

A thick dark blue vertical bar runs along the left edge of the page. A blue arrow-shaped banner points to the right from this bar, containing the text 'Trabajo Fin de Grado'. In the bottom-left corner, there are several thin, curved, light blue lines that sweep upwards and to the right.

Trabajo Fin de Grado

Aplicación web: *Ñam*

Jennifer López Melchor

Ciclo Formativo de Desarrollo de Aplicaciones Web (DAW)

Proyecto Final de Grado Superior

Curso 2020-2021

Instituto Tecnológico Telefónica

ÍNDICE

1. INTRODUCCIÓN	2
2. MÓDULOS FORMATIVOS APLICADOS EN EL TRABAJO.....	3
3. OBJETIVOS DEL PROYECTO.....	6
4. HERRAMIENTAS Y LENGUAJES UTILIZADOS	7
5. FASES DEL PROYECTO	9
5.1. Fase de definición y diseño.....	9
5.2. Fase de modelado de datos.....	10
5.3. Fase de desarrollo	13
<i>5.3.1. Back-End con SpringBoot</i>	<i>13</i>
<i>5.3.2. Front-End con Angular 8</i>	<i>17</i>
<i>5.3.3. Base de datos relacional con PostgreSQL</i>	<i>22</i>
<i>5.3.4. Especificaciones del proyecto y casos de uso</i>	<i>28</i>
6. CONCLUSIONES Y MEJORAS DEL PROYECTO	41
7. BIBLIOGRAFÍA	44

1. INTRODUCCIÓN

Nam es una aplicación cuya temática principal es la alimentación y cuya filosofía surge de motivaciones personales. Desde mi punto de vista y como experiencia personal, para alimentarse de manera equilibrada se requiere organización y tiempo, todo lo contrario a la improvisación a la que solemos recurrir en el día a día y que, finalmente nos lleva a una alimentación basada en ultra-procesados, con el riesgo que esto supone para la salud.

Por tanto, la idea inicial es que esta aplicación sea el comienzo de un ejercicio de responsabilidad y organización a través del cual se mejoren los hábitos alimenticios del usuario.

Así mismo, también se pretende que la aplicación sirva de inspiración al usuario, en el sentido de que pueda descubrir y aprender a cocinar nuevas recetas, dejando a un lado la monotonía que a veces supone cocinar los alimentos de una única forma y haciendo que su relación con la comida sea más satisfactoria.

Pero, ante todo, este proyecto supone un ejercicio de aprendizaje, de culminación de una etapa que, a todas luces, ha sido un ejercicio de superación, tanto personal como profesional.

2. MÓDULOS FORMATIVOS APLICADOS EN EL TRABAJO

En primer lugar, es preciso destacar que todos los módulos formativos del grado superior han sido de utilidad durante la realización de este proyecto, pues han supuesto la base de los conocimientos que se ven reflejados en este trabajo.

Sin embargo, existen una serie de asignaturas que han cobrado especial relevancia durante todo este proceso, como son las siguientes:

- **Bases de datos:**

Toda la información de la aplicación (sobre usuarios, recetas, ingredientes, categorías, etc) está almacenada en una base de datos de carácter relacional sobre la que se hacen las correspondientes operaciones CRUD (*Create, Read, Update, Delete*). Para ello, se ha utilizado SQL como lenguaje de consultas y PostgreSQL como sistema gestor de nuestra base de datos.

- **Lenguajes de marcas y sistemas de gestión de información:**

En este sentido, el uso de HTML (*HyperText Markup Language*) y CSS (*Cascading Style Sheets*) han sido claves para el desarrollo del proyecto. Cada pantalla de la aplicación se corresponde con un componente de nuestro Front-End en Angular, y cada uno de estos componentes tiene, entre otros, un archivo HTML y otro CSS que nos permiten dar forma y estilo a la parte visual de la aplicación. Por tanto, los contenidos impartidos en esta asignatura han sido de gran utilidad.

- **Diseño de interfaces web:**

Los conocimientos adquiridos en el curso de esta asignatura también han sido de gran utilidad, especialmente para la realización del *Mock-up* del proyecto, que ha permitido dar forma a la aplicación y clarificar cuáles serían las funcionalidades a implementar y el esqueleto de cada pantalla. Por otro lado, se ha tenido en consideración el cumplimiento

de los criterios de usabilidad y accesibilidad necesarios, así como un diseño amigable y estilo apropiado para mejorar la experiencia del usuario.

- **Despliegue de aplicaciones web:**

En relación al despliegue de la aplicación, se han aprovechado las funcionalidades de las tecnologías usadas, es decir, en cuanto al Back-End, Spring tiene embebido un servidor Tomcat que nos permite desplegar la aplicación localmente desde el propio IDE (*Integrated Development Environment*).

Y en cuanto al Front-End con Angular, aprovechamos que también dispone de un servidor propio que nos facilita el despliegue de manera local, utilizando simplemente los comandos *ng build* y *ng serve*.

Sin embargo, a pesar de que las tecnologías usadas facilitan enormemente el despliegue de la aplicación de manera casi automatizada, ha sido necesario recurrir a los contenidos de esta asignatura para entender cómo funciona el proceso de despliegue.

- **Programación:**

Podríamos afirmar que esta asignatura ha supuesto uno de los pilares fundamentales sobre la que se sustenta el desarrollo de este proyecto, ya que para el desarrollo e implementación de cada una de las funcionalidades de la aplicación, se ha tenido que recurrir a los conocimientos sobre programación orientada a objetos adquiridos en esta asignatura, reforzando y ampliando dichos conocimientos a través de la práctica con los lenguajes de Java y Typescript.

- **Desarrollo web en entorno cliente:**

Del mismo modo, han sido de gran utilidad los conocimientos sobre Javascript adquiridos a través de esta asignatura para el desarrollo del Front-End de la aplicación. Aunque el

lenguaje de programación que utilizamos para el desarrollo *front* con Angular ha sido Typescript, es necesario destacar que es muy parecido a Javascript, por lo que la base del conocimiento con este lenguaje ha sido fundamental.

- **Desarrollo web en entorno servidor:**

También han resultado indispensables los conocimientos sobre Java adquiridos en el transcurso de esta asignatura para el desarrollo del Back-End de la aplicación, así como los fundamentos de JPA (*Java Persistence API*), que nos han permitido entender la correlación entre un sistema orientado a objetos y una base de datos de carácter relacional.

- **Entornos de desarrollo:**

El contenido de esta asignatura ha sido relevante, por un lado, en cuanto a la correcta utilización de un IDE, que en este caso es *IntelliJ Idea* y el conocimiento de las herramientas principales que este ofrece, como por ejemplo, utilizar el IDE en modo *debug* para ir siguiendo, paso a paso, la ejecución del código. Así mismo, los contenidos de esta asignatura también han supuesto de utilidad para llevar un control de versiones del proyecto a través de Git.

3. OBJETIVOS DEL PROYECTO

En este apartado, haremos una diferenciación entre objetivos generales o globales del proyecto, y otros más específicos.

- **Objetivos generales:**

- Afianzar el aprendizaje de los contenidos impartidos en las distintas asignaturas del grado superior.
- Conocer el flujo real que sigue una aplicación web, desde el entorno del cliente (Front-End), pasando por el entorno servidor (Back-End) hasta su interacción con la base de datos.
- Desarrollar una aplicación completa y funcional.
- Desarrollar una aplicación web con un diseño atractivo y funcional, que proporcione una experiencia de usuario amigable.
- Establecer y organizar tiempos de trabajo para cada una de las etapas que conforman el proceso de diseño y desarrollo de la aplicación.

- **Objetivos específicos:**

- Conocer y aplicar los fundamentos básicos de Angular para el desarrollo del Front-End de la aplicación.
- Conocer y aplicar los fundamentos básicos de Spring para el desarrollo del Back-End de la aplicación.
- Elaborar un modelo de datos lógico y funcional, que proporcione la mayor escalabilidad posible a la aplicación, con vistas a futuras ampliaciones y cambios.
- Establecer conexiones entre el entorno cliente (Front-End), servidor (Back-End) y la base de datos.
- Aplicar, de forma práctica, las operaciones CRUD (*Create, Read, Update y Delete*).
- Desarrollar un Back-End capaz de proveer al entorno cliente de los datos necesarios para cada funcionalidad de la aplicación, evitando sobrecargar las respuestas de las peticiones con datos innecesarios.

4. HERRAMIENTAS Y LENGUAJES UTILIZADOS

En este apartado es necesario mencionar, en primer lugar, que el desarrollo de la aplicación está dividido en dos sub-proyectos: el Front-End, es decir, el entorno cliente, y el Back-End o entorno servidor. Ambos se han desarrollado como proyectos independientes, utilizando tecnologías y herramientas distintas, y conectándose entre ellos a través de peticiones REST, mediante el protocolo HTTP.

Por un lado, para el desarrollo del Front-End de la aplicación, se ha utilizado el lenguaje Typescript, apoyándonos en el *framework* Angular 8. Así mismo, es preciso destacar el uso de HTML, CSS y Bootstrap para el diseño de cada uno de los componentes que conforman la aplicación.

En cuanto al Back-End, el lenguaje utilizado ha sido Java, apoyándonos en el *framework* SpringBoot.

Por otro lado, se ha utilizado SQL (*Structured Query Language*) para realizar las respectivas operaciones CRUD sobre nuestra base de datos relacional; Y PostgreSQL como sistema gestor de la base de datos.

Entre otras herramientas utilizadas, destacamos Postman, que nos ha permitido probar de manera sencilla e intuitiva las peticiones realizadas sobre nuestra aplicación.

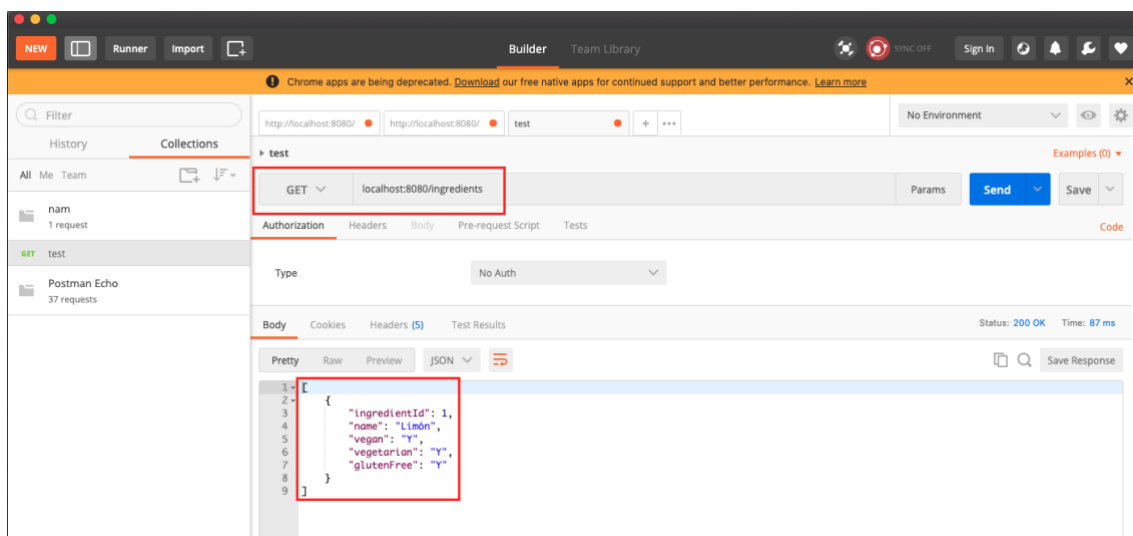


Imagen 1. Ejemplo de uso de Postman

Por otro lado, como IDE se ha utilizado IntelliJ IDEA, que también nos ha proporcionado una experiencia de usuario intuitiva y sencilla, detectando las configuraciones para cada *framework* y facilitando su compilación y despliegue.

Y para llevar un control de versiones durante todo el proceso de desarrollo de la aplicación se ha utilizado Git, subiendo nuestro proyecto y sus respectivos cambios a un repositorio remoto, que en este caso ha sido GitHub, al que se puede acceder desde el siguiente enlace:

<https://github.com/jlmelchor?tab=repositories>

En resumen, mostramos una tabla con las principales tecnologías y herramientas utilizadas durante todo el proceso:

Fase de definición y diseño						
Elaboración del <i>Mock-up</i>	AdobeXD					
Elaboración del DER	https://app.diagrams.net/					
Fase de desarrollo						
Front-End	Angular 8	Typescript	HTML5	CSS3	Bootstrap	SweetAlert
Back-End	SpringBoot		Java		Maven	
Base de datos	PostgreSQL			SQL		
Otras herramientas y tecnologías utilizadas durante todo el proceso						
Postman	Visual Studio Code	IntelliJ Idea		Git		GitHub

Tabla 1. Herramientas, lenguajes y tecnologías utilizados durante el desarrollo del proyecto

5. FASES DEL PROYECTO

En este apartado, trataremos de definir cada una de las etapas seguidas desde la fase de diseño y modelado de datos hasta el resultado final.

5.1. Fase de definición y diseño

En primer lugar, se procedió a elaborar un *Mock-up*, es decir, un boceto o maqueta de nuestra aplicación, definiendo de manera precisa el diseño de cada una de las pantallas y concretando las funcionalidades a implementar en cada una de ellas. Para ello, se ha utilizado AdobeXD.

El resultado final se puede ver a través del siguiente enlace:

<https://xd.adobe.com/spec/4cb04619-b01f-4929-72eb-cf44b5a28117-3a81/>

Como se puede observar, se trata de un *Mock-up* que permite no sólo visualizar el diseño de la aplicación, sino navegar entre sus diferentes pantallas e intuir sus funcionalidades.

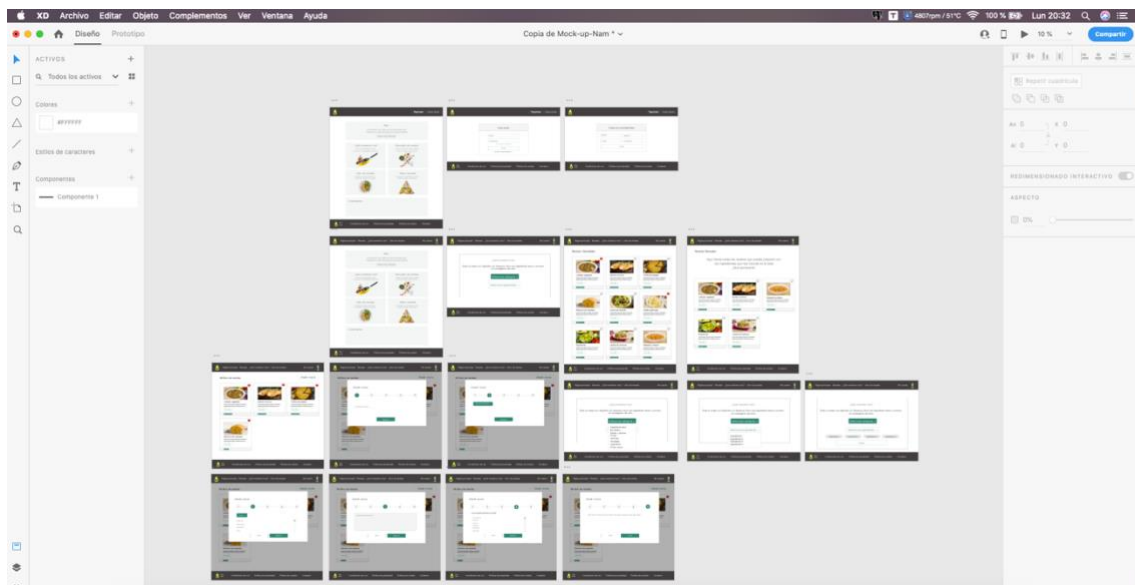


Imagen 2. Uso de AdobeXD para la elaboración del Mock-Up

Esta primera fase nos permitió disponer de una base sólida sobre la que comenzar el desarrollo de la aplicación, conociendo al detalle cómo sería su diseño y las funcionalidades a implementar.

5.2. Fase de modelado de datos

Posteriormente, se procedió a elaborar un Diagrama Entidad-Relación (DER), que permitió dar forma al modelo de datos sobre el que se sustenta nuestra aplicación.

Para ello, se ha utilizado una herramienta de Google que nos permite crear diagramas de manera sencilla y eficiente:

<https://www.draw.io/>

El DER de nuestra aplicación es el siguiente:

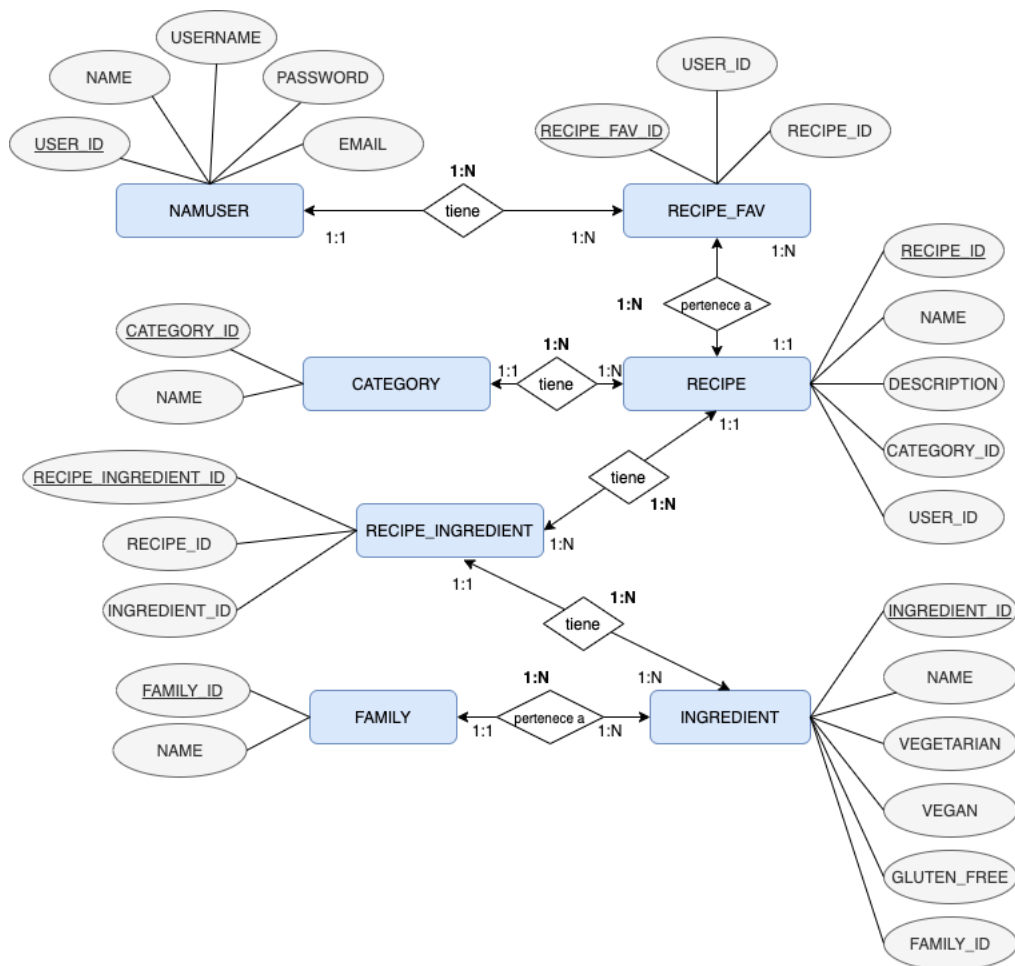


Imagen 3. Diagrama Entidad-Relación Ñam

En azul observamos las entidades del diagrama, que se corresponden con el nombre de las tablas de nuestro modelo de datos. Y en color gris, destacamos los atributos de cada entidad, que se corresponden con las columnas de cada una de las tablas. Pasamos a concretar brevemente cada una de estas entidades:

ENTIDAD	ATRIBUTOS	DEFINICIÓN
NAMUSER (Usuario de la aplicación)	USER_ID	<i>Primary key:</i> Atributo clave e identificador único.
	NAME	Nombre del usuario.
	USERNAME	<i>Login</i> del usuario, es decir, el nombre con el que iniciará sesión en la aplicación.
	PASSWORD	Contraseña del usuario.
	EMAIL	Email del usuario.
RECIPE_FAV (Receta favorita del usuario)	RECIPE_FAV_ID	<i>Primary key:</i> Identificador único.
	USER_ID	<i>Foreign key:</i> Clave ajena, identificador del usuario (de la entidad NAMUSER).
	RECIPE_ID	<i>Foreign key:</i> Clave ajena, identificador de la receta (de la entidad RECIPE).
RECIPE (Receta)	RECIPE_ID	<i>Primary key:</i> Identificador único.
	NAME	Nombre de la receta.
	DESCRIPTION	Descripción de la receta.
	CATEGORY_ID	<i>Foreign key:</i> Clave ajena, identificador de la categoría a la que pertenece la receta (de la entidad CATEGORY).
	USER_ID	<i>Foreign key:</i> Clave ajena, identificador del usuario al que pertenece la receta (de la entidad NAMUSER). Si la receta es añadida por el propio usuario, este campo estará relleno, para indicar que la receta es exclusiva del usuario en cuestión.
CATEGORY (Categoría a la que pertenece cada receta)	CATEGORY_ID	<i>Primary key:</i> Identificador único.
	NAME	Nombre de la categoría (por ejemplo: plato principal)
RECIPE_INGREDIENT (Asociación entre recetas e ingredientes)	RECIPE_INGREDIENT_ID	<i>Primary key:</i> Identificador único.
	RECIPE_ID	<i>Foreign key:</i> Clave ajena, identificador de la receta (de la entidad RECIPE).
	INGREDIENT_ID	<i>Foreign key:</i> Clave ajena, identificador del ingrediente (de la entidad INGREDIENT).
INGREDIENT (Ingrediente)	INGREDIENT_ID	<i>Primary key:</i> Identificador único.
	NAME	Nombre del ingrediente.
	VEGETARIAN	‘Y’ o ‘N’, si el ingrediente es apto para vegetarianos.
	VEGAN	‘Y’ o ‘N’, si el ingrediente es apto para veganos.

	GLUTEN_FREE	‘Y’ o ‘N’, si el ingrediente está libre de gluten o no.
	FAMILY_ID	<i>Foreign key</i> : Clave ajena, identificador de la familia a la que pertenece el ingrediente (de la entidad FAMILY).
FAMILY (Familias de alimentos)	FAMILY_ID	Identificador único de la familia.
	NAME	Nombre de la familia (por ejemplo, fruta)

Tabla 1: Modelo de datos

Una vez definidos tanto el diseño como el modelo de datos de la aplicación, ya disponemos de un “esqueleto”, una base sólida sobre la que se sustentará el desarrollo posterior.

5.3. Fase de desarrollo

En esta tercera fase se procede al desarrollo propio de la aplicación.

Después de invertir un tiempo en formación sobre los *frameworks* de SpringBoot y Angular 8, se procedió a realizar las configuraciones iniciales para cada uno de ellos. A continuación, describimos, en líneas generales, los primeros pasos que se siguieron tanto para el Front como para el Back-End de la aplicación:

5.3.1. Back-End con SpringBoot

Para la configuración inicial del Back-End de la aplicación, en primer lugar accedimos a Spring Initializr (<https://start.spring.io/>) e indicamos las características generales con las que queremos configurar nuestro proyecto:

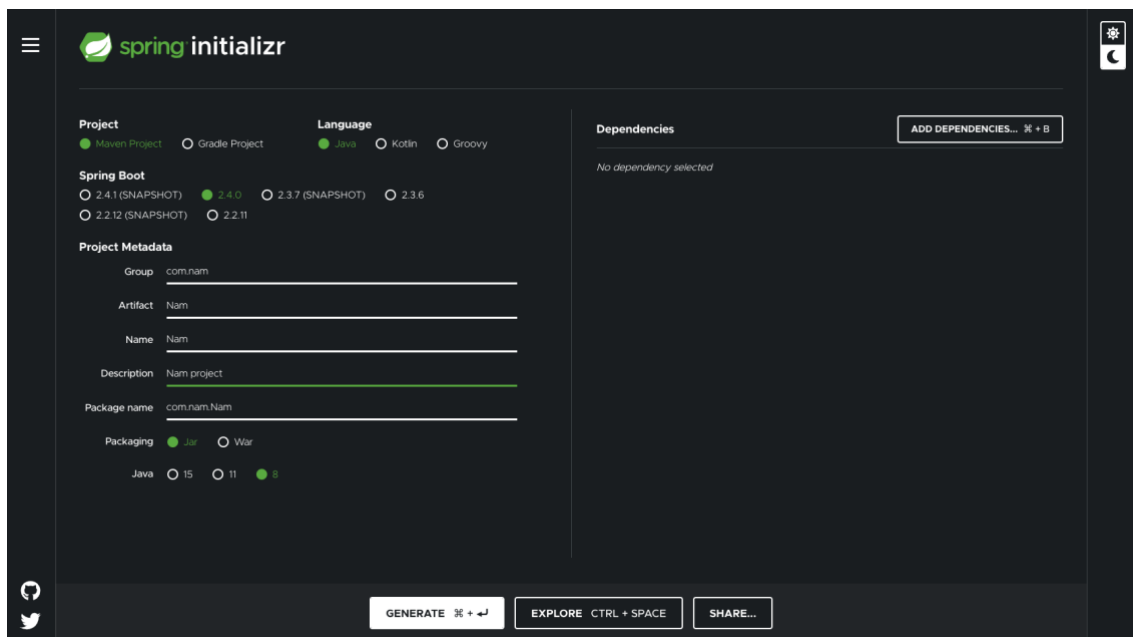


Imagen 4. Spring Initializr

Esto nos facilitó un descargable que contiene la base del proyecto con las configuraciones necesarias.

Como se puede observar en la imagen, se ha seleccionado Maven como gestor del proyecto, de todas nuestras librerías y dependencias (para ello, previamente hubo que instalar Maven y añadirlo como variable de entorno). Esto implica, entre otras cosas, que el proyecto dispondrá de un fichero llamado *POM.xml* (*Project Object Model*), que será el encargado de almacenar la información relativa a todas las dependencias de nuestro

proyecto. Además, Maven nos servirá como automatizador de ciertas tareas, como la compilación del código y su empaquetado, por lo que será de gran utilidad.

En el propio IDE, podemos ver un apartado en el que estas tareas se pueden hacer de forma gráfica e intuitiva con un simple *click*:

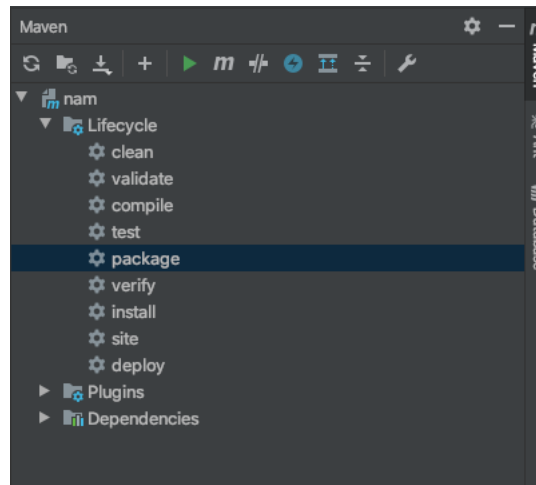


Imagen 5. Maven desde el IDE

Por otro lado, siguiendo con las configuraciones iniciales del proyecto, seleccionamos Java 8 como lenguaje de programación y Jar como tipo de empaquetado para nuestro proyecto.

Una vez hecho esto y generada la base del proyecto con Spring Initializr, abrimos con IntelliJ el archivo *POM.xml* del proyecto y ejecutamos desde terminal el comando *mvn clean install*, que nos permitirá limpiar todas las clases compiladas del proyecto e instalar el artefacto en el repositorio local.

En este momento, ya tenemos configurado el proyecto correctamente y podemos empezar con el desarrollo. Para ello, creamos nuestro primer controlador con el primer *endpoint* de prueba de nuestra aplicación (“Hello world”):

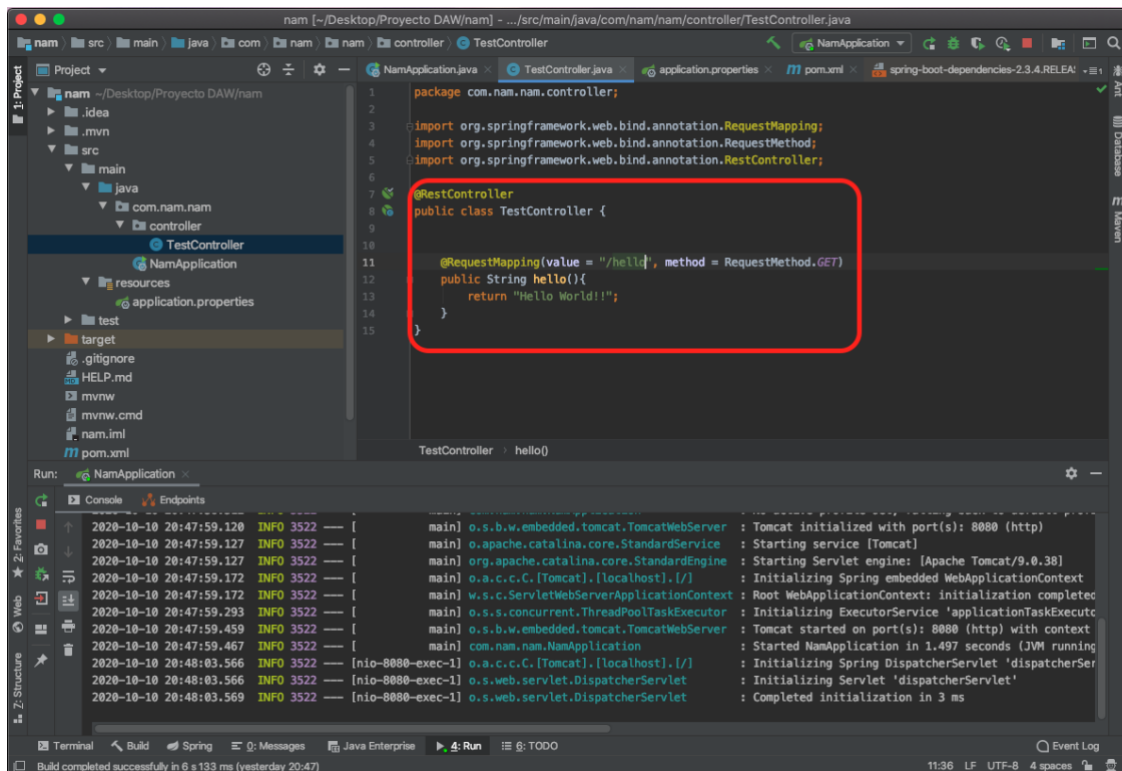


Imagen 6. Primer controlador con Spring

Con la automatización que nos proporciona el IDE y utilizando el servidor Tomcat embebido en Spring, hacemos *build* del proyecto y vemos que se despliega correctamente en el puerto 8080:

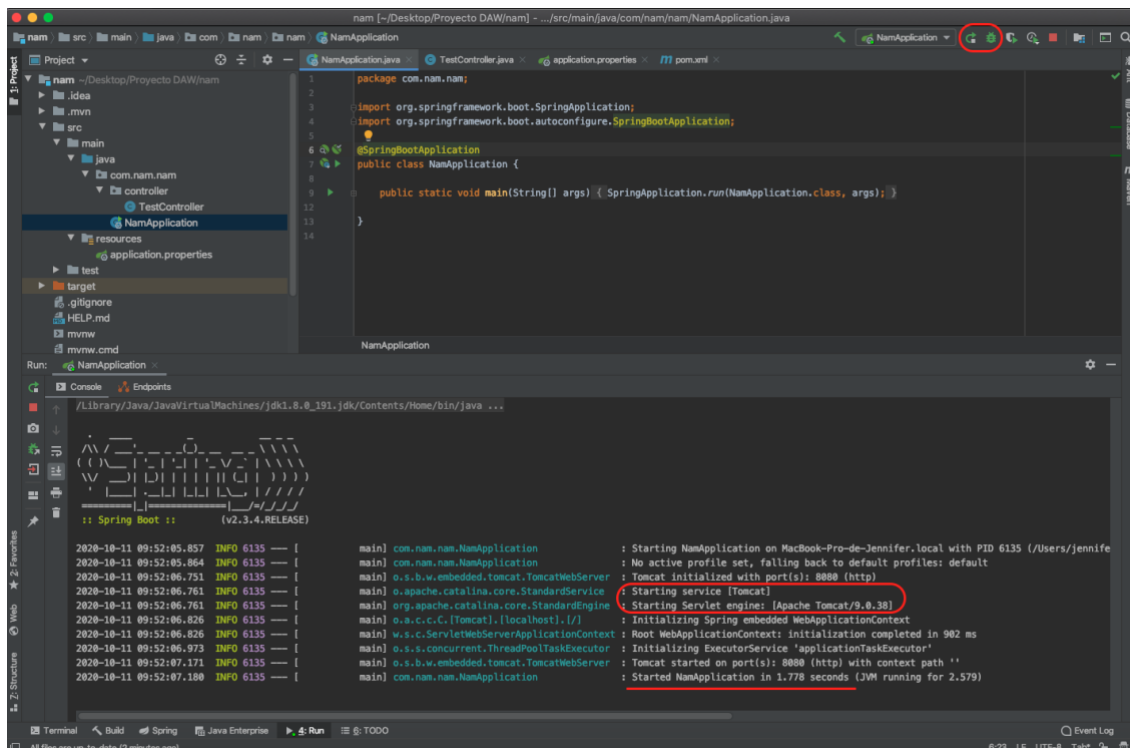


Imagen 7. Despliegue local del Back-End en el puerto 8080

Para probar que nuestro primer controlador funciona correctamente, hacemos uso de Postman:

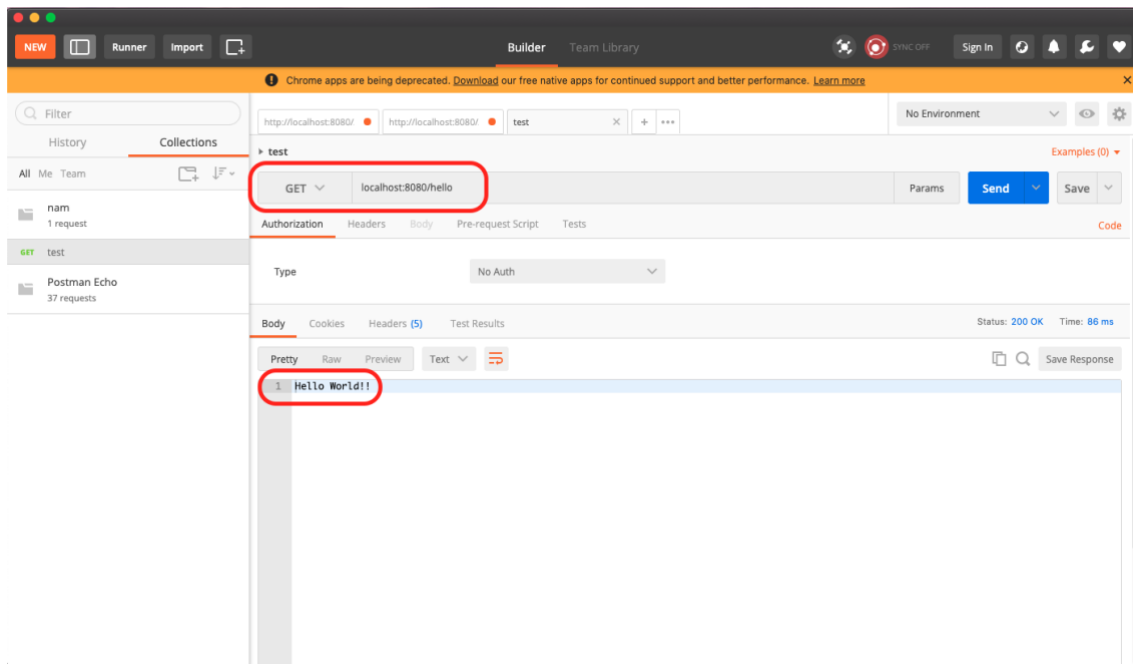


Imagen 8. Prueba con Postman

Hacemos una petición *Get* al *endpoint* que acabamos de crear, apuntando al puerto 8080 (que es donde se ha levantado nuestro proyecto) y podemos observar que, tanto el estado como el contenido de la respuesta son los esperados.

5.3.2. Front-End con Angular 8

En cuanto al front-end de la aplicación, en primer lugar se hicieron las instalaciones necesarias: Node JS, Typescript y Angular CLI.

Para generar la que será la base de nuestro proyecto, simplemente desde la terminal y apuntando al directorio de nuestra preferencia, ejecutamos el comando `ng new nam-front` (donde *nam-front* sería el nombre de nuestro proyecto). Con ello, se nos instalan todos los paquetes necesarios para el desarrollo de una aplicación con Angular 8:

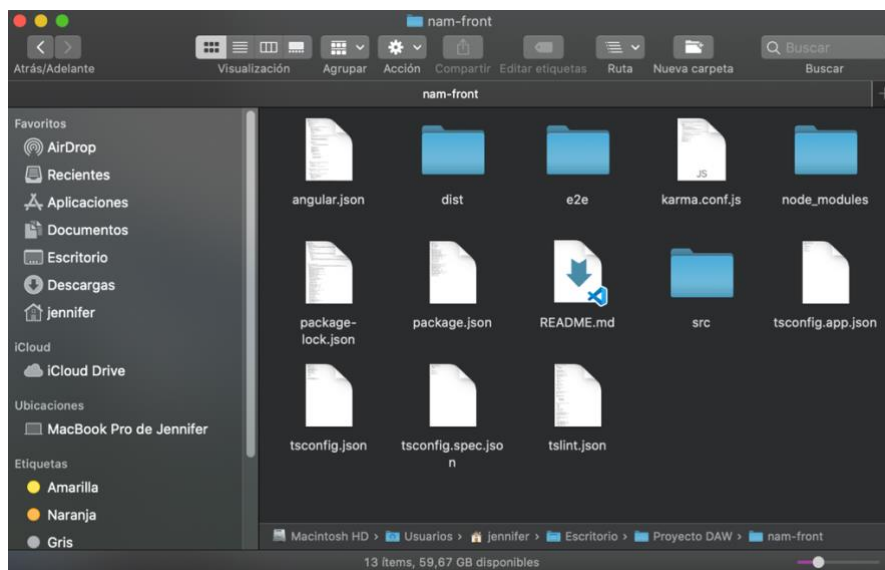


Imagen 9. Directorios generados tras lanzar el comando `ng new nam-front` en Angular

Con esto, ya podemos desplegar nuestra aplicación de Angular, simplemente ejecutando desde la terminal el comando `ng serve`. Esto nos levantará un servidor local en el puerto 4200:

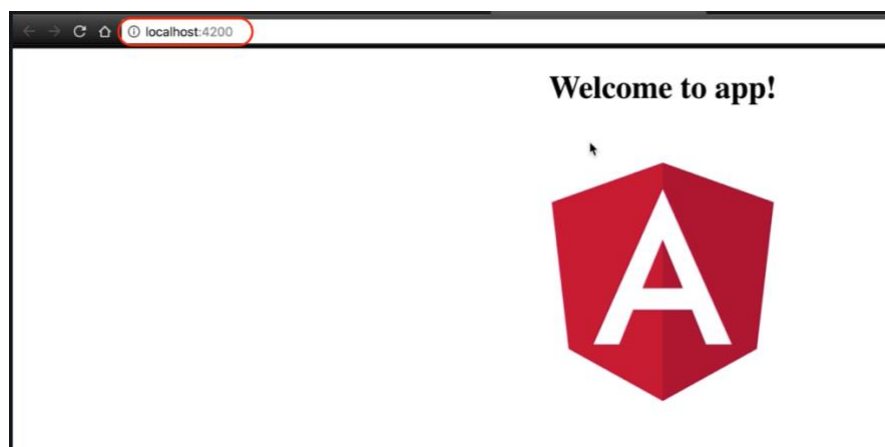


Imagen 10. Despliegue local del Front-End con Angular en el puerto 4200

En la estructura del proyecto, hay varios archivos de especial relevancia y que es necesario mencionar:

angular.json: básicamente, en este archivo se incluye el esqueleto de la aplicación. Por ejemplo, en el apartado de “*styles*” indicaremos la fuente de los estilos que vamos a usar para el desarrollo de la aplicación, como es el caso de Bootstrap:

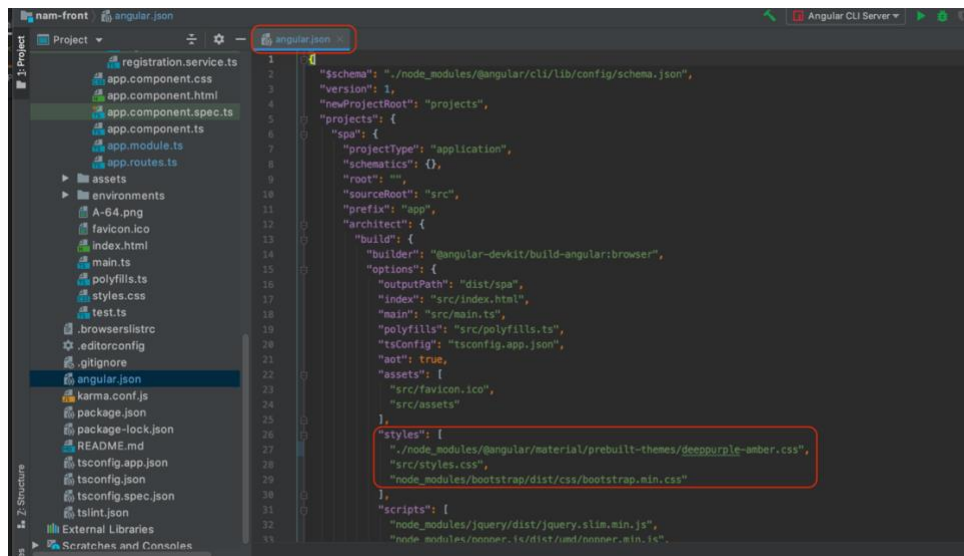


Imagen 11. Contenido del archivo angular.json

package.json: generalmente, no realizaremos modificaciones sobre este archivo, pero es importante conocerlo porque aquí se incluyen todas las dependencias de nuestro proyecto:

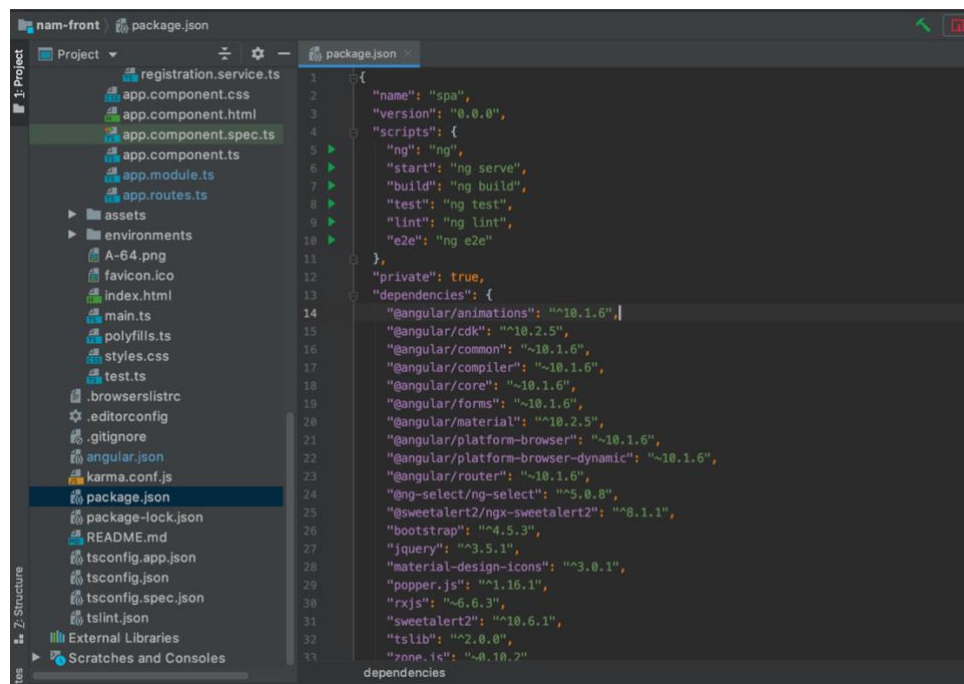


Imagen 12. Contenido del archivo package.json

Debido a la relevancia de estos archivos y por ser el directorio donde vamos a invertir casi la totalidad del tiempo de desarrollo, es importante conocer, a grandes rasgos, el contenido de la carpeta *src* del proyecto.

Como fichero principal, destacamos el *index.html*, que constituye el archivo raíz del proyecto y el punto de entrada a nuestra aplicación.

En este punto, es preciso destacar que Angular trabaja con el concepto de componente, de modo que cada pantalla de nuestra aplicación se corresponderá con un componente.

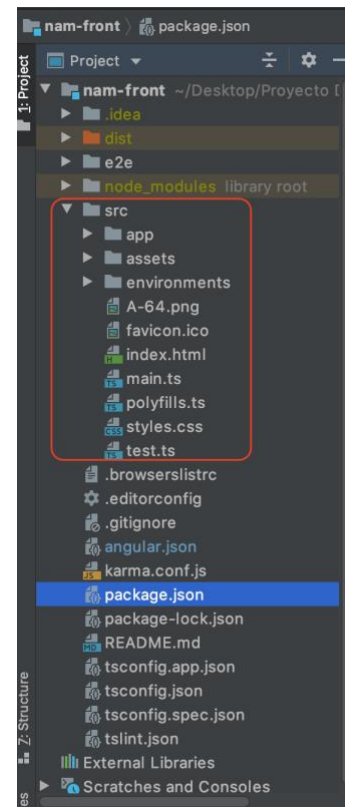


Imagen 13. Estructura de directorios del front-end

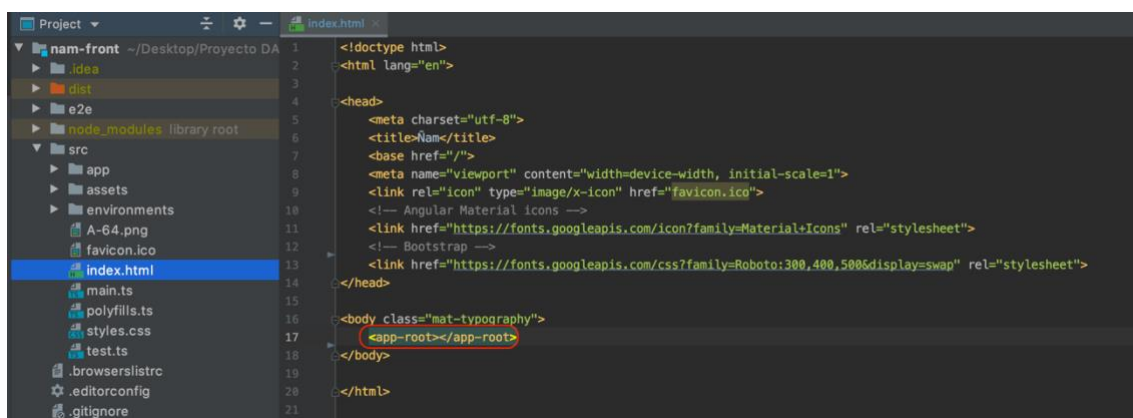


Imagen 14. Contenido del archivo *index.html*

Tal y como se observa en la imagen anterior, dentro del *<body>* del archivo, tenemos la referencia al componente *<app-root>*, que básicamente, nos dirige al que será el componente principal de nuestro proyecto:

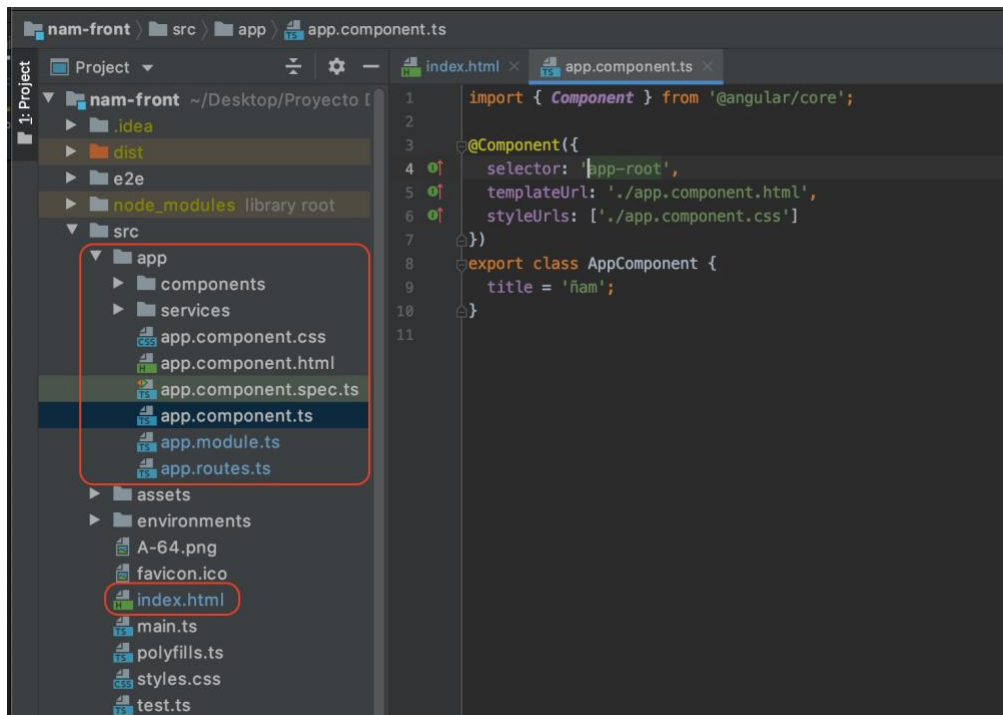


Imagen 15. Contenido del archivo `app.component.ts` y estructura de directorios

Una vez que nos hemos familiarizado, de manera global, con la estructura del proyecto, procedemos a crear nuestros primeros componentes. Para ello, basta con introducir el comando `ng generate component (nombre)` desde la terminal del propio IDE, y automáticamente se genera el componente con cuatro archivos:

- HTML: el archivo con este formato nos permitirá maquetar y dar forma a cada componente.
- CSS: nos servirá para dar estilo al componente.
- TS: se trata de un archivo en formato Typescript, en el que incluiremos la lógica y el tratamiento de los datos a usar en el componente.
- SPEC.TS: para la elaboración de tests unitarios sobre nuestro componente.

Y para desplegar el proyecto, como apuntábamos anteriormente, basta con introducir en la terminal del IDE (sobre la carpeta raíz de nuestro proyecto) los comandos `ng build` y `ng serve`. Vemos que el proyecto se levanta correctamente en el puerto 4200 y nos muestra los cambios realizados en los primeros componentes de nuestra aplicación:

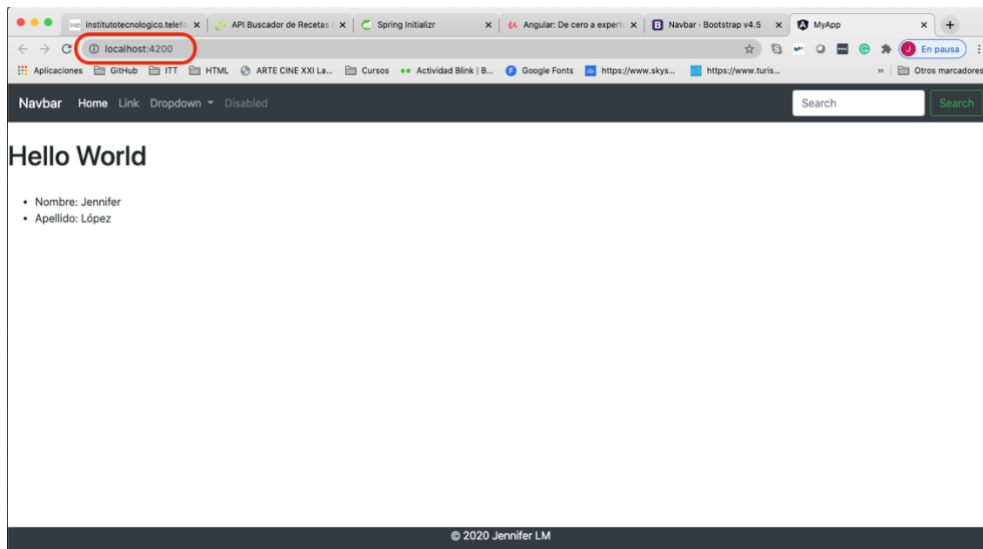


Imagen 16. Despliegue local del Front-End

Finalmente, es necesario destacar otro concepto fundamental de Angular: los servicios. Los servicios nos servirán para establecer conexiones entre componentes (por ejemplo, si necesitamos transferir datos entre ellos) y también serán el nexo de unión con nuestro Back-End. Es a través de los servicios donde, utilizando el protocolo HTTP, haremos la correspondiente petición (get, post, put o delete) al endpoint de nuestro Back-End que corresponda:

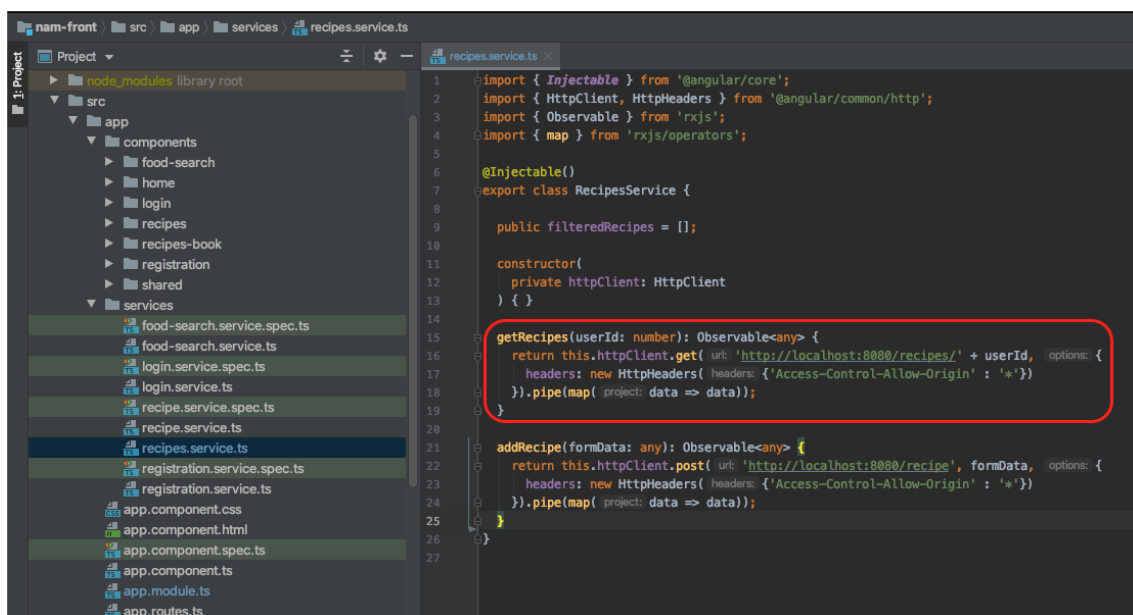


Imagen 17. Contenido del servicio de recetas

Por tanto, como se puede observar en la estructura de directorios, tenemos dos carpetas claramente diferenciadas: la de componentes, en la que se incluye un componente por cada pantalla de la aplicación y la de servicios que consume cada componente.

5.3.3. Base de datos relacional con PostgreSQL

Para establecer la conexión con la base de datos, en primer lugar añadimos las dependencias de PostgreSQL y JPA al archivo *POM.xml* de nuestro Back-End:

```
<dependency>
  <groupId>org.postgresql</groupId>
  <artifactId>postgresql</artifactId>
  <scope>runtime</scope>
</dependency>

<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
```

Imagen 18. Dependencias PostgreSQL y JPA en el archivo *POM.xml*

Después, basta con añadir en el archivo *application.properties* del proyecto la *url*, el nombre de usuario y la contraseña de nuestra base de datos:

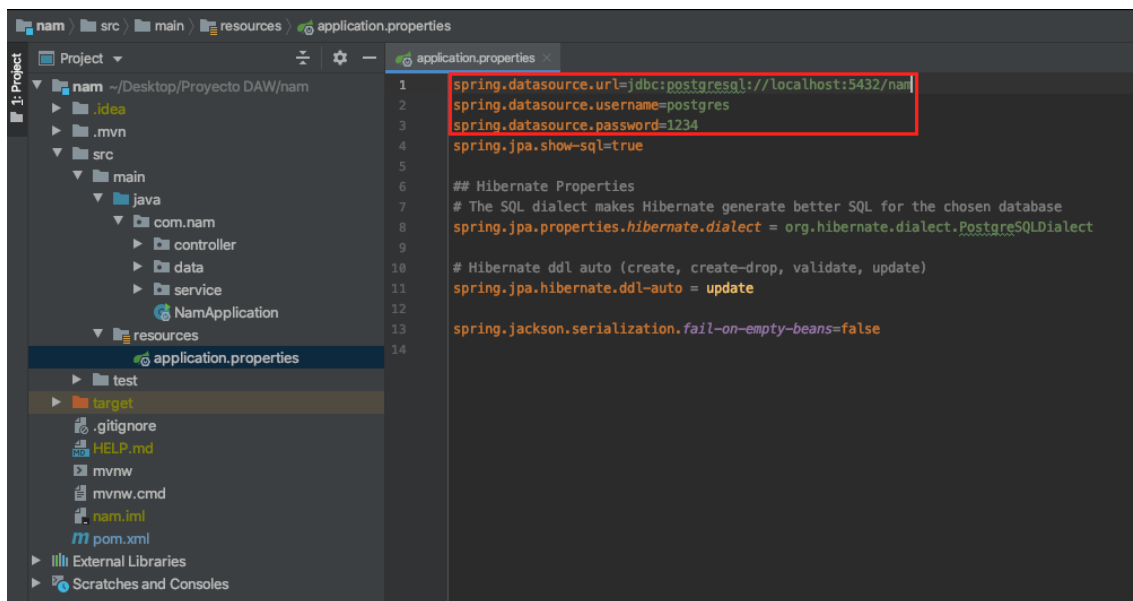


Imagen 19. Datos para la conexión con PostgreSQL en el archivo *application.properties*

Y a partir de este momento, ya está configurada la conexión de nuestro Back-End con la base de datos. Para comprobarlo, creamos una entidad a la que llamamos “NamUser”, que hará referencia al usuario de la aplicación:

```

1 package com.nam.data.model;
2
3 import javax.persistence.*;
4
5 @Entity
6 @Table(name = "NAM_USER")
7 public class NamUser {
8
9     @Id
10    @GeneratedValue(strategy = GenerationType.IDENTITY)
11    @Column(name = "USER_ID", nullable = false)
12    private long userId;
13
14    @Column(name = "NAME", nullable = false)
15    private String name;
16
17    @Column(name = "SURNAME", nullable = false)
18    private String surName;
19
20    @Column(name = "PASSWORD", nullable = false)
21    private String password;
22
23    @Column(name = "EMAIL", nullable = false)
24    private String email;
25
26    public NamUser() {
27
28    }
29

```

Imagen 20. Entidad NamUser

Utilizamos las anotaciones correspondientes que nos proporciona Springboot para indicar que esa clase es una entidad (@Entity), el nombre de la tabla a la que hará referencia dicha entidad (@Table), las columnas que tendrá la tabla y que se corresponderán con los atributos de la clase (@Column), el identificador de la entidad (@Id)...

Posteriormente, creamos un repositorio para dicha entidad (utilizamos la anotación @Repository) y le decimos que extienda de JPA, indicando el nombre de la entidad a la que hará referencia (en este caso, NamUser):

```

1 package com.nam.data.repository;
2
3 import com.nam.data.model.NamUser;
4 import org.springframework.data.jpa.repository.JpaRepository;
5 import org.springframework.stereotype.Repository;
6
7 @Repository
8 public interface UserRepository extends JpaRepository<NamUser, Long> {
9
10 }
11

```

Imagen 21. Repositorio de la entidad NamUser

Y si compilamos nuestro proyecto, podremos ver cómo se actualiza automáticamente la base de datos y aparece la tabla que acabamos de crear:

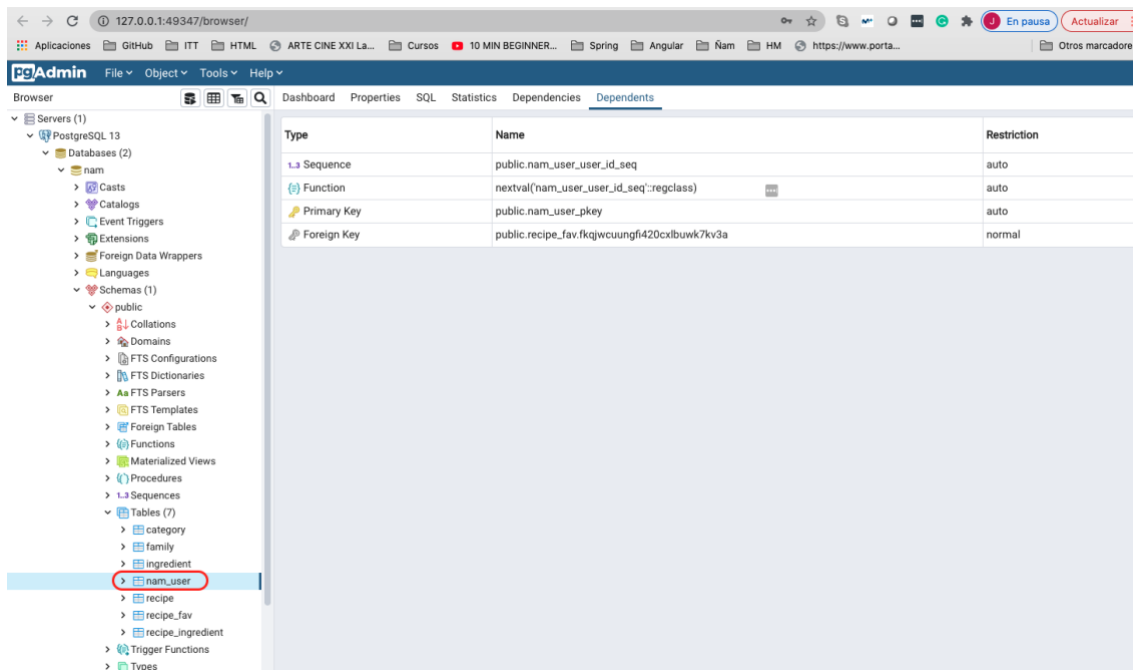


Imagen 22. Actualización automática de la BBDD al compilar el Back-End

Esta actualización de la base de datos se realiza porque, además de las configuraciones que acabamos de hacer, en el *application.properties* del proyecto le hemos indicado que se actualice la base de datos siempre que se levante la aplicación:

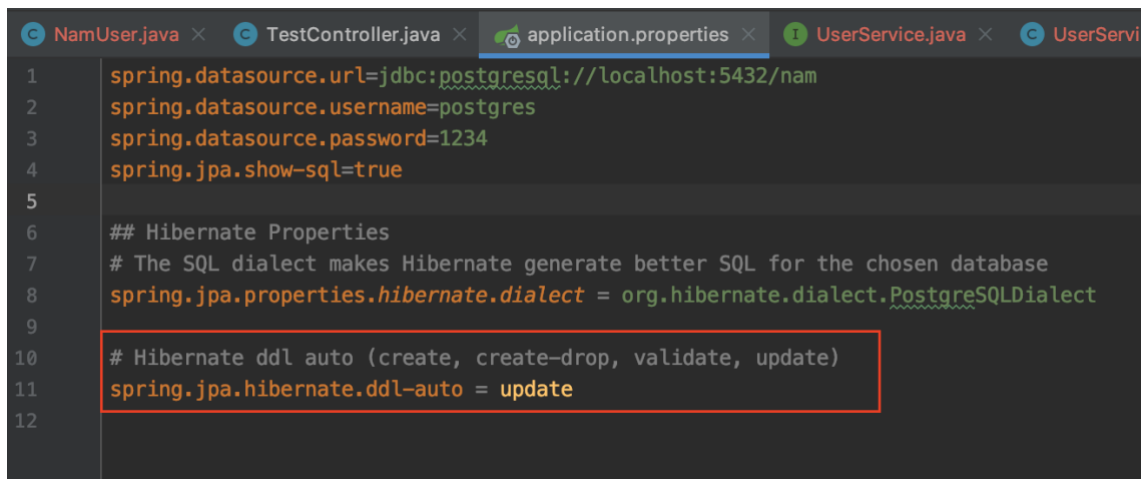
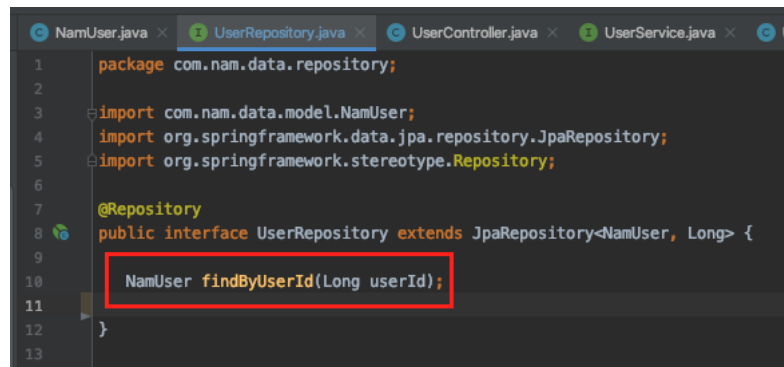


Imagen 23. Parámetros del archivo application.properties que permiten una actualización automática de la BBDD

Si vamos un paso más allá, podemos probar el flujo completo utilizando el cliente Postman. Insertaremos un registro en la tabla de usuarios que acabamos de crear y desde Postman, haremos una petición *Get* a un endpoint del back-end que nos devuelva la información de ese usuario:

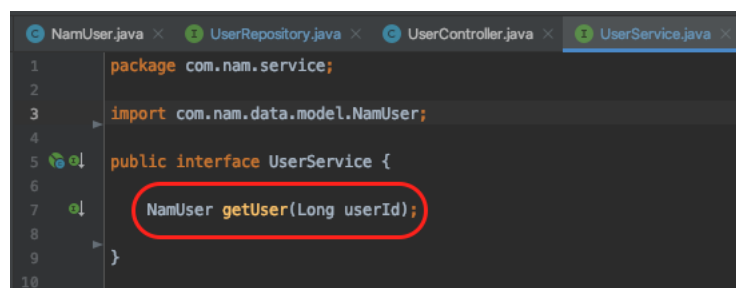
Para ello, en el repositorio de nuestra entidad, creamos un método que nos busque el usuario por su identificador:



```
1 package com.nam.data.repository;
2
3 import com.nam.data.model.NamUser;
4 import org.springframework.data.jpa.repository.JpaRepository;
5 import org.springframework.stereotype.Repository;
6
7 @Repository
8 public interface UserRepository extends JpaRepository<NamUser, Long> {
9
10     NamUser findById(Long userId);
11
12 }
13
```

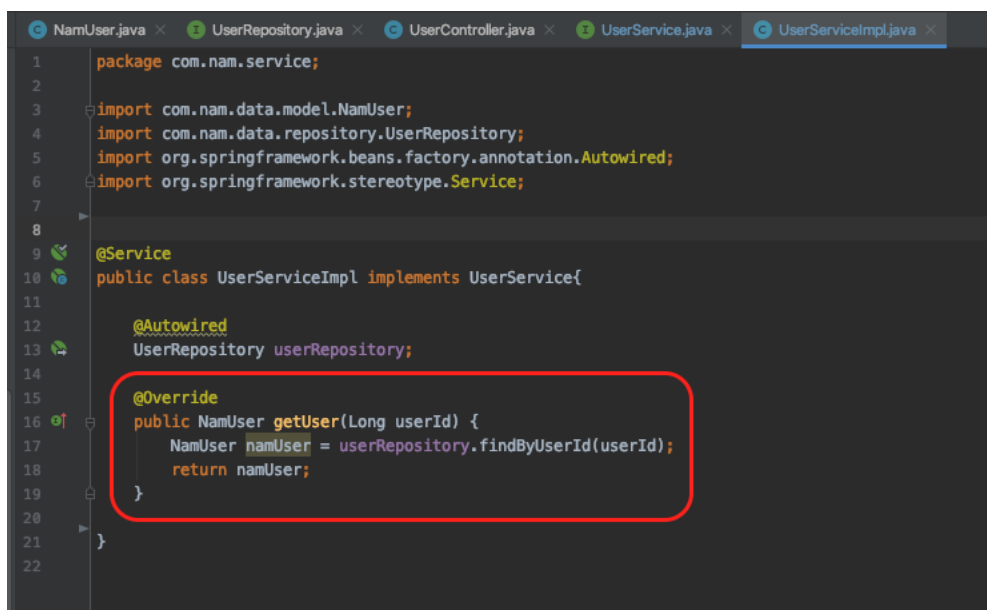
Imagen 24. Método del repositorio de la entidad NamUser

Llamaremos a este método desde un servicio. Para ello, nos creamos la interfaz y la implementación correspondiente que llame a ese método del repositorio (*findById*):



```
1 package com.nam.service;
2
3 import com.nam.data.model.NamUser;
4
5 public interface UserService {
6
7     NamUser getUser(Long userId);
8
9 }
10
```

Imagen 25. Interfaz UserService con el método getUser()

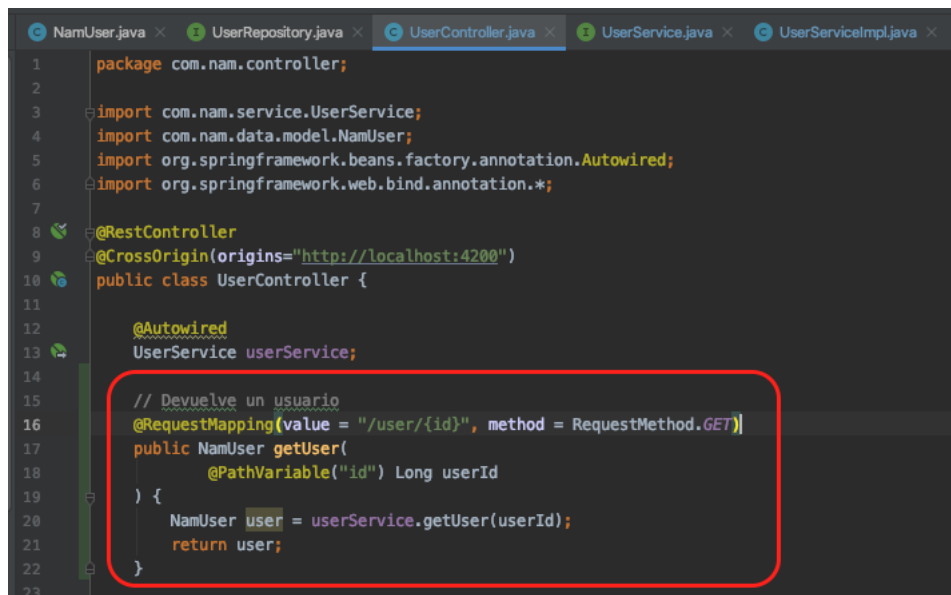


```
1 package com.nam.service;
2
3 import com.nam.data.model.NamUser;
4 import com.nam.data.repository.UserRepository;
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.stereotype.Service;
7
8
9 @Service
10 public class UserServiceImpl implements UserService{
11
12     @Autowired
13     UserRepository userRepository;
14
15     @Override
16     public NamUser getUser(Long userId) {
17         NamUser namUser = userRepository.findById(userId);
18         return namUser;
19     }
20
21 }
22
```

Imagen 26. Implementación de la interfaz UserService y del método getUser()

Como se observa en la imagen, hemos utilizado la anotación `@Service` para indicar que esa clase será un servicio y, además se ha utilizado `@Autowired` para inyectar el repositorio al que llamaremos para utilizar el método `findById`.

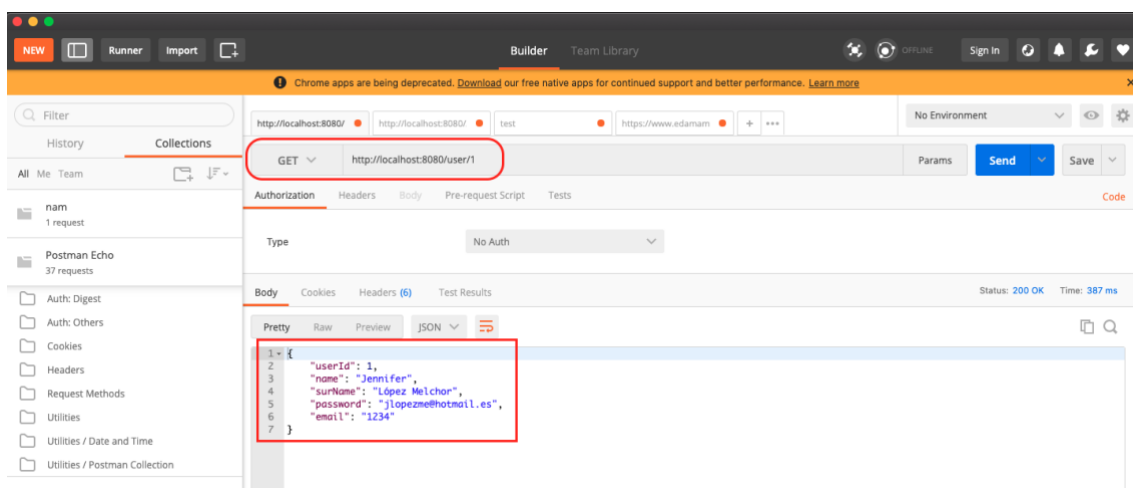
A su vez, este servicio será llamado desde un controlador, que contendrá el *endpoint* al que llamaremos desde Postman:



```
1 package com.nam.controller;
2
3 import com.nam.service.UserService;
4 import com.nam.data.model.NamUser;
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.web.bind.annotation.*;
7
8 @RestController
9 @CrossOrigin(origins="http://localhost:4200")
10 public class UserController {
11
12     @Autowired
13     UserService userService;
14
15     // Devuelve un usuario
16     @RequestMapping(value = "/user/{id}", method = RequestMethod.GET)
17     public NamUser getUser(
18         @PathVariable("id") Long userId
19     ) {
20         NamUser user = userService.getUser(userId);
21         return user;
22     }
23 }
```

Imagen 27. Controlador con un endpoint que devuelve un usuario a través del método `getUser()`

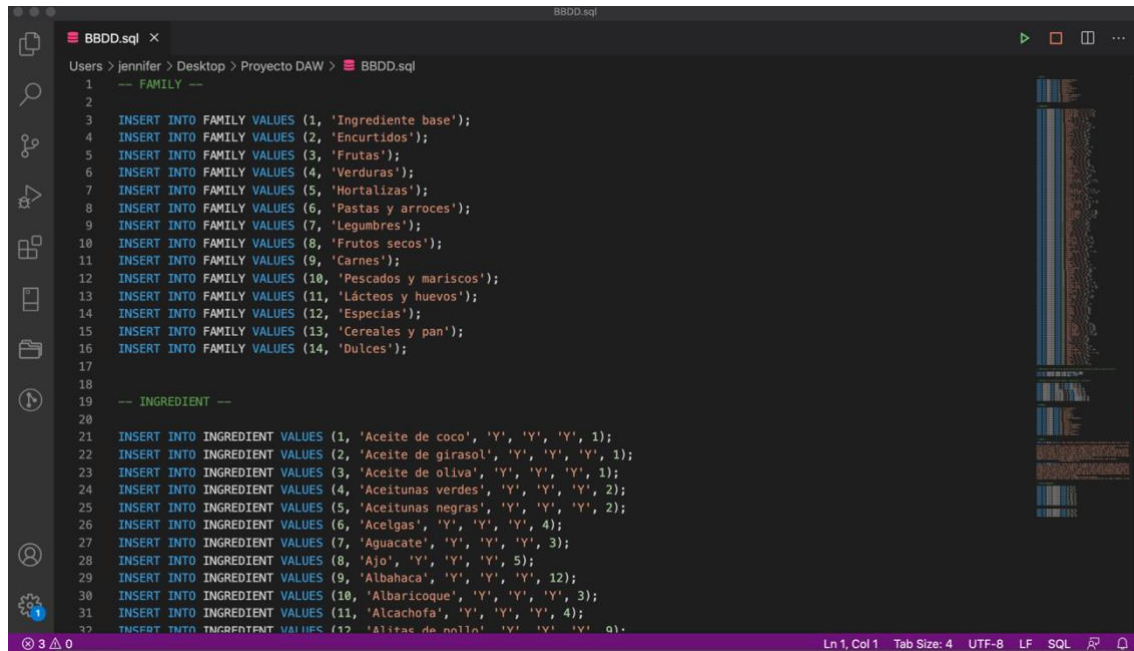
Ahora, si desde Postman hacemos una petición a ese endpoint y le pasamos el identificador de un usuario que tengamos en base de datos, nos devolverá la información correspondiente a dicho usuario en formato *.json*:



```
1 {
2   "userId": 1,
3   "name": "Jennifer",
4   "surname": "López Melchor",
5   "password": "jlopez@hotmail.es",
6   "email": "1234"
7 }
```

Imagen 28. Prueba de funcionamiento del endpoint desde Postman

Finalmente, es necesario destacar que para disponer de datos con los que probar nuestra aplicación, se elaboró y ejecutó un fichero (cuyo nombre es *BBDD.sql*, adjunto a este archivo) con “*inserts*” para las distintas tablas de nuestra base de datos.



```
1  -- FAMILY --
2
3  INSERT INTO FAMILY VALUES (1, 'Ingrediente base');
4  INSERT INTO FAMILY VALUES (2, 'Encurtidos');
5  INSERT INTO FAMILY VALUES (3, 'Frutas');
6  INSERT INTO FAMILY VALUES (4, 'Verduras');
7  INSERT INTO FAMILY VALUES (5, 'Hortalizas');
8  INSERT INTO FAMILY VALUES (6, 'Pastas y arroces');
9  INSERT INTO FAMILY VALUES (7, 'Legumbres');
10 INSERT INTO FAMILY VALUES (8, 'Frutos secos');
11 INSERT INTO FAMILY VALUES (9, 'Carnes');
12 INSERT INTO FAMILY VALUES (10, 'Pescados y mariscos');
13 INSERT INTO FAMILY VALUES (11, 'Lácteos y huevos');
14 INSERT INTO FAMILY VALUES (12, 'Especias');
15 INSERT INTO FAMILY VALUES (13, 'Cereales y pan');
16 INSERT INTO FAMILY VALUES (14, 'Dulces');
17
18
19 -- INGREDIENT --
20
21 INSERT INTO INGREDIENT VALUES (1, 'Aceite de coco', 'Y', 'Y', 'Y', 1);
22 INSERT INTO INGREDIENT VALUES (2, 'Aceite de girasol', 'Y', 'Y', 'Y', 1);
23 INSERT INTO INGREDIENT VALUES (3, 'Aceite de oliva', 'Y', 'Y', 'Y', 1);
24 INSERT INTO INGREDIENT VALUES (4, 'Aceitunas verdes', 'Y', 'Y', 'Y', 2);
25 INSERT INTO INGREDIENT VALUES (5, 'Aceitunas negras', 'Y', 'Y', 'Y', 2);
26 INSERT INTO INGREDIENT VALUES (6, 'Acelgas', 'Y', 'Y', 'Y', 4);
27 INSERT INTO INGREDIENT VALUES (7, 'Agua de coco', 'Y', 'Y', 'Y', 3);
28 INSERT INTO INGREDIENT VALUES (8, 'Ajo', 'Y', 'Y', 'Y', 5);
29 INSERT INTO INGREDIENT VALUES (9, 'Albahaca', 'Y', 'Y', 'Y', 12);
30 INSERT INTO INGREDIENT VALUES (10, 'Albaricoque', 'Y', 'Y', 'Y', 3);
31 INSERT INTO INGREDIENT VALUES (11, 'Alcachofa', 'Y', 'Y', 'Y', 4);
32 INSERT INTO INGREDIENT VALUES (12, 'Alitas de pollo', 'Y', 'Y', 'Y', 1);
```

Imagen 29. Captura de pantalla del fichero BBDD.sql

En este sentido, se valoró e investigó la posibilidad de utilizar una API de recetas, como la siguiente:

<https://developer.edamam.com/es/api-recetas-edamam>

En un principio, la idea era utilizar esta API u otra similar para abastecer a la aplicación con grandes cantidades de información, de modo que no fuera necesario realizar inserciones en base de datos sobre recetas o ingredientes, sino que se haría una petición a un *endpoint* de esta API que nos devolviese la información necesaria. Sin embargo, la opción disponible de manera gratuita tenía vigencia de un mes, por lo que descartamos la idea y por ello trabajamos con datos que se han ido insertando sobre nuestra propia base de datos.

5.3.4. Especificaciones del proyecto y casos de uso

En este apartado, mostraremos el resultado final de nuestra aplicación: *Ñam*.

- **Página principal:**

La primera página que nos aparece al iniciar la aplicación es la siguiente:

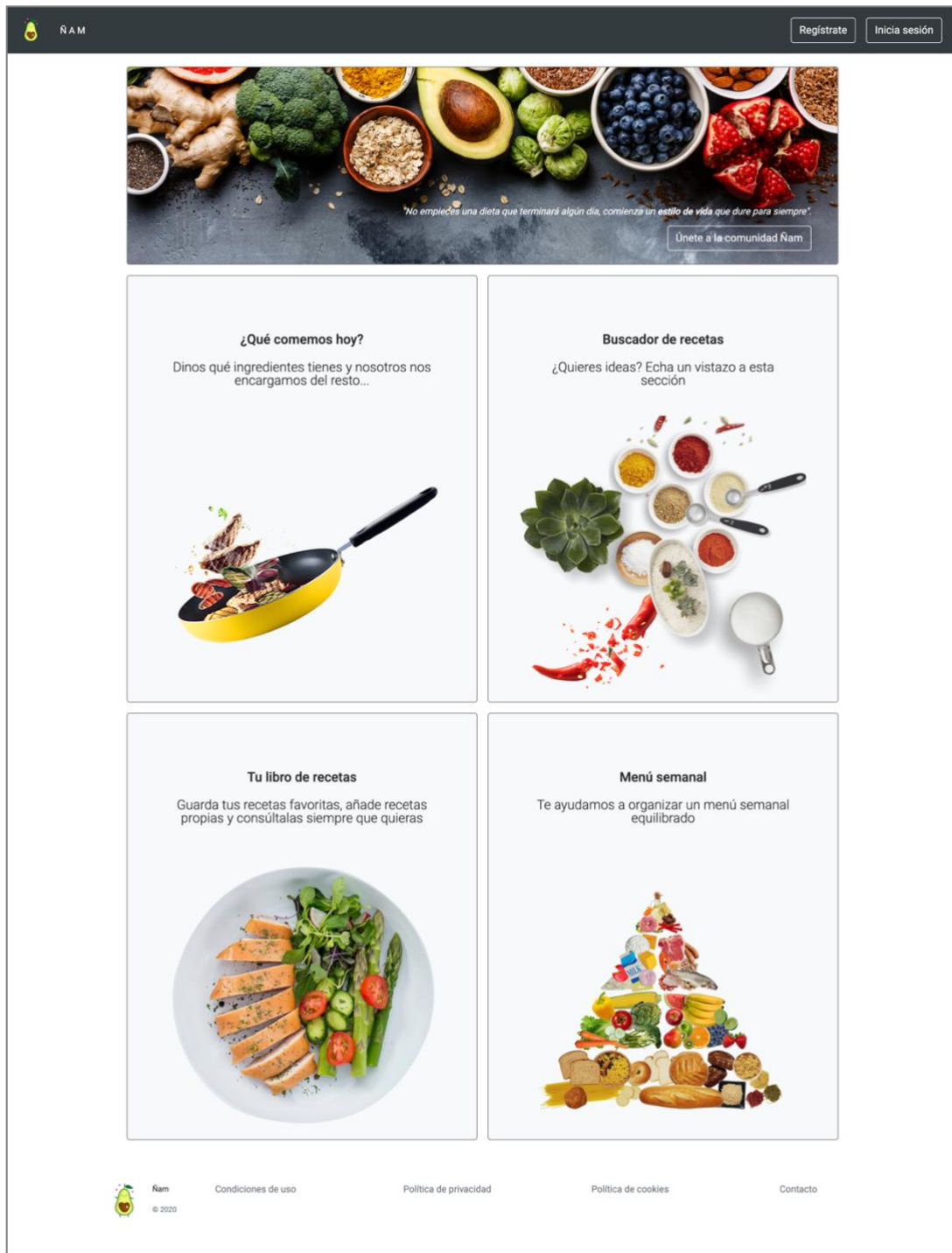


Imagen 30. Ñam - Página principal

Esta primera página es, por un lado, la puerta de acceso al usuario para registrarse o iniciar sesión y, por otro, una descripción global de la aplicación y de sus funcionalidades principales, de modo que con un simple vistazo, el usuario pueda ver en qué consiste *Ñam* y registrarse o iniciar sesión, si considera oportuno.

Es necesario destacar que, tal y como está planteada ahora mismo, el usuario sólo podrá acceder a las 4 funcionalidades principales de la aplicación si se registra y/o inicia sesión.

En la siguiente tabla, mostramos una síntesis de los casos de uso para la página principal de la aplicación:

Casos de uso de la página principal
Visualización de las funcionalidades principales de la aplicación.
Acceso al formulario de registro desde el botón de “Únete a la comunidad <i>Ñam</i> ”.
Acceso al formulario de registro desde el botón de “Regístrate”.
Acceso al formulario de inicio de sesión desde el botón de “Inicia sesión”.

Tabla 2. Casos de uso de la página principal

- **Registro e inicio de sesión:**

Si el usuario decide registrarse, accederá a la siguiente página:

Imagen 31. Página de registro de usuario

En la siguiente tabla, mostramos una síntesis de los casos de uso para la página de registro del usuario:

Casos de uso de la página de registro de usuario
Inserción de datos de registro en el formulario.
Envío de datos y posterior registro en la aplicación a través del botón de “Enviar”.
Retorno a la página principal, haciendo <i>click</i> sobre el logo de la aplicación o el nombre de la misma en el <i>navbar</i> .
Acceso al formulario de inicio de sesión desde el botón de “Inicia sesión”.

Tabla 3. Casos de uso de la página de registro de usuario

Una vez registrado, se redirige al usuario a la página de inicio de sesión, en la que ya podría introducir sus datos:

Imagen 32. Página de inicio de sesión

En este sentido, si el email o la contraseña son incorrectos, se mostrará un mensaje al usuario:

Imagen 33. Página de inicio de sesión con usuario y/o contraseña incorrectos

En caso contrario, se muestra un mensaje de bienvenida al usuario y se le da acceso a la página de recetas.

En la siguiente tabla, mostramos una síntesis de los casos de uso para la página de inicio de sesión del usuario:

Casos de uso de la página de inicio de sesión
Inserción de datos de inicio de sesión en el formulario.
Envío de datos y posterior inicio de sesión en la aplicación a través del botón de “Enviar”.
Visualización de mensaje de error en caso de introducir datos incorrectos.
Visualización de mensaje de bienvenida y acceso a la pantalla de recetas en caso de introducir datos correctos.
Acceso al formulario de registro de usuario desde el botón de “Regístrate”.
Acceso al formulario de registro de usuario desde la parte inferior del formulario, en “¿No tienes cuenta? Regístrate aquí”.
Retorno a la página principal de la aplicación, haciendo <i>click</i> sobre el logo de la aplicación o el nombre de la misma en el <i>navbar</i> .

Tabla 4. Casos de uso de la página de inicio de sesión

- **Página de recetas:**

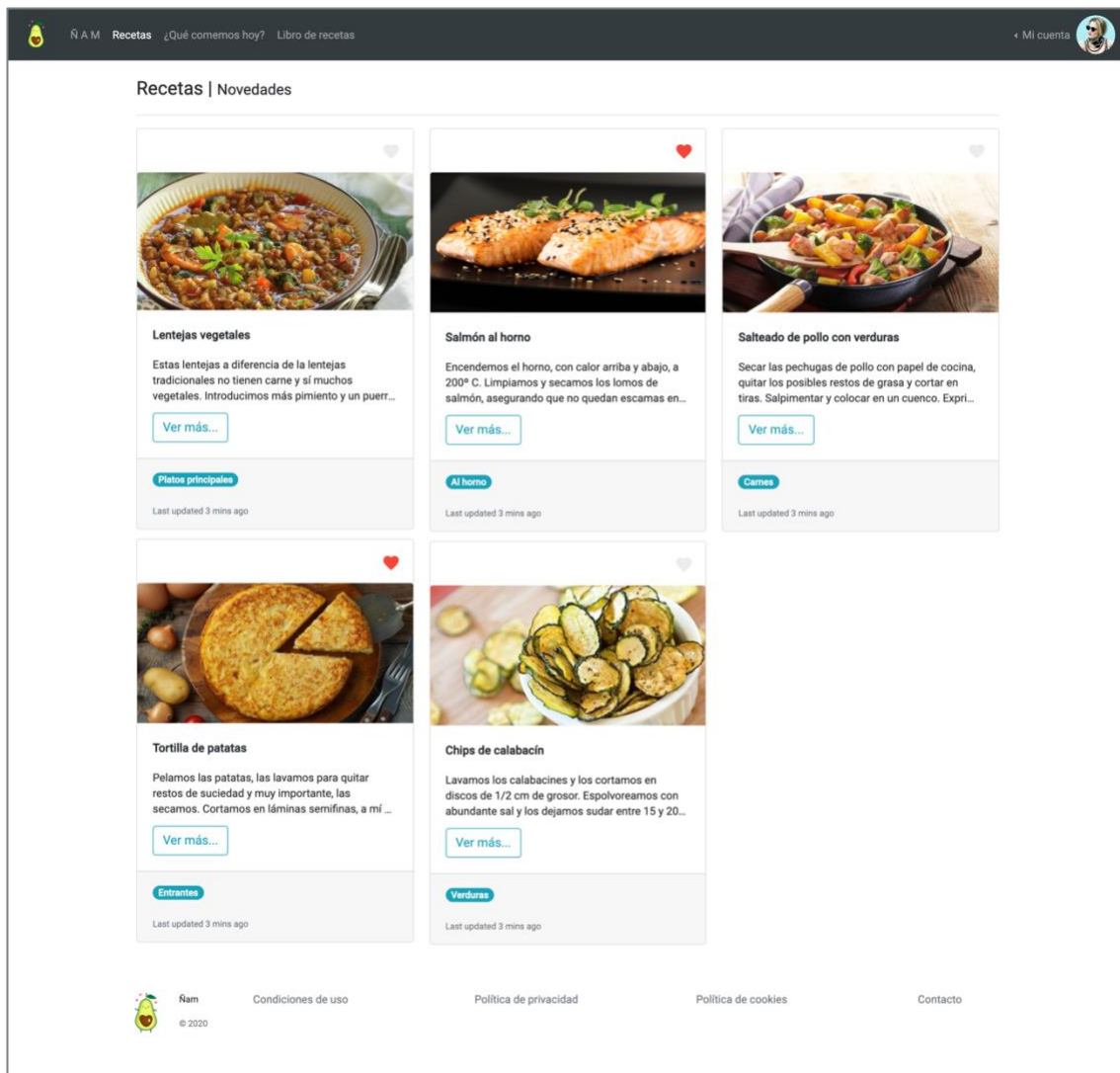


Imagen 34. Página de recetas

Como se observa en la imagen y, en la línea de lo que se comentó en la descripción de la página principal, observamos que una vez que el usuario inicia sesión, en el *navbar* de la aplicación ya aparecen las principales opciones de menú, que son las que darán acceso al usuario a las distintas funcionalidades de la aplicación:

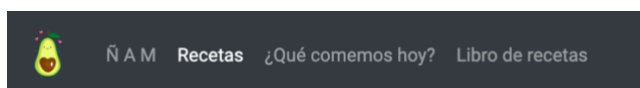


Imagen 35. Navbar con visibilidad de opciones de menú

Dentro de esta página de recetas, el usuario podrá echar un vistazo general a cada receta, la categoría a la que pertenece, añadirla a favoritos (lo que implica que la receta se guardará en su “libro de recetas”, que explicaremos posteriormente) y también podrá

acceder a una información más detallada de la receta para consultar sus ingredientes y descripción completa, si pulsa sobre el botón “*Ver más...*”



Imagen 36. Ejemplo de receta

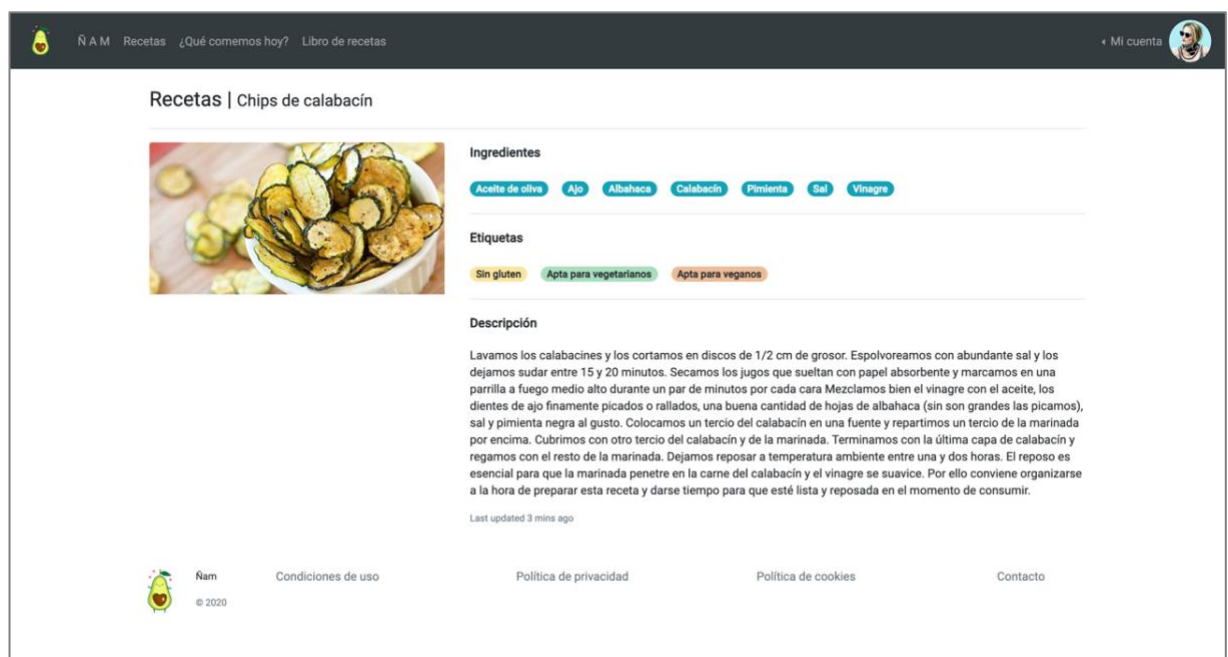


Imagen 37. Detalles de la receta al pulsar sobre el botón “*Ver más...*”

En esta página, además de la información sobre los ingredientes y descripción de la receta, se ha incluido un sistema de etiquetado, de modo que se indica al usuario si la receta está dentro de las categorías de “sin gluten”, “apta para vegetarianos” o “apta para veganos”, en función de los ingredientes de la misma.

En la siguiente tabla, mostramos una síntesis de los casos de uso para la página de recetas que acabamos de describir:

Casos de uso de la página de recetas
Visualización general de las recetas disponibles (foto, nombre, parte de la descripción y categoría a la que pertenece).
Añadir a favoritos una o varias recetas, haciendo <i>click</i> sobre el botón con forma de corazón de las recetas.
Acceso a información detallada sobre cada receta, haciendo <i>click</i> sobre el botón “Ver más...”
Visualización de los ingredientes de la receta.
Visualización de las etiquetas de la receta, si corresponden con las opciones de “Apta para vegetarianos”, “Apta para veganos” y/o “Sin gluten”.
Visualización de la descripción completa de la receta.
Retorno a la página principal de la aplicación, haciendo <i>click</i> sobre el logo de la aplicación o el nombre de la misma en el <i>navbar</i> .
Retorno a la página de recetas, haciendo <i>click</i> sobre la opción correspondiente en el <i>navbar</i> .
Acceso a la pantalla “¿Qué comemos hoy?”, desde la opción de menú del <i>navbar</i> .
Acceso a la pantalla “Libro de recetas”, desde la opción de menú del <i>navbar</i> .

Tabla 5. Casos de uso de la página de recetas

- **¿Qué comemos hoy?**

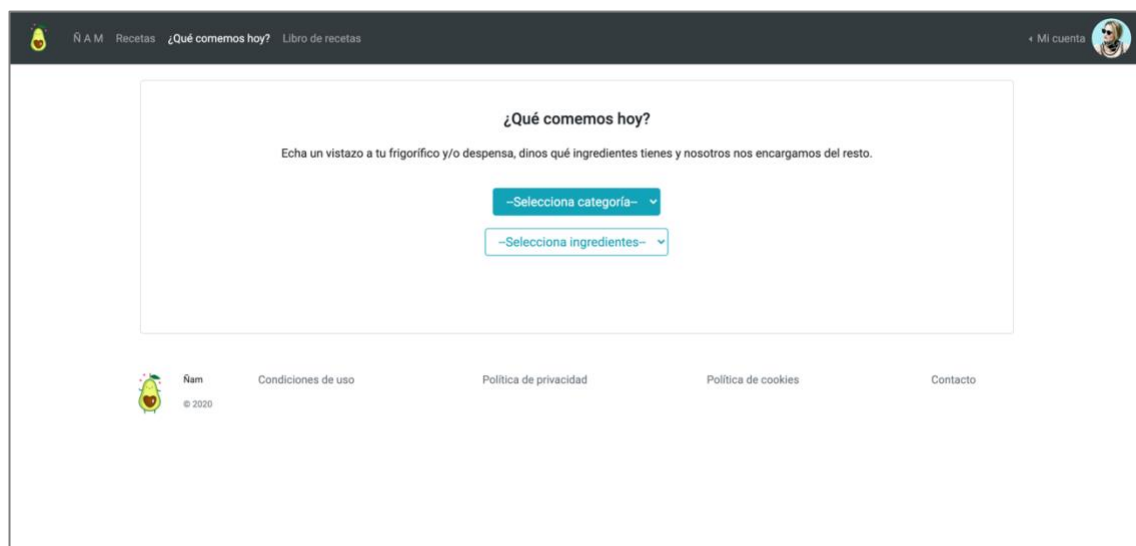


Imagen 38. Página “¿Qué comemos hoy?”

Otra de las funcionalidades de la aplicación es buscar, en función de los ingredientes que tenga disponibles, una serie de recetas a elaborar con dichos ingredientes. Para ello, puede filtrar todos los ingredientes por familias, de modo que sea más sencilla su búsqueda y pueda ir seleccionando fácilmente los ingredientes que correspondan.

Por tanto, el primer desplegable muestra las distintas familias de ingredientes (por ejemplo, fruta, lácteos, etc) y el segundo cargará, en función de la selección anterior, los ingredientes que correspondan a dicha familia:

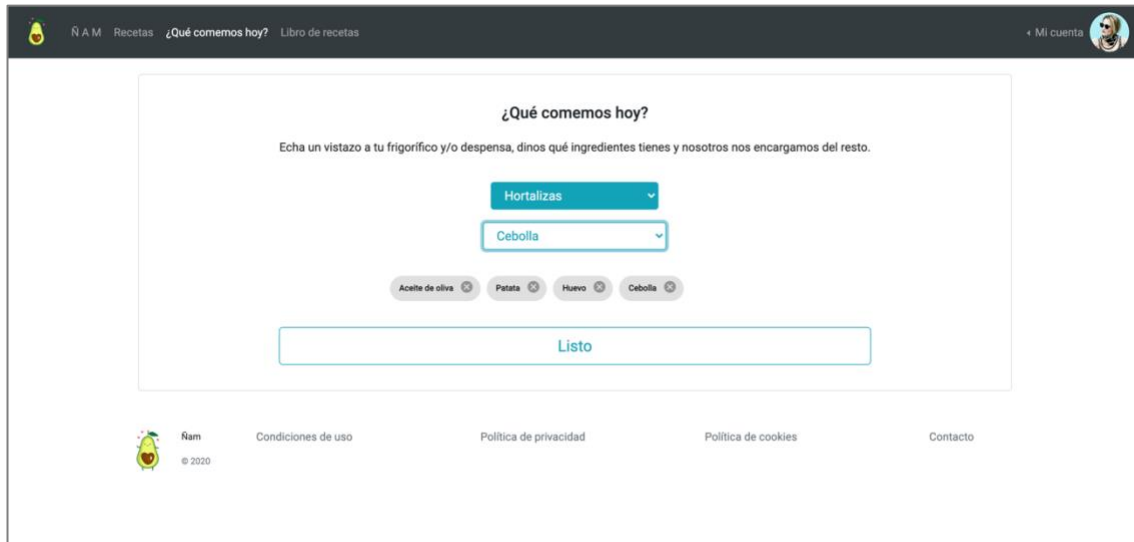


Imagen 39. Selección de ingredientes por categorías

Una vez que el usuario selecciona los ingredientes, se le mostrarán las recetas que pueda elaborar con dichos alimentos:

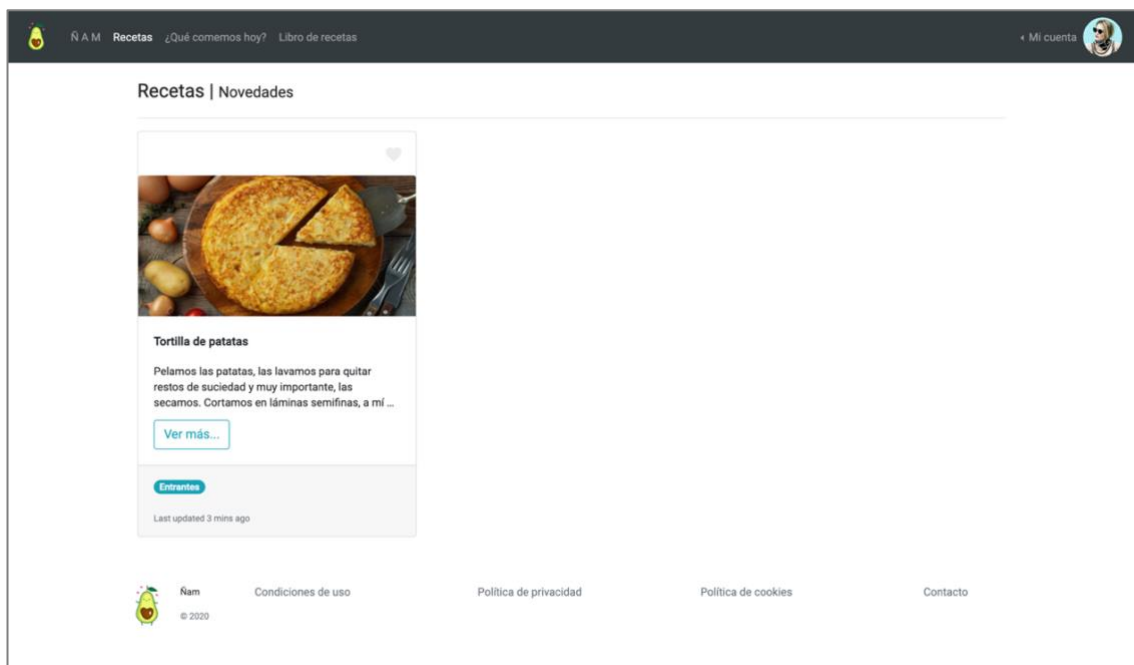


Imagen 40. Resultado de la búsqueda con los ingredientes seleccionados en la página anterior

Desde aquí, puede consultar la información detallada de cada receta, igual que si lo hiciese desde la página descrita en el apartado anterior y también podría añadir esta receta a favoritos, de modo que se incluya en su “libro de recetas” personal.

En la siguiente tabla, mostramos una síntesis de los casos de uso para la página que acabamos de describir:

Casos de uso de la página “¿Qué comemos hoy?”
Filtrar ingredientes por familias.
Seleccionar familia de ingredientes.
Seleccionar ingredientes.
Eliminar ingredientes de la selección pulsando sobre el icono de borrado del ingrediente seleccionado.
Hacer la búsqueda de recetas que coincidan con la lista de ingredientes, haciendo <i>click</i> sobre el botón “Listo”.
Visualización de la información general sobre las recetas que coinciden con la búsqueda (foto, nombre, descripción y categoría a la que pertenece).
Acceso a información detallada sobre cada receta, haciendo <i>click</i> sobre el botón “Ver más”.
Añadir a favoritos una o varias recetas, haciendo <i>click</i> en el botón con forma de corazón sobre las recetas.
Visualización de la descripción completa de la receta.
Retorno a la página principal de la aplicación, haciendo <i>click</i> sobre el logo de la aplicación o el nombre de la misma en el <i>navbar</i> .
Retorno a la página de recetas, haciendo <i>click</i> sobre la opción correspondiente en el <i>navbar</i> .
Acceso a la página “Libro de recetas”, desde la opción de menú del <i>navbar</i> .

Tabla 6. Casos de uso de la página “¿Qué comemos hoy?”

- **Libro de recetas:**

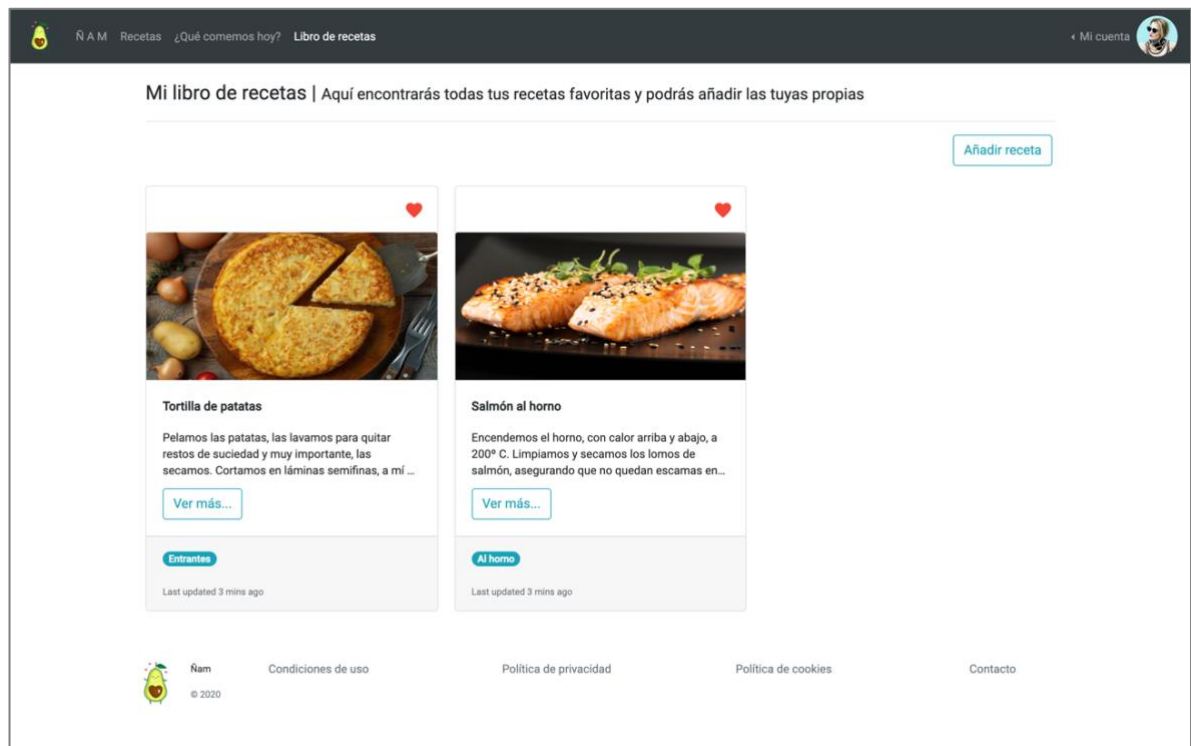


Imagen 41. Nam – Libro de recetas

En este apartado, el usuario podrá consultar todas las recetas que haya añadido como “favoritas”, así como otras recetas que haya añadido personalmente. Para ello, desde el botón de “Añadir receta”, tendrá acceso a lo siguiente:

1. Insertar nombre de la receta:

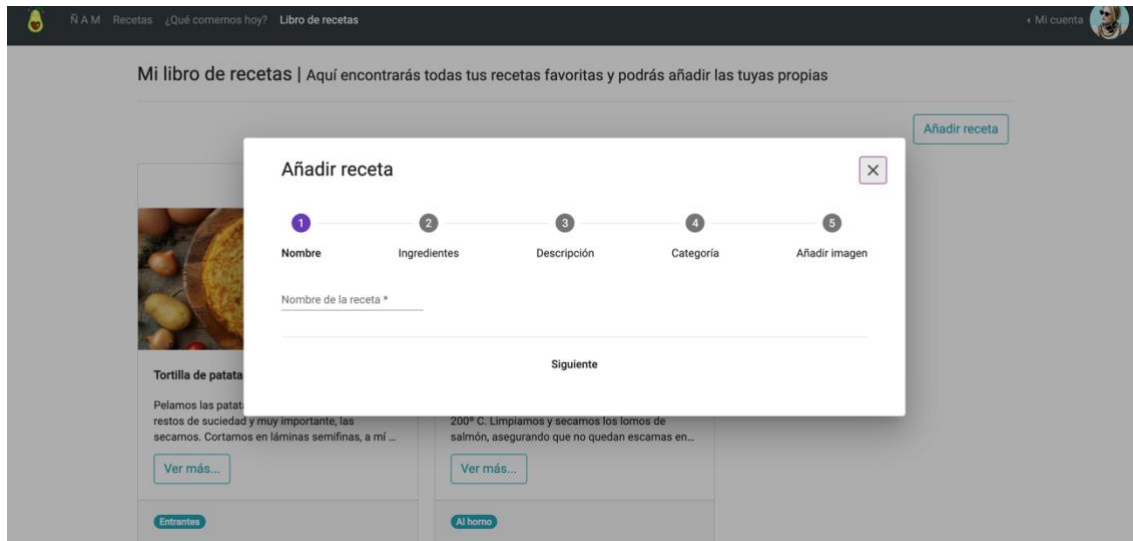


Imagen 42. Ventana modal para añadir receta – Paso 1: Añadir nombre

2. Seleccionar ingredientes:

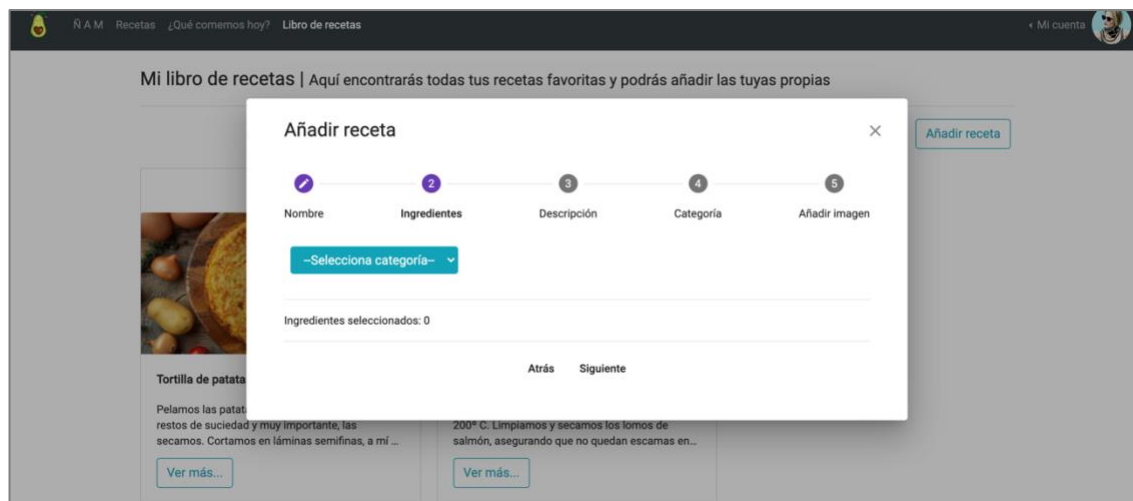


Imagen 43. Ventana modal para añadir receta – Paso 2: Añadir ingredientes

Para mejorar la experiencia de usuario, también puede filtrar por categorías e ir seleccionando los ingredientes correspondientes:

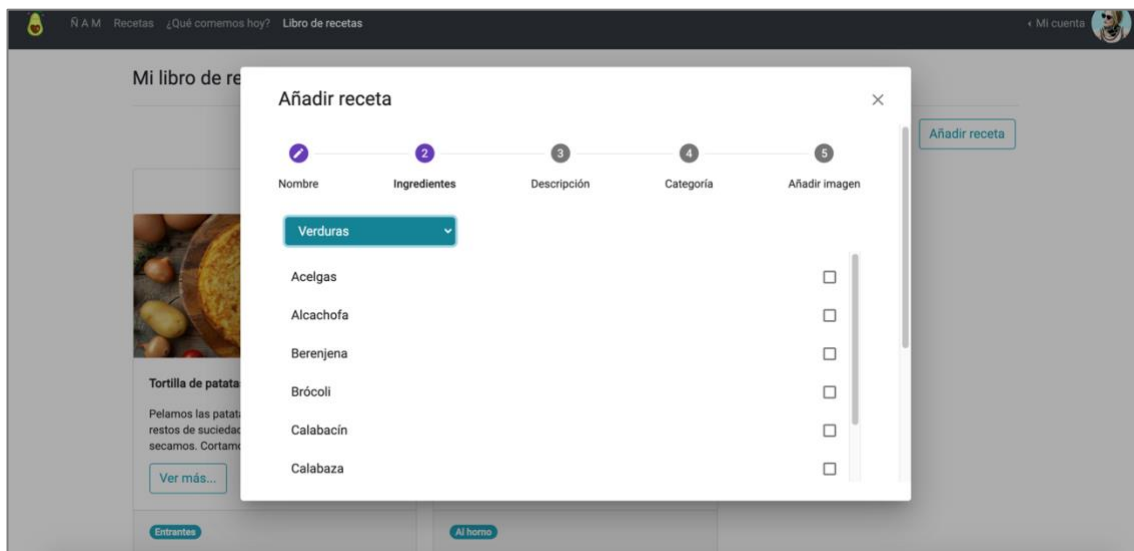


Imagen 44. Ventana modal para añadir receta – Paso 2: Filtrado de ingredientes por categorías

3. Indicar la descripción de la receta:

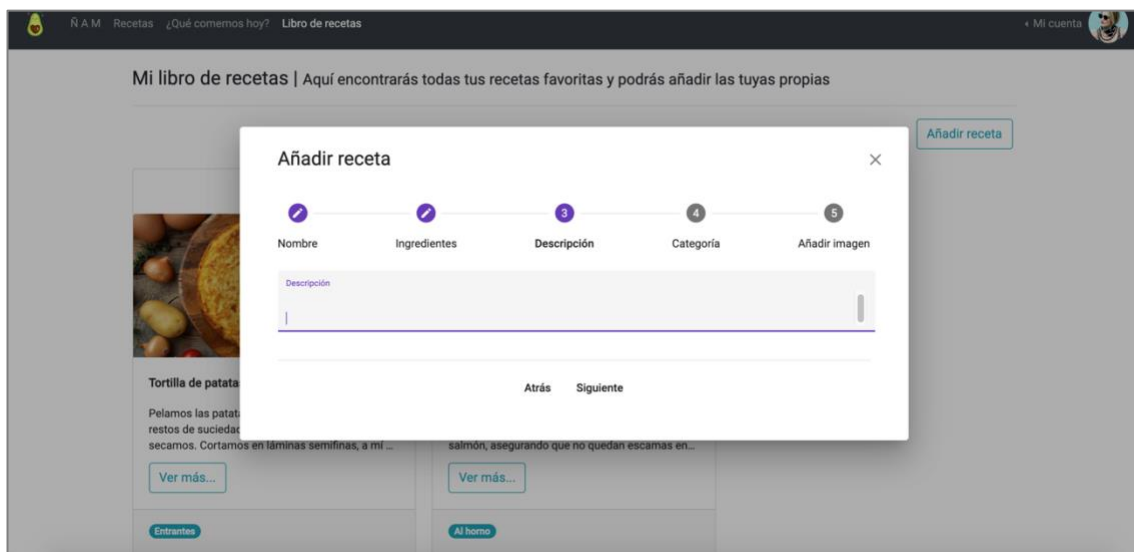


Imagen 45. Ventana modal para añadir receta – Paso 3: Añadir descripción

4. Seleccionar la categoría a la que pertenece:

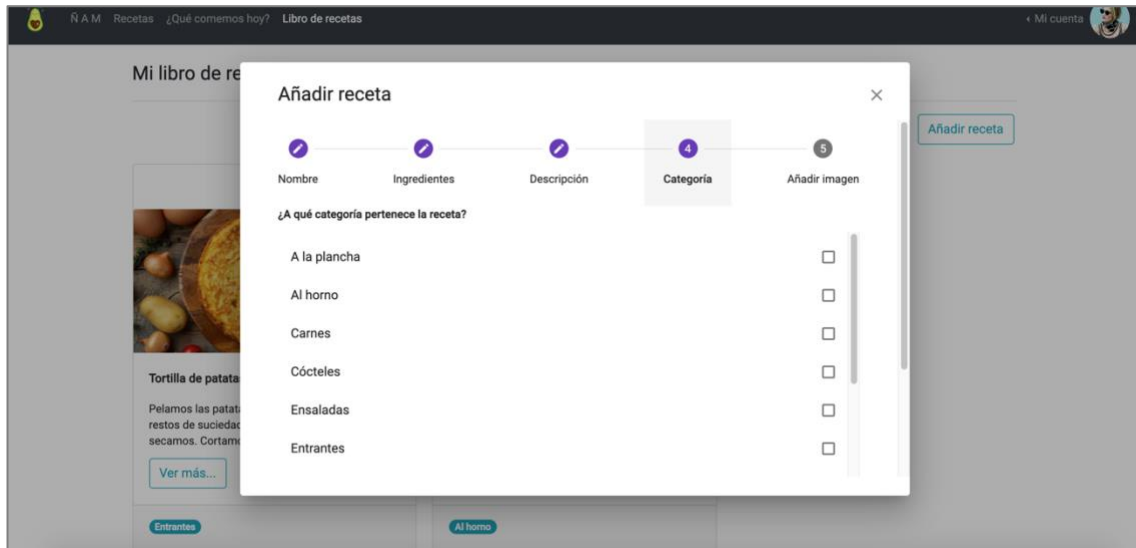


Imagen 46. Ventana modal para añadir receta – Paso 4: Añadir categoría

5. Adjuntar una foto de la receta:

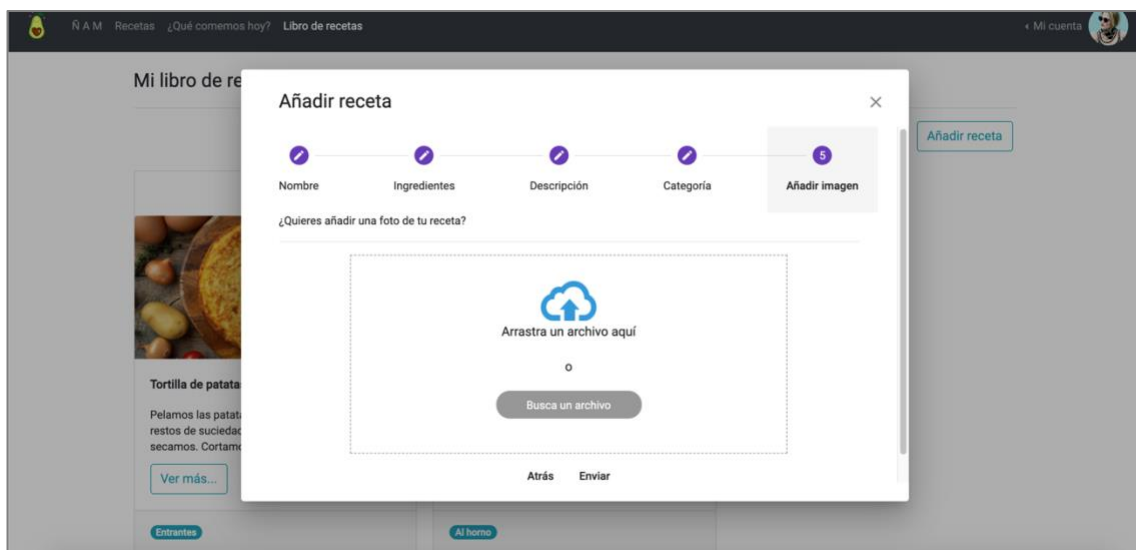


Imagen 47. Ventana modal para añadir receta – Paso 5: Añadir imagen

Una vez que el usuario ha rellenado todos los datos y da al botón de enviar, se guarda la receta para ese usuario y ya aparecerá en su “libro de recetas”, junto con sus recetas favoritas.

En la siguiente tabla, mostramos una síntesis de los casos de uso para la página que acabamos de describir:

Casos de uso de la página “Libro de recetas”
Visualizar recetas añadidas como favoritas por el usuario.
Visualizar recetas propias añadidas por el usuario.
Quitar de favoritos una o varias recetas, haciendo <i>click</i> sobre el botón con forma de corazón.
Visualización de la información general sobre las recetas (foto, nombre, descripción y categoría a la que pertenece).
Acceso a información detallada sobre cada receta, haciendo <i>click</i> sobre el botón “Ver más”.
Visualización de la descripción completa de la receta.
Añadir una receta.
Añadir nombre de la receta.
Seleccionar ingredientes de la receta y filtrarlos por familias.
Añadir la descripción de la receta.
Seleccionar la categoría a la que pertenece la receta.
Añadir una imagen sobre la receta.
Retorno a la página principal de la aplicación, haciendo <i>click</i> sobre el logo de la aplicación o el nombre de la misma en el <i>navbar</i> .
Acceso a la página de recetas, haciendo <i>click</i> sobre la opción correspondiente en el <i>navbar</i> .
Acceso a la pantalla “¿Qué comemos hoy?”, desde la opción de menú del <i>navbar</i> .

Tabla 7. Casos de uso de la página “Libro de recetas”

6. CONCLUSIONES Y MEJORAS DEL PROYECTO

Una vez finalizado el tiempo de desarrollo del proyecto y unos días previos a la entrega del mismo, tengo algunas conclusiones y propuestas de mejora que me gustaría dejar plasmados en este apartado.

Como bien indicaba en la introducción de este documento, este proyecto surgió de motivaciones personales, partiendo del interés tanto por la temática de la alimentación, como por las tecnologías y herramientas que durante todo este proceso he aprendido a utilizar.

A día de hoy, considero que la aplicación no está desarrollada en su totalidad tal y como me hubiese gustado. Por falta de tiempo, ha quedado en el tintero el desarrollo de algunas funcionalidades. Igualmente, considero que me queda mucho por aprender y descubrir de las tecnologías usadas y de la programación web en general. Es por ello que, más allá de la entrega y posterior defensa de este trabajo, pretendo continuar con su desarrollo, para dejar cerrado el proyecto que desde un principio tenía en mente, con todas sus funcionalidades y, por supuesto, para seguir aprendiendo y creciendo técnicamente.

En síntesis, los puntos clave que me gustaría dejar plasmados como propuestas de mejora y futuros desarrollos para este proyecto, son los siguientes:

- Menú semanal

En función de los datos del usuario (género, altura, peso, frecuencia de actividad física, etc), se le propondrá una selección de recetas organizadas por días de la semana, de modo que para cada día, disponga de una combinación de menús que le ayuden a alimentarse de manera equilibrada (por ejemplo, que se incluya fruta y verdura a diario, legumbres 3 días a la semana, cierta cantidad de proteínas en función de su peso y actividad física, etc). Este menú “base” que proporciona la aplicación, podrá ser modificado y personalizado por el usuario.

Para ello, sería necesario, por un lado, una pantalla donde el usuario pueda introducir sus datos personales. Se podría utilizar la opción de menú “*Datos personales*” ya existente en el *navbar* de la aplicación:

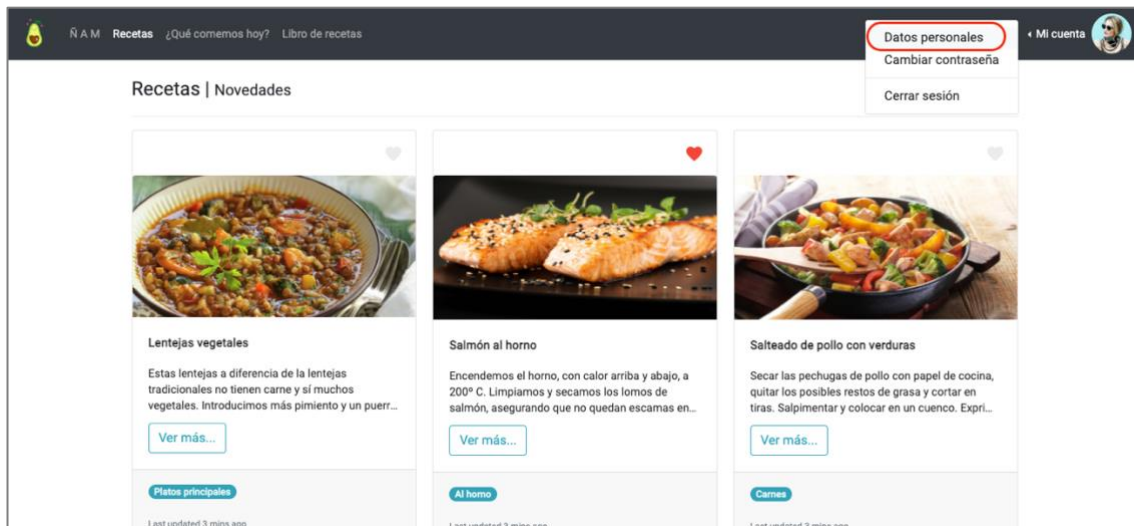


Imagen 48. Navbar – Menú de configuración de datos del usuario

También será necesario elaborar una pantalla donde el usuario pueda consultar y modificar estos menús proporcionados por el sistema.

Y a grandes rasgos, será necesario implementar la lógica que permita combinar, en función de las categorías de las recetas y de las familias de los ingredientes, una selección de recetas y menús que puedan considerarse equilibrados. Considero que el modelo de datos propuesto actualmente permite hacer esta escalabilidad y adaptarse a este nuevo desarrollo.

- Consulta de histórico de menús

Se ubicará en otro módulo distinto, pudiendo consultar los menús que se han elaborado anteriormente.

- Cambio de contraseña

En relación con la última imagen mostrada, también queda pendiente el desarrollo y la implementación del cambio de contraseña del usuario desde la opción de menú que ya existe en el *navbar* de la aplicación.

- *Testing* del código

Me gustaría incluir pruebas unitarias tanto para el Back-End como para el Front-End de la aplicación.

Si bien es cierto que para cada componente de Angular ya se generan archivos de *tests* (los que tienen extensión *.spec.ts*) que ya incluyen alguna prueba sobre el propio componente, sería de gran utilidad aprender a utilizar las tecnologías de Karma y Jasmine que incluye el propio *framework* y permiten la realización de pruebas unitarias sobre el código.

Así mismo, en cuanto al Back-End, me gustaría utilizar las tecnologías de Mockito y JUnit para la realización de pruebas unitarias sobre el código.

- Adaptación de *Ñam* como aplicación móvil

Una vez implementados todos los apartados anteriores y habiendo dejado cerradas las funcionalidades principales del proyecto, me gustaría comenzar una nueva andadura por el universo de las aplicaciones móviles y empezar a aprender de aquellas tecnologías que me permitan adaptar este proyecto y desarrollarlo como aplicación móvil.

Para finalizar, me gustaría destacar mi satisfacción con el trabajo realizado durante la elaboración de este proyecto, pues considero que he podido abarcar todos los objetivos planteados y poner en práctica los conocimientos adquiridos durante estos dos años de formación en el Instituto Tecnológico de Telefónica.

Sin duda, para mi ha sido una gran experiencia con la que culminar esta etapa de formación en el grado superior de Desarrollo de Aplicaciones Web. Nunca imaginé que lo que en su día comenzó con la simple curiosidad por el desarrollo web, dedicándome a otro sector completamente distinto a este, me llevaría hasta el momento actual, dando un giro de 360° a mi trayectoria profesional.

7. BIBLIOGRAFÍA

<https://start.spring.io/>

<https://material.angular.io/>

<https://www.udemy.com/course/angular-2-fernando-herrera/>

<https://getbootstrap.com/>

<https://openwebinars.net/blog/que-es-rest-conoce-su-potencia/>

<https://developer.edamam.com/food-database-api-docs>

<https://www.javaguides.net/2019/08/spring-boot-spring-data-jpa-postgresql-example.html>

<https://www.baeldung.com/spring-requestmapping>