentación (Google Docs)	20min 1h 20min 20min	15min 43min 10min 10min
Formación	• Android básico ○ Android imprevistos • Correow • RSS • Moodle • G.A.U.R • BD	40h 20h 5h 5h 5h 5h 5h	40h. 15h 4h. 30min. 4h. 4h. 3h

Implementación	• Login
	• Correow
	• RSS
	• Horario
	  ○ Inserción
	  ○ Selección
	  ○ Visualización
	  ○ BD
	• Otros
	• Pruebas
		1h
		30h
		30h
		
		30h
		15h
		20h
		10h
		5h
		5h
			1h + 6h
			25h.
			18h.
			
			24h
			10h
			6h
			6h + 4h
			30min
			30min

Diseño	• Home • Correow • RSS • Horario • Otros • Pruebas	2h 2h 2h 3h 2h 5h	0h 0h 0h 0h 0h 0h
Documentación	• Memoria • Presentación	70h 20h	3h 0h
Reuniones	Reuniones presenciales/BBC y correos	10h	4h

Tabla 9. Tiempos iteración 6

Iteración 7 (29/04/21 - 14/05/21)

Reuniones iteración 7

29 de abril 2021 reunión de inicio de la iteración

Temas tratados

- Enseñé lo que llevaba hasta ese momento.
- Hablamos sobre mejoras y otras tareas a implementar.

Acuerdos adoptados

- Entrar en el apartado de diseño
- Solucionar bugs
- Realizar pruebas con alumnos
- Seguir investigando la agenda correow.

Resumen iteración 7

Esta iteración se centró principalmente en el **diseño y solución de problemas**. También se añadieron algunas otras características.

Me basé en un **diseño** lo más **minimalista** posible y con **colores** parecidos a los de **eGela**, tonos entre marrón, amarillo y beige. Intenté crear mis propios temas de android, para el **modo oscuro y claro**, pero esto me llevó bastante tiempo, así que en vez de crearlo desde cero, modifiqué uno de los disponibles.

Me basé, como ya he comentado, en que el usuario tenga **siempre cerca del dedo** las partes accionables como botones y en la parte más alejada la visualización de información.

Ahora al pulsar sobre una asignatura de usuario se puede ver la **información** de dicha **asignatura e información del docente**, como sus tutorías y despacho.

También implementé los **botones editables** para tener acceso rápido a diferentes webs que un usuario puede necesitar acceder, por ejemplo; su webHost, la otra página, externa eGela, que usan algunos profesores, etc.

Implementé una forma de ver el **calendario**. Intenté que se pudiera hacer zoom sobre ella, pero tardé más de lo que debía sin éxito. Esto era, como ya he comentado, debido a la forma en la que tenía hecha la aplicación con popups. Me pareció suficiente dejar el calendario sin zoom.

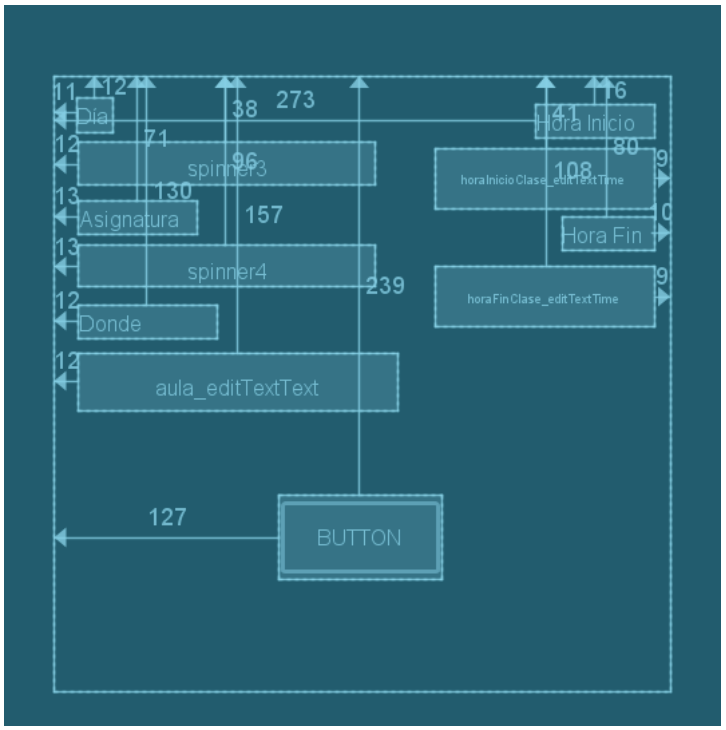
Realicé pruebas con algunos compañeros de clase tanto de cuarto como de segundo (ya que me quedaba ISO)

Tras estas pruebas con alumnos hice algunas mejoras en cuanto al diseño y funcionalidad. Puse un botón para recordar la contraseña y ldap y mejorar algunos apartados para que quedaran más claros.

La aplicación está disponible en 3 idiomas: euskera, inglés y español. Para cambiar entre ellos, por ahora, depende del idioma del smartphone. Si tu smartphone está en euskera, la aplicación estará en euskera.

Trabaje en la memoria.

Imágenes

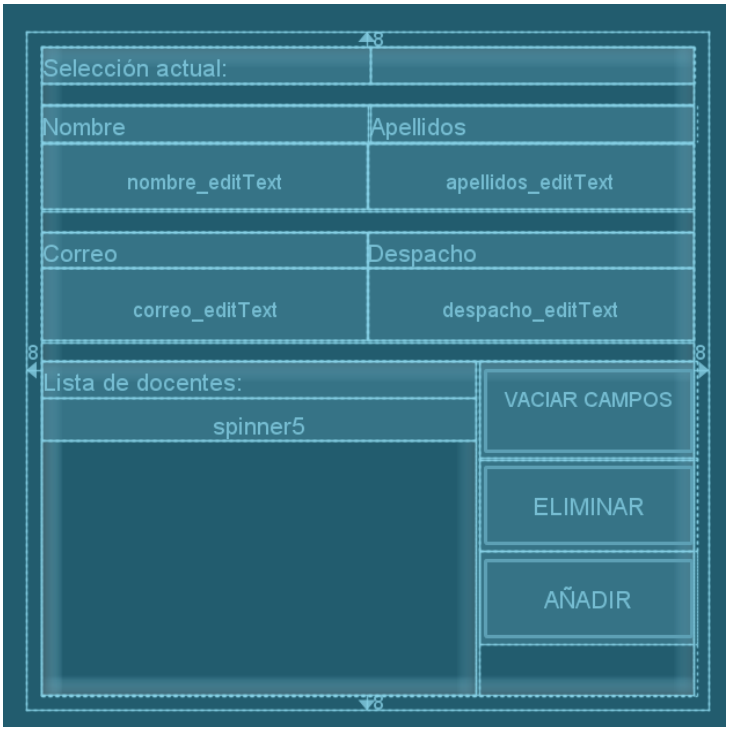


Con todos los activitys y popups se colocaron de mejor forma los elementos.

Esto también ayuda a la hora de verse en otros dispositivos.

En este ejemplo de la izquierda, los elementos podrían llegar a chocar entre ellos.

Figura 48. Elementos desordenados



Podemos apreciar en la imagen de la izquierda como queda más limpio el popup de añadir docentes.

dp es una medida que depende del tamaño de pantalla del dispositivo. Aquí cada elemento tiene su medida en esta unidad.

Figura 49. Elementos ordenados

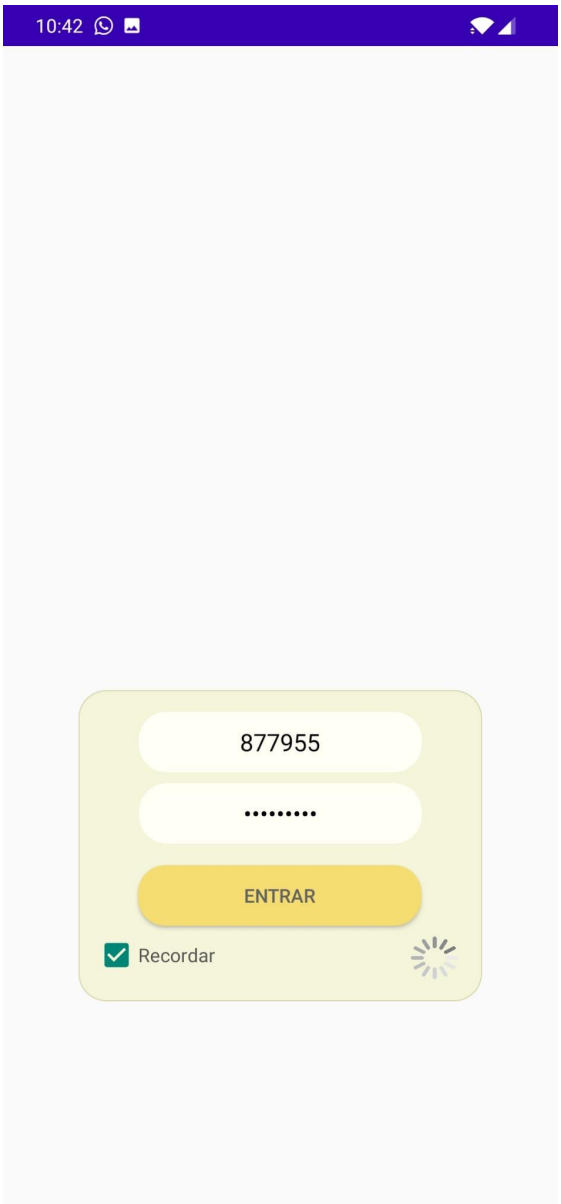


Figura 50. Login con diseño

LOGIN

Añadí el botón para recordar e hice un mejor diseño. No lo puse centrado para que estuviera más accesible al pulgar como ya he comentando anteriormente.

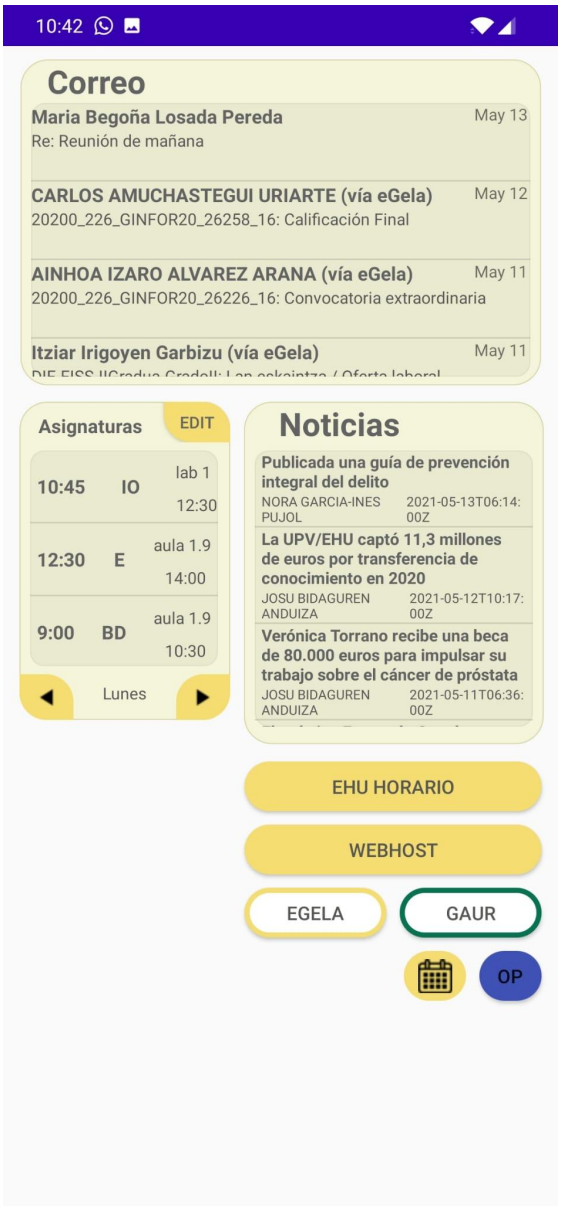


Figura 51. Home con diseño

HOME

Hice un mejor diseño. Agrupé las asignaturas por días.

Se puede apreciar, en amarillo, los botones editables y los botones para el calendario y para las opciones (OP).

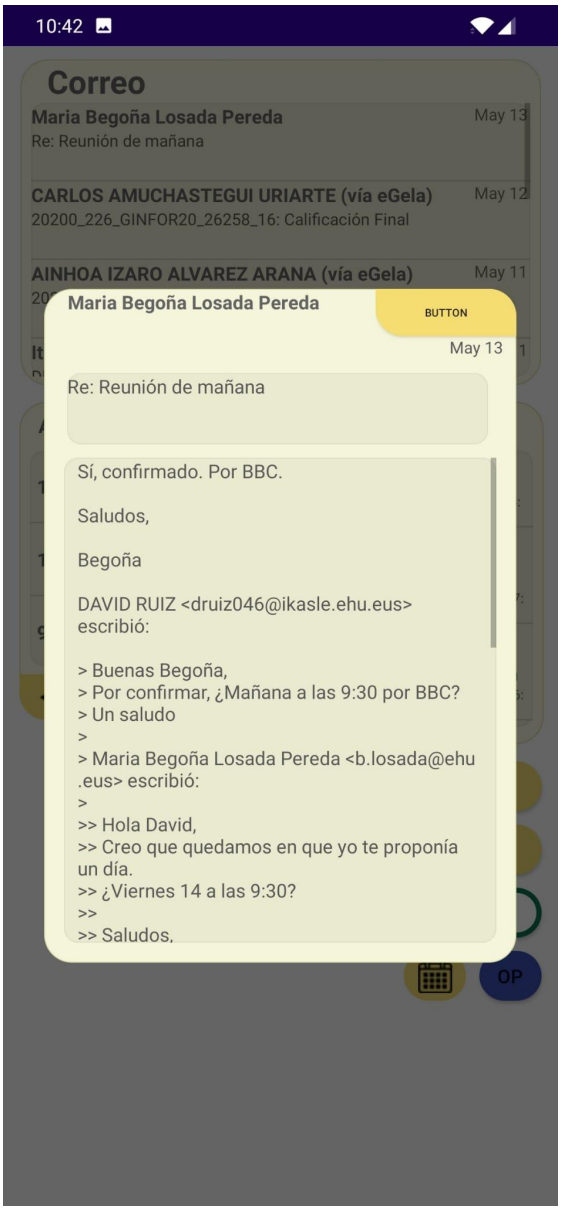


Figura 52. Correo ampliado con diseño

Correo Ampliado

A falta de darle un mejor diseño, así se mostraba en esta iteración la vista ampliada de un correo.

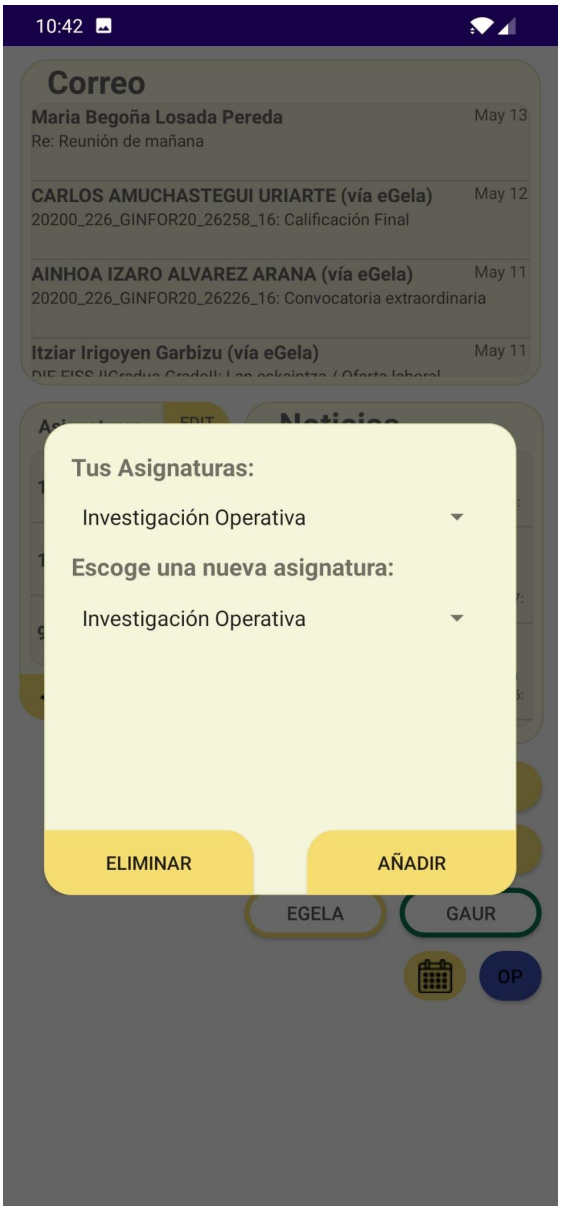


Figura 53. Selección de asignaturas con diseño

Añadir/Eliminar tus asignaturas

Dos listas desplegables para escoger y ver tus asignaturas. Se pueden añadir nuevas de la lista de todas las asignaturas y eliminar de tu lista de asignaturas.

Figura 54. Vista asignatura con diseño

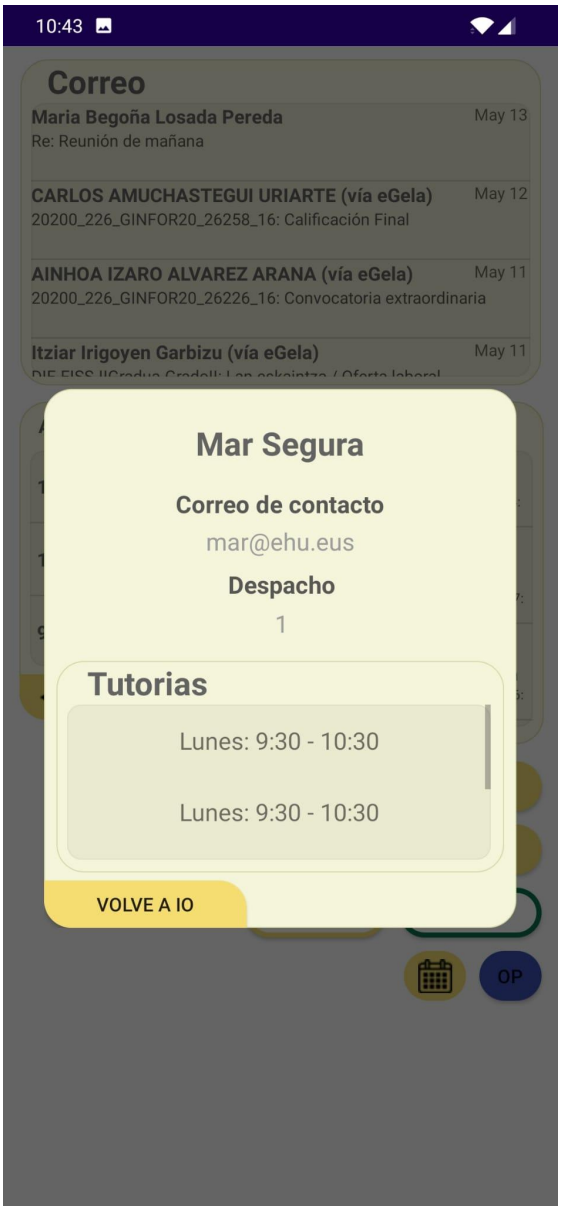


Figura 55. Vista docente con diseño

Popup información clase (asignatura)

Tras el usuario pulsar en una asignatura en el home, se abre este pop up donde se ve la información de la clase

Popup información docente y tutorías

Al pulsar en “ver docente” en popup anterior, se accede a la información del docente que imparte esa asignatura y sus tutorías. Las tutorías se repiten.

Figura 56. Calendario

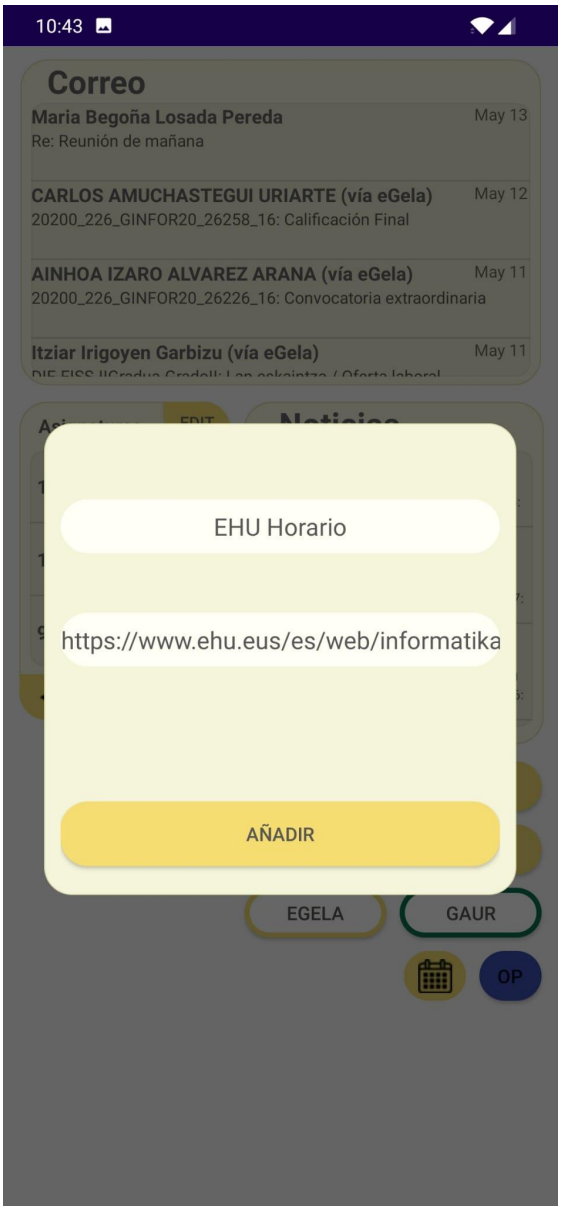


Figura 57. Botón editable con diseño

Calendario

Al pulsar al botón cuyo logo es un calendario, se abre este popup donde se ve el calendario. No se puede realizar zoom sobre él.

Accesos directos editables

A falta de darles un mejor diseño, son dos botones que un usuario puede editar, la información se guarda en el propio dispositivo. Si se mantiene pulsado el botón durante unos segundos, aparece ese popup. Si se pulsa rápidamente, abre la página web especificada.

Tiempos iteración 7

	Tarea	Estimado	Real
Instalación	• Android Studio y conf. Dispositivos • Librerías. • Servidor	5h 1h 1h	4h. 1h. 1h.
Seguridad	• Copia App Android • Nube App Android (GitHub) • Copia Documentación • Nube Documentación (Google Docs)	20min 1h 20min 20min	20min 45min 10min 10min
Formación	• Android básico ○ Android imprevistos • Correow • RSS • Moodle • G.A.U.R • BD	40h 20h 5h 5h 5h 5h 5h	40h 20h 4h 30min 4h 4h 3h

Implementación	• Login
	• Correow
	• RSS
	• Horario
	  ○ Inserción
	  ○ Selección
	  ○ Visualización
	  ○ BD
	• Otros
	• Pruebas
		1h
		30h
		30h
		
		30h
		15h
		20h
		10h
		5h
		5h
			1h + 6h
			25h
			18h
			
			28h
			12h
			12h
			6h + 4h
			30min
			30min

Diseño	• Home • Correow • RSS • Horario • Otros • Pruebas	3h 3h 3h 3h 3h 5h	2h 2h 2h 2h 2h 2h
Documentación	• Memoria • Presentación	70h 20h	9h 0h
Reuniones	Reuniones presenciales/BBC y correos	10h	4.5h

Tabla 10. Tiempos iteración 7

Iteración 8 (14/05/21 - 28/05/21)

Resumen iteración 8

Se trabajó principalmente en la memoria y se obtuvo una primera versión de la memoria. También se realizaron unos cuantos cambios en el proyecto. Se arreglaron algunos problemas para la inserción.

- Se implementó la forma de cambiar entre idiomas.
    - Refrescar la página
    - Cambiar entre idiomas
    - Recordar el idioma seleccionado
- Se hizo la parte de tareas y se escondió las noticias.
    - Se realizó una forma para añadir enlaces, por ejemplo, a grupos de Whatsapp.
    - También para añadir tareas normales.
- Se añadió URLs a las asignaturas.

Imágenes

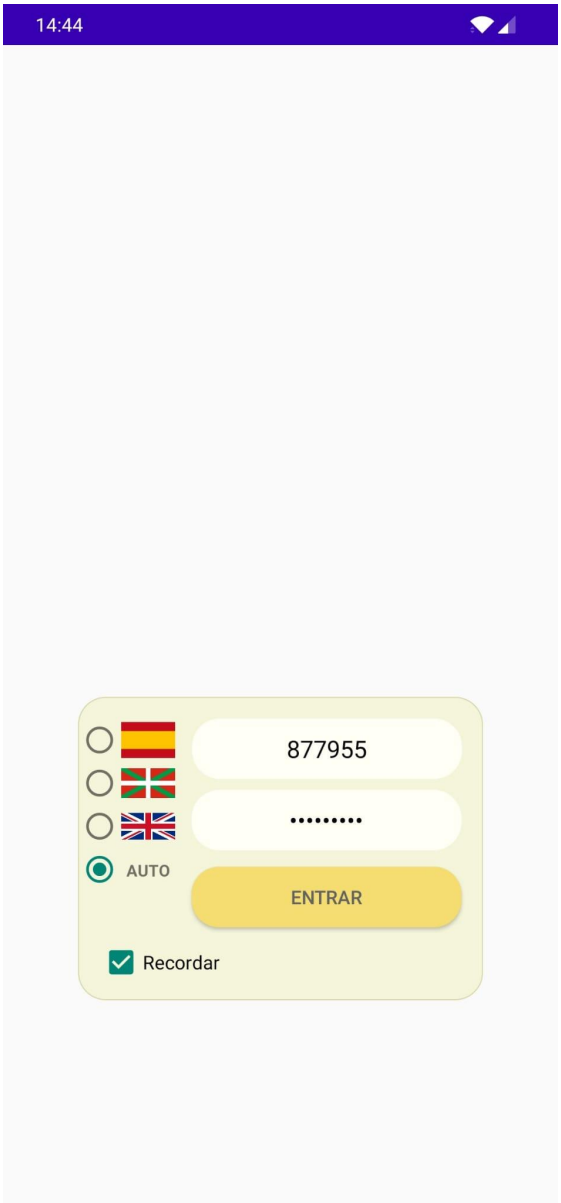


Figura 58. Login iteración 8

LOGIN

Se puede seleccionar entre el idioma español, euskera e inglés. También se puede poner en automático para que el sistema elija el que más convenga (entre esos 3) teniendo en cuenta el idioma de su smartphone.

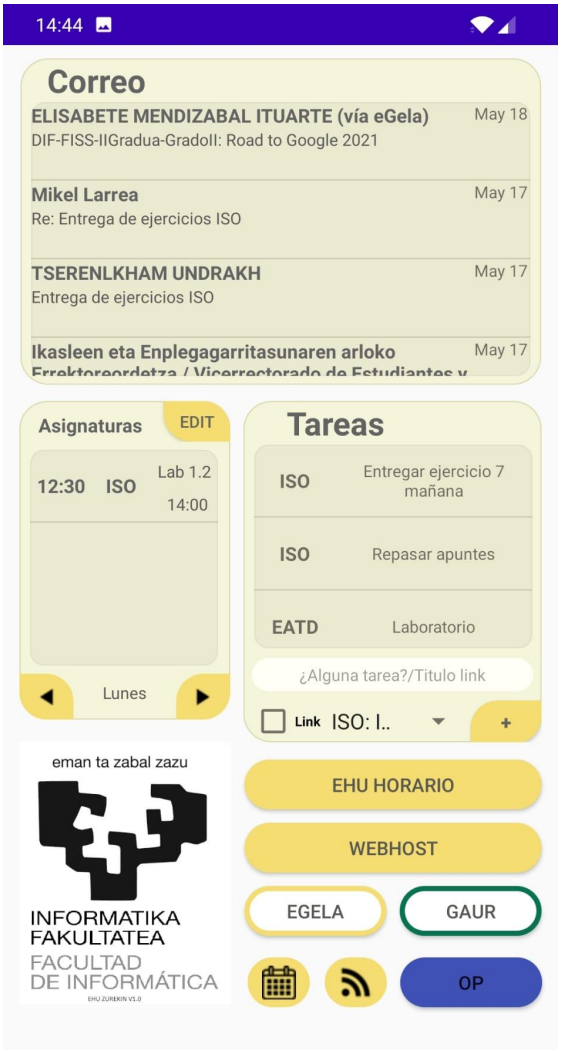


Figura 59. Home iteración 8

HOME

Ahora, en vez de los RSS, hay un apartado donde se pueden ver las tareas a realizar y añadir nuevas. Puede seleccionarse si es link o no y la asignatura. Si se pulsa sobre una tarea que sea link, por ejemplo para un grupo de whatsapp, este le llevará al link.

Las noticias están ahora en el botón con logo de RSS. (El que se parece al logo del wifi).

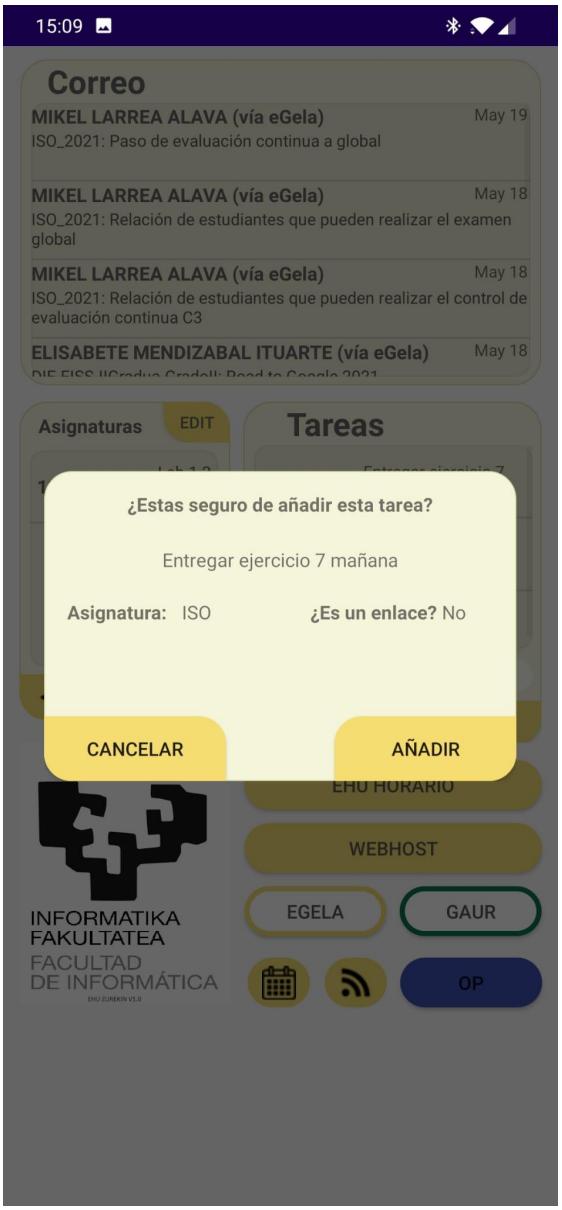


Figura 60. Confirmar tarea iteración 8

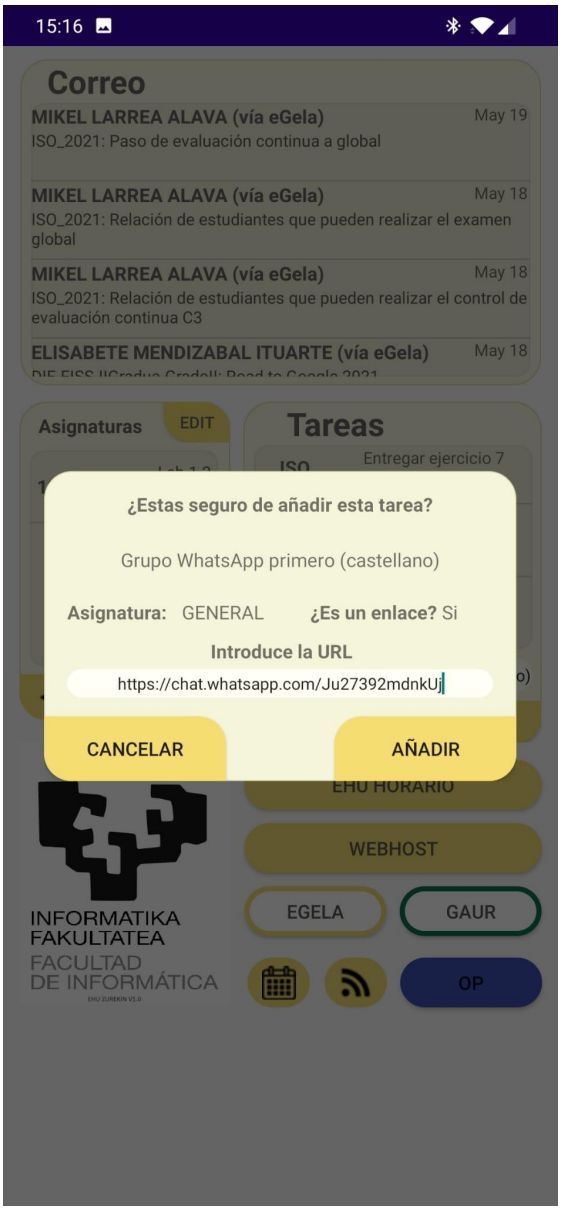


Figura 61. Confirmar enlace iteración 8

CONFIRMAR TAREA (no es enlace)

Es un pequeño popup de confirmación para añadir una tarea. Se muestra el texto de la tarea, la asignatura y si es un enlace o no.

En home habría que poner de que trata la tarea.

CONFIRMAR TAREA (es enlace)

En el caso de ser enlace aparecerá un nuevo campo para poder introducir el link de este.

En home habría que darle un nombre al enlace que posteriormente vamos a añadir.

En el ejemplo podemos ver cómo compartir, por ejemplo, el enlace de Whatsapp del grupo de primero de castellano.

Figura 62. Noticias en iteración 8

Figura 63. URLs en añadir asignatura iteración 8

Noticias

Al pulsar el botón anteriormente mencionado, se abre este popup donde podemos ver las noticias.

Añadir asignatura

Se han incluido dos campos nuevos para poner hasta dos URLs.

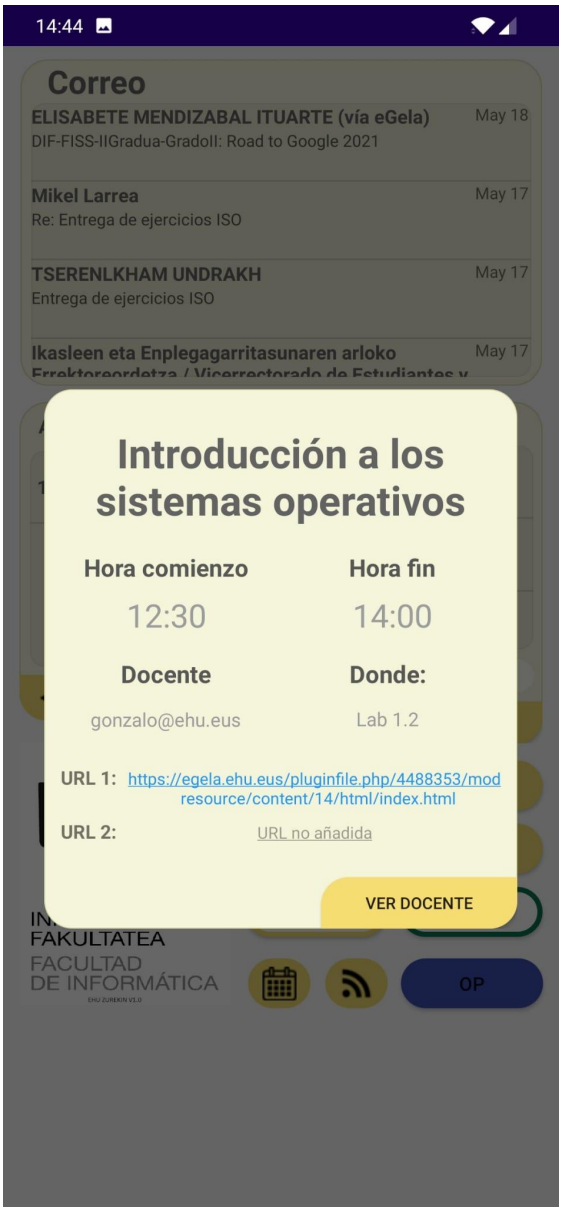


Figura 64. Vista info asignatura iteración 8

Popup información clase (asignatura)

Ahora aparecen dos campos nuevos donde aparecen, si los hay, los enlaces de la asignatura. Si los clicas te llevan a dicha página.

Reuniones iteración 8

28 de mayo 2021 reunión de fin de la iteración

Temas tratados

- Enseñé lo que llevaba hasta ese momento.
- Comentamos la memoria.

Acuerdos adoptados

- Finalizar la memoria y corregir los fallos.
- Últimos retoques para la aplicación.

1 hora de reunión.

Tiempos iteración 8

	Tarea	Estimado	Real
Instalación	• Android Studio y conf. Dispositivos • Librerías. • Servidor	5h 1h 1h	4h. 1h. 1h.
Seguridad	• Copia App Android • Nube App Android (GitHub) • Copia Documentación • Nube Documentación (Google Docs)	20min 1h 20min 20min	22min 50min 15min 15min
Formación	• Android básico ○ Android imprevistos • Correow • RSS • Moodle • G.A.U.R • BD	40h 20h 5h 5h 5h 5h 5h	40h. 20h. 4h. 30min. 4h. 4h. 3h.

Implementación	• Login
	• Correow
	• RSS
	• Horario
	  ○ Inserción
	  ○ Selección
	  ○ Visualización
	  ○ BD
	• Tareas
	• Otros
	• Pruebas
		1h
		30h
		30h
		
		30h
		15h
		20h
		10h
		10h
		5h
		5h
			1h + 10h.
			25h.
			18h.
			
			28h.
			12h.
			12h.
			6h + 4h.
			7h.
			3h.
			1h.

Diseño	• Home • Correow • RSS • Horario • Tareas • Otros	3h 3h 3h 3h 3h 3h	2h. 2h. 2h. 2h. 2h. 1.5h.
Documentación	• Memoria • Presentación	70h 20h	50h 0h
Reuniones	Reuniones presenciales/BBC y correos	10h	5.5h

Tabla 11. Tiempos iteración 8

Iteración 9 (14/05/21 - 11/06/2021) y retoques finales (20/06/2021)

Resumen iteración 9

En esta iteración trabajé principalmente en la memoria solucionando los errores que había. También hice pruebas con la aplicación y se solucionan algunos Bugs. Terminé de comentar el código.

Había un error al añadir clases y cuando se editaba un botón editable no se actualizaba. Esos errores fueron solucionados.

Hice unos retoques, como darle el logo a la aplicación, darle un nombre de versión. Al no tener mucha idea de diseñar, hice un logo bastante sencillo utilizando como base el de la EHU.

Tuve varios problemas con los índices en la memoria ya que drive no tiene demasiadas opciones para manejarlos.

Finalmente, una vez terminada la memoria, se realizó el poster y se comenzó a preparar la presentación.

Tiempo final

	Tarea	Estimado	Real
Instalación	• Android Studio y conf. Dispositivos • Librerías. • Servidor	5h 1h 1h	4h. 1h. 1h.
Seguridad	• Copia App Android • Nube App Android (GitHub) • Copia Documentación • Nube Documentación (Google Docs)	20min 1h 20min 20min	22min. 50min. 20min. 20min.
Formación	• Android básico ○ Android imprevistos • Correow • RSS • Moodle • G.A.U.R • BD	40h 20h 5h 5h 5h 5h 5h	40h. 20h. 4h. 30min. 4h. 4h. 3h.

Implementación	• Login
	• Correow
	• RSS
	• Horario
	  ○ Inserción
	  ○ Selección
	  ○ Visualización
	  ○ BD
	• Tareas
	• Otros
	• Pruebas
		1h
		30h
		30h
		
		30h
		15h
		20h
		10h
		10h
		5h
		5h
			1h + 10h.
			25h.
			18h.
			
			28h.
			12h.
			12h.
			6h + 4h.
			7h.
			3h.
			1h.

Diseño	• Home • Correow • RSS • Horario • Tareas • Otros	3h 3h 3h 3h 3h 3h	2h. 2h. 2h. 2h. 2h. 2h.
Documentación	• Memoria • Presentación • Poster	70h 20h 5h	68h. 15h. 3h.
Reuniones	Reuniones presenciales/BBC y correos	10h	6h.
FINAL			314h

Tabla 12. Tiempo final

CONCLUSIONES

Resultado final

La únicas diferencias respecto al diseño y modelo inicial es el hecho de que EHU Zurekin sea una aplicación y no un launcher y que debido a los permisos de Gaur y eGela, se use una base de datos en su lugar.

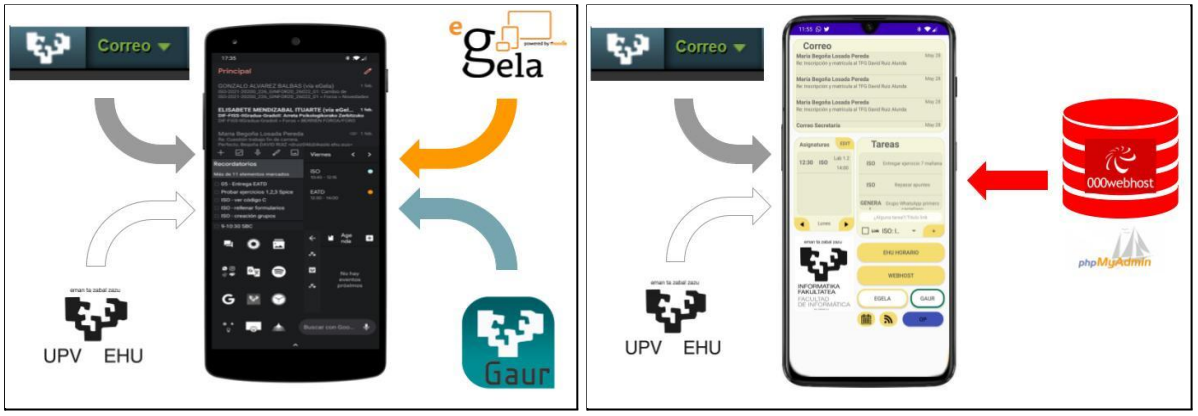


Figura 65. Diseño prototipo inicial vs aplicación final

Proyecto

Mediante este enlace se puede ver el código de la aplicación.

https://github.com/davidruizalunda/EHU-UPV-AndroidApp.git

Vídeo EHU Zurekin

Mediante este enlace se puede ver la aplicación final a modo de vídeo.

http://tiny.cc/EHUZurekin

Conclusiones sobre herramientas, lenguajes y otros

Durante el proyecto he sacado algunas conclusiones acerca de las herramientas usadas y los lenguajes.

Servidor

000webhost, en su versión gratuita, ha sido suficiente para este proyecto. No obstante, a futuro, es mejor que el servidor sea de pago en esta o en otra plataforma. El servidor actual no soportaría a tantas personas como las que tiene la universidad. 000webhost me ha dado algún que otro fallo y es un servidor bastante lento.

Android Studio

Android Studio ha demostrado ser una herramienta muy potente y muy intuitiva. Es, sin duda, el mejor entorno de desarrollo que he usado. No obstante es únicamente para la plataforma Android y no sirve para IOS. Además Android Studio es un entorno para trabajar prácticamente desde 0 y de forma nativa. En futuras versión se podría implementar la aplicación haciendo uso de Flutter, Firebase, React Native y otras ayudas para facilitar la implementación.

Java

Java ha sido una muy buena elección por la amplia comunidad que tiene pero, para futuras versiones, debería estudiarse el paso a kotlin ya que es un lenguaje que se promueve mucho por sus mejoras respecto a java. Encima, es interoperable con el código Java.

Conclusión final

EHU Zurekin 0.1 consigue el objetivo de agrupar algunos de los servicios de la EHU y lo hace de una forma fácil, rápida e intuitiva. Mejora la productividad de un alumno al poder encontrar la información que necesita al instante y teniéndola siempre actualizada. Nunca más un alumno va necesitar realizar las preguntas que vimos al principio.

No obstante, EHU Zurekin, en esta versión 0.1, no pretende salir a la luz. Tampoco es posible lanzarla debido a los permisos (lanzarla bajo el nombre de “EHU”). EHU Zurekin busca ser una primera propuesta para una posible aplicación en la que se interese la EHU. Si se consigue trabajar de mano con la EHU, obtener los permisos necesarios y se añaden más funciones y mejoras a esta aplicación, EHU Zurekin podría llegar a ayudar a muchos alumnos. Y no solo alumnos de la EHU, muchas otras universidades sufren de los mismos problemas y buscan una aplicación que agrupe todas esas páginas que tienen. El concepto de EHU Zurekin, una aplicación que agrupa los distintos servicios de la universidad, podría servir para ayudar a más alumnos y más universidades.

Dejo el proyecto abierto a que otro alumno/a pueda continuar desarrollando esta aplicación siempre y cuando se ponga en contacto conmigo para asesorarle y respete la licencia CC: BY-NC-SA.

Agradecer a Begoña por la dirección y el apoyo, a mis compañeros y compañeras que probaron la aplicación y me dieron su feedback y en en especial, a Iñigo.

MEJORAS

Ahora pasaré a comentar una serie de mejoras que se podrían implementar en futuras versiones.

1. Mejorar el código.
    a. Aplicar los conocimientos en profundidad de las asignaturas IS2, CS, etc.
    b. Sistemas de Gestión de Seguridad.
2. Cambiar servidor.
    a. Buscar un servidor mejor, más estable y rápido.
3. Conseguir permisos para usar Moodle y Gaur.
    a. Intentar usar esa información para mostrarla en la aplicación.
4. El Idioma del dispositivo se selecciona automáticamente en el login si no se marca para recordar.
    a. Utilizar alguna api para poder usar la aplicación en todos los idiomas
5. Añadir respuesta rápida a un correo.
6. Cambiar Strings fecha de la base de datos por algún tipo date y seleccionar automáticamente el día en el horario.
7. Guardar en la memoria interna del dispositivo toda la información. (En caché) Y solo acceder a la base de datos cuando ocurran cambios. Esta mejora, es para que se pueda seguir usando la aplicación cuando no haya internet.
8. Zoom en el calendario.
9. Añadir una forma de gestionar las tareas. (Tipo checklist)
10. Añadir botón con más opciones
    a. Añadir tarjeta universidad en foto (¿NFC?)
    b. Visa de exámenes
    c. Vista de notas
    d. Mensajes de eGela, entregas, transparencias... etc.
11. Modo oscuro
12. Mejorar el diseño adaptable a los dispositivos
13. EHU Zurekin para IOS
14. ¿Globalizar la aplicación? ¿Más universidades?
15. Más mejoras...

BIBLIOGRAFÍA

General

1. **Curso Android desde cero con Android Studio (La Geekipedia De Ernesto)**

   https://www.youtube.com/watch?v=tyx05coXixw&list=PLyvsggKtwbLX06iMtXnRGX5lyjiiMaT2y&index=1

2. **developers** (Google)

   https://developer.android.com/?hl=es

3. **EHU**

   https://www.ehu.eus/es/

4. **Moodle**

   https://moodle.org/?lang=es

5. **php.net**

   https://www.php.net/

6. **stackoverflow Inglés (EN-SO)**

   https://stackoverflow.com/

7. **stackoverflow Español (ES-SO)**

   https://es.stackoverflow.com/

8. **Obtener códigos de color HTML**

   https://htmlcolorcodes.com/es/

9. **javamail-android (code Google)**

   https://code.google.com/archive/p/javamail-android/downloads

10. **jsoup (jsoup)**

    https://jsoup.org/download

11. **Increíbles imágenes gratis para descargar**

    https://pixabay.com/es/

12. **Split GIF Image in frames**

    https://ezgif.com/split

Implementación correow

13. **How to fetch mails in Android using IMAP/POP3 protocols (EN-SO)**

    https://stackoverflow.com/questions/44228088/how-to-fetch-mails-in-android-using-imap-pop3-protocols

14. **Developing an email client app on android (EN-SO)**

    https://stackoverflow.com/questions/8009351/developing-an-email-client-app-on-android

15. **Learn How to use Java Mail API to send and receive emails (eduonix)**

    https://blog.eduonix.com/java-programming-2/learn-use-java-mail-api-send-receive-emails/

16. **Mini tutorial JavaMail – Leer correos con JavaMail (aepi)**

    https://asociacionaepi.es/mini-tutorial-javamail-leer-correos-con-javamail/

17. **Leer correos con JavaMail (chuidiang)**

    http://www.chuidiang.org/java/herramientas/javamail/leer-correo-javamail.php

18. **JavaMail API - Checking Emails (tutorialspoint)**

    https://www.tutorialspoint.com/javamail_api/javamail_api_checking_emails.htm

Implementación Gaur y EHU

19. **Is there any Moodle API that still exists? (EN-SO)**

    https://stackoverflow.com/questions/41572651/is-there-any-moodle-api-that-still-exists?noredirect=1&lq=1

20. **campusa Noticias de la Universidad del País Vasco (EHU campusa)**

    https://www.ehu.eus/es/campusa/noticias

21. **Creación lector RSS (Android es Fácil Creative)**

    https://www.youtube.com/watch?v=FfiRbBL6kso

Personalización

22. **Custom Popup Dialog | Android Studio (Tutorial)**

    https://www.youtube.com/watch?v=-iLG0oucUI4

23. **Android Custom ListView**

    https://www.javatpoint.com/android-custom-listview