



# DISEÑO E IMPLEMENTACIÓN DE UNA APP SOBRE DESARROLLO SOSTENIBLE

CON BACK-END DE ARQUITECTURA BASADA EN MICROSERVICES  
Y DE UNA REACT NATIVE FRONT-END APP

## Memoria del proyecto

Universitat Politècnica de Catalunya (UPC) - BarcelonaTech  
Facultat d'Informàtica de Barcelona (FIB)  
Fecha de defensa: 28 de octubre de 2016  
Directora: Claudia Patricia Ayala Martínez - ESSI  
Titulación: Grado en Ingeniería Informática  
Especialidad: Ingeniería de Software

Autora: Laura Chacón Chacón  
[laura.chacon@est.fib.upc.edu](mailto:laura.chacon@est.fib.upc.edu)



UNIVERSITAT POLITÈCNICA DE CATALUNYA

BARCELONATECH

Facultat d'Informàtica de Barcelona



## RESUMEN

En este proyecto se ha desarrollado una aplicación móvil para iOS y Android que pretende concienciar al usuario sobre el cuidado del medio ambiente. El usuario podrá registrar las acciones cotidianas que lleva a cabo e impactan en el medio ambiente. De esta manera la aplicación podrá hacerle un seguimiento de sus actos y mostrarle si mejora o empeora su conducta medioambiental.

Con este marco medioambiental de fondo, se ha decidido aprender e investigar dos tecnologías actuales y cada vez más utilizadas para desarrollar cada parte del sistema. Para la parte del Back-End se ha decidido diseñar una arquitectura basada en microservicios. Se han implementado cinco servicios que se comunican entre sí y donde cada uno corresponde a un área de negocio de la aplicación. Estos cinco servicios se ejecutan en el cloud, concretamente en Amazon Web Services.

Para la parte del Front-End se ha utilizado React Native, un framework de Facebook que ha permitido construir las interfaces de usuario para iOS y Android. De esta manera se han creado dos aplicaciones nativas con una librería intermedia sin necesidad de implementarlas individualmente en base a sus APIs originales.



## ABSTRACT

In this project, a mobile application for iOS and Android that aims to raise awareness about the care of the environment has been developed. The application allows the user to record her environment related daily actions and get an overview of her environment footprint.

With this environmental background, it has been decided to learn and use two modern technologies that are gaining a lot of popularity for building software systems.

On one hand, the Back-End of the system is made out of five Python microservices that communicate with each other via HTTP and run on the Amazon Web Services cloud.

On the other hand, the Front-End is built on top of the React Native framework, which allows the engineer to build two native applications, one iOS app and one Android app, with a single codebase.



## RESUM

En aquest projecte s'ha dut a terme una aplicació mòbil per iOS i Android que pretén conscienciar a l'usuari sobre la cura del medi ambient. L'usuari podrà enregistrar les accions quotidianes que du a terme i impacten en el medi ambient. D'aquesta manera l'aplicació podrà fer-li un seguiment dels seus actes i mostrar-li si millora o empitjora la seva conducta mediambiental.

Amb aquest context mediambiental de fons, s'ha decidit aprendre i investigar dues tecnologies actuals i que cada cop són mes utilitzades per desenvolupar cada part del sistema. Per a la part del Back-End s'ha decidit dissenyar una arquitectura basada en microserveis. S'han implementat cinc serveis que es comuniquen entre ells i on cadascun correspon a una àrea de negoci de l'aplicació. Aquests cinc serveis s'executen al cloud, concretament a Amazon Web Services.

Per la part del Front-End s'ha utilitzat React Native, un framework de Facebook que ha permès construir les interfícies d'usuari per iOS i Android. D'aquesta manera s'han creat dues aplicacions natives amb una llibreria intermèdia sense necessitat de implementar individualment, en base a les seves APIs originals.



## AGRADECIMIENTOS

A mi directora Claudia Ayala y a mis profesores de la Universidad Politécnica de Cataluña que me han aportados los conocimientos y me han hecho de guía durante este largo y constructivo camino.

A mi padre, Luís, por enseñarme lo que es la constancia en cualquier aspecto de la vida y por el gran esfuerzo que ha realizado para que yo haya podido recibir esta educación. Por mostrarme lo que significa no rendirse nunca ante las peores circunstancias y por desear siempre lo mejor para mi vida.

A mi madre, Mª Carmen, por su comprensión, apoyo y cariño incondicional sin el que no podría haber realizado este camino. Por el amor recibido y la paciencia con la que trataba cada dificultad que se me presentaba durante este trayecto.

A todos los compañeros de universidad, tanto de la UPC como de la KTH, que se han acabado convirtiendo en amigos incondicionales que me han permitido disfrutar y hacer de la etapa universitaria la mejor de mi vida.

Al resto de mi gran familia y a mis amigas, sobre todo a Elena, Alba, Irene y Raquel, personas de las que siempre he recibido palabras de cariño y ánimo cada vez que el camino se complicaba.

Por último, a mi hermano Jordi, la persona sin la cual no habría podido llegar hasta aquí. Por sus enseñanzas tanto personales como profesionales, por ser mi guía y por nunca dejar de confiar en mí incluso cuando yo dejaba de hacerlo. Nunca tendré suficientes días de vida para agradecerle lo que ha hecho por mí.



## ÍNDICE

Resumen .....	2
Abstract.....	3
Resum .....	4
Agradecimientos.....	5
1. Contexto.....	9
1.1. Introducción .....	9
1.2. Actores implicados .....	10
1.2.1. Trámites legales.....	10
2. Estado del arte.....	11
2.1. Formulación del problema .....	11
2.2. Contextualización .....	12
2.2.1. React Native .....	12
2.2.2. Microservices.....	13
2.2.3. Desarrollo sostenible.....	13
2.3. Estudio de mercado.....	14
2.4. Conclusiones del estudio de mercado .....	14
3. Objetivos .....	15
4. Alcance.....	15
4.1. Posibles obstáculos.....	16
5. Metodología y rigor .....	16
5.1. Metodología de trabajo.....	16
5.2. Herramientas de desarrollo .....	17
5.3. Herramientas de seguimiento.....	17
6. Planificación inicial general .....	17
6.1. Planificación estimada del proyecto .....	17
6.2. Recursos.....	17
6.3. Plan de acción y valoración de alternativas.....	18
6.4. Consideraciones globales .....	19
6.5. Descripción de las tareas.....	19
6.6. Gestión del proyecto .....	19
6.6.1. Búsqueda y puesta en marcha.....	19
6.6.2. Desarrollo y testeo de la parte del Back-End.....	20



6.6.3. Desarrollo y testeo de la parte del Front-End .....	20
6.6.4. Mejoras.....	20
6.6.5. Documentación y presentación .....	20
6.7. Calendario.....	21
6.7.1. Estimación de horas fase inicial .....	21
6.7.2. Diagrama de Gantt .....	21
6.8. Gestión económica.....	23
6.8.1. Consideraciones iniciales .....	23
6.8.2. Identificación y estimación de costes .....	23
6.8.3. Costes directos por actividad fase inicial.....	24
6.8.4. Costes indirectos .....	25
6.8.5. Contingencia .....	26
6.8.6. Coste total .....	26
6.9. Control de gestión .....	27
7. Ejecución real.....	27
8. Sostenibilidad y compromiso social .....	33
8.1. Económica .....	33
8.2. Social.....	33
8.3. Ambiental .....	34
9. Requisitos y especificación.....	36
9.1. Requisitos funcionales.....	36
9.1.1. Diagrama general de casos de uso .....	36
9.1.2. Descripción de los casos de uso.....	37
9.2. Requisitos no funcionales.....	43
9.2.1. Requisitos de percepción .....	43
9.2.2. Requisitos de capacidad de uso y humanidad.....	44
9.2.3. Requisitos de desempeño .....	45
9.2.4. Requisitos operacionales y ambientales.....	47
9.2.5. Requisitos de mantenimiento y soporte .....	47
9.2.6. Requisitos de seguridad .....	47
9.2.7. Requisitos legales .....	48
10. Diseño .....	49
10.1. Diseño y arquitectura del back-end .....	50
10.1.1. Amazon Web Services .....	51



10.1.2. Microservices .....	52
10.1.3. Falcon .....	75
10.2. Diseño y arquitectura del front-end .....	75
10.2.1. Diseño de la interfaz de usuario .....	75
10.2.2. React native .....	85
10.2.3. Redux .....	86
11. Implementación.....	90
11.1. Lenguajes de programación .....	92
12. Fase de Pruebas.....	92
12.1. Fase de pruebas del Back-End.....	92
12.1.1. Fase de pruebas por servicio .....	92
12.1.2. Mocking .....	94
12.1.3. Fase de pruebas global.....	95
12.2. Fase de pruebas del front-end .....	98
13. Conclusiones finales .....	100
13.1. Limitaciones y dificultades .....	100
13.2. Trabajo futuro.....	100
13.3. Competencias técnicas.....	101
14. Bibliografía .....	102
15. Índice de tablas.....	104
16. Índice de ilustraciones.....	106
17. Índice de figuras .....	107



## 1. CONTEXTO

Este proyecto es un Trabajo Final de Grado de la especialidad de Ingeniería de Software de la Facultad de Informática de Barcelona (Universidad Politécnica de Cataluña). Se trata de un proyecto de modalidad A donde el autor contempló la posibilidad de crear una aplicación sobre desarrollo sostenible con la cual aprender React Native y arquitectura microservices.

### 1.1. INTRODUCCIÓN

Seguramente hayas oído hablar del desarrollo sostenible, ¿pero a que se refiere exactamente esta palabra? Hace referencia a aquel desarrollo que es capaz de satisfacer las necesidades actuales sin comprometer los recursos y posibilidades de las futuras generaciones. Este concepto de desarrollo sostenible abarca varias áreas. Para este proyecto se enfoca en el lado ambiental.

La preocupación por el medio ambiente se inicia a finales de la década de los 60 y ha crecido continuamente desde entonces, de modo que prácticamente se contempla en todas las actividades domésticas o industriales [1]. Aprovechando las nuevas tecnologías y en concreto el mundo de las aplicaciones móviles, se ha decidido enfocar este trabajo de final de grado en desarrollar una aplicación móvil en iOS y Android que fomente una vida más sostenible. Esto se pretende conseguir haciendo que el usuario introduzca datos sobre los hábitos que lleva a cabo y que pueden tener repercusión en el medio ambiente para que luego el sistema pueda ofrecerle un seguimiento de estos hábitos y mostrárselos gráficamente. De esta manera el usuario podrá ver el progreso de su huella ecológica, si aumenta o disminuye.

Con este contexto medioambiental de fondo, la intención del autor también es aprender sobre tecnologías actuales y modernas. En concreto a usar React Native para el Front-End<sup>1</sup> y un Back-End<sup>2</sup> con arquitectura basada en “microservices” en el Cloud. El aprendizaje y uso de estas herramientas tienen gran importancia en este proyecto.

---

<sup>1</sup> Parte del desarrollo que se dedica de la parte frontal, desde la estructura del sitio hasta los estilos como colores, fondos, tamaños hasta llegar a las animaciones y efectos [2]. La parte de sistema que se ejecuta en los dispositivos de los usuarios, ya sea un ordenador, Tablet o móvil.

<sup>2</sup> El área que se dedica a la parte lógica, es el encargado de que todo funcione como debería. Desde la programación de las funciones del sitio hasta bases de datos [2]. Es la parte del sistema que se ejecuta en un conjunto de servidores.



## 1.2. ACTORES IMPLICADOS

En este apartado se definirán los principales actores de este proyecto, es decir, todas las personas que tenga algún tipo de relación con el proyecto:

- Directora del proyecto: La directora del proyecto es Claudia Patricia Ayala Martínez. Será la persona que guiará y supervisará el trabajo del autor del proyecto.
- Equipo desarrollador del sistema: El equipo desarrollador del sistema lo forma el jefe de proyecto, un diseñador, un desarrollador y un responsable de pruebas. El éxito del proyecto depende del cumplimiento de las responsabilidades individuales y de la buena gestión y comunicación entre los miembros del equipo. Cabe recordar que en la práctica será el autor del proyecto quien ejercerá cada uno de los roles.

Por lo tanto, el autor tiene el objetivo de desarrollar el sistema con todas las garantías dentro del tiempo planificado. Fruto de esta actividad se espera obtener un beneficio didáctico.

- Administrador del sistema: El administrador tiene a su cargo gestionar la aplicación una vez el sistema esté acabado y se encarga de gestionar todo lo relacionado con los datos que el usuario registra en el sistema. El autor del proyecto será el administrador del sistema durante la versión de prueba inicial.
- Gestor: Este sistema ofrece una funcionalidad que muestra una serie de hechos relacionados con el medio ambiente al usuario. Esta información es añadida y gestionada por un gestor que se encarga de mantenerla actualizada en la base de datos del sistema. Inicialmente el gestor será el autor del proyecto y la plantilla de gestores podrá aumentar en caso que el número de usuarios crezca. De esta manera cumplimos con el requisito #13 (Ver Sección 1.1.1.23 Página 41).
- Usuarios: Este proyecto está enfocado en hacer tomar conciencia a sus usuarios de su huella ecológica. La persona que utilice esta aplicación es consciente de que tiene unos hábitos poco sostenibles y desea ver el progreso de sus hábitos. De esta manera puede ver si está llevando a cabo rutinas sostenibles o por el contrario debe cambiar para mejorar estas rutinas. Su objetivo es llevar a cabo prácticas que sean más responsables con el medio ambiente.
- Equipos de desarrollo de aplicaciones sobre desarrollo sostenible: Indirectamente la competencia actúa como parte interesada. Si tienen el sistema pueden probarlo y coger ideas para mejorar sus propios sistemas.

### 1.2.1. TRÁMITES LEGALES

En este proyecto necesitamos cumplir con la seguridad de los datos que son necesarios para el sistema. Por lo tanto se tiene que en cuenta la LOPD, Ley Orgánica de protección

de datos de carácter personal para poder protegerlos y garantizarlos, es decir asignar una protección alta a los datos de los usuarios [2].

## 2. ESTADO DEL ARTE

### 2.1. FORMULACIÓN DEL PROBLEMA

El haber escogido el tema sobre el medio ambiente es el resultado de una motivación propia sumado a la problemática actual que supone la degradación del medio ambiente. Según el barómetro del CIS de diciembre del 2015 [3], se puede observar que es un tema que interesa. A la pregunta sobre cuál era el nivel de interés que despertaba en ellos los temas sobre ecología y medio ambiente, un 28,9% de los encuestados respondió que mucho y un 48,3% contestó que bastante. En esta respuesta se puede ver reflejado que gran parte de la población intenta informarse e interesarse por todo lo relativo a lo medioambiental.

El barómetro del CIS también muestra algunos de los hábitos de los españoles. En la figura 1 puede observarse que más de un 70% de los encuestados afirman ser habituales en una serie de acciones consideradas buenas para el medio ambiente. La rutina donde más se puede observar que es necesario un cambio es en el uso de transporte público. Sólo un 25,6% de los encuestados dice usar habitualmente el transporte público para desplazarse en su localidad. Con la creación de esta App se espera concienciar al usuario y motivarlo a mejorar este hábito.

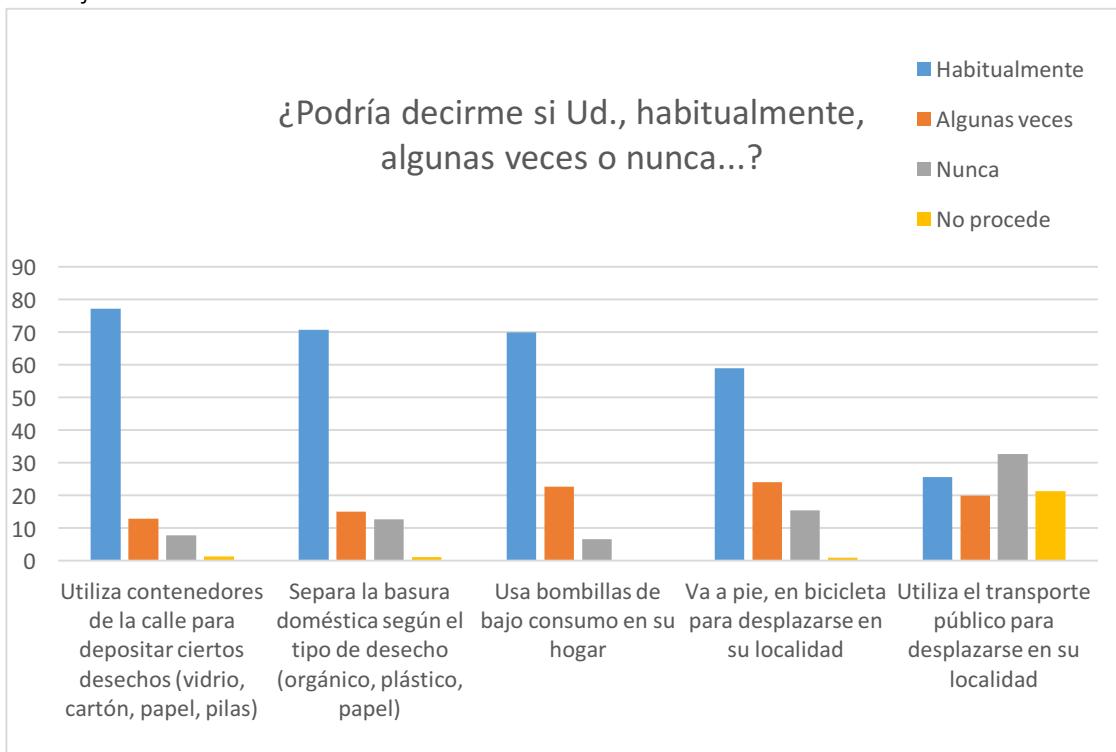


Figura 1. Gráfico con las respuestas a una pregunta del barómetro del CIS 2015.



Pese a esto, la problemática no se soluciona ni tiende a solucionarse. Muchos científicos, inversores, empresarios, geólogos y políticos piensan que se avecina una grave crisis energética mundial en los próximos 10 años. La dificultad para afrontar y reconocer el problema junto con la inercia de la vida que llevamos hará que no tomemos medidas a tiempo [4].

A partir de aquí surgió la idea de desarrollar una aplicación móvil para concienciar a las personas de que un cambio en nuestros hábitos es necesario mediante un seguimiento de sus hábitos medioambientales. ¿Y por qué no hacerlo mediante una aplicación móvil? Según el IIV Estudio Anual de Mobile Marketing publicado en septiembre de 2015, el número de Smartphones en España ha aumentado en un 35% desde 2012 alcanzando los 15 millones. Actualmente es el principal dispositivo con el que navegan el 85% de los usuarios, frente al 67% que prefiere el ordenador, o al 45% que se decanta por la Tablet [5].

Por todos los motivos expuestos anteriormente, sumado a mi posición como desarrolladora de software, decidí centrar mi proyecto en la implementación y desarrollo de una App móvil que motive a quien la use a reducir su impacto medioambiental.

## 2.2. CONTEXTUALIZACIÓN

### 2.2.1. REACT NATIVE

Para la parte del Front-End del sistema se ha decidido usar React Native como framework<sup>3</sup>. El motivo principal por el que se usa esta herramienta es que puedes crear una aplicación nativa<sup>4</sup> tanto para Android como para iOS, y reusar gran parte del código para ambas aplicaciones.

El framework oficial que ofrece Google para crear aplicaciones Android te obliga a usar Java, de manera que si creas una App de esta manera solo funcionará para dispositivos que tienen Android como sistema operativo. Si más adelante se decidiera hacer esa misma App para dispositivos iOS, se tendría que desarrollar el mismo código en Objective-C. Estaríamos hablando de dos lenguajes distintos, dos frameworks distintos con funciones distintas y documentación distinta. Esto implica que cada vez que se deba hacer un cambio en una de las versiones se deberá hacer el cambio en la otra versión. Es por estos motivos entre otros, que durante los últimos 5 años ha habido muchos intentos por parte de desarrolladores de crear un framework común que te permita crear una única App que funcione en ambos sistemas. React Native es uno de estos intentos de crear un framework común. Es decir, permite compilar aplicaciones escritas en JavaScript para Web, iOS y Android desde una única base de código, lo que significa que los desarrolladores pueden aprender un idioma único de programación y desplegar su software en las tres plataformas. Y esta es exactamente una de las intenciones del

<sup>3</sup> Conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar.

<sup>4</sup> Aplicación que se desarrolla de forma específica para un determinado sistema operativo.



proyecto, desarrollar su App en Android y iOS aprovechando la mayor parte del código en el proceso de creación de ambas versiones.

### 2.2.2. MICROSERVICES

Para la parte del Back-End se utilizará una arquitectura basada en microservices en lugar de desarrollar una arquitectura monolítica. Un sistema monolítico es el que se organiza en su totalidad de forma vertical, la información tiene que atravesar desde la base de datos, pasando por una o varias partes de todo el sistema, hasta llegar al usuario. Esto genera varias dificultades a la hora de programar, como una gran dependencia entre las partes que lo forman o que si aparece un error crítico toda la aplicación deja de funcionar. Por las dificultades que presenta la implementación de una arquitectura monolítica sumado a una motivación personal por aprender nuevas maneras de desarrollar arquitecturas software, se decidió implementar una arquitectura con microservices. Esta arquitectura se caracteriza por permitir consultar información a un servicio específico. Contrariamente a la arquitectura monolítica, ninguna parte depende necesariamente de otras partes, y cada parte, de forma independiente, sabe responder ante cualquier situación. Además cada uno de los servicios suele encapsular funcionalidades más simples y se pueden escalar o reducir, probar, implementar y administrar de forma independiente [6].

Es decir que mientras que una aplicación monolítica es un programa grande con muchas responsabilidades, las aplicaciones basadas en microservices se componen de varios programas pequeños, cada uno con una única responsabilidad [7].

### 2.2.3. DESARROLLO SOSTENIBLE

Una vez se decidió que herramientas se usarían para desarrollar el proyecto, se buscó un tema motivador y en alza actualmente que permitiera usar y aprender de estas nuevas tecnologías. El hecho de que el autor haya recibido un curso sobre la relación del desarrollo sostenible con las nuevas tecnologías y otro sobre los aspectos medioambientales de la informática, hizo que se planteara el medioambiente como tema del proyecto. Otro hecho que hizo decantarse por este tema fue la importancia que está tomando la sostenibilidad en nuestra sociedad. Estudios como el realizado por la empresa Kantar TNS a petición de la Comisión Europea reflejan que ocho de cada diez ciudadanos europeos consideran importante o muy importante la conservación de la naturaleza. También muestra que tres de cada cuatro ciudadanos europeos manifiestan su preocupación por la pérdida de biodiversidad. El cambio climático es, según el Eurobarómetro, el cuarto mayor problema global después de la pobreza, el terrorismo y la situación económica [8].

## 2.3. ESTUDIO DE MERCADO

---

Para situar el proyecto dentro de un marco de soluciones existentes, es necesario realizar un estudio de mercado teniendo en cuenta las necesidades de los usuarios. Estas necesidades son principalmente conocer sus hábitos medioambientales y cómo mejorarlos. Ha sido de gran ayuda realizar este estudio sobre aplicaciones móviles ya existentes del que se pudieron sacar conclusiones útiles de cara a definir los objetivos y funcionalidades del proyecto.

Algunas de las aplicaciones móviles más comunes que ayudan a los usuarios a convertirse en un ciudadano más sostenible son las que se presentarán a continuación. Tienen el objetivo común de que promueven el consumo consciente y facilitan información valiosa a las personas sobre lo que hacen y consumen [9]:

- Goodguide: Se caracteriza por mostrar los productos de supermercado junto con unos ratings con el impacto social, medioambiental y la salud que se desprende de cada uno. Muestra si estos productos se acomodan a un estilo de vida basado en el consumo responsable [10].
- JouleBug: Esta aplicación fue diseñada para fomentar una vida sostenible a través de juegos y la interacción social haciendo del uso de esta App simple y divertido. Esta App puede ser muy útil para grupos. Por ejemplo, si se adopta a nivel corporativo, los empleados pueden ver la participación y puntuación de otros de manera que compiten entre sí y de esta manera entran en una competición sobre reducir su huella ecológica [11].
- Locavore: Esta App te ayuda a descubrir que alimentos se producen en la estación del año actual. Te ofrece la localización de los mercados agrícolas y tiendas donde puedes encontrar los alimentos y cómo cocinarlos [12].

## 2.4. CONCLUSIONES DEL ESTUDIO DE MERCADO

---

Después de realizar un estudio de aplicaciones sobre el medio ambiente se ha podido llegar a varias conclusiones. La primera es que no existe ninguna App de uso generalizado entre los usuarios de Smartphones sobre cómo mejorar el medio ambiente. Se ha podido observar que hay un gran número de aplicaciones que en conjunto abarcan varios ámbitos relacionados con el medio ambiente.

Otra conclusión a la que se ha llegado es que para este sistema se pretende fomentar la vida sostenible mediante el seguimiento de diversas rutinas de los usuarios. No se pretende centrarse solo en un tipo de acción medioambiental, sino dar una idea del impacto general que tienen los hábitos del usuario. Utilizar los actos que las personas realizan diariamente y que tienen repercusión en el medioambiente.



### 3. OBJETIVOS

El objetivo principal del proyecto se puede dividir en tres. Un objetivo general y dos personales con el fin de crecer como ingeniera de software adquiriendo nuevos conocimientos:

- Crear una solución software relacionada con el medio ambiente que motive al usuario a reducir su impacto medio ambiental.
- Aprender React Native desarrollando una App en Android y iOS con dicho framework.
- Aprender y crear una arquitectura orientada a servicios o de microservices y evaluar las ventajas y desventajas de ésta en comparación con una arquitectura clásica monolítica.

### 4. ALCANCE

El sistema resultante de este proyecto será una aplicación móvil que se caracterizará por haber usado unas tecnologías actuales para cada una de las dos partes en las que se dividirá el desarrollo del sistema, Back-End y Front-End. Una vez finalizado el sistema se espera haber aprendido el framework React Native para la parte del Front-End del sistema. Respecto a la parte del Back-End, se espera interiorizar cómo desarrollar una arquitectura microservices, de manera que una vez finalizado el proyecto se pueda ser capaz de comparar y evaluar con conocimiento los dos tipos de arquitecturas empleadas, la monolítica y la orientada a servicios.

Una vez se haya diseñado, implementado y testeado el proyecto, se espera que éste pueda ofrecer un serie de funcionalidades. Permitirá al usuario añadir información sobre las acciones cotidianas relacionadas sobre el medio ambiente que lleva a cabo durante el día. Estas acciones con las que se llevará a cabo un seguimiento son las relacionadas con el consumo de:

- Agua
- Temperatura (calefacción y aire acondicionado)
- Comida
- Transporte

Con esta información, el sistema le mostrará al usuario el progreso de estos hábitos para que él sea consciente de los cambios que se producen en sus actos. Además la aplicación dispondrá de una sección con hechos relacionados con el medioambiente que el usuario podrá consultar.



## 4.1. POSIBLES OBSTÁCULOS

Los condicionantes más destacables a los que se debe hacer frente durante el desarrollo del sistema son:

- Restricción temporal: El Trabajo Fin de Grado tiene una fecha límite fijada. Cualquier tipo de problema, inconveniente, bug o estancamiento durante la realización del mismo se deberá gestionar adecuadamente para no retrasarlo y en su caso se volverá a planificar el trabajo para que no tenga repercusiones mayores.
- Problemas con el aprendizaje de las tecnologías: El desarrollo del sistema irá ligado con el grado de aprendizaje autónomo que se vaya adquiriendo del framework React Native y de microservices.

## 5. METODOLOGÍA Y RIGOR

### 5.1. METODOLOGÍA DE TRABAJO

Para el desarrollo del proyecto se pensó inicialmente en una metodología en cascada. Esta metodología se puede definir como una serie de actividades a ser seguidas en orden. La estrategia principal era definir y seguir el progreso del desarrollo del software hacia una serie de fines bien definidos, es decir, se codifica y reparan los errores. Se trata de un proceso constante de codificación y reparación. Este modelo contemplaba las siguientes fases: Análisis y especificación de requisitos, diseño, codificación, integración y pruebas, liberación y mantenimiento.

Se harán reuniones semanales (o quincenales si no fuese posible) con la directora del proyecto para comprobar que se sigue la planificación temporal y que el desarrollo del software se ajusta a los requisitos. También será en estas reuniones donde la directora dará el visto bueno definitivo del trabajo realizado.

Finalmente y viendo que esta metodología no era la que más se ajustaba a las necesidades del sistema, se decidió usar una metodología ágil basada en un desarrollo iterativo que acepta mejor los cambios que puedan aparecer a lo largo del proyecto.

La metodología que se ha seguido es SCRUM. Se han realizado varias iteraciones llamadas *sprints*.

En el desarrollo e implementación del back-end, el tiempo de cada sprint ha sido de 2-3 semanas. El tiempo ha podido variar dependiendo de las tareas que se han llevado a cabo. Cada sprint pretendía desarrollar uno de los cinco servicios que finalmente componen el sistema.



## 5.2. HERRAMIENTAS DE DESARROLLO

Las herramientas de desarrollo principales que serán usadas serán el framework React Native basado en el lenguaje de programación Javascript. Para la parte del Back-End se utilizará Python como lenguaje de programación y el framework Falcon. Para llevar a cabo la arquitectura basada en microservices se hará uso de Amazon Web Services. El editor de texto que se utilizará es Emacs. Finalmente, como base de datos se utilizará DynamoDB y S3 que son dos servicios que ofrece Amazon Web Services. Otro servicio ofrecido por Amazon Web Services que se utilizará es EC2. Se entrará en detalle en estas tecnologías en el apartado 10.1. Diseño y arquitectura del back-end que aparece más adelante.

## 5.3. HERRAMIENTAS DE SEGUIMIENTO

Como no se trata de un proyecto a desarrollar por un equipo de varias personas, sino por una sola, no se ha tenido en cuenta herramientas de trabajo simultáneo. Se trabajará, no obstante, con herramientas de control de versiones, conjuntamente con gestores de repositorios en red (Git y Github), con tal de garantizar la disponibilidad del código y facilitar la recuperación de fallos.

# 6. PLANIFICACIÓN INICIAL GENERAL

En este apartado aparece explicado todo lo que se planificó inicialmente para este proyecto. En el apartado 7 de este documento se explica y argumenta todas las diferencias que se han producido entre la planificación real y la planificación real que finalmente se ha llevado a cabo.

## 6.1. PLANIFICACIÓN ESTIMADA DEL PROYECTO

El proyecto tiene una duración estimada de cuatro meses, con inicio el 26 de enero y finalización el 27 de junio, teniendo un margen de unos 20 días antes de la presentación para posibles desviaciones que puedan aparecer. Tiene una carga de trabajo de 540 horas aproximadamente, englobando todas las tareas necesarias para la realización del proyecto.

## 6.2. RECURSOS

Los recursos previstos para la realización de este proyecto son:

- **Recursos personales:** una persona, con una dedicación de 30 horas semanales durante todo el periodo de desarrollo del proyecto.

- Recursos materiales:

Recurso	Tipo	Finalidad
Ordenador portátil DELL Latitude E7240 Memoria DDR3L de 4 GB con distribución Ubuntu 14.02.2 LTS	Herramienta de desarrollo	Para desarrollar la aplicación móvil y la memoria.
MackBook Air	Herramienta de desarrollo	Para desarrollar la aplicación móvil.
Iphone 5 con iOS 9.2.1	Herramienta de desarrollo	Para testear la aplicación móvil iOS desarrollada.
Motorola Moto G (3ª Generación)	Herramienta de desarrollo	Para testear la aplicación móvil Android desarrollada.
GNU Emacs	Herramienta de desarrollo	Editor de texto para desarrollar la aplicación.
Microsoft Office Word 2013 Microsoft Office Powerpoint 2013	Herramienta de desarrollo	Para realizar la documentación del proyecto
Git v1.9.5	Herramienta de desarrollo	Para el control de versiones del repositorio del código fuente.
Correo electrónico Gmail	Herramienta de comunicación	Para comunicarme con la directora.
Amazon Web Service	Herramienta de desarrollo	Para poder utilizar la aplicación y para que el móvil pueda consumir de la API.
Adobe Reader XI	Herramienta de desarrollo	Para visualizar documentos.
Microsoft Project 2013	Herramienta de gestión	Para planificar el proyecto.

Tabla 1 Recursos materiales

### 6.3. PLAN DE ACCIÓN Y VALORACIÓN DE ALTERNATIVAS

Se pueden producir desviaciones temporales en las iteraciones del desarrollo y en el tiempo dedicado a aprender el funcionamiento de las nuevas herramientas. En caso que se diera una desviación de más de 5 días en la realización de una tarea se llevará a cabo una nueva planificación.

Puede darse el caso de que se tenga que cancelar la implementación de una funcionalidad que se esté desarrollando o modificarlo de manera que se pueda llevar a cabo en las fechas acordadas. La funcionalidad que no se llevaría a cabo sería la que ofrece al usuario hechos y consejos relacionados con el medio ambiente. Que finalmente no se lleve a cabo esta funcionalidad no afecta a la funcionalidad principal del sistema que es la de ofrecer un seguimiento al usuario de sus hábitos medioambientales. Son independientes la una de la otra por lo que no llevar a cabo una no afectará a la otra.

## 6.4. CONSIDERACIONES GLOBALES

Este proyecto será llevado a cabo por una sola persona de manera que ninguna de las tareas podrá hacerse en paralelo. Debido a la metodología escogida y a la naturaleza del proyecto, este proyecto debe elaborarse de manera secuencial. No se empezará una tarea hasta que la anterior esté finalizada.

## 6.5. DESCRIPCIÓN DE LAS TAREAS

En esta sección se explicarán las diferentes fases en que se dividirá el proyecto. También se dará una idea general de lo que se llevará a cabo en cada una de las fases. En la figura 1 podemos ver representado el nombre de cada fase con las horas requeridas para cada una. Se puede observar que es la parte de la implementación del Back-End y Front-End la que requiere de más horas.

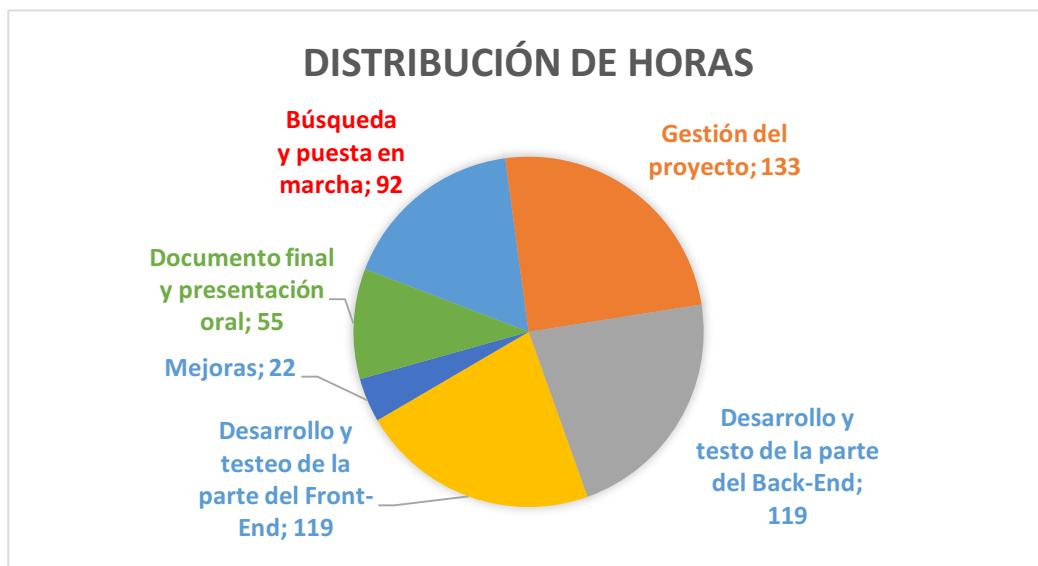


Figura 1 Gráfica con el reparto de horas por fase

## 6.6. GESTIÓN DEL PROYECTO

En la primera fase, el jefe de proyecto se dedicará a la elaboración del documento de gestión del proyecto que se irá mejorando durante la asignatura de GEP. Para llevar a cabo estos documentos se siguen unas pautas e indicaciones fijas de la asignatura. Este proceso no tiene ninguna dependencia de precedencia ya que tiene unas fechas de entregas fijas y hay que seguir el calendario estrictamente.

### 6.6.1. BÚSQUEDA Y PUESTA EN MARCHA

En esta fase del proyecto se preparará el entorno de trabajo. Durante esta fase se utilizarán 80 horas de autoaprendizaje del framework React Native. Durante este tiempo también se aprenderá sobre la arquitectura microservices con la que se implementará la parte del Back-End. Esta fase no tiene ninguna dependencia de precedencia.



### 6.6.2. DESARROLLO Y TESTEO DE LA PARTE DEL BACK-END

Esta fase se utilizará para implementar la parte del Back-End de nuestro sistema. Se desarrollarán todos los servicios necesarios para el correcto comportamiento del sistema y se testearán. Durante esta fase se llevará a cabo el análisis de requisitos, diseño y especificación y finalmente testeo. Estos pasos que se siguen en esta fase se harán específicamente para cada servicio que se implemente. De esta manera hasta que un servicio no pase todas las pruebas requeridas no se pasará a implementar el siguiente. Al ser una arquitectura microservices ningún servicio depende de otro. Esto facilita el trabajo de implementación y testeo ya que no se crean dependencias entre servicios. También será durante esta fase donde se configurará lo relativo a las bases de datos, es decir Amazon S3 y DynamoDB. También se testeará, de manera que se confirme su correcto funcionamiento.

### 6.6.3. DESARROLLO Y TESTEO DE LA PARTE DEL FRONT-END

Durante esta fase se llevarán a cabo los mismos pasos que para el desarrollo y testeo de la parte del Back-End, es decir, el análisis de requisitos, especificación y diseño, implementación y testeo. Durante esa fase, y como parte del diseño, se crearán los mockups de la aplicación con una herramienta online y dejaremos definido el aspecto que tendrá la aplicación móvil tanto para Android como para iOS. A medida que se vaya desarrollando se irá probando de manera que cada función implementada funcione como se espera. Durante esta fase también se juntarán las partes del Back-End y Front-End y se realizarán varias pruebas que ratifiquen el correcto funcionamiento de la aplicación.

### 6.6.4. MEJORAS

Esta etapa es opcional. Esta es última fase que forma parte del proceso de desarrollo del sistema y consistirá en buscar mejoras tanto en el funcionamiento, en el diseño o en la implementación de nuevas funcionalidades. La planificación de esta fase del proyecto se hará en el momento en que se sepa el tiempo disponible para dedicar a estas nuevas mejoras. La dependencia de precedencia de esta fase son las dos anteriores.

### 6.6.5. DOCUMENTACIÓN Y PRESENTACIÓN

Es durante esta fase que se terminará la documentación requerida. También se revisará toda la hecha anteriormente con el fin de dejar constancia de los cambios que se han dado durante las fases posteriores a la creación de la documentación por primera vez.

Durante esta fase también se preparará la defensa oral del proyecto. La fecha de esta presentación no está todavía definida.

## 6.7. CALENDARIO

En esta sección aparece la estimación de horas por cada rol que se calculó al inicio del proyecto, y el diagrama de Gantt que muestra las fechas y períodos de tiempo que cada etapa del proyecto requerían durante la fase inicial.

### 6.7.1. ESTIMACIÓN DE HORAS FASE INICIAL

Tarea	Responsable	Horas
Búsqueda y puesta en marcha		92
Búsqueda y puesta en marcha	Jefe de proyecto	24
	Programador	68
Gestión del proyecto		133
Definición del alcance y contextualización	Jefe de proyecto	34
	Jefe de proyecto	31
	Jefe de proyecto	21
	Jefe de proyecto	8
	Jefe de proyecto	8
	Jefe de proyecto	32
		119
Análisis de requisitos	Analista	12
Diseño y Especificación	Diseñador	16
	60% Programador / 40% tester	91
Desarrollo y testeo de la parte del Back-End		119
Análisis de requisitos	Analista	12
	Diseñador	16
	60% Programador / 40% tester	91
Desarrollo y testeo de la parte del Front-End		119
Diseño y especificación	Analista	12
	Diseñador	16
	60% Programador / 40% tester	91
Mejoras		22
Análisis y diseño	50% Analista / 50% Diseñador	7
	60% Programador / 40% tester	15
Implementación y pruebas		
Documento final y presentación oral		55
Redacción de la memoria	Jefe de proyecto	41
	Jefe de proyecto	14
<b>Total</b>		<b>540</b>

Tabla 2 Estimación de horas fase inicial

### 6.7.2. DIAGRAMA DE GANTT

El diagrama de Gantt de la figura 2 que aparece a continuación contiene todas las fases que se han explicado en los apartados anteriores que corresponden con la planificación que se hizo durante la fase inicial del proyecto. En la tabla 3 observamos por cada fase su respectiva duración, fechas de inicio y finalización y responsable de la tarea.

Resource Names	Duration	Start	Finish
<b>1. Búsqueda y puesta en marcha</b>	<b>20 days</b>	<b>Tue 26/01/16</b>	<b>Mon 22/02/16</b>
Búsqueda y puesta en marcha	5 days	Tue 26/01/16	Mon 01/02/16
Aprendizaje autónomo	15 days	Tue 02/02/16	Sun 21/02/16
<b>2. Gestión del proyecto</b>	<b>29 days</b>	<b>Mon 22/02/16</b>	<b>Thu 31/03/16</b>
Definición del alcance y contextualización	8 days	Mon 22/02/16	Wed 02/03/16
Planificación temporal	7 days	Thu 03/03/16	Fri 11/03/16
Gestión económica y sostenible	5 days	Sat 12/03/16	Thu 17/03/16
Presentación preliminar	2 days	Fri 18/03/16	Sun 20/03/16
Pliego de condiciones	2 days	Mon 21/03/16	Tue 22/03/16
Documento final y presentación oral	7 days	Wed 23/03/16	Thu 31/03/16
<b>3. Desarrollo y testeo de la parte del back-end</b>	<b>26 days</b>	<b>Fri 01/04/16</b>	<b>Fri 06/05/16</b>
Análisis de requisitos	3 days	Fri 01/04/16	Tue 05/04/16
Diseño y Especificación	4 days	Wed 06/04/16	Sun 10/04/16
Implementación y pruebas	20 days	Mon 11/04/16	Fri 06/05/16
<b>4. Desarrollo y testeo de la parte del front-end</b>	<b>26 days</b>	<b>Sat 07/05/16</b>	<b>Fri 10/06/16</b>
Análisis de requisitos	3 days	Sat 07/05/16	Tue 10/05/16
Diseño y Especificación	4 days	Wed 11/05/16	Mon 16/05/16
Implementación y pruebas	19 days	Tue 17/05/16	Fri 10/06/16
<b>5. Mejoras</b>	<b>5 days</b>	<b>Sat 11/06/16</b>	<b>Thu 16/06/16</b>
Análisis y diseño	2 days	Sat 11/06/16	Sun 12/06/16
Implementación y pruebas	3 days	Mon 13/06/16	Wed 15/06/16
<b>6. Documento final y presentación oral</b>	<b>11 days</b>	<b>Fri 17/06/16</b>	<b>Fri 01/07/16</b>
Redacción de la memoria	8 days	Fri 17/06/16	Tue 28/06/16
Preparación presentación oral	3 days	Wed 29/06/16	Fri 01/07/16

Tabla 3 Diagrama de Gantt fase inicial

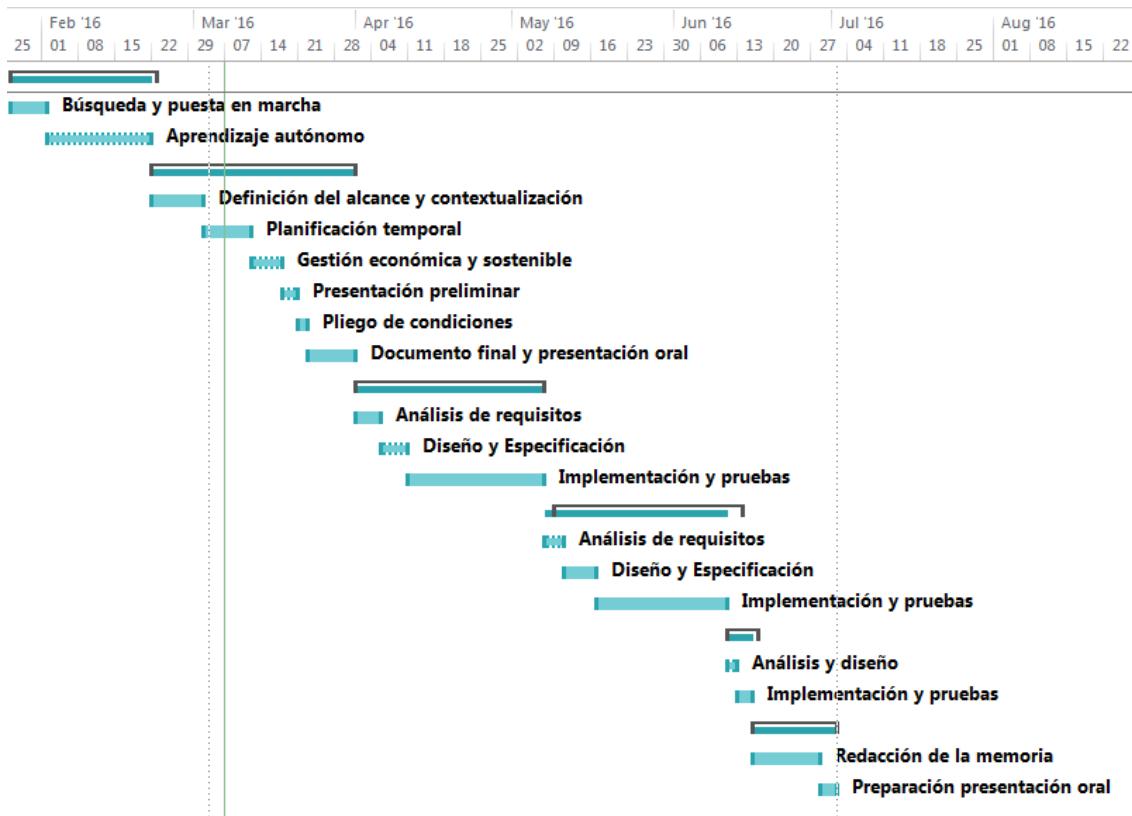


Figura 2 Diagrama de Gantt fase inicial

## 6.8. GESTIÓN ECONÓMICA

Una vez se ha definido el problema y se ha propuesto la solución, se requiere hacer un estudio económico y otro del impacto social y ambiental con el fin de conocer si el proyecto es viable o no.

En las próximas secciones se lleva a cabo los costes de los recursos necesarios para el desarrollo del proyecto. En concreto los recursos humanos, las partes hardware, las partes software y los gastos generales.

### 6.8.1. CONSIDERACIONES INICIALES

Se ha de tener en cuenta que este se trata de un proyecto universitario. Es por eso que el cálculo de los recursos humanos se ha hecho para dos casos. El caso 1 que se hace partiendo de que la autora hace todos los roles y los lleva a cabo como becaria y lleva a cabo todas las tareas. El caso 2 será teniendo en cuenta el salario específico de cada rol sin ser becario.

### 6.8.2. IDENTIFICACIÓN Y ESTIMACIÓN DE COSTES

La identificación y estimación de costes del proyecto es el paso previo a la elaboración del presupuesto. El primer recurso que se tiene en cuenta para el cálculo de costes es el humano.

La remuneración utilizada para esta estimación corresponde con el precio mínimo por hora que se establece en el informe Page Personnel del año 2016 [13]. Suponemos que el año tiene 250 días laborables.

Rol	Horas estimadas	Remuneración (€/h) (Caso 1)	Remuneración (€/h) (Caso 2)	Coste estimado € (Caso 1)	Coste estimado € (Caso 2)
Jefe de proyecto	212	8	20	1.696	4.240
Analista	27,5	8	16	1.489,6	1.955,1
Diseñador	35,5	8	10,5	220	440
Programador	186,2	8	10,5	630,4	866,8
Tester	78,8	8	11	284	372,75
<b>Total</b>	<b>540</b>			<b>4320 €</b>	<b>7874,65 €</b>

Tabla 4 Coste estimado de recursos humanos por casos

Hace falta recordar que las horas estimadas para el programador incluyen las de autoaprendizaje del framework React Native y de cómo llevar a cabo una arquitectura microservices.

### 6.8.3. COSTES DIRECTOS POR ACTIVIDAD FASE INICIAL

En la siguiente tabla podemos observar el desglose por cada actividad que forman las fases del proyecto consideradas durante la fase inicial:

Nombre de la actividad	Horas estimadas	Recursos	Coste € (Caso 1)	Coste € (Caso 2)
Búsqueda y puesta en marcha	92		736	1194
Búsqueda y puesta en marcha	24	Jefe de proyecto	192	480
Aprendizaje autónomo	68	Programador	544	714
Gestión del proyecto	133		952	2680
Definición del alcance y contextualización	34	Jefe de proyecto	272	680
Planificación temporal	31	Jefe de proyecto	248	620
Gestión económica y sostenible	21	Jefe de proyecto	168	420
Presentación preliminar	8	Jefe de proyecto	64	160
Pliego de condiciones	8	Jefe de proyecto	64	160
Documento final y presentación oral	32	Jefe de proyecto	256	640
Desarrollo y testo de la parte del Back-End	119		952	1333,7
Análisis de requisitos	12	Analista	96	192
Diseño y Especificación	16	Diseñador	128	168
Implementación y pruebas	91	60% Programador / 40% tester	728	973,7
Desarrollo y testeo de la parte del Front-End	119		952	1333,7
Análisis de requisitos	12	Analista	96	192
Diseño y especificación	16	Diseñador	128	168
Implementación y pruebas	91	60% Programador / 40% tester	728	973,7
Mejoras	22		176	253,25
Análisis y diseño	7	50% Analista / 50% Diseñador	56	92,75
Implementación y pruebas	15	60% Programador / 40% tester	120	160,5
Documento final y presentación oral	55		440	1100
Redacción de la memoria	41	Jefe de proyecto	328	820
Preparación presentación oral	14	Jefe de proyecto	112	280
<b>Total</b>	<b>540</b>		<b>3592 €</b>	<b>6700,65 €</b>

Tabla 5 Costes directos por actividad por casos de fase inicial

#### 6.8.4. COSTES INDIRECTOS

- **Hardware:**

Los recursos hardware que se han usado se estima que tendrán una vida útil de 4 años. Para establecer un coste de amortización por hora de uso se coge como referencia que 1 año de vida útil son 250 días laborables a un ritmo de trabajo de 8 horas diarias.

Para realizar el cálculo de la amortización se hará de la siguiente manera:

$$\frac{\text{Precio}}{4 \text{ años} * (250 \text{ días laborales} * 8 \text{ horas/día})}$$

Recurso hardware	Precio (€)	Horas estimadas	Coste amortización (€/h)	Coste estimado (€)
Ordenador portátil DELL Latitude E7240 Memoria DDR3L	899	329	0,11	101
Iphone 5 16GB	279	98	0,03	10
Motorola Moto G	199	98	0,02	5
Amazon Web Server	200	261	0,02	5
<b>Total</b>				<b>120 €</b>

Tabla 6 Costes estimado por hardware

- **Software:**

Los recursos software deberán ser renovados cada 3 años. Para establecer un coste de amortización por hora de uso se coge como referencia que 1 año de vida útil son 250 días laborables a un ritmo de trabajo de 8 horas diarias.

Para realizar el cálculo de la amortización se hará de la siguiente manera:

$$\frac{\text{Precio}}{3 \text{ años} * (250 \text{ días laborales} * 8 \text{ horas/día})}$$

Recurso software	Precio (€)	Horas estimadas	Coste amortización (€/h)	Coste estimado (€)
GNU Emacs	0	No trasciende	0	0
Microsoft Office Word 2013	100	188	0,01	2
Microsoft Office Excel 20163				
Git v1.9.5	0	No trasciende	0	0
Correo electrónico Gmail	0	No trasciende	0	0
Adobe Reader XI	0	No trasciende	0	0
Microsoft Project 2013	765	30	0,12	98
Ubuntu 14.04	0	No trasciende	0	0
<b>Total</b>				<b>100 €</b>

Tabla 7 Costes estimado por software

- **Gastos generales**

Durante todo el transcurso del proyecto se estará utilizando el ordenador portátil con un consumo energético. También se ha de tener en cuenta la recarga de batería de los dos Smartphones utilizados. Se considera que cada uno se tiene que recargar cada 2 días y durante 3 horas. Respecto a la conexión a internet, se utilizará una de 30MB. También se tiene en cuenta el desplazamiento en transporte público en el área metropolitana de Barcelona utilizando una T-Jove de 1 zona [14].

Recurso	Precio	Cantidad	Coste estimado (€)
Consumo energético	0,13 €/kWh	Consumo diario de 1,8 kWh durante 5 meses	35,67
Conexión a Internet	27 €/mes	5 meses	135
Desplazamientos	105 € cada 3 meses	5 meses	210
<b>Total</b>			<b>385,77 €</b>

Tabla 8 Gastos generales iniciales

#### 6.8.5. CONTINGENCIA

Como a medida de contingencia se establece un margen del 20% de la suma de los costes directos, costes de hardware y software y de imprevistos.

Recursos	Coste € (Caso 1)	Coste € (Caso 2)
Costes directos	3888	6794,65
Recursos hardware	120	120
Recursos software	100	100
Imprevistos	380,67	380,67
<b>Total Contingencia</b>	<b>897,73 €</b>	<b>1479,06 €</b>

Tabla 9 Contingencia fase inicial

#### 6.8.6. COSTE TOTAL

En la tabla 16 podemos observar el coste total para cada caso que se planificó durante la fase inicial. Para realizar el cálculo final tenemos en cuenta que:

- No se llevará a cabo ningún aumento de precio durante el proyecto, ya que no es de larga duración.
- Dado que es un proyecto sin ánimo de lucro no se añade al presupuesto final ningún beneficio sobre el coste total.

Recursos	Coste € (Caso 1)	Coste € (Caso 2)
Costes directos	3592	6700,65
Recursos hardware	120	120
Recursos software	100	100
Imprevistos	380,67	380,67
Contingencia	897,73	1479,06
Gastos generales	385,77	385,77
<b>Total</b>	<b>5411,89 €</b>	<b>9142,27 €</b>

Tabla 10 Coste total fase inicial

## 6.9. CONTROL DE GESTIÓN

A medida que el proyecto avance y se pueda determinar el coste real de los recursos y de las horas de trabajo dedicadas se podrán calcular desviaciones de la siguiente manera:

- Desvío de mano de obra en precio = (coste estimado – coste real) \* consumo de horas real
- Desvío en la realización de una tarea en precio = (coste estimado – coste real) \* consumo horas reales
- Desvío de un recurso en precio = (coste estimado – coste real) \* consumo real
- Desvío en la realización de una tarea en consumo = (consumo estimado – consumo real) \* coste real
- Desvío total en la realización de tareas = coste total estimado tarea – coste total real tarea
- Desvío total en recursos = coste total estimado recursos – coste total real recursos
- Desvío total costes fijos = coste total coste fijos presupuestado – coste total fijo real

## 7. EJECUCIÓN REAL

Finalmente el proyecto ha tenido una duración de 8 meses, con inicio del 26 de enero y finalización el 26 de septiembre. Los días posteriores a la fecha de finalización son dedicados a la exposición oral de la cual se ha fijado para el 28 de octubre. Este proyecto ha tenido una carga de trabajo de 800 horas aproximadamente. El aumento de horas para realizar este sistema se ha producido en las etapas de desarrollo y testeo tanto de la parte del back-end como la del front-end.

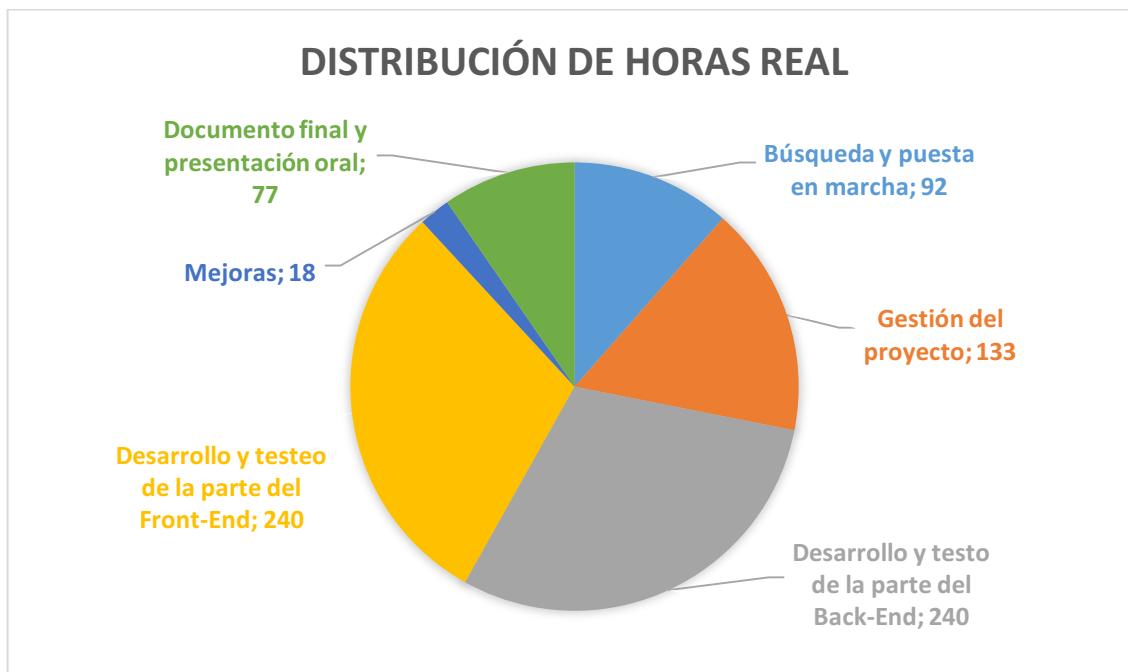


Figura 3 Distribución de horas real

Como finalmente se han necesitado 800 horas para llevar a cabo este sistema, la planificación de horas por fase aparece la reflejada en la tabla 11.

Tarea	Responsable	Horas
Búsqueda y puesta en marcha		92
Búsqueda y puesta en marcha	Jefe de proyecto	24
Aprendizaje autónomo	Programador	68
Gestión del proyecto		133
Definición del alcance y contextualización	Jefe de proyecto	34
Planificación temporal	Jefe de proyecto	31
Gestión económica y sostenible	Jefe de proyecto	21
Presentación preliminar	Jefe de proyecto	8
Pliego de condiciones	Jefe de proyecto	8
Documento final y presentación oral	Jefe de proyecto	32
Desarrollo y testeo de la parte del Back-End		240
Análisis de requisitos	Analista	12
Diseño y Especificación	Diseñador	16
Implementación y pruebas	60% Programador / 40% tester	212
Desarrollo y testeo de la parte del Front-End		240
Análisis de requisitos	Analista	12
Diseño y especificación	Diseñador	16
Implementación y pruebas	60% Programador / 40% tester	212
Mejoras		18
Análisis y diseño	50% Analista / 50% Diseñador	5
Implementación y pruebas	60% Programador / 40% tester	13
Documento final y presentación oral		77
Redacción de la memoria	Jefe de proyecto	56
Preparación presentación oral	Jefe de proyecto	21
<b>Total</b>		<b>800</b>

Tabla 11 Estimación de horas real

El diagrama de Gantt que hay a continuación contiene todas las fases que finalmente se han llevado a cabo. En la figura 4 y en la tabla 12 podemos observar cómo se han dedicado más días a la fase de desarrollo y testeo de la parte del back-end. La estimación inicial fue de 26 días y finalmente se han necesitado 57 días. Esto ha sido debido a que el desarrollador del sistema ha necesitado dedicar más tiempo en la implementación de cada servicio y en el testeo del mismo. Lo mismo ha ocurrido con la fase de desarrollo y testeo de la la parte del front-end. Donde inicialmente se planificaron 26 días, finalmente se han necesitado 52 días. Esto se ha dado debido a que el desarrollador era nuevo en el

uso del framework y además al ser dicho framework muy novedoso, la documentación que se ha podido encontrar como soporte ha sido escasa.

Para la fase de mejoras se ha dedicado un día menos del especificado en la fase inicial ya que el autor del proyecto ha dado prioridad a crear la documentación final antes que a hacer pequeñas mejoras en el sistema.

Resource Names	Duration	Start	Finish
<b>1. Búsqueda y puesta en marcha</b>	<b>20 days</b>	<b>Tue 26/01/16</b>	<b>Mon 22/02/16</b>
Búsqueda y puesta en marcha	5 days	Tue 26/01/16	Mon 01/02/16
Aprendizaje autónomo	15 days	Tue 02/02/16	Sun 21/02/16
<b>2. Gestión del proyecto</b>	<b>29 days</b>	<b>Mon 22/02/16</b>	<b>Thu 31/03/16</b>
Definición del alcance y contextualización	8 days	Mon 22/02/16	Wed 02/03/16
Planificación temporal	7 days	Thu 03/03/16	Fri 11/03/16
Gestión económica y sostenible	5 days	Sat 12/03/16	Thu 17/03/16
Presentación preliminar	2 days	Fri 18/03/16	Sun 20/03/16
Pliego de condiciones	2 days	Mon 21/03/16	Tue 22/03/16
Documento final y presentación oral	7 days	Wed 23/03/16	Thu 31/03/16
<b>3. Desarrollo y testeo de la parte del back-end</b>	<b>57 days</b>	<b>Fri 01/04/16</b>	<b>Mon 20/06/16</b>
Análisis de requisitos	3 days	Fri 01/04/16	Tue 05/04/16
Diseño y Especificación	4 days	Wed 06/04/16	Sun 10/04/16
Implementación y pruebas	51 days	Mon 11/04/16	Mon 20/06/16
<b>4. Desarrollo y testeo de la parte del front-end</b>	<b>52 days</b>	<b>Tue 21/06/16</b>	<b>Wed 31/08/16</b>
Análisis de requisitos	3 days	Tue 21/06/16	Thu 23/06/16
Diseño y Especificación	4 days	Fri 24/06/16	Wed 29/06/16
Implementación y pruebas	45 days	Thu 30/06/16	Wed 31/08/16
<b>5. Mejoras</b>	<b>3 days</b>	<b>Thu 01/09/16</b>	<b>Sun 04/09/16</b>
Análisis y diseño	2 days	Thu 01/09/16	Fri 02/09/16
Implementación y pruebas	2 days	Sat 03/09/16	Sun 04/09/16
<b>6. Documento final y presentación oral</b>	<b>24 days</b>	<b>Mon 05/09/16</b>	<b>Thu 06/10/16</b>
Redacción de la memoria	16 days	Mon 05/09/16	Mon 26/09/16
Preparación presentación oral	8 days	Tue 27/09/16	Thu 06/10/16

Tabla 12 Daigramma de Gantt real

Para la fase de documentación final y presentación oral se ha pasado de creer necesitar 11 días a finalmente requerir 24 días. Debido a que finalmente el proyecto ha necesitado más trabajo del planificado en la fase inicial, hace falta redactar y justificar todo lo que se ha añadido al proyecto. Por cada fase se muestra su respectiva duración, fechas de inicio y finalización y responsable de la tarea.

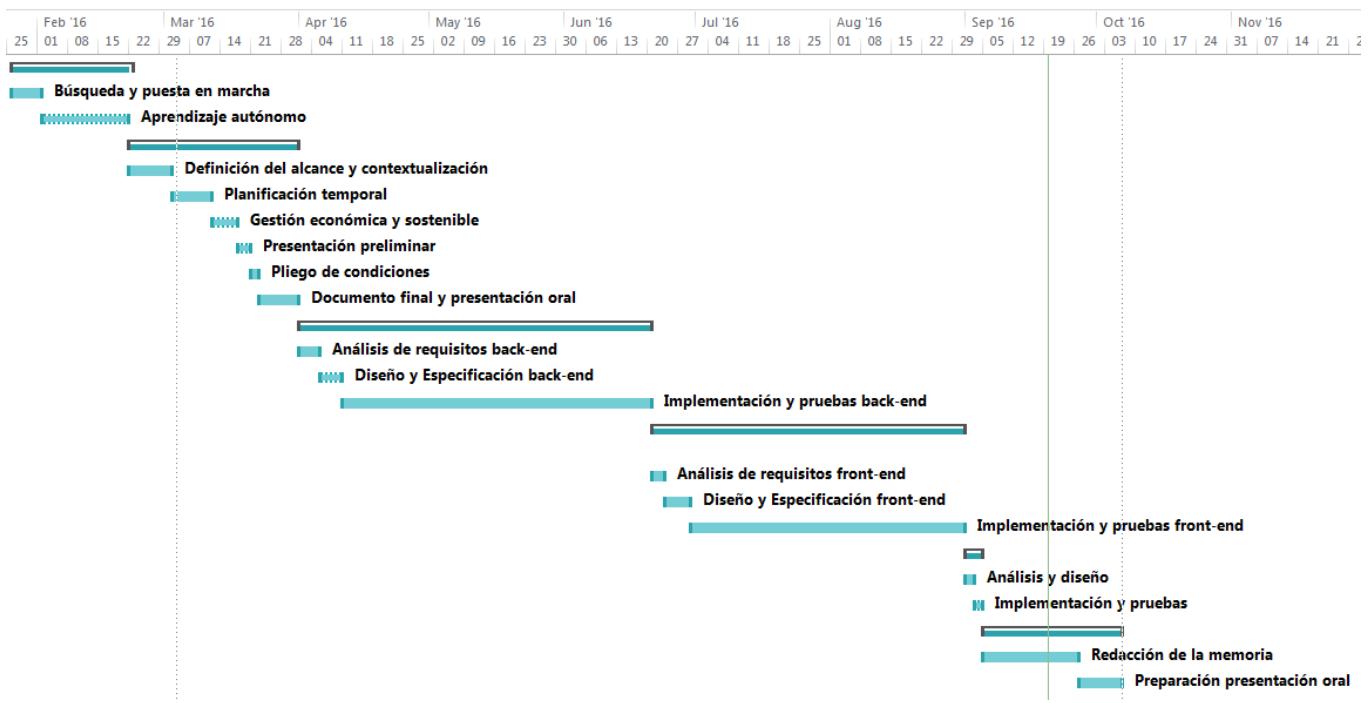


Figura 4 Diagrama de Gantt real

Dado que la planificación varió desde que fue definida al inicio hasta el momento de la finalización del proyecto, el coste final de los recursos humanos por casos ha variado. Si la tabla 13 la comparamos con la tabla 4 (coste de recursos humanos en la planificación inicial) que aparece en la sección 6.8.2 página 23 observamos la diferencia de costes. Para el caso 1 se ha pasado de 4.320€ a 6.400€ y en el caso 2 de 7.874€ a 11.595€ en costes de recursos humanos. Recordemos que el caso 1 que se hace partiendo de que la autora hace todos los roles y los lleva a cabo como becaria y lleva a cabo todas las tareas. El caso 2 será teniendo en cuenta el salario específico de cada rol sin ser becario.

Rol	Horas estimadas	Remuneración (€/h) (Caso 1)	Remuneración (€/h) (Caso 2)	Coste estimado € (Caso 1)	Coste estimado € (Caso 2)
Jefe de proyecto	235	8	20	1.880	4.700
Analista	26,5	8	16	212	424
Diseñador	34,5	8	10,5	276	362
Programador	330,2	8	10,5	2.642	3.467
Tester	174,8	8	11	1.398	1.923
Total	801			6.408 €	10.876,15 €

Tabla 13 Coste final de recursos humanos por casos

Nombre de la actividad	Horas estimadas	Recursos	Coste € (Caso 1)	Coste € (Caso 2)
Búsqueda y puesta en marcha	92		736	1194
Búsqueda y puesta en marcha	24	Jefe de proyecto	192	480
Aprendizaje autónomo	68	Programador	544	714
Gestión del proyecto	133		1072	2680
Definición del alcance y contextualización	34	Jefe de proyecto	272	680
Planificación temporal	31	Jefe de proyecto	248	620
Gestión económica y sostenible	21	Jefe de proyecto	168	420
Presentación preliminar	8	Jefe de proyecto	64	160
Pliego de condiciones	8	Jefe de proyecto	64	160
Documento final y presentación oral	32	Jefe de proyecto	256	640
Desarrollo y testo de la parte del Back-End	240		1920	2628,4
Análisis de requisitos	12	Analista	96	192
Diseño y Especificación	16	Diseñador	128	168
Implementación y pruebas	212	60% Programador / 40% tester	1696	2268,4
Desarrollo y testeo de la parte del Front-End	240		1920	2628,4
Análisis de requisitos	12	Analista	96	192
Diseño y especificación	16	Diseñador	128	168
Implementación y pruebas	212	60% Programador / 40% tester	1696	2268,4
Mejoras	18		176	205,35
Análisis y diseño	5	50% Analista / 50% Diseñador	40	66,25
Implementación y pruebas	13	60% Programador / 40% tester	104	139,1
Documento final y presentación oral	77		616	1540
Redacción de la memoria	56	Jefe de proyecto	448	1120
Preparación presentación oral	21	Jefe de proyecto	168	420
<b>Total</b>	<b>801</b>		<b>5.368 €</b>	<b>10.876,15 €</b>

Tabla 14 Costes directos por actividad por casos reales

Relativo a los gastos generales también ha habido cambios. La tabla 15 hace referencia tanto al consumo energético tanto de los Smartphones como de los portátiles. También se tiene en cuenta la conexión a internet y los desplazamientos. Para estos valores se han tenido en cuenta un período de tiempo de 8 meses. Inicialmente y como muestra la tabla 8 de la sección 6.8.3 se planificó para 5 meses. Es por eso que podemos observar que los gastos han pasado de 385€ a 582€ que supone un incremento de 197€.

Recurso	Precio	Cantidad	Coste estimado (€)
Consumo energético	0,13 €/kWh	Consumo diario de 1,8 kWh durante 8 meses	51
Conexión a Internet	27 €/mes	8 meses	216
Desplazamientos	105 € cada 3 meses	8 meses	315
<b>Total</b>			<b>582,00 €</b>

Tabla 15 Gastos generales reales

Respecto a la contingencia, establecida en un margen del 20%, ha aumentado. Como los costes directos se han visto incrementados también lo ha hecho la contingencia. Como se puede observar en la tabla 9 de la sección 6.8.4, la contingencia inicialmente era de 897€ para el caso 1 y de 1479€ para el caso 2. El incremento observado de la contingencia inicial con la contingencia real que aparece en la tabla 16 es de 296€ para el caso 1 y de 816 para el caso 2.

Recursos	Coste € (Caso 1)	Coste € (Caso 2)
Costes directos	5368	10876
Recursos hardware	120	120
Recursos software	100	100
Imprevistos	380,67	380,67
<b>Total Contingencia</b>	<b>1.193,73 €</b>	<b>2.295,33 €</b>

Tabla 16 Contingencia real

Finalmente, después de calcular todos los costes reales que han aparecido anteriormente, el coste total real que ha supuesto la realización de este proyecto es de 7.743€ para el caso 1 de 14.353€ para el caso 2. A diferencia de lo que se calculó durante la fase inicial que fue de 5411€ en el caso 1 y de 9.142€ para el caso 2 como aparece en la tabla 10 de la sección 6.8.5. El coste total por lo tanto ha aumentado en 2.332€ en el caso 1 y 5.211€ para el caso 2.

Recursos	Coste € (Caso 1)	Coste € (Caso 2)
Costes directos	5368	10876
Recursos hardware	120	120
Recursos software	100	100
Imprevistos	380	380
Contingencia	1193	2295
Gastos generales	582	582
<b>Total</b>	<b>7.743,00 €</b>	<b>14.353,00 €</b>

Tabla 17 Coste total real

## 8. SOSTENIBILIDAD Y COMPROMISO SOCIAL

A continuación presentamos el estudio sobre sostenibilidad que se ha hecho referente al proyecto. En la tabla 10 presentamos los resultados, que se explican más adelante, en forma de la matriz propuesta en la guía de la asignatura.

Sostenibilidad		Económica	Social	Ambiental
PPP		Factura 8 / 10	Impacto personal 9 / 10	Consumo de diseño 9 / 10
Vida útil		Plan de viabilidad 18 / 20	Impacto social 19 / 20	Huella ecológica 19 / 20
Valoración		26 / 30	28 / 30	28 / 30
Total		82 / 90		

Tabla 18 Valoración sostenibilidad

### 8.1. ECONÓMICA

Para considerar este proyecto económicamente viable se ha tenido en cuenta el coste en caso de que se produjeran cambios en la planificación temporal además de cualquier tipo de reparación en los recursos usados para el desarrollo del sistema.

En caso de que las tecnologías utilizadas para desarrollar el sistema ya estuvieran aprendidas, el tiempo para realizar el proyecto sería menor. De esta manera el coste también disminuiría. Pero no hay que olvidar que dos de los objetivos del proyecto son aprender estas herramientas. Es por esto que el tiempo de autoaprendizaje es necesario. Cabe la posibilidad de que partes del proyecto se hayan podido utilizar de otros trabajos, pero la naturaleza didáctica del proyecto hace que se descarte hacer uso de otros sistemas.

En cuestión de sostenibilidad económica, este proyecto merece una valoración de  $\frac{26}{30}$ . No obtiene el máximo de nota ya que el proyecto podría llevarse a cabo en menos tiempo. Esto solo sería posible aumentando el personal del proyecto, pero dado el carácter de proyecto de final de carrera de este sistema esa posibilidad queda descartada.

### 8.2. SOCIAL

Actualmente el país está pasando por una crisis económica que afecta directamente al medio ambiente. Los casos de corrupción se cuentan ya por centenares y detrás de ellos quedan los impactos al medio ambiente con espacios naturales llenos de hormigón o suelos contaminados por enterramientos ilegales de residuos peligrosos, entre otros [15]. Cabe destacar que dada esta situación de crisis económica, se ven perjudicados varios ministerios presentados por el Gobierno en los Presupuestos del Estado. Se ven especialmente afectados los apartados de medio ambiente, y materias como la



conservación de la biodiversidad, el desarrollo rural sostenible y la lucha contra el cambio climático. Éste último sufre el mayor recorte con un 48% menos pasando de 101 millones a 52 millones [16]. Este proyecto que lleva a cabo pretende hacer mejorar la sociedad y por lo tanto esta situación problemática por la que atraviesa el país. Enseña a los usuarios cuál es su huella ecológica, el progreso que tienen los hábitos que realiza durante el día. Algunos autores sostienen que si cada habitante del mundo en desarrollo dejara la misma huella ecológica que el habitante promedio de los países de ingreso alto, se necesitarían seis planetas como la Tierra. No podemos seguir a este ritmo de manera que es necesario concienciar a la gente del problema y enseñarles la realidad del problema [17].

El sistema resultante de este proyecto puede cambiar el estilo de vida del usuario. Le mostrará el progreso de sus hábitos relacionados con el medioambiente. Además le aportará información adicional sobre temas medioambientales.

Cabe recordar que cualquier sector que lleve a cabo prácticas poco sostenibles se verá perjudicado por este sistema. Principalmente será el sector de motor ya que este sistema incita a usar la bicicleta y dejar de lado cualquier vehículo que consuma combustibles no renovables. También se verá afectado el sector alimentario, sobretodo el cárnico y el que comercialice alimentos procedentes de pescado. Las compañías encargadas de gestionar los servicios relacionados con el agua que se consume en los hogares también serán perjudicadas. Por último también se verá afectadas las compañías eléctricas que reciban beneficios del consumo de aire acondicionado y calefacción.

En cuestión de sostenibilidad social, este proyecto merece una valoración de  $\frac{28}{30}$ . Le corresponde casi la nota máxima ya que este sistema permitirá mejorar la sociedad. Los escasos recursos deben emplearse de forma adecuada y racional, y en este proyecto ha primado diseñar, usando nuevas tecnologías, desarrollar un sistema que cree conciencia de la problemática que supone actualmente la degradación del medio ambiente.

### 8.3. AMBIENTAL

Todos los recursos necesarios para llevar a cabo este proyecto ya se usaban antes, por lo tanto no ha sido necesario comprar nuevos. En el caso del portátil, éste tiene un consumo eléctrico que se tiene en cuenta, pero debido a que ya se usa para otras funciones aparte del desarrollo del proyecto, no supone un gran incremento en este consumo.

Cabe destacar también que el consumo e impacto ambiental de mi TFG no supone ningún gasto en papel ya que todo se hará de manera digital. La única energía necesaria para la realización del proyecto es la eléctrica que consume el portátil. También la que se use en proceso de recarga de las baterías de los teléfonos móviles usados.

Además al ser un producto de software se podrá reaprovechar cualquier parte del código para otros proyectos. Tampoco requiere de ningún coste de fabricación ni se generará ningún tipo de contaminación. Además no nos tendremos que preocupar del reciclaje, simplemente se borrará el software del disco duro y de la Play Store.



Una de las principales metas que tiene la implantación de este sistema es que cada usuario sea consciente de su huella ecológica para así poder reducirla. Es por esto que en cuestión de sostenibilidad ambiental, este proyecto merece una valoración de  $\frac{28}{30}$ . El sistema pretende reducir el uso indiscriminado de los recursos naturales y el desarrollo de sustancias artificiales ofreciendo una aplicación móvil a los usuarios. Además se ahorrará tanto en papel como en tinta aunque cierto es que se consumirá energía extra tanto del ordenador portátil como de los teléfonos móviles.

## 9. REQUISITOS Y ESPECIFICACIÓN

A continuación, se van a describir los requisitos, tanto funcionales como no funcionales que se han tenido en cuenta para que el sistema sea considerado de calidad. Dado que el sistema no ha sido desarrollado para un cliente definido, los requisitos han sido generados por el autor del proyecto a partir del estudio de mercado explicado al inicio del proyecto y consultados y revisados por la directora del proyecto.

### 9.1. REQUISITOS FUNCIONALES

En este apartado se definen los casos de uso del sistema mediante diagramas generales de casos de uso relacionándolos con los actores implicados (partes interesadas definidas en la sección 1.2 página 9).

También se van a describir los casos de uso del sistema mediante una tabla que donde se describe:

- Actor principal.
- Precondición.
- Disparador
- Escenario principal de éxito
- Extensiones

#### 9.1.1. DIAGRAMA GENERAL DE CASOS DE USO

A continuación se presentan los diagramas de casos de uso del sistema relacionándolos con los actores implicados.

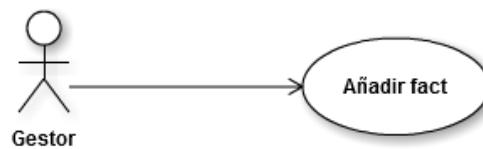


Figura 5 Diagrama de casos de uso del gestor

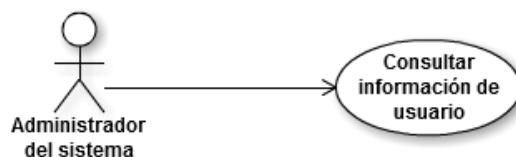


Figura 6 Diagrama de casos de uso del administrador

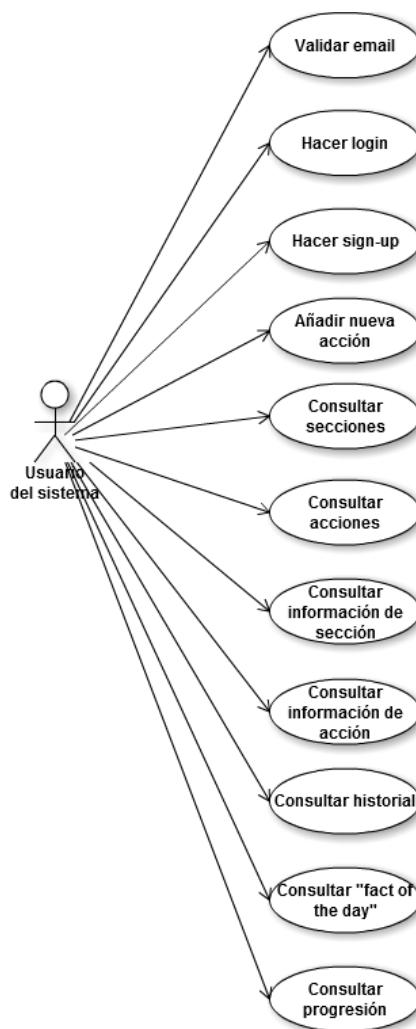


Figura 7 Diagrama de casos de uso del usuario del sistema

### 9.1.2. DESCRIPCIÓN DE LOS CASOS DE USO

#### 1.1.1.1 VALIDAR EMAIL

##### **Actor principal**

Usuario del sistema

##### **Disparador**

El usuario quiere hacer uso de la aplicación.

##### **Precondiciones**

El usuario no ha iniciado sesión ni se ha registrado en el sistema.

##### **Escenario principal de éxito**

1. El usuario abre la aplicación.
2. El sistema muestra la pantalla de validación de email.
3. El usuario hace click en la casilla correspondiente a introducir el email.
4. El sistema muestra el teclado del dispositivo.
5. El usuario escribe el email que quiere validar en el sistema.
6. El sistema valida el email
  - 6.1. Si el email ya existe en el sistema se realiza el caso de uso *Hacer login*.
  - 6.2. Si el email no existe en el sistema se realiza el caso de uso *Hacer sign-up*.

Tabla 19 Descripción del caso de uso "Validar email"



#### 1.1.1.2 HACER LOGIN

##### Actor principal

Usuario del sistema

##### Disparador

El usuario quiere iniciar sesión en el sistema.

##### Precondiciones

El usuario ha validado el email y este email ya había sido registrado previamente en el sistema.

##### Escenario principal de éxito

1. El usuario hace click en la casilla correspondiente a introducir la contraseña que corresponde al email introducido en el paso anterior.
2. El sistema muestra el teclado del dispositivo.
3. El usuario escribe la contraseña que quiere validar en el sistema.
4. El usuario hace click en el botón de *login*.
5. El sistema valida la contraseña.
6. El sistema autoriza al usuario para utilizar el sistema. El flujo continúa en el caso de uso *Añadir nueva acción*.

Tabla 20 Descripción del caso de uso "Hacer login"

#### 1.1.1.3 HACER SIGN-UP

##### Actor principal

Usuario del sistema

##### Disparador

El usuario quiere registrarse en el sistema.

##### Precondiciones

El usuario ha validado el email y este email no ha sido registrado previamente en el sistema.

##### Escenario principal de éxito

1. El usuario hace click en la casilla correspondiente a introducir la contraseña que corresponde al email introducido en el paso anterior.
2. El sistema muestra el teclado del dispositivo.
3. El usuario escribe la contraseña que quiere validar en el sistema.
4. El usuario hace click en el botón de *sign-up*.
5. El sistema valida la contraseña que es correcta.
6. El sistema autoriza al usuario para utilizar el sistema. El flujo continúa en el caso de uso *Añadir nueva acción*.

##### Extensiones

- 6.a. El sistema valida la contraseña, que es incorrecta.
- 6.b. El sistema muestra por pantalla un mensaje de texto "Invalid password".

*Se repite el paso 3 hasta que el usuario decida terminar.*

Tabla 21 Descripción del caso de uso "Hacer sign-up"

#### 1.1.1.4 AÑADIR NUEVA ACCIÓN

**Actor principal**

Usuario del sistema

**Disparador**

El usuario ha llevado a cabo una nueva acción y quiere registrarla en el sistema.

**Precondiciones**

El usuario tiene la sesión iniciada en el sistema y ha abierto la aplicación.

**Escenario principal de éxito**

1. El usuario hace click en la pestaña “New action” del menú.
2. El sistema muestra la pantalla “New action”.
3. El usuario navega por las diferentes secciones de la pantalla hasta encontrar la acción que quiere registrar.
4. El usuario hace click en la acción que ha llevado a cabo para marcarla.
5. El sistema muestra el botón de “Ok” como activo.
6. El usuario hace click en el botón de “Ok”.
7. El sistema muestra un loading spinner mientras lleve a cabo el registro de la acción en el sistema.
8. El sistema registra la acción en el sistema.
9. El sistema actualiza el valor del “score” mostrado por pantalla mediante una animación.
10. El sistema desmarca la acción previamente registrada y muestra el botón de “Ok” como inactivo.

*El usuario repite este proceso tantas veces como deseé.*

**Extensiones**

- 5.a. El usuario no quiere registrar la acción

5.a.1. El usuario desmarca la acción que quería registrar.

5.a.2. El sistema muestra el botón de “Ok” como inactivo.

Tabla 22 Descripción del caso de uso "Añadir nueva acción"

#### 1.1.1.5 CONSULTAR SECCIONES

**Actor principal**

Usuario del sistema

**Disparador**

El usuario quiere consultar las secciones que ofrece el sistema.

**Precondiciones**

El usuario tiene la sesión iniciada en el sistema y ha abierto la aplicación

**Escenario principal de éxito**

1. El usuario hace click en la pestaña “New action” del menú.
2. El sistema muestra la pantalla “New action” que contiene las secciones a consultar por el usuario.
3. El usuario consulta las diferentes secciones que el sistema ofrece. El sistema ofrece cuatro secciones por las que el usuario puede navegar:
  - Food
  - Water
  - Transportation
  - Temperature

Tabla 23 Descripción del caso de uso "consultar secciones"

#### 1.1.1.6 CONSULTAR ACCIONES

##### **Actor principal**

Usuario del sistema

##### **Disparador**

El usuario quiere consultar las acciones que ofrece el sistema.

##### **Precondiciones**

El usuario tiene la sesión iniciada en el sistema y ha abierto la aplicación

##### **Escenario principal de éxito**

1. El usuario hace click en la pestaña “New action” del menú.
2. El sistema muestra la pantalla “New action” que contiene las secciones a consultar por el usuario.
3. El usuario navega por las diferentes secciones y selecciona la que quiere consultar sus acciones.
4. El sistema muestra las acciones de la sección seleccionada por el usuario.
5. El usuario consulta las diferentes acciones que el sistema ofrece. El sistema ofrece cuatro acciones por cada una de las cuatro secciones:

Food:	Water:	Transportation:	Temperature:
<input type="radio"/> Meat	<input type="radio"/> Short shower	<input type="radio"/> Bike	<input type="radio"/> Turn off cooling
<input type="radio"/> No meat	<input type="radio"/> Long shower	<input type="radio"/> Plane	<input type="radio"/> Turn off heating
<input type="radio"/> Fish	<input type="radio"/> Take a bath	<input type="radio"/> Car	<input type="radio"/> Wear a sweater
<input type="radio"/> No fish	<input type="radio"/> Wash with cold water	<input type="radio"/> Public transportation	<input type="radio"/> Open a window

Tabla 24 Descripción del caso de uso "consultar acciones"

#### 1.1.1.7 CONSULTAR INFORMACIÓN DE SECCIÓN

##### **Actor principal**

Usuario del sistema

##### **Disparador**

El usuario quiere consultar la información de una sección ofrecida por el sistema.

##### **Precondiciones**

El usuario tiene la sesión iniciada en el sistema y ha abierto la aplicación.

##### **Escenario principal de éxito**

1. El usuario hace click en la pestaña de “New action” del menú.
2. El sistema muestra la pantalla “New action” que contiene las secciones a consultar por el usuario.
3. El usuario navega por las diferentes secciones y selecciona la sección requerida.
4. El sistema muestra el botón de “info” de la sección seleccionada junto con las acciones de la sección.
5. El usuario hace click en el botón “info”.
6. El sistema muestra una pantalla con la información de la sección.
7. El usuario consulta la información de la sección.

Tabla 25 Descripción del caso de uso "consultar información de sección"

#### 1.1.1.8 CONSULTAR INFORMACIÓN DE ACCIÓN

**Actor principal**

Usuario del sistema

**Disparador**

El usuario quiere consultar la información de una acción ofrecida por el sistema.

**Precondiciones**

El usuario tiene la sesión iniciada en el sistema y ha abierto la aplicación.

**Escenario principal de éxito**

1. El usuario hace click en la pestaña de “New action” del menú.
2. El sistema muestra la pantalla “New action” que contiene las secciones a consultar por el usuario.
3. El usuario navega por las diferentes secciones y selecciona la sección requerida.
4. El sistema muestra el botón de “info” de la sección seleccionada junto con las acciones de la sección.
5. El usuario hace click en el botón “info”.
6. El sistema muestra una pantalla con la información de las acciones.
7. El usuario consulta la información de la acción.

Tabla 26 Descripción del caso de uso "consultar información de acción"

#### 1.1.1.9 CONSULTAR HISTORIAL

**Actor principal**

Usuario del sistema

**Disparador**

El usuario quiere consultar el historial de sus acciones que el sistema ofrece.

**Precondiciones**

El usuario tiene la sesión iniciada en el sistema y ha abierto la aplicación.

**Escenario principal de éxito**

1. El usuario hace click en la pestaña de “History” del menú
2. El sistema muestra la pantalla “History” que contiene las secciones a consultar por el usuario.
3. El usuario navega y consulta el historial de las acciones que ha llevado a cabo hasta el momento.

Tabla 27 Descripción del caso de uso "consultar historial"

#### 1.1.1.10 CONSULTAR “FACT OF THE DAY”

**Actor principal**

Usuario del sistema

**Disparador**

El usuario quiere consultar el hecho del día que el sistema ofrece.

**Precondiciones**

El usuario tiene la sesión iniciada en el sistema y ha abierto la aplicación.

**Escenario principal de éxito**

1. El usuario hace click en la pestaña de “Fact of the day” del menú.
2. El sistema muestra la pantalla “Fact of the day” que contiene las secciones a consultar por el usuario.
3. El usuario navega y consulta el hecho del día.

Tabla 28 Descripción del caso de uso "consultar fact of the day"

#### 1.1.1.11 CONSULTAR PROGRESIÓN

**Actor principal**

Usuario del sistema

**Disparador**

El usuario quiere consultar la progresión de su puntuación.

**Precondiciones**

El usuario tiene la sesión iniciada en el sistema y ha abierto la aplicación.

**Escenario principal de éxito**

1. El usuario hace click en la pestaña de “Progression” del menú.
2. El sistema muestra la pantalla “Progression” que contiene un gráfico que muestra la evolución de la puntuación del usuario.
3. El usuario consulta la progresión de su puntuación hasta el momento.

Tabla 29 Descripción del caso de uso "consultar progresión"

#### 1.1.1.12 AÑADIR FACT

**Actor principal**

Gestor

**Disparador**

El gestor quiere añadir un nuevo *fact* al sistema.

**Precondiciones**

El gestor tiene permisos para modificar el contenido desde *Amazon Web Services*.

**Escenario principal de éxito**

1. El gestor inicia sesión en la consola de AWS.
2. Modifica el fichero de AWS que contiene todos los *facts* que se muestran en la aplicación.
3. Sube a AWS el fichero actualizado.

Tabla 30 Descripción del caso de uso "Añadir fact"

#### 1.1.1.13 CONSULTAR INFORMACIÓN DE USUARIO

**Actor principal**

Administrador del sistema

**Disparador**

El usuario quiere consultar la información de un usuario en la base de datos.

**Precondiciones**

El administrador del sistema tiene permisos para consultar el contenido de la DynamoDB.

**Escenario principal de éxito**

1. El administrador del sistema inicia sesión en la consola de AWS.
2. Accede al bucket que desea de DynamoDB.
3. Accede a la tabla que desea y consulta la información del usuario.

Tabla 31 Descripción del caso de uso "Consultar información de usuario"

## 9.2. REQUISITOS NO FUNCIONALES

Antes de describir todos los requisitos, vamos a explicar qué indica cada apartado de las tablas de Volére [18] de la descripción de los requisitos no funcionales:

- Número de requisito: es simplemente un identificador, por si tenemos que referirnos a él en otros apartados del documento.
- Tipo: bajo qué tipo agrupamos el requisito.
- Descripción: el requisito en sí, lo que queremos que se cumpla en el sistema.
- Justificación: cuál es el motivo por el que queremos que se cumpla este requisito.
- Criterio de satisfacción: cómo sabremos si el requisito se ha cumplido.
- Satisfacción del usuario: de 1 a 5 (de menos a más), cómo se sentirá de satisfecho el usuario si se cumple el requisito.
- Insatisfacción del usuario: de 1 a 5 (de menos a más), cómo se sentirá de insatisfecho el usuario si no se cumple el requisito.
- Prioridad: la importancia que tiene en el sistema que este requisito se cumpla.

### 9.2.1. REQUISITOS DE PERCEPCIÓN

#### 1.1.1.14 REQUISITOS DE APARIENCIA

Número de requisito	#01	Tipo						
Descripción	El diseño de la interfaz resulta atractivo para el usuario							
Justificación	La interfaz resultará agradable a la vista de los usuarios mediante una combinación de colores que invitará a los usuarios a interaccionar fácilmente con nuestro sistema y de esta manera sentirse cómodos.							
Criterio de satisfacción	Se realizará una encuesta a los usuarios que hayan usado más de una vez el sistema, y el 90% o más confirmará el atractivo del diseño.							
Satisfacción del usuario	4	Insatisfacción del usuario	3	Prioridad	Alta			

Tabla 32 Tabla de Volére del requisito #01

#### 1.1.1.15 REQUISITOS DE ESTILO

Número de requisito	#02	Tipo						
Descripción	El diseño de la interfaz es sencillo y ordenado							
Justificación	La mayor parte de los usuarios no suelen tener un nivel alto en aplicaciones móvil, por ello la interfaz deberá ser simple y no se utilizarán elementos que puedan ocasionar confusiones.							
Criterio de satisfacción	Se realizará una encuesta a los usuarios donde el 80% o más deberá llevar a cabo las acciones con facilidad							
Satisfacción del usuario	4	Insatisfacción del usuario	2	Prioridad	Alta			

Tabla 33 Tabla de Volére del requisito #02

## 9.2.2. REQUISITOS DE CAPACIDAD DE USO Y HUMANIDAD

### 1.1.1.16 REQUISITOS DE FACILIDAD DE UTILIZACIÓN

Número de requisito	#03	Tipo				
Descripción	El sistema es fácil de utilizar para los usuarios.					
Justificación	Los usuarios son de diversas edades y serán los mayores críticos para este sistema, por ello el sistema será útil y simple en el momento de hacer uso de él.					
Criterio de satisfacción	Cuando se hagan las pruebas de testing con usuarios, se evaluará la interacción que estos tengan con el sistema y se irán midiendo los tiempos que cada usuario requiera. El 95% de los usuarios no debería tener problemas que impliquen un exceso de tiempo en las pruebas.					
Satisfacción del usuario	4	Insatisfacción del usuario	2	Prioridad Alta		

Tabla 34 Tabla de Volére del requisito #03

### 1.1.1.17 REQUISITOS DE APRENDIZAJE

Número de requisito	#04	Tipo				
Descripción	El sistema se utiliza sin recibir una formación previa.					
Justificación	Los usuarios no deberán necesitar una formación para poder utilizar el sistema.					
Criterio de satisfacción	El 85% o más de las personas que no han trabajado con sistemas similares no deberán tener problemas con la utilización del sistema.					
Satisfacción del usuario	4	Insatisfacción del usuario	2	Prioridad Alta		

Tabla 35 Tabla de Volére del requisito #04

### 1.1.1.18 REQUISITOS DE COMPRENSIÓN Y CORTESÍA

Número de requisito	#05	Tipo				
Descripción	El sistema tiene un registro lingüístico formal, comprensible por cualquier usuario.					
Justificación	Los usuarios deben entender el lenguaje del sistema sin ningún tipo de complicaciones.					
Criterio de satisfacción	Se utiliza un lenguaje comprensible, por lo cual no se esperan complicaciones. Durante la fase de testeo se observará que los usuarios que lleven a cabo las pruebas de usabilidad no tengan ningún problema con el registro lingüístico utilizado.					
Satisfacción del usuario	4	Insatisfacción del usuario	2	Prioridad Media		

Tabla 36 Tabla de Volére del requisito #05

Número de requisito	#06	Tipo				
Descripción	El sistema utiliza iconos fáciles de identificar.					
Justificación	Con la utilización de iconos ya conocidos, evitaremos que el usuario se confunda y que el uso del sistema sea intuitivo.					
Criterio de satisfacción	Serán reutilizados iconos estándar de otros sistemas que son conocidos por usuarios habituales de tecnología.					
Satisfacción del usuario	4	Insatisfacción del usuario	2	Prioridad Media		

Tabla 37 Tabla de Volére del requisito #06

#### 1.1.1.19 REQUISITOS DE ACCESIBILIDAD

Número de requisito	#07	Tipo			
Descripción	El sistema funcionará con una demora máxima de 5 segundos en ejecutar cualquier acción.				
Justificación	El sistema tiene que complacer al usuario en varios sentidos. Uno de ellos es en la rapidez para evitar que los usuarios dejen de usar el sistema.				
Criterio de satisfacción	En la etapa de testing, se medirá el tiempo que puede llegar a tardar el proceso de carga de las diversas páginas del sistema.				
Satisfacción del usuario	4	Insatisfacción del usuario	2	Prioridad	Media

Tabla 38 Tabla de Volére del requisito #07

Número de requisito	#08	Tipo			
Descripción	Los textos que se muestran en el sistema son completamente legibles				
Justificación	Los textos serán legibles para cualquier tipo de usuario independientemente de algún problema de visión que puedan tener, también se utilizarán fuentes de textos conocidas para el usuario.				
Criterio de satisfacción	Se realizará una prueba a los usuarios para ver su capacidad de lectura. Un 99% o más de los usuarios deberán quedar satisfechas con la lectura de los textos.				
Satisfacción del usuario	4	Insatisfacción del usuario	2	Prioridad	Media

Tabla 39 Tabla de Volére del requisito #08

#### 9.2.3. REQUISITOS DE DESEMPEÑO

##### 1.1.1.20 REQUISITOS DE CONFIABILIDAD Y DISPONIBILIDAD

Número de requisito	#09	Tipo			
Descripción	El sistema ha de estar disponible al menos el 99 % del tiempo.				
Justificación	El usuario debe poder acceder al sistema en cualquier momento. Esto aumentará su fiabilidad y confianza en el servicio.				
Criterio de satisfacción	Dado que se usará Amazon Web Services, se usa el compromiso del Acuerdo de nivel de servicios de Amazon Web Services que ofrece una disponibilidad del 99,95% en todas y cada una de las regiones de AWS.				
Satisfacción del usuario	5	Insatisfacción del usuario	2	Prioridad	Media

Tabla 40 Tabla de Volére del requisito #09

##### 1.1.1.21 REQUISITOS DE CAPACIDAD

Número de requisito	#10	Tipo			
Descripción	El sistema es escalable.				
Justificación	Dependiendo del número de usuarios que utilicen la aplicación, el sistema ha de poder soportar cualquier cambio o petición de los usuarios.				
Criterio de satisfacción	Dado que se usará Amazon Web Services, a medida que aumente el conjunto de datos, Amazon DynamoDB los distribuirá automáticamente entre los recursos de máquina suficientes para satisfacer sus necesidades de almacenamiento.				
Satisfacción del usuario	5	Insatisfacción del usuario	2	Prioridad	Alta

Tabla 41 Tabla de Volére del requisito #10

Número de requisito	#11	Tipo						
Descripción	El sistema debe poder dar respuesta de manera habitual cumpliendo todos los requisitos anteriores y posteriores con, como máximo, 5.000 usuarios simultáneos.							
Justificación	El sistema debe aguantar un mínimo considerablemente alto de usuarios simultáneos para que el usuario no pierda su confianza en él.							
Criterio de satisfacción	El sistema responderá correctamente y cumpliendo los requisitos de latencia con al menos 5.000 usuarios simultáneos.							
Satisfacción del usuario	5	Insatisfacción del usuario	2	Prioridad	Alta			

Tabla 42 Tabla de Volére del requisito #11

#### 1.1.1.22 REQUISITOS DE ESCALABILIDAD Y EXTENSIBILIDAD

Número de requisito	#12	Tipo						
Descripción	El sistema debe estar abierto a la ampliación y ser capaz de aumentar su potencia de procesamiento en todo momento, tanto optimizando el código como aumentando el número de procesadores.							
Justificación	El número de usuarios crecerá y posiblemente no se cumplirán los requisitos de capacidad, por tanto se deben ampliar con facilidad.							
Criterio de satisfacción	El diseño del software seguirá patrones de diseño específicos para la extensibilidad del sistema, pudiendo añadir funcionalidades extra sin modificar en un 5 % el trabajo realizado. AWS mantiene un servicio disponible en todo momento para la escalabilidad del hardware, de tal forma que se pueda añadir más memoria, más almacenamiento y más procesadores.							
Satisfacción del usuario	1	Insatisfacción del usuario	5	Prioridad	Alta			

Tabla 43 Tabla de Volére del requisito #12

#### 1.1.1.23 REQUISITOS DE LONGEVIDAD

Número de requisito	#13	Tipo						
Descripción	El sistema debe aumentar su plantilla de gestores para poder abastecer al creciente número de usuarios.							
Justificación	El sistema debe estar en todo momento actualizada con nuevos <i>facts</i> añadidos por el gestor o gestores.							
Criterio de satisfacción	Una de las premisas de este proyecto era ofrecer una serie de hechos denominados <i>facts</i> que permiten al usuario saber más sobre el miedo ambiente. El gestor es el encargado de añadir esta información por lo que si aumentan los usuarios más gestores serán contratados.							
Satisfacción del usuario	1	Insatisfacción del usuario	5	Prioridad	Alta			

Tabla 44 Tabla de Volére del requisito #13

## 9.2.4. REQUISITOS OPERACIONALES Y AMBIENTALES

### 1.1.1.24 REQUISITOS DE LANZAMIENTO

Número de requisito	#14	Tipo	
Descripción	El sistema será actualizado sin ocasionar inconvenientes.		
Justificación	El sistema se actualizará sin ocasionar molestias a los usuarios y sin ocasionar fallos al propio sistema.		
Criterio de satisfacción	Dado que nuestro sistema se ejecuta en el cloud, se harán los cambios de manera offline hasta corroborar el correcto funcionamiento y una vez confirmado, se subirá la nueva versión al cloud.		
Satisfacción del usuario	2	Insatisfacción del usuario	3
	Prioridad	Media	

Tabla 45 Tabla de Volére del requisito #14

## 9.2.5. REQUISITOS DE MANTENIMIENTO Y SOPORTE

### 1.1.1.25 REQUISITOS DE ADAPTABILIDAD

Número de requisito	#15	Tipo	
Descripción	El sistema se podrá utilizar de manera totalmente correcta tanto en sistemas operativos iOS como Android.		
Justificación	El usuario debe poder utilizar el sistema desde su sistema operativo móvil.		
Criterio de satisfacción	Se realizarán pruebas de usabilidad en los sistemas operativos iOS y Android antes del lanzamiento del sistema y en todos los casos deben superarse.		
Satisfacción del usuario	1	Insatisfacción del usuario	5
	Prioridad	Alta	

Tabla 46 Tabla de Volére del requisito #15

## 9.2.6. REQUISITOS DE SEGURIDAD

### 1.1.1.26 REQUISITOS DE INTEGRIDAD

Número de requisito	#16	Tipo	
Descripción	El sistema ha de realizar copias de seguridad de la base de datos de toda la información que ha de mantener persistente.		
Justificación	La integridad física de los datos es esencial para el éxito del sistema. No se puede permitir que por un fallo técnico se pierda esta información.		
Criterio de satisfacción	Amazon DynamoDB almacena tres copias geográficamente distribuidas de cada tabla para aumentar la disponibilidad y la durabilidad de los datos.		
Satisfacción del usuario	2	Insatisfacción del usuario	4
	Prioridad	Alta	

Tabla 47 Tabla de Volére del requisito #16

Número de requisito	#17	Tipo			
Descripción	El sistema ha de definir restricciones que impidan romper la integridad lógica de los datos, ni permitir que por cualquier tipo de error se pierden datos de los usuarios.				
Justificación	La integridad lógica de los datos es también esencial para el éxito del sistema. Debemos evitar que aparezcan datos en formatos incorrectos o corruptos.				
Criterio de satisfacción	Desde la propia capa de presentación pondremos todas las facilidades posibles para que se almacenen los datos de la manera que nosotros deseamos. Si, aun así, no se obtienen en el formato deseado, disponemos de métodos en las capas inferiores que los estandarizarán.				
Satisfacción del usuario	1	Insatisfacción del usuario	2	Prioridad	Alta

Tabla 48 Tabla de Volére del requisito #17

#### 1.1.1.27 REQUISITOS DE INMUNIDAD

Número de requisito	#18	Tipo			
Descripción	El sistema ha de estar protegido contra los ataques informáticos más habituales, como virus, spyware o código malicioso.				
Justificación	Los ataques informáticos ponen en riesgo la disponibilidad del sistema, además de la integridad de los mismos y su privacidad, con lo cual dejarían de cumplirse la mayoría de los requisitos anteriores y los siguientes. Además, no solo estaría en peligro el sistema, sino también el usuario, que podría perder confianza en el sistema.				
Criterio de satisfacción	El usuario estará protegido contra ataques maliciosos, como virus, spyware, troyanos y otros tipos de ataques informáticos habituales, debido al diseño robusto del sistema.				
Satisfacción del usuario	2	Insatisfacción del usuario	5	Prioridad	Alta

Tabla 49 Tabla de Volére del requisito #18

#### 9.2.7. REQUISITOS LEGALES

##### 1.1.1.28 REQUISITOS DE CUMPLIMIENTO

Número de requisito	#19	Tipo			
Descripción	Todos los datos de carácter personal han de ser tratados cumpliendo la LOPD ESPAÑOLA (Ley Orgánica de Protección de Datos 15/1999)				
Justificación	El sistema tiene que cumplir con la legislación nacional.				
Criterio de satisfacción	Se contratará a un abogado para llevar toda la documentación adecuada en regla y confirmar que el sistema cumple con las leyes vigentes.				
Satisfacción del usuario	2	Insatisfacción del usuario	5	Prioridad	Alta

Tabla 50 Tabla de Volére del requisito #19

## 10. DISEÑO

Después de haber explicado todos los requisitos tanto funcionales como no funcionales, en esta sección pasaremos a presentar el desarrollo y diseño del software de manera específica.

El diseño consiste en definir los controladores, módulos y datos de manera que se satisfagan parte de los requisitos que se han definido en la sección anterior.

Se ha adoptado una arquitectura en tres capas (presentación, dominio y datos) como se puede observar en la figura 8 que aparece a continuación. Se decidió utilizar esta arquitectura para gestionar el desarrollo del sistema por niveles. Si una capa se ve afectada por algún fallo o inconveniente, será más simple cambiar dicha capa sin generar daños en los demás módulos. Así obtenemos un sistema que escala más. En la capa lógica se encuentran los cinco servicios implementados. Estos servicios están explicados más adelante individualmente en la sección 10.1.2 de la página 51.

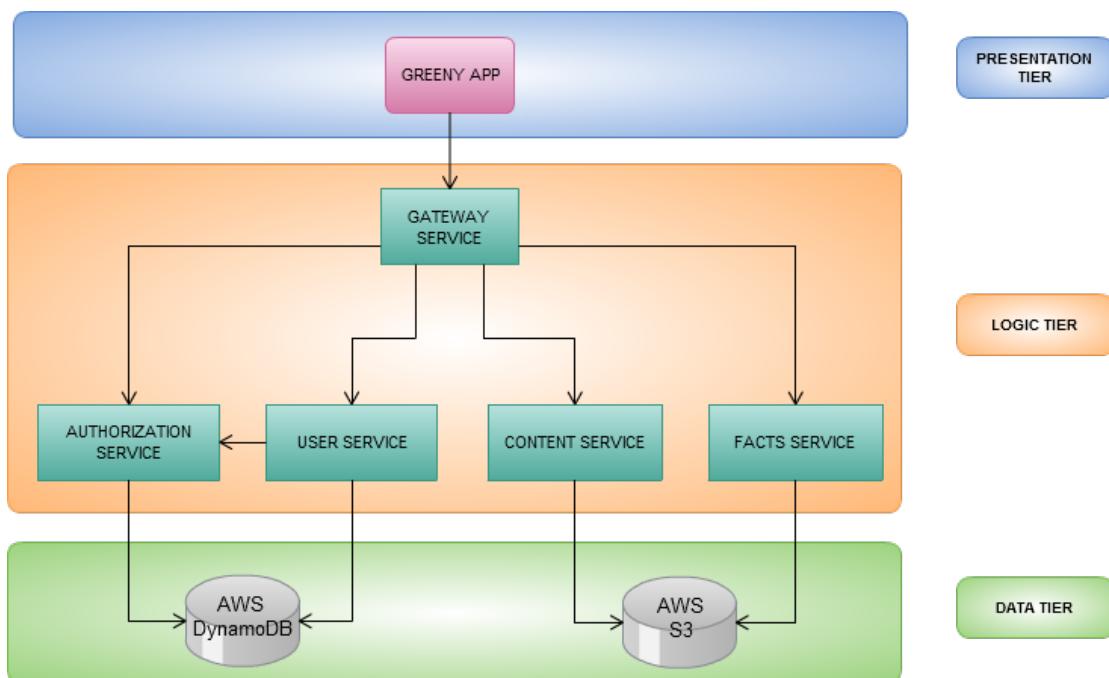


Figura 8 Diseño interno de capas del sistema

Para desarrollar este sistema se ha utilizado el patrón MVC (modelo-vista-controlador). Esta es una arquitectura que tiene el objetivo de separarla lógica de negocio y los datos de una aplicación de la interfaz de usuario.

- El modelo, se encuentra en la capa de datos y en la capa de dominio. Este gestiona los accesos a la información con la que el sistema trabaja. Permite realizar consultas y actualizaciones en la base de datos del sistema.
- La vista es la que presenta la información que gestiona el controlador y la muestra de manera útil y adecuada para el usuario.
- El controlador responde a las peticiones que el usuario le hace y las envía a la parte del modelo que las gestiona. Es el paso intermedio entre la vista y el modelo.

## 10.1. DISEÑO Y ARQUITECTURA DEL BACK-END

En esta sección se entrará en detalle de las decisiones que se han tomado a la hora de diseñar la parte del back-end de este sistema.

Las APIs del sistema (microservicios) han sido diseñadas siguiendo la mayoría de los principios REST y cada una de ellas está explicada en el apartado 10.1.2 de esta sección donde entramos en detalle de cada uno de los servicios del sistema.

Los principios de REST que se han seguido para llevar a cabo este sistema han sido varios. Se ha hecho uso correcto de las URIs, es decir que, los nombres de las URIs no implican ninguna acción y por lo tanto no contienen ningún verbo, cada URI es única e identifica un único recurso y mantienen una jerarquía lógica. Para comprobar que se hace buen uso de las URIs, se pueden observar en los diagramas de secuencia que van apareciendo en este apartado por cada servicio que nombre toman las URIs en las peticiones.

Otro principio REST que cumplen las APIs de este sistema es que hace un uso correcto de HTTP. Para manipular los distintos recursos, HTTP ofrece unos métodos que también se han usado para este sistema. En concreto se han usado los métodos GET, POST y PUT.

En el trozo de código que aparece a continuación podemos observar como mediante HTTP llevamos a cabo una petición POST para validar la contraseña del usuario al iniciar sesión. En concreto este código pertenece al controlador *Validate Password* del *Gateway Service* de nuestro sistema.

```
import falcon, requests, json

class ValidatePassword(object):
    def on_post(self, req, resp, uid):
        r = requests.post(
            "http://127.0.0.1:8002/users/" + str(uid) + "/validate_password",
            data=json.dumps(req.context['body']),
            headers={"Content-Type": "application/json",
                     "Accept": "application/json"})
        body = json.loads(r.content)
        req.context['result'] = body
        field_name = "HTTP_" + str(r.status_code)
        resp.status = getattr(falcon, field_name)
```

### 10.1.1. AMAZON WEB SERVICES

Una de las tecnologías que se han usado para gestionar todo lo relacionado con bases de datos ha sido Amazon Web Services que ofrece servicios cloud computing<sup>5</sup> de manera que las empresas, y en este caso el autor del proyecto, se ahorran dinero en infraestructuras y pagan por lo que consumen. AWS alquila almacenamiento de datos por gigabyte y potencia informática por hora.

De todos los servicios que ofrece AWS, para este sistema se han utilizado tres. Cada uno de estos servicios son explicados en detalle en el apartado del microservicio donde son utilizados:

- Amazon S3: Amazon Simple Storage System es una API sencilla que permite guardar y recuperar archivos. Se utiliza para guardar ficheros en un disco, externo e ilimitado [19].
- Amazon EC2: Amazon Elastic Compute Cloud es un servicio que ofrece servidores en los centros de datos de Amazon y que en este sistema se ha utilizado para hospedar las APIs y poder hacer uso de ellas en cualquier momento.

El compromiso del Acuerdo de nivel de servicios de Amazon EC2 es ofrecer una disponibilidad del 99,95% en todas y cada una de las regiones de Amazon EC2 [20]. Con este alto nivel de disponibilidad que ofrece EC2 se satisface el requisito #09 (Ver Sección 1.1.1.23 Página 44).

El tener nuestras APIs en una instancia de EC2 permite actualizarlas en el momento que el administrador del sistema quiera. El administrador puede llevar a cabo nuevas funcionalidades de manera offline, someter el sistema a pruebas y una vez esté todo listo, hacer la actualización en EC2. De esta manera cumplimos con el requisito no funcional #14 (Ver Sección 1.1.1.23 Página 45).

- Amazon DynamoDB: Amazon DynamoDB es un servicio de base de datos NoSQL que ofrece un desempeño muy fácil de escalar, predecible y rápido. Evita a sus usuarios, y por lo tanto al autor del proyecto, las cargas administrativas que supone escalar bases de datos ya que AWS se ocupa de ello.

Además, Al crear una tabla, DynamoDB realiza de forma automática las particiones de los datos entre un número suficiente de servidores como para cubrir la capacidad de solicitud que necesita. Es decir que puede escalar perfectamente una única tabla en cientos de servidores de manera que se cumplen los requisitos #10 (Ver Sección 1.1.1.23 Página 44) y #12 (Ver Sección 1.1.1.23 Página 45).

Amazon DynamoDB almacena tres copias geográficamente distribuidas de cada tabla para aumentar la disponibilidad y la durabilidad de los datos de manera que la seguridad de los datos especificada en el requisito #16 (Ver Sección 1.1.1.23 Página 45) queda cubierta.

---

<sup>5</sup> Consiste en la posibilidad de ofrecer servicios a través de Internet. Busca tener todos nuestros archivos e información en Internet sin preocuparse por poseer capacidad suficiente para almacenar información en nuestro ordenador. [27]

### 10.1.2. MICROSERVICES

Este tipo de arquitectura utilizada para llevar a cabo la parte del back-end de este sistema ha sido explicada en el apartado 2.2.2 de este documento. Cabe recordar que una de las cualidades de los microservicios es que cada uno suele encapsular funcionalidades simples, las cuales se pueden escalar o reducir, implementar y administrar de forma independiente. Esto permite una más fácil extensibilidad del sistema y la capacidad de añadir funcionalidades extra sin necesidad de modificar parte del trabajo realizado cumpliendo con el requisito #12 (Ver Sección 1.1.1.23 Página 45).

El back-end de nuestro sistema está formado por 5 servicios:

- Gateway Service: Este servicio recibe todas las requests del Cliente y las redirige al servicio adecuado.
- Facts Service: Servicio encargado de gestionar los *facts*. Los *facts* son hechos breves relacionados con el medio ambiente que son mostrados en la aplicación para concienciar al usuario sobre la problemática medioambiental.
- Content Service: Servicio encargado de gestionar el contenido estático del sistema. Este contenido es el relacionado con las secciones y acciones del sistema. Como sección entendemos cada uno de los 4 ámbitos en los que el usuario puede registrar acciones. Estas secciones son comida, agua, transporte y temperatura. Cada sección tiene cuatro acciones que son los actos cotidianos que el usuario puede llevar a cabo:

		Acción			
Sección	Comida	Comer carne	No comer carne	Comer pescado	No comer pescado
	Agua	Ducha corta (< 5 minutos)	Ducha larga (> 5 minutos)	Darse un baño	Lavar con agua fría
	Transporte	Bicicleta	Coche	Transporte público	Avión
	Temperatura	Encender la calefacción	Encender el aire acondicionado	Ponerse un sweater	Abrir la ventana

- Authentication Service: Servicio encargado de gestionar todo lo relativo a la autenticación de los usuarios.
- User Service: Servicio encargado de gestionar toda la información del usuario relativa a las acciones que ha registrado en el sistema.

Cada uno de estos servicios es explicado a continuación entrando en detalle en las decisiones que se han tomado para implementarlos.

#### 1.1.1.29 GATEWAY SERVICE

Se ha implementado una API Gateway como único punto de entrada a las peticiones del cliente. Las peticiones llegan a este servicio que las enruta al servicio adecuado. De esta manera si por ejemplo el sistema tiene que validar la contraseña de un usuario, la petición le llegará al *Gateway Service* y éste se la pasa al *Authentication Service* que es el servicio que trata esta acción.

En el siguiente diagrama de la figura 9 podemos observar un ejemplo de este sistema que sigue esta secuencia. La petición le llega al *Gateway Service*(GWS) y éste se la pasa al *Authentication Service*(AS):

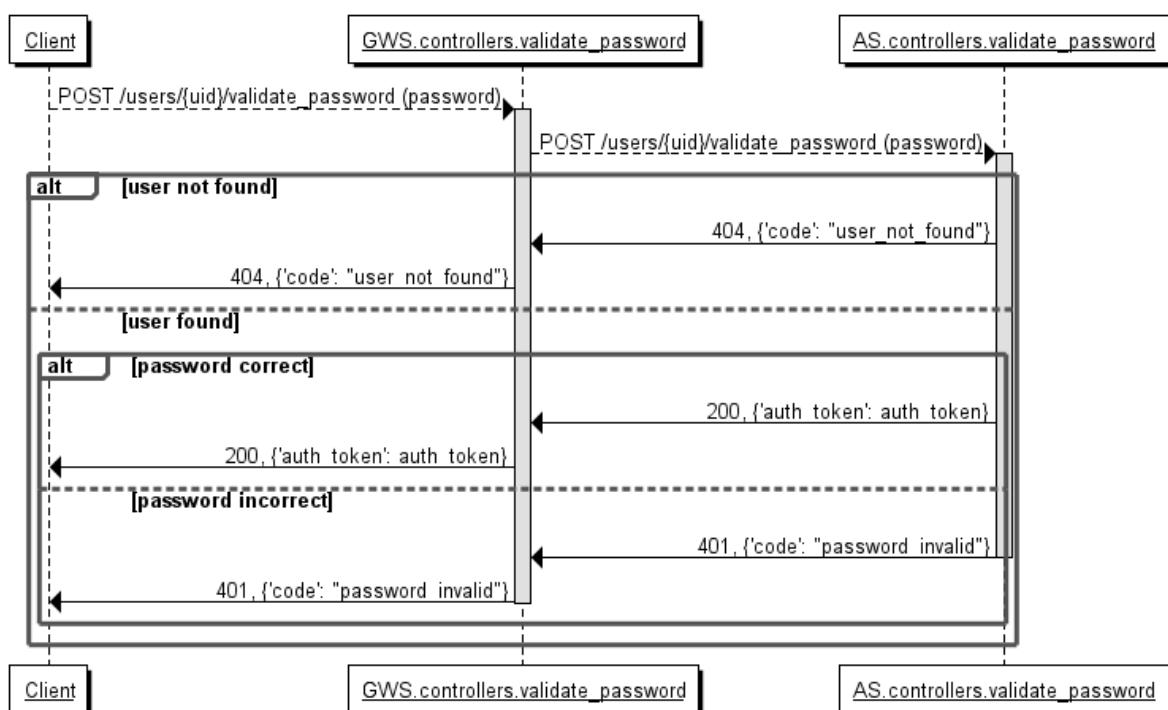


Figura 9 Diagrama de secuencia "validate password"

#### 1.1.1.29.1 Authentication token

Para este sistema se ha implementado un sistema de seguridad que permite autorizar al cliente para poder realizar una petición. Es el caso por ejemplo de la petición que obtiene el historial de acciones del usuario. Para evitar que cualquiera pueda acceder a esta información, a cada usuario se le asigna un *authentication token* al registrarse que forma parte de la cabecera de la petición. De esta manera, antes de proceder a reenviar a *User Service* la petición que obtiene el historial de acciones, se valida el *authentication token* del usuario que quiere obtener el historial.

A continuación podemos observar los diagramas que muestran la secuencia explicada:



Figura 10 Diagrama de secuencia GWS "Get action history"

#### 1.1.1.30 FACTS SERVICE

Este servicio es el encargado de gestionar los *facts*. Como *fact* entendemos un hecho relacionado con el medio ambiente y que puede ser relevante para el usuario de cara a concienciarlo a llevar una vida más amigable con el medio ambiente. Cada día se le mostrará al usuario un nuevo *fact*.

Este servicio hace uso de Amazon S3 (Amazon Simple Storage Service). Para ello se ha creado un bucket llamado greeny-facts. Los buckets son los contenedores que ofrece este servicio para almacenar datos. Este servicio es recomendado para guardar contenido estático y debido a que *Facts Service* trata contenido que sabemos que no cambiará, se ha decidido hacer uso de él. Dentro de este bucket tenemos un objeto llamado facts.json donde están todos los facts que le pueden aparecer al usuario.

El fichero facts.json tiene esta forma:

```
{  
    "facts": [  
        {  
            "id": 1,  
            "display": "Earth's temperatures in 2015 were the hottest ever recorded (source: NASA). Why does this matter? Because a change of even 1 degree Fahrenheit - which may sound small - can upset the delicate balance of ecosystems, and affect plants and animals that inhabit them.."   
        },  
        {  
            "id": 2,  
            "display": "Experts predict that one-fourth of Earth's species will be headed for extinction by 2050 if the warming trend continues at its current rate."   
        },  
        {  
            "id": 3,  
            "display": "As ocean waters warm, they expand, causing sea-levels to rise. Melting glaciers compound the problem by dumping even more fresh water into the oceans."   
        },  
        {  
            "id": 4,  
            "display": "Scientists estimate that globally glaciers are losing 92 cubic kilometres of ice per year. That's as much water used by Canada's homes, farms and factories over six years."   
        }  
    ]  
}
```

Figura 11 Aspecto del fichero facts.json

Para leer y obtener el contenido de este fichero facts.json se ha usado la librería de Python Boto 3 que permite escribir software que haga uso de servicios de Amazon como, en este caso, S3. Para ello, y como se puede observar en el código que aparece a continuación, se ha creado un cliente de S3 pasándole la región y las credenciales de AWS. Una vez configurado, ya podemos obtener el contenido deseado especificando el bucket al que queremos acceder y el fichero del que queremos obtener el contenido.

```
client = boto3.client(  
    's3', region_name='eu-west-1',  
    aws_access_key_id=os.environ['ACCESS_KEY_ID'],  
    aws_secret_access_key=os.environ['SECRET_ACCESS_KEY'])  
  
class Fact:  
    def __init__(self, **kwargs):  
        self.fact_id = kwargs.get("fact_id", None)  
        self.display = kwargs.get("display", None)  
  
    def get_fact_id(self):  
        return self.fact_id  
  
    def get_display(self):  
        return self.display  
  
    def read_facts():  
        response = client.get_object(  
            Bucket='greeny-facts',  
            Key='facts.json')  
        return response['Body'].read()
```

En el diagrama de clases de la figura 12 se puede observar en la capa de negocio todos los controladores del *Gateway Service* y del *Facts Service*, que solo tiene un controlador.

Para esta funcionalidad que gestiona la obtención de facts, solo aparece una comunicación entre servicios, la del *Gateway Service* con el *Facts Service*.

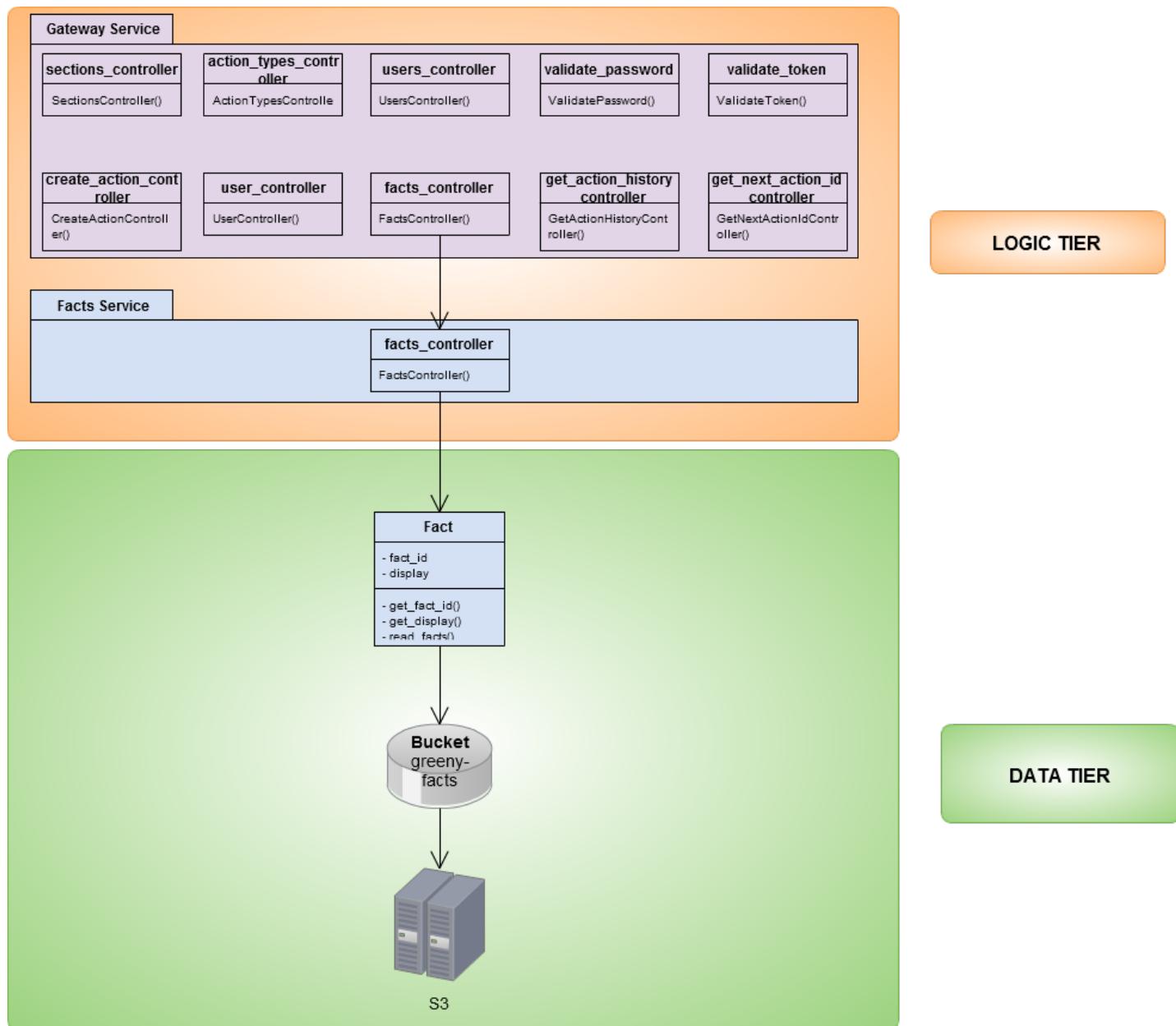


Figura 12 Diagrama de clases "Facts Service"

Como vemos en el diagrama que aparece a continuación en la figura 13, al *Gateway Service* le llega una petición desde el cliente que reenvía al *Facts Service*. El controlador de este servicio, *facts\_controller*, trata la petición leyendo de Amazon S3 y retornando uno de los facts al *Gateway service* o por el contrario devolviendo un mensaje de error.

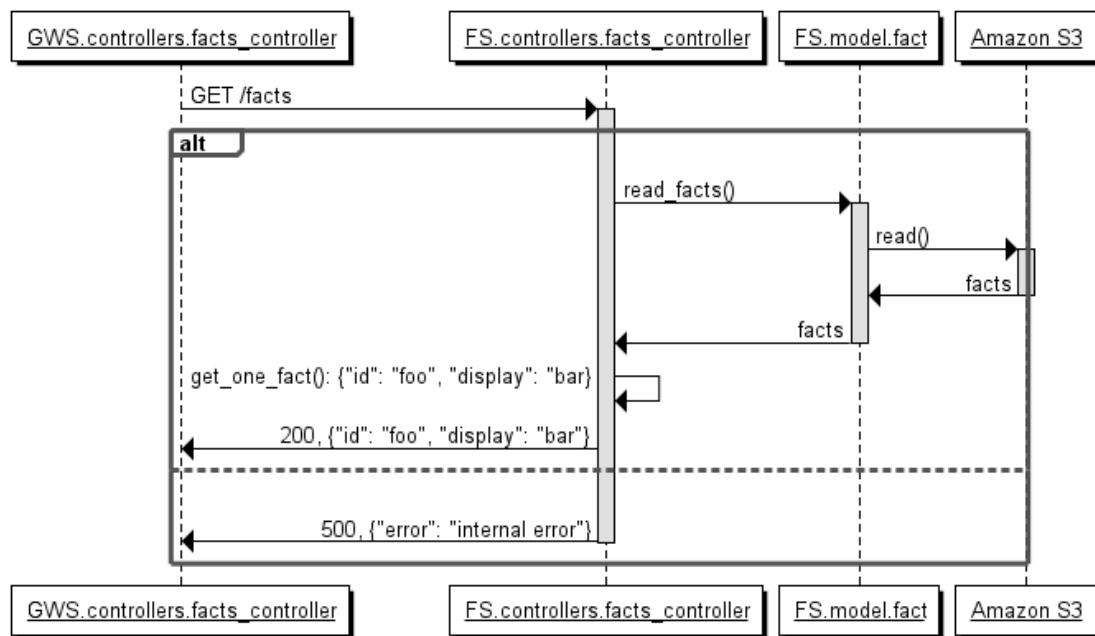


Figura 13 Diagrama de secuencia "Facts Controller"

#### 1.1.1.31 CONTENT SERVICE

Este servicio es el encargado de gestionar las secciones y las acciones del sistema. Como secciones entendemos los 4 grupos en los que se distribuyen las acciones: comida, agua, transporte y temperatura. Como acciones entendemos las 16 (4 por sección) actividades que un usuario puede llevar a cabo que están relacionadas con el medio ambiente.

De igual manera que el *Facts Service*, este servicio utiliza Amazon S3 (Amazon Simple Storage Service). Para ello se ha creado un bucket llamado *greeny-content*. Los buckets son los contenedores que ofrece este servicio para almacenar datos. Este servicio de AWS es recomendado para guardar contenido estático y debido a que *Content Service* trata contenido que sabemos que no cambiará, se ha decidido hacer uso de él. Dentro de este bucket tenemos cinco objetos con el siguiente aspecto:

```
{
  "sections": [
    {
      "id": "food",
      "display": "info food"
    },
    {
      "id": "water",
      "display": "info water"
    },
    {
      "id": "transportation",
      "display": "info transportation"
    },
    {
      "id": "temperature",
      "display": "info temperature"
    }
  ]
}
```

Figura 17 Aspecto del fichero sections.json

```
{
  "action_types": [
    {
      "id": "short_shower",
      "display": "Short shower",
      "points": 5
    },
    {
      "id": "long_shower",
      "display": "Long shower",
      "points": -5
    },
    {
      "id": "bath",
      "display": "Take a bath",
      "points": -15
    },
    {
      "id": "cold_water",
      "display": "Wash with cold water",
      "points": 3
    }
  ]
}
```

Figura 14 Aspecto del fichero water\_action\_types.json

```
{
  "action_types": [
    {
      "id": "meat",
      "display": "Meat",
      "points": -5
    },
    {
      "id": "no_meat",
      "display": "No meat",
      "points": 5
    },
    {
      "id": "fish",
      "display": "Fish",
      "points": -3
    },
    {
      "id": "no_fish",
      "display": "No fish",
      "points": 3
    }
  ]
}
```

Figura 16 Aspecto del fichero food\_action\_types.json

```
{
  "action_types": [
    {
      "id": "bike",
      "display": "Bike",
      "points": 10
    },
    {
      "id": "car",
      "display": "Car",
      "points": -5
    },
    {
      "id": "public_transport",
      "display": "Public transport",
      "points": 5
    },
    {
      "id": "plane",
      "display": "Plane",
      "points": -15
    }
  ]
}
```

Figura 15 Aspecto del fichero transportation\_action\_types.json

```
{  
    "action_types": [  
        {  
            "id": "heating",  
            "display": "Turn on heating",  
            "points": -5  
        },  
        {  
            "id": "cooling",  
            "display": "Turn off Cooling",  
            "points": -5  
        },  
        {  
            "id": "sweater",  
            "display": "Use sweater",  
            "points": 5  
        },  
        {  
            "id": "window",  
            "display": "Open window",  
            "points": 5  
        }  
    ]  
}
```

Figura 18 Aspecto del fichero temperature\_action\_types.json

Para leer y obtener el contenido de estos ficheros se ha usado la librería de Python Boto3 que, por el mismo motivo que en *Facts Service*, permite escribir software que haga uso de servicios de Amazon como, en este caso, S3. Para ello, y como se puede observar en el código que aparece a continuación, se ha creado un cliente de S3 pasándole la región y las credenciales de AWS. Una vez configurado, ya podemos obtener el contenido deseado especificando el *bucket* al que queremos acceder y el fichero del que queremos obtener el contenido. En el caso que aparece a continuación se obtienen las secciones accediendo al *sections.json*.

```
client = boto3.client(  
    's3',  
    region_name='eu-west-1',  
    aws_access_key_id=os.environ['ACCESS_KEY_ID'],  
    aws_secret_access_key=os.environ['SECRET_ACCESS_KEY'])  
  
def read_sections():  
    response = client.get_object(  
        Bucket='greeny-content',  
        Key='sections.json'  
    )  
    return response['Body'].read()
```

En el diagrama de clases de la figura 19 se puede observar en la capa de negocio todos los controladores del *Gateway Service* y del *Content Service*, que tiene dos controladores.

Para esta funcionalidad que gestiona la obtención del contenido de la aplicación, aparecen las dos comunicaciones entre servicios, la del *Gateway Service* con el *Content Service*.

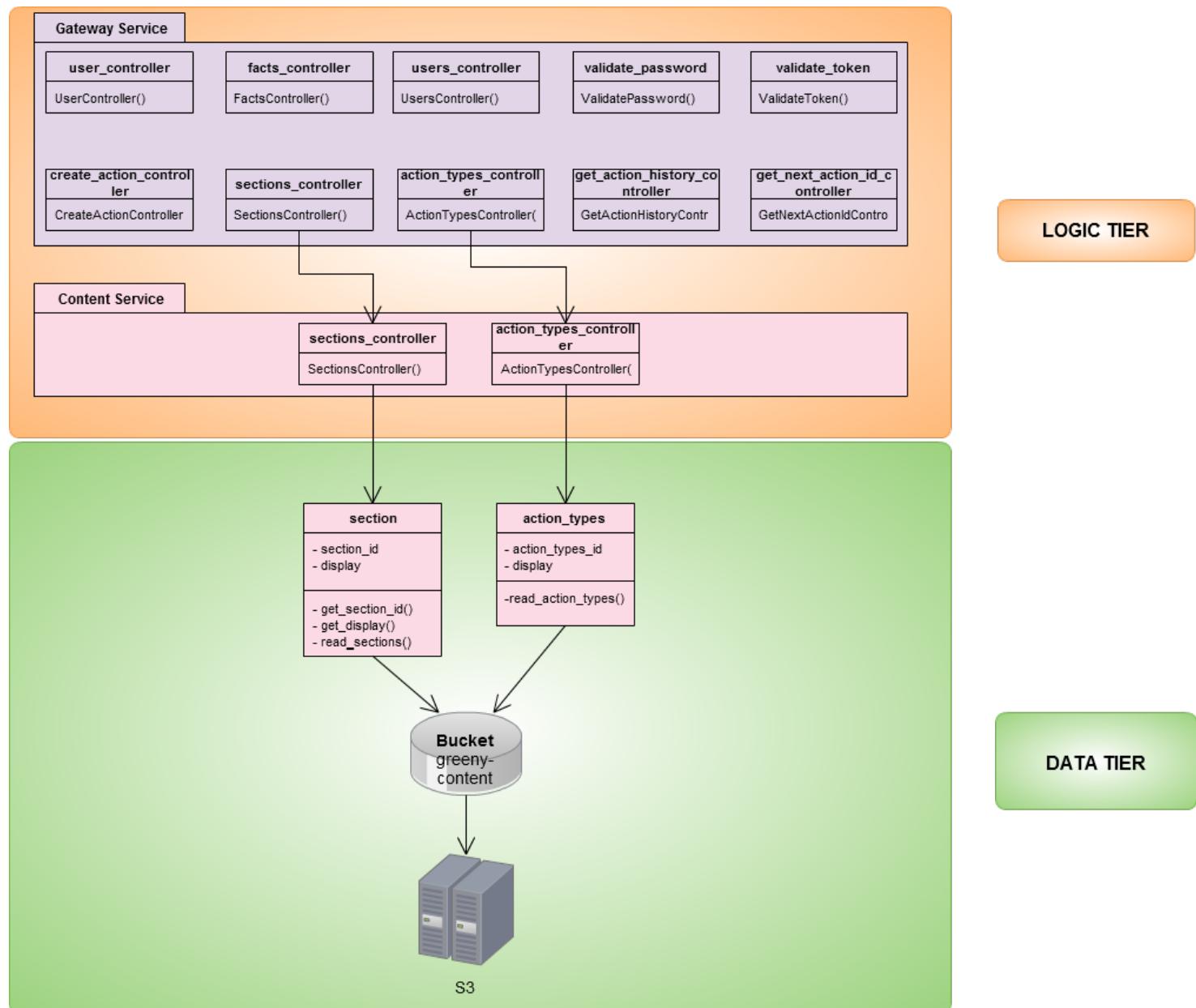


Figura 19 Diagrama de clases "Content Service"

En el siguiente diagrama de la figura 20 observamos cómo se desarrolla la petición que obtiene las secciones. Desde el *Gateway Service* se reenvía la petición al *Content Service* que lee las secciones de Amazon S3. En caso de éxito, se devuelve un status code 200 y en el body de la response el contenido del fichero *sections.json*. En caso contrario se devuelve un status code 500 y un mensaje de error.

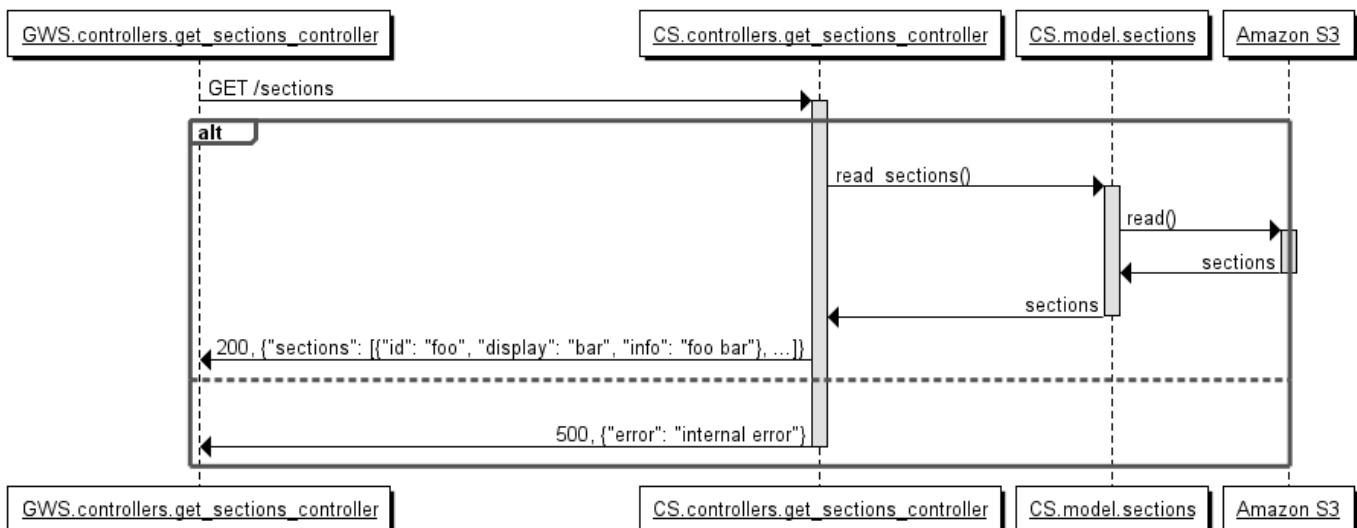


Figura 20 Diagrama de secuencia "get sections controller"

En la figura 21 se puede observar la secuencia que se lleva a cabo para obtener las acciones de las secciones que se le muestran al usuario. Para ello, el *Gateway Service* reenvía la petición que le ha llegado al *Content Service*. Este lee las acciones y las devuelve en el body de la response junto con un status code 200. En caso que ocurra algún error se devolverá el mensaje de error con un status code 500.

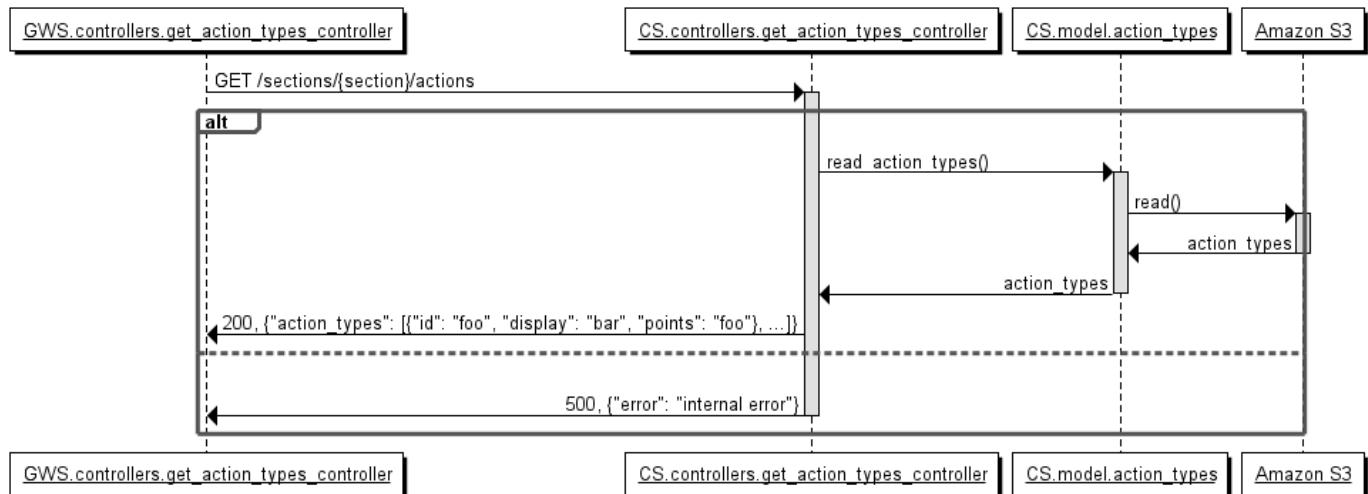


Figura 21 Diagrama de secuencia "get action types controller"

### 1.1.1.32 AUTHENTICATION SERVICE

Este servicio es el encargado de gestionar todo lo relativo al sistema de autenticación del usuario, es decir:

- o Validar la contraseña de un usuario que quiere iniciar sesión.
- o Validar el *authentication token* de un usuario que quiere llevar a cabo alguna acción que requiere permiso.
- o Guardar la contraseña de un usuario que quiere registrarse en el sistema.
- o Crear el *authentication token* de un usuario que quiere registrarse en el sistema.

El *Authentication Service* hace uso de otro servicio que ofrece *Amazon Web Services*. En concreto de *Amazon DynamoDB*. Este servicio ha ahorrado al autor del proyecto todo el trabajo tanto administrativo como tareas de instalación, configuración, aprovisionamiento y revisión del software. Todo esto lo hace *Amazon* por el cliente.

Como se puede observar en el diagrama de clases de la figura 22, en la capa de datos tenemos una tabla llamada *as\_users* que guarda y actualiza los siguientes datos:

Tabla <i>as_users</i>	
Atributo	Descripción
Uid	<i>Primary key</i> Identificador UUID del usuario
Password	Contraseña del usuario
Auth_token	Authentication token del usuario

Tabla 51 Tabla *as\_users*

En la figura 22 también podemos observar el diagrama de clases del *Authentication Service*. Podemos observar la comunicación que aparece entre tres servicios distintos. La funcionalidad de crear un nuevo usuario es competencia del *User Service* pero, el registrar un usuario en el sistema contiene acciones de las que se tiene que encargar el *Authentication Service*. Estas acciones son crear el *authentication token* con el que el usuario se autentica y guardar la contraseña que el usuario guarda en el sistema.

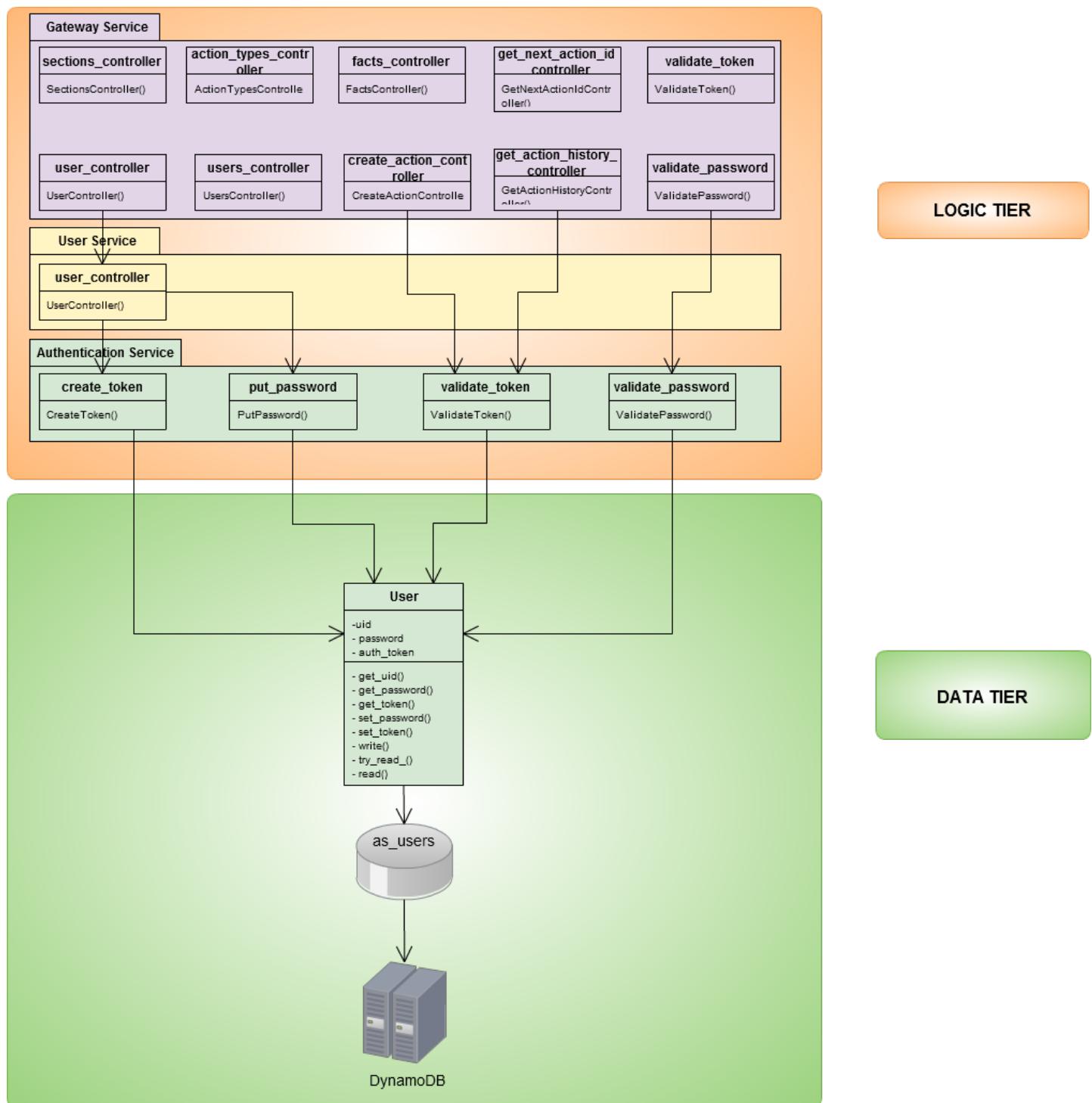


Figura 22 Diagrama de clases "Authentication Service"

En la figura 23 podemos observar la secuencia que se sigue para crear el *authentication token* de un nuevo usuario en el sistema. Al *Authentication Service* le llega una petición de *User Service* del controlador que registra o inicia sesión a un usuario en el sistema. Para ello, obtenemos leyendo de DynamoDB el usuario con *uid* pasado en el query string de la petición. Recordemos que ese *uid* corresponde al identificador existente si el usuario ya estaba registrado, o al nuevo identificador en caso que el usuario se quiera registrar.

Si el usuario estaba registrado, se genera un nuevo *authentication token* y se actualiza escribiendo en DynamoDB este valor. Si el usuario es nuevo y no estaba registrado, se genera un nuevo *authentication token* y se escribe en la DynamoDB.

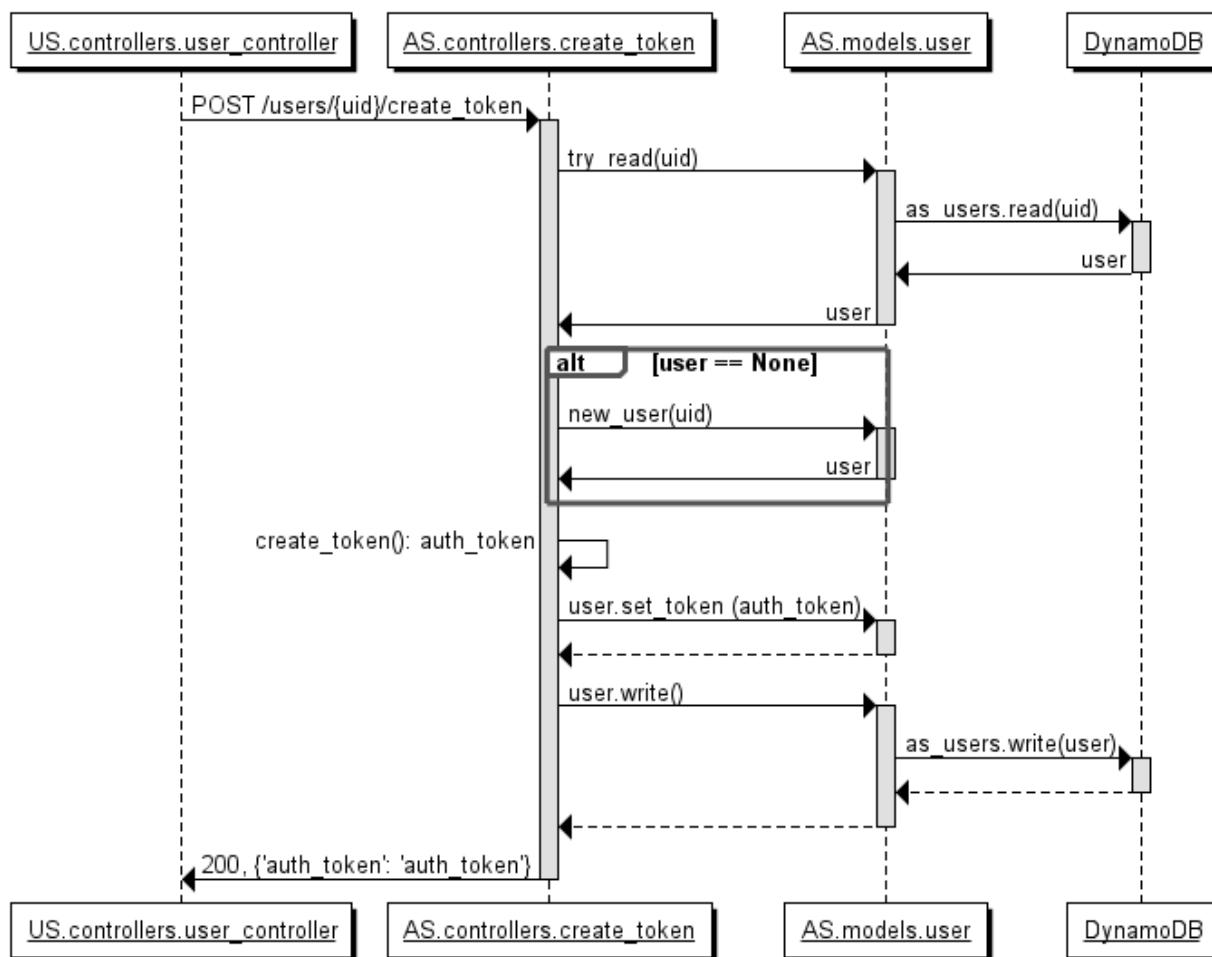


Figura 23 Diagrama de secuencia "create token"

En la figura 24 observamos la secuencia que existe para guardar la contraseña de un usuario que se quiere registrar.

Esta petición llega, igual que en *create token*, de *User Service*. Esta funcionalidad se lleva a cabo en este servicio ya que recordemos que es el encargado de gestionar todo lo relacionado con la autenticación de los usuarios. El *User Service* delega este trabajo al *Authentication Service* corroborando la base de las arquitecturas microservices que pretenden separar en servicios las diferentes lógicas de negocio del sistema.

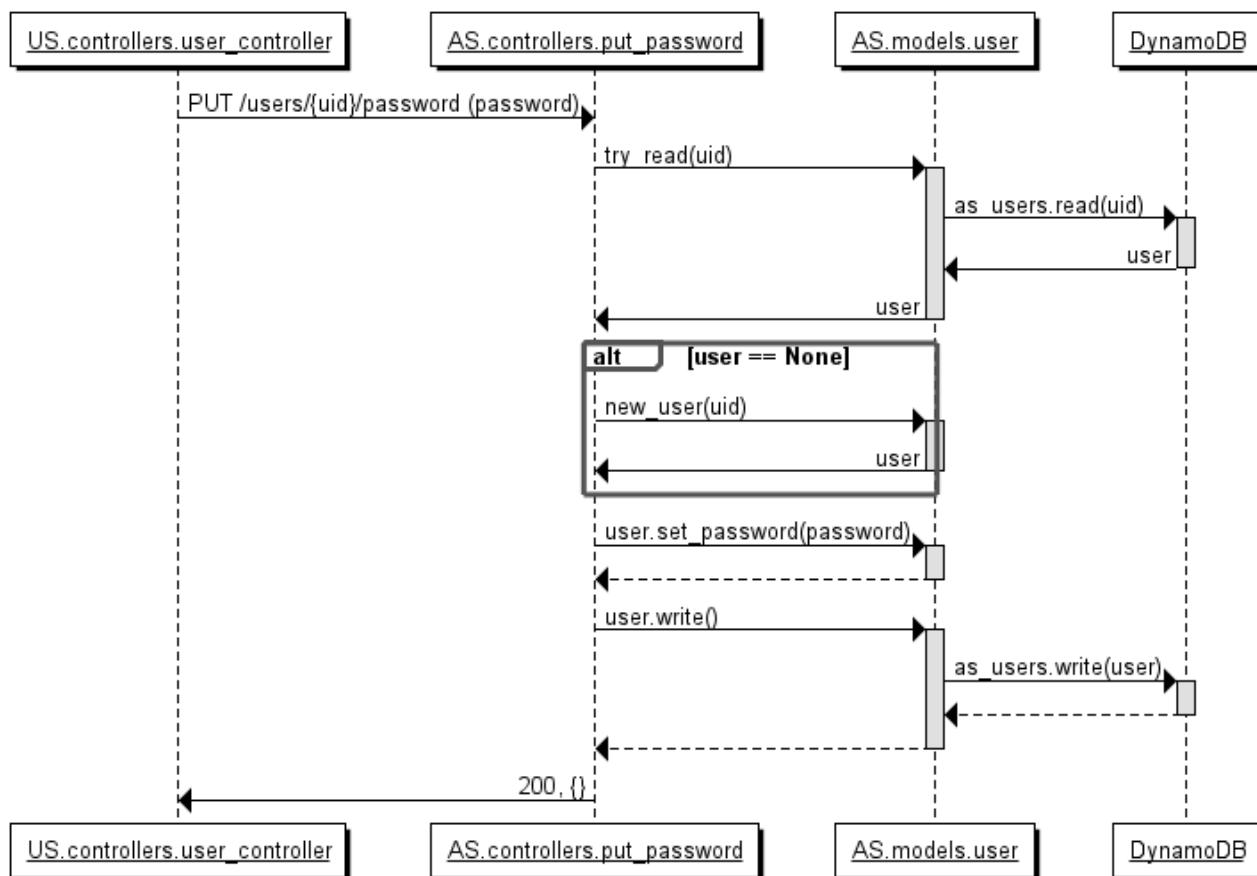


Figura 24 Diagrama de secuencia "put password"

En caso de que el usuario quiera entrar en el sistema deberá introducir su contraseña. En la figura 25 observamos la secuencia que valida la contraseña que ha introducido un usuario. Primero se comprueba que el usuario con *uid* pasado en la URL exista para devolver un mensaje de error en caso necesario. Si existe, leemos de DynamoDB la contraseña guardada para ese *uid* y comprobamos si son iguales.

- En caso afirmativo se devuelve el *authentication token* que permitirá al usuario llevar a cabo nuevas acciones y un *status code* de 200.
- En caso negativo, se devuelve un mensaje de error y un *status code* de 401.

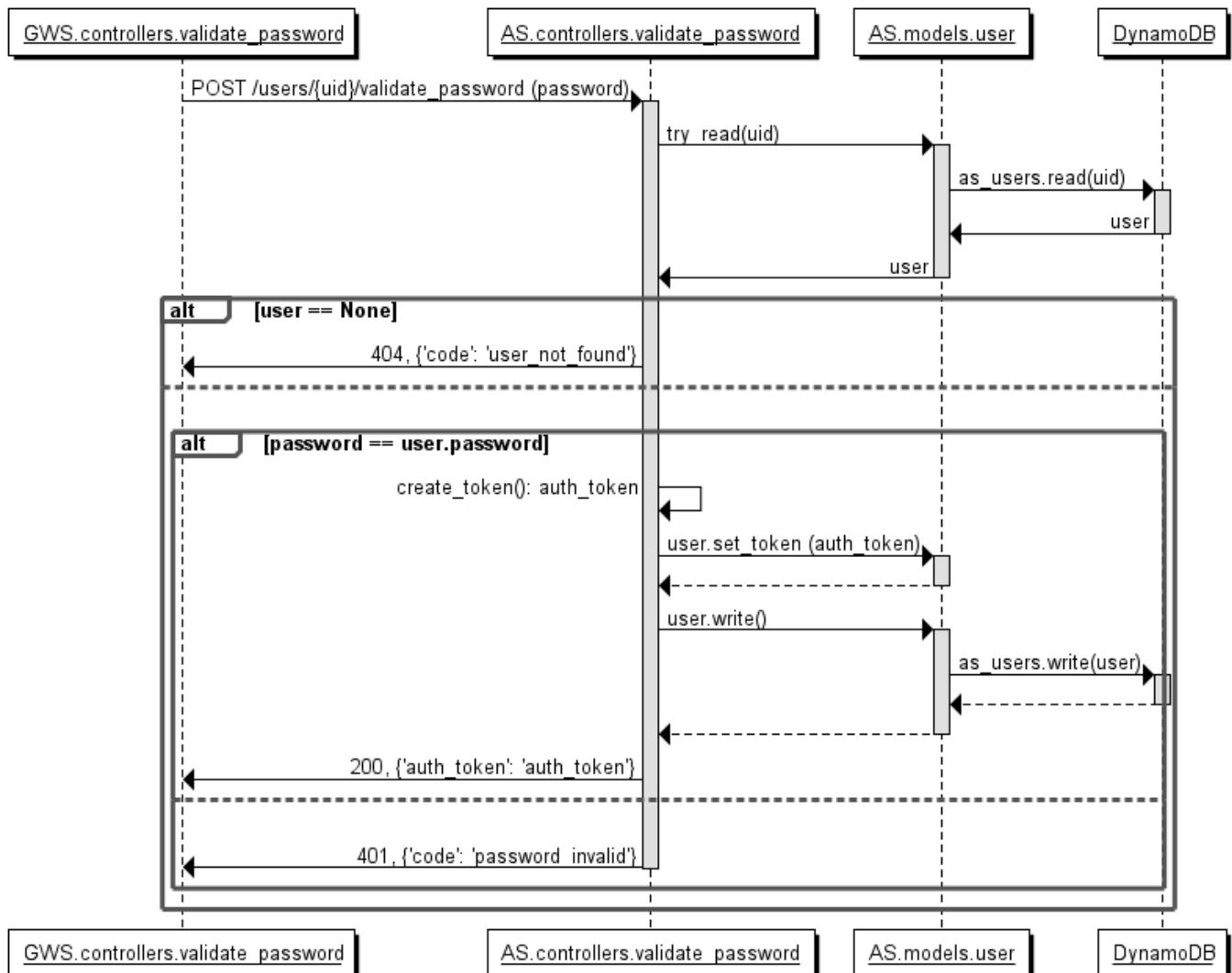


Figura 25 Diagrama de secuencia "validate password"

En la figura 26 podemos observar el diagrama de secuencia de la funcionalidad que valida el *authentication token* de un usuario.

De esta manera se dará permiso o no al usuario para llevar a cabo una acción en el sistema. Leemos de DynamoDB el usuario con *uid* pasado por la URL de la petición y con el *authentication token* de este usuario y con el que hemos recibido en el cuerpo de la petición, hacemos una comparación y si son iguales permitimos al usuario llevar a cabo la acción y si no son iguales denegamos el acceso.

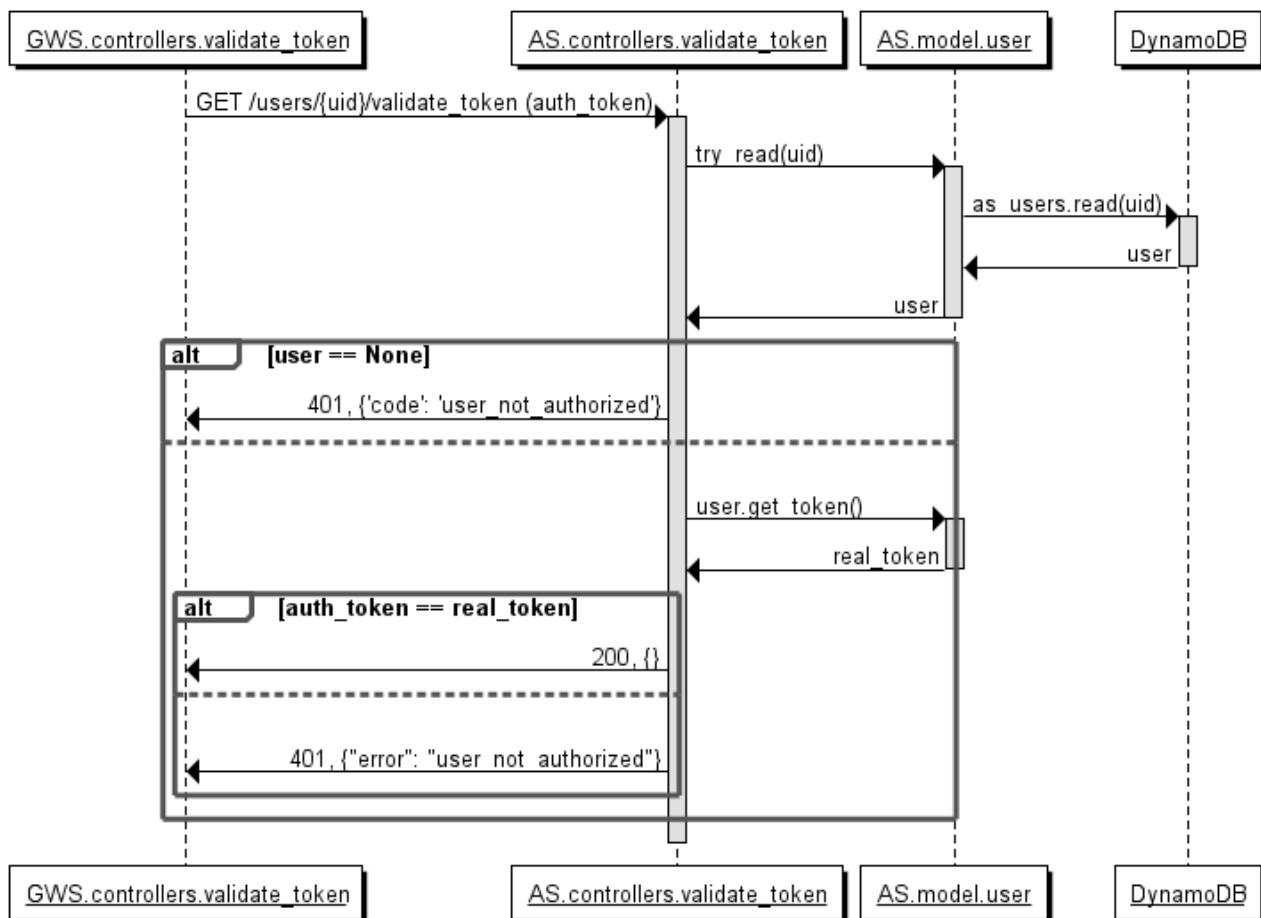


Figura 26 Diagrama de secuencia "validate token"

#### 1.1.1.33 USER SERVICE

Este servicio es el encargado de gestionar todo lo que concierne a las acciones que lleva a cabo el usuario.

El *User Service* hace uso de *Amazon DynamoDB*, igual que *Authentication Service*. Este servicio ha ahorrado al autor todo el trabajo tanto administrativo como tareas de instalación, configuración, aprovisionamiento y revisión del software. Todo esto lo hace *Amazon* por el cliente.

A continuación se explica el aspecto que tienen las tablas que forma la capa de datos de este servicio. En total se hace uso de 4 tablas

La tabla 52 contiene los atributos y la descripción de cada uno de los atributos que guarda esta tabla *us\_users* en *DynamoDB*. Se utiliza para obtener el email de un usuario a partir del identificador de éste.

Tabla us_users	
Atributo	Descripción
Uid	<i>Primary key</i> Identificador UUID del usuario
Email	Correo electrónico del usuario

Tabla 52 Tabla us\_users

La tabla 53 *us\_email\_to\_uid* contiene los atributos y la descripción de cada uno de ellos. Esta tabla de *DynamoDB* se utiliza para obtener el identificador de un usuario a partir de su email. Esta tabla es usada para validar el email de un usuario.

Tabla us_email_to_uid	
Atributo	Descripción
Email	<i>Primary key</i> Correo electrónico del usuario
uid	Identificador UUID del usuario

Tabla 53 Tabla us\_email\_to\_uid

La tabla 54 *us\_uid\_to\_next\_action\_id* guarda el *action\_id* que se asignará a la siguiente acción que el usuario quiere realizar. Se ha llevado a cabo de esta manera ya que así se evitan problemas como crear dos acciones con el mismo *action\_id*.

Tabla us_uid_to_next_action_id	
Atributo	Descripción
uid	<i>Primary key</i> Identificador UUID del usuario
Next_action_id	Identificador UUID de la próxima acción del usuario

Tabla 54 Tabla us\_uid\_to\_next\_action\_id

La tabla 55 muestra el aspecto de *us\_actions* que se utiliza para guardar las acciones que el usuario ha registrado en el sistema. Cada acción debe ser única. No puede haber una acción que pertenezca a más de un usuario, ni un usuario puede tener dos acciones iguales. Es por eso que esta tabla se ha configurado con una primary key compuesta por el *uid* y el *action\_id*.

Tabla us_actions	
Atributo	Descripción
uid	<i>Primary key</i> Identificador UUID del usuario
Action_id	<i>Sort key</i> Identificador UUID de la acción
Action_Type	Tipo de acción
Datetime	Fecha de registro de la acción
Score	Puntuación de la acción
Section	Sección de la acción. Este valor puede tomar los siguientes valores: Food, Water, Transportation, Temperature

Tabla 55 Tabla us\_actions

En la figura 27, podemos observar el diagrama de clases del *User Service*. Aparece la comunicación entre dos servicios distintos, el *Gateway Service* y el *User Service*. En la figura también se pueden ver las tablas que han sido explicadas anteriormente.

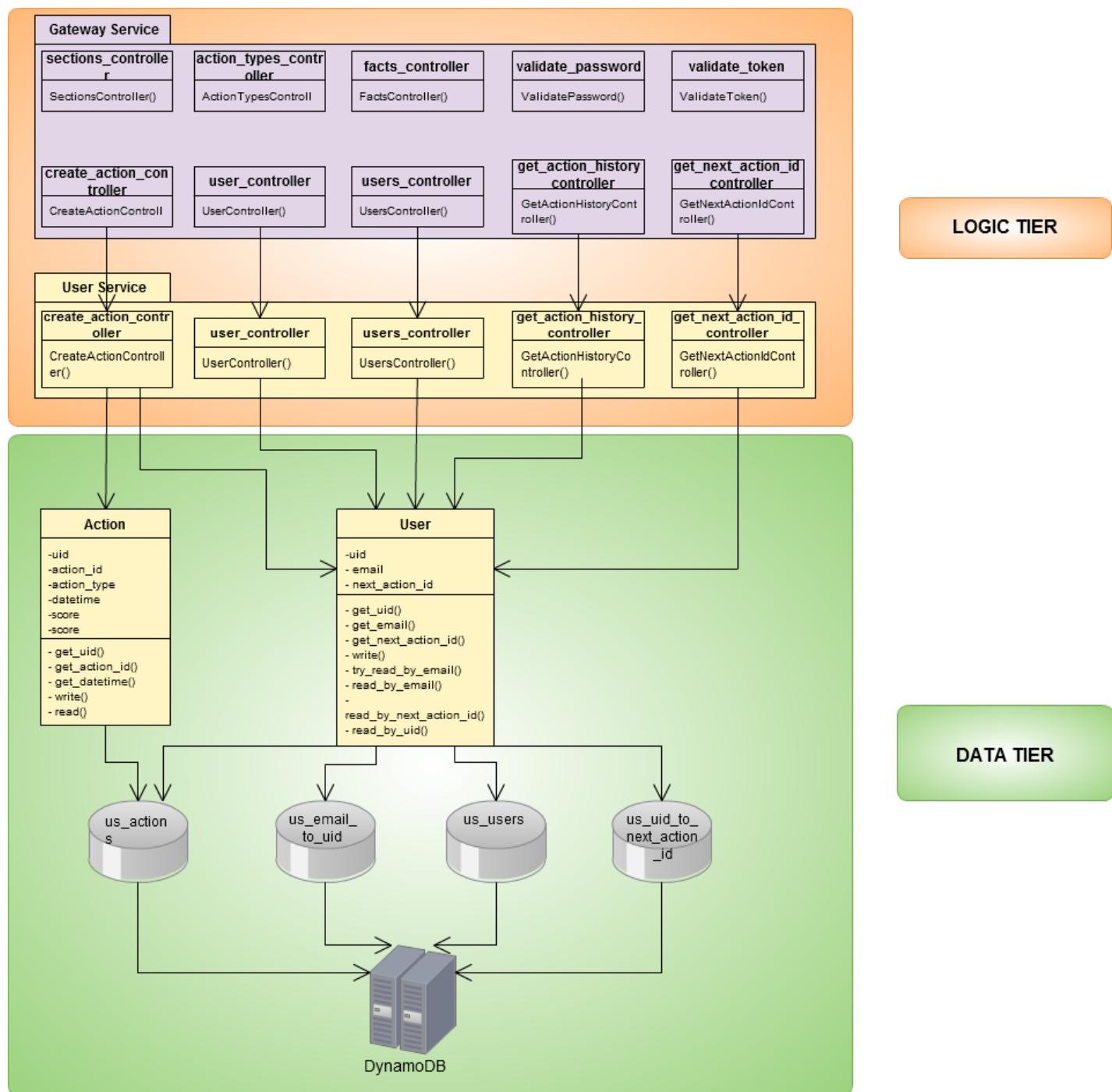


Figura 27 Diagrama de clases "User Service"

En la figura 28 podemos observar la secuencia que se sigue para crear una acción que un usuario ha llevado a cabo. Una vez el *User Service* recibe la petición, genera la fecha actual y escribe la acción en la tabla *us\_actions* de *DynamoDB*. Una vez guardada la acción, se genera el *next\_action\_id* que corresponde al *action\_id* de la próxima acción que el usuario lleve a cabo.

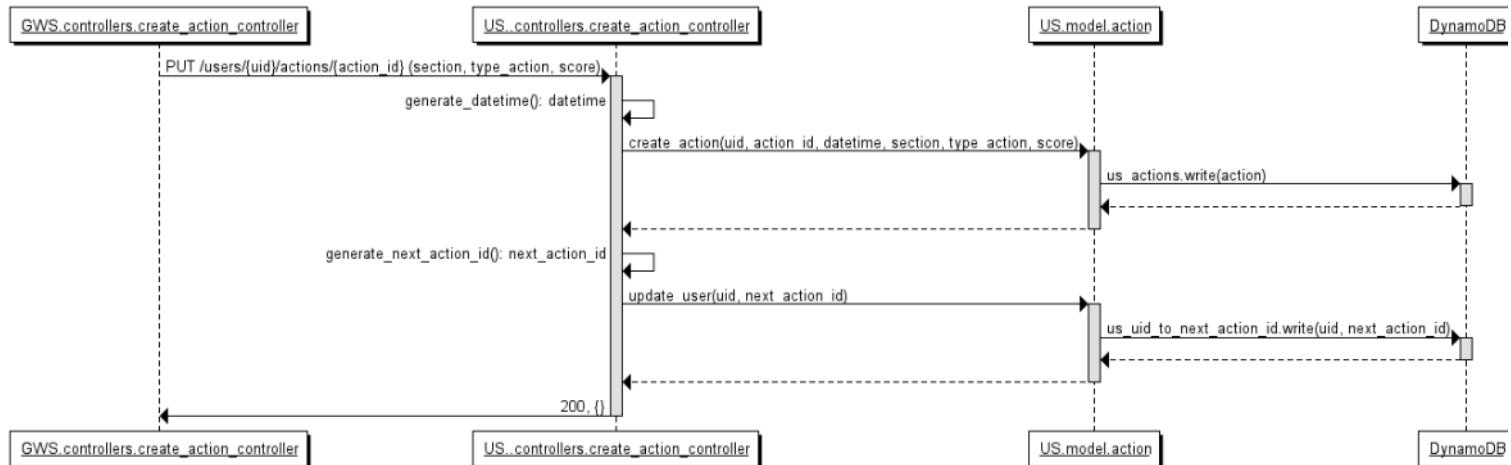


Figura 28 Diagrama de secuencia "create action controller"

A continuación, en la figura 29, podemos observar la secuencia que sigue la acción que obtiene el historial de acciones de un usuario. El *Gateway Service* reenvía la petición al *User Service* que a partir del identificador *uid* lee la historia de usuario de *DynamoDB*.

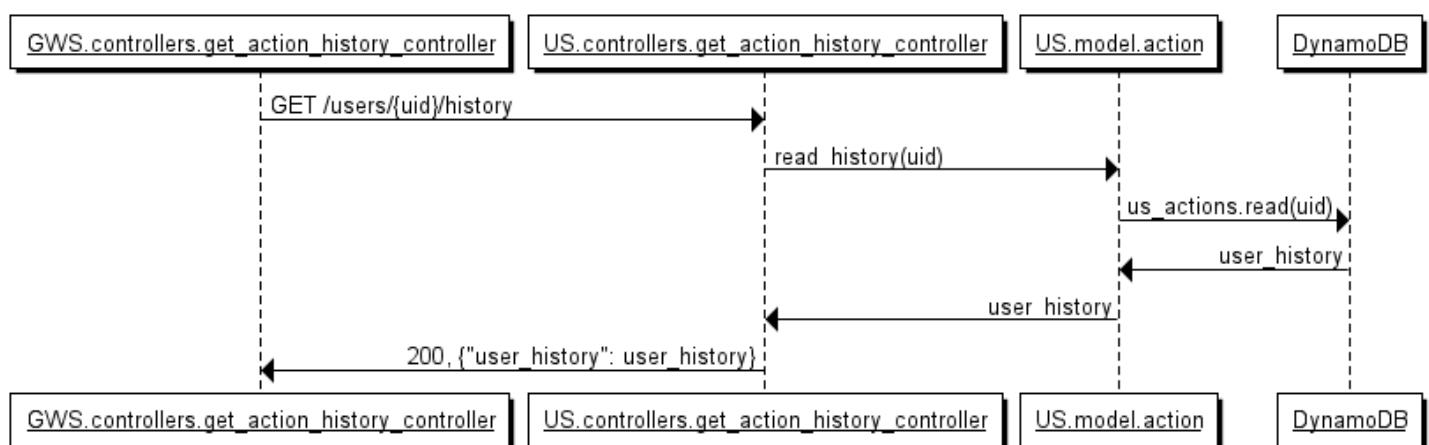


Figura 29 Diagrama de secuencia "get action history controller"

En la figura 30 se puede observar el diagrama de secuencia que obtiene el identificador de la nueva acción que el usuario lleve a cabo. Para ello, se hace una lectura en *DynamoDB* de la tabla *us\_uid\_to\_next\_action\_id* del usuario con identificador *uid* y de este objeto cogemos el *next\_action\_id* y lo metemos en el *body* de la response. En caso de éxito el status code será un 200.

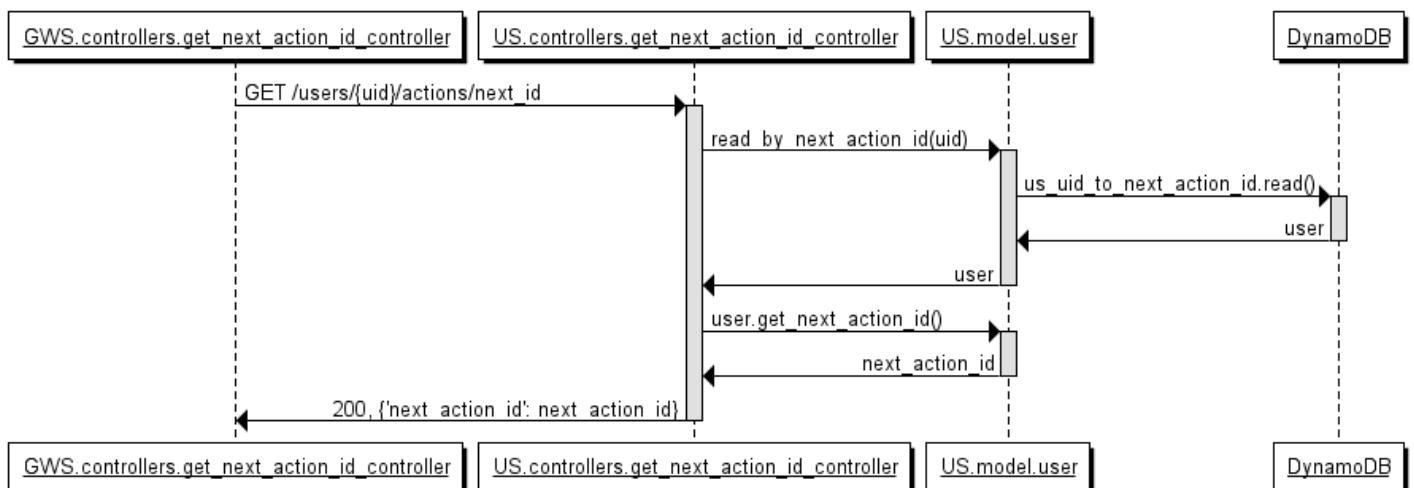


Figura 30 Diagrama de secuencia "get next action id controller"

A continuación, en la figura 31 encontramos el diagrama de secuencia del *User Controller*. Este controlador se encarga de gestionar todo lo que supone registrar a un usuario en el sistema. Inicialmente le llega la petición al *User Service* desde el *Gateway Service* y genera un *next\_action\_id* aleatorio que corresponderá al identificador de la primera acción que el usuario guarde en el sistema. Escribiremos en la tabla *as\_users* de *DynamoDB* el *uid*, *email* y *next\_action\_id* del usuario. A continuación se realiza una petición a *Authentication Service* que creará y escribirá en *DynamoDB* el *authentication token* que dará permiso al usuario para realizar futuras peticiones.

Por último se realiza otra petición a *Authentication Service* que guarda la contraseña en *DynamoDB* que el usuario ha introducido.

La respuesta de esta petición contiene un body con el *authentication token* del usuario recién creado y un status code de 200.

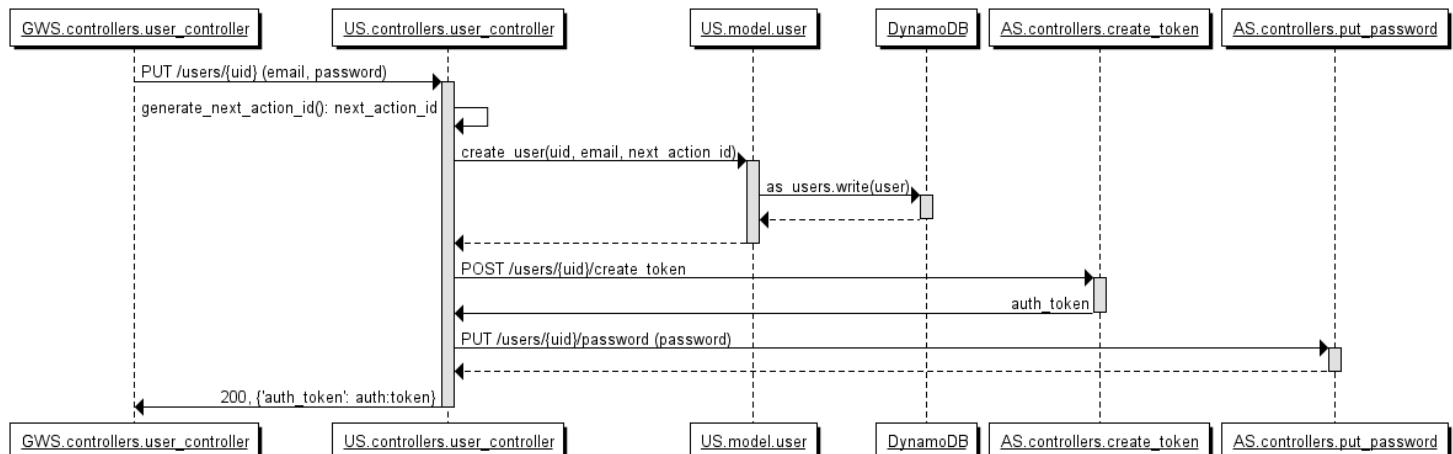


Figura 31 Diagrama de secuencia "user controller"

En la figura 32 se puede observar la secuencia del *Users Controller* que devuelve una respuesta dependiendo de si el usuario está registrado en el sistema o no. Inicialmente se lee de la tabla *us\_email\_to\_uid* de *DynamoDB* el usuario con el email pasado en el body de la request. Si esta lectura resulta fallida y no se encuentra ningún usuario con ese email significará que el usuario es nuevo en el sistema. Por eso se generará un nuevo identificador de manera aleatoria y se meterá en el body de la response junto a un array vacío que se utilizará como condición para llevar a cabo futuras funcionalidades que dependan de este resultado.

En caso que la lectura haya sido exitosa y se retorne un usuario existente, se obtiene el identificador de este usuario y se mete en un array que formará parte del body de la response.

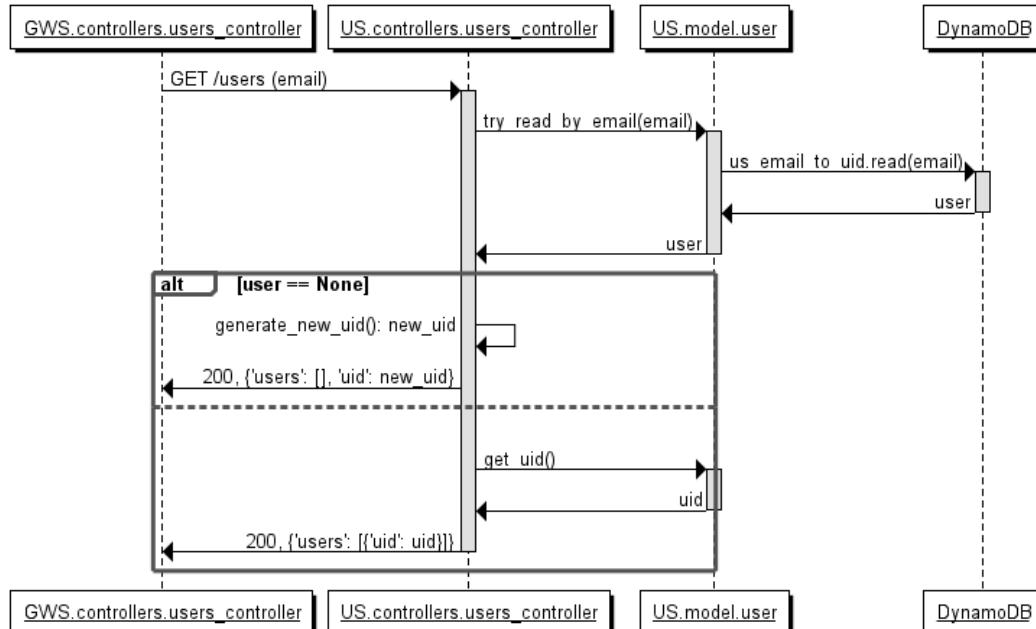


Figura 32 Diagrama de secuencia "users controller"

### 10.1.3. FALCON

Falcon ha sido el framework de Python utilizado para crear las APIs en la nube. Las APIs en la nube tienen que ser capaces de gestionar peticiones de manera rápida y hacer un uso eficiente del hardware. Falcon es entre los WSGI frameworks disponibles, el más rápido. Es capaz de procesar peticiones de manera más veloz que cualquier otro web framework de Python.

Por ejemplo, en el *Content Service*, que es el encargado de leer contenido de la base de datos y meterlo en la response, el código del controlador que obtiene las secciones de la aplicación quedaría de la siguiente manera:

```
class GetSectionsController(object):
    def on_get(self, req, resp):
        sections = model.section.read_sections()
        resp.body = sections
        resp.status = falcon.HTTP_200

from controllers.get_sections_controller import GetSectionsController
def add_routes(api):
    api.add_route('/sections', GetSectionsController())
```

En esta parte de código observamos como todas las peticiones GET a la url */sections* deben leer de la base de datos y meter el contenido en el body de la response además de asignar al status el valor que pertoca.

## 10.2. DISEÑO Y ARQUITECTURA DEL FRONT-END

En este apartado se entrará en detalle de todas las decisiones que se han tomado para llevar a cabo la parte del Front-End del sistema. Se hablará de la interfaz de usuario, de las herramientas utilizadas y los componentes que han formado parte de esta parte del sistema.

### 10.2.1. DISEÑO DE LA INTERFAZ DE USUARIO

A continuación aparecen las pantallas con las que el usuario puede interaccionar haciendo referencia a los requisitos funcionales y no funcionales del sistema. Para cada funcionalidad aparece la pantalla que corresponde al aspecto que tendría en iOS y en algunos casos el aspecto que tendría en Android, dado que el aspecto es igual en gran parte de las funcionalidades en ambas plataformas.

En las ilustraciones 1 y 2 se puede observar el aspecto que tiene la pantalla de validar email que cumple el requisito funcional de la tabla 19 (Ver sección 9.1.2 Página 36) en iOS y en Android.

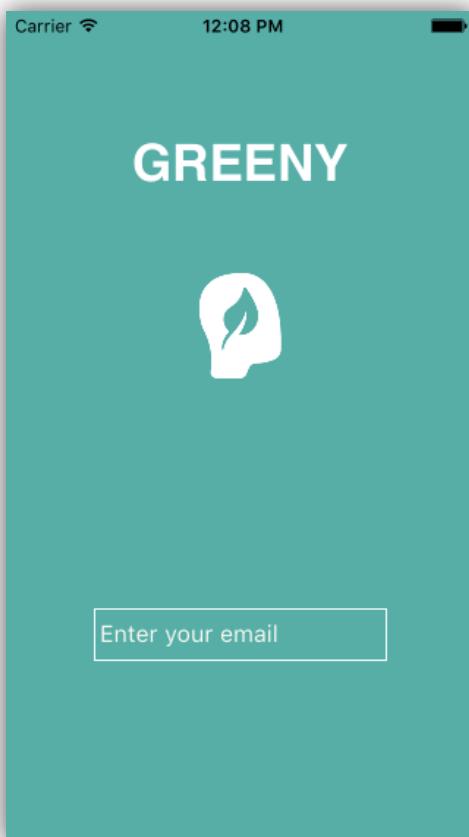


Ilustración 1 Validar email iOS

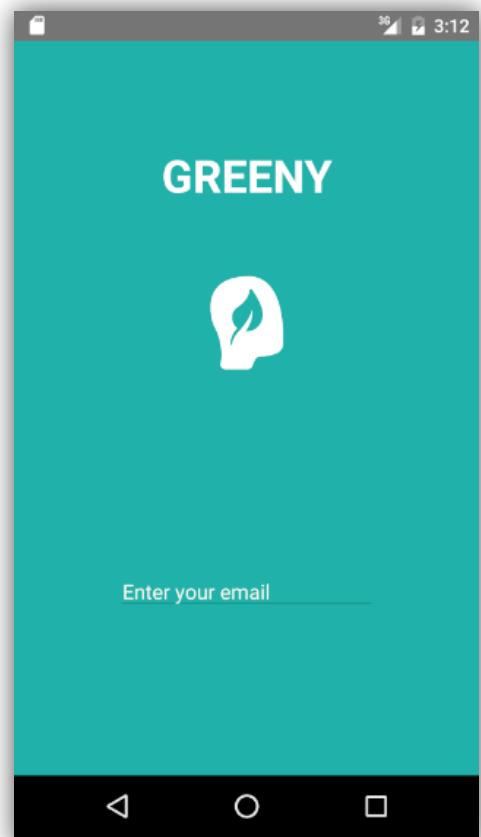


Ilustración 2 Validar email Android

Los componentes y librerías que se han usado que ofrece React Native han sido:

- Text: Para mostrar el nombre de la aplicación: Greeny.
- TouchableOpacity: Para hacer táctil la sección donde el usuario tiene que escribir su email. De esta manera al hacer click en este espacio, aparece el teclado que permite al usuario escribir.
- TextInput: Permite al usuario escribir texto mediante el teclado. Se le da un valor por defecto: Enter your email.
- Image: Componente utilizado para mostrar el logo de la aplicación. El ícono utilizado ha sido hecho por Freepik [21].
- Animated: Librería utilizada para animar la secuencia que pasa de esta pantalla a la siguiente (Login o Sign up). La animación se aplica al TextInput de manera que este contenedor se mueva hacia la derecha hasta que desaparezca de la pantalla. A la vez que aparecen los nuevos contenedores por la izquierda colocándose como aparece en las ilustraciones 2 y 3. Esto se obtiene mediante translaciones y variando la posición de los componentes respecto al ancho de la pantalla.
- Dimensions: Método que permite obtener las dimensiones de la pantalla que permiten hacer la animación del TextInput.
- Easing: Función que permite dar un efecto concreto a nuestra animación. Para esta se ha definido como elastic.

En la ilustración 3 se muestra la pantalla de Registro del sistema que cumple el requisito funcional de la tabla 21 (Ver sección 9.1.2 Página 37). Los componentes y librerías que se han usado que ofrece React Native han sido:

- Text: Para mostrar el nombre de la aplicación: Greeny. También para darle el texto Sign up al botón de registro.
- TouchableOpacity: utilizado para crear el botón táctil que el usuario debe usar para registrarse una vez ha introducido la contraseña. Esto se consigue gracias a la propiedad `onPress()` que ofrece este componente.
- TextInput: Permite al usuario escribir texto mediante el teclado. Se le da un valor por defecto: Enter your new password.
- Image: Componente utilizado para mostrar el logo de la aplicación. El icono utilizado ha sido hecho por Freepik [21].

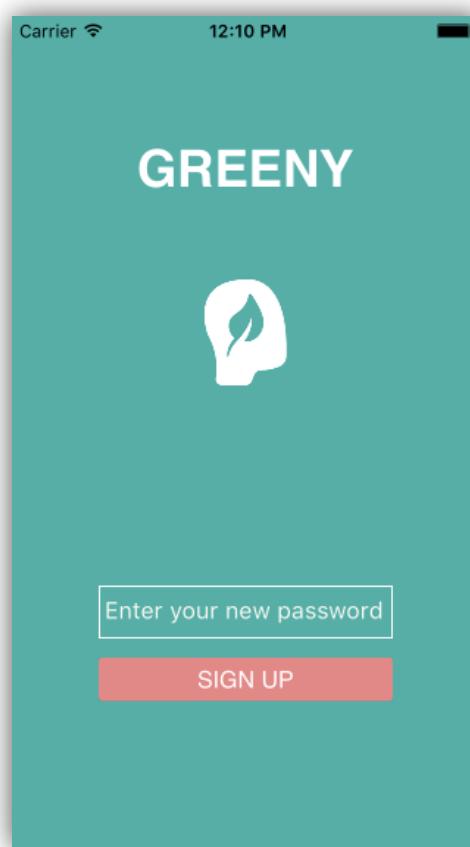


Ilustración 3 Registrarse iOS

En las ilustraciones 4 y 5 se muestra la pantalla de inicio de sesión del sistema que cumple el requisito funcional de la tabla 20 (Ver sección 9.1.2 Página 37). Los componentes y librerías que se han usado que ofrece React Native han sido:

- Text: Para mostrar el nombre de la aplicación: Greeny. También para darle el texto Login al botón de registro.
- TouchableOpacity: utilizado para crear el botón táctil que el usuario debe usar para registrarse una vez ha introducido la contraseña. Esto se consigue gracias a la propiedad `onPress()` que ofrece este componente.
- TextInput: Permite al usuario escribir texto mediante el teclado. Se le da un valor por defecto: Enter your new password.
- Image: Componente utilizado para mostrar el logo de la aplicación. El icono utilizado ha sido hecho por Freepik [21].

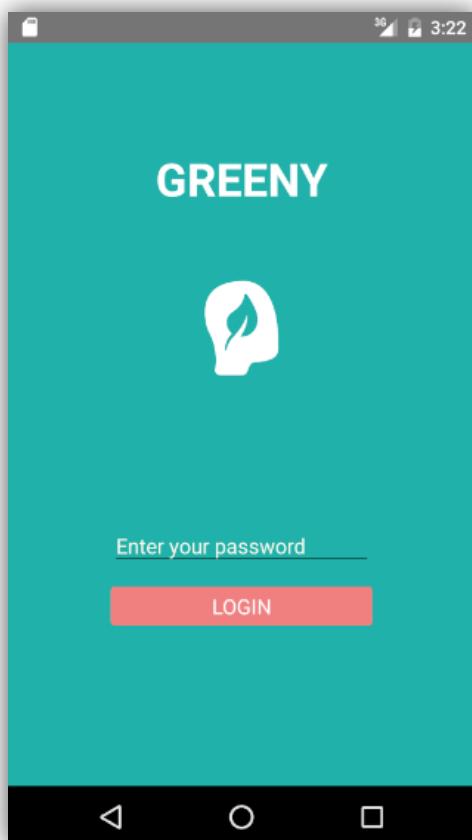


Ilustración 5 Iniciar sesión Android

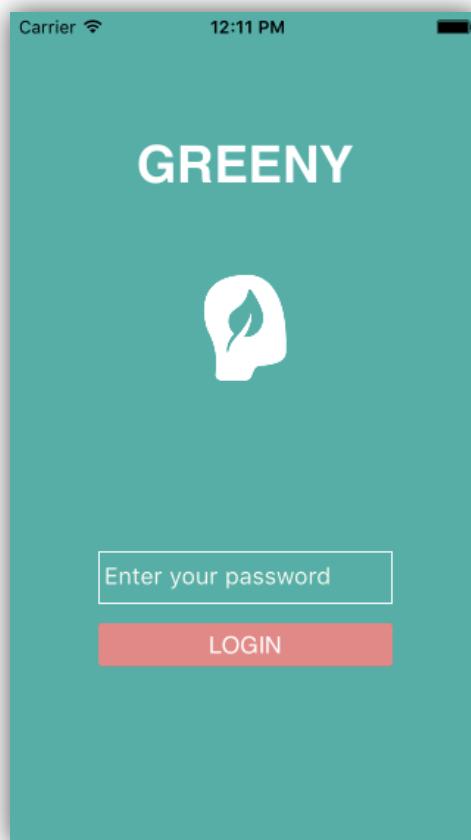


Ilustración 4 Iniciar sesión iOS

En las ilustraciones 7, 8, 9 y 10 se muestra la pantalla que permite al usuario añadir una nueva acción al registro de sus acciones y que corresponde al requisito funcional de la tabla 22 (Ver sección 9.1.2 Página 38). Cada una de las pantallas corresponde a una de las cuatro secciones. A cada sección se le ha dado un color que lo identifique y cada ícono de una acción se caracterizará por tener el color de su sección.

Los componentes y librerías que se han usado que ofrece React Native han sido:

- TabBar: Este componente ha sido implementado por el autor haciendo uso de touchableHighlight, Text e Image tal como se puede observar en la ilustración 6. Se han implementado cuatro pestañas, una por cada pantalla que el usuario puede consultar. Cada una de las pestañas contiene un ícono que hace referencia a la funcionalidad y un texto que ayude al usuario a saber a qué pantalla puede navegar. La pestaña en la que el usuario se encuentra aparece marcada mostrando el ícono y el texto con el color que caracteriza la aplicación (Light Sea Green). En caso contrario

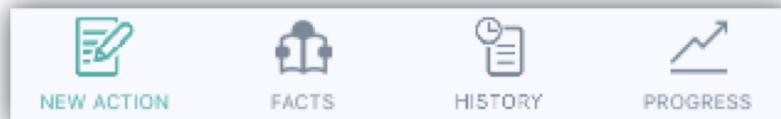


Ilustración 6 Tab Bar

se muestra en color Light Slate Gray. En la ilustración 6 podemos observar como la pestaña *New Action* está marcada.

Esta barra de navegación aparece en la parte inferior de todas las pantallas exceptuando las de autenticación.

- Text: Este componente ha sido utilizado para el texto del botón de info, para el texto que define cada acción, para el “total score” y el número que representa la puntuación del usuario. Este componente también está usado en el botón de “Ok” de la parte superior derecha y para el título de la pantalla, en este caso, new action.
- TouchableHighlight: Componente utilizado para crear la sección táctil de cada una de las cuatro acciones de cada sección.
- ActivityIndicator: Muestra un spinner circular para mostrarle al usuario que el registro de una acción se está llevando a cabo tal y como se muestra en las ilustracion 14 y 15 que son explicadas más adelante.
- Animated: Esta librería se ha utilizado en varias ocasiones para esta pantalla. Se ha animado el contenedor circular de la puntuación. En el momento que un usuario registra una acción, la puntuación cambia y se decidió animar el contenedor haciendo que parpadeara para que el usuario fuera consciente del cambio. Otra animación que se lleva a cabo en esta pantalla es la del botón “OK” que variando el valor de la opacidad hacemos que se muestre como un botón activo o inactivo.
- Image: Este componente se ha usado en muchas ocasiones para esta pantalla. Todos los casos que se ha necesitado poner un icono en la pantalla se ha hecho uso del componente Image, tanto para los iconos es de las acciones como para los iconos de las secciones.
- Modal: Este componente se ha utilizado para presentar un contenido por encima de la vista principal. Se ha usado para mostrar la información de cada sección que se muestra al hacer click sobre el botón info. Esta pantalla se explicará más adelante junto a la ilustración 11.

A continuación, en la tabla 56, encontramos las acciones por sección que serán mostradas al usuario en esta pantalla

		Acción				
		Food	Meat	No meat	Fish	No fish
Sección	Food	Meat	No meat	Fish	No fish	
	Water	Short shower	Long shower	Bath	Cold water	
	Transportation	Bike	Car	Public transportation	Plane	
	Temperature	Heating	Cooling	Sweater	Window	

Tabla 56 Acciones por sección

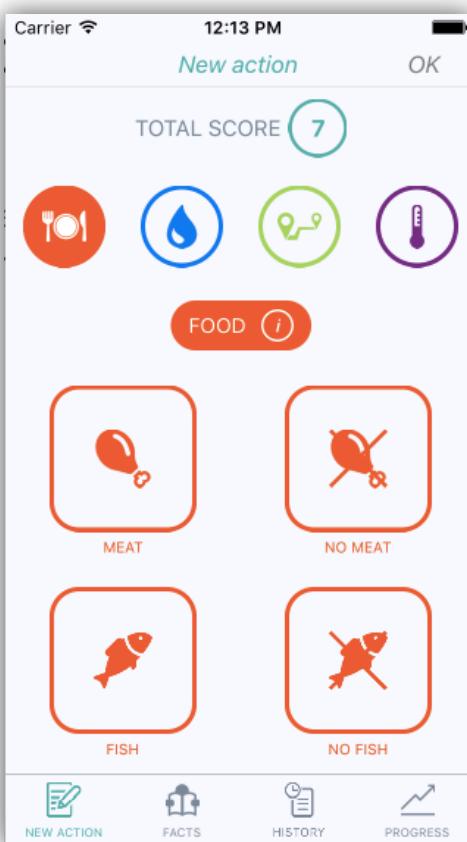


Ilustración 9 Añadir acción en sección "Comida" iOS

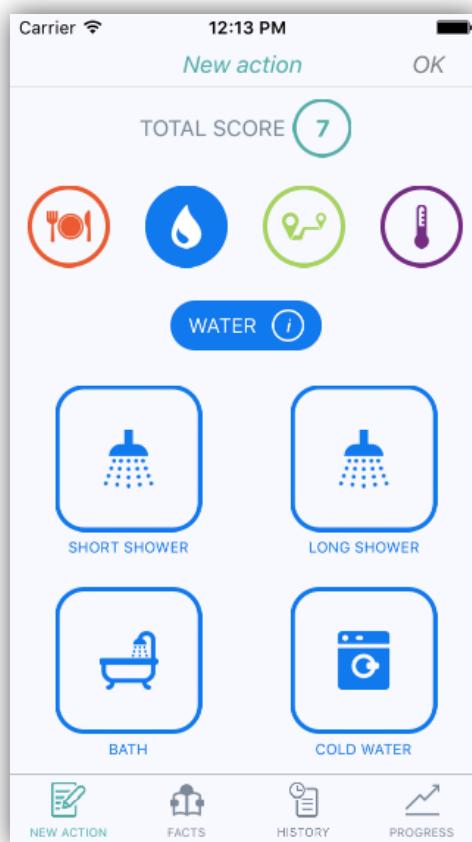


Ilustración 8 Añadir acción en sección "agua" iOS

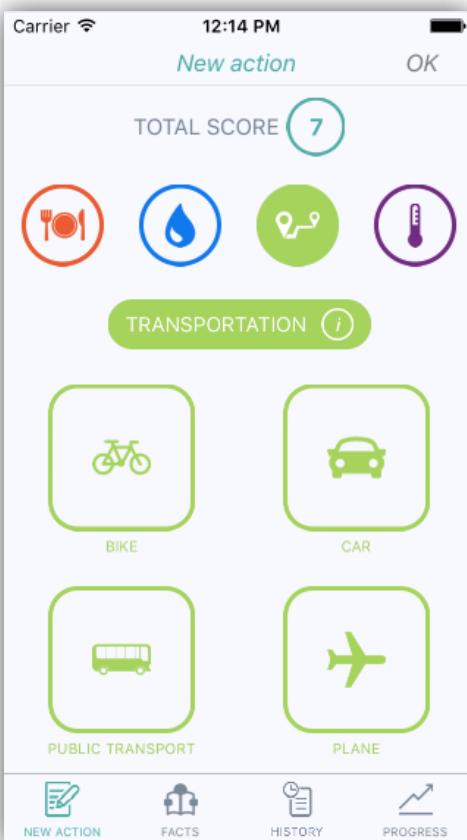


Ilustración 10 Añadir acción en sección "Transporte" iOS

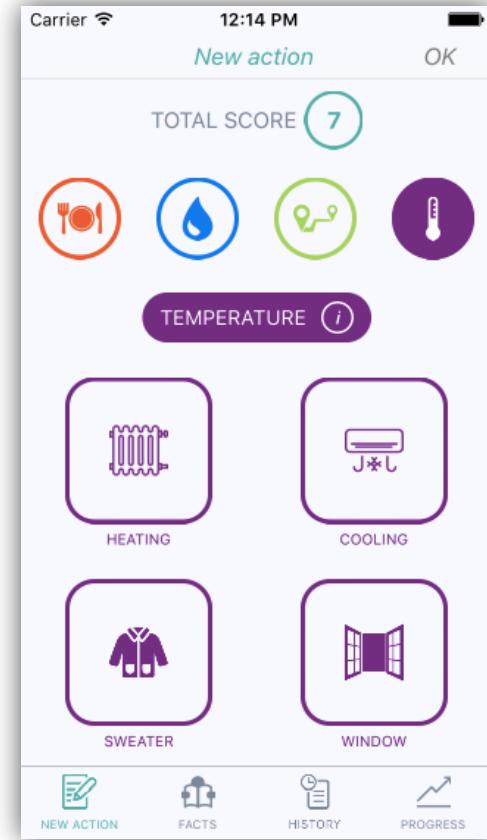


Ilustración 7 Añadir acción en sección "Temperatura" iOS

En la ilustración 11 podemos observar la pantalla que se muestra una vez el usuario ha hecho click en el botón Info que aparece en cada sección de la pantalla “Añadir acción” de las ilustraciones 7, 8, 9 y 10. Esa pantalla hace referencia a los casos de uso consultar información de sección de la tabla 25 (Ver sección 9.1.2 Página 38) y consultar información de acción de la tabla 26 (Ver sección 9.1.2 Página 38).

Cada sección mostrará una información específica para el campo que se está tratando y una tabla que muestra la puntuación que recibe el usuario por cada acción que registra en el sistema. En este documento solo se muestra la pantalla de información de la sección de Temperatura pero las otras tres secciones tienen el mismo diseño mostrando un texto y una tabla.

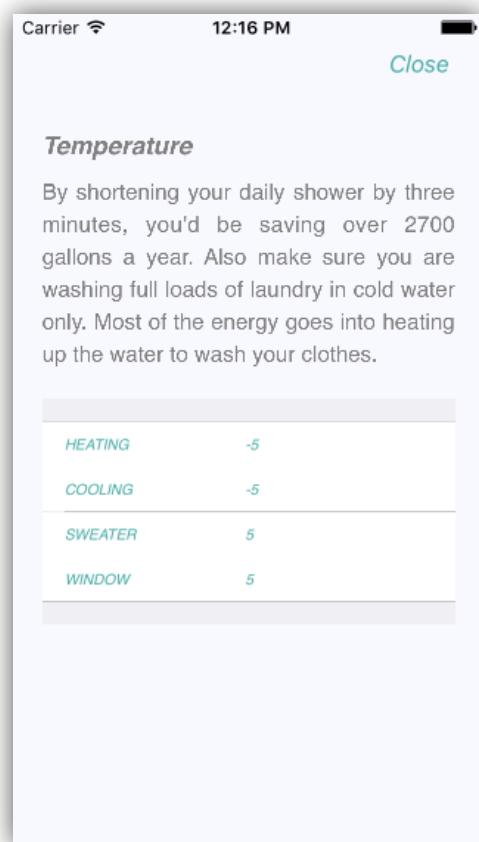


Ilustración 11 Información sección “Temperatura” iOS

En la ilustración 12 podemos observar el aspecto que tiene la pantalla de añadir una nueva acción cuando el usuario ha seleccionado una acción. Este es el paso previo a confirmar la acción para que se registre en el sistema.

En la ilustración 12 podemos observar como la acción ha sido marcada cambiando el color de fondo del contenedor de blanco al color de la sección, en este caso verde. Otro cambio que ocurre es que el botón de Ok, para poder confirmar la acción, ha pasado de estar inactivo a estar activo si lo comparamos con las ilustraciones 7, 8, 9 y 10. Es para crear este efecto que se ha hecho uso de la librería Animated de React Native. El botón “OK” va variando el valor de su opacidad permitiendo que se muestre activo o inactivo.

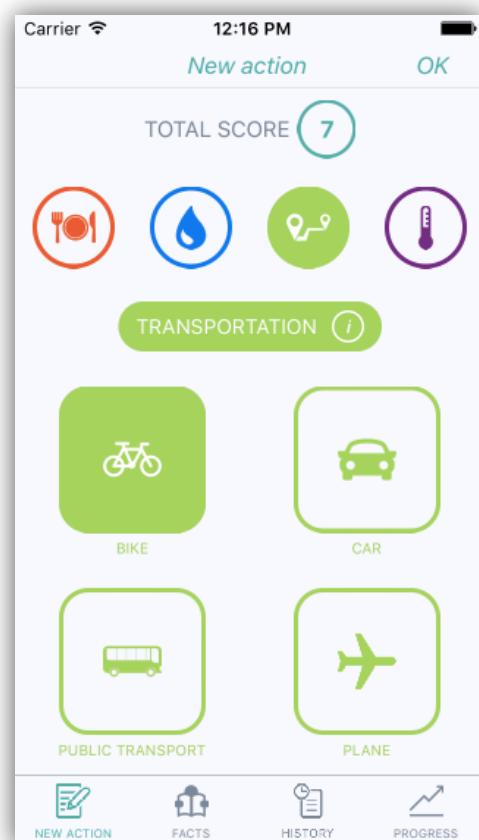


Ilustración 12 Acción seleccionada iOS

En las pantallas que aparecen a continuación en las ilustraciones 13 y 14 vemos lo que el usuario ve una vez ha marcado una acción y ha hecho click en el botón de “OK” para registrarla. Observando ambas ilustraciones podemos observar como el spinner (señalado con una flecha) es distinto en Android que en iOS pese a que nuestro código es el mismo, no hemos diferenciado de iOS y Android. Aquí podemos observar otra vez la gran cualidad que React Native ofrece.

Lo que ve es un spinner en la parte superior derecha, es decir, una especie de aguja giratoria que representa un “Cargando...”. Mientras se trata el registro de la acción, la aplicación muestra este spinner que permite que el usuario sepa que la aplicación está procesando la información que acaba de facilitar. Para ello hacemos uso de un componente que ofrece React Native que se llama ActivityIndicator.

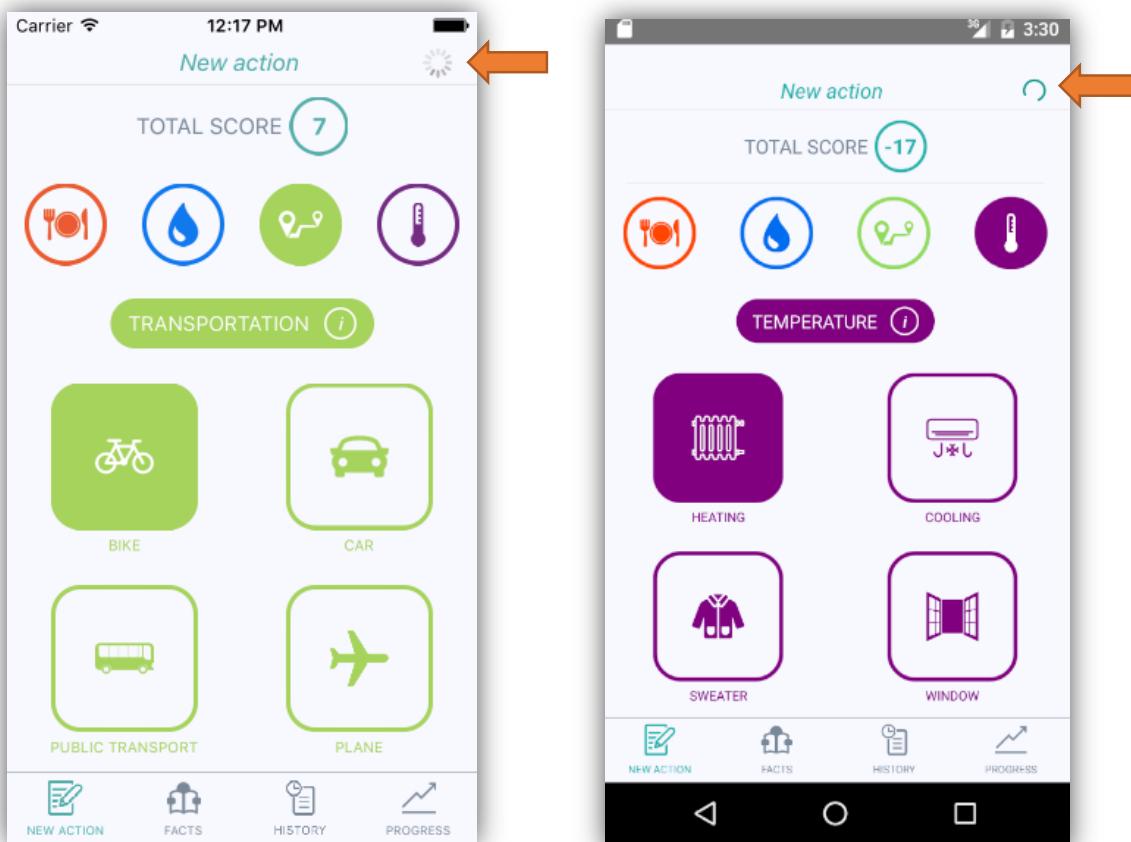


Ilustración 14 Spinner mostrándose iOS

Ilustración 13 Spinner mostrándose Android

En la ilustración 15 se muestra la pantalla de “Fact of the day” del sistema que cumple el requisito funcional de la tabla 28 (Ver sección 9.1.2 Página 40).

Esta pantalla consta del Tab Bar bar que permite navegar por las cuatro funcionalidades principales de la aplicación. Por diseño y por usabilidad, y como se ha explicado anteriormente, la pestaña en la que se encuentre el usuario estará resaltada cambiando el color de la fuente y del ícono al color que caracteriza la aplicación (Light Sea Green). Por lo tanto, para esta pantalla estará resaltada la pestaña “Facts” del menú inferior.

En esta pantalla aparece un título “Fact of the day” para el que se ha usado el componente Text y debajo el hecho mostrado durante ese día. Este hecho se actualiza cada vez que el usuario consulta esta pestaña y ha pasado un día desde entonces. Los hechos que aparecen actualmente son elegidos por el autor del proyecto. En un futuro se espera poder trabajar con un experto en medio ambiente que pueda facilitar hechos relevantes e interesantes para el usuario.

La pantalla que aparece en la ilustración 16, es la pantalla que muestra el historial de acciones al usuario que cumple el requisito funcional de la tabla 27 (Ver sección 9.1.2 Página 40). Consta del tab bar con cuatro botones que permite navegar por las cuatro funcionalidades principales de la aplicación. Podemos observar como la pestaña “history” aparece marcada.

En esta pantalla el usuario podrá observar las acciones que ha llevado a cabo y que han sido registradas en el sistema hasta el momento. En una lista ordenada cronológicamente de más reciente a más antiguo, aparecerán las acciones representadas con el texto que las identifica y el color de la sección a la que corresponden. Además se muestra en forma de barra vertical dividida en los días que el usuario ha registrado alguna acción,



Ilustración 15 Pantalla “Fact of the day” iOS

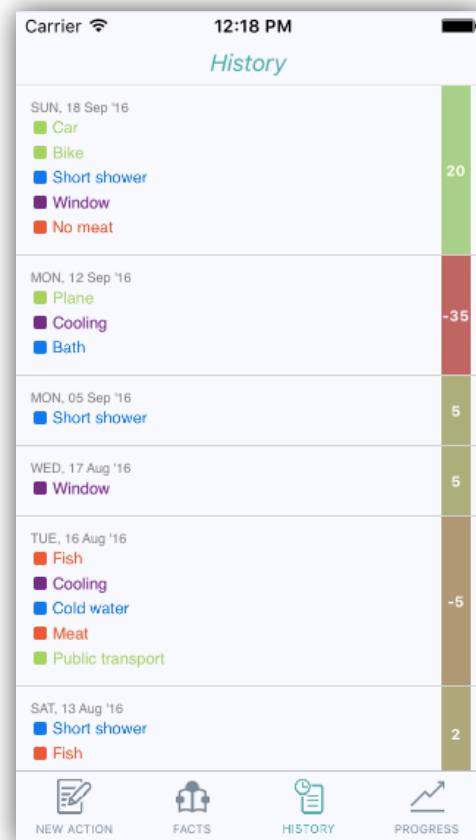


Ilustración 16 Pantalla “history” iOS

la puntuación que resulta de llevar a cabo las acciones de ese día. Dependiendo de la puntuación, la barra tomará un color u otro. El color varía de verde, si la puntuación es alta, a rojo, si la puntuación es baja.

Tal y como se ve en la imagen, la barra de puntuaciones varía sus colores. Para mejorar el diseño de la barra se ha intentado hacer uso de un componente de React Native llamado react-native-linear-gradient [22]. Se implementó con la intención de que el cambio de un color a otro fuera cambiando de manera suave y progresiva y crear un gradiente que hiciera el diseño más atractivo. Dado que este componente no es oficial de React Native ha presentado errores que no han permitido que se mostrara el gradiente como se deseaba. Es por eso que finalmente no se ha utilizado.

En esta pantalla se ha usado también un componente de React Native llamado ListView. Este componente permite mostrar listas como la que se muestra en esta pantalla.

Por último, la ilustración 17 muestra el aspecto que tiene la pantalla que muestra al usuario el progreso de su puntuación que cumple el requisito funcional de la tabla 29 (Ver sección 9.1.2 Página 41). Mediante un gráfico que toma el valor de la puntuación en el eje de las ordenadas y los días en el eje de las abscisas, el usuario puede ver el desarrollo de sus acciones.

Para llevar a cabo esta funcionalidad se ha hecho uso del componente react-native-chart [23] que es un componente creado por una persona ajena a React Native que decidió desarrollar esto dado que React Native no ofrecía ningún componente parecido y compartirlo con el resto de usuarios. Dado que no es un componente oficial ha mostrado errores. Cuando la puntuación alcanza número negativos, el eje de las ordenadas muestra unos números erróneos. El eje de las abscisas tampoco deja mostrar los valores modificados tal y como quería el autor del proyecto. Se espera que en un futuro aparezca un componente que permita mostrar gráficos de la manera deseada.



Ilustración 17 Pantalla "progression" iOS

## 10.2.2. REACT NATIVE

React Native es el framework que se ha utilizado para llevar a cabo la parte del front-end del sistema y de esta manera obtener dos aplicaciones nativas, una para iOS y otra para Android cumpliendo así el requisito #15 (Ver Sección 1.1.1.23 Página 47). Hay un concepto muy claro para toda app implementada con este framework y es que el código debe volver a utilizarse lo más posible. Este concepto puede parecer contradictorio ya que para este proyecto se pretende diseñar dos aplicaciones nativas para cada plataforma. Pero esta es la gracia de React Native, hay una lista de componentes reutilizables como *button*, *container*, *list row*, *header*, etc. Y aparte hay otros componentes únicos y específicos para cada versión.

A continuación tenemos un trozo de código donde observamos el aspecto que tiene un fichero que sigue las normas de React Native. El código corresponde al componente *Facts*. Todos los componentes de nuestro sistema tienen este aspecto con una función *render()* y un StyleSheet que añade los estilos a cada componente:

```
import React, { Component } from 'react';
import {
    StyleSheet,
    View,
    Text,
    Image,
    TextInput
} from 'react-native';
import NavigationBar from './navigationBar';
import * as colors from './colors';

const styles = StyleSheet.create({
    parent: {
        flex: 1,
        backgroundColor: colors.MAIN_BACKGROUND_COLOR
    },
    container: {
        flex: 1,
        alignItems: 'center',
        justifyContent: 'center'
    },
    icon_name: {
        fontWeight: 'bold',
        fontFamily: 'Helvetica',
        fontSize: 16,
        color: colors.APP_COLOR
    },
    textInput_container: {
        alignItems: 'center',
        marginTop: 30,
    },
    text: {
        height: 250,
        width: 250,
        fontFamily: 'Helvetica',
        fontSize: 15,
    }
});
```

```
        color: 'gray',
        textAlign: 'justify',
        lineHeight: 25
    }
});

export default class Facts extends Component {
    constructor(props) {
        super(props);
    }

    render() {
        const { fact, title } = this.props;
        return (
            <View style={styles.parent}>
                <NavigationBar
                    title={title}/>
                <View style={styles.container}>
                    <View style={styles.textInput_container}>
                        <Text style={styles.text}>
                            {fact}\n
                        </Text>
                    </View>
                </View>
            );
    }
}
```

### 10.2.3. REDUX

Redux es una librería clave para este sistema basada en el concepto de Flux, que facilita la labor de conectar fuentes de diferentes entornos como por ejemplo APIs, de forma sencilla, y con una manera de testear fácil [24].

Para entender brevemente como funciona esta librería que se ha usado hace falta primero saber que elementos lo caracterizan:

- Redux tiene un *App state* que se guarda en un objeto *Store*. La única manera de modificar el *App state* es haciendo *dispatch* de una acción.
- El *App state* puede ser cambiado desde cualquier parte y se utilizará para actualizar la interfaz de usuario que reflejará el cambio en el estado.
- Las acciones son usadas para desencadenar la transferencia del *App state* actual a un nuevo *App state*.
- El poder de Redux se encuentra en los *reducers* que son utilizados para modificar el *App state*. Las acciones representan un cambio, pero no especifican como cambia el *App state*. Esto es trabajo de los *reducers*.

Una vez entendidos los elementos que forman parte de Redux, observamos que aspecto tiene un ciclo en Redux con la siguiente imagen (ilustración 18):

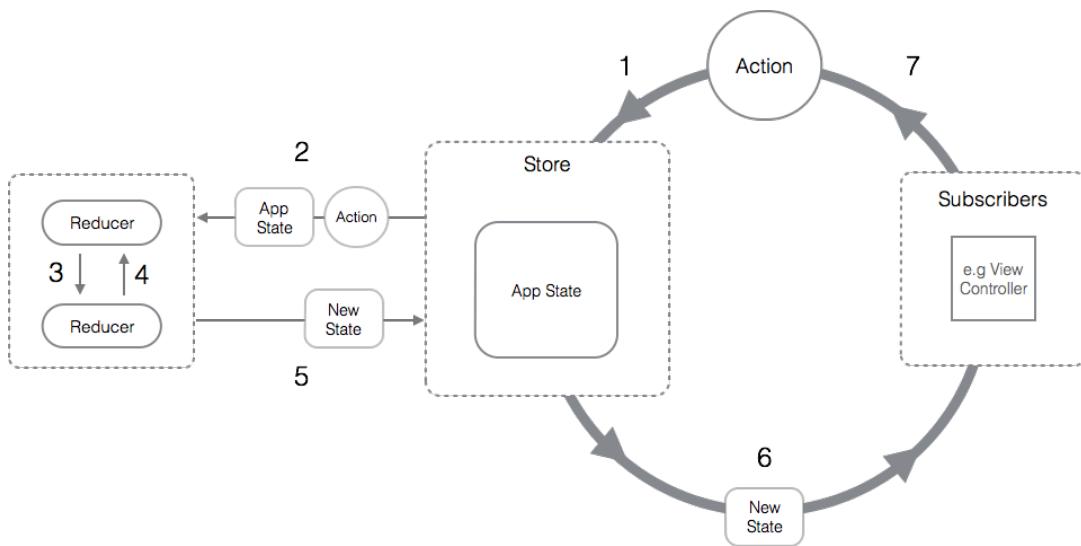


Ilustración 18 Redux flow [25]

1. Una acción es creada por el controlador de una de las *views* o cualquier elemento que pretenda cambiar el *App State*.
2. El *Store* llama al *Reducer* pasándole el estado actual y la acción.
3. El *Reducer* llama a los otros *Reducers* necesarios para tratar la acción.
4. Estos *reducers* devuelven al *Reducer* el *App state* cambiado.
5. El *Reducer* combina los estados que le han llegado de los otros *reducers* para crear un nuevo estado *New state* y el *Store* actualiza el *App State*.
6. El *Store* notifica a los controladores de las *views* sobre el nuevo *App state* para que pueda actualizar el contenido de las *views* con el contenido del *App state*.
7. Los controladores de las *views* pueden crear nuevas acciones que reflejen cualquier interacción del usuario y llamar al *dispatch* del *Store*.

Y el ciclo se repite [25].

Una vez entendido el funcionamiento a grandes rasgos podemos ver la secuencia que se ha seguido en este sistema. Para entenderlo mejor, en la figura 33 aparece el diagrama de secuencia de validar si el email de un usuario está en el sistema o no. Recordemos que en el sistema tenemos tres reducers aunque para esta secuencia solo entran en juego dos, *Authentication Reducer* y *User Reducer*.

El estado del *Authentication Reducer*, que recordemos se actualiza con cada nueva acción, guarda los siguientes valores:

- *isLoggedIn*: evalúa a true si el usuario tiene la sesión iniciada o false si no la tiene iniciada. Toma el valor de false inicialmente.
- *isEmailRegistered*: evalúa a true si el usuario está registrado o false si no lo está. Toma el valor de false inicialmente.
- *uid*: guarda el identificador del usuario. Toma el valor de null inicialmente.
- *email*: guarda el email del usuario. Toma el valor de null inicialmente.
- *error*: guarda el mensaje de error que puede darse en caso que el proceso de autenticación falle. Toma el valor de null inicialmente.



- *token*: guarda el authentication token del usuario. Toma el valor de null inicialmente

El estado del *User Reducer*, que recordemos se actualiza con cada nueva acción, guarda los siguientes valores:

- *history*: guarda en un vector todas las acciones que el usuario ha registrado en el sistema. Inicialmente toma el valor de un vector vacío.
- *score*: guarda la puntuación total que el usuario tiene hasta el momento. Toma el valor de null inicialmente.
- *nextActionId*: guarda el identificador de la siguiente acción que el usuario lleve a cabo. Toma el valor de null inicialmente.

Desde el *container GreenyApp*, donde tenemos los estados que han sido explicados anteriormente, comprobamos que el usuario no tenga iniciada la sesión accediendo al parámetro *isLoggedIn* del *Authentication Reducer*. Como no lo está, se renderiza el componente *Authentication* que corresponde a la pantalla de autenticación del sistema donde el usuario debe introducir su email.

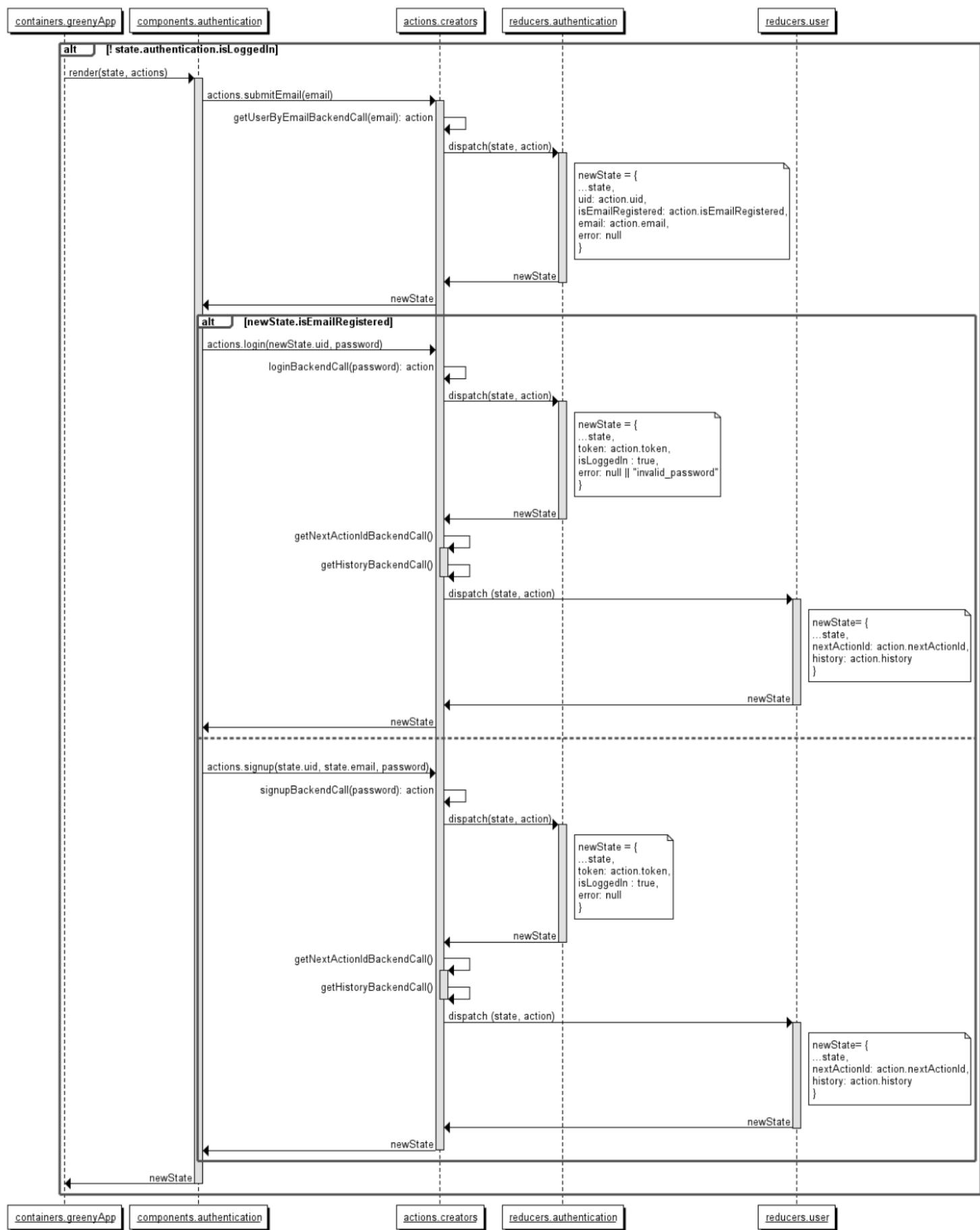
A continuación el usuario introduciría el email y haría click en el botón de validar. Esta acción llamada *submitEmail* desembocará en una llamada al back-end con este email para comprobar si el usuario con este email está registrado o no en el sistema. Esta llamada devuelve la acción que se ha llevado a cabo con todos los parámetros necesarios para actualizar el estado. Esta acción le llega al *Authentication Reducer* que genera un nuevo estado a partir de la información que le llega. En este caso se actualizan los campos *uid*, *isEmailRegistered*, *email* y *error*.

Una vez tenemos este estado actualizado el sistema accede al campo *isEmailRegistered* para saber si debe mostrar la pantalla de Registro o de Iniciar sesión.

En caso que el email esté registrado y el usuario quiera iniciar sesión, el usuario introducirá la contraseña y hará click en el botón de Login que llevará a cabo la acción *login*. En este punto se realizará la llamada al back-end con la contraseña facilitada por el usuario. Esta llamada devuelve una acción que contiene los parámetros necesarios para actualizar el estado. Esta acción le llega al *Authentication Reducer* que genera un nuevo estado a partir de la información que le llega. En este caso se actualizan los campos *token*, *isLoggedIn* y *error*.

Una vez actualizado el estado seguimos con el proceso de *login*. Ahora se hacen dos llamadas más al back-end, una para obtener el *nextActionId* y otra para obtener el historial del usuario. La información que se obtiene de estas llamadas se utiliza para actualizar el estado de nuevo. En este caso la acción le llega al *User Reducer* que actualiza los campos *nextActionId* y *history*.

Después de estas dos actualizaciones del estado ya tenemos el estado con la información deseada. Este estado le llega al *container greenyAp* que decidirá accediendo a los campos del estado, que debe renderizar.



*Figura 33 Secuencia de validar un email en el front-end*

## 11. IMPLEMENTACIÓN

A continuación se explicarán algunos aspectos relativos a la implementación del sistema.

El formato del proyecto esta subdividido en diferentes paquetes con el objetivo de ordenar los tipos de ficheros necesarios. En el caso del front-end, el esqueleto de ficheros tiene el aspecto que aparece a continuación en la ilustración 19:

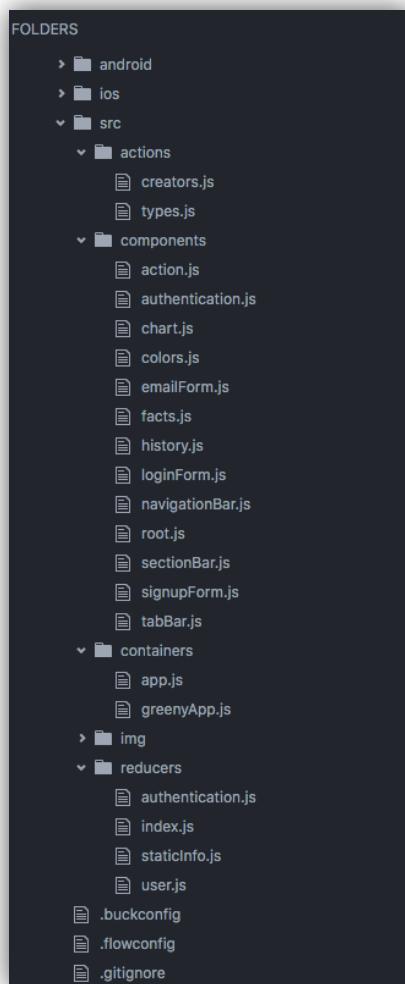


Ilustración 19 Esqueleto ficheros Front-end

Esta parte del proyecto está subdividida en 3 directorios, de los que *iOS* y *Android* se crean al iniciar un nuevo proyecto de *React Native*. Los directorios y ficheros que han sido añadidos durante el transcurso de la implementación del sistema han sido los pertenecientes a /src. En este directorio podemos observar que se encuentran las acciones, componentes, contenedores, imágenes y reducers que usa esta parte de nuestro sistema.

En el caso del back-end el esqueleto de cada microservicio sigue el mismo patrón. El *Gateway Service* contiene un directorio llamado /controllers que contiene todos los ficheros python que corresponden con cada endpoint de este servicio además de un fichero main.py (ilustración 21).

El resto de microservicios, *User Service* (figura 23), *Authentication Service* (figura 22), *Facts Service* (figura 20) y *Content Service* (figura 24) mantienen el mismo aspecto. Todos constan de un fichero main.py, de un Makefile y de tres directorios:

- /controllers
- /model
- /tests

```
content_service/
  controllers
    get_action_types_controller.py
    get_sections_controller.py
    __init__.py
  main.py
  Makefile
  model
    action_types.py
    __init__.py
    section.py
  README.md
  tests
    test_get_action_types_controller.py
    test_get_sections_controller.py
```

Ilustración 24 Esqueleto ficheros Content Service

```
gateway_service/
  controllers
    actions_types_controller.py
    create_action.py
    create_user.py
    facts_controller.py
    get_action_history.py
    get_next_action_id.py
    __init__.py
    sections_controller.py
    user_controller.py
    users_controller.py
    validate_password.py
    validate_token.py
  main.py
```

Ilustración 21 Esqueleto ficheros Gateway Service

```
user_service/
  controllers
    create_action_controller.py
    get_action_history_controller.py
    get_next_action_id_controller.py
    __init__.py
    user_controller.py
    users_controller.py
  main.py
  Makefile
  model
    action.py
    __init__.py
    user.py
  tests
    test_create_action.py
    test_get_action_history.py
    test_get_next_action_id.py
    test_user.py
    test_users.py
  views
```

Ilustración 23 Esqueleto ficheros User Service

```
facts_service/
  controllers
    facts_controller.py
    __init__.py
  main.py
  Makefile
  model
    fact.py
    __init__.py
  tests
    test_facts_controller.py
  views
```

Ilustración 20 Esqueleto ficheros Facts Service

```
authentication_service/
  api_server.txt
  controllers
    common.py
    create_token.py
    __init__.py
    put_password.py
    validate_password.py
    validate_token.py
  main.py
  Makefile
  model
    __init__.py
    user.py
  tests
    test_create_token.py
    test_put_password.py
    test_validate_password.py
    test_validate_token.py
```

Ilustración 22 Esqueleto ficheros Authentication Service



## 11.1. LENGUAJES DE PROGRAMACIÓN

Para implementar este sistema se han usado dos lenguajes de programación distintos, uno para cada parte del sistema, Front-End y Back-End.

Para la parte del Back-End se ha utilizado el lenguaje de programación Python. El autor lo escogió con la intención de aprender en profundidad un lenguaje moderno que además ofrece una sintaxis sencilla y simple.

Para la parte del Front-End se tuvo que utilizar Javascript. El motivo es que el framework React Native está basado en Javascript por lo que no se ha podido escoger otro lenguaje para esta parte del sistema.

## 12. FASE DE PRUEBAS

### 12.1. FASE DE PRUEBAS DEL BACK-END

Para testear el Back-End se ha dividido el proceso en dos partes en las que el testeo se ha llevado de diferente manera. Primero se ha testado cada uno de los servicios por separado después de la implementación de cada uno de ellos. Una vez implementados y testeados los servicios se ha hecho un testeo global y ya corriendo la aplicación en la nube. En los siguientes apartados se entra en detalle.

#### 12.1.1. FASE DE PRUEBAS POR SERVICIO

Para testear cada uno de los servicios se ha utilizado una librería que ofrece el framework Falcon llamada testing. Esta librería nos proporciona, entre otras muchas, la función *simulate\_request()* que simula una petición a una aplicación WSGI. A esta función se le pasa por parámetro el método usado en la petición (GET, POST, etc.), la URL, los headers que se deben incluir en la request y el query string y el body en caso de que sean necesarios.

Para cada controlador de cada servicio se ha creado un test que valida el correcto comportamiento del servicio. Para ello, el resultado de las peticiones simuladas se compara haciendo uso de la función *assertEqual()* donde se puede observar que este resultado obtenido contiene los valores esperados. A continuación aparece el trozo de código correspondiente al testo del controlador que valida la contraseña de un usuario que está intentando entrar en la aplicación. En concreto se puede observar cómo se valida:

- Que la contraseña es correcta en la función *test\_validate\_password\_correct()*.
- Que la contraseña es incorrecta en la función *test\_validate\_password\_incorrect()*.
- Que el usuario del que se intenta validar su contraseña no existe, en la función *test\_validate\_password\_user\_not\_exists()*.
- Que el authentication token del usuario ha sido actualizado después de la petición en la función *test\_user\_already\_has\_token()*

```
class TestValidatePassword(testing.TestCase):
    def before(self):
        self.api = main.create_api()

    def test_validate_password_correct(self):
        uid = str(uuid.uuid4())
        password = "foo"
        user = User(uid=uid, password=password)
        user.write()
        body = self.req(uid, password)
        user = model.user.read(uid)
        body = json.loads(body)
        self.assertEqual(self.srmock.status, falcon.HTTP_200)
        self.assertEqual(body['token'], user.get_token())
        self.assertEqual(password, user.get_password())

    def test_validate_password_incorrect(self):
        uid = str(uuid.uuid4())
        password = "foo"
        user = User(uid=uid, password=password)
        user.write()
        body = self.req(uid, "foo2")
        body = json.loads(body)
        self.assertEqual(self.srmock.status, falcon.HTTP_401)
        self.assertEqual(body, {'errors': [{'code': "password_invalid"}]})

    def test_validate_password_user_not_exists(self):
        uid = str(uuid.uuid4())
        password = "foo1"
        body = self.req(uid, password)
        self.assertEqual(self.srmock.status, falcon.HTTP_404)
        body = json.loads(body)
        self.assertEqual(body, {'errors': [{'code': "user_not_found"}]})

    def test_user_already_has_token(self):
        uid = str(uuid.uuid4())
        password = "foo"
        old_token = "abc"
        user = User(uid=uid, password=password, token=old_token)
        user.write()
        body = self.req(uid, password)
        user = model.user.read(uid)
        body = json.loads(body)
        self.assertEqual(self.srmock.status, falcon.HTTP_200)
        self.assertEqual(body['token'], user.get_token())
        self.assertEqual(password, user.get_password())
```

```
    self.assertEqual(old_token, user.get_token())

def req(self, uid, password):
    headers = [('Accept', 'application/json'),
               ('Content-Type', 'application/json'),]
    return self.simulate_request(
        '/users/' + uid + '/validate_password',
        headers=headers,
        decode='utf-8',
        method="POST",
        body=json.dumps({'password': password}))
```

### 12.1.2. MOCKING

Para poder testear alguno de los controladores ha sido necesario hacer uso de la librería de Python mock. Esta librería te permite remplazar partes de tu sistema para poder llevar a cabo los tests. En el caso de este sistema hemos mockeado todas las peticiones previas que se realizan para poder llevar a cabo la creación de un nuevo usuario.

El siguiente trozo de código corresponde al test del controlador que añade un nuevo usuario al sistema. El controlador corresponde al *User Service*. Para llevar a cabo esta funcionalidad, se tienen que hacer dos peticiones, una que cree y guarde el *authentication token* y otra que guarde la contraseña del usuario. Ambas acciones se llevan a cabo en el *Authentication Service*. Es aquí donde hacemos uso del Mocking para poder validar correctamente el funcionamiento del controlador. En la función `test_put_user()` se puede observar que se mockea una petición POST que corresponde a la creación del token devolviendo el *authentication token*. La siguiente petición mockeada es PUT y corresponde a la petición que guarda la contraseña del usuario que se pretende crear.

```
class TestUser(testing.TestCase):
    def before(self):
        self.api = main.create_api()

    @patch('requests.post')
    @patch('requests.put')
    def test_put_user(self, mock_requests_post, mock_requests_put):
        mock_requests_post.return_value = TestUser.MockResponse({'auth_token':
            "abc"})
        mock_requests_put.return_value = TestUser.MockResponse('')
        uid = str(uuid.uuid4())
        email = "laurac@gmail.com"
        password = str(uuid.uuid4())
        body = {'email': email, 'password': password}
```

```
body = self.req(uid, body)
token = json.loads(body)['auth_token']
self.assertEqual(self.srmock.status, falcon.HTTP_200)
user = model.user.read_by_email(email)
self.assertEqual(uid, user.get_uid())
self.assertEqual("abc", token)

def req(self, uid, body):
    headers = [('Accept', 'application/json'),
               ('Content-Type', 'application/json'),]
    return self.simulate_request('/users/' + uid,
                                 headers=headers,
                                 decode='utf-8',
                                 method="PUT",
                                 body=json.dumps(body))

class MockResponse:
    def __init__(self, content):
        self.content = content
```

### 12.1.3. FASE DE PRUEBAS GLOBAL

Una vez todos los servicios fueron implementados y testeados se hizo un test del sistema para corroborar que todos se comunicaban correctamente. En este test se llevan a cabo todas las peticiones posibles.

El siguiente trozo de código corresponde al aspecto que tiene parte de este test. Cada función simula la petición que se realiza para llevar a cabo las diversas funcionalidades que ofrece el sistema. Al final podemos observar los pasos que se han seguido en la variable *steps*.

```
def get_user_with_email_assert_not_found(ctx):
    query_string = {'email': ctx["email"]}
    body = get("/users", query_string, None, 200)
    assert "users" in body.keys()
    assert "uid" in body.keys()
    assert body['users'] == []
    return {"uid": body["uid"]}

def signup(ctx):
    request_body = {'email': ctx["email"], 'password': ctx["password"]}
    body = put("/users/" + str(ctx["uid"]), request_body, None, 200)
    assert "auth_token" in body.keys()
    return {"auth_token": body["auth_token"]}

def get_user_with_email_assert_found(ctx):
```

```
query_string = {'email': ctxt["email"]}
body = get("/users", query_string, None, 200)
assert "users" in body.keys()
assert body['users'] == [{"uid": ctxt["uid"]}]
return {}

def login(ctxt):
    request_body = {"password": ctxt["password"]}
    body = post("/users/" + ctxt["uid"] + "/validate_password", request_body, 200)
    assert "token" in body.keys()
    assert body['token'] != ctxt["auth_token"]
    return {"auth_token": body["token"]}

def get_facts(ctxt):
    body = get("/facts", {}, None, 200)
    assert "display" in body.keys()
    return {}

def get_sections(ctxt):
    body = get("/sections", {}, None, 200)
    assert "sections" in body.keys()
    assert len(body['sections']) > 0
    return {}

def get_action_types(ctxt):
    body = get("/sections/water/actions", {}, None, 200)
    assert "action_types" in body.keys()
    assert len(body["action_types"]) > 0
    return {}

def get_next_action_id(ctxt):
    body = get("/users/" + ctxt["uid"] + "/actions/next_id", {}, None, 200)
    assert "next_action_id" in body.keys()
    return {"next_action_id": body["next_action_id"]}

def create_action(ctxt):
    url = "/users/" + ctxt["uid"] + "/actions/" + ctxt["next_action_id"]
    request_body = {"section": "foo", "action_type": "bar", "score": "5"}
    body = put(url, request_body, ctxt["auth_token"], 200)
    return {}

def create_action_unauthorized(ctxt):
    url = "/users/" + ctxt["uid"] + "/actions/" + ctxt["next_action_id"]
    request_body = {}
    body = put(url, request_body, "foobar", 401)
    return {}
```

```
def get_history(ctxt):
    body = get("/users/" + ctxt["uid"] + "/history", {}, ctxt["auth_token"], 200)
    assert "user_history" in body.keys()
    assert len(body["user_history"]) == 2
    return {}

def run_steps(steps):
    def run_step(ctxt, step):
        new_ctxt_fields = step(ctxt)
        ctxt.update(new_ctxt_fields)
        return ctxt
    reduce(run_step, steps, {})

steps = [setup,
         get_user_with_email_assert_not_found,
         signup,
         get_user_with_email_assert_found,
         login,
         get_facts,
         get_sections,
         get_action_types,
         get_next_action_id,
         create_action,
         get_next_action_id,
         create_action,
         create_action_unauthorized,
         get_history]
run_steps(steps)
```

## 12.2. FASE DE PRUEBAS DEL FRONT-END

Las pruebas para el Front-End han consistido inicialmente en seguir el manual de usuario de la aplicación. El tester, en este caso el autor del proyecto, ha introducido datos reales como si se tratara de un usuario y ha comprobado que las salidas proporcionadas han sido las esperadas tal como se especifican en los casos de uso:

Caso de uso	Testeado con éxito (✓/✗)
Validar email	✓
Registrarse	✓
Iniciar sesión	✓
Consultar secciones	✓
Consultar acciones	✓
Consultar info secciones	✓
Consultar info acciones	✓
Añadir acción	✓
Consultar historial	✓
Consultar fact	✓
Consultar progresión	✓

A continuación se ha medido el tiempo que puede llegar a tardar el proceso de carga de las diversas pantallas de la App. Ninguna se ha demorado más de 5 segundos por lo que se satisface el requisito #07 (Ver Sección 1.1.1.23 Página 44).

Una vez el tester ha comprobado que todo funciona como se esperaba, se decidió pedir a varias personas ajenas al proyecto que llevaran a cabo una serie de tareas y contestaran unas preguntas para evaluar la usabilidad del sistema.

Estas pruebas se han realizado a 3 personas de distintos rangos de edad a los que inicialmente se les pidió llevar a cabo acciones para comprobar que todas las funcionalidades que ofrece el sistema se llevaban a cabo con facilidad. Antes de empezar con este test se les ha explicado en qué consiste la aplicación y cuál es su finalidad. En un futuro se pretende realizar estos test a un número superior de usuarios para confirmar y mejorar la interfaz de usuario.

La tabla 57 muestra el tiempo en minutos que cada usuario ha necesitado para llevar a cabo las tareas. En la columna Media observamos el tiempo medio que los usuarios han necesitado para realizar la tarea y podemos comparar este tiempo con el que aparece en la columna Tiempo esperado. De esta manera podemos observar que los usuarios han necesitado menos tiempo del esperado. Así se cumple el requisito #02 (Ver Sección 1.1.1.23 Página 42) ya que más del 80% de los usuarios ha llevado a cabo las acciones con facilidad. También se cumple el requisito #03 (Ver Sección 1.1.1.23 Página 42) ya que más del 95% de los usuarios no han tenido problemas que implicaran un exceso de tiempo en las pruebas.

Tiempo esperado	User1	User2	User3	Media
-----------------	-------	-------	-------	-------

<b>Registrarse</b>	0.15	0.14	0.18	0.11	<b>0.14</b>
<b>Iniciar sesión</b>	0.15	0.12	0.10	0.10	<b>0.11</b>
<b>Consultar sección "water"</b>	0.30	0.10	0.08	0.10	<b>0.09</b>
<b>Consultar sección "temperature"</b>	0.30	0.05	0.04	0.04	<b>0.04</b>
<b>Consultar acción "bike"</b>	0.30	0.05	0.08	0.07	<b>0.07</b>
<b>Consultar acción "meat"</b>	0.30	0.04	0.04	0.03	<b>0.04</b>
<b>Consultar acción "window"</b>	0.30	0.05	0.05	0.03	<b>0.04</b>
<b>Consultar info sección "water"</b>	0.30	0.20	0.15	0.14	<b>0.16</b>
<b>Consultar puntos acción "bike"</b>	0.30	0.10	0.20	0.08	<b>0.13</b>
<b>Añadir acción "meat"</b>	0.30	0.21	0.15	0.16	<b>0.17</b>
<b>Consultar historial</b>	0.30	0.18	0.20	0.10	<b>0.16</b>
<b>Añadir acción "cooling"</b>	0.30	0.11	0.10	0.08	<b>0.1</b>
<b>Consultar fact</b>	0.30	0.06	0.05	0.04	<b>0.05</b>
<b>Consultar progresión</b>	0.30	0.05	0.05	0.03	<b>0.04</b>

Tabla 57 Tiempo en minutos por usuario en llevar cada funcionalidad

Dado que los usuarios que han llevado a cabo las pruebas no han recibido información previa sobre el uso de la App, se cumple el requisito #04 (Ver Sección 1.1.1.23 Página 43). Se satisface ya que el 85% o más de las personas encuestadas no han trabajado con un sistema similar y no han tenido problemas con la utilización del sistema.

La persona que ha realizado este test a los tres usuarios no ha observado complicaciones por parte de los usuarios al realizar las tareas por culpa del lenguaje por lo que se cumplen los requisitos #05 (Ver Sección 1.1.1.23 Página 43) y #08 (Ver Sección 1.1.1.23 Página 44). Tampoco ha observado confusión debido a los iconos de la App por lo que se satisface el requisito #06 (Ver Sección 1.1.1.23 Página 43).

Una vez el usuario ha acabado con las tareas se le ha pedido que contestaran a una serie de preguntas relacionadas con el diseño de la interfaz: ¿Te ha parecido atractivo el diseño de la aplicación? ¿Te ha parecido intuitivo el diseño de la aplicación? Los tres usuarios han contestado si a ambas preguntas por lo que el requisito #01 (Ver Sección 1.1.1.23 Página 42) se cumple.



## 13. CONCLUSIONES FINALES

Una vez finalizado el trabajo, puedo afirmar que los tres objetivos principales definidos al inicio del proyecto han sido cumplidos; se ha creado una solución software relacionada con el medio ambiente que motiva al usuario a reducir su impacto medioambiental, se ha aprendido React Native desarrollando una app en Android y iOS con dicho framework y se ha aprendido y creado una arquitectura microservices que se ejecuta en el cloud.

He llegado al final habiendo implementado todas las funcionalidades que se definieron al principio y de acuerdo con el alcance del proyecto. Aun así han habido desviaciones que han afectado al presupuesto que han sido detalladas en la sección 7 de este documento, página 27.

Personalmente, he ampliado muchos mis conocimientos sobre el desarrollo de software. He podido poner en práctica conocimientos aprendidos durante la carrera y aprender de nuevos. Me he enfrentado a nuevas tecnologías que no había usado nunca y que por ser actuales y poco conocidas aún no disponían ni de mucha ni buena documentación, lo que complicó las cosas y en muchos momentos me hizo verme superada por la situación. Han habido muchos días que me he quedado bloqueada y acababa pensando que no iba a poder terminarlo por falta de conocimiento. Aun así acababa consiguiendo avanzar y completar las diferentes fases de este proyecto.

Es el primer proyecto en el que yo misma he tomado tantas las decisiones iniciales como finales y donde he desarrollado todas las partes de un software de manera individual.

### 13.1. LIMITACIONES Y DIFICULTADES

Inicialmente se dijo que en caso de falta de tiempo, se dejaría de implementar la funcionalidad “fact of the day” del sistema. Finalmente se ha podido llevar a cabo todo lo definido inicialmente aunque hubo dificultades y limitaciones relacionadas con los componentes y módulos que forman la interfaz de usuario como se ha explicado en el apartado 10.2.1 página 84. Hubo dos componentes (uno que se eliminó y otro que se dejó) para llevar a cabo las funcionalidades que no funcionaban como se esperaba. Esto es debido a que React Native no ofrecía estos componentes y se tuvieron que buscar proyectos de open source que otros usuarios habían desarrollado y que, en estos dos casos, tenían limitaciones.

### 13.2. TRABAJO FUTURO

El proyecto ha cumplido los objetivos planteados en el inicio, pero el sistema tiene la capacidad de evolucionar mucho ya que se ha diseñado haciendo muy escalable. Añadir nuevas secciones y acciones a la aplicación sería sencillo así como añadir nuevas funcionalidades más genéricas.



Los componentes y módulos open source utilizados que no funcionan como deberían por tener errores en su implementación han de ser tratados una vez desaparezcan esos errores.

### 13.3. COMPETENCIAS TÉCNICAS

Al inscribir el proyecto se definieron varias competencias técnicas que se debían cumplir al término del mismo. A continuación podemos ver las competencias definidas y su justificación de cumplimiento:

- **CES1.2: Dar solución a problemas de integración en función de las estrategias, de los estándares y de las tecnologías disponibles.** [Bastante]: Este proyecto está orientado a mejorar y dar una solución a la situación medioambiental por la que la sociedad pasa. Se pretende concienciar a los usuarios de que sus acciones cotidianas tienen una repercusión negativa en el medio ambiente. Por el momento la aplicación realiza un seguimiento de las acciones que tienen que ver con la comida, el agua, el transporte y la temperatura. Otras secciones podrían ser añadidas para que el usuario sepa mejor que hacer para mejorar.
- **CES1.3: Identificar, evaluar y gestionar los riesgos potenciales asociados a la construcción de software que se puedan presentar.** [En profundidad]: Los riesgos principales a la hora de desarrollar este software era el poder implementar la arquitectura de microservicios y ejecutarla en el cloud para que estuviera siempre disponible. Otro riesgo fue el uso de React Native que dispone de poca documentación debido a que Facebook ha liberado este framework recientemente.
- **CES1.4: Desarrollar, mantener y evaluar servicios y aplicaciones distribuidas con soporte de red.** [En profundidad]: Con la ayuda de Amazon Web Services y en concreto EC2 se han podido subir y ejecutar los cinco servicios que forman la API de este sistema. De esta manera la aplicación siempre puede acceder a la API a través de la red.
- **CES1.5: Especificar, diseñar, implementar y evaluar bases de datos.** [Un poco]: Se ha conseguido manejar un nuevo tipo de base de datos del cual no teníamos conocimientos previos. El usar Amazon Web Services, en concreto S3 y DynamoDB, ha sido clave para el correcto desarrollo de este sistema.
- **CES1.7: Controlar la calidad y diseñar pruebas en la producción de software.** [Bastante]: Se han creado pruebas para ambas partes del sistema, back-end y front-end. Se han testeado cada servicio por separado y luego conjuntamente. Para la aplicación se han llevado a cabo tests de usabilidad. Más información sobre el sistema de pruebas en el apartado 12 página 91.
- **CES1.9: Demostrar comprensión en la gestión y gobierno de los sistemas de software.** [Un poco]: Durante el tiempo de diseño e implementación del back-end y del front-end han aparecido problemas que no se supieron abordar desde un principio. Sin embargo, la aplicación ha acabado funcionado correctamente y se han podido solucionar estos problemas.

- CES2.1: Definir y gestionar los requisitos de un sistema software. [En profundidad]: Se han definido y cumplido con satisfacción los requisitos definidos en la sección 9 página 36. La mayoría de ellos eran relacionados con la escalabilidad del sistema y gracias al diseño que nuestro sistema se han podido cumplir.
- CES2.2: Diseñar soluciones apropiadas en uno o más dominios de aplicación, usando métodos de ingeniería del software que integren aspectos éticos, sociales, legales y económicos. [En profundidad]: Al definir la matriz de sostenibilidad y haber obtenido un 82 sobre 90 hizo ver que íbamos por buen camino para cumplir este objetivo. Los problemas temporales que han aparecido han podido afectar a la gestión económica, pero se ha podido solventar sin tener repercusión.

## 14. BIBLIOGRAFÍA

- [1] J. L. S. Sancho, «Tecnologías para un desarrollo sostenible,» [En línea]. Available: <http://www.rac.es/ficheros/doc/00431.pdf>. [Último acceso: 22 Febrero 2016].
- [2] «Ley Orgánica de Protección de Datos de Carácter Personal (España),» [En línea]. Available: [https://es.wikipedia.org/wiki/Ley\\_Org%C3%A1nica\\_de\\_Protecci%C3%B3n\\_de\\_Datos\\_de\\_Car%C3%A1cter\\_Personal\\_%28Espa%C3%B1a%29](https://es.wikipedia.org/wiki/Ley_Org%C3%A1nica_de_Protecci%C3%B3n_de_Datos_de_Car%C3%A1cter_Personal_%28Espa%C3%B1a%29). [Último acceso: 2016].
- [3] «Barómetro de diciembre 2015,» Diciembre 2015. [En línea]. Available: [http://datos.cis.es/pdf/Es3121mar\\_A.pdf](http://datos.cis.es/pdf/Es3121mar_A.pdf). [Último acceso: 26 Febrero 2016].
- [4] «¿Sabe cuánto petróleo queda realmente? ¿Cree que podrá permitirse volar dentro de 20 años? ¿Le intriga el futuro de la economía mundial?,» [En línea]. Available: <http://www.cenit-del-petroleo.com/>. [Último acceso: 26 Febrero 2016].
- [5] iabspain, «Estudio anual sobre Mobile Marketing,» 29 Septiembre 2015. [En línea]. Available: <http://www.iabspain.net/mobile-marketing/>. [Último acceso: 26 Febrero 2016].
- [6] M. Fussell, «¿Por qué usar un enfoque de microservicios para crear aplicaciones?,» [En línea]. Available: <https://azure.microsoft.com/es-es/documentation/articles/service-fabric-overview-microservices/>. [Último acceso: 23 Febrero 2016].

- [7] H. Loftis, «Why Microservices Matter,» [En línea]. Available: [https://blog.heroku.com/archives/2015/1/20/why\\_microservices\\_matter](https://blog.heroku.com/archives/2015/1/20/why_microservices_matter). [Último acceso: 23 Febrero 2016].
- [8] M. Varela, «Aumenta el interés por el medioambiente en Europa,» 1 Febrero 2016. [En línea]. Available: <http://hablandoenvidrio.com/aumenta-el-interes-por-el-medioambiente-en-europa/>. [Último acceso: 24 Febrero 2016].
- [9] A. Pope, «5 smartphone apps that help sustainability,» Enero 2016. [En línea]. Available: <http://www.canadiangeographic.ca/magazine/jf16/5-smartphone-apps-for-sustainability.asp>. [Último acceso: 25 Febrero 2016].
- [10] «GoodGuide Ratings,» [En línea]. Available: <http://www.goodguide.com/>. [Último acceso: 25 Febrero 2016].
- [11] «Joulebug,» [En línea]. Available: <http://joulebug.com/>. [Último acceso: 25 Febrero 2016].
- [12] «Locavore,» [En línea]. Available: <http://www.getlocavore.com/>. [Último acceso: 25 Febrero 2016].
- [13] PagePersonnel, «Estudios de remuneración 2016,» 2016. [En línea]. Available: [http://www.pagepersonnel.es/sites/pagepersonnel.es/files/er\\_tecnologia16\\_2.pdf](http://www.pagepersonnel.es/sites/pagepersonnel.es/files/er_tecnologia16_2.pdf). [Último acceso: 7 Marzo 2016].
- [14] T. M. d. Barcelona, «Metro Barcelona Bus Barcelona Billetes y tarifas,» [En línea]. Available: <http://www.tmb.cat/es/sistema-tarifari-integrat/-/ticket/TJove>. [Último acceso: 9 Marzo 2016].
- [15] G. España, «El medio ambiente importa,» 2016. [En línea]. Available: <http://www.greenpeace.org/espana/es/Trabajamos-en/medio-ambiente-en-espana/>. [Último acceso: 10 Marzo 2016].
- [16] P. Cáceres, «Las políticas de medio ambiente se esfuman de los Presupuestos del Estado,» 4 Abril 2012. [En línea]. Available: <http://www.elmundo.es/elmundo/2012/04/03/natura/1333465693.html>. [Último acceso: 11 Marzo 2016].
- [17] S. d. A. R. J. M. R. P. Ignacio Quiroz Bartolo, «Desarrollo sustentable, ¿discurso político o necesidad urgente?,» 7 Enero 2016. [En línea]. Available: <http://revistamira.com.mx/desarrollo-sustentable-discurso-politico-o-necesidad-urgente/>. [Último acceso: 9 Marzo 2016].
- [18] B. I. Center, «Caso de éxito: Amazon Web Services reina en el mundo del Cloud,» 17 Abril 2015. [En línea]. Available:

- <http://www.centrodeinnovacionbbva.com/noticias/caso-de-exito-amazon-web-services-reina-en-el-mundo-del-cloud>. [Último acceso: 10 Octubre 2016].
- [19] «Amazon EC2 – Hospedaje de servidores virtuales,» Amazon Web Services, [En línea]. Available: <https://aws.amazon.com/es/ec2/>. [Último acceso: 9 Octubre 2016].
- [20] Freepik, «Flat icon,» [En línea]. Available: [http://www.flaticon.com/free-icon/green-thinking\\_78452#term=head%20leaf&page=1&position=2](http://www.flaticon.com/free-icon/green-thinking_78452#term=head%20leaf&page=1&position=2). [Último acceso: Julio 2016].
- [21] React-native-community, «react-native-linear-gradient,» [En línea]. Available: <https://github.com/react-native-community/react-native-linear-gradient>. [Último acceso: Agosto 2016].
- [22] T. Auty, «react-native-chart,» 1 Febrero 2015. [En línea]. Available: <https://github.com/tomauby/react-native-chart>. [Último acceso: Agosto 2016].
- [23] P. Parra, «Redux tutorial,» 28 Octubre 2015. [En línea]. Available: <https://medium.com/@pedroparra/redux-tutorial-bb8e87eebb09#.gsz4tgc2h>. [Último acceso: Septiembre 2016].
- [24] A. kroll, «Redux for iOS: Unidirectional Data Flow for mobile app development in Swift,» 30 Enero 2016. [En línea]. Available: <http://blog.jtribe.com.au/redux-for-ios/>. [Último acceso: Septiembre 2016].
- [25] C. Arturo, «Explicando que es Front-End, que es Back-End y sus características,» Explicando que es Front-end y que es Back-End, sus diferencias, las herramientas y lenguajes que utiliza cada uno y cual escoger., [En línea]. Available: <http://www.falconmasters.com/web-design/que-es-front-end-y-que-es-back-end/>. [Último acceso: 22 Febrero 2016].
- [26] «¿Qué es el cloud computing?,» [En línea]. Available: <https://debitoor.es/glosario/definicion-cloud-computing>. [Último acceso: 10 Octubre 2016].

## 15. ÍNDICE DE TABLAS

Tabla 1 Recursos materiales.....	18
Tabla 2 Estimación de horas fase inicial.....	21
Tabla 3 Diagrama de Gantt fase inicial.....	22
Tabla 4 Coste estimado de recursos humanos por casos.....	23

Tabla 5 Costes directos por actividad por casos de fase inicial .....	24
Tabla 6 Costes estimado por hardware .....	25
Tabla 7 Costes estimado por software.....	25
Tabla 8 Gastos generales iniciales.....	26
Tabla 9 Contingencia fase inicial .....	26
Tabla 10 Coste total fase inicial.....	26
Tabla 11 Estimación de horas real .....	28
Tabla 12 Daigrama de Gantt real .....	29
Tabla 13 Coste final de recursos humanos por casos.....	30
Tabla 14 Costes directos por actividad por casos reales .....	31
Tabla 15 Gastos generales reales.....	32
Tabla 16 Contingencia real .....	32
Tabla 17 Coste total real.....	32
Tabla 18 Valoración sostenibilidad.....	33
Tabla 19 Descripción del caso de uso "Validar email" .....	37
Tabla 20 Descripción del caso de uso "Hacer login" .....	38
Tabla 21 Descripción del caso de uso "Hacer sign-up" .....	38
Tabla 22 Descripción del caso de uso "Añadir nueva acción" .....	39
Tabla 23 Descripción del caso de uso "consultar secciones" .....	39
Tabla 24 Descripción del caso de uso "consultar acciones" .....	40
Tabla 25 Descripción del caso de uso "consultar información de sección" .....	40
Tabla 26 Descripción del caso de uso "consultar información de acción" .....	41
Tabla 27 Descripción del caso de uso "consultar historial" .....	41
Tabla 28 Descripción del caso de uso "consultar fact of the day" .....	41
Tabla 29 Descripción del caso de uso "consultar progresión" .....	42
Tabla 30 Descripción del caso de uso "Añadir fact" .....	42
Tabla 31 Descripción del caso de uso "Consultar información de usuario" .....	42
Tabla 32 Tabla de Volére del requisito #01.....	43
Tabla 33 Tabla de Volére del requisito #02.....	43
Tabla 34 Tabla de Volére del requisito #03.....	44
Tabla 35 Tabla de Volére del requisito #04.....	44
Tabla 36 Tabla de Volére del requisito #05.....	44
Tabla 37 Tabla de Volére del requisito #06.....	44
Tabla 38 Tabla de Volére del requisito #07.....	45
Tabla 39 Tabla de Volére del requisito #08.....	45
Tabla 40 Tabla de Volére del requisito #09.....	45
Tabla 41 Tabla de Volére del requisito #10.....	45
Tabla 42 Tabla de Volére del requisito #11.....	46
Tabla 43 Tabla de Volére del requisito #12.....	46
Tabla 44 Tabla de Volére del requisito #13.....	46
Tabla 45 Tabla de Volére del requisito #14.....	47



Tabla 46 Tabla de Volére del requisito #15.....	47
Tabla 47 Tabla de Volére del requisito #16.....	47
Tabla 48 Tabla de Volére del requisito #17.....	48
Tabla 49 Tabla de Volére del requisito #18.....	48
Tabla 50 Tabla de Volére del requisito #19.....	48
Tabla 51 Tabla as_users.....	63
Tabla 52 Tabla us_users .....	69
Tabla 53 Tabla us_email_to_uid .....	69
Tabla 54 Tabla us_uid_to_next_action_id .....	69
Tabla 55 Tabla us_actions .....	70
Tabla 56 Acciones por sección .....	79
Tabla 57 Tiempo en minutos por usuario en llevar cada funcionalidad .....	99

## 16. ÍNDICE DE ILUSTRACIONES

Ilustración 1 Validar email iOS.....	76
Ilustración 2 Validar email Android .....	76
Ilustración 3 Registrarse iOS.....	77
Ilustración 4 Iniciar sesión iOS.....	78
Ilustración 5 Iniciar sesión Android .....	78
Ilustración 6 Tab Bar .....	78
Ilustración 7 Añadir acción en sección "Temperatura" iOS .....	80
Ilustración 8 Añadir acción en sección "agua" iOS.....	80
Ilustración 9 Añadir acción en sección "Comida" iOS .....	80
Ilustración 10 Añadir acción en sección "Transporte" iOS .....	80
Ilustración 11 Información sección "Temperatura" iOS .....	81
Ilustración 12 Acción seleccionada iOS .....	81
Ilustración 13 Spinner mostrándose Android .....	82
Ilustración 14 Spinner mostrándose iOS .....	82
Ilustración 15 Pantalla "Fact of the day" iOS .....	83
Ilustración 16 Pantalla "history" iOS.....	83
Ilustración 17 Pantalla "progression" iOS .....	84
Ilustración 18 Redux flow [25].....	87
Ilustración 19 Esqueleto ficheros Front-end .....	90
Ilustración 20 Esqueleto ficheros Facts Service .....	91
Ilustración 21 Esqueleto ficheros Gateway Service .....	91
Ilustración 22 Esqueleto ficheros Authentication Service .....	91
Ilustración 23 Esqueleto ficheros User Service .....	91
Ilustración 24 Esqueleto ficheros Content Service .....	91

## 17. ÍNDICE DE FIGURAS

Figura 1 Gráfica con el reparto de horas por fase .....	19
Figura 2 Diagrama de Gantt fase inicial .....	22
Figura 3 Distribución de horas real .....	27
Figura 4 Diagrama de Gantt real .....	30
Figura 5 Diagrama de casos de uso del gestor.....	36
Figura 6 Diagrama de casos de uso del administrador.....	36
Figura 7 Diagrama de casos de uso del usuario del sistema .....	37
Figura 8 Diseño interno de capas del sistema .....	49
Figura 9 Diagrama de secuencia "validate password" .....	53
Figura 10 Diagrama de secuencia GWS "Get action history" .....	54
Figura 11 Aspecto del fichero facts.json .....	55
Figura 12 Diagrama de clases "Facts Service".....	57
Figura 13 Diagrama de secuencia "Facts Controller" .....	58
Figura 14 Aspecto del fichero water_action_types.json .....	59
Figura 15 Aspecto del fichero transportation_action_types.json .....	59
Figura 16 Aspecto del fichero food_action_types.json .....	59
Figura 17 Aspecto del fichero sections.json.....	59
Figura 18 Aspecto del fichero temperature_action_types.json .....	60
Figura 19 Diagrama de clases "Content Service" .....	61
Figura 20 Diagrama de secuencia "get sections controller" .....	62
Figura 21 Diagrama de secuencia "get action types controller" .....	62
Figura 22 Diagrama de clases "Authentication Service" .....	64
Figura 23 Diagrama de secuencia "create token" .....	65
Figura 24 Diagrama de secuencia "put password" .....	66
Figura 25 Diagrama de secuencia "validate password" .....	67
Figura 26 Diagrama de secuencia "validate token" .....	68
Figura 27 Diagrama de clases "User Service" .....	71
Figura 28 Diagrama de secuencia "create action controller" .....	72
Figura 29 Diagrama de secuencia "get action history controller" .....	72
Figura 30 Diagrama de secuencia "get next action id controller" .....	73
Figura 31 Diagrama de secuencia "user controller" .....	74
Figura 32 Diagrama de secuencia "users controller".....	74
Figura 33 Secuencia de validar un email en el front-end .....	89