1. **Presentación del grupo de trabajo y objetivos de proyecto.**

El proyecto trata del desarrollo de una web de temática libre en la que plasmar los conocimientos adquiridos durante el semestre con los lenguajes de programación: HTML, CSS, JavaScript, JQuery, PHP, MySQL...

Añadiendo además otros factores a tener en cuenta como son la estructura (front-end y back-end) o la accesibilidad, entre otras.

**2. Documento de requisitos y prototipos sobre los que se ha trabajado.**

1. **Cambios efectuados y justificación sobre el documento de requisitos original y wireframes originales.**

Pese a mantenernos bastante fieles al documento de requisitos original, en aquel momento no teníamos del todo claro lo que queríamos, lo que nos ha permitido tomar lo que teníamos como una base y mejorarlo, potenciarlo y sacarle más provecho del que tenía.

Un claro ejemplo de ello es la página principal, lo que teníamos claro era el menú de opciones que clasificaría las recetas y permitiría la navegación; sin embargo, aparte de la cabecera y el pie de página, el resto se quedó en blanco. Eso nos ha permitido dar rienda suelta a las posibilidades que ofrece el mundo web y reorientarlo.

Podemos afirmar que, salvo ese cambio, que es el más notable, hemos sido fieles al documento original.

1. **Detalles de análisis y diseño.**
2. **Modelo (DB)**

Primero de todo, realizamos un modelo entidad-relación que nos permitiera ver las entidades que participarían en la web. A continuación, lo pasamos a tablas. Sin embargo, ese no fue el producto final ya que, tras realizar una parte de la web, para ajustarnos más a la implementación, consideramos modificar ligeramente lo que ya teníamos.

Partimos de las tablas: usuario, categoría, receta, foto. Como mencionado, se vio que, en las recetas, para darles un mejor formato y más personalidad, era más útil desplegar la tabla recetas en otras dos: pasos e ingredientes.

1. **Patrones de diseño**

Los patrones que hemos considerado más adecuados son:

- Para el controlador: el *front controller* 🡪 delegar las tareas a un controlador que irá llamando a lo que sea necesario según la información que venga.

- Para las vistas: *template view* 🡪 usar plantillas permite ahorrar código y que éste sea más limpio, delegando lo que es la información importante al servidor.

- Para los datos: *table gateway* 🡪 tras realizar una clase php para la conexión con la base de datos, ésta es usada por un gateway que gestiona las operaciones sobre la base de datos evitando así que haya la gestión de sentencias SQL en múltiples ficheros.

1. **Detalles de implementación.**

**a. Patrones**

Como hemos mencionado anteriormente hemos seleccionado el patrón *front controller* para la implementación del controlador. De esta manera hemos creado un fichero php con nombre *index*, este fichero recibirá todas las peticiones realizadas por el cliente. Vamos a definir las acciones que debe realizar en base a dos parámetros, acción e id. En un swicth el controlador ira comprobando la acción y dentro de esta el id y realizara acciones necesarias con ayuda del modelo y la vista.

El modelo que hemos mencionado es un fichero php con nombre modelo que se encarga de realizar todas las consultas a la base de datos mediante la clase *DBConexion*. De esta manera implementa el patrón *table Gateway*. El modelo únicamente contiene todas las funciones necesarias para responder a la solicitud del controlador, proporcionándole datos de la base de datos.

La vista mencionada es un fichero php con nombre vista que se encarga de servir al cliente la página web. Contiene todas las funciones necesarias para devolver una vista al cliente. Estas funciones toman como base las plantillas HTML que hemos definido, a partir de las plantillas y con los datos que el modelo le ha proporcionado al controlador modifica la plantilla creando la página final que obtendrá el cliente como resultado. De esta manera la vista implementa el patrón *template view*.

1. **Detalles de despliegue. (todavía no lo hemos dado)**
2. **Auditoría rendimiento.**
3. **Pruebas realizadas.**

Prueba 1: página principal.

Prueba 2: listado de recetas.

Prueba 3: vista de una receta.

Prueba 4: galería de imágenes de una receta.

Prueba 5: perfil de usuario.

Prueba 6: formulario de creación de una receta.

Prueba 7: formulario de modificación de una receta.

Prueba 8: formulario de registro / login.

1. **Resultados obtenidos.**

**Prueba 1**: la auditoria de rendimiento de Chrome otorga a la página principal un 87%. Los resultados obtenidos son los siguientes: first content Paint 0.9 s, speed index 1.1 s, largest contentful paint 2.2 s, time to interactive 1.3 s, total blocking time 0 y cumulative layout shift 0.006 s. Además ofrece una serie de indicaciones, eliminar los recursos que bloquean el renderizado, eliminar los ficheros CSS no utilizados y precargar solicitudes de claves.

**Prueba 2**: la auditoria de rendimiento de Chrome otorga al listado de recetas un 89%. Los resultados obtenidos son los siguientes: first content Paint 1 s, speed index 1 s, largest contentful paint 2 s, time to interactive 1.1 s, total blocking time 0 y cumulative layout shift 0 s. Además ofrece una serie de indicaciones, utilizar imágenes de tamaño adecuado, eliminar los recursos que bloquean el renderizado, eliminar ficheros CSS y JS no utilizados.

**Prueba 3**: la auditoria de rendimiento de Chrome otorga a la vista de una receta un 89%. Los resultados obtenidos son los siguientes: first content Paint 0.9 s, speed index 0.9 s, largest contentful paint 2 s, time to interactive 1.5 s, total blocking time 0 y cumulative layout shift 0.011 s. Además ofrece una serie de indicaciones que son las mismas que ha ofrecido en la prueba anterior.

**Prueba 4**: la auditoria de rendimiento de Chrome otorga a la galería de imágenes de una receta un 89%. Los resultados obtenidos son los siguientes: first content Paint 0.8 s, speed index 0.8 s, largest contentful paint 2 s, time to interactive 1.1 s, total blocking time 0 y cumulative layout shift 0 s. Además ofrece una serie de indicaciones, utilizar imágenes de tamaño adecuado, eliminar los recursos que bloquean el renderizdo y eliminar los ficheros CSS que no se utilizan.

**Prueba 5:** la auditoria de rendimiento de Chrome otorga al perfil de usuario un 95%. Los resultados obtenidos son los siguientes: first content Paint 0.9 s, speed index 1 s, largest contentful paint 1.3 s, time to interactive 1.5 s, total blocking time 30 ms y cumulative layout shift 0 s. Además ofrece una serie de indicaciones, eliminar los recursos que bloquean el renderizado, utilizar imágenes con tamaño adecuado, eliminar los ficheros CSS no utilizados y precargar solicitudes de claves.

**Prueba 6**: la auditoria de rendimiento de Chrome otorga al formulario de creación de receta un 96%. Los resultados obtenidos son los siguientes: first content Paint 0.9 s, speed index 0.9 s, largest contentful paint 1.3 s, time to interactive 1 s, total blocking time 0 s y cumulative layout shift 0 s. Además ofrece una serie de indicaciones, eliminar los recursos que bloquean el renderizado y eliminar los ficheros CSS que no se utilizan.

**Prueba 7**: la auditoria de rendimiento de Chrome otorga al formulario de modificación un 91%. Los resultados obtenidos son los siguientes: first content Paint 1.2 s, speed index 1.2 s, largest contentful paint 1.5 s, time to interactive 1.2 s, total blocking time 0 s y cumulative layout shift 0 s. Además ofrece una serie de indicaciones, eliminar los recursos que bloquean el renderizado, eliminar los ficheros CSS no utilizados y utilizan imágenes de tamaño adecuado.

**Prueba 8**: la auditoria de rendimiento de Chrome otorga al formulario de login/registro un 94%. Los resultados obtenidos son: first content Paint 0.9 s, speed index 1 s, largest contentful paint 1.4 s, time to interactive 1.3 s, total blocking time 0 s y cumulative layout shift 0 s. Además ofrece una serie de recomendaciones, eliminar los recursos que afectan al renderizado y eliminar los ficheros CSS no utilizados.

1. **Modificaciones propuestas.**

**Prueba 1**: en base a los resultados obtenidos proponemos utilizar imágenes más pequeñas o que no afecten tanto al renderizado y eliminar todos los ficheros CSS y JS que no se utilizan.

**Prueba 2**: en base a los resultados obtenidos proponemos redimensionar las imágenes al tamaño en el que van a ser mostradas en la web y eliminar todos los ficheros CSS y JS que no se utilizan.

**Prueba 3**: en base a los resultados obtenidos las modificaciones propuestas son las mismas que han sido propuestas en la prueba anterior.

**Prueba 4**: en base a los resultados obtenidos proponemos redimensionar las imágenes al tamaño en el que van a ser mostradas en la web y eliminar los ficheros CSS que no se utilizan.

**Prueba 5**: en base a los resultados obtenidos las modificaciones propuestas son redimensionar las imágenes al tamaño en el que van a ser mostradas y eliminar los ficheros CSS que no se utilizan.

**Prueba 6**: en base a los resultados obtenidos se propone eliminar todos los ficheros CSS que no se utilizan.

**Prueba 7**: en base a los resultados obtenidos se propone redimensionar las imágenes al tamaño en el que van a ser mostradas y eliminar los ficheros CSS no utilizados.

**Prueba 8**: en base a los resultados obtenidos se propone eliminar todos los ficheros CSS que no se utilizan.

1. **Auditoria SEO.**

**a. Pruebas realizadas.**

Para las pruebas de SEO hemos realizado una prueba por página y para ello hemos utilizado las herramientas de desarrolladores de Chrome.

Prueba 1: página principal.

Prueba 2: listado de recetas.

Prueba 3: vista de una receta.

Prueba 4: galería de imágenes de una receta.

Prueba 5: perfil de usuario.

Prueba 6: formulario de creación de una receta.

Prueba 7: formulario de modificación de una receta.

Prueba 8: formulario de registro / login.

1. **Resultados obtenidos.**

**Prueba 1**: la auditoria SEO de Chrome a otorgado a la página principal un 90%. Indica que le falta un atributo *meta description*. Además, hay que añadir que 4 auditorias no han podido ser evaluadas.

**Prueba 2**: la auditoria SEO de Chrome a otorgado a la página que contiene un listado de recetas un 80%. Las indicaciones recibidas son las siguientes: falta un atributo *meta description* ylas imágenes de la página no tienen atributo *alt*.

**Prueba 3**: la auditoria SEO de Chrome a otorgado a la página de la vista de una receta un 80%. Los motivos son los mismos que en la prueba anterior, falta el atributo *meta description* y las imágenes no tienen el atributo *alt.*

**Prueba 4**: la auditoria SEO de Chrome a otorgado a la galería de imágenes un 80%. Los motivos vuelven a ser los mismos, falta el atributo *meta description* y las imágenes no tienen el atributo *alt.*

**Prueba 5**: la auditoria SEO de Chrome a otorgado al perfil de usuario un 80%. Los motivos vuelven a ser los mismos, falta el atributo *meta description* y las imágenes no tienen el atributo *alt.*

**Prueba 6**: la uditoria SEO de Chrome a otorgado al formulario de creación de una receta un 90%. La indicación que ofrece es que falta el atributo *meta descripction.*

**Prueba 7**: la auditoria SEO de Chrome a otorgado al formulario de modificación de una receta un 80%. Los motivos vuelven a ser los mismos, falta el atributo *meta description* y las imágenes no tienen el atributo *alt.*

**Prueba 8**: la auditoria SEO de Chrome a otorgado al formulario de registro/login un 90%. La indicación que nos ofrece es que falta el atributo *meta description.*

1. **Modificaciones propuestas.**

**Prueba 1**: la modificación que se propone es añadir el atributo indicado a la plantilla HTML del header.

**Prueba 2**: debemos introducir el atributo *alt* en todas las imágenes, ya que también es importante para valorar la accesibilidad, además de añadir el atributo *meta description* a la plantilla HTML del header.

**Prueba 3:** las modificaciones adecuadas con las mismas que las propuestas para la prueba anterior.

**Prueba 4**: las modificaciones adecuadas vuelven a ser las mismas que han sido propuestas para la prueba 2.

**Prueba 5**: las modificaciones adecuadas vuelven a ser las mismas que han sido propuestas para la prueba 2.

**Prueba 6**: la modificación adecuada en este caso es la misma que ha sido propuesta para la prueba 1.

**Prueba 7**: las modificaciones adecuadas son las mismas que han sido propuestas para la prueba 2.

**Prueba 8**: la modificación adecuada en este caso en la misma que ha sido propuesta en la prueba 1.

1. **Auditoría accesibilidad.**

**a. Auditoría realizada.**

**b. Resultados obtenidos.**

**c. Modificaciones propuestas.**

1. **Pruebas de usabilidad.**

**a. Pruebas realizadas.**

**b. Resultados obtenidos.**

**c. Modificaciones propuestas.**

1. **Gestión de configuración.**
2. **Uso de repositorios**

Para esta práctica final hemos utilizado un repositorio común en GitLab, gracias al cual podemos gestionar correctamente los cambios y las versiones, de forma coordinada y eficaz.

1. **Integración continua**

**c. Trabajo en equipo**

Consideramos que ha habido un buen balance de cargas. Tras unos días en los que fuimos delegando las diferentes tareas principales en los distintos miembros del grupo, finalmente supimos organizarnos bien y trabajar a buen ritmo.

Comenzamos realizando pequeñas tareas, como la creación del logotipo (dibujado por Carla, así que está libre de copyright), el nombre de la empresa, etc. Vlad y Ainara comenzaron ideando la estructura de la página (header, footer, etc). Mientras, Carla y Jorge plantearon una primera versión de la base de datos de la que se hablaba en puntos anteriores y que posteriormente corregiríamos en clase entre todos, y Carla se ocupó del poblado de la misma con recetas de ejemplo para poder realizar pruebas durante la implementación de los modelos y la conexión con la base de datos.

Algo que vino de novedad fueron las aportaciones de Jorge; como ya tenía algo de experiencia en realizar webs dio a conocer el mundo de Bootstrap, un framework de CSS muy completo. Para enseñar su funcionamiento y el gran abanico de posibilidades que ofrecía, se encargó de diseñar las plantillas (o vistas, en lo que al patrón MVC se refiere). Dicha tarea consiste, en primer lugar, en el diseño de plantillas HTML, estilado con CSS, definición del comportamiento mediante Javascript. Por otro lado, en el lado del servidor, Jorge creó una serie de clases que permitían rellenar los “placeholders” y listas de elementos que aparecen en dichas plantillas, además de implementar diferentes clases, que permiten hacer uso de las diferentes plantillas. Por último, y para permitir el acceso a todas esas vistas, procedió a crear un documento “.htaccess”, que redirigiese todas las URLs a un mismo script, que también implementó (“controller.php”). Dicho script se encarga de crear y mostrar las vistas y, en el futuro, se encargará de realizar los cambios pertinentes en el modelo (haciendo de nuevo referencia al patrón MVC).

[…]

1. **Conclusiones**
2. **Anexos**