Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA
MENCIÓN EN INGENIERÍA DE SOFTWARE

Dielthy: Una web para dietas especiales

Alumno: Javier Alaguero Martín

Tutor: Yania Crespo González-Carvajal

A mi familia, que siempre me apoyó

Agradecimientos

A mi familia, que siempre me ha apoyado y me ha animado a lograr todo lo que me propongo.

A mis compañeros de carrera, que me ayudaron a crecer personal y académicamente, sin los que ahora no estaría donde estoy y sin los que este viaje no habría sido lo mismo.

A mi tutora Yania, por implicarse día a día con sus alumnos y ayudarlos en todo lo necesario.

A mi novia Laura, cuyo apoyo diario me anima a seguir adelante siempre.

Gracias

Resumen

El objetivo de este proyecto es desarrollar una web de consulta de los diferentes aditivos alimentarios existentes. Está enfocado especialmente a personas veganas, vegetarianas o que sufren algún tipo de intolerancia común, como la intolerancia al gluten o a la lactosa. Además, se pretende que la web se convierta en una red social para que los usuarios intercambien recetas, ideas, opiniones... Adicionalmente, para facilitar la interacción de un usuario en movilidad, se desarrolla un bot de Telegram integrado con el sistema que permite buscar la información de un aditivo de forma más rápida.
El proyecto se ha desarrollado empleando el framework Vue.js, HTML5, CSS y el lenguaje JavaScript siguiendo una metodología ágil (Scrum).

Abstract

The purpose of this project is to develop a website to query food additives. It is specially focused on vegan, vegetarian people or those who suffer some common type of food intolerance, like lactose or gluten intolerance. Furthermore, it is intended that the website turns into a social network in which users can share recipes, ideas, opinions... Aditionally, to make easier mobile user interaction, a Telegram bot integrated with the system is developed, which allow for searching additive information faster.
The project has been developed using Vue.js framework, HTML5, CSS and JavaScript language following an agile methodology (Scrum).

Índice general

Lista de Figuras

Lista de Tablas

Capítulo 1

Introducción, objetivos y motivación

1.1. Introducción

El número de personas en el mundo que deciden seguir una dieta vegetariana o vegana ha crecido exponencialmente [34]. Asimismo, se ha visto un aumento considerable en las personas que sufren algún tipo de intolerancia a ciertos alimentos, llegando a alcanzar el 25 % de la población. Entre ellas, las más comunes a día de hoy son el gluten y la lactosa [35]. Muchas aplicaciones y webs han optado por mostrar qué alimentos se pueden o se deben comer si se opta por esta clase de alimentación, ya sea porque su cuerpo así lo determina, por ética, salud, o cualquier motivo de otra índole. En la Sección 1.5.1 se explicarán los objetivos de este proyecto y de qué forma ayudará a estos colectivos.

1.2. Motivación

La idea de este proyecto surgió tras observar de forma cercana cómo personas vegetarianas y veganas sufrían un desconocimiento de los distintos alimentos que podían o no ingerir, en especial debido a la cantidad de aditivos alimentarios que contienen, identificados únicamente por un código numérico, de los cuales es imposible recordar toda su información a la hora de realizar una compra rutinaria.

Existen otras alternativas y buscadores, si bien es cierto que ninguno de ellos hace hincapié en el intercambio social entre personas en la misma situación. En la siguiente sección se presenta una revisión de trabajos similares al que se pretende desarrollar con este proyecto.

1.3. Alternativas

Antes de abordar este proyecto se realizó una búsqueda de las distintas alternativas existentes similares a la idea que se pretende desarrollar. En los próximos apartados se describirá cada una de ellas haciendo énfasis en las ventajas y desventajas competitivas halladas.

1.3.1. cocinarvegano.com

Cocinarvegano [3] constituye la fuente principal de inspiración de este proyecto.

Ventajas:

- Contiene una lista bastante completa de los distintos aditivos alimentarios y si son aptos o no.
- Incluye un recetario muy extenso.
- Recetas con alimentos de todo tipo y para todos los gustos.
- Posibilidad de búsqueda por categorías.
- Enlaces directos a su cuenta en distintas redes sociales.
- Posibilidad de buscar recetas sin gluten o sin lactosa.

Desventajas:

- Difícil acceso a la lista de aditivos alimentarios.
- Interfaz poco amigable.
- Demasiada publicidad.
- Muy reducida capacidad del usuario para subir sus propias recetas o comentar sus ideas en la web.

1.3.2. e-aditivos.com

E-aditivos [4] es la web con la lista más completa de aditivos alimentarios de entre las alternativas analizadas.

Ventajas:

- Listado muy completo de aditivos alimentarios.
- Interfaz atractiva y bastante intuitiva.
- Cuenta con aplicación para Android.
- Cuenta con un bot para Telegram.

Desventajas:

- Poca información al respecto sobre cada aditivo.
- Muy reducida posibilidad del usuario de aportar sus propias ideas y opiniones.
- El bot de Telegram no funciona y no muestra ningún tipo de información para saber cómo utilizarlo.
- Lista únicamente centrada en personas veganas y vegetarianas y en la toxicidad general de los propios aditivos.
- La aplicación únicamente existe para Android, dejando de lado el alto porcentaje de usuarios de iOS.

1.3.3. aditivos-alimentarios.com

Aditivos-alimentarios [5] es la web más cercana a lo que se pretende llevar a cabo en este proyecto.

Ventajas:

- Mucha información sobre cada aditivo.
- Interfaz intuitiva y bien diseñada.
- Posibilidad de los usuarios para enviar sus comentarios sobre cada aditivo.
- La publicidad no resulta muy molesta para el seguimiento y uso de la web.

Desventajas:

- No se centra en ningún tipo de dieta especial, tan sólo provee la información del aditivo en cuestión.
- Por la publicidad presente y la existencia de una tienda se puede extraer como objetivo principal el lucro personal.
- El origen de cada aditivo debe ser extraído de la descripción por el propio usuario, no se muestra de forma gráfica o en un apartado propio.

1.4. Nicho de mercado y usuarios objetivo

Tras haber recabado información acerca de los principales competidores existentes actualmente, se puede deducir que el principal nicho de mercado se encuentra en las personas veganas y vegetarianas, cuyo número se ha visto muy incrementado en los últimos años. Sin embargo, con el fin de acercar esta herramienta a otras personas con tipos de dietas diferentes a éstas, se ha decidido expandir el alcance de la web para incluir a aquellos que sufren intolerancia a la lactosa y el gluten.

A pesar de que el proyecto se encuentra bastante centrado en todos estos colectivos, cualquiera sería capaz de acceder a la web y utilizarla. No se circunscribe su uso a personas cuya alimentación se encuentre restringida por los factores anteriormente comentados. Es por ello que los usuarios objetivo de la web constituyen un abanico amplísimo.

Se pretende dotar al aplicativo de una interfaz sencilla que permita la utilización de sus funcionalidades a todo tipo de usuarios. Quienes no mantengan una fluida relación con la tecnología, especialmente personas de edad más avanzada, dispondrán de la herramienta web, mientras que, en un intento por un mayor acercamiento a los grupos de edad más jóvenes, se provee de un bot en Telegram, aplicación de mensajería instantánea muy utilizada.

1.5. Objetivos del proyecto

1.5.1. Objetivos de desarrollo

El principal objetivo en el que se centra el desarrollo de este proyecto es proveer de una herramienta de consulta de aditivos alimentarios, en primera instancia a personas veganas y vegetarianas, pero accesible y extensible a

cualquier persona que tan sólo desee información. En vista de que el factor más importante y la mayor fuerza de estos colectivos con dietas restringidas es el componente social, se pretende dotar a la web de un marcado carácter de red interpersonal que favorezca un medio adecuado de comunicación entre unos y otros para compartir todas sus inquietudes al respecto y fomentar el aprendizaje entre ellos.

Por todo lo anterior, se ha decidido integrar un foro (únicamente para usuarios identificados en la web) junto a la principal funcionalidad, que es la consulta de los aditivos alimentarios. Como objetivo secundario, con menor prioridad, se encontraría crear un bot sencillo para la red social Telegram de manera que quien posea esta aplicación sea capaz de acceder a toda esta información de manera más sencilla. Telegram ha experimentado un gran auge en cuanto a descargas y popularización se refiere en los últimos años [69] y especialmente en los últimos meses, durante el confinamiento ocasionado por la COVID-19.

1.5.2. Objetivos personales

Dado que este proyecto se enmarca en un contexto educativo, como Trabajo de Fin de Grado, también tiene unos objetivos de formación personal que se enumeran a continuación:

- Conocer en profundidad el framework Vue.js.
- Mejorar mi capacidad y destreza en el desarrollo web. *Front-end* empleando HTML, CSS y JavaScript, pues es la faceta a la que me gustaría, en principio, enfocar mi futura carrera profesional.
- Acercar la herramienta que se pretende desarrollar a personas cercanas a mi entorno, de cuyas necesidades se han extraído los objetivos de este proyecto.
- Entender y aprender la metodología y los principios de Scrum, que constituye un marco en el que se basan numerosos procesos de desarrollo actualmente y que puede decirse que es uno de los que alcanzan mayor popularidad hoy en día [70] [72].

1.6. Estructura de la memoria

El resto del documento está estructurado de la siguiente forma:

- En el Capítulo 2 se describe la planificación inicial del proyecto, realizada en el marco de desarrollo Scrum, abordando su adaptación a este caso concreto.
- En el Capítulo 3 se describen todas las tecnologías, lenguajes y herramientas empleadas para el desarrollo y la implementación.
- A continuación, en el Capítulo 4 se realizará una estimación de costes y un plan de posibles riesgos que pudieran acaecer durante el desarrollo del proyecto.
- En el Capítulo 5 se explicará el avance del proyecto. Esta documentación se desarrollará a lo largo de la realización del mismo reflejando los eventos Scrum conocidos como *sprint planning*, *sprint review* y *sprint retrospective*.

- Los Capítulos 6, 7 y 8 estarán dedicados al diseño, implementación y pruebas, respectivamente.
- Por último, en el Capítulo 9 se finaliza esta memoria con unas breves conclusiones, así como posibles futuras mejoras que podrían ser incluidas.
- Se añaden algunos anexos de interés como manuales de usuario y de despliegue, así como el detalle de los archivos que se incluyen con la entrega de este Trabajo de Fin de Grado.

Capítulo 2

Desarrollo y planificación del proyecto

2.1. Scrum

2.1.1. Principios y definición de Scrum

Scrum [6] es un marco de trabajo, como se muestra en la Figura 2.2, un proceso de desarrollo ágil, mediante el cual se realiza un desarrollo incremental del proyecto en cuestión. Es un proceso de gestión que reduce la complejidad del producto, apoyándose en gran medida en la fuerza del trabajo en equipo. Se sustenta en tres pilares fundamentales:

- **Transparencia**. Los aspectos más significativos tienen que ser visibles para todos aquellos interesados en el resultado, definiendo para ello todo el proceso siguiendo un estándar común que favorezca el entendimiento entre todas las partes involucradas.
- **Inspección**. Los involucrados en el marco de Scrum deben inspeccionar frecuentemente el progreso para detectar variaciones indeseadas.
- **Adaptación**. Habiendo detectado variaciones en el producto, debe ser posible adaptar el proceso de desarrollo, siendo, por tanto, ajustado de forma adecuada.

2.1.2. Equipo Scrum (*Scrum Team*)

Los equipos Scrum (Figura 2.1) son autoorganizados y multifuncionales, permitiendo así mejorar la flexibilidad, la creatividad y la productividad. Poseen todas las competencias necesarias para llevar a cabo el trabajo sin depender de personas ajenas al propio equipo. Los componentes del equipo Scrum son:

***Product Owner*:** Es el máximo responsable de optimizar el valor del producto y el trabajo del equipo de desarrollo. Es también el responsable de modificar el *Product Backlog*, artefacto que se explicará posteriormente con más detalle, en la Sección 2.1.4. De esta forma, se podría resumir su función en mantener actualizada la lista de tareas que el equipo debería desarrollar a lo largo del proceso.

Equipo de Desarrollo (*Development Team*): Consta de los profesionales que se encargan de desarrollar y entregar cada incremento (ver definición de incremento en la Sección 2.1.4). Cada uno de estos últimos debe

aportar una funcionalidad solicitada probada y completa. La organización será la encargada de conformar el equipo, dotado de suficiente independencia como para autoorganizarse y optimizar, así, el proceso de creación de los incrementos (ver Sección 2.1.4).

***Scrum Master*:** Constituye la cabeza visible del Equipo de Desarrollo, asegurándose de que el equipo trabaja acorde a las reglas y principios de Scrum. Es el líder del equipo, quien dirige las interacciones con personas externas al mismo.

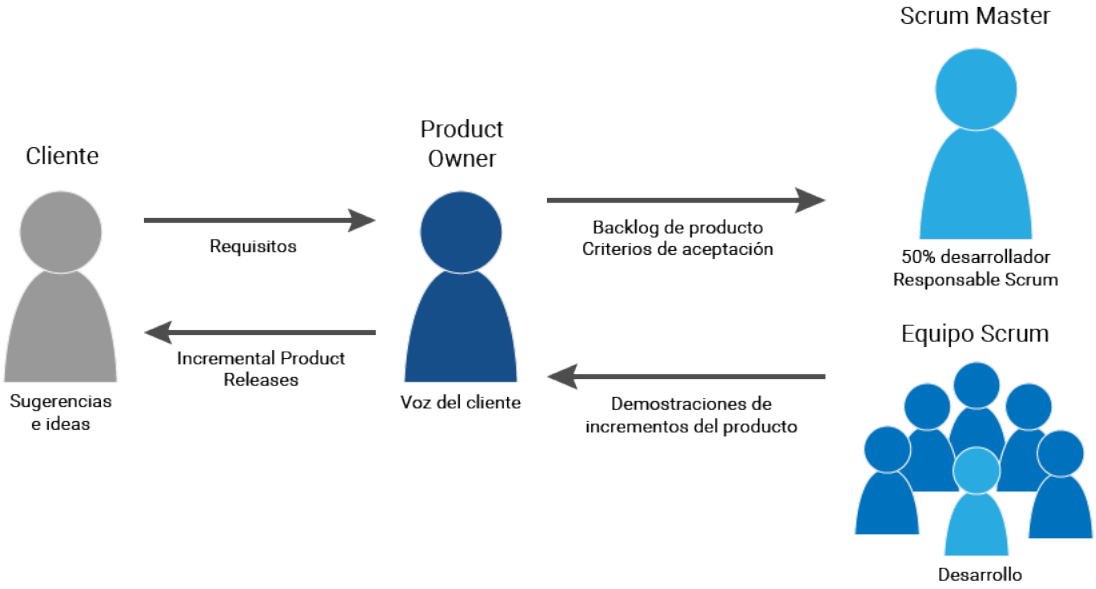

Figura 2.1: Roles del Equipo Scrum. Tomada de [36]

2.1.3. Eventos Scrum

El marco de Scrum está dividido en múltiples eventos:

Sprint: Constituye el evento fundamental del marco Scrum. Es un bloque de tiempo, con una duración entre 1 y 4 semanas, en la cual se realiza un incremento (explicado en detalle en la Sección 2.1.4) utilizable y potencialmente desplegable. Lo ideal es mantener una duración fija a lo largo del proceso de desarrollo. Cada sprint comienza inmediatamente después de la finalización del anterior. Mantiene unos requisitos y objetivos claros y estáticos a lo largo de su duración.

***Sprint Planning*:** Con anterioridad al comienzo del sprint se realiza una planificación del mismo, en el que participan todos los miembros del equipo Scrum. En él se fijan los objetivos del sprint y los desarrollos que se realizarán en el incremento en cuestión. También se asigna una carga de trabajo estimado a cada tarea. Esta estimación se denomina punto de historia. Deberá marcarse, al inicio, a cuántas horas de trabajo equivale. Cada tarea, de esta forma, tendrá una cantidad determinada de puntos de historia, lo que aportará un idea inicial de las horas de trabajo que conllevará cada una. El equipo de desarrollo completo participa en el *Sprint Planning*, fijando en él, de forma consensuada, los puntos de historia para cada una de las tareas que se abordarán en el sprint. Existen diversas escalas y formas de asignación de puntos de historia a tareas (lineal, serie de Fibonacci...).

***Sprint Goal*:** Es una meta que se establece en el Sprint Planning, desglosada mediante las historias de usuario del backlog (ver Sección 2.1.4) que pertenecen al sprint. El equipo de Desarrollo mantiene en mente este objetivo, implementando de la forma más adecuada la funcionalidad para llegar a satisfacerlo.

***Daily Scrum*:** . Es una reunión de unos 15 minutos en la que el equipo de Desarrollo pone en común las actividades desarrolladas durante las 24 horas anteriores y fija las que se realizarán en las siguientes. El objetivo es evaluar el progreso del equipo y sincronizar a todos sus miembros, alineándolos en la consecución del Sprint Goal.

***Sprint Review*:** . A la conclusión de cada sprint se lleva a cabo una revisión para inspeccionar el incremento creado a lo largo del mismo y modificar y adaptar, si es necesario, el backlog (artefacto explicado en la Sección 2.1.4). Se realiza un actualización del estado de las funcionalidades requeridas y los requisitos solicitados. Participan en él tanto el *Product Owner* como el Equipo de Desarrollo, así como el *Scrum Master*. El objetivo es facilitar la retroalimentación entre todas las partes y fomentar la colaboración del Equipo Scrum completo.

***Sprint Retrospective*:** . Constituye una oportunidad para el Equipo Scrum de evaluar esfuerzos y problemas y abordar soluciones o mejoras que puedan optimizar el trabajo en el desarrollo de próximos sprints.

Sprint 0

Se trata de un sprint especial, realizado al comienzo del proyecto, en el cual se realizan tareas de estudio y preparación de la arquitectura, planificación inicial de tareas y versión básica del *backlog* (artefacto descrito en detalle en la Sección 2.1.4) y, en definitiva, todos los preparativos previos a la implementación. El uso de esto genera una gran controversia, si bien se ha ido extendiendo. Entre sus detractores muchos no lo recomiendan, mientras que otros sólo consideran que es un nombre genérico mal empleado que se le da a la agrupación de todas las tareas de planificación previas al primer sprint.

Sprint de release

Siguiendo la metodología ágil Scrum, cada iteración debe concluir con un incremento potencialmente entregable. Sin embargo, puede pretenderse dedicar un único sprint encargado exclusivamente de que el equipo complete toda la preparación para desplegar el sistema en producción. Esto se realizaría al finalizar un número fijo de sprints considerados suficiente como para materializar dicha entrega. Al igual que el sprint 0, posee numerosos detractores que consideran esta actividad una mala práctica poco ajustada al marco Scrum y que, por tanto, debe evitarse.

2.1.4. Artefactos Scrum

En el contexto del marco Scrum existen varios artefactos que generan valor en diversas formas. Están diseñados explícitamente para favorecer la transparencia de los puntos clave para todas las partes a lo largo de todo el proceso de desarrollo. Entre ellos se incluyen:

***Product Backlog*:** Es una lista ordenada de requisitos del producto, denominados historias de usuario. Es importante destacar que esta lista se encuentra ordenada por prioridad. En un primer momento, se realiza una detección y definición previa de los mismos, pudiendo variar a lo largo de todo el proceso de desarrollo. Por ello, el *Product Backlog* es cambiante, sujeto a todo tipo de variaciones. En él se incluyen todas las mejoras, correcciones, evoluciones y funcionalidades de las que se debe dotar al producto. El responsable de actualizar y refinar esta lista es el *Product Owner* pero en su adecuación trabaja, además, el Equipo de Desarrollo, siendo un proceso continuo y variable. Sus elementos se examinan y se modifican en función de los requisitos

que se pretendan acometer o abordar, pudiendo surgir otros nuevos o modificaciones sobre los ya existentes en cualquier momento.

***Sprint Backlog*:** Está constituido por la lista de requisitos e historias de usuario del *Product Backlog* que serán abordados a lo largo del sprint actual. Es estático, no pudiendo ser modificado mientras el sprint se encuentra en curso, a fin de mantener una adecuada cohesión y coordinación entre los objetivos a implementar y los miembros del equipo, favoreciendo así la optimización del trabajo. Hace visible la lista de tareas necesarias para alcanzar el *Sprint Goal*. Contiene el suficiente detalle como para poder observar el avance realizado en el *Daily Scrum*. Pertenece únicamente al Equipo de Desarrollo.

Incremento: Es la suma de todos los elementos completados del *Product Backlog* hasta la fecha. Cada incremento debe estar en condiciones de ser utilizado.

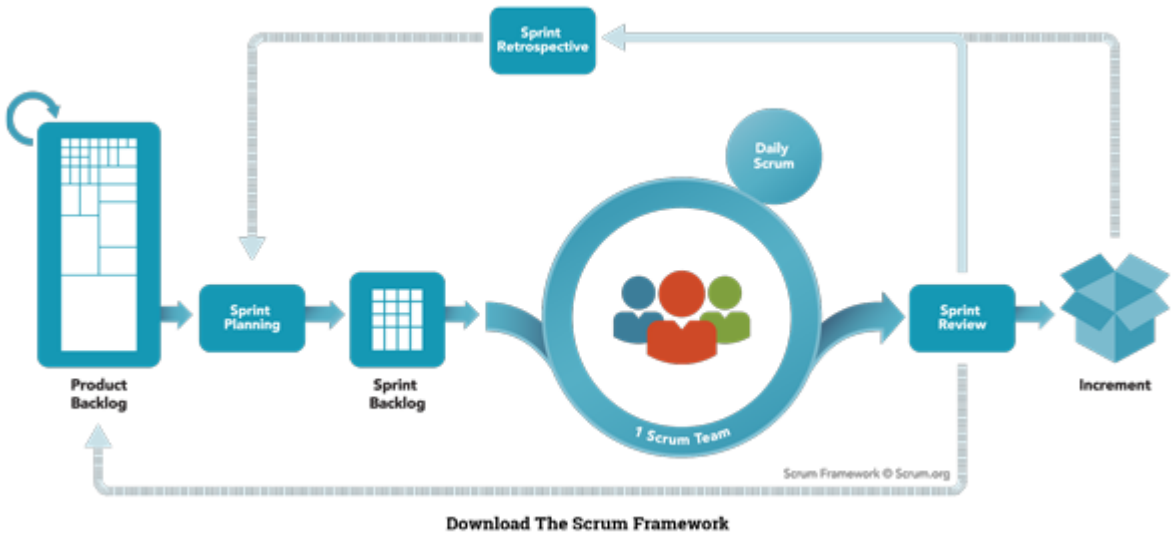

Figura 2.2: Eventos y artefactos del marco Scrum. Tomada de [37]

2.2. Adaptación de Scrum a este proyecto

El marco general de Scrum debe ser adaptado a todos los contextos donde se aplique. De forma particular en este contexto se requiere una importante adaptación debido especialmente a las características del personal. Siendo un proyecto que se realiza por un alumno como trabajo de fin de grado y contando con la tutora del mismo.

El alumno se encargará de ser a la vez el Equipo de Desarrollo y el *Product Owner*. Por su parte, la tutora desempeñará las funciones de *Scrum Master* y agente interesado. Por tanto, el *Product Backlog* quedará a cargo del alumno mientras que la tutora se encargará de guiarle en la correcta aplicación de la metodología Scrum y aportar asesoramiento en otro tipo de aspectos.

La duración de los sprints será de 2 semanas, finalizando cada uno de ellos los martes, en los que se realizará el *Sprint Review*, la *Sprint Retrospective* y el *Sprint Planning* del siguiente. Además, cada martes se realizará también una reunión semanal de seguimiento del proyecto.

El mantenimiento del *Product Backlog* se realizará empleando Trello y el seguimiento de las issues se realizará utilizando GitLab *issue tracker*. En él se mostrará tanto el *Product Backlog* como el *Sprint Backlog*. La forma de gestionar GitLab *issue tracker* y Trello será explicado en las Secciones 3.1 y 3.7, respectivamente.

En el *Sprint Planning* se estima una carga de trabajo por tarea, medida en puntos de historia. A cada historia de usuario se le asigna una cantidad de puntos de historia. Estos conceptos están relacionados con una medida de la productividad utilizada en entornos de desarrollo *agile* como es la velocidad (*velocity*). La velocidad se define como la cantidad de puntos de historia que puede hacer el equipo de desarrollo en cada sprint.

Para decidir cuántas historias de usuario y tareas entran en cada sprint se fija una velocidad inicial, estimada según la experiencia propia previa del equipo de desarrollo (el alumno en este caso). Esta velocidad inicial se modificará en el sprint 3, tras haber realizado alguna iteración que ya permita establecer una medida más fiable. Dicha velocidad inicial tomará un valor de, aproximadamente, 6 puntos de historia. Se ha establecido que 1 punto de historia equivale, aproximadamente, a 3 horas de trabajo. Cada tarea se puntuará de forma lineal en base al tiempo que el desarrollador cree que llevará su realización, en una escala entre 0 y 10, siendo 0 el mínimo y 10 el máximo. Teniendo en cuenta que cada sprint consta de unas 30 horas se puede deducir que se incluirá un total de 10 puntos de historias por cada uno. Sin embargo, se dejará un margen, sin abordar nunca el máximo de puntos de historia posibles por sprint, puesto que se debe reservar tiempo para la realización de la presente memoria de Trabajo de Fin de Grado.

Merece la pena destacar, por otra parte, que se realizará un sprint 0. Teniendo en cuenta el tiempo del que dispone el alumno para realizar el proyecto, los primeros meses serán dedicados exclusivamente a realizar la planificación y documentación iniciales, plantear la arquitectura y estructura del proyecto, poner en marcha todas las herramientas necesarias y dejar preparada por completo el entorno de desarrollo de la aplicación. De este modo se pretende que los sprints, cuya duración total abarcará alrededor de unos 4 meses, sean prácticamente dedicados sólo a la implementación y las tareas paralelas de documentación de esta memoria. Entre los meses de octubre y enero se desarrollará este sprint 0, cuyas dedicaciones serán incluidas en los costes calculados con posterioridad, pero no en la planificación de sprints. Esta actividad permitirá al alumno profundizar en su documentación e información al respecto, teniendo claros todos los aspectos relevantes para la correcta realización del proyecto en cuestión.

2.2.1. Planificación inicial de los sprints

Habiendo consultado la guía docente correspondiente al Trabajo Fin de Grado del Grado en Ingeniería Informática de la Universidad de Valladolid [2], se ha podido observar que la cantidad de horas correspondiente al desarrollo del mismo es de 300 horas. Dada la disponibilidad del alumno para su realización, se emplearán sprints de 30 horas de trabajo a lo largo de 2 semanas. De esta forma, la carga de trabajo resultará ser de 15 horas semanales. Se ha realizado una primera división de las tareas en función del tiempo, de lo que se ha deducido que se necesitarán al menos 9 sprints, dejando un margen de 1 sprint para corrección de errores, refactorización del código y finalización de todos los detalles pendientes del presente documento. La completa planificación inicial puede observarse en la Tabla 2.1.

2.2.2. Descripción de historias de usuario

La especificación de cada una de las historias de usuario del proyecto, se detalla en la Tabla 2.2. Se trata de una sencilla descripción de posibles historias compuestas o *epics* [71]. En el Capítulo 5 se detallarán y desglosarán las mismas.

Una vez vistas las historias de usuario que se abordarán a lo largo del proyecto y la planificación inicial que

Sprint	Comienzo	Finalización	Observaciones
Sprint 1	11/02/2020	25/02/2020	
Sprint 2	25/02/2020	10/03/2020	
Sprint 3	10/03/2020	24/03/2020	
Sprint 4	24/03/2020	07/04/2020	
Sprint 5	07/04/2020	21/04/2020	
Sprint 6	21/04/2020	05/05/2020	
Sprint 7	05/05/2020	19/05/2020	
Sprint 8	19/05/2020	02/06/2020	
Sprint 9	02/06/2020	16/06/2020	
Sprint 10	16/06/2020	30/06/2020	Susceptible de ser eliminado. Sprint de refuerzo

Tabla 2.1: Planificación inicial de sprints

se seguirá, se debe realizar una previsión de riesgos y una estimación de costes, que se detallarán a lo largo del Capítulo 4.

2.2.3. Modelo conceptual del dominio

Tras el análisis de las historias de usuario y las necesidades que conllevará su implementación se elabora un diagrama conceptual del dominio, que muestra las entidades necesarias para cubrir las funcionalidades requeridas. Este modelo puede observarse en la Figura 2.3.

2.2.4. Tareas realizadas

Una descripción más detallada de las tareas realizadas se abordará en el Capítulo 5.

Número	Título	Descripción
1	Consulta de aditivo	Como usuario quiero consultar la información de un aditivo para ver si se adecúa a mi tipo de dieta.
2	Registro	Como usuario quiero registrarme en la web para acceder a todas sus funcionalidades.
3	Inicio de sesión	Como usuario quiero iniciar sesión para poder añadir comentarios en el foro y modificar mi perfil.
4	Consulta de perfil	Como usuario quiero consultar y editar mi perfil para verificar que mi información es correcta y actualizarla si es necesario.
5	Comentarios en foro	Como usuario quiero comentar en un tema abierto para compartir mis opiniones o dudas con el resto de la comunidad.
6	Eliminación de usuarios	Como administrador quiero eliminar a los usuarios para evitar que personas ofensivas inunden el foro.
7	Eliminación de comentarios	Como administrador quiero eliminar comentarios del foro para mantenerlo bajo control.
8	Visualización del foro	Como usuario quiero visualizar el foro para estar al tanto de los últimos mensajes de los usuarios.
9	Abrir tema	Como usuario quiero abrir un tema para compartir mis opiniones con otros.
10	Denuncias de comentarios	Como usuario quiero denunciar un comentario del foro para que se proceda a tomar las medidas oportunas.
11	Atender denuncias	Como administrador quiero atender las denuncias de los comentarios de los usuarios para moderar adecuadamente el foro.
12	Bot de Telegram	Como usuario quiero consultar los aditivos alimentarios desde un bot de la aplicación de mensajería Telegram para acceder a esa información de forma más sencilla y rápida.
13	Internacionalización de textos de la aplicación	Como usuario quiero que los textos que aparezcan en la web se muestren en más de un idioma para poder adaptarlo a aquel que más me convenga.
14	Eliminación de comentario propio	Como usuario quiero eliminar un comentario que haya escrito en el foro.
15	Desestimación de denuncias de un comentario	Como administrador quiero desestimar las denuncias sobre un comentario en el foro.

Tabla 2.2: Detalle de las historias de usuario

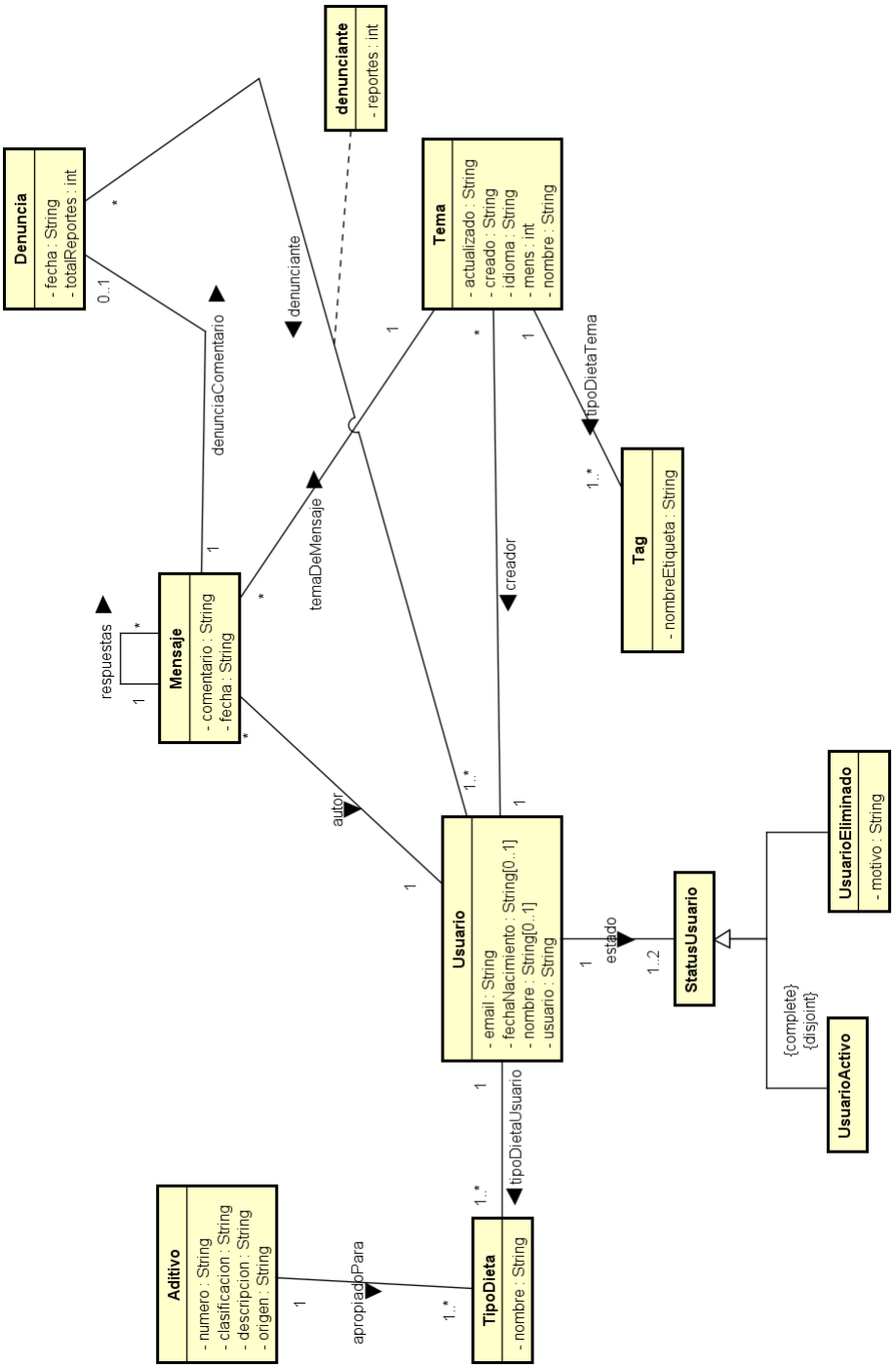

Figura 2.3: Modelo conceptual del dominio

Capítulo 3

Tecnologías utilizadas

3.1. Git

Git [7] es una herramienta gratuita de control de versiones de código abierto. Por control de versiones se entiende la gestión de los cambios realizados a lo largo del proceso de desarrollo de un proyecto software o la modificación de su configuración. Git se encuentra organizado en repositorios, cada uno de los cuales almacena los archivos que constituyen un determinado proyecto. Permite dividirlo en ramas. Cada una de ellas emerge desde otra y constituye una copia exacta de todos los archivos y contenidos que en ese momento existieran en todos los puntos del proyecto. Git es un sistema de control de versiones distribuido y permite coordinar repositorios locales y remotos. De esta forma se permite mantener un contenido estático de manera remota a la par que se van realizando los cambios de código pertinentes y se realizan las pruebas correspondientes de manera local. Una vez finalizado el trabajo se pueden de nuevo unificar las ramas, de modo que se combinen los nuevos cambios con la versión anterior. A esto se le denomina *fusionar* y en términos de git se utiliza la acción *merge*.

Para realizar un seguimiento de los cambios realizados en el repositorio local es necesario añadir los archivos modificados a lo que se denomina el *aréa de stage*. Después se ejecuta un *commit*, mediante el cual es posible añadir un mensaje aclaratorio de los cambios realizados. Los commits realizados en local se sincronizan con el repositorio remoto realizando una subida de código a través de un *push*. Esto actualiza la nueva versión en el repositorio remoto, reflejando en éste el mismo histórico de commits realizado en el repositorio local. Una sencilla explicación de este proceso puede observarse en la Figura 3.1.

3.1.1. GitLab e integración continua

Como parte de las herramientas de las que nos provee la universidad de manera gratuita se encuentra GitLab. GitLab es un entorno que nos permite gestionar los proyectos de Git de forma eficaz mediante una interfaz de usuario muy completa y sencilla. La forma de trabajo mediante el empleo de Git es de muy diversas características, pudiendo cada equipo de desarrollo organizarse de la manera mejor y más eficiente que estime oportuna. No existe, por tanto, una forma única de trabajar con Git. Es por ello que no se seguirá un estándar en particular. Se dispondrá de una rama master, desde la que se subirán las entregas al finalizar cada sprint, y de una rama develop desde la que saldrán las distintas ramas por cada una de las tareas que se pretendan realizar, cada una de ellas con un nombre identificativo. Esta forma de trabajo es similar a la conocida como “rama por tarea”.

GitLab, además, provee de una serie de herramientas, entre las que se encuentra la integración continua (CI/CD). Esto permite que el proyecto sea compilado, ejecutado, desplegado, etc., de manera automática con cada una de

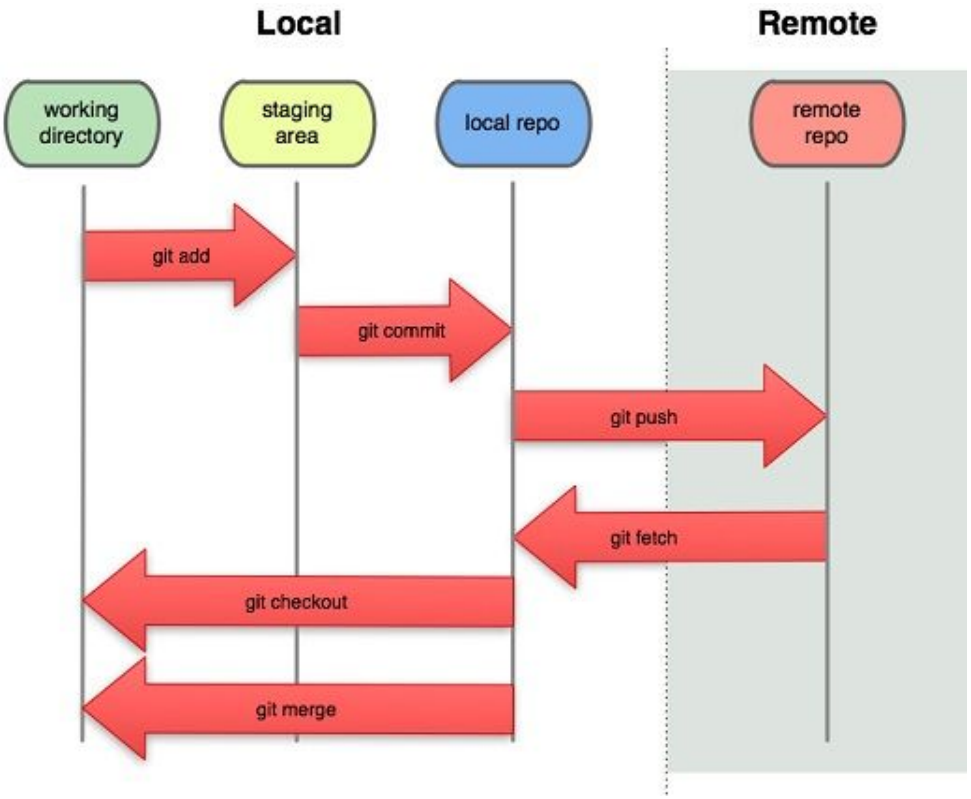

Figura 3.1: Funcionamiento de Git. Tomada de [50]

las subidas que se realicen al repositorio remoto. Para la correcta realización de los tests del aplicativo, en la rama develop se ejecutarán los tests unitarios, mientras que en la rama master, se ejecutarán los tests end-to-end y el despliegue de la aplicación en una plataforma de hosting, dado que, tratándose de una web, la aplicación (no así el código) deberá ser de dominio público y accesible. En el resto de ramas tan sólo se ejecutarán tests unitarios. La configuración de CI/CD en Gitlab [8] se basa en un archivo YAML, cuya estructura y creación resulta bastante sencilla, que se coloca en el directorio raíz del repositorio y que gestiona cada uno de los comandos que se ejecutan en las distintas fases o *stages* por las que va pasando la compilación, testeo y despliegue del proyecto en cuestión. La estructura final del archivo YAML del proyecto puede consultarse en la Sección 7.3.

3.1.2. Issues en Gitlab

La gestión *issues* en Gitlab [22] es una herramienta tremendamente útil en la gestión de las tareas a realizar a lo largo del desarrollo de un proyecto. Permite mantener una relación de los próximos cambios que se realizarán, asignar determinadas subidas al repositorio a una tarea, controlar el tiempo empleado en cada una de ellas, discutir determinados cambios entre los miembros del equipo... En este proyecto se emplearán issues para cada tarea que surja a lo largo de la implementación de las historias de usuario, que pueden consultarse en la Tabla 2.2. Esto permitirá un seguimiento mucho mayor por parte de la tutora de los avances que el alumno irá realizando con el paso del tiempo.

3.2. Vue.js

Vue.js [9] es un framework progresivo que permite crear interfaces de usuario de forma sencilla. Se emplea en el desarrollo *Front-end* de aplicaciones web de tipo Single Page Application (SPA) [10]. El desarrollo de este tipo de aplicaciones está caracterizado por la descomposición de la misma en múltiples vistas bajo una misma página, sin necesidad de recargar el navegador al cambiar entre ellas. Tan sólo existe un único punto de entrada a ellas: el fichero index.html, desde el cual se puede acceder a las diversas partes de las que consta la aplicación. La mayor ventaja es la experiencia de usuario que proporciona, con cargado rápido e interacción agradable con la página.

Vue.js es un framework que nos permite componentizar la parte frontal de una web. Constará, por tanto, de numerosos archivos con extensión .vue, que constituirán cada uno de los componentes, basados en la idea de Componentes de un Solo Archivo (*Single File Components*). Estos se dividen en tres fragmentos, encapsulados cada uno de ellos entre etiquetas estilo XML, y que, por tanto, tendrán tanto etiqueta de apertura como de cierre:

Template: Código HTML que crea los elementos visuales. Se encuentra entre etiquetas <template>.

Script: código JavaScript o TypeScript que permite gestionar toda la lógica de los elementos visuales y los datos que mostrarán estos. Se engloba entre etiquetas <script>. Este fragmento, a su vez, se puede dividir en numerosas partes:

a. *name*: Nombre del componente.

b. data. Función que almacena las variables del estado del componente. En el template se permitirá usar estas variables para mostrar datos.

c. *props*: Constituyen el conjunto de atributos a los que se puede dar valor cuando este componente es utilizado por otro.

d. *components*: Declaración de todos los componentes que utiliza el componente en cuestión. Esta parte contendrá una referencia a todos los componentes en su template en formato HTML, pudiendo pasar como atributos las props que declare cada uno de ellos.

e. *methods*: Conjunto de métodos del componente. Pueden ser utilizados por eventos de elementos del template o por la propia lógica interna del componente.

f. *computed*: Aúna el conjunto de variables que requieren de un cálculo previo en base a otras. Está muy relacionado con la reactividad, pues si una variable de las que depende cambia, se recalculará el resultado.

g. *watch*: Permite realizar acciones en base a los cambios de una *prop*, de un elemento de la parte *data* o de una variable de la parte *computed*. Pone de manifiesto la reactividad en Vue (Figura 3.2), permitiendo "observar" constantemente los cambios en el estado del componente y pudiendo implementar reacciones ante dichos cambios.

Style: Clases CSS y sus propiedades que permiten crear estilos para los elementos visuales del componente. Se colocan entre etiquetas <style>.

Cabe destacar que estas secciones pueden contener el código como tal o bien hacer referencia a ficheros externos que contendrán el código.

Haciendo uso de los elementos visuales declarados en el template, Vue genera un DOM virtual. La interacción con este DOM y/o los cambios que se produzcan en los datos y variables de los que hace uso hará que se actualice en consonancia. La propiedad que hace posible este comportamiento es a lo que se denomina reactividad. Puede observarse un esquema más aclaratorio en la Figura 3.2 y una explicación más profunda en la Sección 6.3.

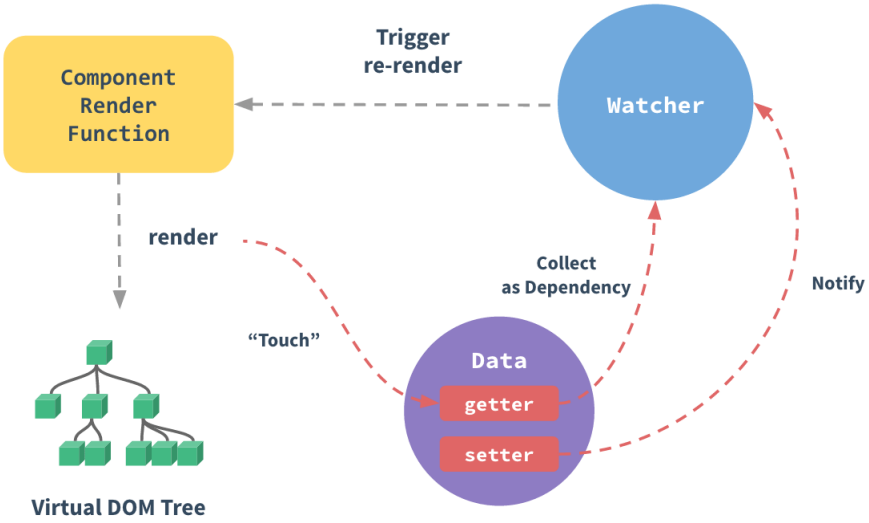

Figura 3.2: Reactividad en Vue. Tomada de [38]

3.2.1. Ciclo de vida

Cada uno de los componentes que conforman una aplicación Vue es una instancia [39]. Estos avanzan a lo largo de una serie de pasos desde su inicialización hasta su destrucción, en lo que se conoce como ciclo de vida, representado en la Figura 3.3. A la izquierda de la figura se pueden observar los distintos métodos y etapas por los que pasa la instancia de Vue, que pueden ser utilizados (e implementar lógica en ellos) en un componente en su sección script, explicada en la Sección 3.2.

3.3. NPM

Node Package Manager (NPM) [11] es un gestor de paquetes desarrollado en lenguaje JavaScript que permite añadir dependencias, librerías y módulos a las aplicaciones que estemos desarrollando. En este proyecto se añaden numerosas dependencias, pasando a continuación a explicar brevemente las de mayor trascendencia para el desarrollo del mismo y las que se han añadido adicionalmente a las que por defecto agrega Vue.

3.3.1. @vue/cli-plugin-unit-jest

El plugin para testing unitario en Vue [12] permite gestionar la ejecución y creación de los tests unitarios desarrollados para la aplicación haciendo uso de la herramienta Jest [13]. Se empleará para los tests unitarios del proyecto, explicados en la Sección 8.2. Es el gestor de testing unitario que se encuentra por defecto en Vue.js.

3.3.2. @vue/cli-plugin-vuex

Este plugin permite incorporar Vuex [14] como dependencia en el proyecto. Vuex [15] es una herramienta que permite gestionar el estado global de la aplicación desarrollada en Vue.js y la compartición de datos entre componentes desde muy diferentes puntos del flujo de la misma. Consta de 3 partes claramente diferenciadas

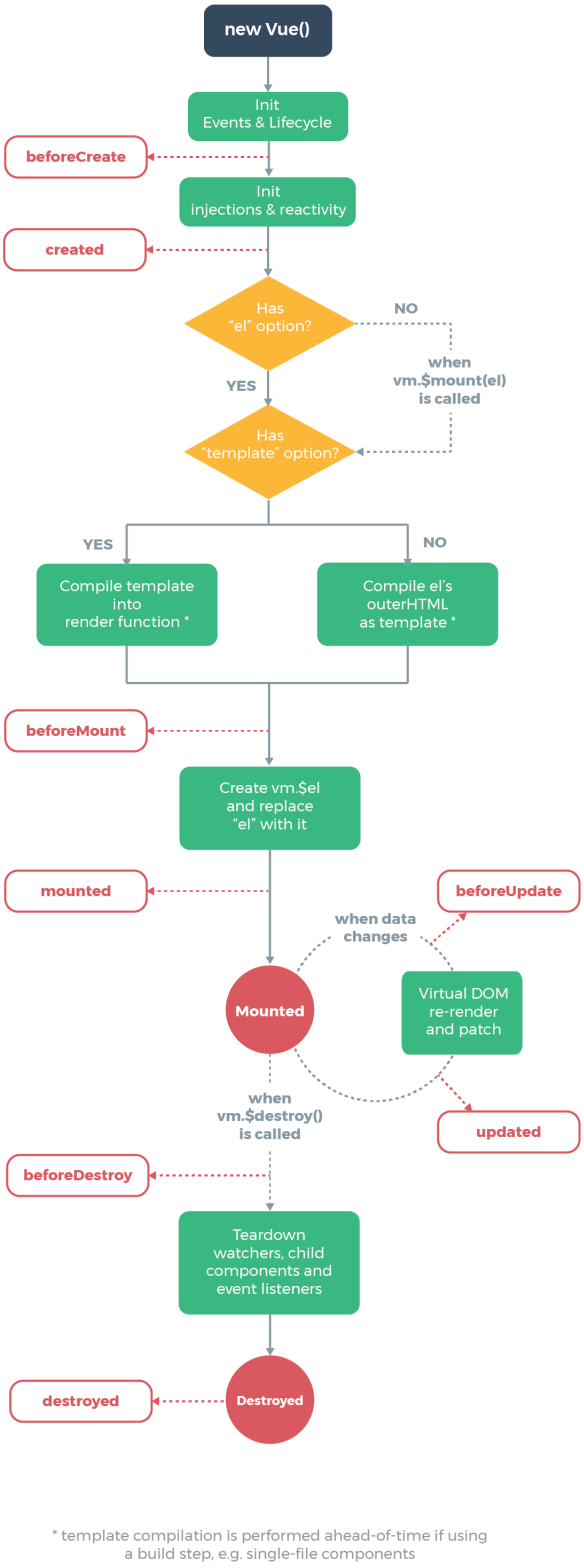

Figura 3.3: Ciclo de vida de una instancia Vue. Tomada de [39]

State: Almacena datos.

Actions: Actúan de intermediarias entre los componentes y el estado (*state*).

Mutations: Modifican el estado. Son llamadas desde las *actions*.

En la Figura 3.4 pueden observarse las relaciones entre los distintos elementos que conforman Vuex y su interacción con los componentes. Esta herramienta puede añadirse en la propia creación de la aplicación.

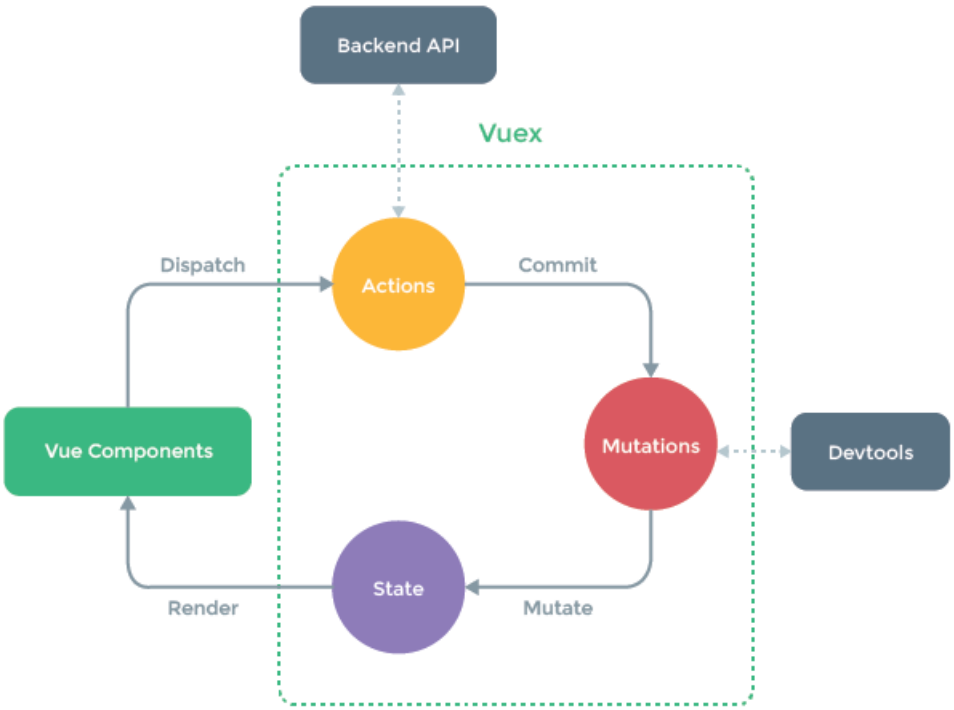

Figura 3.4: Flujo de Vuex. Tomada de [15]

3.3.3. @vue/cli-plugin-router

El plugin de router para Vue [16] facilita la creación y manejo de rutas en Vue. Vue Router [17] facilita el paso entre rutas y vistas a través de una aplicación SPA creada con Vue.js. Con una simple declaración de las rutas posibles y comandos sencillos en la lógica de la aplicación o elementos visuales que proporciona el propio framework es posible cambiar la vista que se muestra en la página. Esta herramienta puede añadirse en la propia creación de la aplicación.

3.3.4. @vue/cli-plugin-e2e-cypress

El plugin de testing automático que facilita por defecto Vue [18] emplea Cypress [19]. Cypress es una herramienta que permite realizar tests, tanto unitarios, como de integración, como end-to-end, de forma sencilla, rápida y amigable. Se asemeja, en cierto modo, a algunos webdrivers, si bien su arquitectura, simplicidad y herramientas de debugging lo diferencian de los demás y lo hacen una gran utilidad a incorporar. Se empleará para los test end-to-end que se explican en la Sección 8.3.

3.3.5. vuetify

Este paquete [20] permite añadir Vuetify al proyecto. Vuetify [21] es un framework de componentes que permite añadir diversos elementos visuales a las vistas de la aplicación. Esto facilita sobremanera la labor del desarrollador, ya que lo único que debe hacer es aunar dichos elementos para que se ajusten al diseño visual deseado y gestionar su cohesión. La lógica interna de los mismos corre a cargo de los desarrolladores de cada componente. Es la librería recomendada por Vue.js, si bien no se añade por defecto en la creación del proyecto.

3.3.6. @mdi/js

Este paquete [53] de Material Design Icons permite la importación y uso de los mismos a lo largo de la aplicación. La lista completa de iconos, de cuya lista se han extraído los utilizados en el desarrollo del proyecto, puede ser consultada en [54].

3.3.7. i18n

El paquete i18n [55] es un plugin que permite la internacionalización de los textos en diferentes idiomas de forma sencilla empleando notación de objetos JavaScript. Puede ser configurado en unos determinados idiomas de tal forma que, refiriéndose a cada texto con el mismo nombre de variable, es posible aportar traducciones de páginas web completas de forma automática.

3.3.8. express

El paquete express [62] es una infraestructura web sencilla que proporciona un conjunto de características para las aplicaciones creada con Node.js. Se emplea en la implementación de la función que controla el bot de Telegram, puesto que provee de herramientas para gestionar las peticiones HTTP.

3.4. Firebase

Firebase [25] es una plataforma digital creada por Google que permite agilizar y facilitar el desarrollo tanto de aplicaciones móviles como web. Gracias a esta herramienta se puede implementar una base de datos en tiempo real, funciones que traten los datos, análisis del rendimiento de las aplicaciones, envío de notificaciones o gestión de usuarios y autenticación, entre otros [26].

En el desarrollo de este proyecto se emplean, en particular, las siguientes utilidades:

- Hosting: almacenamiento de los ficheros que componen y despliegan una web.
- Firestore: base de datos NoSQL en tiempo real de Firebase.
- Authentication: gestión de usuarios y sesiones.
- Functions: funciones desplegadas en la nube que cumplen algún propósito específico. En este proyecto el bot de Telegram se desplegará haciendo uso de estas Cloud Functions de Firebase.

3.5. Astah

Entre las herramientas más utilizadas por el alumnado a lo largo del desarrollo del Grado en Ingeniería Informática se encuentra Astah. Astah [27] es un software que permite la creación de modelos de abstracción de los sistemas informáticos con el objetivo de comprender y analizar en profundidad su funcionamiento, relaciones entre sus elementos, errores que puedan surgir... Para ello se sirve del Lenguaje Unificado de Modelo (Unified Modeling Language). UML [28] es un lenguaje estándar creado para facilitar la comprensión de un sistema informático por parte de cualquier analista o arquitecto software con tan sólo visualizar los modelos desarrollados con el mismo, sin necesidad de revisar su implementación. En este proyecto se emplea en las Secciones 2.2.3, 6.4, 6.7 y 7.6.

3.6. InVision

InVision [29] es una web que permite crear bocetos y diseños sencillos para proyectos de diversas plataformas. Se utilizará en la Sección 6.5.1. Para ello se empleará la herramienta que provee la propia página web: InVision Studio.

3.7. Trello

Trello [33] es una herramienta que permite la creación, gestión y compartición de tableros en línea. En ellos se pueden crear listas. Éstas pueden contener tarjetas más específicas. Esto permite llevar un control del trabajo, una mejor coordinación de equipo y una adecuada gestión de las tareas a realizar. No sólo es aplicable en ámbitos de proyectos empresariales sino que puede ser utilizada a nivel personal, pudiendo desde coordinar una fiesta de cumpleaños hasta gestionar listas de tareas en casa.

En este proyecto se empleará Trello para llevar el control del backlog global, organizando el tablero en tantas listas como sprints haya. Esto permite a la tutora y al alumno mantenerse al tanto de cualquier cambio que se produzca.

3.8. Webscraper

Webscraper [41] es una herramienta online que permite realizar *web scraping* [42]. Esta técnica se basa en extraer grandes cantidad de información de páginas web de forma automatizada. Se emplean algoritmos de búsqueda y expresiones regulares para precisar la obtención de la información.

En este proyecto se ha empleado esta herramienta para generar una lista de aditivos alimentarios de la web e-aditivos [4], ayudándose de un pequeño script en Python para convertir los datos a formato JSON e insertarlos de forma fácil y rápida en la base de datos de Firebase.

Capítulo 4

Plan de riesgos y estimación de costes

4.1. Plan de riesgos

Un riesgo [30] es un evento o condición probable cuya ocurrencia tiene un efecto en los objetivos de un proyecto. Como elementos clave de un riesgo encontramos:

- Está relacionado con el futuro.
- Lleva asociados tanto una causa como un efecto.

En todo proyecto es vital la adecuada planificación y gestión de los riesgos que puede conllevar. Estos se puede dividir en:

- **Riesgos de proyecto**: Son aquellos directamente relacionados con la no consecución de los objetivos marcados por el equipo de desarrollo y el jefe de proyecto.
- **Riesgos de negocio**: Entre ellos se incluyen aquellos relacionados con temas económicos y de mercado, como son los ingresos derivados de las ventas del producto en el mismo, su popularidad, relevancia...

Para llevar a cabo un correcto plan de riesgos es recomendable seguir unas determinadas pautas y pasos:

1. Identificación de riesgos.
2. Análisis y establecimiento de prioridades.
3. Planificación de riesgos.
4. Monitorización y control de los riesgos.

4.1.1. Riesgos encontrados en este proyecto

A lo largo del desarrollo del sprint 0 se ha realizado un análisis de los riesgos de mayor relevancia de cara a la realización del proyecto. Merece la pena subrayar que se trata de riesgos de proyecto. No se contempla la idea

de abordar la salida al mercado de este proyecto. Para cada uno de estos riesgos se muestra un título, una breve descripción, su probabilidad (muy baja, baja, media, alta, muy alta), su impacto (muy bajo, bajo, medio, alto, crítico), un plan de mitigación que reduzca su probabilidad y el plan de contingencia que se adoptará en caso de que se materialice. El resultado de este análisis se muestra en las Tablas 4.1 a 4.10

Título	Falta de formación
Descripción	Un miembro del equipo de desarrollo no dispone de la suficiente formación en las tecnologías empleadas
Probabilidad	Baja
Impacto	Medio
Plan de mitigación	Se han elegido tecnologías con las que se ha estado y se está trabajando en prácticas en empresa
Plan de contingencia	Pedir ayuda

Tabla 4.1: Riesgo de falta de formación

Título	Enfermedad de un desarrollador
Descripción	Un miembro del equipo de desarrollo cae enfermo y no es capaz de desarrollar las tareas pendientes
Probabilidad	Baja
Impacto	Alto
Plan de mitigación	Se ha previsto un sprint de refuerzo en la planificación inicial. Se ha provisto de margen suficiente en cada sprint para la realización de las tareas, incluso algunas a desarrollar en varios sprints por su magnitud
Plan de contingencia	Retrasar ciertas tareas al sprint siguiente

Tabla 4.2: Riesgo de enfermedad

Título	Retraso en el desarrollo de las tareas
Descripción	Las tareas de un sprint llevan más tiempo del esperado
Probabilidad	Media
Impacto	Medio
Plan de mitigación	Se ha previsto un sprint de refuerzo en la planificación inicial. Se ha provisto de margen suficiente en cada sprint para la realización de las tareas, incluso algunas a desarrollar en varios sprints por su magnitud
Plan de contingencia	Retrasar ciertas tareas al sprint siguiente

Tabla 4.3: Riesgo de retraso de las tareas

Título	Ausencia de *Scrum Master*
Descripción	Por disponibilidad horaria el *Scrum Master* (rol ejercido por la tutora) se ausenta o no se encuentra disponible para acudir a los eventos y reuniones importantes del desarrollo del proyecto
Probabilidad	Media
Impacto	Bajo
Plan de mitigación	Se acuerdan reuniones con suficiente antelación para prever imprevistos y reservar horarios
Plan de contingencia	El desarrollador asume una mayor responsabilidad

Tabla 4.4: Riesgo de ausencia del *Scrum Master*

Título	Cambio en las historias de usuario
Descripción	Las historias de usuario cambian, añadiendo o restando funcionalidad al sistema en función del avance del proyecto
Probabilidad	Media
Impacto	Medio
Plan de mitigación	El proyecto se aborda desde un desarrollo ágil, lo que permite fácilmente la adición o borrado de historias de usuario en el backlog. Se ha creado la planificación con tiempo suficiente para atender a imprevistos y cambios de requisitos
Plan de contingencia	Paso de los cambios a siguientes sprints. Replanificación de los mismos para hacerlos frente

Tabla 4.5: Riesgo por cambio en las historias de usuario

Título	Ausencia de información
Descripción	No se encuentra la información necesaria sobre los aditivos alimentarios o, al menos, no de forma sencilla, causando retrasos por su búsqueda
Probabilidad	Media
Impacto	Alto
Plan de mitigación	Obtención de información de la fuente que aporte una lista lo más completa posible, tomando como requisito imprescindible la facilidad de obtención de la misma
Plan de contingencia	Se comparará la información obtenida con distintas fuentes

Tabla 4.6: Riesgo por ausencia de información

Título	Completitud de información
Descripción	Los aditivos alimentarios encontrados pueden no corresponderse con el total de los existentes
Probabilidad	Media
Impacto	Bajo
Plan de mitigación	Obtención de información de la fuente que aporte una lista lo más completa posible, tomando como requisito imprescindible la facilidad de obtención de la misma
Plan de contingencia	Se comparará la información obtenida con distintas fuentes

Tabla 4.7: Riesgo por completitud de información

Título	Veracidad de información
Descripción	Los aditivos alimentarios encontrados y si son aptos o no a cada tipo de dieta pueden no corresponderse con información completamente veraz debido al diseño de la base de datos y la propia veracidad de la información aportada por la fuente elegida
Probabilidad	Alta
Impacto	Medio
Plan de mitigación	Obtención de información de la fuente que aporte una lista lo más completa posible, tomando como requisito imprescindible la facilidad de obtención de la misma
Plan de contingencia	Se comparará la información obtenida con distintas fuentes

Tabla 4.8: Riesgo por completitud de información

Título	Insuficiente cuota gratuita de Firebase
Descripción	Los accesos gratuitos a base de datos que ofrece Firebase pueden llegar a ser insuficientes y requerir de un pequeño pago para seguir adelante con el proyecto
Probabilidad	Muy baja
Impacto	Alto
Plan de mitigación	Se ha consultado previamente que el plan gratuito de Firebase goza de entre 20.000 y 50.000 accesos diarios, que se ha estimado que resultarán suficientes dadas las horas diarias que se dedicarán a trabajar en el proyecto. Se ha previsto un aumento del 30 % en el presupuesto estimado
Plan de contingencia	Se afrontará este pequeño pago sin plan de contingencia determinado, pues el plan de mitigación permite asumir este coste

Tabla 4.9: Riesgo de excesivo uso de Firebase

Título	Planificación optimista
Descripción	No se cumplen las previsiones y el tiempo de desarrollo es mayor del esperado al inicio
Probabilidad	Media
Impacto	Medio
Plan de mitigación	Se ha previsto un aumento de un 30 % en el presupuesto estimado para hacer frente a sobrecoste inesperados por aumento del número de horas de desarrollo
Plan de contingencia	Se descartarán funcionalidades minoritarias que no sea posible realizar por falta de tiempo debido al retraso en otras tareas

Tabla 4.10: Riesgo de planificación optimista

4.2. Planificación presupuestaria inicial

Consultando el Boletín Oficial del Estado (BOE) [31], se puede observar en las tablas salariales que el sueldo para un Analista Programador, Diseñador web (categoría C III) a partir del 31/12/2019 es de 20.896'44 € brutos al año, y su jornada laboral es de 1.800 h/año. Una empresa paga a la Seguridad Social, aproximadamente un 31 % del salario base del trabajador [32]. Por tanto, el coste total para la empresa de un desarrollador de este tipo es de 27.374'34 €. Dividiendo esta cifra entre el total de horas se obtiene que el coste por hora será de 15'21 €. Las horas de las que se dispondrá en el proyecto se estimaron en el Capítulo 2, deduciendo unas 300 horas en total. Por lo tanto, el presupuesto total para contar con un desarrollador en el proyecto será de 4.563 €.

No es necesario añadir ningún tipo de coste fijo por luz, agua o gas, pues el trabajo se desarrolla desde casa o desde edificios de entidades públicas, como la Universidad, corriendo, por tanto, los costes a cargo de entidades externas al proyecto.

La herramienta GitLab será gratuita, puesto que se utilizará la versión que provee la Escuela de Ingeniería Informática de la Universidad de Valladolid.

Tanto Astah como Webscraper como Firebase se utilizarán en su plan gratuito, por lo que no supondrán tampoco coste alguno.

Para el diseño del logo se empleará la web gratuita freelogodesign. La información de los aditivos se obtendrá de la web e-aditivos, como ya se comentó en la Sección 3.8, por lo que también será gratuita.

Consultando la tabla de amortización lineal, en su sección para equipos para procesos de información [43] se observa que el coeficiente lineal máximo es del 25 %. En el desarrollo del proyecto se emplea un ordenador Toshiba Satellite C55-A-1PL, valorado en 658'16 €. Por tanto, el coste de usar este hardware será, aproximadamente, de 658'16 €*25 % anual*4/12 años, lo que da un total de 54'85 € de amortización.

La suma total asciende a 4.617'85 €. Para hacer frente a posibles sobrecostes, ya sea por el uso de determinadas herramientas o aumento de horas dedicadas al desarrollo del proyecto se eleva el presupuesto un 30 %. Esto se traduce en un presupuesto total de 6.003'21 €.

Tipo de coste	Precio (€)
Herramientas	0 €
Costes fijos	0 €
Fuentes, logos, licencias...	0 €
Equipo de desarrollo	4.563 €
Material	54'85 €
Total	4.617'85 €
Total normalizado	6.003'21 €

Tabla 4.11: Resumen del presupuesto del proyecto

4.3. Análisis final de costes

Una vez finalizado el proyecto se recopila la cantidad total de horas empleadas (la cual puede consultarse realizando la suma del total de horas de cada uno de los sprints relatados en el Capítulo 5). A la cifra total se ha de sumar, además, una revisión posterior, en la cual se han empleado 22'5h una vez terminada una primera versión final de la documentación. Este tiempo se debe, principalmente, a que la introducción de herramientas que permitiesen detectar la cobertura de los tests (ver Secciones 8.2 y 8.3) ha conllevado un tiempo de búsqueda de información y ha requerido de un importante esfuerzo de implementación. También se deben añadir 18'5h dedicadas a la creación de una presentación para la defensa del proyecto. Por lo tanto, el cómputo total de horas empleadas es 303. En base a esto se realiza un balance final simulado del presupuesto de este proyecto, cuyos costes se estimaron en la Sección 4.2. En ella se detalla que el coste por hora es de 15'21 €, lo que daría un total de 4.608'63 €. El resto de costes se han mantenido respecto a los estimados al inicio. Puede consultarse un resumen de esta simulación en la Tabla 4.12. En ésta se aprecia que el coste simulado sería de 4.663'48 €. Este presupuesto es asumible, puesto que, pese a haber sobrepasado un poco la cifra estimada, se aportó un colchón de un 30 % para hacer frente a los posibles sobrecostes.

Tipo de coste	Precio (€)
Herramientas	0 €
Costes fijos	0 €
Fuentes, logos, licencias...	0 €
Equipo de desarrollo	4.608'63 €
Material	54'85 €
Total	4.663'48 €

Tabla 4.12: Resumen de los costes simulados del proyecto

Como contraste, en la Tabla 4.13 puede verse un resumen de los costes reales del proyecto. Éstos resultan ser 0 €, pues ni los materiales ni el equipo de desarrollo, únicos tipos de coste que mantenían valor superior a 0, han requerido de ningún tipo de pago por parte de la Universidad de Valladolid, entidad para la que se ha realizado este proyecto como trabajo de fin de grado del alumno.

Tipo de coste	Precio (€)
Herramientas	0 €
Costes fijos	0 €
Fuentes, logos, licencias...	0 €
Equipo de desarrollo	0 €
Material	0 €
Total	0 €

Tabla 4.13: Resumen de los costes reales del proyecto

Capítulo 5

Seguimiento del proyecto

5.1. Seguimiento de los sprints realizados

Se ha realizado un seguimiento sprint a sprint del proyecto. Para ello, se han planificado reuniones semanales con la *Scrum Master* (la tutora) para la correcta supervisión del avance del desarrollador (el alumno). El martes de cada dos semanas se tomará como finalización del sprint en curso y comienzo del siguiente. En cada uno de los sprints se elegirán al principio las tareas a abordar en el mismo. Existirán varios tipos de tareas dependiendo de su función. Así, tendremos:

Chore: tipo de tareas asociadas con la presente documentación o configuración o puesta en marcha de ciertos elementos necesarios para la realización del proyecto.

HU: tareas asociadas con alguna historia de usuario en particular, acompañado por su identificador numérico, que puede ser consultado en la Sección 2.2.2.

Bug: tarea relativa a la resolución de algún bug encontrado.

Q: tarea que se relaciona con mejoras observadas. En este grupo también se incluyen las tareas asociadas a mejoras observadas por la tutora o por los usuarios de la aplicación.

Junto a cada tarea se especifica el tiempo en horas-hombre (HH) empleadas en la realización de la misma. Asimismo, se especificará el estado de la tarea en concreto, a fin de especificar qué tareas se encuentran incompletas y deberán ser pasadas al siguiente sprint.

5.1.1. Sprint 0

Como ya se comentó en la Sección 2.2, en este proyecto se realiza un sprint 0 durante algunos meses, a fin de tener preparada la mayor parte posible de la documentación inicial del mismo y el diseño lógico inicial de la aplicación. Sin embargo, debido a la disponibilidad horaria no fue posible la realización de todas las tareas que se pretendía, resultando en un total de 21 horas empleadas. Las tareas 6 y 7 pasan a formar parte del siguiente sprint. Puede verse un detalle de las tareas realizadas en la Tabla 5.1.

Tarea	Tipo	Descripción	HH	Estado
T - 001	Chore	Definición inicial del backlog	1'5h	Completada
T - 002	Chore	Documentación de la introducción y los objetivos	3'5h	Completada
T - 003	Chore	Documentación de la planificación	4h	Completada
T - 004	Chore	Documentación de las tecnologías que se utilizarán	9h	Completada
T - 005	Chore	Documentación de los riesgos y el plan presupuestario	3h	Completada
T - 006	Chore	Creación de la arquitectura y estructura inicial del proyecto	0h	Incompleta
T - 007	Chore	Adecuación de las herramientas de despliegue e integración continua	0h	Incompleta
Total			**21h**	

Tabla 5.1: Tareas del sprint 0

5.1.2. Sprint 1

En este sprint se comienzan a documentar todas las decisiones tomadas referentes al diseño, tanto lógico como físico de la aplicación, así como la realización de bocetos de la tarea a realizar. Para este sprint se ha tomado una sola historia de usuario, estimada en 5 puntos de historia debido a que se requiere completar previamente la tarea 6 traspasada del sprint anterior.

A pesar de consumir horas de búsqueda de información y realización de pruebas para completar la tarea 7 no se ha conseguido el objetivo. Esto es debido al poco conocimiento acerca de la integración continua en GitLab y su correcto funcionamiento con Cypress. Esto ha causado que no se hayan completado todas las tareas que se pretendía.

Se han consumido un total de 32'75h en este sprint. Las tareas 7 y 16 serán consultadas con la tutora. La tarea 7 quedará parada y se abordará en sprints posteriores, mientras que la 16 se pasará al siguiente sprint. Se ha implementado la historia de usuario 1 pero la realización de sus tests (tareas 17 y 18) se realizará en el siguiente sprint.

El desglose completo de las tareas realizadas en este sprint puede consultarse en la Tabla 5.2.

Tarea	Tipo	Descripción	HH	Estado
T - 008	Chore	Documentación arquitectura y diseño	3h	Completada
T - 009	HU - 01	Realización de bocetos	3h	Completada
T - 006	Chore	Creación de la arquitectura y estructura inicial del proyecto	3h	Completada
T - 007	Chore	Adecuación de las herramientas de despliegue e integración continua	6h	Incompleta
T - 010	Chore	Obtención y preparación de datos iniciales de aditivos alimentarios	6h	Completada
T - 011	Chore	Documentación del seguimiento de los sprints 0 y 1	1h	Completada
T - 012	HU - 01	Maquetación pantalla principal	2h	Completada
T - 013	HU - 01	Maquetación pantalla aditivo	4'5h	Completada
T - 014	HU - 01	Búsqueda de aditivo por código	3'5h	Completada
T - 015	Chore	Documentación tecnologías añadidas	0'25h	Completada
T - 016	Chore	Modelos de diseño de los componentes creados para el sprint 1	0'5h	Incompleta
T - 017	HU - 01	Tests unitarios	0h	Incompleta
T - 018	HU - 01	Tests end-to-end	0h	Incompleta
Total			**32'75h**	

Tabla 5.2: Tareas del sprint 1

5.1.3. Sprint 2

En este sprint se abordan las tareas incompletas del sprint anterior, a excepción de la tarea 7, y se realizan todas aquellas asociadas a las historias de usuario 2 y 3 (diseño de bocetos, maquetaciones, modelo de diseño de componentes y funcionamiento). Además, se realiza una mejora en la funcionalidad de la historia de usuario 1 tras contrastar opiniones con la tutora.

Se decide añadir una nueva historia de usuario, la número 13, que permita la internacionalización de todos los textos de la aplicación para que sea posible que aparezca en diferentes idiomas. Eso implica que en cada uno de los posteriores sprints se requerirá de una tarea asociada a esta historia de usuario para añadir en el fichero correspondiente las cadenas de texto necesarias.

Respecto de la historia de usuario 3 se debe destacar que se trata de una historia de usuario compuesta, en la que se incluye, además del inicio de sesión, el cierre de sesión y el bloqueo de determinadas rutas de la web en función de este hecho. Su correspondiente implementación de funcionalidad y tests se incluye dentro de esta historia de usuario.

Durante los días previos al cierre de sprint se experimentaron problemas con la plataforma de GitLab que impedían conectar con el repositorio remoto. Eso causó que se tuvieran varias ramas abiertas con tareas dependientes unas de otras que no pudieron ser fusionadas. Esto ocasionó una pérdida de aproximadamente 1 hora a mayores entre unas tareas y otras para resolver todos los conflictos que surgieron cuando se resolvió el problema.

Pese al incidente comentado, se han completado satisfactoriamente todas las tareas propuestas al inicio del sprint, en las que se ha empleado un total de 30'5 horas. No existen tareas que deban ser pasadas al sprint siguiente.

El desglose completo de las tareas de este sprint puede ser consultado en la Tabla 5.3.

Tarea	Tipo	Descripción	HH	Estado
T - 016	Chore	Modelos de diseño de los componentes creados para el sprint 1	0'5h	Completada
T - 017	HU - 01	Tests unitarios	5'5h	Completada
T - 018	HU - 01	Tests end-to-end	2'25h	Completada
T - 019	Q	Mejora de la experiencia de usuario en la búsqueda de un aditivo por código	1h	Completada
T - 020	HU - 02	Realización de bocetos	0'75h	Completada
T - 021	HU - 03	Realización de bocetos	0'75h	Completada
T - 022	HU - 02	Maquetación pantalla registro	2h	Completada
T - 023	HU - 03	Maquetación pantalla inicio de sesión	0'5h	Completada
T - 024	HU - 02	Registro y control de errores	7h	Completada
T - 025	HU - 03	Inicio de sesión y control de errores	2h	Completada
T - 026	HU - 02	Tests unitarios	2h	Completada
T - 027	HU - 02	Tests end-to-end	1'5h	Completada
T - 028	HU - 03	Tests unitarios	1'25h	Completada
T - 029	HU - 03	Tests end-to-end	1h	Completada
T - 030	Chore	Modelos de diseño de los componentes creados para el sprint 2	0'25h	Completada
T - 031	Chore	Documentación del seguimiento del sprint 2	0'75h	Completada
T - 032	HU - 13	Preparación y adecuación de las herramientas de internacionalización	0'25h	Completada
T - 033	HU - 13	Internacionalización de las cadenas de texto introducidas en los componentes del sprint 2	1'25h	Completada
Total			**30'5h**	

Tabla 5.3: Tareas del sprint 2

5.1.4. Sprint 3

Durante este sprint se abordan las historias de usuario 4, 8 y 9. Puede verse un desglose de las tareas realizadas en el mismo en la Tabla 5.4.

En este sprint se modifica notablemente la velocidad inicial establecida en la Sección 2.2. Ésta pasará a tomar un valor de 9 puntos de historia por los 6 que tenía al principio, debido a que no se requiere la reserva de excesivos puntos de historia para tareas de tipo chore y se ha adquirido destreza en las tecnologías empleadas.

La tarea 48, que se corresponde con los tests unitarios de la historia de usuario 9, se cancela. Se sustituyen todos los tests unitarios de dicha historia de usuario por tests end-to-end. Tras haber probado numerosas formas de montar el componente (un diálogo modal) con la herramienta Jest se han visto problemas de alojamiento de memoria que no permitían finalizar al test, que quedaba en bucle infinito, sin razón aparente. El tiempo en esta tarea ha sido, casi en su totalidad, dedicado a buscar información al respecto, pero no se ha encontrado solución.

Respecto de la tarea 44, correspondiente a los tests end-to-end de la historia de usuario 8, se debe reseñar que muchos de estos tests se han sustituido por tests unitarios. Esto se debe al gran acoplamiento que existiría entre la base de datos y el frontal de la aplicación. Lo ideal, en estos casos, sería disponer de varios entornos, cada uno de ellos con su propia base de datos: desarrollo, preproducción y producción. Con esta medida se controla qué datos

Tarea	Tipo	Descripción	HH	Estado
T - 034	HU - 04	Realización de bocetos	0'5h	Completada
T - 035	HU - 04	Maquetación pantalla del perfil	3'5h	Completada
T - 036	HU - 04	Ver perfil	1h	Completada
T - 037	HU - 04	Editar perfil y control de errores	1'5h	Completada
T - 038	HU - 04	Tests unitarios	0'75h	Completada
T - 039	HU - 04	Tests end-to-end	1h	Completada
T - 040	HU - 08	Realización de bocetos	0'5h	Completada
T - 041	HU - 08	Maquetación pantalla principal del foro	4h	Completada
T - 042	HU - 08	Funcionamiento pantalla principal del foro	4'5h	Completada
T - 043	HU - 08	Tests unitarios	3h	Completada
T - 044	HU - 08	Tests end-to-end	0'75h	Completada
T - 045	HU - 09	Realización de bocetos	0'25h	Completada
T - 046	HU - 09	Maquetación pantalla de crear nuevo tema	1'5h	Completada
T - 047	HU - 09	Añadir nuevo tema y control de errores	1'5h	Completada
T - 048	HU - 09	Tests unitarios	1'5h	Cancelada
T - 049	HU - 09	Tests end-to-end	1h	Completada
T - 050	HU - 13	Internacionalización de las cadenas de texto introducidas en el sprint 3	0'25h	Completada
T - 051	Chore	Modelos de diseño de los componentes creados para el sprint 3	0'25h	Completada
T - 052	Chore	Documentación del seguimiento del sprint 3	0'75h	Completada
Total			**28h**	

Tabla 5.4: Tareas del sprint 3

aparecen en la base de datos de desarrollo y, aunque también exista acoplamiento, los tests seguirán pasando, pues los usuarios no interaccionan con estos datos y los desarrolladores estarán informados en todo momento de los cambios. En este proyecto no se contempla la inclusión de diferentes entornos.

Cabe destacar que este sprint se ha visto caracterizado por la crisis del coronavirus COVID-19 [56] y el estado de alarma decretado por el Gobierno el día 14 de marzo de 2020. Esto último obligó a la cuarentena completa del país, recluyendo a todos los habitantes del mismo en sus casas por un período indefinido de tiempo. Esto afectó, especialmente, a las reuniones periódicas con la tutora, que pasaron a ser vía online.

Se han empleado un total de 28 horas. Pese a haber incrementado la velocidad inicial se han completado las tareas de forma rápida y eficiente, más de lo que cabía esperar al inicio del sprint, pues todos los sprints están estimados en unas 30 horas. No existe ninguna tarea que deba ser pasa a un sprint posterior.

5.1.5. Sprint 4

Durante este sprint se abordan las historias de usuario 5, 10 y 11. Puede verse el desglose completo de las tareas realizadas en la Tabla 5.5.

Sobre la tarea 56 (tests unitarios de los comentarios en un tema) cabe destacar que, por los mismos problemas

Tarea	Tipo	Descripción	HH	Estado
T - 053	HU - 05	Realización de bocetos	0'5h	Completada
T - 054	HU - 05	Maquetación pantalla de comentarios	3'5h	Completada
T - 055	HU - 05	Funcionamiento comentarios	3'25h	Completada
T - 056	HU - 05	Tests unitarios	1'5h	Completada
T - 057	HU - 05	Tests end-to-end	1'75h	Completada
T - 058	HU - 10	Realización de bocetos	0'25h	Completada
T - 059	HU - 10	Maquetación acción de denunciar comentario	1'5h	Completada
T - 060	HU - 10	Funcionamiento denuncia de comentario	1h	Completada
T - 061	HU - 10	Test unitarios	1h	Completada
T - 062	HU - 10	Tests end-to-end	0'5h	Completada
T - 063	HU - 11	Realización de bocetos	0'5h	Completada
T - 064	HU - 11	Maquetación pantalla para atender denuncias	3'5h	Completada
T - 065	HU - 11	Funcionamiento pantalla denuncias	4h	Completada
T - 066	HU - 11	Tests unitarios	1'5h	Completada
T - 067	HU - 11	Tests end-to-end	0'75h	Completada
T - 068	HU - 13	Internacionalización de las cadenas de texto introducidas en los componentes del sprint 4	0'25h	Completada
T - 069	Chore	Documentación del seguimiento del sprint 4	0'5h	Completada
T - 070	Chore	Modelos de diseño de los componentes creados en el sprint 4	0'25h	Completada
T - 071	Q	Añadir diálogo de confirmación de denuncia de comentario	0'5h	Completada
T - 072	Chore	Documentar fase de testing	2'25h	En progreso
T - 073	Q	Añadir visualización de carga al registro	0'25h	Completada
Total			**29h**	

Tabla 5.5: Tareas del sprint 4

comentados en la Sección 5.1.4 para los diálogos modales, se han realizado tests unitarios hasta donde se ha podido y se sustituyen los restantes por tests end-to-end. Por tanto, los tests de la funcionalidad completa de añadir un comentario a un tema se realizan en su totalidad en la tarea 57. Asimismo, en esta última tarea se debe reseñar que se han sustituido numerosos tests por tests unitarios debido al excesivo acoplamiento que existiría entre los datos almacenados y los tests. Es por ello que ambas tareas y, por ende, ambos tipos de tests, se complementan entre sí, aportando la suficiente completitud a éstos.

La tarea 72, correspondiente a la documentación del capítulo de testing, quedará en progreso hasta el momento en que finalicen los desarrollos, pues será entonces cuando acabará la fase de testing y podrá completarse.

El cómputo total de horas empleadas en este sprint es de 29. El desarrollo del sprint ha sido satisfactorio, completando todas las tareas dentro del tiempo estimado. Por tanto, la velocidad, que cambió en el sprint 3, se mantiene adecuada en este nivel.

5.1.6. Sprint 5

A lo largo del sprint 5 se abordan las historias de usuarios 6 y 7. Además, surge la historia de usuario 14, que queda reflejada junto a las demás en la Tabla 2.2. Asimismo, se retoma la tarea 72 del sprint anterior, que quedó en progreso ya que la documentación de la fase de testing se realiza de forma incremental. Esta tarea vuelve a quedar en progreso. Sin embargo, sí se realizan numerosos avances al respecto, pues se plantean y diseñan los escenarios y las pruebas de usuario, explicado todo en la Sección 8.5.

En este sprint se emplean un total de 28'75 horas. Ha resultado satisfactorio, completando todas las tareas previstas (excepto la ya comentada tarea 72).

Tarea	Tipo	Descripción	HH	Estado
T - 072	Chore	Documentación fase de testing	4'75h	En progreso
T - 074	HU - 06	Realización de bocetos	0'25h	Completada
T - 075	HU - 06	Maquetación eliminar usuario	0'75h	Completada
T - 076	HU - 06	Funcionamiento eliminar usuario	2h	Completada
T - 077	HU - 06	Tests unitarios	0'75h	Completada
T - 078	HU - 06	Tests end-to-end	1h	Completada
T - 079	HU - 07	Realización de bocetos	0'25h	Completada
T - 080	HU - 07	Maquetación eliminar comentario (administrador)	5h	Completada
T - 081	HU - 07	Funcionamiento eliminar comentario (administrador)	2'75h	Completada
T - 082	HU - 07	Tests unitarios	0'5h	Completada
T - 083	HU - 07	Tests end-to-end	1h	Completada
T - 084	HU - 14	Realización de bocetos	0'25h	Completada
T - 085	HU - 14	Maquetacion eliminar comentario (usuario)	0'25h	Completada
T - 086	HU - 14	Funcionamiento eliminar comentario (usuario)	1'75h	Completada
T - 087	HU - 14	Tests unitarios	0'25h	Completada
T - 088	HU - 14	Tests end-to-end	0'5h	Completada
T - 089	HU - 13	Internacionalización de las cadenas de texto introducidas en los componentes del sprint 5	0'25h	Completada
T - 090	Chore	Documentación del seguimiento del sprint 5	0'5h	Completada
T - 091	Chore	Realizar despliegue	2h	Completada
T - 092	Q	Añadir registro de usuarios eliminados	2h	Completada
T - 093	Chore	Añadir licencia	2h	Completada
Total			**28'75h**	

Tabla 5.6: Tareas del sprint 5

5.1.7. Sprint 6

Durante el sexto sprint se añade la historia de usuario número 15, documentada debidamente en la Tabla 2.2. Ésta se aborda, junto con la 12, en el desarrollo de este sprint, en el cual el objetivo claro pasa a ser completar los desarrollos previstos para realizar los tests con usuarios de la fase *User Acceptance Testing* (UAT), ver Sección 8.5, y obtener y documentar las conclusiones extraídas de los mismos. Por ello, la tarea 72 se retoma. Todas las tareas de este sprint se encuentran desglosadas en la Tabla 5.7.

A la finalización del sprint, la tarea 72 de nuevo vuelve a quedar en progreso pues, aunque se pretendía al inicio documentar y extraer conclusiones de la fase UAT, no ha sido posible su desarrollo porque un usuario demoró su respuesta más tiempo del esperado. El tiempo total empleado en todas las tareas ha sido de 30'25 horas.

Tarea	Tipo	Descripción	HH	Estado
T - 072	Chore	Documentación fase de testing	3'25h	En progreso
T - 094	HU - 12	Funcionamiento del bot	3'5h	Completada
T - 095	HU - 12	Tests de sistema	0'75h	Completada
T - 096	HU - 15	Realización de bocetos	0'5h	Completada
T - 097	HU - 15	Maquetación desestimar denuncia	0'25h	Completada
T - 098	HU - 15	Funcionamiento desestimar denuncia	1'5h	Completada
T - 099	HU - 15	Tests unitarios	0'25h	Completada
T - 100	HU - 15	Tests end-to-end	1'25h	Completada
T - 101	HU - 13	Internacionalización de las cadenas de texto introducidas en los componentes del sprint 6	0'25h	Completada
T - 102	HU - 13	Traducción de todas las cadenas de texto a inglés	2h	Completada
T - 103	HU - 13	Introducción de cambio de idioma en la web y selección de idioma en los temas del foro	5'25h	Completada
T - 104	Chore	Preparación tests UAT	2'5h	Completada
T - 105	Chore	Documentación del seguimiento del sprint 6	0'25h	Completada
T - 106	Q	Mejora de la visualización para móvil (web responsive)	8'25h	Completada
T - 107	Bug	Solución de error al desestimar los reportes de una respuesta	0'5h	Completada
Total			**30'25h**	

Tabla 5.7: Tareas del sprint 6

5.1.8. Sprint 7

En este séptimo sprint se abordan todas las tareas relativas a las incidencias surgidas en la fase UAT (desglosadas en la Tabla 8.6), así como la extracción y documentación de las conclusiones que se derivan de las opiniones de los usuarios, por lo que se continúa con la tarea 72. En la Tabla 5.8 únicamente se añade en la descripción el número identificativo de la incidencia a la que hace referencia la tarea.

En total se han empleado 30 horas para la realización de todas las tareas previstas para este sprint. La tarea 120 se deja en progreso, puesto que, para completarla, es necesario terminar la tarea 7, que quedó interrumpida en el sprint 1 (véase la Sección 5.1.2). Ésta última se retomará en el sprint siguiente.

5.1.9. Sprint 8

En este sprint se abordan, esencialmente, tareas chore que permiten cerrar el proyecto. Además, se retoman las tareas 7, incompleta desde el sprint 1 (Sección 5.1.2), y 120, del sprint anterior. El resumen de las tareas realizadas puede observarse en la Tabla 5.9.

En total se han empleado 32'25h en el desarrollo de las tareas de este sprint, que ha resultado satisfactorio, pues se han completado todas las tareas previstas.

Tarea	Tipo	Descripción	HH	Estado
T - 072	Chore	Documentación fase de testing	5'5h	Completada
T - 108	Chore	Documentación del seguimiento del sprint 7	0'5h	Completada
T - 109	Q	Incidencia 1	0'75h	Completada
T - 110	Q	Incidencia 2	1h	Completada
T - 111	Q	Incidencia 3	1'5h	Completada
T - 112	Q	Incidencia 6	0'5h	Completada
T - 113	Q	Incidencia 7	0'75h	Completada
T - 114	Q	Incidencia 9	0'75h	Completada
T - 115	Q	Incidencia 8	3h	Completada
T - 116	Q	Incidencia 10	2h	Completada
T - 117	Q	Incidencia 5	1'75h	Completada
T - 118	Q	Incidencia 11	3'25h	Completada
T - 119	Q	Incidencia 4	0'75h	Completada
T - 120	Chore	Documentación fase de implementación	3'5h	En progreso
T - 121	Chore	Modelo de la base de datos	4h	Completada
T - 122	Chore	Documentación de diseño para el sprint 7	0'5h	Completada
Total			**30h**	

Tabla 5.8: Tareas del sprint 7

Tarea	Tipo	Descripción	HH	Estado
T - 007	Chore	Adecuación de las herramientas de despliegue e integración continua	12h	Completada
T - 120	Chore	Documentación fase de implementación	2'25h	Completada
T - 123	Chore	Documentación del seguimiento del sprint 8	0'5h	Completada
T - 124	Chore	Cálculo del coste final	1'75h	Completada
T - 125	Chore	Documentación de conclusiones y líneas futuras	2h	Completada
T - 126	Chore	Agradecimientos	0'25h	Completada
T - 127	Chore	Documentación del manual de usuario	3h	Completada
T - 128	Chore	Documentación del manual de despliegue	2'5h	Completada
T - 129	Q	Revisión de documentación completa	8h	Completada
Total			**32'25h**	

Tabla 5.9: Tareas del sprint 8

Capítulo 6

Diseño

En el diseño de este proyecto se aplican los modelos SaaS y BasS. El diseño arquitectónico se basa fundamentalmente en el patrón MVVM, desarrollando un diseño guiado por componentes. En este Capítulo se presentan estos modelos y patrones, y se documentan otros elementos importantes del diseño como el diseño de la interfaz de usuario por sprints y los modelos de despliegue tanto de la aplicación web como del bot de Telegram.

6.1. SaaS

Software-as-a-Service (SaaS) [40] es un modelo de distribución de software mediante el cual un programa se aloja en un servidor externo, accesible a través de Internet. Se trata de un concepto muy popularizado hoy en día debido al auge de la computación y almacenamiento de datos en la nube. Es de especial relevancia en el desarrollo web, permitiendo a los clientes acceder al mismo a través de un navegador.

6.1.1. Ventajas

- Se basa en un modelo de suscripción, por lo que los usuarios pagan cuando realmente lo utilizan.
- Puede desplegarse y ponerse en marcha rápidamente.
- Favorece el acceso desde cualquier lugar con tan sólo disponer de conexión a Internet.

6.1.2. Desventajas

- Requiere de conexión constante a Internet.
- Los datos y el software se encuentran en un servidor de un proveedor externo por lo que la confidencialidad y la seguridad pueden verse comprometidas.

6.2. BaaS

Backend-as-a-Service (BaaS) [23] es un modelo que permite a un desarrollador abstraer la parte del servidor, de manera que sólo debe preocuparse por el funcionamiento de la aplicación en cuestión (Figura 6.1). Es de importante

trascendencia para el desarrollo móvil y web, permitiendo actuar de puente entre el lado *Front-end* y los servicios *Back-end*. Este modelo permite gestionar desde una base de datos hasta funciones que sirvan esos datos o realicen operaciones sobre ellos. Se basa en la idea de almacenar todo tipo de estructuras y funciones típicas del lado del servidor en proveedores en la nube que se encargarán del correcto funcionamiento, mantenimiento y disponibilidad de los servicios, así como de la autenticación, asunto que podría traer más de un quebradero de cabeza (Figura 6.2). Como toda tecnología conlleva sus ventajas y sus inconvenientes [24].

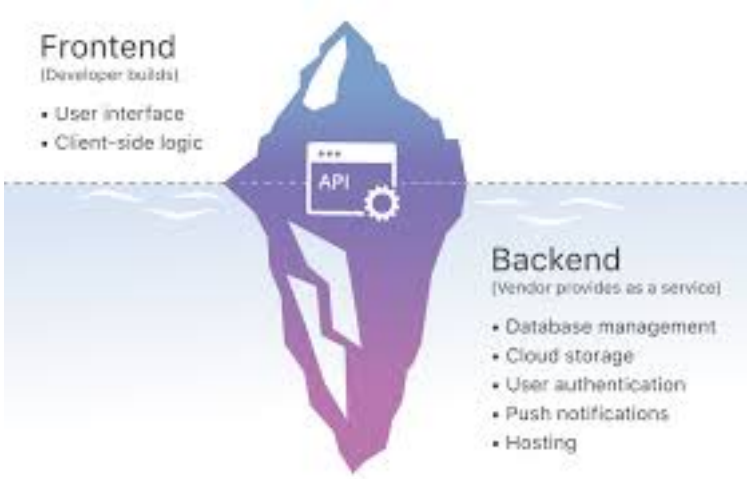

Figura 6.1: Visión modelo BaaS por parte de un desarrollador. Tomada de [51]

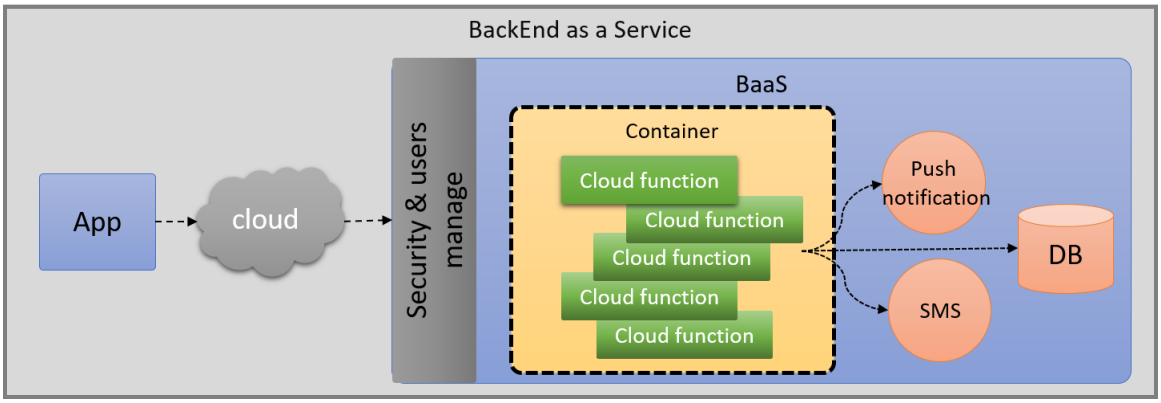

Figura 6.2: Funcionamiento básico del modelo BaaS. Tomada de [52]

6.2.1. Ventajas

- Ofrece almacenamiento en la nube, favoreciendo el acceso desde cualquier lugar.
- No requiere coste de mantenimiento de un servidor físico sino que es el proveedor quien se encarga de ello, cobrando una cuota en función del uso.
- Permiten obtener un servicio *Back-end* completo, con acceso a datos en tiempo real y ejecución de código en la nube.

6.2.2. Desventajas

- Menor flexibilidad en la construcción del *Back-end*.

- Dependencia del proveedor en caso de caída de los servicios o de pobre rendimiento o inadecuado funcionamiento.
- Se necesita conexión a Internet para acceder a los servicios que ofrece.

En este proyecto se utiliza Firebase (ver apartado 3.4) como soporte al modelo BaaS.

6.3. MVVM

Modelo-Vista-Modelo de vista (Model-View-ViewModel, MVVM) [44] es un patrón arquitectónico de software centrado en la separación completa entre el código de la interfaz de usuario y el de la lógica de negocio. Consta de 3 partes:

Modelo (Model): conjunto de estructuras de datos que almacenan la lógica de negocio y el estado de la aplicación.

Vista (View): capa que se corresponde con la interfaz de usuario, que muestra sus elementos al exterior y captura las interacciones del usuario con los mismos.

Modelo de vista (ViewModel): se encarga de enlazar la interfaz de usuario con el modelo, conteniendo todos los métodos necesarios para interactuar con el modelo tras la captura de una interacción por parte de la vista.

La principal característica que permite diferenciar este patrón de otros similares como MVC es que los datos del modelo se actualizan automáticamente por medio de un binder cuando el usuario cambia los que la vista está mostrando. De esta forma, las actualizaciones de la vista dejan de ser manuales para pasar a ser automáticas [45]. En la Figura 6.3 puede observarse gráficamente la interacción entre los elementos del patrón MVVM.

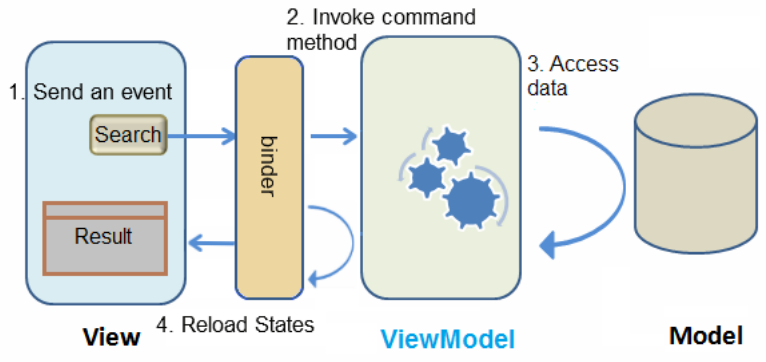

Figura 6.3: Interacción entre los elementos del patrón MVVM. Tomada de [45]

6.3.1. MVVM en este proyecto

La utilización del framework Vue.js en este proyecto (Sección 3.2) facilita la aplicación del patrón MVVM, pues cada instancia Vue está centrada en actuar como modelo de vista [46]. De esta forma, aprovecha el binding de dos direcciones que provee este patrón (Modelo → Vista Vista → Modelo). Los componentes del patrón aplicados a este framework son:

Modelo: conjunto de datos que contiene el estado de Vuex y variables de la sección data.

Vista: sección template.

Modelo de vista: instancia Vue y sección computed.

La instancia Vue relaciona los datos del modelo y los elementos de la vista mediante el binding en dos direcciones, muy en consonancia con la reactividad explicada en la Sección 3.2, empleando para ello:

Directivas [47]: atributos HTML que permiten relacionar datos con elementos visuales o manipular estos últimos. Entre las más importantes destacan:

- **v-model:** permite relacionar un elemento visual con una variable en ambas direcciones, de forma que si uno cambia el otro también, y viceversa (binding en dos direcciones).
- **v-bind:** permite relacionar una variable con un elemento visual en una dirección (Modelo → Vista). Se emplea, generalmente, para atributos HTML. Puede sustituirse por dos puntos.
- **v-for:** realiza un bucle en la capa visual del elemento que contenga esta directiva en base a un array o una lista de elementos, con quien establece una relación de una única dirección.
- **v-if, v-else, v-else-if:** mediante su uso se permite controlar si determinados elementos visuales deberán ser tenidos en cuenta por la instancia Vue para generar la interfaz de usuario o no. Con el empleo de estas directivas un elemento que no cumpla las condiciones no se incluirá en el DOM virtual de Vue.
- **v-show:** similar a la anterior, sólo que en este caso el elemento se incluye en el DOM virtual pero no se muestra.
- **v-on:** permite especificar un evento con el que crear un listener en ese elemento visual y asociarlo a un método que controle las acciones a realizar. Se puede sustituir por @.

Double mustache binding: permite mostrar datos en la interfaz de usuario. Se emplea escribiendo la variable a mostrar entre dobles llaves.

A modo de resumen general, la Figura 6.4 muestra un esquema básico de la estructura del patrón MVVM sobre el framework Vue.js y la Figura 6.5 muestra el funcionamiento básico del binding en dicho framework.

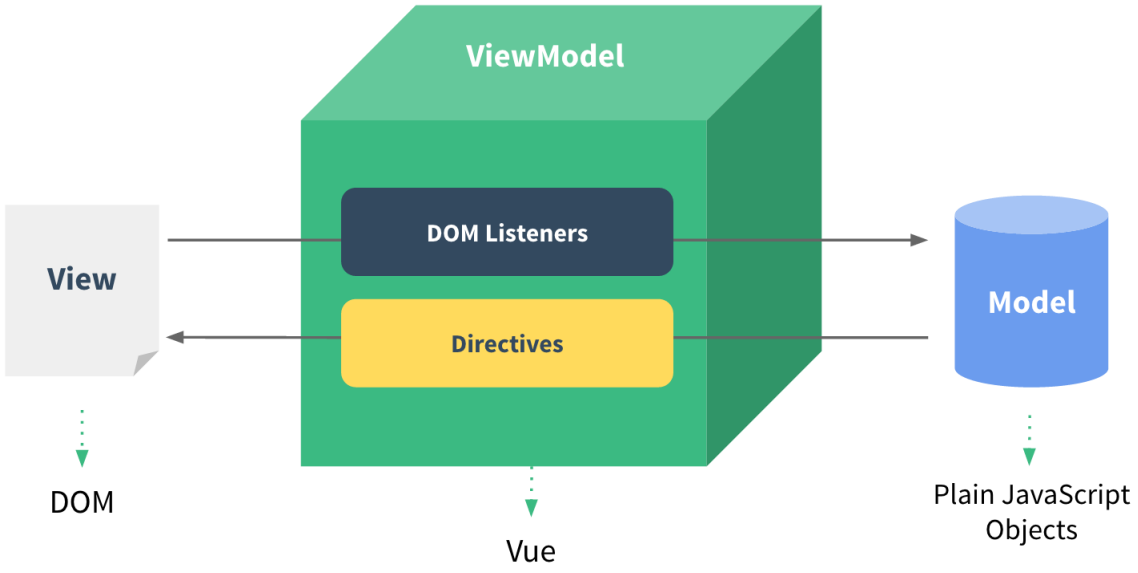

Figura 6.4: MVVM en Vue. Tomada de [46]

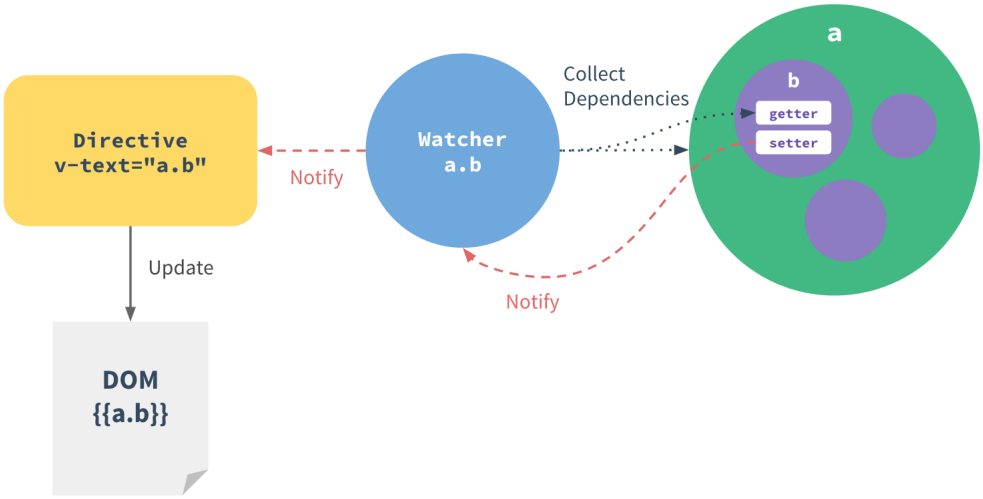

Figura 6.5: Funcionamiento del binding en Vue. Tomada de [46]

6.4. Diseño guiado por componentes o Atomic Web Design

El diseño guiado por componentes [48] es un patrón de diseño que permite desgranar un sistema software en múltiples partes más pequeñas (posiblemente también éstas en subpartes, y así sucesivamente). Es especialmente utilizado hoy en día para el desarrollo de páginas web. La principal ventaja de esto es la reutilización de los componentes que se crean para generar otros más complejos. Brad Frost [49] compara este hecho con el modelo atómico tradicional de la Física. De esta forma, encontramos que una web consta de átomos, moléculas, organismos,

templates y páginas, cada uno de ellos formado a su vez por elementos del tipo anterior a él. A esta metáfora se refiere el nombre *Atomic Web Design*.

6.4.1. Ventajas

- Los componentes son autocontenidos y encapsulados, almacenando cada uno su propio estado interno.
- Son independientes.
- Favorecen la reutilización.
- Simplifican la fase de testing.
- Facilitan el mantenimiento del sistema.
- Permiten aislar errores.

6.4.2. Desventajas

- Aumento de la carga de trabajo en diseño. Se debe descomponer el sistema y observar sus partes reutilizables para convertirlas en componentes.
- Mayor carga en la fase de testing. Se debe probar cada componente por separado y después probar su integración con el resto de componentes.

En este proyecto se emplea Atomic Web Design para su desarrollo, en el que cada uno de los componentes que se crean será una instancia Vue, contenida en un fichero con extensión .vue, basado en el concepto Single File Component. Cada uno de ellos consta de las partes explicadas en la Sección 3.2.

6.4.3. Componentes creados en el desarrollo del proyecto

Como se explicó en la Sección 3.2, los componentes en Vue siguen la filosofía Single File Component, formados por tres partes claramente diferenciadas: template, script y style. Estas se incluyen en un fichero de extensión .vue mediante el que se manifiesta el componente en sí. En la Figura 6.6 se expresan claramente estas relaciones de forma más gráfica mediante un diagrama de un componente genérico en Vue. Todos los componentes creados en el desarrollo de este proyecto siguen esta estructura. Cabe destacar, como se explicó también en la Sección 3.2, que el artefacto script consta de las partes especificadas en dicha sección.

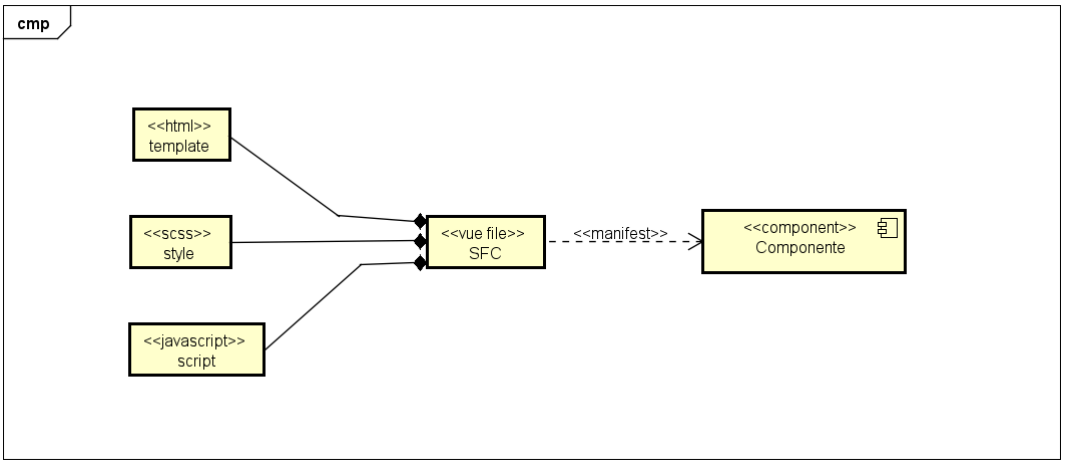

Figura 6.6: Modelo de diseño de un componente genérico en Vue

A continuación se enumeran y describen los componentes creados para el desarrollo del proyecto:

- App: contiene la aplicación completa.
- HeaderBar: barra superior de la web, utilizada por App.
- LoginRegister: componente que permite dirigirse a las pantallas de inicio de sesión y de registro y que, en caso de haber iniciado ya sesión, mostrará las opciones y el nombre del usuario. Es utilizado por HeaderBar.
- Main: pantalla principal de la aplicación.
- AditiveInfo: pantalla de información de un aditivo en cuestión.
- Register: pantalla de registro de un usuario.
- Profile: pantalla que muestra la información del perfil de un usuario.
- PrivacityPolicy: pantalla que muestra la política de privacidad de la web.
- Login: pantalla de inicio de sesión.
- ResetPassword: componente usado por Login que muestra un pequeño diálogo modal para restaurar la contraseña.
- Forum: pantalla principal del foro.
- TopicForum: componente que muestra los metadatos de un tema del foro. Es utilizado por Forum.
- TopicTag: pequeño componente que gestiona la lógica para mostrar la etiqueta de un tema del foro. Es utilizado por TopicForum.
- AddTopic: componente que gestiona la lógica de una ventana modal que permite añadir un nuevo tema al foro. Es utilizado por Forum.
- Topic: pantalla que muestra los comentarios de un tema concreto.
- CommentTopic: componente que muestra la información asociada a un comentario de un tema. Es usado por Topic y por sí mismo para mostrar las respuestas de un comentario.

- AddComment: componente que gestiona la lógica de una ventana modal que permite añadir un comentario o respuesta en un tema. Es usado por Topic.
- Reports: pantalla de que muestra todas las denuncias a comentarios realizados por los usuarios. Sólo es accesible para el administrador.
- ReportInfo: componente que muestra la información de cada denuncia particular. Es utilizado por Reports.
- BottomBar: componente que muestra la información de la licencia y los derechos de autor. Es utilizado directamente por App.

La Figura 6.7 muestra gráficamente la relación de todos estos componentes mencionados. Cabe destacar que, pese a que se observa que App establece relaciones de dependencia con prácticamente todos los demás componentes, éstas se manifiestan a través de <router-view>, un componente que provee Vue Router que actúa como envoltorio de todos los demás. Únicamente HeaderBar y BottomBar se relacionan directamente con App.

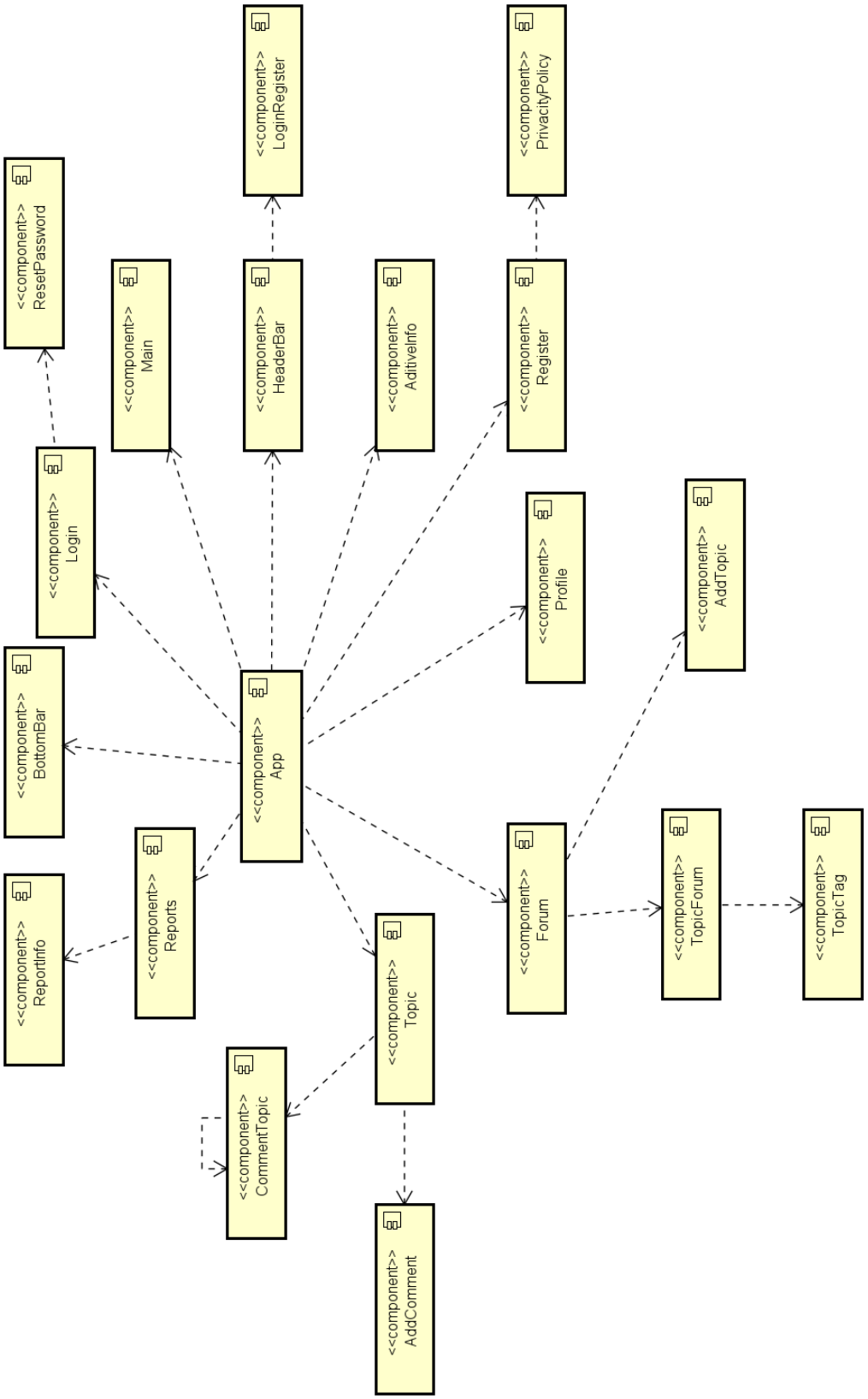

Figura 6.7: Diagrama de componentes

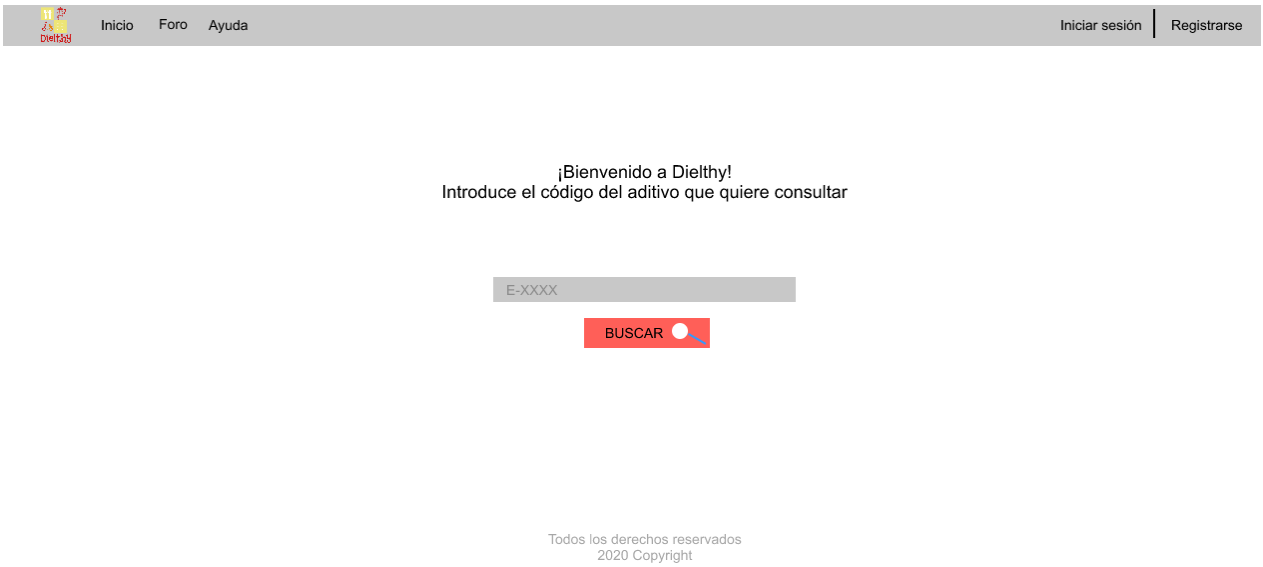

Figura 6.8: Boceto página principal

6.5. Diseño de la interfaz de usuario

La interfaz de usuario buscará acercarse a usuarios del mayor rango de edad posible y con conocimientos técnicos sobre el manejo de las webs muy variados. Es por ello que se buscará la sencillez y simplicidad, resultando esto en una buena facilidad de uso y aprendizaje para cualquiera que desee visitar la página.

6.5.1. Bocetos

Los bocetos que se muestran a continuación se han realizado con la herramienta InVision, explicada en la Sección 3.6. Los colores son meramente orientativos en este momento del desarrollo del producto.

Se organiza en apartados según el sprint en el que se ha ido produciendo cada boceto, excepto el sprint 0 en el cual no se realizó ningún boceto por dedicarlo íntegramente a tareas chore.

También se indica si en algún sprint ha habido que introducir modificaciones en bocetos producidos en sprints anteriores, así como las diferencias sustanciales que existen entre la implementación final y el boceto inicial. Todas las historias de usuario a las que se hace referencia se encuentran explicadas en la Tabla 2.2.

Sprint 1

Durante el primer sprint se realizaron los bocetos asociados a la historia de usuario número 1 (Figuras 6.8 y 6.9).

En la implementación se cambió el diseño del boceto de la Figura 6.8. Se añadió el logo en la parte central de la pantalla, sobre el texto, puesto que la pantalla quedaba excesivamente vacía e insulsa.

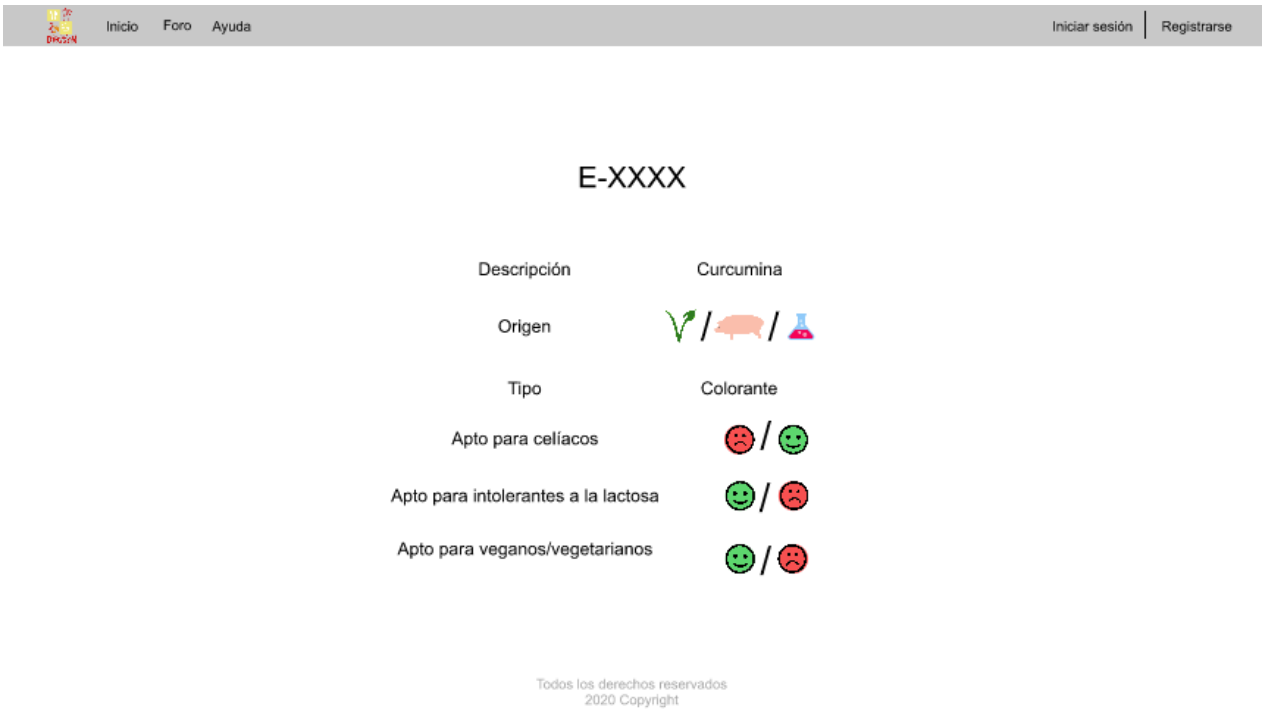

Figura 6.9: Boceto página de información de un aditivo

Sprint 2

Durante el segundo sprint se realizaron los bocetos asociados a las historias de usuario 2 y 3, que se corresponden con el registro y el inicio de sesión (Figuras 6.10 y 6.11).

Respecto a los cambios que se produjeron en la implementación, se destaca la inclusión de la aceptación de las políticas de privacidad como requisito indispensable para registrarse (Figura 6.10).

Sprint 3

Durante el tercer sprint se realizaron bocetos asociados a las historias de usuario 4, 8 y 9, que se corresponden con la edición/vista del perfil, la pantalla principal del foro y la ventana de creación de tema en el foro (Figuras 6.12, 6.13, 6.14 y 6.15).

El boceto de la Figura 6.12 difiere de la implementación real en la eliminación del campo contraseña, considerado innecesario a la hora de ver la información del perfil de un usuario. Asimismo, se añade una opción en la parte superior de la pantalla, bajo la cabecera general, que, en caso de no haber verificado la cuenta por medio del correo enviado al email indicado, muestra una alerta al respecto y permite reenviar dicho correo de verificación. También se cambia el botón editar por un icono de un lápiz.

Respecto del boceto de la Figura 6.14, se debe destacar la inclusión en la implementación final de un filtro por nombre de tema, así como un botón en la parte superior derecha que permite abrir el diálogo modal de la Figura 6.15 para iniciar un nuevo tema. También se ha decidido mostrar la etiqueta de cada tema, cuya función es relacionar un tema con un tipo de dieta y cuyo filtro se corresponde con el primero de los dos selectores que se observan en la figura.

Figura 6.10: Boceto página de registro

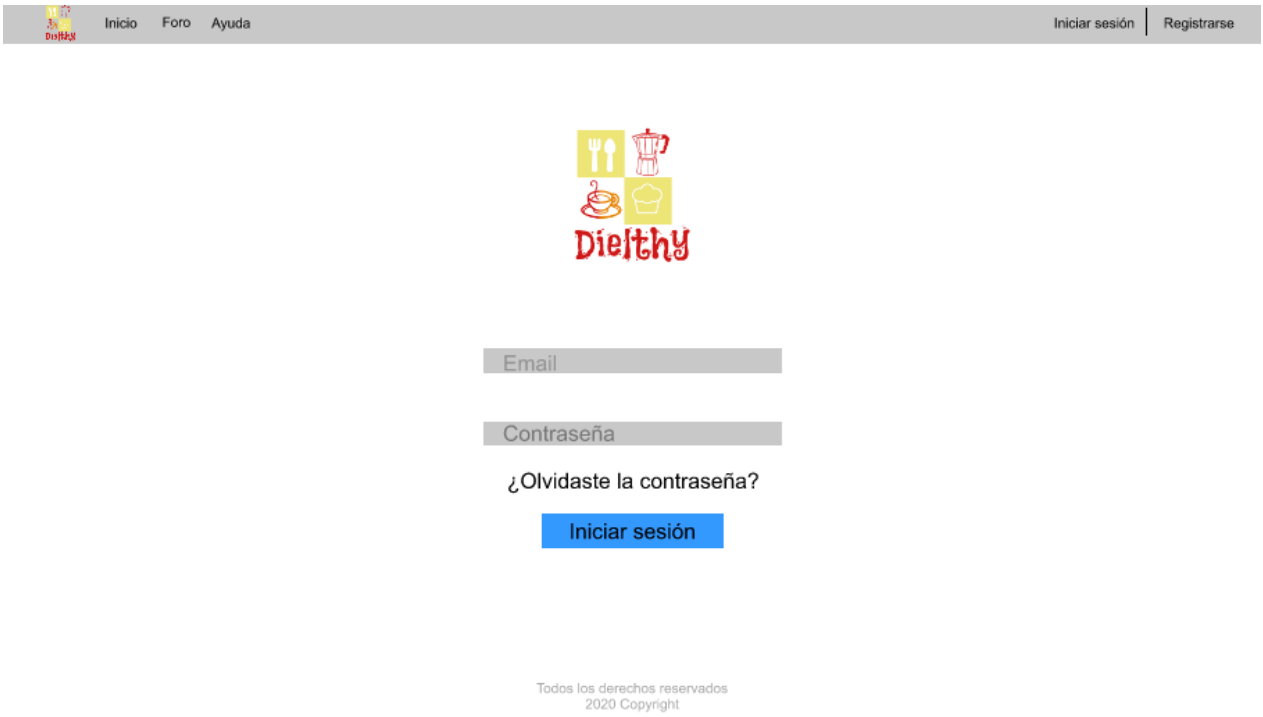

Figura 6.11: Boceto página de inicio de sesión

Figura 6.12: Boceto página de vista de perfil

Figura 6.13: Boceto página de edición de perfil

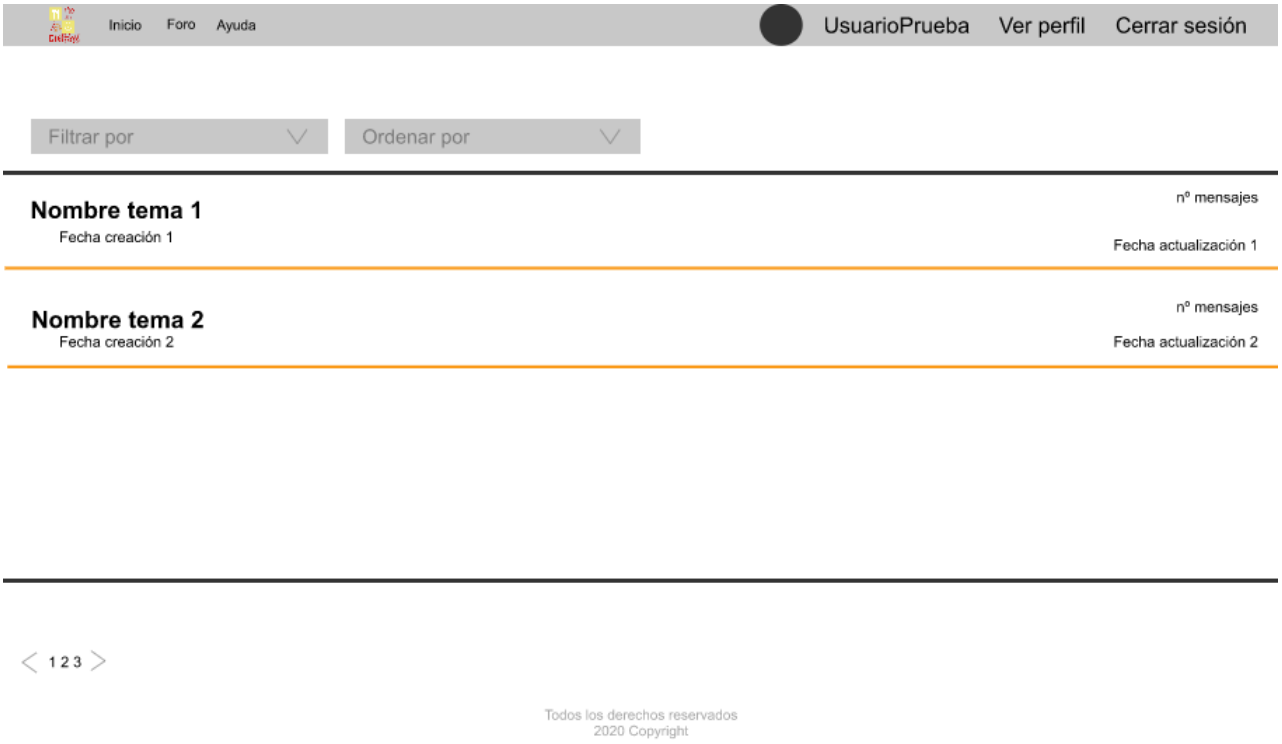

Figura 6.14: Boceto página principal del foro

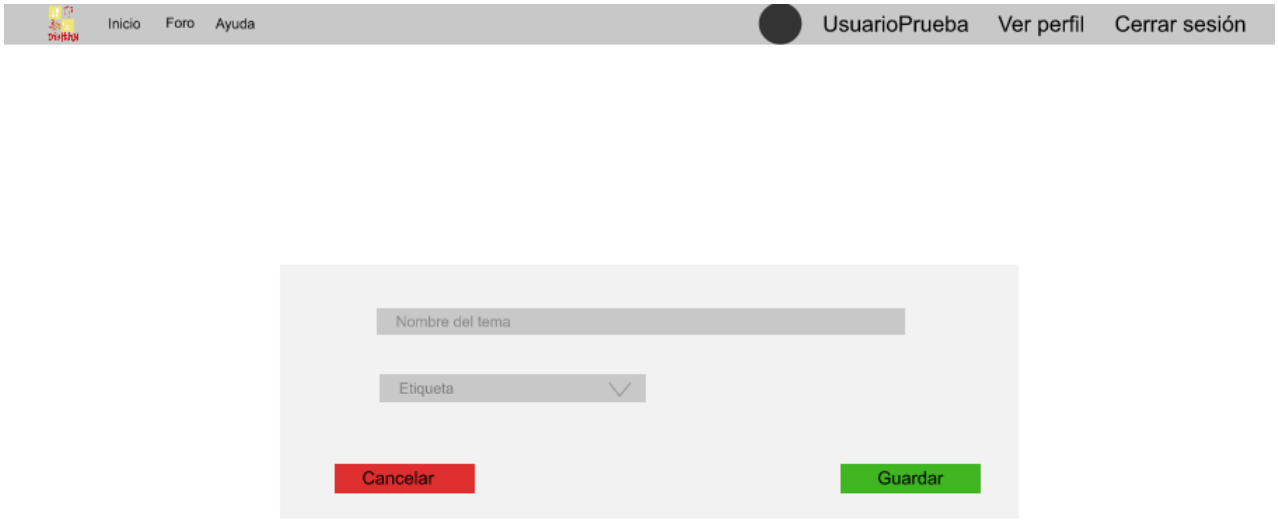

Figura 6.15: Boceto pantalla de nuevo tema

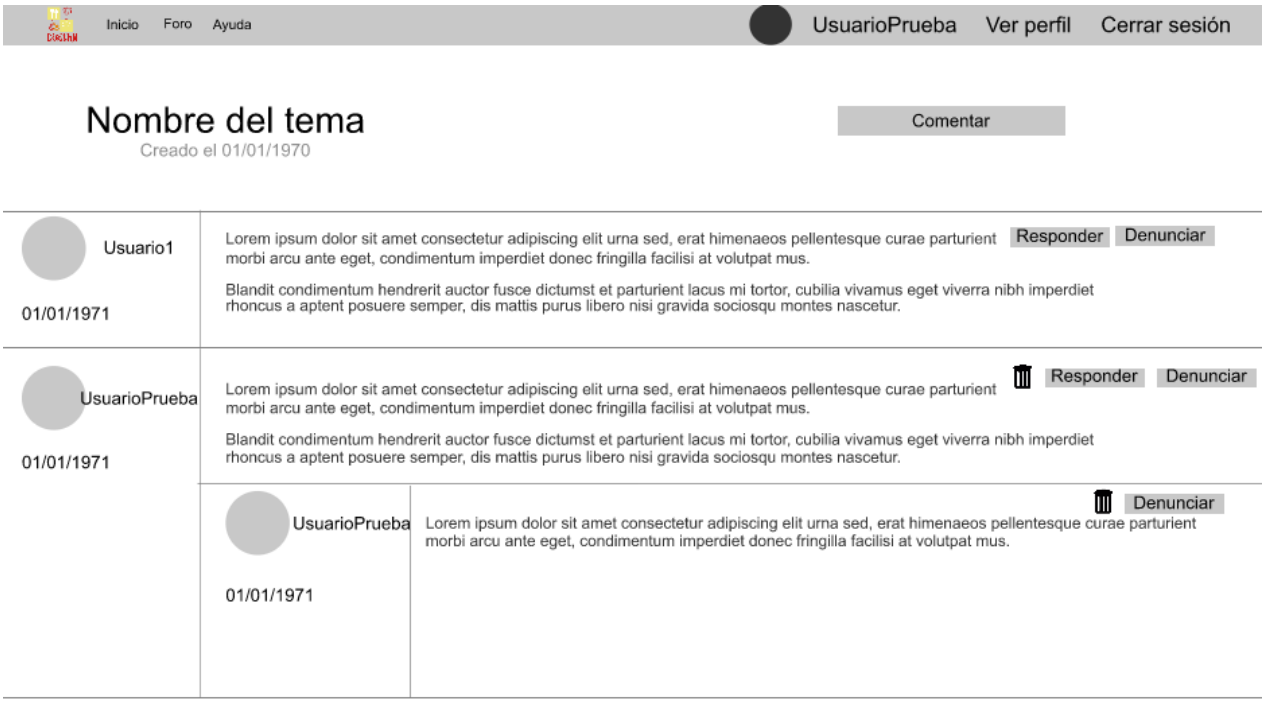

Figura 6.16: Boceto página de comentarios

Sprint 4

Durante este cuarto sprint se realizan los bocetos asociados a las historias de usuario 5, 10 y 11, que se corresponden con la adición de comentarios y respuestas al foro y toda la funcionalidad asociada a la denuncia de los comentarios (Figuras 6.16 y 6.17).

Respecto del boceto de la Figura 6.16 se debe destacar que, en la implementación final, se han sustituido los botones Denunciar y Reportar por iconos que, al pasar el ratón por encima, dejan ver dichos textos.

En la implementación final de la pantalla de reportes/denuncias (Figura 6.18) se ha añadido en la parte derecha la fecha del comentario y el desplegable del número de usuarios mostrará la información del usuario que realizó el reporte además de la fecha en la que lo efectuó.

Sprint 5

En este sprint se abordan las historias de usuario 6, 7 y 10 que se corresponden con la eliminación de comentarios, tanto por parte de un administrador como de un usuario, así como el borrado de usuarios (Figuras 6.18, 6.16 y 6.12).

El boceto de la historia de usuario 7 (eliminación de comentario por parte de un administrador) se incluye en la Figura 6.18 mediante el icono de una papelera en cada uno de los comentarios reportados.

El boceto de la historia de usuario 6 (eliminación de usuarios) queda reflejado en la Figura 6.12, en la que se añade un icono de una papelera, sólo visible para un usuario con permisos de administrador.

El boceto correspondiente a la historia de usuario 10 (eliminación de un comentario propio de un usuario) se muestra en la Figura 6.16. En ella se añade el icono de una papelera que únicamente aparecerá en los comentarios

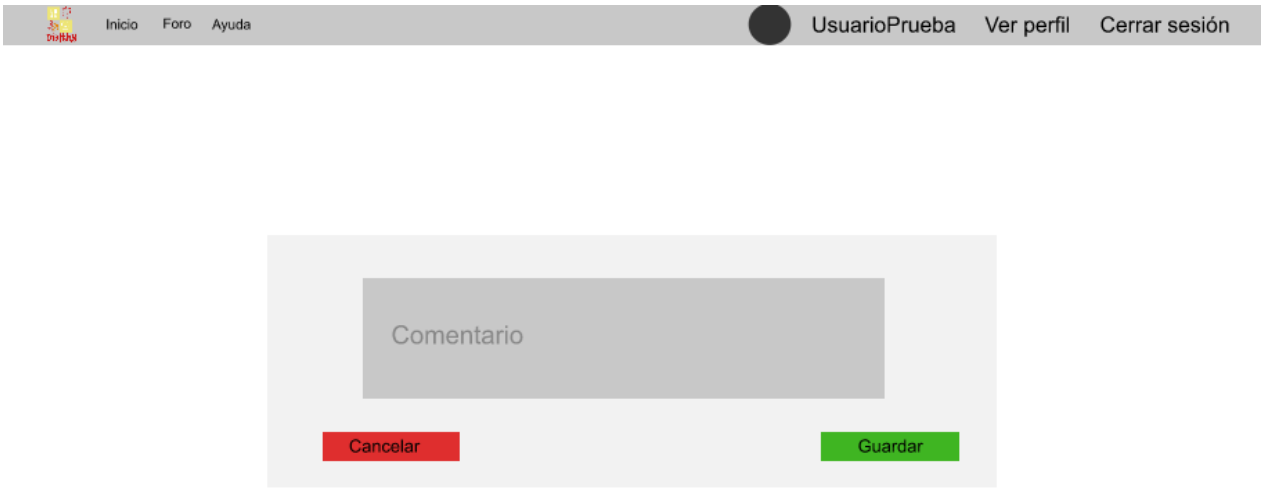

Figura 6.17: Boceto pantalla de añadir comentario/respuesta

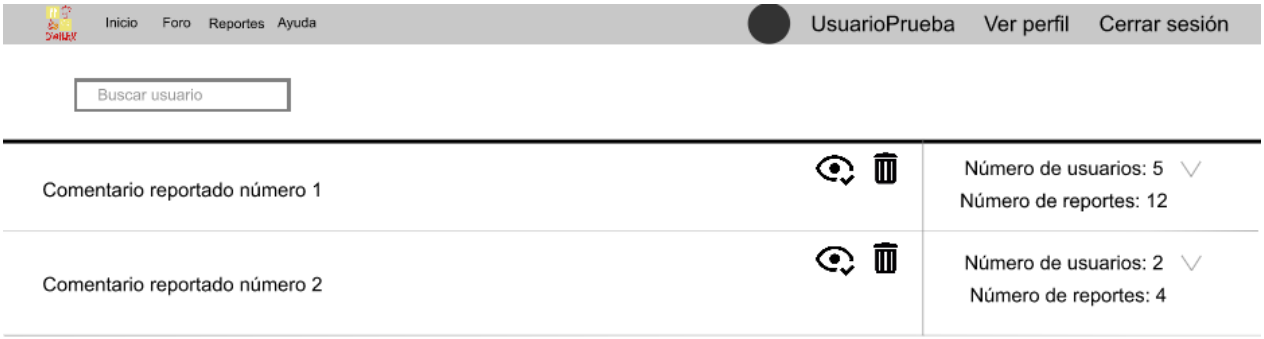

Figura 6.18: Boceto pantalla de denuncias de comentarios (perfil administrador)

que el usuario haya realizado y permitirá eliminar dicho comentario.

Sprint 6

En este sprint se abordan las historias de usuario 12, 13 y 15.

El boceto correspondiente a la historia de usuario 15 se incluye en la Figura 6.18. En ella, se añade un icono de un ojo que simboliza la acción de desestimar un reporte.

De la historia de usuario 12, correspondiente al bot de Telegram, se ha decidido no incluir bocetos debido a la complejidad que supone, pues se trata de un bot de consulta con el que simplemente se intercambian mensajes, sin elementos visuales de por medio ni disposición de éstos.

La inclusión en la historia de usuario 13 del cambio de idioma de la web causa que se introduzca un elemento en la parte superior derecha de la aplicación, un selector con las banderas de los países de los idiomas disponibles. Asimismo, el boceto del foro, en la Figura 6.14, cambia sustancialmente con respecto a la implementación final pues se añade un filtro más para buscar temas por un idioma concreto, así como el idioma del propio tema junto al mismo. El boceto de la Figura 6.17 también cambiará con respecto a la versión implementada, ya que se añade un selector que permita escoger el idioma del tema a crear.

Sprint 7

En este sprint no se abordan historias de usuario. Tan sólo las incidencias recogidas en la Tabla 8.6. Al tratarse de mejoras no excesivamente sustanciales en el diseño de la aplicación se opta por no modificar los bocetos. Sin embargó, sí merece la pena destacar los cambios que sufre la implementación final respecto de los bocetos iniciales y/o cambios posteriores realizados a lo largo del desarrollo de los sprints.

Respecto al boceto de la información del aditivo (Figura 6.9) se han cambiado las caras por iconos tachados o no, en función de si el aditivo es apto o no para cada tipo de dieta, manteniendo los colores verde o rojo. Se utilizan un trigo en el caso de los celíacos, un queso para los intolerantes a la lactosa y una hoja para los veganos y vegetarianos.

En el boceto de la Figura 6.14 se añade la hora a la fecha de actualización del tema y un botón para escoger si la ordenación será ascendente o descendente. Las etiquetas de cada tema pasan a ser múltiples, modificando el filtro correspondiente, así como el selector de la etiqueta del tipo de dieta en la modal de adición de un nuevo tema (Figura 6.17).

Se debe reseñar, asimismo, que el botón Ayuda de la barra de la parte superior de la web se decide eliminar. Aparece en todos los bocetos.

En el manual de usuario que se incluye en el Anexo A se pueden consultar las capturas de las pantallas finalmente implementadas después de todos los cambios y las mejoras introducidas sobre los bocetos iniciales.

6.6. Diseño de la base de datos

Una vez realizado el modelo conceptual del dominio de la aplicación (ver Figura 2.3) se repara en el diseño de la base de datos. Ésta será de tipo NoSQL, tal como se comentó en la Sección 3.4. Eso lleva a plantear ciertas modificaciones en el modelo conceptual. Debido a la eliminación de restricciones propias de una base de datos relacional, las entidades con multiplicidad 1..* y cuya única información almacenada sea un simple atributo pasan a convertirse en arrays. Así, el usuario mantendrá un array de tipos de dieta, tal como hará el tema con sus etiquetas. En el caso de los aditivos, relacionados también con los tipos de dieta, se ha optado por 3 atributos

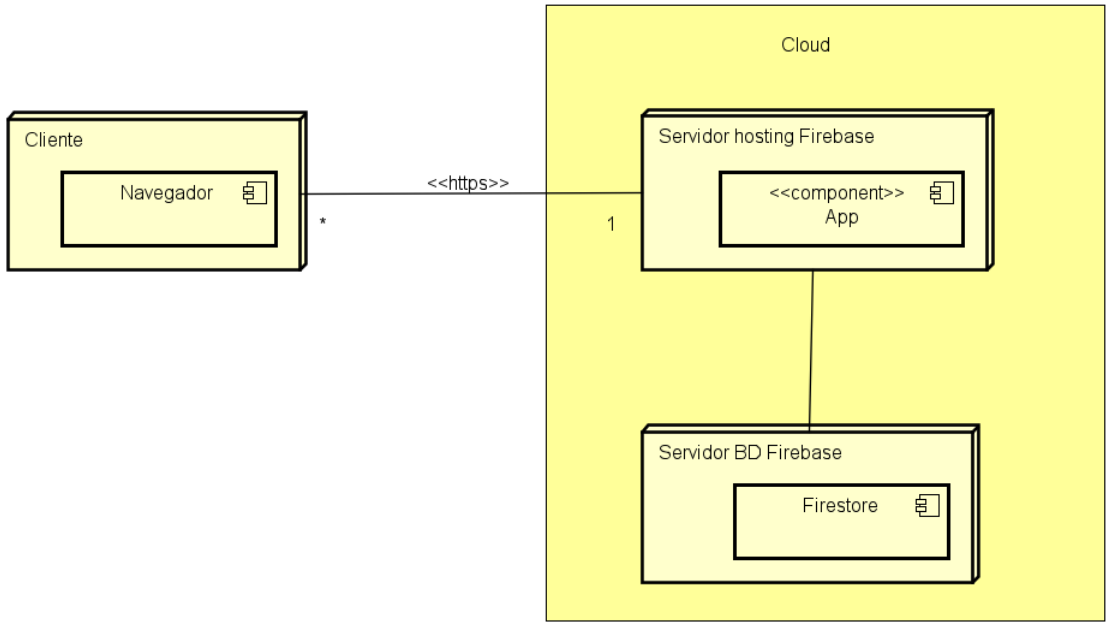

Figura 6.19: Diagrama de despliegue

booleanos, los cuales resultan más sencillos de cara al desarrollador para la implementación de la parte frontal, pues sólo se desea mostrar si un aditivo es apto o no para los tipos de dieta incluidos en la aplicación.

En cuanto a la herencia, se decide almacenar todos los atributos de la superclase en ambas subclases, pues resultará útil para mostrar la información del usuario en ambos escenarios.

La relación recursiva "respuestas" se convierte en un array de entidades "Mensaje" debido a la facilidad que supone a la hora de consultar la lista de comentarios.

Respecto a la clase asociación "denunciante" se ha tomado la decisión de incluirla junto a la entidad "Denuncia". Esto se debe a la posibilidad que ofrecen las colecciones (como se denominan las tablas en bases de datos NoSQL) de almacenar documentos (así se nombran las entidades de cada una de las colecciones) con mapas, arrays y casi todo tipo de estructuras de datos complejas. De esta forma, se facilitan las consultas y se ahorra la creación de una colección denunciante.

La aplicación de todas estas decisiones y cambios puede verse en la Sección 7.6.

6.7. Modelo de despliegue de la web

El despliegue se realizará empleando el servicio de hosting que ofrece Firebase, sobre el que se hospedarán todos los archivos necesarios para el correcto funcionamiento de la web. Como ya se comentó en la Sección 3.4, la base de datos utilizada será Firestore, ofrecida también por Firebase. Esto se alojará en la nube, en servidores completamente desconocidos para el desarrollador. A pesar de ello, en la Figura 6.19 puede verse una estructura aproximada del despliegue que realizarán los propios servidores de la nube.

6.8. Diseño del bot de Telegram

En el proceso de diseño del bot de Telegram se busca una gran sencillez, del mismo modo que la web. Los bots de Telegram pueden prepararse para recibir cualquier tipo de mensaje pero, para determinadas acciones en particular, es conveniente indicar comandos. Éstos consisten en una cadena alfanumérica precedida de /. En el bot que nos atañe se han planteado varios comandos:

- `/start`. En los bots, en la primera conversación con estos, Telegram proporciona un botón en la parte inferior cuyo click envía automáticamente este comando. En el bot que se pretende realizar tan sólo dará la bienvenida.
- `/help`. Comando básico de ayuda que informará del correcto uso del bot.
- `/e` seguido de un código de aditivo. Proporciona la información de un aditivo. Muestra la misma información que se puede apreciar en la Figura 6.9, sólo que se hará a través de un mensaje simple en la conversación.

Este bot se desplegará empleando el servicio de functions de Firebase. Esto se debe al gran peso que se otorga en este proyecto a las plataformas serverless y a que no se dispone de un medio propio en el que desplegar este código. Además, permite aprovechar lo máximo posible los servicios que proporciona Firebase al proyecto creado. La Figura 6.20 muestra un diagrama aproximado (pues se desconoce la estructura que posee Firebase en la nube) de despliegue del bot.

Functions admite como entornos Node.js (lenguaje JavaScript), Python (lenguaje Python) y Go (lenguaje Go). Dado que para la web se emplea JavaScript, este caso no será una excepción y se utilizará Node.js como entorno de ejecución. También se apoyará en el uso de Express, una infraestructura que actúa como middleware facilitando la labor del desarrollador.

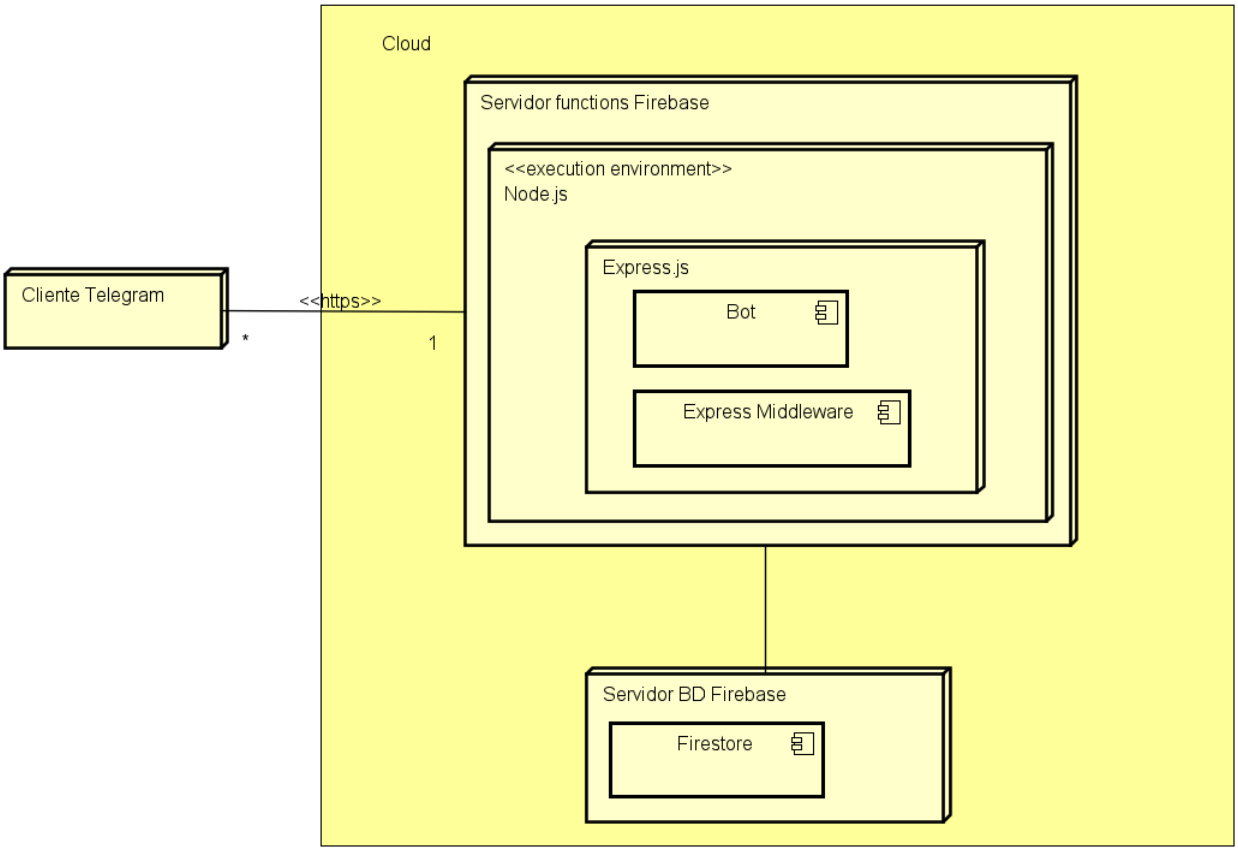

Figura 6.20: Diagrama de despliegue del bot de Telegram

Capítulo 7

Implementación

En este capítulo se muestra la estructura final del código, tanto del bot como de la web. También se detalla el proceso de integración continua, así como las decisiones tomadas durante el desarrollo del proyecto y los cambios que han supuesto. En último término, se describe la licencia que se ha optado por conferir al mismo.

7.1. Estructura del código de la web

A continuación se proporciona la organización básica final del código.

```
|   .babelrc
|   .browserslistrc
|   .eslintrc.js
|   .firebaserc
|   .gitignore
|   babel.config.js
|   cypress.json
|   firebase.json
|   jest.setup.js
|   nyc.config.js
|   jest.config.js
|   package.json
|   README.md
|
+---public
|   index.html
|   logo-dielthy.png
|
+---src
|   |   App.vue
|   |   main.js
|   |
|   +---api (se omite su contenido por brevedad)
|   |
|   +---assets  (se omite su contenido por brevedad)
|   |
```

```
|   +---components  (se omite su contenido por brevedad)
|   |
|   +---lang
|   |   |   i18n.js
|   |   |
|   |   +---locale
|   |   |       en.js
|   |   |       es.js
|   |
|   +---plugins
|   |       firebase.js
|   |       firebaseConfig.js
|   |       vuetify.js
|   |
|   +---router
|   |       index.js
|   |
|   +---store
|   |   |   index.js
|   |   |
|   |   \---modules (se omite su contenido por brevedad)
|   |
|   +---utils
|   |       sharedFunctions.js
|   |
|   +---views   (se omite su contenido por brevedad)
|   |
+---tests
    +---e2e (se omite su contenido por brevedad)
    |
    +---unit    (se omite su contenido por brevedad)
```

Los primeros ficheros son de configuración de Node, despliegue de Firebase, Cypress, Jest... y el README. También se incluye un fichero `.gitignore` que permite excluir ciertos directorios en la subidas que se realicen al repositorio remoto de Git, como son `node_modules` (módulos y dependencias del proyecto) o `dist` (véase el Apéndice B.2).

- La carpeta **public** contiene el fichero index que se utiliza como punto de entrada a la aplicación y el logo que se utiliza en la barra del navegador.
- En **src** se alojan todos los archivos del código que ejecuta la aplicación. Consta de:
  - **api**. Carpeta que contiene los archivos que realizan llamadas a la base de datos y no requieren el uso del estado global de la aplicación.
  - **assets**. Contiene los archivos estáticos que se usan a lo largo de la aplicación, como las imágenes.
  - **components**. En esta carpeta se encuentran los componentes reutilizables de la aplicación, aquellos que se utilizan en bucles o que son susceptibles de ser usados más de una vez.
  - **lang**. Constituye el conjunto de todos los ficheros necesarios para la internacionalización de la web.
  - **plugins**. Incluye ficheros de configuración generales para plugins que se utilizan en la aplicación al completo, tales como Firebase o la librería de componentes Vuetify.

  - **router**. Contiene los archivos necesarios para gestionar el paso entre las distintas rutas de la web.
  - **store**. Aglutina todo lo necesario para administrar el estado global de la aplicación mediante Vuex, tanto las estructuras de datos que almacena como la modificación de éstas.
  - **utils**. Directorio creado para contener ficheros de utilidades, como funciones o constantes, comunes a los distintos componentes.
  - **views**. Almacena todos los componentes de un solo uso, generalmente aquellos que son usados directamente por las rutas especificadas para la aplicación.

- En **tests** se encuentran todos los archivos con el código de los tests automáticos que se ejecutan para cada una de las historias de usuario contempladas o los componentes desarrollados. Se divide en **e2e** y **unit**, que contienen los tests end-to-end y unitarios, respectivamente.

7.2. Estructura del código del bot de Telegram

A continuación se proporciona la estructura básica del código del bot de Telegram.

```
|   .firebaserc
|   .gitignore
|   firebase.json
|   package.json
|
+---functions
|   |   .gitignore
|   |   index.js
|   |   package.json
```

Los primeros ficheros son de configuración de Firebase y Node, además del `.gitignore` para indicar qué archivos no debe incorporar Git al repositorio.

En la carpeta **functions** el archivo `index.js` contiene el código en sí que regula el funcionamiento del bot.

7.3. CI/CD

Una vez realizados los desarrollos se pasó a automatizar el proceso de testing y despliegue mediante la herramienta de integración continua de GitLab, cuya explicación puede verse en la Sección 3.1.1. Lo ideal sería haber realizado esta tarea al inicio pero, como se comenta en la Sección 5.1.2, se deja aparcada tras intentar completarla sin éxito por considerarla no prioritaria. Cabe destacar que únicamente se realiza integración continua del código empleado para la web, no para el bot de Telegram. También es importante reseñar que la integración continua en el repositorio público de GitLab no funciona puesto que el archivo `firebaseConfig.js` no contiene la información necesaria para iniciar adecuadamente la base de datos, lo cual hará que existan errores en todo tipo de tests, así como el despliegue. Los motivos que llevan a la ocultación de los datos de dicho fichero se encuentran explicados en el Apéndice B.2.

A continuación se muestra la apariencia final del archivo YAML:

```
stages:
  - test
  - deploy

cache:
  paths:
    - node_modules/

test_unit:
  image: node:latest
  stage: test
  script:
    - npm install --progress=false
    - npm run test:unit
  only:
    - develop

test_e2e:
  image: node:latest
  stage: test
  script:
    - apt-get update -yqqq
    - apt-get install -y libgtk2.0-0 libgtk-3-0 libnotify-dev libgconf-2-4 libnss3 libxss1 libasound2
    libxtst6 xauth xvfb fonts-liberation libappindicator3-1 xdg-utils -y
    - wget http://dl.google.com/linux/direct/google-chrome-stable_current_amd64.deb
    - dpkg -i google-chrome-stable_current_amd64.deb; apt-get -fy install
    - Xvfb :99 &
    - export DISPLAY=:99
    - npm install --progress=false
    - npm run test:e2e-ci
  only:
    - master

deploy_production:
  image: node:latest
  stage: deploy
  script:
    - npm install -g firebase-tools
    - npm install
    - npm run build
    - firebase use --token $FIREBASE_DEPLOY_KEY default
    - firebase deploy --non-interactive --token $FIREBASE_DEPLOY_KEY
  only:
    - master
```

En los apartados siguientes se explica el contenido de este archivo YAML para GitLab CI/CD.

7.3.1. Preparación

Al inicio del archivo YAML aparecen las fases por las que pasará la ejecución de la integración continua. Esto puede verse en:

```
stages:
  - test
  - deploy
...
```

Posteriormente, se incluye lo siguiente:

```
...
cache:
  paths:
    - node_modules/
...
```

Esto indica a la herramienta que debe mantener en caché las rutas especificadas y elegir esa copia si se encuentra disponible. Esto permite que módulos utilizados no se tengan que descargar una y otra vez.

En cada uno de los trabajos puede apreciarse lo siguiente:

```
...
    image: node:latest
...
```

Esto permite especificar la imagen docker que debe utilizar GitLab para desplegar el entorno de integración continua. En este caso se elige el contenedor de node más reciente posible, lo que posibilita la ejecución de scripts npm.

7.3.2. Testing

Dentro de esta fase se incluye una subfase o trabajo por cada uno de los dos tipos de tests que se realizan en el proyecto: unitarios y end-to-end.

Con respecto a los tests unitarios, el trabajo para su realización es el siguiente:

```
...
test_unit:
  image: node:latest
  stage: test
  script:
    - npm install --progress=false
    - npm run test:unit
  only:
    - develop
...
```

Además de la elección de la imagen docker, también se especifica la fase en la que debe ejecutarse el trabajo y con lo siguiente puede seleccionarse en qué ramas se hará.

```
...
    only:
        - develop
...
```

En este caso sólo será en `develop`. Esto se debe a la consideración de que los tests unitarios son bastante rápidos y, ya que las fusiones a la rama `develop` son bastante habituales, su ejecución permitirá observar en apenas unos pocos minutos si alguna funcionalidad ha sido alterada sin darse cuenta.

La ejecución del trabajo en sí consta de:

```
...
    script:
        - npm install --progress=false
        - npm run test:unit
...
```

Esto permite instalar todas las dependencias del proyecto sin mostrar su progreso y ejecutar los tests unitarios como tal. El segundo de los comandos puede verse en el Apéndice B.4.

En referencia a los tests end-to-end, el trabajo que ordena su ejecución es el siguiente:

```
...
test_e2e:
  image: node:latest
  stage: test
  script:
    - apt-get update -yqqq
    - apt-get install -y libgtk2.0-0 libgtk-3-0 libnotify-dev libgconf-2-4 libnss3 libxss1 libasound2
    libxtst6 xauth xvfb fonts-liberation libappindicator3-1 xdg-utils -y
    - wget http://dl.google.com/linux/direct/google-chrome-stable_current_amd64.deb
    - dpkg -i google-chrome-stable_current_amd64.deb; apt-get -fy install
    - Xvfb :99 &
    - export DISPLAY=:99
    - npm install --progress=false
    - npm run test:e2e-ci
  only:
    - master
...
```

Como en el caso de los tests unitarios, lo primero que se observa, además de la selección de imagen docker, es la fase en la que se realiza dicho trabajo. Al final del mismo puede verse que la rama en la que se ejecutarán estos tests será únicamente `master`. Dicha elección se debe a su lentitud en la ejecución y su gran dependencia de la base de datos, y a que las fusiones sobre esta rama son poco frecuentes. El trabajo como tal consta de:

```
...
     script:
```

```
        - apt-get update -yqqq
        - apt-get install -y libgtk2.0-0 libgtk-3-0 libnotify-dev libgconf-2-4 libnss3 libxss libasound2
        libxtst6 xauth xvfb fonts-liberation libappindicator3-1 xdg-utils -y
        - wget http://dl.google.com/linux/direct/google-chrome-stable_current_amd64.deb
        - dpkg -i google-chrome-stable_current_amd64.deb; apt-get -fy install
        - Xvfb :99 &
        - export DISPLAY=:99
        - npm install --progress=false
        - npm run test:e2e-ci
...
```

Lo primero que se observa es la actualización e instalación de ciertas librerías de dependencias, en especial Xvfb, que permite mantener en ejecución un navegador sin mostrar nada por consola y sin interfaz gráfica:

```
...
         - apt-get update -yqqq
         - apt-get install -y libgtk2.0-0 libgtk-3-0 libnotify-dev libgconf-2-4 libnss3 libxss1
         libasound2 libxtst6 xauth xvfb fonts-liberation libappindicator3-1 xdg-utils -y
...
```

Con posteridad, se instala una versión del navegador de Google Chrome integrado en consola:

```
...
        - wget http://dl.google.com/linux/direct/google-chrome-stable_current_amd64.deb
        - dpkg -i google-chrome-stable_current_amd64.deb; apt-get -fy install
...
```

Más tarde, se crea una instancia de un navegador empleando Xvfb en un display en particular y se indica a la consola que dicho display será el principal:

```
...
        - Xvfb :99 &
        - export DISPLAY=:99
...
```

Después se instalan las dependencias del proyecto y se ejecutan los tests end-to-end como tal:

```
...
        - npm install --progress=false
        - npm run test:e2e-ci
...
```

7.3.3. Despliegue

El trabajo que permite el despliegue del artefacto de la web, cuyo proceso de obtención puede verse en el Apéndice B.2, es el siguiente:

```
...
deploy_production:
  image: node:latest
  stage: deploy
  script:
    - npm install -g firebase-tools
    - npm install
    - npm run build
    - firebase use --token $FIREBASE_DEPLOY_KEY default
    - firebase deploy --non-interactive --token $FIREBASE_DEPLOY_KEY
  only:
    - master
...
```

Como en los anteriores trabajos, lo primero que se observa es la imagen docker y la fase en la que se efectúa dicho trabajo. Al final se aprecia que la única rama en la que se hará será `master`. La ejecución de éste se realiza mediante:

```
...
        - npm install -g firebase-tools
        - npm install
        - npm run build
        - firebase use --token $FIREBASE_DEPLOY_KEY default
        - firebase deploy --non-interactive --token $FIREBASE_DEPLOY_KEY
...
```

Inicialmente se instala la herramienta Firebase CLI, necesaria para realizar el despliegue. Después se instalan todas las dependencias y se genera el artefacto mediante `npm run build`. Tras ello, se indica a Firebase que debe usar cierto token para realizar todas las operaciones posteriores e iniciar sesión en la cuenta a la que pertenece dicho token. En este caso se ha definido en Gitlab una variable de entorno que contiene el valor de un token generado previamente para poder acceder al proyecto en Firebase. Por último, se ejecuta `firebase deploy` con ciertos parámetros que indican el token y que este despliegue debe ser realizado sin interfaz interactiva. Esto es una simplificación del proceso descrito más detalladamente en el Apéndice B.2.

7.4. Decisiones tomadas a lo largo del proyecto

Durante la implementación del proyecto ha sido necesario tomar multitud de decisiones, de las que se resaltarán las más importantes a continuación.

Como plataforma para el desarrollo de la base de datos y la lógica *Back-end* se ha escogido Firebase, ya explicado en la Sección 6.2. Esto se debe a la multitud de servicios que ofrece, a su flexibilidad y a su facilidad de uso, pues se pretende que en este proyecto la mayor carga de trabajo se encuentre en la parte *Front-end*. Además, por el simple hecho de pertenecer a Google y encontrarse bajo la plataforma Google Front End, Firebase posee numerosas herramientas de detección de brechas de seguridad, ataques DDoS y protección frente a otro tipo de amenazas que podría sufrir la aplicación [65] [66]. Esta seguridad existe bajo el nombre de Google Cloud Armor [64].

Con respecto a la forma de acceso a la base de datos, Firebase provee tanto de una API como de un SDK propio. Se ha optado por este segundo método, puesto que su facilidad de uso es más que evidente, dejando que

Firebase gestione por sí mismo las llamadas a las URL adecuadas. Basta, entonces, con preocuparse únicamente por mantener ciertas referencias a la base de datos en la capa frontal y ejecutar ciertos métodos provistos por el SDK.

En relación a la forma de almacenamiento local empleado para la aplicación se ha decidido utilizar el estado global de la aplicación para compartir información entre componentes. Este estado se gestiona a través de Vuex, explicado en la Sección 3.3.2. Asimismo, se emplea también el estado de cada componente para datos que no requieren ser compartidos. Existe, sin embargo, una excepción, pues se ha optado también por hacer uso del almacenamiento de sesión del navegador. Esto se debe a que resulta la forma más sencilla encontrada para mantener la información asociada al permiso que posee el usuario en la web, teniendo por objetivo el control de las rutas a las que el usuario está autorizado a acceder.

7.5. Cambios en implementación de los diseños iniciales

A lo largo de la realización de cada una de las historias de usuario, en el momento de implementarlas, surgieron determinadas circunstancias o se observaron ciertas características que hicieron evolucionar los diseños iniciales. Estos cambios se relatan en la Sección 6.5.1 en cada uno de los sprints. Pueden verse imágenes de la apariencia final de la web en el Apéndice A.

7.6. Cambios en implementación del modelo conceptual

En este proyecto, como se comentó en la Sección 3.4, se emplea Firestore, una base de datos NoSQL. Eso ocasiona que, en implementación, el modelo conceptual sufra determinados cambios. Esto se debe a que no se trata de una base de datos relacional al uso sino que consta de estructuras de datos, llamadas colecciones, donde ciertas restricciones relacionales, como, por ejemplo, los grupos repetitivos, pueden ser eliminadas. La consecuencia directa de esto es una mayor facilidad de implementación para el desarrollador. Los cambios efectuados en el modelo conceptual del dominio (Figura 2.3) pueden observarse en la Figura 7.1. Ésta muestra las entidades que se han mantenido como colecciones, los atributos surgidos de eliminar las relaciones del modelo conceptual (marcados en rojo) y aquellos que se añaden para facilitar la implementación de la parte *Front-end* (marcados en azul). Una descripción más detallada de las decisiones tomadas para la base de datos puede encontrase en la Sección 6.6.

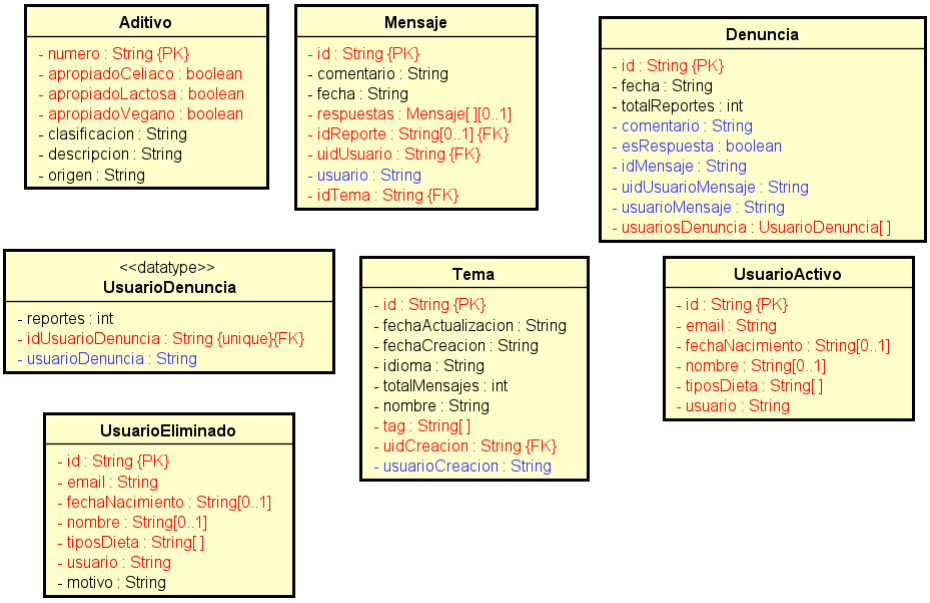

Figura 7.1: Cambios del modelo conceptual en implementación para una base de datos NoSQL.

7.7. Licencia

Este proyecto se ha dotado de una licencia Creative Commons. De entre las que ofrece esta plataforma se ha optado por una de tipo Atribución-NoComercial-SinDerivadas [67]. Según la misma, se permite la copia y distribución del material en cualquier medio o formato, por lo que el código será público. Sin embargo, no se autoriza el uso comercial de la aplicación ni la creación, remezcla o transformación (y posterior distribución) en base a ella. Del mismo modo, se debe brindar crédito de manera adecuada y razonable, un enlace a la licencia y si se han realizado modificaciones.

Capítulo 8

Testing

8.1. Introducción

El testing en software [57] es la realización de pruebas que nos lleva a obtener información acerca de su calidad. Sus principales objetivos son:

- Localización y subsanación de errores.
- Aumento del nivel de confianza en el software desarrollado.
- Disminución de la aparición de nuevos defectos.

Estas pruebas se pueden clasificar según determinadas condiciones:

- Según la forma de interactuar con el código:
  - Estáticas: se realizan sin necesidad de ejecutar el código, tan sólo mediante su lectura y comprensión.
  - Dinámicas: necesitan ejecutar el código.
- Según la forma de interactuar con el software:
  - Manuales: requieren de una navegación normal de un usuario.
  - Automáticas: se emplean herramientas que simulan el comportamiento de un usuario.
- Según lo que verifican:
  - Funcionales: prueban características de funcionamiento del software. Se subdividen en:
    - Unitarias: pequeños fragmentos de código son probados de forma aislada.
    - Integración: prueban varios fragmentos de código juntos para comprobar su correcta interacción.
    - End-to-end: prueban el flujo completo desde el punto de vista del usuario final.
    - Aceptación: enfocadas a la funcionalidad completa del software. Suele ser el usuario quien las realiza.
  - No funcionales: se centran en características que no tienen que ver con la funcionalidad, como son la seguridad, el rendimiento, la accesibilidad, la usabilidad... Las pruebas de aceptación de usuarios tienen una componente funcional y otra no funcional, ya que con el usuario se puede obtener información acerca de la aceptación de la usabilidad, el rendimiento, etc.

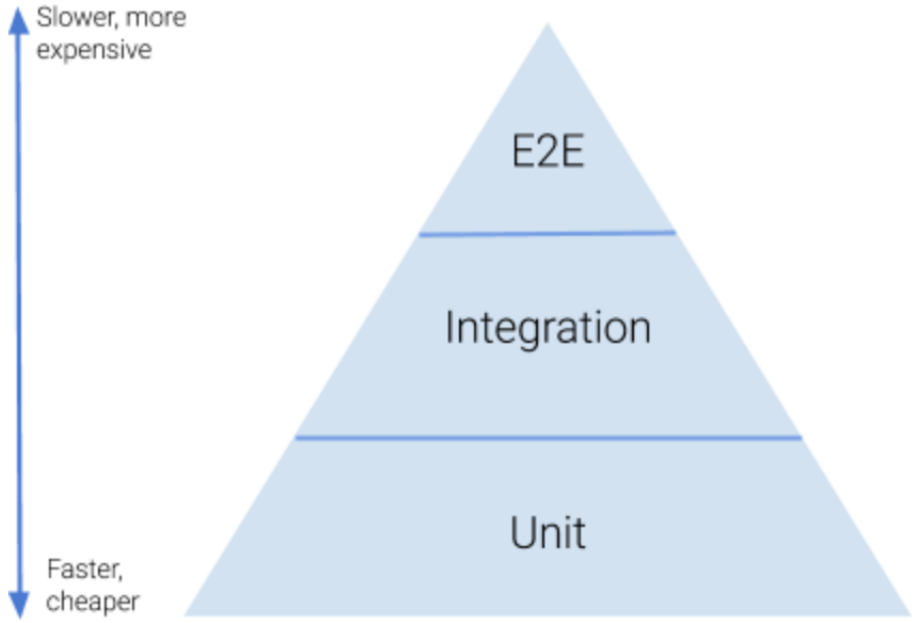

Figura 8.1: Grupos de tests software funcionales. Tomada de [58]

En este proyecto, teniendo presente que no se cuenta con un usuario final y que la mayor carga de trabajo se encuentra en la parte frontal de la aplicación, se decide incluir pruebas o tests dinámicos, automáticos y funcionales únicamente en *Front-end*. A modo de aclaración acerca del testing puede consultarse la Figura 8.1, en la que se muestra la jerarquización de las pruebas funcionales realizadas en este proyecto. Con su incorporación se busca mantener un control de la creciente funcionalidad de la aplicación, pudiendo comprobar de forma rápida si la introducción de algún fragmento de código causa que el software desarrollado anteriormente falle y en qué punto lo hace.

Pese a que las pruebas realizadas son automáticas y codificadas por el propio equipo de desarrollo, se decide además realizar pruebas con usuarios reales que simularían ser los clientes finales de la aplicación, quienes deben valorar si se cumplen las expectativas y las funcionalidades acordadas. Este tipo de pruebas son manuales y se explican en la Sección 8.5. Además, debido a la dificultad que existe para la creación de tests automatizados que prueben la funcionalidad del bot de Telegram desarrollado, se deciden realizar tests de sistema, explicados en la Sección 8.4. También se tratará de pruebas manuales y funcionales.

8.2. Tests unitarios

En este proyecto los tests unitarios se realizan empleando la herramienta Jest, ya comentada en la Seccion 3.3.1. Son tests de rápida ejecución, cuyo desarrollo resulta en cierta medida costoso. Esto es debido, no tanto a la complicación en el uso de Jest, sino más bien a la dificultad que se puede extraer de la propia realización de tests en *Front-end*, pues resulta difícil enfocar adecuadamente unas pruebas que cubran el 100 % del código en un ámbito tan visual. Pese a ello, se han realizado todos los tests posibles que se han considerado necesarios para la verificación del funcionamiento correcto de la aplicación. Estos tests se agrupan en suites, las cuales son grupos de pruebas enfocadas a comprobar cierto comportamiento. Se ha intentado crear una suite por cada uno de los componentes de la aplicación (en la medida de lo posible, pues algunos no requieren ningún tipo de test), si bien no siempre ha sido así, pues en algunas historias de usuario no se han creado nuevos componentes y se ha diseñado una suite para

dicha historia de usuario que prueba en aislamiento el componente afectado. En todos los tests se emplean mocks u objetos de prueba a fin de mantener el control sobre los datos de los que dispone cada componente, ya que en este tipo de tests no se interacciona con la base de datos.

A la finalización del proyecto se cuenta con 63 tests agrupados en 12 suites. El resumen de la cobertura de estos tests puede verse en la Figura 8.2. Se puede apreciar que ésta ronda el 35 %-40 %. Parece una cobertura un tanto baja pero existen varios componentes que no pudieron ser probados debido a que se trataba de ventanas modales de diálogo, las cuales estaban integradas en otros componentes y no fue posible las realización de sus tests unitarios. Además, debe tenerse en cuenta, como ya se ha comentado previamente en esta sección, que resulta difícil diseñar tests que cubran el 100 % del código desarrollado en la parte frontal de la aplicación, pues está dotado de características visuales que no pueden ser probadas de forma unitaria. Se ha considerado que, con los tests generados, se pueden observar fallos en las principales funcionalidades, por lo que la cobertura, pese a encontrarse incluso por debajo del 50 %, resulta adecuada.

Figura 8.2: Cobertura de los tests unitarios

A modo de ejemplo puede aportarse la historia de usuario 1, correspondiente a la búsqueda de un aditivo alimentario por su código. Para ella se crean dos suites, una por cada uno de los componentes desarrollados para la misma. Se trata de AditiveInfo.vue y Main.vue, explicados en la Sección 6.4.3. Los tests realizados para cada suite son:

- Suite Main.vue
  - Mostrar correctamente mensaje de bienvenida a la web.
  - Añadir al input de la vista 'E-' al escribir texto en él.
  - Borrar todo el texto del input una vez que el usuario elimina todos los caracteres excepto 'E-'.
  - Mostrar un mensaje de error al buscar un aditivo con el input vacío.
  - Realización de la llamada al método para buscar un aditivo cuando el input contiene texto.
- Suite AditiveInfo.vue
  - Renderizado correcto de las etiquetas y textos (aditivo de origen vegetal, apto para celíacos, intolerantes a la lactosa y veganos).
  - Renderizado correcto en caso de aditivo de origen animal.
  - Renderizado correcto en caso de aditivo de origen químico o síntético.
  - Renderizado correcto en caso de aditivo de origen indeterminado.
  - Renderizado correcto en caso de aditivo no apto para intolerantes a la lactosa.
  - Renderizado correcto en caso de aditivo no apto para veganos.
  - Renderizado correcto en caso de aditivo no apto para celíacos.

8.3. Tests end-to-end

En este proyecto, los tests end-to-end incluyen, a su vez, los tests de integración. Se encuentran muy enfocados a probar el funcionamiento global de la aplicación, simulando incluso la interacción del usuario con los componentes.

Figura 8.3: Cobertura tests end-to-end

Se ha decidido crear, en la medida de lo posible, una suite por cada una de las historias de usuario abordadas en el desarrollo del aplicativo.

La implementación de estos tests se realiza mediante la herramienta Cypress, comentada en la Sección 3.3.4. Ésta permite la creación de un contenedor en el que se ejecuta un navegador web, tal como lo haría un usuario final, donde se despliega la aplicación y se interacciona con ella de forma automatizada. Es por ello que estos tests son muy lentos, dependientes de la conexión a Internet y del funcionamiento de los servicios, pues prueban también la interacción entre el *Front-end* y el *Back-end*. Esto último genera diversos problemas de acoplamiento de datos ya comentados en algunos sprints del Capítulo 5, e incluso que algunos tests sólo puedan ser ejecutados una vez con éxito. Lo ideal para subsanar esta situación sería disponer de entornos y bases de datos diferentes (desarrollo, preproducción y producción) y aisladas entre sí, de modo que los desarrolladores tengan siempre control sobre los datos almacenados en la base de datos del entorno de desarrollo y puedan adaptar los tests a estos datos. En el caso de este proyecto no se dispone de esta característica, por lo que muchas pruebas end-to-end han debido ser sustituidas por tests unitarios que permitieran evitar este acoplamiento.

Al término del proyecto se cuenta con 30 tests agrupados en 14 suites diferentes. Puede verse un resumen de la cobertura de los tests end-to-end en la Figura 8.3. Se puede apreciar que se sitúa en torno al 50 %, lo cual resulta bastante adecuado teniendo en cuenta que numerosos componentes no pudieron ser probados por completo (e incluso algunas historias de usuario no poseen tests end-to-end) debido, como se ha comentado previamente en esta misma sección, al gran acoplamiento existente entre la base de datos y los tests. Se ha considerado que cumplen con su función, la cual es detectar errores en alguna de las funcionalidades principales implementadas.

Continuando con el ejemplo de la historia de usuario 1 se citan a continuación todos los tests end-to-end realizados para la misma:

- Búsqueda de un aditivo correcto.
- Búsqueda de un aditivo con el texto del input vacío.
- Búsqueda de un aditivo con el texto del input vacío y comprobación de la eliminación del error al escribir sobre el input.
- Búsqueda de un aditivo y vuelta a la página principal.
- Redirección a la página principal cuando se recarga la página de la información de un aditivo.
- Búsqueda de un aditivo que no existe.

8.4. Tests de sistema

Los tests de sistema [63] son aquellos en los que se prueba el sistema completo desplegado en su entorno real de explotación. Se encuentran a medio camino entre los tests de integración y de aceptación de usuario.

En este proyecto tan sólo se emplean para realizar pruebas sobre el bot de Telegram creado para la consulta de aditivos. Para ello, se diseña un banco de pruebas, que muestra una descripción de la cadena de texto probada en cada test, el resultado esperado y si la prueba resulta exitosa o no. Puede verse un resumen en la Tabla 8.1.

Descripción	Resultado esperado	Resultado
Cadena “Hola”	“Para consultar cómo utilizar el bot envía /help”	Éxito
Cadena “/help”	“Para comenzar a usar el bot teclea el comando /e seguido de un espacio y el código del aditivo que deseas consultar”	Éxito
Cadena “/e”	“No se ha especificado un código de aditivo”	Éxito
Cadena “/start”	“Bienvenido al bot de Dielthy. Aquí podrás consultar la información de los aditivos”	Éxito
Cadena “/e 100”	“Código: E-100 Clasificación: Colorante Descripción: Curcumina Origen: Vegetal Apto para celíacos: Sí Apto para veganos/vegetarianos: Sí Apto para intolerantes a la lactosa: Sí”	Éxito
Cadena “/e100”	“Código: E-100 Clasificación: Colorante Descripción: Curcumina Origen: Vegetal Apto para celíacos: Sí Apto para veganos/vegetarianos: Sí Apto para intolerantes a la lactosa: Sí”	Éxito

Tabla 8.1: Banco de pruebas para el bot de Telegram

8.5. Tests UAT

8.5.1. Contextualización

Al final de este proyecto, una vez finalizados los desarrollos previstos inicialmente, se realizan pruebas con usuarios reales, simulando la fase denominada UAT (User Acceptance Testing, Pruebas de Aceptación de Usuario) [60]. Ésta se realiza con posterioridad a otro tipo de tests, antes de la salida a producción del software, y tiene por objetivo determinar la completitud de las funcionalidades esperadas y la calidad de las mismas, pues es el cliente quien toma la decisión final de aprobar o no el producto desarrollado. Para su realización se diseñan numerosos escenarios o casos de prueba acotados que deberán efectuar los usuarios escogidos para esta fase, que pretenden abarcar el flujo normal de utilización de la aplicación.

8.5.2. Adaptación de UAT a este proyecto

En este proyecto, al no disponer de usuarios finales, se selecciona un grupo de 10 usuarios, de los cuales el 10 % contará con permisos de administrador, de diferentes edades y con conocimientos de informática y tecnologías de la información y las comunicaciones (TICs) muy diversos, con diferentes tipos de dieta. Esto permite simbolizar y cubrir el nicho de mercado contemplado en la Sección 1.4.

Se diseñan 13 casos de prueba, que pueden verse en detalle en la Tabla 8.2.

En el caso de los usuarios con permisos de administrador en la web se añaden algunos escenarios más, cuya descripción puedes observarse en la Tabla 8.3.

Número de escenario	Descripción
Escenario 1	Registrarse y cerrar sesión
Escenario 2	Buscar el aditivo con código 1105. En caso de poseer cuenta en Telegram se pedirá la misma tarea de consulta desde el bot de Telegram
Escenario 3	Cambiar en el perfil la información correspondiente al tipo de dieta
Escenario 4	Buscar en el foro un tema con título “prueba”
Escenario 5	Crear un nuevo tema en el foro
Escenario 6	Consultar quién ha escrito el último comentario en el tema “Prueba” y ver su perfil
Escenario 7	Reportar el comentario “Prueba reportar” del tema “Prueba”
Escenario 8	Añadir un comentario en el tema “Prueba” y después eliminarlo
Escenario 9	Añadir un comentario en el tema “Prueba” y después eliminarlo
Escenario 10	Consultar el tema que ha sido actualizado más recientemente
Escenario 11	Consultar el primer tema que se creó
Escenario 12	Consultar todos los temas relacionados con la dieta vegana
Escenario 13	Cambiar el idioma a inglés y buscar en el foro todos los temas cuyo idioma sea el inglés

Tabla 8.2: Escenarios de prueba de la fase UAT

Número de escenario	Descripción
Escenario 1	Reportar un comentario y eliminarlo desde la lista de reportes
Escenario 2	Eliminar un usuario y consultar su perfil
Escenario 3	Reportar un comentario y desestimar el reporte

Tabla 8.3: Escenario de prueba de los usuarios con permisos de administrador de la fase UAT

En cada uno de los escenarios, cada usuario deberá valorar, en una escala del 1 al 5, la dificultad de realización de la tarea solicitada, el tiempo que le lleva realizarla y podrá aportar sugerencias de mejora de cualquier tipo o reportar cualquier error detectado.

Al finalizar la ejecución de todos los casos de prueba comentados los usuarios deberán valorar, también en una escala entre 1 y 5, una serie de características generales de la aplicación, así como aportar ciertos datos que permitan obtener resultados y conclusiones para mejoras aún por realizar en la aplicación antes de dar por finalizado su desarrollo. Estas características están inspiradas en [59], donde se proporciona un ejemplo de cuestionario básico para la recopilación de datos. Se han escogido las siguientes:

- Agradable
- Entendible

- Fácil de aprender
- Útil
- Interesante
- Rápido
- Original
- Atractivo
- Cómodo
- Innovador

Se añaden, además, de cosecha propia:

- Satisfacción general
- Nivel de recomendación
- Sugerencias generales o errores detectados

Respecto del resto de características del comentado formulario proporcionado en [59] se ha decidido **eliminar** las que siguen:

- Creativo
- Valioso
- Impredecible
- Impulsor de apoyo
- Bueno
- Fácil
- Novedoso
- Seguro
- Activante
- Cubre expectativas
- Eficiente
- Claro
- Pragmático
- Simpático

El motivo de su exclusión es la cobertura que ofrecen las características escogidas o el poco valor que aportan en el caso concreto de este proyecto. Además se pretende hacer un cuestionario adecuado en extensión para que los usuarios de prueba no lo perciban como una carga excesiva sino que resulte ameno y leve.

Al inicio de la prueba se pedirá al usuario que informe de:

Rango de edad	Relación con las TICs	Tipo de dieta
0-20	4	Ninguno
41-60	3	Ninguno
21-40	4	Celíaco/a
41-60	2	Ninguno
21-40	5	Vegano/a o vegetariano/a
21-40	4	Vegano/a o vegetariano/a
21-40	5	Celíaco/a
0-20	3	Celíaco/a
41-60	5	Celíaco/a
21-40	1	Celíaco/a + intolerante a la lactosa

Tabla 8.4: Perfiles de los usuarios que ejecutan los escenarios de prueba

- Rango de edad al que pertenece.
- Tipo de dieta.
- Nivel de relación con las TICs.

Esto permitirá agrupar a los usuarios en función de su perfil, de modo que se puedan obtener conclusiones con una perspectiva más cercana a la situación individual de cada uno de ellos.

Para la obtención de los resultados de las valoraciones y la recogida de datos se crea un formulario empleando la herramienta Google Forms [61], que permite recopilar respuestas de forma rápida y sencilla haciendo uso de la nube.

La situación ideal para estas pruebas sería realizarlas en presencia del desarrollador, de modo que éste fuera capaz de observar el comportamiento del usuario ante el sistema y realizara sus propias anotaciones al respecto. Sin embargo, debido a la pandemia del coronavirus [56] que aún asola el país y las sucesivas prórrogas del estado de alarma decretado por el Gobierno (situación ya comentada en la Sección 5.1.4) que aún mantienen a la población en cuarentena, esto no será posible. Por ello, las pruebas se realizan de forma online y asíncronamente, recibiendo el desarrollador las respuestas de los usuarios ante los escenarios planteados.

8.5.3. Resultados y conclusiones

En la Tabla 8.4 puede verse una resumen de los perfiles de los usuarios tomados como modelo para ejecutar los escenarios de prueba diseñados.

Una vez recogidos los resultados de las pruebas de los usuarios, se procede a realizar un análisis de los mismos. Para ello, haciendo uso de la herramienta Microsoft Excel, se han diseñado varios gráficos a fin de describir de forma más visual las impresiones de los usuarios ante los casos de prueba propuestos.

Se ha decidido agrupar varios escenarios por cada gráfica, teniendo 2 gráficas por cada grupo: una para la dificultad y otra para el tiempo empleado. Éstas pueden consultarse en las Figuras 8.4 a 8.9.

Para los escenarios de prueba de los usuarios con permisos de administrador, dado que sólo se cuenta con las valoraciones de 1 persona (10 % del total, como se comentó en la Sección 8.5.2), los resultados se muestran en forma de tabla, en la Tabla 8.5.

La descripción de los escenarios a los que se hace referencia puede encontrarse en las Tablas 8.2 y 8.3.

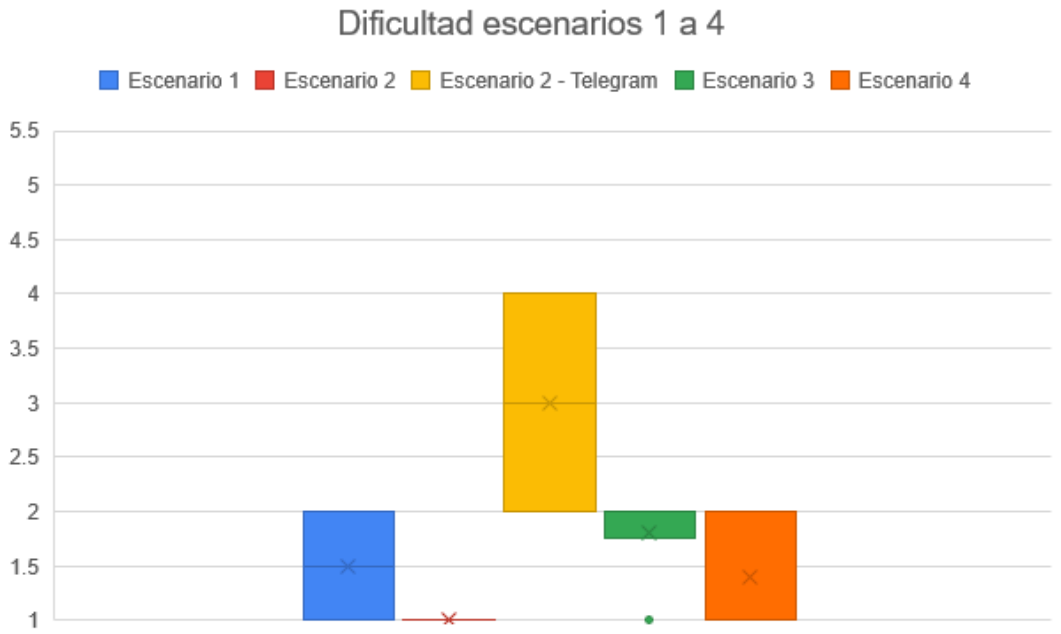

Figura 8.4: Dificultad escenarios 1 al 4

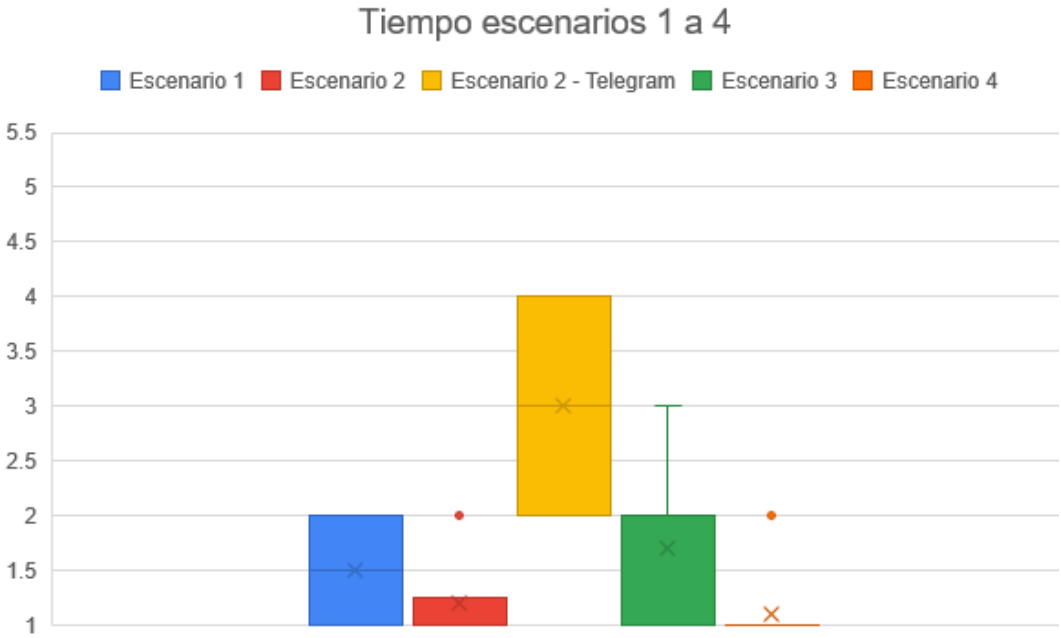

Figura 8.5: Tiempo empleado escenarios 1 al 4

Escenario	Dificultad	Tiempo empleado
Escenario 1	1	1
Escenario 2	1	1
Escenario 3	2	1

Tabla 8.5: Escenarios usuario con permisos de administrador

En todos los escenarios se puede observar que la mayor parte de las valoraciones, tanto en dificultad como en tiempo, se encuentran entre 1 y 3, lo cual puede catalogarse como un rango medio-bajo. Se ha considerado por ello que el proyecto cumple las expectativas iniciales propuestas, pues pretendía acercarse al mayor rango de usuarios

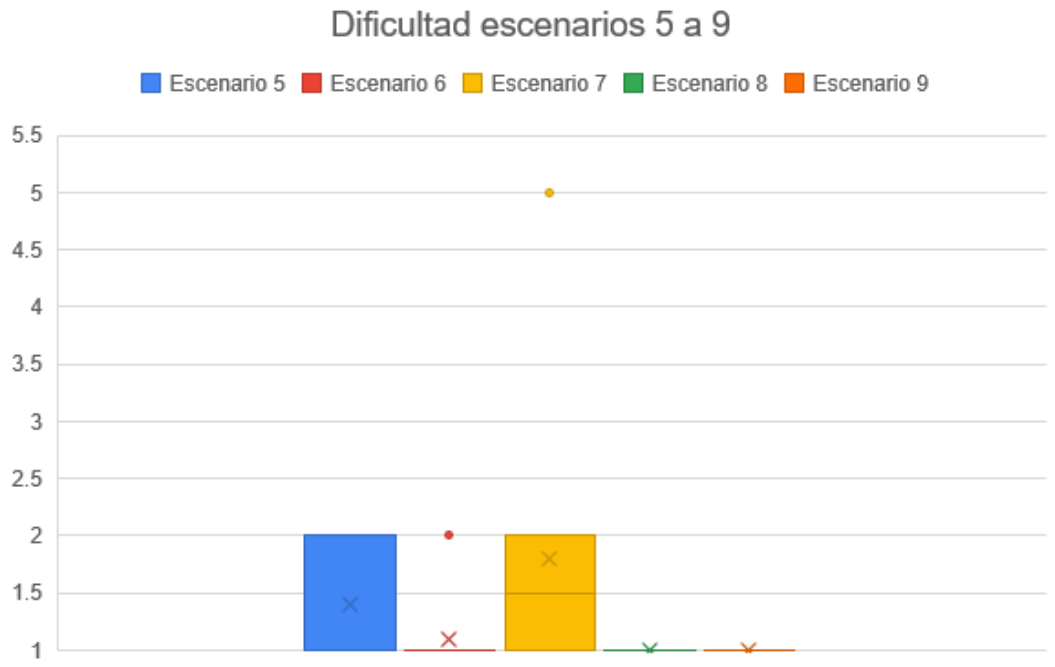

Figura 8.6: Dificultad escenarios 5 al 9

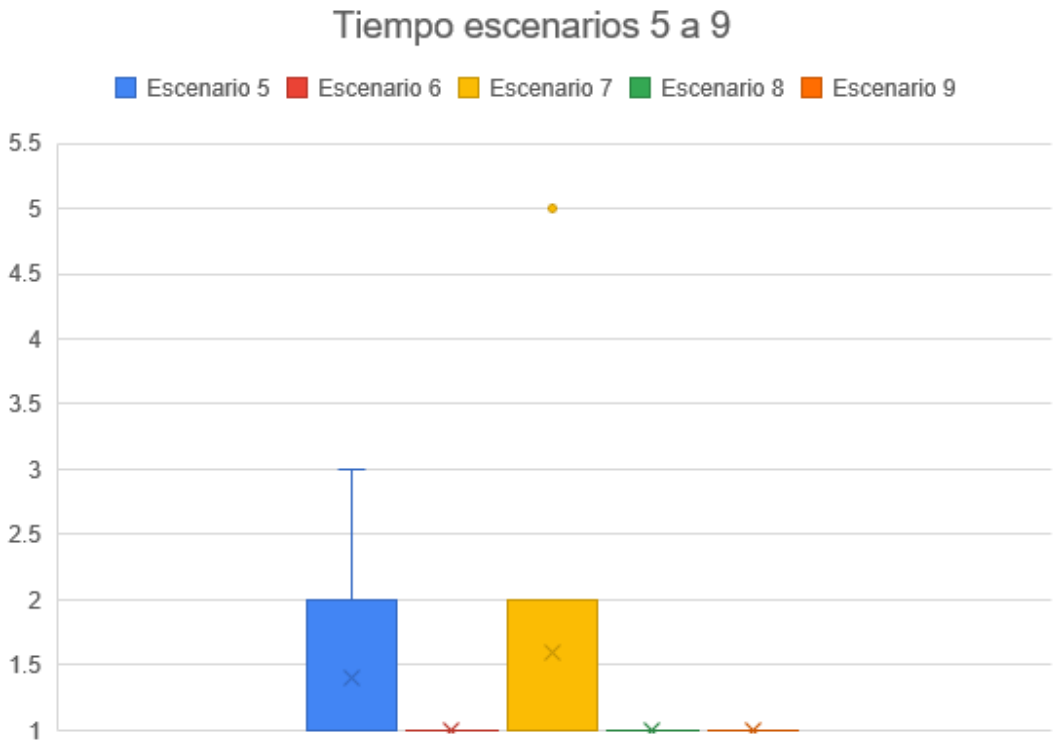

Figura 8.7: Tiempo empleado escenarios 5 al 9

posible y, teniendo en cuenta la variedad de ellos con los que se han realizado las pruebas, puede considerarse que dicho objetivo ha sido cubierto.

Merecen quizá especial atención ciertos escenarios en los que la valoración difiere de la tónica general.

En la Figura 8.4 y la Figura 8.5, los usuarios que probaron el bot de Telegram, los cuales constituyen únicamente

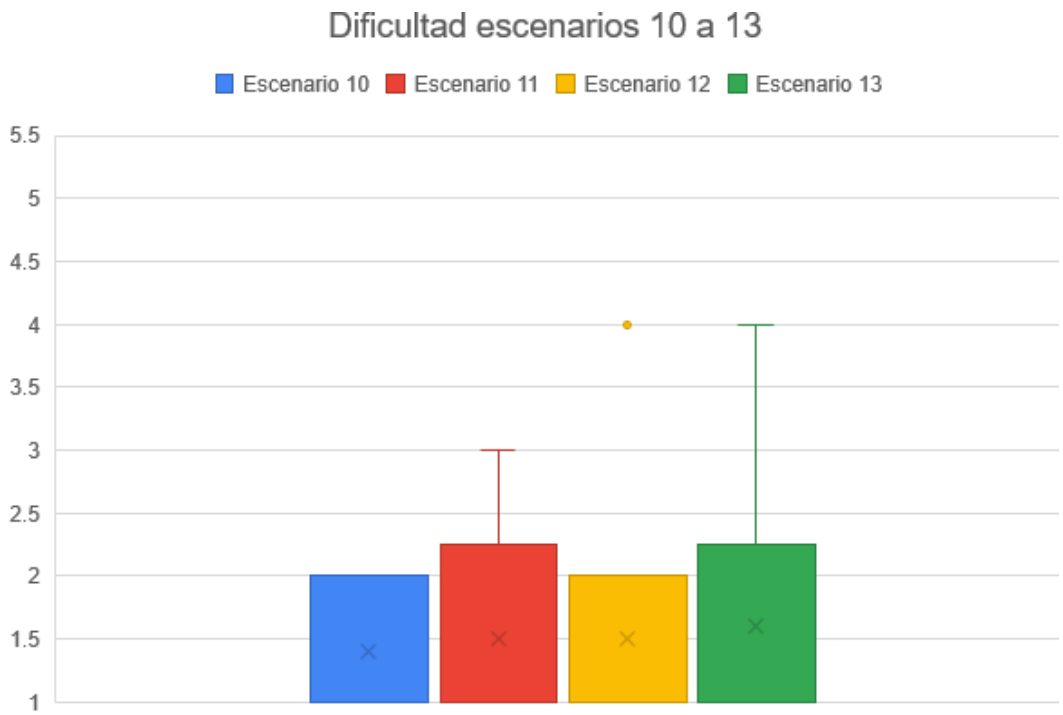

Figura 8.8: Dificultad escenarios 10 al 13

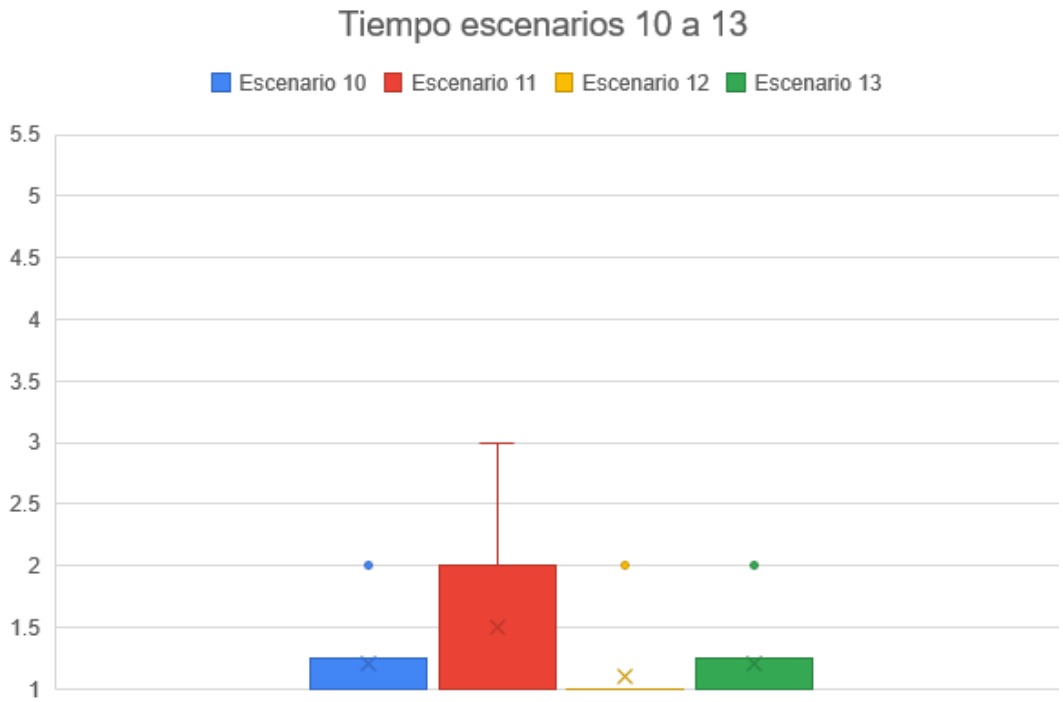

Figura 8.9: Tiempo empleado escenarios 10 al 13

el 20 % del total, difieren mucho en sus valoraciones, tomando especial relevancia los que superan el 3. Se pone especial atención en mejorar este caso, cambiando los mensajes de bienvenida del bot y pasando de una interfaz basada en texto a una basada en iconos a fin de mejorar la facilidad de uso, la eficiencia y el tiempo que el usuario toma para buscar la información de un aditivo. En el escenario 3 se puede apreciar que, especialmente la dificultad, ha resultado ser más alta de lo esperado con prácticamente todos los valores concentrados en torno al 3. Esto se debe a la forma de selección del tipo de dieta, incidencia que será tomada en cuenta para mejorar este aspecto.

Respecto a las Figuras 8.6 y 8.7 los valores se encuentran en torno a lo esperado, en un rango entre 1 y 3. Destaca el escenario 7, en el cual, tanto el tiempo como la dificultad, presentan un valor aislado en el 5. Esto se debe a que, tras consultar con el usuario que reportó esta valoración, se vio que no había sido capaz de ver el mensaje "Prueba reportar" porque Google Chrome traducía automáticamente la palabra "reportar" por "informar". Al inicio del cuestionario se había establecido en las instrucciones que, si no se podía completar la tarea solicitada, se marcara la dificultad máxima y el tiempo máximo. Fue el caso de este usuario.

En las Figuras 8.8 y 8.9 destaca por encima de los demás (pues el resto de valores se encuentran de nuevo en un rango medio-bajo) la dificultad de los escenarios 12 y 13, donde existen puntos aislados en el 4. Esto es debido a que la respuesta del último usuario se demoró en exceso y ya se habían resuelto ciertas incidencias que conllevaron cambios profundos en la base de datos que provocaron ciertos errores en la versión desplegada, cuya versión no había sido actualizada.

Los valores recogidos en la Tabla 8.5, correspondientes a las pruebas de los usuarios con permisos de administrador en la web, se encuentran en valores óptimos. Tanto la dificultad como el tiempo empleado en los 3 escenarios resulta ser mínimo.

En las Figuras 8.10 y 8.11 se encuentran reflejadas visualmente las valoraciones de los usuarios respecto a las características generales de la web solicitadas en el cuestionario.

Figura 8.10: Opinión general de la web, primera parte

Se puede observar en ambas gráficas que todos los valores oscilan entre 3 y 5, lo que constituye un rango medio-alto. Merece la pena reseñar la facilidad de aprendizaje, la rapidez, la comodidad y la utilidad, características importantes para lograr el objetivo de acercarse al mayor número de personas posible, con características de edad y tipos de dieta dispares. Sobresale la valoración de lo atractiva que resulta la web, donde se ha recogido alguna valoración por debajo de 3, por lo que se trabajará en cambiar especialmente los colores para hacerla más llamativa.

De todo ello puede concluirse que la web cumple con lo esperado, donde las tareas no resultan complicadas de cara a los usuarios y donde las funcionalidades de la misma resultan intuitivas, en cierta medida útiles para la mayor parte de los usuarios y cómodas. No se ha creado, por lo que se ha podido extraer de los datos recogidos, un producto demasiado innovador o atractivo, pero sí lo suficiente como para que la satisfacción y el nivel de recomendación se mantengan en un rango medio-alto.

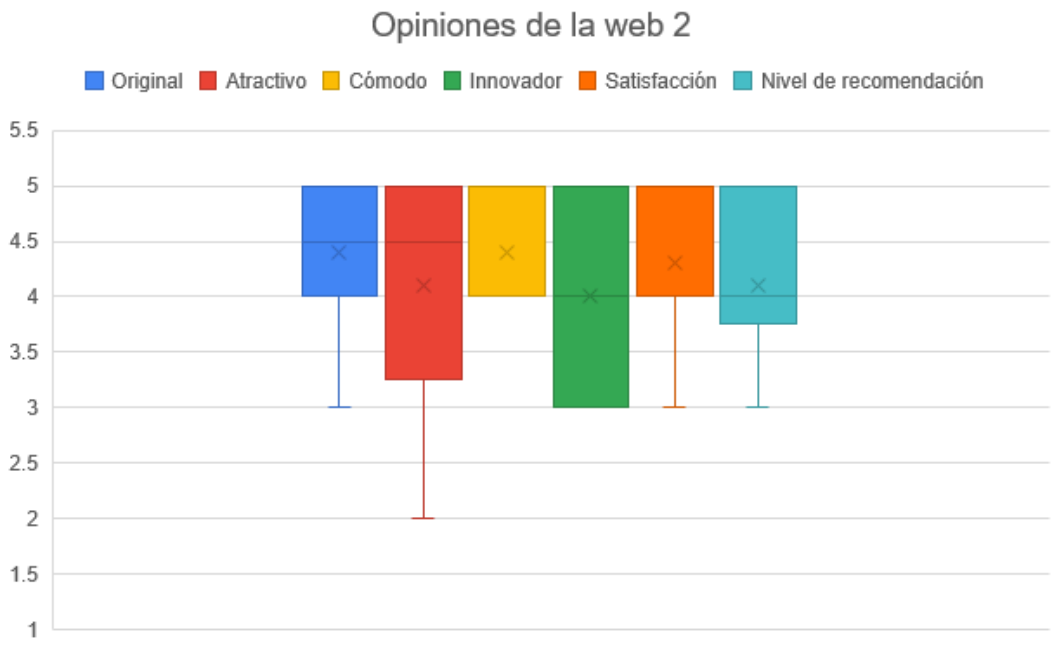

Figura 8.11: Opinión general de la web, segunda parte

Además de tener en cuenta la valoración de los usuarios también se ha prestado atención a todas las sugerencias que éstos han realizado con respecto a las tareas que debían completar. Se ha hecho una selección entre ellas, junto a la tutora, para escoger las más interesantes y que resultan de mayor relevancia de cara al funcionamiento y las características no funcionales de la aplicación. Estas tareas se encuentran desglosadas en la Tabla 8.6, donde se otorga a cada una un número de referencia para poder identificarla, una breve descripción y la prioridad (alta o baja) que se ha dado a dicha tarea. Durante el sprint 7 (Sección 5.1.8) se realiza la implementación de las mismas.

Además de estas incidencias también se decide cambiar la acción "Reportar" por "Denunciar", pues se ha visto que a muchos de los usuarios les resulta confuso dicho término.

Número	Descripción	Prioridad
1	Añadir modal de confirmación para cerrar sesión	Alta
2	Cambiar el tipo de calendario para introducir la fecha de nacimiento	Alta
3	Cambiar la forma de selección para el tipo de dieta en el registro	Alta
4	Introducción de iconos diferentes para cada tipo de dieta en la visualización de la información de un aditivo, tanto en el bot de Telegram como en la web	Baja
5	Mostrar en qué afecta el aditivo que se busca a la persona que haya iniciado sesión en la web	Alta
6	Abstraer la búsqueda de temas en el foro de la tildes	Alta
7	Añadir al filtro por tipo de dieta del foro una opción “Mi dieta” para consultar los temas relacionados con la dieta del usuario dado de alta	Alta
8	Añadir etiquetas múltiples tanto al filtrar los temas del foro como al añadir un tema	Alta
9	Añadir la hora a la fecha de última actualización de un tema	Alta
10	Añadir criterio de ordenación ascendente o descendente para los temas del foro	Alta
11	Conservar los filtros que el usuario ha seleccionado en el foro una vez que se entra en un tema	Baja

Tabla 8.6: Tareas seleccionadas entre las sugerencias e incidencias de la fase UAT

Capítulo 9

Conclusiones y líneas futuras

9.1. Conclusiones

Una vez finalizado el desarrollo del proyecto se puede decir que éste ha resultado muy satisfactorio. Se han completado los objetivos y tareas planteadas al inicio del mismo. Respecto de los objetivos de desarrollo (Sección 1.5.1):

- Se ha implementado la consulta de los aditivos alimentarios desde la web.
- Se ha desarrollado el bot de Telegram propuesto al inicio.
- Se ha creado un foro simple de forma satisfactoria que cumple sus funciones con eficiencia.

Además de todo ello se ha conseguido que la web sea responsive, es decir, tenga una buena visualización en dispositivos de menor resolución. Tan sólo se ha realizado el desarrollo para móviles por considerarlo el más extendido. Este objetivo surgió a lo largo del proyecto pero se optó por implementarlo, ya que esto supondría una mayor usabiliad para aquellos usuarios que no dispusieran de la aplicación Telegram y que usaran la aplicación en movilidad.

En relación a los objetivos personales (Sección 1.5.2):

- Conocer en profundidad el framework Vue.js: pese a que ya se contaba con una buena base del mismo, el desarrollo del proyecto ha permitido profundizar en ciertos conocimientos del mismo, especialmente en su funcionamiento interno por toda la información que se ha debido recabar y plasmar en la presente memoria
- Mejorar mi capacidad y destreza en el desarrollo web *Front-end*: el planteamiento de la arquitectura del proyecto y la creación de todos los componentes de la aplicación ha permitido conocer esta faceta en profundidad, así como mejorar la eficiencia en los desarrollos.
- Acercar la herramienta que se pretende desarrollar a persona cercanas a mi entorno: se ha creado una aplicación rápida, simple y eficiente, lo que aumenta las posibilidades de que personas cercanas a mi entorno la utilicen de cara al futuro.
- Entender y aprender la metodología y los principios de Scrum: el desarrollo del proyecto en su totalidad aplicando este marco de trabajo ha contribuido sobremanera a entender en profundidad cómo trabajar de este modo.

9.2. Mejoras futuras

Durante el desarrollo del proyecto se han ido observando determinadas mejoras que podrían implementarse en un futuro. Entre ellas destacan:

- Posibilidad de subir foto de perfil para un usuario.
- Mejora de visualización para tablets.
- Posibilidad de mensajes privados entre usuarios.
- Inclusión de recetas elaboradas por los usuarios.
- Creación de una lógica *Back-end* consistente en servidores propios.
- Despliegue en dominio propio para customizar la URL de acceso a la web.

Apéndice A

Manual de usuario

Este manual se divide en dos secciones, la primera dedicada al uso de la aplicación web y la segunda, al uso del bot de Telegram.

A.1. Uso de la aplicación web

La página principal de la web, cuya apariencia puede verse en la Figura A.1, da la bienvenida a la misma. A ella se puede acceder a través del botón "Inicio" de la barra superior o haciendo click en la pequeña imagen del logo de la parte superior izquierda. En ella se puede introducir el código del aditivo alimentario que se pretende consultar.

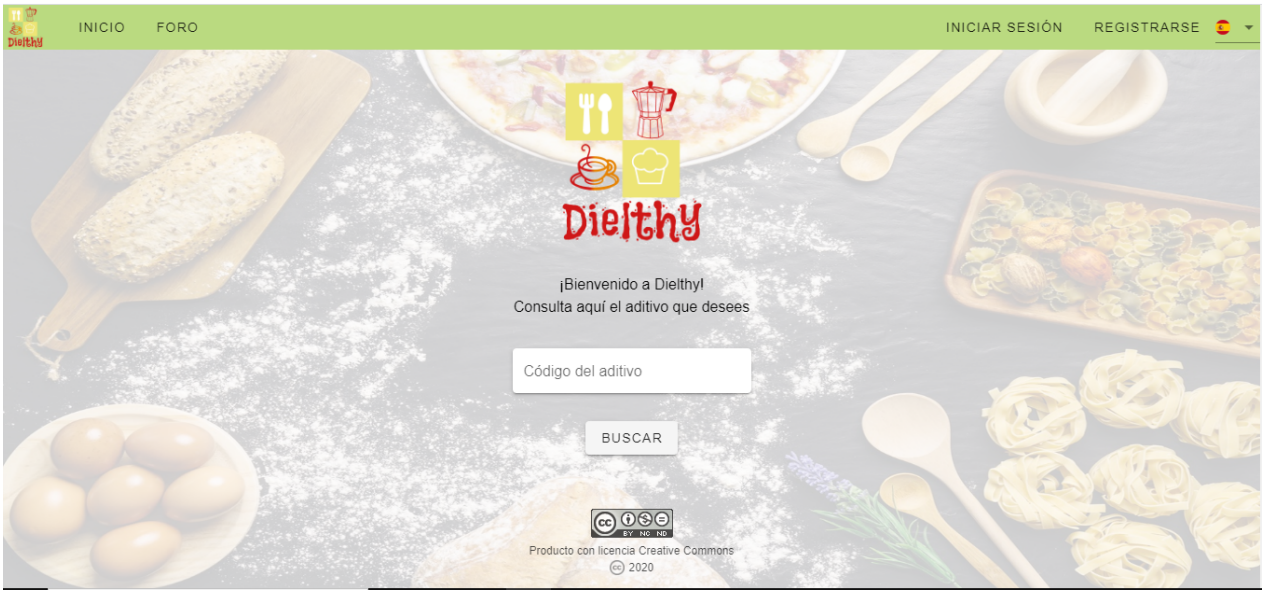

Figura A.1: Apariencia final de la página principal

La consulta de un aditivo conduce a una página en la que se presenta la siguiente información del mismo (ver Figura A.2):

- Código del aditivo.

- Descripción.
- Clasificación según su tipo.
- Origen.
- Apto o no para veganos/vegetarianos.
- Apto o no para celíacos.
- Apto o no para intolerantes a la lactosa.

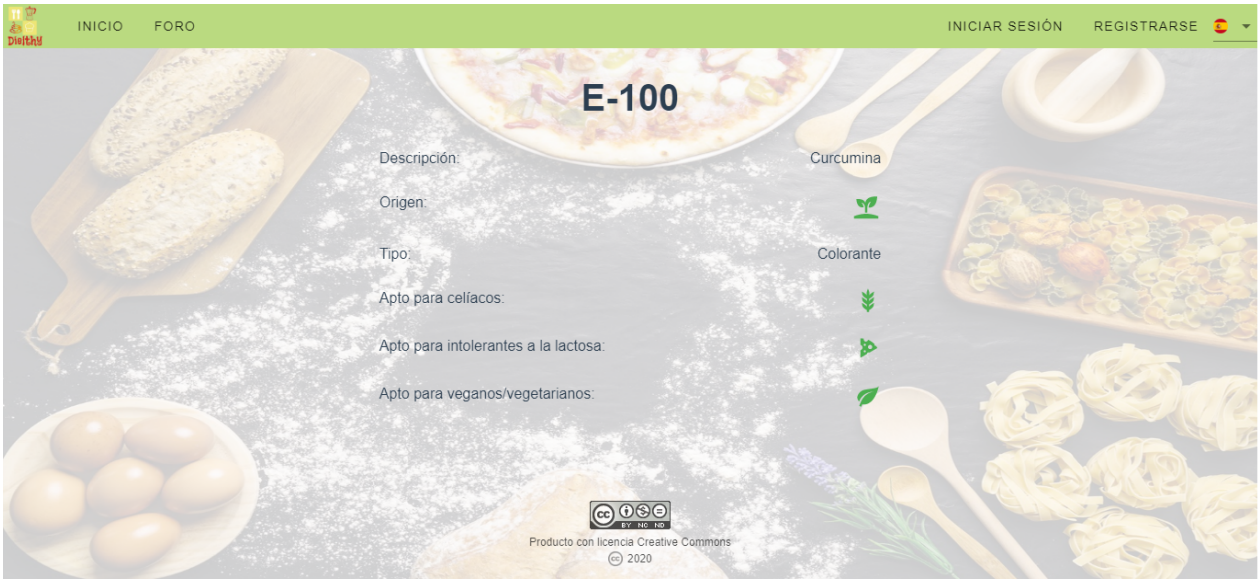

Figura A.2: Apariencia final de la página de información de un aditivo

En la parte superior derecha se encuentran los botones “Iniciar sesión” y “Registrarse”. El primero de ellos lleva a una página (Figura A.3) con un sencillo formulario donde se debe introducir el email y la contraseña, lo que permitirá acceder a la cuenta personal, siempre y cuando el usuario se encuentre registrado previamente. En caso contrario se mostrará un error.

En esa misma página, haciendo click en el link de la parte inferior del formulario es posible recuperar la contraseña. Esto abre un cuadro de diálogo, que se muestra en la Figura A.4, donde se solicita el email para reenviar un correo de recuperación de la misma. En caso de que el email no se corresponda con ningún usuario o que no sea posible enviar dicho correo se muestra un error.

El botón “Registrarse” permite acceder a un formulario para darse de alta en la web, el cual puede consultarse en la Figura A.5. En la parte inferior de cada uno de los campos se proporciona información acerca de la obligatoriedad de los mismos.

Haciendo click en el link de “política de privacidad” es posible visualizar dichas políticas (ver Figura A.6).

Una vez que el usuario se ha registrado o ha iniciado sesión, los botones de la parte superior derecha se sustituyen por “Ver perfil” y “Cerrar sesión”, además de añadirse el nombre de usuario. Haciendo click en este último se solicita confirmación y se puede cerrar sesión (ver Figura A.7).

El botón “Ver perfil” lleva a una página en la que poder consultar la información personal propia. Ésta puede verse en la Figura A.8. En la parte superior aparece un cuadro en el que, en caso de que el email no haya sido verificado haciendo click en el enlace enviado por email a la cuenta del usuario, se indica esta circunstancia.

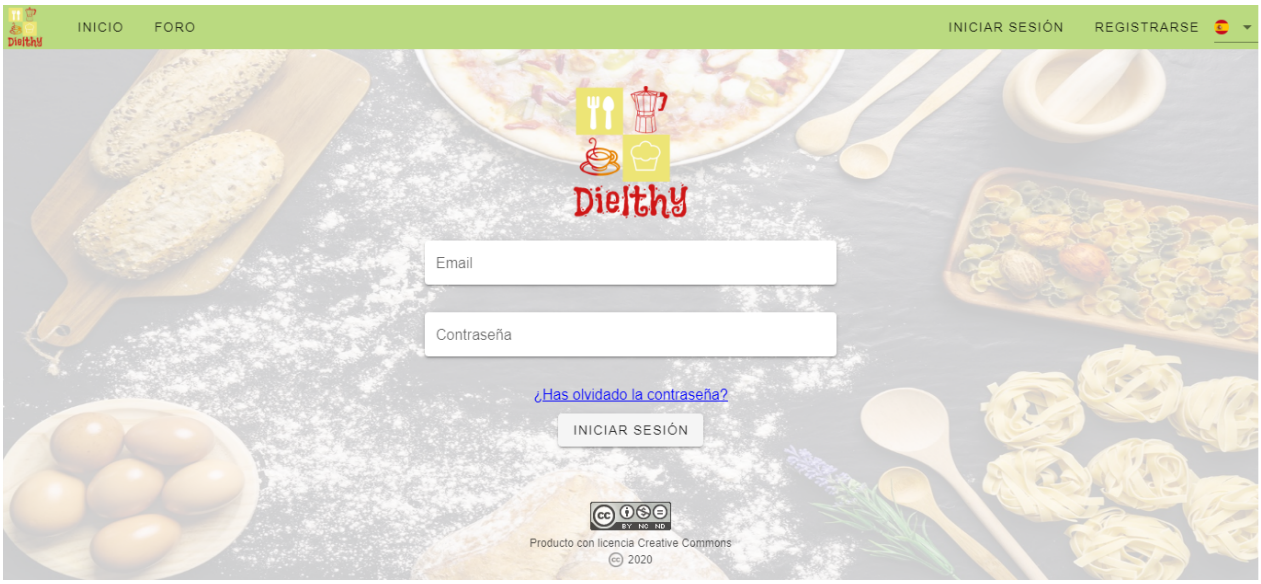

Figura A.3: Apariencia final de la página de login

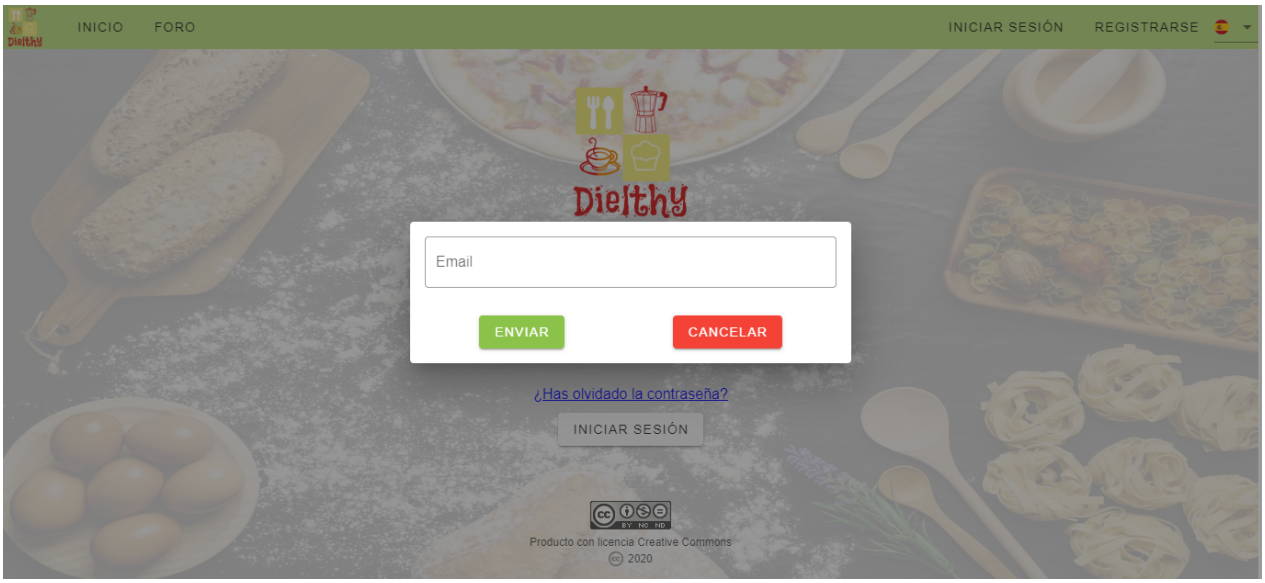

Figura A.4: Apariencia final del diálogo de recuperación de contraseña

Haciendo click en el pequeño lápiz situado a la derecha del nombre de usuario es posible ver un formulario similar al del registro donde cambiar la información de la cuenta, así como la contraseña. Éste puede verse en la Figura A.9.

El botón "Foro" de la barra superior permitirá acceder al foro (ver Figura A.10) siempre y cuando el usuario haya iniciado sesión en la web. Esta página consiste en una serie de filtros y un botón de "Nuevo tema" en la parte superior y una lista de temas. Éstos se muestran ordenados por defecto según la fecha de última actualización de cada uno, que puede verse en la parte derecha. Todos muestran, a mayores, su idioma, sus etiquetas de tipo de dieta, su fecha y usuario de creación y el número de comentarios, contando también con las respuestas a cada uno.

El botón "Nuevo tema" da acceso a un cuadro de diálogo (Figura A.11) donde se solicita un nombre para el tema a crear, la etiqueta de tipo de dieta (que puede ser múltiple) con la que se corresponderá el tema y el idioma del mismo.

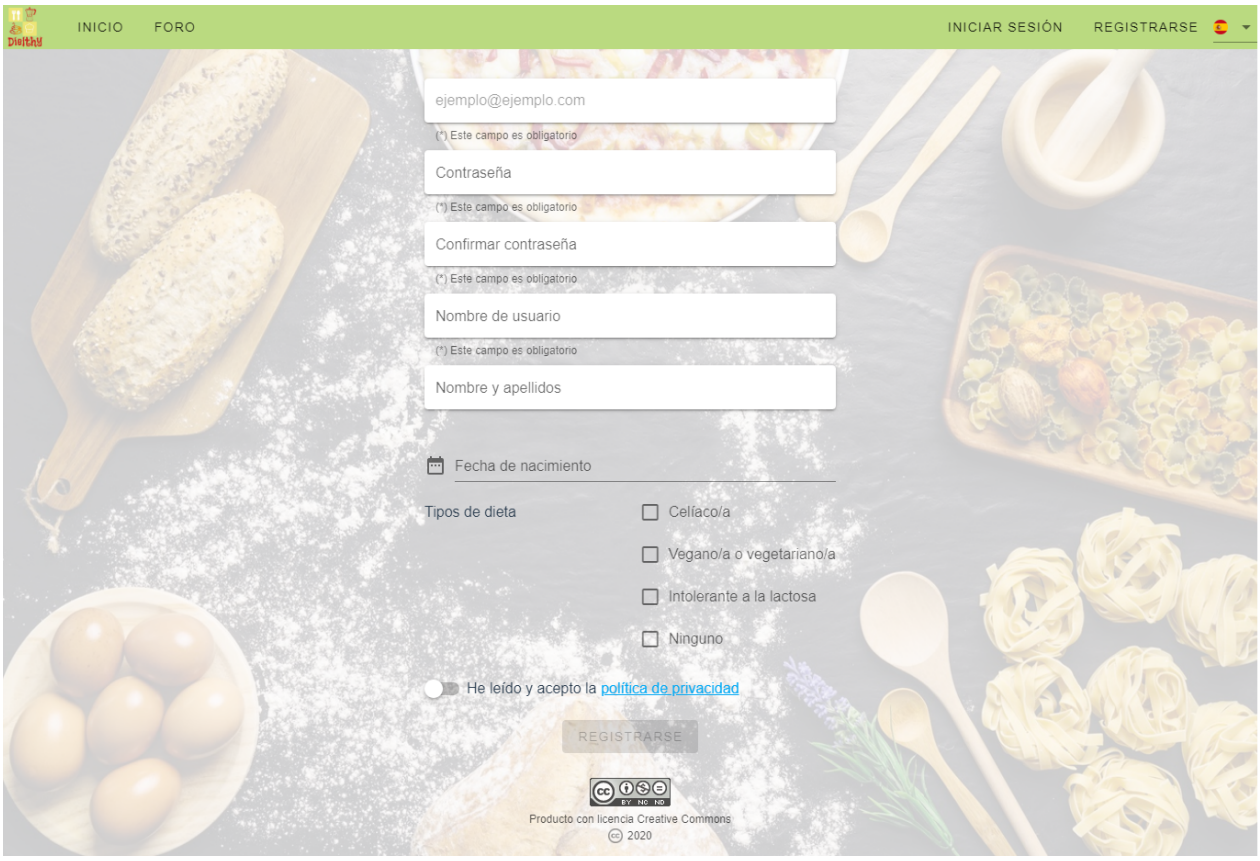

Figura A.5: Apariencia final del formulario de registro

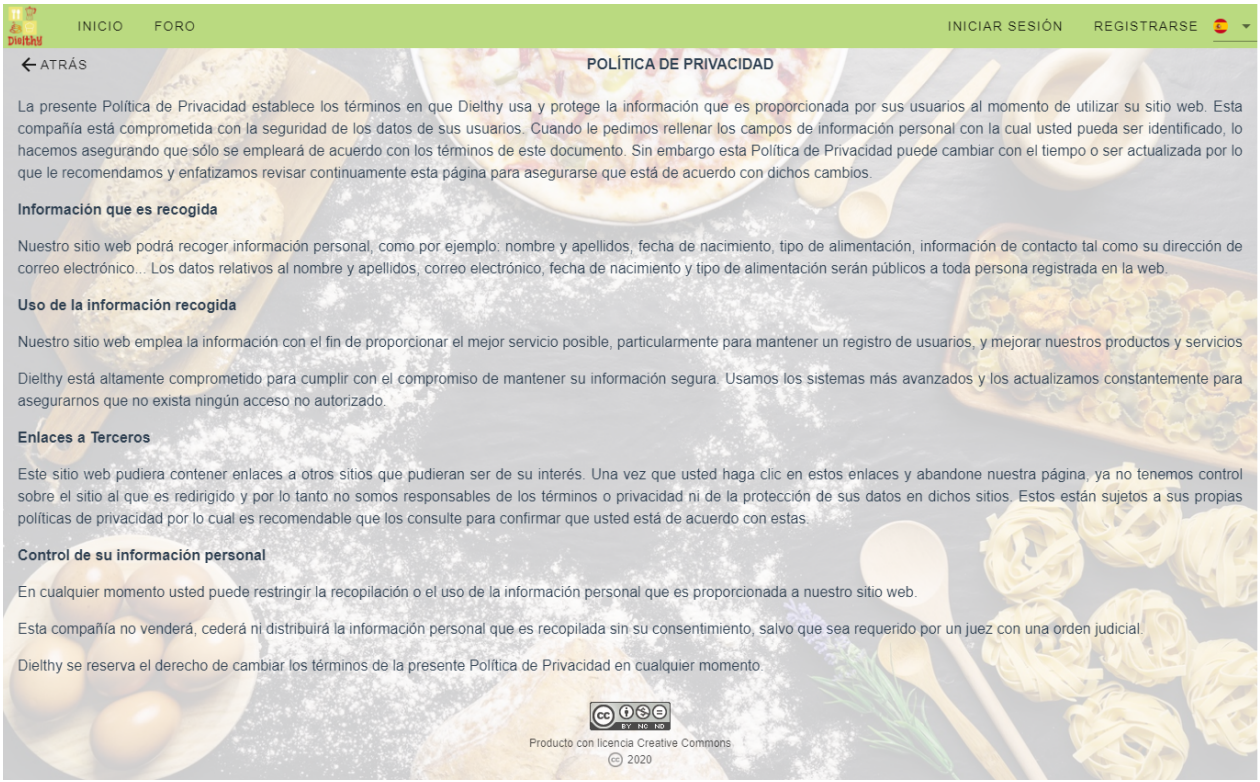

Figura A.6: Apariencia final de las políticas de privacidad

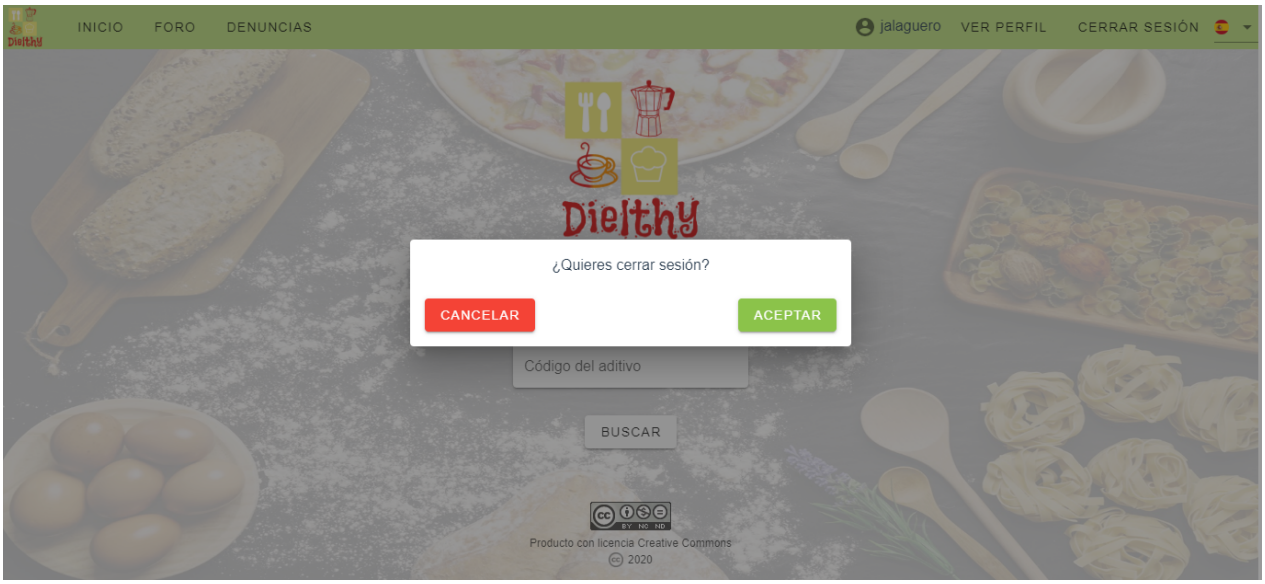

Figura A.7: Apariencia final del diálogo para cerrar sesión

Figura A.8: Apariencia final de la visualización del perfil del usuario

Haciendo click en alguno de los nombres de usuario que aparecen en la página es posible acceder al perfil de ese usuario, pudiendo consultar su información personal, tal como se muestra en la Figura A.8, con la excepción de que el icono del lápiz que aparece junto al nombre de usuario no será visible en caso de ser un perfil diferente al del usuario que ha iniciado sesión. Además, el aviso de perfil no verificado de la parte superior de la pantalla tampoco será visible, pues sólo lo será si el perfil que se visita es el del mismo usuario que ha iniciado sesión.

Si se hace click en uno de los nombres de los temas que aparecen en la página principal del foro pueden verse los comentarios de dicho tema, con la fecha de cada uno de ellos y el usuario al que corresponde (Figura A.12). En cada comentario principal se añaden 3 posibles botones para responder, denunciar y eliminar el comentario. Estos dos últimos nunca aparecen a la vez, puesto que no se permite que un usuario denuncie un comentario propio y tampoco que un usuario elimine un comentario de otro. En un nivel inferior, dentro de cada comentario principal, se encuentran las respuestas a éste. No es posible responder a las mismas, tan solo aparece disponible esta opción en el comentario principal. Si se hace click sobre un nombre de usuario se puede ver la información asociada a su

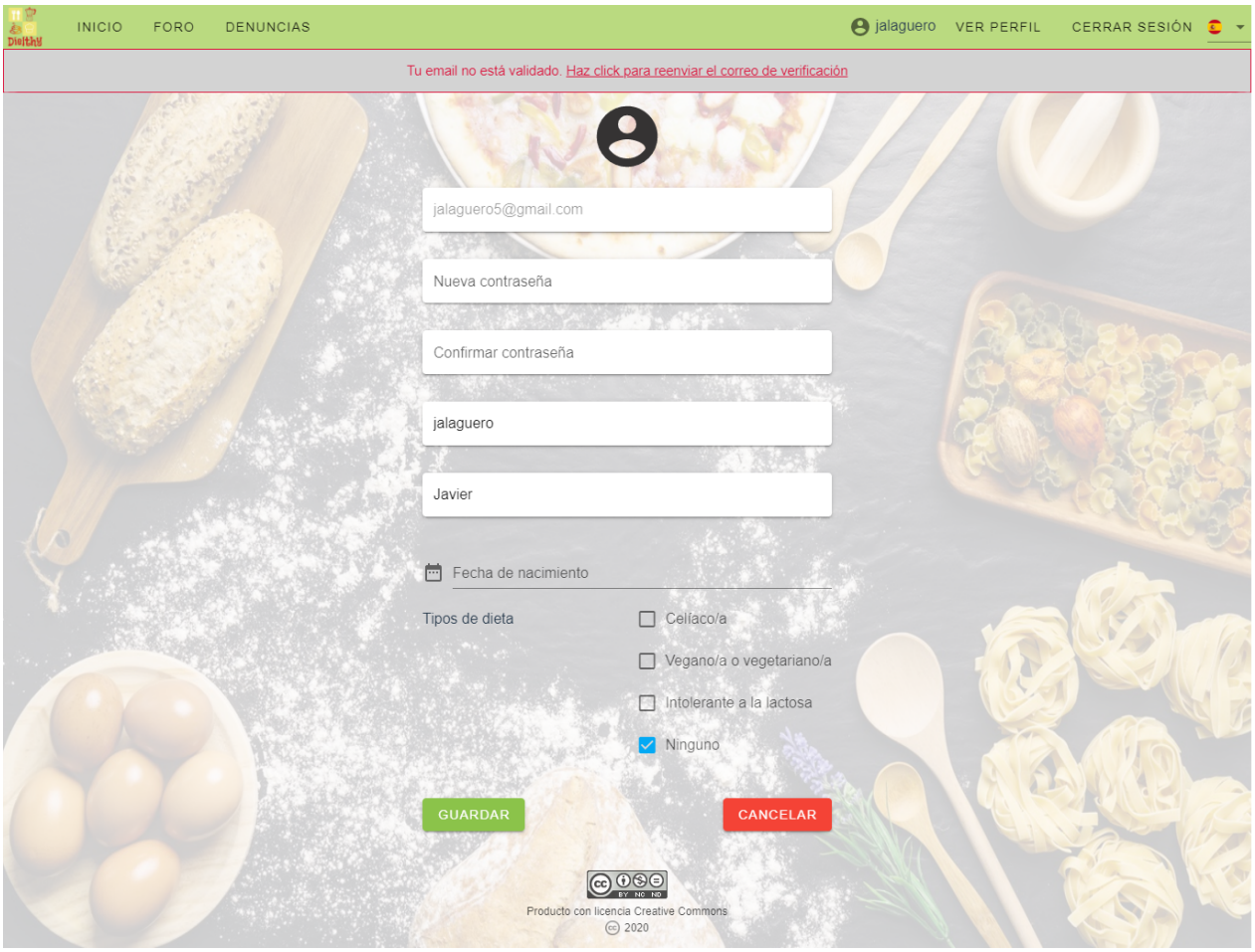

Figura A.9: Apariencia final de la edición del perfil del usuario

perfil como se muestra en la Figura A.8, donde el icono del lápiz no será visible en caso de tratarse de un perfil diferente al del usuario que ha iniciado sesión.

El botón para responder se simboliza con una flecha que apunta hacia la izquierda. Conduce a un cuadro diálogo, que puede verse en la Figura A.13 donde se solicita la respuesta que se pretende dar.

El botón para eliminar comentario o respuesta se simboliza mediante una papelera. Lleva a un diálogo de confirmación (ver Figura A.14).

En el caso del botón para denunciar un comentario o respuesta se simboliza con una señal octogonal de atención. Un click conduce a un nuevo diálogo de confirmación, el cual puede verse en la Figura A.15.

En la parte superior de la página de los comentarios de un tema (Figura A.12) se puede observar un botón "Comentar" que lleva a un diálogo como el de la Figura A.13. En este caso, posibilita añadir un comentario nuevo al tema en cuestión.

Las funciones de administrador de la web están restringidas, de modo que un usuario convencional que se registra en la aplicación no será dado de alta como administrador a menos que se le dote de dicha característica directamente en base de datos. Al iniciar sesión como administrador se puede observar que en la parte superior se añade un botón "Denuncias". Éste da acceso a todos los comentarios que han sido denunciados por otros usuarios en la web, ordenados descendentemente por número de denuncias (ver Figura A.16). Para cada uno se muestra el contenido del comentario, ya sea una respuesta a otro o no, el número de denuncias totales, un desplegable con los usuarios que lo denunciaron y cuántas veces, de modo que puedan evitarse cuentas *troll* que denuncien un

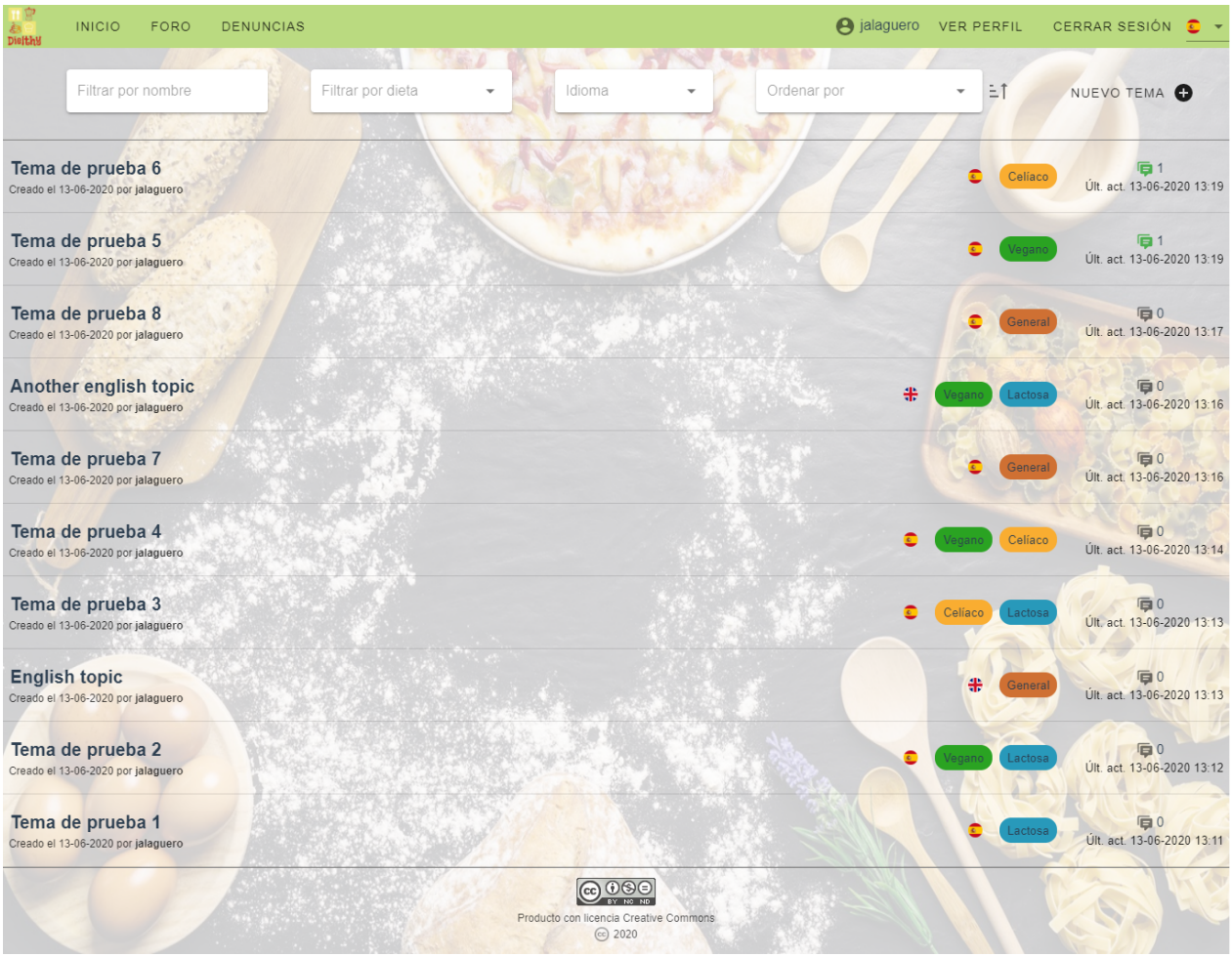

Figura A.10: Apariencia final de la página del foro

comentario reiteradamente sin motivo aparente, y la fecha del comentario. Además aparecen dos botones: uno para desestimar las denuncias de ese comentario y otro para eliminarlo en caso de que el administrador lo considere necesario. En ambos casos se muestran diálogos de confirmación. Además es posible filtrar las denuncias según el nombre de usuario de alguno de los denunciantes.

Además de esta nueva página para el administrador también se le provee de los permisos necesarios para eliminar completamente un usuario, de modo que no pueda, incluso, volver a registrarse en la web de nuevo con el mismo email. Esto es posible gracias a un botón con un icono de una papelera a la derecha del nombre de usuario en la página de visualización del perfil del mismo (ver Figura A.17). Al hacer click en dicho botón se abrirá un cuadro de diálogo en el que se solicitará el motivo de la eliminación, el cual se mostrará en la parte superior de la pantalla una vez se confirme la acción (ver Figuras A.18 y A.19). Para los usuarios sin permisos de administrador no se mostrará la información de un usuario borrado (Figura A.20)

A.2. Uso del bot de Telegram

El bot de Telegram es capaz de responder a cualquier mensaje, solicitando la introducción correcta de los comandos ante los que puede reaccionar. La lista de los comandos aceptados y su función puede hallarse en la Sección 6.8.

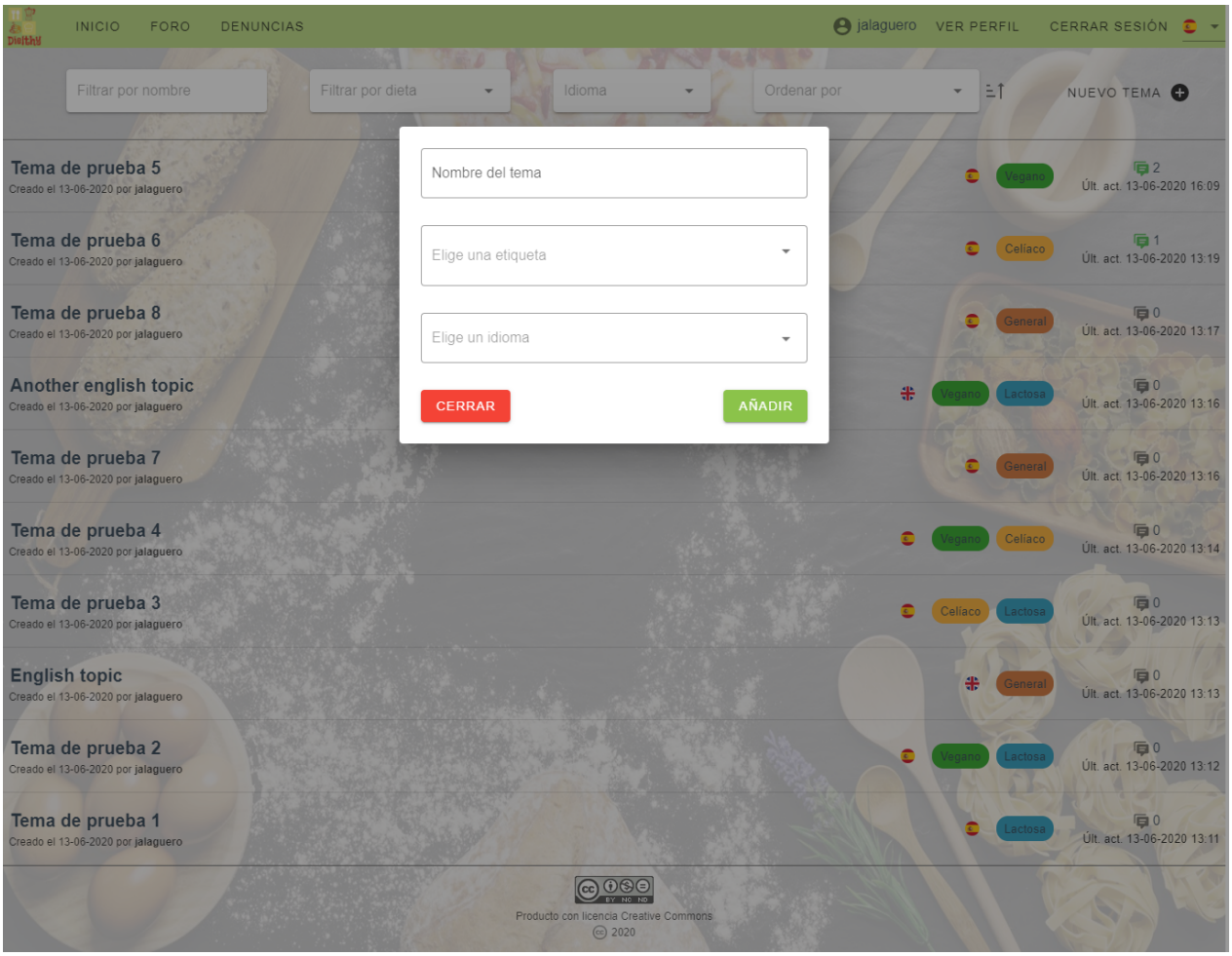

Figura A.11: Apariencia final del diálogo para crear nuevo tema

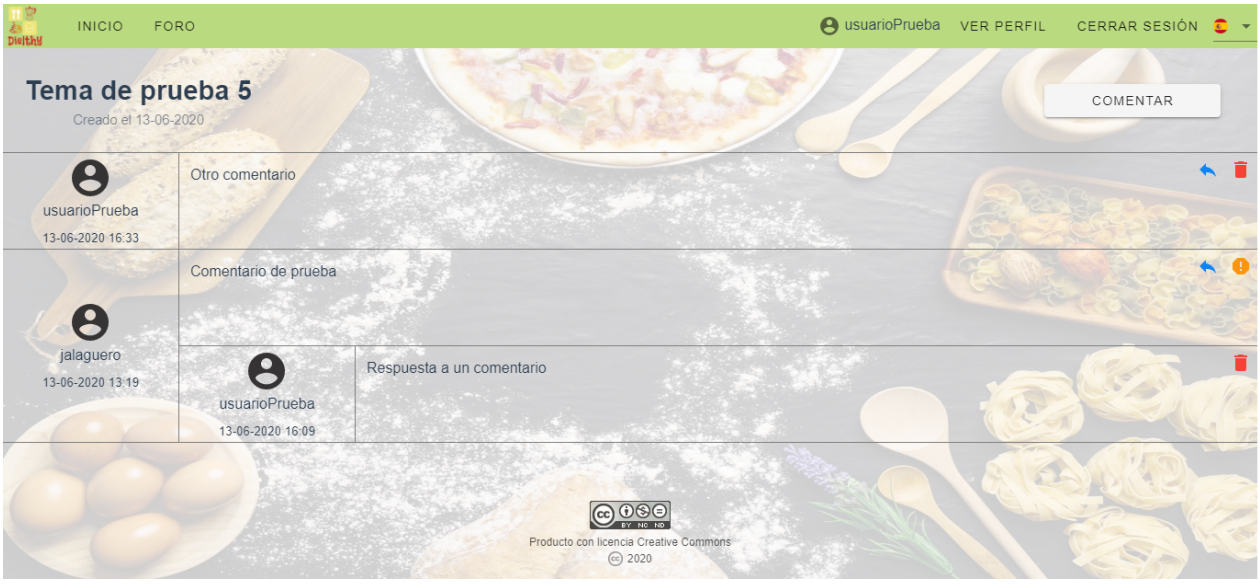

Figura A.12: Apariencia final de la página de comentarios de un tema

El comando principal es `/e` seguido del código de un aditivo. Muestra la información de dicho aditivo o, por el contrario, un mensaje de error en caso de que éste no se encuentre.

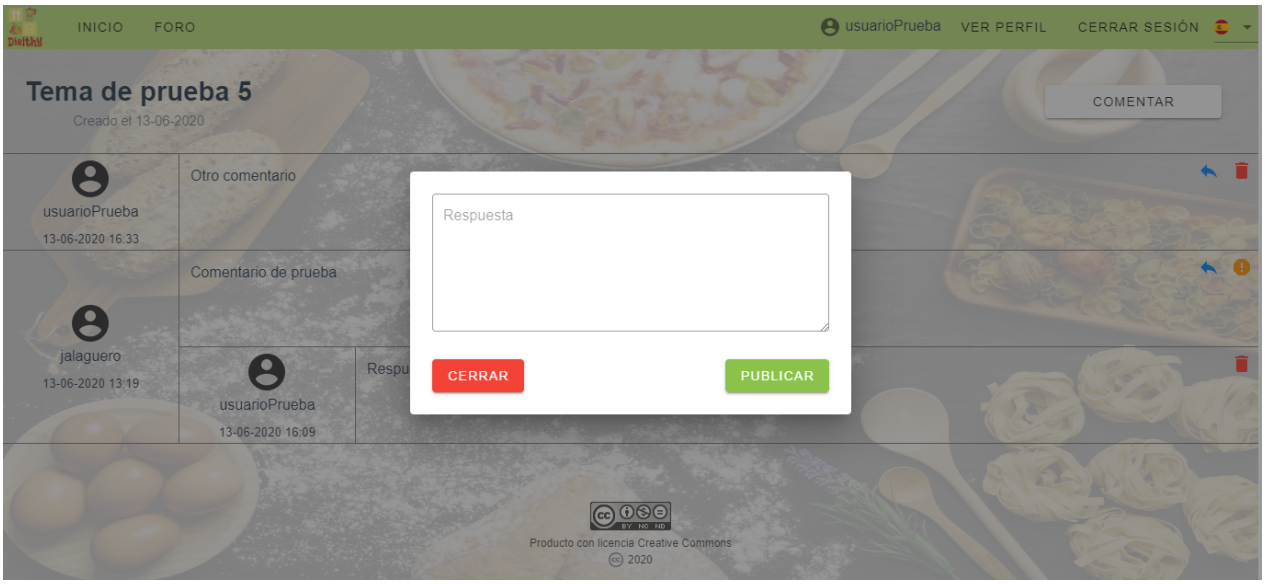

Figura A.13: Apariencia final del diálogo para responder a un comentario

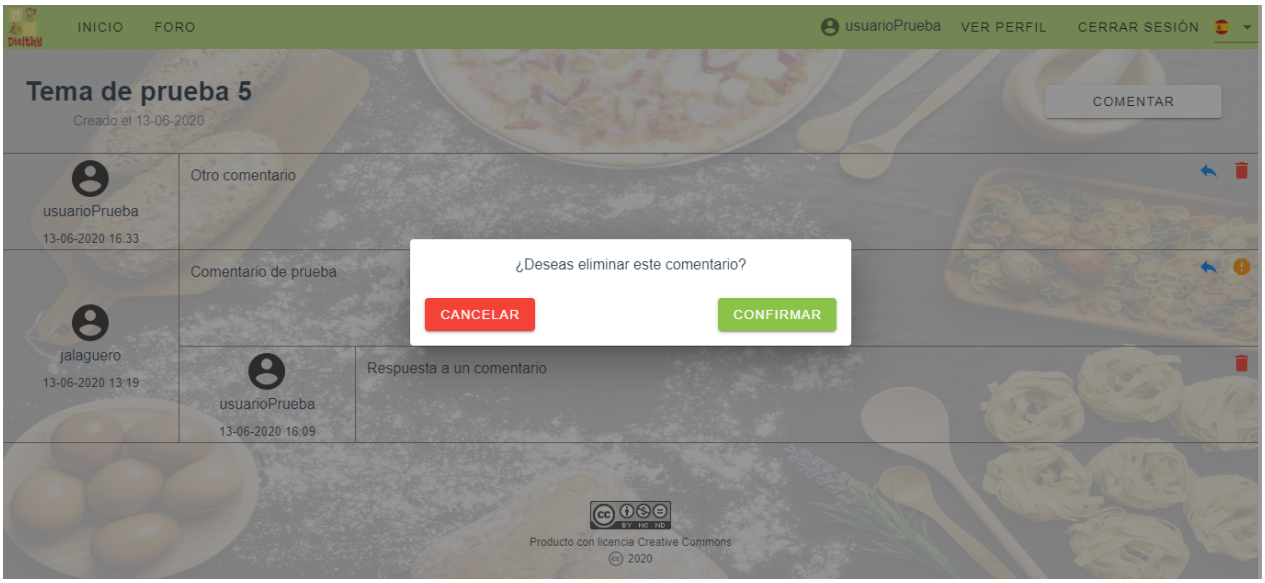

Figura A.14: Apariencia final del diálogo para eliminar un comentario o respuesta

Todas las respuestas que proporciona el bot a cada tipo de mensaje explicado pueden verse en la Figura A.21.

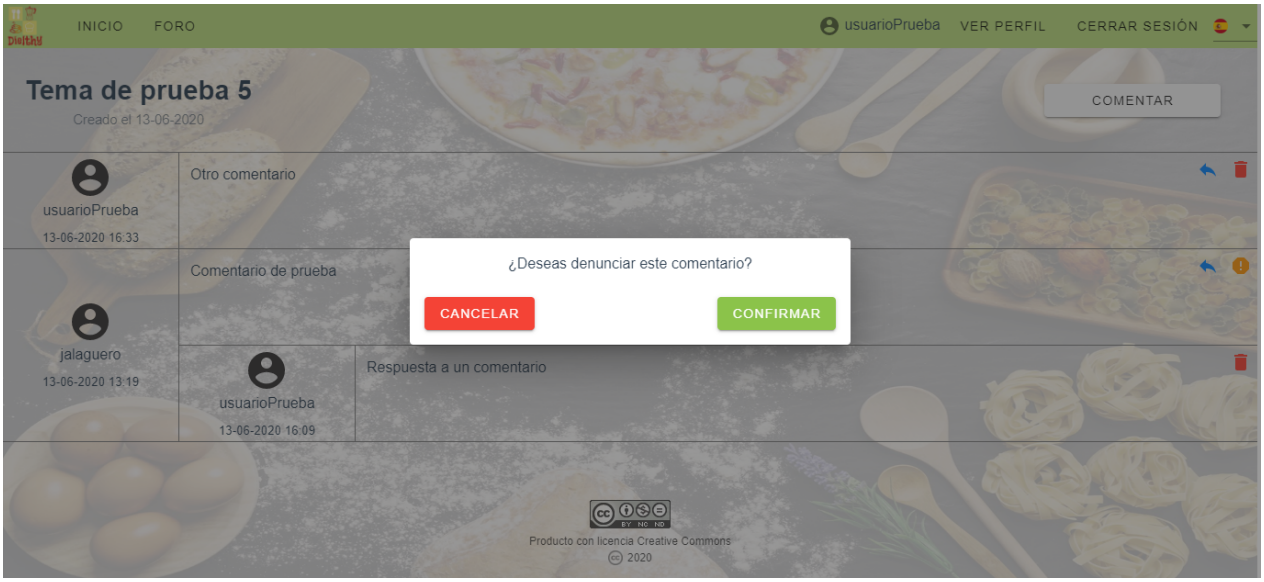

Figura A.15: Apariencia final del diálogo para denunciar un comentario o respuesta

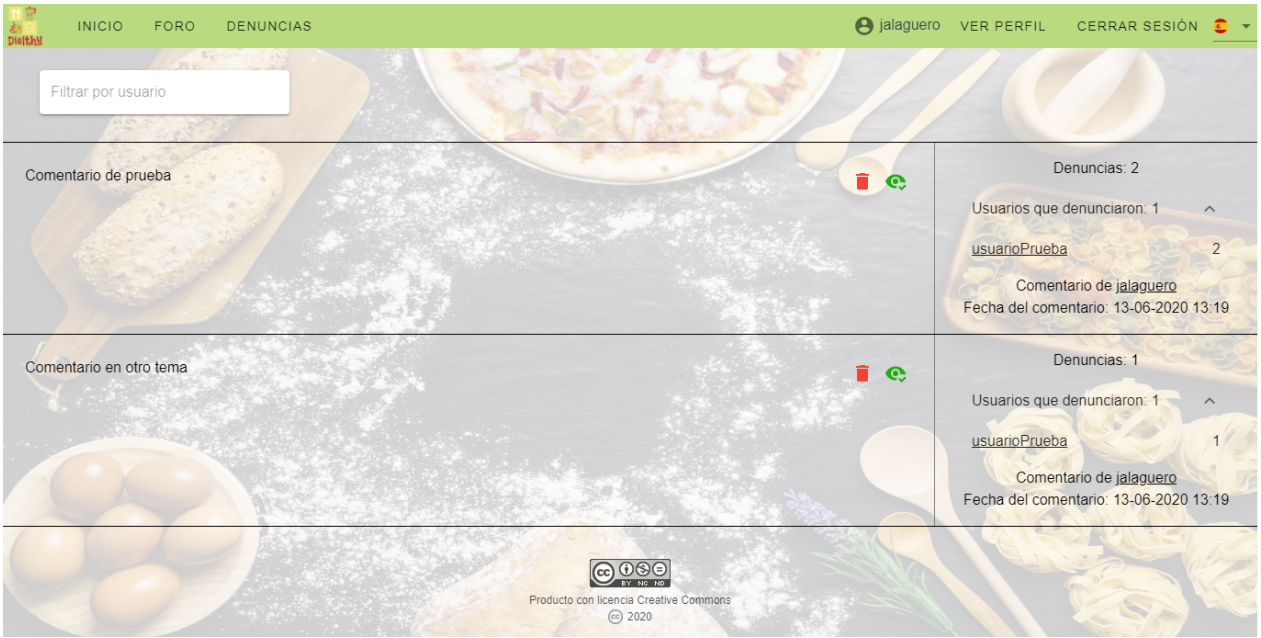

Figura A.16: Apariencia final de la página de denuncias de los comentarios

Figura A.17: Apariencia final de la visualización del perfil por parte de un administrador

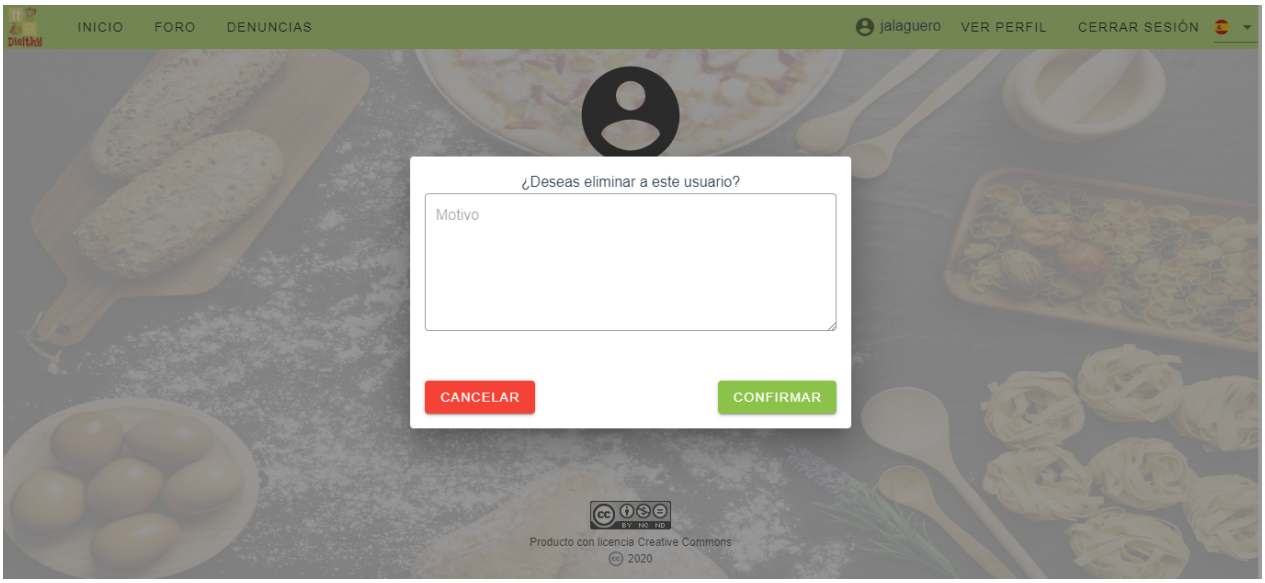

Figura A.18: Apariencia final del diálogo para borrar usuario

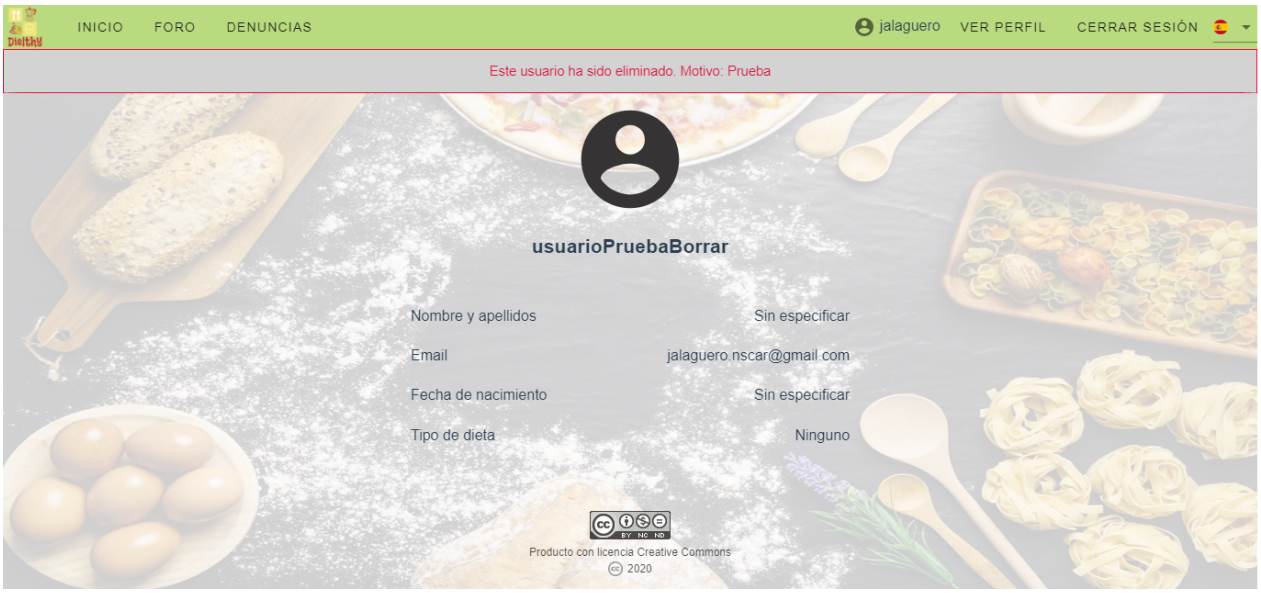

Figura A.19: Apariencia final de la visualización del perfil de un usuario borrado (administrador)

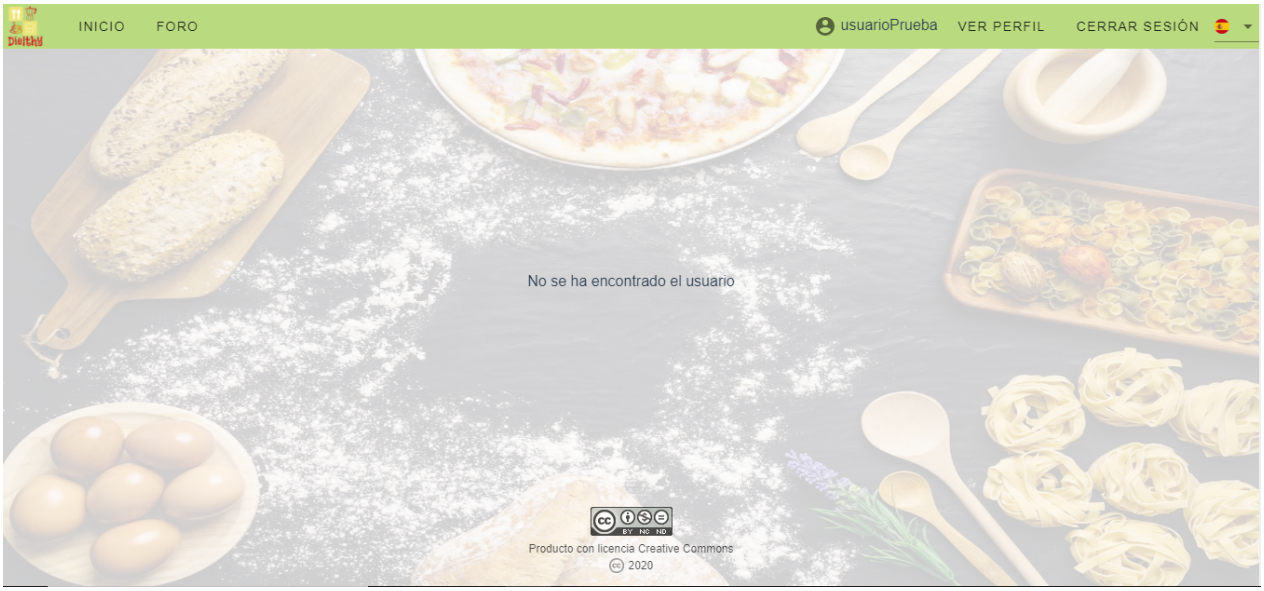

Figura A.20: Apariencia final de la visualización del perfil de un usuario borrado (no administrador)

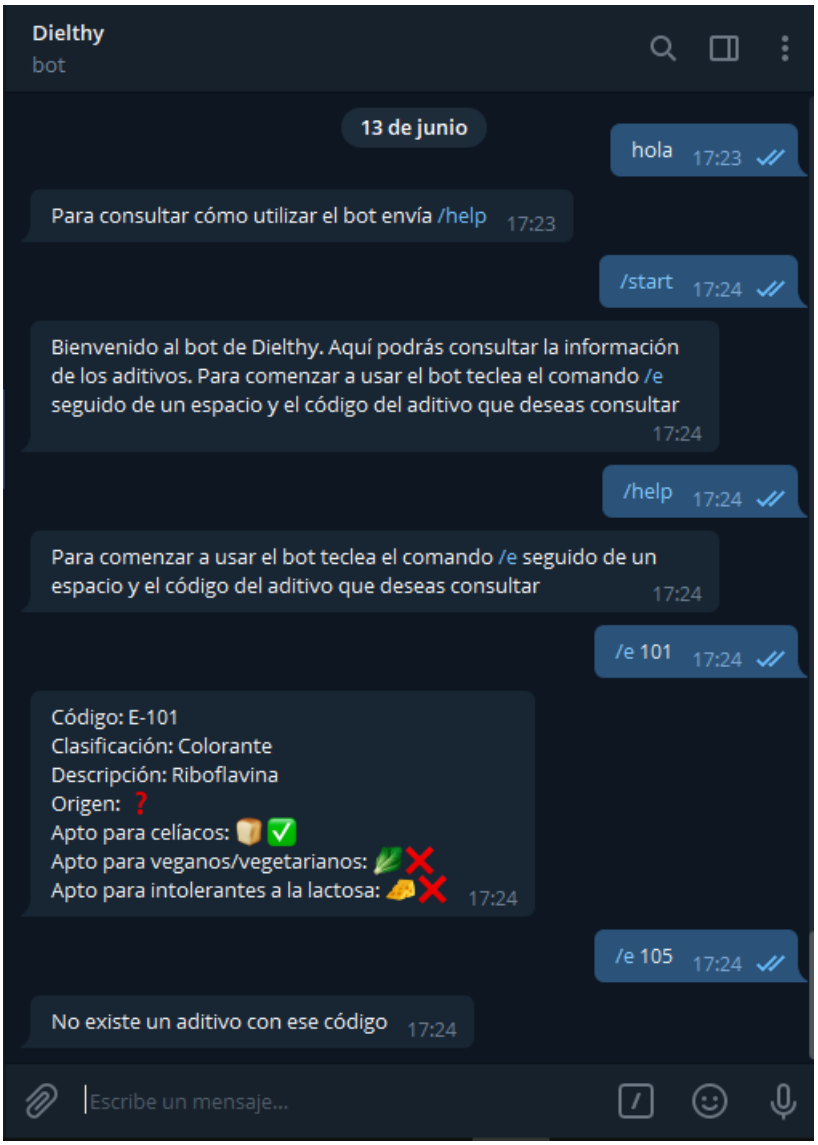

Figura A.21: Apariencia final del bot de Telegram

Apéndice B

Manual de despliegue

B.1. Herramientas requeridas

Para preparar el entorno para el despliegue de la web se necesitan las siguientes herramientas:

- Node JS versión 8+.
- npm versión 6+.
- Firebase CLI versión 8.1+.

B.2. Proceso de preparación del entorno y despliegue de la web

Para poder realizar el despliegue de la web se necesita generar un artefacto con todos los ficheros estáticos necesarios para el funcionamiento de la aplicación. Para ello se cuenta con varios scripts que facilitan la tarea.

Con el código en mano (el cual se localiza en `https://gitlab.com/JAlaguero_5/dielthyweb`), en la carpeta raíz del proyecto, ejecutando `npm install` se instalan todas las dependencias y paquetes del proyecto.

Mediante el comando `npm run serve` (alias para `vue-cli-service serve`) se crea un servidor web local en el puerto 8080 desde donde es posible consultar la aplicación de forma local. Todos los cambios realizados en los datos que contiene la web afectarán a los datos que se muestran en la versión desplegada.

El artefacto que permite realizar el despliegue como tal, se genera gracias a `npm run build`. Esto es realmente un alias para `vue-cli-service build`. Dicho comando crea una carpeta `dist` en la raíz del proyecto que estará listo para ser desplegado en un servidor web real.

Dado que la web se encuentra desplegada en la herramienta hosting de Firebase se debe realizar dicho despliegue a través de la misma. Para ello, se debe iniciar sesión en Firebase CLI ejecutando `firebase login` en una consola de comandos habitual. Tras ello se redirigirá a la web de Firebase donde deberá iniciarse sesión con una de las cuentas con permisos en el proyecto. Posteriormente, en la raíz del proyecto, desde la consola de comandos se debe ejecutar `firebase deploy`. Este comando despliega el contenido de la carpeta `dist` en el servicio de hosting de Firebase, en uno de sus servidores, de modo que, pocos minutos después, es posible consultar esta versión desplegada en `https://tfg-dielthy.web.app`. Aún a día de hoy puede encontrarse en dicha URL.

Cualquier despliegue realizado, ya sea en un servidor local o en remoto, debe contar con un archivo javascript de configuración (en nuestro caso `firebaseConfig.js`). Su ubicación puede consultarse en la Sección 7.1. Su contenido es restringido, ejerciendo el derecho que el propietario posee sobre su obra desde el mismo momento de su creación [68]. Tendrá él y sólo él, por tanto, la potestad de ceder el derecho a la visualización de dicho archivo que permitirá hacer funcionar la web, tanto en local como en cualquier despliegue realizado, de forma correcta. Esta protección se debe que en dicho fichero se encuentran determinadas credenciales que permiten establecer conexión con la base de datos del proyecto, alojado en la cuenta personal de Google del creador. A continuación se provee una estructura básica del contenido de este archivo, sin mostrar dichas credenciales.

```
export default {
  apiKey: '',
  authDomain: '',
  databaseURL: '',
  projectId: '',
  storageBucket: '',
  messagingSenderId: '',
  appId: ''
};
```

B.3. Proceso de preparación del entorno y despliegue del bot de Telegram

Disponiendo del código (el cual puede hallarse en `https://gitlab.com/JAlaguero_5/dielthybot`), en el directorio raíz del proyecto y en la carpeta `functions` debe ejecutarse `npm install` para instalar todas las dependencias y módulos que se requieren. Después de eso puede ejecutarse `npm run serve` (alias para `firebase emulators:start --only functions`), lo que desplegará de forma local un servidor, de manera similar a como se explica en el Apéndice B.2 para la web.

Si lo que se pretende es desplegar la función de forma remota para ser accedida a través de Telegram, cosa que no es posible si se encuentra desplegada de forma local, debe iniciarse sesión en Firebase CLI mediante `firebase login`, si es que no se ha hecho ya con anterioridad en el sistema. Esto se debe a que el código de la función se aloja en el mismo proyecto que la web, en el apartado functions, por lo que debe ser necesario este inicio de sesión y, además, debe hacerse con una cuenta con permisos suficientes en el proyecto para realizar el despliegue pretendido. Posteriormente, basta con ejecutar `npm run deploy` (alias para `firebase deploy --only functions`).

El bot puede ser encontrado en @DielthyBot en Telegram.

B.4. Scripts adicionales útiles

Además de los scripts ya comentados en la Sección B.2, pueden encontrase otros adicionales de utilidad para otras tareas:

- `npm run test:unit`. Permite ejecutar los tests unitarios. Es un alias para `vue-cli-service test:unit`.
- `npm run test:e2e`. Posibilita la ejecución de los tests end-to-end. Es un alias para `vue-cli-service test:e2e`.

- `npm run test:e2e-ci`. Realiza la misma acción que el script anterior, sólo que en este caso se pasa como parámetro `--headless` para ejecutar estos tests sin interfaz gráfica.

Apéndice C

Resumen de enlaces adicionales

Los enlaces útiles de interés en este Trabajo Fin de Grado son:

- `https://gitlab.com/JAlaguero_5/dielthybot`. Repositorio que contiene el código del bot de Telegram, estructurado de la forma que se explica en la Sección 7.2.
- `https://gitlab.com/JAlaguero_5/dielthyweb`. Repositorio en el que se aloja el código de la web, organizado como se explica en la Sección 7.1.
- `https://tfg-dielthy.web.app`. Enlace en el que puede encontrarse la versión desplegada de la web.

Bibliografía

[1] Página oficial de Overleaf, *Documentación de Overleaf*, Consultado a lo largo del desarrollo de toda la documentación, Disponible en https://www.overleaf.com/learn/latex/Main_Page

[2] Universidad de Valladolid, *Guía docente del Trabajo Fin de Grado, Grado en Ingeniería Informática, Mención en Ingeniería de Software, Universidad de Valladolid*, Consultado el 15/10/2019, Disponible en https://www.alojamientos.uva.es/guia_docente/uploads/2019/545/46976/1/Documento.pdf

[3] cocinarvegano, *Web oficial cocinarvegano*, Consultado el 09/10/2019, Disponible en https://www.cocinarvegano.com

[4] e-aditivos, *Web oficial e-aditivos*, Consultado el 10/10/2019, Disponible en https://www.e-aditivos.com

[5] aditivosalimentarios, *Web oficial aditivos-alimentarios*, Consultado el 10/10/2019, Disponible en https://www.aditivos-alimentarios.com

[6] Ken Schwaber y Jeff Sutherland, *La Guía de Scrum*, Consultado el 14/10/2019, Disponible en https://www.scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-Spanish.pdf

[7] Página oficial de Git, *Git*, Consultado el 17/12/2019, Disponible en https://www.git-scm.com/doc

[8] Samuel Alfageme, *Integración Continua rápida y sencilla con GitLab CI*, Consultado el 22/01/2020, Disponible en https://www.solidgeargroup.com/gitlab_countinuous_integration_intro/?lang=es

[9] Página oficial de Vue.js, *Vue.js*, Consultado el 22/01/2020, Disponible en https://www.vuejs.org/v2/guide/

[10] Miguel Ángel Álvarez, *Qué es una SPA*, Consultado el 22/01/2020, Disponible en https://www.desarrolloweb.com/articulos/que-es-una-spa.html

[11] Yanina Muradas, *Qué es NPM y para qué sirve*, Consultado el 23/01/2020, Disponible en https://www.openwebinars.net/blog/que-es-node-package-manager/

[12] NPM archive, *vue/cli-plugin-unit-jest*, Consultado el 23/01/2020, Disponible en https://www.npmjs.com/package/@vue/cli-plugin-unit-jest

[13] Página oficial de Jest, *Jest*, Consultado el 23/01/2020, Disponible en https://www.jestjs.io/

[14] NPM Archive, *vue/cli-plugin-vuex*, Consultado el 23/01/2020, Disponible en https://www.npmjs.com/package/@vue/cli-plugin-vuex

[15] Página oficial de Vue.js, *Vuex*, Consultado el 23/01/2020, Disponible en https://vuex.vuejs.org/

[16] NPM Archive, *vue/cli-plugin-router*, Consultado el 23/01/2020, Disponible en https://www.npmjs.com/package/@vue/cli-plugin-router

[17] Página oficial de Vue.js, *Vue Router*, Consultado el 23/01/2020, Disponible en `https://router.vuejs.org/guide/`

[18] NPM Archive, *vue/cli-plugin-e2e-cypress*, Consultado el 23/01/2020, Disponible en `https://www.npmjs.com/package/@vue/cli-plugin-e2e-cypress`

[19] Página oficial de Cypress, *Cypress*, Consultado el 23/01/2020, Disponible en `https://docs.cypress.io/guides/`

[20] NPM Archive, *vuetify*, Consultado el 23/01/2020, Disponible en `https://www.npmjs.com/package/vuetify`

[21] Página oficial de Vuetify, *Vuetify.js*, Consultado a lo largo de todo el proyecto, Disponible en `https://www.vuetifyjs.com/es-MX/`

[22] Página oficial de Gitlab, *Issues*, Consultado el 27/01/2020, Disponible en `https://docs.gitlab.com/ee/user/project/issues/`

[23] Adrián Alonso Vega, *Backend as a Service o cómo prescindir de un backend developer*, Consultado el 27/01/2020, Disponible en `https://www.medium.com/codenares/backend-as-a-service-o-como-prescindir-de-un-backend-developer-facdde6e3132`

[24] George Batschinski, *What is a BaaS*, Consultado el 28/01/2020, Disponible en `https://www.blog.back4app.com/2019/07/24/backend-as-a-service-baas/`

[25] Página oficial de Firebase, *Firebase*, Consultado el 28/01/2020, Disponible en `https://www.firebase.google.com/`

[26] Valentina Giraldo, *¿Qué es Firebase?*, Consultado el 28/01/2020, Disponible en `https://www.rockcontent.com/es/blog/que-es-firebase/`

[27] Página oficial de Astah, *Astah*, Consultado el 29/01/2020, Disponible en `http://www.astah.net/`

[28] Pagína oficial de UML, *What is UML*, Consultado el 29/01/2020, Disponible en `https://www.uml.org/what-is-uml.htm`

[29] Página oficial de InVision, *InVision*, Consultado a lo largo de todo el proyecto, Disponible en `https://www.invisionapp.com/`

[30] Bob Hughes and Mike Cotterell, *Software Project Management*, 5th ed. Europa, Oriente Medio y África: McGraw-Hill Education, 2009, pp 163-166. Consultado el 29/01/2020

[31] Ministerio de Empleo y Seguridad Social, Gobierno de España, *XVII Convenio colectivo estatal de empresas de consultoría, y estudios de mercado y de la opinión pública*, Consultado el 30/01/2020, Disponible en `https://www.boe.es/boe/dias/2018/03/06/pdfs/BOE-A-2018-3156.pdf`

[32] Javier Santos Pascualena, *¿Cuánto cuesta contratar un trabajador?*, Consultado el 30/01/2020, Disponible en `https://www.infoautonomos.com/blog/cuanto-cuesta-contratar-un-trabajador/`

[33] Página oficial de Trello, *Trello*, Consultado a lo largo del desarrollo del proyecto, Disponible en `https://www.trello.com/es`

[34] Noemí Alba, *¿Qué porcentaje de la población española es vegana?*, Consultado el 04/02/2020, Disponible en `https://www.buenoyvegano.com/2018/11/09/porcentaje-poblacion-espanola-vegana/`

[35] Europa Press, *Un 25 % de la población padece alguna intolerancia alimentaria*, Consultado el 04/02/2020, Disponible en `https://www.heraldo.es/noticias/salud/2018/10/05/un-poblacion-padece-alguna-intolerancia-alimentaria-1270302-2261131.html`

[36] Universidade de Vigo, *Scrum*, Consultado el 14/10/2019, Disponible en `https://www.desire.webs.uvigo.es/contenidos/scrum/`

[37] Joel Francia, *¿Qué es Scrum?*, Consultado el 14/10/2019, Disponible en `https://www.scrum.org/resources/blog/que-es-scrum`

[38] Página oficial de Vue.js, *Reactividad en profundidad*, Consultado el 07/02/2020, Disponible en `https://es.vuejs.org/v2/guide/reactivity.html`

[39] Página oficial de Vue.js, *La instancia Vue*, Consultado el 07/02/2020, Disponible en `https://es.vuejs.org/v2/guide/instance.html`

[40] David Onieva, *Qué es el software SaaS y qué ventajas nos puede aportar*, Consultado el 10/02/2020, Disponible en `https://www.softzone.es/2018/08/06/software-saas-ventajas-aportar/`

[41] Página oficial Webscraper, *Webscraper*, Consultado el 10/02/2020, Disponible en `https://webscraper.io/`

[42] Marq Martí, *¿Qué es el Web scraping? Introducción y herramientas*, Consultado el 10/02/2020, Disponible en `https://sitelabs.es/web-scraping-introduccion-y-herramientas/`

[43] Agencia Tributaria, *Tabla de coeficientes de amortización lineal*, Consultado el 10/02/2020, Disponible en `https://www.agenciatributaria.es/AEAT.internet/Inicio/_Segmentos_/Empresas_y_profesionales/Empresas/Impuesto_sobre_Sociedades/Periodos_impositivos_a_partir_de_1_1_2015/Base_imponible/Amortizacion/Tabla_de_coeficientes_de_amortizacion_lineal_.shtml`

[44] Shirivo, *¿Modelo-Vista-Vista Modelo? ¿Eso es un patrón?*, Consultado el 13/02/2020, Disponible en `https://shirivo.wordpress.com/2015/06/30/modelo-vista-vista-modelo-eso-es-un-patron/`

[45] Adictos al trabajo, *MVC y MVVM*, Consultado el 15/02/2020, Disponible en `https://www.adictosaltrabajo.com/2012/10/07/zk-mvc-mvvm/`

[46] Página oficial de Vue.js, *Getting Started*, Consultado el 15/02/2020, Disponible en `https://012.vuejs.org/guide`

[47] Página oficial de Vue.js, *Directives*, Consultado el 15/02/2020, Disponible en `https://012.vuejs.org/guide/directives.html`

[48] Adrián Alonso Vega, *Atomic Web Design o Diseño Guiado por Componentes*, Consultado el 15/02/2020, Disponible en `https://adrianalonso.es/arquitectura-del-software/atomic-web-design-o-diseno-guiado-por-componentes/`

[49] Brad Frost, *Atomic Design*, Consultado el 15/02/2020, Disponible en `https://bradfrost.com/blog/post/atomic-web-design/`

[50] Javier Rodríguez, *Tutorial de Git y GitHub para uso de control de versiones*, Consultado el 17/12/2019, Disponible en `https://www.javierrguez.com/tutorial-basico-git-github-uso-control-versiones/`

[51] Cloudflare, *What is BaaS?*, Consultado el 28/01/2020, Disponible en `https://www.cloudflare.com/learning/serverless/glossary/backend-as-a-service-baas/`

[52] Óscar Blancarte, *Qué es el Backend-as-a-Service (BaaS)*, Consultado el 28/01/2020, Disponible en `https://www.oscarblancarteblog.com/2018/06/18/backend-as-service-baas/`

[53] NPM Archive, *mdi/js*, Consultado el 24/02/2020, Disponible en `https://www.npmjs.com/package/@mdi/js`

[54] Material UI, *Material Design Icons*, Consultado el 24/02/2020, Disponible en `https://materialdesignicons.com/`

[55] kazupon, *i18n*, Consultado el 04/03/2020, Disponible en `https://kazupon.github.io/vue-i18n/`

[56] Wikipedia, *Pandemia de enfermedad por coronavirus de 2019-2020*, Consultado el 17/03/2020, Disponible en `https://es.wikipedia.org/wiki/Pandemia_de_enfermedad_por_coronavirus_de_2019-2020`

[57] Carlos Lucena Herrera, *Qué es el Testing de Software*, Consultado el 06/04/2020, Disponible en `https://openwebinars.net/blog/que-es-el-testing-de-software/`

[58] Jessie Leung, *The Test Pyramid*, Consultado el 06/04/2020, Disponible en `https://medium.com/better-programming/the-test-pyramid-80d77535573`

[59] Andreas Hinderks, Martin Schrepp y Jörg Thomaschewski, *User Experience Questionnaire*, Consultado el 19/04/2020, Disponible en `https://www.ueq-online.org/`

[60] Los Andes Training, *¿Qué son las pruebas de aceptación?*, Consultado el 19/04/2020, Disponible en `https://losandestraining.com/2017/08/23/que-son-las-pruebas-de-aceptacion/`

[61] Google, *Google Forms*, Consultado el 21/04/2020, Disponible en `https://www.google.es/intl/es/forms/about/`

[62] ExpressJS, *Express*, Consultado el 24/04/2020, Disponible en `https://expressjs.com/es/`

[63] Software Testing Fundamentals, *System Testing*, Consultado el 25/04/2020, Disponible en `http://softwaretestingfundamentals.com/system-testing/`

[64] Google, *Google Cloud Armor*, Consultado el 13/05/2020, Disponible en `https://cloud.google.com/armor`

[65] Stackoverflow, *Are Google Cloud Functions Protected from DDoS attacks?*, Consultado el 13/05/2020, Disponible en `https://stackoverflow.com/questions/47948561/are-google-cloud-functions-protected-from-ddos-attacks`

[66] Frederic Lardinois, *Google brings DDoS protection and other new security features to its cloud*, Consultado el 13/05/2020, Disponible en `https://techcrunch.com/2018/03/21/google-brings-new-security-features-to-its-cloud/`

[67] Creative Commons, *Atribución-NoComercial-SinDerivadas*, Consultado el 15/05/2020, Disponible en `https://creativecommons.org/licenses/by-nc-nd/4.0/deed.es`

[68] LexGoApp, *Cómo se protege el software: ¿propiedad intelectual o industrial?*, Consultado el 24/05/2020, Disponible en `https://lexgoapp.com/blog/como-se-protege-el-software-propiedad-intelectual-o-industrial/`

[69] Laura Sacristán, *Telegram alcanza los 400 millones de usuarios y se prepara para ofrecer videollamadas grupales este año*, Consultado el 04/06/2020, Disponible en `https://www.xatakamovil.com/aplicaciones/telegram-alcanza-400-millones-usuarios-se-prepara-para-ofrecer-videollamadas-grupales-este-ano`

[70] Michael Carraz *et al.*, *State of the Developer Nation*, 16th Ed., 2019, pp 30-33. Consultado el 04/06/2020. Disponible en `https://slashdata-website-cms.s3.amazonaws.com/sample_reports/ZAamt00SbUZKwB9j.pdf`

[71] José Huerta, *¿Qué es una épica qué no es?*, Consultado el 04/06/2020, Disponible en `https://josehuerta.es/gestion/agile/que-es-una-epica-y-que-no-es`

[72] State of agile, *State of agile report*, Consultado el 05/06/2020, Disponible en `https://stateofagile.com/#ufh-i-615706098-14th-annual-state-of-agile-report/7027494`