



TRABAJO FIN DE GRADO

IdealMenú

Plataforma web para la gestión y creación de una dieta personalizada

Convocatoria de Junio – Curso 2021 / 2022

Ana Roderop Paredes
48844755T

ana.roderop@um.es

Tutora: Begoña Moros Valle

ÍNDICE DE CONTENIDOS

I.	Resumen	6
II.	Extended abstract.....	8
1	Introducción.....	13
2	Estado del arte	15
2.1	Posibles mejoras a herramientas ya existentes	16
2.2	Tecnologías utilizadas	16
2.2.1	MEAN Stack	17
2.2.2	Front-end.....	18
2.2.3	Back-end	19
2.2.4	Base de datos	20
2.2.5	Herramientas usadas para el proyecto	21
3	Análisis de objetivos y metodología.....	22
3.1	Objetivos	22
3.2	Metodología	23
4	Diseño y resolución del trabajo realizado	25
4.1	Casos de uso	25
4.1.1	Autenticación.....	25
4.1.2	Recetas	26
4.1.3	Alimentos	26
4.1.4	Pirámide alimenticia	27
4.1.5	Dieta.....	27
4.2	Diseño de la aplicación.....	28
4.2.1	Base de datos	28
4.2.2	Back-end	31
4.2.2.1	Routes	32
4.2.2.2	Models.....	34
4.2.2.3	Controllers.....	34

4.2.2.3.1	Alimento	34
4.2.2.3.2	Pirámide	35
4.2.2.3.3	Receta	36
4.2.2.3.4	Receta Simple.....	37
4.2.2.3.5	Usuario.....	37
4.2.2.3.6	Plan	39
4.2.3	Front-end.....	47
4.2.3.1	Componentes	47
4.2.3.1.1	Página principal.....	47
4.2.3.1.2	Página de registro	48
4.2.3.1.3	Página de inicio de sesión	51
4.2.3.1.4	Página de inicio	52
4.2.3.1.5	Base de datos de recetas.....	53
4.2.3.1.6	Añadir receta simple.....	54
4.2.3.1.7	Añadir receta	56
4.2.3.1.8	Información sobre una receta	57
4.2.3.1.9	Base de datos de alimentos	58
4.2.3.1.10	Información pirámides.....	59
4.2.3.1.11	Plan semanal.....	60
4.2.3.1.12	Plan diario	61
4.2.3.1.13	Editar perfil	61
4.2.3.2	Modelos.....	62
4.2.3.3	Servicios.....	63
4.2.3.4	Guards	63
5	Conclusiones y vías futuras.....	64
5.1	Conclusión	64
5.2	Vías futuras	64
6	Bibliografía.....	66

ÍNDICE DE FIGURAS

Figura 1. MEAN stack	17
Figura 2. Arquitectura MEAN stack	17
Figura 3. Casos de uso para autenticación.	25
Figura 4. Casos de uso para las recetas	26
Figura 5. Casos de uso para los alimentos	27
Figura 6. Casos de uso para las piramides.	27
Figura 7. Casos de uso para la dieta.....	28
Figura 8. Arquitectura de la pila MEAN.....	28
Figura 9. Diseño de la base de datos.....	29
Figura 10. Pirámide alimenticia recomendada por la SENC.....	40
Figura 11. Página principal	47
Figura 12. Página de registro.....	48
Figura 13. Registro con datos introducidos.....	49
Figura 14. Registro contraseñas no coinciden.....	49
Figura 15. Registro email ya registrado	50
Figura 16. Registro exitoso	51
Figura 17. Página de inicio de sesión.....	51
Figura 18. Email no tiene cuenta	52
Figura 19. Contraseña incorrecta.....	52
Figura 20. Página de inicio.....	53
Figura 21. Base de datos de recetas simples y completas	53
Figura 22. Despliegue receta simple.....	54
Figura 23. Añadir receta.....	55
Figura 24. Alimento por añadir y añadido	55
Figura 25. Añadir receta.....	56
Figura 26. Ingrediente añadido y por añadir.	57
Figura 27. Información receta	58
Figura 28. Base de datos de alimentos	58
Figura 29. Añadir alimento	59
Figura 30. Pirámide regular.....	60
Figura 31. Plan semanal	60
Figura 32. Editar perfil.....	61
Figura 33. Cambio de contraseña	62

ÍNDICE DE TABLAS

Tabla 1. Frameworks más usados para el front-end (Tápanes, 2022)	18
Tabla 2. Tiempos estimados de la relación del proyecto	23
Tabla 3. Rutas para Usuario	32
Tabla 4. Rutas para Piramide.....	32
Tabla 5. Rutas para Alimento.....	32
Tabla 6. Rutas para Receta.....	33
Tabla 7. Rutas para Receta Simple	33
Tabla 8. Rutas para Plan.....	33

I. Resumen

Hoy en día la preocupación por nuestra salud es muy elevada. Conforme pasan los años, tenemos más y más conocimiento sobre todo lo que tiene que ver con nuestra salud. Durante los últimos años se ha ido demostrando cómo mantener una vida saludable puede ayudar en gran medida a numerosas enfermedades. Además, según la Organización Mundial de la Salud (OMS) muchos de los riesgos para el desarrollo de enfermedades crónicas están ligados con la alimentación y el ejercicio físico (OMS, 2003).

Esto nos lleva a deducir que la alimentación no saludable y el sedentarismo puede llegar a ser responsable de gran cantidad de fallecimientos debido a enfermedades crónicas. Por todo ello, la sanidad mundial está poniendo bastante empeño en ayudar y disminuir todos estos problemas. Se intenta concienciar a la población de todos los peligros que podría llevar no mantener una vida saludable, como también de intentar mostrar de qué formas podríamos ayudar a nuestra salud, simplemente tenemos que mantener una dieta equilibrada y saludable junto a la realización de ejercicio físico.

Debido a todo esto, es muy importante que la población tenga el máximo conocimiento sobre todos los factores que afectan a la salud, por ello existen cada vez más plataformas y tecnologías que ayudan tanto a profesionales como a gente de la calle a lidiar con las dificultades que pueda llevar saber cómo mantener al día en todos estos ámbitos.

En consecuencia, poder tener una aplicación que nos facilite los diferentes conocimientos necesarios para poder llevar una dieta equilibrada y sana, y que también a su vez nos ayude a organizar dichas dietas resultaría de gran ayuda para conseguir mantener una vida saludable.

Por ello, el objetivo de este TFG es el desarrollo de una plataforma para la creación de planes semanales de alimentación basándose en la pirámide alimenticia dirigido a personas no expertas en la materia. Para favorecer la accesibilidad, esta plataforma estará disponible a través de una página web, sin necesidad de descargar ninguna aplicación y para poder verla desde cualquier sitio.

Para poder conseguir dicho objetivo de este TFG, el proyecto ha sido dividido en 4 partes principales. Primeramente, cuenta con una parte front-end en la cual se organiza toda la vista de la aplicación, es decir la interfaz. También cuenta con la base de datos en la cual se almacenan los diferentes elementos que han sido utilizados. Finalmente cuenta con una

parte back-end en la cual se encuentra toda la lógica de la aplicación, esta lógica contiene la última parte principal, el algoritmo que crea el plan semanal de dieta.

Estos componentes han sido desarrollados mediante diferentes tecnologías, que se agrupan en lo que se llama una pila MEAN (MongoDB, Express, Angular, NodeJS). Primeramente, Mongo DB ha sido utilizado para la base de datos, es una base de datos no relacional en la cual se pueden crear diferentes colecciones que contienen documentos. Express y NodeJS ha sido utilizado para el back-end, estas tecnologías utilizan JavaScript y permiten el uso de una API REST para mantener una comunicación con la interfaz. Y por último Angular, esta tecnología permite crear a través de librerías y componentes una interfaz. Esta tecnología tiene grandes facilidades para la creación de una página web ya que tiene una cantidad importante de librerías y componentes que permite trabajar con ella de forma mucho más fácil. Esta interfaz se comunicará con el back-end a través de peticiones API REST.

II. Extended abstract

Today the concern for our health is very elevated. As the years pass by, we acquire more and more knowledge about everything that has to do with our health. In the last few years, it has been shown how maintaining a healthy way of life can greatly help in numerous diseases. In addition, according to the World Health Organization (WHO) almost 50% of the factors for the development of chronic diseases are linked to diet and physical exercise.

This leads us to deduce that unhealthy eating and sedentary lifestyle can be the blame for a large number of deaths due to these chronic diseases mentioned before. Due to all of this, world health is putting a lot of effort into helping to reduce all these problems. One of the most important things is to make the population be aware of all the dangers and problems that one could have if not maintaining a healthy life. Furthermore, trying to show in what possible ways we could help our health, we should simply have to maintain a balanced and healthy diet connected with physical exercise.

Due to all this, it is very important that people have the maximum knowledge about all the factors that affect health, how to maintain a healthy lifestyle, acknowledging your own body and what it can handle, in terms of exercising, but also in terms of having a healthy diet combined with it. Which is why, nowadays, there are more and more platforms and technologies that help both professionals and non-professional on the subject of how to deal with the difficulties that can take knowing how to keep you up to date in all these areas.

Taking all this into account, to be able to have an application that provides us with the different knowledge that is necessary to be able to have a balanced and healthy diet, as well as helping us organize these diets, is a great addition to one's way of living, being able to not have to worry about this matter, makes everything easier. The goal is not only to have your plan made out for you, but also being able to organize the different recipes and food that is used to create this plan. You will be able to add as many recipes and foods as you would like, so the app can have more variability to take for the creation of the diet.

Therefore, the objective of this project is that any user could have this platform with which they could organize their weekly diet. The platform creates a weekly meal plan for you based on the recommended food pyramid. This is made with different recipes stored in our database. At the moment, the app only makes plans for the regular pyramid, it does

not take into consideration the vegetarian and vegan pyramids, although, at the moment, the user will have the chance to observe these pyramids for educational purposes.

This platform has been made on a web page without the need to download any application and be able to see it from anywhere. I think that the web gives more freedom of use without having to consider if the app is compatible with the different devices.

In order to achieve this objective of this TFG, the project has been divided into 4 main parts, the front-end, the back-end, the database and the algorithm that creates the diet.

Firstly, the front-end part which includes the entire view of our application, that is, our interface, what the user will see and interact with. It also has the database in which the different elements necessary for our application are stored, all the recipes, foods, user, pyramids, and weekly plans. Finally, it has a back-end part where the application logic is located, and this logic contains the last main part of our application, the algorithm that creates the weekly diet plan.

These components have been developed using different technologies, which are grouped into what is called a MEAN stack (MongoDB, Express, Angular, NodeJS). First of all, we have MongoDB, this technology is a database system, a non-relational system. In this, we can store the different elements in what are called collections instead of tables, and in these collections, you have what is named, documents, the different objects stored in the collection. This database system is more flexible than a relational one, due to the fact that you do not have to follow a strict template of the documents, that is, each document does not have to have the same structure as the rest, this allow you to be more flexible and lets you stored exactly what you need, without the necessity of having to fill elements that you wouldn't want to, this means that you could have a missing property on a object for example, a property that a single element does not need, so, with this, you would not have to waste space in the database if it is not necessary. This database also lets you easily make JSON based queries to find different documents in the multiple collections.

Secondly, we have Express and NodeJS, these technologies were used to make the back-end part of the application. Express is a server-side Javascript framework. Due to its simplicity, it is the most used web application framework for Node. Therefore, if you already know JavaScript, then it will be really easy for you to program an application using Express. Express provides a simple routing for requests made by clients. It also provides a middleware that is responsible for making decisions to give the correct responses for the requests made by the client.

On the other hand, NodeJS is a Javascript runtime that powers the server side of the application. NodeJS also provides a rich library of various JavaScript modules which simplifies the development of web applications using NodeJS to a great extent, one of these libraries is Express.

Together, Express and NodeJS, make a great combo, they are very easy to use and to connect with the other parts of the application. It also allows us to use an API Rest to communicate with the client-side of the application with requests.

The back-end of the app is structured in different parts: models, controllers and routes. The models will contain the different objects created in the app, the one that will be stored in the database. These will be, what are called for Mongo, the schemas, another way of saying the structure of the object saved in the database. The routes will set up the different request routes that could be sent by the client and redirected to a controller function. These will contain all the different possible routes that the user can trigger from every action possible in the front-side of the application. Last but not least, the controllers, these will have the functionality of all the application, with the database queries necessary in the process. From the routes, each route redirects to each one of its functions in the controllers, giving logic to each possible action.

In the back-end, we have our algorithm. This is used for the creation of the weekly plan for the user to use. The way this algorithm works is quite complicated. Starting with a template of the recommended food pyramid, which tells you how many times you should eat each different category of food. From this, it organizes the different recipes and foods into the five different meals for each day. For the breakfast, brunch, and snack, it uses simple recipes, that is, simple foods to have in the “small” meals. For lunch and dinner it uses complete recipes, that is full recipes with measurements and preparation steps so the user can learn how to prepare them. I separated these two types of recipes because only during lunch and dinner you should have bigger meals, however in the small ones you should only have snacks. The way this algorithm works is the following, first of all, it starts assigning the mid-day meal, the lunch, this is divided into first plate and main plate. It starts with the first plate, it takes from the pyramid the different intervals of times each category should be eaten to organize the different recipes in those categories each day. Then it goes to the main plate, it does the same but with its respective categories. After that, it continues with dinners, using what it is recommended, it takes from the database the recipes which categories match with what should be used. Then, after assigning the whole recipes, the complex one, it goes to the simple ones, the ones used for breakfast,

brunch and snack, assigning the categories according to the different intervals of each one of them. The algorithm has a bit more complexity than these, this was a simple way of explaining what it does. After all of these, the user will have their weekly plan ready for them.

Lastly, but not least, we have Angular, it is an open-source JavaScript framework written in TypeScript, which combines HTML for the making of the page, CSS to give format and style to this page and TypeScript which with the functionality to the elements. Angular has multiple libraries and components to use. It is an “easy” way of handling a plain HTML and its JavaScript functionality, because TypeScript has more flexibility than JavaScript but not allowing you the same freedom as JavaScript has, that is, you will be able to make less mistakes because you will be able to structure everything better. The main library used is Angular Material, it is the most used one and personally, the best one, it has all the components that you would want to use. Also, you can set up a global theme on your app, namely, having a global CSS file which can allow you to organize and reduce code. Angular will help build interactive and dynamic single page applications, also known as SPAs throughout all of its features that include templating, two-way binding, modularization, RESTful API handling, dependency injection, and AJAX handling.

The front-end part of the application is divided into the following parts: components, guards, models and services. Starting off with the components, here we will have all the different pages or subpages of the application, separating all the different parts and views, allowing us to organize everything better, being also able to reuse a component or to nest ones inside others, without having thousands of lines of code in the same file. It is a great way of working and having everything mapped out. Each component will have a HTML file, a TypeScript file and a CSS file, like it was explained in the paragraph before this one, the HTML will have the plain view of the page, the CSS file will give this style, and the TypeScript will give it functionality, these files will have the calls to the services, and those are the ones that communicates with the back-end. Then we have the guards, a guard is exactly what it looks like, guarding the different components. In my case, I have a single guard which controls whether the user is logged in or not to let or forbid them to enter into different pages. To control this, I use cookies and local storage from the web browser, everything someone logs in, a cookie is set, and a user is stored, the guard will check if these are currently stored, because when the user logs out, those two are erased. Then, we have the models, like in the back-end, the structure of each object used in the database. And lastly, the services, these will be the ones having the API REST calls to

communicate with the back-end and the database. These are the connection between both projects, handling the requests and responses of all the possible functionality the application has, all these requests will match all the different possible routes previously established in the backend.

All these different parts make what we will see, as our web application.

1 Introducción

El avance y evolución del ser humano aumenta de forma exponencial, esto engloba tanto a los avances tecnológicos como pueden ser los avances médicos. Se podría decir que la tecnología y la medicina en los últimos 50 años, ha pegado un salto descomunal, y se podría decir que ambos están ligados, ya que el avance que ha tenido la medicina es en bastante porcentaje gracias a las nuevas tecnologías que han permitido avances en el diagnóstico o tratamiento de enfermedades (Jaramillo-Antillón, 2001). Toda esta evolución, proporciona a la población cada vez más conocimiento sobre posibles modos de mejorar nuestra salud, da la posibilidad de poder perfeccionar nuestra forma de vida y la calidad de esta.

Uno de los pilares que ayudan a mejorar nuestra calidad de vida, e incluso a alargarla, es la alimentación. Está juega un papel fundamental en la salud de un ser humano (DKV Equipo, 2021). Además, este avance mencionado previamente también ha proporcionado una amplitud en la variedad de alimentos que se pueden consumir, no siendo todos ellos saludables para obtener este objetivo. Por ello, es fundamental una alimentación saludable, tener el conocimiento de cuáles son aquellos alimentos que se deben consumir, y cuáles son aquellos que se deben intentar evitar.

Esto deriva a la búsqueda de una buena alimentación (FAO, 2014), para poder ayudarnos a mantener una vida larga y saludable. Realmente, no es complicado, simplemente es importante informar a la población de los conocimientos necesarios para poder tomar estas decisiones, la necesidad de contar siempre con profesionales que nos la proporcionen, alcanzar este objetivo de mantener una buena alimentación sin necesidad de mucho esfuerzo y al alcance de cualquiera (Alianza Team, 2019).

Por todo esto, se busca la forma de ayudar y proporcionar esta información. Con este proyecto es justo lo que se quiere conseguir, facilitar a cualquiera una ayuda extra creando y organizando unos planes de comida, siguiendo recomendaciones de profesionales, para que todos puedan seguirlos.

El desarrollo de este Trabajo Fin de Grado se ha dividido en distintas partes. En primer lugar, se lleva a cabo el estado del arte, aquí se describe todo lo que tenga relación con el contexto del problema que se quiere solucionar, se habla de herramientas similares que el mercado ofrezca, y posibles soluciones a estas que se hayan implementado en este

proyecto. También se habla sobre las distintas tecnologías que se han empleado en la realización de este proyecto. Después, se pasa a los objetivos y metodología empleados, estableciendo qué objetivos se quiere alcanzar con este trabajo y qué metodologías se han utilizado para conseguirlo. Luego, se pasa al apartado de diseño y desarrollo, donde se explica técnicamente cómo se ha desarrollado la aplicación. Y por último las conclusiones y posibles vías futuras.

2 Estado del arte

Hoy en día el conocimiento sobre la salud y la alimentación aumenta cada vez más. Gracias al avance de tecnologías y a los avances médicos nos damos cuenta de cómo cada vez más la alimentación afecta directamente a la salud. Organizaciones como la Organización Mundial de la salud realiza numerosos estudios los cuales demuestran que una dieta no saludable o una vida sedentaria puede causar numerosas enfermedades crónicas las cuales pueden derivar en la muerte. Esto ha hecho que surgieran gran cantidad de herramientas y plataformas las cuales ayudan a profesionales y no profesionales a organizar tantos conocimientos y poder consultar de esta forma las diferentes pautas que uno debe llevar para mantener una vida saludable.

Muchas de las plataformas que ya existen simplemente nos facilitan recetas o planes previamente creados para intentar guiarnos en nuestro día a día, es decir, plantillas previamente creadas sin ser personalizadas.

Algunas de las plataformas y aplicaciones más utilizadas en el ámbito alimenticio son:

- **Nooddle**¹: También conocida como ekilu, es una aplicación cuyo propósito es "come sano, con lo que tienes a mano". Esta herramienta intenta que puedas improvisar de forma sana con los ingredientes que tienes a mano en tu nevera.
- **FatSecret**²: es una aplicación que principalmente se basa en la ayuda para perder peso. Podrás hacerte un diario de dieta, hacer seguimiento de todo lo que comes, y hacer seguimiento de todos los nutrientes que ingieres.
- **Nootric**³: podrás crearte tus planes con recetas gratis. También cuenta con una parte de pago con la cual podrías llevar un seguimiento a medida por parte de un nutricionista.
- **Fitia**⁴: con esta aplicación podrás contar calorías, visualizar tu comida y calcular los macronutrientes presentes en tus recetas.

La mayoría de estas aplicaciones o plataformas se basan en llevar cuenta de los nutrientes ingerido durante las comidas. Esto es claramente un factor importante si estamos

¹ Web oficial de Noodle: <https://ekilu.com/es>

² Web oficial de FatSecret: <https://www.fatsecret.es/>

³ Web oficial de Nootric: <https://www.nootric.com/es>

⁴ Web oficial de Fitia: <https://fitia.app/>

centrados en una ganancia, pérdida o mantenimiento de peso. Algunas, aparte de esto, combinan bases de datos de recetas saludables las cuales podrías incluir en tu día a día, con la opción de organizarte tus propios planes basándote en dichas recetas. Todo esto ayuda bastante a cualquier persona que no esté dotada con los conocimientos de un profesional o nutricionista.

2.1 Posibles mejoras a herramientas ya existentes

Como se ha dicho en el apartado anterior la mayoría de las aplicaciones se basa en un seguimiento bastante exhaustivo de una dieta, en el cual tendrás que dedicarle bastante tiempo al registro de recetas o alimentos ingeridos para poder llevar cuenta de los todos los nutrientes.

Hoy en día la gente no tiene ni el tiempo ni las ganas de estudiar exhaustivamente cuáles son las pautas que uno debe seguir y qué comidas y alimentos se deben tomar para intentar llevar una vida saludable.

Conforme pasan los años, existen y se crean más malos hábitos en muchos ámbitos, pero sobre todo en el de la alimentación. Cada vez surgen más lugares de alimentación denominados como “comida basura” (QuiromSalud, 2018). Por ello es cada vez más complicado intentar abstenerse de estos malos hábitos y seguir una dieta equilibrada.

Pero no todo el mundo tiene estos conocimientos, por eso es muy importante informar a la población de qué hábitos hay que seguir, pero a su vez estos deben centrarse más en qué categorías de alimentos son necesarias o cuántas veces han de tomarse para mantener una dieta sana.

Por eso se ha creado esta aplicación, tienes la opción de ampliar con recetas a tu gusto las cuales son usadas para crear de forma automática un plan para ti sin necesidad de crearlo tú mismo.

2.2 Tecnologías utilizadas

En este apartado comentaremos las distintas tecnologías que han sido usadas para la creación de esta aplicación, todas ellas se juntan en lo que se denomina como pila MEAN, esto es un acrónimo para MongoDB, Express, Angular y NodeJS, todas explicadas en los siguiente apartados, pero primero explicaremos la pila.

2.2.1 MEAN Stack

MEAN stack o pila MEAN es un framework basado en JavaScript para el desarrollo web.



Figura 1. MEAN stack

Esto utilizada MongoDB como base de datos, Express y NodeJS para el backend y Angular para el frontend.

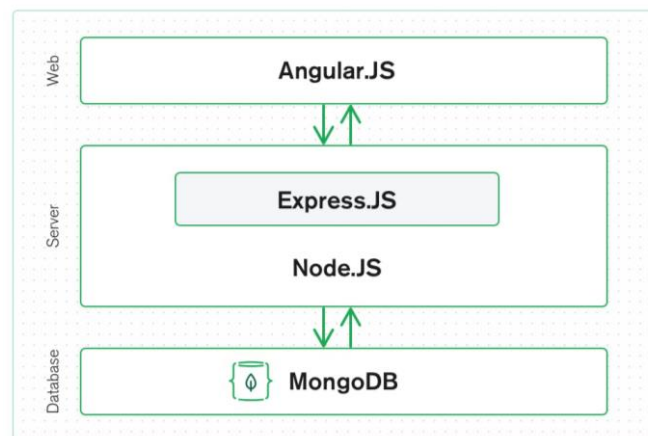


Figura 2. Arquitectura MEAN stack

Como se puede observar en la Figura 2, su arquitectura es sencilla, contamos con Angular para la parte web o parte del cliente, Express, el cual se encuentra incluido en NodeJS forman el servidor y MongoDB la base de datos. Todos estos serán explicados en más detalle en los siguientes apartados.

2.2.2 Front-end

El front-end de una aplicación es lo que se denominada, el lado cliente de esta, es decir, la interfaz, la cual contiene todas las páginas visuales y vistas de nuestra aplicación. Para esta aplicación se ha utilizado para desarrollar esta parte Angular⁵.

Angular es una plataforma de desarrollo que incluye un marco basado en componentes para crear aplicaciones web escalables, una amplia colección de bibliotecas que incluyen una amplia variedad de funciones incluido el enrutamiento, gestión de formularios, comunicación cliente-servidor o componentes básicos cómo tablas, botones, etc.

Angular utiliza HTML, TypeScript y CSS. HTML es un lenguaje de marcas de hipertexto, es el componente más básico de una página web, con el creamos las diferentes etiquetas que forman parte de nuestra página. CSS es un lenguaje de diseño gráfico para crear y definir el estilo de dichas etiquetas mencionadas previamente. Y por último, TypeScript, este es un lenguaje de programación libre y de código abierto, es también un súper conjunto de Javascript.

Tabla 1. Frameworks más usados para el front-end (Tápanes, 2022)

Nombre	Precio	Código Abierto	Licencia	Tipo de Aplicación	Curva de Aprendizaje	DOM Virtual	Creador
Angular	Gratuito	Sí	MIT	Una página (SPA)	Media	No	Google
React	Gratuito	Sí	MIT	Una página (SPA)	Media	Sí	Facebook
Vue	Gratuito	Sí	MIT	Una página (SPA)	Leve	Sí	Vue Core Team
Ember	Gratuito	Sí	MIT	Varias páginas	Media	No	Ember Core Team
Qooxdoo	Gratuito	Sí	MIT	Aplicaciones enriquecidas (RIA)	Alta	No	Qooxdoo Core Team
Dojo	Gratuito	Sí	BSD o AFL	Aplicaciones enriquecidas (RIA)	Alta	Sí	JS Foundation

Como se puede observar en la Tabla 1, Angular es de los frameworks más usados del mercado y esto se debe a numerosas ventajas que éste ofrece: Typescript utiliza

⁵ Web oficial de Angular: <https://angular.io/>

características de lenguajes orientados a objetos que permiten una más sencilla organización de objetos y variables, utiliza directivas con las cuales se podrán realizar una gran variedad de operaciones en el código HTML. Proporciona también un enlace bidireccional de datos con el cual comunica los ficheros HTML con sus funcionalidades, y también tiene la opción de realizar pruebas unitarias o de integración para la comprobación del código. Estas son algunas de las muchas ventajas que ofrece este framework.

En cuanto a Angular, se ha utilizado en particular Angular Material⁶. Esta es una librería de estilos basada en la guía de diseño de Material Design⁷. Material Design es un sistema adaptable de pautas herramientas y componentes los cuales son utilizados en el diseño de interfaces de usuario, además facilita la colaboración entre diseñadores y desarrolladores ayudando así a una mejora creación de productos atractivos para los clientes. Angular Material tiene una gran cantidad de componentes plantilla los cuales tienen un sencillo uso y una sencilla incorporación en nuestro proyecto Angular.

2.2.3 Back-end

El back-end de una aplicación es la parte del desarrollo web que se encarga de que la parte lógica de la página web funcione. Consiste en un conjunto de acciones que suceden en nuestra página web, pero sin ser visualizadas, es decir, funcionalidades de cada página. Esta parte es también denominada la parte server-side de la aplicación.

Para esta parte del proyecto se ha utilizado NoseJS⁸, este es un entorno de tiempo de ejecución de JavaScript. JavaScript es un lenguaje que, antes, solo era ejecutable en los navegadores, pero gracias a al entorno de NodeJS, javascript será ejecutable en cualquier sitio. Realmente lo que hace es convertir el código en código máquina más rápido.

NodeJS utiliza un sistema de entrada y salida sin bloqueo, es decir, hablamos de entradas y salidas refiriéndonos a peticiones y respuestas.

Junto a esto, se ha utilizado ExpressJS⁹, es una infraestructura de aplicaciones web para NodeJS. Express es utilizado para la creación de APIs. Ambas se han utilizado para la creación y uso de una API REST, esto es una interfaz de programación de aplicaciones

⁶ Web oficial de Angular Material: <https://material.angular.io/>

⁷ Web oficial de Material Design: <https://material.io/>

⁸ Web oficial de NodeJS: <https://nodejs.org/es/>

⁹ Web oficial de Express: <https://expressjs.com/es/>

(API) que se junta con la arquitectura REST y permite interaccionar con servicios web RESTful.

NodeJS y Express son de las tecnologías más usadas juntas para el desarrollo de la parte back-end de una aplicación web.

Para mantener un control sobre la autenticación en la aplicación, es decir, llevar un control sobre los usuarios que se conectan, se ha utilizado Json Web Token¹⁰. Esto es un estándar abierto que sirve para la creación de tokens de acceso lo cual nos permite llevar un control sobre la identidad o privilegios que queramos establecer en la aplicación.

2.2.4 Base de datos

Una base de datos es un lugar donde almacenar un conjunto de elementos. Estas se dividen en dos diferentes tipos: las relacionales o no relacionales.

Las bases de datos relacionales o SQL, son las más amplias. Estas son las bases de datos que se crearon primero y las que llevan más tiempo en el mercado, siempre se han usado. Este tipo de bases de datos no permiten diferencias, ya que tienen poca flexibilidad, todos los documentos deben mantener el mismo formato, y todos deben ser estructuralmente idénticos. Es más costoso su trabajo y mantenimiento.

Por otro lado, están las bases de datos no relacionales o NoSQL, estas son menos amplias. Fueron creadas después que las relacionales y llevan menos tiempo en el mercado. Tienen bastante más flexibilidad que las SQL ya que permite el almacenamiento de elementos sin tantas restricciones. Es mucho más sencillo el trato de los elementos, permite el almacenamiento de objetos que no estén estructurados exactamente igual. Si se necesita tratar con objetos más manejables y que se adapten a cada situación de forma distinta, usar este tipo de base de datos será la mejor opción. Por esto, en la aplicación se ha usado, MongoDB¹¹, es una base de datos no relacional de código abierto. De las bases de datos no relacionales es la más usada.

Se ha decidido usar una base de datos no relacional por la flexibilidad que aporta para los diferentes elementos que se necesitan almacenar, y MongoDB es la mejor opción. Permite

¹⁰ Web oficial de JSON Web Token: <https://jwt.io/>

¹¹ Web oficial de MongoDB: <https://www.mongodb.com/es>

un modelado de datos más manejable, una gran flexibilidad y una sencilla sintaxis para las consultas.

2.2.5 Herramientas usadas para el proyecto

Es importante destacar las distintas herramientas que han sido utilizadas para desarrollar el proyecto. Se ha utilizado un repositorio Git para su almacenamiento, en concreto, GitHub¹², esto es de vital importancia para mantener actualizado y organizado tu proyecto con la opción de visualizar cada paso que has ido dando, y pudiendo volver atrás si fuera necesario.

Para la programación se ha utilizado la aplicación de Visual Studio Code¹³, es una de las aplicaciones más usadas en el ámbito de la programación. Algunas de las ventajas que tiene podría ser su gran cantidad de extensiones que te ayudan en el proceso de desarrollo, ofrece un autocompletado en varios de los lenguajes, al tener una terminal integrada es más sencillo para este tipo de desarrollo web.

Para la parte de la base de datos se ha utilizado la aplicación de Robo3T¹⁴, esta aplicación sirve para la gestión de bases de datos de MongoDB y tiene una sencilla interfaz la cual permite un sencillo uso de ella.

Por último, se ha utilizado como navegador Firefox¹⁵. Cualquier otro navegador sería valido pero se ha decidido hacer uso de este, debido a su privacidad, a su rapidez y a su ligero consumo (BBC New Mundo, 2018).

¹² Web oficial de GitHub: <https://github.com/>

¹³ Web oficial de Visual Studio Code: <https://code.visualstudio.com/>

¹⁴ Web oficial de Robo3T: <https://robomongo.org/>

¹⁵ Web oficial de Firefox: <https://www.mozilla.org/es-ES/firefox/new/>

3 Análisis de objetivos y metodología

En esta sección se van a abordar los objetivos a los que se quiere llegar con el desarrollo de este proyecto, así como la metodología empleada para alcanzar estos objetivos.

3.1 Objetivos

El objetivo de este TFG crear un plan de comidas semanal de tal forma que el usuario pueda organizarse una dieta sin necesidad de tener grandes conocimientos en los ámbitos alimenticios. En la aplicación se podrá observar las diferentes pirámides alimenticias dependiendo de la dieta y actualmente será capaz de generar lo que se considera un plan equilibrado y sano basándose en dicha pirámide siguiendo una dieta regular. Este plan es realizado escogiendo diferentes recetas almacenadas en la base de datos. El usuario podrá visualizar dicha base de datos de recetas y también de alimentos e incluso tendrá la opción de añadir nuevas recetas para que éstas puedan ser escogidas para el plan. Sí al añadir alguna receta el usuario no pudiera encontrar algún ingrediente también tendrá la opción de añadir alimentos a la base de datos de alimentos. Si el usuario no está conforme con su plan siempre tendrá la opción de generarse uno distinto.

Para conseguir el objetivo de este proyecto se han realizado las siguientes tareas:

1. Estudiar minuciosamente aplicaciones o plataformas existentes en el mercado que tratan sobre nutrición, buscando defectos y posibilidades de mejora.
2. Escoger las tecnologías a usar tanto back-end como front-end, para elegir la mejor opción para realizar este proyecto.
3. Estructurar las ideas que se querían plasmar en el proyecto realizando un diseño de aplicación para tener así una idea general de todo lo que se quería abarcar.
4. Elegir las fuentes para seleccionar las recetas y alimentos que se iban a almacenar en la base de datos.
5. Implementar las vistas del front-end.
6. Crear el esquema de base de datos:
 - a. Insertar datos sobre alimentos con sus nutrientes.
 - b. Implementar un programa para crear de forma automática los recetas para insertar en la base de datos, a falta de relacionar.
 - c. Establecer las relaciones entre los ingredientes de las recetas con los alimentos.

7. Implementar la funcionalidad de la aplicación (front-end y back-end).
8. Realizar pruebas para comprobar que toda aplicación funciona.
9. Redactar la memoria.

A continuación, mostramos en una tabla el tiempo aproximado que nos ha llevado a realizar cada tarea.

Tabla 2. Tiempos estimados de la relación del proyecto

Tarea	Tiempo estimado
1	20 h
2	5 h
3	30 h
4	10 h
5	50 h
6a	10 h
6b	15 h
6c	100 h
7	80 h
9	5 h
10	20 h
Total	345 h

3.2 Metodología

Para poder obtener el objetivo propuesto al inicio para la aplicación y que se han mencionado en el apartado anterior se ha seguido una metodología tradicional en cascada para seguir cada uno de los siguientes pasos de forma cronológica, ya que cada uno de ellos depende de los anteriores. El proyecto se puede dividir en cinco importantes partes:

1. Análisis y estudio de las tecnologías. En esta fase podemos incluir toda la parte que implica el análisis previo tanto de plataformas o aplicaciones relacionadas con la nutrición ya existentes, como de las tecnologías que podían ser usadas para esta aplicación web.

2. Diseño y organización de la aplicación. En este apartado se organizan todas las partes que se quieren incluir en la aplicación realizando distintos diseños de aplicación y creando los diferentes casos de uso que se quieren tratar.
3. Definición y creación del esquema de la base de datos.
4. Implementación. Desarrollo del código de todas las partes del proyecto en las tecnologías seleccionadas.
5. Pruebas. Comprobar que todas las partes de la aplicación funcionan correctamente.

4 Diseño y resolución del trabajo realizado

En este apartado se va a presentar el diseño, la implementación y la resolución de este proyecto.

4.1 Casos de uso

Comenzaremos este apartado hablando sobre los casos de uso que se han tratado en esta aplicación. Un caso de uso es una técnica para la captura de requisitos potenciales de un sistema, con cada caso de uso se facilitan uno o más escenarios con los que se indican cómo debería interactuar el sistema con el usuario. Estos casos de uso engloban los requisitos del sistema, es decir, todo lo que la aplicación tiene que hacer.

En este caso para este proyecto solo hay un solo actor, este es el usuario que vaya a usar la aplicación.

Vamos a dividir estos casos de uso en diferentes partes dependiendo de qué acciones realicen a lo largo de la aplicación.

4.1.1 Autenticación

En este apartado simplemente vamos a incluir las acciones que un usuario puede realizar para autenticarse o que tienen que ver con el perfil, esto es iniciar sesión, registrarse, editar el perfil o cerrar la sesión.

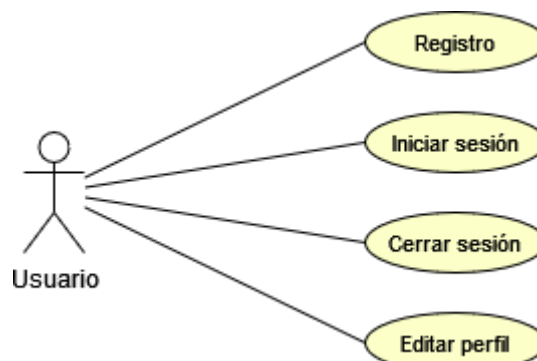


Figura 3. Casos de uso para autenticación.

- Registro: Un usuario podrá introducir unas credenciales para crearse una cuenta en la aplicación, estas credenciales engloban el nombre y apellidos, email y contraseña.
- Inicio de sesión: Un usuario podrá iniciarse en la aplicación con las credenciales de su email y contraseña.
- Edición de perfil: Un usuario podrá, en cualquier momento, cambiar cualquiera de los datos introducidos en el registro tanto nombre y apellidos como el email o la contraseña.
- Cerrar sesión: Un usuario podrá, en cualquier momento, salirse de la aplicación y cerrar su sesión.

4.1.2 Recetas

Los casos de uso relacionados con las recetas van a ser aquellos que realizan visualización y actualización en la base de datos, ya que el usuario tiene la opción de consultar o añadir recetas para ser usadas en la aplicación.

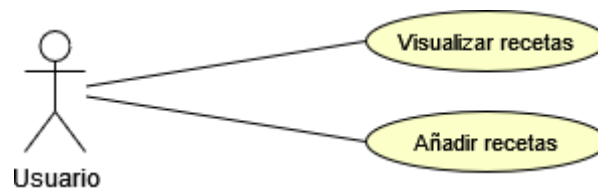


Figura 4. Casos de uso para las recetas

- Visualizar recetas: el usuario tendrá la opción de consultar la base de datos de las recetas tanto las complejas, usadas para las comidas y las cenas, como las recetas simples, usadas para el desayuno almuerzo y merienda.
- Añadir recetas: el usuario tendrá la opción de añadir recetas a la base de datos para poder ser consideradas en la creación de la dieta.

4.1.3 Alimentos

Los casos de uso relacionados con los alimentos son, como en las recetas, una visualización y actualización en la base de datos, ya que el usuario tiene la opción de consultar o añadir alimentos para ser usados en la aplicación.

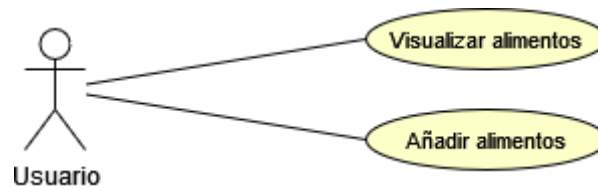


Figura 5. Casos de uso para los alimentos

- Visualizar alimentos: el usuario tendrá la opción de consultar la base de datos de los alimentos usados para las recetas.
- Añadir alimentos: el usuario tendrá la opción de añadir algún alimento a la base de datos para poder ser considerado como ingrediente en la creación de las recetas.

4.1.4 Pirámide alimenticia

En caso de uso relacionado con la pirámide es solo uno, ya que, si tu plan es llevar una dieta saludable, solo hay una forma recomendada en la que deberías organizar tus comidas, siguiendo la pirámide alimenticia recomendada por los expertos, por lo que este objeto solo será consultable y no editable.

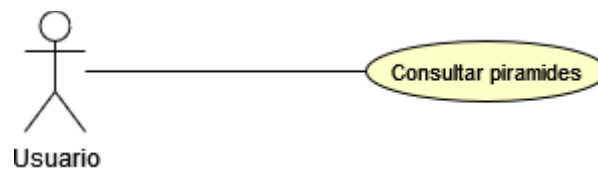


Figura 6. Casos de uso para las pirámides.

- Consultar pirámides: el usuario tendrá la opción de consultar la pirámide con la que se le va a realizar su dieta, que, a día de hoy, simplemente es la dieta regular. Aunque también tendrá la opción de consultar cómo son las pirámides para una dieta tanto vegetariana como vegana.

4.1.5 Dieta

En este apartado se comentan los casos de uso relacionados con la creación y la consulta de la dieta que se crea para el usuario, podrá crear una dieta, consultarla o sustituirla. Un usuario podrá crear más de una dieta, pero cada una será sustituida por la anterior.

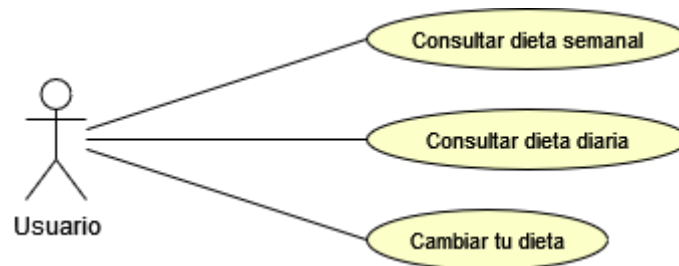


Figura 7. Casos de uso para la dieta.

- Consultar dieta semanal: el usuario podrá consultar su plan semanal de dieta. Si es la primera vez que se realiza esta acción se le creará dicha dieta.
- Consultar dieta diaria: el usuario podrá consultar de su plan semanal cualquier día en detalle, es decir, visualizar más a fondo la dieta de un día concreto.
- Cambiar la dieta: el usuario podrá cambiar la dieta semanal que se le ha asignado, de esta forma se le volverá a crear de nuevo otra dieta, sustituyendo la que tenía previamente.

4.2 Diseño de la aplicación

En este apartado se va a mostrar el diseño de la aplicación explicando cómo se han utilizado las tecnologías de la MEAN stack. Comenzaremos con la base de datos para dar forma a los objetos que vamos a tratar en el resto de los apartados del front-end y del back-end.

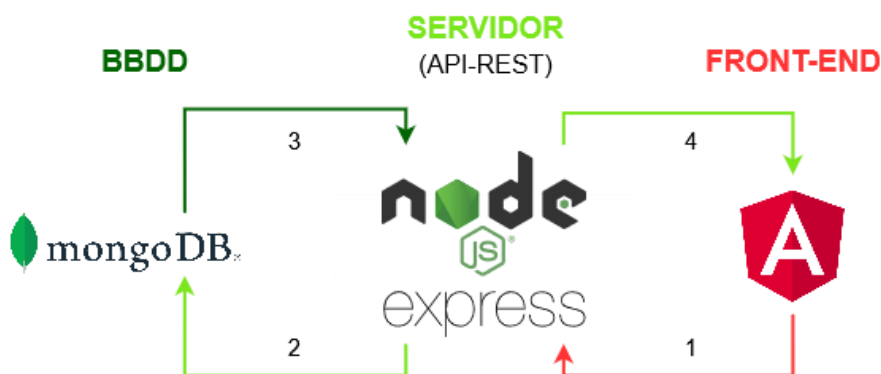


Figura 8. Arquitectura de la pila MEAN

4.2.1 Base de datos

Siguiendo la propuesta de la pila MEAN, la base de datos utilizada ha sido la base de datos no relacional MongoDB. En la Figura 9 se muestra el modelo conceptual del

esquema de base de datos de la aplicación representado mediante un diagrama de clases de UML.

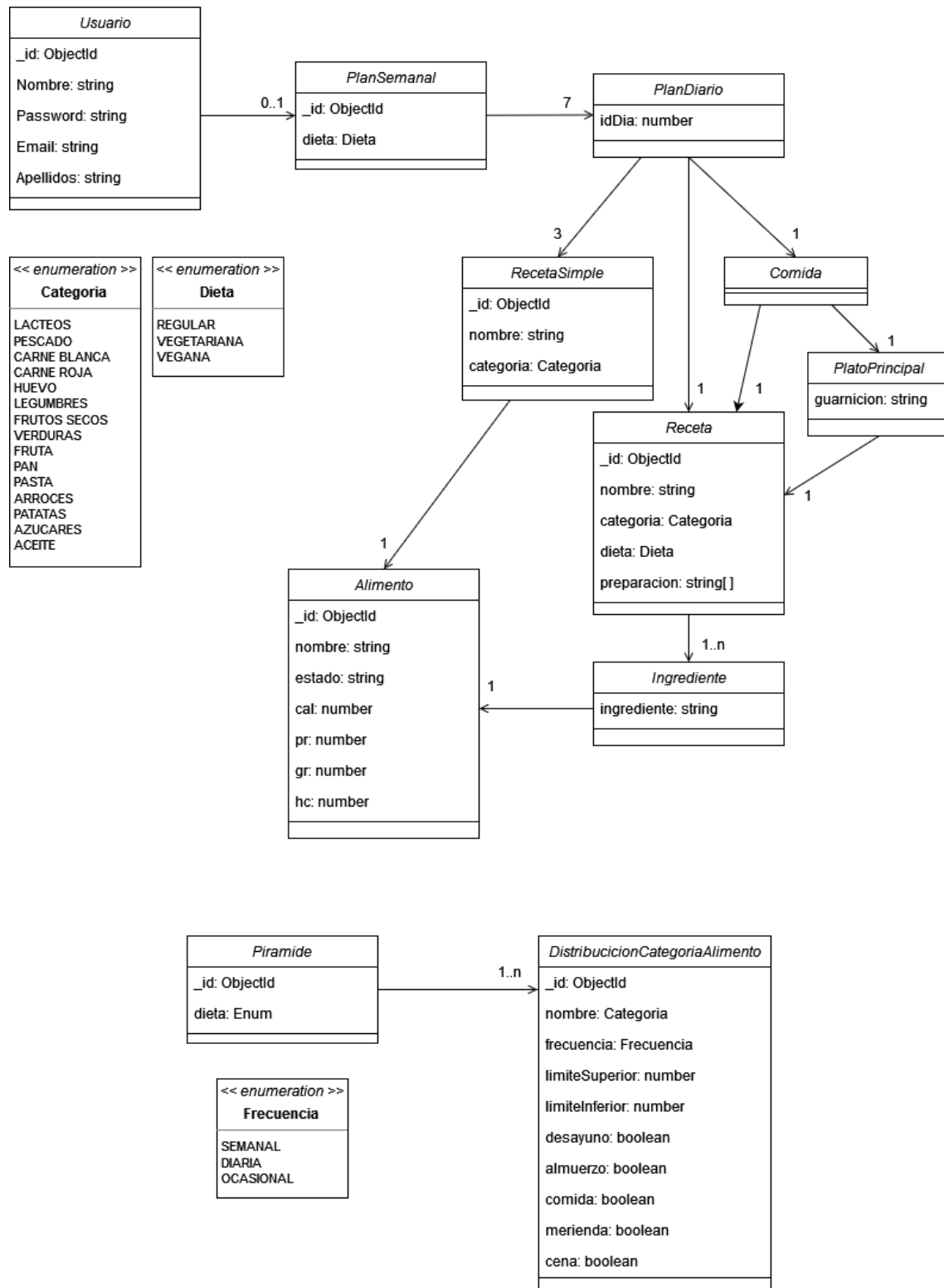


Figura 9. Diseño de la base de datos

Vamos a comenzar explicando la clase Usuario. Un usuario se caracteriza por el nombre, apellidos, email, contraseña y un plan semanal, que puede tener o no tener.

La clase **Plan Semanal** almacena la dieta que se va a crear. Esta clase contiene una propiedad que indica de qué clase de dieta se trata: regular, vegetariana o vegana. Aunque actualmente solo se tratan con dietas regulares la aplicación está creada de tal forma que se pueda incluir de forma sencilla en un futuro una dieta vegana o vegetariana. La clase **Plan Semanal** tiene también un listado de planes diarios, en concreto siete, para los siete días de la semana

La clase **Plan Diario** sirve para almacenar las comidas que se van a tomar durante ese día. Esta clase tendrá 3 **Recetas Simples**, para el desayuno almuerzo y merienda, una **Receta** para la cena y otra **Receta** para la comida principal.

La clase **Receta Simple** se caracteriza por un nombre, un listado de categorías a las que puede pertenecer, y un listado de **Alimentos** que incluye la receta.

La clase **Comida**, está formada por dos propiedades: el primer plato, este será una **Receta**, y el **Plato Principal**.

El **Plato Principal** almacena un par de propiedades, la **Receta** de dicho plato principal y su guarnición.

La clase **Receta** tiene las siguientes propiedades: categoría, es decir a qué categoría alimentaria pertenece esta receta, nombre, dieta, preparación, esto es, un listado de pasos que hay que seguir para la elaboración de la receta, y un listado de **Ingredientes**.

La clase **Ingrediente** relaciona el nombre de dicho ingrediente con el **Alimento** con el que se relaciona

La clase de **Alimento** tiene como propiedades: nombre, estado (crudo, frito, cocido...) y los nutrientes que este contiene, es decir, sus calorías, proteínas, grasas e hidratos de carbono.

Por otro lado, la clase **Pirámide** almacena simplemente la dieta a la que pertenece y un listado de **Distribuciones de Categorías de Alimentos**, esto es cada una de las distintas categorías a las que puede pertenecer un alimento.

Para finalizar, la clase **Distribuciones de Categorías de Alimentos** tiene como propiedades el nombre, límite inferior y superior, esto es el intervalo de veces que se debe

tomar esta categoría de alimento, frecuencia, para definir si el intervalo anterior es diario o semanal, y variables auxiliares que nos indican para cada categoría en qué momentos del día pueden ser tomadas.

Por último, como se ha explicado al inicio del apartado, el tipo de relaciones establecidas entre los objetos. Aquellas clases que tienen su propia colección en la base de datos son los usuarios, recetas simples, recetas complejas, los alimentos, la pirámide y los planes de comida, es decir, para relacionar cualquiera de estos objetos entre sí, se realiza a través del id, por ejemplo, un usuario tiene un plan de comida, en las propiedades del usuario aparecerá el id del plan de comidas que le corresponde. En cambio, para el resto de los objetos no mencionados serán almacenados en sí en el objeto, porque son lo suficientemente simples para ello o no tiene sentido que tengan su propia colección en la base de datos, por ejemplo, un plan semanal tiene un listado de planes diarios, los cuales se almacenan directamente en el objeto de plan semanal, ya que los planes diarios no se almacenan individualmente.

4.2.2 Back-end

Para el desarrollo backend, como se ha explicado en apartados anteriores, se han utilizado las tecnologías de Express y NodeJS. Explicaremos sección a sección cómo se ha organizado este proyecto.

La carpeta principal de este proyecto contendrá de primeras 3 archivos para conectarlos con la base de datos y la aplicación en un puerto concreto. Esta clase principal recogerá las diferentes rutas por las que puedes ser llamadas desde el frontal y las redirigirá a sus respectivos ficheros de rutas.

10. /api → /routes/userRoutes.js
11. /api/piramide → /routes/piramideRoutes.js
12. /api/alimento → /routes/alimentoRoutes.js
13. /api/receta → /routes/recetaRoutes.js
14. /api/receta/simple → /routes/recetaSimpleRoutes.js
15. /api/plan → /routes/planRoutes.js

Cada una de estas rutas que sea recibida desde el frontal será redirigida a su respectivo fichero para ser tratada.

El proyecto del back-end está organizado en tres carpetas: controllers, models y routes.

4.2.2.1 Routes

La carpeta Routes contendrá unos archivos, estos archivos simplemente servirán también para redirigir, es decir no tendrán un gran funcionamiento, simplemente redirige cada una de las peticiones lanzadas desde el frontal a su correspondiente fichero de controladores dónde ya se implementa cada una de las funciones.

A continuación, mostraremos una secuencia de tablas donde se podrá observar las diferentes peticiones con sus rutas, tipos de operaciones, a dónde se le va a redirigir y una breve explicación, ya que cada una de estas funciones serán explicadas con más detalle en el apartado de controlador.

Tabla 3. Rutas para Usuario

PETICION	TIPO DE OPERACION	REDIRIGIR	BREVE EXPLICACION
/	GET	getUsers	Devuelve todos los usuarios
/register	POST	createUser	Crea y almacena un nuevo usuario
/login	POST	loginUser	Autentica la existencia de un usuario para loguearse
/authenticate	POST	authenticate	Comprueba si hay algún usuario logueado
/getUser	GET	getUser	Devuelve el usuario actualmente logueado por su id
/getUserByEmail	GET	getUserByEmail	Devuelve el usuario actualmente logueado por su email
/update	PUT	editUser	Edita un usuario previamente almacenado

Tabla 4. Rutas para Piramide

PETICION	TIPO DE OPERACION	REDIRIGIR	BREVE EXPLICACION
/	GET	getPiramides	Devuelve todas las pirámides
/getPiramide	GET	getPiramideById	Devuelve una pirámide por id
/getPiramideByDieta	GET	getPiramideByDieta	Devuelve una pirámide por dieta

Tabla 5. Rutas para Alimento

PETICION	TIPO DE OPERACION	REDIRIGIR	BREVE EXPLICACION
----------	-------------------	-----------	-------------------

/	GET	getAlimentos	Devuelve todos los alimentos
/getNumAlimentos	GET	getNumAlimentos	Devuelve el número de alimentos almacenados
/create	POST	createAlimento	Crea y almacena un nuevo alimento
/getAlimento	GET	getAlimento	Devuelve un alimento por id
/getMultipleAlimento	GET	getMultipleAlimento	Devuelve múltiples alimentos por su id

Tabla 6. Rutas para Receta

PETICION	TIPO DE OPERACION	REDIRIGIR	BREVE EXPLICACION
/	GET	getRecetas	Devuelve todas las recetas
/getNumRecetas	GET	getNumRecetas	Devuelve el número de recetas almacenados
/create	POST	createReceta	Crea y almacena una nueva receta
/getReceta	GET	getReceta	Devuelve una receta por id
/getRecetasByCategoria	GET	getRecetasByCategoria	Devuelve múltiples alimentos por su id

Tabla 7. Rutas para Receta Simple

PETICION	TIPO DE OPERACION	REDIRIGIR	BREVE EXPLICACION
/	GET	getRecetas	Devuelve todas las recetas
/getNumRecetas	GET	getNumRecetas	Devuelve el número de recetas almacenadas
/create	POST	createReceta	Crea y almacena una nueva receta
/getReceta	GET	getReceta	Devuelve una receta por id

Tabla 8. Rutas para Plan

PETICION	TIPO DE OPERACION	REDIRIGIR	BREVE EXPLICACION
/create	POST	createPlan	Crea y almacena un nuevo plan
/getPlan	GET	getMyPlan	Me devuelve el plan del usuario

Todo esto engloba las diferentes peticiones que podemos realizar en la aplicación. Se explicarán con más detalle en los siguientes apartados.

4.2.2.2 Models

En la carpeta Models estarán los archivos que almacenarán los esquemas de la base de datos. Para MongoDB se utiliza una librería llamada Moongose¹⁶, esta es una librería basada en esquemas para modelar los datos.

Estos esquemas comparten las mismas propiedades que los elementos de las bases de datos explicados en el apartado anterior, ya que son los mismos objetos tratados, y es aquí donde se estructura dichos objetos almacenados en la base de datos.

Cada esquema define sus propiedades, el tipo de variable que es y sí va a ser una propiedad obligatoria u opcional. Estos esquemas también pueden ser almacenados uno dentro de otro, de esta forma se pueden relacionar entre sí las distintas clases.

4.2.2.3 Controllers

La carpeta Controllers contendrá los diferentes controladores de la aplicación. Aquí se encuentran las funciones que se han mencionado en el apartado Routes, a dónde será redirigido el usuario dependiendo de la petición establecida.

Se va a explicar función a función lo que realiza cada una.

4.2.2.3.1 Alimento

La funcionalidad que ofrece el controlador de Alimento es:

getNumeroAlimentos

¹⁶ Web oficial de Moongose: <https://mongoosejs.com/>

Esta función simplemente cuenta el número de alimentos que hay almacenados en la base de datos. Esto se realiza con la función `count()`.

getAlimentos

Esta función devuelve todos los alimentos almacenados en la base de datos. Esto se realiza con la función de `find()` sin ningún parámetro.

createAlimento

Esta función crea un nuevo alimento y lo almacena en la base de datos. Para ello obtenemos el número de alimentos ya almacenados para aumentar en uno el id colocando así este nuevo alimento el último. Una vez creado el objeto lo almacenamos en la base de datos con la función `save()`.

getAlimentoById

Esta función recibe como parámetro un id y busca en la base de datos el alimento que corresponda a dicho id. Esto lo realiza con la función `find()` añadiéndole de parámetro este id de la siguiente forma: `find({idAlimento: id})`

getMultipleAlimento

Esta función ha sido creada para no tener que repetir la función anterior en el caso de necesitar más de un objeto, de esta forma no hay que ir esperando al tiempo de petición y respuesta, sino que lo podemos recibir más de un alimento en la misma llamada. Esta función sirve para obtener numerosos objetos por sus ids. Para ello utilizaremos de nuevo la función `find` añadiéndole el parámetro de búsqueda `$in` de la siguiente forma: `find({idAlimento:{"$in": [id1,id2,...]}})`.

4.2.2.3.2 Pirámide

La funcionalidad que ofrece el controlador de la clase Pirámide es:

getPiramides

Esta función devuelve todas las pirámides almacenadas en la base de datos. Simplemente utilizamos la función `find()` sin ningún parámetro.

getPiramideById

Esta función devuelve la pirámide asociada al id que se le establece a la función por parámetro de la siguiente forma `find({idPiramide: id})`.

getPiramideByDieta

Esta función va a devolver la pirámide asociada a la dieta que se le envía por parámetro. Como se ha explicado en otros apartados, aunque la aplicación actualmente solo crea un plan para una dieta regular, también ofrece la posibilidad de visualizar las pirámides para dietas veganas y vegetarianas. Esto se realiza de la siguiente forma `find({dieta: dieta})`

4.2.2.3.3 Receta

La funcionalidad que ofrece el controlador de la clase Receta es:

getNumeroRecetas

Esta función cuenta el número de recetas que hay almacenadas en la base de datos. Esto se realiza con la función `count()`.

getRecetas

Esta función devuelve todas las recetas almacenadas en la base de datos. Esto se realiza con la función de `find()` sin ningún parámetro.

createReceta

Esta función crea una nueva receta y la almacena en la base de datos. Para ello obtenemos el número de recetas ya almacenados para aumentar en uno el id colocando así esta nueva receta la última. Una vez creado el objeto lo almacenamos en la base de datos con la función `save()`.

getRecetaById

Esta función recibe como parámetro un id y busca en la base de datos la receta que corresponda a dicho id. Esto lo realiza con la función `find()` añadiéndole de parámetro este id de la siguiente forma: `find({idReceta: id})`

getRecetasByCategoria

Esta función recibe como parámetro una categoría y busca en la base de datos las recetas que correspondan a dicho categoría. Esto lo realiza con la función `find()` añadiéndole de parámetro esta categoría de la siguiente forma: `find({categoria: categoria})`

4.2.2.3.4 Receta Simple

La funcionalidad que ofrece el controlador de la clase Receta Simple es:

getNumeroRecetas

Esta función cuenta el número de recetas simples que hay almacenadas en la base de datos. esto se realiza con la función `count()`.

getRecetas

Esta función devuelve todas las recetas simples almacenadas en la base de datos. Esto se realiza con la función de `find()` sin ningún parámetro.

createReceta

Esta función crea una nueva receta simple y la almacena en la base de datos. Para ello obtenemos el número de recetas simples ya almacenados para aumentar en uno el id colocando así esta nueva receta simple la última. Una vez creado el objeto lo almacenamos en la base de datos con la función `save()`.

getRecetaById

Esta función recibe como parámetro un id y busca en la base de datos la receta simple que corresponda a dicho id. Esto lo realiza con la función `find()` añadiéndole de parámetro este id de la siguiente forma: `find({idReceta: id})`

4.2.2.3.5 Usuario

La funcionalidad que ofrece el controlador de la clase Usuario es:

getUsers

Esta función devuelve todos los usuarios almacenados en la base de datos, esto se realiza con la función `find()` sin ningún parámetro.

createUser

Función crea un nuevo usuario y lo almacena en la base de datos, pero primero comprueba si hay algún usuario ya creado que tenga el mismo email ya que esta propiedad debe ser única. Para realizar esto se utiliza: `findOne({email: email})`. Si esto devuelve algún usuario quiere decir que si existe otro usuario con el mismo correo electrónico y se devolverá un mensaje de error, si no es así el usuario será guardado en la base de datos.

loginUser

Esta función permite a un usuario iniciar sesión en la aplicación. Para ello primero comprueba que este usuario existe comprobando de la misma forma que la función anterior si ya está registrado ese email: `findOne({email: email})`. Si esto devuelve un usuario quiere decir que existe. Entonces comprobamos si la contraseña del usuario en la base de datos coincide con la contraseña proporcionada desde la aplicación, si esta contraseña no coincide se devuelve un mensaje de error indicando que la contraseña es incorrecta.

Por otro lado, si no ha encontrado el usuario devolverá un mensaje de error indicando que la cuenta no existe.

Y si todo va bien, es decir, el usuario existe y la contraseña es correcta, entonces se devolverá un mensaje informando de que todo ha ido bien. En este paso, como se ha explicado en un apartado previo, se utiliza para el control de autenticación de los usuarios JSON Web Token, para realizar lo que se denomina como una ‘firma’ junto con el id de nuestro usuario y poder así llevar un control de autenticado. Se realiza de la siguiente manera: `jwt.sign({ _id: usuarioGuardado._id }, "clave_secreta");`

authenticate

Esta función comprueba si hay actualmente un usuario que haya iniciado sesión. Para ello, tenemos que comprobar si el token proporcionado como parámetro y el que tendría que haberse obtenido cuando dicho usuario se ha logueado. Se verifica si dicho token existe como clave secreta. Se realiza de la siguiente manera:

`jwt.verify(token, "clave_secreta", function (err, data){...})`. Dentro de la función establecida en el método comprobamos si en efecto esto sucede, ya que nos devolverá la variable `err` en caso de error y `data` si todo va bien. Dependiendo de esto devolveremos un mensaje de error o de éxito a la aplicación.

getUser

Esta función devuelve el usuario que esté en el sistema actualmente, para ello utiliza la cookie que se creó al loguearse desde la aplicación, la cual contiene el id del usuario, y con este lo busca en la base de datos. Se tiene que decodificar esta cookie a través de JSON Web Token de la siguiente manera: `jwt.decode(cookie)`. Esta función devuelve el id y a través de este se busca el usuario: `findById(id)`

getUserByEmail

Esta función devuelve el usuario asociado al correo electrónico proporcionado como parámetro. Buscamos por parámetro como se ha hecho anteriormente: `find({email: email})`

editUser

De la misma forma que obtenemos el usuario que este en el sistema actualmente, decodificamos a través de la cookie guardada el id del usuario, y utilizamos esto para encontrar y actualizar este usuario. Esto lo realizamos de la siguiente manera: `findByIdAndUpdate(id, {$set: user})`. Esta función busca el usuario por id en la base de datos y con el parámetro `$set` indicamos qué queremos sustituir lo encontrado con el usuario pasado la función.

4.2.2.3.6 Plan

Por último, el controlador de la clase del Plan contiene el algoritmo que va a permitir crear un plan de dieta. La información necesaria para la creación de la dieta ha sido obtenida de artículos de nutricionistas de la Sociedad Española de Nutrición Comunitaria (SENC, 2018)

getMyPlan

Esta función devuelve el plan asociado al usuario conectado. Cada usuario tiene la información del id de su plan asociado, que será pasado como parámetro a esta función para buscar así el plan en la base de datos. Esto lo realizamos de la misma forma que hemos visto anteriormente con `find({id: id})`

createPlan

Esta función es el algoritmo principal de la aplicación. Se irá explicando paso por paso lo que se ha ido realizando.



Figura 10. Pirámide alimenticia recomendada por la SENC

En la figura 10, se puede observar la pirámide de la que se habla a lo largo de la explicación de este algoritmo.

Actualmente este algoritmo genera un plan de comidas para una dieta regular, por lo que comenzamos buscando en la base de datos la pirámide correspondiente a dicho tipo de dieta. Con esta pirámide vamos recorriendo las distintas categorías de alimentos para realizar un cálculo de intervalos de cada una. Para ello, se han creado dos mapas que asocian cada categoría con el número de veces que se debe tomar. Se ha realizado una separación en dos mapas debido a que uno gestiona los intervalos semanales y el otro los diarios.

Como cada categoría tiene un límite inferior y otro superior, se escoge un número de forma aleatoria entre ambos límites y se almacena en su respectivo mapa, estableciendo como clave el nombre de dicha categoría. Esto permitirá posteriormente ir gestionando

estos intervalos de forma rápida dependiendo de la categoría que se esté tratando. El mapa de intervalos semanales simplemente asocia el nombre de la categoría con el valor generado para el intervalo, mientras que el mapa de intervalos diarios asocia el nombre con un array de siete elementos, que representan los siete días de la semana.

Una vez realizada esta parte comenzaremos con la asignación de comidas. Vamos a comenzar con las comidas principales ya que van a ser las más complejas. Una comida principal debe estar formada por un primer plato y un plato principal, se comenzará asignando estos primeros platos.

Para poder seguir una dieta sana y equilibrada las verduras deben tomarse entre 2 y 3 veces al día por lo que se deberían incluir en este primer plato, por lo que solo serán asignadas ensaladas, cremas o sopas. Para buscar aquellas recetas que se ciñan a estas características se ha realizado la siguiente consulta en la base de datos:

```
{
  "$and": [
    {
      "$or": [
        {"nombre": { "$regex": "ensalada", "$options": "i" }},
        {"nombre": { "$regex": "Crema", "$options": "i" }},
        {"nombre": { "$regex": "Sopa", "$options": "i" }}
      ]
    },
    {
      "$or": [
        {"categoria": "VERDURAS"},
        {"categoria": "LEGUMBRES"}
      ]
    }
  ]
}
```

Esta consulta devolverá todas las recetas que tengan como nombre ensalada, crema o sopa, y que pertenezcan a la categoría de verduras o legumbres. Sirve para buscar aquellas recetas que cumplan tanto la primera parte de la consulta como la segunda, por ello se usa el operador `$and`. Dentro de esta cada una de las partes de esta consulta utilizan el operador `$or`, con esto se indica que simplemente se tiene que cumplir una de las

opciones establecidas. Se utiliza el operador `$options` con la opción `'i'` para indicar que se ignoren las diferencias entre mayúsculas y minúsculas.

Una vez obtenidas todas las recetas que nos devuelve esta consulta seleccionaremos aleatoriamente siete de ellas para repartirlas en cada uno de los días del plan. Si el número de recetas obtenidas fuese menor de siete, se tiene en cuenta para rellenar el resto de días con repeticiones de estas recetas. Además, se disminuye del mapa de intervalos las veces que se ha usado cada categoría.

Una vez seleccionados los primeros platos para toda la semana, se configuran los platos principales. Siguiendo las recomendaciones de la SENC, se han establecido las siguientes pautas para seguir una dieta equilibrada que incluye la organización de las comidas principales de la semana de la siguiente manera:

- 1 día plato de arroz como ingrediente principal
- 1 día plato de legumbres como ingrediente principal
- 1 día plato de pasta como ingrediente principal
- Los 4 días restantes deben ser repartidos entre platos de carne y pescado, con una guarnición de arroz pasta o legumbres.

Comenzamos buscando las recetas de los 3 días de arroz, legumbres y pasta. Para ello se buscan en la base de datos las recetas realizando una consulta por parámetro de la siguiente manera `find({"categoria": "ARROCES"})`. La consulta será igual para recuperar las recetas de pasta, cambiando el nombre de la categoría por `"PASTA"`, y, para las recetas de legumbres se llevará a cabo de la siguiente manera:

```
{
  "$and": [
    {
      "nombre": {"$regex": "^(?!ensalada)", "$options": "i"}
    },
    { "categoria": "LEGUMBRES" }
  ]
}
```

Esta consulta devuelve aquellas recetas que pertenecen a la categoría de legumbres pero que no incluyen en su nombre la palabra ensalada. Como en el primer plato hemos incluido también posibles ensaladas de legumbres, para el plato principal se busca algún plato de legumbres pero que no sea una ensalada.

Se escoge aleatoriamente una receta de cada una de las consultas realizadas (una por categoría) y se asignan, también de forma aleatoria a tres días de la semana, controlando que no coincidan las legumbres en el primer plato y el principal. A continuación, se actualizan los mapas de intervalos para llevar la cuenta de las categorías usadas hasta el momento.

Después, se configura el plan diario de los cuatro días restantes. Siguiendo las recomendaciones nutricionales el resto de los días de la semana deben incluir recetas de carne y pescado, por lo que se asignarán dos días para cada tipo. Dentro de la carne se encuentran las categorías de carne blanca y roja. La carne roja debe ser tomada de forma ocasional, por lo que se generará un número aleatoriamente entre 0 y 1 para decidir si esta semana se va a tomar carne roja o no. Si se obtiene 0 la carne tomada esta semana será solo blanca, pero si es 1 se tomará, en estas comidas principales, una de cada. La carne roja al tratarse de una categoría que solo debe tomarse de forma ocasional, no se va a tratar como el resto de categorías con sus intervalos, sino que simplemente se escoge de esta forma al azar sí durante esta semana se va a tomar. Dependiendo de esto sustituiremos una de las dos comidas de carne blanca, por la carne roja, como momento excepcional en el que se toma.

Para esto hemos realizado las siguientes consultas:

```
[
  {
    "$lookup": {
      "from": "alimentos",
      "localField": "ingredientes.alimento",
      "foreignField": "idAlimento",
      "as": "vinculados"
    }
  },
  {
    "$match": {
      "categoria": "PESCADO"
    }
  }
]
```

Se realiza la consulta de esta manera para realizar una unión entre colecciones. El parámetro `$lookup` permite realizar esta unión entre dos colecciones, estableciendo la

propiedad de unión. Al incluir este parámetro en vez de utilizar la función `find()`, se debe utilizar `aggregate()`.

Esta consulta se le realiza a la colección de recetas, por lo que en el parámetro de unión establecemos qué propiedad se quiere utilizar de esta colección para la unión, junto con la propiedad que se quiera utilizar de la colección ajena. En este caso se va a unir con la colección de los alimentos, de esta forma por cada ingrediente se tendrá el alimento correspondiente cogido de su colección. Se une el alimento de cada ingrediente de la receta (esta propiedad es un id) con el objeto de la colección de Alimentos con dicho id. Esta unión se realiza para poder trabajar con las características y propiedades de cada uno de los ingredientes directamente sin necesidad de ir a buscar uno a uno a la colección de Alimentos.

El proceso es análogo para la selección de recetas de la categoría de carne blanca. Finalmente, se escogerán dos recetas de pescado, y una o dos recetas de carne blanca dependiendo de si en este caso se ha incluido o no una receta de carne roja. Las cuatro recetas obtenidas y se asignarán aleatoriamente a los cuatro días restantes de la semana. Conforme se van asignando las recetas a los platos principales se buscarán sus respectivas guarniciones. Esto se va a realizar de la siguiente forma: con cada plato se busca en el mapa de intervalos si todavía queda la categoría de pasta por asignar, si es así, se busca si alguno de los ingredientes del plato pertenece a la categoría de pasta, si ninguno de ellos pertenece entonces asignaremos en este plato como guarnición la pasta. Si no se cumplen ninguna de estas condiciones se realiza la misma búsqueda pero con la categoría de arroces y si tampoco se ha realizado una asignación aquí se realiza lo mismo pero con la categoría de patatas.

Concluido este proceso se han generado las 7 recetas principales del Plan Semanal: una de arroz, una de pasta, una de legumbres, dos recetas de pescado con sus respectivas guarniciones y dos recetas de carne también con sus guarniciones.

Procedemos a las cenas, según la SENC, las verduras se deben tomar de forma diaria y además no deben tomarse demasiados hidratos de carbono durante la noche. Por ello, para poder cumplir con las pautas recomendadas, se ha decidido establecer las verduras como categoría imprescindible durante la cena. Primero comprobamos si queda alguna carne blanca o pescado por asignar de acuerdo al mapa de intervalos, si es así buscaremos algunas recetas de estas categorías que entre sus ingredientes incluyan verduras. El resto de los días serán rellenados con recetas simplemente de verduras.

Vamos a suponer que sí que quedan recetas de carne blanca o pescado por asignar buscaremos entonces a través de la siguiente consulta:

```
[
  {
    "$lookup": {
      "from": "alimentos",
      "localField": "ingredientes.alimento",
      "foreignField": "idAlimento",
      "as": "vinculados"
    }
  },
  {
    "$match": {
      "$and": [
        { "categoria": "CARNE BLANCA" },
        { "vinculados.categoria": "VERDURAS" }
      ]
    }
  }
]
```

Como ya hemos explicado previamente utilizamos esta consulta para realizar una unión y poder acceder a propiedades de cada uno de los alimentos vinculados como ingredientes. Hacemos lo mismo con la categoría de pescado si es necesario. Esta consulta devolverá recetas de carne blanca o pescado que contengan verduras como ingredientes.

Cogemos entonces de los resultados tantas recetas de carne blanca y pescado cómo queden en sus respectivos intervalos. Filtramos estos resultados con aquellas recetas que ya han sido usadas previamente, para no elegir la misma receta.

Buscamos también recetas de verduras para ir rellenando los días restantes, simplemente buscados por el parámetro de categoría con `find({categoría: "VERDURAS"})`.

A continuación, se recorren los días de la semana comprobando los platos principales para, en el caso que sea necesario, no colocar las recetas restantes de carne blanca o pescado en aquellos días que tengan como plato principal estas categorías. El resto de los días serán rellenados con recetas de verduras. Actualizamos los mapas de intervalos con las categorías utilizadas.

Una vez finalizadas las recetas complejas de la comida y la cena, vamos a pasar al reparto de las recetas simples utilizadas para los desayunos, almuerzos y meriendas.

Comenzamos con los desayunos, escogemos siete recetas simples aleatoriamente, las colocamos en cada uno de los días y actualizamos las categorías que han sido usadas.

Pasamos entonces a los almuerzos y meriendas. Para esta parte, buscamos recetas que incluyan:

- Lácteos y no cereales:
`find({$and: [{categoria: "LACTEOS"}, {categoria: {$ne: "PAN"}}]})`
- Cereales y no lácteos:
`find({$and: [{categoria: "PAN"}, {categoria: {$ne: "LACTEOS"}}]})`
- Lácteos y cereales:
`find({$and: [{categoria: "PAN"}, {categoria: "LACTEOS"}]})`
- Frutas:
`find({categoria: "FRUTA"})`

Realizamos todas estas búsquedas para ir repartiendo recetas que se ajusten a los intervalos que queden. Se recorre día a día y se comprueba lo siguiente. Primero se comprueba si todavía en ese día quedan por asignar tanto recetas de pan como de lácteos, si esto se cumple, se elige de forma aleatoria una de las recetas que contienen ambas categorías, si esto no fuera así, se comprueba entonces individualmente cada una de las categorías. Si quedan para asignar en la categoría del pan, se escoge entonces una receta que contenga esta categoría, pero no la de lácteos, si esto tampoco fuera así entonces se comprobaría si quedan por asignar de la categoría de lácteos, si esto es así se asigna entonces una receta aleatoria de aquí. Y si este tampoco fuera el caso, y no nos quedan por asignar ninguna de ambas categorías, se escoge aleatoriamente una de las recetas de la fruta.

Cada día se van realizando estas comprobaciones y se repite el mismo proceso para asignar una merienda. De esta forma se va asignando día a día tantas recetas sean necesarias por cada categoría.

Por último, como tenemos cada una de las comidas del día almacenadas por separado, se crean los objetos de Planes Diarios para ir asignando estas comidas seleccionadas previamente. Una vez creado el array de planes diarios, se crea el objeto Plan Semanal,

se busca el ultimo id almacenado en esta colección para aumentar en uno el id, se asigna como dieta, la dieta regular ya que actualmente es la única que se trata, y por último para los planes diarios se asigna este array recientemente construido.

Se guarda el objeto en la base de datos, y como esta función se le ha pasado por parámetro el usuario al que se le está creando este plan, será necesario actualizar la propiedad correspondiente:

```
findOneAndUpdate(userId, {$set:{planComida: idPlan}}).
```

Se busca el usuario en la base de datos de usuarios por su id y establece su propiedad de plan de comida con el id del nuevo plan que acabamos de crear.

4.2.3 Front-end

En este apartado vamos a explicar toda la implementación del proyecto front-end. Como se ha mencionado anteriormente, este proyecto se ha realizado con Angular, y se puede dividir, como se ha explicado en apartados anteriores, en los componentes, modelos, servicios y guards.

4.2.3.1 Componentes

En este apartado se van a explicar los distintos componentes del proyecto, las distintas vistas de las distintas páginas o subpáginas.

4.2.3.1.1 Página principal



Figura 11. Página principal

Como se puede observar en la figura 11, esta página principal es sencilla, simplemente se incorpora, cómo se hará en todas, la cabecera con el logo de la aplicación y algunas funcionalidades, en este caso se podrá acceder al inicio de sesión y al registro.

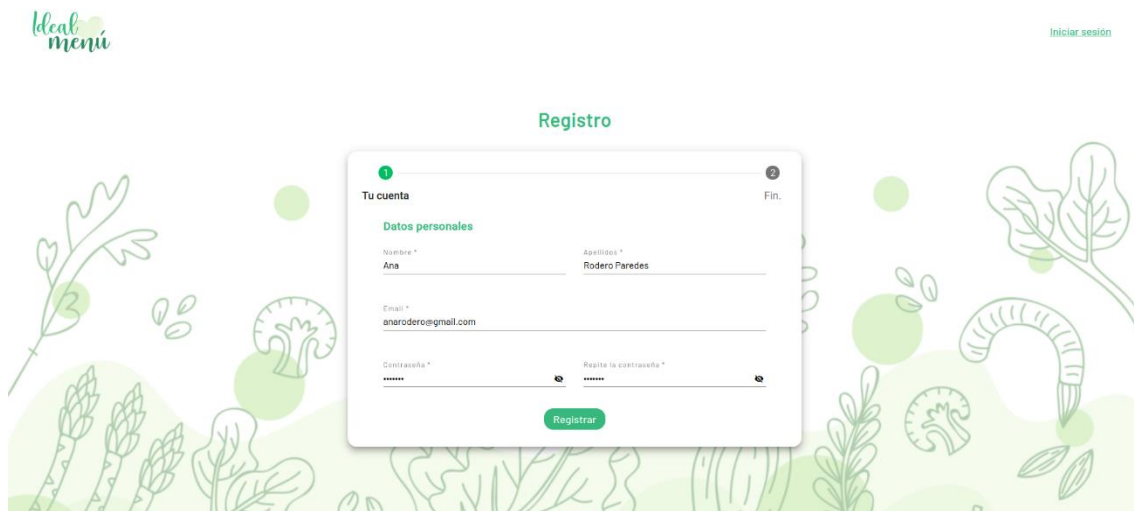
En la página se incluye un sencillo slogan para captar la atención del usuario con otro acceso al registro.

4.2.3.1.2 Página de registro

Figura 12. Página de registro

A esta ventana se accede para registrarse en la aplicación. esta ventana simplemente mostrará un formulario con los datos necesarios para el registro de un usuario. Estos datos son el nombre y apellidos, el correo electrónico, y la contraseña, la cual se pedirá dos veces por seguridad.

Como se puede observar en la figura 12, el botón de registro se encuentra deshabilitado. este botón no se habilitará hasta que todos los datos obligatorios sean rellenados.

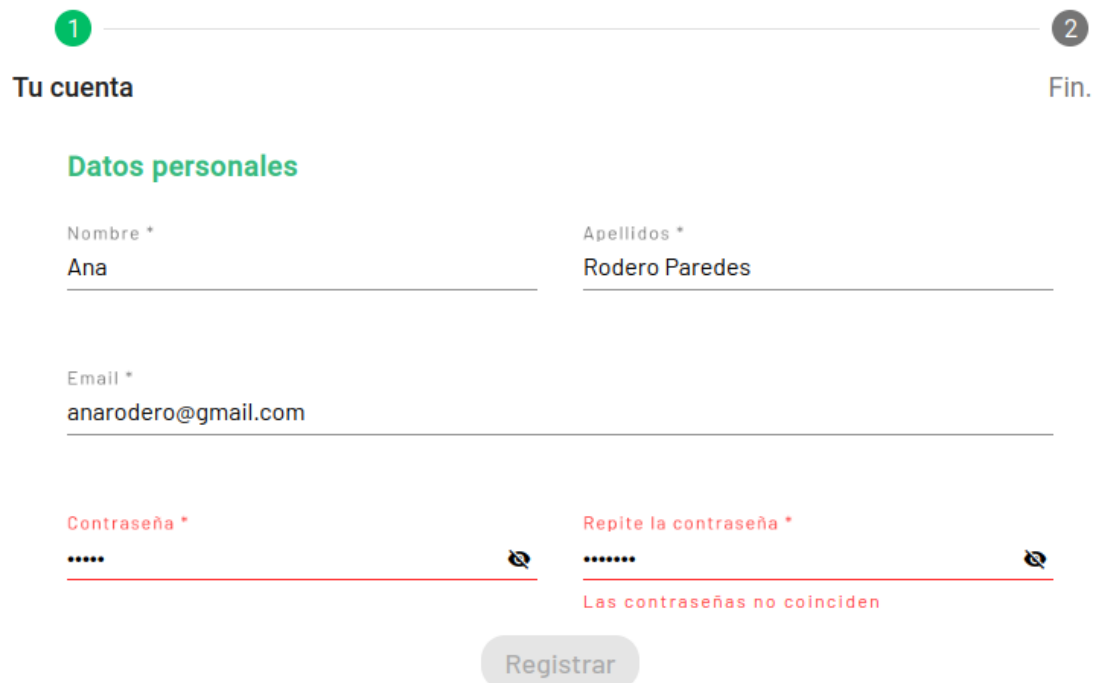


The screenshot shows the 'Registro' (Registration) page of the IdealMenú application. The page has a green and white theme with a background illustration of various vegetables like asparagus, mushrooms, and leafy greens. At the top left is the 'IdealMenú' logo, and at the top right is a link to 'Iniciar sesión' (Log in). The main heading is 'Registro'. Below it, a white form box contains the registration steps. Step 1, 'Tu cuenta' (Your account), is active and shows the 'Datos personales' (Personal data) section. The fields are filled with: 'Nombre *' (Name) as 'Ana', 'Apellidos *' (Surnames) as 'Rodero Paredes', 'Email *' as 'anarodero@gmail.com', and 'Contraseña *' (Password) and 'Repite la contraseña *' (Repeat password) both masked with dots. A green 'Registrar' button is at the bottom of the form. Step 2, 'Fin.' (End), is indicated by a small circle with the number 2.

Figura 13. Registro con datos introducidos

Como se puede ver en la figura 13, una vez se han rellenado todos los campos, el botón de registro se habilitará.

Este formulario lleva un control de ciertos errores. Si al escribir ambas contraseñas, estas no son iguales, se marcarán los campos como error indicando que las contraseñas no coinciden, como se observa en la figura 14.



This screenshot shows the same registration form as Figure 13, but with an error. The 'Contraseña *' (Password) field is marked with a red border and contains five dots. The 'Repite la contraseña *' (Repeat password) field is also marked with a red border and contains six dots. Below the repeat password field, a red error message reads 'Las contraseñas no coinciden' (Passwords do not match). The 'Registrar' button at the bottom is now greyed out and disabled. The progress indicator at the top shows step 1 as active and step 2 as completed.

Figura 14. Registro contraseñas no coinciden

Al registrar el usuario, si ya existe un usuario con ese correo electrónico, no se registrará entonces el usuario y se mostrará un error en el campo indicando que se email ya ha sido utilizado, como se puede observar en la figura 15.

The screenshot shows a registration form titled 'Tu cuenta' with a progress bar at the top. The progress bar has a green circle with the number '1' on the left and a green circle with a pencil icon on the right, labeled 'Fin.'. Below the title, the section 'Datos personales' is highlighted in green. The form contains several input fields: 'Nombre *' with the value 'Ana', 'Apellidos *' with the value 'Rodero Paredes', 'Email *' with the value 'ana@gmail.com', 'Contraseña *' with masked characters, and 'Repite la contraseña *' with masked characters. A red error message 'Este email ya esta registrado' is displayed below the email field. At the bottom, there is a grey button labeled 'Registrar'.

Figura 15. Registro email ya registrado

Si al registrar el usuario, todo va bien, el usuario será redirigido al paso dos del registro, cómo se puede ver en la figura 16. Esta vista simplemente durará brevemente, para indicarle al usuario que su registro se ha realizado de forma correcta y entonces será redirigido a la página de inicio de sesión para que entonces el usuario pueda loguearse con las credenciales que acaba de crearse.

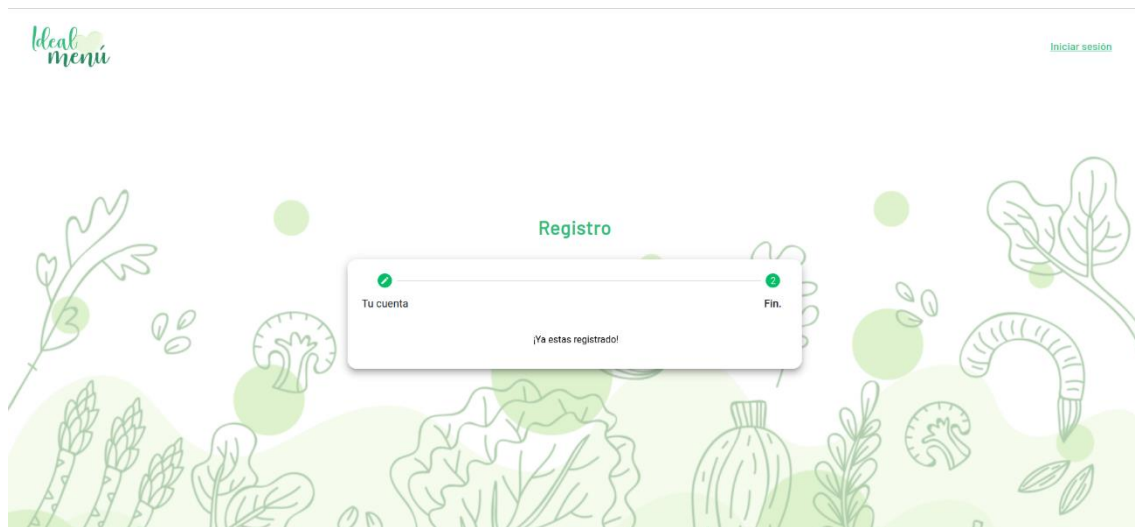


Figura 16. Registro exitoso

4.2.3.1.3 Página de inicio de sesión

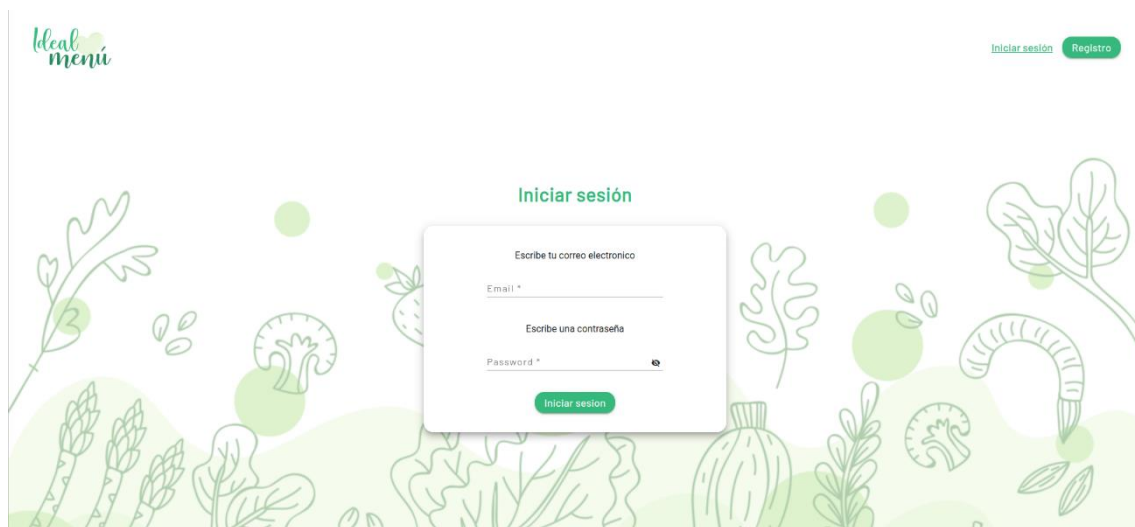
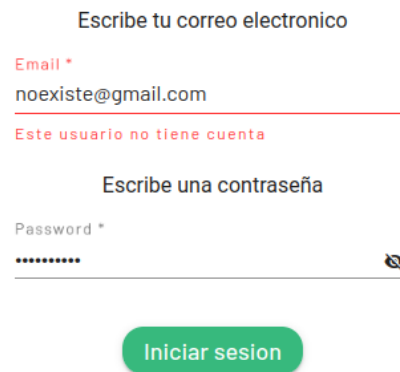


Figura 17. Página de inicio de sesión

Esta página muestra un simple formulario de correo electrónico y contraseña para que el usuario pueda iniciar sesión en la aplicación.

Esta página también lleva un control de errores. Si al iniciar sesión, se ha introducido un correo electrónico que no pertenece a ningún usuario previamente registrado, se muestra entonces un error indicando que ese usuario no existe, como se ve en la figura 18.

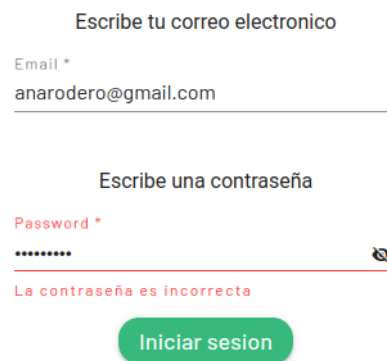


Formulario de inicio de sesión con los siguientes elementos:

- Encabezado: "Escribe tu correo electronico"
- Etiqueta: "Email *"
- Entrada de texto: "noexiste@gmail.com"
- Mensaje de error: "Este usuario no tiene cuenta" (en rojo)
- Encabezado: "Escribe una contraseña"
- Etiqueta: "Password *"
- Entrada de texto: "*****" (con icono de ojo para alternar visibilidad)
- Botón: "Iniciar sesion" (verde)

Figura 18. Email no tiene cuenta

También, si al iniciar sesión, la contraseña introducida no coincide con la registrada para el usuario con dicho correo electrónico, se mostrará entonces un error indicando que la contraseña es incorrecta, como se puede observar en la figura 19.



Formulario de inicio de sesión con los siguientes elementos:

- Encabezado: "Escribe tu correo electronico"
- Etiqueta: "Email *"
- Entrada de texto: "anarodero@gmail.com"
- Encabezado: "Escribe una contraseña"
- Etiqueta: "Password *"
- Entrada de texto: "*****" (con icono de ojo para alternar visibilidad)
- Mensaje de error: "La contraseña es incorrecta" (en rojo)
- Botón: "Iniciar sesion" (verde)

Figura 19. Contraseña incorrecta

Si las credenciales han sido introducidas de forma correcta, el usuario será redirigido a la página de inicio de la aplicación.

4.2.3.1.4 Página de inicio

Cuando el usuario introduce sus credenciales, será redirigido a esta página de inicio. Aquí se muestran las diferentes funcionalidades que el usuario se puede realizar. Se ha dividido en tres bloques, la base de datos para visualizar sus recetas y alimentos, el plan semanal, y las pirámides de alimentos de una dieta regular y de una dieta vegetariana o vegana. Todo esto se puede ver en la figura 20.

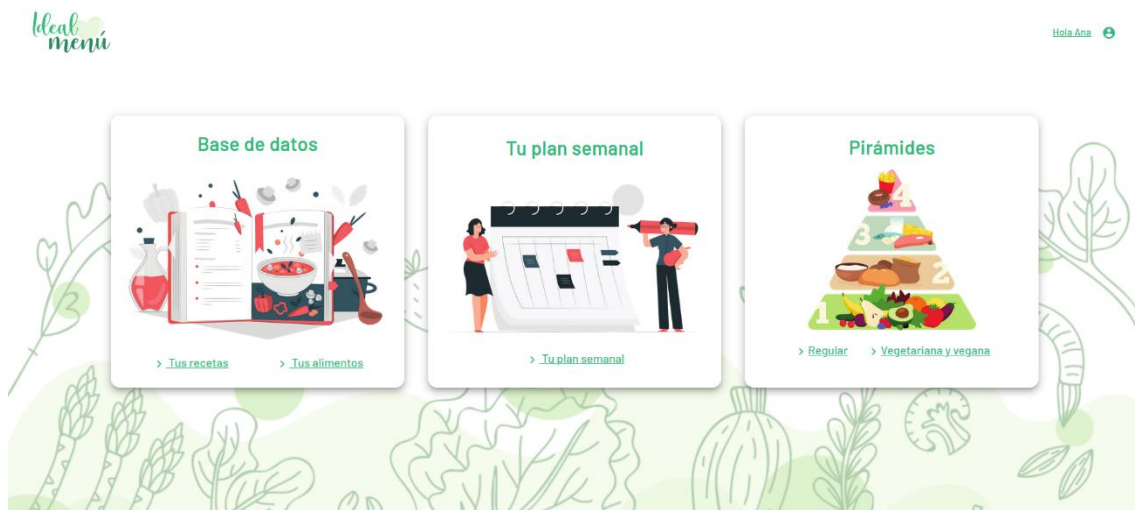


Figura 20. Página de inicio

4.2.3.1.5 Base de datos de recetas

En esta página el usuario podrá visualizar la base de datos de tanto las recetas simples para desayunos almuerzos y meriendas, como las completas para comidas y cenas.

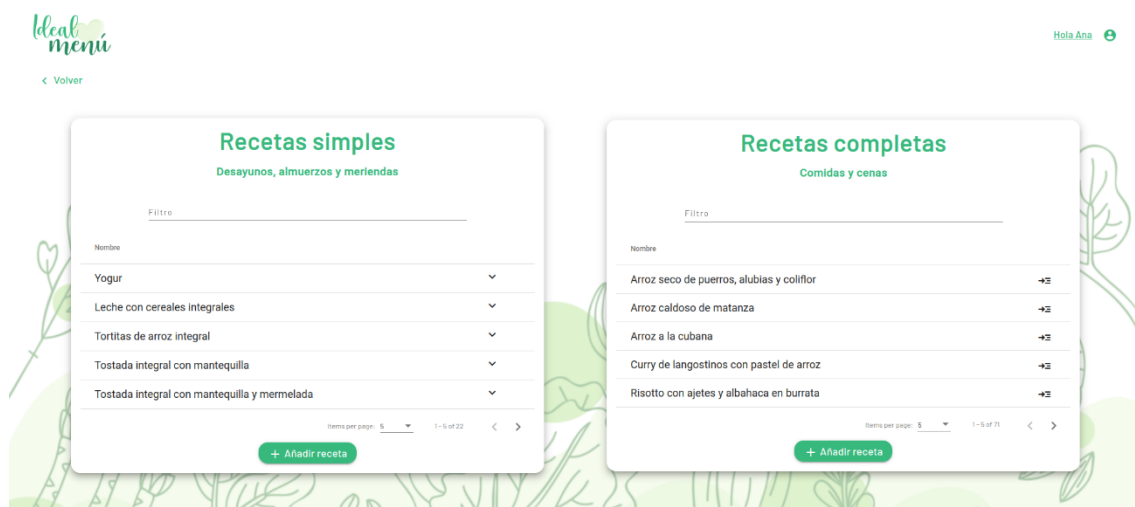


Figura 21. Base de datos de recetas simples y completas

Como se puede observar en la figura 21, estas tablas están paginadas, con la opción de modificar cuántos elementos se quieren visualizar por página, teniendo como opción 5 o 10. Además estas tablas podrán ser filtradas. Cómo se puede observar ambas tienen un campo llamado filtro, en el cual el usuario podrá filtrar el nombre de las recetas. Incluso también si el usuario pulsa sobre la cabecera de nombre, podrá ordenar de forma alfabética, tanto ascendente como descendente, las recetas.

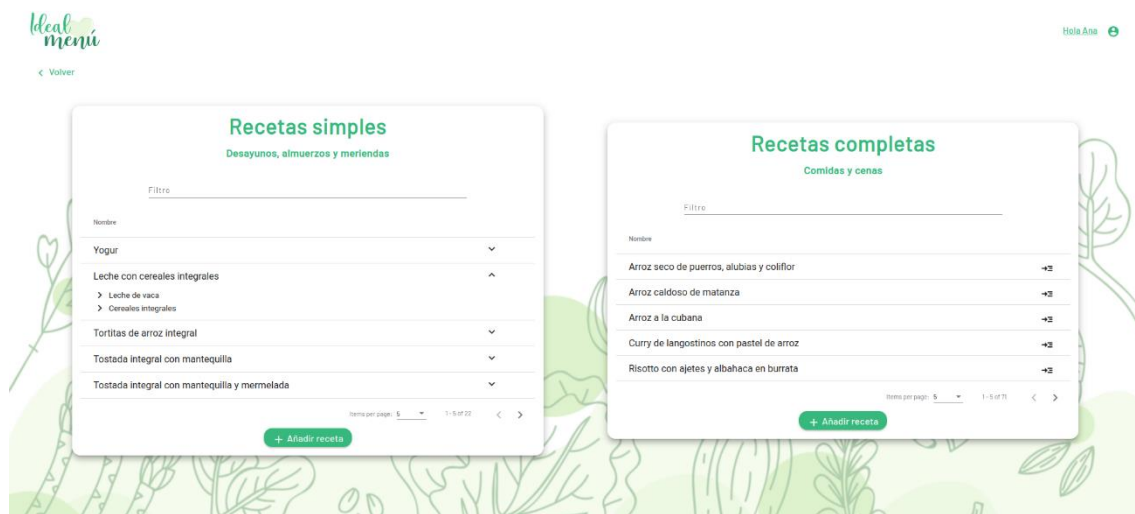


Figura 22. Despliegue receta simple

Como las recetas simples no tienen elaboración ni pasos, se puede observar en la figura 22, las filas de la tabla de las recetas simples serán desplegables mostrando los alimentos que conforman dicha receta.

En el caso de las recetas completas, al tener más propiedades y ser más complejas, el usuario podrá pulsar en el icono que aparece en cada una de las filas de esta tabla, esto le redirigirá a una página donde se explica con detalle dicha receta, esto se profundiza en otro apartado.

Ambas tablas tienen sus respectivos botones para añadir recetas, estos botones redirigirán al usuario a sus páginas para añadirlas, se mostrarán en los siguientes apartados.

4.2.3.1.6 Añadir receta simple

Esta página contiene un formulario para poder añadir una receta simple. Tiene su título, un listado de alimentos, y un seleccionable de categorías para elegir a qué categorías de alimentos pertenece esta receta simple.

IdealMenú

Hola Ana

< Volver

Añade una receta simple

Título *

Alimentos

> Añade un alimento...

Selecciona a que categorías pertenecen estos alimentos

Categoría *

Crear receta

Figura 23. Añadir receta simple

Inicialmente la página se visualiza como aparece en la figura 23. Para añadir alimentos a dicha receta el usuario tendrá que pulsar en ‘Añade un alimento...’. Este botón hace visible un cuadrante, el cual se puede observar en la figura 24, con un autocompletado dónde, conforme se va escribiendo en el campo, se va filtrando en la base de datos de alimentos. Una vez seleccionado un alimento, se habilitará el botón de añadir ingrediente que, al pulsarlo, el alimento será añadido al listado de alimentos que pertenecen a la receta, como se visualiza en la figura 24.

Título *

Alimentos

> frambuesas

Busca un alimento

Alimento *

Añadir ingrediente

Selecciona a que categorías pertenecen estos alimentos

Categoría *

Crear receta

Figura 24. Alimento por añadir y añadido

La receta debe tener todos los campos del formulario obligatorios rellenados, tanto el título, como al menos un alimento, y al menos una categoría seleccionada. Una vez rellenado esto el botón de crear receta se habilitará. Si se pulsa el botón para crear la receta, esta será añadida a la base de datos y el usuario será redirigido a la página de base de datos de recetas.

4.2.3.1.7 Añadir receta

Esta página contiene un formulario para poder añadir una receta. Tiene su título, la opción de seleccionar a qué dieta pertenece, su categoría, un listado de ingredientes, y un listado de pasos para la preparación de dicha receta.

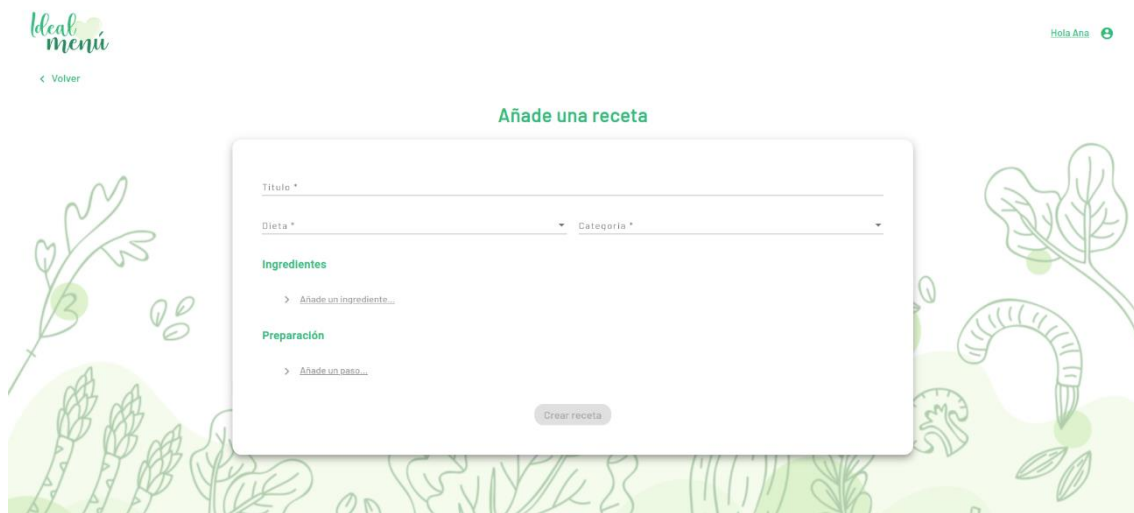


Figura 25. Añadir receta

Inicialmente la página se visualiza como aparece en la figura 25. Para añadir alimentos a dicha receta el usuario simplemente tendrá que pulsar en ‘Añade un ingrediente...’. Este botón hace visible un cuadrante, el cual se puede observar en la figura 26, con tres campos: una cantidad, un seleccionable de medidas y un autocompletado donde, conforme se va escribiendo en el campo, se va filtrando en la base de datos de alimentos. Una vez rellenado los tres campos, se habilitará el botón de añadir ingrediente que, al pulsarlo, el alimento será añadido al listado de alimentos que pertenecen a la receta, como se visualiza en la figura 26.

Título *

Dieta * Categoría *

Ingredientes

> 150 gr de harina de avena

Busca un alimento

Cantidad * 20 Medida * ml Alimento * leche de cabra

Añadir ingrediente

Preparación

1) Primer Paso...

Escribe el siguiente paso *

Segundo paso...

Añadir paso

Crear receta

Figura 26. Ingrediente añadido y por añadir.

De la misma forma sucede con la preparación, si el usuario pulsa en ‘Añade un paso...’, se habilitará un cuadrante (figura 26), en el cual aparece un cuadro de texto donde se podrá escribir el paso de la preparación que se quiere añadir. Una vez haya algo escrito, el botón de añadir pasos se habilitará, y al pulsarlo el paso será añadido como se puede visualizar en la figura 26.

La receta debe tener todos los campos del formulario obligatorios rellenados, tanto el título, como la dieta y categorías seleccionadas, y al menos un alimento y un paso. Una vez rellenado esto el botón de crear receta se habilitará. Si se pulsa el botón para crear la receta, esta será añadida a la base de datos y el usuario será redirigido a la página de base de datos de recetas.

4.2.3.1.8 Información sobre una receta

El usuario es redirigido a esta página cuando se encuentra situado en la página de la base de datos de recetas, y en la tabla de recetas completas, pulsa sobre algún icono de alguna de las filas de las recetas. Esto se puede observar en la figura 27.



Figura 27. Información receta

En esta ventana se visualiza el título la dieta y la categoría a la que pertenece esta receta. también aparece en el listado de ingredientes y de pasos para la preparación de dicha receta.

Se puede ver también un botón “volver” el cual redirigirá al usuario de vuelta a la base de datos de recetas.

4.2.3.1.9 Base de datos de alimentos

En esta ventana el usuario podrá visualizar datos de alimentos almacenados hasta el momento.

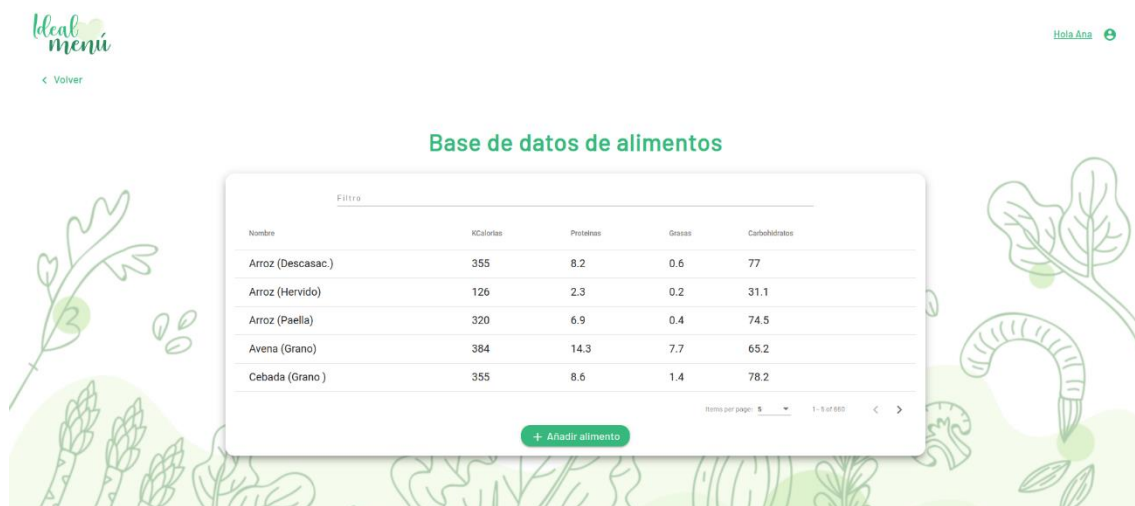
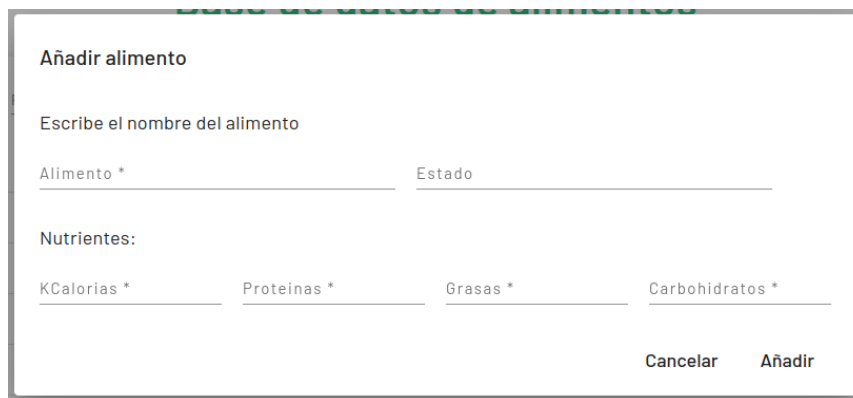


Figura 28. Base de datos de alimentos

Como se puede ver en la figura 28, esta tabla, como la de las recetas, cuenta con un paginado ajustable de 5 o 10 páginas, con un filtrado por nombre, y en este caso por una ordenación en cada una de las columnas tanto ascendente como descendente. Estas columnas de las que se hablan son el nombre del alimento y sus nutrientes, siendo estos, las kilocalorías, proteínas, grasas y carbohidratos que dicho alimento contenga. Estas medidas son por cada 100 gramos de alimento.

También se tiene la opción de añadir algún alimento. Habrá que pulsar en el botón de añadir alimento, el cual hará que se muestre el siguiente modal.



The modal titled 'Añadir alimento' contains the following elements:

- A label 'Escribe el nombre del alimento' above a text input field.
- Two input fields: 'Alimento *' and 'Estado'.
- A label 'Nutrientes:' above four input fields: 'KCalorías *', 'Proteínas *', 'Grasas *', and 'Carbohidratos *'.
- Two buttons at the bottom right: 'Cancelar' and 'Añadir'.

Figura 29. Añadir alimento

Como se puede observar en la figura 29, este formulario nos pide que escribamos el nombre del alimento, sus nutrientes y su estado, este último campo no es obligatorio.

En todo momento podemos cancelar esta operación y no añadir ningún alimento, o al pulsar añadir este alimento será añadido a la base de datos de alimentos. El modal se cerrará y este alimento se habrá añadido a la tabla.

4.2.3.1.10 Información pirámides

En este apartado podemos incluir las tres ventanas que nos muestran la información de las tres posibles pirámides. Aunque, como ya se ha comentado previamente, esta aplicación por el momento solo utiliza la pirámide regular, el usuario también tiene la opción de visualizar cómo son las pirámides para una dieta vegana y vegetariana.



Figura 30. Pirámide regular

Como se observa en la figura 30, esta ventana muestra la información de la pirámide, organizando la distribución de los alimentos que se deben tomar diaria y semanalmente. Lo mismo se hará con las pirámides para la dieta vegana y vegetariana, mostrando para cada una sus respectivas categorías.

4.2.3.1.11 Plan semanal

Esta ventana muestra el usuario el plan que se le ha creado de forma personalizada.

Tu plan semanal

LUNES	MARTES	MIÉRCOLES	JUEVES	VIERNES	SABADO	DOMINGO
Desayuno Galletas de chocolate	Desayuno Tostada integral con huevo	Desayuno Tortitas de arroz integral	Desayuno Galletas integrales	Desayuno Leche con cereales integrales	Desayuno Tostada integral con mantequilla	Desayuno Zumo de naranja
Almuerzo Tostada integral con tomate	Almuerzo Batido de avena con fruta	Almuerzo Tostada integral con tomate	Almuerzo Sandwich de pavo con queso	Almuerzo Sandwich de pavo con tomate	Almuerzo Yogur con fruta	Almuerzo Tostada integral con mantequilla y mermelada
Comida Crema fría de espinacas, berros y brócoli Salmon con miel y limón - Guarnición: Pasta	Comida Ensalada templada de legumbres Pato estofado - Guarnición: Pasta	Comida Ensalada de legumbres Garbanzos casados	Comida Ensalada de berenjenas escaldadas caseras Espaguetis con verduras y tomate	Comida Ensalada de aguacate y tomate Pollo mantequilla con coco - Guarnición: Patata	Comida Ensalada de berenjena marroquí (Zaalouk) Tortilla de bacalao y pimiento verde	Comida Crema de aguacate Risotto de langostinos con curry y leche de coco
Merienda Yogur con fruta	Merienda Yogur con fruta	Merienda Batido de avena con fruta	Merienda Batido de avena con fruta	Merienda Yogur con fruta	Merienda Zumo de naranja	Merienda Batido de avena con fruta
Cena Crema fría de espinacas, berros y brócoli	Cena Salmon con miel y limón	Cena Tortilla de bacalao y pimiento verde	Cena Albóndigas vegetarianas de tofu con salsa de tomate	Cena Ensalada de aguacate y tomate	Cena Gazpacho de fresones	Cena Ensalada de berenjena marroquí (Zaalouk)

[Cambiar de plan](#)

Figura 31. Plan semanal

Como se puede observar en la figura 31, en este plan se podrá visualizar una tabla, organizada por cada uno de los días y por cada una de las comidas.

En cada uno de los días de la semana, el usuario podrá pulsar en el icono que aparece al lado del día, esto le redirigirá a una pantalla con la información de solo ese día en concreto, explicada en el siguiente apartado.

En cualquier momento el usuario también tendrá la opción de pulsar el botón de cambiar de plan, si ya ha finalizado la semana o al usuario no le gusta el plan que se le ha asignado, siempre podrá cambiárselo. Al pulsar aquí se le autogenerará otro plan distinto.

4.2.3.1.12 Plan diario

Esta página simplemente muestra el usuario un día en concreto para poder visualizarlo de forma más limpia y sencilla. En esta ventana simplemente se visualizará un solo día, con cada una de sus comidas.

4.2.3.1.13 Editar perfil

En todo momento desde la cabecera el usuario tendrá la opción de editar su perfil.

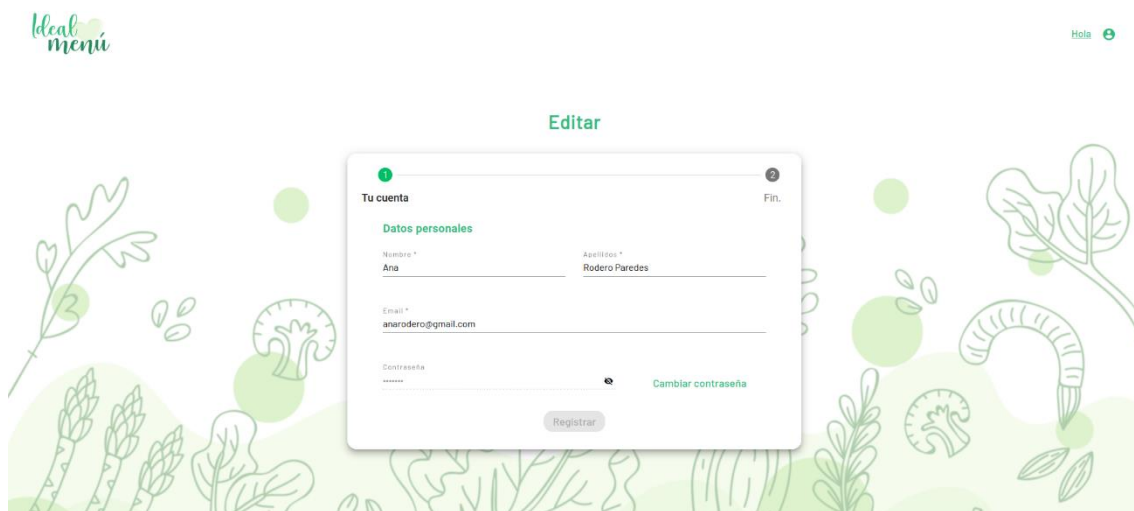


Figura 32. Editar perfil

En esta página el usuario verá un formulario parecido al del registro, pero con sus datos ya rellenados, podrá modificar estos datos como quiera, e incluso podrá cambiar la contraseña. De primeras el campo de contraseña está habilitado, aunque tendrá la opción de visualizarlo ya que inicialmente no es visible.

Si el usuario quiere cambiar la contraseña tendrá que pulsar el botón de ‘Cambiar contraseña’, esto hará visible una nueva parte, mostrada en la figura 33.

1 Tu cuenta 2 Fin.

Datos personales

Nombre *
Ana

Apellidos *
Rodero Paredes

Email *
anarodero@gmail.com

Cambio de contraseña ✕ ⬇

Escribe tu contraseña actual

Contraseña *

Escribe tu nueva contraseña

Nueva contraseña * Repite la nueva contraseña *

Registrar

Figura 33. Cambio de contraseña

El usuario necesitará escribir de nuevo su contraseña actual, y escribir la nueva contraseña dos veces. Para guardar estos cambios deberá pulsar al icono de guardar que aparece al lado de ‘Cambio de contraseña’. Este cambio se guardará sí la contraseña actual coincide con la que el usuario tiene en este momento, y si las nuevas contraseñas coinciden entre sí. Estos campos tienen la misma validación que en el registro, por lo que los fallos serán visualizados de la misma forma.

Si las condiciones se cumplen, se realizará el cambio de contraseña. Si por otro lado el usuario no quiere realizar este cambio de contraseña, siempre podrá cancelar la operación pulsando en el icono de la cruz al lado de ‘Cambio de contraseña’.

Mientras se esté realizando un cambio de contraseña, el botón para guardar la edición de perfil quedará deshabilitado, cuando no se esté realizando este cambio y todos los campos confirmen sus validaciones, el usuario podrá entonces guardar los cambios de su nuevo perfil

4.2.3.2 Modelos

En este apartado del proyecto se definen los distintos modelos de objetos que se van a tratar, estos son los mismos descritos en la parte del back-end, en el caso de Angular, se definen interfaces que representan estos objetos, con la misma estructura.

4.2.3.3 Servicios

Los servicios sirven como conexión entre el proyecto del front-end y del back-end. Aquí se almacenarán las distintas peticiones realizadas al back-end, es decir, las llamadas mediante API REST.

Se han creado tantos servicios como objetos distintos controla la aplicación (alimento, pirámide, plan de comida, receta y usuario). Cada uno de estos gestiona sus llamadas propias. Conforme cada componente necesita información de la base de datos o alguna funcionalidad establecida en el back-end, el componente llama al servicio del objeto que está tratando, el cual lanzará la petición al back-end.

Estos servicios tendrán tantas llamadas como posibles rutas de petición establecidas en el back-end.

4.2.3.4 Guards

Un guard es, como bien dice su nombre, un guardia que gestiona la entrada de los distintos componentes, es decir, cuando permitirle al usuario visualizar o no una página.

En el caso de este proyecto, se ha creado un guard global, que controla cuando el usuario se encuentra dentro de la aplicación o no. Ya que cuando el usuario no ha introducido sus credenciales todavía, solo debe tener acceso a la página de inicio, a la página del registro y a la de iniciar sesión. Una vez que el usuario ha introducido sus credenciales, podrá acceder a todas menos las previamente dichas. De esta forma se controla que mientras un usuario no esté dentro de la aplicación, este no pueda visualizar paginas de dentro de la aplicación, permitiéndole ver solo la “fachada” de la aplicación.

5 Conclusiones y vías futuras

5.1 Conclusión

En este TFG se ha diseñado y desarrollado una aplicación web para ayudar a sus usuarios a mantener una vida sana. Con ella se ha contribuido a mejorar la salud de quienes la usen, proporcionando la ayuda de organizar unos planes de comidas creados a partir de las recomendaciones de los expertos, gracias a un algoritmo diseñado para ello.

Tras haber finalizado la realización de este proyecto, se puede decir que se han cumplido los objetivos establecidos, algunos de estos objetivos tal vez se abrían querido finalizar de otra manera, añadiendo tal vez más funcionalidades, pero considero que han sido resueltos, aun así, de forma correcta.

Muchas de las pautas y tiempos que me había establecido al comenzar, han aumentado considerablemente, en concreto, la creación de la base de datos. Le he tenido que dedicar mucho tiempo a este apartado, ya que quería crear de cero mi propia base de datos, y para ello he tenido que buscar y crear todo esto. Cabe destacar que se ha partido de una base de datos de alimentos base, aunque le he tenido que añadir a esta, bastantes alimentos. También me ayudó el algoritmo que me cree para la lectura de mis recetas, pero aún así tenía que relacionar manualmente cada uno de los ingredientes con el alimento, por lo que esto ha sido un proceso bastante trabajoso y largo.

Por otro lado, para el algoritmo principal de la aplicación, he tenido que buscar la información y la forma de hacerlo de la mejor manera posible siguiendo las recomendaciones de profesionales, esta parte ha sido también bastante trabajosa ya que había que tener muchas cosas en cuenta, y me ha llevado más tiempo del que había previsto al comienzo.

En conclusión y pese algunos baches e impedimentos, he podido terminar el proyecto planteado, con un resultado del que me encuentro bastante orgullosa.

5.2 Vías futuras

Mirando al futuro, hay algunas cosas que se podrían añadir a este proyecto para llevarlo un poco más allá.

Sobre la base de datos, la idea sería ampliarla a grandes rasgos. Al haber creado la base de datos en el contexto del TFG, actualmente no se encuentra muy poblada, por lo que se podría ampliar añadiendo muchas más recetas, ya que conforme más recetas haya más variedad tendrá el algoritmo para la creación de los planes.

Por otro lado, sobre el algoritmo principal, ampliar e implementar la idea inicial de que no solo se tuviera en cuenta una dieta regular, sino que tanto si eres vegetariano como vegano pudieras crearte también tus planes. Las recetas que se crean con la aplicación y las almacenadas en la base de datos tienen ya de base un poco de esta idea, ya que cada receta está relacionada con el tipo de dieta en el que se puede tomar, por lo que sería más sencillo incluir estos algoritmos de creación de dietas vegetarianas y veganas.

Por último, podría pasarse esta aplicación web a una app, dependiendo de la opción de los usuarios, que tras haber usado durante un tiempo la aplicación web, se les podría hacer un encuesta para ver si estarían interesados en la propuesta.

6 Bibliografía

- Alianza Team. (2019). Tips de nutrición para mejorar tu calidad de vida. Obtenido de <https://www.alianzateam.com/tips-de-nutricion-para-mejorar-calidad-de-vida/>
- BBC New Mundo. (2018). 3 razones por las que Firefox puede ser mejor que Google Chrome. Obtenido de <https://www.bbc.com/mundo/noticias-44563040>
- DKV Equipo. (2021). Por qué es importante llevar una alimentación saludable. *DKV quiero cuidarme*. Obtenido de <https://quierocuidarme.dkv.es/alimentacion/por-que-es-importante-llevar-una-alimentacion-saludable>
- FAO. (2014). Aprende cómo una buena alimentación puede mejorar tu salud. *Objetivos de desarrollo sostenible*. Obtenido de <https://www.fao.org/zhc/detail-events/es/c/214216/>
- Jaramillo-Antillón, J. (2001). Evolución de la medicina: pasado, presente y futuro. *Acta Médica Costarricense*, 43(3), 105-113. Obtenido de http://www.scielo.sa.cr/scielo.php?script=sci_arttext&pid=S0001-60022001000300003&lng=en&tlng=es
- OMS. (2003). *Dieta, Nutrición y prevención de enfermedades crónicas*. Ginebra. Obtenido de http://apps.who.int/iris/bitstream/handle/10665/42755/WHO_TRS_916_spa.pdf;jsessionid=A51D62ED6AD4BEE832A861C88E03CB29?sequence=1
- QuiromSalud. (2018). Los riesgos de consumir comida basura o fast food. *Tu canal de salud*. Obtenido de <https://www.tucanaldesalud.es/es/tusaludaldia/articulos/riesgos-consumir-comida-basura-fast-food>
- SENC. (2018). Guía de la alimentación saludable para atención primaria y colectivos ciudadanos. Obtenido de <https://www.nutricioncomunitaria.org/es/noticia/guia-alimentacion-saludable-ap>
- Tápanes, Y. (2022). Los 6 mejores framework frontend. *Saasradar*. Obtenido de <https://saasradar.net/mejores-framework-frontend/>