upna
Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

Web

ForoUpna

Jhonny F., Chicaiza Palomo

Proyecto de IWE

Índice

La pagina web se encuentra en la url:

http://eim-alu-83201.lab.unavarra.es/

Presentación y objetivos

Este proyecto consiste en la creación de una página web del tipo Foro que sirva para la interactuación entre los estudiantes de la UPNA. En esta página se podrá registrar y subir una foto para su perfil, cada usuario podrá crear un foro y podrá interactuar con los demás foros creado, habrá un buscador para facilitar la búsqueda de los foros por el título. En la creación de un foro de podrá subir una foto para que acompañe a este.

Los objetivos principales para desarrollar son:

- Creación de usuario
- Modificación de los datos de un usuario
- Creación de foro
- Buscador de foros
- Creación de comentarios dentro de un foro
- Fácilmente usable para las diferentes plataformas web, móvil.
- Se implementarán reglas que permitan una mejor a accesibilidad ya se desde el diseño y desde el código.

Requisitos y prototipos

En primer lugar, se deja el link del wireframe realizado para el diseño de la página web: **lnik: wireframe**

1. Barra de Navegación:
   Esta constara con links de acceso a los diferentes sitios de la pagina web, y se adaptara según si el usuario esté logueado o no. Estará fija en todas las páginas de la web.

2. Sistema de login y registro:
   Para que el usuario se registre es necesario un email, contraseña y un nombre de usuario que será único. Para el login se podrá realizar con el nombre de usuario y la contraseña.

   Para que el sistema se más seguro se ha limitado el tiempo de la sesión si el usuario esta inactivo a 800 segundos.

3. Página principal:
   Consta de dos partes, la parte superior donde aparecerán los foros más comentados y la parte inferior donde aparecerán los últimos comentarios que se han hecho. Desde las tarjetas del foros mas comentados o comentarios el usuario podrá acceder a este.

4. Perfil:
   El usuario podrá visualizar sus datos de perfil aquí junto a su foto. Además, podrá modificarlos si este lo desea.

5. Buscador:
   El usuario podrá buscar el foro por el título, se listarán y se podrá acceder a estos con un click, se realizará una paginación si no caben en la página, para cargar las paginas se utiliza Ajax.

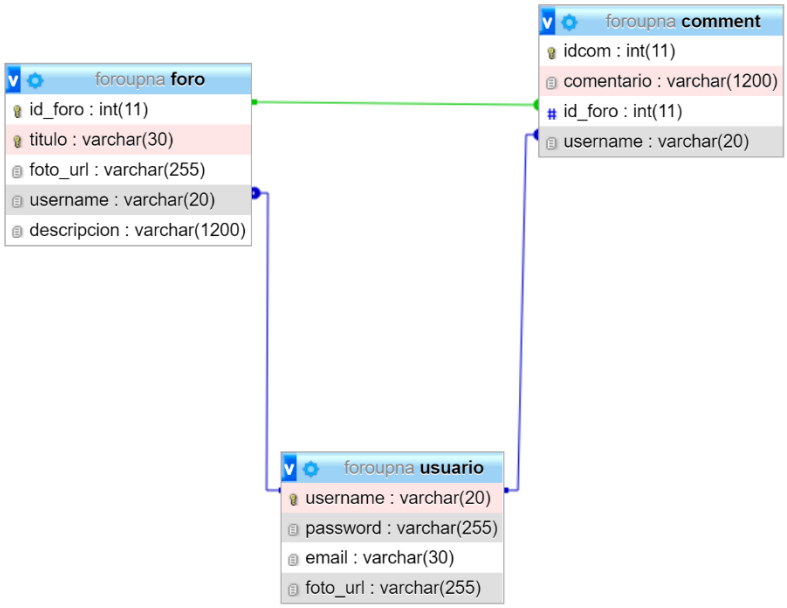
6. Foro:

   La parte principal de la página web. En primer lugar, se visualizará el nombre del foro y la foto, si no subió foto se pondrá una por defecto. Después aparecerá el comentario que puso el creador del foro, posteriormente aparecer un textarea donde el usuario logueado podrá escribir su comentario y añadirlo al foro. Finalmente se listaran los comentarios del foro, si son demasiados se realizara la paginación de estos, se podrán cargar los distintos comentarios mediante llamadas Ajax.

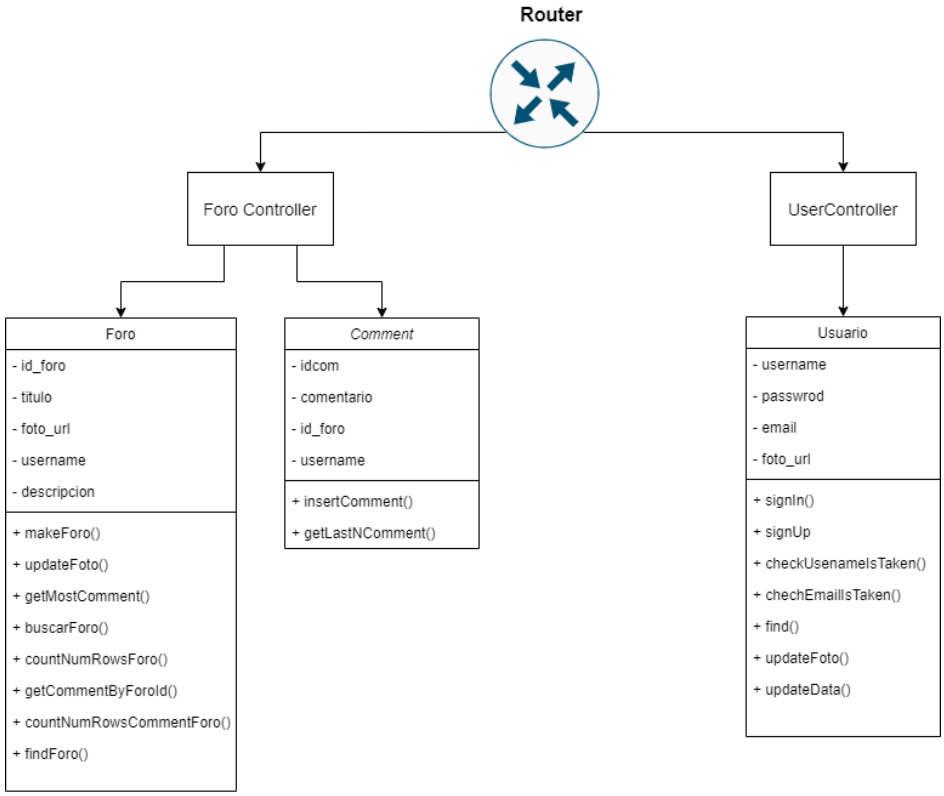
   Todas estas funcionalidades están implantadas y correctamente funcionales.

Análisis y diseño

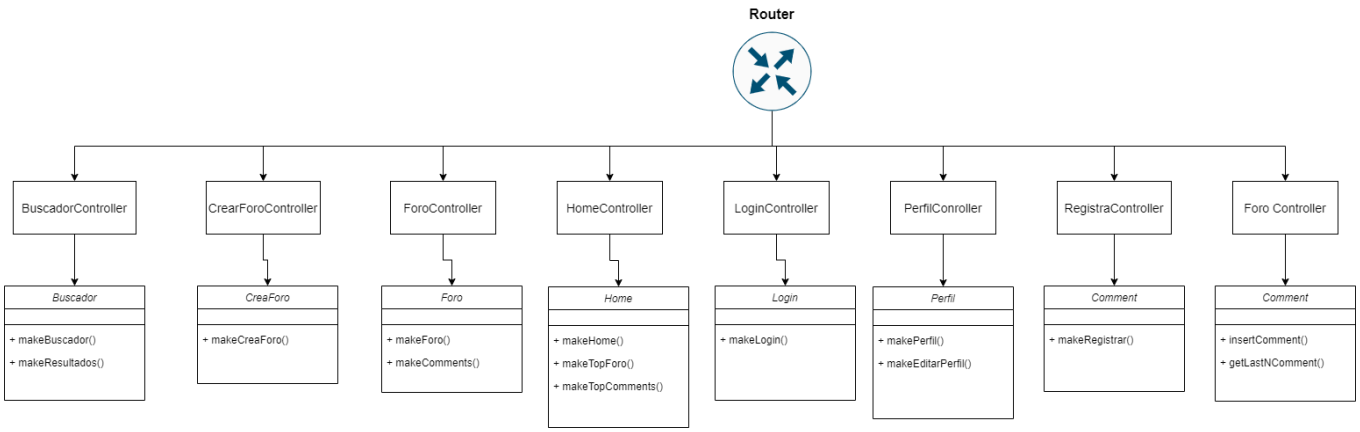
La base de datos se ha realizado con MySql, es decir se ha utilizado una base de datos relacional, la estructura de esta es la siguiente:



Para el control de esta se ha realizado un api que controla las llamadas al backend, con la siguiente estructura:



Con lo que respecta al frontend se ha utilizado la modelo vista controlador MVC facilitar la construcción de las vistas, y una mayor estabilidad. Con el mismo diseño de router, controlador y modelo que el api del backend:



Como en el frontend necesita realzar llamadas al api se ha realizado en este un controlador de las llamadas al api, para que sea mas seguro. **BackendConxController.php.**

Para la paginación se ha realizado un modelo común que es capaz de realizar la paginación tanto como para el buscador o como para los comentarios del foro.

Para la carga dinámica de datos en el buscador se ha utilizado JavaScript con Ajax y para el estilo se ha utilizado css.

Implementación

Backend:

Para la implementación se ha utilizado el PHP orientado a objetos y se ha aplicado el patrón MVC.

Se ha replicado el diseño de Laravel, mediante el modelado de rutas y Composer para añadir las dependencias necesarias.

Router.php:

dir: WebForoUpna/backend/public/router.php

Este fichero consta de todas las rutas accesibles con las que se puede interactuar con el api, ejem:

```
48  /*----------  User Routes  ----------*/
49  $route->post(basename(__FILE__) . '/user/signup', [UserController::class, 'signUp']);
50  $route->post(basename(__FILE__) . '/user/login', [UserController::class, 'signIn']);
51  $route->get(basename(__FILE__) . '/user/logout', [UserController::class, 'logout']);
52  $route->get(basename(__FILE__) . '/user/getUser', [UserController::class, 'getUser']);
53  $route->get(basename(__FILE__) . '/user/find/{username}', [UserController::class, 'findUse
54  $route->post(basename(__FILE__) . '/user/updata', [UserController::class, 'upData']);
55  $route->post(basename(__FILE__) . '/user/upfoto', [UserController::class, 'uploadFoto']);
56  /*----------  End of User Routes  ----------*/
```

Controladores:

dir: WebForoUpna/backend/src/Controller

En esta carpeta se han implementado los controladores listados en la descrita en la sección anterior, estos se encargan de sanear los datos y verificar que son válidos, verificado además que el cliente tenga los privilegios necesarios si es requerido, como para crear foro o comentar.

Modelos:

dir: WebForoUpna/backend/src/Models

En esta carpeta se encuentran todos los modelos listados en la gráfica de la sección anterior. Estos se encargan de realizar las acciones requeridas contra la base de datos, insertar, actualizar, obtener datos o verificar.

Database.php:

dir: WebForoUpna/backend/src/Models/Database.php

Para la conexión con la base de datos se ha utilizado el modelo singelton que establece una única conexión para no sobrecargar el api.

Config.php:

dir: WebForoUpna/backend/src/config/config.php

Este fichero contiene los datos para autentificarse con la base de datos.

Sanitazer.php:

dir: WebForoUpna/backend/src/Controllers/Sanitaizer.php

Este se encarga de limpiar los datos con los que se consultan al api de condigo malicioso.

Session.php:

dir: WebForoUpna/backend/src/Models/Session.php

Se encarga de mantener los datos de sesión y controlarlos.

FileController.php:

dir: WebForoUpna/backend/src/Controller/FileController.php

Este se encarga de guardar las fotos cunado se van a actualizar. Para el almacenamiento de las fotos se ha utilizado un sistema de fichero, dado que es más rápido, se guardan en la carpeta backend/public/imgs, dentro de esta podemos ver las fotos de perfil o foro en /perfil o / foro respectivamente.

FrontEnd:

Al igual que en le backend se ha utilizado para la implementación el diseño PHP orientado a objetos y se ha aplicado el patrón MVC.

Para la construcción de las vistas se replicado el funcionamiento de Angular.

Router.php:

dir: WebForoUpna/frontend/public/router.php

Este fichero consta de todas las rutas accesibles con las que el cliente puede interactuar para navegar por la web, ejem:

```
49  /*----------  Home Routes  ----------*/
50  $route->get(basename(__FILE__) . '/', [HomeController::class, 'redirect']);
51  $route->get(basename(__FILE__) . '/home', [HomeController::class, 'showHome']);
52  /*----------  End of Home Routes  ----------*/
53
54  /*----------  Login Routes  ----------*/
55  $route->get(basename(__FILE__) . '/login', [LoginController::class, 'showLogin']);
56  /*----------  End of Login Routes  ----------*/
57
58  /*----------  Registrar Routes  ----------*/
59  $route->get(basename(__FILE__) . '/registrar', [RegistrarController::class, 'showRegistrar']
60  /*----------  End of Registrar Routes  ----------*/
61
62  /*----------  Buscador Routes  ----------*/
63  $route->get(basename(__FILE__) . '/buscador', [BuscadorController::class, 'showBuscador']);
64  $route->post(basename(__FILE__) . '/buscador/find', [BuscadorController::class, 'buscar']);
65  /*----------  End of Buscador Routes  ----------*/
66
67  /*----------  Perfil Routes  ----------*/
```

Controladores:

dir: WebForoUpna/fronend/src/Controller

En esta carpeta se han implementado los controladores listados en la descrita en la sección anterior, estos se encargan de sanear los datos y verificar que son válidos, además si es necesario datos de la base de datos mediante llamadas al api.

Modelos:

dir: WebForoUpna/frontend/src/Models

En esta carpeta se encuentran todos los modelos listados en la gráfica de la sección anterior. Estos modelos se encargan de crear todas las vistas que se visualizan, modificándolas según si el usuario este logueado o no, por lo que se han creado plantillas fácilmente modificables al estilo de Angular, cada modelo consta de un template, css y js si es necesario.

/Templates:

dir: WebForoUpna/frontend/src/Models/templates/

En esta carpeta se pueden encontrar todas las vistas html que se utilizan en la web, para que sean fácilmente modificables por los modelos.

Dentro tenemos la carpeta /res/:

Aquí están tantos los css como los js en las carpetas /css y js respectivamente.

Si es requerido como en el caso de buscador se han descompuesto los html en componentes, al estilo de Angular, para la composición iterativa de los elementos repetidos.

Paginado.php:

dir: WebForoUpna/frontend/src/Models/Paginado.php

Como este es un componente que se repite tano en diseño y funcionalidad en dos partes de la web se ha realizado este modelo que permite construir el html de esta sección independientemente de la página a la que pertenezca.

Session.php:

dir: WebForoUpna/frontend/src/Models/Session.php

Al igual que en el backend hacemos en el frontend un control de los datos de sesión con esta implementación.

BackendConx.php & BackendConxController.php:

dir: ../frontend/src/Controllers/BackendConxController.php

dir: ../frontend/src/Models/BackendConx.php

Como es necesario hacer llamadas al api se han creado estos elementos que permiten realizar una conexión con el api, el controlador sanea los datos y el modelo permite realizar una única conexión al api mediante el modelo singelton, para no sobrecargar el servicio del frontend. Para hacer las llamadas se ha utilizado la librería **Guzzle**.

Resurce.php & ResurceController.php:

dir: ../frontend/src/Controllers/ResurceController.php

dir: ../frontend/src/Models/Resurce.php

Estos elementos se han realizado debido a que la web necesita acceder a los recursos que están en carpetas del servidor, como lo css y js. Para que tener un acceso seguro a ellos se ha realizado eso, que da únicamente acceso a estos ficheros.

Header.php:

dir: WebForoUpna/frontend/src/Models/Header.php

Al estilo de angular esta es la vista principal donde se montan todas las demás, dado que el Header es un elemento persistente.

Sanitazer.php:

dir: WebForoUpna/frontend/src/Controllers/Sanitaizer.php

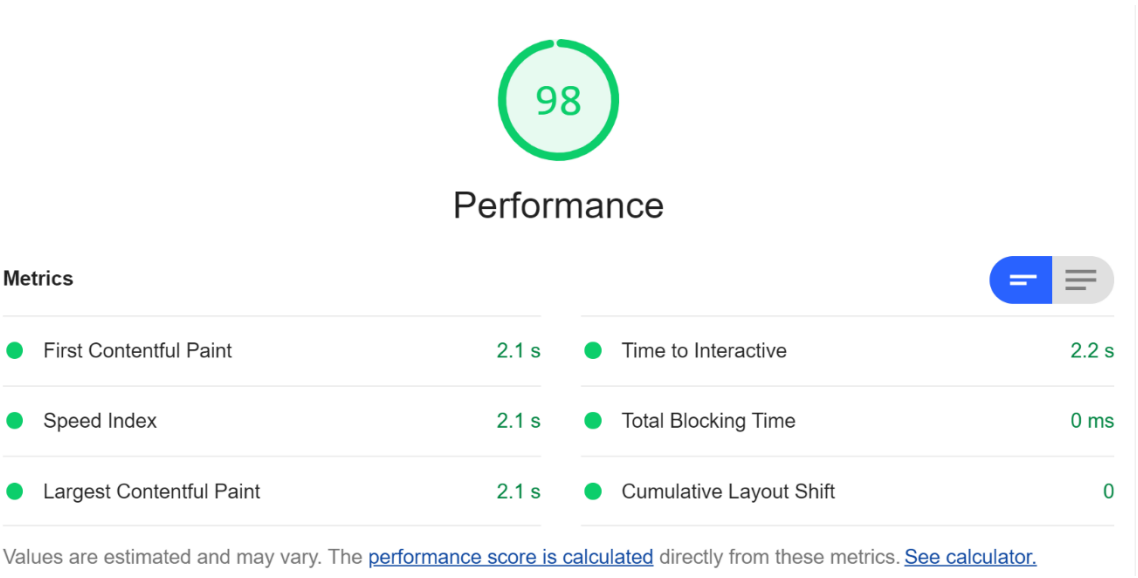
Este se encarga de limpiar los datos con los que se consultan al servicio del frontend de condigo malicioso.

Auditoria de Rendimiento y SEO

Para esto se ha utilizado la herramienta de Google Chrome **lighthouse**, accesible desde el menú F12, en una de estas pestañas.

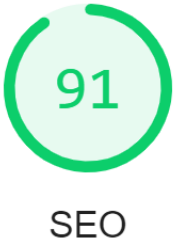
Se ha hecho la prueba en versión móvil, debido a que la herramienta valora en primer lugar la versión móvil, es decir si en el móvil tiene un buen rendimiento en la versión web el rendimiento será mucho mejor.

Se ha pasado esta prueba a todas la paginas de la web y se ha obtenido este resultado, +-1 dependiendo de la página:



Se ha utilizado la herramienta **lazy loading** en todas las etiquetas **img**, para que se carguen de forma dinámica cuando sea necesario, pese a que no hay muchas imágenes.

Con lo que respecta al **SEO**, se ha utilizado la también la herramienta **lighthouse** que ofrece un scanner se SEO, todas las pagina ofrecen una puntuación de 91 -+1:



Para las que imanes que se suben sean ligeras se recalan auténticamente, aun tamaño mucho más pequeño mediante el **FileController** del Backend.

Los diseños de las fotos están predefinidos en el css y no dependen de que, si se caga la foto o no, con lo que podemos evitar también CLS.

Se ha añadido la etiqueta meta con los keywords y la descripción de la página, además del favicon.

Accesibilidad

Con lo que respecta a la accesibilidad en primer lugar destacar que en los inputs de los formularios se les han puesto **labels** asociados para que se sepa la definición de cada uno.

Se ha utilizado la etiqueta role perteneciente la metodología **ARIA**, donde ha sido oportuno, para definir el role en un formulario, un contenedor. Esto se puede ver en el template de home, login, registrar. Se ha utilizado también la etiqueta **aria-describedby="..."** para describir que fin de un formulario, lo que contiene un contendor esto se pude encontrar en el contenedor de los resultados de búsqueda, formulario de registrar. Finalmente se ha utilizado **aria-required="true | false"**, con el fin de definir si los campos de un formulario son requeridos o no, esto se puede ver en todos los input de los formularios que hay en la página web.

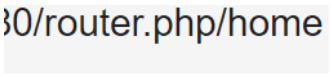
Análisis según las reglas WCAG:

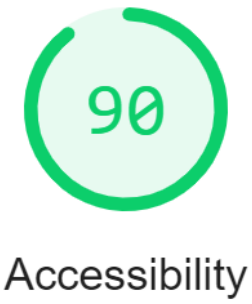
- Percibible:
  Para esto se pude ver que la página web tiene un diseño en color muy contrastado, dado que se ha utilizado el blanco y negro. Se han definido correctamente los límites de los contenidos siendo fácilmente divisables. Además, todas las imágenes tienen texto alternativo, incluso las que se añaden por el usuario.
- Operable:

Es fácilmente operable, la navegación entre las paginas es fácil mediante la barra de navegación, los formularios no tienen límite de tiempo. Se puede acceder desde cualquier página con el tabulador.

- Entendible:
  La interactuación es muy fácil, se proporciona el mismo esquema de una página del estilo, ofreciendo desde la ventana home datos del fin de la página que es la discusión en los foros. Todos los inputs tienen labels.
- Robusta:
  Esta página es totalmente operable desde cualquier dispositivo, ya que tiene un diseño multiplataforma que se ha implementado con css.

La herramienta **lighthouse** permite realizar un escáner de accesibilidad han todas las pagina se ha obtenido el valor siguiente +- 5 puntos:





Usabilidad

Para esto se ha utilizado las 10 heurísticas de Nielsen:

Visibilidad de estatus:

En todos los formularios se les de feedback a usuario mediante la librería **sweetalert2**, que proporciona respuestas muy amigables a los usuarios del estado el formulario.

Equivalencia de sistema con el mundo real:

Para el buscador se ha utilizado el icono de la lupa que identifica fácilmente esta herramienta, dado que es la más utilizada visible en Google.

Control por el usuario y libertad:

Esta se puede ver en el formulario de actualizar los datos del usuario, dado que, si los ha cambiado y no quiere hacer submit, con hacer click en la barra de navegación en el botón de perfil puede escapar de esa página.

Consistencia y estándar:

Esto se puede ver en el header, que es el que se utilizan en muchas páginas, con el nombre de la página web en grande arriba y que si se hace click en él se puede ir al home de la página, esto se hace como estándar. El navegador se encuentra en la parte superior como en la mayoría de las páginas.

Reconocimiento antes que recordar:

Todos los campos son fácilmente visibles, y no se interpolan en ningún caso.

Prevención de erro:

En los formularios se han establecido un control de los campos, y si un campo es erróneo en la respuesta se utiliza un **sweetalert2**¸ en el que describe el error.

Flexibilidad y eficiencia de uso:

Como la página tiene muy fin muy concreto que es el de foro, todas las funciones de la página web son accesibles desde la barra de navegación.

Diseño minimalista y estético:

El diseño desde el principio fue minimalista, donde solo se ha utilizado el blanco y negro. Sin sobrecargarlo sobre con colorines innecesarios.

Ayudar al usuario con errores:

En todos formularios se dan respuesta con la librería **sweetalert2**, diciendo si la acción se ha hecho con éxito o no.

Ayuda y documentación:

Esta parte no se ha implementado debido a que esta página es totalmente funcional sin necesidad de ayuda extra.

Se ha dejado usara esta página varios usuarios más y se ha podido ver en las respuestas que es fácilmente usable, sin necesitar información extra.

CI/CD Despliegue y Gestión Configuración

Para el despliegue se ha utilizado el GitLab runner CI/CD, que hace una copia de los archivos de /var/www/**pre** a la carpeta /var/www/**html**. El script implementado es el siguiente.

```
stages:
    - deploy

deploy_a:
    tags:
        - guacamole
    stage: deploy
    script:
        - echo "Copiamos backend a la crpeta html"
        - echo "$RootPass" | sudo -S cp -r WebForoUpna/backend/ /var/www/html/
        - sleep 5

deploy_b:
    tags:
        - guacamole
    stage: deploy
    script:
        - echo "Copiamo frontend a la carpeta html"
        - echo "$RootPass" | sudo -S cp -r WebForoUpna/frontend/ /var/www/html/
        - sleep 5
```

Para la gestión de proyecto se ha utilizado la herramienta de GitLab Boards, en la que se ha ido añadiendo el tiempo estimado y el tiempo gastado para cada vez que se hacia una tarea.

Se modifico en php.ini:

Para permitir subir fotos de tamaños más grandes:

upload_max_filesize: 20M

post_max_size: 20M

Para usar esa liberia que permite modificare el tamaño de las fotos:

extension=gd

Para instalar las dependencias hacer el directorio de backend y fontend:

$ composser install

El api de **backend** se ejecuta el **localhost** en el puerto **1234** y el servicio del **frontend** se ejecuta en el puerto **80**, abierto a cualquier ip. De este modo conseguimos aislar el acceso de backend al exterior para mayor seguridad.

Conclusiones

Se ha aprendido ha aprendido a utilizar php orientado a objetos, para el diseño de una web.

Se ha utilizado **Postman** para hacer la prueba de las llamadas al api de backend.

Se ha realizado una web con distintas metodologías de accesibilidad y usabilidad, cosa que es muy importante la sociedad.

Se ha realizado un diseño MVC para el frontend imitando a angular. Para el backend se ha utilizado un modelo se api, que gestionan todas las llamadas contra la base de datos.

En general se ha intentado implementar todo lo dado en la asignatura en este pequeño proyecto.

Anexo

- Composer
- Guzzle
- PHPRoute
- Autoload
- Postman
- Jquery
- Instalación de composser:

```
curl -sS https://getcomposer.org/installer -o composer-setup.php
sudo php composer-setup.php --install-dir=/usr/local/bin --
filename=composer
```