Universitat Oberta de Catalunya

uoc.edu

On-Time app: Aplicación móvil para el registro de la jornada laboral.

Sergio Jiménez Jiménez
Máster universitario en Desarrollo de Aplicaciones para Dispositivos Móviles
Desarrollo de aplicaciones cross-platform Android / IOS

Pau Dominkovis Coll

09/10/2019

C) Copyright

FICHA DEL TRABAJO FINAL

Título del trabajo:	*On-Time app: Aplicación móvil para el registro de la jornada laboral.*
Nombre del autor:	*Sergio Jiménez Jiménez*
Nombre del consultor/a:	*Pau Dominkovis Coll*
Nombre del PRA:	*Pau Dominkovis Coll*
Fecha de entrega (mm/aaaa):	10/2019
Titulación:	*Máster universitario en Desarrollo de Aplicaciones para Dispositivos Móviles*
Área del Trabajo Final:	*Desarrollo de aplicaciones cross-platform Android / IOS*
Idioma del trabajo:	*Español*
Palabras clave	*Check in: acción por la cual se registra inicio/pause o salida de la jornada laboral.* *Jornada laboral: tiempo que cada trabajador dedica a la ejecución del trabajo para el cual fue contratado*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

Tras la implantación de la nueva ley de trabajo en España, todas las empresas deben registrar la jornada de trabajo de sus empleados. Los sistemas de control de acceso pueden ser costosos para pequeñas empresas o pymes con lo que el objetivo es ofrecer una solución mediante la cual los empleados puedan registrar su jornada laboral usando una app móvil y reducir así costes mientras se cumple la ley.

Abstract (in English, 250 words or less):

After the implantation of the new employment law in Spain, all the business must track the working time of its employees, However the acquisition of control access systems can be costly for small business that’s why the aim of this project is to create a mobile application that allow the employees to track their working time in order to reduce costs while staying in compliance with the law.

Índice

Lista de figuras

1. Introducción

1.1 Contexto y justificación del Trabajo

El Gobierno ha implantado en España una reforma laboral que modifica el Estatuto de los Trabajadores y reconoce su derecho de conocer la duración real de su jornada, así como su horario y distribución. Esto es, fichar a la entrada y la salida del puesto de Trabajo [1].

La implantación de sistemas de control de acceso puede ser costoso para pequeñas pymes y empresas sin embargo el registro mediante aplicaciones móviles o web también es válido y una solución menos costosa.

Otros proveedores ofrecen soluciones a este problema:

Fichapp [2]:
Esta aplicación se aproxima mucho a la solución que se pretende ofrecer, pero no tiene cliente para IOS por el momento

Sesametime [3]:
Ofrece control horario con geolocalización y de vacaciones entre otras funcionalidades. Es una solución completa pero no incluye la función de registrar usando la red interna de la empresa.

Control del horario [4]:
Interfaz poco atractiva, pero ofrece control de horario con geolocalización, una vez más, no hay opción a usar la red interna de la empresa.

Todas estas alternativas cubren las funcionalidades básicas que se pretenden desarrollar (fichar entradas, salidas, vacaciones) sin embargo ninguna de ellas incorpora la opción de registrar la jornada usando la red interna de la empresa en lugar de únicamente la geolocalización, esta función es mucho más rápida ya que dependiendo de la gama del teléfono, la obtención de coordenadas GPS será más o menos precisa y llevara más o menos tiempo, algo que puede ser un poco tedioso en ocasiones, sobre todo si dentro del edificio las señal GPS es débil. Este es el añadido que se pretende incorporar, fichar mientras el móvil del empleado está en la red, una opción rápida, cómoda y más precisa.

Además, On-Time app permitirá al usuario consultar sus registros y solicitor vacaciones.

1.2 Objetivos del Trabajo

El objetivo de este trabajo final de master es diseñar y desarrollar una aplicación cross-platform Android/IOS que facilite la tarea del registro de la jornada laboral para las empresas y empleados a un coste reducido. Así como un back-end que de soporte a la configuración y guardado de datos de esta aplicación.

Lo que se busca es una solución flexible que permita las siguientes funciones:

Requisitos funcionales

- Como usuario quiero poder loguearme y ver mis registros anteriores
- Como usuario, quiero poder registrar el inicio de mi jornada laboral junto con mi ubicación o información de conexión de red.
- Como usuario quiero poder cambiar mi contraseña.
- Como usuario quiero poder registrar la finalización de mi jornada laboral junto con mi ubicación o conexión de red wifi
- Como usuario quiero poder registrar salidas de la oficina durante la jornada laboral (salida a comer al médico)
- Como usuario quiero poder registrar mis periodos vacacionales, durante los cuales no hare registros de entrada/salida a la oficina (normalmente).
- Como usuario quiero poder notificar registros erróneos junto con la corrección. (sera via email al administrador)
- Como administrador quiero poder acceder a una página web en la que agregar usuarios o ver los registros.

Requisitos no funcionales.

- El proceso de registro de la jornada laboral ha de realizarse desde un dispositivo móvil.
- El proceso de gestión de usuarios solo estar disponible desde el navegador.
- Los usuarios deben ser autenticados usando el email y contraseña con el que fueron registrados.
- La información de registro de jornada será enviada al servidor, en caso de que el usuario no tenga red en ese momento, la app no funcionará.

1.3 Enfoque y método seguido

El enfoque a seguir será el desarrollo de una aplicación nueva ya que no hay ninguna alternativa de código abierto que proporcione una solución al problema y pueda ser adaptada.

1.4 Planificación del Trabajo

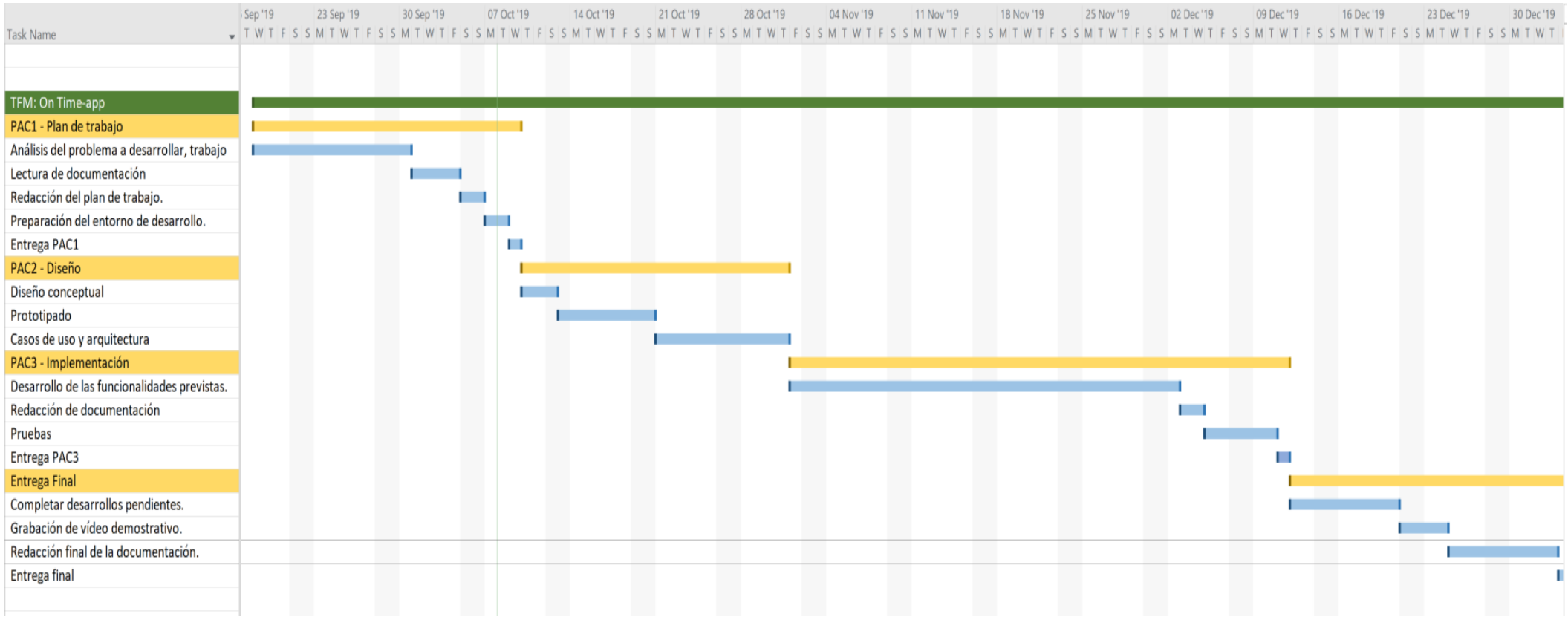
La planificación de entregas propuestas para este trabajo es, siguiendo las fechas de entregas establecidas en la página de la UOC, las siguientes:

Fecha	Nombre de la actividad	Descripción
10 de Octubre	PAC1	Plan de trabajo
30 de Octubre	PAC2	Diseño
11 de Diciembre	PAC3	Implementación
3 de Enero	Entrega Final	Memoria y video demostrativo de la aplicación
13 – 17 de Enero	Defensa virtual	Respuesta a las preguntas planteadas por el tribunal

El proyecto tiene, como fecha de inicio el 18 de Septiembre de 2019 y como fecha final de entrega el 3 de Enero de 2020. Los 108 días comprendidos entre estas fechas se dividen en 15 fines de semanas (30 días), 3 días festivos (Trabajo en Suiza, Basilea, solo son festivos los días 25, 26 de Diciembre y 1 de Enero estos festivos se contabilizan como 2h dejando un colchón de 12h) y 45 días laborables. El número de horas dedicadas los días laborables se reducirá a 2 horas máximas dependiendo de factores externos ajenos al proyecto. Los días no laborables (festivos y fines de semana las jornadas de trabajo soportarán la mayor carga de trabajo 6h y de ellas dependerá el éxito final del proyecto).

Tareas	Total Días	Horas	Fecha inicio	Fecha fin
TFM: On-Time app	108	338/350	18/09/2019	03/01/2020
PAC1 - Plan de trabajo	22	68	18/09/2019	09/10/2019
Análisis del problema a desarrollar, trabajo	13	42	18/09/2019	30/09/2019
Lectura de documentación	4	8	01/10/2019	04/10/2019
Redacción del plan de trabajo.	2	12	05/10/2019	06/10/2019
Preparación del entorno de desarrollo.	2	4	07/10/2019	08/10/2019
Entrega PAC1	1	2	09/10/2019	09/10/2019
PAC2 - Diseño	22	68	10/10/2019	31/10/2019
Diseño conceptual	3	10	10/10/2019	12/10/2019
Prototipado	8	28	13/10/2019	20/10/2019
Casos de uso y arquitectura	11	30	21/10/2019	31/10/2019
PAC3 - Implementación	42	132	31/10/2019	11/12/2019
Desarrollo de las funcionalidades previstas.	33	106	31/10/2019	02/12/2019
Redacción de documentación	2	4	03/12/2019	04/12/2019
Pruebas	6	20	05/12/2019	10/12/2019
Entrega PAC3	1	2	11/12/2019	11/12/2019
Entrega Final	23	70	12/12/2019	03/01/2020
Completar desarrollos pendientes.	9	26	12/12/2019	20/12/2019
Grabación de vídeo demostrativo.	4	16	21/12/2019	24/12/2019
Redacción final de la documentación.	9	26	25/12/2019	02/01/2020
Entrega final	1	2	03/01/2020	03/01/2020

On-Time app: Detalle de la planificación



1.5 Breve sumario de productos obtenidos

Memoria:
Documentación del Desarrollo del proyecto para su posterior evaluación

Aplicación móvil:
Producto principal, se trata de una aplicación cross-platform con lo que se generaran dos paquetes, uno para Android y otro para IOS (no se entregara ya que no estoy en posesión de la licencia necesaria pero el código será ejecutable en emulador IOS) la tecnología elegida es Flutter[5] por razones que se expondrán en posteriores apartados.

Web de gestión:
Web simple para registrar nuevos usuarios y dar soporte a la configuración de la aplicación móvil, esta web será simple y escrita en Angular, no es el producto principal, tan solo un soporte.

Back-end:
API REST que dará soporte tanto a la web de gestión como a las aplicaciones móviles será el encargado de almacenar datos y de contener la lógica de negocio.

Manual de usuario:
Documentación que describirá el funcionamiento de los productos resultantes de este trabajo.

Video de presentación final:
Breve video mostrando las funcionalidades desarrolladas.

1.6 Breve descripción de los otros capítulos de la memoria

1. **Introducción:** Esta sección.

2. **Ley de control horario:** Introducción a la nueva ley de control horario por la que se va a crear la aplicación

3. **Usuarios y contexto de uso:** publico objetivo de la app y en qué contexto tiene sentido su uso

4. **Diseño conceptual:** Mock ups y prototipos de baja fidelidad.
5. **Diseño técnico:** Diseño de la app

6. **Diseño de la arquitectura de la aplicación**: diagramas de clases y arquitectura de la aplicación así como patrones y organización de la base de datos y entidades.

7. **Conclusiones:** Breve resumen de lecciones aprendidas y objetivos alcanzados

8. **Glosario**

9. **Bibliografía**

10. **Anexos:** Contiene manuales acerca de cómo compilar y ejecutar los distintos componentes de la aplicación.

2. Ley de control horario 12/03/2019

Según la nueva ley de control horario que entró en vigor el 12/03/2019:

- Todas las empresas deberán llevar el registro de horario de sus empleados independientemente de la jornada que estos tengan.
- La empresa estará obligada a guardar el registro de horario de los empleados durante cuatro años.
- Así mismo, deberá estar disponible para los empleados y los sindicatos.
- Los empleados deben conocer la distribución y duración de la jornada laboral ordinaria.
- Los sindicatos deben conocer mensualmente las horas extras realizadas por los empleados.

Es decir, todas las empresas deberán registrar la jornada laboral de los empleados. Además, las empresas deben garantizar el registro horario de la jornada que deberá incluir el horario concreto de inicio y finalización de la jornada de trabajo de cada persona trabajadora, sin perjuicio de la flexibilidad horaria que establece en propio precepto estatutario.

3. Usuarios y contexto de uso

Para llevar a cabo el Diseño Centrado en el Usuario, se realizará un test con usuarios que nos ayudara a determinar las funcionalidades a implementar e identificar los problemas que pueden aparecer al cubrir las necesidades expuestas.

Para reducir el alcance de la aplicación, y dado que el registro de la jornada laboral puede verse afectada por varios factores (como la falta de conectividad) o casos más especiales como trabajadores nocturnos que llevan su jornada más allá de las 00:00 de un día concreto, vamos a tomar como público objetivo de los test, aquellas personas que tienen en común las siguientes características: únicamente usuarios o trabajadores de oficina donde hay conexión a internet con jornada regular de lunes a viernes en horario 8h de trabajo diurno.

Es importante este dato dado que la muestra de usuarios a la que se dirige el test determinara las funcionalidades necesarias ya que como aquí se expone las necesidades pueden variar según el sector de trabajo.

Los perfiles en los que se dividirá a los usuarios, siempre que cumplan los requisitos previos para la realización del test son:

Usuario experto. Usuario con un nivel de formación elevado con amplios conocimientos de Internet usa varias apps en su teléfono móvil, sabe lo que es un Sistema operativo y se siente cómodo configurándolo.

Usuario avanzado. Usuario con amplios conocimientos de Internet, sabe que es IOS o Android y usa varias apps sin problema.

Usuario medio. Usuario que utiliza Internet para comunicarse con sus amigos y familiares y para mantenerse informado, pero no sabe distinguir entre sistemas operativos o configurarlos.

3.1Entrevista

Documento de “screening” para el filtrado de los usuarios según los perfiles establecidos.
Los objetivos de un test de "screening" son:

- Determinar, a partir de un conjunto de preguntas, si una persona tiene las características generales y específicas necesarias para convertirse en un buen participante para la prueba.
- Obtener información sobre las personas, que permitirá identificar, clasificar y seleccionar posibles participantes para la prueba.
- Obtener información histórica (incluida experiencia, actitudes y preferencias) de las personas, lo que permitirá entender el comportamiento y rendimiento de los participantes durante el test.

3.2 Cuestionario previo

La primera sección de preguntas a realizar trata de filtrar aquellas personas que no encajan en ninguno de los perfiles deseados, además de evitar a personas no comprometidas.

Estas preguntas nos ayudan a establecer un primer filtrado:

Filtro inicial

¿Cree que le va a resultar interesante participar en este test? (Si/No)
¿Cuál es su jornada laboral habitual?
¿Trabaja usted en una oficina? (Si/No)
¿Dispone siempre de conectividad a internet durante su jornada laboral? (Si/No)
¿Registran actualmente su jornada de trabajo? (Si/No)

Manejo de aplicaciones y terminal moviles

¿Tiene un Smartphone? (Si/No)
¿Qué Sistema Operativo tiene su Smartphone? (Android/IOS/Otro/No sabe)
¿Cuántas aplicaciones utiliza habitualmente en su smartphone?
¿Sabe usted que son los permisos por aplicación y como modificarlos? (Si/No)
¿Cuánto tiempo dedica a su smartphone de media al día? (0, 4 o menos, Entre 4 y 8, 8 o más)
¿Cuál es el uso principal que le da a su smartphone? (Email, Prensa, Redes sociales (What's app Facebook ...) Juegos, Compras)

3.3 Guion de la entrevista

Perfil de usuario

- Edad
- Sexo
- Estudios
- Estado civil

Teléfono Actual

- Año de compra
- Sistema operativo
- Numero de aplicaciones instaladas

Registro jornada

- ¿Qué le interesa poder consultar sobre su jornada laboral?
- ¿Encontraría útil poder registrar ausencias como vacaciones?
- ¿Encontraría útil registrar su jornada laboral exclusivamente mediante GPS?
- ¿Le resultaría útil registrar su jornada laboral usando la red de la empresa?
- ¿Teniendo en cuenta que cualquier modificación sobre errores en los fichajes debe ser aprobado, le resultaría útil hacerlo desde la app o cree que un simple email es suficiente?

3.3.1 Entrevistados

Entrevista 1 – Mujer 28 años.

Perfil de usuario

- Edad: 28
- Sexo: Mujer
- Estudios: Universitarios
- Estado civil: Soltera

Teléfono Actual

- Año de compra: 2018
- Sistema operativo: IOS
- Numero de aplicaciones instaladas: 16

Registro jornada

- Quiero poder consultar los registros de días anteriores para en caso de necesidad poder saber cuánto tiempo debo recuperar, en ocasiones tengo algún examen de un master que estoy cursando y tengo que salir antes del trabajo, esto no es un problema, pero debo recuperar esas horas. Saber cuántas horas he trabajado cada día sería útil.
- Dispongo normalmente de 22 días laborales de vacaciones de los que puedo disponer. Normalmente nos ponemos de acuerdo en el departamento para que siempre haya alguien. Mi supervisor es quien aprueba la solicitud, pero es mera formalidad, es el departamento quien realmente se auto organiza. No suele haber problema.
- Me parece Buena idea, pero en mi oficina no siempre hay conexión GPS o Buena señal, yo creo que es algún tema relacionado con la cobertura, pero no estoy segura, normalmente uso el WIFI de la empresa.
- Definitivamente, ya que en mi oficina no se recibe muy bien la señal GPS.
- Creo que, ya que desde la app se permitirá fichar, lo suyo es que se puedan editar errores, pero un email al administrador/manager me parece bien igualmente.

Entrevista 2 – Hombre 26 años.

Perfil de usuario

- Edad: 26
- Sexo: Hombre
- Estudios: Universitarios
- Estado civil: Soltero

Teléfono Actual

- Año de compra: 2017
- Sistema operativo: Android
- Numero de aplicaciones instaladas: 26

Registro jornada

- Quiero poder consultar los registros de días anteriores y del día actual, También estaría bien saber si al pedir las vacaciones estas han sido aprobadas y ver el total de horas trabajadas en un día concreto.

- Dispongo normalmente de 25 días laborales de vacaciones de los que puedo disponer. Yo solicito los días que quiero estar de vacaciones y mi superior los aprueba o deniega.
- Fichar usando el GPS me parece suficiente en mi opinión así se sabe que estás en tu puesto de trabajo y las horas de llegada y salida.
- Me parece buena idea, pero en mi caso no creo que lo use, normalmente uso mi tarifa de datos y es más que suficiente, nunca uso la red de la empresa en mis dispositivos personales.
- Enviar 1 email con la corrección me parece suficiente, de todas formas la información de la ubicación y demás deja de ser relevante si es una corrección de hora es tu supervisor que confía en ti o sabe que es real esa corrección.

Entrevista 3 – Mujer 36 años.

Perfil de usuario

- Edad: 36
- Sexo: Mujer
- Estudios: Universitarios
- Estado civil: Casada

Teléfono Actual

- Año de compra: 2019
- Sistema operativo: IOS
- Numero de aplicaciones instaladas: 22

Registro jornada

- En mi día a día como manager suelo hacer horas extra, saber cuánto tiempo trabajo cada día sería útil así puedo saber cuántas horas me excedo de mi jornada habitual. Por supuesto consultar los fichajes que he realizado es útil. En cuanto a mis vacaciones suelo solicitarlas bajo mi propio criterio.
- Dispongo normalmente de 25 días laborales de vacaciones de los que puedo disponer. Yo solicito los días que quiero estar de vacaciones y realmente nadie los aprueba, se confía en mi criterio, pero debe quedar constancia de los días que he estado de vacaciones.
- Fichar usando el GPS me parece suficiente.
- No lo veo necesario la verdad.
- Preferiría que fuese a través de la aplicación así no me bombardearan el email con peticiones de corrección. Aunque a la hora de revisar las peticiones quizá sea más cómodo en el ordenador que en mi móvil.

Entrevista 4 – Hombre 23 años.

Perfil de usuario

- Edad: 23
- Sexo: Hombre
- Estudios: Universitarios
- Estado civil: Soltero

Teléfono Actual

- Año de compra: 2018

- Sistema operativo: Android
- Numero de aplicaciones instaladas: 24

Registro jornada

- Quiero poder consultar los registros de días anteriores y del día actual, con lo que me pagan no quiero invertir ni 1 minuto de más en la oficina.
- Dispongo normalmente de 22 días laborales de vacaciones de los que puedo disponer. Yo solicito los días que quiero estar de vacaciones y mi superior los aprueba o deniega.
- Fichar usando el GPS me parece Buena idea, pero mi teléfono tarda mucho en recibir señal es un móvil chino barato que compre en Gearbest si puedo evitar usar el GPS sería lo mejor.
- Como dije antes el GPS no me va muy bien así que WIFI es perfecto en mi caso.
- Enviar 1 email con la corrección me parece suficiente, me parece pesado hacer muchas gestiones desde el teléfono móvil la verdad que lo uso para WhatsApp Instagram y poco más.

3.4 Perfiles encontrados

Candidato	Uso Smartphone	Implicación problema de fichajes	Perfil
1	Medio	Medio	Experto
2	Alto	Alto	Experto
3	Medio	Medio	Avanzado
4	Medio	Alto	Medio

Usuario Administrador: Es el perfil de usuario con más permisos para registrar otros usuarios o asignación de trabajadores a managers.

Usuario Manager: Es el perfil de usuario con permisos medios, además de fichar, gestiona las vacaciones de su equipo o gente al cargo

Usuario Trabajador: perfil de usuario básico, solo fiche y consulta el estado de sus registros y total de horas trabajadas o peticiones de vacaciones.

3.4.1 Contexto de uso

La aplicación pretende proveer de una **funcionalidad básica** extendible en el futuro para el registro de la jornada laboral, así como los días de vacaciones y permitir al administrador extraer estos informes de cara a los sindicatos.

La aplicación está destinada a persona que trabajan con conexión a internet constante y que quieren registrar su jornada laboral en régimen estándar diurno. También está destinada a los managers o líderes de equipo que coordinan las vacaciones de los miembros de su equipo.

3.4.2 Conclusiones

Dado que la aplicación pretende solventar el problema del fichaje de horas, el usuario objetivo es el trabajador medio. Se realizan preguntas a diferentes trabajadores de entornos laborales similares al mío (oficina o factoría de software) de donde se extraen las siguientes conclusiones:

La antigüedad media de los terminales es de 2 años algunos terminales son de gama baja/ media y otros con Sistema IOS. No todos los usuarios se sienten cómodos poniendo aplicaciones relacionadas con el trabajo en sus terminales personales sin embargo todos ellos ven con Buenos ojos la posibilidad de medir el tiempo trabajado de una forma fácil desde 1 dispositivo móvil.

4. Diseño conceptual

4.1. Escenarios de uso

Escenario empleado.
A sus 26 años Sergio es un empleado de una empresa de producción de software.

En su día a día Sergio trabaja a jornada completa en dicha empresa donde tuene que registrar la hora de entrada, salida y la pausa para comer.

Durante los fines de semana le gusta irse de viaje con amigos con lo que en ocasiones solicita días libres a su empresa para extender algún que otro fin de semana.

Durante la semana suele quedar con amigos para tomar algo o jugar un partido de futbol, en ocasiones cuando queda, se le olvida registrar el final de su jornada laboral, pero al día siguiente lo corrige enviando un email a su superior.

En alguna ocasión ha tenido que viajar para trabajar en las oficinas de otra sede en reino unido, durante estos viajes registrar su jornada normalmente cuando entra y sale de la oficina.

Escenario manager.
A sus 36 años Ana es una empleada de una pequeña empresa de producción de software.

En su día a día Ana gestiona una línea de producción de la empresa donde tiene 10 trabajadores a su cargo y 2 proyectos bajo su gestión, trabaja a jornada completa en dicha empresa donde tiene que registrar la hora de entrada, salida y la pausa para comer.

Entre sus tareas está el garantizar que los proyectos cumplen la fecha de entrega y gestionar las vacaciones de los empleados que tiene a su cargo de forma que el trabajo pueda seguir llevándose a cabo en tiempo y forma.

En ocasiones cuando Sergio olvida fichar la salida de la oficina, es Ana quien aprueba la corrección sugerida por Sergio.

Escenario administrador.
Juan de 39 años es el encardo de recursos humanos entre sus funciones están el entrevistar candidatos cuando se necesita contratar a alguien nuevo o dar de alta en los sistemas a nuevos usuarios.

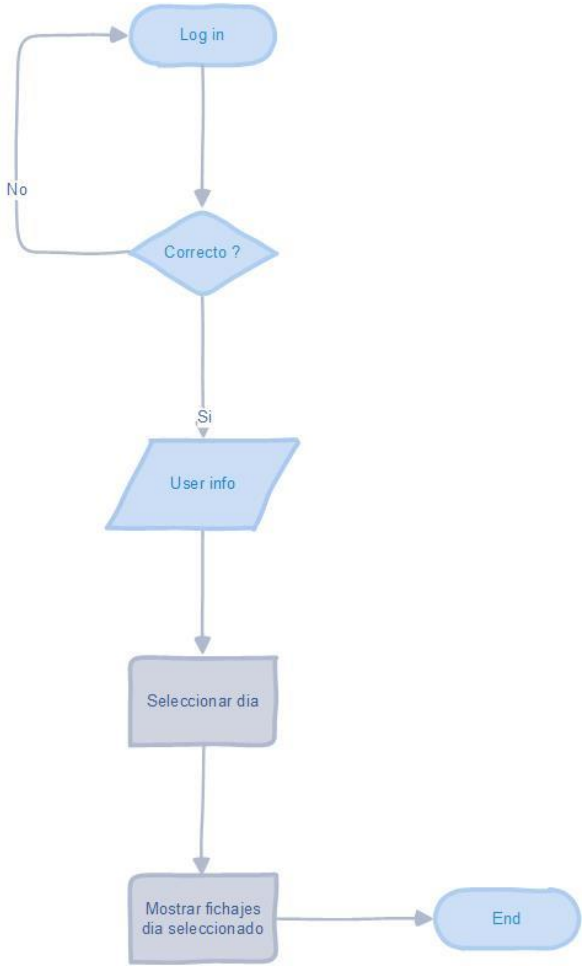
Al igual que el resto de empleados, Juan También registra su jornada laboral al llegar, salir o parar para comer.

En ocasiones tiene que extraer informes que contienen el cómputo de horas que los empleados han registrado.

4.2. Flujos de interacción

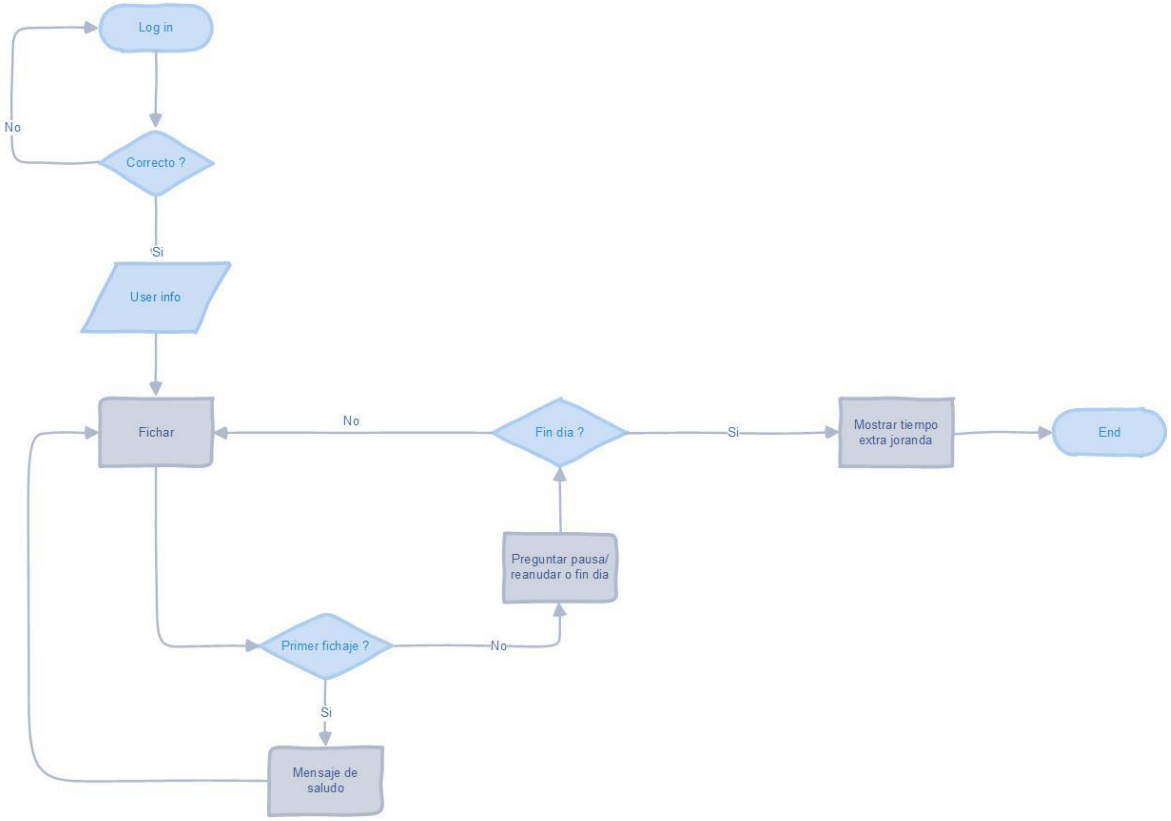
Escenario 1 - Ver Fichajes

Durante la jornada laboral el usuario se dispone a fichar, pero antes de hacerlo revisa días anteriores para comprobar que todo está en orden.



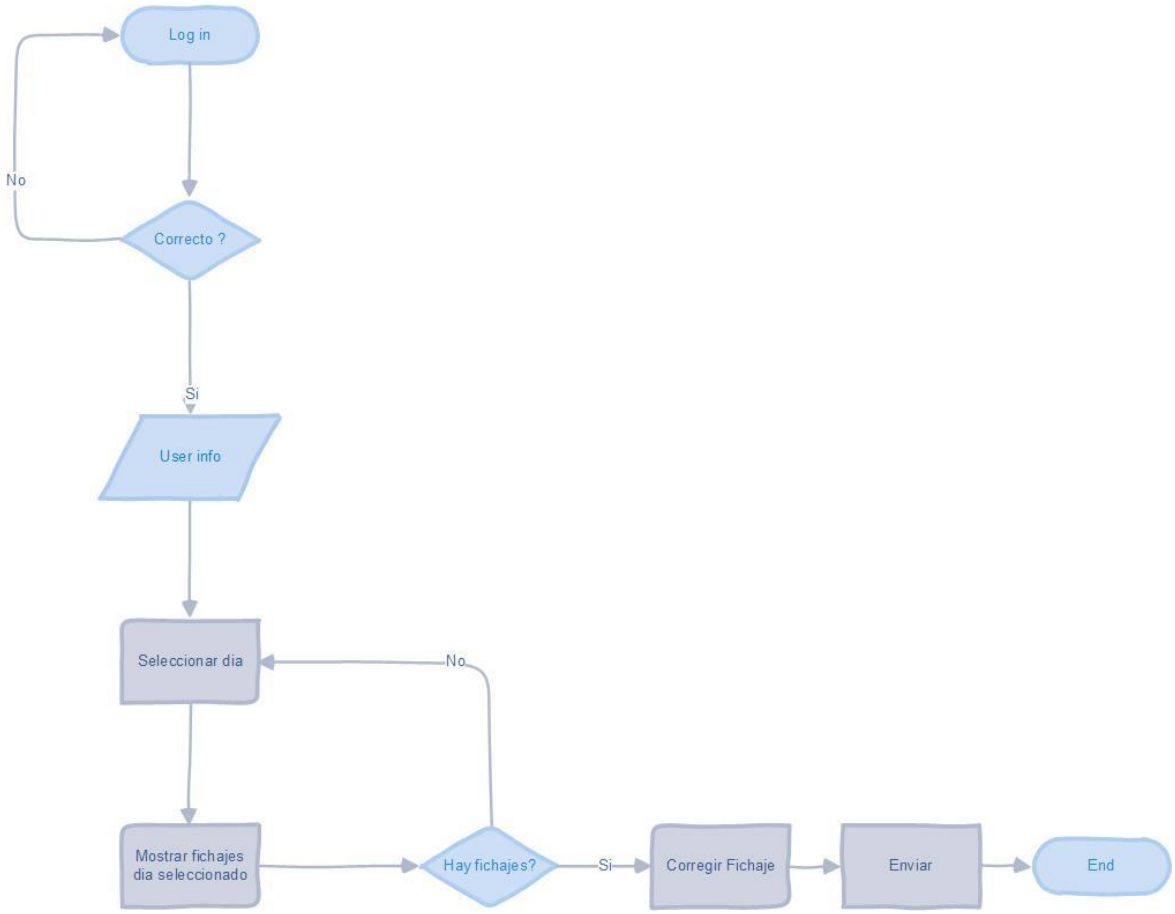
Escenario 2 - Fichar

Durante el transcurso del día el usuario fichara varias veces, al llegar, en la pausa de la comida, al reanudar su jornada y al finalizar el día.



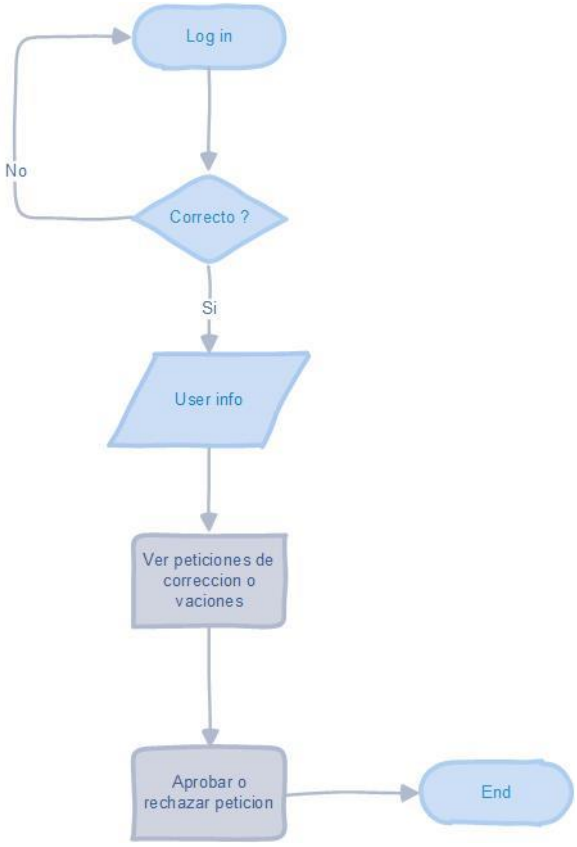
Escenario 3 - Corregir Fichaje

En ocasiones el usuario puede olvidar fichar al llegar o salir del trabajo en cuyo caso puede incluir fichajes anteriores que deberán ser aprobados a posteriori, (esta función queda fuera del alcance de la aplicación).



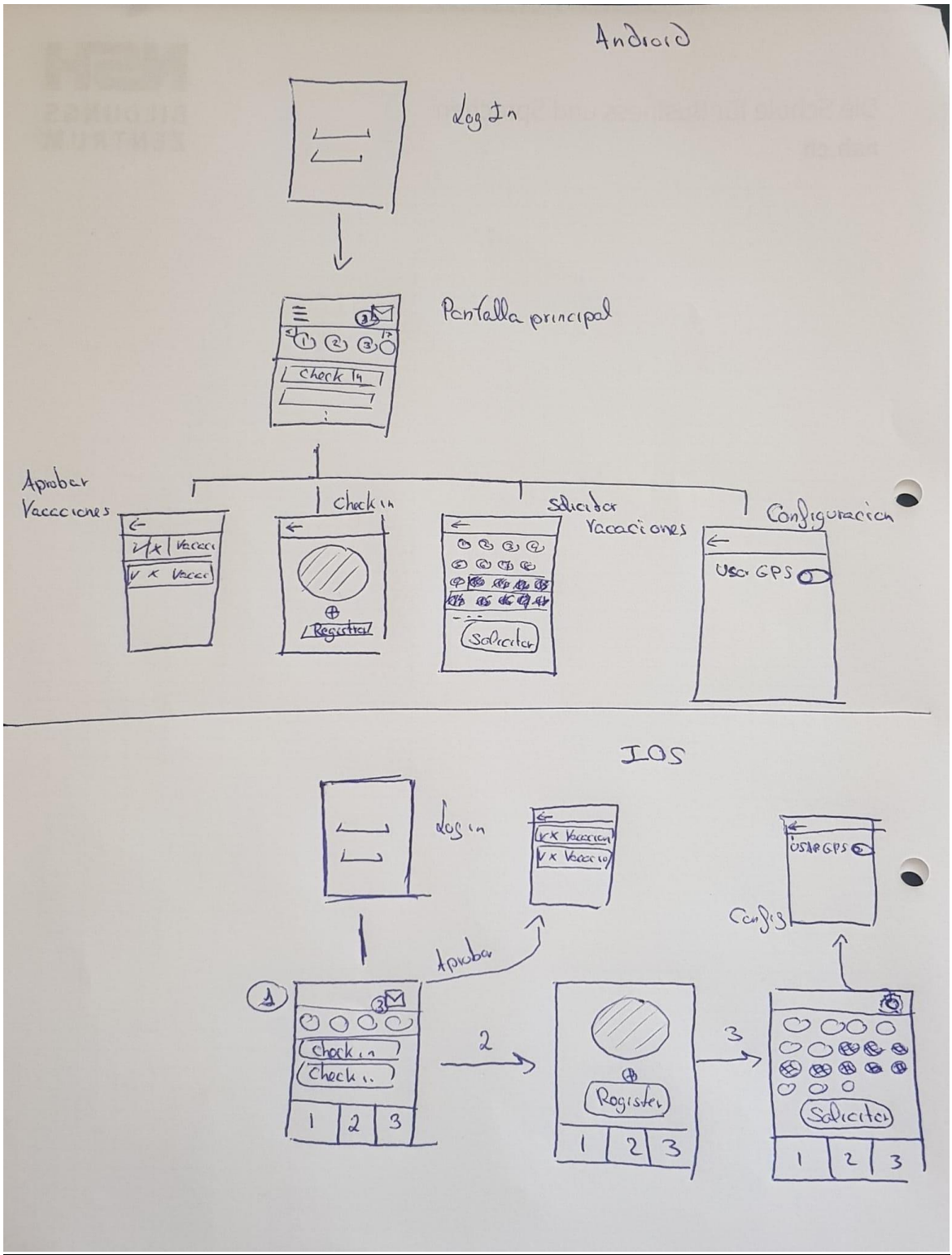
Escenario 5 - Aprobar vacaciones

El usuario con rol de manager puede aprobar peticiones de vacaciones realizadas por sus subordinados o rechazarlas.



3.2 Prototipado

Navegación, Sketch



Como se puede apreciar la principal diferencia es que en Android siempre volvemos a la pantalla de “Mi semana” mientras que en IOS esto no es necesario **siguiendo los estándares** de navegación/aplicación de cada Sistema operativo.

Prototipos comunes Android/IOS

Log in

La pantalla de log-in es común a Android e IOS, simplemente autoriza al usuario a entrar a la aplicación siempre y cuando el usuario haya sido dado de alta por el administrador vía web.

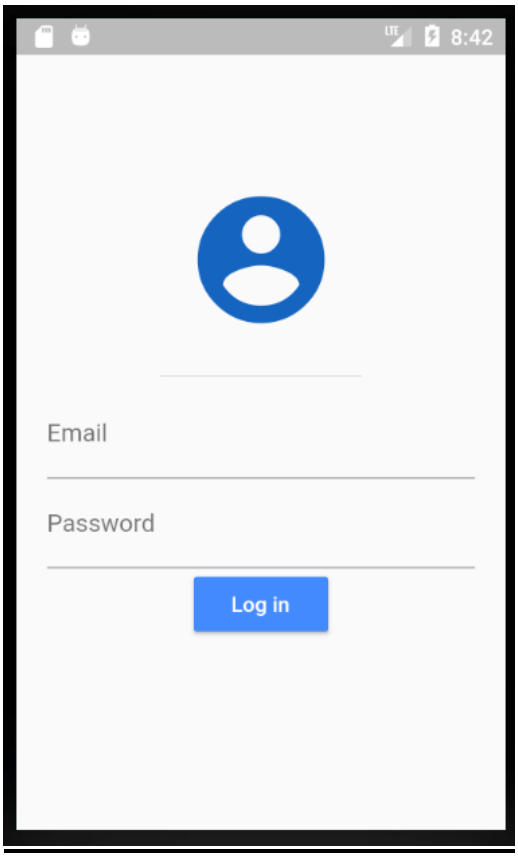


Ilustración 1: Log in app

Prototipos IOS

Pantalla mi semana

Esta es la pantalla principal donde se podrá revisar los fichajes realizados en días previos presionando sobre los días del calendario mostrado. Corresponde con la opción “1” de navegación mostrada en el sketch para IOS.

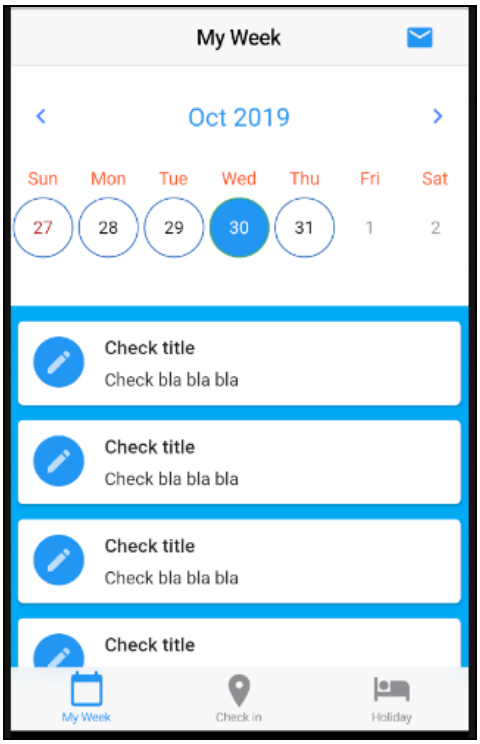


Ilustración 2: Ver Fichaje IOS

Pantalla de peticiones

Al presionar sobre el icono del mensaje en la pantalla “Mi semana” accedemos a esta donde se verán las peticiones de vacaciones u otras peticiones en el futuro. Al cerrarse volvemos a la pantalla “Mi semana”.

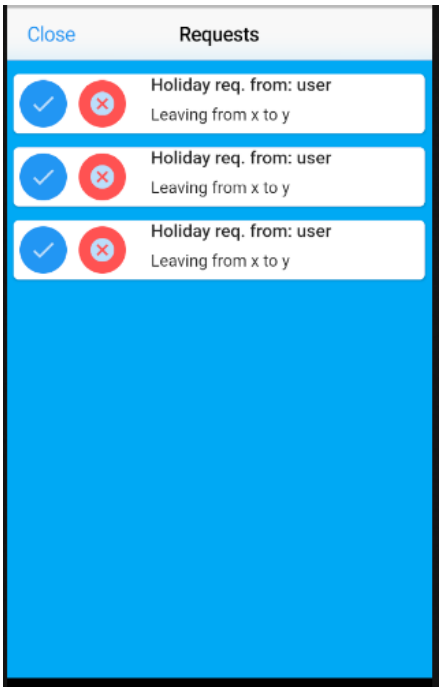


Ilustración 3: Ver peticiones IOS

Pantalla check-in

Esta pantalla puede ser la de wifi (aquí mostrada) o la de GPS (mostrada en los prototipos Android), según la configuración seleccionada por el usuario, el comportamiento y navegación se mantienen, únicamente cambia la información que es almacenada. Esta pantalla se corresponde con la opción “2” del sketch de navegación.

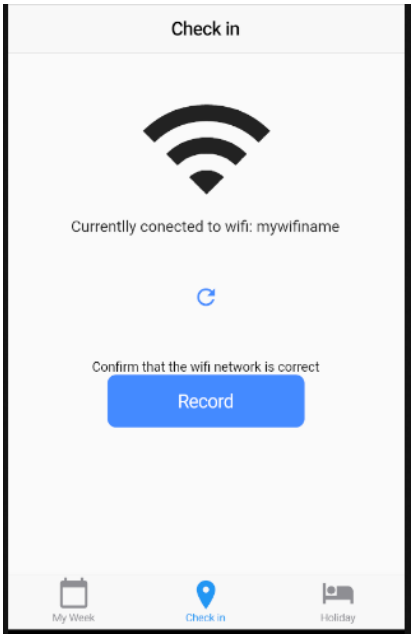


Ilustración 4: Fichaje Wifi IOS

Pantalla vacaciones

Esta pantalla es la de solicitud de vacaciones el usuario selecciona desde cuando hasta cuando desea solicitar vacaciones y presiona el botón de 'Request'. Esta pantalla corresponde con la opción "3" del sketch de navegación y desde ella se puede llegar al menú de configuración.

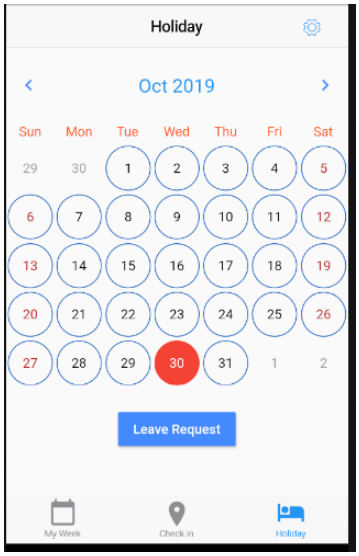


Ilustración 5: Solicitar vacaciones IOS

Configuración

Menú de configuración al cerrarlo se Vuelve a la pantalla de solicitud de vacaciones.

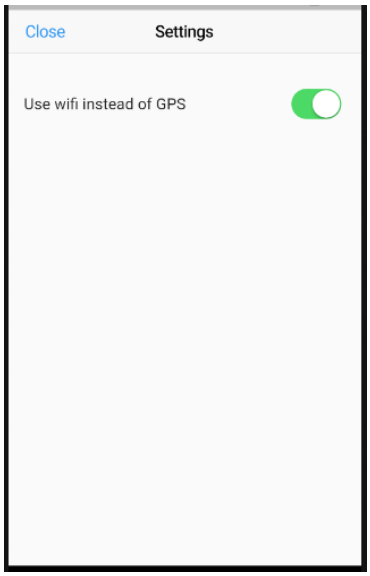


Ilustración 6: Configuración IOS

Prototipos Android

Pantalla mi semana

Esta es la pantalla principal donde se podrá revisar los fichajes realizados en días previos presionando sobre los días del calendario mostrado. Contiene el menú de navegación en Android.

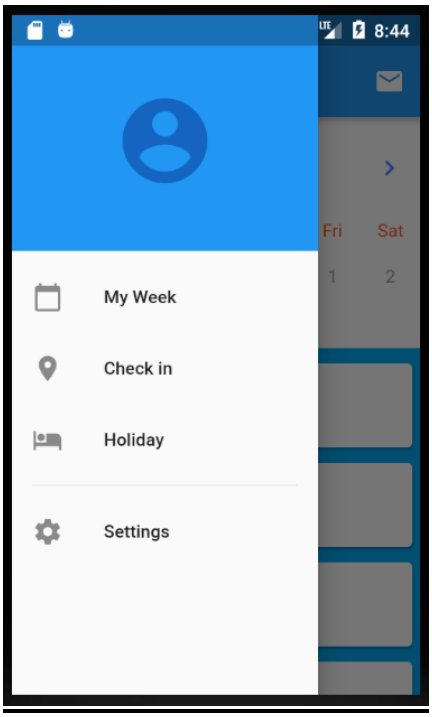


Ilustración 7: Menú Android

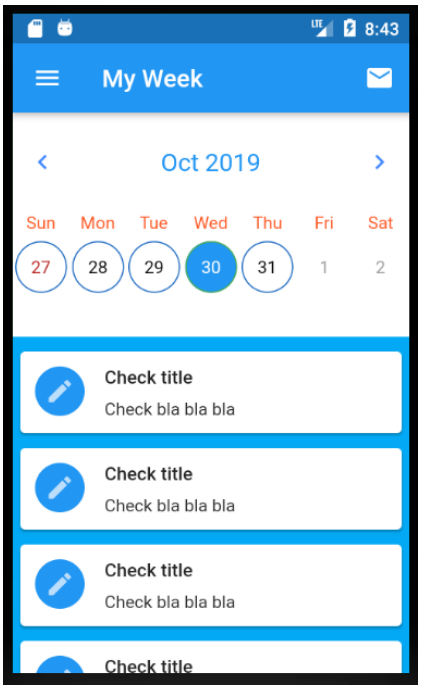


Ilustración 8: Ver Fichaje Android

Pantalla de peticiones

Al presionar sobre el icono del mensaje en la pantalla “Mi semana” accedemos a esta donde se verán las peticiones de vacaciones u otras peticiones en el futuro. Al cerrarse volvemos a la pantalla “Mi semana”.

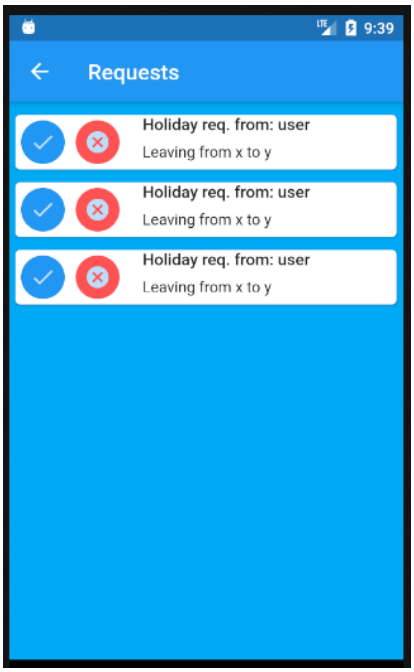


Ilustración 9: Ver peticiones Android

Pantalla check-in

Esta pantalla puede ser la de WIFI (mostrada en los prototipos IOS) o la de GPS (aquí mostrada), según la configuración seleccionada por el usuario, el comportamiento y navegación se mantienen, únicamente cambia la información que es almacenada. Al cerrarse se navega de vuelta a la pantalla “Mi semana”

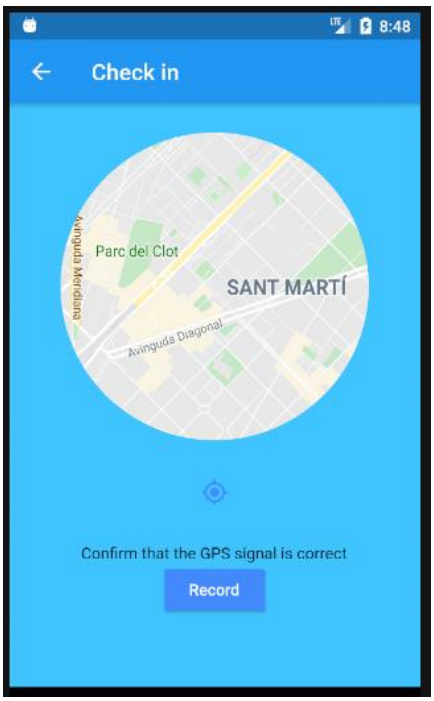


Ilustración 10: Fichar GPS Android

Pantalla vacaciones

Esta pantalla es la de solicitud de vacaciones el usuario selecciona desde cuando hasta cuando desea solicitar vacaciones y presiona el botón de ‘Request’. Esta al cerrarse navega a la pantalla “Mi semana”.

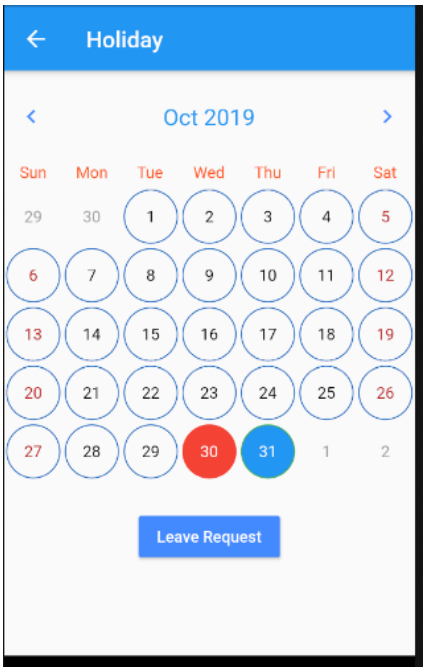


Ilustración 11: Solicitar Vacaciones Android

Configuración

Menú de configuración al cerrarlo se Vuelve a la pantalla “Mi semana”

Ilustración 12: Configuración Android

3.2 Evaluación

Cada uno de los voluntarios llevará a cabo un test de usuario de forma independiente, durante dicho test, se registrará información útil para el análisis del comportamiento del usuario.

Con los test de usuarios pretendemos detectar cuando se equivoca o se detiene el usuario, así como el por qué. Para obtener esta información usaremos el método 'think-aloud retrospectivo' lo cual consiste en instar al usuario a realizar ciertas acciones en la aplicación y posteriormente solicitamos que exprese verbalmente que no entiende o que ha pensado para llevar la acción de la forma en la que se ha llevado a cabo.

Las tareas a realizar para el test son:

1. Registre su jornada laboral actual, teniendo en cuenta las distintas paradas que a lo largo del día, entrada, comida y salida.

2. Supongamos que se quiere hacer una pausa e irse de vacaciones, notifique esta ausencia.

3. Supongamos que el terminal del evaluado es antiguo o que la señal GPS dentro de la oficina no es correcta, cambie la configuración para fichar con WIFI

Una vez finalizadas las tareas se llevará a cabo un cuestionario post test

Facilidad de uso de la aplicación (0 poco; 10 mucho) 0□ 1□ 2□ 3□ 4□ 5□ 6□ 7□ 8□ 9□ 10□
Estaba claro para que es cada pantalla en todo momento (0 poco; 10 mucho) 0□ 1□ 2□ 3□ 4□ 5□ 6□ 7□ 8□ 9□ 10□
La disposición de los elementos le parece razonable (0 poco; 10 mucho) 0□ 1□ 2□ 3□ 4□ 5□ 6□ 7□ 8□ 9□ 10□
Impresiones generales (0 muy mala; 10 muy buena) 0□ 1□ 2□ 3□ 4□ 5□ 6□ 7□ 8□ 9□ 10□
Valore la dificultad de llevar a cabo las tereas solicitadas con la app (0 poco; 10 muy difícil) 0□ 1□ 2□ 3□ 4□ 5□ 6□ 7□ 8□ 9□ 10□

5. Diseño técnico

5.1 Diagrama de casos de uso

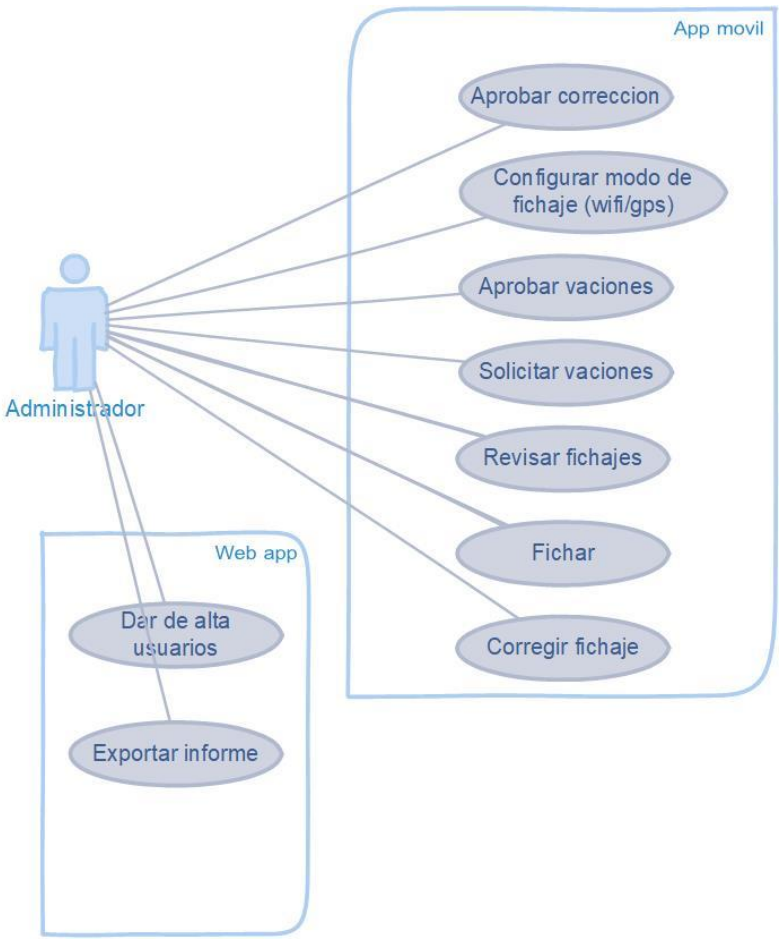


Ilustración 13: Caso de uso Administrador

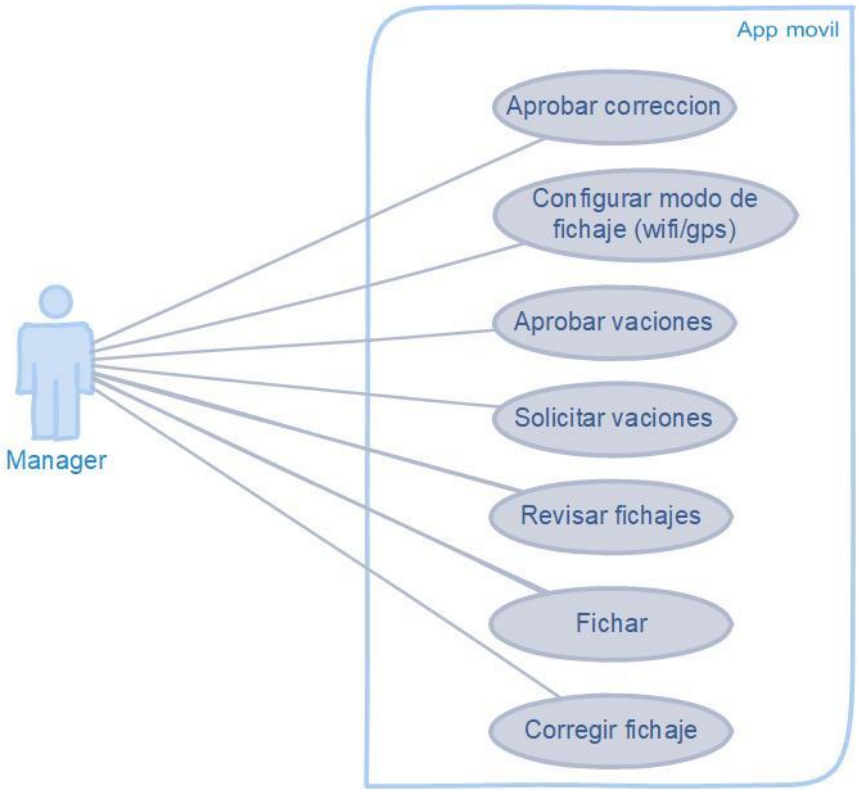


Ilustración 14: Caso de uso Manager

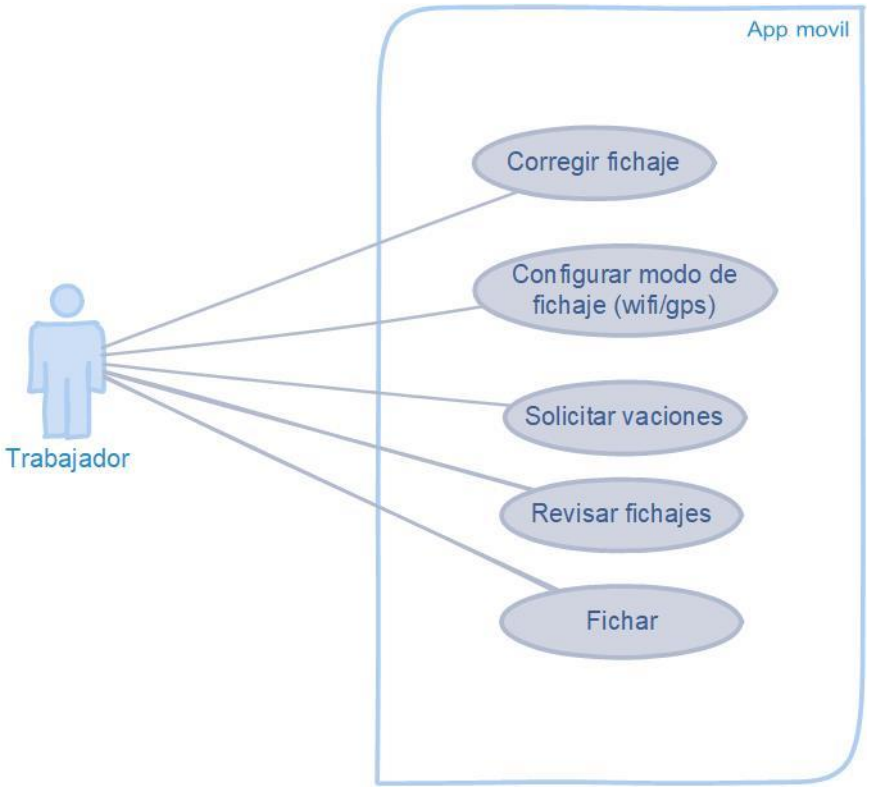


Ilustración 15: Caso de uso Trabajador

5.2 Listado de casos de uso

5.2.1 Pantalla de inicio. Usuario no registrado

Identificador	REQ_001
Nombre	Inicio de sesión de usuario no registrado.
Prioridad	Alta
Descripción	Los usuarios no registrados por un administrador en la aplicación no podrán acceder.
Actores	-
Pre-Condiciones	Usuario/email y contraseña
Proceso	El usuario escribe un usuario o email y contraseña no registrado en la aplicación
Post-Condiciones	Mensaje de error informando de usuario no registrado

5.2.2 Pantalla de inicio. Usuario registrado

Identificador	REQ_002
Nombre	Inicio de sesión de usuario registrado.
Prioridad	Alta
Descripción	Los usuarios no registrados por un administrador en la aplicación no podrán acceder.
Actores	Manager, Administrador, Trabajador
Pre-Condiciones	Usuario/email y contraseña
Proceso	El usuario escribe un usuario o email y contraseña previamente registrado en la aplicación
Post-Condiciones	La aplicación muestra el menú principal

5.2.3 Pantalla principal. Revisar fichajes

Identificador	REQ_003
Nombre	Revisar fichajes realizados
Prioridad	Alta
Descripción	Los usuarios deben de poder revisar fichajes realizados anterior mente
Actores	Manager, Administrador, Trabajador
Pre-Condiciones	El usuario ya ha realizado algún fichaje para poder revisarlo
Proceso	El usuario accede a la aplicación y en el menú principal selecciona el día cuyos fichajes quiere revisar
Post-Condiciones	La aplicación muestra el listado de fichajes que se realizó ese día

5.2.4 Pantalla principal. Revisar fichajes, editar fichaje

Identificador	REQ_004
Nombre	Editar entrada salida o pausa
Prioridad	Alta
Descripción	Los usuarios deben de poder editar fichajes anteriores.
Actores	Manager, Administrador, Trabajador
Pre-Condiciones	El usuario está registrado y accede a la pantalla principal, deben existir fichajes previos para editar
Proceso	El usuario accede a la aplicación y en la pantalla principal selecciona el día cuyos fichajes quiere editar
Post-Condiciones	La aplicación muestra la pantalla de editar fichaje y si se ha realizado correctamente la petición de edición.

5.2.5 Pantalla de fichajes. Fichar

Identificador	REQ_005
Nombre	Fichar entrada salida o pausa
Prioridad	Alta
Descripción	Los usuarios deben de poder fichar la entrada en la oficina, salida pausas
Actores	Manager, Administrador, Trabajador
Pre-Condiciones	El usuario está registrado y accede a la pantalla de Fichar
Proceso	El usuario accede a la aplicación y en el menú principal selecciona la opción para fichar, una vez el terminal recopila ubicación o red de conexión el usuario aprieta el botón de fichar y confirma la información
Post-Condiciones	La aplicación muestra la pantalla de fichar y si se ha realizado correctamente el fichaje

5.2.6 Pantalla de vacaciones. Solicitar vacaciones

Identificador	REQ_006
Nombre	Solicitar vacaciones
Prioridad	Baja
Descripción	Los usuarios deben de poder fichar la entrada en la oficina, salida pausas
Actores	Manager, Administrador, Trabajador
Pre-Condiciones	El usuario está registrado y accede a la pantalla de Fichar
Proceso	El usuario accede a la aplicación y en el menú principal selecciona la opción para fichar, una vez el terminal recopila ubicación o red de conexión el usuario aprieta el botón de fichar y confirma la información
Post-Condiciones	La aplicación muestra la pantalla de fichar y si se ha realizado correctamente el fichaje

5.2.7 Pantalla de vacaciones. Aprobar vacaciones/correcciones

Identificador	REQ_007
Nombre	Aprobar vacaciones o correcciones de fichajes
Prioridad	Baja
Descripción	Los usuarios deben poder aprobar o rechazar las solicitudes de vacaciones y correcciones de fichajes.
Actores	Manager
Pre-Condiciones	El usuario está registrado y accede a la aplicación
Proceso	Si tiene peticiones pendientes de revisión aparecerá una notificación, al tocar entrará en la pantalla de revisión de peticiones.
Post-Condiciones	Una vez en la pantalla de revisión de peticiones podrá aceptarlas o rechazarlas.

5.2.7 Pantalla de configuración. Selección WIFI o GPS

Identificador	REQ_008
Nombre	Fichar con WIFI o GPS
Prioridad	Media
Descripción	Los usuarios deben poder elegir si fichar usando WIFI cuando están en la red de la empresa o GPS.
Actores	Manager, Administrador, Trabajador
Pre-Condiciones	El usuario está registrado y accede a la aplicación
Proceso	En el menú selecciona configuración y selecciona la opción deseada
Post-Condiciones	-

5.2.9 Web de gestión. Registrar usuario

Identificador	REQ_009
Nombre	Registrar usuario
Prioridad	Media
Descripción	El administrador debe poder registrar nuevos usuarios en el Sistema y asignarles un manager
Actores	Administrador
Pre-Condiciones	-
Proceso	En la web de gestión de usuarios rellena el formulario con los datos del nuevo usuario
Post-Condiciones	-

5.2.10 Web de gestión. Informe de horas

Identificador	REQ_010
Nombre	Informe de tiempo trabajado
Prioridad	Media
Descripción	El administrador debe poder extraer un informe con las horas trabajadas por todos los empleados.
Actores	Administrador
Pre-Condiciones	-
Proceso	En la web de gestión de usuarios tocar el botón de “Generar informe”
Post-Condiciones	-

6. Diseño de la arquitectura de la aplicación

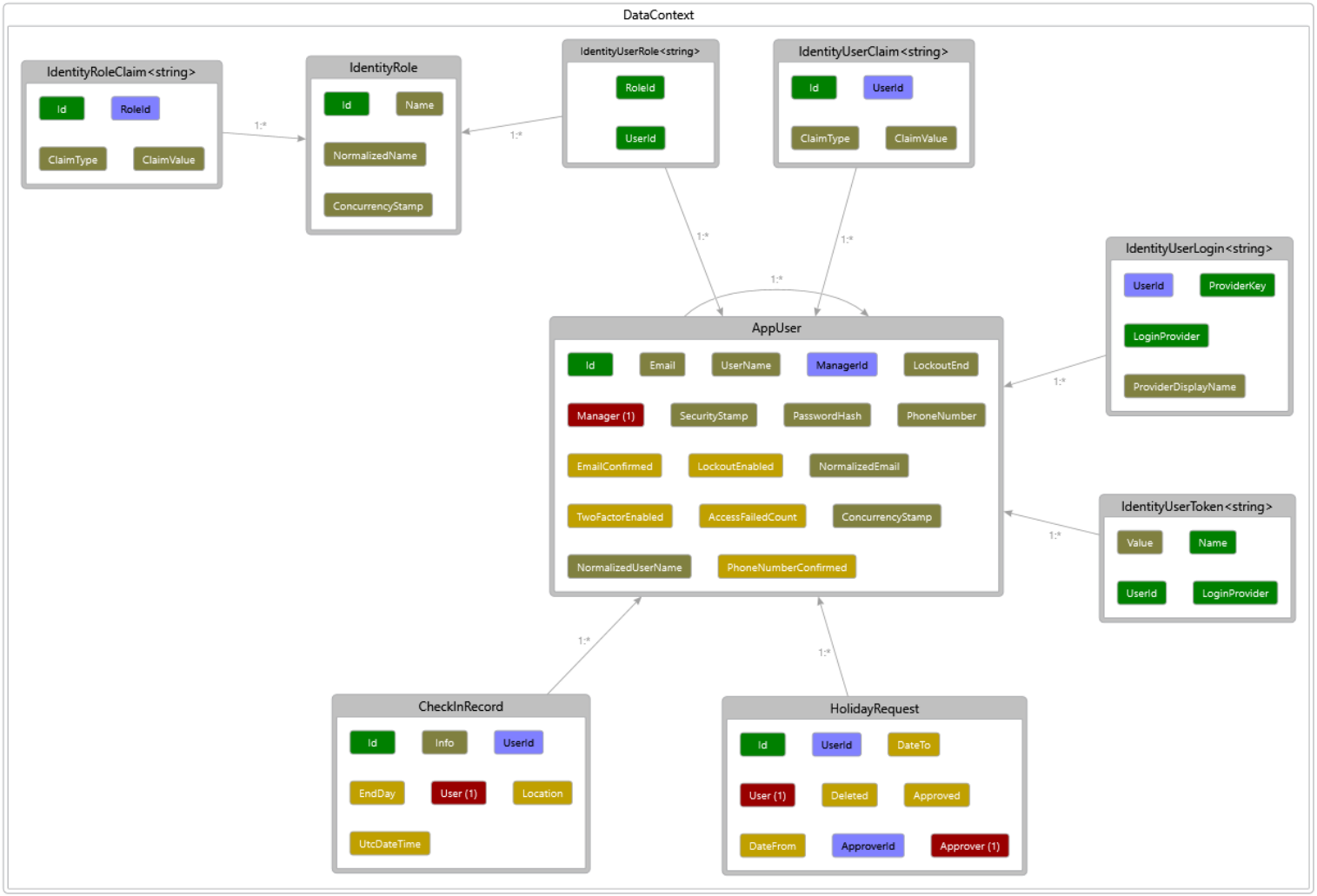
6.1 Servicio Rest

6.1.1 Diseño de la base de datos

La pieza clave del diseño de la base de datos es el AppUser, esta entidad se relaciona con los check-In y los Holiday request, el resto como los Roles o los UserClaim son entidades necesarias para la implementación de JWT como método de autenticación entre otras cosas. Además, un AppUser puede contener otro AppUser como manager de ahí la relación con sigo mismo.

6.1.2 Diseño de entidades y clases

Las entidades son las mismas que se muestran en el apartado 6.1.1 ya que se usa un ORM Entity Framework en este caso y cada entidad corresponde con su tabla en base de datos.



6.1.3 Arquitectura del sistema

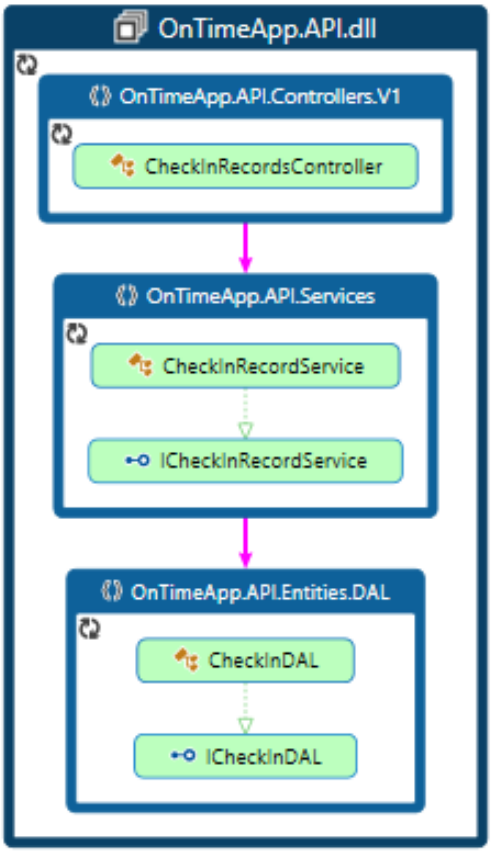
A nivel tecnológico se ha usado .net Core con C# para implementar el back-end de la aplicación o API Rest, se ha implementado autenticación con JWT[6] y se ha documentado la API Rest usando Swagger[7] para

posteriormente generar el código de los clientes que usaran la API usando Swagger codegen[8]

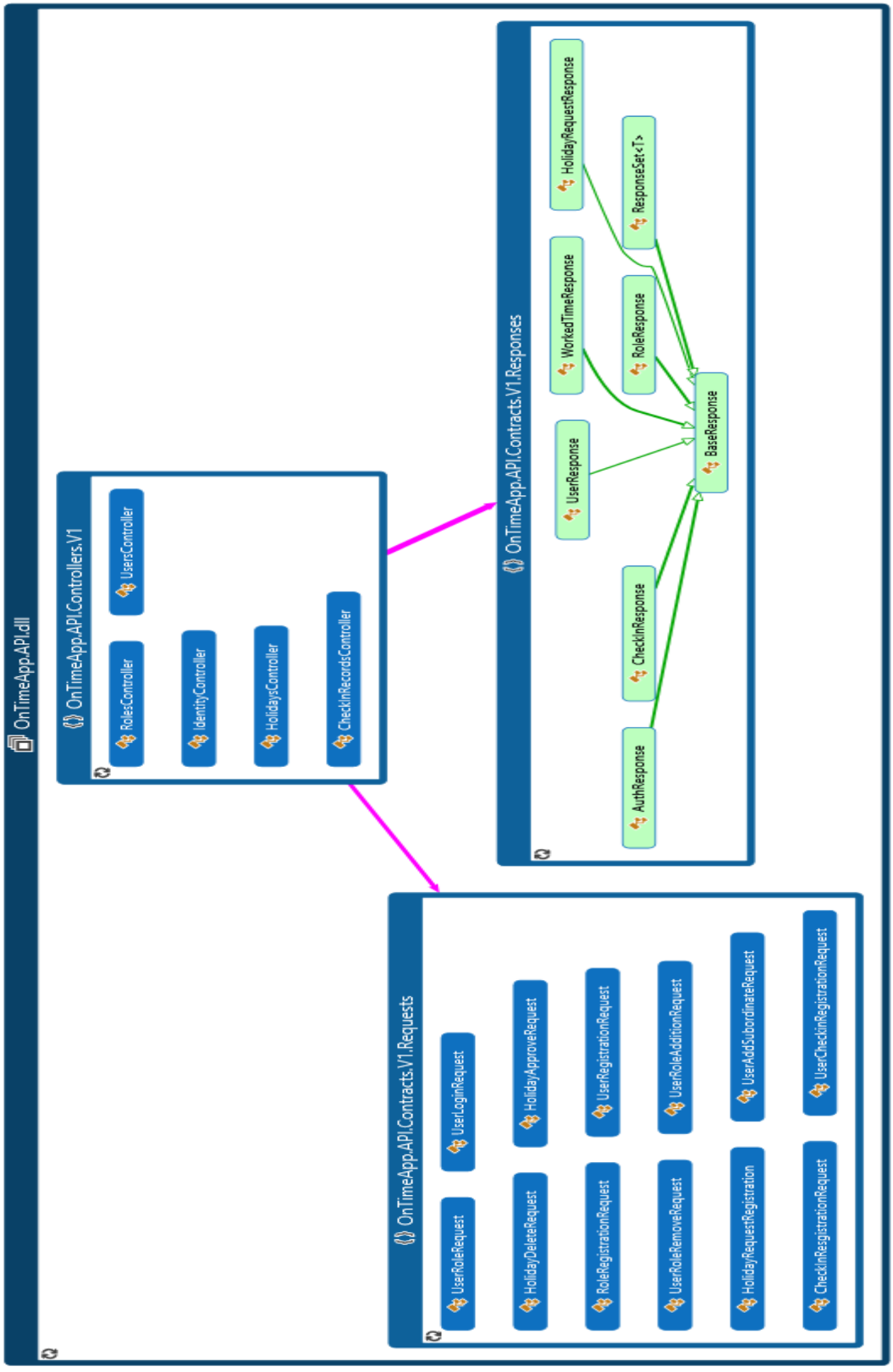
También se ha utilizado inyección de dependencias a la hora de gestionar qué implementación se debe servir cuando se necesita cierta interfaz, todo esto está forzado por MVC de .net así que no entrare en detalle y focalizaremos en explicar el N-Tier que se ha construido.

El código se ha distribuido usando una clásica arquitectura N-Tier con 3 capas, **Capa de Persistencia** usando Entity Framework en la implementación, **capa de servicio** que contiene la lógica de negocio y **capa de presentación** o (.net MVC controller en este caso) que transforma las respuestas del servicio en los contratos definidos para comunicación con los diferentes clientes de la API también conocidos como DTO[8].

Véase un sub-set de las clases y su distribución del N-Tier a continuación. “CheckInRecordsController” corresponde a la capa de **presentación**, “CheckInRecordService” a la de **servicio** y “CheckInDAL” a la de **persistencia**.



A continuacion se muestran los diferentes DTO y su interacción con los controladores:



Ventajas:

- **Swagger:** Estándar de documentación de API Rest nos permite generar posteriori el código del cliente que usara dicha API esto también es conocido como Open API en nuestro caso usamos **Open API V2**.
- **JWT:** permite replicar el back-end en diferentes nodos e incluir un balanceador de carga ya que permite hacer login contra un nodo y si la siguiente petición se redirige a otro nodo por el balanceador de carga, todo sigue funcionando, hemos creado un Stateless Rest Service
- **N-Tier:** Arquitectura sencilla que nos permite reemplazar la implementación en cada capa de forma fácil sin que las capas superiores se vean afectadas.
- **Inyección de dependencias:** Nos permite centralizar y configurar que implementación se usa y si ciclo de vida en toda la aplicación.
- **DTO:** Tras cerrar el diseño de estos contratos que serán expuestos al resto de aplicaciones, el resto de capas pueden cambiar siempre y cuando el resultado sea mapeado a estas interfaces o contratos.
- **.Net Core:** Se ejecuta en multitud de sistemas Linux/Windows/Mac al igual que java pero no hay que pagar licencia por la JVM y las herramientas de desarrollo de Microsoft son bastante potentes y convenientes, entre otras cuestiones C# es menos verboso y permite realizar código más limpio (Además es un lenguaje con el que me siento más cómodo desarrollando). También podríamos haber utilizado Python pero .net Core MVC trae varias funcionalidades como JWT de fácil configuración.

Todo esto aporta sencillez en el ddesarrollo y testabilidad ya que cada capa puede ser testeada de forma independiente al resto y sabemos en qué capa esta cada parte de la aplicación así que si necesitamos ajustar algo en la lógica de negocio solo cambiamos una capa o si se necesita ajustar algo a nivel de transacción También solo modificamos una capa, esta segregación de responsabilidad claramente definida nos ayudara a mantener la aplicación de forma que los cambios no se tengan que hacer a nivel de aplicación si no de capa.

6.2 Aplicación Móvil

6.2.1 Diseño de la base de datos

No se almacena nada en base de datos local, pero si hace uso de los llamados “shared preferences” para almacenar la configuración de usuario, por ejemplo si se debe usar solo WIFI o GPS a la hora de hacer check-in, se deja el trabajo pesado del lado del back-end o servicio Rest.

6.2.2 Diseño de entidades y clases

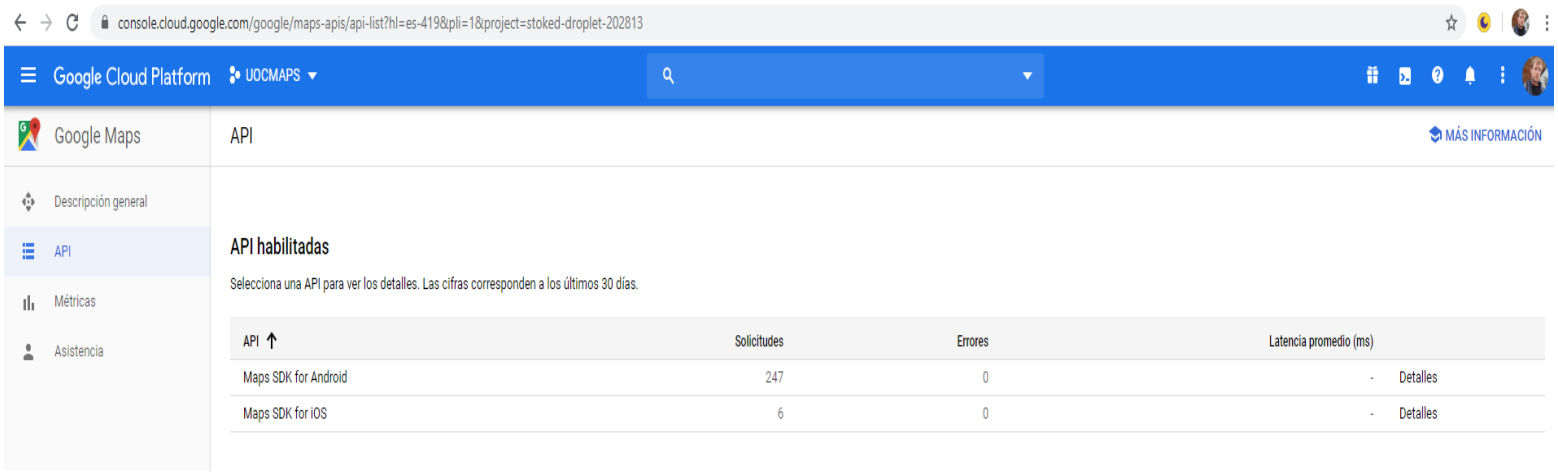
Las entidades usadas corresponden a los DTO expuestos por el API Rest definida previamente.

6.2.3 Arquitectura del sistema

A nivel tecnológico se ha usado una nueva tecnología en desarrollo de aplicaciones móviles llamada Flutter basada en el lenguaje de programación Dart.

También se ha usado Firebase a la hora de proveer de geolocalización a las aplicaciones Android e IOS.

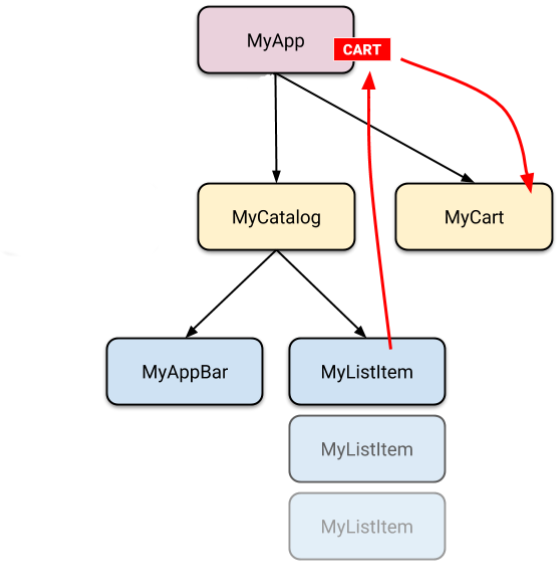
Véase a continuación el panel de administración de Google Cloud



También se hace uso de una librería autogenerada por Swagger que expone cómo interactuar con el API Rest descrito previamente.

La aplicación describe una serie de widgets con y sin estado, no se usa una arquitectura MVC o nada similar, pero sí que hace uso de un principio fundamental en Flutter que es el uso de StatefullWidgets junto con “Lifting state up”[10].

Esto consiste básicamente en que si dos widgets comparten estado, el estado se traslada al widget padre, en el siguiente ejemplo[11] se puede aprecia como el estado se traslada a widget root dado que se necesita en “MyCart” y La lista de elementos. Este principio se ha seguido a lo largo de toda la aplicación.



Además de este principio y para reducir ciertas complejidades se ha añadido también **un estado global** de aplicación que expone una serie de “ValueNotifiers”[12] u observers que notifican cuando un valor ha sido cambiado de forma que la interfaz se refresca solo al ser notificada por estos cambios.

No todo el estado de la aplicación está contenido aquí, hay elementos o estados gestionados exclusivamente por la pantalla que hace uso de ellos.

Existen multitud de formas de gestionar el estado, pero para el caso que nos ocupa esto es más que suficiente.

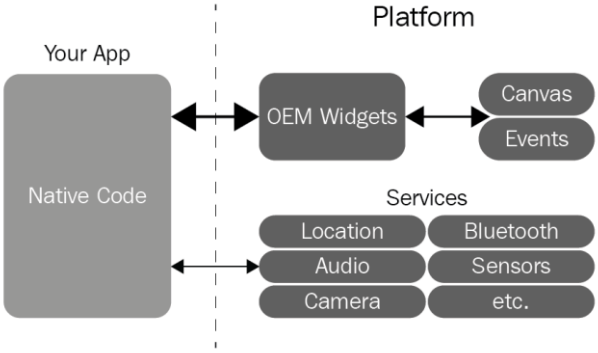
Ventajas:

Flutter: Único lenguaje y API para diversas plataformas Android e IOS con velocidad Nativa. El look and feel de cada aplicación corresponde con la de su Sistema operativo, así como la navegación y animaciones, mientras se mantiene la lógica de negocio (o comunicación con API REST en este caso) compartida entre los diferentes sistemas.

Mock ups: Los mock ups fueron creados con Flutter y reusados en la ampliación final.

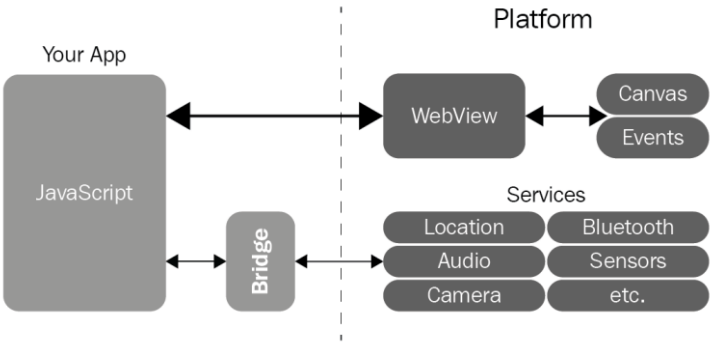
A continuación, una serie de diagramas que permiten entender las diferencias entre Flutter y el resto de Frameworks actuales y por qué se ha elegido para el desarrollo.

Native



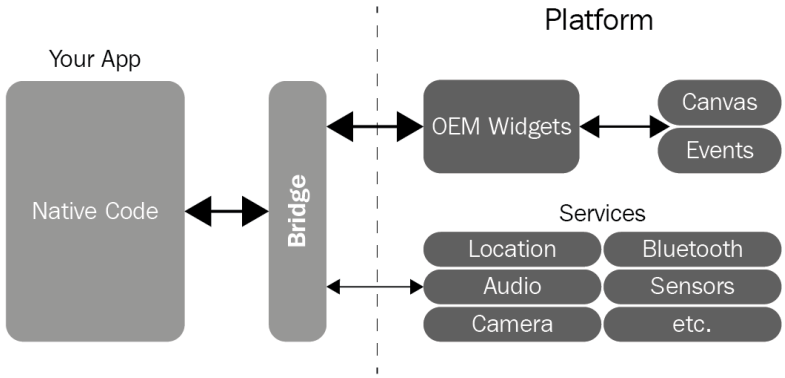
Al usar componentes nativos debemos escribir **dos aplicaciones completamente distintas,** algo que queremos evitar.

Basados en Web View (como Ionic)



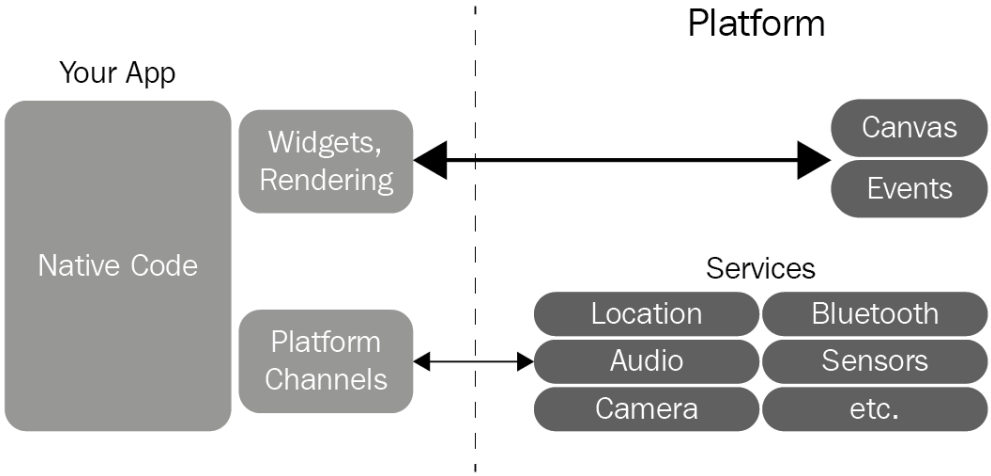
Ionic es uno de los framework que hemos visto en el master, como vemos si necesitamos hacer uso de elementos como la geolocalización necesita hacer uso de un Bride además la aplicación luce igual si no se aplican distintos css sin importar la plataforma.

Xamarin o react native



Utilizan un Bridge no solo para servicios sino también para crear los Widgets lo que puede resultar peligroso en términos de rendimiento.

Flutter



Flutter no usa ningún Bridge ni widgets específicos de la plataforma, únicamente renderiza widget en lugar de trabajar con un canvas y eventos.

Potencialmente podríamos tener apps estilo IOS en Android y al revés, es simplemente que Flutter proporciona librerías de widgets que actúan y lucen como la plataforma a la que van dirigidos.

Además, Flutter **puede realizar AOT compilación que transforma el código Dart en código nativo binario** lo que hace que se ejecute muy rápido y el redibujado de la interfaz tenga un buen rendimiento.

Durante el desarrollo junto con el APK se empaqueta la máquina virtual de Dart lo que permite **live reloading** de la aplicación, una vez en release esta máquina virtual no va con la app.

La app que se adjunta en este trabajo es una versión debug ya que para ejecutarla en el emulador de Android únicamente funcionan las versiones en debug, las versiones release no son funcionales a día de hoy en el emulador.

6.3 Página web

En este apartado vamos a ser muy generalistas ya que no es el objetivo principal de este trabajo y es más un complemento o demostración de lo que es posible, una vez más se ha generado usando Swagger un cliente typescript

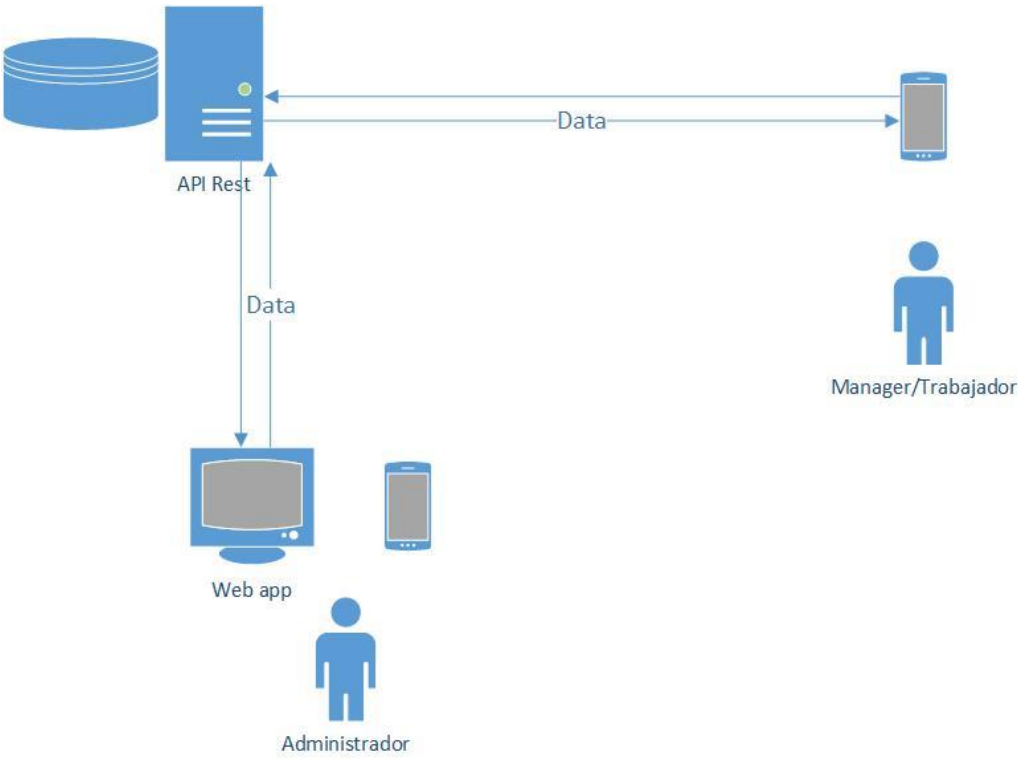
No se sigue ninguna arquitectura en particular ya que es un pequeño desarrollo para dar soporte a la configuración y demostrar las posibilidades del Sistema. Al estar escrita en Angular se siguen las buenas prácticas del framework, para saber más acerca de la arquitectura del framework y de la arquitectura que esta fuerza sobre nuestra web se puede visitar: https://angular.io/guide/architecture

Ventajas

Angular: Usar angular da un set de utilidades consistentes dado que hay mucha variedad de herramientas para el Desarrollo web Angular nos da un marco de trabajo consistente.

Typescript: sintaxis parecida a C# y se transpila a JavaScript sin tener que preocuparnos de que el navegador entienda o no las funcionalidades que de JavaScript que se usan.

A continuación, se muestra un diagrama de como interactuaran los diferentes componentes del Sistema de forma muy simplificada, ténganse en cuanta que al tratarse de un API Rest sin estado, el back end puede ser escalado e incluir balanceadores de carga, aquí mostramos la más simple de las configuraciones posibles usada durante el desarrollo:



7. Conclusiones

Conclusiones generales

Durante la realización del trabajo **hemos aprendido a desarrollar aplicaciones en Flutter**, este conocimiento es sin duda algo preciado, también hemos aprendido que la gestión de proyectos es más compleja de lo que parece y seamos honestos aun teniendo una planificación, cuando se lleva a cabo otro trabajo a tiempo completo es muy difícil de seguir, en mi caso han surgido varios viajes por trabajo que me han impedido seguir la planificación necesitando entonces por mi parte un esfuerzo extra en periodos vacacionales.

Si bien es verdad que en una planificación pueden surgir imprevistos, esto no me parece realista ya que al desempeñar otra actividad a tiempo completo es muy complicado que se pueda seguir dicha planificación. Aun teniendo en cuenta el horario laboral.

Reflexión critica

Se han logrado todos los objetivos, sin embargo, algunas funcionalidades se han recortado como es el caso de la edición de fichajes erróneos.

El recorte de funcionalidades o reducción del alcance es debido a que en un inicio se fue demasiado ambicioso, una aplicación para Android e IOS más un back-end y una web son demasiado pera tan corto periodo de tiempo, aun así, **se han entregado todos los elementos a los que nos comprometimos** con posibilidad de expandirlos.

Análisis critico

Como comentaba en párrafos anteriores seguir la planificación no ha sido posible viéndose alterada y necesitándose una inversión extra de tiempo durante el periodo vacacional.

7.1 Líneas de trabajo futuro

La aplicación web puede mejorarse enormemente dando funciones avanzadas de gestión de usuarios, cambio de contraseñas ... etc. además de permitir a los usuarios realizar las mismas acciones que en la app móvil, pero en formato web, muchos usuarios preferirán este formato ya que no tendrán que instalar aplicaciones de trabajo en sus dispositivos personales.

Edición de fichajes erróneos es una funcionalidad que hemos dejado de lado para reducir la complejidad tanto de la aplicación móvil como del back-end en un futuro podría incorporarse.

Se pueden incluir más funciones de gestión de personal como por ejemplo una vista que muestre qué miembros del equipo están en la oficina o están de vacaciones, pero para ello habría que replantear la distribución de pantallas en IOS

Notificaciones para usuarios cuando se aprueban o deniegan peticiones en lugar de ver el cambio directamente en la pantalla de vacaciones sin ningún aviso.

8. Glosario

- **DTO:** del inglés data transfer object, objetos usados simplemente para transferencia de información deben ser de solo lectura y ser serializarles para poder ser llamados DTOs.

- **Swagger/Open Api:** Open Api es un estándar de documentación de API REST que nos permite generar el código fuente de los clientes que consumiran dicho API

- **JWT:** JSON Web Token (abreviado JWT) es un estándar abierto basado en JSON permiten la propagación de identidad y privilegios o claims en inglés. Por ejemplo, un servidor podría generar un token indicando que el usuario tiene privilegios de administrador y proporcionarlo a un cliente.

9. Bibliografía

Diagramas de arquitectura de frameworks móviles y Flutter provienen del libro: https://www.packtpub.com/application-development/google-flutter-mobile-development-quick-start-guide

1. https://www.up-spain.com/blog/ley-control-horario-pymes/
2. http://fichapp.es/
3. https://www.sesametime.com/es/
4. https://www.controldelhorario.com/
5. https://flutter.dev/
6. https://es.wikipedia.org/wiki/JSON_Web_Token
7. https://swagger.io/
8. https://github.com/swagger-api/swagger-codegen
9. https://en.wikipedia.org/wiki/Data_transfer_object
10. https://github.com/brianegan/flutter_architecture_samples/tree/master/vanilla
11. https://flutter.dev/docs/development/data-and-backend/state-mgmt/simple
12. https://api.flutter.dev/flutter/foundation/ValueNotifier-class.html

10. Anexos

No se explicara como instalar cada una de las dependencias necesarias, dado que todas ellas tienen una página web que explica cómo hacerlo, esta sección está dedicada a explicar cómo ejecutar el proyecto.

Listado de dependencias

- Node v12.14.0 LTS, https://nodejs.org/
- .Net core 2.1 LTS https://dotnet.microsoft.com/download/dotnet-core
- Flutter v1.12.13 hotfix5-stable, https://flutter.dev/docs/get-started/install
- Xcode 11,
- Android Studio

10.1 Repositorio de código en GitHub.

https://github.com/Seryusjj/OnTimeApp desde aquí podemos descargar todo el código fuente necesario usando el comando:

git clone https://github.com/Seryusjj/OnTimeApp.git

Una vez tenemos el código Fuente vamos a ejecutar la aplicación

10.2 Herramientas de Desarrollo back-end.

El back end está escrito con .Net core, para el Desarrollo se pueden utilizar multiples IDES como **Visual studio 2019**, Visual Studio code o **Rider** de Jetbrains, este ultimo es el que se ha usao para el Desarrollo en windows pero puede usarse cualquier otro.

10.2 Ejecución del back-end.

Si únicamente queremos ejecutar el back end, podemos hacerlo desde línea de commandos. Tras situarnos en el directorio root del Proyecto **OnTimeApp/OnTimeApp.Api** ejecutamos el commando:

donet run -p OnTimeApp.API

```
C:\Users\Sergio\Desktop\OnTimeApp\OnTimeApp.API>"C:\Program Files\dotnet\dotnet.
exe" run -p OnTimeApp.API
Usando la configuración de inicio de OnTimeApp.API\Properties\launchSettings.jso
n...
info: Microsoft.AspNetCore.DataProtection.KeyManagement.XmlKeyManager[0]
      User profile is available. Using 'C:\Users\Sergio\AppData\Local\ASP.NET\Da
taProtection-Keys' as key repository and Windows DPAPI to encrypt keys at rest.
Hosting environment: Development
Content root path: C:\Users\Sergio\Desktop\OnTimeApp\OnTimeApp.API\OnTimeApp.API

Now listening on: https://localhost:5001
Now listening on: http://localhost:5000
Application started. Press Ctrl+C to shut down.
```

Esto lanzara el back-end junto con el servidor web de desarrollo que ya incluye certificados ssl para el uso de https.

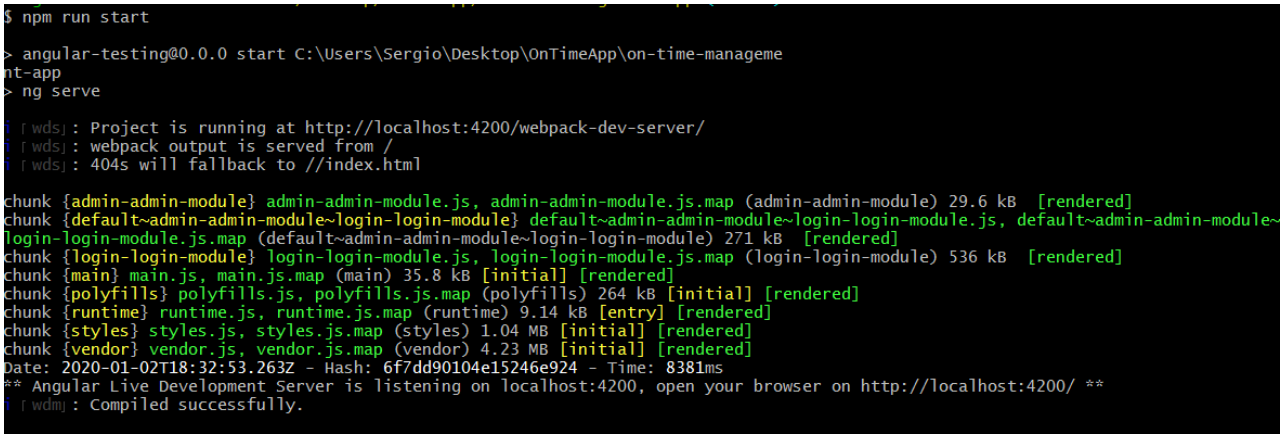
10.4 Herramientas de Desarrollo web.

Para desarrollar nuevas funcionalidades de la aplicacion web se pueden usar editores como Visual Studio Code o Webstorm para Angular.

10.5 Ejecutar web de administración.

En caso de querer ejecutar la web, tras ejecutar el back-end basta con ejecutar desde la raiz del Proyecto **OnTimeApp/on-time-management-app** los siguientes comandos:

npm install
npm run start



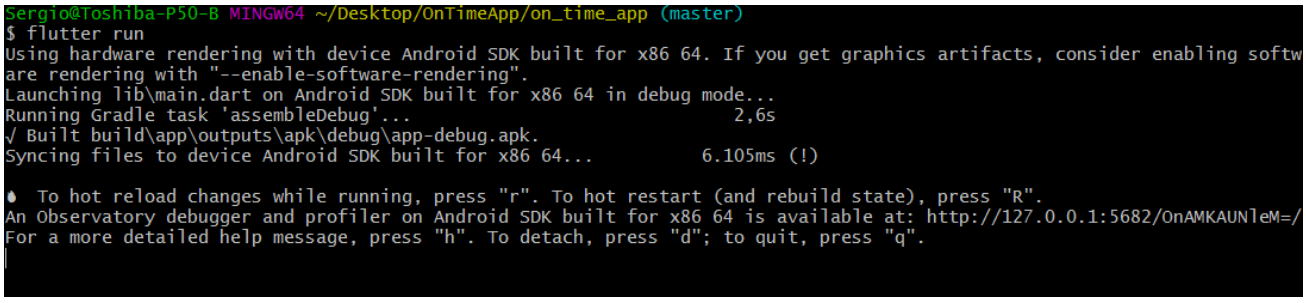
10.6 Herramientas de Desarrollo Flutter

Basta con Android Studio o Visual Studio Code.

10.7 Ejecutar Flutter app en Android.

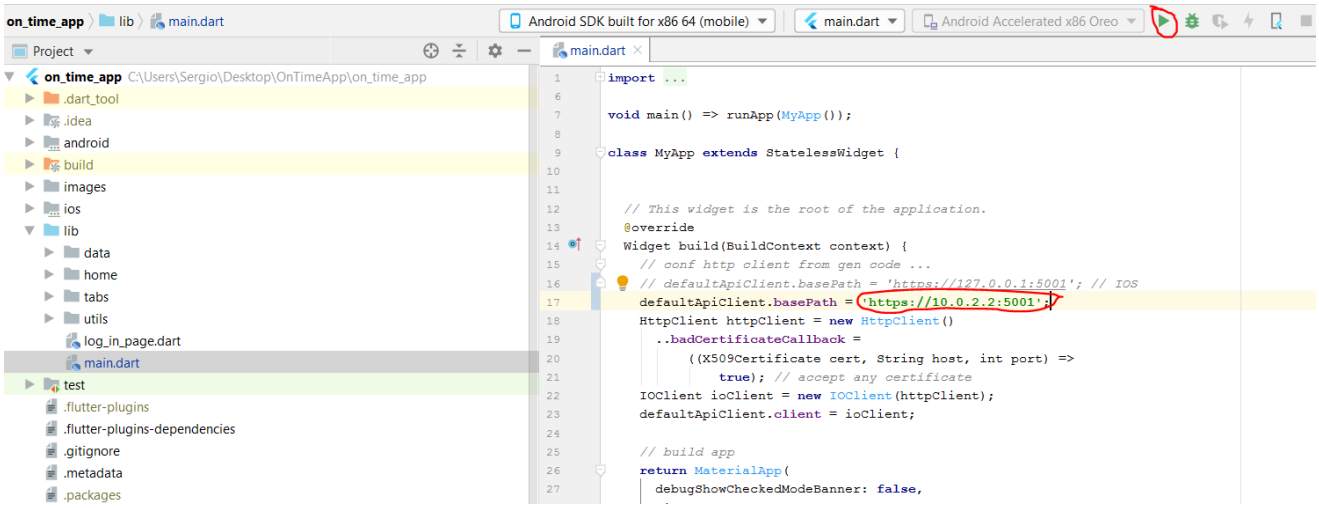
Ejecutar el emulador de Android y desde el directorio **OnTimeApp/on_time_app** ejecutar

flutter run



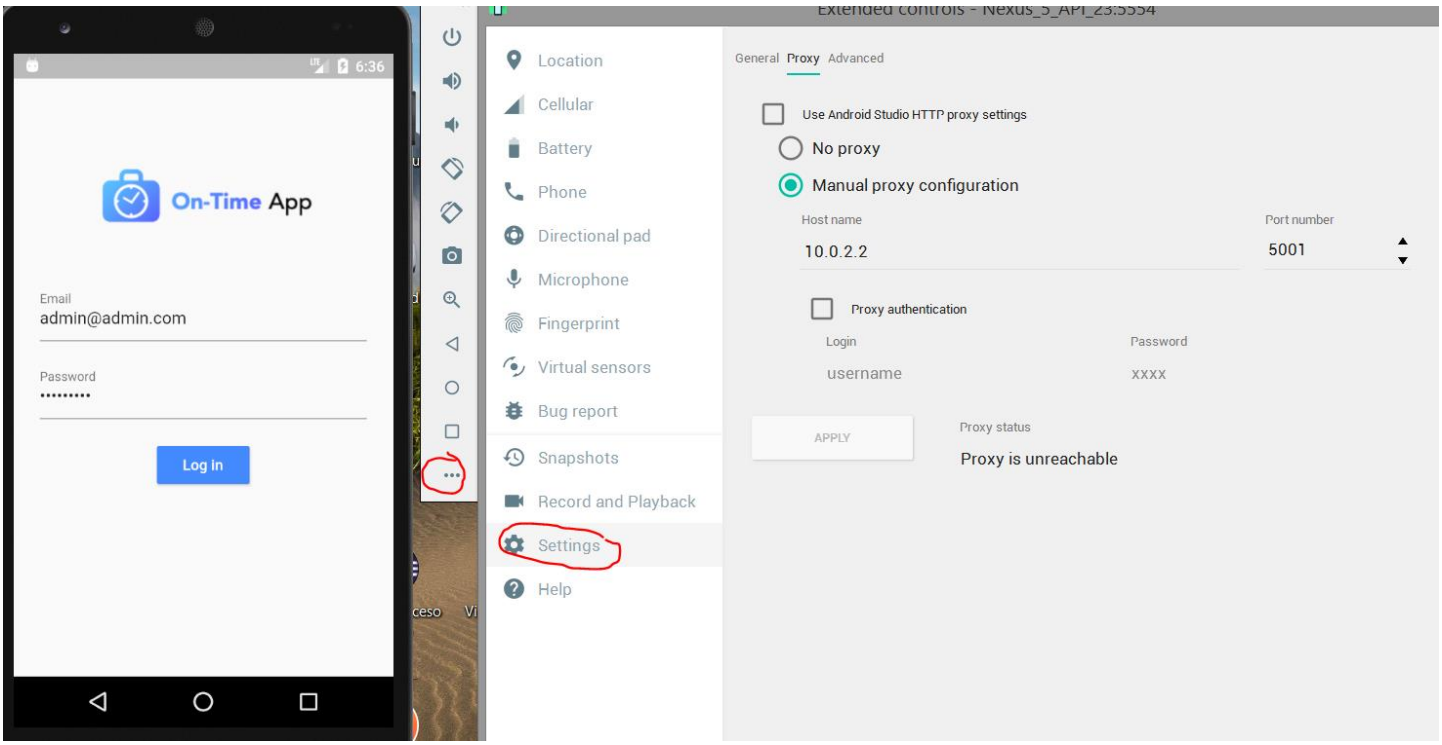
Este commando descargará las dependencias, compilará la app y la instalará en el dispositivo o emulador que encuentre.

También se puede hacer desde Android Studio si lo preferimos usando su interfaz de usuario



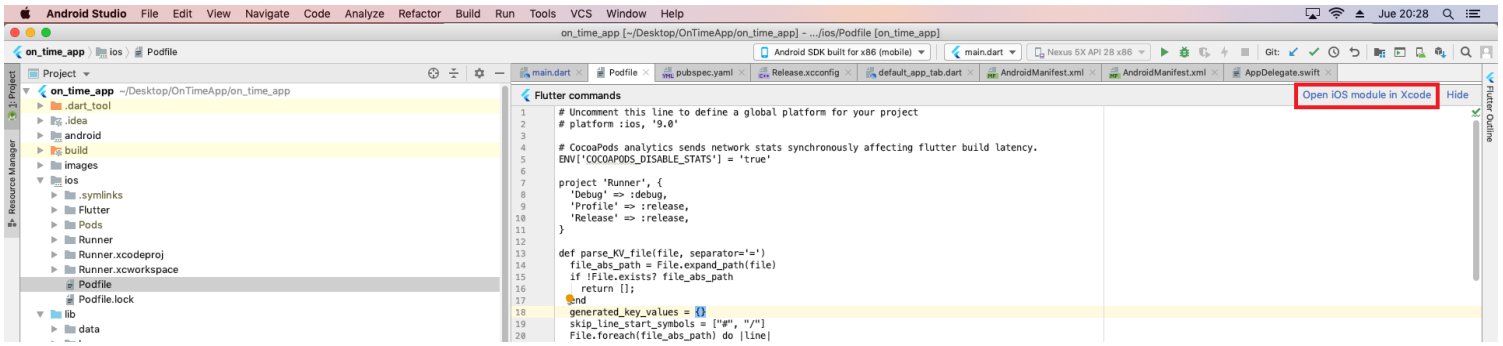
Nótese redondeado en rojo donde se ha escrito el root de la API Rest que coincide con la configuración de emulador, justo en la línea superior en código comentado está el root usado para IOS, se puede ver que es 127.0.0.1(IOS) en lugar de 10.0.2.2
(Android).

Tenganse en cuenta que el emulador debe estar debidamente configurado para acceder a back end que se estará ejecutando en la misma máquina. Para Android hemos de configurar proxy 10.0.2.2:5001 para que el tráfico sea redirigido.

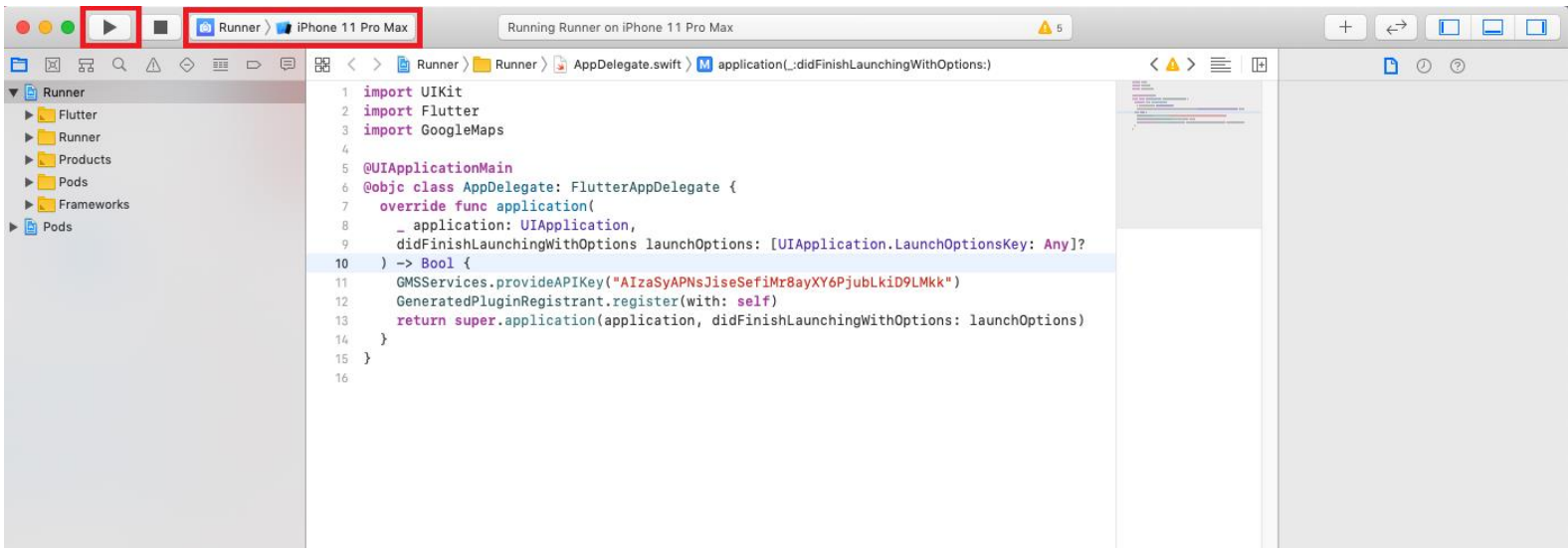


10.8 Ejecutar Flutter app en IOS.

Desde android Studio abrir IOS -> Podfile donde el IDE nos indicará '**Open IOS module en Xcode'** esto ejecutará el Podfile y descargara las dependencias necesarias, una vez en Xcode trabajar normalmente ejecutando el target en el emulador.



Una vez abierto Xcode ejecutar en el emulador



10.9 Swagger y codegen.

En el repositorio encontramos una carpeta llamada generated donde podremos ver los scripts con los que se ejecutó **swagger-codegen-cli.jar** a la hora de generar los clients de flutter y typescript para el back-end junto con la definicion de la API generda por swagger Open Api V2

Estos scripts no es necesario volver a ejecutarlos a no ser que se añada nueva funcionalidad a la API Rest