

MEMORIA ESCRITA DEL PROYECTO

CFGS Desarrollo de Aplicaciones Web

Título del proyecto

Autor: Juan José Saborido Barranco

Tutor: Fernando Méndez Castellar

Fecha de entrega: 14/06/2024

Convocatoria: 2S2324 (semestre de la convocatoria)

Documentos del proyecto: [Proyecto DAW - Saborido Barranco. Juan José](#)



Índice de contenidos

1. INTRODUCCIÓN	2
1.1. Motivación	2
1.2. Abstract	3
1.3. Objetivos propuestos (generales y específicos)	4
2. Estado del Arte	6
3. METODOLOGÍA USADA	8
4. TECNOLOGÍAS Y HERRAMIENTAS UTILIZADAS EN EL PROYECTO	11
5. PLANIFICACIÓN, DIAGNÓSTICO Y CONTEXTO LABORAL	14
6. ANÁLISIS DEL PROYECTO	17
7. DISEÑO DEL PROYECTO	26
8. DESPLIEGUE Y PRUEBAS	35
9. CONCLUSIONES	46
10. VÍAS FUTURAS	47
11. BIBLIOGRAFÍA/WEBGRAFÍA	48
12. ANEXOS	49
1.1. MANUAL DE USUARIO	72

1. Introducción

1.1. Motivación

Dadas las necesidades y el consumo de nuestra sociedad actual, cada vez se hace más relevante la demanda por parte de los consumidores de tener un control sobre qué comemos, cuándo lo comemos y cómo lo comemos, para así evitar desequilibrios alimenticios y problemas de nutrición o, simplemente, para mejorar nuestra salud física a través de una buena dieta. Poco a poco vamos tomando conciencia de qué alimentos son bajos en grasas, no tienen azúcares añadidos, son alimentos procesados o 100% naturales, etcétera.

En este sentido, la educación también es un aspecto bastante importante y, desde pequeños, se nos educa a los/as niños/as que debemos comer frutas y verduras y aparte de las comidas que más nos gustan. La organización de todas estas comidas puede ser, muchas veces, un trabajo que puede llegar a suponer un verdadero quebradero de cabeza para todas las personas que están a cargo de dichos menores o que, sencillamente, quiere aplicar lo que les enseñaron años atrás sus padres y madres.



Fig. 1 Bebé tomando biberón

Puesto que existe cada vez una mayor preocupación por estas cuestiones y teniendo en cuenta que el tiempo para la mayoría de dichas personas es algo limitado, se hace cada vez más necesario el apoyo de las diversas tecnologías para tratar todas estas cuestiones relacionadas con una dieta y alimentación equilibrada.

1.2. Abstract

Due to the needs and consumption of our current society, the demand from consumers to have control over what we eat, when we eat and how we eat, is becoming more and more relevant in order to avoid nutritional imbalance and nutritional problems or just to improve our physical health through a good diet. Little by little we are becoming aware of which foods are low in fat, do not have added sugars, are processed foods or are 100% natural, and so on.

Furthermore, the education here is also a very important aspect and, since we are young, we are taught that we should eat fruits and vegetables and not only the dishes that we enjoy the most and the organization of all these meals can be, quite often, a job that can be a real headache for all those people who are in charge of those kids or maybe they just want to apply all the knowledge that they received from their parents in the past.



Fig. 1 Bebé tomando biberón

Due to the fact that there exists a bigger concern about these questions and having in mind that the time is very limited for a lot of people, the support of various technologies is becoming more and more necessary to treat all these issues related to diet and balanced nutrition.

1.3. Objetivos propuestos (generales y específicos)

La aplicación web que vamos a crear, aspira a hacerse un hueco en el mercado de aplicaciones nutricionales, ofreciendo una interfaz simple, pero efectiva para el usuario, respondiendo a las necesidades de los mismos, teniendo en cuenta siempre su feedback o cómo podría mejorar la experiencia de usuario.

Con esto en mente, podemos separar nuestros objetivos en generales y específicos, donde los primeros se centrarán en tener un propósito de carácter general, enfocándose sobre todo en los resultados globales y los segundos tienen un propósito más específico y potencialmente más técnicos que harán posible cumplir con los objetivos generales del proyecto

Objetivos generales:

- **Creación de un calendario personal por usuario:** cada usuario podrá tener guardada en su base de datos un calendario personalizado, en función de sus recetas y que le servirá para calcular los valores nutricionales de las comidas que elija a lo largo de la semana.
- **Creación/Edición/Eliminación de recetas por parte de los usuarios:** cada usuario contará con una lista de recetas propias que podrá editar, añadir o eliminar
- **Cálculo de nutrientes de cada receta, de cada consumo diario y de consumo semanal.**
- **Separación de los usuario por roles.**
- **Diferentes vistas en función del tipo de usuario** que acceda a la aplicación (usuario registrado o administrador).
- **Separación total entre cliente y servidor:** haciendo que tanto uno como el otro estén completamente separados y comunicados entre sí.

Objetivos específicos:

- Implementación de un sistema de registro y de login, donde el primero permite insertar un nuevo usuario en nuestra base de datos en caso de que no haya nadie actualmente registrado ya con dicho correo y con sus correspondientes validaciones.
- Permitir modificar el calendario para que este se amolde a las necesidades del momento de los usuarios. Además, siempre que se quiera durante la modificación de dicho calendario, se podrán calcular los valores nutricionales temporales o definitivos de dicho calendario, teniendo valores totales en función del día, en función de las comidas y totales de todo el calendario.
- Permitir a los usuarios introducir recetas en la aplicación, con los diferentes ingredientes y cantidades y tener una tabla de ingredientes genéricos donde se encuentren los valores nutricionales de los mismos, guardándose así los valores nutricionales totales de dicha receta y actualizándose en caso de que esta se modifique.
- Hacer distinción entre usuarios y administradores, donde los primeros tengan control sobre sus recetas y su calendario y los segundos tengan control de los propios usuarios, pudiendo banear a alguno en caso de que así se considere oportuno.
- Capacidad, por parte de la aplicación, de poder mantener una sesión activa a menos que el usuario decida poner fin a la misma.
- Garantizaremos en todo momento la integridad y seguridad de nuestros datos codificando la contraseña proporcionada por el usuario y blindando nuestra aplicación frente a posibles intentos de inyección SQL.

2. Estado del Arte

El término de “receta” es un concepto bastante antiguo. El primer recetario del que se tiene constancia histórica se llama “De Re Coquinaria” y se remonta a la época romana y es atribuido a un gourmet romano llamado Marcus Gavius Apicius que vivió en el Siglo I después de Cristo, durante el reinado del Emperador Tiberio, aunque no se sabe con certeza y pudo ser escrito por diferentes cocineros de la época. En dicho recetario, muchas de las recetas hacían uso de un ingrediente llamado “Sifio”, que es una planta cuya resina se usaba con fines medicinales y gastronómicos y que se extinguió en el Siglo primero, así que dicho dato apoya la teoría de que, efectivamente, el recetario data de aquella época, aunque otros creen que el recetario que conservamos a día de hoy data del Siglo V después de Cristo pero, de cualquier modo, es el registro de recetario más antiguo que conservamos hasta el día de hoy.



Fig. 2 Portada original De Re Coquinaria

A partir de entonces, disponemos de numerosos documentos donde se detallan diferentes recetas de comida que nos ayudan a entender y comprender el cómo y el qué se comía en las diferentes épocas de la Historia, pasando por diferentes culturas y civilizaciones, desde el ya mencionado recetario romano, donde seguro que se usó en diferentes fiestas de los grandes señores, hasta el recetario de cocina que tenían nuestros abuelos y abuelas en casa en formato libro y ahora los diferentes recetarios de cocina que se pueden ver y buscar por internet (a veces, incluso con un vídeo demostración).

Además, gracias a los avances científicos, sobre todo en Química y Biología, tenemos mucho mayor conocimiento y comprensión de qué alimentos son más saludables y cómo procesa nuestro organismo dichos alimentos, pudiendo tener así, un mayor control sobre nuestra dieta y alimentación.

Por tanto, dada la extensa tradición culinaria que existe hoy día, por comodidad y gracias a la tecnología, podemos crear una aplicación que recopile todas estas recetas y que nos permita aplicar todos los conocimientos que tenemos actualmente sobre salud y alimentación, para así conseguir tener una dieta sana y equilibrada.

A día de hoy existen muchos tipos de dietas en función del objetivo que se quiera conseguir. Por poner algunos ejemplos, hay dietas para ganar volumen, para definir músculo, para perder peso e incluso dietas basadas en respetar el medio ambiente, como lo son las dietas veganas y vegetarianas. También se pueden tener en consideración cuestiones como la persona a la que va dirigida y diferentes problemas de salud que pueda tener como, por ejemplo, hacer dieta para celíacos, intolerantes a la lactosa,...

Teniendo en cuenta todo esto, la demanda por parte de la sociedad para tener un mayor control sobre nuestra alimentación se hace cada vez más importante y, debido a esto, disponemos en el mercado de diferentes aplicaciones como pueden ser “MyFitnessPal”, una aplicación tanto para web como para smartphone que nos permite saber qué cantidad de calorías hemos consumido a lo largo del día y llevar un control exhaustivo sobre las mismas, pero también es cierto que dicha aplicación tiene una interfaz mejorable y que no te da tanto control sobre tus recetas propias (aparte que es una aplicación que, para acceder a ciertas funciones, tienes que pagar, cuando podría ser perfectamente gratuita y la versión gratis, es una versión bastante limitada en lo que a funcionalidades se refiere).

Del mismo modo, existen otras muchas más aplicaciones como “MyFitnessPal” en el mercado tales como Hiki, Macros, Easyfit, Fitatu,... Pero todas ellas tendrían problemas similares a los ya mencionados y, además, una de las principales ventajas de crear nuestra propia aplicación es que podríamos meterle las funcionalidades que nosotros y nosotras consideremos oportunas. Como se suele decir coloquialmente y referente ámbito culinario aunque aplicable a cualquier situación similar, “esta aplicación se haría a lo Juan Palomo, yo me lo guiso, yo me lo como”.

3. Metodología usada

Para llevar a cabo este extenso proyecto y poder dividirlo en pequeñas partes, haremos uso de una máxima que se decía mucho en la antigüedad y que se sigue usando a día de hoy, que consiste en “divide y vencerás”, por lo tanto vamos a llevar a cabo una metodología Kanban. Esta metodología consiste en dividir el proyecto en pequeñas tareas sobre qué queda por hacer (con la etiqueta “to do”), qué se está haciendo (con la etiqueta “in progress”) y qué se ha hecho ya (con la etiqueta “done”) y así tener un control de en qué fase del proyecto se encuentra dicha tarea en sí y quién se está encargando de realizarla.

Esta técnica nació en Toyota y es una palabra japonesa que significa “tarjetas visuales” dado que es bastante fácil de comprobar a simple vista el estado actual del proyecto y actualmente es un modelo que está siendo aplicado bastante en la gestión y desarrollo de proyectos de software.

Por ejemplo, supongamos que tenemos que realizar un “proyecto” que consiste en hacer las tareas domésticas de la casa (por poner un ejemplo cotidiano y sencillo) y que dichas tareas las deben realizar entre cuatro integrantes de la casa: Juan, que es el padre de la familia, Ana, la madre, Álvaro, el hijo y María, la hermana de Juan. Entre los cuatro deciden organizarse usando la ya mencionada metodología Kanban, así pues para dicho ejemplo podrían dividirse las tareas en: María se encarga de hacer la colada, Juan es responsable de hacer la comida, Ana es quien friega los platos y Álvaro se va a dedicar a barrer el suelo después de comer. En un inicio todas estas tareas están en el estado inicial que “to do” (aún están por hacer), pero cuando llega la hora de la comida, Juan mueve su tarea de “hacer la comida” desde la sección de “to do” a la sección “in progress”. Cuando la comida está ya hecha, la tarea de “hacer la comida” pasa a ponerse en la sección de “done” y cuando terminan de comer, tanto Álvaro como Ana mueven sus respectivas tareas a la sección “in progress” y en ese momento, tendríamos una tarea ya realizada y acabada, en la sección “to do” (“hacer la comida”), dos tareas en proceso en la sección “in progress” (“fregar los platos” y “barrer la cocina”) y una última tarea aún por realizar en la sección “to do” (“hacer la colada”).

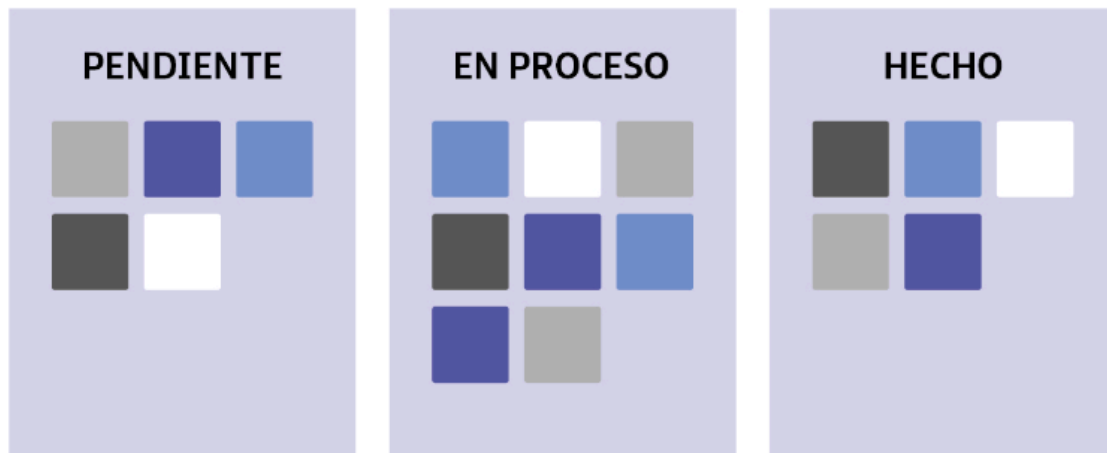


Fig. 3: Imagen de un modelo Kanban sacado del libro de Entornos de Desarrollo

Pues bien, ahora que ya hemos explicado en qué consiste a grandes rasgos la metodología que vamos a utilizar, vamos a aplicarla en nuestro proyecto, para ello vamos a separar las diferentes funcionalidades de nuestra aplicación (como el login, el registro, la visualización de recetas,...) y así, con un vistazo rápido, puedo ver qué me queda aún por hacer de mi aplicación sin temor a que se me olvide nada.

Debido a que en este proyecto voy a estar yo solo trabajando sobre el mismo, no será necesario que escriba quién es el encargado de llevar a cabo cada tarea, aunque en caso de que hubiera más integrantes en el proyecto, estaría bien ponerlo, para así evitar que una misma persona pueda llevar a cabo varias funcionalidades a la vez antes de cerrar las que ya ha empezado a hacer.

Así pues, algunas de las tareas que realizaremos, serán las siguientes:

- Creación de Base de Datos: en este paso crearemos una base de datos desde cero, creando a su vez las tablas necesarias, que en principio serán tres, pero que terminarán siendo cuatro como ya veremos más adelante y estableciendo las relaciones entre ellas haciendo uso de sus claves primarias y claves foráneas.
- Creación de los diagramas de uso: aquí estableceremos las posibles casuísticas que puedan darse a la hora de manipular nuestra aplicación, previendo así los posibles problemas futuros que puedan surgir.
- Creación de usuario con un registro a través de nuestra web: insertando dicho usuario nuevo en la tabla correspondiente de la base de datos.

- Funcionalidad de login: permitir a un usuario loguearse en nuestra página web y permitiéndole así realizar acciones en función del rol que le asignemos, siempre que su sesión permanezca activa.
- Funcionalidad de crear/modificar/eliminar recetas: hacer posible que, una vez registrados y logueados los usuarios, estos puedan crear/modificar/eliminar recetas haciendo uso de los alimentos que disponemos en nuestra base de datos.
- Creación de una web inicial que será visible para todo usuario que intente acceder a nuestra aplicación y que hará de página de inicio.
- Creación de las diferentes páginas por las que navegará el usuario: crearemos una serie de vistas concretas que solo serán accesibles por usuarios autorizados y donde puedan interactuar con nuestro modelo de base de datos en función de sus permisos.

Teniendo en cuenta todas estas funcionalidades, quedaría un diagrama Kanban del siguiente modo:

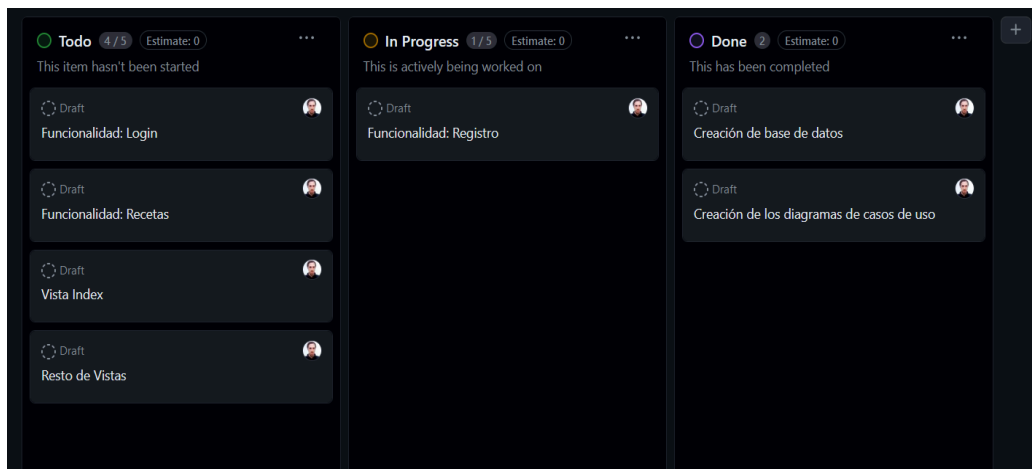


Fig. 4 Ejemplo práctico del modelo Kanban de este proyecto

Donde se puede apreciar de un momento concreto las tareas ya realizadas, las que estoy realizando y las tareas que aún tengo pendientes de hacer y donde cada tarea está asignada a mi persona (por eso sale una foto mía en la parte derecha de cada tarea).

4. Tecnologías y herramientas utilizadas en el proyecto

Las herramientas necesarias para llevar a cabo este proyecto van a ser las siguientes:

- Utilizaremos **VirtualBox** para crear una máquina virtual con Windows 11. Este paso no es estrictamente necesario, pero aprovechando que aprendimos a hacerlo en la asignatura de “Sistemas Informáticos” y teniendo en cuenta de que el usuario puede tener instalado cualquier programa en su ordenador personal que pueda generar conflicto de puertos con XAMPP (teniendo en cuenta que vamos a usar XAMPP como veremos más adelante) y a fin de evitar dichos conflictos, será mucho más sencillo crearnos una máquina virtual con un sistema operativo que controlemos (o, en mi caso, que sea cómodo de trabajar como es Windows 11) y en dicha máquina instalaremos todos los programas necesarios para llevar a cabo este proyecto. Aunque, como bien he dicho antes, esto es puramente opcional y bien se podría usar el ordenador particular sin necesidad de crear ninguna máquina virtual.
- Usaremos el paquete de **XAMPP** para poder ejecutar nuestro código PHP y poder conectarnos a la base de datos que crearemos en PHPMyAdmin. Dicho paquete trae consigo el sistema gestor de base de datos MySQL, el servidor web Apache y los intérpretes para los lenguajes de PHP y Perl. Dicho paquete aprendimos a instalarlo y a trabajar con él tanto en la asignatura de “**Despliegue de aplicaciones web**” y, paralelamente, en “Desarrollo web entorno servidor”.
- Necesitaremos un editor de texto para generar nuestros diferentes códigos y páginas webs y en mi caso me he decantado por usar **Visual Studio Code**, puesto que ofrece una mayor comodidad a la hora de trabajar con él (aunque también podría valer cualquier otro editor de texto como Notepad++, Atom,... A gusto del consumidor).
- Y por último, pero no menos importante, necesitaremos un navegador web que nos permita visualizar todo el contenido de nuestras páginas webs. En mi caso usaré **Mozilla Firefox** como mi navegador por defecto, pero también haré pruebas en Google Chrome y en Microsoft Edge para corroborar que, efectivamente, todo se ve como debería verse y que no hay problemas por visualizar nuestras webs en diferentes navegadores.

Y con respecto a las tecnologías, tenemos que hablar de los diferentes lenguajes sobre los que he decidido llevar a cabo este proyecto y por qué:

- Usaremos el patrón de diseño **Modelo-Vista-Controlador** para trabajar en este proyecto. Este patrón de diseño es uno de los más predominantes en el mundo laboral (si no es el que más, quizás junto al Modelo-Vista-Vista-Modelo) y consiste en separar la aplicación en tres partes principales:
 - El modelo es quién se encargará de todo lo referente a interactuar con la base de datos, ya sea desde una consulta a una actualización de algún dato concreto o varios datos concretos de una tabla.
 - La vista es toda aquella interfaz que el usuario podrá ver y será visible desde nuestra aplicación, ya que no queremos que los usuarios vean las tablas en sí, sino que queremos que vean la información de dichas tablas como nosotros queramos mostrársela.
 - El controlador será el encargado de responder a diferentes eventos o peticiones que haga el usuario sobre nuestro modelo. A su vez, es el encargado de modificar la vista en función de las acciones del usuario en aquellos casos en los que sea oportuno modificarla.
- Así pues, para todo lo referente a la parte de backend, usaremos **PHP**. Este lenguaje lo hemos visto en la asignatura de “**Desarrollo web entorno servidor**” y es uno de los lenguajes más extendidos en el mercado a día de hoy, nos ayudará principalmente a aplicar la lógica referente a cómo gestionar nuestra base de datos desde nuestra página web (es decir, se encargará de hacer de “controlador” en la mayoría de los casos en el patrón de diseño MVC).
- Del mismo modo, para la parte de frontend haremos uso de **HTML**, **CSS** y **JavaScript**:
 - El **HTML** es un lenguaje de marcado muy utilizado (posiblemente el que más) para la creación de páginas web. Dicho lenguaje consiste en separar por etiquetas cada una de las partes de un documento para la presentación de contenidos web y dichas etiquetas pueden ser de texto, imágenes, vídeos, audios,... De hecho, World Wide Web Consortium (conocido comúnmente como W3C), es la organización encargada de gestionar un correcto uso de dichas etiquetas para que exista un estándar y así mantener cierta homogeneidad referente a la escritura e implementación de dicho lenguaje. Y todo lo referente a este lenguaje lo vimos en el módulo, principalmente, en la

asignatura de “**Lenguaje de marcas y sistemas de gestión de la información**” (aunque también ha estado presente a lo largo de otras asignaturas, ahí fue donde más nos centramos en este lenguaje).

- El lenguaje **CSS** es un lenguaje de diseño gráfico que nos permite dar forma, presentación y una visualización más agradable de las diferentes páginas creadas con HTML. Dicho lenguaje de diseño se nos fue impartido en la asignatura de “**Diseño de interfaces**” y gracias a este mismo lenguaje, podemos hacer más atractivas nuestras páginas webs haciendo más probable que el usuario prefiera seguir usando nuestra aplicación web, antes que irse a cualquier otra o que termine con una experiencia indeseable de nuestra aplicación y, por lo tanto, es un aspecto muy importante a tener en cuenta y a cuidar. Respecto a esta tecnología cabe destacar que me he apoyado ligeramente en Bootstrap solo con el fin de querer hacer más atractiva la vista principal de la aplicación, pero sin entrar demasiado en detalle ni abusar del uso de la misma.
- Y, por último, el lenguaje **JavaScript**, es el lenguaje de programación interpretado del lado del cliente por excelencia (al menos, hasta día de hoy), permitiéndonos crear páginas webs dinámicas y mejorando mucho la experiencia de usuario. En la asignatura de “**Desarrollo web entorno cliente**” fue donde entramos en profundidad en este lenguaje y aprendimos a poder modificar nuestras páginas HTML a través de eventos que interactúen con el usuario, a la vez que trabajar con objetos y diferentes tipos de datos siempre y cuando nos fuera necesario.
- Y, respecto a la base de datos, usaremos lenguaje **MySQL**, lenguaje con el que trabajamos en la asignatura de “**Bases de Datos A**” principalmente (aunque, tangencialmente, lo vimos también en otras asignaturas, cuando nos fue necesario para trabajar con bases de datos). Dicho lenguaje es un sistema de gestión de bases de datos relacional que viene implementado en el paquete XAMPP y será donde almacenemos todos los datos referentes a nuestra página web, desde los diferentes ingredientes que podrán ser usados en las recetas hasta las tablas de usuarios que tendremos registrados en nuestra aplicación.

5. Planificación, Diagnóstico y Contexto Laboral

La planificación de este proyecto ha sido clave a la hora de poder organizar los tiempos de un modo estimado y así poder hacer frente a diferentes imprevistos.

Como bien comenté en la sección de “Metodología usada”, he aplicado un modelo de Kanban para así poder separar todo el proyecto en pequeñas tareas y centrándome en una hasta acabarla antes de pasar a la siguiente tarea.

Así pues, mi proyecto se ha podido dividir en las siguientes fases:

- **Fase de planteamiento de la idea del proyecto.** En esta fase que comenzó el 21 de Febrero, empecé a idear este proyecto con la idea de presentarlo como proyecto de final de ciclo formativo superior, teniendo en mente varias ideas alternativas (por si alguna era rechazada) y esta fase abarcó hasta el 3 de Marzo, que fue cuando se me dio el apto para el proyecto.
- **Fase de preparación de la base de datos.** En esta fase creamos las diferentes tablas de la base de datos y creamos las diferentes dependencias entre ellas, estableciendo así el diagrama Entidad-Relación de toda nuestra Base de Datos.
- **Fase de preparación de la memoria escrita para el feedback.** En esta fase empezamos a elaborar la memoria escrita para así poder llevar a cabo la entrega en el plazo establecido para el feedback. Esta entrega se llevó a cabo el 17 de marzo y contenía un pequeño adelanto de la memoria escrita para así poder comprobar que cumplía con las especificaciones y normativa establecidas.
- **Fase de elaboración del código.** En esta fase (que es la fase que más tiempo sin lugar a dudas me ha llevado) he preparado el código para satisfacer todas las necesidades del proyecto, tomando decisiones sobre cómo hacer las diferentes funcionalidades y buscando información a la vez que aprendiendo a cómo hacer el código de una manera eficiente y con un resultado que me satisficiera, refactorizando en más de una ocasión el código y llevando a cabo las modificaciones pertinentes.
- **Fase de pruebas.** Esta fase ha abarcado prácticamente lo mismo que la fase de elaboración del código, dado que siempre que se implementaba una nueva funcionalidad, tenía que comprobarse seguidamente que dicha funcionalidad funcionaba correctamente y esta fase siempre se iba intercalando entre elaboración de código y elaboración de código.

- **Fase de elaboración de la memoria escrita.** Una vez finalizado el código y corroborado que funciona la aplicación como se desea, se ha procedido a terminar de elaborar la memoria escrita, modificando la que previamente se había entregado en el feedback y añadiendo todo lo restante hasta acabarla.
- **Fase de elaboración del PowerPoint y del vídeo presentación.** La última fase del proyecto ha consistido en elaborar el PowerPoint para realizar la presentación y grabar el vídeo para su posterior entrega.

En un principio, todas estas fases tuvieron unos tiempos establecidos a modo de previsión pero, como suelen ocurrir en estos casos, han surgido contratiempos que han hecho que ciertas fases se hayan visto prolongadas en el tiempo mientras que otras, por suerte, se han podido reducir el tiempo estimado con respecto a lo que se pensaba que iba a tomar terminar dicha fase.

En el siguiente **diagrama de Gantt** puede observarse la evolución del proyecto:

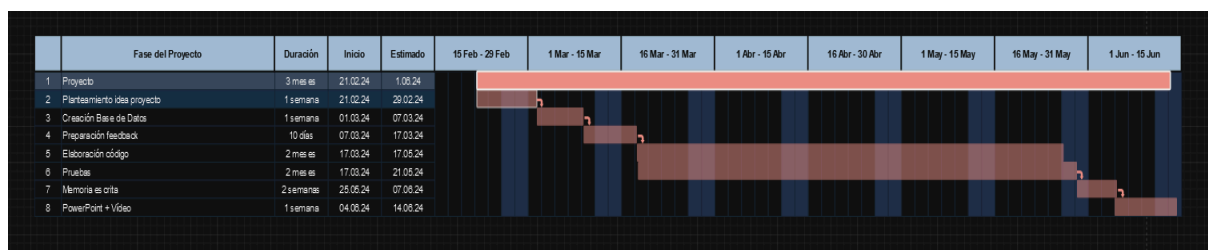


Diagrama de Gantt del proyecto (ver Anexos)

Cada una de las fases ha sido precedida por su fase posterior (exceptuando el “proyecto global” que ha abarcado todo y la fase de código + fase de pruebas que han sido prácticamente simultáneas como se ha mencionado antes). Y respecto a los tiempos, si bien ha habido diferencia entre lo “Estimado” y la duración real, se ha podido asemejar bastante al resultado final, siendo útil de cara a la de organizar los tiempos del proyecto.

Y además de tener en cuenta nuestros tiempos estipulados en nuestro diagrama de Gantt, también hemos llevado a cabo un análisis **DAFO** de nuestro proyecto para así estudiar las diferentes problemáticas y virtudes a las que se puede enfrentar este proyecto, teniéndose en cuenta los factores internos (las debilidades y fortalezas) y los factores externos (las amenazas y oportunidades) pudiendo clasificarlo todo del siguiente modo.

- **Debilidades:** La principal debilidad de nuestro proyecto es, como cabe esperar, la inexperiencia a la hora de trabajar y llevar a cabo un proyecto grande desarrollado completamente por una única persona, sin apoyo externo (sin haber tirado de librerías externas aparte de la ya mencionada Bootstrap cuyo uso ha sido muy puntual) y con el tiempo tan limitado que se ha tenido. Esto, obviamente, ha supuesto un problema en el resultado final del proyecto como, por ejemplo, que la interfaz sea un html bastante básico y un css bastante limitado también (haciendo que el resultado de las vistas se antoje relativamente arcaico en comparación con las aplicaciones web que se destilan hoy en día).
- **Amenazas:** También debemos tener en cuenta las amenazas a las que se enfrentará nuestro proyecto como, por ejemplo, la competitividad de otras aplicaciones en el mercado que están mejor desarrolladas (la mayoría habrán sido desarrolladas por un equipo de varias personas y sin limitaciones de tiempo) y cuyas marcas están ya asentadas en el mercado actual.
- **Fortalezas:** Como puntos positivos tenemos las fortalezas de nuestro proyecto que son, por ejemplo, el código que está bastante bien ejecutado, se han tenido en cuenta diversas problemáticas que se pudieran generar por parte de los usuarios y que, al estar el servidor separado completamente del cliente, se ofrece una mayor seguridad permitiendo, por ejemplo, llevar a cabo validaciones tanto en cliente como en servidor y proporcionando así una mayor seguridad frente a posibles intrusiones externas (aunque, como siempre, este aspecto se podría mejorar aún más, pero actualmente considero que está bastante cuidado este aspecto) o que, al ser un proyecto desarrollado por nosotros, podemos tener un control completo sobre nuestras funcionalidades en función de nuestras necesidades, haciendo que esta sea una aplicación completamente personalizable y amoldable según nuestros criterios.
- **Oportunidades:** Por último, se puede hablar de las oportunidades de nuestro proyecto entre las que podemos destacar que apenas hay aplicaciones web gratuitas que puedan ofrecer (o terminar ofreciendo) tantas funcionalidades y, como bien se ha comentado antes, podemos tener un control sobre las funcionalidades en función del feedback que podamos recibir de nuestros usuarios que, al ser en un principio pocos, podrán recibir un trato más personalizado de la aplicación haciendo que esta sea más versátil y optimizable.

6. Análisis del proyecto

En este apartado exploraremos las necesidades y finalidad del proyecto, exponiendo las diferentes casuísticas por parte de diferentes tipos de usuario que quieran hacer uso de nuestra aplicación. Para ello, vamos a empezar a separar y enumerar los diferentes requisitos funcionales y no funcionales de nuestra aplicación.

Requisitos funcionales:

1. Registro de usuarios: los usuarios deberán poder registrarse en nuestra aplicación web creando una cuenta con su nombre de usuario, su correo y su contraseña. Dicho correo deberá ser único en la aplicación, haciendo que un usuario no se pueda registrar dos veces con el mismo correo.
2. Login: una vez que el usuario haya sido registrado, deberá poder loguearse con sus credenciales de nombre de usuario y contraseña de un modo seguro.
3. Creación de recetas: una vez logueados los usuarios, estos deberían poder crear nuevas recetas haciendo uso de los ingredientes que tengamos en nuestra base de datos, añadiendo diferentes cantidades de cada ingrediente y siendo dicha receta de carácter personal para cada usuario (es decir, cada usuario tendrá sus propias recetas).
4. Modificación de recetas: una vez que el usuario está logueado en el sistema y tenga creadas sus recetas, este podrá modificarla siempre que quiera, añadiendo nuevos ingredientes, eliminando otros o modificando las cantidades necesarias de los mismos.
5. Eliminación de recetas: si así lo desea el usuario, este podrá eliminar una receta que haya creado previamente.
6. Gestión de las comidas en el calendario: con el usuario ya logueado y con sus recetas ya creadas, se le permitirá introducir sus propias recetas en su calendario para así poder organizar las comidas de la semana.
7. Cálculo de los valores nutricionales semanales, diarios y por comidas: una vez que el usuario haya creado su propio calendario, se podrá calcular los valores nutricionales del mismo.
8. Logout: una vez que el usuario haya terminado de trabajar con su sesión lo que quiera, podrá desloguearse de la aplicación, manteniendo así su privacidad y protección de sus datos frente a otros posibles usuarios que usen posteriormente el mismo dispositivo.

Requisitos no funcionales:

1. Disponibilidad: La aplicación web debe estar disponible en todo momento, pudiendo llevarse a cabo labores de mantenimiento de manera puntual y siempre intentando que la experiencia del usuario se vea afectada en la menor medida posible (aunque lo deseable sea que no se vea afectada en absoluto).
2. Compatibilidad: La aplicación web debe ser compatible con todos los navegadores webs del mercado, viendo que nuestra aplicación no se vea afectada si el usuario decide usar un navegador web distinto al nuestro.
3. Escalabilidad: Nuestra aplicación web deberá poder soportar un aumento en el número de usuarios que tengamos en nuestro sistema sin que esto afecte al rendimiento del mismo.
4. Usabilidad: La interfaz de la aplicación web deberá ser intuitiva y fácil de manejar para el usuario, evitando cargarla demasiado con elementos que puedan molestar e incordiar para un correcto uso de dicha aplicación.
5. Accesibilidad: La aplicación web deberá, en la medida de lo posible, tener en consideración que dicha aplicación podría ser usada por personas con algún tipo de discapacidad y se velará porque la experiencia de dichos usuarios sea lo más cercana posible a la experiencia del resto de usuarios, cumpliendo así con las normas WCAG.
6. Navegabilidad: A la hora de navegar por nuestra aplicación web, hay que conseguir que todos los sitios de relevancia para los usuarios sean accesibles a través de pocos clics a enlaces, evitando engorronar la aplicación con demasiadas páginas web que puedan resultar irrelevantes para el usuario y siempre permitiéndole volver a la página principal cuando así lo desee.
7. Seguridad: Nuestra aplicación web debe ser segura, blindándose frente a posibles ataques externos de personas que quieran hacer un uso malicioso de dicha aplicación web, protegiendo así, la privacidad de los datos proporcionados por nuestros usuarios, así como se estipula en la Ley Orgánica de Protección de Datos Personales y Garantía de los Derechos Digitales (LOPDGDD cuya entrada en rigor fue el 07/12/2018, con enlace en la bibliografía).
8. Mantenibilidad: El código fuente de nuestra aplicación web debe estar bien documentado facilitando así su mantenimiento y posibles actualizaciones futuras.

Con todo esto en mente, pasaremos a establecer los diagramas de casos de uso de los posibles usuarios que quieran utilizar nuestra aplicación.

Caso 1: Un usuario nuevo llega a nuestro sistema. En este caso solo podrá o registrarse (puesto que es posible que no esté registrado aún en nuestro sistema) o loguearse, en caso de tener ya una cuenta con nosotros.

Una vez se loguee, tendrá acceso a un calendario que en principio se mostrará vacío y una serie de botones que, en caso de clicar en ellos, se le desplegarán diferentes ventanas (concretamente elementos <dialog>) que le permitirán llevar a cabo diferentes acciones o que dichos botones simplemente hagan algún cambio en el calendario (como actualizar las recetas del desplegable de sus recetas o calcular los valores nutricionales del mismo) .

Los diferentes botones son los siguientes:

- **Add recipe:** botón que le desplegará una ventana donde podrá crear su receta asignándole un nombre a susodicha receta, añadiendo nuevos ingredientes o eliminando ingredientes (hasta un mínimo de un ingrediente por receta) y estableciendo una cantidad concreta de dicho ingrediente (en g).
- **Show recipes:** botón que le desplegará una ventana donde podrá ver todas sus recetas creadas hasta el momento y en cada receta habrá dos botones adicionales, el primero para editar la receta concreta, pudiendo modificar los valores de dicha receta e incluso añadir o eliminar ingredientes (hasta un mínimo de uno) y el segundo botón que le servirá para eliminar la receta completamente de su lista de recetas.
- **Refresh calendar:** botón que le permitirá actualizar las recetas de su calendario una vez que se hayan añadido/modificado o eliminado las recetas concretas.
- **Calculate nutritional values:** botón que le mostrará en el calendario los valores nutricionales totales semanales, por día o por comida.
- **Save calendar:** botón que le permitirá guardar un calendario concreto en la base de datos.
- **Close session:** botón que cerrará la sesión del usuario y le mandará de vuelta a la vista de login.

Así pues, quedaría un diagrama de casos de uso tal que así:

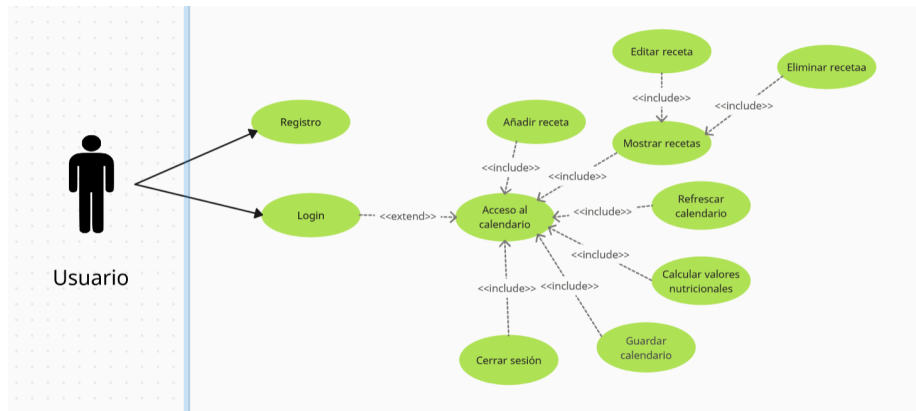


Diagrama casos de uso para un usuario que llega al sistema (ver Anexos)

Caso 2: Un administrador accede al sistema a través del login y, una vez lo haga, tendrá una lista de todos los usuarios donde podrá banear a cualquier usuario que desee, ya sea por mala conducta o por un uso malintencionado de la aplicación.

En un principio se le iban a añadir más funcionalidades a este usuario administrador pero, finalmente, se terminó concluyendo que carecía de sentido algunas funcionalidades que se habían pensado en un principio.

También cabe destacar que la vista de administrador será completamente independiente de la vista de usuario, haciendo que ningún usuario pueda acceder a ella de manera malintencionada, comprobándose en todo momento que el usuario que está intentando acceder a esta vista se trata de, efectivamente, un usuario con el rol de “superadmin” y no con ningún otro rol de usuario posible o se le llevará a otra vista distinta donde solo podrá volver hacia atrás para acceder a la vista del login.

Este es un aspecto de seguridad bastante importante y vital para un correcto funcionamiento de la aplicación.

Si tenemos en cuenta todo esto, obtendremos un diagrama de casos de uso del siguiente modo:

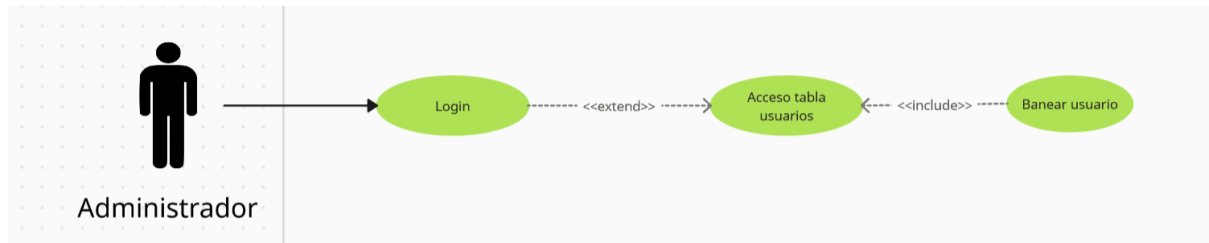


Diagrama de casos de uso para un Administrador (ver anexos)

Y, por último, con respecto al tema de la Base de Datos relacional, tendremos las siguientes tablas con sus respectivos campos (recordemos que con PK nos referimos a las claves primarias de la tabla y que con FK nos referiremos a las claves foráneas de las mismas tablas):

- Tabla users:
 - user_id (PK)
 - username
 - email
 - password_user
 - rol_user
- Tabla ingredients:
 - ingredient_id (PK)
 - ingredient
 - kcal
 - proteins
 - cholesterol
- Tabla recipes:
 - recipe_id (PK)
 - user_id_fk (FK)
 - name_recipe
 - total_kcal
 - total_proteins
 - total_fat
 - total_carbohydrates

- Tabla recipe_ingredients:
 - ingredient_id_fk (PK)
 - recipe_id_fk (PK)
 - quantity

- Tabla calendar:
 - calendar_id (PK)
 - user_id_fk (FK)
 - monday_breakfast_recipe_fk (FK)
 - monday_lunch_recipe_fk (FK)
 - monday_dinner_recipe_fk (FK)
 - tuesday_breakfast_recipe_fk (FK)
 - tuesday_lunch_recipe_fk (FK)
 - tuesday_dinner_recipe_fk (FK)
 - wednesday_breakfast_recipe_fk (FK)
 - wednesday_lunch_recipe_fk (FK)
 - wednesday_dinner_recipe_fk (FK)
 - thursday_breakfast_recipe_fk (FK)
 - thursday_lunch_recipe_fk (FK)
 - thursday_dinner_recipe_fk (FK)
 - friday_breakfast_recipe_fk (FK)
 - friday_lunch_recipe_fk (FK)
 - friday_dinner_recipe_fk (FK)
 - saturday_breakfast_recipe_fk (FK)
 - saturday_lunch_recipe_fk (FK)
 - saturday_dinner_recipe_fk (FK)
 - sunday_breakfast_recipe_fk (FK)
 - sunday_lunch_recipe_fk (FK)
 - sunday_dinner_recipe_fk (FK)

Relaciones entre las tablas:

- Un usuario tendrá un único calendario y cada calendario pertenece a un único usuario. Relación 1:1.
- Un usuario puede tener una o varias recetas pero toda receta pertenecerá siempre a un único usuario. Relación 1:N.
- Cada comida del calendario tendrá siempre una única receta (una comida en cada almuerzo/cena del día), pero una receta puede estar en varias comidas del calendario. Relación 1:N.
- Cada ingrediente puede pertenecer a más de una receta y cada receta puede tener varios ingredientes. Relación N:M.

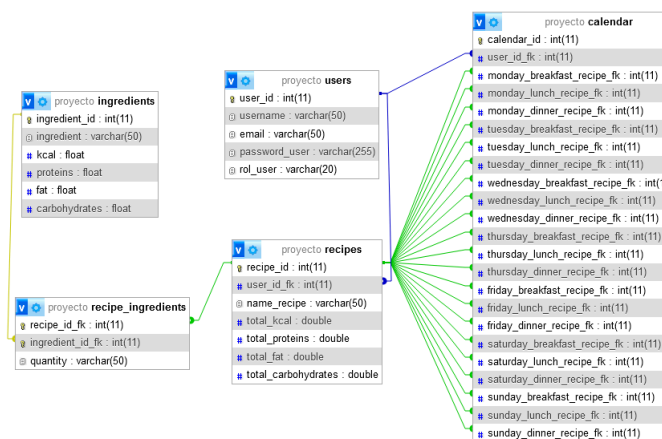


Diagrama Entidad-Relación (ver anexos).

Cabe destacar que en dicho diagrama hemos necesitado una tabla adicional intermedia a la que hemos llamado “recipe_ingredients”. Esto se debe a que la relación entre ambas es N:M y, por lo tanto, hay que crear una tabla intermedia cuya clave primaria sea una clave conjunta de las dos claves foráneas de ambas tablas (ingredient_id_fk y recipe_id_fk).

Una vez enseñado los diagramas de casos de uso y el Diagrama Entidad-Relación, sólo nos quedaría por exponer el Diagrama de clases.

Para este proyecto no he usado demasiadas clases, porque no he tenido una necesidad real para hacerlo, pero sí es cierto que he usado dos clases en JavaScript: la clase Recipe y la clase ListRecipes.

- **Recipe:**

- Atributos:

- idRecipe (atributo de tipo number): atributo que tomará el valor id_recipe de la tabla recipes de la base de datos.
 - nameRecipe (atributo de tipo string): atributo que determinará el nombre de la receta.
 - numberIngredients (atributo de tipo number): atributo que determina el número de ingredientes que tiene una receta.
 - ingredients (atributo de tipo array): atributo que consiste en un array de json con todos los ingredientes que pertenecen a esa receta y sus respectivas cantidades.

- Métodos:

- constructor(): Método constructor para los objetos de tipo Recipe.
 - setIdRecipe(idRecipe): Método setter para el atributo idRecipe.
 - setNameRecipe(nameRecipe): Método setter para el atributo nameRecipe.
 - getNumberIngredients(): Método getter para el atributo numberIngredients.
 - getRecipeIngredients(): Método getter para el atributo ingredients.
 - increaseNumberIngredients(): Método que incrementa en uno el número de ingredientes de la receta.
 - decreaseNumberIngredients(): Método que disminuye en uno el número de ingredientes de la receta.
 - pushIngredientsRecipe(ingredient): Método que añade el ingrediente al array de ingredients.

- **ListRecipes:**

- Atributos:

- recipes (atributo de tipo array): atributo que consiste en un array con todas las recetas para el usuario que tenga iniciada su sesión.
 - numberRecipes (atributo de tipo number): atributo que determina la cantidad de recetas que pertenecen a dicho usuario.

- Métodos:
 - constructor(): Método constructor para los objetos de tipo ListRecipes.
 - getRecipes(): Método getter para el atributo recipes.
 - getNumberRecipes(): Método getter para el atributo numberRecipes.
 - increaseNumberRecipes(): Método que incrementa en uno el número de recetas de la lista.
 - pushRecipes(recipe): Método que añade la receta al array de recetas.

Cabe destacar que los objetos de tipo ListRecipes están compuestos a su vez por objetos de tipo Recipe, haciendo uso de la “composición” de la programación orientada a objetos (una lista de recetas está compuesta por las recetas y una lista de recetas sin recetas carece de sentido).

Y, por lo tanto, el diagrama de clases resultante que nos quedaría sería el siguiente:

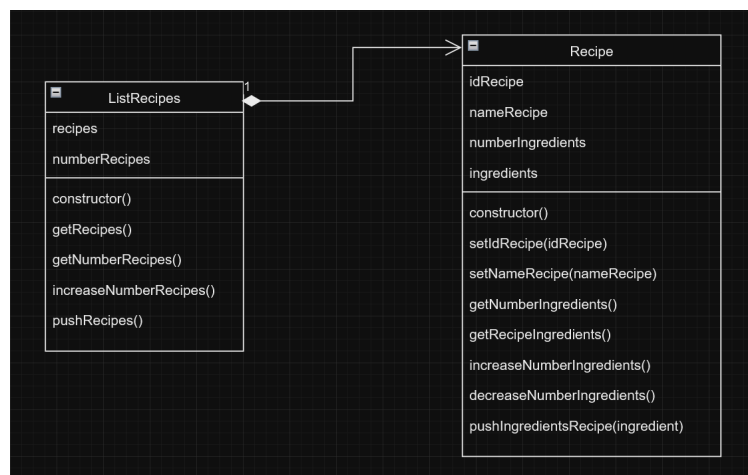


Diagrama de clases (ver anexos)

Con todo esto establecido ya, procederemos en el siguiente apartado de la memoria a explicar cómo ha ido el diseño del proyecto, como he decidido diseñarlo y por qué he decidido diseñarlo de ese modo y no de ningún otro.

7. Diseño del proyecto

Partiendo de la base, como ya se ha mencionado previamente (más concretamente en el apartado 4), vamos a utilizar como patrón de diseño el Modelo-Vista-Controlador, donde el **Modelo** se va a encargar de todo lo referente a la Base de Datos, la **Vista** va a ser todo lo que se va a mostrar al usuario y el **Controlador** va a ser el encargado de añadir interactividad a las vistas y de hacer capa intermedia entre la vista y el modelo y, paralelamente, como también se ha mencionado en el apartado de objetivos, me he centrado en querer separar completamente la parte del frontend de la parte del backend, haciendo que en la parte del backend pueda estar solo el PHP y la base de datos y en la parte del frontend esté solo el HTML, CSS y JavaScript, es decir, me he negado rotundamente a mezclar el HTML con PHP, por ejemplo, haciendo que en mis archivos con extensión .php solo haya código PHP, en mis archivos .js solo haya código de JavaScript, que en mis archivos .css solo haya lenguaje CSS y que en mis archivos .html solo haya código HTML (aunque en este último caso sí que hay alguna funcionalidad de javascript incrustada en el html como, por ejemplo, eventos onClick() de algunos botones, pero dicha funcionalidad del onClick ha sido definida en su archivo .js correspondiente y no en el propio html). Con todo esto quiero decir que he conseguido modularizar y separar lo máximo posible todos y cada uno de los lenguajes, quedando una estructura de archivos como muestro a continuación:

- Carpeta **Controller**:
 - **addRecipe.php**: este archivo php es el que va a controlar las peticiones de los usuarios a la hora de querer añadir una receta en la parte del servidor.
 - **banUser.php**: este archivo php es el que va a controlar las peticiones del superadmin a la hora de querer banear a un usuario en la parte del servidor.
 - **checkSession.php**: este archivo php es el que va a controlar si realmente hay una sesión de usuario activa en el lado del servidor y qué tipo de usuario es quién tiene dicha sesión activa.
 - **closeSession.php**: este archivo php es el que va a controlar el cierre de las sesiones en el lado del servidor para los usuarios.
 - **deleteRecipe.php**: este archivo php es el que va a controlar las peticiones de los usuarios a la hora de querer borrar una receta en la parte del servidor.
 - **editRecipe.php**: este archivo php es el que va a controlar las peticiones de los usuarios a la hora de querer editar una receta en la parte del servidor.

- **getIngredients.php:** este archivo php será el encargado de devolver al frontend los ingredientes de la base de datos.
- **getRecipes.php:** este archivo php será el encargado de devolver al frontend las recetas de la base de datos del usuario que tenga iniciada su sesión.
- **getRecipesValues.php:** este archivo php será el encargado de devolver al frontend los valores nutricionales de las recetas de la base de datos del usuario que tenga iniciada su sesión.
- **getUsers.php:** este archivo php será el encargado de devolver al frontend los usuarios de la base de datos.
- **login.js:** este archivo JavaScript será el encargado de establecer todas las funcionalidades de la vista index.html (que será la vista inicial de nuestra aplicación, que constará de un login, por eso este nombre de archivo) y mandará las peticiones necesarias a su archivo php correspondiente.
- **login.php:** este archivo php será el encargado de validar los datos proporcionados por el frontend y de devolver al frontend un “ok” si el usuario ha podido loguearse correctamente o un error si ha habido algún problema con las credenciales.
- **recipes.js:** este archivo JavaScript será el encargado de establecer todas las funcionalidades referentes a las recetas de la vista userView.html, mandando las peticiones a los archivos php correspondientes y modificando la vista para mostrar los resultados pertinentes.
- **register.js:** este archivo JavaScript será el encargado de establecer todas las funcionalidades referentes al registro, validando que los datos introducidos son correctos y mandando la petición al archivo php correspondiente en caso de que todo sea según lo esperado.
- **register.php:** este archivo php será el encargado de validar los datos proporcionados por el frontend, para que no haya posibles errores en los datos introducidos y devolver al frontend un “ok” si el usuario ha podido registrarse correctamente o un error si ha habido algún problema con los datos de contacto proporcionados.
- **saveCalendar.php:** este archivo php será el encargado de devolver al frontend un “ok” si el usuario ha podido guardar su calendario correctamente en la base de datos o un error si ha habido algún problema con las credenciales.

- **superadmin.js:** este archivo JavaScript será el encargado de establecer todas las funcionalidades referentes a la vista del superadmin, validando que efectivamente el usuario que está intentando acceder tenga el rol de superadmin y proporcionándole la funcionalidad de banear a algún usuario si así lo desea.
- **userView.js:** este archivo JavaScript será el encargado de establecer todas las funcionalidades referentes a la vista del usuario una vez que este se ha logueado con éxito, validando que efectivamente el usuario se ha logueado previamente y ofreciendo las diferentes funcionalidades para que el usuario pueda mandar peticiones a diferentes archivos php para poder gestionar su sesión en el lado del servidor desde el frontend.
- Carpeta **Model:**
 - **connection.php:** este archivo php contiene todo lo referente a la conexión con la base de datos de PHPMyAdmin, permitiendo abrir una conexión o cerrándola en los casos que sea necesario.
 - **proyecto.sql:** este archivo sql contiene todo lo referente a como están montadas las bases de datos y donde se llegarán todas las peticiones después de haber pasado por todas las diferentes validaciones, tanto de los archivos php, como de los archivos JavaScript que pudieran haber.
 - **recipesFunctions.php:** este archivo php contiene todas las funciones referentes a la gestión de las recetas en la base de datos, realizando las diferentes sentencias SQL (con prepared statements, para evitar inyecciones de SQL) y ejecutando dichas consultas cuando se haya verificado que todo está correcto y devolviendo en cada caso los valores pertinentes (generalmente un true si todo ha ido bien o una variable con los valores de la respuesta de la base de datos).
 - **usersFunctions.php:** este archivo php contiene todas las funciones referentes a la gestión de los usuarios en la base de datos, realizando las diferentes sentencias SQL (con prepared statements, para evitar inyecciones de SQL) y ejecutando dichas consultas cuando se haya verificado que todo está correcto y devolviendo en cada caso los valores pertinentes (generalmente un true si todo ha ido bien o una variable con los valores de la respuesta de la base de datos).

- Carpeta **View:**

- **index.css:** este archivo css contiene todos los estilos para la vista index.html, determinando el color de fondo, la posición de los diferentes elementos html, etcétera.
- **index.html:** este archivo html contiene todas las etiquetas html y su estructura para la vista inicial de la aplicación que consistirá en un login con un par de campos para que el usuario pueda acceder así a nuestro sistema en función del rol que tenga.
- **register.css:** este archivo css contiene todos los estilos para la vista register.html (del mismo modo que index.css contiene todos los estilos para index.html), determinando el color de fondo, la posición de los diferentes elementos html, etcétera.
- **register.html:** este archivo html contiene todas las etiquetas html y su estructura para la vista de la aplicación que mostrará a los usuarios unos campos que estos podrán rellenar para registrarse en nuestro sistema.
- **superadmin.css:** este archivo css contiene todos los estilos para la vista superadmin.html, determinando el color de fondo, la posición de los diferentes elementos html, etcétera.
- **superadmin.html:** este archivo html contiene todas las etiquetas html y su estructura para la vista de la aplicación que consistirá en una tabla con todos los usuarios registrados hasta este momento y con botones que permitan al superadmin banear al usuario que elija.
- **userBanned.html:** este archivo html contiene todas las etiquetas html y su estructura para la vista de la aplicación que se mostrará a todos aquellos usuarios que han sido baneados de la aplicación, impidiéndoles así acceder a la aplicación.
- **userView.css:** este archivo css, junto a los diferentes apoyos que se obtienen gracias a usar Bootstrap, contiene todos los estilos para la vista userView.html, determinando el color de fondo, la posición de los diferentes elementos html, etcétera. Cabe destacar que al ser esta potencialmente la vista más importante de la aplicación, es en este archivo css donde más se ha intentado hacer hincapié en establecer unos estilos más agradables visualmente hablando.

- **userView.html:** este archivo html contiene todas las etiquetas html y su estructura para la vista de la aplicación que consistirá en la vista más importante de la aplicación, donde el usuario podrá ver un calendario y, en dicho calendario, podrá elegir mediante un desplegable la receta que más guste y con diferentes botones para diferentes funcionalidades (que pueden ir desde “cerrar sesión” a que se les despliegue un modal para que puedan introducir/modificar/eliminar sus recetas).
- **userViewRejected.html:** este archivo html contiene todas las etiquetas html y su estructura para la vista de la aplicación que se mostrará a todos aquellos usuarios que intenten acceder a la aplicación sin haberse logueado como es debido o después de haber cerrado su sesión (para evitar problemas de que una tercera persona pueda acceder a sus datos sin su permiso).

De todos estos archivos, cabe destacar que el único archivo .html que contiene alguna funcionalidad de Bootstrap es el archivo “userView.html” y se tomó esta decisión porque, al ser la vista en la que el usuario presuntamente iba a invertir más tiempo en su uso de la aplicación, se optó por intentar ofrecerle una experiencia de usuario más agradable haciendo que, por ejemplo, todas las funcionalidades referentes a la propia vista y evitando que el usuario tuviera que navegar por diferentes vistas en función de lo que quisiera hacer (por ejemplo, si el usuario quiere añadir una receta, con clicar en el botón “addRecipe”, se le abriría un modal con respecto a los datos para que pueda añadir una receta... Sin necesidad de moverse a otra vista que le ofreciera esta opción, evitando así recargas innecesarias por parte del navegador). Por tanto, con el fin de la comodidad para el usuario, se optó por introducir Bootstrap en el proyecto, aunque sea de un modo muy tangencial (y sin abusar de las funcionalidades que ofrece Bootstrap de por sí).

Ahora que ya hemos expuesto la estructura de los archivos de nuestro proyecto, como se puede observar, en la vista solo hay archivos html y css, en el controlador solo hay archivos js y php y en el modelo solo hay archivos php y sql y lo verdaderamente interesante de todo esto es la dinámica que siguen los diferentes archivos.

Durante el módulo se nos ha enseñado a trabajar de manera independiente el cliente y el servidor, pero algo que siempre me he planteado era el como ocurría dicha comunicación cliente-servidor que sabía que en el mundo laboral se daba (ya que las diversas aplicaciones que hay en el mercado, raro es que usen única y exclusivamente un lenguaje de programación) e investigando mucho, pude encontrar una función nativa en JavaScript

(al menos, desde el año 2015-2017, dependiendo del navegador, pero que a día de hoy, está estandarizado) que es la función `fetch()` (cuya documentación se encuentra en la sección de webgrafía) y, en esencia, consiste en mandar desde JavaScript una petición al servidor y devolviendo una promesa en función de la respuesta recibida, así que me cree un par de funciones (una para peticiones post y otra para peticiones get) que mandaran cierta información (o no) a un archivo php concreto, este archivo php ejecutara su script y devolviera una respuesta en función de si todo había ido bien o no y dicha respuesta de vuelta la gestiona luego JavaScript para mostrarle al usuario la información pertinente.

Por tanto, el flujo de información sería:

1. Se carga el html y el css para mostrárselo al usuario.
2. El usuario introduce datos que JavaScript recogerá y enviará al controlador de PHP en formato de JSON.
3. El archivo php de la carpeta controller recibirá los datos en formato JSON (en el posible caso de que existieran dichos datos), los descodificará y llamará a alguna función que se encuentra en algún archivo php concreto de la carpeta model.
4. Dicha función intentará ejecutar una sentencia SQL en nuestra base de datos con los datos proporcionados.
5. Si todo ha ido bien, se le dirá al archivo php de la carpeta controller que mande una respuesta con un 200 (que es un “ok”) y, si no ha ido bien, se mandará una respuesta con un 400 (que es un “bad request”) y codificando los posibles datos de vuelta en formato JSON.
6. El 200 o el 400 y los potenciales datos en formato JSON llegarán de vuelta al archivo JavaScript correspondiente y, dependiendo de este valor modificará el html y/o css de la vista para que este pueda observar los cambios.

Todo esto equivaldría a seguir el siguiente esquema:

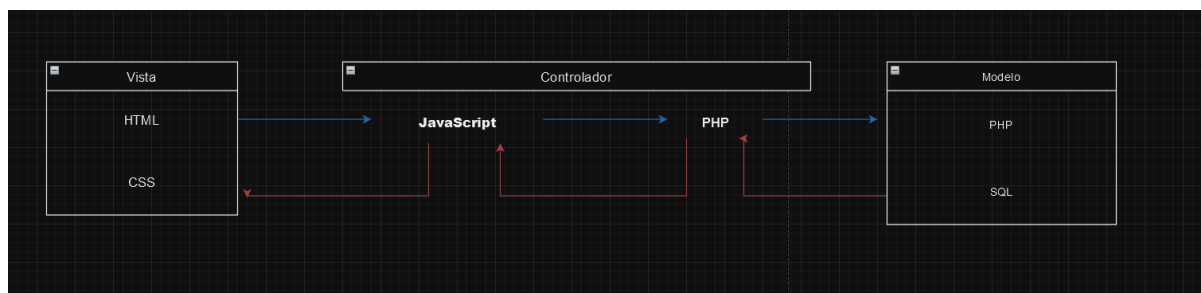


Diagrama de flujo de la aplicación (ver anexos)

Dishes At Will (DAW) - Juan José Saborido Barranco

Donde las líneas azules equivalen a las solicitudes y las líneas rojas equivaldrían a las respuestas.

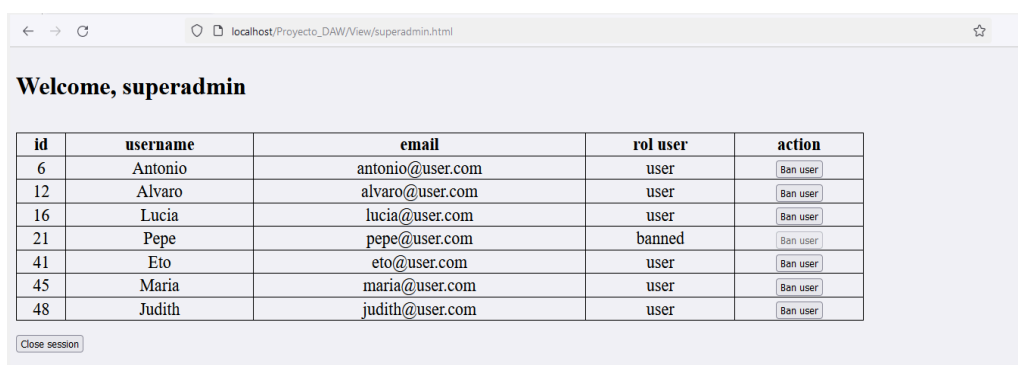
Y, respecto a las ya mencionadas respuestas, solo comentar que se han elegido la respuesta 200 y la respuesta 400 por convenio y generalizar, pero que realmente podría haberse usado diferentes tipos de código de respuesta en función de la situación (por ejemplo, en vez de devolver un 400, podría haberse devuelto un 401, que significa “unauthorized” o incluso un 418 que significa “i’m a teapot”, pero realmente en todos los casos, habría un error en la petición fetch y la promesa no se cumpliría igualmente). Estos valores salen de los códigos de respuesta de las peticiones http que también se encuentra un enlace en la bibliografía.

Una vez explicado todo el funcionamiento general de nuestra aplicación respecto a cómo se comunican el backend y el frontend, procederemos a explicar el diseño de cada vista.

De las vistas de userBanned.html y userViewRejected.html hay poca cosa que comentar, simplemente son vistas sencillas donde se le ofrece al usuario un enlace para que vuelva a la vista de index.html (con el fin de fomentar la navegabilidad de la web evitando llegar a puntos muertos dentro de la aplicación).

De las vistas index.html y register.html simplemente se le ha aplicado un color de fondo ligeramente gris (para evitar que fuera blanco puro) y se han centrado sus elementos para así dar una sensación de “equilibrio” de dicha vista y donde desde index.html se puede acceder a la vista register.html y viceversa.

Ahora bien, hablando de la vista de superadmin.html, podremos ver la siguiente vista:



The screenshot shows a web browser window with the address bar displaying 'localhost/Proyecto_DAW/View/superadmin.html'. The page content includes a heading 'Welcome, superadmin' and a table with the following data:

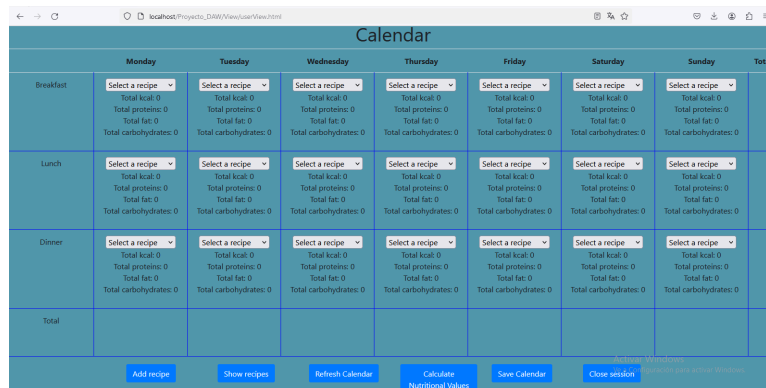
id	username	email	rol user	action
6	Antonio	antonio@user.com	user	Ban user
12	Alvaro	alvaro@user.com	user	Ban user
16	Lucia	lucia@user.com	user	Ban user
21	Pepe	pepe@user.com	banned	Ban user
41	Eto	eto@user.com	user	Ban user
45	Maria	maria@user.com	user	Ban user
48	Judith	judith@user.com	user	Ban user

Below the table, there is a button labeled 'Close session'.

Donde, como puede observarse, hay una tabla con la información de los diferentes usuarios y un botón al lado de cada uno de ellos para poder banearlos además de un botón que le permitirá cerrar su sesión y volver al login.

Dishes At Will (DAW) - Juan José Saborido Barranco

Y, con respecto a la vista de `userView.html` tenemos el siguiente diseño:



	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday	Total
Breakfast	Select a recipe Total kcal: 0 Total proteins: 0 Total fat: 0 Total carbohydrates: 0	Select a recipe Total kcal: 0 Total proteins: 0 Total fat: 0 Total carbohydrates: 0	Select a recipe Total kcal: 0 Total proteins: 0 Total fat: 0 Total carbohydrates: 0	Select a recipe Total kcal: 0 Total proteins: 0 Total fat: 0 Total carbohydrates: 0	Select a recipe Total kcal: 0 Total proteins: 0 Total fat: 0 Total carbohydrates: 0	Select a recipe Total kcal: 0 Total proteins: 0 Total fat: 0 Total carbohydrates: 0	Select a recipe Total kcal: 0 Total proteins: 0 Total fat: 0 Total carbohydrates: 0	
Lunch	Select a recipe Total kcal: 0 Total proteins: 0 Total fat: 0 Total carbohydrates: 0	Select a recipe Total kcal: 0 Total proteins: 0 Total fat: 0 Total carbohydrates: 0	Select a recipe Total kcal: 0 Total proteins: 0 Total fat: 0 Total carbohydrates: 0	Select a recipe Total kcal: 0 Total proteins: 0 Total fat: 0 Total carbohydrates: 0	Select a recipe Total kcal: 0 Total proteins: 0 Total fat: 0 Total carbohydrates: 0	Select a recipe Total kcal: 0 Total proteins: 0 Total fat: 0 Total carbohydrates: 0	Select a recipe Total kcal: 0 Total proteins: 0 Total fat: 0 Total carbohydrates: 0	
Dinner	Select a recipe Total kcal: 0 Total proteins: 0 Total fat: 0 Total carbohydrates: 0	Select a recipe Total kcal: 0 Total proteins: 0 Total fat: 0 Total carbohydrates: 0	Select a recipe Total kcal: 0 Total proteins: 0 Total fat: 0 Total carbohydrates: 0	Select a recipe Total kcal: 0 Total proteins: 0 Total fat: 0 Total carbohydrates: 0	Select a recipe Total kcal: 0 Total proteins: 0 Total fat: 0 Total carbohydrates: 0	Select a recipe Total kcal: 0 Total proteins: 0 Total fat: 0 Total carbohydrates: 0	Select a recipe Total kcal: 0 Total proteins: 0 Total fat: 0 Total carbohydrates: 0	
Total								

[Add recipe](#)
[Show recipes](#)
[Refresh Calendar](#)
[Calculate Nutritional Values](#)
[Save Calendar](#)
[Close session](#)

Vista `userView.html` (ver anexos)

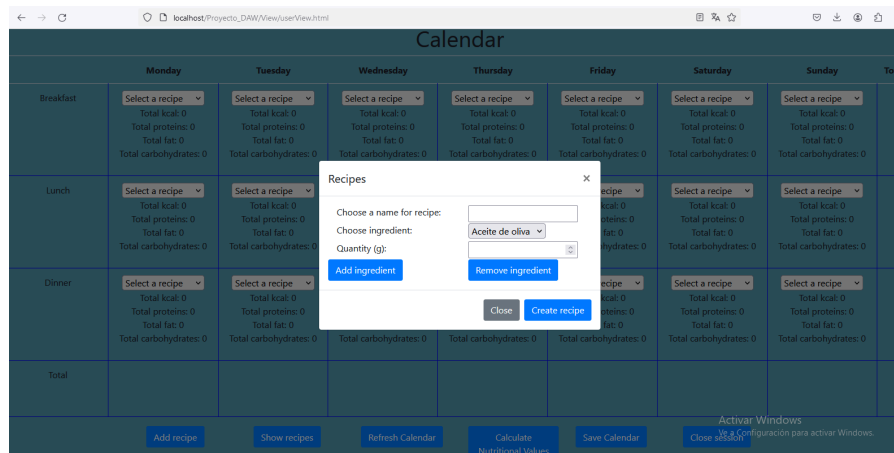
Donde en dicha vista tenemos, para empezar, un calendario con los diferentes días de la semana y tres posibles comidas por día (desayuno, almuerzo y cena) y en cada caso, un desplegable que contendrá todas las recetas que tenga el usuario con la sesión activa y, debajo del calendario, una serie de botones, con cada botón ofreciendo una funcionalidad distinta.

Por ejemplo, los cuatro últimos botones son:

- **Refresh calendar:** el usuario, al clicar sobre este botón, actualizará todas las recetas del desplegable del calendario, evitando así que pueda haber algún dato de receta que no esté actualizado como es debido.
- **Calculate nutritional values:** al hacer click aquí, se rellenará en el calendario los valores nutricionales de cada receta seleccionada además de que se calcularán los valores totales tanto diarios como semanales. Respecto a este botón cabe destacar que en un principio valoré que la propia aplicación avisara al usuario si superaba los valores nutricionales recomendados, pero después de informarme al respecto, comprobé que dichos valores dependen de la edad, la altura, la actividad deportiva del usuario, etcétera, haciendo que tuviera que meterme en detalles técnicos sobre nutrición y como dichas cuestiones se desviaban del objetivo real del proyecto, decidí descartarlo.
- **Save calendar:** botón que permitirá al usuario guardar dichas selecciones del calendario en nuestra base de datos.
- **Close session:** este botón será el encargado de cerrar la sesión al usuario cuando este considere que ha terminado de utilizar nuestra aplicación y mandará al usuario de vuelta a la vista `index.html`

Dishes At Will (DAW) - Juan José Saborido Barranco

Y, con respecto a los dos primeros botones, que son los que contienen una funcionalidad mayor, tenemos el primero de ellos que es **“Add recipe”** que desplegará un modal donde el usuario podrá introducir un nombre para la receta, un ingrediente concreto de nuestra base de datos, una cantidad de dicho ingrediente y la posibilidad de añadir más ingredientes si así lo desea (o eliminar ingredientes en caso de error).



Vista userView.html con el modal addRecipe desplegado

Como se puede observar en la barra de direcciones, la vista sigue siendo userView.html pero, sin embargo, se le está permitiendo al usuario, gracias a los diferentes modal creados, introducir una nueva receta sin tener que navegar a otra vista distinta, ofreciendo así una mayor experiencia de usuario.

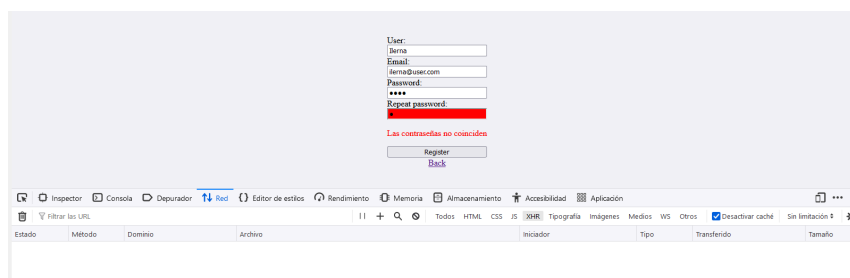
Y, por último, está el botón **“Show recipes”** que contendrá un modal con una tabla donde se mostrarán todos los datos de las recetas creadas hasta ahora y por cada receta tendrá dos botones, el primero para editar la receta (abriendo a su vez un nuevo modal similar al de addRecipe pero mostrando los datos actuales de la receta en cuestión) y el segundo para borrar la receta de la base de datos haciendo antes que el usuario confirme su decisión a través de un window.confirm() de JavaScript para asegurarnos de que no ha habido ningún error por parte del usuario.

Y con todo esta parte del diseño explicado, vamos a empezar a ofrecer una serie de pruebas a modo de demostración de que se ha conseguido cumplir con las funcionalidades de la aplicación como así se ha querido.

8. Despliegue y pruebas

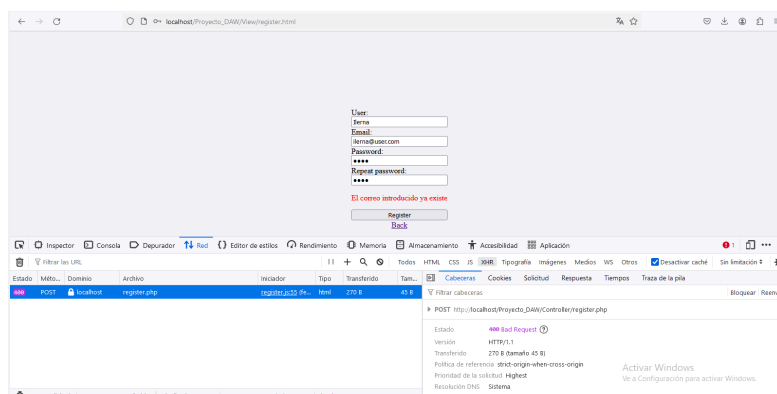
Para esta sección procederemos a realizar una serie de pruebas tanto de caja blanca como de caja negra, donde las primeras consisten en ver cómo reacciona nuestra aplicación sabiendo internamente cómo se comporta y las segundas consisten en comprobar las respuestas de nuestra aplicación sin preocuparnos de qué procesos internos están teniendo lugar.

Supongamos que somos un usuario nuevo que llega a nuestro sistema, veremos la vista de index.html (ver anexos), así que tendrá que hacer click en la parte de “Register here” para registrarse, lo que le llevará a la vista de register.html (ver anexos), aquí deberá de introducir sus datos de registro pero, supongamos que por algún casual, el usuario se equivoca y deja vacío el campo User e intenta registrarse, entonces dicho campo se iluminará de rojo y le aparecerá un mensaje en rojo que le informará al usuario que se le ha olvidado de introducir un usuario (ver anexos). En el supuesto caso de que el usuario deje el campo del email vacío o que, en su defecto, no introduzca un email válido, éste se iluminará en rojo y mostrará un mensaje de error de que el email no es correcto (ver anexos). Cabe destacar que esta validación tiene lugar gracias a las famosas expresiones regex que nos permiten comprobar si un email es, efectivamente, un email o no (aunque hay diferentes niveles de validaciones, el sistema, por ejemplo, no comprueba que dicho email exista, por ejemplo). E igualmente con las contraseñas ocurre lo mismo, si se deja el primer campo vacío se muestra el error al usuario rellenando el campo en rojo y mostrando un mensaje (ver anexos) e igual ocurre con el campo “Repeat password” que, además en este caso, si no coinciden las contraseñas, tampoco dejará registrarse al usuario en el sistema (ver anexos). Esto ocurre con todos los campos y en caso de que el usuario rellene un campo que antes daba conflicto, dicho campo volverá a la normalidad y procederá a comprobarse si el siguiente campo es correcto o no. Lo importante de esto es que en ninguno de estos casos se intentará mandar los datos al lado del servidor, como se puede apreciar desde la consola en la pestaña de “Red”/”Network” como se muestra en la siguiente imagen:



Dishes At Will (DAW) - Juan José Saborido Barranco

Ahora bien, si el usuario finalmente introduce todos los campos como es debido, se mandará el nuevo registro a nuestra base de datos y se intentará insertar dicho registro y, en caso de que todo haya ido bien, se mandará al usuario a la vista de index.html. Sin embargo, si el usuario intenta registrarse con un correo que actualmente ya existe en nuestra base de datos, podrá observarse que se han ejecutado las validaciones en JavaScript y se ha mandado una petición de tipo POST al archivo register.php, donde se le ha devuelto una respuesta con un 400 y con un mensaje de error que dice “El correo introducido ya existe”, mensaje de error que se le mostrará al usuario del mismo modo que se han mostrado el resto (dicho mensaje puede verse en la pestaña de “respuesta” en la sección de Red)



Register.html con error del servidor (ver anexos)

Lo importante aquí es señalar que efectivamente la petición ha llegado al servidor (el backend) y que éste la ha rechazado o no en función de los datos introducidos y ha mandado una respuesta a JavaScript (el frontend) para que este procese dicha respuesta y la muestre por pantalla (arriba del botón de “Register” puede observarse el mensaje en rojo “El correo introducido ya existe” que es la misma respuesta que nos ha devuelto el servidor) o mande al usuario a la vista index.html si la respuesta ha sido satisfactoria.

Cabe destacar también, que a pesar de que se han llevado a cabo validaciones en el frontend (JavaScript), se han realizado las mismas validaciones en el backend (php) porque desde el frontend se puede engañar al sistema para saltarse dichas validaciones, pero en el backend es algo mucho más complicado esto y con tal de mejorar la ciberseguridad de nuestra aplicación web, estas validaciones en concreto se llevan a cabo tanto en el frontend como en el backend por lo que pudiera pasar (aunque ya digo que, como mínimo, toda validación debe existir en el backend y, opcionalmente, en el frontend). En este caso, por ejemplo, con fin de evitar realizar peticiones al servidor con posibles errores en los datos

Dishes At Will (DAW) - Juan José Saborido Barranco

proporcionados que, recordemos, mandar peticiones al servidor supone un consumo de recursos adicional, opté por realizar dichas comprobaciones en el frontend y, una vez se mandara la petición con los datos que sean, realizar las comprobaciones en el backend para evitar posibles sorpresas.

En la misma línea de la ciberseguridad de nuestra aplicación, a la hora de llevarse a cabo el INSERT de nuestro nuevo registro en la base de datos, no solo se ha trabajado con prepared statements para evitar inyecciones SQL (en esta consulta y en todas), sino que, además, se ha hashado la contraseña con el fin de que, si por algún casual, alguien accede a nuestro sistema, este no pueda saber qué contraseñas usan nuestros usuarios, quedando guardado el registro en nuestra base de datos del siguiente modo:

	user_id	username	email	password_user	rol_user
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	1	Juan Jose	juanjo@admin.com	\$2y\$10\$ANDCOO7gow7szgYgRkRumOhjOCuAAltD98AdWsNO1h...	superadmin
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	6	Antonio	antonio@user.com	\$2y\$10\$AOGMr31Nsl8gRbzFxQReURYLsIGNOVs4QdVxgjAW...	user
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	12	Alvaro	alvaro@user.com	\$2y\$10\$Kkbpolyg3ctXcsMIBGi8JnI IQoOyUjQ/duxWA3yCkF...	user
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	16	Lucia	lucia@user.com	\$2y\$10\$BYOazgezeOUTrCG7FBh8sOyDhPKFey0XQKMBLo5xoV94...	user
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	21	Pepe	pepe@user.com	\$2y\$10\$mRtNZol32XLW0T Eu84fuzQwtLHYW2wBIRe9.cl.d...	banned
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	41	Eto	eto@user.com	\$2y\$10\$AOGMr31Nsl8gRbzFxQReURYLsIGNOVs4QdVxgjAW...	user
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	45	Maria	maria@user.com	\$2y\$10\$Yozlcy9L7l6hnr38nxTgpu7mw86byMdRX37HGHFPV5g...	user
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	48	Judith	judith@user.com	\$2y\$10\$KLu956lftmPhvosM3Z0Uw.3a6oOYGP9HEgpreJMoYQL...	user
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	49	Ilerna	ilerna@user.com	\$2y\$10\$co9x61aIfA3QX.32eSGLv2h/s3mi3kmR6FmCr...	user

Tabla users con datos concretos (ver anexos)

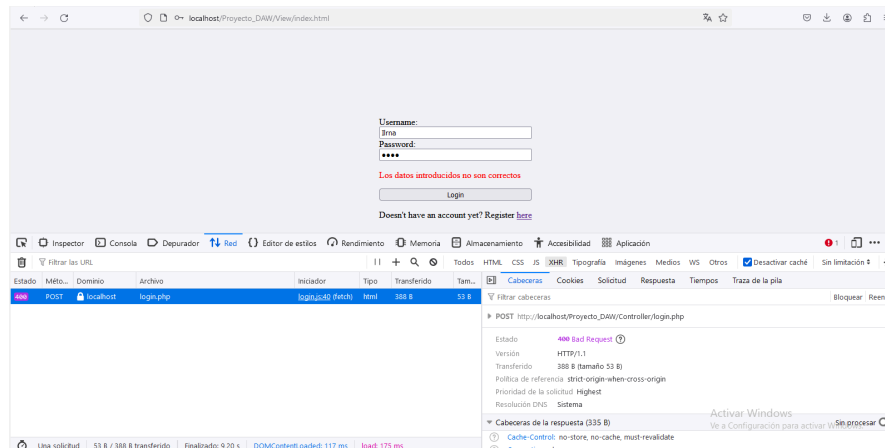
Pero a modo de información y esto es algo que nunca debe hacerse (porque cualquier persona puede terminar leyendo este documento...), todos los usuarios registrados hasta este momento usan la misma contraseña que es 1234 a excepción del usuario Juan José que es superadmin y usa la contraseña asdf (aunque dicha información solo la dejo escrita por si el lector quiere realizar pruebas con la aplicación también, pero dichas contraseñas deberían cambiarse en caso de ser una aplicación real y, por supuesto, nada de dejarlas por escrito en ningún lado).

Ahora que ya tenemos creado un usuario nuevo (recordemos, Ilerna con contraseña 1234), vamos a proceder a loguearnos en la vista de index.html.

Del mismo modo que ocurría con el registro, si el usuario intenta loguearse con algún campo vacío, el propio frontend evitará enviar una petición al servidor (para no repetirme, ocurriría lo mismo que en el caso anterior, por evitar no poner excesivas capturas de pantalla en los anexos, aunque si se desea, se puede comprobar desplegando la aplicación sin problema), lo que sí que vamos a observar es qué ocurre si intenta mandar datos incorrectos de sesión como, por ejemplo, que en vez de introducir el usuario "Ilerna", escriba "Ilrna" sin la "e", en tal caso como dicho usuario no existe en nuestra base de datos, se

Dishes At Will (DAW) - Juan José Saborido Barranco

enviará la petición al archivo login.php y este responderá con un 400 y una respuesta que se mostrará como texto justo encima del botón de Login del siguiente modo:



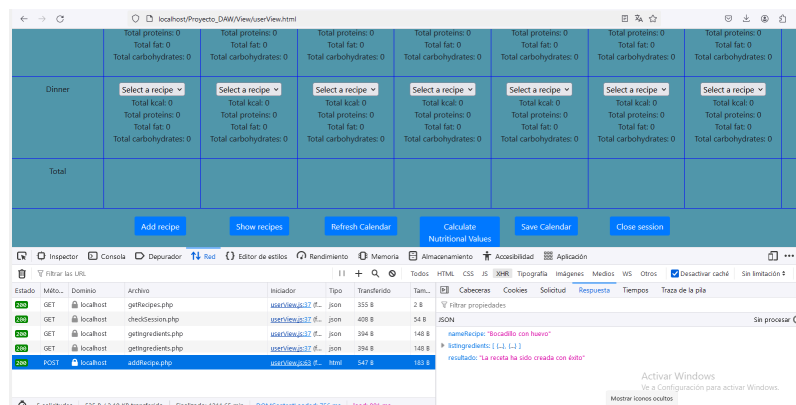
Vista index.html con datos de login incorrectos (ver anexos)

Esto quiere decir que se ha intentado realizar la petición al backend y que este ha rechazado dicha petición (justo como en el caso anterior del registro) pero, por otro lado, si el usuario corrige dicho error, se le mandará a la vista `userView.html` que es el grosor de nuestra aplicación y en el `login.php` se inicia una nueva sesión para el usuario (aunque esto ocurre internamente con un `session_start()` en el lado del servidor) y, nada más cargar la página, podemos ver que se realizan dos peticiones a dos archivos php distintos, el primero que es `getRecipes.php` que carga las recetas que puede tener ya creadas el usuario y el segundo que es `checkSession.php` que comprueba que, efectivamente hay sesión activa para dicho usuario y devuelve los valores de `active_session`, `user_id` y `rol_user`, donde el primer valor determina que, efectivamente, dicho usuario tiene sesión activa, el segundo valor nos permitirá que, a la hora de crear o modificar recetas, sepamos de qué usuario se trata y el tercer valor nos sirve para efectivamente confirmar que la vista que debería estar mostrándose sea la de `user` y no la de `superadmin` o `banned` (ver anexos para corroborar esta información).

Como es lógico, nuestro nuevo usuario `Ilerna` no tiene de momento ninguna receta creada así que, lo normal, es que cliquee sobre el botón de “Add recipe” y empiece a crear sus recetas. Al hacer click sobre Add recipe se desplegará el modal asociado a dicho botón y se hará una petición al archivo `getIngredients.php` que pedirá los ingredientes de la base de datos de la tabla `ingredients` y los devolverá al frontend para que JavaScript recoja todos esos ingredientes y los muestre en el desplegable correspondiente (ver anexos).

Dishes At Will (DAW) - Juan José Saborido Barranco

Ahora nuestro usuario podrá empezar a darle un nombre a la receta, seleccionar ingredientes y sus cantidades y añadir más ingredientes a su receta. Cabe destacar de que si el usuario escribe algo y entonces se acuerda de que le falta un nuevo ingrediente, los datos introducidos hasta entonces no se eliminan (porque el html no se refresca), sino que lo que ocurre es que se le añade/elimina el elemento html concreto,...por desgracia esta funcionalidad no se puede demostrar con imágenes, así que encomiendo encarecidamente al lector/a que lo pruebe por sí mismo/a. Lo que sí que se puede demostrar con imágenes es que, cuando el usuario le da al botón de “Create recipe” este manda una petición al archivo addRecipe.php y este realiza el insert en la base de datos con los datos proporcionados (si es que todo ha ido bien). A modo de ejemplo, crearé una receta que sea “Bocadillo con huevo” y que tenga 150g de pan blanco y 75g de huevos y el resultado será este:



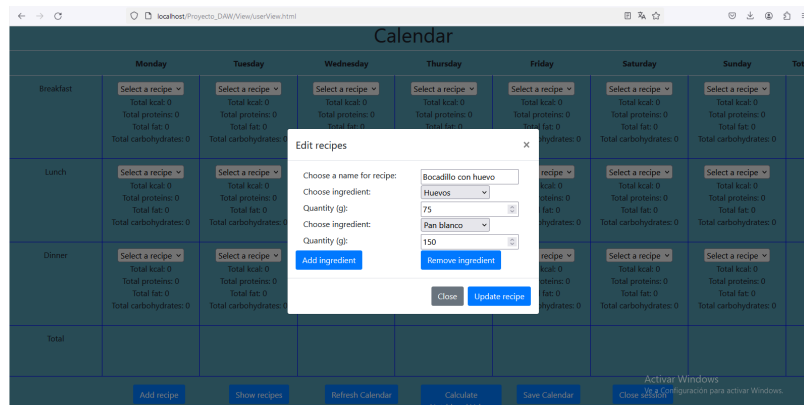
Respuesta del backend después de añadir receta “Bocadillo con huevo” (ver anexos)

Y en la pestaña de “Consola” podrá leerse el mensaje “La receta ha sido creada con éxito”. Si este proceso se lleva a cabo todas las veces que sean hasta que el usuario cree todas las recetas que crea necesario, podemos proseguir con las pruebas.

Ahora que ya hemos creado varias de las recetas que queramos tener para usar en nuestro calendario, toca hablar del siguiente botón qué es el de “Show recipes”. Dicho botón nos desplegará el modal showRecipes y en dicho modal veremos la información de todas las recetas creadas hasta entonces. Cabe destacar que, al hacer click en dicho botón y desplegarse el modal, automáticamente se lleva a cabo una petición al archivo getRecipes.php para así poder cargar todas las recetas que tenga el usuario con toda su información pertinente (ver anexos). Y, desde este modal, podremos editar e incluso borrar la receta que seleccionemos. Empezaremos por editar la receta “Bocadillo con huevo”.

Dishes At Will (DAW) - Juan José Saborido Barranco

Para ello, haremos click en el botón de “Edit” y acto seguido se ocultará el modal showRecipes y se activará el modal editRecipes, mostrándose lo siguiente:



Modal editRecipes con los datos de la receta seleccionada (ver anexos)

Con la peculiaridad de que se han rellenado automáticamente los campos y los ingredientes con los datos que tiene actualmente la receta seleccionada y, si modificamos dichos datos (como, por ejemplo, cambiar la cantidad de huevos a 50g y le damos al botón “Update recipe” veremos como se manda una petición al archivo editRecipe.php con la respuesta de que “todo ha ido bien” y, si le volvemos a dar al botón de “Show recipes” veremos como dicha información ha sido actualizada con éxito (ver anexos).

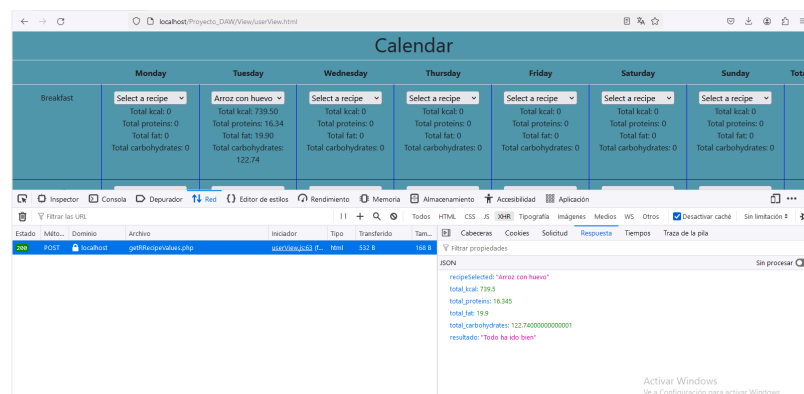
Por el contrario, si desde dicho modal clicamos sobre el botón “Delete” nos saltará una ventana para confirmar nuestra decisión y, si le damos a Cancelar no pasará nada pero si le damos a aceptar se le mandará una petición al backend al archivo deleteRecipe.php con un resultado de “Todo ha ido bien” y la receta eliminada del modal de showRecipes (ver anexos).

Aviso importante: a modo de agilizar las pruebas, a partir de ahora, voy a hacer uso del usuario “Lucia”, cuya contraseña es 1234, y su id de usuario es 16, puesto que, con esta usuaria, ha sido con quien he llevado a cabo varias pruebas durante el desarrollo del proyecto y, por tanto, tiene ya varias recetas creadas con las que poder trabajar. Así que cerraré la sesión de Ilerna haciendo click en el botón “Close session” (botón que explicaré más adelante su funcionamiento), me loguearé con el usuario Lucia y proseguiré las pruebas a partir de ahí, aunque cualquier acción es extrapolable a cualquier otro usuario siguiendo los mismos pasos.

Dishes At Will (DAW) - Juan José Saborido Barranco

Ahora nos tocaría hablar del siguiente botón que es “Refresh calendar”, dicho botón lo que hace es volver a cargar las recetas de la base de datos e introducir dichas recetas en el menú desplegable del calendario. Esto se debe a que, como no se vuelve a recargar la página (puesto que el usuario se mueve entre modals, pero la vista no se actualiza) hay situaciones donde el usuario cree/edite/elimine una receta y estos cambios no se vean reflejados en el propio calendario, pues al hacer click aquí, todos los datos se actualizan por si pudiera haber algún error.

Una vez tenemos en nuestro calendario actualizado con los desplegables con todas las recetas, podemos seleccionar cada una de dichas recetas en el lugar del calendario que corresponda y, en tal caso, el cliente mandará una petición al servidor, más concretamente al archivo getRecipeValues.php con los valores nutricionales de dicha receta (que se han calculado automáticamente en el lado del servidor, cuando la receta fue creada) y lo que hace el cliente es coger dicha respuesta y mostrarla en su lugar correspondiente del calendario con los valores redondeados a dos decimales. Por ejemplo, supongamos que el usuario decide que el martes para desayunar va a comer Arroz con huevo, el desayuno de los campeones, receta que en su día la creó y que figura en nuestra base de datos, pues bien, nada más seleccionar en el desplegable “Arroz con huevo” ocurre lo siguiente:



Resultado al elegir Arroz con huevo para desayunar el martes (ver anexos)

Como puede observarse, se han mostrado los valores justo debajo de la selección con sus valores nutricionales de la receta, pudiendo así ver cuánto de saludable es dicha decisión y pudiendo optimizar cada campo para sacar el máximo provecho a la dieta.

Dishes At Will (DAW) - Juan José Saborido Barranco

Por otro lado, el siguiente botón se trata del de “Calculate nutritional values”, que lo que hace dicho botón es calcular los totales tanto diarios como semanales como por comidas de cada una de las recetas que ha seleccionado el usuario, pudiendo saber así el total de cada una de sus decisiones y, una vez más, dándole la posibilidad al usuario de optimizar sus elecciones (ver anexos para observar el resultado). Cabe destacar que para calcular dichos valores totales, no se ha realizado ninguna petición al servidor, sino que se ha cogido el valor de cada uno de los campos del frontend y se han realizado los cálculos con los mismos (optimizando así un poco más la aplicación, evitando hacer peticiones al backend innecesarias).

Después de esto, el usuario puede querer guardar su calendario si es que de verdad le ha convencido su resultado, para esto se ha creado el siguiente botón “Save calendar”. Dicho botón mandará una petición al archivo saveCalendar.php que insertará o actualizará el calendario asociado a dicho usuario en nuestra base de datos (manteniendo siempre un calendario por usuario).

localhost/Proyecto_DAW/View/user/View.html

Dinner

Tortilla

Huevos

Arroz con huevo

Tortilla

Papas

Huevos

Tortilla de papas

Total

Total kcal: 127.20
Total proteins: 0.00
Total fat: 14.00
Total carbohydrates: 31.99

Total kcal: 171.87
Total proteins: 2.40
Total fat: 22.00
Total carbohydrates: 60.19

Total kcal: 799.50
Total proteins: 16.34
Total fat: 19.90
Total carbohydrates: 122.74

Total kcal: 127.20
Total proteins: 0.00
Total fat: 14.00
Total carbohydrates: 31.99

Total kcal: 479.50
Total proteins: 0.00
Total fat: 53.20
Total carbohydrates: 13.50

Total kcal: 171.87
Total proteins: 2.40
Total fat: 22.00
Total carbohydrates: 60.19

Total kcal: 939.45
Total proteins: 21.21
Total fat: 59.63
Total carbohydrates: 74.51

Total kcal: 1545.95
Total proteins: 21.21
Total fat: 126.83
Total carbohydrates: 120.00

Total kcal: 1850.82
Total proteins: 39.95
Total fat: 101.53
Total carbohydrates: 257.44

Total kcal: 1806.15
Total proteins: 37.55
Total fat: 93.53
Total carbohydrates: 229.24

Total kcal: 1038.57
Total proteins: 18.74
Total fat: 55.90
Total carbohydrates: 214.92

Total kcal: 778.27
Total proteins: 2.40
Total fat: 89.20
Total carbohydrates: 105.68

Total kcal: 1238.52
Total proteins: 23.61
Total fat: 95.63
Total carbohydrates: 166.69

Total kcal: 1850.82
Total proteins: 39.95
Total fat: 101.53
Total carbohydrates: 257.44

Total kcal: 10109.20
Total proteins: 183.41
Total fat: 664.15
Total carbohydrates: 1351.41

Add recipe

Show recipes

Refresh Calendar

Calculate Nutritional Values

Save Calendar

Close session

Inspector

Consola

Depurador

Red

Editor de estilos

Rendimiento

Memoria

Almacenamiento

Accesibilidad

Aplicación

Filtrar las URLs

||

+

?

×

Todos

HTML

CSS

JS

XHR

Tipografía

Imágenes

Medios

WS

Otros

Desactivar caché

Sin limitación

Ente

Metas...

Domino

Archivo

Incisor

Tipo

Transferido

138...

15448

138...

Filtrar preferencias

Cache

Cookies

Seguridad

Respuesta

Tiempos

Taza de la pila

index

POST

localhost

localhost/calcular.php

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

localhost/...

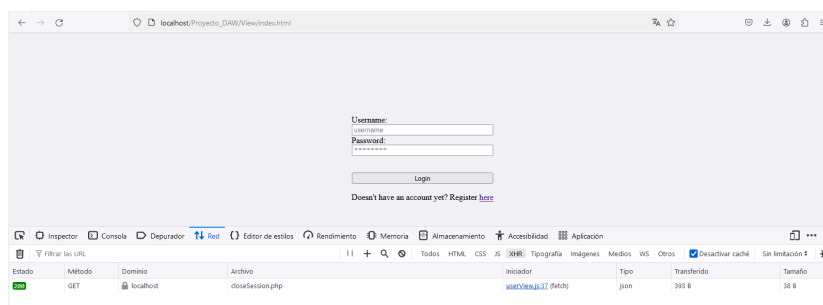
localhost/

Respuesta con el calendario guardado (ver anexos)

Y si se quiere comprobar en la base de datos que, efectivamente se ha actualizado dicho calendario para el usuario, bastará con observar dicho cambio en PhpMyAdmin (ver anexos) con los ids respectivos del usuario (en este caso recordemos que Lucia tiene id 16) y los ids de cada una de las recetas en cada campo concreto.

Finalmente, nos queda por hablar del último botón que es “Close session” que, para sorpresa de nadie, lo que hace es, efectivamente, cerrar la sesión del usuario activo pero para esto permítanme demostrarles diferentes casuísticas.

Supongamos por algún casual que el usuario, por error o intencionadamente, decide irse momentáneamente a otra página web y luego decide volver a nuestra aplicación, sería cuanto menos tedioso el obligar al usuario a tener que volver a introducir otra vez sus credenciales en la vista de index.html, pues en principio eso no ocurre gracias a que iniciamos en el servidor una variable de sesión para dicho usuario, así que en el caso de que un usuario deje temporalmente nuestra aplicación no va a suponerle ningún problema puesto que puede volver a ella sin perder su sesión. Sin embargo, cuando el usuario decida que, efectivamente, ha terminado de trabajar con nuestra sesión, podrá cerrarla haciendo click en dicho botón, evitando así, que una tercera persona pueda acceder a su sesión sin sus credenciales (aumentando de este modo la seguridad de nuestra aplicación web). Para esto, al hacer click en el botón, se manda una vez más, una petición al servidor al archivo closeSession.php y se mandará al usuario de vuelta a la vista index.html como se muestra a continuación:

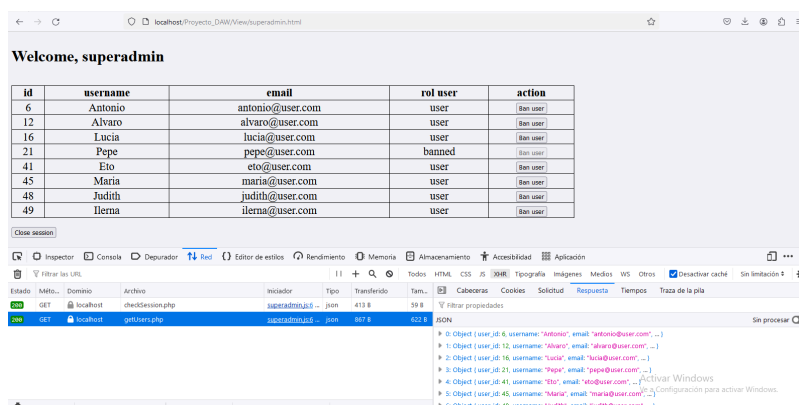


Petición al archivo closeSession.php para cerrar la sesión (ver anexos)

Y en el caso de que ahora mismo un usuario intente acceder de nuevo a dicha vista introduciendo la dirección manualmente en la URL, se le redirigirá a la vista userViewRejected.html y esta vez devolviendo un 400 el archivo checkSession.php (ver anexos). Vista que, por supuesto, ofrece un enlace para volver a index.html y que el usuario pueda introducir sus credenciales para volver a loguearse.

Por último, nos queda hablar de nuestro usuario “superadmin”, dicho usuario (con username “Juan José” y con contraseña “asdf”) tendrá acceso a una vista personalizada llamada superadmin.html. Al cargar dicha vista, lo primero que hace la aplicación es comprobar que, en efecto, quién está intentando acceder a ella se trata del usuario con rol superadmin y, si esto es cierto, muestra el contenido de la misma, realizando a su vez una petición al recurso del servidor getUsers.php que le proporcionará al frontend una lista con todos los usuarios, email, roles y un botón para poder banear a dicho usuario.

Dishes At Will (DAW) - Juan José Saborido Barranco



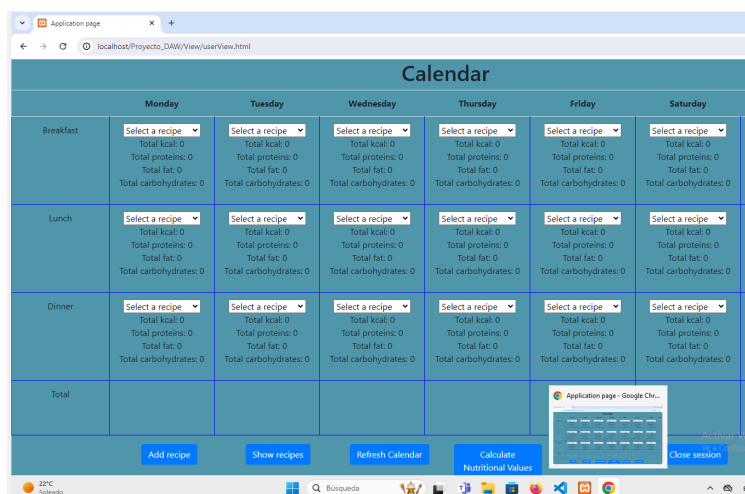
Vista superadmin.html con getUsers.php (ver anexos)

Y, con fines científicos simplemente, vamos a proceder a banear al usuario Ilerma que hemos creado previamente. Al hacer click en dicho botón, se mandará una petición al archivo banUser.php que actualizará el rol del usuario Ilerma de user a banned (ver anexos para comprobar el resultado). El html se cambiará para que se desactive el botón (un usuario baneado no tiene sentido que se pueda volver a banear) y se cambiará el rol user en la tabla de dicha vista para que pueda visualizarse el cambio sin necesidad de que se vuelva a cargar la página ni nada.

Y el superadmin, al hacer click en el botón de “Close session” se le cerrará la sesión del mismo modo que ocurría con los usuarios normales.

Ahora bien, si en este mismo momento, intentamos loguearnos con el usuario de Ilerma ocurrirá que, en vez de mandarnos a la vista de userView.html como ocurrió antes, se nos mandará a la vista de userBanned.html a pesar de que el login.php haya dado un “ok” como respuesta, porque efectivamente dicho usuario existe en nuestra base de datos y, por tanto, puede seguir logueandose (ver anexos).

Finalmente quedaría comprobar que todas estas vistas se ven igual independientemente de qué navegador estemos utilizando. Hasta ahora hemos utilizado mozilla firefox para llevar a cabo todas estas pruebas pero, a modo de demostración, comprobaremos que dicha aplicación funciona también tanto en google chrome como en microsoft edge y para ello, tomaremos un par de capturas de cada uno de ellos (aunque hay que decir que esta aplicación ha sido completamente testada en dichos navegadores y sin ningún tipo de problema).



Vista userView.html en google chrome (ver anexos)

Y si comprobamos la vista de index.html (ver anexos también), podremos observar que no hay cambios importantes y que la aplicación funciona perfectamente (recordemos que la vista register.html es muy similar a la de index.html y las vistas de userViewRejected.html y userBanned son bastante sencillas y la de superadmin.html es una versión más simple que la de userView.html, por eso solo corroboramos su funcionamiento en estas dos vistas, aunque ya digo que funciona igual). Y análogamente, si abrimos microsoft edge y vamos a nuestra aplicación, podremos ver un resultado similar, como era de esperar (ver anexos).

Con esta última prueba, quedaría por concluida esta sección aunque, por supuesto, se pueden realizar muchas más pruebas y ánimo a que se lleven a cabo por parte del lector diferentes experimentos y que me proporcione feedback si así lo considera oportuno.

9. Conclusiones

Para ser completamente honestos, personalmente al menos, estoy bastante contento con el proyecto en sí, si bien tiene sus carencias y sus posibles mejoras, considero que el resultado es bastante sólido.

Gracias a este proyecto he podido afianzar mucho más mis conocimientos sobre html, css y sql, pero sobre todo todo, de JavaScript y PHP, donde no solo he conseguido fortalecer todo lo aprendido durante el ciclo formativo, sino que, además, he ampliado mucho más dichos conocimientos. He aprendido nociones sobre ciberseguridad, gestión de diferentes problemáticas, cómo enfrentarlas y cómo solventarlas, gestión de sesiones, gestión de usuarios, hacer funcionalidades CRUD (las famosas por sus siglas "Create, Read, Update, Delete"), el uso de las promesas de JavaScript y el entendimiento de la asincronía de dicho lenguaje y, de lo que más orgulloso estoy, es del hecho de que he podido separar completamente el entorno cliente y el entorno servidor haciendo que se comuniquen entre sí con la función fetch y las peticiones de tipo http.

Aunque no ha sido una tarea fácil, he podido cumplir con mis objetivos, he aprendido mucho durante el proceso y, si bien ha sido duro llevar a cabo un proyecto de estas características por uno mismo, compaginando prácticas y proyecto y con los tiempos establecidos, echando la vista atrás, siento que ha sido una experiencia muy satisfactoria.

Solo me quedaría agradecer todo el apoyo mostrado a mi familia y amigos, a Fernando, mi tutor, por haber estado siempre ahí respondiendo todas las posibles dudas que pudieran surgirme respecto a la memoria escrita, a todos/as los/as profesores/as que me han acompañado durante mis dos años de aprendizaje (y que gracias a más de uno/a he visto alimentadas mis ganas de aprender y de querer mejorar en este mundo de la informática), a Ilerna, por haberme permitido llevar a cabo todo este proyecto con ellos y, por supuesto, al lector/a que esté leyendo esto, gracias por llegar hasta aquí.

A todo el mundo, gracias.

10. Vías futuras

La aplicación, como bien he comentado, tiene margen de mejora. Estoy seguro que se podría mejorar aún más la ciberseguridad del sistema, se le podrían añadir más funcionalidades a la aplicación (como, por ejemplo, que el superadmin pueda desbanear a un usuario o que los usuarios pudieran interactuar entre sí, pudiendo comentar las recetas de algún usuario conocido, tener sistema de “amigos”, etcétera) y se le podría mejorar mucho más el apartado estético de las diferentes vistas añadiéndoles más CSS o bien haciendo un mayor uso de Bootstrap o de algún framework que también cuide este aspecto como Tailwind o Bulma.

Se podría aumentar, por supuesto la cantidad de ingredientes que hay en la base de datos (que, al final, he rellenado unos valores que pueden considerarse “comunes” pero que dista mucho de la cantidad de ingredientes que hay en el mercado actual), podría ofrecerse un sistema de suscripción con mejoras en la aplicación (si es que en algún momento se llegara a buscar la comercialización de la misma) y se podría crear un calendario mucho más amplio que no solo abarque la semana actual, sino que sea capaz de abarcar un calendario, por ejemplo, anual.

Otra posibilidad es crear una lista de posibles dietas que el usuario pueda decidir y, en función de la dieta seleccionada, la propia aplicación ofreciera ideas de posibles calendarios que seguir en función de las recetas del usuario.

Como puede verse, hay un mundo entero de posibilidades que se podrían llevar a cabo en esta aplicación, dependiendo de los intereses personales y generales que puedan ir surgiendo por parte de la comunidad, pero esa es otra historia para otro futuro proyecto que pueda tomar este proyecto como base.

Hasta este momento, he llegado hasta aquí, y el futuro ya determinará qué caminos pueda tomar esta aplicación.

11. Bibliografía/Webgrafía

Figuras e imágenes:

- Fig. 1:
<https://www.dicyt.com/noticias/una-sustancia-de-la-leche-materna-contribuye-al-desarrollo-neurologico-del-bebe>
- Fig. 2:
<https://www.worldhistory.org/image/7519/de-re-coquinaria-the-art-of-cooking/>
- Fig 3: Libro de Entornos de Desarrollo de Ilerna, página 64 del libro.
- Página web de la Ley de Protección de Datos:
<https://www.boe.es/buscar/act.php?id=BOE-A-2018-16673>
- Página web con los datos nutricionales de diferentes productos:
<https://www.dietas.net/tablas-y-calculadoras/tabla-de-composicion-nutricional-de-los-alimentos/>
- Documentación de la función fetch:
<https://developer.mozilla.org/es/docs/Web/API/fetch>
- Códigos de respuesta de peticiones HTTP:
<https://developer.mozilla.org/en-US/docs/Web/HTTP/Status>

12. Anexos

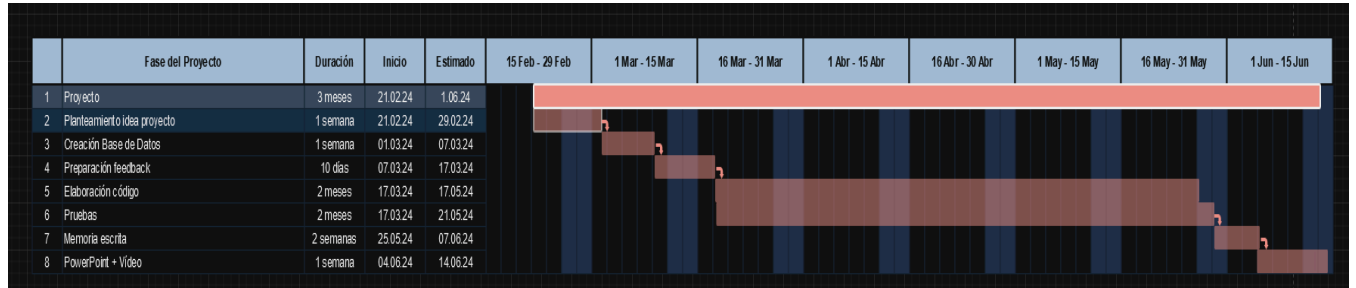


Diagrama de Gantt ampliado

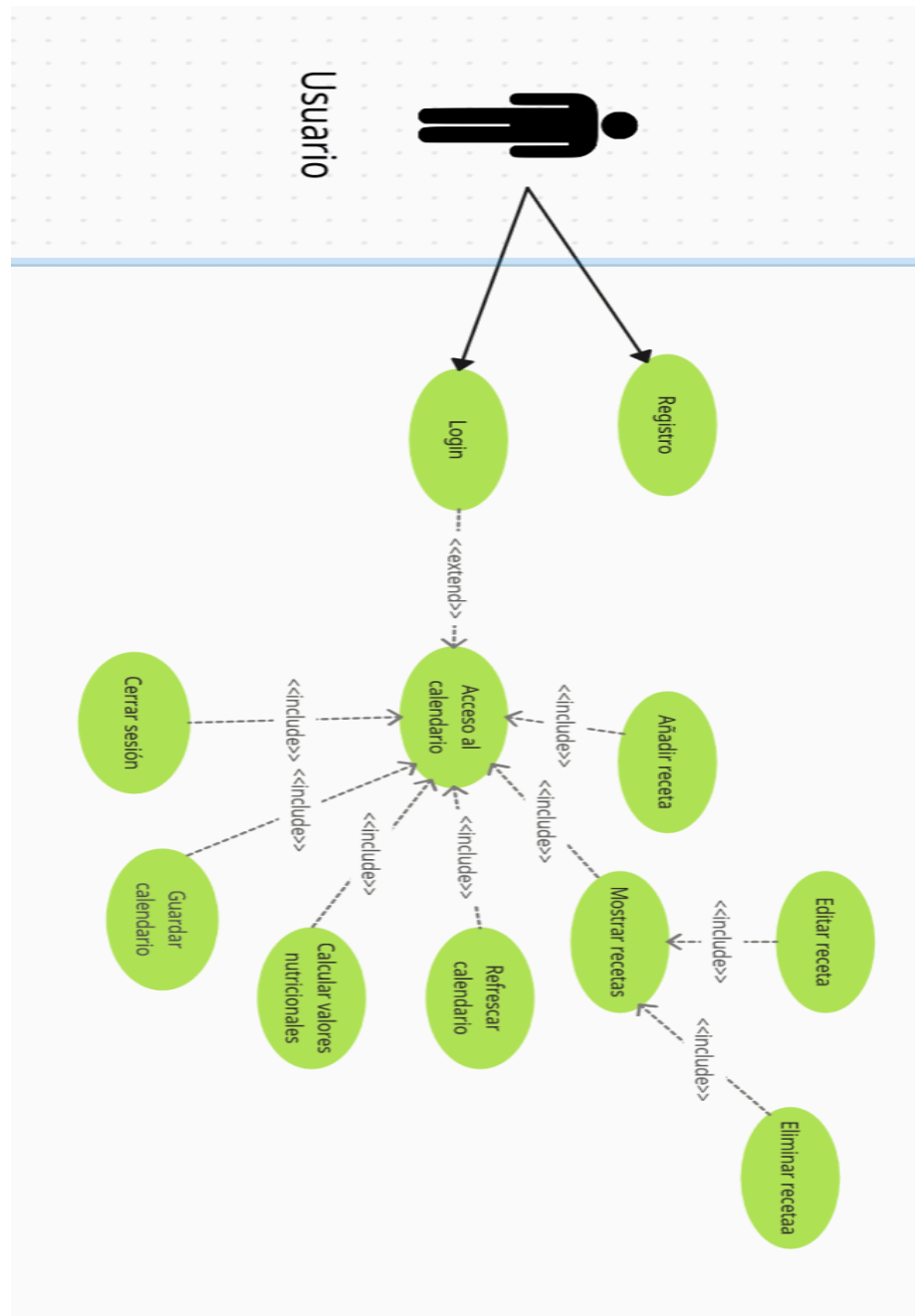


Diagrama casos de uso para un usuario

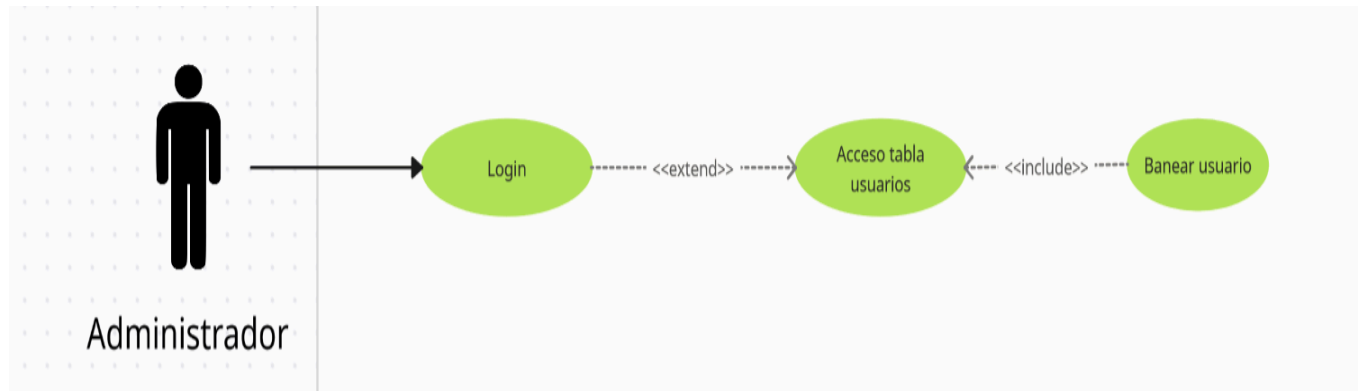


Diagrama de casos de uso para un administrador

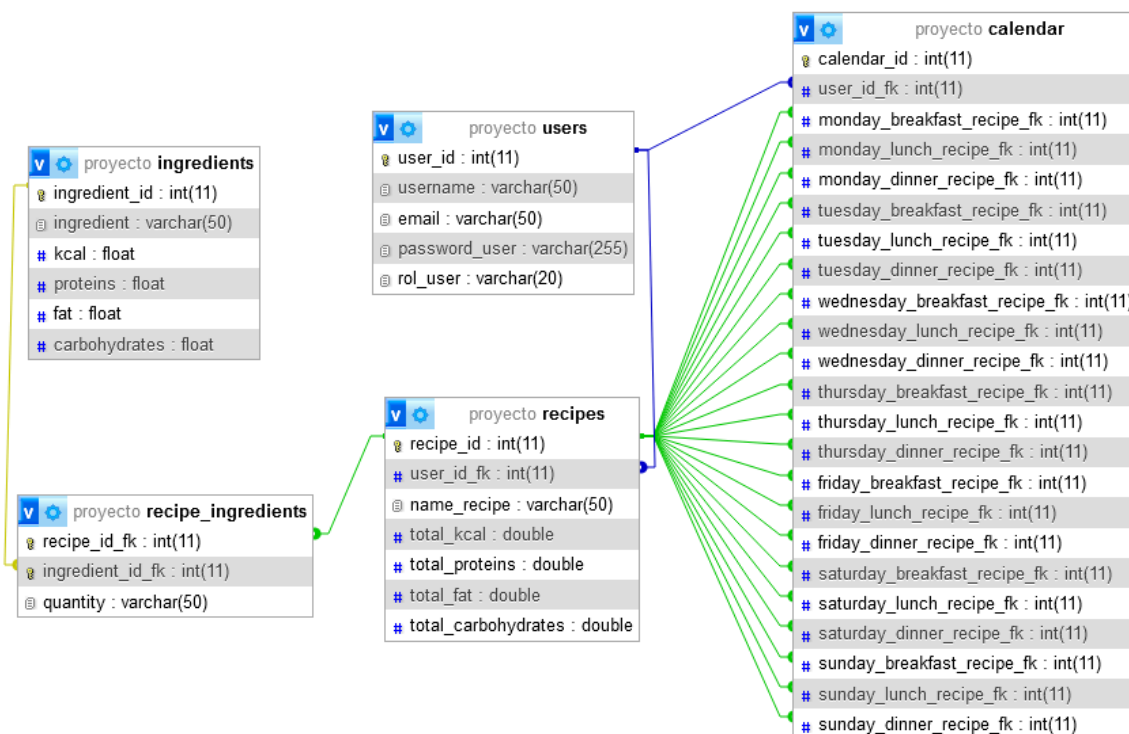


Diagrama Entidad-Relación

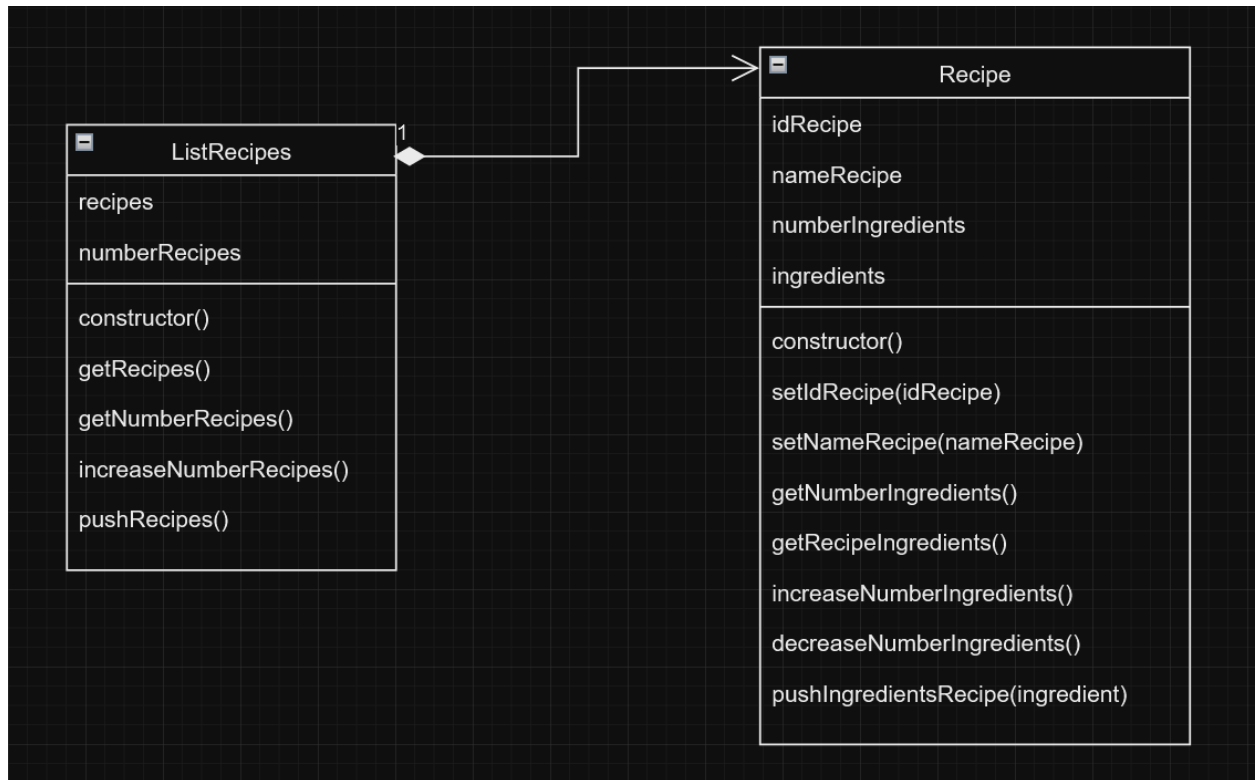


Diagrama de clases

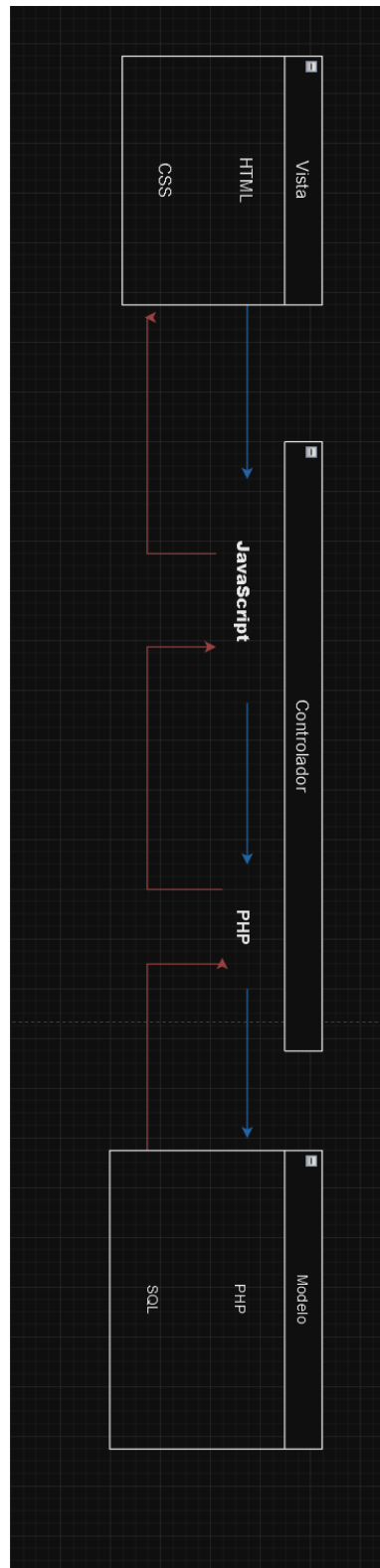


Diagrama de flujo de la aplicación

←

→

G

localhost/Proyecto_DAW/userView.html

🏠

🔍

🌟

📄

🔖

🔍

🔍

🔍

☰

Calendar

	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday	Total
Breakfast	<div>Select a recipe ▾</div> <div>Total kcal: 0</div> <div>Total proteins: 0</div> <div>Total fat: 0</div> <div>Total carbohydrates: 0</div>	<div>Select a recipe ▾</div> <div>Total kcal: 0</div> <div>Total proteins: 0</div> <div>Total fat: 0</div> <div>Total carbohydrates: 0</div>	<div>Select a recipe ▾</div> <div>Total kcal: 0</div> <div>Total proteins: 0</div> <div>Total fat: 0</div> <div>Total carbohydrates: 0</div>	<div>Select a recipe ▾</div> <div>Total kcal: 0</div> <div>Total proteins: 0</div> <div>Total fat: 0</div> <div>Total carbohydrates: 0</div>	<div>Select a recipe ▾</div> <div>Total kcal: 0</div> <div>Total proteins: 0</div> <div>Total fat: 0</div> <div>Total carbohydrates: 0</div>	<div>Select a recipe ▾</div> <div>Total kcal: 0</div> <div>Total proteins: 0</div> <div>Total fat: 0</div> <div>Total carbohydrates: 0</div>	<div>Select a recipe ▾</div> <div>Total kcal: 0</div> <div>Total proteins: 0</div> <div>Total fat: 0</div> <div>Total carbohydrates: 0</div>	
Lunch	<div>Select a recipe ▾</div> <div>Total kcal: 0</div> <div>Total proteins: 0</div> <div>Total fat: 0</div> <div>Total carbohydrates: 0</div>	<div>Select a recipe ▾</div> <div>Total kcal: 0</div> <div>Total proteins: 0</div> <div>Total fat: 0</div> <div>Total carbohydrates: 0</div>	<div>Select a recipe ▾</div> <div>Total kcal: 0</div> <div>Total proteins: 0</div> <div>Total fat: 0</div> <div>Total carbohydrates: 0</div>	<div>Select a recipe ▾</div> <div>Total kcal: 0</div> <div>Total proteins: 0</div> <div>Total fat: 0</div> <div>Total carbohydrates: 0</div>	<div>Select a recipe ▾</div> <div>Total kcal: 0</div> <div>Total proteins: 0</div> <div>Total fat: 0</div> <div>Total carbohydrates: 0</div>	<div>Select a recipe ▾</div> <div>Total kcal: 0</div> <div>Total proteins: 0</div> <div>Total fat: 0</div> <div>Total carbohydrates: 0</div>	<div>Select a recipe ▾</div> <div>Total kcal: 0</div> <div>Total proteins: 0</div> <div>Total fat: 0</div> <div>Total carbohydrates: 0</div>	
Dinner	<div>Select a recipe ▾</div> <div>Total kcal: 0</div> <div>Total proteins: 0</div> <div>Total fat: 0</div> <div>Total carbohydrates: 0</div>	<div>Select a recipe ▾</div> <div>Total kcal: 0</div> <div>Total proteins: 0</div> <div>Total fat: 0</div> <div>Total carbohydrates: 0</div>	<div>Select a recipe ▾</div> <div>Total kcal: 0</div> <div>Total proteins: 0</div> <div>Total fat: 0</div> <div>Total carbohydrates: 0</div>	<div>Select a recipe ▾</div> <div>Total kcal: 0</div> <div>Total proteins: 0</div> <div>Total fat: 0</div> <div>Total carbohydrates: 0</div>	<div>Select a recipe ▾</div> <div>Total kcal: 0</div> <div>Total proteins: 0</div> <div>Total fat: 0</div> <div>Total carbohydrates: 0</div>	<div>Select a recipe ▾</div> <div>Total kcal: 0</div> <div>Total proteins: 0</div> <div>Total fat: 0</div> <div>Total carbohydrates: 0</div>	<div>Select a recipe ▾</div> <div>Total kcal: 0</div> <div>Total proteins: 0</div> <div>Total fat: 0</div> <div>Total carbohydrates: 0</div>	
Total								

Add recipe

Show recipes

Refresh Calendar

Calculate Nutritional Values

Save Calendar

Close session

Active Windows

🔍 Configuración para activar Windows

here'. In the bottom right corner, there is a Windows watermark that says 'Activar Windows' and 'Ve a Configuración para activar Windows.'" data-bbox="120 134 942 414"/>

Username:
username

Password:

Login

Doesn't have an account yet? Register [here](#)

Activar Windows
Ve a Configuración para activar Windows.

Vista index.html

Back' link. In the bottom right corner, there is a Windows watermark that says 'Activar Windows' and 'Ve a Configuración para activar Windows.'" data-bbox="120 520 959 721"/>

User:
Username

Email:
example@example.com

Password:

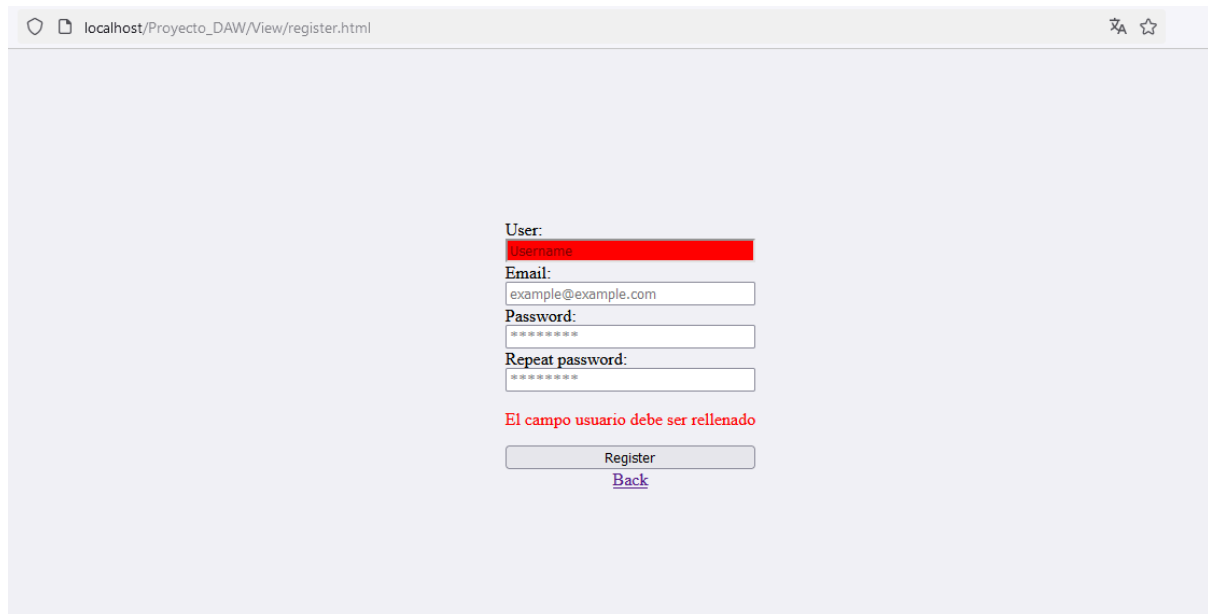
Repeat password:

Register

[Back](#)

Activar Windows
Ve a Configuración para activar Windows.

Vista register.html



localhost/Proyecto_DAW/View/register.html

User:

Email:

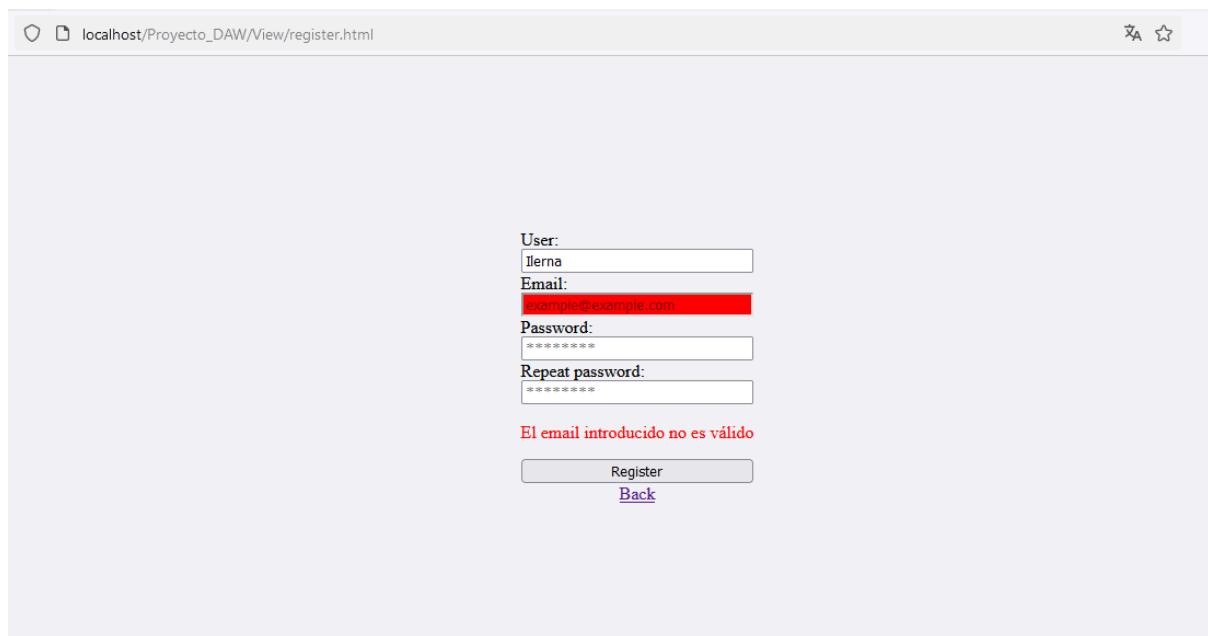
Password:

Repeat password:

El campo usuario debe ser rellenado

[Back](#)

Vista register.html con error por el campo User vacío



localhost/Proyecto_DAW/View/register.html

User:

Email:

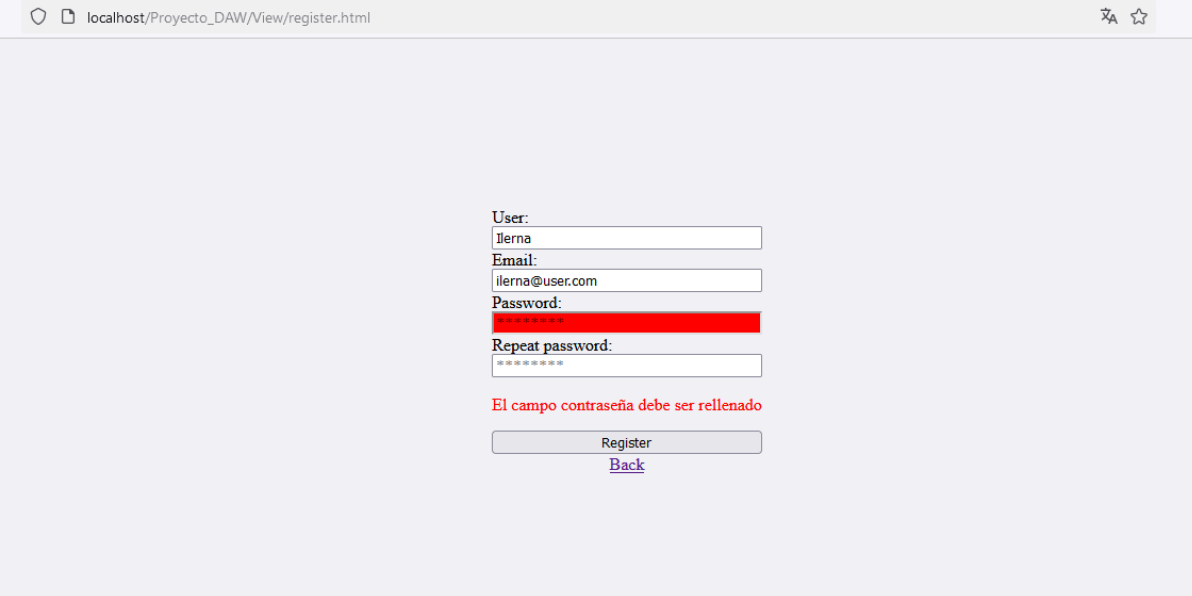
Password:

Repeat password:

El email introducido no es válido

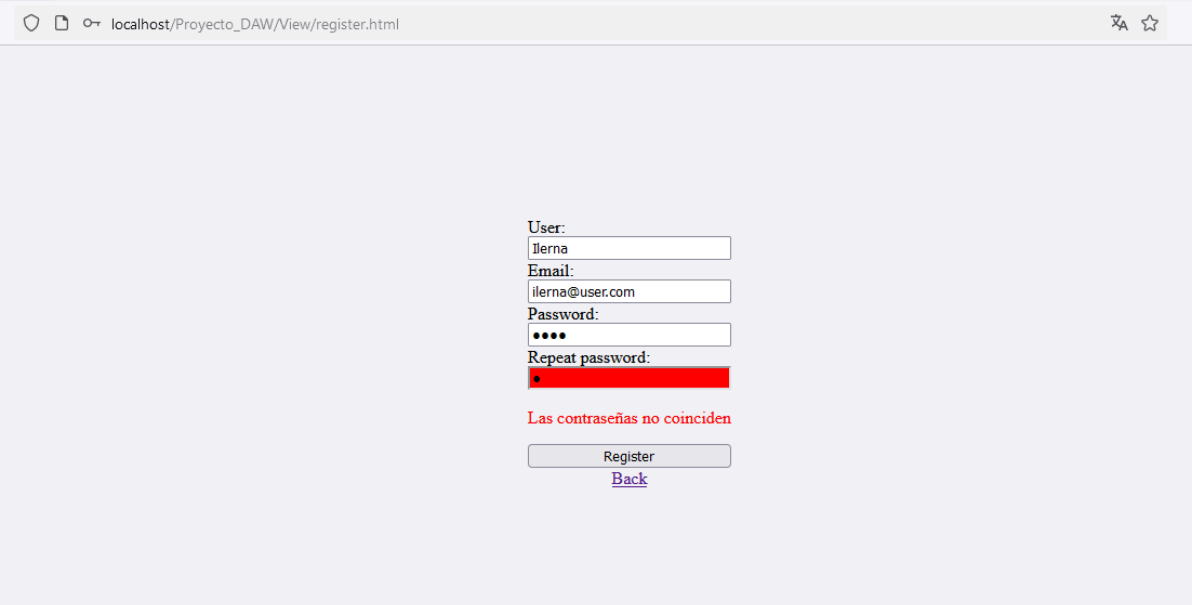
[Back](#)

Vista register.html con error por el email vacío o no válido



A screenshot of a web browser window showing a registration form. The browser's address bar displays 'localhost/Proyecto_DAW/View/register.html'. The form contains the following fields: 'User:' with the value 'ilerna', 'Email:' with the value 'ilerna@user.com', 'Password:' which is highlighted in red, and 'Repeat password:' which is empty. Below the 'Repeat password:' field, a red error message reads 'El campo contraseña debe ser rellenado'. At the bottom of the form are a 'Register' button and a 'Back' link.

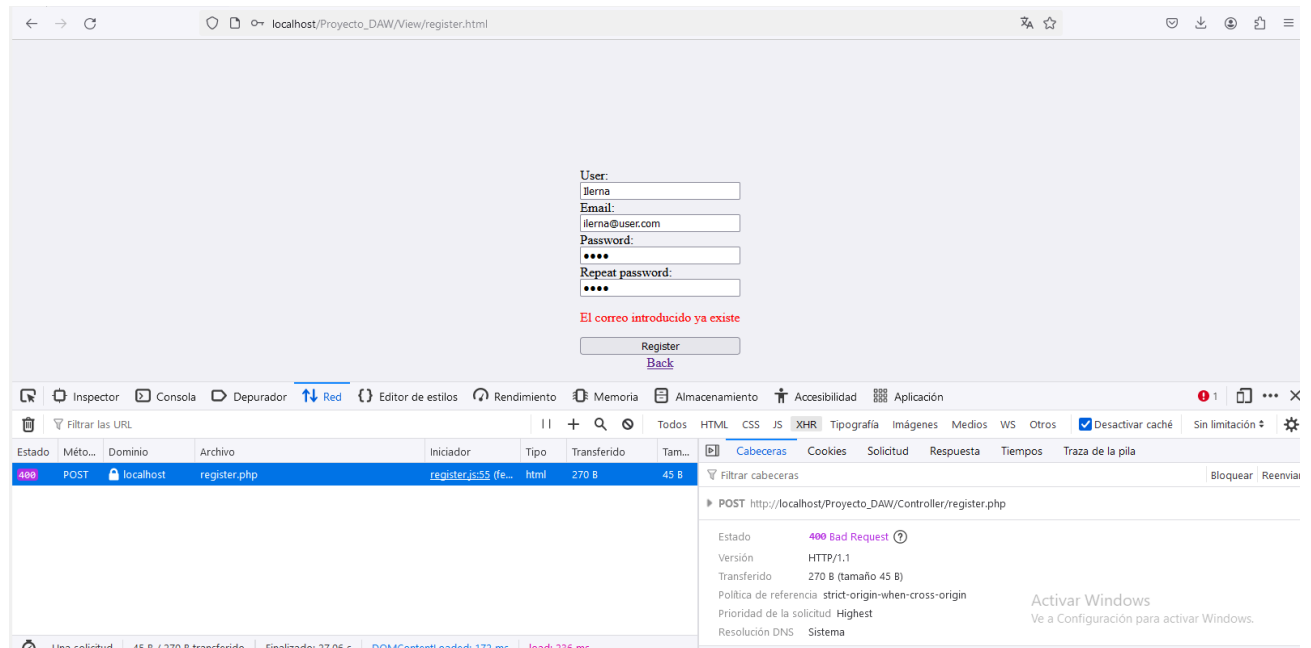
Vista register.html con error por el campo Password



A screenshot of the same web browser window showing the registration form. The 'User:' field contains 'ilerna' and the 'Email:' field contains 'ilerna@user.com'. The 'Password:' field is filled with four dots and the 'Repeat password:' field is filled with a single dot. Both fields are highlighted in red. Below them, a red error message reads 'Las contraseñas no coinciden'. The 'Register' button and 'Back' link are still present at the bottom.

Vista register.html con contraseñas no coincidentes

Dishes At Will (DAW) - Juan José Saborido Barranco



Vista register.html con error del servidor




























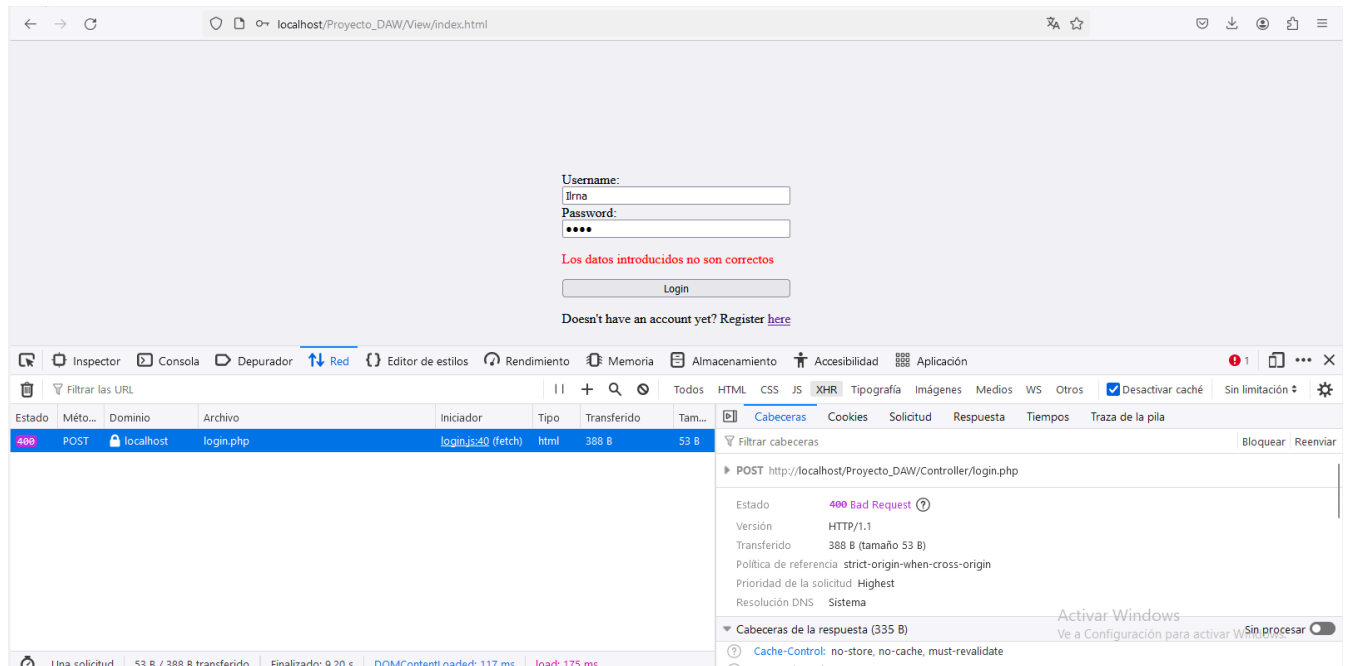
				user_id	username	email	password_user	rol_user
<input type="checkbox"/>	 Editar	 Copiar	 Borrar	1	Juan Jose	juanjo@admin.com	\$2y\$10\$AndCOO7gow7szgYgRkRumOhJrOCuAAItD98AdWsNO1h...	superadmin
<input type="checkbox"/>	 Editar	 Copiar	 Borrar	6	Antonio	antonio@user.com	\$2y\$10\$jAOGMr31Nsl/8gRbzFxQReURYLSIGNOVs4QdVxgjAW...	user
<input type="checkbox"/>	 Editar	 Copiar	 Borrar	12	Alvaro	alvaro@user.com	\$2y\$10\$KKbpoLyg3ctXcsMIBGi8JuN.IQoOyUjQ/duxWA3yCkF...	user
<input type="checkbox"/>	 Editar	 Copiar	 Borrar	16	Lucia	lucia@user.com	\$2y\$10\$BYOazgeZeOUTrCG7FBh8sOyDbPKEy0XQkMBLo5xoV94...	user
<input type="checkbox"/>	 Editar	 Copiar	 Borrar	21	Pepe	pepe@user.com	\$2y\$10\$mRtNZol32XLW0ft.Eu84fuzQwbLHYW2w/BIRe9.cl.d...	banned
<input type="checkbox"/>	 Editar	 Copiar	 Borrar	41	Eto	eto@user.com	\$2y\$10\$jAOGMr31Nsl/8gRbzFxQReURYLSIGNOVs4QdVxgjAW...	user
<input type="checkbox"/>	 Editar	 Copiar	 Borrar	45	Maria	maria@user.com	\$2y\$10\$Yozlcy9L7l6hnw38xTgpu7mw86byMdRX37HGhFPV5g...	user
<input type="checkbox"/>	 Editar	 Copiar	 Borrar	48	Judith	judith@user.com	\$2y\$10\$KUu9f5vHmPlvosM3Z0Uw.3a6oOYGP9HEgpreJMoYQI...	user
<input type="checkbox"/>	 Editar	 Copiar	 Borrar	49	Ilerma	ilerma@user.com	\$2y\$10\$co/9x61alfA3QX.32eSGL.v2h/sj3mii3kmR6FmHCr...	user

Tabla users con datos concretos

Dishes At Will (DAW) - Juan José Saborido Barranco



The screenshot shows a web browser at the URL `localhost/Proyecto_DAW/View/index.html`. The login form has the following fields:

- Username: `ilma`
- Password: `****`

Below the form, a red error message states: **Los datos introducidos no son correctos**. A "Login" button is present, along with a link to "Register here".

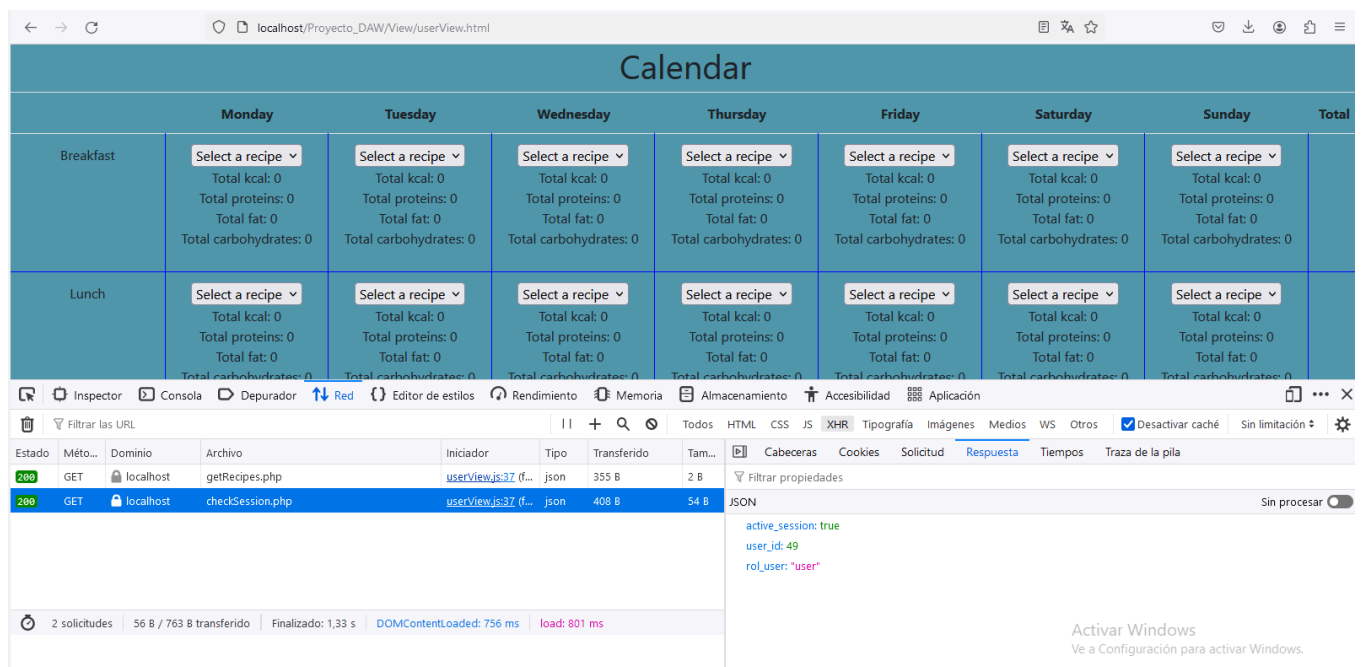
The developer console shows a **400 Bad Request** error for the `POST http://localhost/Proyecto_DAW/Controller/login.php` request. The error details include:

- Estado: **400 Bad Request**
- Versión: HTTP/1.1
- Transferido: 388 B (tamaño 53 B)
- Política de referencia: strict-origin-when-cross-origin
- Prioridad de la solicitud: Highest
- Resolución DNS: Sistema

The console also shows the response headers for the `POST` request:

- Cache-Control: no-store, no-cache, must-revalidate

Vista index.html con datos de login incorrectos



The screenshot shows a web browser at the URL `localhost/Proyecto_DAW/View/userView.html`. The page displays a "Calendar" view with a table of days and meal categories.

	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday	Total
Breakfast	Select a recipe v Total kcal: 0 Total proteins: 0 Total fat: 0 Total carbohydrates: 0	Select a recipe v Total kcal: 0 Total proteins: 0 Total fat: 0 Total carbohydrates: 0	Select a recipe v Total kcal: 0 Total proteins: 0 Total fat: 0 Total carbohydrates: 0	Select a recipe v Total kcal: 0 Total proteins: 0 Total fat: 0 Total carbohydrates: 0	Select a recipe v Total kcal: 0 Total proteins: 0 Total fat: 0 Total carbohydrates: 0	Select a recipe v Total kcal: 0 Total proteins: 0 Total fat: 0 Total carbohydrates: 0	Select a recipe v Total kcal: 0 Total proteins: 0 Total fat: 0 Total carbohydrates: 0	
Lunch	Select a recipe v Total kcal: 0 Total proteins: 0 Total fat: 0 Total carbohydrates: 0	Select a recipe v Total kcal: 0 Total proteins: 0 Total fat: 0 Total carbohydrates: 0	Select a recipe v Total kcal: 0 Total proteins: 0 Total fat: 0 Total carbohydrates: 0	Select a recipe v Total kcal: 0 Total proteins: 0 Total fat: 0 Total carbohydrates: 0	Select a recipe v Total kcal: 0 Total proteins: 0 Total fat: 0 Total carbohydrates: 0	Select a recipe v Total kcal: 0 Total proteins: 0 Total fat: 0 Total carbohydrates: 0	Select a recipe v Total kcal: 0 Total proteins: 0 Total fat: 0 Total carbohydrates: 0	

The developer console shows a **200 OK** response for the `GET http://localhost/Proyecto_DAW/Controller/checkSession.php` request. The response is in JSON format:

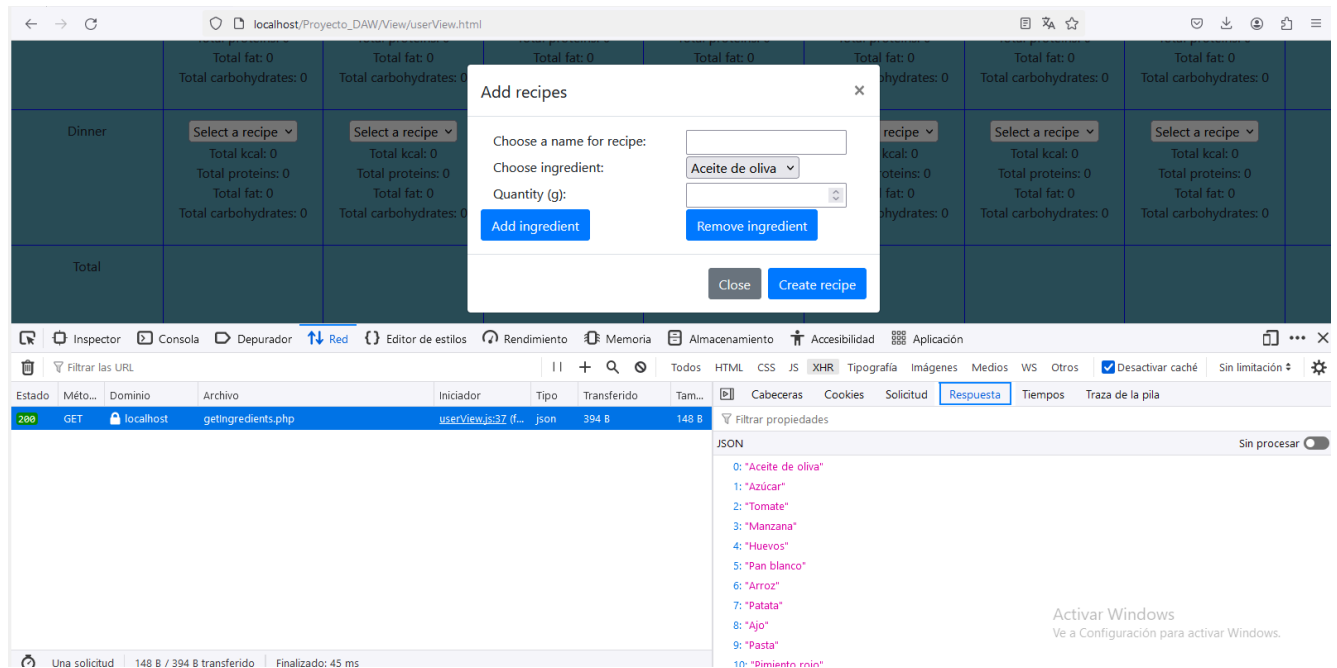
```

{
  "active_session": true,
  "user_id": 49,
  "rol_user": "user"
}

```

Vista userView.html con los datos de respuesta de checkSession.php

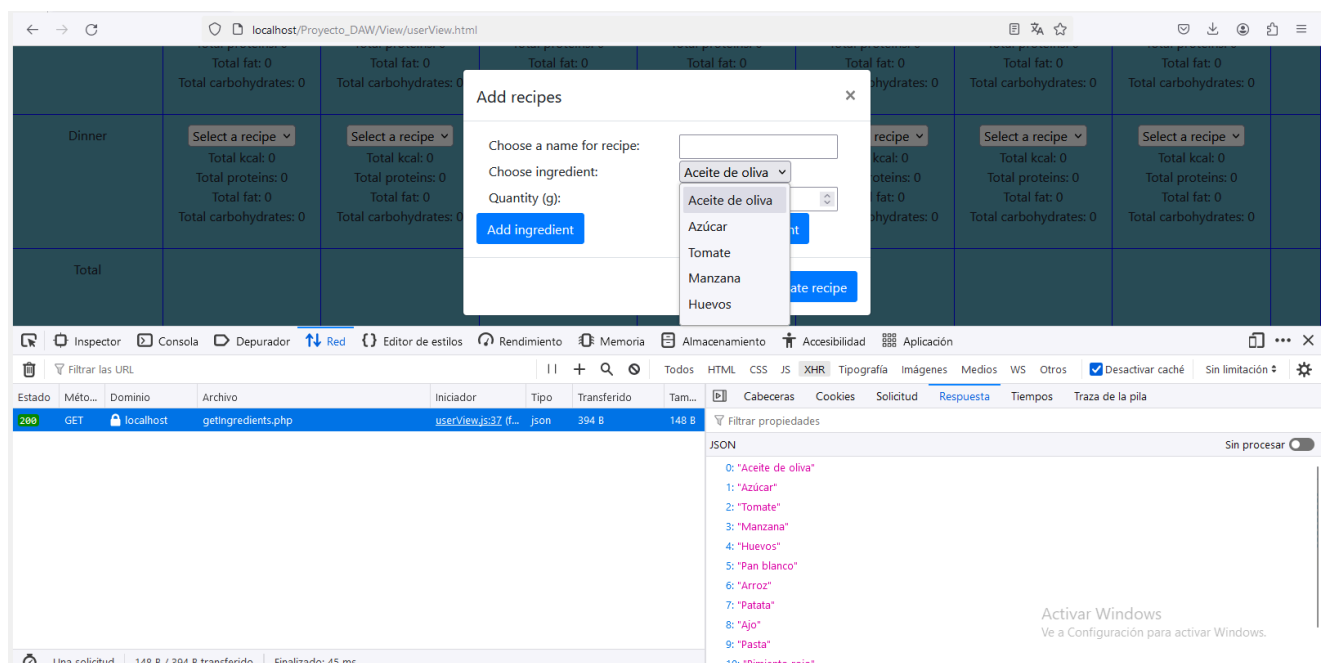
Dishes At Will (DAW) - Juan José Saborido Barranco



The screenshot shows the DAW application interface with a modal titled "Add recipes" open. The modal contains fields for "Choose a name for recipe:", "Choose ingredient:" (with a dropdown menu showing "Aceite de oliva"), and "Quantity (g):". There are buttons for "Add ingredient", "Remove ingredient", "Close", and "Create recipe".

The browser's developer console shows the response from the `getIngredients.php` endpoint, which is a JSON array of ingredients:

```
JSON
0: "Aceite de oliva"
1: "Azúcar"
2: "Tomate"
3: "Manzana"
4: "Huevos"
5: "Pan blanco"
6: "Arroz"
7: "Patata"
8: "Ajo"
9: "Pasta"
10: "Dimiñento rxin"
```

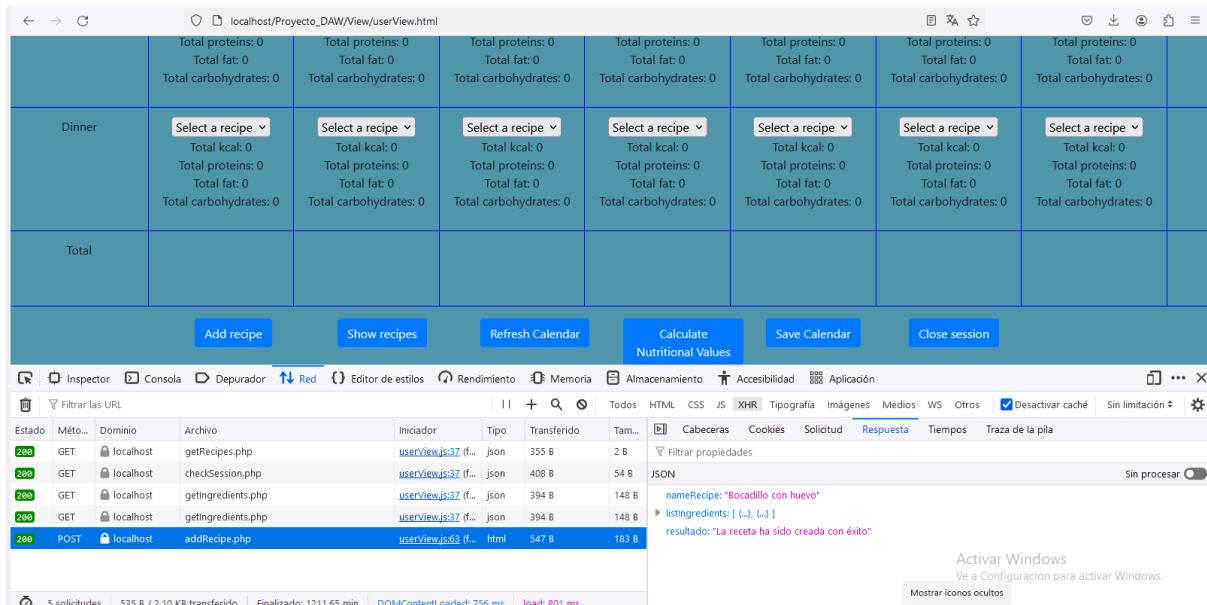


This screenshot shows the same "Add recipes" modal as the previous one, but with the "Choose ingredient:" dropdown menu open. The menu lists the following ingredients: "Aceite de oliva", "Azúcar", "Tomate", "Manzana", and "Huevos".

The browser's developer console shows the same JSON response from `getIngredients.php` as in the previous screenshot.

Modal addRecipe recién desplegado con la respuesta de `getIngredients.php` (menú sin desplegar y menú desplegado respectivamente)

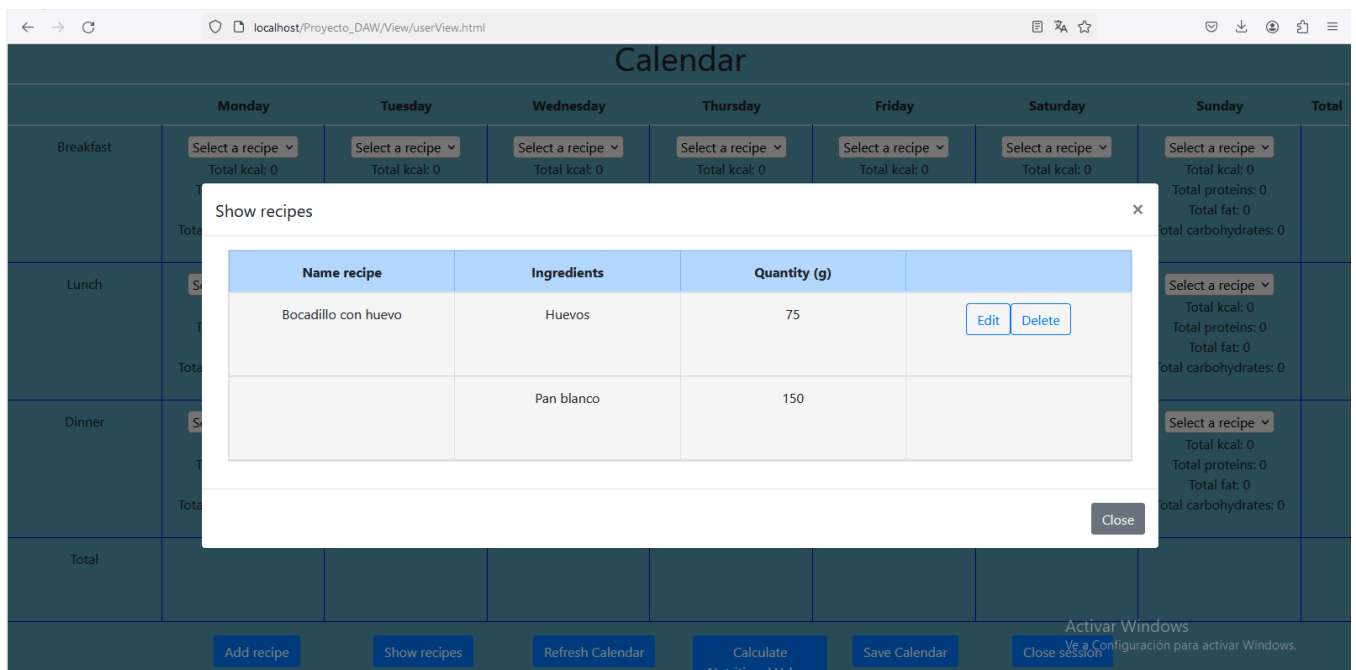
Dishes At Will (DAW) - Juan José Saborido Barranco



The screenshot shows the user view of the DAW application. The main table displays recipes for Breakfast, Lunch, and Dinner, with columns for Total kcal, Total proteins, Total fat, and Total carbohydrates. The 'Add recipe' button is visible at the bottom. The console log shows the response for the 'addRecipe.php' endpoint, indicating that the recipe 'Bocadillo con huevo' was successfully added.

Estado	Método	Dominio	Archivo	Iniciador	Tipo	Transferido	Tamaño	Respuesta
200	GET	localhost	getRecipes.php	userView.js:37 (f...)	json	355 B	2 B	JSON
200	GET	localhost	checkSession.php	userView.js:37 (f...)	json	408 B	54 B	
200	GET	localhost	getIngredients.php	userView.js:37 (f...)	json	394 B	148 B	
200	GET	localhost	getIngredients.php	userView.js:37 (f...)	json	394 B	148 B	
200	POST	localhost	addRecipe.php	userView.js:63 (f...)	html	547 B	183 B	<pre> nameRecipe: "Bocadillo con huevo" listingredients: [(-), (-)] resultado: "La receta ha sido creada con éxito" </pre>

Respuesta del backend después de añadir receta "Bocadillo con huevo"

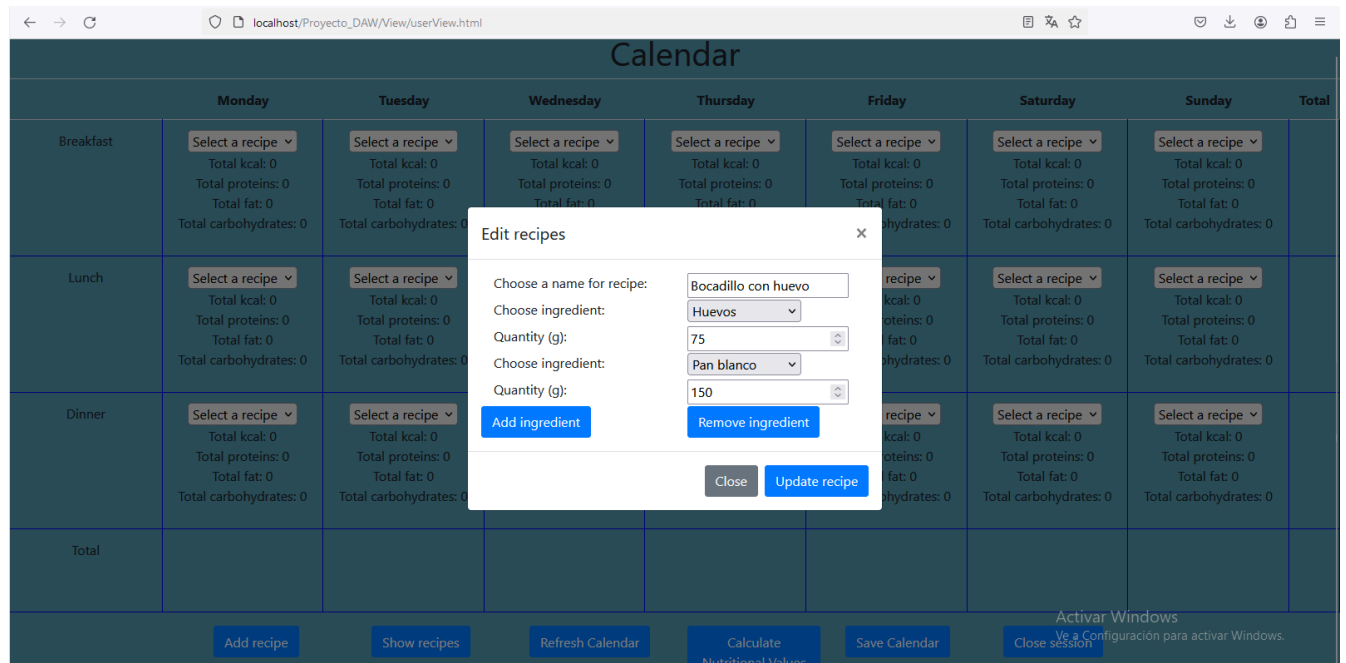


The screenshot shows the user view of the DAW application with the 'Show recipes' modal active. The modal displays a table of recipes with columns for Name recipe, Ingredients, and Quantity (g). The recipe 'Bocadillo con huevo' is shown with ingredients 'Huevos' (75g) and 'Pan blanco' (150g). The 'Edit' and 'Delete' buttons are visible for each recipe.

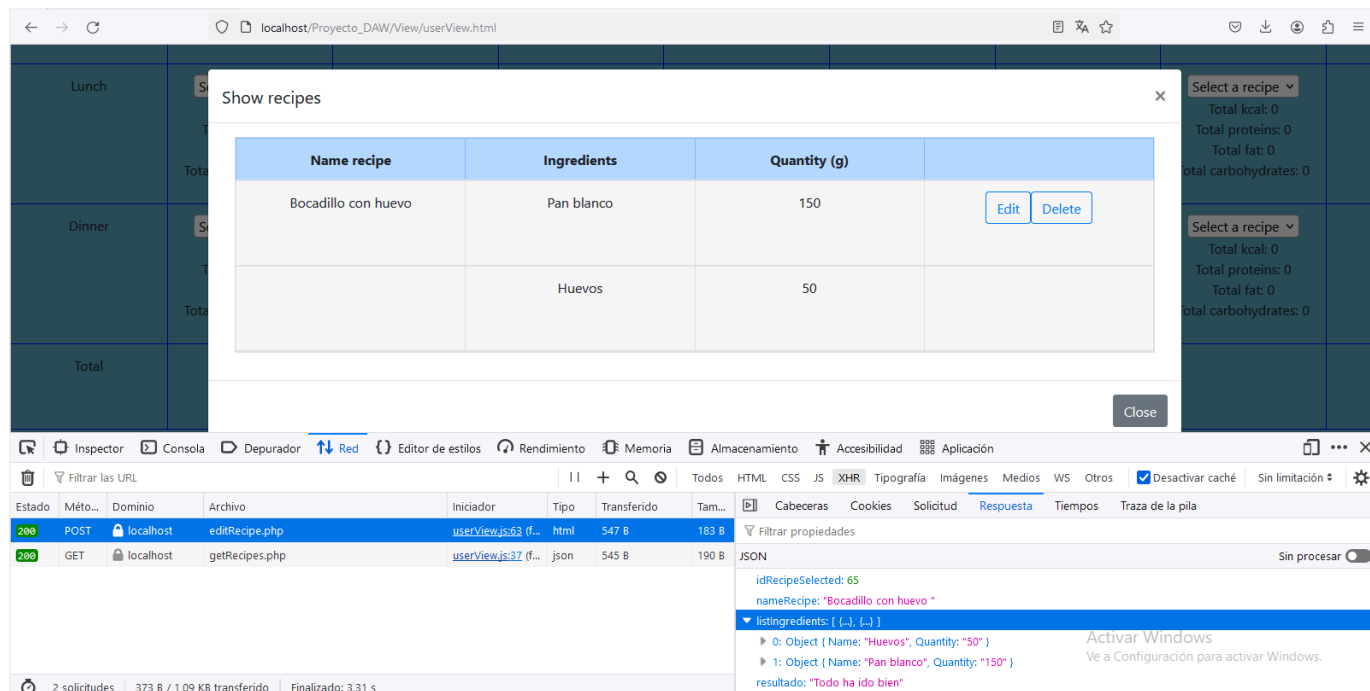
Name recipe	Ingredients	Quantity (g)
Bocadillo con huevo	Huevos	75
	Pan blanco	150

Vista userView.html con el modal showRecipes activado

Dishes At Will (DAW) - Juan José Saborido Barranco

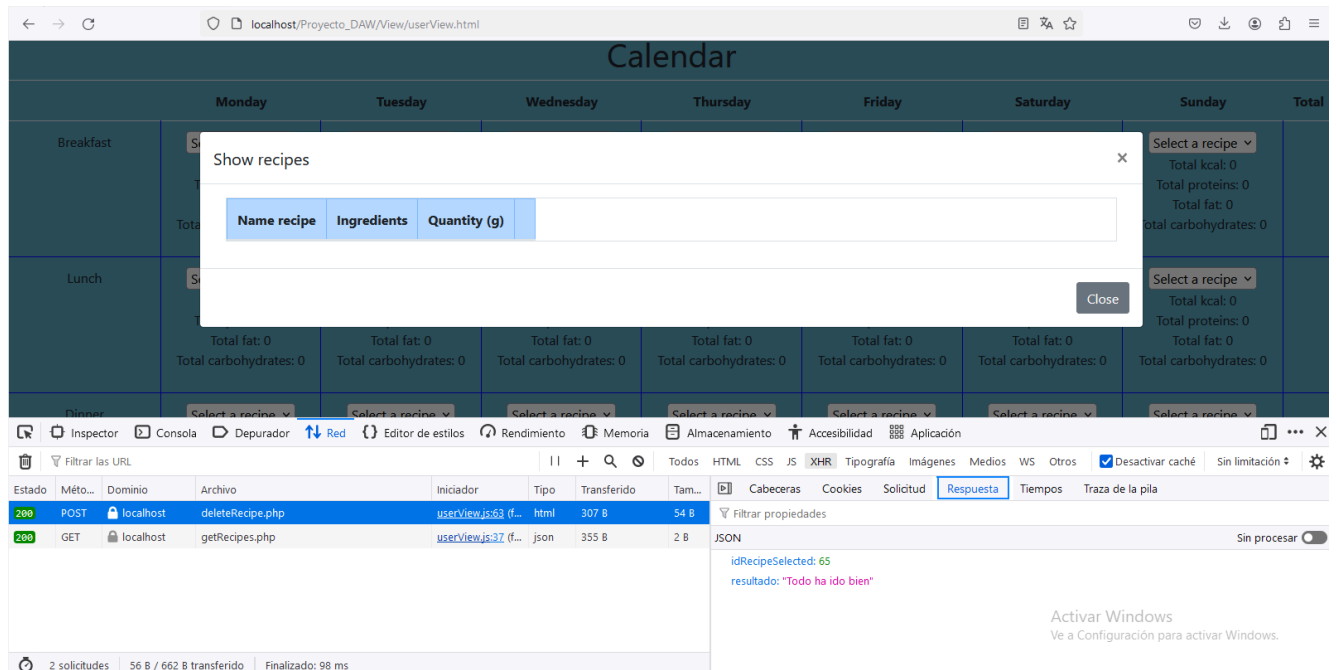


Modal editRecipes con los datos de la receta seleccionada



Modal showRecipes con la respuesta después de haber actualizado la receta

Dishes At Will (DAW) - Juan José Saborido Barranco



Calendar

Monday Tuesday Wednesday Thursday Friday Saturday Sunday Total

Breakfast

Lunch

Dinner

Show recipes

Name recipe Ingredients Quantity (g)

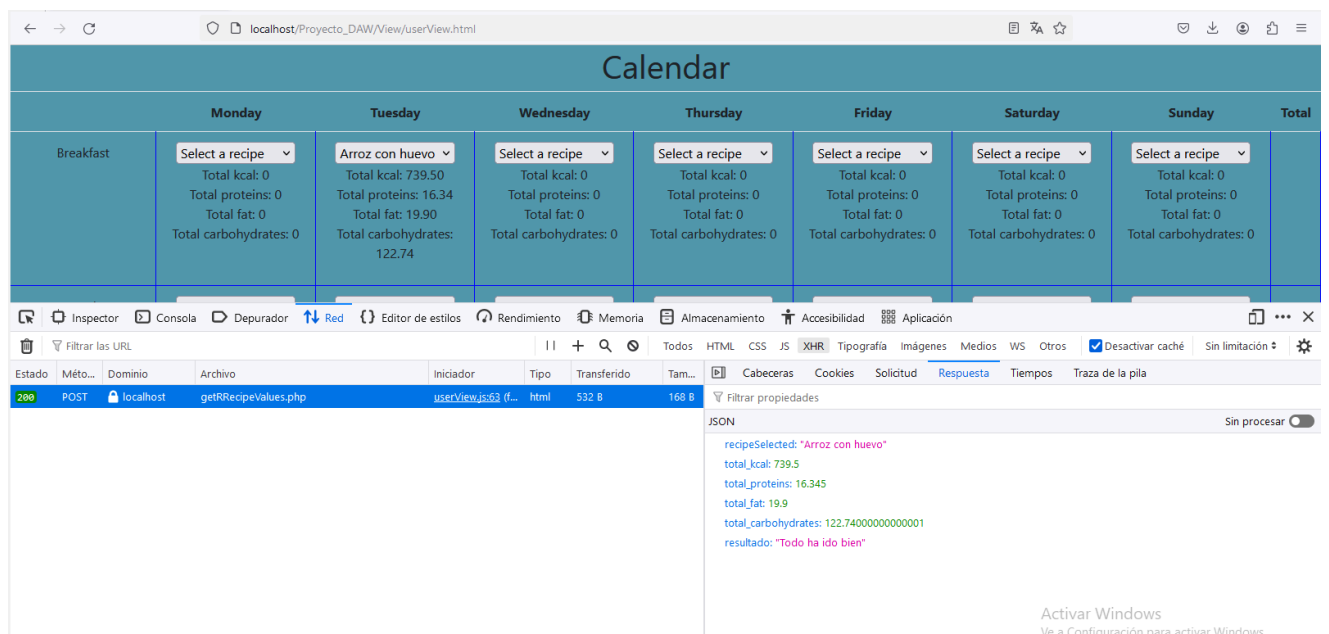
Close

2 solicitudes 56 B / 662 B transferido Finalizado: 98 ms

```

{
  "idRecipeSelected": 65,
  "resultado": "Todo ha ido bien"
}
```

Modal showRecipes después de haber eliminado la receta



Calendar

Monday Tuesday Wednesday Thursday Friday Saturday Sunday Total

Breakfast

Arroz con huevo

Total kcal: 739.50
Total proteins: 16.34
Total fat: 19.90
Total carbohydrates: 122.74

2 solicitudes 532 B / 168 B transferido Finalizado: 10 ms

```

{
  "recipeSelected": "Arroz con huevo",
  "total_kcal": 739.5,
  "total_proteins": 16.345,
  "total_fat": 19.9,
  "total_carbohydrates": 122.74000000000001,
  "resultado": "Todo ha ido bien"
}
```

Resultado al elegir Arroz con huevo para desayunar el martes

Dishes At Will (DAW) - Juan José Saborido Barranco

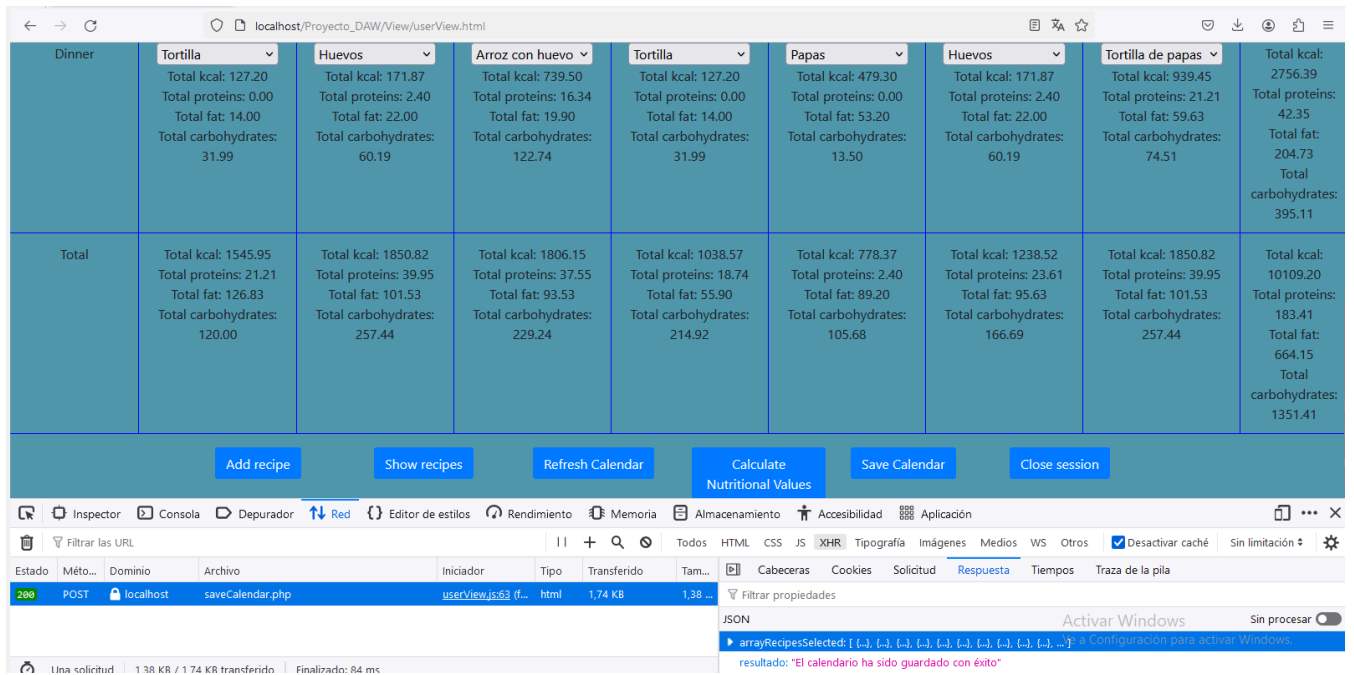
	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday	Total
Breakfast	Tortilla de papas Total kcal: 939.45 Total proteins: 21.21 Total fat: 59.63 Total carbohydrates: 74.51	Arroz con huevo Total kcal: 739.50 Total proteins: 16.34 Total fat: 19.90 Total carbohydrates: 122.74	Tortilla de papas Total kcal: 939.45 Total proteins: 21.21 Total fat: 59.63 Total carbohydrates: 74.51	Huevos Total kcal: 171.87 Total proteins: 2.40 Total fat: 22.00 Total carbohydrates: 60.19	Tortilla Total kcal: 127.20 Total proteins: 0.00 Total fat: 14.00 Total carbohydrates: 31.99	Tortilla de papas Total kcal: 939.45 Total proteins: 21.21 Total fat: 59.63 Total carbohydrates: 74.51	Arroz con huevo Total kcal: 739.50 Total proteins: 16.34 Total fat: 19.90 Total carbohydrates: 122.74	
Lunch	Papas Total kcal: 479.30 Total proteins: 0.00 Total fat: 53.20 Total carbohydrates: 13.50	Tortilla de papas Total kcal: 939.45 Total proteins: 21.21 Total fat: 59.63 Total carbohydrates: 74.51	Tortilla Total kcal: 127.20 Total proteins: 0.00 Total fat: 14.00 Total carbohydrates: 31.99	Arroz con huevo Total kcal: 739.50 Total proteins: 16.34 Total fat: 19.90 Total carbohydrates: 122.74	Huevos Total kcal: 171.87 Total proteins: 2.40 Total fat: 22.00 Total carbohydrates: 60.19	Tortilla Total kcal: 127.20 Total proteins: 0.00 Total fat: 14.00 Total carbohydrates: 31.99	Huevos Total kcal: 171.87 Total proteins: 2.40 Total fat: 22.00 Total carbohydrates: 60.19	
Dinner	Tortilla Total kcal: 127.20 Total proteins: 0.00 Total fat: 14.00 Total carbohydrates: 31.99	Huevos Total kcal: 171.87 Total proteins: 2.40 Total fat: 22.00 Total carbohydrates: 60.19	Arroz con huevo Total kcal: 739.50 Total proteins: 16.34 Total fat: 19.90 Total carbohydrates: 122.74	Tortilla Total kcal: 127.20 Total proteins: 0.00 Total fat: 14.00 Total carbohydrates: 31.99	Papas Total kcal: 479.30 Total proteins: 0.00 Total fat: 53.20 Total carbohydrates: 13.50	Huevos Total kcal: 171.87 Total proteins: 2.40 Total fat: 22.00 Total carbohydrates: 60.19	Tortilla de papas Total kcal: 939.45 Total proteins: 21.21 Total fat: 59.63 Total carbohydrates: 74.51	

Vista antes de clicar en el botón “Calculate nutritional values”

Calendar								
	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday	Total
Breakfast	Tortilla de papas Total kcal: 939.45 Total proteins: 21.21 Total fat: 59.63 Total carbohydrates: 74.51	Arroz con huevo Total kcal: 739.50 Total proteins: 16.34 Total fat: 19.90 Total carbohydrates: 122.74	Tortilla de papas Total kcal: 939.45 Total proteins: 21.21 Total fat: 59.63 Total carbohydrates: 74.51	Huevos Total kcal: 171.87 Total proteins: 2.40 Total fat: 22.00 Total carbohydrates: 60.19	Tortilla Total kcal: 127.20 Total proteins: 0.00 Total fat: 14.00 Total carbohydrates: 31.99	Tortilla de papas Total kcal: 939.45 Total proteins: 21.21 Total fat: 59.63 Total carbohydrates: 74.51	Arroz con huevo Total kcal: 739.50 Total proteins: 16.34 Total fat: 19.90 Total carbohydrates: 122.74	Total kcal: 4596.42 Total proteins: 98.71 Total fat: 254.69 Total carbohydrates: 561.19
Lunch	Papas Total kcal: 479.30 Total proteins: 0.00 Total fat: 53.20 Total carbohydrates: 13.50	Tortilla de papas Total kcal: 939.45 Total proteins: 21.21 Total fat: 59.63 Total carbohydrates: 74.51	Tortilla Total kcal: 127.20 Total proteins: 0.00 Total fat: 14.00 Total carbohydrates: 31.99	Arroz con huevo Total kcal: 739.50 Total proteins: 16.34 Total fat: 19.90 Total carbohydrates: 122.74	Huevos Total kcal: 171.87 Total proteins: 2.40 Total fat: 22.00 Total carbohydrates: 60.19	Tortilla Total kcal: 127.20 Total proteins: 0.00 Total fat: 14.00 Total carbohydrates: 31.99	Huevos Total kcal: 171.87 Total proteins: 2.40 Total fat: 22.00 Total carbohydrates: 60.19	Total kcal: 2756.39 Total proteins: 42.35 Total fat: 204.73 Total carbohydrates: 395.11
Dinner	Tortilla Total kcal: 127.20 Total proteins: 0.00 Total fat: 14.00 Total carbohydrates: 31.99	Huevos Total kcal: 171.87 Total proteins: 2.40 Total fat: 22.00 Total carbohydrates: 60.19	Arroz con huevo Total kcal: 739.50 Total proteins: 16.34 Total fat: 19.90 Total carbohydrates: 122.74	Tortilla Total kcal: 127.20 Total proteins: 0.00 Total fat: 14.00 Total carbohydrates: 31.99	Papas Total kcal: 479.30 Total proteins: 0.00 Total fat: 53.20 Total carbohydrates: 13.50	Huevos Total kcal: 171.87 Total proteins: 2.40 Total fat: 22.00 Total carbohydrates: 60.19	Tortilla de papas Total kcal: 939.45 Total proteins: 21.21 Total fat: 59.63 Total carbohydrates: 74.51	Total kcal: 2756.39 Total proteins: 42.35 Total fat: 204.73 Total carbohydrates: 395.11
Total	Total kcal: 1545.95 Total proteins: 21.21 Total fat: 126.83 Total carbohydrates: 120.00	Total kcal: 1850.82 Total proteins: 39.95 Total fat: 101.53 Total carbohydrates: 257.44	Total kcal: 1806.15 Total proteins: 37.55 Total fat: 93.53 Total carbohydrates: 229.24	Total kcal: 1038.57 Total proteins: 18.74 Total fat: 55.90 Total carbohydrates: 214.92	Total kcal: 778.37 Total proteins: 2.40 Total fat: 89.20 Total carbohydrates: 105.68	Total kcal: 1238.52 Total proteins: 23.61 Total fat: 95.63 Total carbohydrates: 166.69	Total kcal: 1850.82 Total proteins: 39.95 Total fat: 101.53 Total carbohydrates: 257.44	Total kcal: 10109.20 Total proteins: 183.41 Total fat: 664.15 Total carbohydrates: 1351.41

Vista después de clicar en el botón “Calculate nutritional values”

Dishes At Will (DAW) - Juan José Saborido Barranco




JSON response from the browser console:

```

{
  "arrayRecipesSelected": [
    { "id": 1, "recipe": "Tortilla", "kcal": 127.2, "proteins": 0.0, "fat": 14.0, "carbs": 31.99 },
    { "id": 2, "recipe": "Huevos", "kcal": 171.87, "proteins": 2.4, "fat": 22.0, "carbs": 60.19 },
    { "id": 3, "recipe": "Arroz con huevo", "kcal": 739.5, "proteins": 16.34, "fat": 19.9, "carbs": 122.74 },
    { "id": 4, "recipe": "Tortilla", "kcal": 127.2, "proteins": 0.0, "fat": 14.0, "carbs": 31.99 },
    { "id": 5, "recipe": "Papas", "kcal": 479.3, "proteins": 0.0, "fat": 53.2, "carbs": 13.5 },
    { "id": 6, "recipe": "Huevos", "kcal": 171.87, "proteins": 2.4, "fat": 22.0, "carbs": 60.19 },
    { "id": 7, "recipe": "Tortilla de papas", "kcal": 939.45, "proteins": 21.21, "fat": 59.63, "carbs": 74.51 }
  ]
}
  
```

Respuesta con el calendario guardado



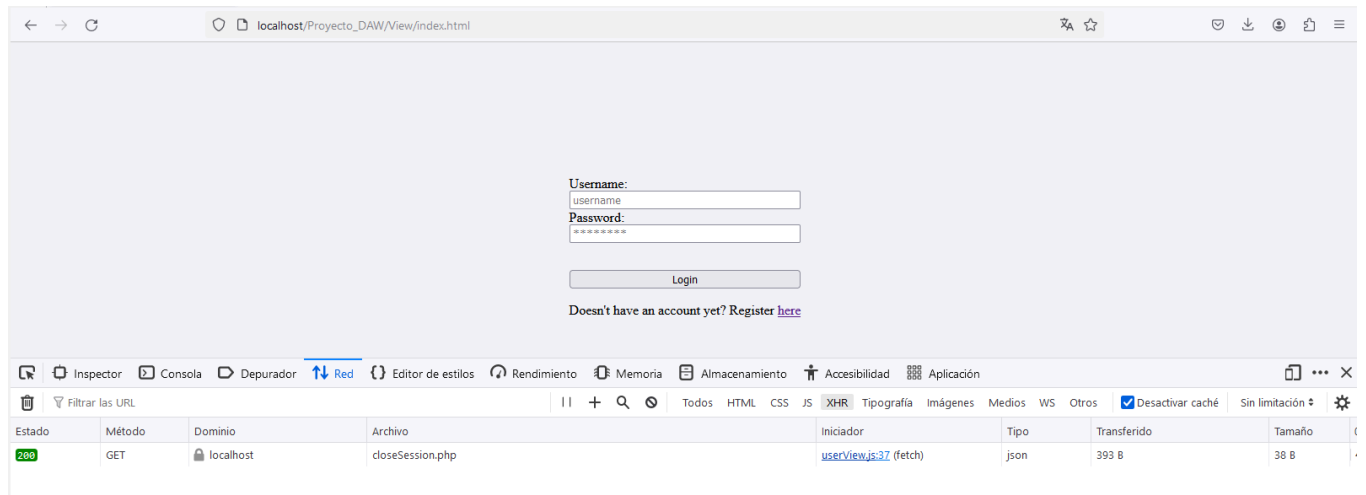
SQL Query: `SELECT * FROM `calendar``

Result:

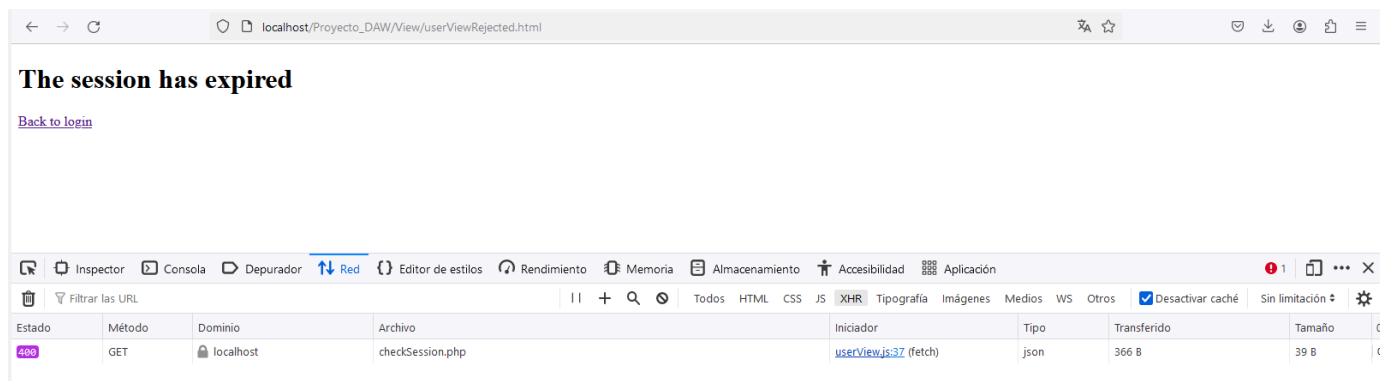
calendar_id	user_id_fk	monday_breakfast_recipe_fk	monday_lunch_recipe_fk	monday_dinner_recipe_fk	tuesday_breakfast_recipe_fk	tuesday_lunch_recipe_fk
2	16	23	63	23	14	13

Calendario insertado/actualizado en la base de datos con los ids respectivos tanto del usuario como de las recetas

Dishes At Will (DAW) - Juan José Saborido Barranco



Petición al archivo `closeSession.php` para cerrar la sesión



Vista `userViewRejected.html`

Dishes At Will (DAW) - Juan José Saborido Barranco

← → ↻ localhost/Proyecto_DAW/View/superadmin.html ☆ ⌵ ⌵ ⌵ ⌵ ⌵

Welcome, superadmin

id	username	email	rol user	action
6	Antonio	antonio@user.com	user	<button>Ban user</button>
12	Alvaro	alvaro@user.com	user	<button>Ban user</button>
16	Lucia	lucia@user.com	user	<button>Ban user</button>
21	Pepe	pepe@user.com	banned	<button>Ban user</button>
41	Eto	eto@user.com	user	<button>Ban user</button>
45	Maria	maria@user.com	user	<button>Ban user</button>
48	Judith	judith@user.com	user	<button>Ban user</button>
49	Ilerna	ilerna@user.com	user	<button>Ban user</button>

Close session

Inspector Console Depurador **Red** Editor de estilos Rendimiento Memoria Almacenamiento Accesibilidad Aplicación

Filtrar las URL

Estado	Método	Dominio	Archivo	Iniciador	Tipo	Transferido	Tamaño	Acciones
200	GET	localhost	checkSession.php	superadmin.js:6	json	413 B	59 B	
200	GET	localhost	getUsers.php	superadmin.js:6	json	867 B	622 B	

Filtrar propiedades

JSON

```

0: Object (user_id: 6, username: "Antonio", email: "antonio@user.com", ...)
1: Object (user_id: 12, username: "Alvaro", email: "alvaro@user.com", ...)
2: Object (user_id: 16, username: "Lucia", email: "lucia@user.com", ...)
3: Object (user_id: 21, username: "Pepe", email: "pepe@user.com", ...)
4: Object (user_id: 41, username: "Eto", email: "eto@user.com", ...)
5: Object (user_id: 45, username: "Maria", email: "maria@user.com", ...)
6: Object (user_id: 48, username: "Judith", email: "judith@user.com", ...)

```

2 solicitudes 681 B / 1.28 KB transferido Finalizado: 387 ms DOMContentLoaded: 312 ms load: 333 ms

Vista superadmin.html con getUsers.php

← → ↻ localhost/Proyecto_DAW/View/superadmin.html ☆ ⌵ ⌵ ⌵ ⌵ ⌵

Welcome, superadmin

id	username	email	rol user	action
6	Antonio	antonio@user.com	user	<button>Ban user</button>
12	Alvaro	alvaro@user.com	user	<button>Ban user</button>
16	Lucia	lucia@user.com	user	<button>Ban user</button>
21	Pepe	pepe@user.com	banned	<button>Ban user</button>
41	Eto	eto@user.com	user	<button>Ban user</button>
45	Maria	maria@user.com	user	<button>Ban user</button>
48	Judith	judith@user.com	user	<button>Ban user</button>
49	Ilerna	ilerna@user.com	banned	<button>Ban user</button>

Close session

Inspector Console Depurador **Red** Editor de estilos Rendimiento Memoria Almacenamiento Accesibilidad Aplicación

Filtrar las URL

Estado	Método	Dominio	Archivo	Iniciador	Tipo	Transferido	Tamaño	Acciones
200	POST	localhost	banUser.php	superadmin.js:3	html	299 B	46 B	

Filtrar propiedades

JSON

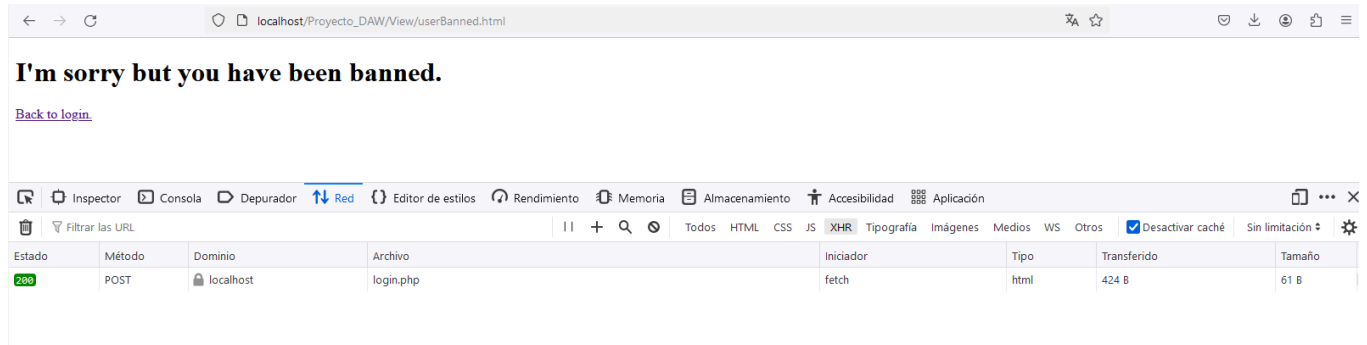
```

idUser: "49"
resultado: "Todo ha ido bien"

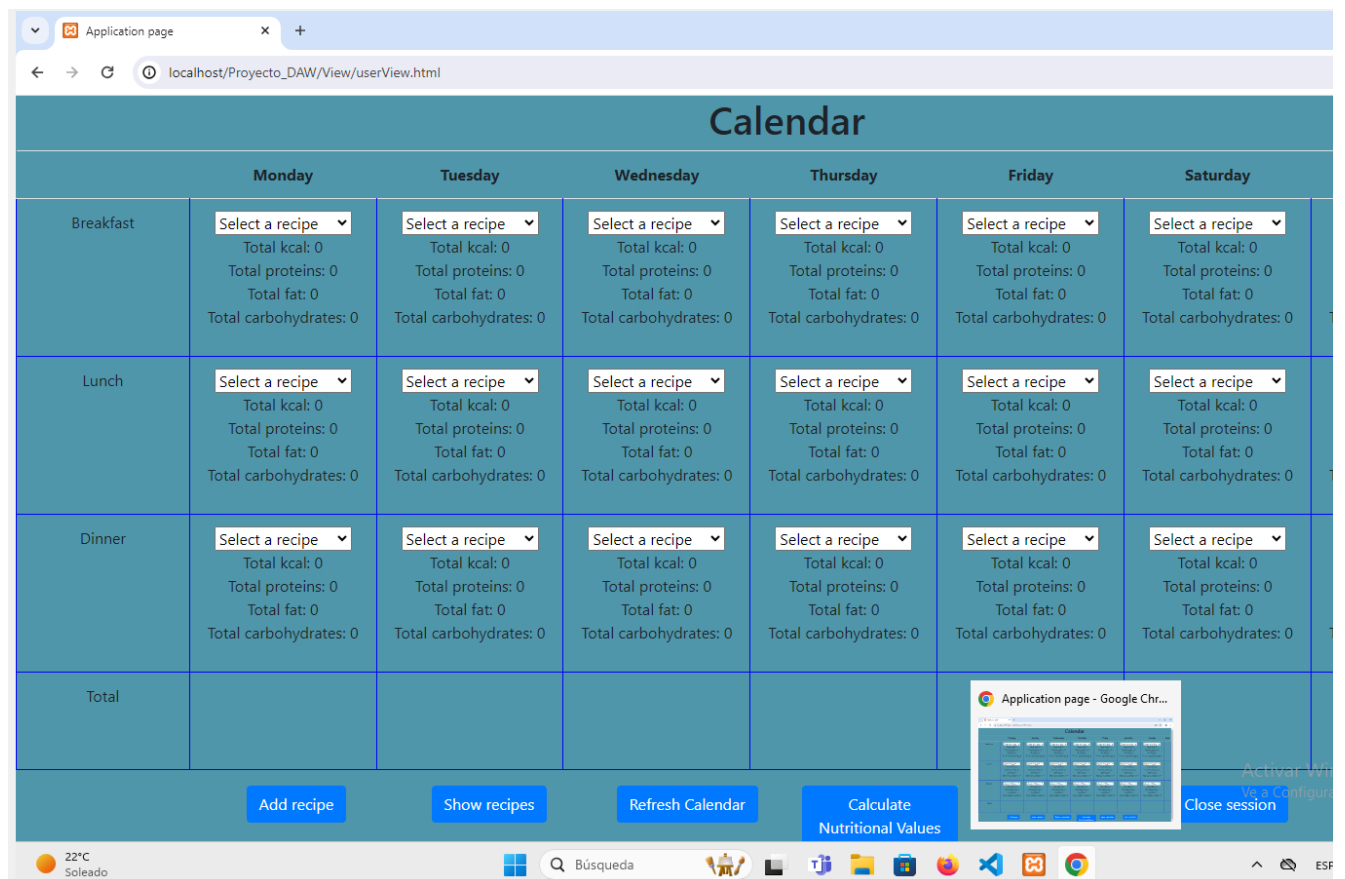
```

Vista después de banear al usuario Ilerna

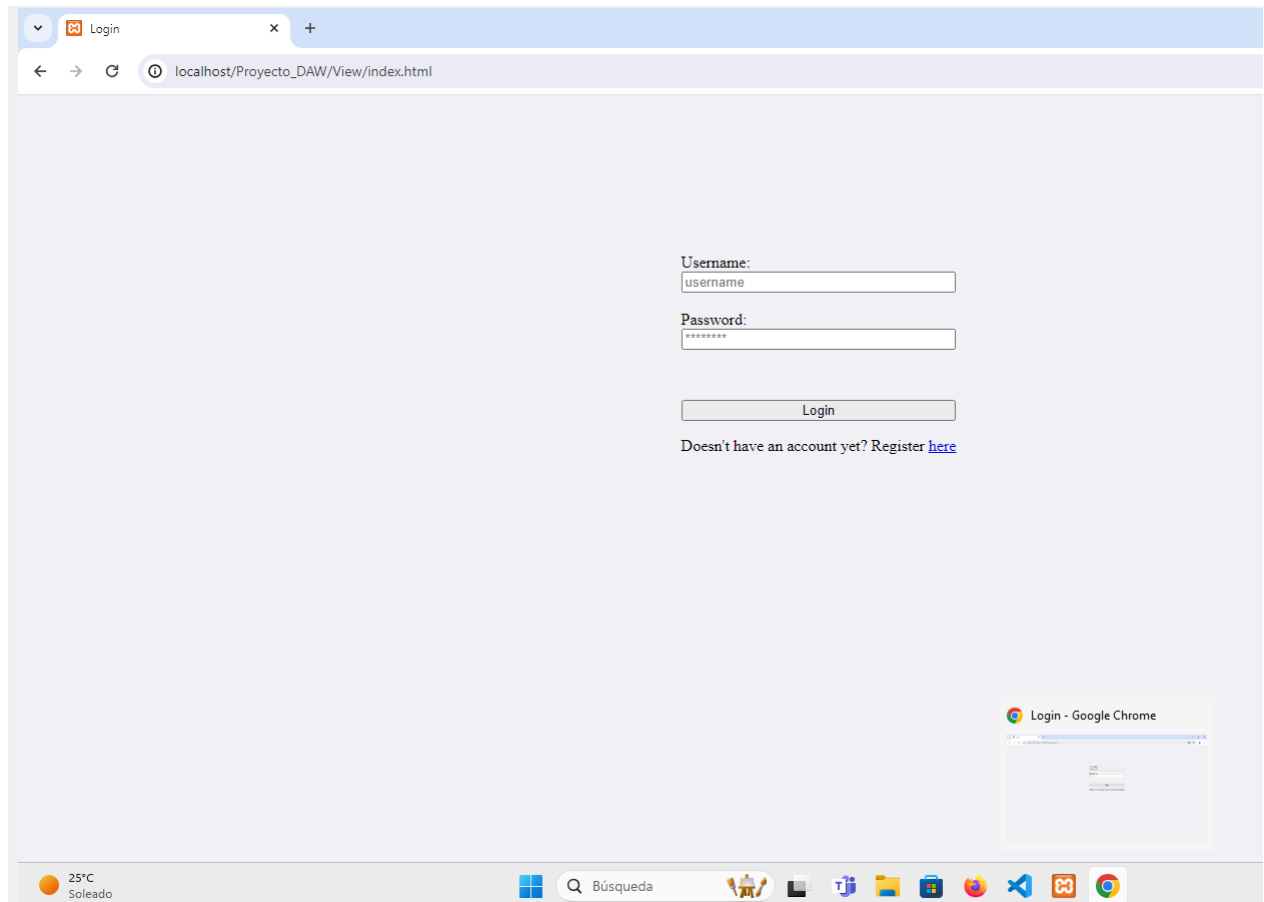
Dishes At Will (DAW) - Juan José Saborido Barranco



Vista userBanned.html

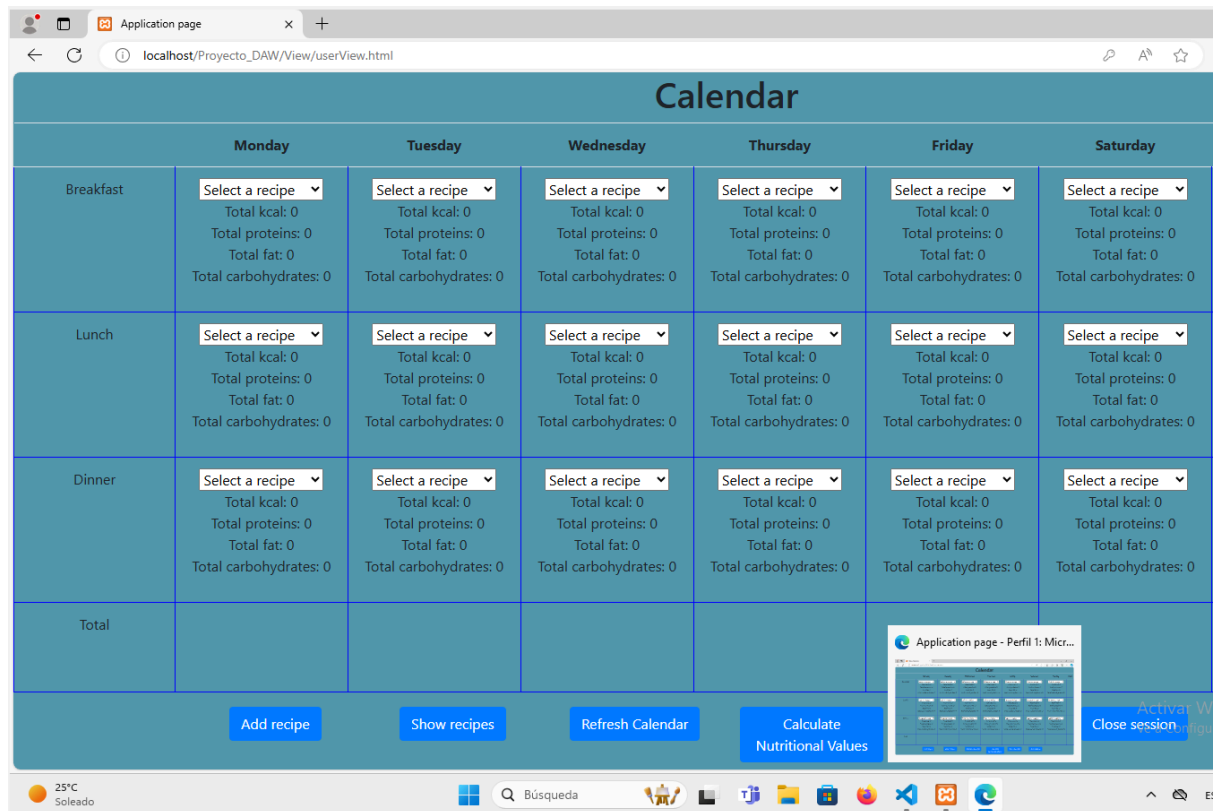


Vista userView.html en google chrome



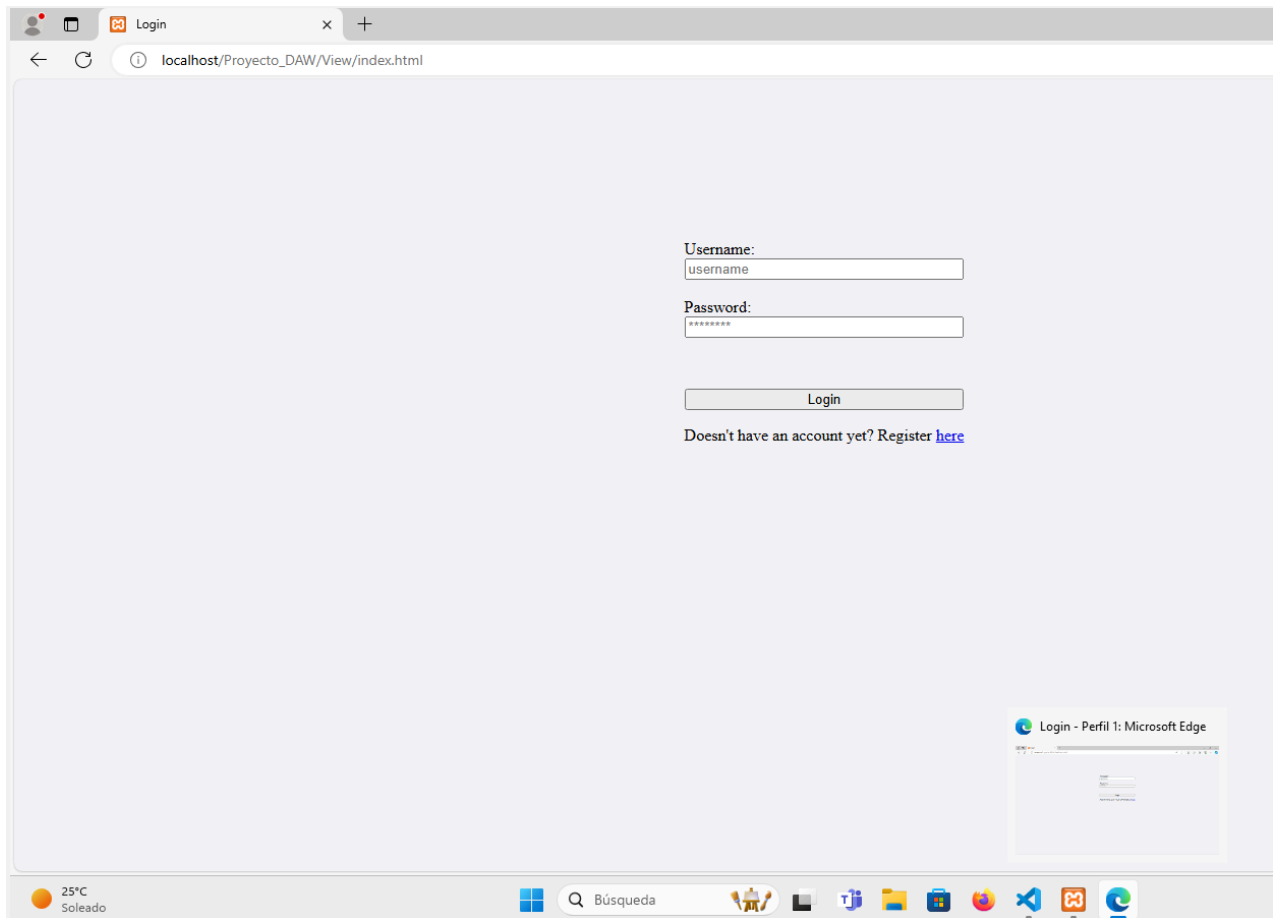
Vista index.html en google chrome

Dishes At Will (DAW) - Juan José Saborido Barranco



The screenshot shows a web application running in Microsoft Edge. The browser address bar displays `localhost/Proyecto_DAW/View/userView.html`. The application title is "Calendar". The main content area is a table with columns for days of the week (Monday to Saturday) and rows for meals (Breakfast, Lunch, Dinner, and Total). Each meal row contains a "Select a recipe" dropdown menu and nutritional information (Total kcal, Total proteins, Total fat, Total carbohydrates). The bottom of the application features a row of buttons: "Add recipe", "Show recipes", "Refresh Calendar", "Calculate Nutritional Values", and "Close session". A small window titled "Application page - Perfil 1: Micr..." is visible in the background.

Vista userView.html en microsoft edge



Vista index.html en microsoft edge

1.1. Manual de Usuario

Para poder levantar esta aplicación, ha sido necesario el uso del paquete XAMPP, como bien se ha comentado y, con Apache y PhpMyAdmin inicializados, desde PhpMyAdmin, se ha de “Importar” la base de datos proporcionada en la carpeta “Model” del proyecto y el proyecto descargado debe copiarse a la carpeta htdocs de la carpeta xampp y, una vez hecho esto, desde cualquier navegador, accedemos a la dirección localhost/Proyecto_DAW/View/index.html (asumiendo que esa sea la ruta del proyecto y que tenga ese nombre) y podremos observar la vista inicial de nuestro proyecto.

Una vez aquí, el usuario puede sentirse libre de testear todo lo que desee la aplicación para comprobar que todo funciona como es debido, recordándole que si quiere usar los usuarios ya creados en la base de datos a modo de prueba, todos tienen como contraseña “1234” salvo el usuario “Juan José”, que por algo tiene el rol de superadmin, usa la contraseña “asdf” y más allá de esto, recordarle que haga un uso responsable de la aplicación y que la disfrute cuanto desee.

Si desea ponerse en contacto conmigo para comentar cualquier cuestión referente a este proyecto, siéntase libre de escribirme tanto al correo del campus como a mi correo personal.

Un saludo y gracias por todo.