Planificador de menús para *veggies* “Menureverde”

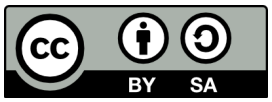
José Luis Plaza Molina
Grado Multimedia
TFG - Aplicaciones Interactivas

Consultor: Kenneth Capseta Nieto
Profesor: Carlos Casado Martínez
11 de junio de 2018

Créditos

Memoria

Aplicación Web



Aplicaciones de terceros

A pesar de que, tanto la memoria como la aplicación web están bajo licencias *Creative Commons*, se utilizan herramientas de terceros:

Frameworks HTML, CSS, JavaScript:

- Vue (https://vuejs.org/)
- Bulma (https://bulma.io/)
- Node.js (https://nodejs.org)
- Firebase (https://firebase.google.com/)

Abstract

Es un hecho que, en España, Europa y en el resto de países occidentales crece el número de personas que deciden ser vegetarianos o veganos. Además, entre los no vegetarianos, crece el número de personas que han disminuido el consumo de alimentos procedentes de animales. En resumen, existe una mayor sensibilidad por los alimentos que ingerimos, su procedencia y por el medio ambiente en general, ya que la industria cárnica es un factor clave en la contaminación atmosférica.

Por otro lado, existe mucha controversia entre si las dietas veganas son saludables o no. En realidad, una dieta vegana puede ser igual de mala que una dieta omnívora donde solo se coman productos procesados cargados de azúcar, fritos, etc. Lo que sí que es verdad, sin embargo, es más fácil caer en la falta de nutrientes en una dieta vegana que en una dieta omnívora, por eso es recomendable informarse bien antes de empezar, o bien, acudir a algún dietista / nutricionista bien formado que le pueda asesorar.

Como consecuencia de lo anteriormente expuesto he decidido realizar una aplicación web donde las personas que decidan llevar una dieta vegana, o vegetariana, puedan elaborar un menú semanal con recetas de los mejores blogs nacionales. En la misma aplicación obtendrán información nutricional con la que podrán ver si falta o sobra cualquier nutriente y poder solucionarlo añadiendo o quitando recetas del menú.

Para realizar esta aplicación web, en cuanto a los aspectos técnicos, se ha optado por utilizar JavaScript tanto en la parte cliente como en la servidora, además de HTML5 y CSS, como lenguajes de marcado y estilos respectivamente. Además, se han utilizado varios *frameworks* de última generación, como Vuejs, que permite dividir la aplicación en componentes y facilita la comunicación entre estos, Firebase, para la autenticación y base de datos o Bulma, como *framework* CSS.

Finalmente, el desarrollo de la aplicación no va a seguir una metodología concreta, aunque sí que bebe de las metodologías ágiles, teniendo una versión alfa limitada funcionalmente en las primeras semanas, pudiendo así obtener *feedback* por parte del profesor y usuarios. De esta forma se irán incorporando funcionalidades poco a poco hasta tener la versión final.

Abstract (English version)

It is a fact that in Spain, Europe and many other western countries, the number of people who decide to become vegetarian or vegan is growing. Furthermore, among those who are not vegetarian, the amount of people who have reduced their consumption of meat and animal products is also on the rise. In summary, there is a greater awareness about the food that we eat, where it comes from, and of the environment in general, given that the meat industry plays a key role in air pollution.

On the other hand, there is a lot of controversy about vegan diets and whether they are healthy or not. In fact, a vegan diet can be just as unhealthy as an omnivorous diet if individuals only eat processed foods that are full of sugar or deep fried. What is true, however, is that it can be more difficult to obtain all the necessary nutrients from a vegan diet than an omnivorous one, therefore it is advisable for people to inform themselves before starting such a diet, or seek advice from a professional dietitian or nutritionist who can offer guidance.

Considering the preceding information, it was decided to create a web application wherein people who decide to follow a vegan or vegetarian diet can devise a weekly menu with recipes from the best Spanish blogs. In this application, users will obtain nutritional information and will be able to see if they are lacking any nutrients or exceeding their intake, and will be able to resolve this by adding or removing recipes from their menu.

To make this web application, regarding the technical aspects, JavaScript has been utilized both for the client side and the server side, as well as HTML5 and CSS as markup language and style, respectively. In addition, the latest generations of several frameworks have been used, such as Vue.js, which allows the application to be divided into components and facilitates communication between them, Firebase, for authentication and database, and Bulma as CSS framework.

Finally, the development of the application will not follow a specific methodology, although it does use agile methodologies, having an alpha version functionally limited in the first weeks, thereby being able to obtain feedback from the course tutor and users. In this way, features will be incorporated gradually until the final version is available.

Notaciones y convenciones

Títulos	**Times 16 pt**
Subtítulos	**Times 14 pt**
Texto cuerpo	Arial 11 pt
Idioma extranjero	*Arial cursiva 11 pt*
`Código fuente`	`Consolas 10 pt`
Encabezado y pies de página	Arial 9 pt

Índice

1. Introducción

Nos os engaño si os digo que hará por lo menos quince años que hice mi primer curso de HTML y CSS, la verdad es que siempre me ha apasionado este mundo, pero nunca he podido profundizar en él ya que mi trayectoria profesional iba por otros derroteros. Pasado ya un tiempo, al cambiar de trabajo tuve más tiempo libre y decidí unirme al grado multimedia, desde entonces siempre me he ido decantando más por las asignaturas de programación que por las de diseño.

Ahora, finalizando el grado multimedia como estoy, sigo sin poder considerarme un desarrollador web, se trata de un terreno muy difícil y cambiante y hay que estar en continua formación. Es por todo esto que he decidido realizar una aplicación web, con el fin de seguir aprendiendo y mejorando en este terreno.

Por otro lado, mi vida cambió cuando me detectaron celiaquía, una enfermedad autoinmune que normalmente ataca al intestino en presencia de gluten. Entonces, empecé a hacerme preguntas, ¿Por qué cada día hay más intolerancias, enfermedades autoinmunes, alergias, etc.? Y comencé a cuidar la alimentación, a cambiar hábitos de vida para controlar el estrés, me hice más consciente del mundo que nos rodea y nuestra relación con él.

A causa de esto, me convencí de que mi alimentación debía aproximarse al veganismo por mi salud y por la del planeta. Pero enseguida me encontré con muchas dudas sobre la alimentación: si no bebes leche te va a faltar calcio, falta de proteínas, Vitamina B12, vitamina D etc. Muchas de ellas infundadas, pero otras no tanto.

Todo esto me ha llevado a realizar esta aplicación web donde los vegetarianos y veganos pueden elaborar sus menús con las recetas de los mejores blogueros nacionales e internacionales y despejar así cualquier duda sobre la falta de nutrientes del mismo.

Enlace a la aplicación web: https://menureverde.firebaseapp.com

2. Descripción

Como ya se ha adelantado en apartados anteriores, este proyecto pretende diseñar y construir una aplicación web donde los usuarios puedan buscar recetas y organizar su menú semanal, obteniendo, además, información nutricional y alarmas en caso de falta de nutrientes. Para su realización se ha optado por diversas soluciones, algunas de las cuales se detallan a continuación:

- **Diseño y desarrollo del *frontend***. Con el fin de acelerar el trabajo y, además, aprender el uso de herramientas actuales, se ha utilizado Bulma como *framework* CSS y el ecosistema Vuejs como *framework* Javascript, el cual permite trabajar con componentes de una manera muy efectiva, así como poder crear una SPA (*Single Page Aplication*). Este diseño se podrá ver en cualquier dispositivo: *Responsive Design.*

- **Gestión de usuarios**: Para permitir que los usuarios de la aplicación puedan guardar sus recetas favoritas y elaborar sus menús se hace necesario crear el módulo de gestión de usuarios. Para ello se ha utilizado FirebaseUI Auth, en la página de Firebase podemos leer lo siguiente:

  "*FirebaseUI proporciona una solución de autenticación directa que controla los flujos de IU para los usuarios que acceden con direcciones de correo electrónico y contraseñas, números de teléfono y con proveedores de identidad federada populares, que incluyen el Acceso con Google y el Acceso con Facebook."*

  Los usuarios, una vez logados, podrán buscar desayunos, comidas, meriendas o cenas y organizarlos en su menú semanal, además de guardar sus recetas favoritas.

- **Base de datos**: Se hace necesario tener una base de datos donde enlazar a los usuarios con sus preferencias y dónde poder almacenar sus menús. Por eso se ha optado por Firebase de nuevo, en concreto Firebase Data Base.

- **APIS**: Para este proyecto se ha utilizado la API buscador de recetas de Edamam, la cual permite obtener información nutricional de los alimentos.

- **Gestión de menús:** Una vez logados los usuarios pueden ver todas las recetas ordenadas por popularidad, en esta primera pantalla hay un buscador donde se puede buscar y filtrar por tipos de alimentos, intolerancias, etc. Después, se pueden agregar las recetas al día que elijamos arrastrando y soltando.

- **Servidor de desarrollo / Servidor producción:** Se ha utilizado vue-cli. Así mediante Node y NPM se puede iniciar un proyecto rápidamente con todas las dependencias necesarias. NPM tiene su propio servidor de desarrollo, el cual se puede ejecutar desde el terminal, escribiendo `npm run dev`. Como servidor de producción se ha optado finalmente Firebase Hosting.

3. Objetivos

3.1. Objetivos generales

Como Trabajo Final de Grado, este proyecto tiene como objetivo la puesta en valor de todo lo aprendido durante el Grado Multimedia. Este objetivo se concreta en los siguientes puntos:

- Aprender a realizar una aplicación web únicamente con HTML 5, CSS 3 y JavaScript, usando los marcos de desarrollo más utilizados en este momento.
- Planificar y estructurar de manera realista en tareas asumibles en el tiempo utilizando alguna metodología actual.
- Elaborar una memoria del proyecto según una estructura prefijada.
- Elaborar una presentación del desarrollo y resultados finales del proyecto.

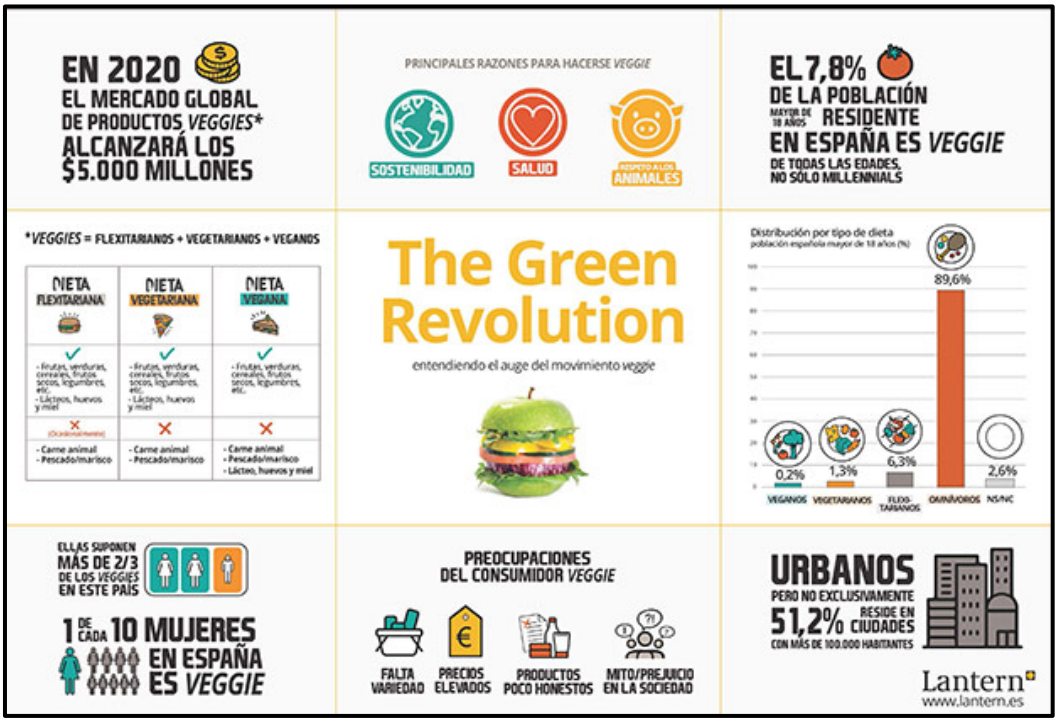
3.2. Objetivos específicos

Además de los objetivos generales se pueden destacar unos objetivos específicos en este proyecto, son los siguientes:

- Realizar eficientemente una aplicación web responsiva para planificar menús veganos y vegetarianos.
- Crear una base de datos de alimentos y otra de recetas con información nutricional con referencias a blogs de prestigio nacional e internacional.
- Crear un solución óptima y adecuada para que la página la pueda utilizar cualquier persona de manera sencilla.
- Crear una propuesta de diseño que atraiga a los potenciales usuarios.

4. Marco teórico / Escenario

Según un estudio de la consultora Lantern, llamado "*The Green Revolution*" cerca del 8% de la población en España se declara *Veggie*, que es un anglicismo que se utiliza para enmarcar tanto a veganos y vegetarianos, como a los flexitarianos, que solo comen productos de procedencia animal esporádicamente. Se trata de una tendencia a nivel mundial estando alejados todavía del 12 % de población que opta por este tipo de dietas en el Reino Unido. Además, este porcentaje aumenta claramente entre los jóvenes, lo que hace indicar que es una tendencia imparable.



Siguiendo con este reciente estudio, se pueden saber los motivos para adoptar este tipo de dietas. El principal motivo es esencialmente ético y se basa en el respeto a la vida animal. Como segundo motivo tenemos la creciente consciencia en torno a la sostenibilidad de nuestro planeta, ya que es sabido que los recursos para generar un kilo de carne son mucho más costosos y contaminantes que los necesarios para producir un kilo de legumbres. Finalmente, un tercer aspecto es la mejora de la salud personal, en el año 2015 la OMS publicó un informe vinculando el consumo excesivo de carne a una mayor predisposición a contraer enfermedades coronarias o incluso cáncer.

El estudio no viene si no a confirmar lo que cada uno de nosotros percibimos en nuestro día a día. No hay más que observar como las cadenas de supermercados se llenan de productos para *veggies* y es que la industria se ha dado cuenta que los que siguen este tipo de dietas están dispuestos a pagar más por los productos:

Como se ha visto anteriormente una de las razones para hacerse *veggie* es la salud. Sin embargo, hay mucha información contradictoria, incluso de fuentes profesionales como médicos y nutricionistas, a pesar de haber números estudios serios que avalan la alimentación vegana en cualquier edad. Uno de los principales motivos es la industria cárnica y lechera que invierte millones de dólares o de euros en lavar su imagen contratando para ello numerosos profesionales de la salud.

En este contexto, nace Menureverde, donde consultando bases de datos de alimentos públicas se podrán elaborar menús semanales con información nutricional, para que no quede ningún tipo de duda..

5. Contenidos

Cuando se accede a la aplicación aparece la cabecera de la página web y un buscador. A partir de aquí es posible acceder a cualquiera de los siguientes escenarios:

1. **Inicio.** Se trata de la página de presentación. En ella se puede ver un buscador. Si se realiza una búsqueda, aparecerá un listado con las coincidencias, con la información básica de cada receta. Además, se pueden realizar las siguientes operaciones de filtrado:

    - Solo veganas
    - Sin gluten
    - Sin lactosa
    - Sin soja

    1.1. **Vista de receta.** Esta será la vista que se podrá ver si se selecciona cualquier receta del listado de recetas. Desde aquí se podrá ver la descripción completa de la receta, así como su información nutricional. Lo que no se verá desde aquí será la elaboración, para eso habrá un enlace al blog del chef creador de la receta. En esta vista también se podrá agregar la receta como favorita, siempre que el usuario esté correctamente logado.

2. **Mis recetas.** Esta sección muestra el listado de recetas guardadas como favoritas, siempre que el usuario esté correctamente logado

3. **Nosotros.** Página donde se da a conocer al autor y su motivación para realizar el proyecto

4. **Contacto.** Aquí aparecen los métodos de contacto

5. **Login.** Pulsando en el botón de Login aparecerá un *popup* que permite la autenticación con una cuenta de Google.

6. Metodología

Al tratarse de un trabajo individual es difícil elaborar una metodología ágil como Scrum, que están orientadas a equipos de trabajo. Debido a esto la metodología que se ha usado es la de dividir el trabajo en tareas y subtareas y asignarles un marco temporal. Después de eso se ha elaborado un diagrama de Gantt con la herramienta Gantter. Todo esto está más desarrollado en el apartado de planificación, aunque a modo de introducción estás son las tareas principales en las que se divide el proyecto:

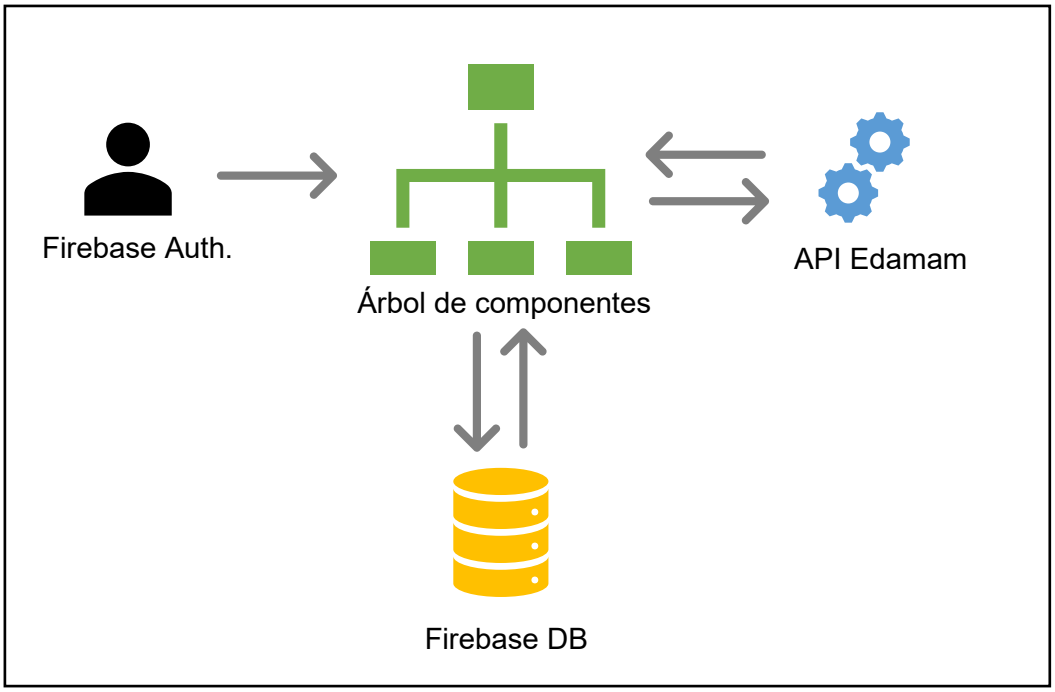
- Planificación, 6 días.
- Login de usuarios, 10 días.
- Guardar recetas como favoritas, 16 días.
- Filtros de búsqueda, 11 días.
- Planificador, 26 días.
- Presentación TFG, 12 días.

Hay que tener en cuenta que hay parte del trabajo realizado en el semestre anterior y que no se ha tenido en cuenta.

Finalmente, será el tutor, mediante las pruebas de evaluación continua el que vaya comprobando que el trabajo se está realizando según lo planeado y proponga actualizaciones, como si se tratara del cliente.

7. Arquitectura de la aplicación

La aplicación presentada es del tipo ***Single Page Application*** (Aplicación de Página Única), donde no es necesario recargar la página cada vez que se realiza una petición. Para poder realizarla se ha hecho uso del *framework* Vuejs, el cual permite la creación de una aplicación a modo de árbol de componentes.



Por otro lado, se utilizan varios servicios de Firebase, como es el *hosting*, la autenticación y la base de datos en tiempo real para poder guardar las preferencias de los usuarios registrados. En los siguientes puntos se detallan más todos estos aspectos

7.1. Vue.js

VueJS, es el *framework* JavaScript utilizado en este proyecto. Se trata de un *framework* progresivo, en el que, dependiendo de la complejidad del proyecto se irían agregando diferentes módulos.

El módulo principal sería el *core* de VueJS, el cual se encarga de renderizar componentes en el navegador. Estos componentes son pequeñas piezas de código con HTML, CSS y JavaScript que se organizan jerárquicamente formando lo que se denomina árbol de componentes.

Las principales características de VueJS serían las siguientes:

- **Patrón MVVM** (*model–view–viewmodel*). Este patrón trata de separar al máximo la interfaz de usuario de la lógica de la aplicación. En VueJS esto se traduce en que és el propio *framework* el que se encarga de renderizar el HTML. El programador se

puede centrar entonces en el uso de los datos, métodos y plantillas HTML y la lógica que permita indicar que elemento es el que se tiene que renderizar en cada momento.

- **Reactividad.** Esto quiere decir que los datos y el DOM están enlazados. Por lo tanto, si se cambia el valor de los datos, automáticamente se modificará el DOM con la lógica que le se le haya proporcionado.

- **Único Fichero**. Los componentes en VueJS tienen extensión vue y, a diferencia de React contienen el HTML, el CSS y el JavasCript separados por etiquetas.

7.2. Entorno de programación

VueJS, al igual que Angular, proporciona una herramienta de terminal muy útil para construir rápidamente la estructura de las aplicaciones, **vue-cli**. De esta manera para crear la estructura de la aplicación y las primeras dependencias escribiríamos los siguiente:

```
vue init webpack-simple tfg_uoc
```

Una vez creada la estructura se procede a instalar todas las dependencias. Para ello se accede a la carpeta generada en el paso anterior y se ejecuta:

```
npm install
```

Primera conexión con el repositorio de Git https://bitbucket.org/maestrilloliendre/tfg_uoc:

```
git init
git add .
git commit
git commit -m "vue init"
git remote add origin https://bitbucket.org/maestrilloliendre/tfg_uoc
git push origin master
```

Se instala ESlint que es un linter de código que permite crear reglas de estilo para que todo el trabajo tenga el mismo estilo.

```
npm i -D eslint // La opción -D es para que no se instale en producción
```

Se instala la configuración standarjs https://github.com/standard/eslint-config-standard
Y tal y como pone en la información del repositorio se ejecuta:

```
npm install --save-dev eslint-config-standard eslint-plugin-standard eslint-plugin-
promise eslint-plugin-import eslint-plugin-node
```

Se añade también un plugin para que eslint funcione bien con html:

```
npm i -D eslint-plugin-html
```

se crea el fichero .eslintrc con la siguiente información:

```
{
    "extends": "standard",

    "rules": {
        "no-new": 0
    },
    "plugins": ["html"]
}
```

Se integra con Webpack para que en el momento de desarrollo ofrezca esa información:

```
npm i -D eslint-loader
```

Se añade la configuración a webpack.config.js en el apartado de *rules*:

```
{
        test: /\.(js|vue)$/,
        loader: 'eslint-loader',
        enforce: 'pre',
        include: [path.resolve(__dirname, './src')]
},
```

7.3. Utilización de SASS, BULMA y PUG

Al crear la aplicación con vue-cli ya se configuró para que se utilizara SASS como precompilador de CSS. Por lo tanto solo se tiene que crear la carpeta scss y el archivo main.scss dentro.

Para incorporarlo en la aplicación se incorpora en el archivo principal de la aplicación, app.vue:

```
<style lang="scss">
  @import './scss/main.scss'
</style>
```

En los siguientes pasos se describe la instalación de Bulma, que es un *framework* CSS de código abierto basado en Flexbox. Por lo tanto, para instalarlo se utilizarían los sigiuentes comandos:

```
Npm i -S bulma   //-S la dependencia sí que estará en producción
```

Para incorporar Bulma en el proyecto hay que importarlo en el main.scss:

```
@import '../../node_modules/bulma/bulma.sass';
```

También se ha utilizado un preprocesador de HTML llamado Pug, el cual permite escribir mucho menos código:

```
npm i -D pug pug-loader
```

para utilizar pug en las plantillas, hay que indicarlo de la siguiente manera:

```
<template lang="pug">
```

7.4. Firebase Hosting

Firebase Hosting es un servicio que permite alojar contenido estático de manera muy sencilla y gratuita hasta cierto volumen de transferencia. Para empezar a utilizarlo lo primero que hay que hacer es instalar las herramientas de Firebase:

```
npm install -g firebase-tools
```

Nos logamos con nuestra cuenta de Google:

```
firebase login
```

posteriormente se instala todo lo necesario para poder lanzar la aplicación:

```
firebase init
```

Finalmente, se despliega la aplicación con este simple comando:

```
firebase deploy
```

Este comando sube la aplicación a los servidores de Firebase y la hace accesible a través de la siguiente dirección:

https://menureverde.firebaseapp.com/

Ahora podemos acceder a través de la consola de Firebase para configurar nuestro alojamiento pudiendo acceder al historial de implementaciones y poder revertir cualquier estado volviendo a uno anterior.

7.5. Autenticación de Firebase

Para la autenticación de usuarios se ha decidido utilizar Firebase Authentication. Esta herramienta proporciona servicios de autenticación de manera sencilla y con la ventaja de integrarse perfectamente con las bases de datos Firebase Realtime Database. A pesar de

soportar varios métodos de autenticación como email y contraseña, Facebook, Twitter, etc., en un primer momento, por sencillez y comodidad, solo se utilizará la autenticación mediante una cuenta de Google. Lógicamente, antes de empezar, hay que instalar Firebase en el entorno, esto se hace escribiendo en el terminal:

```
npm install firebase --save
```

Una vez instalado Firebase en nuestro proyecto hay que habilitar el servicio de autenticación a través de la consola de Firebase.

En el fichero `main.js` se realiza la importación de Firebase, se genera la instancia de Firebase y comprobamos si hay un usuario conectado:

```
let app

// Guardamos en la variable la configuración de Firebase
let config = {
  apiKey: 'AIzaSyDf9ivIVMAsdkEN30e8GAcIyVFghsgy5IE',
  authDomain: 'menureverde.firebaseapp.com',
  databaseURL: 'https://menureverde.firebaseio.com',
  projectId: 'menureverde',
  storageBucket: 'menureverde.appspot.com',
  messagingSenderId: '1071247826839'
}

// Inicializamos la instancia de Firebase
firebase.initializeApp(config)

// Comprobamos si existe algún usuario conectado y después iniciamos la
aplicación
firebase.auth().useDeviceLanguage()
firebase.auth().onAuthStateChanged(function (user) {
  if (!app) {
    app = new Vue({
      el: '#app',
      render: h => h(App),
      router
    })
  }
  if (user) {
    console.info('Conectado: ', user)
    app.autentificado = true
    app.usuarioActivo = user
  } else {
    console.warn('No conectado')
    app.autentificado = false
    app.usuarioAtivo = null
  }
```

```
})
```

En el apartado de código fuente se puede ver la configuración de las conexiones al pulsar el botón de *login* o de *logout* del componente Header.vue

7.6. Base de datos en tiempo real de Firebase

La mejor manera de explicar lo que es Firebase Realtime Database que he encontrado es con esta breve explicación de su página web:

> *"Firebase Realtime Database es una base de datos alojada en la nube. Los datos se almacenan en formato JSON y se sincronizan en tiempo real con cada cliente conectado. Cuando compilas apps multiplataforma con nuestros SDK de iOS, Android y JavaScript, todos tus clientes comparten una instancia de Realtime Database y reciben actualizaciones automáticamente con los datos más recientes."*

Como se puede comprobar en el apartado anterior ya está configurada la instancia de Firebase, ahora solo es necesario agregar una referencia a la base de datos, la cual está en el mismo fichero *main.js*. Para poder utilizarla en todos los componentes se ha optado por crear una constante y exportarla en los componentes que vayan a hacer uso de la base de datos:

```
// Obtener una instancia a la base de datos
export const database = firebase.database()
```

Sobre esta base se ha creado la estructura de la base de datos de recetas favoritas con la siguiente estructura de datos:

```
favs {
  referenciaFirebase {
    recipeUri: “”
    userUid: “”
  }
}
```

Toda la información guardada en la base de datos va a seguir ese patrón, donde referenciaFirebase es el código basado en el *timestamp* que genera automáticamente Firebase por cada registro. Para guardar las recetas favoritas por usuario, únicamente se necesitan un valor único de la receta, en este caso el URI, y del usuario necesitamos su ID. De esta forma asociamos esa receta a un usuario en concreto.

Para poder trabajar con la base de datos en todos los componentes lo que se ha hecho es realizar un *snapshot* de la base de datos justo en el momento que la instancia se ha creado y antes de que se haya montado. VueJS tiene un ciclo de vida y nos permite ejecutar código en todos sus estados. En este caso la lectura de la base de datos se realiza en el estado `created`. Por lo tanto, en todos los componentes que hagan uso de la base de datos tendremos lo siguiente:

```
  created () {
    database.ref('/favs')
      .on('value', snapshot => this.readDBfavs(snapshot.val()))
  },
```

Como podemos ver se está llamando al método `readDBfavs`, el cual se encarga de actualizar el array `favs[]`, a continuación el código:

```
  methods: {
    readDBfavs (dbFavs) {
      this.favs = []
      for (let key in dbFavs) {
        this.favs.push({
          recipeUri: dbFavs[key].recipeUri,
          userUid: dbFavs[key].userUid,
          key: key
        })
      }
      console.log(this.favs.length)
    },
```

De esta forma ya está todo preparado para poder trabajar con la base de datos en cada componente que haga uso de ella.

7.7. Vue Router

Vue Router es un paquete independiente del *core* de VueJS, el cual permite enlazar las distintas *URLs* con los componentes de la aplicación. A continuación, se pueden ver las distintas rutas configuradas en la aplicación:

```
import Search from '@/components/Search.vue'
import RecipeDetail from '@/components/RecipeDetail.vue'
import Planner from '@/components/Planner.vue'
import Misrecetas from '@/components/Misrecetas.vue'

const routes = [
  {
    path: '/',
    component: Search,
```

```
    name: 'search'
  },
  {
    path: '/misrecetas',
    component: Misrecetas,
    name: 'misrecetas'
  },
  {
    path: '/recipe/:r',
    component: RecipeDetail,
    name: 'recipe'
  },
  {
    path: '/recipefavs/:r',
    component: RecipeDetail,
    name: 'recipefavs'
  },
  { path: '/planner',
    component: Planner,
    name: 'planner'
  }
]

export default routes
```

8. Plataforma de desarrollo

Son varios los recursos que se están utilizando en el proyecto:

Entorno de desarrollo:

- **Portátil Dell Precission 5510**
- **Windows 10 con Ubuntu Bash como CLI**.
- **Visual Studio Code.** Editor de Código de Microsoft con licencia de software libre. Enlace Aquí.
- **Vue–cli sobre Node.js y NPM**. Aquí se pueden ver todas las dependencias del proyecto:
    - Babeljs: Transcompilador que nos permite escribir código con las versiones más recientes de Ecma Script, encargándose de traducirlo si el navegador no lo soporta.
    - Eslint: Se trata de un *linter* que se encarga de buscar fallos sintácticos en tiempo real.
    - Pugjs: Facilita la escritura de HTML.
    - Webpack: Se encarga de crear un único archivo JavaScript para producción.
- **Bulma.** *Framework* CSS para el *Frontend*. Enlace aquí.
- **Vuejs.** *Framework* JavaScript para entornos reactivos y Ajax. Enlace aquí.
- **Firebase.** Base de datos en tiempo real y en la nube. Enlace aquí.

Entorno de producción

Firebase Hosting. Resulta verdaderamente muy rápido pasar de desarrollo a producción. Estos son los pasos que se deben realizar:

```
# npm run build
# npm install -g firebase-tools
# firebase init
# firebase deploy
```

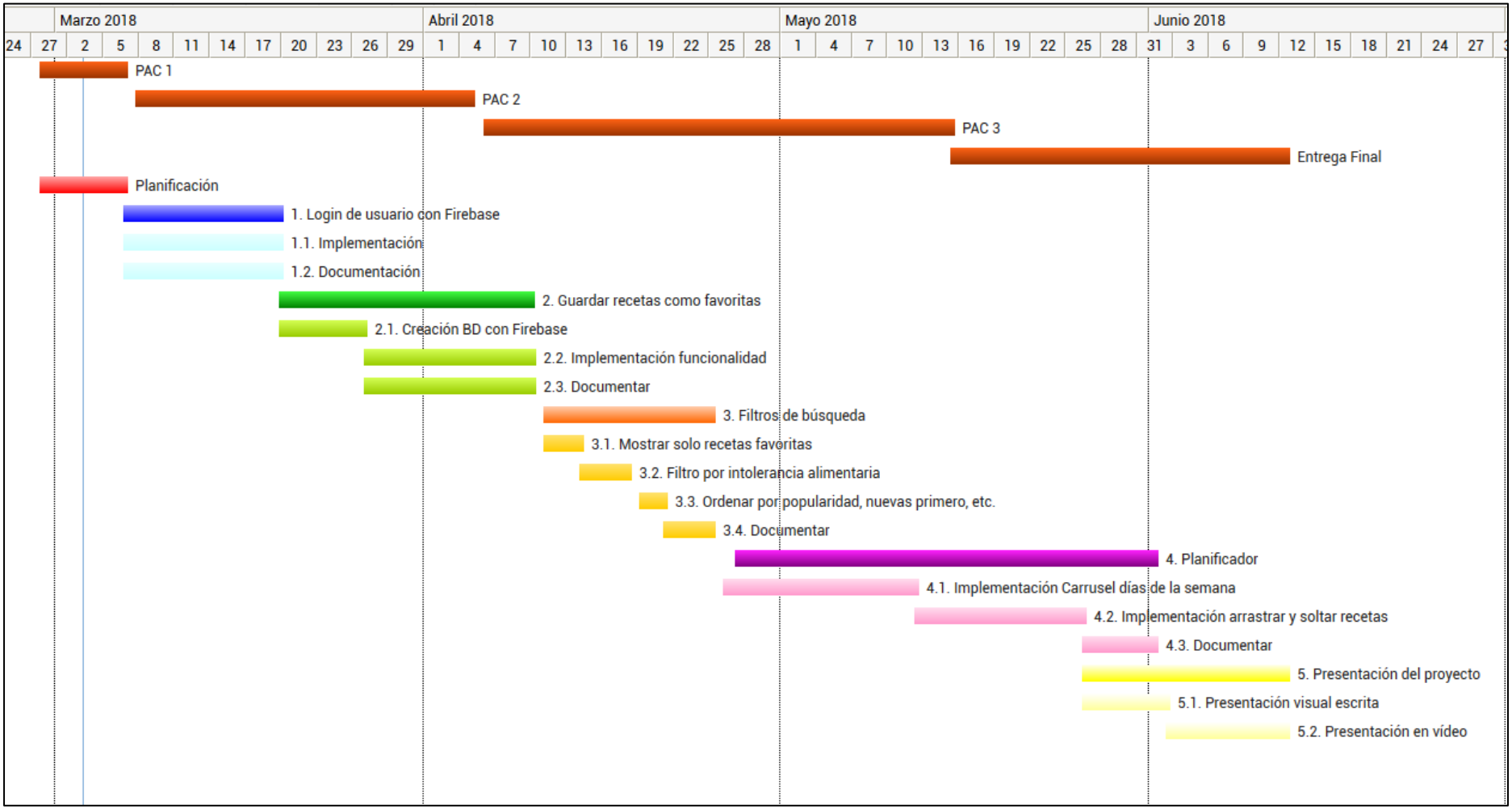
Página en producción: https://menureverde.firebaseapp.com/#/

9. Planificación

Teniendo en cuenta la metodología planteada en apartados anteriores, he creado las distintas tareas y subtareas y las he ordenado en el tiempo, quedando así:

Tarea	Días	Inicio	Fin
PAC 1	**6**	**27/02/2018**	**06/03/2018**
PAC 2	**21**	**07/03/2018**	**04/04/2018**
PAC 3	**28**	**05/04/2018**	**14/05/2018**
Entrega final	**21**	**14/05/2018**	**16/05/2018**
Planificación	**6**	**27/02/2018**	**06/03/2018**
1. Login de usuarios	**10**	**06/03/2018**	**19/03/2018**
1.1 Implementar funcionalidad	10	06/03/2018	19/03/2018
1.2 Documentar	10	06/03/2018	19/03/2018
2. Guardar recetas como favoritas	**16**	**19/03/2018**	**09/04/2018**
2.1 Creación BD con Firebase	6	19/03/2018	26/03/2018
2.2 Implementación funcionalidad	11	26/03/2018	09/04/2018
2.3 Documentar	11	26/03/2018	09/04/2018
3. Filtros de búsqueda	**11**	**10/04/2018**	**24/04/2018**
3.1 Mostrar solo recetas favoritas	4	10/04/2018	13/04/2018
3.2 Filtro intolerancia alimentaria	3	13/04/2018	17/04/2018
3.3 Ordenar por popularidad, nuevas primero, etc	3	18/04/2018	20/04/2018
3.4 Documentar	3	20/04/2018	24/04/2018
4. Planificador	**26**	**26/04/2018**	**31/05/2018**
4.1 Implementar carrusel días de la semana	13	26/04/2018	11/05/2018
4.2 Implementar “arrastrar y soltar” recetas	11	11/05/2018	25/05/2018
4.3 documentar	5	25/05/2018	31/05/2018
5. Presentación del proyecto	**12**	**05/05/2018**	**11/06/2018**
5.1 Presentación visual escrita	6	25/05/2018	01/06/2018
5.2 Presentación en vídeo	7	01/06/2018	11/06/2018

Diagrama de Gantt:



10. Proceso de trabajo

A continuación, se detallan los pasos que hay que realizar con cada apartado o componente del proyecto:

1. **Búsqueda de información**. En esta fase se hace un repaso de lo necesario, es decir, de los posibles componentes de terceros a reutilizar, mejores prácticas para elaborar el componente, si es necesario crear una base de datos, consultas a APIS externas, etc.

2. **Desarrollo**. Ejecución del componente.

3. **Actualización repositorio con Git**. Una vez he realizado cambios, si se está satisfecho con los resultados, se procede a la actualización del repositorio para poder comprobar los resultados en la versión de desarrollo:

   ```
   git add <archivo> o git add .
   git commit -m “mensaje de commit”
   git push origin master
   ```

   Estos comandos habrán actualizado el repositorio ubicado en:

   https://maestrilloliendre@bitbucket.org/maestrilloliendre/tfg_uoc

4. **Actualización versión producción**. Los cambios en la versión de desarrollo han sido probados y se está conforme con los resultados. Entonces se procede a actualizar la versión en producción:

   ```
   Firebase deploy
   ```

11. Apis utilizadas

En la elaboración de “menureverde” se han utilizado las siguientes API’s y Frameworks:

- **Firebase Realtime Database**. Como se ha mencionado en apartados anteriores esta va a ser la base de datos del proyecto. Para empezar a utilizarla tenemos que poner el siguiente fragmento de código:

```
// Set the configuration for your app
// TODO: Replace with your project's config object
var config = {
  apiKey: "apiKey",
  authDomain: "projectId.firebaseapp.com",
  databaseURL: "https://databaseName.firebaseio.com",
  storageBucket: "bucket.appspot.com"
};
firebase.initializeApp(config);

// Get a reference to the database service
var database = firebase.database();
```

- **Vuejs.** Un Framework, también mencionado anteriormente que proporciona reactividad y AJAX. Podemos invocar el *framework* a través de su CDN:

  https://cdn.jsdelivr.net/npm/vue

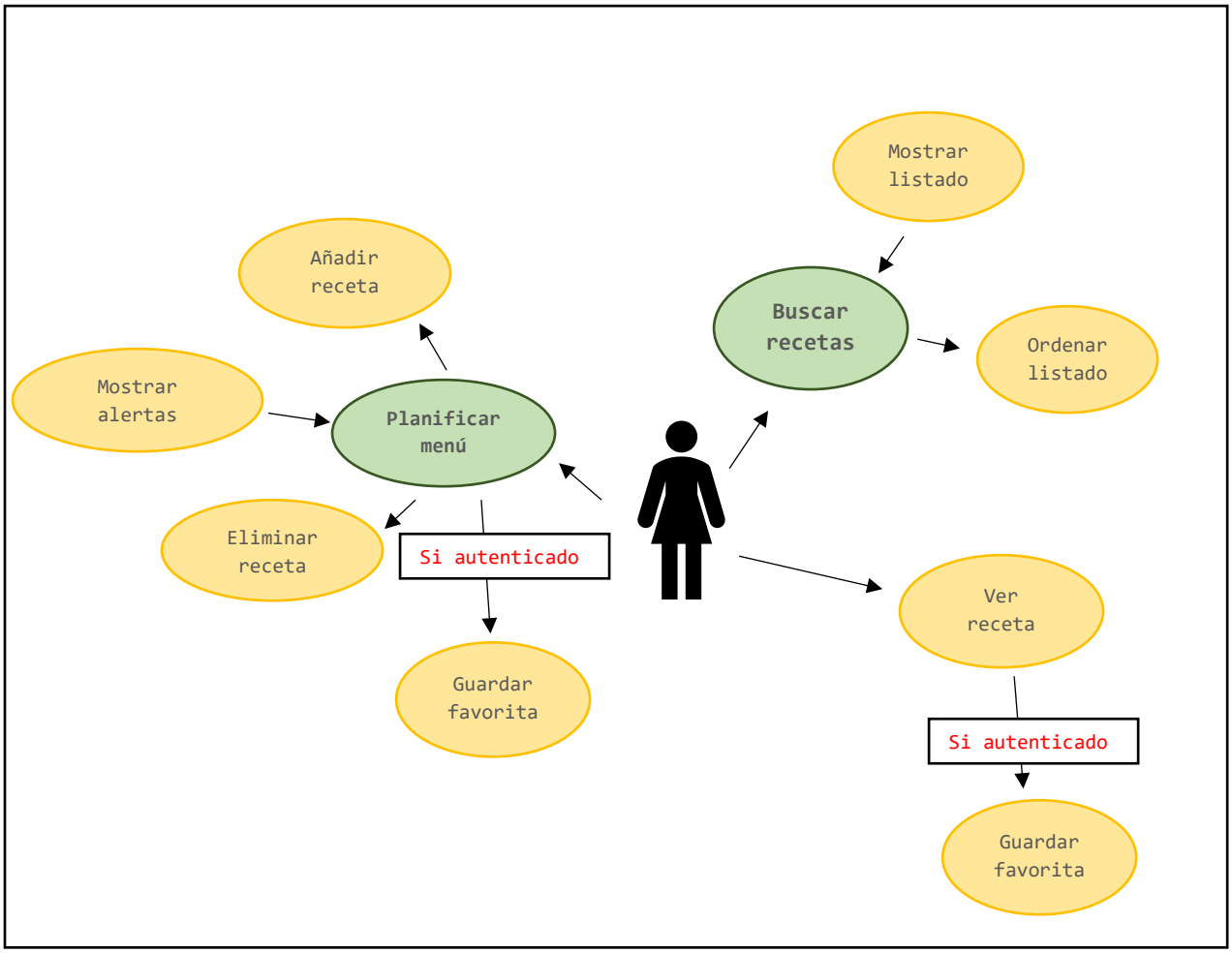
- **Base de datos de recetas Edamam**. Se trata de una API REST desde donde obtener información sobre los nutrientes de los distintos alimentos.

  Ejemplo de *request* tipo *Get* probado en Postman:

```
https://test-es.edamam.com/search?q=paella
valenciana&app_id=4485828c&app_key=73e1c2cd806e015b10451122ef261e2
```

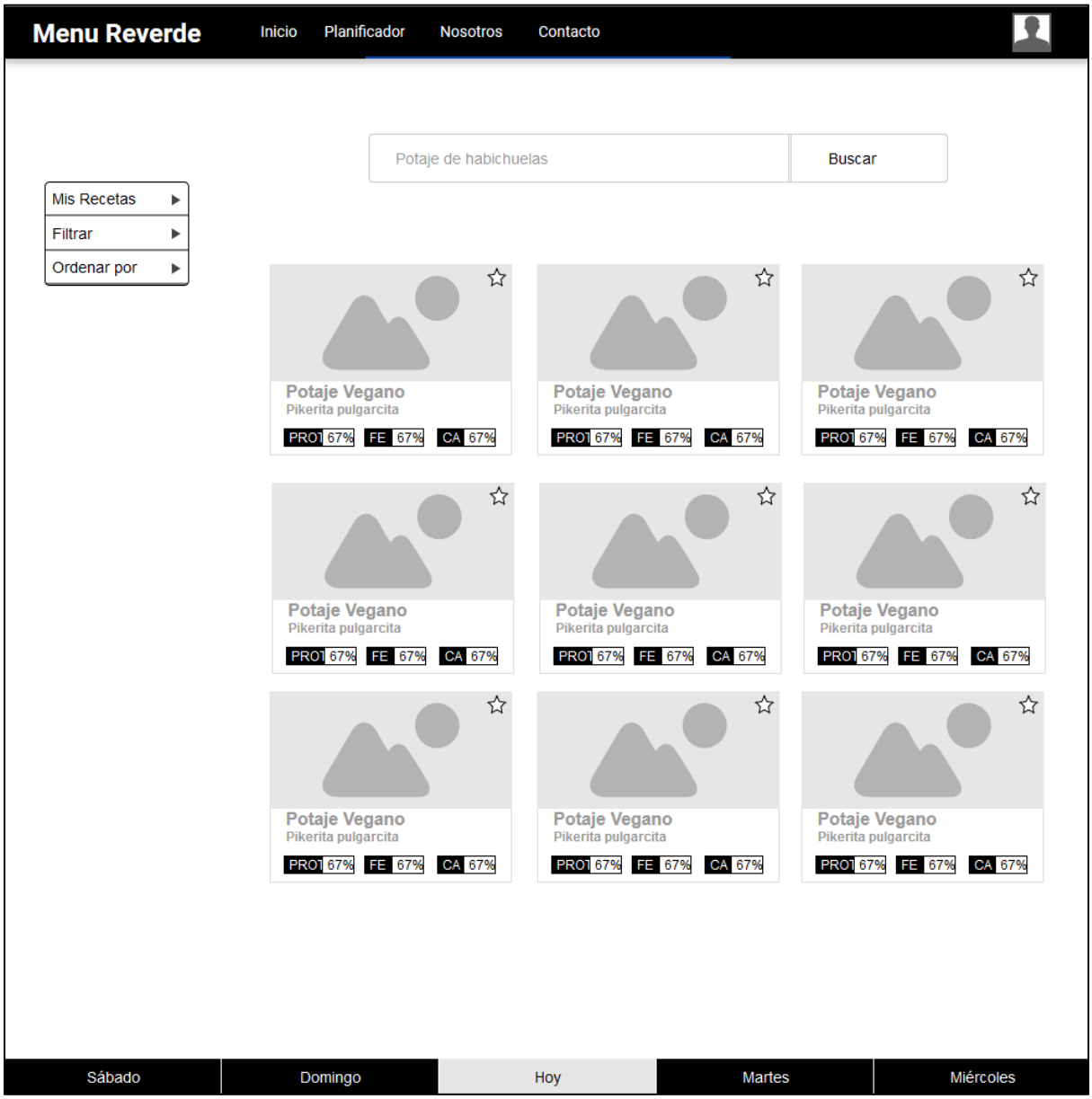
12. Diagrama de casos de uso

Al no utilizarse el paradigma de la programación orientada a objetos en el proyecto no son tan útiles los diagramas de clases, sin embargo, el diagrama de casos de uso puede ser muy útil para tener una perspectiva visual de todos los escenarios.



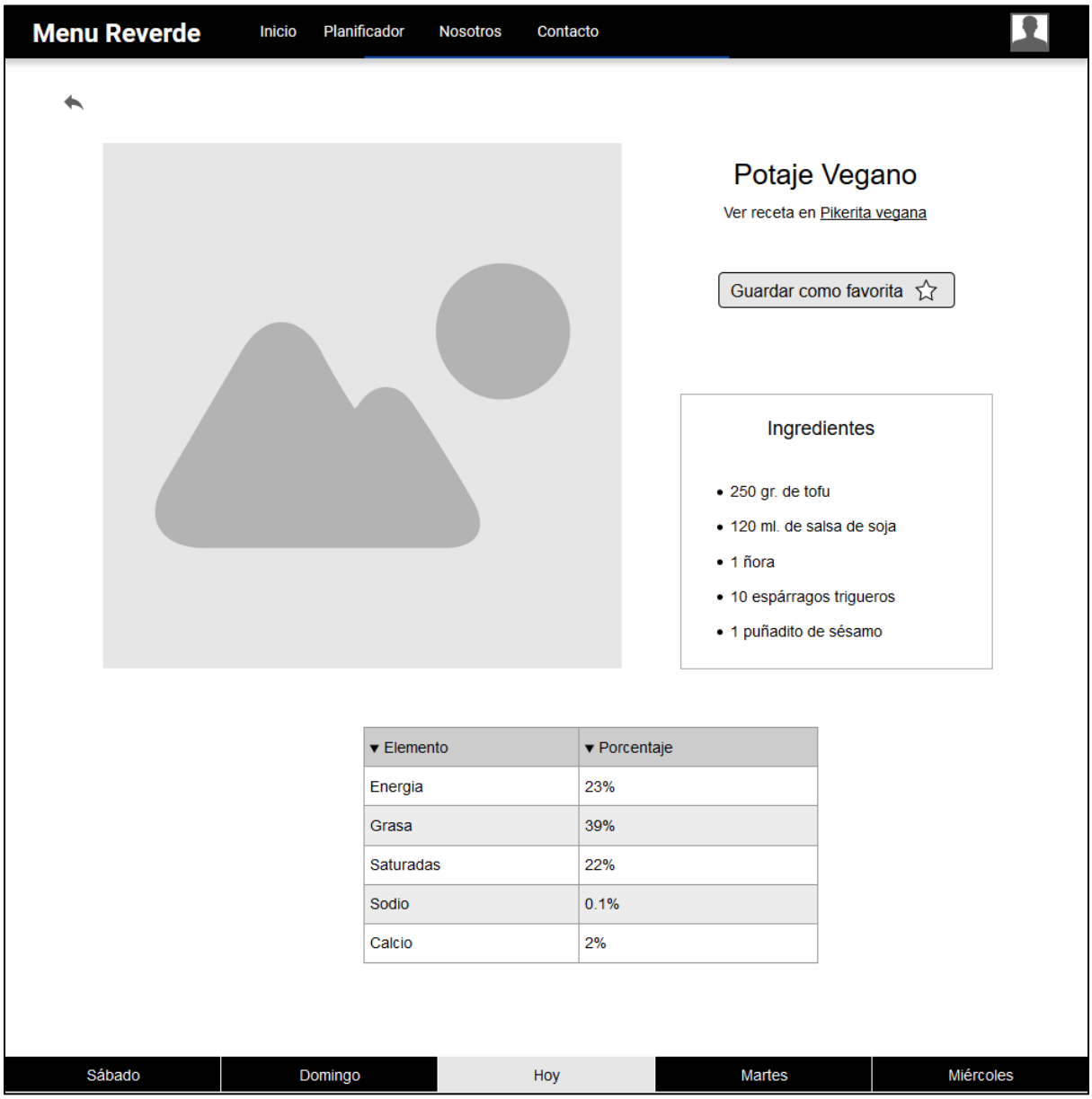
13. Prototipos

Página principal y resultados de búsqueda:

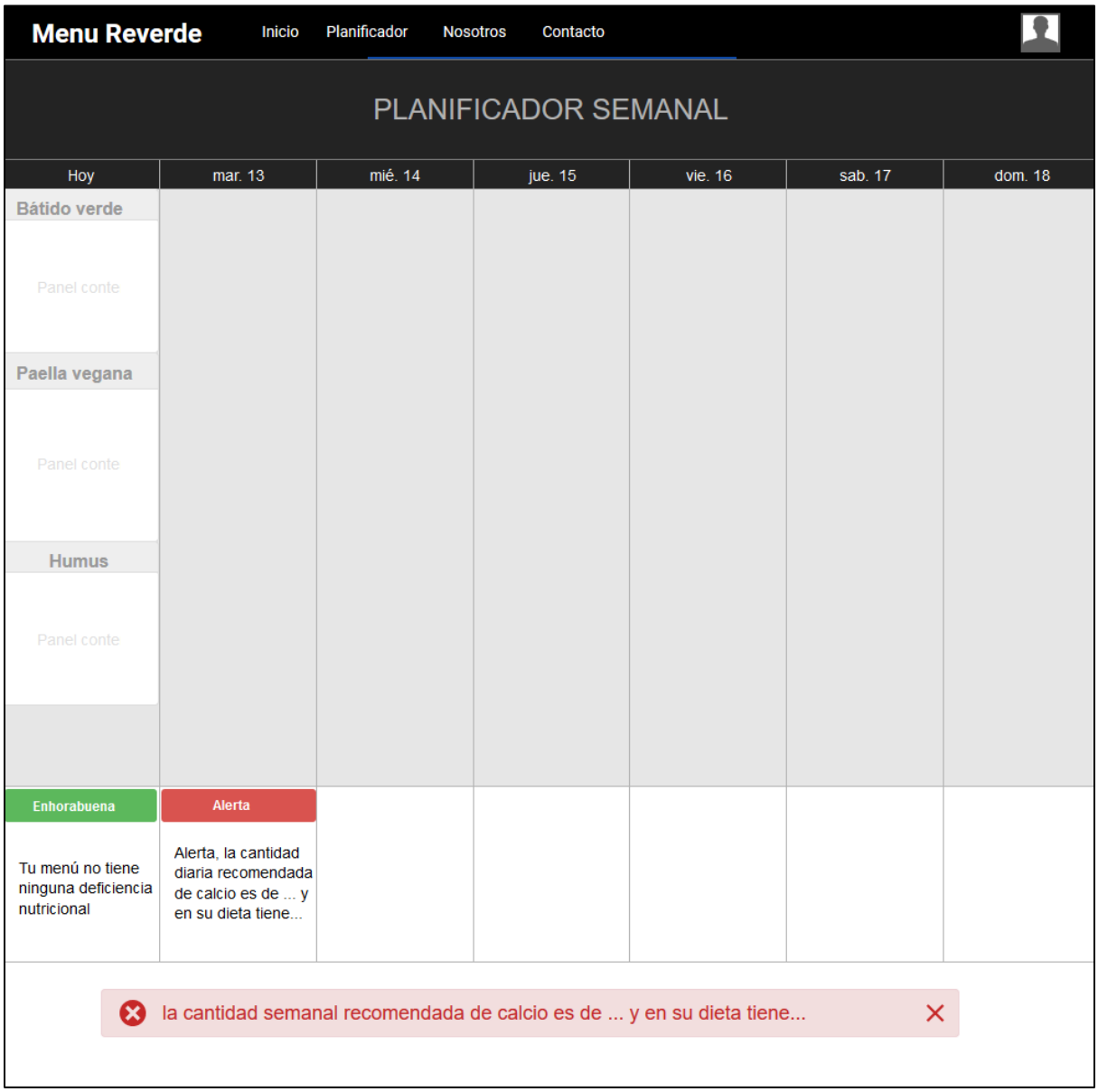


En esta página se puede observar un listado de recetas que bien puede ser el resultado de una búsqueda. A partir de aquí el usuario podrá pinchar en una receta para ver los detalles de la misma, o bien, podrá arrastrar las recetas al día de la semana que quiera para añadirla al menú semanal.

Página vista de receta con la información nutricional:



Planificador semanal:



Aquí se puede apreciar el menú semanal desplegado, viéndose en la parte inferior las posibles alertas alimentarias que se generarán automáticamente. Cabe destacar que se podrá acceder a cualquier día del año, pinchando y arrastrando sobre los días en el calendario.

14. Perfiles de usuario

Menú Reverde está destinado a esa minoría creciente que se preocupa por su salud, al mismo tiempo que se preocupa también por el medio ambiente y los animales.

Algunos de los posibles perfiles tipo a los que iría dirigida la aplicación serían los siguientes:

- ***Veggies*** a los que les gusta cocinar y se preocupan por su salud.

- **Padres** que al tener un niño han investigado sobre la alimentación y tienen dudas si su hijo estará recibiendo los nutrientes necesarios.

- **Personas** que han sido diagnosticadas **con alguna enfermedad** crónica o grave y que se han investigado acerca de la función que tiene la alimentación en la salud.

Además de los diferentes usuarios, la aplicación va a contar con uno o varios usuarios administradores, encargados de crear y actualizar los contenidos.

15. Usabilidad / UX

La usabilidad se mide en función de lo fácil e intuitiva que la aplicación sea de manejar. Otro factor íntimamente relacionado, es la experiencia de usuario, la cual va un paso más allá, tratando la satisfacción del usuario. Esto normalmente ocurre cuando la aplicación está generando algún valor positivo al usuario.

En cuanto a la **usabilidad** se han tenido en cuenta los siguientes factores:

- **Tiempos de carga rápidos.** Vuejs utiliza un renderizado en el navegador, provocando unos tiempos de carga casi instantáneos, dando la sensación en muchas ocasiones que se está utilizando una aplicación de escritorio.

- **Diseño claro y limpio.** En este sentido se utiliza el *framework* CSS Bulma. Antes del lanzamiento de *frameworks* como este se realizan estudios de usabilidad, por lo que, al ser incorporados en el proyecto, se están aprovechando estas características. Por el contrario, se pierde en creatividad y en el poder de sorprender al usuario. Por otro lado, el hecho de utilizar un *framework* hace que el usuario se sienta familiar con el diseño ya que es muy probable que lo haya utilizado con anterioridad.

- **Coherencia.** En este apartado entrarían las respuestas que la aplicación da dependiendo de las acciones del usuario. En este sentido, se intenta informar al usuario de la acción que se está produciendo en cada momento. Por ejemplo, cuando se realiza una búsqueda, al pulsar el botón de buscar aparece el mensaje “Cargando”. Por el contrario, si no se ha producido ninguna coincidencia aparece una notificación en color rojo.

Además, también se han tenido en cuenta aspectos para mejorar la experiencia de usuario:

- **Usabilidad.** El hecho de haber utilizado algunos criterios de usabilidad comentados anteriormente facilita que el usuario pueda cumplir los objetivos que pretendía al acceder a la aplicación.

- **Utilidad.** Se realizan revisiones para que todos los elementos de la aplicación tengan un propósito concreto.

- **Contenido de calidad.** Se espera que la información proporcionada, así como su presentación, sea de un gran valor para el usuario.

- **Facilidad de búsqueda.** Para una buena experiencia de usuario, este tiene que encontrar de manera muy sencilla lo que esté buscando. En este caso, se ha revisado que la manera de acceder a cada apartado sea directa y sencilla. También se ha evitado introducir elementos que puedan despistar al usuario. Por otro lado, el acceso a los dos principales apartados de la aplicación es muy rápido e intuitivo:

    - El buscador de recetas aparece en cuanto cargamos la página y pulsando sobre el botón de “Inicio”.

16. Seguridad

Las páginas web al estar accesibles 24 x 365 a través de Internet están expuestas a multitud de ataques de diferente índole. Desde ataque de denegación de servicio, presentar información modificada o dañada, robar datos de carácter personal, etc…

Es por ello por lo que la seguridad de los sitios web se tiene que aplicar a todos sus elementos, en la aplicación web, en la configuración del servidor, en las políticas de contraseñas, etc.

En el caso de este proyecto, al utilizar servicios de terceros como Firebase o VueJS, se da por seguro que habilitarán por defecto fuertes mecanismos de defensa contra los ataques más conocidos.

17. Versiones de la aplicación

En el transcurso de la elaboración del proyecto se pueden establecer cuatro versiones diferentes de la aplicación que coincidirían con las cuatro entregas.

Versión 1. En esta versión se implementó la búsqueda de recetas.

Versión 2. Implementación de la autenticación.

Versión 3. Se añade la funcionalidad de guardar recetas favoritas.

Versión 4. Se añade la funcionalidad de poder ver las recetas guardadas como favoritas.

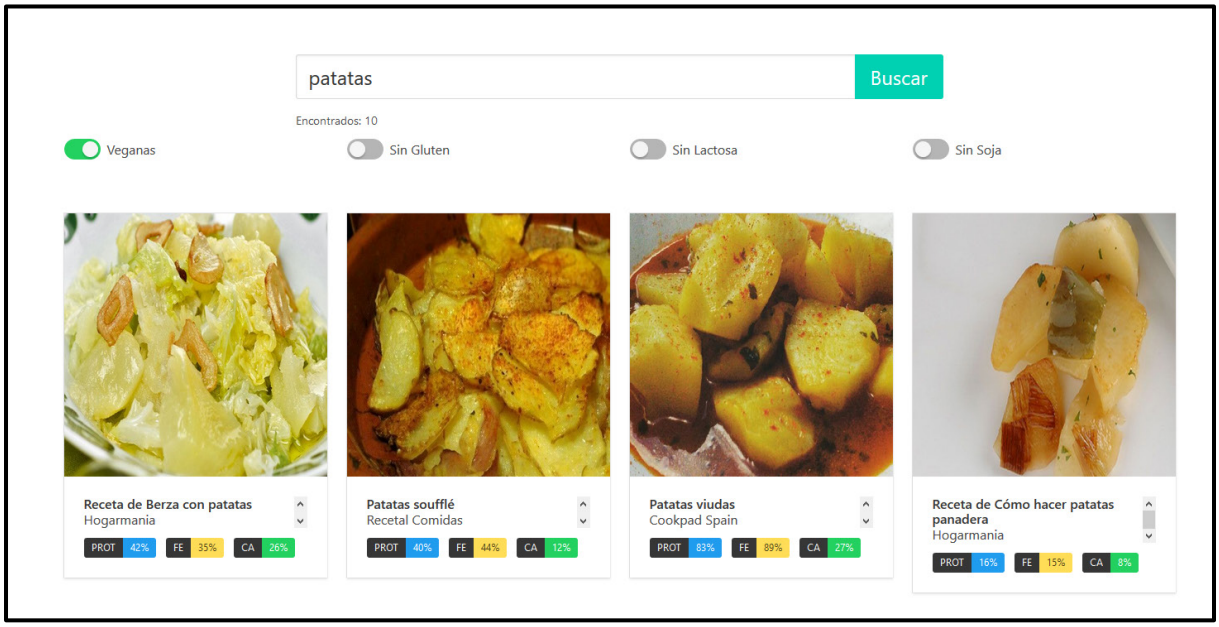
No obstante, al haber trabajado con un repositorio Git en Bitbucket se pueden comprobar todos los *commits* del mismo, los cuales se podrían considerar también distintas versiones de la aplicación:

Author	Commit	Message	Date	Builds
Jose Luis	c46b602	Añadida funcionalidad Mis Recetas, correccion de bugs	2018-06-05	
Jose Luis	c15c361	Añadida funcionalidad Mis Recetas	2018-06-05	
Jose Luis	9f7e80e	versión beta entrega PAC2	2018-05-13	
Jose Luis	a5f05d4	Versión Beta 1 PAC 2	2018-05-13	
Jose Luis	23f9723	Guardar recetas favoritas funcionando	2018-05-13	
Jose Luis	500a212	Filtros alergias funcionando v1	2018-05-04	
Jose Luis	f24ef23	Eliminando algunos bugs	2018-04-30	
Jose Luis	ea267af	Autenticacion funcionando !	2018-04-09	
Jose Luis	c054732	Atenticación funcionando a medias	2018-03-24	
Jose Luis	c7c5d36	Atenticación funcionando a medias	2018-03-24	
Jose Luis	0a414be	Implementacion beta, muy beta del frontend para login de usuaio	2018-03-05	
Jose Luis	b01e5fa	Utilización de vue-carousel para el planificador	2018-02-03	
Jose Luis	131638e	Añado vue-carousel al proyecto. Planificador	2017-12-11	
Jose Luis	554f8a8	Añadido vue-router como dependiencia de desarrollo	2017-12-08	
Jose Luis	b642d49	npm run build, firebase init, firebase deploy	2017-12-03	
Jose Luis	51c84a3	Se añaden transiciones para las notificaciones	2017-12-02	
Jose Luis	f92990e	Creación de Detalle de receta y navegación a inicio	2017-11-30	
Jose Luis	203f481	Modificación búsqueda para que muestre solo recetas vegetarianas	2017-11-30	
Jose Luis	2dbf0fa	Modificaciones en Header, menu hamburguesa	2017-11-30	
Jose Luis	1a6ee85	Navegación a detalle de receta con rutas paramétricas	2017-11-29	
Jose Luis	068acbb	Implementación Vue Router para crear una SPA	2017-11-27	
Jose Luis	3222f3e	Creación de componentes Footer, Header, Loader, Recipe	2017-11-15	
Jose Luis	d22cb53	Consulta a la API de Edamam y moostrando el nombre de las recetas encontradas	2017-11-15	
Jose Luis	d550b22	conexion con la api mediante https://github.com/Huemul/trae	2017-11-15	
Jose Luis	58a2607	Dependencias instaladas y configuradas	2017-11-14	
Jose Luis	31c543a	vue init	2017-11-13	

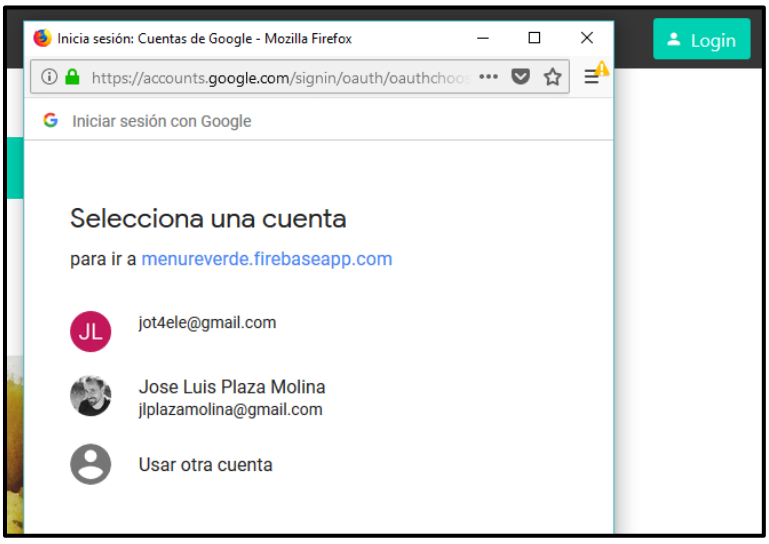
18. Instrucciones de uso

Para que la aplicación funcione hay que habilitar CORS (en español Intercambio de Recursos de Origen Cruzado), esto es debido a que la aplicación hace consultas HTTP a un dominio diferente, en este caso la API de información de recetas e información nutricional de Edamam. Para habilitarlo es necesario instalar una extensión en el navegador que permita activar CORS y deshabilitarlo, cuando no se esté utilizando la aplicación.

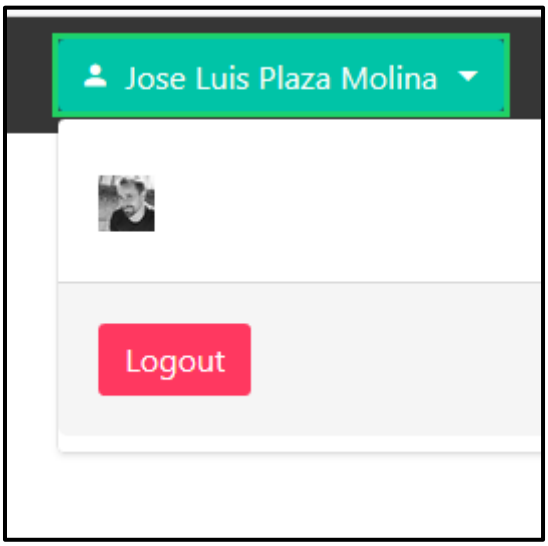
Una vez habilitado CORS ya se pueden realizar búsquedas de recetas y filtrarlas según los gustos o intolerancias del usuario:



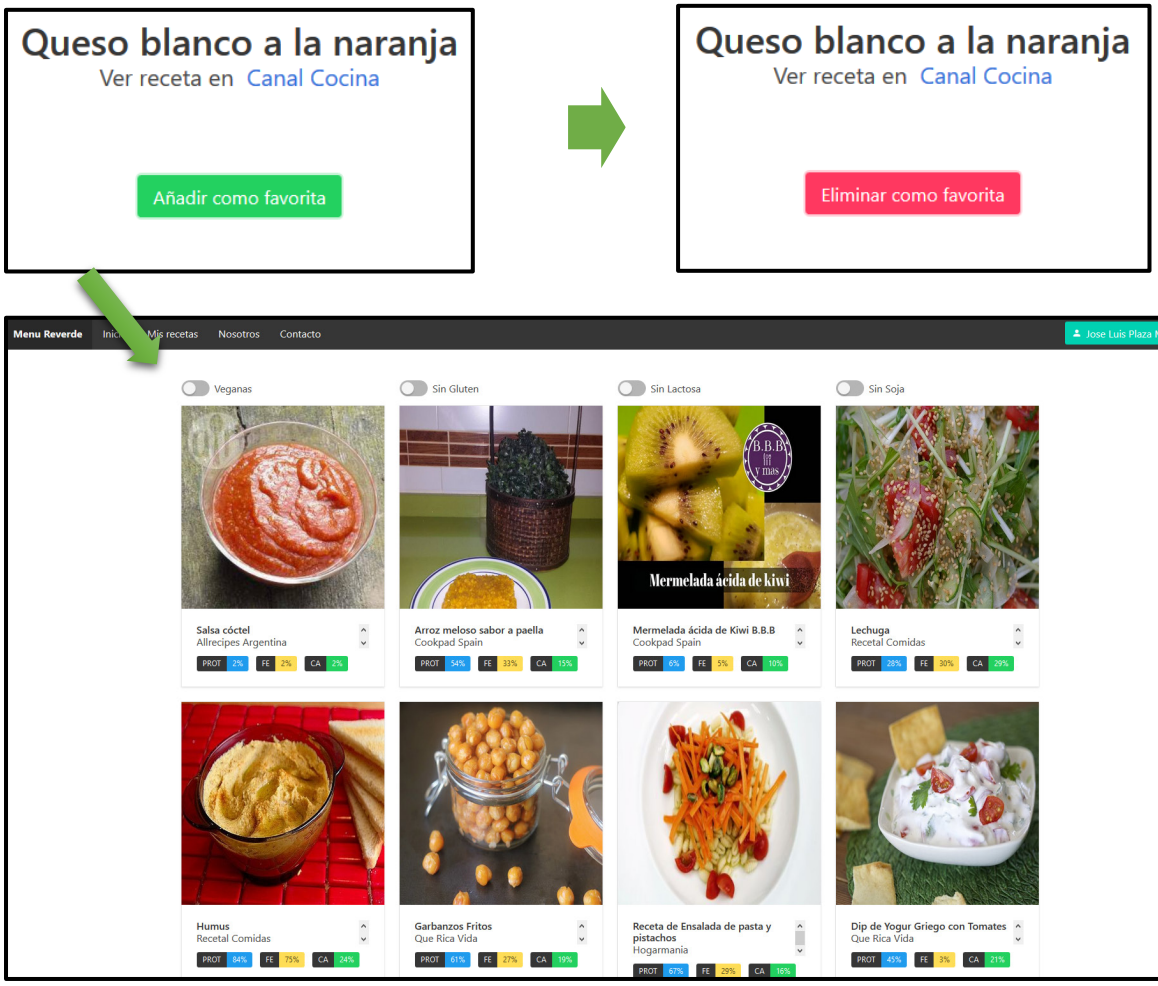
Sin estar logado en la aplicación se puede también seleccionar una receta y ver su información nutricional, pero no se podrá guardar la receta como favorita. Para esto último hay que identificarse en el sistema, lo que es muy sencillo, solo hay que pulsar el botón de *login* y acceder con una cuenta de Google:



Una vez logado el usuario puede deslogarse pulsando el botón “*Logout*”:



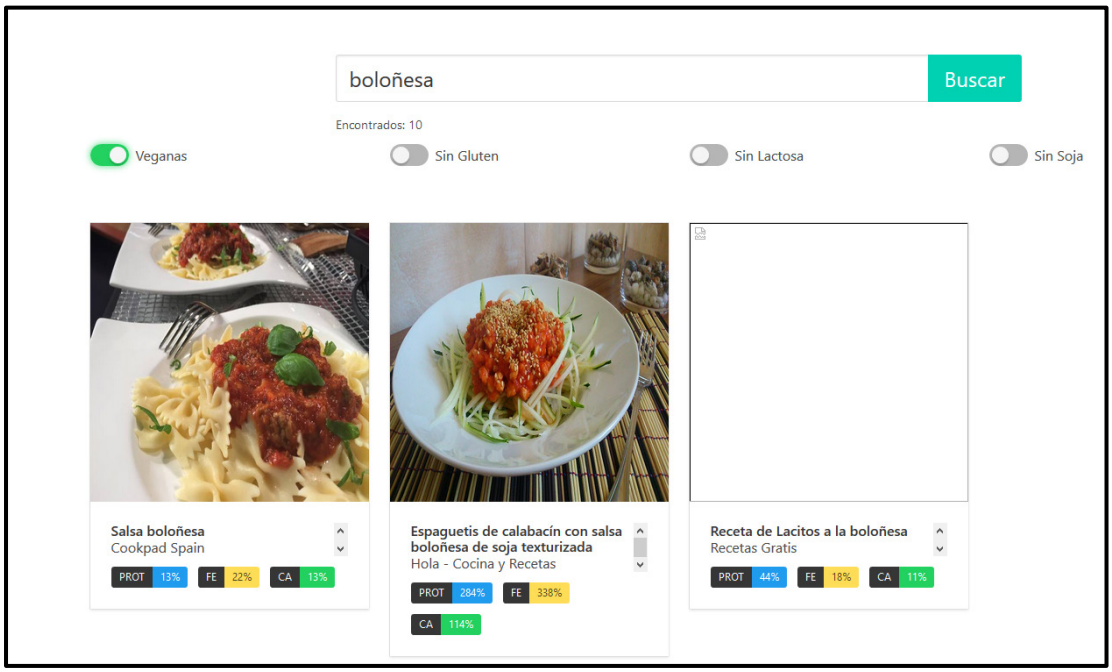
Un usuario logado puede marcar las recetas como favoritas y acceder a sus recetas a través del botón “Mis recetas” del menú de navegación:



19. Bugs

- **Búsquedas limitadas a 10 resultados.** Por alguna limitación de la API, ya que se está usando una cuenta gratuita, cuando se realiza una búsqueda el número máximo de resultados es, únicamente, de 10.

- **Número de recetas encontradas no cambia con filtrado.** Cuando se ha realizado una búsqueda aparece la información del número de recetas encontradas, este número no cambia al realizar una operación de filtrado.



Como se puede comprobar en la imagen, el número de recetas encontradas es de 10, sin embargo, como se observa, tras aplicar el filtro han quedado solo tres recetas.

- **No aparecen todas las recetas.** Al no utilizar una versión de pago de la API de recetas, es posible que en algún momento se exceda el límite de consultas y no aparezcan todas las recetas guardadas como favoritas. En estos casos, si se abren las herramientas de desarrollo del navegador, en la consola se verá el siguiente error:

```
Error: Unauthorized
```

20. Proyección de futuro

A partir de este proyecto son varias las ideas o mejoras que se podrían implementar, tanto como para aumentar sus funcionalidades, como para generar ingresos:

1. **Planificador**. Esta funcionalidad estaba planificada para terminarla a tiempo, pero no ha sido posible, además a nivel técnico sería bastante difícil de implementar. Esta funcionalidad permitiría a cualquier usuario poder organizar un menú semanal simplemente arrastrando y soltando recetas sobre el calendario. Para poder entender mejor su funcionamiento n el apartado de prototipos hay creado uno del planificador.

2. **Lista de la compra.** La cual se debería crear automáticamente con los ingredientes de las recetas incluidas en el menú semanal.

3. **Suscripción premium.** Esta suscripción daría acceso a recetas de blogueros prestigiosos

4. **Creación de recetas.** Mediante esta opción se les permitiría a los usuarios poder subir sus propias recetas a la plataforma.

5. **Perfil nutricionista y menús personalizados.** Mediante una cuenta de usuario con un rol de nutricionista se podrían crear menús personalizados que el resto de usuarios podría comprar, quedándose la plataforma con una parte de ese dinero.

21. Presupuesto

Puesto que el TFG es realizado por una sola persona, para este apartado se ha decidido realizar un supuesto ficticio donde un equipo pequeño se haría cargo del proyecto.

En el caso del hosting, que para el proyecto se ha realizado con la capa gratuita de Firebase, se ha seleccionado el plan Flame de Firebase, el cual incluye 10 GB de almacenamiento y 50 GB/mes de transferencia.

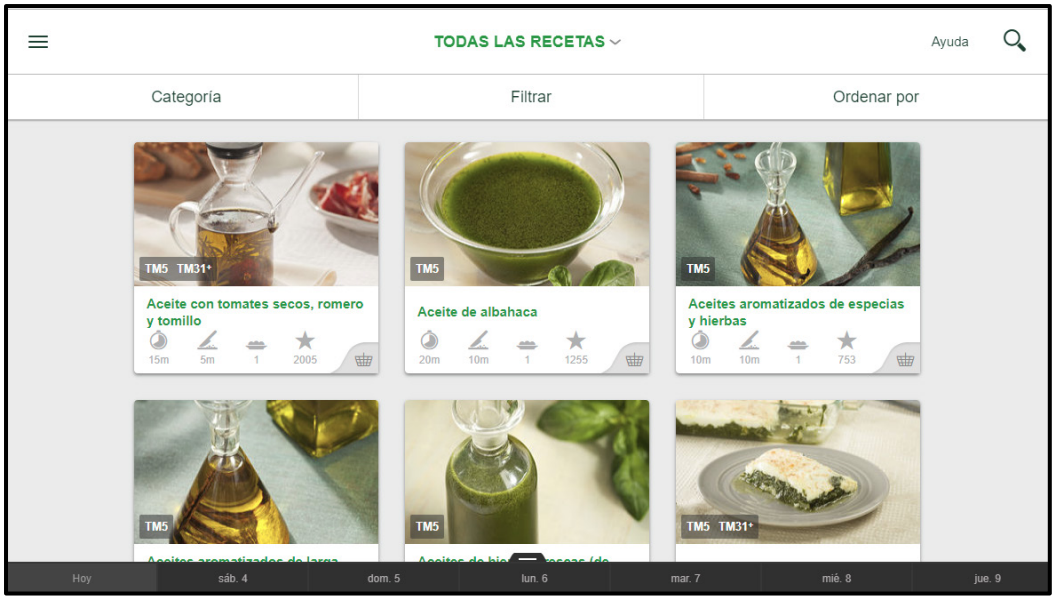
Recurso	Coste por hora	Horas empleadas	Total
Director proyecto	50 €	40 h	2.000 €
Arquitecto de la información	35 €	15 h	525 €
Diseñador UX	30 €	15 h	450 €
Diseñador gráfico	25 €	40 h	1.000
Programador	30 €	50 h	1.500
Dominio (1año)			60 €
Hosting Firebase (1 año)			240 €
Total			**5.775 €**

.

22. Análisis de mercado

Existen varias aplicaciones en el mercado que, además de funcionar como páginas de recetas, se puede elaborar un menú con ellas e incluso elaborar una lista de la compra. Voy a analizar dos de las principales aplicaciones:

- **Cookido.** https://cookidoo.es



Esta aplicación recopila colecciones de recetas para Thermomix. Entre sus principales ventajas encontramos su gran experiencia de usuario, haciendo muy fácil, por ejemplo, añadir recetas en cualquier día de la semana simplemente arrastrando y soltando. También es muy intuitiva la banda dinámica con el calendario y si pinchamos sobre ella nos abre el menú semanal.

Su principal inconveniente es que las recetas son de pago y que está centrada únicamente en el universo Thermomix. Otro inconveniente que le veo es que no tiene información nutricional.

- **Nestle Menu Planner**. https://www.nestlemenuplanner.es

Esta aplicación de muy buen diseñó también, tiene su punto fuerte en la automatización y es que por defecto te ofrece un menú equilibrado, pudiendo cambiar la receta que queramos si no nos gusta. Aparece información de calorías, así como las raciones recomendadas de carne y pescado con alertas de tipo *success* si estás haciendo las cosas bien y de tipo *alarm* si no estás comiendo lo que ellos consideran sano.

Sin embargo, también tiene sus puntos débiles y es que, como es lógico Nestlé nos quiere vender sus productos y la mayoría de las recetas utilizan productos Nestlé. Además, la información nutricional no la considero fiable, al provenir de una multinacional de la industria alimentaria.

Estas dos páginas son, para mí, la mejor referencia al realizar mi aplicación. Los principales factores diferenciales serían que, por un lado, Menureverde está orientada hacia un *target* específico, y por otro lado, que no nos mueve ningún interés más que el de la salud con base científica, la sostenibilidad y la empatía con todos los seres vivos.

23. Marketing y Ventas

Menureverde es un proyecto que no se plantea la monetización como primer objetivo, ya que este es servir de ayuda a la comunidad. Sin embargo, como se ha comentado en apartados anteriores, hay varias funcionalidades extra que se podrían implementar en la aplicación bajo servicios de suscripción.

Bajo estas premisas, un primer objetivo sería convertirse en una aplicación de éxito con un buen volumen de usuarios contentos con los servicios que ofrece. Para dar difusión al proyecto se utilizarán únicamente las redes sociales.

A partir de aquí se estudiaría añadir funcionalidades premium bajo una cuota de suscripción.

24. Conclusiones

Afrontar este proyecto como trabajo fin de grado ha sido todo un reto para mí, ya que no solo he utilizado los conocimientos aprendidos durante el mismo, si no que he querido ampliarlos con nuevas tecnologías como es el caso de VueJS y Firebase.

A lo largo de estos meses ha habido momentos realmente duros que me han hecho replantearme el continuar con el proyecto. Debido a esto, ahora estoy doblemente satisfecho, ya que he abierto la puerta a un mundo realmente impresionante que me gustaría seguir explorando en los próximos años.

Las fases que más me han gustado del proyecto es llevar a la práctica lo que iba aprendiendo y ver cómo, tras varios días de peleas con el código las cosas acaban saliendo. Sólo es cuestión de dedicarle horas y horas.

Por otro lado, se podría decir que lo que menos me ha gustado son las partes formales, como la redacción de la memoria, ya que se tiene un formato muy rígido y estricto, al que me ha costado adaptarme.

A pesar del optimismo, he de decir que no se han cumplido todos los objetivos planteados en la planificación, ya que el planificador de menús ha sido imposible desarrollarlo. Tras varios días de estudio, decidí abandonar este objetivo porque era demasiado difícil técnicamente.

Anexo 1. Entregables del proyecto

- Memoria del Trabajo Final de Grado
- Vídeo de presentación y defensa del TFG
- Presentación del TFG
- Archivo compilado del proyecto

Anexo 2. Código Fuente

En el apartado de arquitectura se ha detallado la parte de código específico de la infraestructura montada para hacer funcionar la aplicación con sus dependencias. En las siguientes líneas se va a explicar con detalle el código de las funcionalidades más importantes, como es la búsqueda, el filtrado, guardar recetas como favoritas, etc.

Anexo 2.1 Búsqueda y resultados de la búsqueda

La mayor parte de la lógica de esta funcionalidad está en el componente "`Search.vue`":

```
main
    transition(name="move")
      pm-notification(v-show="showNotification")
        p(slot="body") No se encontraron resultados

    transition(name="move")
      pm-loader(v-show="isLoading")

    section.section(v-show="!isLoading")
      .container
        .column.is-three-fifths.is-offset-one-fifth
          .field.has-addons
            .control.is-expanded
              input.input.is-large(
                type="text",
                placeholder="Buscar recetas",
                v-model="searchQuery",
                @keyup.enter="search"
              )
              // v-on es la directiva de vue para escuchar eventos. Se puede
sustituir por '@'
            .control
              a.button.is-primary.is-large(@click="search") Buscar
              // a.button.is-danger.is-large &times;
          p
            small {{ searchMessage }}
```

Esta parte del código muestra el HTML simplificado gracias a Pug. Aquí podemos ver los distintos elementos que se renderizan según las distintas condiciones. El funcionamiento es el siguiente:

Una vez que realizamos una búsqueda lanzamos el método "`search`, el cual pone la variable `isLoading` a true mientras se lanza la consulta a la API, Vue se encarga de renderizar automáticamente el componente "`Loader.vue`" cuando se cumple esta condición. Una vez recibida respuesta por la API, si se no se han encontrado resultados aparecerá un mensaje

de error, por el contrario, si la búsqueda ha obtenido resultados se renderizarán los mismos, según el código que se muestra a continuación:

```
.container.results
        .columns.is-multiline
          .column.is-one-quarter(v-for="r in filterRecipes")
            // Cada una de las instancias de r van a crear
            // una instancia nueva del componente recipe
            pm-recipe(
              :class="{ 'is-active': r.recipe.uri === selectedRecipe }",
              :recipe="r",
              @select="setSelectedRecipe")
              // Con @select hemos capturado el evento select emitido por un
componente hijo
```

El método `Search` tiene el siguiente código:

```
search () {
      if (!this.searchQuery) { return } // Si no hay resultados no devuelve
error

      this.isLoading = true

      recipeService.search(this.searchQuery)
        .then(res => {
          // mediante count obtengo el número de registros encontrados en la
búsqueda.
          this.showNotification = res.count === 0 // el resultado sera true si
no se encuentran resultados
          // mediante hits obtengo el array con todas las recetas encontradas
que guardo en recipes[]
          this.recipes = res.hits
          this.isLoading = false
        })
    },
```

Anexo 2.2 Filtrado de recetas

Si al realizar una búsqueda se obtienen resultados, entonces, automáticamente, se renderizan los distintos filtros programados. Siendo el código de la vista el siguiente:

```
.container(v-if="this.recipes.length > 0")
        section
          .columns
            .column
              b-switch(
                type="is-success",
                v-model="veganSelected"
              ) Veganas
            .column
              b-switch(
                type="is-success",
                v-model="glutenfreeSelected"
              ) Sin Gluten
            .column
              b-switch(
                type="is-success",
                v-model="dairyfreeSelected"
              ) Sin Lactosa
            .column
              b-switch(
                type="is-success",
                v-model="soyfreeSelected"
              ) Sin Soja
```

Los cuatro interruptores están conectados al modelo, en concreto a una variable booleana, "`veganSelected`", "`glutenfreeSelected`", "`dairyfreeSelected`" y "`soyfreeSelected`".

La lógica del filtrado la realiza la propiedad computada "`filterRecipes`". Se puede comprobar en la vista de este componente de qué forma se renderizan todos los resultados de la búsqueda:

```
.column.is-one-quarter(v-for="r in filterRecipes")
```

De esta forma tan sencilla nos permite VueJS acceder a todos los elementos de "`filterRecipes`". Aquí tenemos el código de esta propiedad computada:

```
  filterRecipes () {
    let glutenfreeSelected = this.glutenfreeSelected
    let veganSelected = this.veganSelected
    let dairyfreeSelected = this.dairyfreeSelected
    let soyfreeSelected = this.soyfreeSelected
    return this.recipes.filter(function (recipe) {
      let filters = []
      if (glutenfreeSelected) {
        filters.push(recipe.recipe.healthLabels.includes('Gluten-Free'))
      }
      if (dairyfreeSelected) {
        filters.push(recipe.recipe.healthLabels.includes('Dairy-Free'))
      }
      if (veganSelected) {
        filters.push(recipe.recipe.healthLabels.includes('Vegan'))
      }
      if (soyfreeSelected) {
        filters.push(recipe.recipe.healthLabels.includes('Soy-Free'))
      }
      var result = recipe
      if (filters.length > 0) {
        for (var i = 0; i < filters.length; i++) {
          if (i === 0) {
            result = filters[i]
          } else result = result && filters[i]
        }
      }
      return result
    })
  }
},
```

Anexo 2.3 Guardar recetas como favoritas

Para obtener esta funcionalidad se hace uso de la base de datos, la cual se ha explicado detalladamente en el apartado de arquitectura. Partiendo de esto, el componente "`recipeDetail.Vue`" es el que tiene implementada esta funcionalidad y más concretamente el botón "Añadir como favorita" que al pulsarlo guarda la receta en la base de datos cambiando de color a rojo y de texto a "Eliminar como favorita". A continuación, podemos ver el código de la vista:

```
section(v-if='favSelected').section.has-text-centered
  button.button.is-danger(@click="addFav()") Eliminar como favorita
section(v-else='favSelected').section.has-text-centered
  button.button.is-success(@click="addFav()") Añadir como favorita
```

Como se puede ver, es bastante intuitivo. La propiedad computada favSelected valdrá `true` si la receta ya ha sido introducida en la base de datos. El código es el siguiente:

```
computed: {
    // Propiedad computada que detecta dinamicamente si hay una receta
seleccionada
    favSelected () {
      var result
      let actualFav = { recipeUri: this.recipe.uri, userUid: this.user.uid }
      // Compruebo que si el array de recetas favoritas está vacio
      if (this.favs.length > 0) {
        // Se comprueba si la receta ya está añadida como favorita
        let filterRecipe = this.favs.filter(e => ((e.recipeUri ===
actualFav.recipeUri) && (e.userUid === actualFav.userUid)))
        if (filterRecipe.length > 0) {
          result = true
        } else {
          result = false
        }
        return result
      }
    }
  },
```

En ambos casos, al pulsar el botón se ejecuta el método `addFav()`. El cual se encarga de añadir o eliminar el registro de la base de datos. El código de este método es el siguiente:

```
addFav () {
      if (this.userLoged) {
        let actualFav = { recipeUri: this.recipe.uri, userUid: this.user.uid }
        // Compruebo que si el array de recetas favoritas está vacio
        if (this.favs.length > 0) {
          // Se comprueba si la receta ya está añadida como favorita
```

```
          let filterRecipe = this.favs.filter(e => ((e.recipeUri ===
actualFav.recipeUri) && (e.userUid === actualFav.userUid)))
          // Si la receta no está añadida como favorita, la guardamos en la
base de datos
          if (!filterRecipe.length > 0) {
            database.ref('favs').push(actualFav)
            this.favSelected = true
          } else {
            // Si la receta está añadida ya como favorita entonces la borramos
            database.ref('/favs/' + filterRecipe[0].key).remove()
            database.ref('/favs')
              .on('value', snapshot => this.readDBfavs(snapshot.val()))
            this.favSelected = false
          }
        } else {
          database.ref('favs').push(actualFav)
          this.favSelected = true
        }
      } else console.log('usuario no logado')
    }
  }
```

Anexo 2.4 Mis recetas

Para mostrar las recetas favoritas de un usuario se ha creado un componente nuevo llamado "`Misrecetas.Vue`" basado en el componente "`Search.Vue`", ya que toda la parte del renderizado de recetas y la aparición de los filtros es la misma que en este componente. Las diferencias es que la búsqueda de recetas se realiza al cargar el componente y consiste en pasarle a la API la URI de las recetas guardadas como favoritas en la base de datos.

Como se ha visto anteriormente, para ejecutar un método justo al crearse el componente hay que llamarlo dentro del estado "`created`" de la instancia de VueJS. En este caso, además, se ha introducido en la parte de comprobación de la autenticación, para que sólo se ejecute si el usuario está logado:

```
created () {
  database.ref('/favs')
    .on('value', snapshot => this.readDBfavs(snapshot.val()))

  setInterval(() => {
    if (!this.firebaseAppDefined) {
      if (firebase.app()) {
        firebase.auth().onAuthStateChanged((user) => {
          if (user) {
            this.userLoged = true
            this.user = user
            console.info('usuario conectado', this.user.displayName)
            this.search()
          } else {
            this.userLoged = false
            this.user = {}
            console.warn('usuario no conectado')
          }
        })
        this.firebaseAppDefined = true
      }
    }
  }, 100)
  console.info(this.firebaseAppDefined)
},
```

Y este es el código del método "`search`":

```
  search () {
    this.isLoading = true
    if (this.userLoged) {
      if (this.favs.length > 0) {
        for (let key in this.favs) {
```

```
            if (this.user.uid === this.favs[key].userUid) {
              recipeService.getByUri(this.favs[key].recipeUri)
                .then(res => {
                  this.recipes.push(res[0])
                })
            }
          }

          this.isLoading = false
        }
      }
    },
```

Anexo 3. Librerías utilizadas

Las principales dependencias utilizadas en el proyecto han sido explicadas en el apartado de arquitectura y son las siguientes:

- VueJS
- Vue-cli
- Vue Router
- Servicio de *hosting* de Firebase
- Servicio de Autenticación de Firebase
- Base de datos en tiempo real de Firebase
- *Framework* CSS Bulma
- Pug

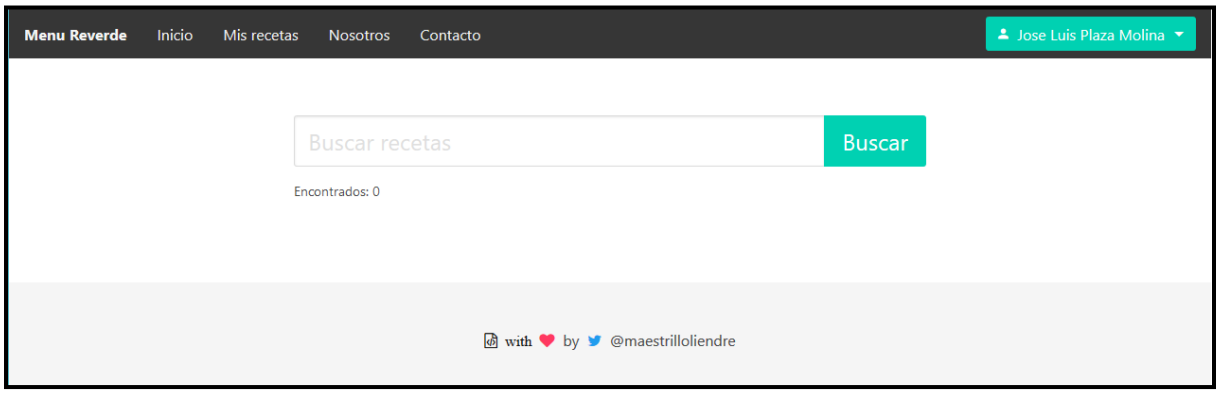
Todo esto corre bajo NodeJS, por lo tanto para ver el total de las dependencias adjunto el contenido del fichero “package.json”:

```
{
  "name": "tfg",
  "description": "Generador de menus para vegetarianos y veganos",
  "version": "1.0.0",
  "author": "maestrilloliendre <profesor.liendres@gmail.com>",
  "license": "MIT",
  "private": true,
  "scripts": {
    "dev": "cross-env NODE_ENV=development webpack-dev-server --open --hot",
    "build": "cross-env NODE_ENV=production webpack --progress --hide-modules"
  },
  "dependencies": {
    "buefy": "^0.6.3",
    "bulma": "^0.6.1",
    "firebase": "^4.11.0",
    "trae": "^1.4.2",
    "vue": "^2.4.4",
    "vue-carousel": "^0.6.5",
    "vuex": "^3.0.1"
  },
  "browserslist": [
    "> 1%",
    "last 2 versions",
    "not ie <= 8"
  ],
  "devDependencies": {
    "babel-core": "^6.26.0",
    "babel-loader": "^7.1.2",
    "babel-preset-env": "^1.6.0",
    "babel-preset-stage-3": "^6.24.1",
```

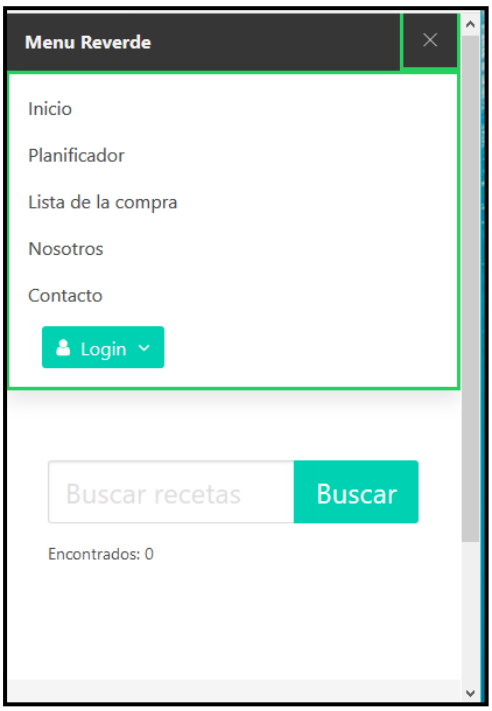
```
    "cross-env": "^5.0.5",
    "css-loader": "^0.28.7",
    "eslint": "^4.11.0",
    "eslint-config-standard": "^10.2.1",
    "eslint-loader": "^1.9.0",
    "eslint-plugin-html": "^4.0.0",
    "eslint-plugin-import": "^2.8.0",
    "eslint-plugin-node": "^5.2.1",
    "eslint-plugin-promise": "^3.6.0",
    "eslint-plugin-standard": "^3.0.1",
    "file-loader": "^1.1.4",
    "node-sass": "^4.5.3",
    "pug": "^2.0.0-rc.4",
    "pug-loader": "^2.3.0",
    "sass-loader": "^6.0.6",
    "vue-loader": "^13.0.5",
    "vue-router": "^3.0.1",
    "vue-template-compiler": "^2.4.4",
    "vuedraggable": "^2.15.0",
    "webpack": "^3.6.0",
    "webpack-dev-server": "^2.9.1"
  }
}
```

Anexo 4. Capturas de pantalla

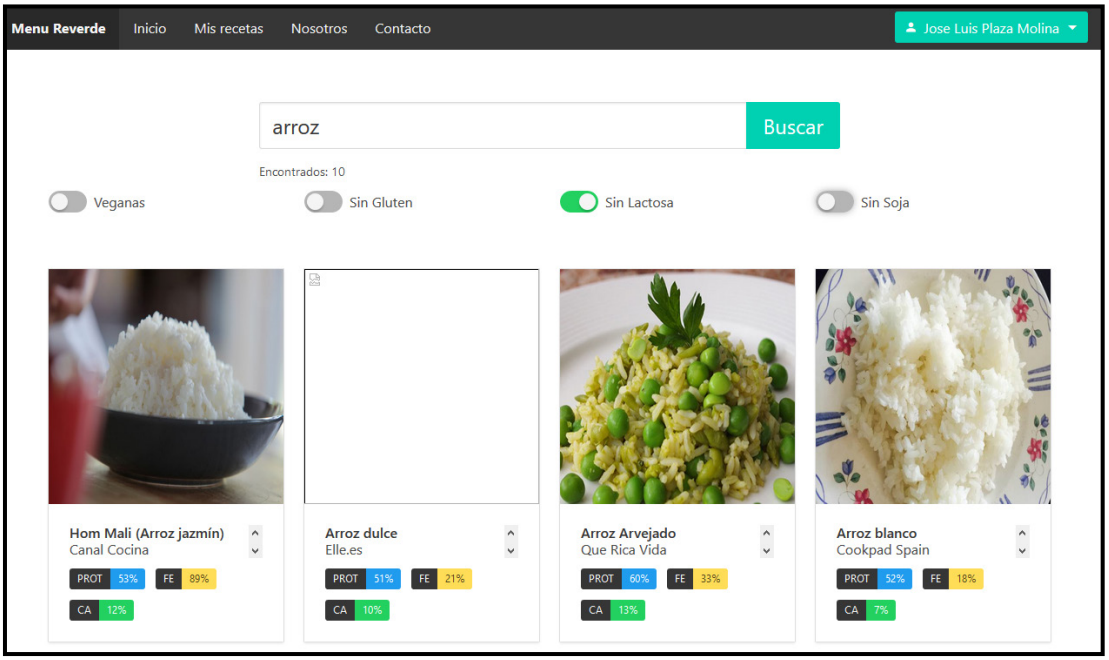
Buscador navegador web:



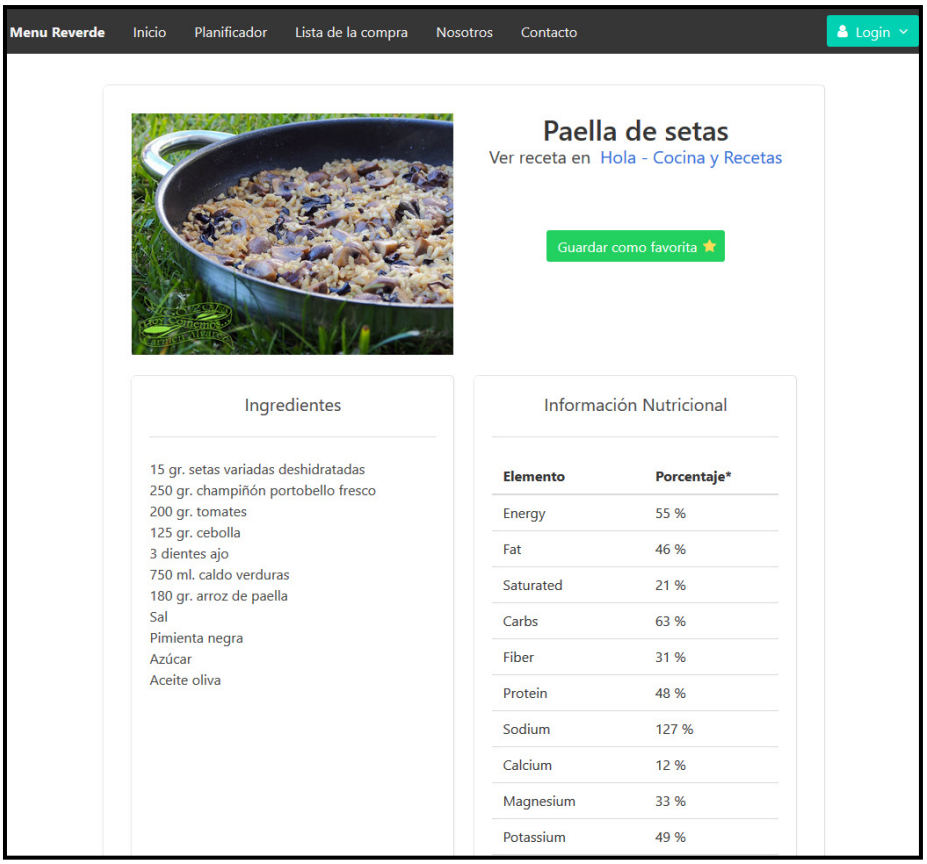
Buscador dispósitivos móviles:



Resultados búsqueda:



Detalle receta:



Anexo 5. Libro de estilo

Este proyecto se ha realizado utilizando un *framework* CSS llamado Bulma. Es por esto que el libro de estilo es el propio que utiliza Bulma. Se ha buscado información al respecto, pero no se ha encontrado ninguna guía de estilo del *framework* en cuestión.

Anexo 6. Resumen ejecutivo

Nombre comercial: Menu Reverde.

Resumen comercial: Aplicación web donde vegetarianos y veganos pueden elaborar su menú semanal con recetas de los mejores blogs nacionales y con información nutricional.

Modelo de negocio: En un principio se ha creado para ofrecer un servicio sin ánimo de lucro. A pesar de esto, a futuro se podrían ofrecer servicios premium bajo suscripción, como pudiera ser la creación de dietas personalizadas.

Productos y servicios: Se ofrece el poder elaborar menús equilibrados nutricionalmente de manera sencilla desde cualquier dispositivo.

Target: Vegetarianos, veganos o personas preocupadas por su salud y que les guste tener las comidas organizadas.

Competencia: Se han encontrado productos comerciales como lo página de Thermomix o la de Nestlé, pero no se consideran competencias como tal ya que ambas tienen como *target* a sus clientes.

Plan de márketing: Se utilizarán las redes sociales para promocionar el producto.

DAFO:

DEBILIDADES	**FORTALEZAS**
Bajo presupuesto. Mercado pequeño	Gratuito. Orientado a un público muy específico
AMENAZAS	**OPORTUNIDADES**
Problemas económicos	La gente cada vez está más preocupada por su salud y por lo que come.

Anexo 7. Bibliografía

Cómo autenticar mediante el Acceso con Google con JavaScript | Firebase. (2018). https://firebase.google.com/docs/auth/web/google-signin?hl=es-419

Desarrolla aplicaciones con VueJS. (2018) (1ª ed.). Madrid. https://www.gitbook.com/book/jdonsan/desarrolla-aplicaciones-con-vuejs

Documentación oficial VueJS. (2018). Vuejs.org. https://vuejs.org/v2/guide/