

TRABAJO FIN DE MÁSTER

UNIVERSIDAD DE ALMERÍA

ESCUELA SUPERIOR DE INGENIERÍA

**“Aplicación web para fomentar la  
promoción y la movilidad académica  
internacional”**

**Curso 2019/2020**

**Alumno/a:**

Sergio Gil Martínez

**Director/es:**

José Antonio Álvarez Bermejo

MÁSTER EN INGENIERÍA INFORMÁTICA



# Agradecimientos

Quiero dedicar este trabajo a las personas que me dieron la vida y la han alumbrado incondicionalmente. Gracias papá y mamá por estar siempre ahí.

Gracias Cristina por ser mi luz.



# Índice general

<b>1</b>	<b>Motivación y objetivos</b>	<b>10</b>
1.1	Motivación: Fomentar la promoción y la movilidad académica internacional	10
1.2	Objetivos y alcance del Proyecto . . . . .	11
1.3	Organización de la memoria . . . . .	11
<b>2</b>	<b>Tecnologías Web</b>	<b>12</b>
2.1	Disyuntiva: Aplicación Móvil vs. Aplicación Web . . . . .	12
2.2	Plataforma tecnológica seleccionada . . . . .	12
2.3	Método y enfoque de desarrollo . . . . .	13
2.4	Tecnologías utilizadas . . . . .	13
2.4.1	Angular . . . . .	13
2.4.2	Firebase . . . . .	14
2.4.3	Visual Studio Code . . . . .	14
2.4.4	JSON . . . . .	15
2.4.5	Bootstrap . . . . .	16
2.4.6	Angular Material . . . . .	16
2.4.7	GitHub . . . . .	16
2.5	Planificación . . . . .	17
<b>3</b>	<b>Aplicación ErasmusWay</b>	<b>19</b>
3.1	Objetivos . . . . .	19
3.1.1	Objetivo 1: Interfaz minimalista . . . . .	19
3.1.2	Objetivo 2: Registro de usuarios . . . . .	19
3.1.3	Objetivo 3: Autenticación de usuarios . . . . .	20
3.1.4	Objetivo 4: Recordar contraseña . . . . .	20
3.1.5	Objetivo 5: Soporte multiplataforma . . . . .	20
3.1.6	Objetivo 6: Comunicación en tiempo real . . . . .	20
3.1.7	Objetivo 7: Gestión de información de estudiantes . . . . .	20
3.1.8	Objetivo 8: Permitir enviar mensajes . . . . .	20
3.1.9	Objetivo 9: Permitir votación en las tarjetas de información . . . . .	21
3.2	Diagrama de casos de uso . . . . .	21
3.3	Actores . . . . .	22
3.3.1	Actor-01: Usuario sin registrar . . . . .	22
3.3.2	Actor-02: Usuario registrado: Sin confirmar . . . . .	22
3.3.3	Actor-03: Usuario registrado: Confirmado . . . . .	22
3.3.4	Actor-04: Usuario que no recuerda su contraseña de acceso . . . . .	22
3.3.5	Actor-05: Estudiante . . . . .	22

3.4	Requisitos funcionales . . . . .	23
3.4.1	RF-01: Registrarse . . . . .	23
3.4.2	RF-02: Confirmación . . . . .	23
3.4.3	RF-03: Autenticación . . . . .	24
3.4.4	RF-04: Recordar contraseña . . . . .	24
3.4.5	RF-05: Validación . . . . .	25
3.4.6	RF-06: Ver información de otros estudiantes . . . . .	25
3.4.7	RF-07: Visualizar información estudiante (perfil) . . . . .	25
3.4.8	RF-08: Insertar información personal . . . . .	26
3.4.9	RF-09: Enviar/recibir mensajes . . . . .	26
3.4.10	RF-10: Votar una tarjeta . . . . .	27
3.4.11	RF-11: Filtrar información por país . . . . .	27
3.4.12	RF-12: Cerrar sesión . . . . .	28
3.5	Requisitos no funcionales . . . . .	28
3.5.1	RNF-01: Diseño minimalista . . . . .	28
3.5.2	RNF-02: Tiempos de carga reducidos . . . . .	28
3.5.3	RNF-03: La información debe mostrarse de forma clara . . . . .	28
3.5.4	RNF-04: Debe de estar en inglés . . . . .	28
3.5.5	RNF-05: Seguridad . . . . .	28
3.6	Requisitos de información . . . . .	29
<b>4</b>	<b>Diseño de la solución</b>	<b>30</b>
4.1	Prototipado . . . . .	30
4.1.1	Login de usuarios . . . . .	30
4.1.2	Registro . . . . .	31
4.1.3	Recordar contraseña . . . . .	31
4.1.4	Home . . . . .	32
4.1.5	Erasmus profile . . . . .	32
4.1.6	Páginas de entrada de datos . . . . .	33
4.1.7	Tarjetas de datos . . . . .	34
4.1.8	Mensajes . . . . .	34
4.2	Diseño de la arquitectura . . . . .	35
4.2.1	Arquitectura general . . . . .	35
4.2.2	Arquitectura de la aplicación . . . . .	35
4.2.3	Servicio de autenticación . . . . .	37
4.2.4	Modelo de datos . . . . .	37
4.2.5	Base de datos en tiempo real . . . . .	38
<b>5</b>	<b>Implementación de la solución</b>	<b>39</b>
5.1	Estructura del proyecto . . . . .	39
5.2	Módulo principal . . . . .	41
5.2.1	HttpClientModule . . . . .	41
5.2.2	AppRoutingModule . . . . .	41
5.2.3	PagesModule . . . . .	42

---

5.3	Auth . . . . .	43
5.3.1	Register . . . . .	44
5.3.2	Login . . . . .	48
5.4	Shared . . . . .	51
5.5	Pages . . . . .	52
5.6	Components . . . . .	55
5.7	Servicios . . . . .	57
5.7.1	Servicio: Academics . . . . .	58
5.7.2	Servicio: Methods . . . . .	59
5.7.3	Servicio: Dates . . . . .	60
5.8	Guards . . . . .	61
5.9	Base de datos . . . . .	63
5.9.1	Autenticación . . . . .	63
5.9.2	Almacenamiento de datos . . . . .	64
5.9.3	Reglas de la base de datos . . . . .	65
5.10	Pruebas . . . . .	66
5.10.1	Llamadas a la API . . . . .	66
5.10.2	Pruebas funcionales . . . . .	68
5.11	Despliegue . . . . .	73
<b>6</b>	<b>Conclusiones y trabajo futuro</b>	<b>75</b>
6.1	Conclusiones . . . . .	75
6.2	Trabajo futuro . . . . .	76

# Índice de figuras

2.1	JSON: Object . . . . .	15
2.2	JSON: Array . . . . .	15
2.3	Planificación del proyecto . . . . .	17
2.4	Horario de la planificación . . . . .	17
3.1	Diagrama de casos de uso . . . . .	21
4.1	Prototipo: Login . . . . .	30
4.2	Prototipo: Registro . . . . .	31
4.3	Prototipo: Recordar contraseña . . . . .	31
4.4	Prototipo: Home . . . . .	32
4.5	Prototipo: Profile . . . . .	32
4.6	Prototipo: Página de entrada de datos . . . . .	33
4.7	Prototipo: Tarjeta de datos . . . . .	34
4.8	Prototipo: Mensajes . . . . .	34
4.9	Arquitectura general . . . . .	35
4.10	Arquitectura de la aplicación [2] . . . . .	36
5.1	Estructura del proyecto . . . . .	40
5.2	Root module . . . . .	41
5.3	AppRoutingModule . . . . .	42
5.4	Estructura PagesModule . . . . .	43
5.5	Estructura Auth . . . . .	43
5.6	Register: Vista form de Registro . . . . .	44
5.7	Register: Declarar form de registro . . . . .	45
5.8	Register: Método de validación personalizado . . . . .	45
5.9	Register form: Validación del form de Registro . . . . .	46
5.10	Register form: Form cumplimentado . . . . .	46
5.11	Register form: Success . . . . .	47
5.12	Register form: Recepción de email . . . . .	47
5.13	Login: onLogin() . . . . .	48
5.14	Login: Verificar correo . . . . .	48
5.15	Login: Verificar cambio contraseña . . . . .	48
5.16	Login: recuperar oobCode . . . . .	49
5.17	Login: Subscribe Auth: comprobar confirmación . . . . .	49
5.18	Login: Subscribe Student: actualizar base de datos . . . . .	49
5.19	Login: Confirmación de cuenta . . . . .	49
5.20	Login: Verificar cambio contraseña . . . . .	50



5.21	Login: Verificar confirmación . . . . .	50
5.22	Login . . . . .	50
5.23	Login: guardar idToken . . . . .	51
5.24	Estructura Shared . . . . .	51
5.25	Modulo Shared . . . . .	52
5.26	Vista general Pages . . . . .	52
5.27	Router Pages . . . . .	53
5.28	Componentes trabajando juntos . . . . .	54
5.29	Estructura Components . . . . .	55
5.30	Componente hijo: Lógica: card-home . . . . .	55
5.31	Componente hijo: Vista: card-home . . . . .	56
5.32	Componente padre: Vista: page Home . . . . .	56
5.33	Visualización del componente hijo en el padre . . . . .	57
5.34	Estructura Services . . . . .	57
5.35	Decorador de un servicio e inyección . . . . .	58
5.36	Servicio: Academics . . . . .	59
5.37	Servicio: Methods . . . . .	60
5.38	Servicio: Methods . . . . .	60
5.39	Guard: CanActivate usando servicio Auth . . . . .	61
5.40	Guard: Servicio: Validación de acceso . . . . .	62
5.41	Rutas protegidas por el guard . . . . .	63
5.42	Firebase: Autenticación . . . . .	63
5.43	Almacenamiento en colecciones de datos . . . . .	64
5.44	Reglas de la base de datos . . . . .	65
5.45	Configuración de Postman . . . . .	66
5.46	Login: post . . . . .	67
5.47	Login: formulario . . . . .	67
5.48	Login: resultado . . . . .	67
5.49	Pruebas: Postman . . . . .	68
5.50	Pruebas funcionales: Guardar info Personal . . . . .	68
5.51	Pruebas funcionales: Guardar info Academica . . . . .	69
5.52	Pruebas funcionales: Guardar info Idiomas . . . . .	69
5.53	Pruebas funcionales: Guardar info Vivienda . . . . .	70
5.54	Página principal . . . . .	70
5.55	Página principal con filtro . . . . .	71
5.56	Envío de un mensaje . . . . .	71
5.57	Botón para votar una tarjeta . . . . .	72
5.58	Confirmación . . . . .	72
5.59	Angular: Versión optimizada . . . . .	73
5.60	Despliegue ErasmusWay . . . . .	73
5.61	ErasmusWay desplegada en producción . . . . .	74



# 1 Motivación y objetivos

## 1.1. Motivación: Fomentar la promoción y la movilidad académica internacional

La internacionalización es un factor clave para el desarrollo de la calidad de las universidades españolas, teniendo el potencial de ser el motor para la reforma del sistema universitario en cuanto a eficiencia, excelencia y competitividad enmarcadas en un entorno global. España, fiel a su tradición, es un país abierto y que actúa como puente entre dos continentes. Como nos indica el monográfico redactado por el Ministerio de Educación, Cultura y Deporte para la Estrategia para la Internacionalización de las Universidades Españolas 2015-2020, nuestro país es uno de los pocos que cuenta con el privilegio de formar parte de dos de las comunidades políticas y culturales más importantes del mundo, la europea y la iberoamericana.

Todo ello se refleja en las aulas de las universidades españolas que son punto de encuentro para los alumnos europeos, latinoamericanos y, cada vez de más, alumnos internacionales procedentes de otros países. La doble adscripción regional de España supone una gran ventaja en una sociedad globalizada como la actual, con una creciente interdependencia económica del exterior, y en un contexto de cooperación y competición global por el talento y por inversiones asociadas al conocimiento.

El desarrollo de un sistema de educación superior internacionalizado debe jugar un papel fundamental para mantener y contribuir a incrementar el atractivo internacional de España, mediante la educación de profesionales altamente cualificados que nuestra economía y sociedad necesitan, así como aportando a los estudiantes los conocimientos y competencias demandados por una economía global.

Además, la internacionalización proporciona oportunidades de mejorar la calidad de las actividades de aprendizaje, docencia, investigación, transferencia e innovación de las universidades, consolidando a España como una referencia en educación internacional en español, e instándola a iniciar el camino para comenzar a competir en el contexto de la educación internacional completamente en inglés y otras lenguas extranjeras. Nuestras universidades deben ser atractivas y reconocidas internacionalmente, capaces de competir con las mejores universidades de otros países, contribuyendo a resolver los grandes retos sociales, muchos de ellos globales, que las actuales sociedades modernas plantean.

A nivel del Espacio Europeo de Educación Superior (EEES), los Ministros de los 47 países del proceso de Bolonia acordaron en 2007 una Estrategia para la Dimensión Exterior del EEES y en 2012 una Estrategia de Movilidad 2020. En la Unión Europea (UE), la Comisión aprobó en 2013 una estrategia de internacionalización donde proponen medidas y objetivos concretos para promover la movilidad internacional de estudiantes

y de personal, y la internacionalización de los sistemas e instituciones de educación superior. Todos los miembros del EEES deben desarrollar estrategias y objetivos con la finalidad de formar parte de un proceso de internacionalización del sistema e instituciones universitarias [8].

## 1.2. Objetivos y alcance del Proyecto

Este trabajo persigue desarrollar un sistema que potencie, en el exterior, la imagen de la Universidad de Almería (UAL) y de su ciudad. Cuando los estudiantes extranjeros buscan un destino para iniciar o continuar su formación académica, previamente se documentan sobre el mismo. Una de las dificultades a las que se enfrentan es que no suelen encontrar información de calidad disponible. Como nos indica el Servicio de Relaciones Internacionales de la UAL, la información más demandada hace referencia a la ciudad, la universidad, las clases, las asignaturas, el campus, la seguridad de la ciudad, dónde alquilar vivienda o el nivel de atención médica que se dispensa en Almería.

Nuestro interés es que quienes ya nos han visitado como estudiantes, pudieran hacer de embajadores de nuestra universidad allende nuestras fronteras. Por ello, mi propuesta consiste en el desarrollo de un sistema, basado en tecnología Web, que permita actuar como punto de encuentro entre los alumnos internacionales que ya han realizado una experiencia Erasmus y los alumnos que están por llegar, e incluso decidiendo si escoger la UAL como universidad de destino fruto del contacto directo entre los alumnos.

## 1.3. Organización de la memoria

Este documento se divide en seis capítulos bien diferenciados. En el primer capítulo se contextualiza el trabajo descrito en esta memoria, considerándose motivada la realización del presente Proyecto. Además, se presentan los objetivos que se pretenden alcanzar.

En el segundo capítulo se realiza un estudio comparativo entre distintos modelos de aplicaciones, móviles y web, y la justificación de la opción más adecuada, así como una descripción en detalle de todas las tecnologías minuciosamente seleccionadas.

En el tercer capítulo se especifican los objetivos que se pretenden alcanzar. Además, se especifican todos los requisitos funcionales, no funcionales y de información que lo componen.

En el cuarto capítulo se describe la arquitectura y el diseño del sistema, dando lugar al diseño general de la herramienta y al diseño individual de cada uno de sus componentes.

En el capítulo quinto se abordan los detalles de implementación que dan lugar a la herramienta final.

El sexto y último capítulo resume las conclusiones, al mismo tiempo que plantea algunas líneas futuras de trabajo.

## 2 Tecnologías Web

### 2.1. Disyuntiva: Aplicación Móvil vs. Aplicación Web

A la hora de acometer la tarea de desarrollar una aplicación nos surgen grandes cuestiones a valorar:

- ¿A quién va dirigida la aplicación?
- ¿Qué rango de edad tendrán sus usuarios?
- ¿A través de que dispositivos van a visualizar la aplicación?
- ¿Cuan fiel va a ser el usuario para utilizar nuestra aplicación?
- ¿Aplicación móvil nativa, híbrida o aplicación web?

Las aplicaciones son las que mejor consideradas están para fidelizar al cliente, sin embargo, deben conseguir ser relevantes y útiles para no desaparecer por falta de uso. Así, un estudio de Google demostró que el 87 % de los clientes decía ser fiel a una marca sin necesidad de tener su aplicación. Es más, un 25 % no sabía que su marca favorita tenía aplicación por lo *que entendemos que obligar* a los usuarios a descargarse una aplicación no es una opción aconsejable puesto que lo pueden percibir como una trampa. De hecho, el 63 % la desinstala a los pocos días de utilizarla.

Por otra parte, ante la comparativa de una aplicación web a una aplicación móvil, dos tercios de los clientes potenciales consiguen el mismo objetivo, por lo que parece que las diferencias son más de funcionamiento, rapidez y usabilidad [9].

### 2.2. Plataforma tecnológica seleccionada

Con los datos anteriormente aportados sobre el uso de aplicaciones móviles y webs, y conociendo previamente que nuestro sistema va ser utilizado por un público objetivo con una edad comprendida entre los 18 y 28 años, que en su mayoría utilizan dispositivos móviles para conectarse, pero que también pueden utilizar dispositivos portátiles, optamos por desarrollar una aplicación web con un diseño responsable que permita la visualización en distintas resoluciones y completamente optimizado para una visualización eficiente en dispositivos móviles.

Parte del éxito de la solución propuesta, residirá en la fiabilidad y optimización de la aplicación, en la estabilidad del sistema en la que se ejecute y, sobre todo, en la eficiencia que experimente el usuario a la hora de utilizarla.

## 2.3. Método y enfoque de desarrollo

Una vez analizadas las características de desarrollo de este proyecto, optamos por desarrollar un producto completamente nuevo, partiendo de un proyecto vacío, en base a los beneficios que reporta dicho método:

- Permite añadir todas las particularidades y características en función de las necesidades que presenten los potenciales usuarios que van a hacer uso de la aplicación.
- En el futuro, se pueden realizar modificaciones para añadir nuevas funcionalidades que detecte el Servicio de Relaciones Internacionales de la UAL.
- Se eliminan los condicionantes y/o problemas que pudieran surgir a la hora de realizar la adaptación de un producto ya existente.

La metodología de diseño centrada en el usuario[11] será la opción elegida, enfocando todo el proceso al estudiante Erasmus, cuya información será proporcionada por el Servicio de Relaciones Internacionales de la UAL. Ésta nos dará feedback de la aplicación desarrollada y nos podrá proponer posibles mejoras sobre la experiencia de usuario.

Para el desarrollo de esta aplicación y con el fin de controlar lo máximo posible la evolución del proyecto (desde la fase inicial hasta la fase final), se hará uso de una planificación lo más clara, realista y detallada posible de todas las fases, junto con las tareas por las que estarán compuestas. Esta planificación puede visualizarse en el diagrama que se presenta en el apartado “2.5. Planificación”.

## 2.4. Tecnologías utilizadas

En este apartado se describen las tecnologías utilizadas y su versión.

### 2.4.1. Angular

Angular es un framework opensource desarrollado por Google para facilitar la creación y programación de aplicaciones web de una sola página, las webs SPA (Single Page Application).

Angular separa completamente el frontend y el backend de la aplicación, evita escribir código repetitivo y mantiene cierto orden y estructura gracias a su patrón MVC (Modelo-Vista-Controlador), por lo que el marco de trabajo nos asegura un desarrollo eficiente, que facilita la realización de futuras modificaciones y actualizaciones, con un menor esfuerzo.

Las webs SPA pueden resultar un poco lentas en su primera apertura, pero después nos proporcionan una navegación casi instantánea. Esto es posible debido a que se realiza la carga de toda la página la primera vez que se accede.

Además, Angular es un marco de trabajo modular y escalable basado en el estándar de componentes web, y con un conjunto de interfaz de programación de aplicaciones (API) que nos permite crear nuevas etiquetas HTML personalizadas que pueden ser

reutilizables. El lenguaje principal de programación de Angular es Typescript, y así toda la sintaxis y modo de plantear el código, lo que añade coherencia y consistencia a la información.

Para desarrollar el frontend de este proyecto se ha utilizado Angular en su versión 9.1.11, Angular Cli en su versión 9.1.10 y Typescript en su versión 3.8.3.

### 2.4.2. Firebase

Firebase es una plataforma para el desarrollo de aplicaciones web y aplicaciones móviles desarrollada por Google en 2014. Es una plataforma ubicada en la nube, integrada con Google Cloud Platform, que usa un conjunto de herramientas para la creación y sincronización de proyectos que serán dotados de alta calidad, haciendo posible el crecimiento del número de usuarios [12]. A continuación se nombran los servicios utilizados para acometer este desarrollo:

- **Realtime database:** Una de las herramientas más destacadas y esenciales de Firebase son las bases de datos en tiempo real. Estas se alojan en la nube, son NoSQL y almacenan los datos como JSON. Permiten alojar y disponer de los datos e información de la aplicación en tiempo real.
- **Autenticación de usuarios:** La identificación de los usuarios de una aplicación web es necesaria en la mayoría de los casos si estos quieren acceder a todas sus características. Firebase ofrece un sistema de autenticación que permite el registro propiamente dicho mediante email y contraseña. Así, esta tarea proporciona más seguridad y simplicidad, considerando también que desde aquí se gestionan los accesos y se consigue una mayor protección de los datos. Destacar que Firebase puede guardar en la nube los datos de inicio de sesión con total seguridad, evitando que una persona tenga que identificarse cada vez que abra la aplicación.
- **Hosting:** Ofrece un servidor para alojar y desplegar la aplicación en producción, es decir, proporciona un hosting estático y seguro. Facilita certificados de seguridad SSL y HTTP2 de forma automática y gratuita para cada dominio, reafirmando la seguridad en la navegación. Funciona situándolas en el CDN (Content Delivery Network) de Firebase, una red que recibe los archivos subidos y permite entregar el contenido.

Para desarrollar el backend de este proyecto se ha utilizado firebase en su versión 8.4.3.

### 2.4.3. Visual Studio Code

Visual Studio Code es un editor de código fuente desarrollado por Microsoft para Windows, Linux y macOS. Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código. También es personalizable, por lo que los usuarios pueden cambiar el tema del editor, los atajos de teclado y las preferencias. Es gratuito y de código abierto, aunque

la descarga oficial está bajo software privativo e incluye características personalizadas por Microsoft.

Visual Studio Code se basa en Electron, un framework que se utiliza para implementar Chromium y Node.js como aplicaciones para escritorio, que se ejecuta en el motor de diseño Blink. Aunque utiliza el framework Electron, el software no usa Atom y en su lugar emplea el mismo componente editor (Monaco) utilizado en Visual Studio Team Services (anteriormente llamado Visual Studio Online) [13]

Para el desarrollo de este Proyecto se va a utilizar Visual Studio Code en su versión 1.48.2 con las extensiones de Angular, Bootstrap y TypeScript, cuyo uso nos proporcionan snippets y soporte para el marco de trabajo y el lenguaje utilizado, además de un control sobre los errores de escritura que pudieran producirse.

#### 2.4.4. JSON

JSON (JavaScript Object Notation - Notación de Objetos de JavaScript) es un formato ligero de intercambio de datos. Leerlo y escribirlo es simple para humanos, mientras que para las máquinas es simple interpretarlo y generarlo. Está basado en un subconjunto del Lenguaje de Programación JavaScript, Standard ECMA-262 3rd Edition - Diciembre 1999. JSON es un formato de texto que es completamente independiente del lenguaje pero utiliza convenciones que son ampliamente conocidas. Estas propiedades hacen que JSON sea un lenguaje ideal para el intercambio de datos [5]. JSON está constituido por dos estructuras:

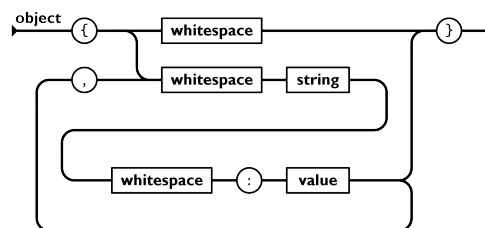


Figura 2.1: JSON: Object

Un objeto es un conjunto desordenado de pares nombre/valor. Un objeto comienza con una llave de apertura «{» y termine con una llave de cierre «}». Cada nombre es seguido por dos puntos «:» y los pares nombre/valor están separados por una coma «,».

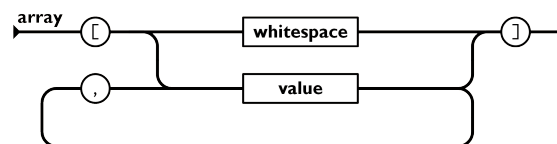


Figura 2.2: JSON: Array

Un arreglo es una colección de valores. Un arreglo comienza con [ corchete izquierdo y termina con ] corchete derecho. Los valores se separan por ,coma.



### 2.4.5. Bootstrap

Bootstrap es un conjunto de herramientas de desarrollo de aplicaciones web, creado por los ex empleados de Twitter Mark Otto y Jacob Thornton. Bootstrap nos proporciona un layout que se adapta a la pantalla del dispositivo utilizado por el usuario.

El framework combina CSS y JavaScript para estilizar los elementos de una página HTML. Esta es una herramienta que proporciona interactividad en la página, por lo que ofrece una serie de componentes que facilitan la comunicación con el usuario, como menús de navegación, controles de página, barras de progreso, entre otras muchas funcionalidades [4].

Además de todas las características que ofrece el framework, su principal objetivo es permitir la construcción de sitios web responsive para dispositivos móviles. Esto significa que las páginas están diseñadas para funcionar en desktop, tablets y smartphones, de una manera muy simple y organizada.

Para nuestro proyecto hemos hecho uso de Bootstrap en su versión 4.4.1

### 2.4.6. Angular Material

Angular Material es una librería de estilos (como Bootstrap) creada por Google y basada en la guía de diseño de Material Design, realizada por el equipo de Angular para integrarse perfectamente con el marco de trabajo de Angular. Angular Material nos proporciona una serie de componentes que enriquecerán mucho el catálogo de interfaces de usuario disponibles al implementar nuestra aplicación.

Para nuestro proyecto hemos hecho uso de Angular Material en su versión 9.2.4

### 2.4.7. GitHub

GitHub es un sistema de gestión de proyectos y control de versiones de código, así como una plataforma de red social diseñada para desarrolladores que nos permite realizar un respaldo de nuestro proyecto y nos da la posibilidad de trabajar en colaboración con personas de todo el mundo, planificar proyectos y realizar un seguimiento del trabajo [3].

- Git: Git es el sistema de control de versiones que nos ayuda a registrar los cambios realizados al código. Además, registra quién realizó los cambios y puede restaurar el código borrado o modificado.
- Hub: El hub de GitHub es lo que convierte una línea de comandos, como Git, en la red social más grande para desarrolladores. Además de contribuir a un determinado proyecto, GitHub le permite a los usuarios socializar con personas de ideas afines. Puedes seguir a personas y ver qué hacen o con quién se conectan.

Para nuestro Proyecto se ha hecho uso de GitHub con la intención de tener un control de versiones, asegurar posibles modificaciones futuras y tener respaldo del desarrollo.

## 2.5. Planificación

Dada la naturaleza académica de este proyecto, se establece un desarrollo en forma de entregas sucesivas, con un horizonte temporal de finalización definido que se inicia el 1 de junio de 2020 y finaliza el 7 de septiembre de 2020. En total hacen unas 14 semanas dedicando un tiempo aproximado de 4 horas diarias hasta completar las 300 horas requeridas, con la intencionalidad de alcanzar los diferentes hitos. A continuación, se muestra un gráfico general con la planificación del proyecto y donde se detalla el horizonte temporal de cada una de ellas teniendo en cuenta la fecha de inicio marcada.

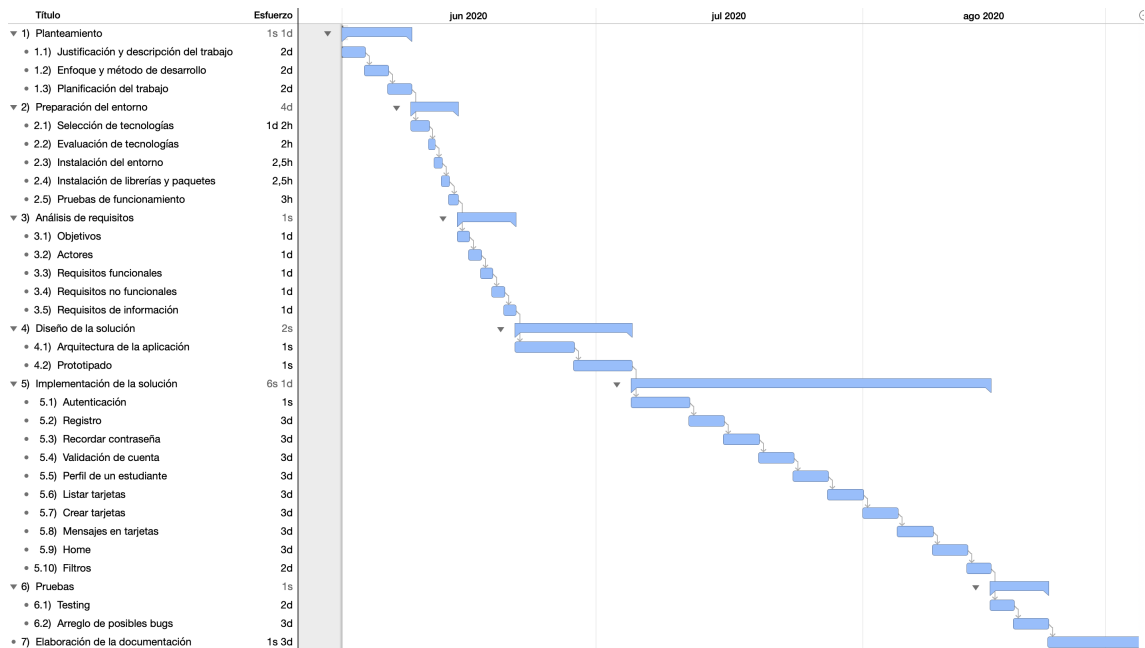


Figura 2.3: Planificación del proyecto



Figura 2.4: Horario de la planificación

- Planteamiento | 26 horas: Iniciamos un proceso de evaluación para describir y justificar el proyecto que se va acometer en este trabajo. Continuaremos plasmando el enfoque de trabajo y método de diseño y desarrollo, finalizando esta fase con la planificación de cada una de las tareas establecidas con una fechas de entrega marcada para asegurar cumplir con los plazos establecidos.

- Preparación del entorno | 13 horas: Continuaremos con la preparación del entorno de trabajo. Este proceso engloba la valoración y selección de las tecnologías más eficientes para acometer el proyecto. Una vez seleccionadas las opciones más adecuadas, procedemos a su instalación. Nos aseguraremos de su funcionamiento realizando una serie de pruebas de uso.
- Análisis de requisitos | 23 horas: Es el primer gran paso para la consecución de nuestra aplicación. Hacemos referencia a la fase de análisis de requisitos, fase estructurada en varias categorías donde describiremos los objetivos que guían el desarrollo de la aplicación, los requisitos funcionales y no funcionales y los requisitos de información.
- Diseño de la aplicación | 46 horas: Planteamiento de la arquitectura y diseño del sistema, así como de la selección del marco de trabajo y prototipado de la aplicación. En esta fase obtendremos una evaluación previa y de bajo coste, al desarrollo del producto final.
- Implementación de la solución | 139 horas: Proceso de implementación del diseño y funcionalidades de la aplicación web, explicando cada uno de los componentes, módulos y servicios que conformarán el sistema.
- Pruebas | 23 horas: Valoración de las posibles pruebas que pueden realizarse en nuestro Proyecto y resultado de los mismos.
- Elaboración de la documentación | 33 horas: Proceso de creación de este documento, donde se recoge todo el proceso seguido y las conclusiones.

## 3 Aplicación ErasmusWay

Para dar solución al planteamiento con el que iniciamos esta memoria y cuya intencionalidad es de la fomentar y promocionar la movilidad académica internacional, se presenta el desarrollo de una aplicación web bautizada con el nombre de ErasmusWay.

La aplicación web va a permitir el registro a todos aquellos usuarios que cumplimenten un formulario, dándoles la posibilidad de compartir la información que ellos deseen respecto a una serie de temas establecidos. Esta información se mostrará en forma de tarjetas, cada una de ellas con una temática distinta y concreta, como son los estudios universitarios, la vivienda o los idiomas. En cada tarjeta se mostrará la información cumplimentada por el usuario, y el resto de usuarios tendrán la posibilidad de enviar mensajes a dicha tarjeta, donde podrán plantear cuestiones que pudieran ser de interés para ellos mismos. Otros estudiantes podrán participar en la conversación e incluso beneficiarse de esta información, llegando al punto de que pueden verse incentivados a iniciar una experiencia Erasmus viendo de primera mano lo que han vivido otros estudiantes.

### 3.1. Objetivos

En la sección de objetivos expondremos el hilo conductor que debe contener la aplicación. Los objetivos van a estar referenciados por una breve descripción y la importancia que tiene en el desarrollo.

#### 3.1.1. Objetivo 1: Interfaz minimalista

- La aplicación debe mostrar una interfaz simple, intuitiva y llamativa que incentive su uso.
- Importancia: Alta

#### 3.1.2. Objetivo 2: Registro de usuarios

- La aplicación debe permitir el registro de usuarios sin la validación de ningún estamento de la Universidad. Es una aplicación de libre uso para que los estudiantes Erasmus puedan tratar consultas y cuestiones entre ellos liberando al Servicio de Relaciones Internacionales de alguna de estas tareas.
- Importancia: Muy alta

### 3.1.3. Objetivo 3: Autenticación de usuarios

- La aplicación debe permitir la autenticación de los usuarios proporcionando la seguridad necesaria para ello.
- Importancia: Muy alta

### 3.1.4. Objetivo 4: Recordar contraseña

- Establecer un protocolo de seguridad para recuperar la contraseña de manera autónoma sin la intervención de la Universidad.
- Importancia: Muy alta

### 3.1.5. Objetivo 5: Soporte multiplataforma

- La aplicación debe funcionar en distintos dispositivos independientemente de su sistema operativo y de su resolución.
- Importancia: Alta

### 3.1.6. Objetivo 6: Comunicación en tiempo real

- La aplicación debe funcionar en la nube estableciendo una comunicación casi instantánea.
- Importancia: Alta

### 3.1.7. Objetivo 7: Gestión de información de estudiantes

- La aplicación debe permitir a los estudiantes compartir información de distintas temáticas con el objetivo de prestar asesoramiento a otros estudiantes menos experimentados o que busquen información para realizar una experiencia Erasmus.
- Importancia: Alta

### 3.1.8. Objetivo 8: Permitir enviar mensajes

- La aplicación debe permitir a los estudiantes poder enviar mensajes a las tarjetas de información de otros estudiantes. Los primeros tendrán la opción de responder en la misma tarjeta. La información será pública para ayudar a otros alumnos.
- Importancia: Alta

### 3.1.9. Objetivo 9: Permitir votación en las tarjetas de información

- La aplicación debe permitir a los estudiantes poder realizar votaciones sobre las tarjetas de otros estudiantes. La información será pública.
- Importancia: Alta

## 3.2. Diagrama de casos de uso

A continuación se representa el diagrama de casos de uso donde se puede visualizar de forma gráfica a los actores que van a participar y al flujo que puede seguir la aplicación.

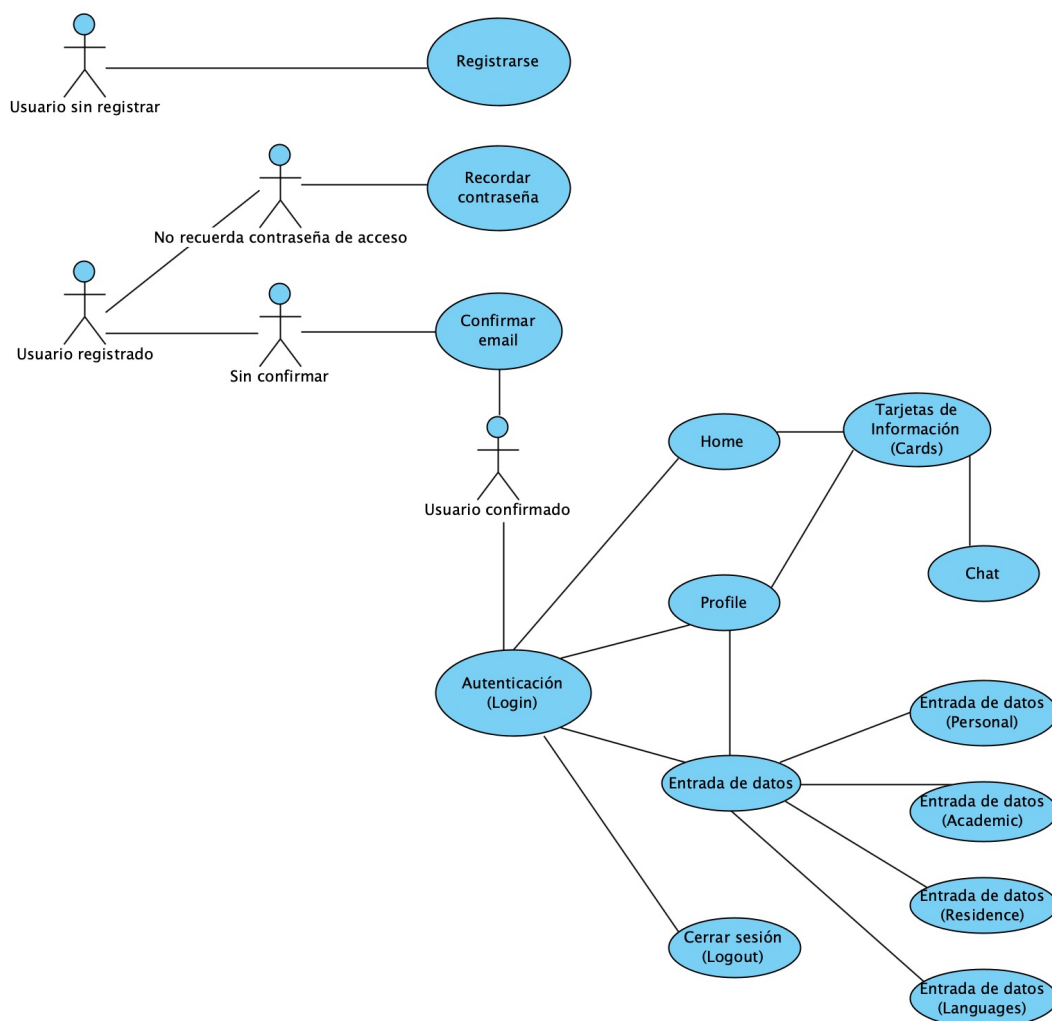


Figura 3.1: Diagrama de casos de uso

### 3.3. Actores

En la sección de actores mostramos a los distintos actores que pueden participar de forma activa en la aplicación.

#### 3.3.1. Actor-01: Usuario sin registrar

- El usuario no puede acceder a la aplicación por carecer de registro.
- |                    |       |             |
|--------------------|-------|-------------|
| Requisito asociado | RF-01 | Registrarse |
|--------------------|-------|-------------|

#### 3.3.2. Actor-02: Usuario registrado: Sin confirmar

- Usuario que ha cumplimentado el proceso de registro y está a la espera de confirmar su registro mediante el acceso a un enlace tras la recepción de un correo de confirmación.
- |                    |       |               |
|--------------------|-------|---------------|
| Requisito asociado | RF-02 | Confirmación  |
|                    | RF-03 | Autenticación |

#### 3.3.3. Actor-03: Usuario registrado: Confirmado

- Usuario que ha cumplimentado el proceso de registro, ha recibido un correo electrónico de confirmación y ha accedido pulsando el enlace.
- |                    |       |               |
|--------------------|-------|---------------|
| Requisito asociado | RF-02 | Confirmación  |
|                    | RF-03 | Autenticación |

#### 3.3.4. Actor-04: Usuario que no recuerda su contraseña de acceso

- Usuario registrado que no recuerda su contraseña de acceso y ha iniciado el proceso de cambiar la contraseña.
- |                    |       |                     |
|--------------------|-------|---------------------|
| Requisito asociado | RF-04 | Recordar contraseña |
|--------------------|-------|---------------------|

#### 3.3.5. Actor-05: Estudiante

- Usuario que se ha autenticado en la aplicación y tiene acceso a cumplimentar sus tarjetas de información, enviar y recibir mensajes y filtrar las tarjetas de otros estudiantes.
- |                    |       |                                      |
|--------------------|-------|--------------------------------------|
| Requisito asociado | RF-05 | Validación                           |
|                    | RF-06 | Ver información de otros estudiantes |
|                    | RF-07 | Insertar información personal        |
|                    | RF-08 | Enviar/recibir Mensajes              |
|                    | RF-09 | Filtrar información                  |
|                    | RF-10 | Cerrar sesión                        |

## 3.4. Requisitos funcionales

En la sección de actores ya hemos establecido la relación establecida entre los actores y los requisitos de la aplicación. Ahora vamos a entrar a conocer con más detalle los requisitos funcionales del sistema.

### 3.4.1. RF-01: Registrarse

Los estudiantes que no dispongan de una cuenta en ErasmusWay, podrán crear una con solo introducir los datos solicitados en la sección de Registro.

- |                |          |                       |
|----------------|----------|-----------------------|
| Actor asociado | Actor-01 | Usuario sin registrar |
|----------------|----------|-----------------------|
- |               |              |
|---------------|--------------|
| Pre-Condición | No existente |
|---------------|--------------|
- |           |        |                                     |
|-----------|--------|-------------------------------------|
| Secuencia | Paso 1 | Pulsar en «Register as new Erasmus» |
|           | Paso 2 | Introducir los datos requeridos     |
|           | Paso 3 | Pulsar en «Register»                |
- |           |   |
|-----------|---|
| Resultado | Se envía un correo electrónico a la dirección introducida en el formulario de registro y se registra una nueva cuenta |
|-----------|---|
- |           |      |   |
|-----------|------|---|
| Excepción | E-01 | El email es único para cada estudiante                      |
|           | E-02 | Todos los datos del formulario de registro son obligatorios |
|           | E-03 | La contraseña debe de contener mínimo 6 caracteres          |
|           | E-04 | Hay que aceptar los términos de uso                         |

### 3.4.2. RF-02: Confirmación

Una vez finalizado el proceso de cumplimentar los datos de registro, el estudiante recibirá un correo electrónico en la dirección indicada cuya funcionalidad es la de confirmar la cuenta de correo electrónico.

- |                |          |                                   |
|----------------|----------|-----------------------------------|
| Actor asociado | Actor-02 | Usuario registrado: Sin confirmar |
|----------------|----------|-----------------------------------|
- |               |  |
|---------------|--|
| Pre-Condición | El usuario debe cumplimentar el formulario de registro (RF-01) |
|---------------|--|
- |           |        |   |
|-----------|--------|---|
| Secuencia | Paso 1 | Recepción de correo electrónico de confirmación           |
|           | Paso 2 | Pulsar enlace   |
|           | Paso 3 | Recibe notificación en pantalla de confirmación de cuenta |
- |           |   |
|-----------|---|
| Resultado | Se actualiza la cuenta al estado de confirmado y permite su autenticación |
|-----------|---|
- |           |   |         |
|-----------|---|---------|
| Excepción | - | Ninguna |
|-----------|---|---------|



### 3.4.3. RF-03: Autenticación

Todas las cuentas confirmadas podrán autenticarse mediante su correo electrónico y contraseña.

- |                |          |                                |
|----------------|----------|--------------------------------|
| Actor asociado | Actor-03 | Usuario registrado: Confirmado |
|----------------|----------|--------------------------------|
- |               |   |
|---------------|---|
| Pre-Condición | La cuenta del usuario debe estar confirmada (RF-02) |
|---------------|---|
- |           |        |  |
|-----------|--------|--|
| Secuencia | Paso 1 | Acceder a la sección de autenticación de la aplicación |
|           | Paso 2 | Introducir «email» y «password»                        |
|           | Paso 3 | Pulsar en «Sign In»                                    |
- |           |   |
|-----------|---|
| Resultado | El usuario es redirigido a la página inicial de la aplicación |
|-----------|---|
- |           |      |                             |
|-----------|------|-----------------------------|
| Excepción | E-01 | El email no existe          |
|           | E-02 | La contraseña es incorrecta |

### 3.4.4. RF-04: Recordar contraseña

Estudiante con cuenta registrada y confirmada que no recuerda su contraseña de acceso.

- |                |          |   |
|----------------|----------|---|
| Actor asociado | Actor-04 | Usuario que no recuerda su contraseña de acceso |
|----------------|----------|---|
- |               |   |
|---------------|---|
| Pre-Condición | La cuenta del usuario debe estar confirmada (RF-02) |
|---------------|---|
- |           |        |                                      |
|-----------|--------|--------------------------------------|
| Secuencia | Paso 1 | Pulsar en «I forgot my password»     |
|           | Paso 2 | Introducir «email»                   |
|           | Paso 3 | Pulsar en «Request new password»     |
|           | Paso 4 | Recepción de correo electrónico      |
|           | Paso 5 | Pulsar enlace del correo electrónico |
|           | Paso 6 | Introducir la nueva contraseña       |
|           | Paso 7 | Pulsar en «New password»             |
- |           |  |
|-----------|--|
| Resultado | El usuario recibirá una notificación en pantalla que confirmará que la contraseña ha sido modificada correctamente |
|-----------|--|
- |           |      |                                     |
|-----------|------|-------------------------------------|
| Excepción | E-01 | El campo email no puede estar vacío |
|           | E-02 | El email no existe                  |

### 3.4.5. RF-05: Validación

Validación que se produce en cada pantalla de la aplicación y en la que se comprueba si el acceso del usuario es válido.

- |                |          |            |
|----------------|----------|------------|
| Actor asociado | Actor-05 | Estudiante |
|----------------|----------|------------|
- |               |  |
|---------------|--|
| Pre-Condición | El usuario ha accedido a la aplicación con unas credenciales correctas (RF-03) |
|---------------|--|
- |           |            |                             |
|-----------|------------|-----------------------------|
| Secuencia | Paso único | Acceso a cualquier pantalla |
|-----------|------------|-----------------------------|
- |           |   |
|-----------|---|
| Resultado | Si el usuario es válido, nos permite la navegación. Si el usuario no es válido, lo expulsa de la aplicación y lo redirige a la pantalla de autenticación. |
|-----------|---|
- |           |   |         |
|-----------|---|---------|
| Excepción | - | Ninguna |
|-----------|---|---------|

### 3.4.6. RF-06: Ver información de otros estudiantes

Listado de todas las tarjetas que contienen información de otros estudiantes.

- |                |          |            |
|----------------|----------|------------|
| Actor asociado | Actor-05 | Estudiante |
|----------------|----------|------------|
- |               |  |
|---------------|--|
| Pre-Condición | El usuario está autenticado con unas credenciales correctas (RF-03) y válidas (RF-5) |
|---------------|--|
- |           |            |  |
|-----------|------------|--|
| Secuencia | Paso único | Acceso a la pantalla principal (/home) |
|-----------|------------|--|
- |           |  |
|-----------|--|
| Resultado | Se muestra un listado clasificado por categorías, con todas las fichas de los estudiantes. |
|-----------|--|
- |           |      |  |
|-----------|------|--|
| Excepción | E-01 | Si no existen fichas cumplimentadas, se muestra una lista vacía y el contador de las fichas a 0. |
|-----------|------|--|

### 3.4.7. RF-07: Visualizar información estudiante (perfil)

Visualización de las tarjetas que contienen la información del estudiante autenticado.

- |                |          |            |
|----------------|----------|------------|
| Actor asociado | Actor-05 | Estudiante |
|----------------|----------|------------|
- |               |  |
|---------------|--|
| Pre-Condición | El usuario está autenticado con unas credenciales correctas (RF-03) y válidas (RF-5) |
|---------------|--|
- |           |            |  |
|-----------|------------|--|
| Secuencia | Paso único | Acceso a la pantalla de perfil (/home/profile) |
|-----------|------------|--|

- |           |   |  |
|-----------|---|--|
| Resultado | Se muestran todas las fichas del estudiante autenticado |  |
|-----------|---|--|
- |           |      |  |
|-----------|------|--|
| Excepción | E-01 | Si no existen fichas cumplimentadas, se muestra las fichas indicando que la información está pendiente de cumplimentar |
|-----------|------|--|

### 3.4.8. RF-08: Insertar información personal

Acceso a cada una de las secciones de «About me», donde existe la posibilidad de cumplimentar la información que a posteriori aparecerá en las fichas personales (RF-07).

- |                |          |            |
|----------------|----------|------------|
| Actor asociado | Actor-05 | Estudiante |
|----------------|----------|------------|
- |               |  |  |
|---------------|--|--|
| Pre-Condición | El usuario está autenticado con unas credenciales correctas (RF-03) y válidas (RF-5) |  |
|---------------|--|--|
- |           |        |  |
|-----------|--------|--|
| Secuencia | Paso 1 | Acceso a cualquier sección de «About me»               |
|           | Paso 2 | Introducir la información en los campos del formulario |
|           | Paso 3 | Pulsar en Save   |
- |           |  |  |
|-----------|--|--|
| Resultado | Se muestra un formulario paso a paso para introducir la información de la ficha seleccionada. Al guardar nos redirige a la pantalla del perfil |  |
|-----------|--|--|
- |           |      |   |
|-----------|------|---|
| Excepción | E-01 | Formulario paso a paso                  |
|           | E-02 | Validación de los campos del formulario |

### 3.4.9. RF-09: Enviar/recibir mensajes

Acceso a una tarjeta de información personal de cualquier estudiante para enviar un mensaje.

- |                |          |            |
|----------------|----------|------------|
| Actor asociado | Actor-05 | Estudiante |
|----------------|----------|------------|
- |               |  |  |
|---------------|--|--|
| Pre-Condición | El usuario está autenticado con unas credenciales correctas (RF-03), son válidas (RF-5) y está visualizando una tarjeta (RF-6) (RF-07) |  |
|---------------|--|--|
- |           |        |  |
|-----------|--------|--|
| Secuencia | Paso 1 | Acceso a una tarjeta de información (/home/card/id/type) |
|           | Paso 2 | Escribir un mensaje en el lugar habilitado para ello     |
|           | Paso 3 | Pulsar en el botón «Send»                                |

- |           |  |
|-----------|--|
| Resultado | Se muestra la información de la ficha del estudiante seleccionado junto con su panel de mensajes |
|-----------|--|
- |           |      |                           |
|-----------|------|---------------------------|
| Excepción | E-01 | Validación de campo vacío |
|-----------|------|---------------------------|

### 3.4.10. RF-10: Votar una tarjeta

Acceso a una tarjeta de información personal de cualquier estudiante para emitir un voto.

- |                |          |            |
|----------------|----------|------------|
| Actor asociado | Actor-05 | Estudiante |
|----------------|----------|------------|
- |               |  |
|---------------|--|
| Pre-Condición | El usuario está autenticado con unas credenciales correctas (RF-03), son válidas (RF-5) y está visualizando una tarjeta (RF-6) (RF-07) |
|---------------|--|
- |           |        |  |
|-----------|--------|--|
| Secuencia | Paso 1 | Acceso a una tarjeta de información (/home/card/id/type) |
|           | Paso 2 | Pulsar el botón «Vote this card»                         |
- |           |  |
|-----------|--|
| Resultado | Recepción en pantalla de una notificación confirmando el voto y aumento del contador de votos de la tarjeta valorada |
|-----------|--|
- |           |      |   |
|-----------|------|---|
| Excepción | E-01 | Un estudiante no puede votarse a sí mismo |
|-----------|------|---|

### 3.4.11. RF-11: Filtrar información por país

Posibilidad de filtrar la información de las fichas mediante un filtro por país.

- |                |          |            |
|----------------|----------|------------|
| Actor asociado | Actor-05 | Estudiante |
|----------------|----------|------------|
- |               |   |
|---------------|---|
| Pre-Condición | El usuario está autenticado con unas credenciales correctas (RF-03), son válidas (RF-5) y se encuentra en la página principal (/home) |
|---------------|---|
- |           |        |                    |
|-----------|--------|--------------------|
| Secuencia | Paso 1 | Selección del país |
|           | Paso 2 | Pulsar en «Filter» |
- |           |  |
|-----------|--|
| Resultado | Se muestran la información de las fichas de todos los estudiantes del país seleccionado en el filtro |
|-----------|--|
- |           |      |                           |
|-----------|------|---------------------------|
| Excepción | E-01 | Validación de campo vacío |
|-----------|------|---------------------------|

### 3.4.12. RF-12: Cerrar sesión

Cerrar sesión en la aplicación.

- |                |          |            |
|----------------|----------|------------|
| Actor asociado | Actor-05 | Estudiante |
|----------------|----------|------------|
- |               |  |
|---------------|--|
| Pre-Condición | El usuario está autenticado con unas credenciales correctas (RF-03), son válidas (RF-5) y pulsa en el botón Logout |
|---------------|--|
- |           |            |                              |
|-----------|------------|------------------------------|
| Secuencia | Paso único | Selección del botón «Logout» |
|-----------|------------|------------------------------|
- |           |  |
|-----------|--|
| Resultado | Elimina las credenciales de acceso y nos redirige a la página de autenticación |
|-----------|--|
- |           |   |         |
|-----------|---|---------|
| Excepción | - | Ninguna |
|-----------|---|---------|

## 3.5. Requisitos no funcionales

Además de los requisitos listados anteriormente y que deben conformar el patrón de diseño de la aplicación, esta debe contener otros requisitos que no siendo tan importantes, son necesarios para el éxito del diseño.

### 3.5.1. RNF-01: Diseño minimalista

La aplicación va a ser utilizada por gente joven, por lo que debe tener un diseño sencillo y moderno.

### 3.5.2. RNF-02: Tiempos de carga reducidos

La aplicación debe permitir un acceso a los datos casi instantáneo que nos permita una navegación rápida y fluida.

### 3.5.3. RNF-03: La información debe mostrarse de forma clara

Debe mostrar los datos de forma clara, concisa y directa.

### 3.5.4. RNF-04: Debe de estar en inglés

El idioma predefinido de la aplicación será el inglés.

### 3.5.5. RNF-05: Seguridad

El acceso y comunicación al servidor debe ser seguro a través de HTTPS.

## 3.6. Requisitos de información

Los requisitos de información son aquellos datos que directamente se almacenan en el dispositivo con el que se accede a la aplicación. En lo que concierne al proyecto recogido en esta memoria, toda la gestión y almacenamiento de los datos van a estar delegados al servicio web al que la aplicación va a estar conectado, y que actuará como backend de esta. Excepcionalmente, en el dispositivo de acceso se van a almacenar tres datos:

- Token: Token de acceso a la aplicación.
- Expiración del token: Tiempo de expiración del token. Por defecto será de 60 minutos. Una vez pasado este tiempo, nos expulsará de la aplicación y nos redirigirá a la página de autenticación.
- Correo (opcional): Si se marca la casilla «remember», el correo quedará guardado para no tener que volver a escribirlo al realizar el login.

# 4 Diseño de la solución

Damos inicio al proceso de diseño de la solución mediante la exposición de un prototipo de alta fidelidad. Un prototipo es una representación visual de la aplicación que nos permite comunicar decisiones de diseño, con la finalidad de realizar una evaluación previa al desarrollo del producto final. Su versatilidad hace que sea sencillo y económico introducir modificaciones y mejoras en el diseño.

## 4.1. Prototipado

A continuación, vamos a presentar todas las pantallas en su versión prototipo que contendrá nuestra aplicación web, explicando y detallando los elementos principales que van a conformar el diseño.

### 4.1.1. Login de usuarios

- Acceso a usuarios sin autenticar y autenticados.
- Proceso de autenticación y validación para acceder a la aplicación.
- Posibilidad de navegar al Registro (4.1.2), Recordar contraseña (4.1.3) y a la Página principal (Home) tras realizar el login (4.1.4).

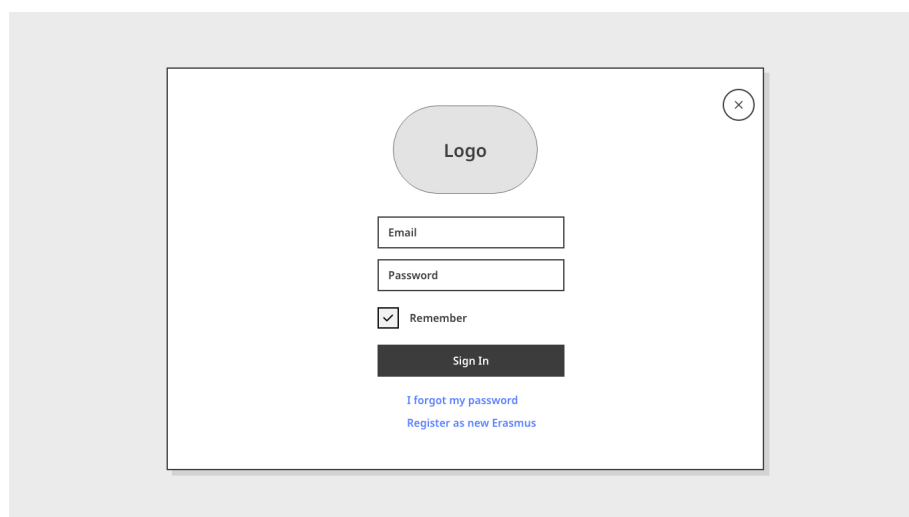


Figura 4.1: Prototipo: Login

### 4.1.2. Registro

- Acceso a usuarios sin autenticar y autenticados.
- Proceso de registro de un nuevo usuario en la aplicación.
- Posibilidad de navegar al Login (4.1.1).

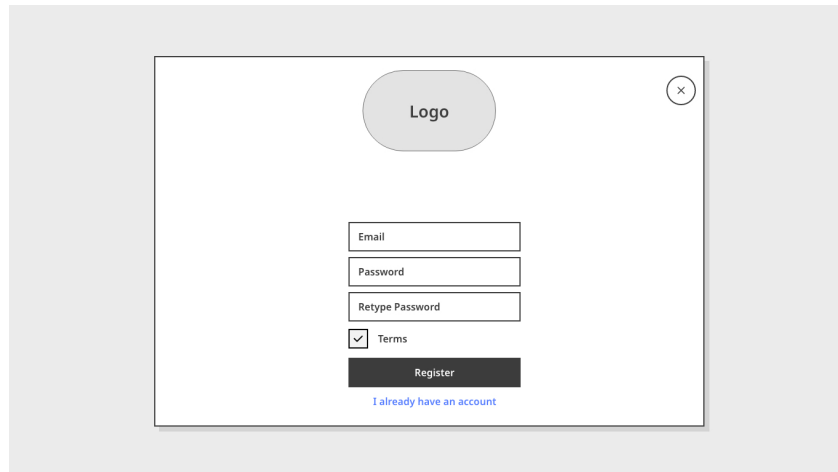
Un prototipo de pantalla de registro. En la parte superior hay un botón ovalado con el texto "Logo" y un ícono de cerrar (X) en la esquina superior derecha. Debajo hay tres campos de texto: "Email", "Password" y "Retype Password". A continuación hay un campo de selección con un ícono de casilla marcada y el texto "Terms". Debajo de eso hay un botón rectangular con el texto "Register". En la parte inferior hay un enlace azul que dice "I already have an account".

Figura 4.2: Prototipo: Registro

### 4.1.3. Recordar contraseña

- Acceso a usuarios sin autenticar y autenticados.
- Proceso para cambiar la contraseña.
- Posibilidad de navegar al Login (4.1.1) y al Registro (4.1.2).

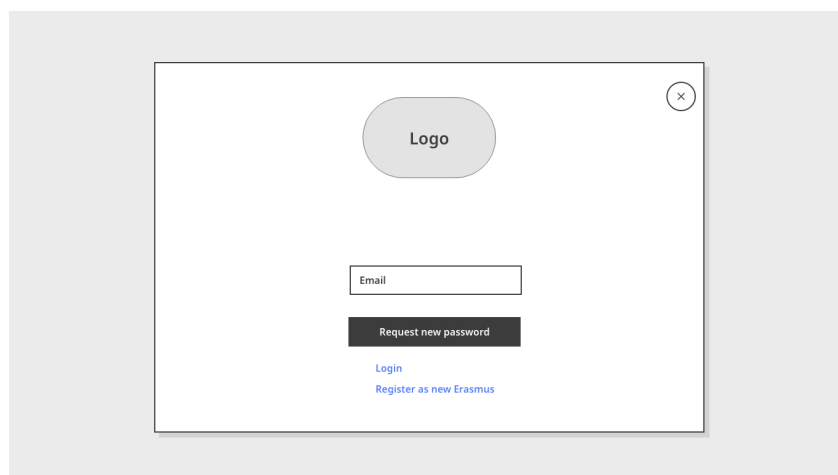
Un prototipo de pantalla de "Recordar contraseña". En la parte superior hay un botón ovalado con el texto "Logo" y un ícono de cerrar (X) en la esquina superior derecha. Debajo hay un campo de texto "Email". Debajo de eso hay un botón rectangular con el texto "Request new password". En la parte inferior hay dos enlaces azules: "Login" y "Register as new Erasmus".

Figura 4.3: Prototipo: Recordar contraseña



#### 4.1.4. Home

- Acceso exclusivo a usuarios autenticados.
- Menú principal que contiene las tarjetas de información de los estudiantes.
- Posibilidad de navegar a Erasmus Profile (4.1.5), las Páginas de entrada de datos y las Tarjetas (4.1.7) con su servicio de Mensajes (4.1.8).

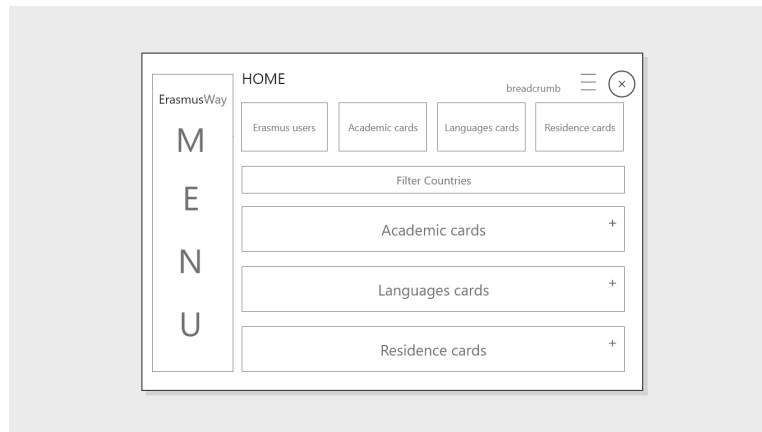


Figura 4.4: Prototipo: Home

#### 4.1.5. Erasmus profile

- Acceso exclusivo a usuarios autenticados.
- Menú que contiene toda la información personal del estudiante autenticado.
- Posibilidad de navegar a Erasmus Profile (4.1.5), las Páginas de entrada de datos y sus propias Tarjetas (4.1.7) con su servicio de Mensajes (4.1.8).

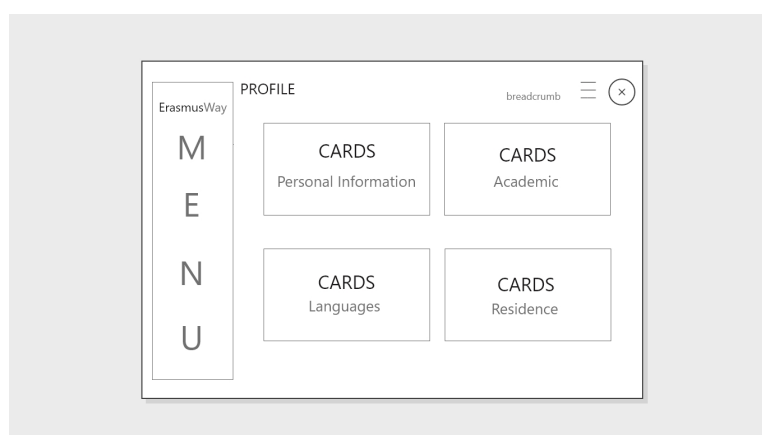


Figura 4.5: Prototipo: Profile

### 4.1.6. Páginas de entrada de datos

- Acceso exclusivo a usuarios autenticados.
- La aplicación contendrá cuatro páginas distintas para introducir datos. Las enumeramos a continuación:
  - Personal: Información personal del estudiante.
  - Academic: Información académica del estudiante.
  - Languages: Información sobre el nivel de idiomas del estudiante.
  - Residence: Información y experiencia adquirida durante su experiencia Erasmus sobre vivienda.
- Formulario paso a paso de entrada de datos del estudiante autenticado. Antes de guardar se mostrará un resumen de todos los datos cumplimentados y redirigirá a la Página de Erasmus profile (4.1.5).
- Posibilidad de navegar a Erasmus Profile (4.1.5), las Páginas de entrada de datos y sus propias Tarjetas (4.1.7) con su servicio de Mensajes (4.1.8).

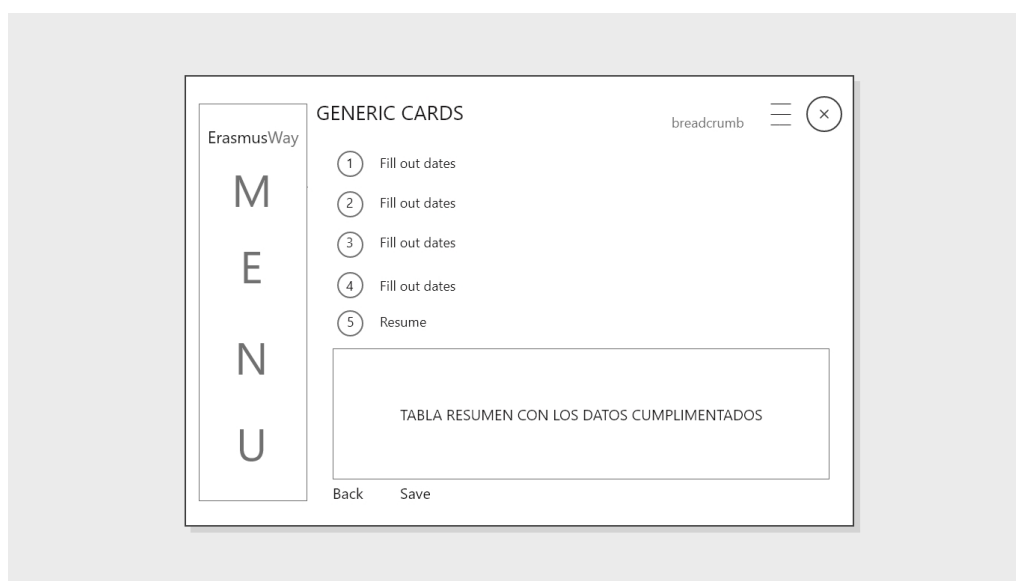


Figura 4.6: Prototipo: Página de entrada de datos

### 4.1.7. Tarjetas de datos

- Acceso exclusivo a usuarios autenticados.
- Tarjeta que muestra la información de un estudiante. Estas tarjetas van a contener la información que cada estudiante ha añadido en las Páginas de entrada de datos.
- Posibilidad de navegar a Home (4.1.4) y al resto de Páginas de entrada de datos (4.1.6), así como visualizar la Tarjeta y el tablón de envío de Mensajes (4.1.8).

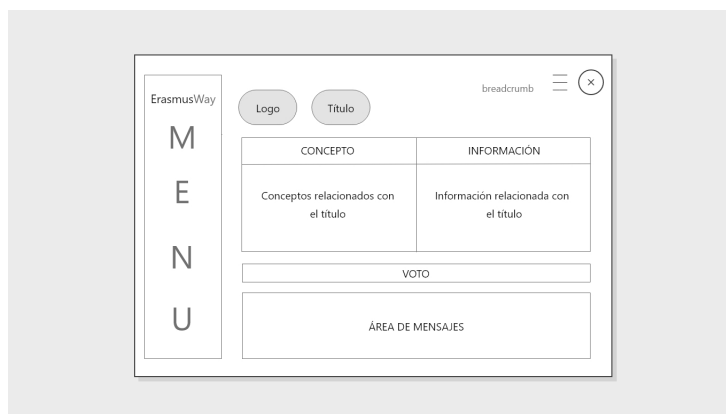


Figura 4.7: Prototipo: Tarjeta de datos

### 4.1.8. Mensajes

- Acceso exclusivo a usuarios autenticados.
- Tablón contenido dentro de las tarjetas de información que muestra los mensajes que se han enviado a ese estudiante y tarjeta.
- Posibilidad de navegar a Home (4.1.4) y al resto de Páginas de entrada de datos.

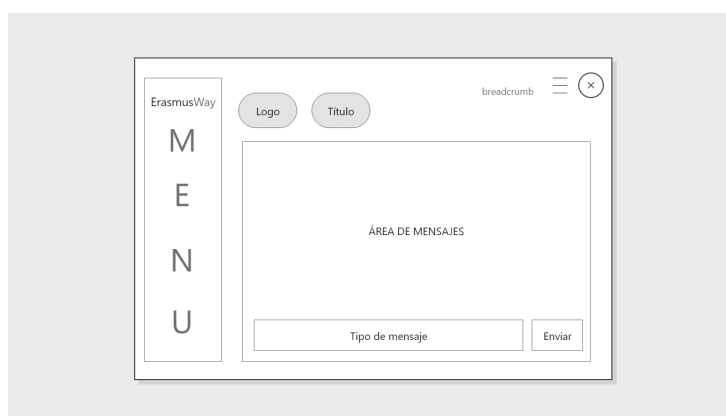


Figura 4.8: Prototipo: Mensajes

## 4.2. Diseño de la arquitectura

### 4.2.1. Arquitectura general

El proyecto expuesto en esta memoria va a utilizar una arquitectura cliente-servidor. Por un lado encontraremos el cliente, que va a ser nuestra aplicación web desarrollada con el marco de trabajo Angular y que será compatible con cualquier dispositivo y resolución. Por otro lado tendremos al servidor que construiremos haciendo uso del servicio de Google Firebase. Utilizaremos Firebase para implementar el sistema de autenticación de usuarios, así como una base de datos en tiempo real para trabajar con los datos de los estudiantes y el envío de mensajes. Además, utilizaremos un pequeño almacenamiento local en el dispositivo de acceso para almacenar información relativa a la sesión del usuario.

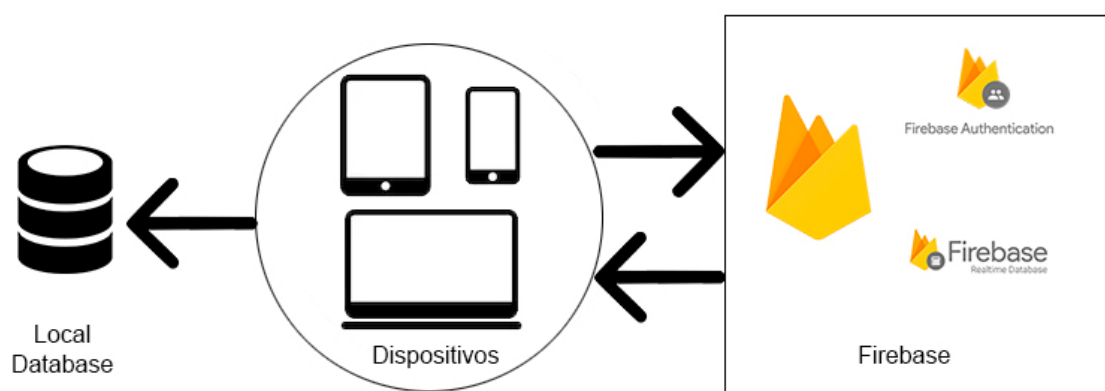


Figura 4.9: Arquitectura general

### 4.2.2. Arquitectura de la aplicación

En cuanto a la aplicación, vamos hacer uso de la arquitectura de Angular, un marco de trabajo que proporciona un patrón de diseño análogo al patrón MVC.

Mediante el uso del mismo, generaremos una vista y controlaremos la lógica de esta mediante el uso de componentes que exportaremos como clases. Además, agregamos lógica con los servicios, encargados de manejar los datos que nuestra aplicación utilizará. Finalmente “encapsulamos” todos componentes y servicios creados en módulos.

Tras ello y como nos indica la documentación de Angular, iniciamos nuestra aplicación mediante el bootstrapping de nuestro root module. Angular toma el control y nos muestra el contenido en el navegador, reaccionando a la interacción con los usuarios/estudiantes que utilicen la aplicación ErasmusWay y de acuerdo a las instrucciones que codificamos en nuestra lógica.

La aplicación generada contará con un root module (AppModule), el cual provee el mecanismo de arranque que inicia nuestra aplicación. Un módulo puede importar funcionalidades de otros módulos, y exportar sus propias funcionalidades. En nuestro proyecto dispondremos de multitud de módulos, separados con la intención de crear un código más limpio y fácil de mantener, así como de disponer de la posibilidad de aplicar lazy loading (carga perezosa) cuando lo creamos necesario.

La aplicación estará compuesta por multitud de componentes, donde cada componente definirá a una clase que contendrá datos y lógica, y que estará vinculada a nuestro template HTML. Antes de que se muestre una vista, Angular evaluará las directivas (lógica) y resuelve la sintaxis del data binding (enlace entre lógica y vista) en el template para modificar los elementos HTML y el DOM, según los datos y la lógica de nuestra aplicación. Angular cuenta con two-way data binding, que significa que los cambios en el DOM también se reflejan en nuestros datos.

Por último, citamos a una de las partes más importantes de este ecosistema, los servicios. Un servicio hace referencia a todos los datos o lógica que no están asociados directamente a una vista y que necesitaremos utilizar en diferentes partes de la aplicación y entre diferentes componentes. Tal como un componente, los servicios son exportados como clases. Los servicios nos permiten ser inyectados en otros componentes como dependencias.

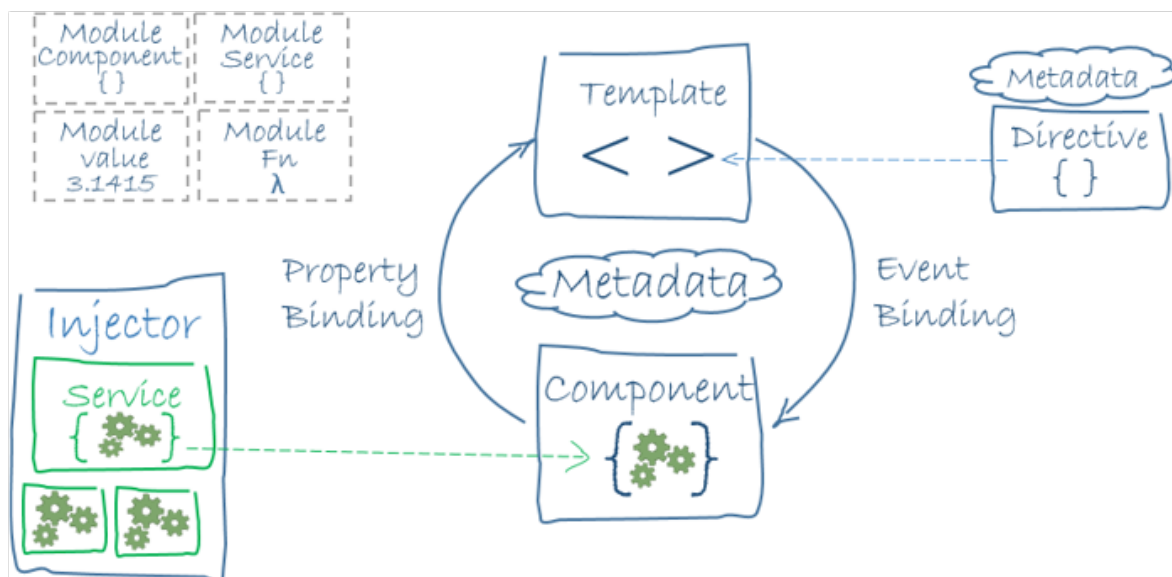


Figura 4.10: Arquitectura de la aplicación [2]

### 4.2.3. Servicio de autenticación

Para realizar el proceso de autenticación con los protocolos de seguridad que hemos establecido, vamos a usar el servicio Firebase Auth [6]. Este es un servicio con el que podemos realizar el proceso de autenticar a los usuarios utilizando únicamente código del lado del cliente mediante el método clásico de inicio de sesión introduciendo el correo electrónico y la contraseña. Además, tenemos a nuestra disposición un sistema de administración de usuarios a través del cual podemos realizar una supervisión de los usuarios almacenados.

### 4.2.4. Modelo de datos

academics	
aboutStudies	string
all_comments	[all_comments]
author	{author}
branch	{branch}
course	{course}
degree	{degree}
idStudent	string
votes	number

languages	
all_comments	[all_comments]
author	{author}
spanish	{spanish}
english	{english}
french	{french}
othersLanguage	string
idStudent	string
votes	number

residences	
aboutHome	string
all_comments	[all_comments]
area	{area}
idStudent	string
place	{place}
price	{price}
votes	number

author	
email	string
experience	boolean
img	string
lastname	string
name	string
country	{country}

all_comments	
author	{author}
date	string
text	string

branch		course		degree		country		area	
id	string	id	string	id	string	id	string	id	string
name	string	name	string	name	string	name	string	name	string

place	
id	string
name	string

price	
id	string
name	string

spanish	
listening	number
reading	number
speaking	number
reading	number

english	
listening	number
reading	number
speaking	number
reading	number

french	
listening	number
reading	number
speaking	number
reading	number

students	
aboutYou	string
dateOfBirth	date
email	string
experience	boolean
idToken	string
img	string
lastname	string
name	string
method	string
country	{country}
needConfirm	boolean
profileComplete	number
registerDate	date
role	number
terms	boolean

#### 4.2.5. Base de datos en tiempo real

Para el almacenamiento en la base de datos, vamos hacer uso del servicio de base de datos en tiempo real de Firebase conocido como «Realtime Database» [7]. Es un servicio alojado en la nube donde los datos se almacenan en formato JSON y se sincronizan en tiempo real con cada cliente conectado.

Firebase nos proporciona un lenguaje flexible de reglas basadas en expresiones, llamado reglas de seguridad. Nos permite definir la estructura de los datos y en qué momento se podrán leer o escribir.

Realtime Database es una base de datos NoSQL y, como tal, tiene diferentes optimizaciones y funcionalidades en comparación con una base de datos relacional. La API de Realtime Database está diseñada para permitir solo operaciones que se puedan ejecutar rápidamente. Este hecho nos permite crear una excelente experiencia de tiempo real que puede servir a millones de usuarios sin afectar a la capacidad de respuesta.

# 5 Implementación de la solución

El objetivo de esta aplicación (de nombre ErasmusWay) es poder implementar un sistema que proporcione a los estudiantes la posibilidad de compartir información de su experiencia Erasmus sobre temáticas relevantes y predefinidas en el sistema mediante el formato de tarjetas. Esta información puede ser de gran ayuda para aquellos alumnos que estén planteándose la posibilidad de vivir su experiencia de estudios Erasmus en la Universidad de Almería. La intencionalidad de este proyecto es que se cree un diálogo entre los estudiantes que los incentive mediante la obtención de información y promocióne a la Universidad de Almería y su ciudad.

Para implementar la aplicación, previamente hemos definido los requisitos y funcionalidades, así como la interfaz deseada y su arquitectura. En esta etapa recogemos el testigo de todo lo anterior y procedemos a iniciar la implementación y pruebas del desarrollo. Para ello se ha optado por utilizar un marco de trabajo basado en Angular para desarrollar el cliente (la aplicación web) y se utilizará Firebase como backend como servicio, tanto para la gestión de usuarios como para la base de datos.

## 5.1. Estructura del proyecto

Angular nos aporta un sistema de trabajo pero deja totalmente a nuestro criterio la estructuración del proyecto. Por ello, en este apartado se pretende mostrar la estructura que se ha creado y las razones por la cuales se ha hecho de esta manera.

Toda la aplicación se ha ido generando desde un proyecto Angular nuevo y vacío. Las carpetas y archivos que hemos ido creando se encuentran dentro de la carpeta «src/app». Vamos a explicar las principales carpetas del directorio.

- Auth: Carpeta donde se encuentra toda la vista y lógica del proceso de autenticación, registro y recordar contraseña.
- Components: Componentes que vamos a utilizar para inyectar en otros componentes. Los llamaremos «componentes hijo».
- Guards: Verificación de acceso a las rutas de usuarios con permisos, en nuestro caso, usuarios autenticados.
- Models: Clases.
- Pages: Páginas de la aplicación.
- Pipes: Pequeñas herramientas utilizadas para modificar el formato de los datos.



- Services: Servicios que vamos a inyectar en otros módulos y componentes, generalmente para trabajar con los datos de la base de datos o para aumentar la lógica de nuestro controlador.
- Shared: Componentes compartidos y visualizados en todas las páginas.
- Routing: Enrutamiento de un módulo.
- Assets: Archivos css, js e imágenes.
- Enviroments: Variables de entorno.
- Style.css: Archivo general con los estilos de la aplicación. Cada componente puede tener su propio css que solo afecta a este.

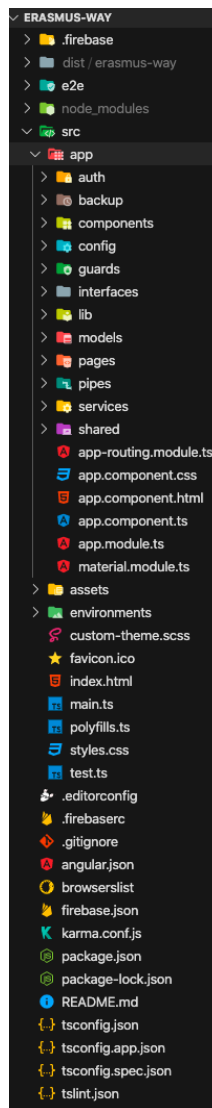


Figura 5.1: Estructura del proyecto

## 5.2. Módulo principal

Como ya hicimos referencia en la introducción al diseño de la arquitectura, la aplicación ErasmusWay arranca mediante el bootstrapping de nuestro root module, encargado de gestionar las cargas de nuestra aplicación. De manera genérica, todos los componentes, librerías, enrutamiento, ...se suelen localizar en el root module y se ejecutan en esta primera llamada, pero en el proyecto que nos atañe y por mantener buenas prácticas de desarrollo, en nuestro proyecto hemos mantenido nuestro NgModule muy liviano con la intencionalidad de liberar a nuestro módulo principal de una carga tan pesada. De esta manera hemos derivado la gestión a cada módulo de manera independiente, manteniendo así un código limpio y modularizado que facilita su mantenimiento, carga y soporte.

```
@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    HttpClientModule,
    PagesModule,
    BrowserModule,
    AppRoutingModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Figura 5.2: Root module

En el NgModule principal se cargan los módulos que vamos a utilizar y con los que se inicia la aplicación. De estas importaciones destacan el módulo «HttpClientModule», el módulo «PagesModule» y el módulo «AppRoutingModule».

### 5.2.1. HttpClientModule

Cliente con los métodos REST habituales que está basado en Observables. Lo vamos a utilizar para hacer llamadas a la API REST (firebase) y obtener los resultados de la misma.

### 5.2.2. AppRoutingModule

Enrutamiento principal de la aplicación que en este primer caso y por motivos académicos, la estamos generando aplicando Lazy loading. Es decir, cuando accedemos a

este componente, se renderiza y se carga exclusivamente este componente, obviando al resto. Angular por norma, carga todo la primera vez, pero aplicando lazy loading se ciñe exclusivamente al componente señalado, por lo que el tiempo de carga es inferior y aumenta su rendimiento.

```
const routes: Routes = [
  {
    path: '',
    redirectTo: 'home',
    pathMatch: 'full'
  },
  {
    path: 'login',
    loadChildren: () => import('./auth/login/login.module')
      .then(m => m.LoginModule)
  },
  {
    path: 'register',
    loadChildren: () => import('./auth/register/register.module')
      .then(m => m.RegisterModule)
  },
  {
    path: 'remember-password',
    loadChildren: () => import('./auth/remember-password/remember-password.module')
      .then(m => m.RememberPasswordModule)
  },
  {
    path: '**',
    redirectTo: 'home',
    pathMatch: 'full'
  }
];
```

Figura 5.3: AppRoutingModuleModule

Como se muestra en la imagen, tenemos una serie de rutas a las que responderá nuestra aplicación. Cuando detecta dicha ruta, la aplicación cargará el módulo especificado y desde el que se gestiona ese componente de manera independiente. Por buenas prácticas, en el routing principal solo estamos gestionando las rutas del «Auth» y además, cada uno de ellos de manera independiente, y la ruta en blanco o desconocida, que derivará a la página principal de forma automática.

### 5.2.3. PagesModule

Módulo independiente creado para centralizar desde un único lugar el funcionamiento de las páginas de la aplicación. Como estamos viendo en este instante en el módulo

general, PagesModule va a contener su propia lógica, módulo, routing y componentes. A su vez cada componente va a tener su propia lógica/controlador y vista.

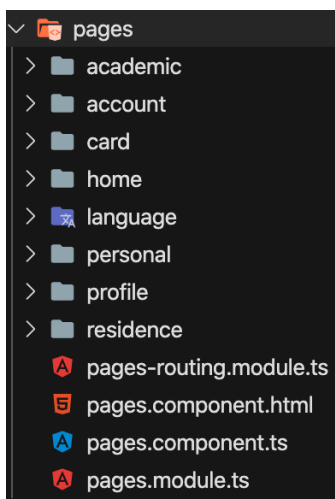


Figura 5.4: Estructura PagesModule

### 5.3. Auth

Como hemos visto en la sección 5.2.2. AppRoutingModuleModule, en el routing principal de la aplicación vamos a manejar el registro, login y recordar contraseña. Todos ellos serán cargados y manejados cuando se acceda a su ruta, es decir, van a tener su propio módulo, enrutamiento, lógica/controlador y vista.

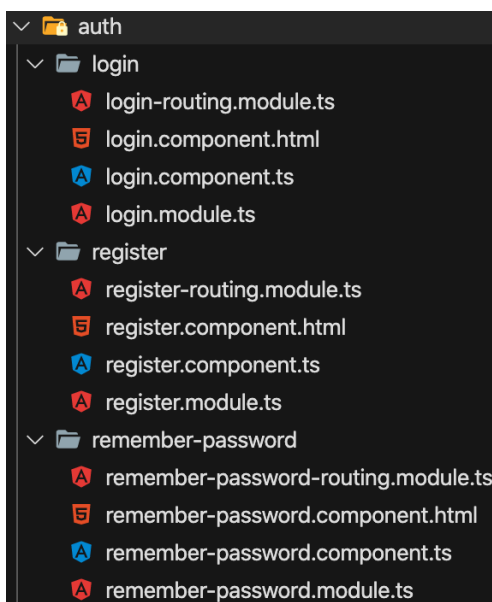


Figura 5.5: Estructura Auth

### 5.3.1. Register

Vamos a ver en detalle el funcionamiento del componente de registro. Esto nos servirá para mostrar la arquitectura de su diseño, así como su funcionamiento.

#### 5.3.1.1. Módulo

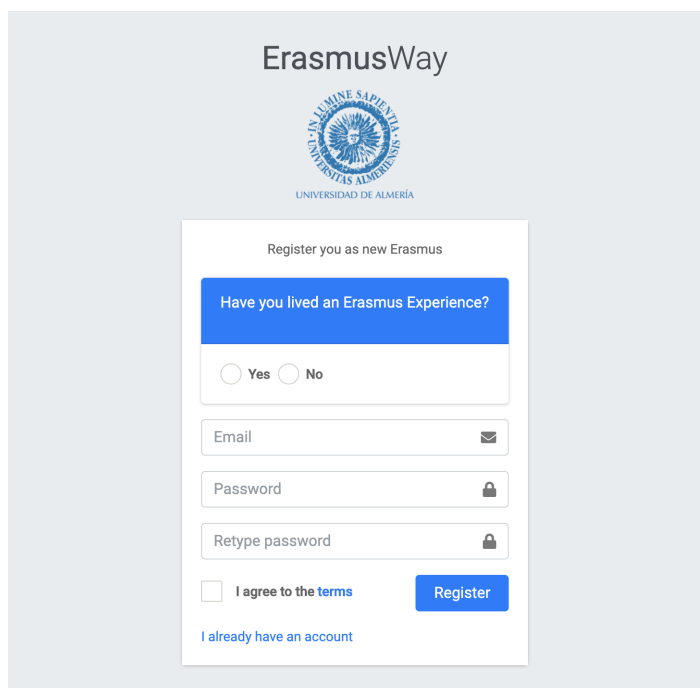
El módulo (`register.module.ts`) va a declarar al propio componente Register y va a importar el Routing interno del módulo, cargar el AngularMaterial y la gestión de formularios.

#### 5.3.1.2. Routing

Cuando se carga el módulo, ya viene implícita la ruta (`register-routing.module.ts`), por lo que damos las instrucciones de respuesta a una ruta vacía que cargará al componente Register. Este a su vez carga la vista y la lógica.

#### 5.3.1.3. Vista

Como hemos visto, una vez se llama al componente, se va a cargar su vista (`register.component.html`) y su lógica. En esta vista, vamos a gestionar un formulario de registro. Para ello, Angular nos ofrece dos formas de manejar y controlar los formularios, una desde la propia vista y otra desde el controlador del componente. En el caso que estamos tratando, lo vamos a manejar desde el controlador del componente en lo que se conoce como «formularios reactivos».



ErasmusWay

UNIVERSIDAD DE ALMERÍA

Register you as new Erasmus

Have you lived an Erasmus Experience?

Yes  No

Email

Password

Retype password

I agree to the terms

[I already have an account](#)

Figura 5.6: Register: Vista form de Registro

El formulario va a contener en su etiqueta <form> una instrucción [formGroup]="registerForm()" que nos indica que es un FormGroup y el nombre del mismo en el controlador. También contiene la llamada a la acción (ngSubmit)="onRegister()" que indica que cuando se pulse en el botón «Register», de tipo submit, se ejecutará la acción onRegister.

#### 5.3.1.4. Controlador

El componente (register.component.ts) recoge la lógica del registro. Este realmente es una clase «decorada» donde en el constructor inicializamos los servicios que vamos a usar en el componente, destacando FormBuilder y el servicio Auth. Este último lo que hace es aumentar la lógica contenida en nuestro controlador.

Dentro del componente, vamos a tener diferentes declaraciones, destacando al formulario donde establecemos el valor inicial de todos los campos y usamos los Validators de Angular para establecer ciertas validaciones por defecto en los campos del formulario.

```
public registerForm = this.fb.group({
  experience: [ '', Validators.required ],
  email: [ '', [Validators.required, Validators.email] ],
  password: [ '', [Validators.required, Validators.minLength(6)] ],
  password2: [ '', [Validators.required, Validators.minLength(6)] ],
  terms: [ , Validators.requiredTrue ],
}, {
  validators: this.passwordEqual('password', 'password2')
});
```

Figura 5.7: Register: Declarar form de registro

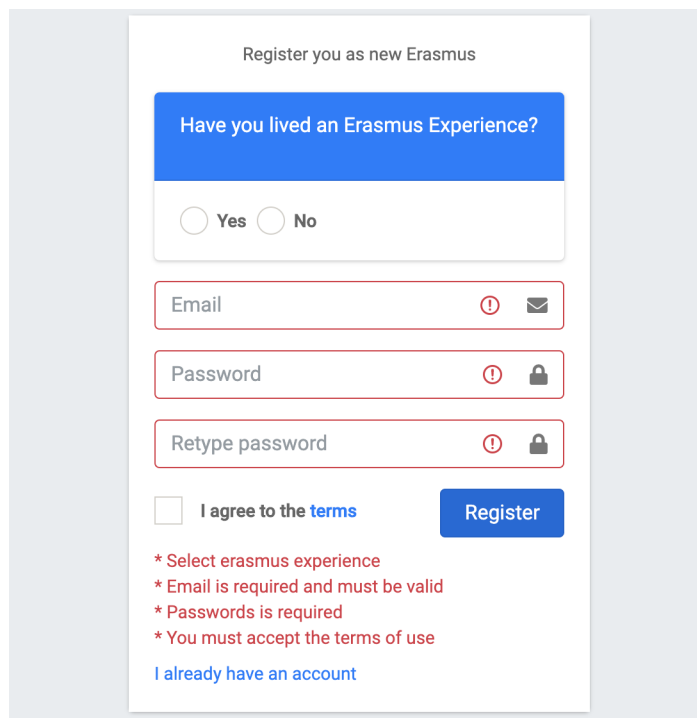
Establecemos los campos requeridos y no nulos, que se respete el formato email, que la contraseña tenga un mínimo de 6 caracteres y una validación personalizada. Esta validación llamará a la función que hemos creado para comprobar que las contraseñas sean iguales.

```
passwordEqual(pass1Name: string, pass2Name: string ) {
  return ( formGroup: FormGroup ) => {
    const pass1Control = formGroup.get(pass1Name); // c
    const pass2Control = formGroup.get(pass2Name); // c

    if ( pass1Control.value === pass2Control.value ) {
      pass2Control.setErrors(null);
    } else {
      pass2Control.setErrors({ noEsIgual: true });
    }
  };
}
```

Figura 5.8: Register: Método de validación personalizado

La codificación anteriormente expuesta se mostrará en la vista una vez se accione el botón «Register» y no se cumplan las validaciones.



Register you as new Erasmus

Have you lived an Erasmus Experience?

Yes  No

Email  ! ✉

Password  ! 🔒

Retype password  ! 🔒

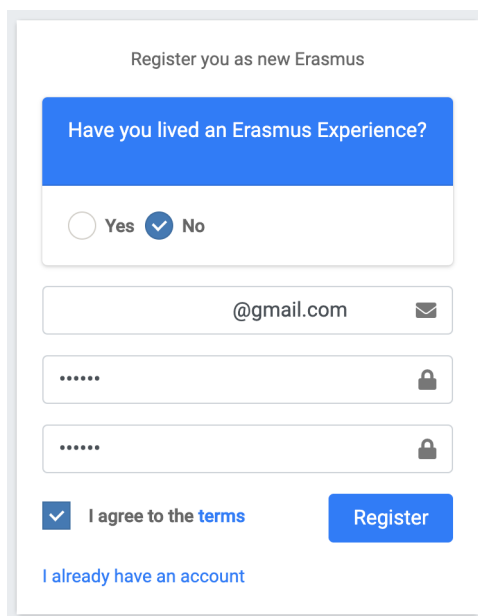
I agree to the terms Register

\* Select erasmus experience  
\* Email is required and must be valid  
\* Passwords is required  
\* You must accept the terms of use

[I already have an account](#)

Figura 5.9: Register form: Validación del form de Registro

En cambio, si los datos introducidos son válidos.



Register you as new Erasmus

Have you lived an Erasmus Experience?

Yes  No

✉

🔒

🔒

I agree to the terms Register

[I already have an account](#)

Figura 5.10: Register form: Form cumplimentado

Finalmente y cuando los datos sean válidos, se ejecuta el siguiente proceso:

1. Se realiza una llamada al método al que hicimos mención al inicio, el método `registerForm()`.
2. Este método se va a subscribir al servicio Auth (lo vemos en la parte de servicios) para realizar el registro y crear un acceso en Firebase Authentication. La subscripción al servicio trae consigo una respuesta y si en la respuesta nos trae el mismo email que en el formulario, es que se ha realizado el registro con éxito.
3. Seguido de esto, creamos el objeto que se va a insertar en la base de datos y preparamos la cabecera que nos exige la API para enviar el correo de verificación.
4. Volvemos a llamar al servicio Auth para subscribirnos al método para enviar un email de verificación.
5. Si el paso anterior tiene éxito, volvemos a llamar al servicio Auth para realizar el registro en la base de datos.
6. Si todo lo anterior se ejecuta con éxito, recibimos una notificación en pantalla y un correo electrónico para confirmar la cuenta.

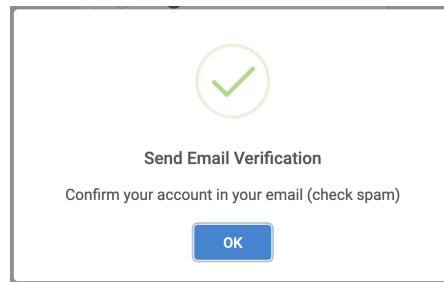


Figura 5.11: Register form: Success

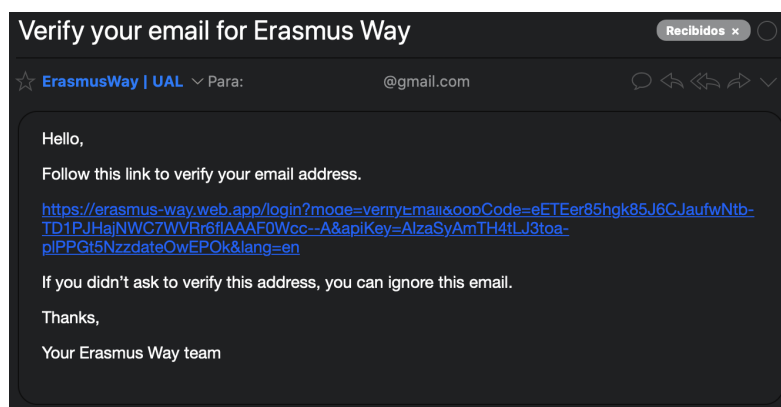


Figura 5.12: Register form: Recepción de email



### 5.3.2. Login

Ahora vamos a ver el componente Login. El módulo, enrutamiento y vista van a ser completamente similares a la explicado en el componente Registro (5.3.1) pero con las referencias al formulario de Login. Vamos adentrarnos en la parte de la lógica, cuyo tarea va a ser distinta de lo visto hasta ahora.

1. Acceso a la ruta /login para introducir nuestras credenciales de acceso y pulsar el botón «Login» que acciona la acción onLogin().

```
158 > public onLogin(): void { ...
242   }
```

Figura 5.13: Login: onLogin()

2. Valoración de acceso 1: Mediante un correo de verificación (Figura 5.12).

```
if ( this.activatedRoute.snapshot.queryParams['mode'] === 'verifyEmail' &&
    this.activatedRoute.snapshot.queryParams['oobCode'] !== undefined ) {
    this.checkVerifyEmail();
}
```

Figura 5.14: Login: Verificar correo

3. Valoración de acceso 2: Confirmación de cambio de contraseña.

```
if ( this.activatedRoute.snapshot.queryParams['mode'] === 'resetPassword' &&
    this.activatedRoute.snapshot.queryParams['oobCode'] !== undefined ) {
    this.checkVerifyResetEmail();
}
```

Figura 5.15: Login: Verificar cambio contraseña

Como hemos visto, la diferencia entre estos tres procesos viene marcada por el lugar desde el que se accede, pero independientemente de esto, una vez se pone en marcha el componente login, bien pulsando en el botón «Sign In», pulsando en el enlace de un correo de verificación o pulsando en el enlace de confirmación de cambio de contraseña, se pone en marcha el ciclo de vida y uno de estos ciclos, concretamente en el `ngOnInit` que implementa nuestro componente login, es el lugar donde se realiza una primera evaluación. Pero...¿qué es el `ngOnInit`? `ngOnInit` forma parte del ciclo de vida de nuestro componente y se ejecuta después del constructor. A diferencia del constructor, `ngOnInit` pertenece al ciclo de vida propio de angular y es aquí donde le ‘decimos’ que el componente ya está listo para darle uso. `ngOnInit` se ejecuta inmediatamente después de hacer las comprobaciones de los enlaces de entrada y salida [1].

Como hemos visto en las Figuras 5.14 y 5.15, se realiza una lectura de la url de acceso para detectar la intención del acceso. Si es válida, pone en marcha el método que corresponda.

### 5.3.2.1. Verificar cuenta

Para la verificación del email recuperamos el oobCode de la url.

```
const body = { oobCode: this.activatedRoute.snapshot.queryParams['oobCode'] };
```

Figura 5.16: Login: recuperar oobCode

Nos subscribimos al servicio Auth para comprobar si el correo se ha confirmado.

```
this.auth.confirmEmailVerification( body )  
  .subscribe( emailVerificationResp => {
```

Figura 5.17: Login: Subscribe Auth: comprobar confirmación

Si no está confirmado, se busca el id del usuario y se subscribe al servicio Student para actualizar el campo needConfirm y ponerlo en True.

```
const value = {  
  needConfirm: true  
};  
  
this.studentsService.patchData(id, value)  
  .subscribe( resp => {
```

Figura 5.18: Login: Subscribe Student: actualizar base de datos

Finalmente muestra notificación de confirmación de cuenta.

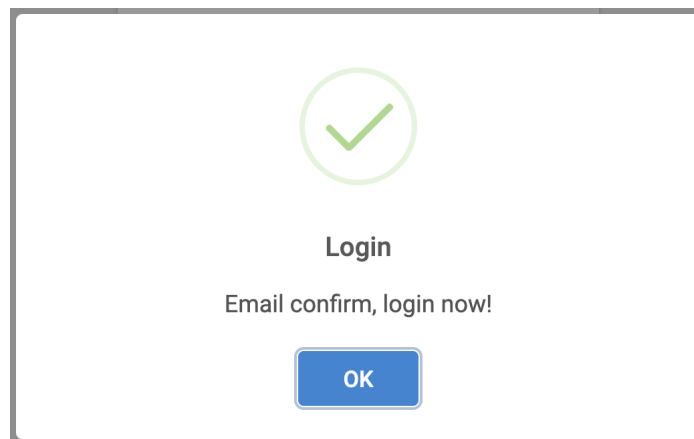


Figura 5.19: Login: Confirmación de cuenta

### 5.3.2.2. Cambiar contraseña

Para el cambio de contraseña, nos subscribimos al servicio Auth y lanzamos un modal en el que introducir una nueva contraseña.

```
public checkVerifyResetEmail() {
  const body = { oobCode: this.activatedRoute.snapshot.queryParams['oobCode'] };

  this.verifyPasswordResetCode$ = this.auth.verifyPasswordResetCode( body );
  this.verifyPasswordResetCodeSubscription = this.verifyPasswordResetCode$
    .subscribe( passwordResetEmailResp => {

    if ( passwordResetEmailResp['requestType'] === 'PASSWORD_RESET' ) {
      $('#newPasswordModal').modal();
    }

  });
}
```

Figura 5.20: Login: Verificar cambio contraseña

### 5.3.2.3. Autenticación

Este es el proceso estándar cuando la situación es distinta de las dos anteriores. Tras la cumplimentación del formulario del login, validar los campos y pulsar en «Sign In», se llama al método onLogin(). Inicialmente realiza una serie de comprobaciones para verificar que el formulario no está vacío y es correcto. Si lo es, comprueba si el correo ha sido confirmado (visto en el punto 5.3.2.1).

A continuación subscribe al servicio students para obtener los datos del usuario que está realizando el proceso de autenticación y comprobar si el correo ha sido verificado.

```
this.studentsService.getFilterData( 'email', this.student.email )
  .subscribe( studentDataResp => {
```

Figura 5.21: Login: Verificar confirmación

Si el correo ha sido verificado, continua y realiza el proceso de autenticación.

```
this.auth.loginAuth( this.student )
  .subscribe( loginAuthResp => {
```

Figura 5.22: Login

Al acceder, realiza una actualización de los datos del estudiante para guardar el idToken en la base de datos y realizar las posteriores validaciones de acceso cuando el usuario navegue por la aplicación.

```
const value = {
  idToken: loginAuthResp['idToken']
}; // console.log('value', value);
const id = Object.keys(studentDataResp).toString();

this.studentsService.patchData(id, value)
  .subscribe( resp => {
```

Figura 5.23: Login: guardar idToken

Finalmente guarda el token, la expiración del token y el correo (solo si se marca «Remember») en el almacenamiento local del dispositivo para concluir el método redirigiéndonos a la página principal.

## 5.4. Shared

Shared hace referencia aquellos componentes que van a ser compartidos en todas las páginas (Pages) de la aplicación. Estos son el breadcrumb, navbar, sidebar y footer. La estructura que presenta dentro del proyecto es la siguiente:

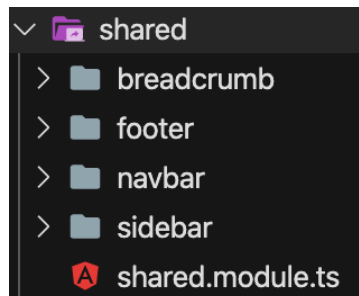


Figura 5.24: Estructura Shared

Cada componente citado anteriormente se va a componer de su vista, contenida en el documento con formato HTML, y su lógica contenida en el documento con el formato TS. En su mayoría, los componentes del módulo Shared son documentos visuales que tienen poca lógica y que utilizan el routerLink de Angular para hacer referencia a las distintas páginas de navegación. El hecho importante a destacar se encuentra contenido en su módulo, donde deben declararse aquellos componentes que van a ser exportables, para hacerlos importables. La importación los habilita para poder utilizarlos en otros componentes, como a continuación veremos en el módulo Pages.

```

@NgModule({
  declarations: [
    BreadcrumbComponent,
    NavbarComponent,
    SidebarComponent,
    FooterComponent
  ],
  exports: [
    BreadcrumbComponent,
    NavbarComponent,
    SidebarComponent,
    FooterComponent
  ],
  imports: [
    CommonModule,
    RouterModule
  ]
})
export class SharedModule { }

```

Figura 5.25: Modulo Shared

## 5.5. Pages

Pages hace referencia a los componentes de las páginas. La estructura del proyecto la hemos visto en el apartado 5.2.3. PagesModule. En su documento general de vista (pages.component.html) podemos comprobar como se utilizan todos los componentes creados en el shared (5.4) y que van a ser comunes a todas las páginas enrutadas.

```

<!-- Navbar -->      You, a month ago * Login, R
<app-navbar></app-navbar>
<!-- /.navbar -->

<!-- Main Sidebar Container -->
<app-sidebar></app-sidebar>
<!-- /.Main Sidebar Container -->

<!-- Content Wrapper. Contains page content -->
<div class="content-wrapper">
  <!-- Content Header (Page header) -->
  <app-breadcrumb></app-breadcrumb>
  <!-- /.content-header -->

  <!-- Main content -->
  <section class="content">
    <div class="container-fluid">
      <router-outlet></router-outlet>
    </div><!-- /.container-fluid -->
  </section>
  <!-- /.content -->
</div>
<!-- /.content-wrapper -->
<app-footer></app-footer>

```

Figura 5.26: Vista general Pages

Las páginas enrutadas utilizan el `<router-outlet>`, las rutas van a estar protegidas por el Guard y solo podrán ser visualizadas por los usuarios que cumplan las condiciones del mismo, en nuestro caso, los usuarios autenticados.

```
const routes: Routes = [
  {
    path: 'home',
    component: PagesComponent,
    canActivate: [ AuthGuard ],
    children : [
      {
        path: '',
        component: HomeComponent,
        data: { title: 'Home' }
      },
      {
        path: 'profile',
        component: ProfileComponent,
        data: { title: 'Profile' }
      },
      {
        path: 'personal',
        component: PersonalComponent,
        data: { title: 'Personal' }
      },
      {
        path: 'account',
        component: AccountComponent,
        data: { title: 'Account' }
      },
      {
        path: 'academic',
        component: AcademicComponent,
        data: { title: 'Academic' }
      },
      {
        path: 'language',
        component: LanguageComponent,
        data: { title: 'Languages' }
      },
      {
        path: 'residence',
        component: ResidenceComponent,
        data: { title: 'Residence' }
      },
      {
        path: 'card/:id/:type',
        component: CardComponent,
        data: { title: 'Information' }
      },
    ]
  }
];
```

Figura 5.27: Router Pages

A continuación mostramos como los componentes creados en el Shared, trabajando con el módulo y enrutamiento del Pages nos muestran el contenido de las páginas. Los componentes Navbar, Sidebar, Breadcrumb y Footer ya los tratamos anteriormente. En cambio, todos los componentes creados en este módulo se renderizarán en la parte gráfica señalada en rojo en la figura 5.28 y que recibe el nombre de Pages. El momento en el que se cargará será cuando detecten su url (5.27).

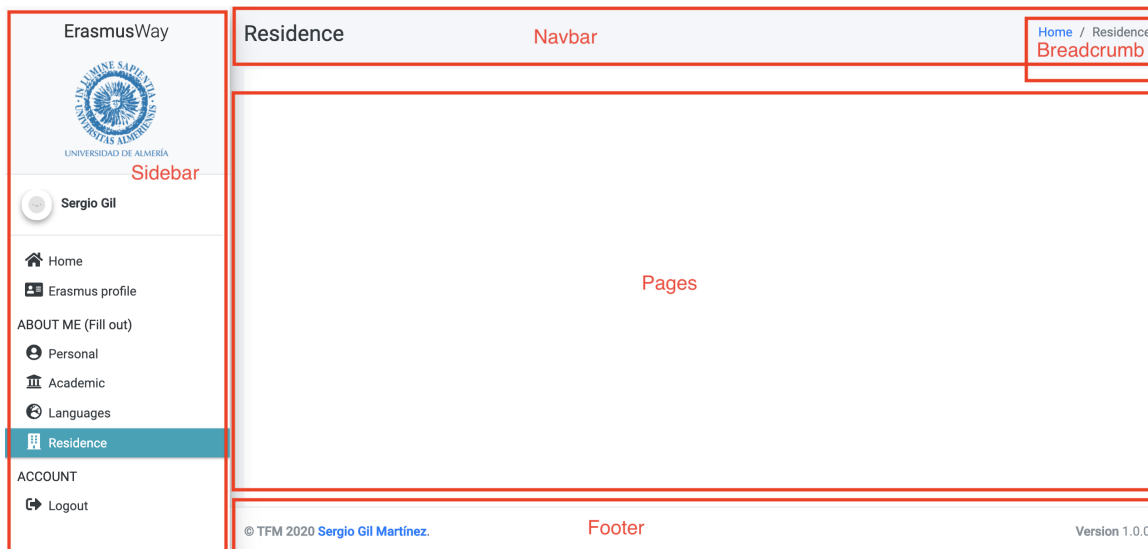


Figura 5.28: Componentes trabajando juntos

Los componentes van a guardar el mismo planteamiento que ya hemos visto en ocasiones anteriores. Van a tener su propia lógica, vista y algunos como el componente card, van a tener su propio CSS. Dentro de la lógica se van a realizar peticiones a la base de datos a través de la inyección de uno o varios servicios. Los servicios se inyectan en el constructor y pueden usarse tantas veces como se deseen. En los servicios vamos a encontrar promesas y observables a los que habrá que subscribirse para hacer uso de ellos. Su uso va a depender de cuando se necesitan los datos, ya que en su mayoría las cargas van a ser asíncronas, pero en ocasiones vamos a necesitar los datos en un momento concreto. Por esta razón, en ocasiones tendremos que esperar la data mediante el uso `async/await` o mediante una promesa.

Dependiendo del componente, se realizarán unas acciones u otras y todo estará conectado directamente a la vista, que mostrará los datos en la primera carga, los datos se modificarán de forma dinámica y sin tener que volver a realizar una carga de toda la página, dando sensación de fluidez y eficiencia propias de una SPA.

En ocasiones, como es el caso del componente Home, referente a la página principal, dentro del componente se va a utilizar otro componente al que llamaremos componente hijo. Esto nos da la posibilidad de enviar los datos desde el componente padre al componente hijo, dejando la lógica y cálculo en el primero y dándonos la posibilidad de usar el componente hijo tantas veces como sea necesario con solo nombrar su etiqueta. Esto nos da pie al siguiente módulo, los componentes (componentes hijos).

## 5.6. Components

El módulo de components, o componentes hijos, va a presentar la siguiente estructura.

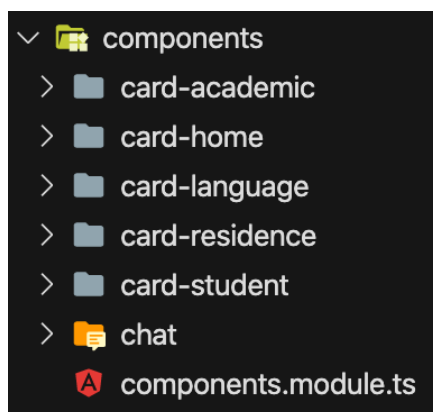


Figura 5.29: Estructura Components

Vamos a mostrar el uso del componente «card-home». El uso del resto de componentes hijos que conforman el módulo va a ser muy similar a este.

La lógica del componente hijo declara tres variables de tipo `Input()`, una de ellas de tipo `number` que mostrará el número de estudiantes, y las otras dos de tipo `string`. Una de ellas contiene el concepto que aparecerá junto al número de estudiantes y la otra, contiene las clases del estilo `css` que se van a aplicar. Este hecho obliga al componente padre a enviarle con carácter obligatorio los datos de las tres variables provocando un error si este hecho no se produce.

```
@Component({
  selector: 'app-card-home',
  templateUrl: './card-home.component.html'
})
export class CardHomeComponent implements OnInit {
  @Input() public num: number;
  @Input() public concept: string;
  @Input() public class: string;

  constructor() { }

  ngOnInit(): void { }
}
```

Figura 5.30: Componente hijo: Lógica: card-home



Las tres variables se van a mostrar en la vista de la siguiente manera.

```
<div [class]="class">
  <div class="inner text-center">
    <h3 class="mt-3">{{ num }}</h3>

    <p>{{ concept }}</p>
  </div>
  <div class="icon">
    <i class="ion ion-bag"></i>
  </div>
</div>
```

Figura 5.31: Componente hijo: Vista: card-home

El componente hijo va a ser muy simple de reutilizar. Para ello será necesario que se importe en el módulo del componente donde se vaya a utilizar. Al componente donde se va a utilizar lo llamaremos componente padre. Para utilizarlo será necesario etiquetarlo en la vista del padre con el nombre del selector y enviarle el valor de las tres variables declaradas, que se declaran y obtienen en la parte de la lógica del componente padre.

```
<div class="row animated fadeIn" *ngIf="loaded">
  <div class="col-lg-3 col-md-6 col-sm-12">
    <!-- small box -->
    <app-card-home [num]="numStudent"
                  [concept]="Erasmus users"
                  [class]='small-box bg-info'></app-card-home>
  </div>
  <!-- ./col -->
  <div class="col-lg-3 col-md-6 col-sm-12">
    <!-- small box -->
    <app-card-home [num]="numAcademic"
                  [concept]="Academic cards"
                  [class]='small-box bg-success'></app-card-home>
  </div>
  <!-- ./col -->
  <div class="col-lg-3 col-md-6 col-sm-12">
    <!-- small box -->
    <app-card-home [num]="numLanguage"
                  [concept]="Languages cards"
                  [class]='small-box bg-warning'></app-card-home>
  </div>
  <!-- ./col -->
  <div class="col-lg-3 col-md-6 col-sm-12">
    <!-- small box -->
    <app-card-home [num]="numResidence"
                  [concept]="Residence cards"
                  [class]='small-box bg-danger'></app-card-home>
  </div>
  <!-- ./col -->
</div>
```

Figura 5.32: Componente padre: Vista: page Home

El componente hijo mostrado en el componente padre se vería así:

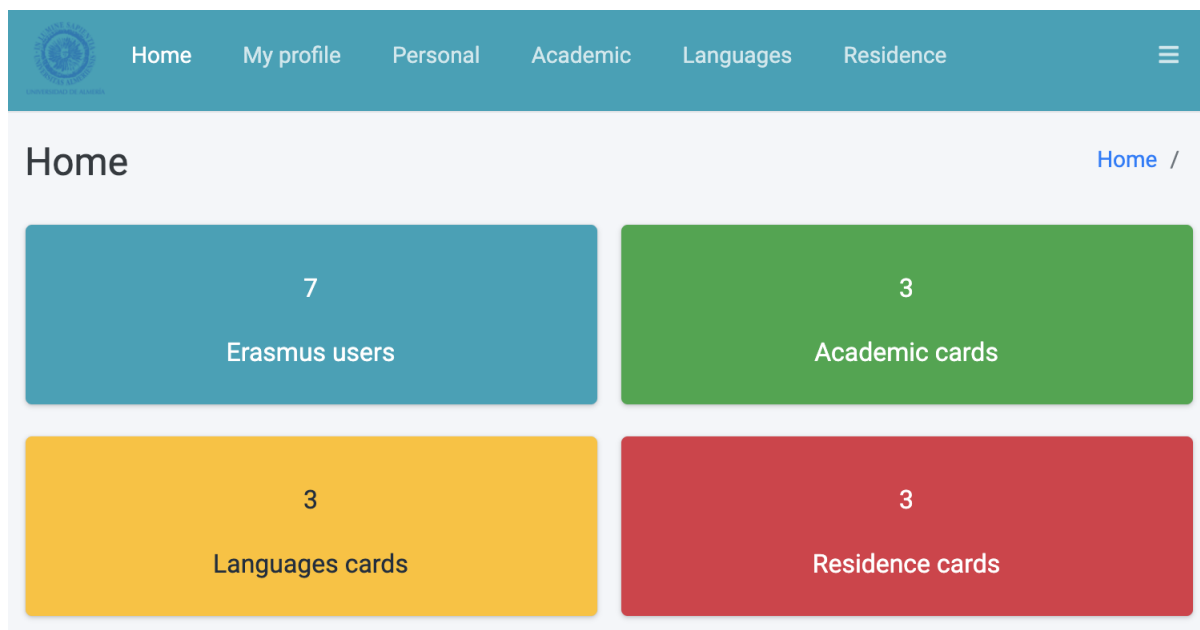


Figura 5.33: Visualización del componente hijo en el padre

Como podemos comprobar, hemos llamado a su selector cuatro veces, por lo que visualizamos ese número de veces al componente hijo. El componente lo podríamos reutilizar indefinidamente y su mantenimiento sería extremadamente sencillo. Solo tendríamos que realizar una vez la modificación y se replicaría en todas partes.

## 5.7. Servicios

Gran parte de la lógica de nuestra aplicación, así como la mayor parte de los datos proceden del uso de los servicios. Motivados por las buenas practicas, hemos creado siete servicios correspondientes a diferentes entidades y/o procesos.

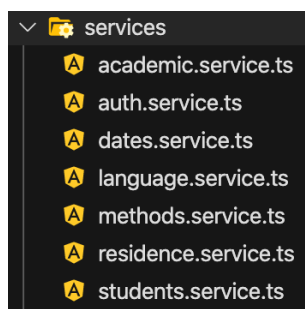


Figura 5.34: Estructura Services

Pero, ¿qué son los servicios?...Los servicios son clases TypeScript y su propósito es el de contener lógica de negocio, clases para acceso a datos o utilidades de infraestructura. Estas clases son perfectamente instanciables desde cualquier otro fichero que las importe.

La particularidad de las clases de servicios está en su decorador: `@Injectable()`. Esta función viene en el `@angular/core` e indica que esta clase puede ser inyectada dinámicamente a quien la demande.

```
@Injectable({
  providedIn: 'root'
})
export class MethodsService {
```

Figura 5.35: Decorador de un servicio e inyección

En la versión de Angular con la que estamos trabajando, los servicios se auto-proveen en el módulo raíz mediante la configuración del `providedIn: 'root'` de su decorador.

Saliendo de las explicaciones teóricas anteriores y volviendo a nuestro proyecto, vamos a proceder a ver como trabajan alguno de los servicios que hemos ido creando para cubrir ciertas necesidades que nos han ido surgiendo. Por un lado tenemos a servicios que trabajan con datos y con lógica, otros servicios trabajan solo con datos y por último nos encontramos con servicios que solo están para manejar lógica.

### 5.7.1. Servicio: Academics

En este servicio nos vamos a encontrar con el manejo de lógica del sistema y con el manejo de data referente exclusivamente a los datos académicos de los estudiantes. Por otro lado y para no caer en una explicación repetitiva, no vamos adentrarnos en los servicios de Language, Residence y Student porque van a ser muy similares a este en cuanto a estructura y funcionalidades.

Como hemos citado al inicio, estos servicios van a manejar datos exclusivamente referenciados a la entidad que les da nombre. De esta forma los separamos para realizar la llamada del servicio solo cuando sea necesario y se nos posibilite modificarlos según las necesidades actuales y futuras que pudieran surgirnos sin afectar al resto de entidades.

En el servicio nos vamos a encontrar con llamadas a la API, es decir, llamadas a nuestra base de datos como servicio Firebase donde vamos a realizar consultas y obtención de datos, inserción y actualización de los mismos. Para realizar estas llamadas nos ceñiremos a una estructura muy concreta que nos demanda nuestra API.

En la sección 5.8. vamos a explicar los Guards y aprovecharemos para entrar en detalles de como declaramos un servicio y consumirlo.

```

export class AcademicService {

  private apiRest = FBASE.APIREST;

  constructor( private http: HttpClient ) { }

  public getFilterData( orderBy: string, equalTo: string ) {
    return this.http.get(
      `${this.apiRest}academics.json?orderBy=${orderBy}&equalTo=${equalTo}&print=pretty`);
  }

  public async getDataFilter( idStudent: string ) {
    const authToken = localStorage.getItem(COLLECTIONS.LOCALTOKEN);
    return await this.getFilterData('idStudent', idStudent).toPromise();
  }

  public getData( id: string ) {
    const authToken = localStorage.getItem(COLLECTIONS.LOCALTOKEN);
    return this.http.get(`${this.apiRest}academics/${id}.json?auth=${authToken}`);
  }

  public getAllData() {
    const authToken = localStorage.getItem(COLLECTIONS.LOCALTOKEN);
    return this.http.get(`${this.apiRest}academics.json?auth=${authToken}`);
  }

  public postData( value: object ) {
    const authToken = localStorage.getItem(COLLECTIONS.LOCALTOKEN);
    return this.http.post(`${this.apiRest}academics.json?auth=${authToken}`, value);
  }

  public patchData(id: string, value: object) {
    const authToken = localStorage.getItem(COLLECTIONS.LOCALTOKEN);
    return this.http.patch(`${this.apiRest}academics/${id}.json?auth=${authToken}`, value);
  }

  public patchDataAuthor(id: string, value: object) {
    const authToken = localStorage.getItem(COLLECTIONS.LOCALTOKEN);
    return this.http.patch(`${this.apiRest}academics/${id}/author.json?auth=${authToken}`, value);
  }

  public sendComment( id: string, value: any ) {
    const authToken = localStorage.getItem(COLLECTIONS.LOCALTOKEN);
    return this.http.put(`${this.apiRest}academics/${id}/all_comments.json?auth=${authToken}`, value);
  }
}

```

Figura 5.36: Servicio: Academics

### 5.7.2. Servicio: Methods

Otro tipo de servicio bien distinto, es el servicio Methods. Este no realiza ninguna llamada a la base de datos sino que se centra únicamente en el manejo de la lógica de ciertas funcionalidades de la aplicación. En este caso, se ciñe a realizar un filtrado de datos en campos de tipo input.

```

@Injectable({
  providedIn: 'root'
})
export class MethodsService {

  constructor() { }

  public filteredOptions(getForm: FormGroup, getField: string, bbddSelect: any[]) {
    return getForm.get(getField).valueChanges
      .pipe(
        startWith(''),
        map( value => typeof value === 'string' ? value : value.name ),
        map( name => name ? this._filter(name, bbddSelect) : bbddSelect.slice() );
      );
  }

  private _filter(name: string, bbddSelect: any[]): any[] {
    const filterValue = name.toLowerCase(); // console.log(filterValue);
    return bbddSelect.filter(option => option.name.toLowerCase().indexOf(filterValue) === 0);
  }
}

```

Figura 5.37: Servicio: Methods

### 5.7.3. Servicio: Dates

Servicio que realiza llamadas a nuestra API para obtener el valor de todos los desplegables que se mostrarán en la aplicación.

```

constructor( private httpClient: HttpClient ) {}

public async getPrices() {
  const url = `${this.api}prices.json`;
  return await this.httpClient.get(url).toPromise();
}

public async getPlaces() {
  const url = `${this.api}places.json`;
  return await this.httpClient.get(url).toPromise();
}

public async getAreas() {
  const url = `${this.api}areas.json`;
  return await this.httpClient.get(url).toPromise();
}

```

```

public async getCountries() {
  const url = `${this.api}countries.json`;
  return await this.httpClient.get(url).toPromise();
}

public async getCourses() {
  const url = `${this.api}courses.json`;
  return await this.httpClient.get(url).toPromise();
}

public async getBranches() {
  const url = `${this.api}branches.json`;
  return await this.httpClient.get(url).toPromise();
}

public async getDegrees() {
  const url = `${this.api}degrees.json`;
  return await this.httpClient.get(url).toPromise();
}

```

Figura 5.38: Servicio: Methods

## 5.8. Guards

En la aplicación vamos a necesitar que determinadas áreas estén protegidas y solo puedan ser accedidas si el usuario está logueado. Para conseguir este propósito utilizaremos los guards. Nuestro guard determinará el acceso bajo una serie de condiciones. Los guards nos ofrecen cuatro tipos que analizaremos para determinar cual usar:

- CanActivate: Comprueba si el usuario puede acceder a una página determinada.
- CanActivateChild: Comprueba si el usuario puede acceder a las páginas hijas de una determinada ruta.
- CanDeactivate: Comprueba si el usuario puede salir de una página.
- CanLoad: Sirve para evitar que la aplicación cargue los módulos perezosamente si el usuario no está autorizado a hacerlo.

En nuestro proyecto, vamos a tener un perfil de estudiante cuya condición de acceso a las páginas va a ser la de estar autenticado y que su autenticación sea válida. Para ello, vamos a necesitar hacer uso del guard CanActivate.

Nuestro guard va a llamar al servicio Auth. Para ello inicialmente lo va a declarar en el constructor, para después comprobar a través de una promesa si la autenticación es válida.

```
@Injectable({
  providedIn: 'root'
})
export class AuthGuard implements CanActivate {

  constructor(
    private auth: AuthService,
    private router: Router
  ) {}

  canActivate(): Promise<boolean> {
    console.log('Guard');

    return new Promise ( resolve => {

      this.auth.validAuthActivate().then( resp => {

        if (!resp) {
          this.router.navigateByUrl('/login');
          resolve(false);
        } else {
          resolve(true);
        }
      }
    )
  }
}
```

Figura 5.39: Guard: CanActivate usando servicio Auth

Como ya hemos indicado, nuestro guard va a consumir al servicio Auth para comprobar si la autenticación es válida y permitir el acceso. En cambio y si no lo es, nos expulsa de la aplicación sin llegar a obtener la vista del componente al que se ha intentado acceder. Todo ello se realiza a través de una promesa que accede a la API para comprobar su validez y su expiración.

```
public validAuthActivate() {  
  
  return new Promise ( resolve => {  
  
    const localToken = localStorage.getItem(COLLECTIONS.LOCALTOKEN);  
    const localExpire = localStorage.getItem(COLLECTIONS.LOCALEXPIRE);  
  
    // Validar que el idToken es real y válido  
    if ( localToken ) {  
  
      const body = {  
        idToken: localToken  
      };  
  
      this.httpPostValidAuthActivate$ = this.http.post(`${ this.url }lookup?key=${ this.apikey }`, body);  
      this.httpPostValidAuthActivateSubscribe = this.httpPostValidAuthActivate$  
        .subscribe( studentResp => {  
  
        if ( localExpire ) {  
  
          // Validar fecha de expiración  
          const expire = Number(localExpire);  
          const expireDate = new Date();  
          expireDate.setTime(expire);  
  
          if ( expireDate > new Date() ) {  
            resolve(true);  
          } else {  
            this.logout();  
            resolve(false);  
          }  
        } else {  
          this.logout();  
          resolve(false);  
        }  
  
        }, (err) => {  
          this.logout();  
          resolve(false);  
        }  
      });  
    } else {  
      this.logout();  
      resolve(false);  
    }  
  });  
}
```

Figura 5.40: Guard: Servicio: Validación de acceso

Para finalizar, marcamos las rutas que van a estar protegidas por el guard CanActivate. Las rutas ya las visualizamos en la Figura 5.27.

```

17  const routes: Routes = [
18    {
19      path: 'home',
20      component: PagesComponent,
21      canActivate: [ AuthGuard ],
22    > children : [ ...
63    ]
64  }
65 ];

```

Figura 5.41: Rutas protegidas por el guard

## 5.9. Base de datos

Para la base de datos, vamos a utilizar Firebase como un servicio. El tratamiento de los datos está implícito en los servicios y tratado en la sección 5.7. En esta sección se pretende mostrar la autenticación de usuarios en el backend, el almacenamiento de datos en colecciones y las reglas de configuración de la base de datos.

### 5.9.1. Autenticación

Muestra de las cuentas que actualmente han realizado el proceso de registro y acceso a la aplicación.

Identificador	Proveedores	Fecha de creación	Inicio de sesión	UID de usuario ↑
[redacted]@gmail.com	✉	31 ago. 2020	5 sept. 2020	2xe5c6PPn9aGaZ
[redacted]@gmail.c...	✉	4 sept. 2020	4 sept. 2020	Z5nkuHKuzHhIOH
[redacted]@hotmail.com	✉	31 ago. 2020	31 ago. 2020	cmjpXt4mOsa3tI2
[redacted]@gmail.com	✉	1 sept. 2020	1 sept. 2020	IRCUszkbqnYPP8
[redacted]@hotmail.com	✉	31 ago. 2020	31 ago. 2020	mnUFi403VcRLa0
[redacted]@hotmail.c...	✉	1 sept. 2020	1 sept. 2020	p1Q5snvwnvNgmi
[redacted]@hotmail.com	✉	1 sept. 2020	1 sept. 2020	wntbB9wdqQh78x

Figura 5.42: Firebase: Autenticación



## 5.9.2. Almacenamiento de datos

Almacenamiento de datos contenido en colecciones.



Figura 5.43: Almacenamiento en colecciones de datos

### 5.9.3. Reglas de la base de datos

Esta última sección hace referencia a las normas establecidas para hacer uso de la base de datos.

```

"rules": {
  "academics": {
    ".read": "auth != null",
    ".write": "auth != null",
    ".indexOn": ["author",
      "author/email",
      "author/nativeCountry/name",
      "idStudent",
      "course/name",
      "votes"]
  },
  "areas": {
    ".read": true,
    ".write": "auth != null"
  },
  "branches": {
    ".read": true,
    ".write": "auth != null"
  },
  "courses": {
    ".read": true,
    ".write": "auth != null"
  },
  "countries": {
    ".read": true,
    ".write": "auth != null"
  },
  "degrees": {
    ".read": true,
    ".write": "auth != null"
  },
  "languages": {
    ".read": "auth != null",
    ".write": "auth != null",
    ".indexOn": ["author",
      "author/email",
      "author/nativeCountry/name",
      "idStudent",
      "votes"]
  },
  "places": {
    ".read": true,
    ".write": "auth != null"
  },
  "prices": {
    ".read": true,
    ".write": "auth != null"
  },
  "residences": {
    ".read": "auth != null",
    ".write": "auth != null",
    ".indexOn": ["author",
      "author/email",
      "author/nativeCountry/name",
      "idStudent",
      "votes"]
  },
  "students": {
    ".read": "auth != null",
    ".write": "auth != null",
    ".indexOn": ["email", "idToken", "nativeCountry", "erasmusExperience", "name", "lastname"],
  }
}

```

Figura 5.44: Reglas de la base de datos

Hemos establecido que todas las colecciones de los despletables van a ser de lectura sin condiciones (`read: true`). En cambio, los datos de los estudiantes van a demandar que el usuario este autenticado y validado en el sistema para poder acceder a los datos, tal y como sucede con las operaciones de escritura. Algunas colecciones las hemos habilitado para que puedan ser indexadas a través de ciertas rutas. Este proceso se realiza en los servicios y es consumido por los distintos componentes de la aplicación.

## 5.10. Pruebas

Vamos a realizar una valoración de si merece la pena realizar pruebas automáticas para nuestro proyecto. A priori debemos contemplarlo y determinar las pruebas a realizar.

Ventajas	Desventajas
Encontrar errores	Trabajo muy extenso
Probar código escrito de otros	Coste de desarrollo aumenta
Detectar errores antes de producción	No tiene mucho sentido si trabajas solo
Generar estadísticas	

Cuadro 5.1: Estudio sobre realizar pruebas

Como el proyecto que estamos tratando en esta memoria ha sido realizado por un único desarrollador, el control sobre la evolución del proyecto ha sido total. Por esta razón creemos que no sería necesario realizar TDD y nos vamos a ceñir a pruebas más funcionales.

### 5.10.1. Llamadas a la API

Por un lado, vamos a comprobar que todas las llamadas realizadas a nuestra API han recibido una respuesta correcta. Todas estas comprobaciones se han realizado previamente a desarrollar los servicios. Para realizar las llamadas se ha utilizado Postman cuya configuración del entorno ha sido la que se muestra en la imagen:

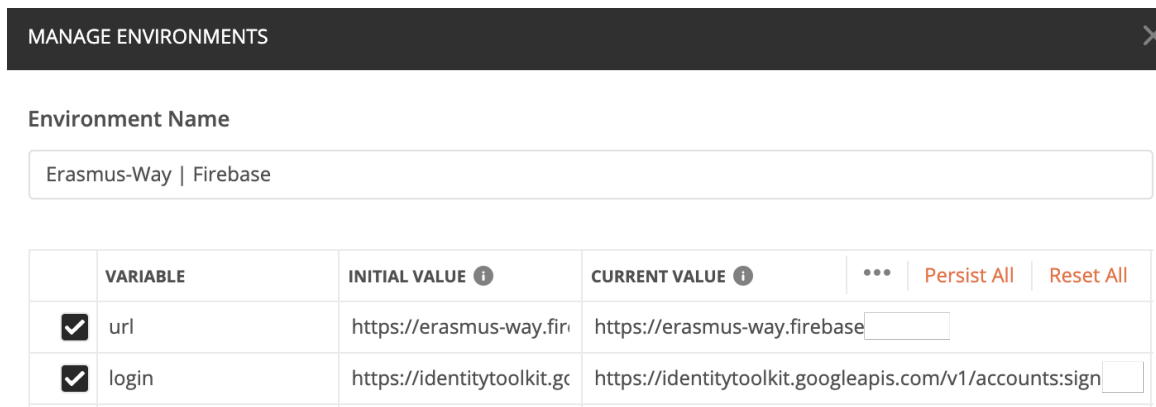


Figura 5.45: Configuración de Postman

A continuación, se muestra un ejemplo de una de las pruebas realizadas.



Se adjunta imagen donde se muestran algunas de las pruebas realizadas en postman.

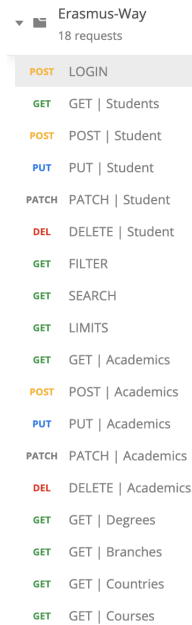


Figura 5.49: Pruebas: Postman

### 5.10.2. Pruebas funcionales

Para acometer las pruebas funcionales, accedemos con el usuario que creamos en la parte del registro para realizar pruebas en todas las secciones de la aplicación.

Vamos a guardar la información personal de nuestro usuario. Recibimos la confirmación y lo verificamos en la sección del perfil:

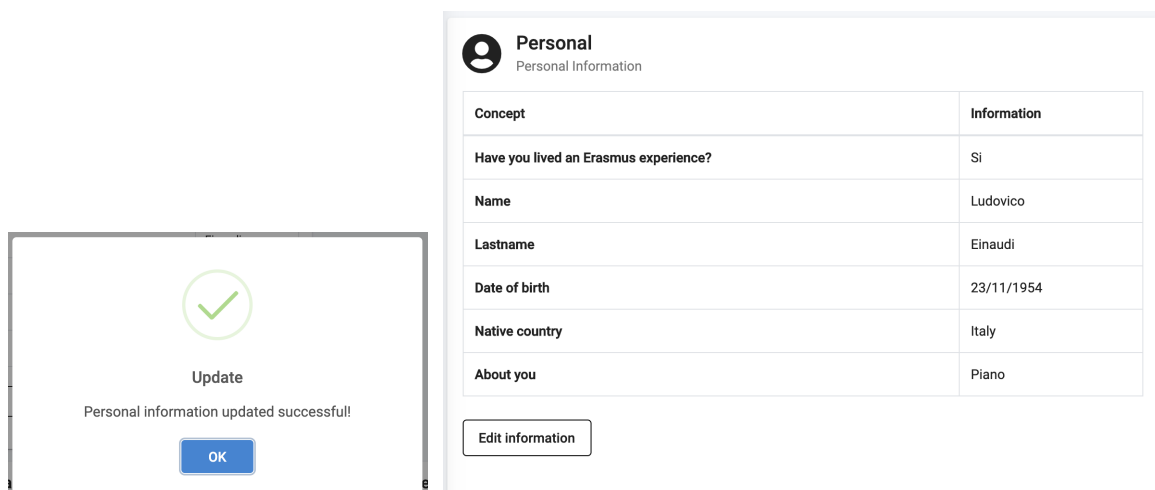


Figura 5.50: Pruebas funcionales: Guardar info Personal

Procedemos a guardar la información académica de nuestro usuario de pruebas. Recibimos la confirmación y lo verificamos en la sección del perfil:

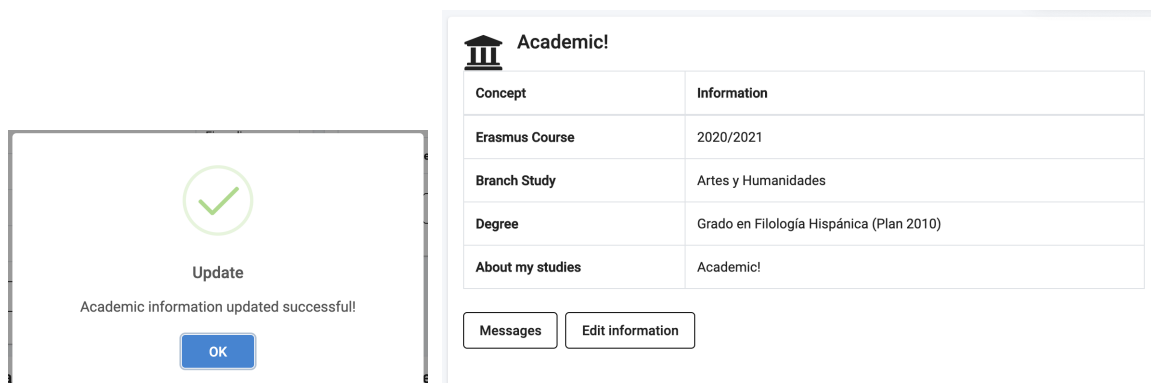


Figura 5.51: Pruebas funcionales: Guardar info Académica

Seguido de lo anterior, procedemos a guardar la información sobre idiomas de nuestro usuario de pruebas. Recibimos la confirmación y lo verificamos en la sección del perfil:

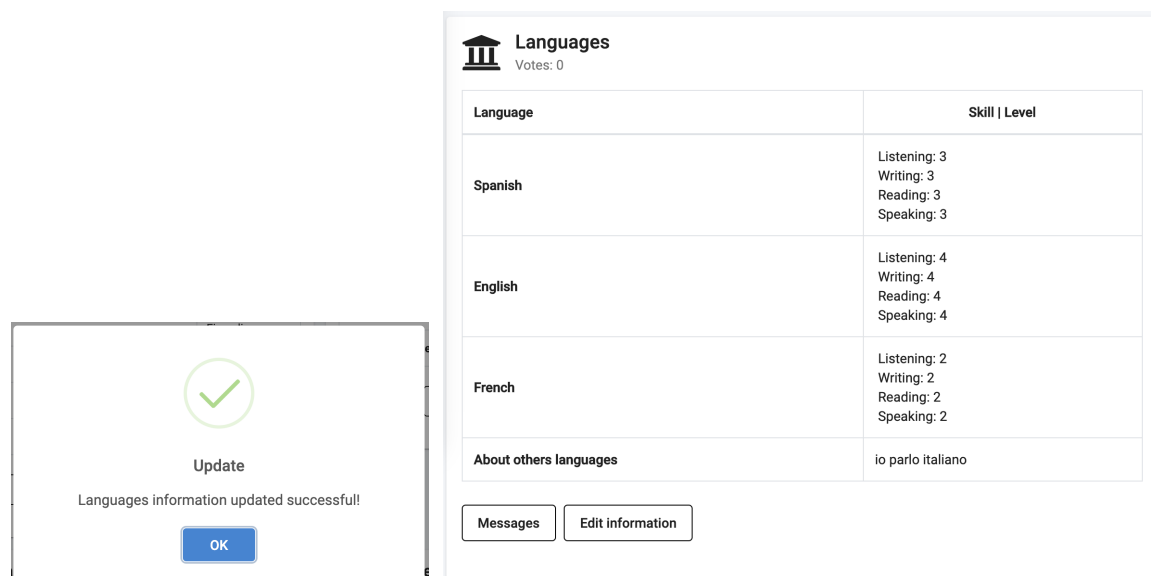


Figura 5.52: Pruebas funcionales: Guardar info Idiomas

Para finalizar la fase de pruebas de insertar datos en las tarjetas de información del estudiante, procedemos a guardar la información sobre vivienda de nuestro usuario de pruebas. Recibimos la confirmación y lo verificamos en la sección del perfil:

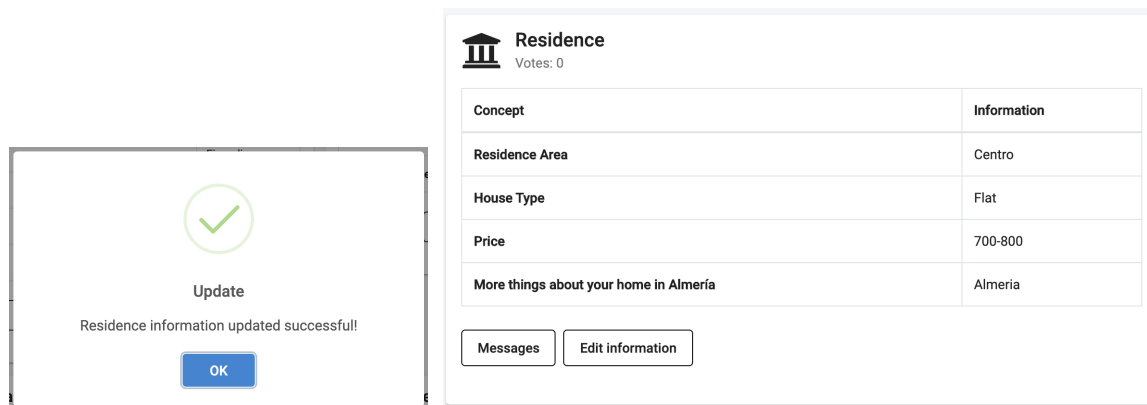


Figura 5.53: Pruebas funcionales: Guardar info Vivienda

Como hemos podido comprobar, todos los datos se han registrado y se muestran como habíamos establecido.

Continuamos ahora comprobando como se muestran las tarjetas en la página principal.

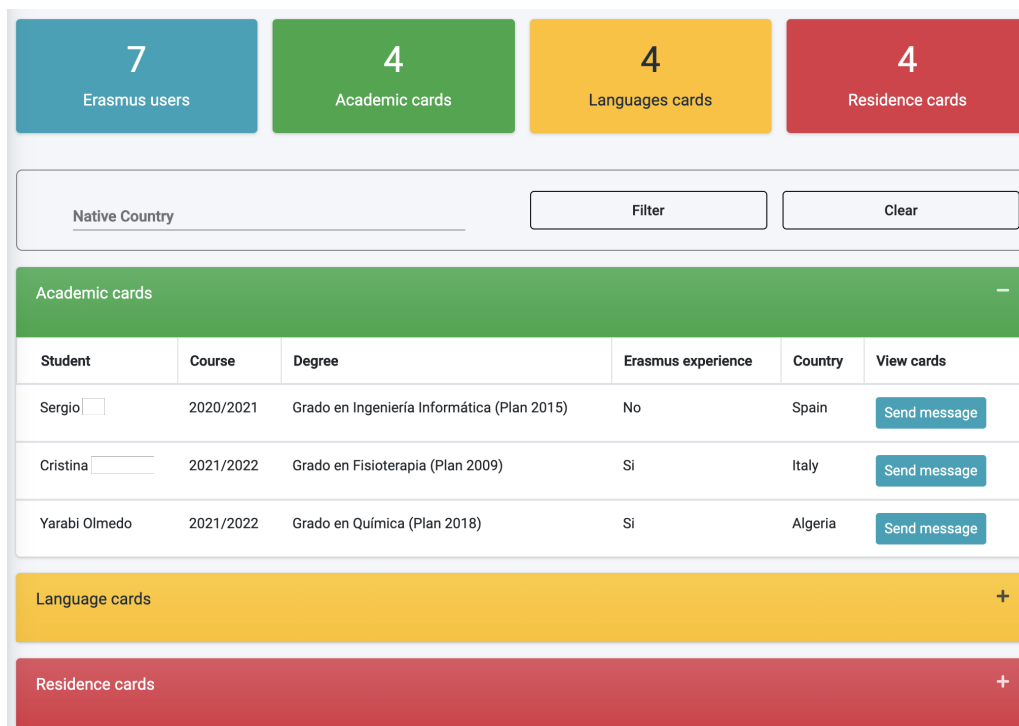


Figura 5.54: Página principal

Vemos existen siete usuarios registrados, de los cuales cuatro de ellos han cumplimentado toda la información. En pantalla se muestran tres de las tarjetas, ya que establecimos que no va aparecer nunca nuestra propia ficha. La visualización de las fichas correspondientes al estudiante autenticado queda reservada a su sección «Profile».

En la imagen anterior (5.54) se muestra a los usuarios registrados y sus tarjetas. Comprobamos que existe un usuario procedente de «Spain». Usemos esto en el filtro y veamos que nos devuelve.

The screenshot shows a web interface with a filter for 'Spain' and three categories of cards:

- Erasmus users:** 7
- Academic cards:** 1
- Languages cards:** 1
- Residence cards:** 1

The filter is set to 'Spain' and the results are displayed in three sections:

#### Academic cards

Student	Course	Degree	Erasmus experience	Country	View cards
Sergio <input type="checkbox"/>	2020/2021	Grado en Ingeniería Informática (Plan 2015)	No	Spain	<a href="#">Send message</a>

#### Language cards

Student	Spanish	English	French	Country	View cards
Sergio <input type="checkbox"/>	Listening: 5 Writing: 5 Reading: 5 Speaking: 5	Listening: 2 Writing: 3 Reading: 3 Speaking: 2	Listening: 1 Writing: 2 Reading: 2 Speaking: 1	Spain	<a href="#">Send message</a>

#### Residence cards

Author	Area	Place	Price	Country	View cards
Sergio <input type="checkbox"/>	Barrio Alto - San Félix - Oliveros - Altamira	Flat	400-500	Spain	<a href="#">Send message</a>

Figura 5.55: Página principal con filtro

¡El filtro funciona como establecimos! Probemos ahora a enviar un mensaje a uno de los estudiantes y veamos el resultado.

The screenshot shows a 'Messages Area' with a message from Ludovico Einaudi and a JSON response from the server:

**Messages Area**

Ludovico Einaudi 05/09/20 20:18

Es un mensaje de prueba

```

-MG8CTCHGVdYcN50kQzy
  aboutStudies: ""
  all_comments
    0
      author
        email: "invirtiendocohelio@gmail.com"
        experience: "true"
        img: "/assets/dist/img/avatar.jpg"
        lastname: "Einaudi"
        name: "Ludovico"
        nativeCountry
          date: "2020-09-05T18:18:14.042Z"
          text: "Es un mensaje de prueba"
      author
      branch
      course
      degree
      idStudent: "-MG84_HIS2_NB-nKC99h"
      votes: 0
  
```

Figura 5.56: Envío de un mensaje

El mensaje se ha enviado correctamente y se ha registrado en la base de datos.



Finalmente y para dar conluir con esta fase de pruebas de funcionamiento, comprobemos que sucede cuando se realiza la votación sobre una tarjeta de información. Si nos fijamos en la figura anterior (5.54), el número de votos almacenado en la base de datos corresponde a 0.

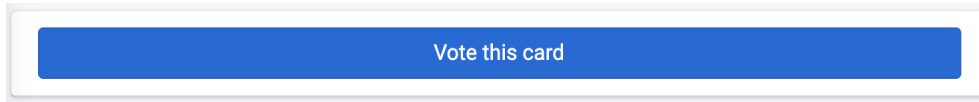


Figura 5.57: Botón para votar una tarjeta

¡Accionemos el botón «Votar»!

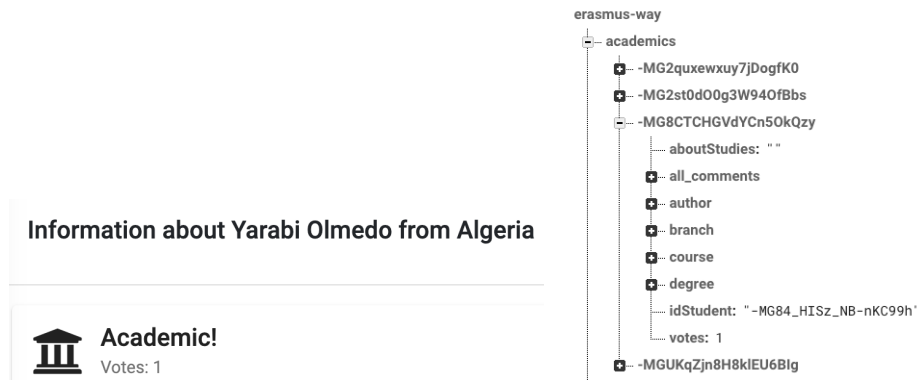


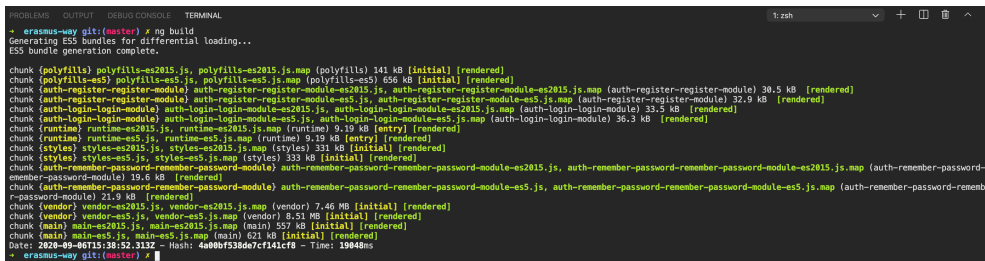
Figura 5.58: Confirmación

Ahora el voto de la tarjeta del usuario ha pasado a ser 1. ¡Funciono correctamente! Además, recordemos que en este aspecto, que únicamente se podrá votar a otros usuarios y que el usuario logueado nunca podrá votarse así mismo.

En las pruebas funcionales hemos comprobado en muy poco tiempo que la aplicación ha sido un éxito y que funciona correctamente.

## 5.11. Despliegue

- Iniciamos el despliegue de la aplicación web[10] emparejando la terminal y aplicación con nuestra cuenta firebase. Para ello, introducimos en nuestra terminal la instrucción «firebase login», seguido de las credenciales de acceso de la plataforma. Previo a este paso, es necesario tener instaladas las herramientas firebase-tools a través de la instrucción «npm install -g firebase-tools».
- Seguido de lo anterior, procederemos a crear una versión optimizada de nuestra aplicación, en la que el código será comprimido, reducido y optimizado para que cargue más rápido, sea de menor tamaño y en general mejore su rendimiento.



```

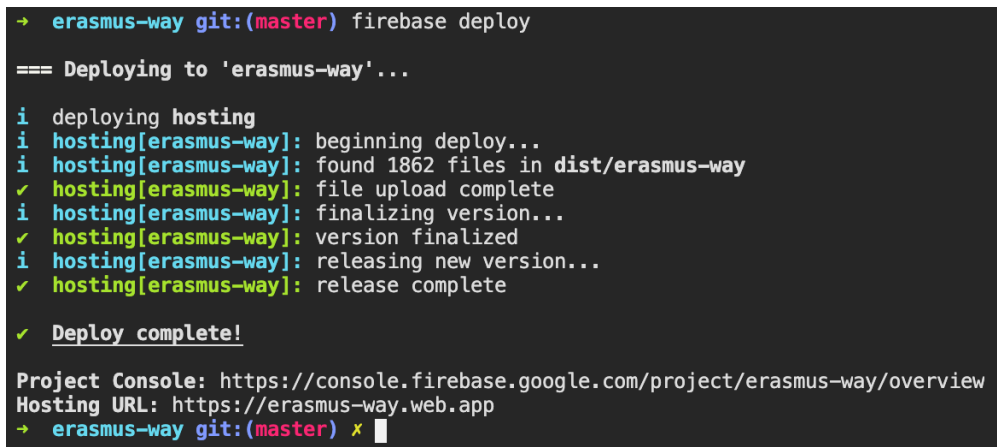
+ erasmus-way git:(master) x ng build
Generating ES5 bundles for differential loading...
ES5 bundle generation complete.

chunk (polyfills) polyfills-es2015.js, polyfills-es2015.js.map (polyfills) 141 kB [initial] [rendered]
chunk (polyfills-es5) polyfills-es5.js, polyfills-es5.js.map (polyfills-es5) 656 kB [initial] [rendered]
chunk (auth-register-register-module) auth-register-register-module-es2015.js, auth-register-register-module-es2015.js.map (auth-register-register-module) 38.5 kB [rendered]
chunk (auth-register-register-module) auth-register-register-module-es5.js, auth-register-register-module-es5.js.map (auth-register-register-module) 32.9 kB [rendered]
chunk (auth-login-login-module) auth-login-login-module-es2015.js, auth-login-login-module-es2015.js.map (auth-login-login-module) 33.5 kB [rendered]
chunk (auth-login-login-module) auth-login-login-module-es5.js, auth-login-login-module-es5.js.map (auth-login-login-module) 36.3 kB [rendered]
chunk (runtime) runtime-es2015.js, runtime-es2015.js.map (runtime) 9.19 kB [entry] [rendered]
chunk (runtime) runtime-es5.js, runtime-es5.js.map (runtime) 9.19 kB [entry] [rendered]
chunk (styles) styles-es2015.js, styles-es2015.js.map (styles) 331 kB [initial] [rendered]
chunk (styles) styles-es5.js, styles-es5.js.map (styles) 353 kB [initial] [rendered]
chunk (auth-remember-password-remember-password-module) auth-remember-password-remember-password-module-es2015.js, auth-remember-password-remember-password-module-es2015.js.map (auth-remember-password-remember-password-module) 19.6 kB [rendered]
chunk (auth-remember-password-remember-password-module) auth-remember-password-remember-password-module-es5.js, auth-remember-password-remember-password-module-es5.js.map (auth-remember-password-remember-password-module) 21.9 kB [rendered]
chunk (vendor) vendor-es2015.js, vendor-es2015.js.map (vendor) 7.46 MB [initial] [rendered]
chunk (vendor) vendor-es5.js, vendor-es5.js.map (vendor) 8.51 MB [initial] [rendered]
chunk (main) main-es2015.js, main-es2015.js.map (main) 557 kB [initial] [rendered]
chunk (main) main-es5.js, main-es5.js.map (main) 621 kB [initial] [rendered]
Date: 2020-09-06T15:38:52.313Z - Hash: 4a80bf5386e7cf141cf8 - Time: 19048ms
+ erasmus-way git:(master) x

```

Figura 5.59: Angular: Versión optimizada

- Desplegamos nuestra aplicación en firebase. Para ello introducimos en la terminal la instrucción «firebase init». Seguimos al asistente seleccionando «Hosting» y el proyecto contenido en la plataforma Firebase, de nombre «erasmus-way».
- Finalizamos con el comando «firebase deploy».



```

→ erasmus-way git:(master) firebase deploy

=== Deploying to 'erasmus-way'...

i deploying hosting
i hosting[erasmus-way]: beginning deploy..
i hosting[erasmus-way]: found 1862 files in dist/erasmus-way
✓ hosting[erasmus-way]: file upload complete
i hosting[erasmus-way]: finalizing version..
✓ hosting[erasmus-way]: version finalized
i hosting[erasmus-way]: releasing new version..
✓ hosting[erasmus-way]: release complete

✓ Deploy complete!

Project Console: https://console.firebase.google.com/project/erasmus-way/overview
Hosting URL: https://erasmus-way.web.app
→ erasmus-way git:(master) x

```

Figura 5.60: Despliegue ErasmusWay

Nuestra aplicación se encontraría desplegada y en producción en la siguiente url:  
<https://erasmus-way.web.app>

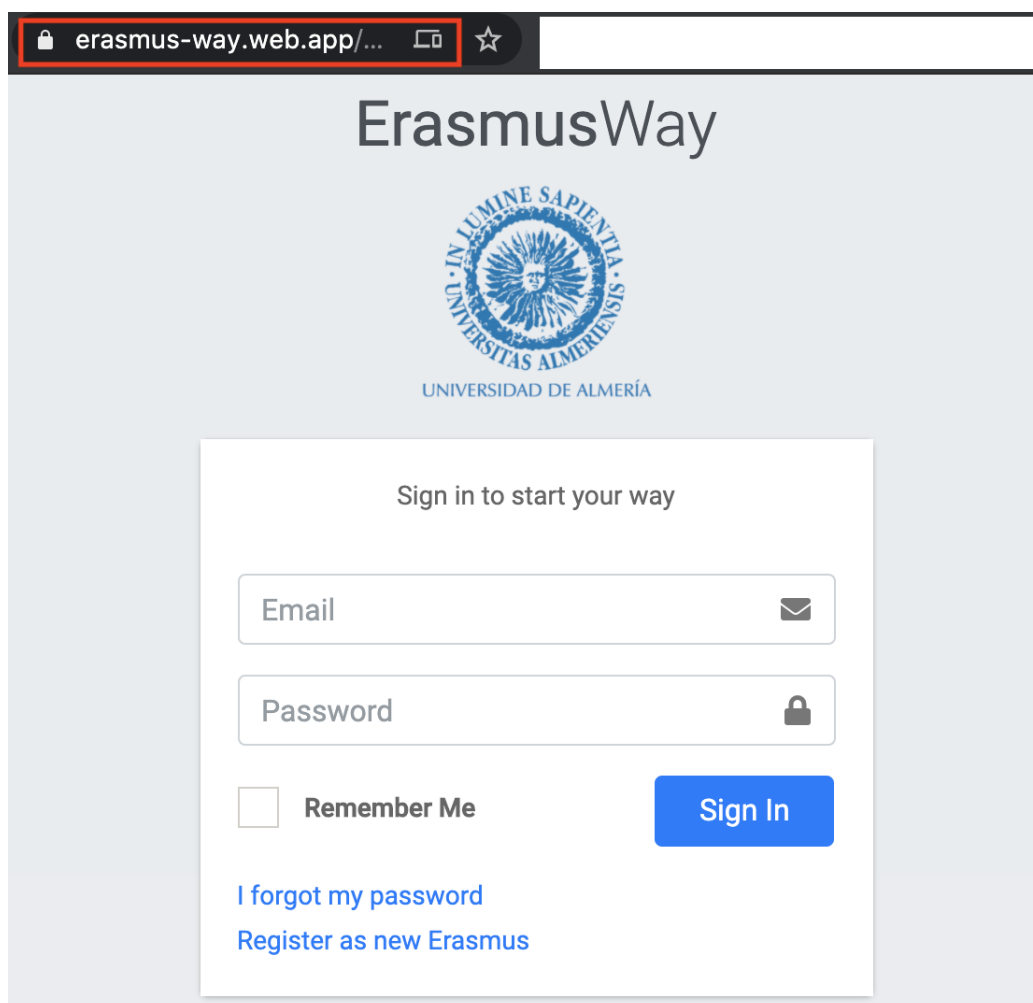


Figura 5.61: ErasmusWay desplegada en producción

## 6 Conclusiones y trabajo futuro

Para concluir la memoria, se exponen las conclusiones y posibles trabajos futuros.

### 6.1. Conclusiones

A priori, el planteamiento a realizar una estancia Erasmus puede suponer al alumno cierto grado de desconocimiento y desconfianza sobre el lugar en el que realizarla. Según el Servicio de Relaciones Internacionales de la Universidad de Almería, los estudiantes plantean multitud de dudas y cuestiones que giran en torno a la cultura del país, su universidad y titulaciones, el idioma, la situación de la vivienda,...etc.

Por ello, se expone el resultado final del proyecto que ha sido la realización de una aplicación basada en tecnologías webs actuales, compatibles con todo tipo de dispositivos y resoluciones, llevando a cabo el desarrollo en todas sus fases, desde el análisis y toma de requisitos, pasando por el diseño y prototipado, la implementación, pruebas y finalizando con el despliegue de la aplicación. Como se indica en los primeros capítulos de esta memoria, la aplicación web va destinada a los estudiantes Erasmus de la Universidad, a los que con la aplicación, de nombre ErasmusWay, se les proporcionará la posibilidad de poder compartir su experiencia con otros estudiantes, así como de comunicarse entre ellos.

Mediante el uso de la aplicación web, por parte de los estudiantes, se pretende aliviar la carga de trabajo informativa del Servicio de Relaciones Internacionales de la Universidad y se pretende poner en el centro a los alumnos que finalizan su estancia Erasmus, convirtiéndolos en embajadores de la misma. De esta manera se procura acercar mucho más a los estudiantes a esta experiencia mediante la creación de un entorno en el que se genera un clima de confianza y colaboración informativa entre iguales.

A la hora del diseño e implementación, se han consultado frecuentemente las páginas oficiales de las herramientas y marcos de trabajo señalados en el capítulo 2. Dichas tecnologías constantemente se actualizan y es de obligada necesidad estar continuamente accediendo a su documentación.

Uno de los puntos más destacables es, sin duda, el nivel de dificultad a la hora de realizar un proyecto con un alcance tan amplio. Por ello, se ha invertido una significativa cantidad de tiempo en investigación para abordar el proyecto de la manera más adecuada y óptima, además de realizar lecturas periódicas de la documentación disponible para saber cómo utilizar las tecnologías seleccionadas y qué posibles funcionalidades implementar dentro de la planificación realizada.

El hecho de realizar el desarrollo con tecnologías web de actualidad ha generado el descubrimiento de una corriente de nuevos conocimientos, así como de la adquisición de una amplia comprensión de los mismos, con especial énfasis en el área de la optimización web. Alcanzar estos conocimientos referentes a la especialización de aplicaciones web puede suponer un importante valor añadido, y generar un enriquecimiento de las habilidades a la hora de llevar a cabo proyectos de un alcance y funcionalidad superior.

## 6.2. Trabajo futuro

Llegados a este punto, se plantean algunos trabajos de interés que podrían complementar y enriquecer la aplicación web recogida en este documento.

- Creación de un sistema de gamificación asociado a la cumplimentación de las tarjetas de información personal que pueda incentivar a los estudiantes a participar en la aplicación ErasmusWay.
- Creación de un sistema de clasificación por puntos con las tarjetas que han recibido más votaciones por parte de otros estudiantes.
- Analizar, detectar e implementar nuevas tarjetas de información que pudieran ser de interés para los estudiantes.
- La aplicación ya establece al inicio una diferenciación entre los alumnos que están actualmente de Erasmus y/o aquellos que ya han finalizado su estancia, así como a los que no han tenido nunca una experiencia Erasmus pero están interesados. Se plantea la posibilidad de crear secciones específicas agrupándolos.

# Bibliografía

- [1] Hooking into the component lifecycle. (2020). Angular.  
<https://angular.io/guide/lifecycle-hooks>
- [2] Introduction to the Angular Docs. (2020). Angular. <https://angular.io/docs>
- [3] B., G. (2019b, mayo 13). ¿Qué es GitHub y para qué se utiliza? Tutoriales Hostinger.  
<https://www.hostinger.es/tutoriales/que-es-github/>
- [4] Otto, M. J. T. (2020). Introduction. Bootstrap.  
<https://getbootstrap.com/docs/4.4/getting-started/introduction/>
- [5] JSON. (2019). How JavaScript Works. <https://www.json.org/json-es.html>
- [6] Firebase Auth REST API. (2020). Firebase.  
<https://firebase.google.com/docs/reference/rest/auth#section-get-account-info>
- [7] Installation & Setup for REST API |. (2020). Firebase.  
<https://firebase.google.com/docs/database/rest/start?authuser=0>
- [8] Cultura y Deporte Secretaría General de Universidades Ministerio de Educación. Estrategia para la internacionalización de las universidades españolas 2015-2020. 2016.
- [9] App o web: Todo está en el estilo de navegación del usuario. (2019). El País Retina.  
[https://retina.elpais.com/retina/2018/08/21/tendencias/1534846474\\_657067.html](https://retina.elpais.com/retina/2018/08/21/tendencias/1534846474_657067.html)
- [10] Saini, S. (2019, 13 diciembre). How to Deploy an Angular App on Firebase. dzone.com.  
<https://dzone.com/articles/how-to-deploy-angular-app-on-firebase>
- [11] Diseño Centrado en el Usuario (DCU). (2019). No Solo Usabilidad.  
<http://www.nosolousabilidad.com/manual/3.htm>
- [12] colaboradores de Wikipedia. (2020, 27 mayo). Firebase. Wikipedia, la enciclopedia libre.  
<https://es.wikipedia.org/wiki/Firebase>
- [13] colaboradores de Wikipedia. (2020b, septiembre 2). Visual Studio Code. Wikipedia, la enciclopedia libre. [https://es.wikipedia.org/wiki/Visual\\_Studio\\_Code](https://es.wikipedia.org/wiki/Visual_Studio_Code)



## Resumen/Abstract

La Comisión de la Unión Europea aprobó en 2013 una estrategia de internacionalización donde propusieron medidas y objetivos para promover la movilidad internacional de estudiantes, instando a las instituciones de educación superior a aplicar estrategias para tal fin.

Cuando los estudiantes extranjeros buscan un destino para iniciar o continuar su formación académica, previamente se documentan. Como nos indica el Servicio de Relaciones Internacionales de la Universidad de Almería, la información más demandada hace referencia a la ciudad, universidad, titulación y asignaturas, alquilar de vivienda y/o idiomas.

En este documento se recoge la solución propuesta para convertir a los antiguos alumnos Erasmus de la Universidad de Almería, en embajadores de la misma a través de una aplicación basada en tecnologías webs actuales, compatibles con todo tipo de dispositivos y resoluciones. Llevaremos a cabo el desarrollo en todas sus fases, desde el análisis y toma de requisitos, pasando por el diseño y prototipado, la implementación, pruebas y finalizando con el despliegue de la aplicación.

In 2013, the European Commission approved an internationalization strategy where they proposed measures and objectives to promote international student mobility, urging higher education institutions to apply strategies for this purpose.

When foreign students search for a destination to start or continue their academic training, it's previously documented. According to the International Relations Service of the University of Almería, the most requested information refers to the city, university, degrees and subjects, housing rental and languages.

This document contains the proposed solution to convert former Erasmus students of the University of Almería into ambassadors through an application based on current web technologies, compatible with all types of devices and resolutions. We will carry out the development in different phases, from the analysis and taking of requirements, going through the design and development of a prototyping, the implementation, tests and ending with the deployment of the application.