

**Planificador Web de Comidas Semanales**

* **MealWeek -**

# Ciclo: Desarrollo de Aplicaciones Web

# Curso: 2022/2023

# tutora: María Del Carmen Buenestado Fernández

# autor: Pedro Jesús Cruces Almendro

# Abstract

El proyecto «MealWeek» consiste en la realización de un sitio web con el que poder administrar y planificar todos los almuerzos y cenas que se realizan durante una semana. Se expondrán todos los requisitos previos necesarios para su desarrollo, así como detalles relevantes que tengas que ver con su proceso de creación. Además, se detallarán todo lo relevante a los tipos de usuarios y las distintas funcionalidades que poseerán y todo el uso de tecnologías necesarias para la construcción de la web.

The «MealWeek» project consists in the creation of a website to manage and plan all lunches and dinners that take place during a week. All the necessary prerequisites for its development will be presented, as well as relevant details that have to do with its creation process. In addition, everything relevant to the types of users and the different functionalities they will have, and all the use of technologies necessary for the construction of the website, will be detailed.

**ÍNDICE**

[1. Justificación 5](#_Toc133333592)

[2. Introducción 6](#_Toc133333593)

[2.1 Presentación 6](#_Toc133333594)

[2.2 Medios Hardware y software a utilizar 7](#_Toc133333595)

[2.2.1 Hardware 7](#_Toc133333596)

[2.2.2 Software 8](#_Toc133333597)

[2.3 Planificación y temporalización 9](#_Toc133333598)

[2.3.1 Etapa 1 – Diseño de la web y de la Base de Datos 9](#_Toc133333599)

[2.3.2 Etapa 2 – Diseño de la API y escribir el código de la primera vista de la web 10](#_Toc133333600)

[2.3.3 Etapa 3 – Escribir el código de interacción entre la vista y la Base de datos 10](#_Toc133333601)

[2.3.4 Etapa 4 – Escribir el resto de vistas y restringir según tipo de usuario 10](#_Toc133333602)

[2.3.5 Etapa 5 – Mejorar la maquetación de la web y de las funcionalidades de cada usuario 10](#_Toc133333603)

[3. Objetivos 11](#_Toc133333604)

[4. Desarrollo 12](#_Toc133333605)

[4.1 Primeros pasos 12](#_Toc133333606)

[4.2 Configuración de archivos e IDE 13](#_Toc133333607)

[4.2.1 Organización por carpetas y archivos 13](#_Toc133333608)

[4.3 Base de datos 16](#_Toc133333609)

[4.4 API 18](#_Toc133333610)

[4.4.1 Ventajas y Desventajas de la arquitectura REST 18](#_Toc133333611)

[4.4.2 Dependencias necesarias 18](#_Toc133333612)

[4.4.3 Endpoints 18](#_Toc133333613)

[4.5 Tipos de usuarios y nivel de acceso 19](#_Toc133333614)

[4.6 Interfaces de usuario 19](#_Toc133333615)

[4.7 Requerimientos generales y específicos 19](#_Toc133333616)

[4.8 Tests 19](#_Toc133333617)

[5. Conclusiones 20](#_Toc133333618)

[Funcionalidades a futuro 20](#_Toc133333619)

[6. Bibliografía prevista para el desarrollo del proyecto 21](#_Toc133333620)

[Anexo I 22](#_Toc133333621)

[Gráfico 1 22](#_Toc133333622)

[Anexo II 23](#_Toc133333623)

[Boceto 1 23](#_Toc133333624)

## Justificación

La creación de un proyecto con estas características, surgen de la necesidad propia del autor para poder gestionar de una forma rápida y sencilla tanto las comidas que suceden a lo largo de una semana en una casa, como la compra que debe de realizarse para el no malgasto de alimentos. Además, todo ello también beneficia a cualquier familia por el ahorro económico de hacer una compra más controlada y ajustada.

La no existencia de herramientas similares que lleguen a todo usuario de una forma simple y directa, hacen que este proyecto sea aún más necesario y que todo el público en general sea objetivo como usuario de la web.

## Introducción

### Presentación

El problema que se plantea en el proyecto es la realización de un portal que permita a cualquier usuario, planificar una semana de comidas y cenas, y permitirle la descarga en PDF tanto de la planificación, como de una lista de la compra que variará en función de lo que el usuario haya seleccionado.

Por lo que nuestra solución partirá por la diferenciación de dos tipos de usuarios: registrados y no registrados. Los usuarios no registrados solo tendrán acceso a la página principal en la que recibirán información acerca del proyecto y todas las funcionalidades que le puede aportar, mientras que los usuarios registrados podrán acceder a la toda la funcionalidad del sitio web.

Para satisfacer todas las funciones de los usuarios registrados, deberemos de contemplar al menos cinco componentes que debe tener nuestra aplicación web:

1. ***Planing Board***: en este componente, el usuario puede ver su planificación semanal y la lista de la compra resultante. Se le facilitará las opciones de editar, con el que podrá editar la planificación, y descargar como PDF. También podrá almacenar el planning en la web, en caso de que le haya gustado, para poder recuperarlo cuando desee.
2. **Editor del *Planning***: desde el editor, podrá construir el *planning* semanal a su gusto con comidas provenientes de su propia colección de comidas favoritas o también podrá generar un *planning* aleatorio de comidas según las comidas que tenga.
3. **Mis Plannings**: sección en donde se guardarán todos los plannings que quiera el usuario y donde podrá recuperarlos.
4. **Página de comida**: cada comida tendrá su propia página en la que tendrá todos los datos que la identifican además de tener una descripción con la preparación de la misma.
5. **Editor de comida**: En el editor de comida, se podrá crear comidas nuevas o editar las que el usuario haya creado. Además, si el usuario no encuentra los ingredientes que necesita, podrá crearlos y añadirlos a la base de datos general.
6. **Perfil de usuario**: En esta vista, el usuario podrá tener acceso tanto a sus datos de usuario, como a su propia colección de comidas favoritas, donde también tendrá acceso a su gestión.

Además, habría que tener formularios de registro para los usuarios no registrados y de log in para el que ya esté registrado.

Con respecto a la API, deberemos de poder escribir y acceder a las tablas de la base de datos y luego restringir los usos según la funcionalidad de la web. Los datos para la conexión a la base de datos deberán de estar protegidos en un archivo de «env», que procuraremos de incluir en el «gitignore» para evitar que se filtren con la subida al repositorio de «Github».

### Medios Hardware y software a utilizar

### Hardware

El hardware necesitado para la creación del proyecto no será especialmente diferente al usado para la creación del cualquier otro sitio web. En concreto, el ordenador utilizado para el desarrollo del proyecto es:

* Procesador: Ryzen 5 3600. 6 núcleos 12 hilos.
* Memoria Ram: 2 módulos de 8Gb Crucial Ballistix
* Disipador: Disipador AMD de serie del procesador
* Placa base: B550 Aorus Elite V2
* Almacenamiento: Western Digital Blue SN550 SSD 1Tb con interfaz NVMe M.2
* Fuente de alimentación: Tacens Radix VII AG 600W
* Tarjeta gráfica: KFA2 NVIDIA GeForce RTX 3060 12Gb
* Monitor: AOC Gaming 24G2U 24" LED IPS FullHD 144Hz FreeSync
* Ratón: Logitech G502 Hero
* Teclado: Mars Gaming MK220

### Software

En cuanto al software, deberemos de utilizar un IDE para la escritura y control del código como es “Visual Studio Code”. En este IDE, podremos instalar diferentes extensiones que nos ayudarán en el progreso de desarrollo del proyecto y nos asegurará la correcta aplicación del código que vayamos escribiendo.

[img IDE]

Necesitaremos de un control de versiones y documentación con el que poder llevar un control del software que vayamos produciendo. Para ello usaremos “Git” con el que podremos mantener este control mediante un repositorio en su sitio web “Github” y que podremos conectar fácilmente con nuestro IDE “Visual Studio Code”. Así desde el propio IDE, podremos realizar los *commit* que irán actualizando nuestro proyecto y podremos ver todos los cambios que han ido sucediendo de versión en versión.

[img GIT]

Para el diseño de bocetos de la web y de las posibles vistas que pueden ser necesarias, usaremos “Adobe XD”. Con dicho software, podremos diseñar fácilmente cualquier vista que necesitemos, además de poder simular las interacciones entre páginas dentro del sitio web y la exportación de algunos elementos a código.

[img Adobe XD]

Para la edición y retoque gráfico de imágenes, logos y diferentes iconos, haremos uso de “Adobe Photoshop” y de “Adobe Illustrator”. Con el primero podremos modificar perfectamente cualquier imagen y adecuarla al formato web, y con el segundo, podremos modificar cualquier logo e icono que vayamos a utilizar, mediante vectores de una forma sencilla.

[img AI y Ps]

Respecto a las tecnologías a usar en el código, nos decantaremos por el *framework* React, ya que nos permitirá crear una página web dinámica mediante componentes que serán fácilmente mantenibles y actualizables. También usaremos CSS *Flex* y *Grid* para maquetar el sitio web, y varias dependencias que nos ayudarán con la escritura del código como “JsPDF” o “Standard”.

[img Next.js]

### Planificación y temporalización

El proyecto se dividirá en diversas etapas que facilitarán la realización del mismo. Además de las etapas que se describirán a continuación, se puede consultar el gráfico 1 del anexo 1, en donde se plantea una temporalización aproximada por semanas de las diferentes etapas que componen el proyecto y su duración.

### Etapa 1 – Diseño de la web y de la Base de Datos

Comprendería el diseño gráfico de todos los elementos y vistas que componen el sitio web, además del diseño de la base de datos con todas las tablas y campos necesarios para el correcto funcionamiento de la web.

### Etapa 2 – Diseño de la API y escribir el código de la primera vista de la web

Esta etapa comprenderá la escritura del código necesario para la creación de la API que se conectará con la base de datos y con la que se accederán a todos sus datos. También se procederá a escribir el código de la primera vista de la web que funcionará como página de inicio para el usuario registrado.

### Etapa 3 – Escribir el código de interacción entre la vista y la Base de datos

En esta etapa, ya dispondremos tanto de la API como de una vista básica de la web con la que podremos comenzar a escribir todo el código, que relacionará la web con la base de datos y con las que podremos cumplir con los principios CRUD (*Create Read Update Delete*) desde el sitio web.

### Etapa 4 – Escribir el resto de vistas y restringir según tipo de usuario

Una vez que ya disponemos de toda la funcionalidad de la base de datos y podemos controlarla desde la web, procedemos a realizar el resto de vistas necesarias para los distintos usuarios y asegurarnos de restringir todos los permisos de cada usuario a su nivel correspondiente.

### Etapa 5 – Mejorar la maquetación de la web y de las funcionalidades de cada usuario

En esta etapa se ultiman los detalles finales relacionados tanto con el diseño como con la funcionalidad para que el usuario final no encuentre ningún tipo de problema, y finalizamos pasando la web a producción.

## Objetivos

* Creación de un sitio web que permita la realización de una planificación semanal de almuerzos y cenas y que genere una lista de la compra en base a dicha planificación
* Permitir que el usuario pueda elegir rellenar la planificación de forma manual, aleatoria o en función de distintos factores como la cantidad de carne semanal, pescado, alergias, etc.
* Permitir que cada usuario disponga de su propia galería de comidas donde podrá guardar las que más le gusten.
* Permitir que cada usuario disponga de una sección donde podrá consultar *plannings* que le haya gustado y que desee repetir.
* Permitir que cada usuario pueda crear comidas nuevas que nutrirán la colección general.

## Desarrollo

### Primeros pasos

Para asentar buenas bases a los cimientos del proyecto, deberemos de plantear en un comienzo la guía de estilo que usaremos en el mismo y algunos bocetos sobre las vistas más importantes.

[img logo]

Para el logotipo, se utiliza la fuente «Titan One» y se mezclan los glifos de la M y la W, siglas del título que da nombre al proyecto. Además, se utilizarán como colores claves el azul *#1A91C6* y el verde *#1AC13D*, combinándolos con blanco y negro dependiendo de la sección de la web para el fondo o para el color de la fuente. También usaremos pequeñas variaciones de estos para el resaltado de enlaces y botones.

[img colores] [img fonts]

Continuando con el diseño, para los *headers* usaremos la misma fuente del logo, para el texto general usaremos la fuente «Itim» y para algunos textos especiales, como los días de la semana de la planificación, usaremos la fuente «Architects Daughter».

Lo mejor para la instalación de las fuentes es convertir exclusivamente las variantes que nos interese al formato web WOFF2. Este formato, es la opción más óptima para cargar las fuentes a nuestro sitio web sin que tengamos que sufrir grandes tiempos de espera y sin tener que depender que el usuario disponga de las fuentes elegidas ya instaladas en su equipo o dispositivo.

[img iconos] [img excalidraw]

Respecto a los bocetos iniciales, podemos usar la herramienta «Excalidraw.com» con la que podremos dibujar fácilmente un pequeño esquema sobre la vista de la web y el cómo se comportará su interactividad. En este boceto (que se puede consultar en el boceto 1 del anexo 2), se puede observar cómo habría dos vistas principales, y el resto de ventanas podrían ser anexadas a estas vistas, usando *modals* o diálogos; o podrían configurarse en alguna vista nueva.

Una vez que el diseño de la web y que los elementos que lo componen están claros sobre la guía de estilo, estaremos listos para proceder a comenzar con la realización del proyecto.

### Configuración de archivos e IDE

Una vez que tenemos la guía de estilo planteada, necesitamos establecer que tipos de tecnologías vamos a usar para la construcción de todo el sitio web. Teniendo en cuenta que vamos a utilizar tecnologías de *front-end* para todo el control del diseño y del cliente, y de *back-end* para la creación y uso que gestione la base de datos, una de las soluciones más adecuadas sería «Next.JS».

[img React] [img Next.js]

«Next.JS» es un *framework* de *React* (que a su vez es un *framework* de *JavaScript*), con el que podremos construir un sitio web con todas las ventajas que nos ofrece de por sí *React*, sumado a la posibilidad de crear componentes que se rendericen en el servidor (*ServerSideRendering*), y la posibilidad de crear fácilmente un enrutado de páginas y *endpoints* para una API.

La instalación de este *framework* es sencilla, pero nos creará archivos y carpetas por defecto que deberemos de entender para su buen uso, ya que, de lo contrario, provocaremos muchos errores que nos entorpecerá la progresión del proyecto.

### Organización por carpetas y archivos

Respetar la organización de directorios en Next.JS es crucial para su funcionamiento. Dentro de la carpeta de nuestro proyecto, donde hemos instalado el framework, encontraremos:

[img screen folders]

* **.*next***: dónde se guardará información y archivos de configuración relevantes para el buen funcionamiento de todo el framework.
* ***node\_modules***: directorio en el que se instalan todos los modulos necesarios para el funcionamiento de la máquina virtual de «node» que ejecutará nuestro proyecto en local y con el que podremos crear una vista previa de nuestro proyecto mientras lo escribimos. También nos genera una compilación de nuestro código cuando lleguemos a la conclusión del proyecto.
* ***Pages***: este directorio es de los más importantes. En su interior, en el que cualquier directorio o archivo .js o .jsx (o sus variantes con *typescript*), serán considerados como páginas y rutas independientes, por lo que no habrá que configurar un *router* como sí podríamos hacer en *React*. Además, en su interior, tenemos el directorio llamado API, en el que al igual que sucede con el directorio pages, cualquier archivo o directorio en su interior será considerado como una ruta, pero esta vez hacia una API. Por lo que la creación y manejo de una API en este *framework* se vuelve muy fácil y conveniente.
* ***Public***: este directorio no esconde muchos secretos, ya que su función es exactamente la misma que en cualquier otro proyecto web.
* ***Styles***: *Next.JS* usa por defecto la tecnología *CSS Modules*, lo que permite separar el archivo *css* en módulos distintos según los componentes de nuestra web o páginas, creando así un rango más específico. Por ejemplo, podríamos escribir los mismos nombres de clases en distintos módulos *css* y estos, no entrarían en conflicto en ningún momento. Este directorio, por tanto, es el utilizado para albergar todos los módulos, además de incluir un globals.css que afectará a todas las páginas en general.
* .***gitignore***: este archivo es uno de los más importantes para la no filtración de datos sensibles. Al usar *Git* como nuestro controlador de versiones, necesita saber que archivos debe o no debe de incluir en el repositorio que estará disponible en la red. Es por ello, que, dentro de este archivo, deberemos de incluir todas las rutas (relativas) que no deberán de subirse al repositorio de *Github*.
* ***jsconfig.json***: este archivo, configura nuestro editor y nos permite incluir o excluir directorios de cara a la compilación.
* ***next.config.js***: archivo en el que incluiremos configuración específica de *Next.JS* y en el que, por ejemplo, deberemos de incluir dominios para poder usar los componentes *Image*.
* ***package.json***: en este archivo de formato *json*, aparece la información relativa a nuestro proyecto, como el nombre y la versión, además de comandos, como “npm run dev” con el que tendremos una vista previa del proyecto o “npm run build” con el que podremos compilarlo. También aparece en detalle, las dependencias necesarias para el funcionamiento de la web y la versión requerida. Además, podremos incluir instrucciones respecto a la configuración del *linter*, con la que ayudarnos a la hora de escribir código.
* ***package-lock.json***: Funciona como una copia de las versiones de las dependencias definidas en el *package.json*. Cualquier cambio que hagamos en el *package.json*, al ejecutar la actualización de una dependencia o al instalarlas todas con “npm install”, se actualizará *package-lock.json* con la versión más reciente instalada.
* ***README.md***: este archivo, como en el resto de proyectos web, funciona como la documentación principal de nuestro proyecto.

Una vez que disponemos y entendemos la jerarquía de los directorios y su función, podremos pasar a configurar el IDE. Para este proyecto, usaremos el *software* *VSCode*, IDE que además de facilitarnos la conexión con nuestro repositorio de *Github*, también nos permite la instalación de diferentes *plugins* que ayudarán con la depuración del código. Entre otros, usaremos un plugin para desplegar un *intellisense* sobre *javascript* y *react*, y un paquete que nos configurará el *linter*. En primer lugar, el *intellisense* nos sugerirá a medida que vamos escribiendo, métodos, clases y atributos que pertenezcan a lo que estemos escribiendo, además de variables, constantes y funciones que tengamos en nuestro documento. Por ejemplo, si escribimos el objeto “document”, al poner el punto, se abrirá un desplegable que te sugerirá todos los métodos accesibles de dicho objeto.

[img eslint] [img screen eslintconfig]

Respecto al *linter*, es el plugin que nos señaliza la sintaxis que no está escrita de manera correcta, o si hay cualquier otro error que hubiese en el código. Este *linter*, se apoya de una configuración que nos permite personalizar a nuestro gusto puesto que hay reglas en sintaxis que corresponden más a un estilo de escritura, que, al correcto funcionamiento de nuestro código, como puede ser el escribir “;” después de cada instrucción en *JavaScript*. Para este caso, instalaremos la dependencia “standard” con el que obtendremos una configuración del *linter* casi perfecta. Para completarla, solo deberemos de añadir las reglas que queramos cambiar al *package.json*, bajo *eslintConfig*, y asignarle un valor *off*, *warn* o *error*, según como queramos que nos lo señalice.

Además, necesitaremos de la instalación de dependencias que nos faciliten, no solo la escritura de código, sino la habilitación de ciertas capacidades a nuestra página web. De hecho, necesitaremos de la instalación de la dependencia “mysql2” con la que podremos establecer conexión con nuestra base de datos y realizar cualquier tipo de *query*, y la dependencia “bcrypt” con la que podremos encriptar *passwords* y realizar su posterior autenticación. Todo ello se explicará con mayor profundidad en las próximas secciones.

[img screen depend. Package json]

Teniendo ya una buena configuración del IDE, podemos comenzar a escribir la solución para nuestro proyecto.

### Base de datos

Para comenzar, nuestra solución deberemos de diseñar una base de datos en la que almacenaremos todos los datos de autenticación de usuarios, comidas y las comidas favoritas que tenga cada usuario. Usaremos MySQL, que se trata de una base de datos relacional con la que podremos, tanto almacenar como consultar datos de forma estricta para poder garantizar la integridad de los mismos. Dentro de la base de datos, el esquema, deberá de estar formado por tres tablas:

[img Tablas y diagramas E/R]

* **Users**: Esta tabla contendrá los datos de cada usuario que se registre en el sitio. Contendrá los campos de *id*, *username*, *password* y *email*, de los cuales id, *username* y *email*, deberán de ser únicos para garantizar la correcta identificación de cada usuario. También hay que considerar de que no es buena práctica almacenar *passwords* en la base de datos sin una correcta gestión de ellos. Es por ello, que necesitaremos de algunas herramientas que nos ayuden a su encriptación y correcto almacenaje (se explicará en la siguiente sección de forma más detallada.)
* **Plannings**: Esta tabla contendrá los plannings guardados de cada usuario. La forman los campos de *planning\_id*, *user\_id*, *plan* y *time*.
* **Meals**: Esta tabla contendrá los datos de cada comida creada por los usuarios y que estará disponible para todos ellos. Contendrá los campos de *id*, *author*, *name*, *description*, *composition* e *ingredients*, de los cuales *id* y *name*, deberán de ser únicos para asegurar que no se almacenan dos entradas iguales. El campo de *author*, nos ayudará a relacionar esta tabla con la de *users*, y el campo de *composition*, será el que nos ayude a identificar la composición de las comidas y así evitar posibles alergias o comidas vegetarianas, etc…
* **Ingredients**:Esta tabla es necesaria para el control de los ingredientes que se añaden a las comidas. Sólo contendrá dos campos: *ingredients\_id* y *name*. Cabe añadir, que el campo *name* deberá ser único para que no exista conflictos de duplicados.
* **Shopping\_List**:Esta tabla es la que especifica que cantidad de ingredientes componen las comidas. De esta manera, podemos generar la lista de la compra de forma adecuada. Estará formada por *shopping\_list\_id*, *meal\_id*, *ingredient\_id*, *quantity* y *type*. Necesitamos de los *id* de las comidas y los ingredientes para su correcta relación y, además, registraremos la cantidad y el tipo de medida usada, como “kg” o “ml”.
* **Favs-Meals**: En esta tabla se almacenarán los datos de las comidas que los usuarios seleccionen como favoritas, y, por tanto, aparecerán en un apartado donde tendrán todas las comidas favoritas. Solo contendrá los campos de *id*, *user\_id* y *meal\_id*, ya que sólo necesitaremos identificar que usuario ha seleccionado como favorita que comida.

### API

Para el almacenaje y la consulta de datos necesitaremos de una API. Esta API se conectará directamente con la base de datos y nos permitirá acceder a todos los registros que necesitemos. Para su correcto uso y para asegurar que estamos creando una aplicación usando buenas prácticas, tendremos que tener en cuenta los principios de la arquitectura REST:

* A
* A
* A
* A
* A

### Ventajas y Desventajas de la arquitectura REST

### Dependencias necesarias

### Endpoints

### Tipos de usuarios y nivel de acceso

### Interfaces de usuario

### Requerimientos generales y específicos

### Tests

## Conclusiones

### Funcionalidades a futuro

## Bibliografía prevista para el desarrollo del proyecto

CSS-Tricks. (2023). *Home*. [Consultado el 06 de 04 de 2023] de CSS-Tricks: <https://css-tricks.com/>

Durán, M. Á. (2023). *Home*. [Consultado el 06 de 04 de 2023] de Midudev: <https://midu.dev/>

MDN. (2023). *Home*. [Consultado el 06 de 04 de 2023] de MDN Web Docs: <https://developer.mozilla.org/en-US/>

Meta Open Source. (2023). *Home*. [Consultado el 06 de 04 de 2023] de React: <https://react.dev/>

Next.JS. (2023). *Home*. [Consultado el 06 de 04 de 2023] de Next.JS: <https://nextjs.org/>

Oracle. (04 de 04 de 2023). *MySQL 8.0 Reference Manual*, revision: 75301. [Consultado el 06 de 04 de 2023] de MySQL: <https://dev.mysql.com/doc/refman/8.0/en/>

Powell, K. (2023). *Home*. [Consultado el 06 de 04 de 2023] de Kevin Powell: <https://www.kevinpowell.co/>

# ANEXOS

## Anexo I

### Gráfico 1

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ETAPAS / SEMANAS** | **ABRIL** | | | | **MAYO** | | | |
| **1** | **2** | **3** | **4** | **1** | **2** | **3** | **4** |
| **Etapa 1 – Diseño de la web y de la Base de Datos** | x | x |  |  |  |  |  |  |
| **Etapa 2 – Diseño de la API y escribir el código de la primera vista de la web** |  | x | x |  |  |  |  |  |
| **Etapa 3 – Escribir el código de interacción entre la vista y la Base de datos** |  |  | x | x | x |  |  |  |
| **Etapa 4 – Escribir el resto de vistas y restringir según tipo de usuario** |  |  |  | x | x | x | x |  |
| **Etapa 5 – Mejorar la maquetación de la web y de las funcionalidades del usuario** |  |  |  |  |  |  | x | x |
| **Conclusión del proyecto** |  |  |  |  |  |  |  | x |

## Anexo II

### Boceto 1

