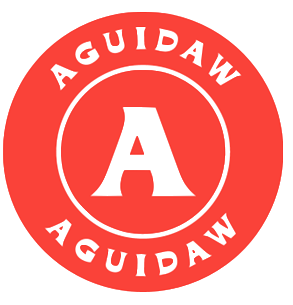
version 1.0

25/11/2022

Erguibide S.l



by: IÑIGO BRUK, GORKA URIARTE, UNAI DÍEZ, Ameer hamza

aguidaw

indice

[RETO2 3](#_Toc120275499)

[Organización del proyecto 4](#_Toc120275500)

[Tareas vistas para realizar el proyecto 4](#_Toc120275501)

[Planificacion temporal 4](#_Toc120275502)

[Especificaciones de requisitos 7](#_Toc120275503)

[Catalogo de requerimiento funcionales 8](#_Toc120275504)

[Catalogo de requerimiento No-funcionales 10](#_Toc120275505)

[Rangos de implementaciones 11](#_Toc120275506)

[Diseño y organizacion de la pagina web 11](#_Toc120275507)

[Prototipado / maquetacion basica la interfaz de las paginas web 11](#_Toc120275508)

[Contenido de la interfaz web 16](#_Toc120275509)

[Metodología del despliegue del entorno 17](#_Toc120275510)

[Modelo de datos 18](#_Toc120275511)

[Contenido del entorno tecnologico (como se controla la pagina web) 21](#_Toc120275512)

[plantilla de las paginas web 21](#_Toc120275513)

[css 21](#_Toc120275514)

[cotrol de imágenes/archivos 22](#_Toc120275515)

[entorno cliente: java script 22](#_Toc120275516)

[php 24](#_Toc120275517)

[Contenido del control-modelo de datos 25](#_Toc120275518)

[Consideraciones sobre la implementacion 27](#_Toc120275519)

[manual de usuario 27](#_Toc120275520)

[Consideracion del entorno cliente: javascript 28](#_Toc120275521)

[fetch (envio y recibos al servidor) 28](#_Toc120275522)

[cookie 28](#_Toc120275523)

[Consideracion del entorno servidor: php 29](#_Toc120275524)

[$\_session 29](#_Toc120275525)

[$\_files 29](#_Toc120275526)

[Consideracion de envio de datos: 30](#_Toc120275527)

[Envió de imágenes 30](#_Toc120275528)

[Consideracion de control de estilos: 30](#_Toc120275529)

[aplicaciones extras utilizadas en el reto 31](#_Toc120275530)

[Incidencias surgidas 31](#_Toc120275531)

[Conclusion 32](#_Toc120275532)

[Anexos 32](#_Toc120275533)

# RETO2

En este proyecto, nuestra empresa tendrá un segundo cliente llamado Aergibide SL. El propio cliente nos ha informado de un problema dentro del departamento de I+D+I: la gestión de conocimiento.

Dado a sus necesidades, este proyecto se centrará en aliviar el exhausto ante las consultas repetitivas que reciben constantemente dentro del propio departamento. También nos han informado de la formalidad de trabajo autónoma del servidor.

El objetivo con este proyecto es que cada usuario pueda ver y filtras todas las preguntas, ver respuesta de cada pregunta y valorar dichas respuestas, insertar una pregunta o respuesta de la propia pregunta, y modificar el usuario del propio usuario.

# Organización del proyecto

## Tareas vistas para realizar el proyecto

En este proyecto, ya no solo vanos a crear la interfaz de la página web, también vamos a crear el entorno de servidor de la página.

Para la parte del servidor, tenemos tres apartados importantes para la función total del reto:

* Diseño de la base de datos (tanto como en ME/R-MR como el script)
* Control de envió de datos a la interfaz de la página.
* Tratamiento y control de datos en el servidor.

En cuanto a la parte de cliente de la página, hemos tenido que controlar mas contenido que en el proyecto del tranvía de euskotren. En este caso el contenido las funciones principales son:

* Control y tratamiento de datos dentro de la interfaz de la pagina
* Funcionalidades para la comunicación con el servidor de la pagina
* Uso del Almacenamiento interno de la parte del cliente de la página web

No obstante, también tendremos que maquetar toda la pagina web con mas libertad que en el proyecto anterior, teniendo dos funcionalidades clave dentro del proyecto:

* Uso de layout de forma responsiva (UX)
* Utilidades del CSS3

Para terminar, el contenido de la pagina web se tiene que implementar un desplegable para el sencillo funcionamiento de la página.

* Uso de un desplegable para la implementación de la pagina web.

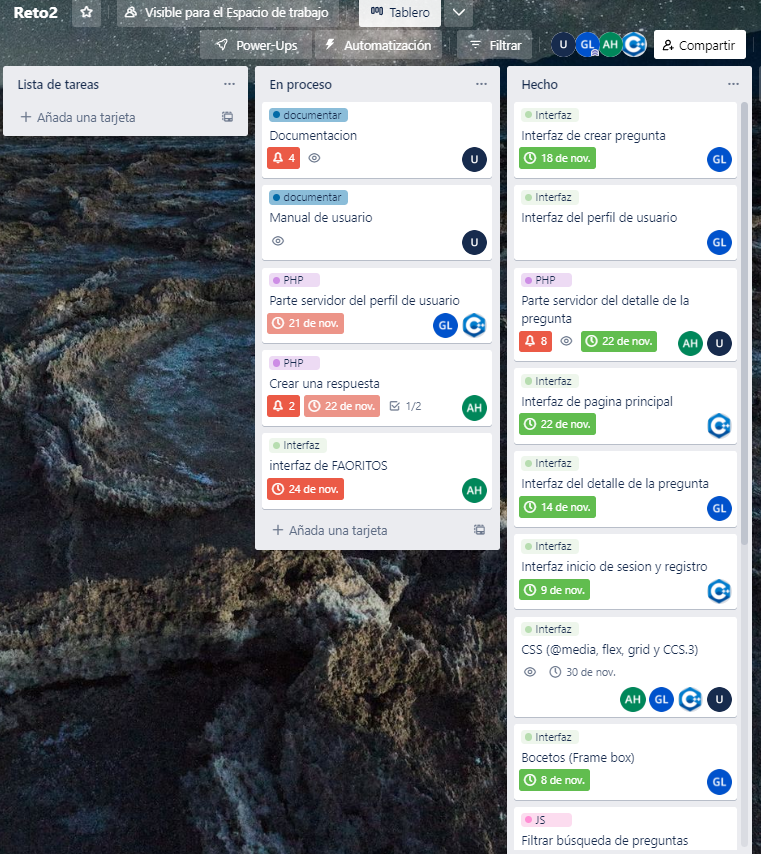
## Planificacion temporal

En la planificación de sete proyecto lo hemos implementado a través de dos aplicaciones

Trello es un software de gestión de proyectos a través de un sistema basado en Kanban.

Dicho sistema se transfiere de forma interactiva en la cual se puede gestionar las actividades a través de tarjetas virtuales, en las cuales, se pueden organizar tareas, meter comentarios y compartir tableros.

Para resumir, Trello es un tablón virtual de organización a través de tarjetas.

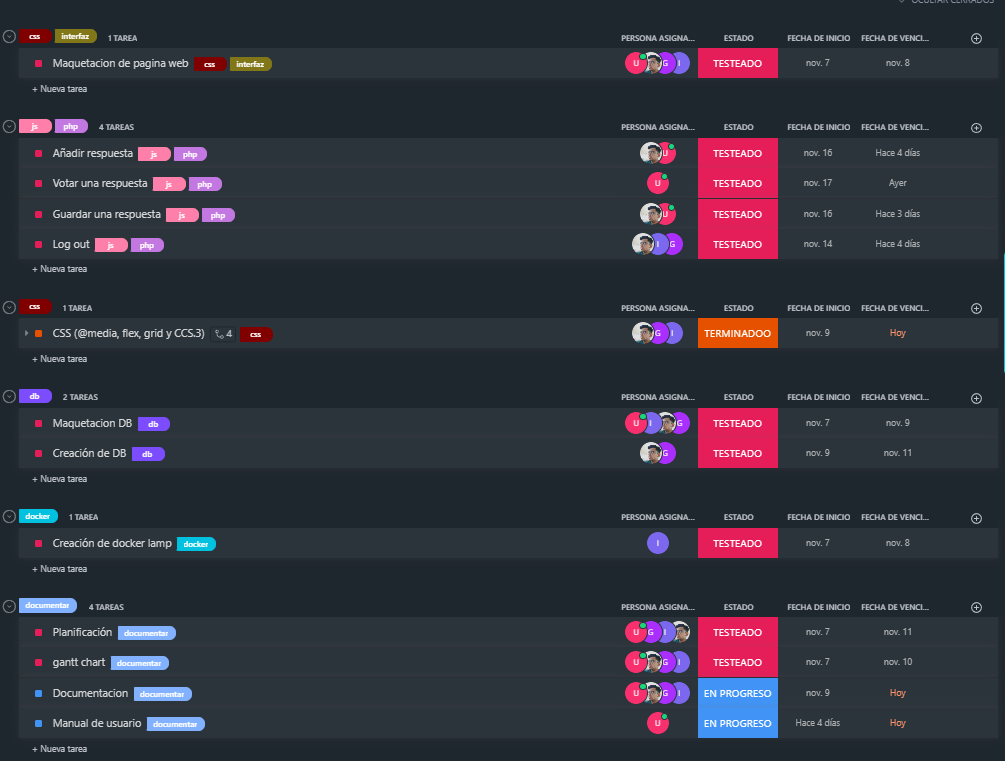


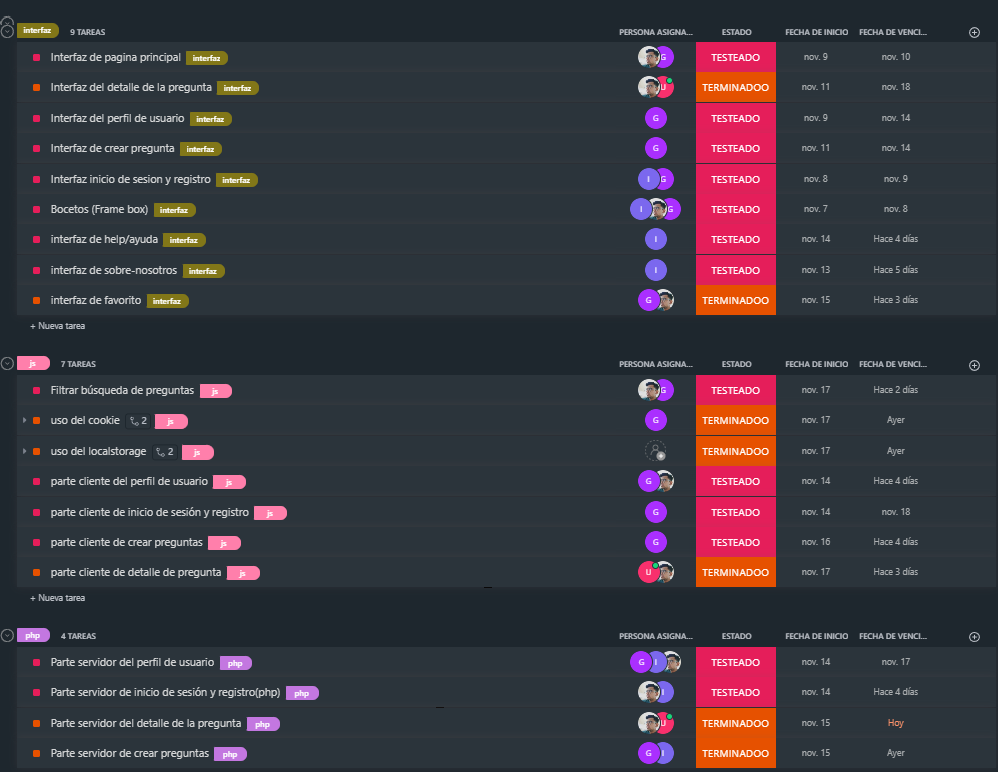
No obstante, para tener toda la planificación programada y sistematizada con un seguimiento interactivo, hemos utilizado la app clickup.

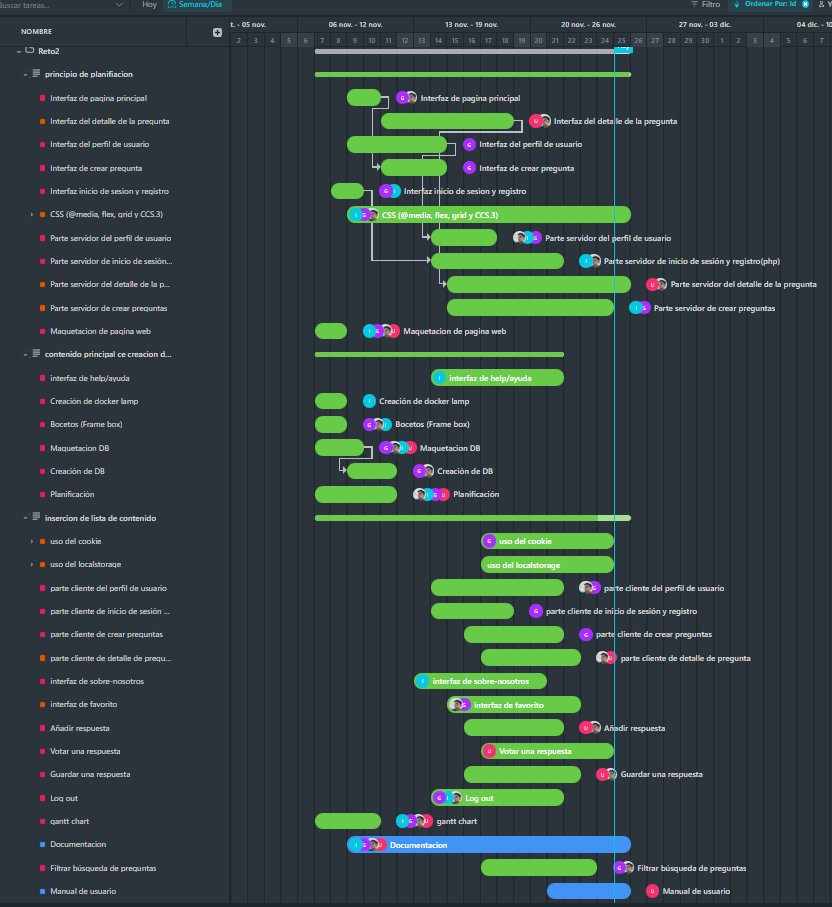
ClickUp es un software de gestión de proyectos basados en varios sistemas de gestión modernas diferentes.

Aparte de que se utiliza el método Kanban en la parte de Trello, ClickUp permite correlacionar las tareas y modificarlos fácilmente utilizando varios modelos de planificación como el Kanban, diagramas de Gantt y mapas conceptuales.

En nuestro caso, hemos implementado ClickUp para la gestión de las tareas implementadas de Trello, importando las tarjetas y relacionarlos utilizando el diagrama de Gantt.







# Especificaciones de requisitos

Todo el contenido que tenemos que hacer en las tareas a realizar anteriormente mencionado, hay que compaginar con las peticiones que el cliente nos ha mencionado con los objetivos del proyecto.

Pare ello hemos realizado varios catálogos de requerimientos para concatenar todas las necesidades para que el proyecto funcione con normalidad.

El objetivo de realizar dichos catálogos es clasificar e identificar todas las necesidades, analizarlas y separarlas y catalogar dichas actividades.

## Catalogo de requerimiento funcionales

En este apartado detallamos los requisitos funcionales para la implementación del proyecto.

Los requisitos funcionales, suelen ser funcionalidades especialmente pedidas por el cliente, aunque en algunos apartados puedan salir ciertas necesidades. ----En este caso no nos piden nada expresamente por el cliente, pero en algunos apartados se nos implementa----.

En este apartado detallamos los requisitos funcionales para la implementación del proyecto.

|  |  |  |  |
| --- | --- | --- | --- |
| nro. | Descripción | Prioridad | Exigencia |
| Módulo de navegación | | | |
| RA01 | El navegador permitirá tener una navegación principal para obtener una fluidez dinámica y sencilla. Los apartados de la navegación son los siguientes:   1. Home: ir al inicio de la pagina 2. Nosotros: ir al apartado de la empresa 3. Help: ir al apartado de ayudas 4. Sing in: iniciar sesión (RU02) 5. Sing up: crear sesión (RU03) 6. Logout: cerrar session (RU06) 7. Favoritos: guardado de preguntas (RPXX) | 1 | E |
| Módulo de usuario | | | |
| RU02 | El usuario se podrá iniciar sesión en la barra de navegación | 1 | E |
| RU03 | El usuario podrá crear un usuario dentro de la pagina | 2 | E |
| RU04 | El usuario podrá ver y modificar todos los datos personales del propio usuario iniciado. Los datos a aparecer son los siguientes:   * Nombre * Apellido * Correo electrónico * Contraseña * Confirmación de contraseña * Alias | 1 | E |
| RU05 | El usuario podrá recuperar su sesión si se le olvida la contraseña del usuario | 2 | D |
| RU06 | El usuario podrá cerrar sesión en la barra de navegación | 2 | E |
| Módulo de preguntas | | | |
| RP06 | En la parte principal de la aplicación se podrá ver todas las preguntas implementadas en la base de datos | 1 | E |
| RP07 | En la parte principal de la aplicación se podrá buscar todas las preguntas de la página a través de una búsqueda intuitiva | 2 | E |
| RP08 | En la parte principal de la aplicación se podrá insertar una nueva pregunta al presionar un botón de búsqueda llamado “Preguntar”. Los datos a implementar en cada pregunta nueva deseada son los siguientes:  Mínimo:   * Titulo * Descripción   Optativos:   * Archivo de implementación * Etiquetas de selección | 1 | E |
| RP09 | En la parte principal de la aplicación se podrá filtrar las respuestas por varios apartados sin necesidad de meter teclado. Los apartados son los siguientes   * Por fecha * Por titulo * Los que tienen más respuestas * Los que tienen menos respuestas | 2 | D |
| RP10 | En la parte principal dentro de la lista de preguntas se limitará el uso de visibilidad de las preguntas a 10 por vista | 3 | O |
| RP11 | En la parte principal dentro de la lista de preguntas se podrá interactuar con una pregunta para ver todo el contenido de la pregunta y sus respuestas a través de un detalle de la pregunta | 1 | E |
| RP12 | En el apartado de detalle de una pregunta se podrá guardar cierta pregunta para tenerlo como favorito . | 2 | D |
| RP13 | En el apartado de favoritos, se podrá ver todas las preguntas guardadas cono favoritos de la pregunta | 2 | D |
| Módulo de respuestas | | | |
| RR14 | En el apartado de detalle de una pregunta se podrá ver todas las respuestas de una pregunta. | 1 | E |
| RR15 | En el apartado de detalle de una pregunta se podrá insertar una nueva respuesta. Los requisitos mínimos para el correcto funcionamiento de una nueva respuesta son estos:   * Un usuario iniciado * Descripción de la respuesta y/o archivo de implementación | 2 | E |
| RR16 | En el apartado de detalle de una pregunta se podrá ver todas las respuestas de una pregunta listada concretamente a través de megusta-nomegusta | 2 | D |
| RR17 | En el apartado de detalle de una pregunta se podrá otra una respuesta | 3 | D |

## Catalogo de requerimiento No-funcionales

En este apartado detallamos los requisitos no-funcionales para la implementación del proyecto.

Los requisitos no funcionales, suelen ser funcionalidades no escritas ni pedidas por el cliente, aunque en algunos apartados puedan salir ciertas necesidades. En este caso no nos piden nada expresamente por el cliente, pero en algunos apartados se nos implementa.

implementación del proyecto.

|  |  |  |  |
| --- | --- | --- | --- |
| nro. | Descripción | Prioridad | Exigencia |
| RN01 | El software se desarrollará íntegramente para el uso local. | 1 | E |
| RN02 | El lenguaje de programación de desarrollo de proyecto del lado cliente será de JS puro | 1 | E |
| RN03 | El lenguaje de programación de desarrollo de proyecto del lado servidor php | 1 | E |
| RN04 | El manejador de base de datos se usará el PDO del php | 1 | E |
| RN05 | Los navegadores comprobados para el manejo del navegador web común actual (IE, Google Chrome, Mozilla Firefox, Opera) | 2 | D |
| RN06 | El módulo del despliegue de la página será utilizado con la tecnología Docker | 1 | E |
| RN07 | El control de la interfaz (UX) de la página web estará estructurada a través de la metodología Flex/Grid | 2 | E |
| RN08 | Todo el sistema de archivos que se realicen en la pagina web se implementara dentro del propio GitHub creada para dicha página web. | 1 | E |

## Rangos de implementaciones

Para realizar el catalogo de ambas partes, hemos sistematizado varios apartados para la fluidez correcta del sistema.

Aparte de las especificaciones de los apartados de requisitos, hemos sistematizado de forma sencilla todos los mínimos de la aplicación:

Cada uno de los apartados del catálogo requisitos hemos implementado diferentes maneras de enumerar los requisitos para poder sistematizar todo

|  |  |  |
| --- | --- | --- |
| Apartado | Formato de numero | Contenido variable |
| funcional | RAXX | A=>atributo de contenido  XX=> 01 incrementado en 1 |
| No-funcional | RNXX | XX=> 01 incrementado en 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| Prioridad | | Exigencia | |
| Numero | **Descripcion** | **Letra** | **Descripcion** |
| 1  2  3 | Alta  Media  Baja | E  D  O | Exigente  Deseable  Opcional |

# Diseño y organizacion de la pagina web

## Prototipado / maquetacion basica la interfaz de las paginas web

Para este proyecto, cada uno realizo una plantilla básica para toda la aplicación web. El objetivo de las plantillas era realizar con unas ilustraciones básicas en papel para poder tener una visualización previa a nuestra página web.

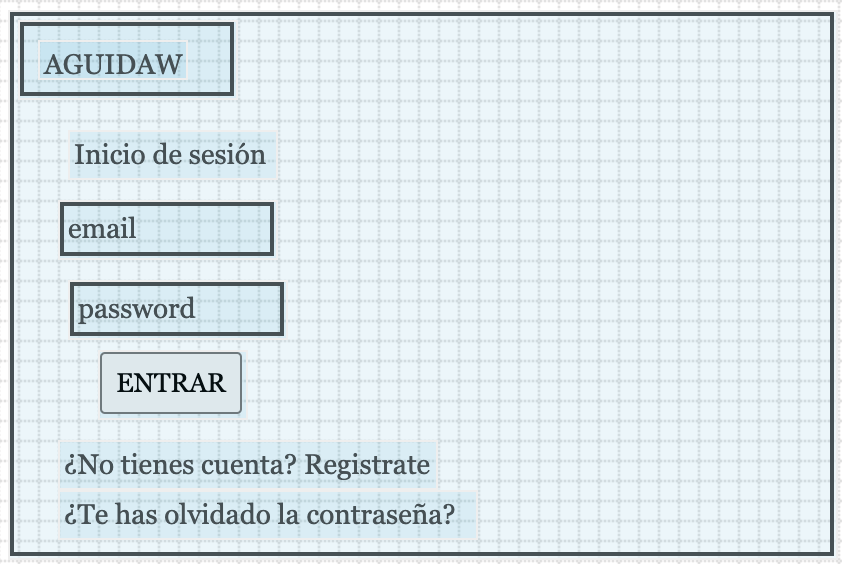
Dichas plantillas se hicieron de manera compartida a través de una página web llamado Framebox.

Framebox, es una manera sencilla de crear un boceto de pagina web como si tuvieras haciendo en un cuaderno.

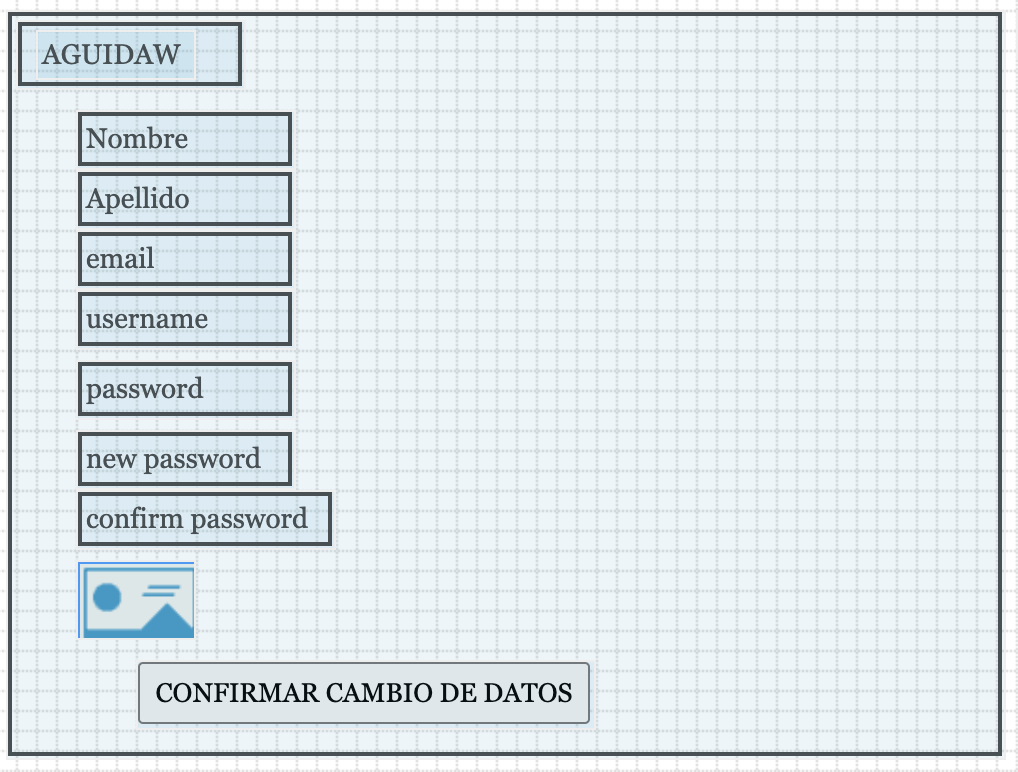
Con ello, realizando un acuerdo entre nosotros, hicimos varios bocetos dependiendo de la pagina web si detallar mucho la estructura de la página.

Los bocetos se han realizado de esta manera:

Inicio\_session



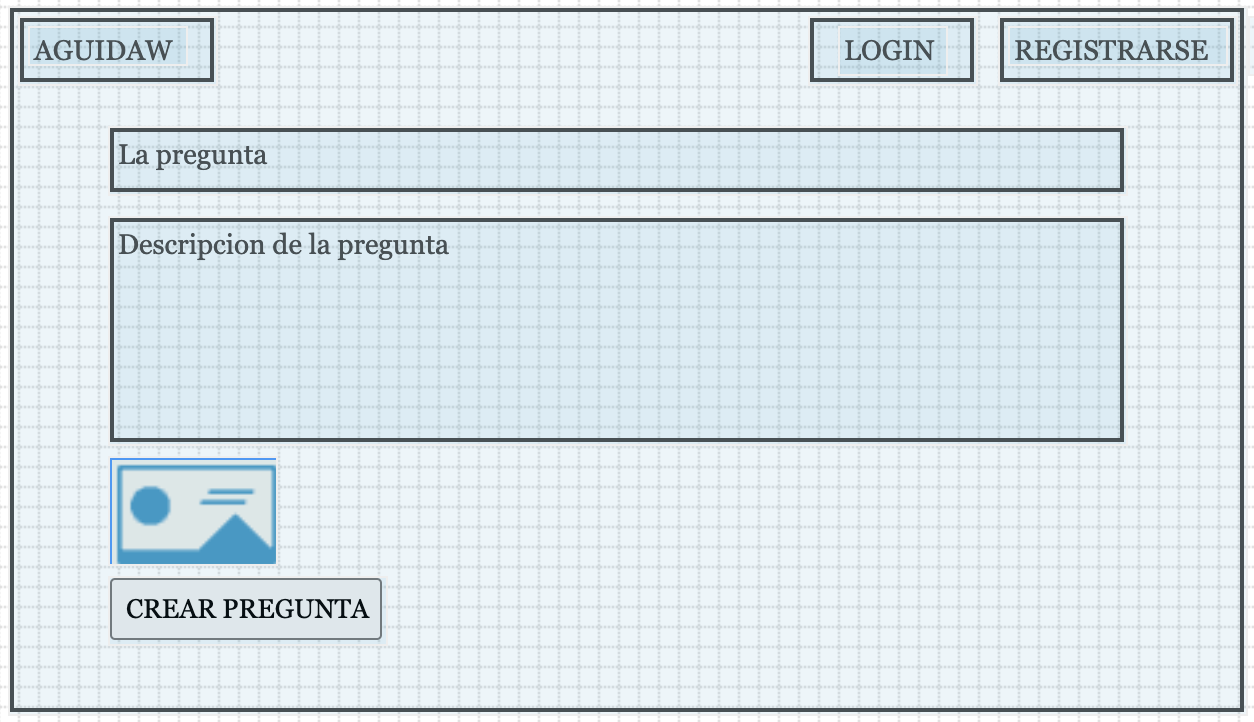
Registrar



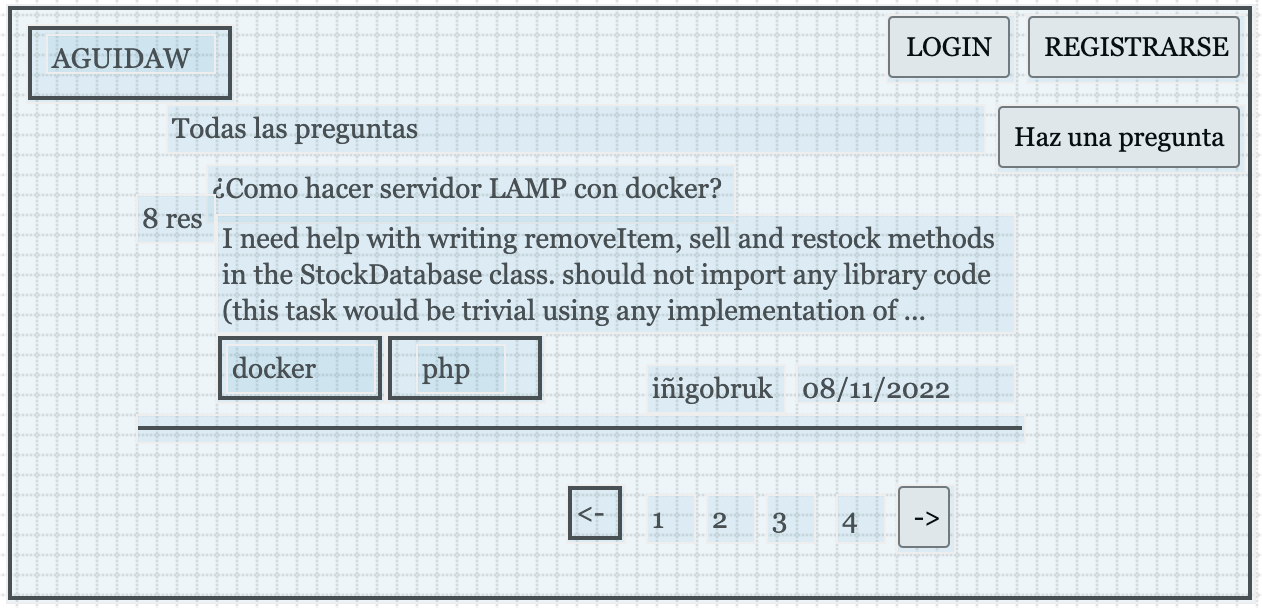
Datos\_usuario

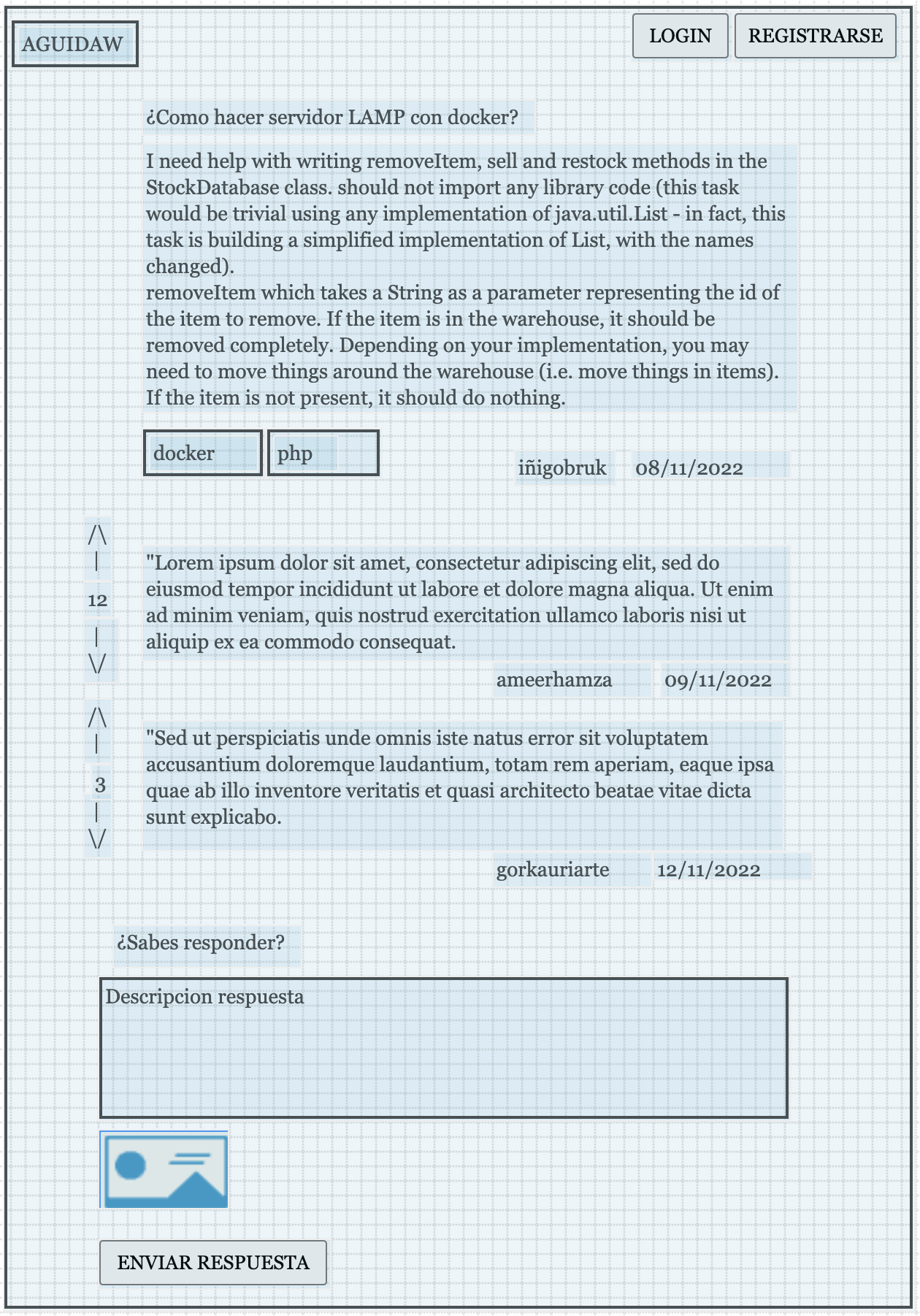
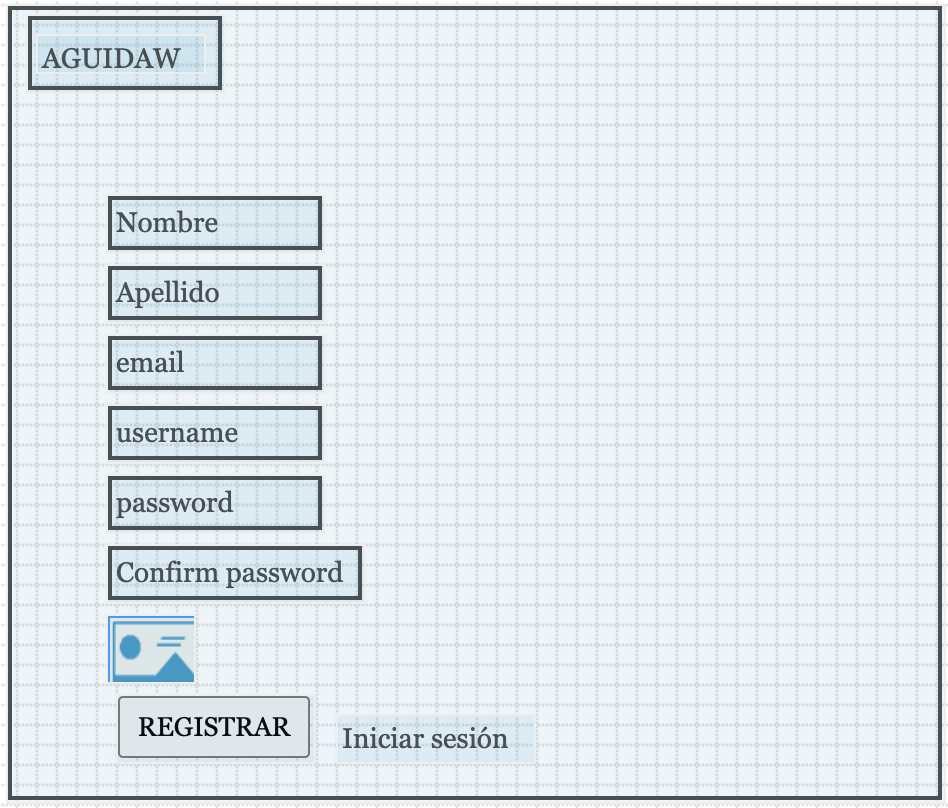


crear\_pregunta



pagina\_principal



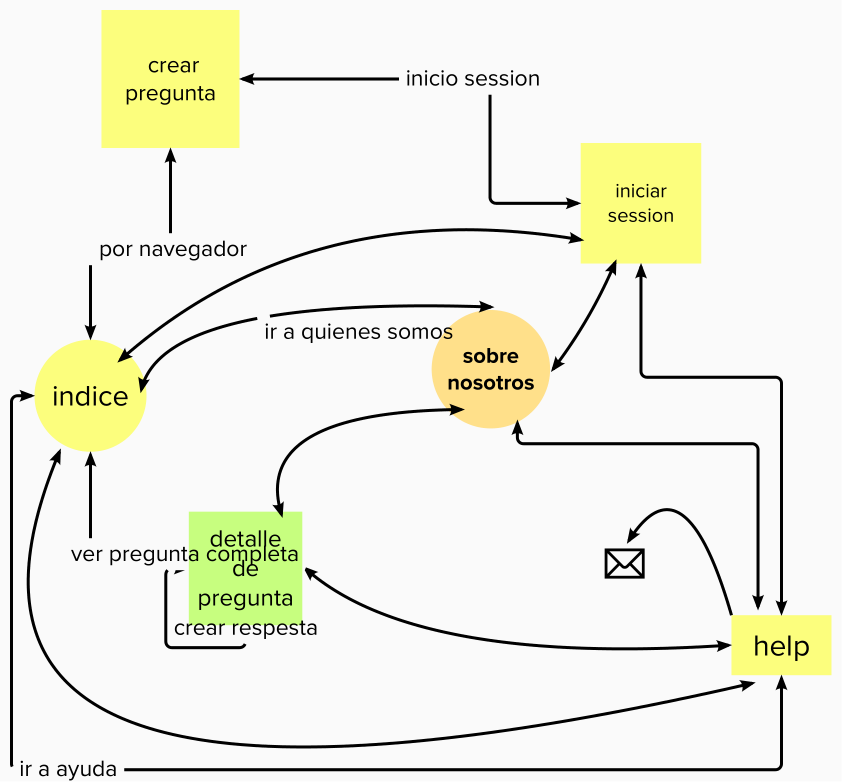
Detalle\_preguntaregistrar

## Contenido de la interfaz web

Para la creación del organigrama de la pagina web, hemos utilizado un mural llamado mural.co, en la cual se realizan mapas mentales para crear murales de distintos apartados.

MURAL permite utilizar plantillas sencillas y editar o crear un muro a través del uso de posits.

En nuestro caso lo hemos utilizado para ver todas las fluctuaciones de las paginas.



## Metodología del despliegue del entorno

Para realizar todas las peticiones (como minimas, como algunas implementaciones extras), hemos tenido que organizar con varias tecnologias diferentes para el posible funcionamiento totalitario del proyecto.

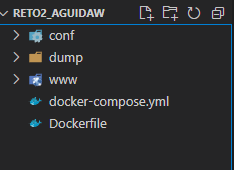
Para ello, hemos utilizado la tecnologia de Docker para poder implementar el sistema de la aplicación.

Docker es un automatizador de despliegue de aplicaciones utilizando contenedores software. Permite crear, probar e implementar contenido a traves de una nube.

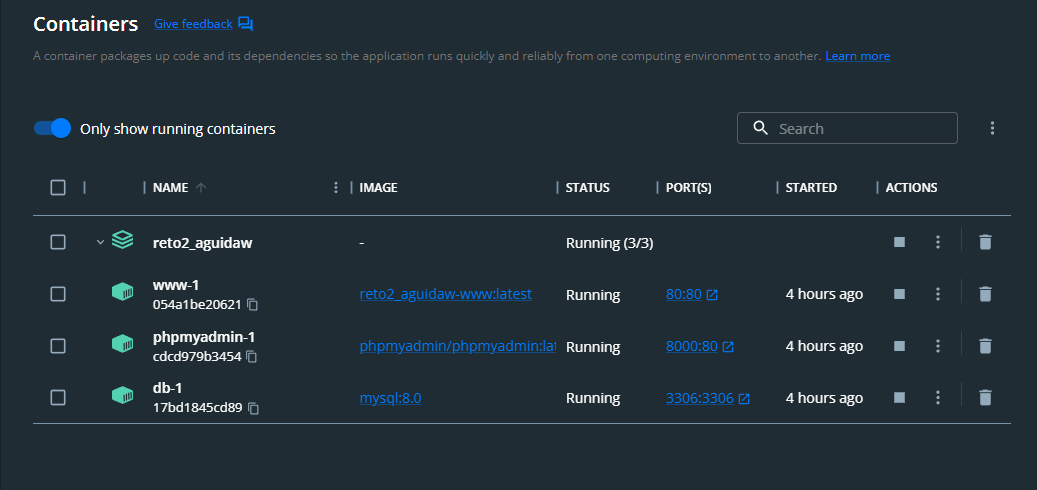
Para poder tenerlo en uso, hemos creado una carpeta normal en nuestro GitHub con varios archivos y carpetas diferentes:

* conf: Configuraciones para el despliegue. Por defecto no tiene ningún archivo.
* dump:
  + dbname.sql: archivo de script de la base de datos.
* www: Carpeta de contenido de la implementación de la página web.
* docker-compose.yml: Configuración del contenido de los contenedores del Docker
* Dockerfile: Script de configuración para la implementación/composición del Docker

Mas adelante hablaremos el contenido que tiene la carpeta www.



En el caso de este proyecto hemos tenido tres contenedores dentro de una carpeta



* reto2\_aguidaw: carpeta con los contenedores necesarios para la apertura del despliegue de la página web.
  + www-1: contenedor de toda la programación de la página web
  + phpmyadmin-1: contenedor de control de la base de datos en interfaz
  + db-1: contenedor de modulo de la base de datos (MySQL)

## Modelo de datos

Todo el contenido de nuestra base de datos esta organizada de esta manera:

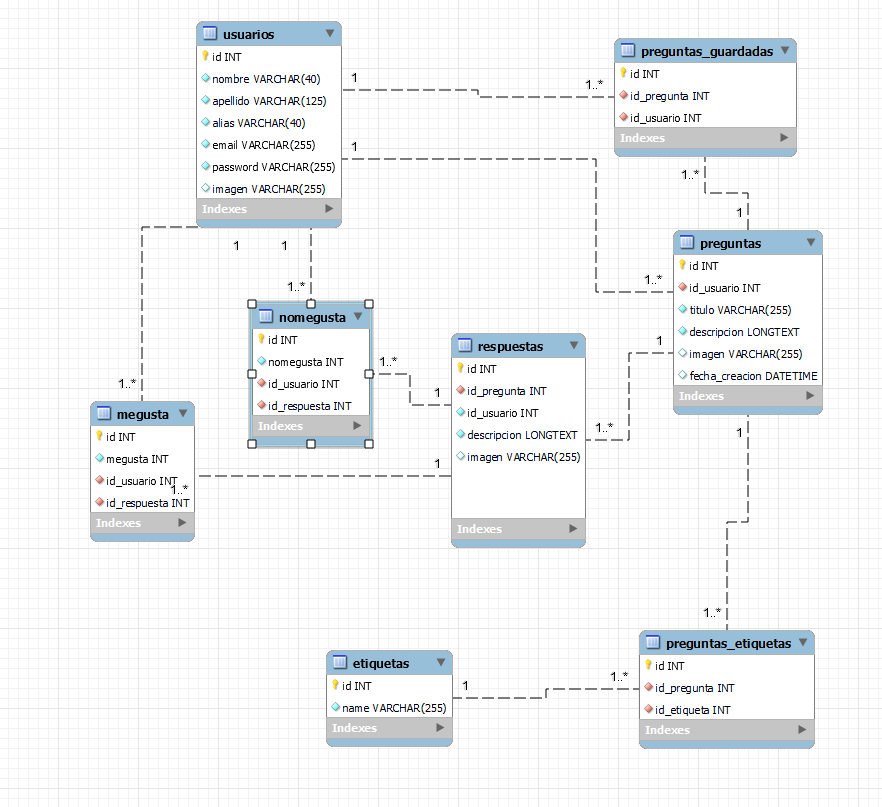


Tabla usuarios:

En esta tabla guardamos todos los datos de un usuario de la página web.

* id: Identidad del usuario (int con incremento automático) (PK)
* nombre: Nombre del usuario (varchar 40)
* apellido: apellido/s del usuario (varchar 125)
* alias: alias del usuario (varchar 40)
* email: correo electrónico del usuario (varchar 255)
* password: Contraseña del usuario (varchar 255)
* imagen: imagen del perfil de usuario (varchar 255) (OPCIONAL)

Tabla preguntas:

En esta tabla guardamos todas las preguntas de la página web.

* id: Identidad de la pregunta (int con incremento automático) (PK)
* id\_usuario: identidad del usuario (FK-> id de la tabla usuario)
* Título: título de la pregunta (varchar 40)
* descripción: contenido detallado de la pregunta (LONGTEXT)
* imagen: imagen del perfil de usuario (varchar 255) (OPCIONAL)
* fecha\_creacion: fecha de creación de la pregunta (DATETIME) ) (OPCIONAL)
  + la fecha se crea de forma automática.

Tabla preguntas\_guardadas: (favoritos)

Esta tabla esta creada de forma relacional dado a que un usuario puede guardad alguna que otra pregunta como favorito varias preguntas guardadas dentro del propio usuario.

* id: Identidad de cada pregunta guardada como favoritos (int con incremento automático) (PK)
* id\_pregunta: identidad de la pregunta (FK-> id de la tabla preguntas)
* id\_usuario: identidad del usuario (FK-> id de la tabla usuario)

Tabla etiquetas:

En esta tabla guardamos todas las etiquetas guardadas para la página web.

* id: Identidad de la pregunta (int con incremento automático) (PK)
* name: nombre de la etiqueta (varchar 255)

Tabla preguntas\_etiquetas:

Esta tabla esta creada de forma relacional dado a que una pregunta puede tener varias preguntas guardadas dentro del propio usuario.

* id: Identidad de la preguntas\_etiquetas (int con incremento automático) (PK)
* id\_pregunta: identidad de la pregunta (FK-> id de la tabla preguntas)
* id etiqueta: identidad de la etiqueta (FK-> id de la tabla etiquetas)

Tabla respuestas:

En esta tabla guardamos todas las respuestas existentes para una pregunta.

* id: Identidad de la respuesta (int con incremento automático) (PK)
* id\_pregunta: identidad de la pregunta (FK-> id de la tabla preguntas)
* id\_usuario: identidad del usuario (FK-> id de la tabla usuario)
* descripción: contenido detallado de la respuesta (LONGTEXT)
* imagen: contenido en modo imagen de la respuesta (varchar 255) (OPCIONAL)

Tabla megusta:

En esta tabla guardamos todas las respuestas que se votan de forma positiva a una respuesta de la página web realizado por un usuario.

* id: Identidad de la pregunta (int con incremento automático) (PK)
* megusta: contenido de megusta (int)
* id\_usuario: identidad del usuario (FK-> id de la tabla usuario)
* id\_respuesta: identidad de la respuesta (FK-> id de la tabla respuesta)

Tabla nomegusta:

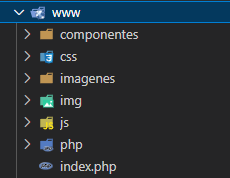
En esta tabla guardamos todas las respuestas que se votan de forma negativa a una respuesta de la página web realizado por un usuario.

* id: Identidad de la pregunta (int con incremento automático) (PK)
* nomegusta: contenido de nomegusta (int)
* id\_usuario: identidad del usuario (FK-> id de la tabla usuario)
* id\_respuesta: identidad de la respuesta (FK-> id de la tabla respuesta)

## Contenido del entorno tecnologico (como se controla la pagina web)

Anteriormente mencionamos que íbamos a detallas mas el contenido de la carpeta de “www”. Ahora es el momento.

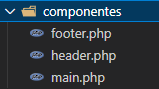
Todas las carpetas dentro de www, esta estructurado para el comportamiento simple de la pagina.

El sistema de archivos del www esta controlado de esta manera:

* componentes.
* css.
* Imágenes.
* Img.
* js.
* Php.
* Index.php

### plantilla de las paginas web

En el apartado de componentes, se implementa el layout general de la pagina, seprando todo el contenido del header, footer,help y main de la pagina.

En el fichero footer.php se implementa la vista general del pie de la página web.

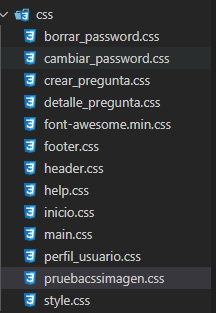
En el fichero header.php se implementa la vista general del encabezado de la página web.

En el fichero main.php se implementa la vista general del apartado de nosotros que está en la barra de navegación que está en el header.php.

### css

En el apartado de css están todas las implementaciones de estilos de todas las paginas web creados durante este proyecto.

La mayoría de los ficheros css están nombrados para sincronizar todos los estilos necesarios con el php de la vista de dicha página:



* borrar\_password.css --> php/borrar\_password.php
* cambiar\_password.css --> php/perfil\_usuario.php
* crear\_pregunta.css --> php/crear\_pregunta.php
* detalle\_pregunta.css --> php/detalle\_pregunta.php
* footer.css --> components/footer.php
* header.css --> components/header.php
* help.css --> php/help.php
* main.css components/main.php
* perfil\_usuario.css --> php/perfil.php

### cotrol de imágenes/archivos

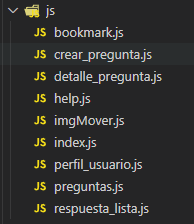
En el apartado de imágenes están guardados todas las imágenes de los usuarios que están implementados dentro de la página web.

En el apartado de img están guardados todas las imágenes que contienen en la página de forma simple.



### entorno cliente: java script

En el apartado de js están todos los archivos de script de JavaScript implementadas en todas las páginas.



En el fichero de bookmark.js se utiliza para controlar los favoritos.

En el fichero de crear\_pregunta.js se utiliza para controlar las etiquetas. Tiene estas funciones:

* mostrarEtiquetasSeleccionadas: depende de cuantas etiquetas se seleccione en la ventana de crear\_preguntas, aparece como se vería al visualizar la pregunta.
* getSelectValues: obtiene todos los valores dentro de

En el fichero detalle\_pregunta.js se utiliza para controlar el contenido de alguna cookie. (mas detalle en el apartado de “consideraciones de js”).

En el fichero help.js se utiliza para controlar el envió de mensajes en la página help. Este fichero solamente se inicializa cuando se envía a través de un XMLhttprequest (más detalle en el apartado de “consideraciones de js”).

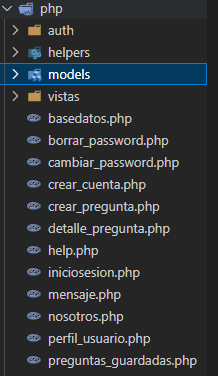
En el fichero de perfil\_usuario.js se utiliza para realizar varios eventos controlando ante ciertos botones. Para ello se utiliza varias funciones:

* habilitar: simplemente habilita la posibilidad de editar el perfil en la página web
* deshabilitar: simplemente deshabilita la posibilidad de editar el perfil en la página web
* mostrarContrasenaActual: al realizar click en el botón que está al lado del apartado de “contraseña actual”, permite ver lo que se escribe en ese contenido. Además, altera la función que realiza dicho botón.
* ocultarContrasenaActual: al realizar click en el botón que está al lado del apartado de “contraseña actual”, oculta el texto de lo que se escribe en ese contenido. Además, altera la función que realiza dicho botón.
* mostrarContrasenaNueva: al realizar click en el botón que está al lado del apartado de “Nueva contraseña”, permite ver lo que se escribe en ese contenido. Además, altera la función que realiza dicho botón.
* ocultarContrasenaNueva: al realizar click en el botón que está al lado del apartado de “Nueva contraseña”, oculta el texto de lo que se escribe en ese contenido. Además, altera la función que realiza dicho botón.
* mostrarContrasenaConfirmar: al realizar click en el botón que está al lado del apartado de “Confirmar contraseña”, permite ver lo que se escribe en ese contenido. Además, altera la función que realiza dicho botón.
* ocultarContrasenaConfirmar: al realizar click en el botón que está al lado del apartado de “Confirmar contraseña”, oculta el texto de lo que se escribe en ese contenido. Además, altera la función que realiza dicho botón.

### php

En el apartado de php esta todo el contenido que necesitamos para crear el entorno servidor de la página. Dentro de ello hay 4 carpetas importantes a tener en cuenta (mas detalle de cada uno mas adelante:

* auth: contiene archivos de controles de usuarios en el entorno servidor.
* helpers: en este caso se utiliza para la administración de archivos para la pagina web y para el control base de la base de datos con la plantilla de archivos.
* models: esta todo el contenido que se utiliza para el control de la base de datos.
* vistas: esta la vista predeterminada de la pagina



En el fichero de actualizar\_perfil.php están todas las configuraciones que se realizan cuando un usuario quiere actualizar los datos.

En el fichero de borrar\_password.php es una vista de recuperación de contraseña por si se olvida el usuario la contraseña.

En el fichero de cambiar\_password.php es una vista de cambio de contraseña de contraseña por si se olvida el usuario la contraseña.

En el fichero de crear\_cuenta.php tiene dos apartados:

* tiene un control sencillo de comprobación de una sesión iniciada
* tiene una vista controlada para la creación de la cuenta con condiciones simples utilizando el apartado de $\_SESSION.

En el fichero de crear\_pregunta.php es una vista de creación de una pregunta, con un script redireccionado a un js.

En el fichero de detalle\_pregunta.php es una vista de creación de una pregunta, con un script redireccionado a un js.

En el fichero de iniciosesion.php tiene dos condiciones basicas:

* Si encuentra una sesión iniciada, directamente cierra dicha sesión y se va al index.php utilizando el apartado de $\_SESSION.
* Si no encuentra una sesión iniciada, se ve una vista de la pagina que pedirá el contenido necesario para iniciar el usuario para iniciar sesión, o para enviar al fichero “borrar\_password.php

En el fichero de nosotros.php es una vista básica que llama a tres componentes diferente:

* header.php: cabecera
* Main.php:contenido central.
* footer.php: pie de pagina.

En el fichero de perfil\_usuario.php es una vista de detalle de la sesión iniciada del usuario, con opción de realizar cambios en el propio perfil de usuario.

En el apartado index.php es el primer archivo que se lee dentro de todas las vistas de la página web. Dado a dicha necesidad, se implementa por separado.

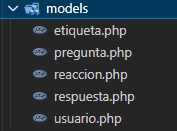
## Contenido del control-modelo de datos

Cualquier dato que se controla dentro de la base de datos hay que ser enviado/recibide de alguna manera.

Para la conexión de la base de datos, se utiliza hay un archivo llamado “basedatos.php” que contiene solamente una función que permite conectarse sin ningún conveniente a la base de datos.

Dentro del contenido de búsqueda de la base de datos, todo el control necesario con la base de datos se mete dentro de la carpeta models, situado dentro de la carpeta php.

Dentro de la carpeta models, se implementa 5 archivos indispensables que nos permite realizar todas las actividades con la base de datos. Dichos archivos tienen un objetivo básico:



* etiqueta.php: realiza todo el contenido necesario con la tabla de etiquetas y pregunta. Tiene estas funciones.
  + etiquetas: busca y devuelve todas las etiquetas
  + buscaEtiquetasDeUnaPregunta: busca las etiquetas que están implementadas dentro de una pregunta.
  + buscaPreguntasPorEtiqueta:busca las preguntas que tienen asignada una etiqueta.
  + crearPreguntaEtiqueta:inserta una nueva fila en la tabla preguntas\_etiquetas
  + buscarEtiquetaPorNombre:busca el nombre de la etiqueta de ina identidad de la etiqueta de la tabla etiquetas.
* pregunta.php: realiza todo el contenido necesario con la tabla de preguntas.
  + preguntas: busca y devuelve todas las preguntas
  + preguntasOrdenadoPor: busca y devuelve todas las preguntas ordenado de una manera deseada.
  + borrarPregunta: borra una pregunta deseada.
  + crearPregunta: inserta una nueva fila en la tabla preguntas, creando una nueva pregunta.
  + actualizarPregunta: cambia el contenido de una pregunta.
  + buscarPreguntasPorTitulo: busca la pregunta que tenga que tenga el contenido del texto como escribe en el titulo.
  + buscarPreguntasPorId: busca una pregunta en concreto.pregunta\_guardada.php: realiza todo el contenido necesario con la tabla de preguntas\_guardadas.
  + esPreguntaMarcada: busca los datos de una pregunta guardada.
  + bookmarkPregunta: implementa una nueva línea en preguntas guardadas.
  + unbookmarkPregunta: borra una linea en preguntas\_guardadas
* reacción.php: realiza todo el contenido necesario con las tablas de megusta y nomegusta con alguna filtración con la tabla de respuestas.
  + buscarUpvoteDeUnaRespuesta: Cuenta la cantidad de megusta que tiene una respuesta en concreto
  + buscarDownvoteDeUnaRespuesta: Cuenta la cantidad de nomegusta que tiene una respuesta en concreto
  + buscarUpvotes: busca si cierto usuario ya tiene insertado alguna reacción en la tabla de megusta dentro de cierta respuesta.
  + buscarDownvotes: busca si cierto usuario ya tiene insertado alguna reacción en la tabla de nomegusta dentro de cierta respuesta.
  + Upvote: inserta una nueva reacción positiva de un usuario a una respuesta concreta en la tabla megusta
  + Downvote: inserta una nueva reacción negativa de un usuario a una respuesta concreta en la tabla nomegusta.
* respuesta.php: realiza todo el contenido necesario con la tabla de respuestas con alguna filtración con la tabla de preguntas.
  + insertarRespusta: inserta una nueva respuesta.
  + cambiarRespusta. Altera una respuesta en concreto.
  + borrarRespusta: Elimina una respuesta concreta.
  + buscarRespuestaPorFecha: busca las respuestas de una pregunta y lo ordena por fecha de creación de forma ascendente.
  + buscarRespuestaPorVotos: busca las respuestas de una pregunta, contabiliza la cantidad de megusta y no me gusta en separado de cada respuesta y lo ordena por megusta/nomegusta.
  + buscarRespuestaDeUnaPregunta: busca todas las respuestas de una pregunta.
  + crearRespuesta: inserta una nueva respuesta.
  + actualizarRespuesta: Altera una respuesta en concreto.
* Usuario: realiza todo el contenido necesario con la tabla de usuarios.
  + buscarUsuarioPorEmail: busca un usuario por correo existente
  + buscarUsuarioPorId: busca un usuario por su identidad.
  + insertarUsuario: inserta un nuevo usuario.
  + actualizarUsuario: cambia los datos de un usuario.

# Consideraciones sobre la implementacion

Todo el contenido mencionado en el diseño de la página web está bien, pero hay cosas que nos gustaría mencionar de una manera más detallada.

## manual de usuario

Para ver de forma mas detallada todas las funciones de nuestra pagina esta en el anexo que se implementa en nuestro apartado de anexos.

## Consideracion del entorno cliente: javascript

Todo el contenido con el lenguaje de programación hemos metido varias funcionalidades nuevas y necesarias para poder controlar de una manera mas sencilla ante el control de almacenamiento y envíos entre el cliente-servidor

### fetch (envio y recibos al servidor)

La mayoría de los datos que realizamos para controlar el entorno cliente, solamente utilizábamos el programa de JavaScript, pero para realizar algunas funcionalidades, hemos tenido que enviar datos al servidor sin la necesidad de recargar los datos.

En este proyecto hemos utilizado dos maneras de enviar el contenido al entorno servidor.

El primer método utilizado ha sido a través del XMLHttpRequest, utilizado exclusivamente en el fichero de help.js

Principalmente, metemos un nuevo XMLHttpRequest a través de un variable.

Después, en la variable creada, cuando recargue la variable, creamos una función anónima para comprobar varias cosas, como el estado de conexión de la conexión, como las condiciones extra que se requieren.

En la parte final, creamos otra variable para crear un FormData (un nuevo formulario) con el contenido del formulario del help para luego enviar dicho formulario.



El segundo método hemos utilizado la metodología de “fetch”, en la cual nos permite comunicar con el servidor con mayor flexibilidad.

Para usar fetch necesitamos dos parámetros: la url de la petición y la segunda la manera de enviar al servidor, teniendo tres apartados dentro:

* method: el método de envío (por defecto el método GET)
* headers: la configuración del contenido del envio (tipo de dato, longitud,etc.)
* body: los datos que se envían al servidor, en general se realiza a través del JSON.Stringfy.

Después del uso del fetch hay varias funcionalidades extra tras su uso:

* then: función posterior al fethc, repetible tras realizar un then.
* Catch: función de errores tanto como al realizar un fetch como al realizar un then.

En nuestro caso hemos implementado varios, tanto en el bookmark.js como en el respuesta\_lista.js

### cookie

Dado a que no todos los datos sean necesarios, alguna que otra cosa también es necesario implementar almacenamientos locales sin la necesidad indispensable del entorno servidor.

En nuestro caso hemos implementado un control de cookies dentro del fichero detalle\_pregunta.php



## Consideracion del entorno servidor: php

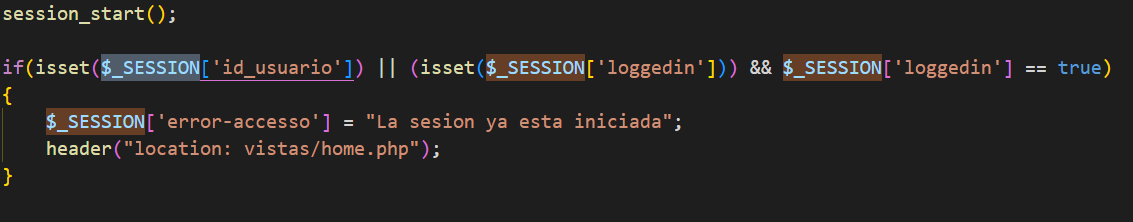
Cualquier dato es importante en varias paginas web, y a veces es un poco

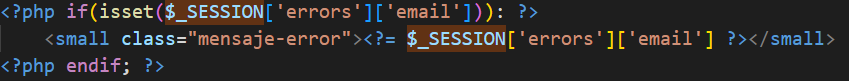
### $\_session

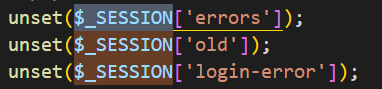
En nuestro caso henos utilizado esta funcionalidad para dos apartados con el control de las sesiones.

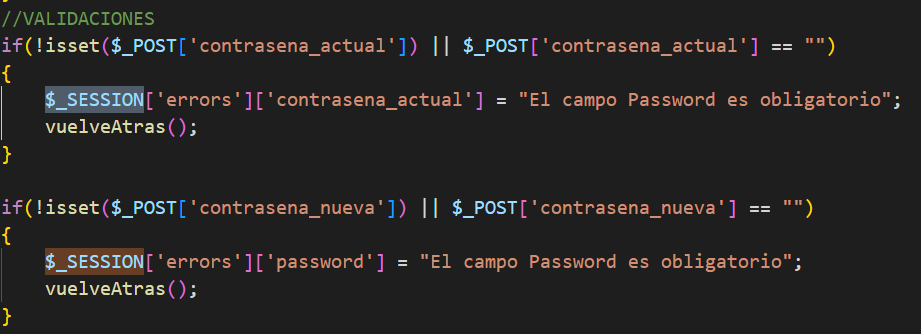
Principalmente lo hemos utilizado para comprobar el control de inicio de session y para clasificar errores de accesso.

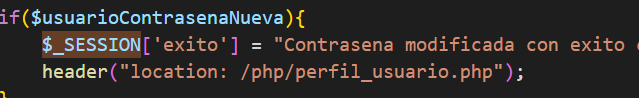
Algunos ejemplos:











### $\_files

Todos los archivos que se insertan en la pagina web, están controlados con el prefijo de $\_FILES.

En nuestro caso lo hemos utilizado para la comprobación de existencias de dicho archivo y utilizando las funciones de subirFoto y borrarFoto que esta controlado en file\_manager.php:

Aquí hay algunos ejemplos:



## Consideracion de envio de datos:

Todos los datos que necesitemos enviar y recibir están en una carpeta llamada “models”. Con ello de forma sencilla podemos separar todo el contenido que necesitemos realizar con la base de datos, pero eso no significa que todos los datos vayan a ser necesarios, por ello hay ciertos criterios en las creaciones de ciertos apartados

## Consideracion de control de estilos:

Todos los estilos, como ya esta mencionado en la plantilla de las paginas web, tiene un estilo detallado de cada pagina. En nuestro caso hemos realizado varias visualizaciones básicas dependiendo del tamaño de la ventana de la pagina web.

Para comenzar, todo el contenido de la pagina esta separado en tres partes:

* Header: 73px
* Main\_part: min height =calc(100vh -73px-30vh)
* Footer: 30 wh

En caso de que el tamaño sea superior a 1000px, la barra de navegador de la cabecera se rellenara con el contenido de la barra de navegación, y el icono/logo estará más centrado .

En caso de que el tamaño este entre 768px y a 1000px, la barra de navegador de la cabecera se rellenara con el contenido de la barra de navegación, pero el logo no estará tan centrado.

En caso de que el tamaño sea inferior a 768px, la barra de navegador de la cabecera tendrá varias líneas, en la cual al pulsar dichas líneas aparecerán las opciones de la navegación en toda la pantalla .

## aplicaciones extras utilizadas en el reto

Todas las aplicaciones mencionadas en los apartados, solamente mencionamos una aplicación de despliegue de la pagina web, una pagina donde se pueden realizar bocetos sencillos de cualquier página web y un par de planificadores. Queda varios mas por comentar que hemos utilizado de forma diaria en el proyecto:

Para el editor de código de nuestra pagina web, hemos utilizado Visual Studio Code.

Visual Studio Code es un editor de código fuente que comparte varios lenguajes de programación y un conjunto de características que puede o no estar en la interfaz del usuario.

En este caso hemos utilizado para la edición de los lenguajes de php, JavaScript, css y YAML (solamente para el docker-compose.yml).

Para el control y compartimiento de archivos de este proyecto se ha utilizado dos aplicaciones para no tener dependencia de una: GitHub y Google Drive.

GitHub es una forja para alojar proyectos utilizando el sistema de control de versiones. Gracias a ello podemos controlar todos los cambios y separar las modificaciones sin la necesidad de alterar con otros compañeros al realizar cambios a los ficheros. En nuestro lo hemos implementado para

Google Drive es un servicio de alojamiento y sincronización de archivos con la posibilidad de compartimiento con multitudes personas. En nuestro caso lo hemos utilizado para realizar copias de seguridad por si no hay disponibilidad de subir a la nube a través del GitHub.

# Incidencias surgidas

Como en este reto henos sido solo nosotros, hemos tenido mucha más libertad ante la realización de contenido, pero henos tenido constantes cambios en cuanto a la reorganización de tareas.

El método de despliegue de la aplicación no ha sido lo mas desplegable, dado a que un ordenador que tenia un antivirus bastante avanzado bloqueaba la posibilidad del método de despliegue de la página.

Las maneras de envío de servidor de forma concurrente también han tenido algún inconveniente, dado al método de los entornos.

# Conclusion

Gracias a este proyecto, hemos aprendido en limitar y sistematizar de una manera mas clara toda la codificación, aunque haya habido coques de confrontación de estilos, se aprende.

# Anexos

# 

DOCUMENTOS EXTERNOS:

* [Boceto](http://framebox.org/ARpKA)s previos
* [Manual de usuario](MANUAL%20DE%20USUARIO.pdf)

GitHub: <https://github.com/gorkauriarte/Reto2_AGUIDAW>