UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIERÍA INDUSTRIAL VALENCIA

TRABAJO FIN DE GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

DESARROLLO DE UNA PLATAFORMA WEB PARA LA GESTIÓN COLABORATIVA DE PROYECTOS MEDIANTE APLICACIONES BASADAS EN INTELIGENCIA ARTIFICIAL

AUTOR: JORDI LLORCA ROMÁN

TUTOR: JOSÉ ANTONIO GIL GÓMEZ

Curso Académico: 2020-21

RESUMEN

Este Trabajo de Fin de Grado tiene por objeto la creación de una plataforma web, enfocada en la publicación de proyectos de distinta índole, donde otras personas tendrán la posibilidad de ofrecer su ayuda para realizar estos proyectos.

La plataforma se realizará con el framework de JavaScript conocido como Angular para la parte del Frontend, y para la parte del Backend se realizará con el framework de PHP conocido como Symfony.

Además, se incluirá un sistema de recomendaciones personalizadas basadas en los anteriores proyectos que hayas participados mediante Inteligencia Artificial.

Palabras claves: Plataforma Web, Inteligencia Artificial, Colaboración, Proyectos, Comunidad; Ingeniería, Informática

RESUM

Aquest Treball de Fi de Grau té per objecte la creació d'una plataforma web, enfocada en la publicació de projectes de diferent índole, on altres persones tindran la possibilitat d'oferir la seva ajuda per realitzar aquests projectes.

La plataforma es realitzarà amb el framework de JavaScript conegut com Angular per la part del Frontend, i per la part del Backend es realitzarà amb el framework de PHP conegut com Symfony.

A més, s'inclourà un sistema de recomanacions personalitzades basades en els anteriors projectes que hagis participats mitjançant Intel·ligència Artificial.

Paraules clau: Plataforma web, Intel·ligència Artificial, Col·laboració, Projectes, Comunitat, Ingeniería, Informática

ABSTRACT

The purpose of this Final Degree Project is to create a web platform, focused on the publication of projects of different kinds, where other people will have the possibility to offer their help to carry out these projects.

The platform will be made with the JavaScript framework known as Angular for the Frontend part, and for the Backend part it will be made with the PHP framework known as Symfony.

In addition, a system of personalized recommendations based on the previous projects that you have participated through Artificial Intelligence will be included.

Keywords: Website, Artificial Intelligence, Collaboration, Projects, Community, Engineering, Informatics

DOCUMENTOS CONTENIDOS EN EL TFG

DOCUMENTO I: MEMORIA

DOCUMENTO II: PRESUPUESTOS

ÍNDICE DE FIGURAS

DOCUMENTO I: MEMORIA

1 Introducción

1.1 Antecedentes

El trabajo surge de una idea personal que pretende ofrecer una herramienta a los estudiantes para poder añadir practicidad a sus sesiones teóricas impartidas en la universidad o en la escuela. Mediante la creación de proyectos y la realización de estos se adquirirán unos conocimientos que de otra forma no se podrían adquirir.

Debido al mundo tan competitivo en que nos encontramos a día de hoy, este tipo de iniciativa cobra más importancia que nunca. Ya que a diferencia que hace años, el valor de un título universitario decae a de forma exponencial. Se tiende a volver a un mundo donde la experiencia y la practicidad cobrará una relevancia muy importante. Se puede ver constantemente como los estudiantes que se incorporan por primera vez al mundo laborar se ven desbordados en un principio, ya que no cuentan con las herramientas para hacer frente a este cambio.

Existen otras herramientas e iniciativas como Generación Espontánea que ofrecen un punto de vista muy similar, ya que permiten la creación de diversos equipos de distinta índole para desarrollar alguna actividad extraescolar, permitiendo así el desarrollo de los componentes del equipo. Este proyecto podría ser un complemento a las actividades desarrolladas en Generación Espontánea.

1.2 Objeto y motivación

La finalidad del presente proyecto es la implementación de una plataforma web que permita la creación de proyectos y la interacción con otros proyectos creados por distintos usuarios.

Es proyecto se lleva a cabo con distintas herramientas: Angular para el Frontend, Symfony para el Backend, MySQL como Base de Datos, Python como lenguaje de programación para crear el sistema de recomendaciones.

La motivación del trabajo es la obtención del título universitario del Grado de Ingeniería en Tecnologías Industriales. Este proyecto permite demostrar los conocimientos adquiridos por el autor.

La finalidad última de este proyecto es la creación de un entorno online donde se puedan propulsar ideas o proyectos de forma colaborativa.

Cabe nombrar que, a día de hoy, las distintas actividades que se llevan a cabo a nivel profesional suelen requerir de diversos campos de aplicación. Si bien esto es cierto para la mayoría de ámbitos, para ámbitos referentes a tecnología es aún más. Por ejemplo, para la creación de un robot, hará falta

personas dedicadas al diseño, a la fabricación de piezas, al ensamble de piezas, por otra parte, hará falta personas dedicadas al software, al desarrollo de inteligencia artificial, etc… De ahí que se haya dado un enfoque informático a este proyecto, aunque siempre se ha planteado desde el punto de vista de un ingeniero industrial.

1.3 Objetivos

Los requisitos mínimos que debe cumplir esta plataforma web son los siguientes:

- Disponer de suficiente capacidad para poder gestionar las distintas peticiones que se generen
- Sistema de interacción con los proyectos sencillo e intuitivo
- Diseño agradable y sencillo
- Rendimiento adecuado, sin mucho tiempo de carga

1.4 Estructura de la memoria

La memoria se ha estructurado de la siguiente manera:

Primeramente, hay una pequeña descripción del proyecto en el Estado del Arte, además de comentar algunas de las tecnologías más importantes en estos momentos.

Seguidamente tenemos una pequeña descripción de las herramientas usadas para desarrollar el proyecto., en la parte de métodos y herramientas, así como la metodología seguida para llevar a cabo el proyecto.

A continuación, están los resultados del proyecto, donde se profundiza en más detalle en cada uno de los aspectos más importantes. Dentro de este apartado, están incluidos todos los resultados referentes a la plataforma web y al modelo de inteligencia artificial

Finalmente, están las conclusiones, donde se analizan los puntos donde hay que mejorar y se plantea algunas implementaciones a futuro.

El código completo de todo el proyecto se puede encontrar en los siguientes enlaces:

Código del Frontend: https://github.com/llorcajor/mbipFront

Código del Backend: https://github.com/llorcajor/mbip

Sistema de recomendación: https://colab.research.google.com/gist/llorcajor/0f8ae7d89e6a405518d4571a99845d47/recommendation-system.ipynb

Página web: http://mbip.es/

Todo el código es libre y puede usarse para cualquier finalidad, bajo la licencia GNU.

2 Estado del Arte

En este apartado se hablará de algunos de los lenguajes de programación más conocidos y se dará un vistazo a las aplicaciones que tienen un enfoque parecido al de esta.

2.1 Definiciones y acrónimos

En este apartado se va a hacer una pequeña explicación de los términos que van a aparecer a partir de este punto en la memoria.

Frontend: El desarrollo web Frontend es todo aquel desarrollo que tenga que ver con la interfaz gráfica en la que el usuario va poder ver e interactuar de forma digital usando HTML, CSS y JavaScript

Backend: El Backend es la parte del desarrollo web que se encarga de que toda la lógica de una página web funcione. Se trata del conjunto de acciones que pasan en una web pero que no vemos como, por ejemplo, la comunicación con el servidor.

BBDD: Base de Datos

API: Una API es un conjunto de definiciones y protocolos que se utiliza para desarrollar e integrar el software de las aplicaciones. API significa interfaz de programación de aplicaciones.

Es una herramienta que permite comunicarse con otros programas sin saber cómo estos están hechos.

2.2 Lenguajes de programación

HTML (HyperText Marcage Language): Lenguaje informático que forma la estructura básica de muchas de las páginas web actuales.

CSS (Cascade Styling Sheet): Lenguaje de estilos utilizado para modificar la apariencia de documentos HTML. CSS describe como debería de mostrarse los elementos.

JS (JavaScript): Lenguaje de programación o de secuencias de comandos que permite implementar funciones complejas en páginas web.

Estos tres lenguajes se han convertido en el estándar en páginas web.

Python: Lenguaje de programación interpretado cuya principal filosofía es que sea legible por cualquier persona con conocimientos básicos de programación.

C: C es un lenguaje de programación (considerado como uno de los más importantes en la actualidad) con el cual se desarrollan tanto aplicaciones como sistemas operativos a la vez que forma la base de otros lenguajes más actuales como Java, C++ o C#.

2.3 Frameworks y librerias

En este apartado se va a hablar de los frameworks, que no son más que programas con ciertas funcionalidades ya creadas para que podamos desarrollar con mayor facilidad y velocidad, y de las librerías, que no son más que un conjunto de funciones ya definidas de las cuales se puede hacer uso sin tener que definirlas desde cero.

Podemos diferenciar dos grupos, los frameworks y librerías orientadas a lenguajes de Frontend (por parte del cliente, lo que vemos al entrar en una página web), y los frameworks y librerías orientadas a los lenguajes de Backend (por parte de servidor, todo lo que se hace por detrás en una página web)

2.3.1 Frameworks de Frontend

React: Es una biblioteca Javascript de código abierto diseñada para crear interfaces de usuario con el objetivo de facilitar el desarrollo de aplicaciones en una sola página. Es mantenido por Facebook y la comunidad de software libre.

Vue.js: Es un framework de JavaScript de código abierto para la construcción de interfaces de usuario y aplicaciones de una sola página. Fue creado por Evan You, y es mantenido por él y por el resto de los miembros activos del equipo central que provienen de diversas empresas como Netlify y Netguru.

2.3.2 Frameworks de Backend

Laravel: Es un framework de código abierto desarrollado para ayudar a desarrollar aplicaciones escritas en PHP. Su filosofía es desarrollar código PHP de forma elegante y simple, evitando el "código espagueti".

2.4 Aplicaciones relacionadas

La idea de este proyecto surge a raíz de otra plataforma web que desarrolla un concepto bastante parecido. Esta plataforma se trata de: https://teamerup.upv.es/

En esta web, se puede publicar una StartUp (explicando de que se trata, que es lo que se pretende conseguir y demás características del proyecto), y otros usuarios pueden decidir inscribirse en el proyecto. Si se acepta esta solicitud se pone en contacto a las dos personas interesadas mediante un correo electrónico.

El enfoque diferenciador que se busca en el proyecto es que no se pueden publicar únicamente StartUps, sino que se pueden publicar toda índole de proyectos, de distintos ámbitos. Se podría crear un proyecto relacionado con la creación de una página web, al igual que un proyecto relacionado con la creación de un cohete para participar en un torneo, o incluso la creación de un proyecto relacionado con la creación de paneles solares.

3 Métodos y herramientas

En este apartado se va a realizar un análisis más exhaustivo y en detalle de los componentes del proyecto.

3.1 Propósito

Como se ha comentado anteriormente, el propósito del proyecto es desarrollar una plataforma web que permita interactuar a personas permitiendo crear proyectos y solicitar unirse a estos para formar equipos.

La página web ofrecerá la opción de crear un proyecto con su descripción, su foto, su nombre y además habrá que seleccionar una Categoría que nos permitirá filtrar dicho proyecto. Al mismo tiempo estos proyectos que se han creado serán visibles para los usuarios registrados y estos podrán optar a inscribirse en dichos proyectos.

Los puntos que se pretenden conseguir con el proyecto son:

- Visualización de los proyectos creados
- Inscripción en los proyectos
- Creación de nuevos proyectos
- Página de detalles para cada proyecto
- Página de detalles para cada usuario
- Página de edición para cada usuario
- Página de edición para cada proyecto
- Visualización de los proyectos inscritos
- Página de visualización de los proyectos propios solicitados por otros usuarios

3.2 Tecnologías usadas

TypeScript: Es un lenguaje de programación y es un superconjunto de JavaScript, es decir, amplia las funcionalidades de JavaScript y cambia la sintaxis.

TypeScript compila a JavaScript.

Angular: Es un framework para aplicaciones web desarrollado en TypeScript, de código abierto, mantenido por Google, que se utiliza para crear y mantener aplicaciones web de una sola página.

PHP: Es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML.

Symfony: Es un framework diseñado para desarrollar aplicaciones web basado en el patrón Modelo Vista Controlador.

LAMP: Es el acrónimo usado para describir un sistema de infraestructura de internet que usa las siguientes herramientas:

- Linux, el sistema operativo;.
- Apache, el servidor web;
- MySQL/MariaDB, el gestor de bases de datos;
- PHP, el lenguaje de programación.

La combinación de estas tecnologías es usada principalmente para definir la infraestructura de un servidor web, utilizando un paradigma de programación para el desarrollo del sistema.

Este es el entorno local que se ha usado para probar y desarrollar toda la parte del Backend.

Postman: Es una herramienta que permite el envío de peticiones HTTP.

Se ha utilizado mucho a lo largo del proyecto para verificar el correcto funcionamiento de las diversas rutas desarrolladas en el Backend.

3.3 Librerías Auxiliares

Bootstrap: Es un framework se css y javascript diseñado para la creación de interfaces. Contiene un amplio abanico de componentes que pueden ser usados de forma sencilla en cualquier plataforma web.

Font-awesome: Es una serie de iconos que pueden ser incrustados directamente con un elemento html.

Serializer: Es un paquete de Symfony usado para convertir objetos a distintos formatos. Para este proyecto en concreto se ha usado para convertir objetos a formato JSON.

Mailer: Es un paquete de Symfony usado para crear y enviar correos electrónicos.

Knp Pagination Bundle: El paquete Knp Pagination Bundle de Symfony se usa para automatizar la paginación.

HttpFoundation: El componente HttpFoundation define una capa orientada a objetos para la especificación HTTP. En el caso de este proyecto se ha usado mayoritariamente para acceder a datos de una petición realizada por el Frontend al Backend.

3.4 Herramientas de desarrollo

Visual Studio Code: Es un editor de código fuente desarrollado por Microsoft para Windows, Linux y macOS. También permite trabajar con diversos lenguajes de programación, entre los que se pueden destacar PHP y TypeScript.

Git: Es un sistema de control de versiones.

GitHub: Es un repositorio para alojar proyectos utilizando el sistema de control de versiones Git.

3.5 Funcionalidades

En este apartado se va a hablar de las distintas funcionalidades que se pueden encontrar y de las limitaciones existentes.

3.5.1 Tipos de usuario

Únicamente existe un tipo de usuario, pero cada usuario va a poder interactuar de forma distinta con los proyectos creados por el mismo usuario o con proyectos creados por otro usuario distinto.

Se podrá editar o eliminar un proyecto por el usuario creador de dicho proyecto, y se podrá solicitar unirse a todos los proyectos pertenecientes a otros usuarios.

3.5.2 Dependencias del sistema

El sistema se tendrá que desplegar en un servidor que sea lo suficientemente potente para gestionar las peticiones que pudieran surgir.

3.6 Requisitos específicos

En este apartado se va a hablar de los requisitos específicos que hacen falta para que toda la plataforma funcione correctamente.

3.6.1 Requisitos funcionales

• **U-R1.** Registrarse. El sistema debe de ser capaz de registrar a cualquier usuario que lo desee.

• **U-R2.** Iniciar sesión. Todo usuario registrado debe ser capaz de iniciar sesión con su usuario y contraseña en cualquier momento.

• **U-R3**. Editar perfil. Los usuarios registrados deben tener la opción de poder editar sus datos siempre que lo requieran. Los datos a editar deben de ser: nombre, apellido, foto y correo electrónico.

• **P-R1**. Visualización de los proyectos. El sistema debe de mostrar todos los proyectos que no sean propios del usuario registrado.

• **P-R2**. Detalles del proyecto. Al pulsar sobre un proyecto, se debe de desplegar una página con toda la información sobre dicho proyecto.

• **P-R3**. Crear/Eliminar proyectos. Se debe de poder crear proyectos en cualquier momento y posteriormente eliminar dichos proyectos por el usuario que los ha creado.

• **P-R4.** Editar proyectos. El usuario que crea un proyecto debe de ser capaz de editarlos en cualquier momento. Debe de ser capaz de modificar el nombre, la descripción, la categoría y la foto.

• **P-R5.** Inscribirse en proyectos. Los usuarios deben ser capaces de inscribirse en cualquier proyecto que seleccionen.

• **RR-R1.** Mostrar las solicitudes recibidas por tus proyectos. El usuario podrá ver que otros usuarios han demandado inscribirse en alguno de sus proyectos. Además, tendrá la opción de rechazar o de aceptar la inscripción.

• **RR-R2.** Listar las inscripciones realizadas. Cualquier usuario siempre tendrá acceso a un listado de todas sus solicitudes realizadas, siempre que estas no hayan sido rechazadas.

4 Resultados

4.1 Base de Datos

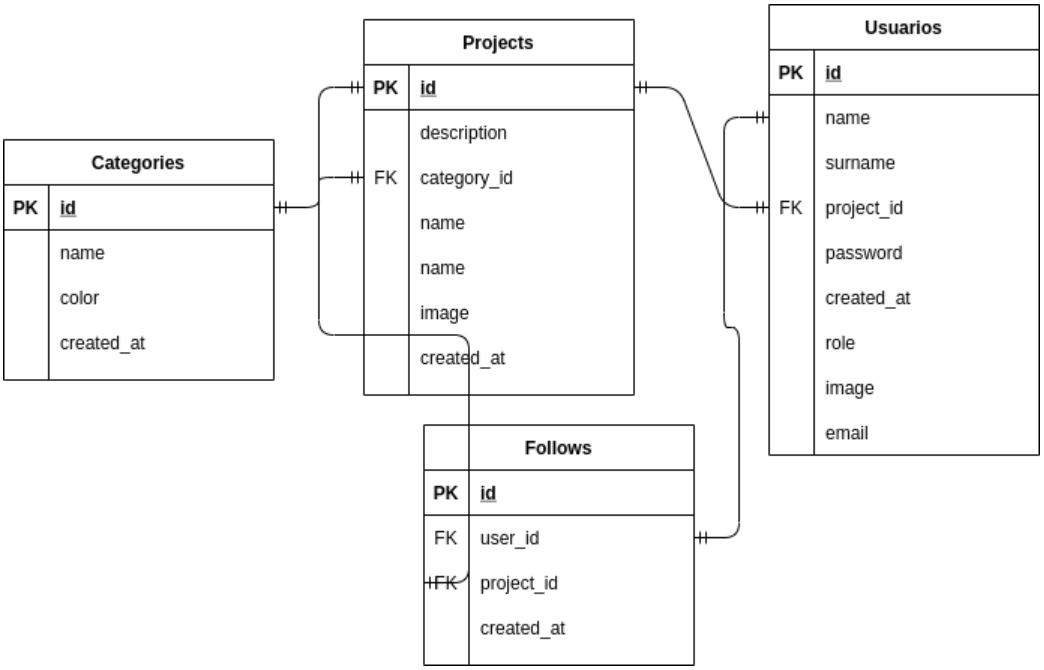

Figura 1. Base de Datos

Se puede observar las relaciones existentes entre las diferentes tablas.

Se tiene las siguientes Relaciones:

En la tabla de proyectos:

- Al crear un nuevo proyecto se almacena en la BBDD el id del usuario que ha creado dicho proyecto. El user_id sería una clave foránea.
- Al crear un nuevo proyecto se almacena en la BBDD el id de la categoría a la que pertenece el proyecto. El category_id sería una clave foránea

En la tabla de follows:

- Al crear una nuevo follow (solicitar unirse a un proyecto), se tendría que guardar en la BD el id del usuario que ha realizado dicho follow, junto con el id del proyecto al que se ha solicitado.

4.1.1 Categorías

En el proyecto se pueden encontrar las siguientes categorías:

- Informática
- Ingeniería
- Automovilismo
- Sanitario
- Social
- Artístico
- Culinario
- Empresarial
- Arquitectura
- Diseño

4.2 Rutas

En este apartado se va a ver las distintas rutas que hay en la plataforma web.

Las rutas sirven para moverse entre los distintos componentes que conforman la página web.

Podemos diferenciar dos tipos de rutas, las pertenecientes al Frontend, que son las encargadas de cambiar la parte visual, y las pertenecientes al Backend, que son las encargadas de gestionar toda la interacción con la base de datos.

4.2.1 Rutas del Frontend

En este apartado se puede observar todas las rutas que se usan en la parte del Frontend.

Hay que hacer mención a una función denominada canActivate, que se encarga de comprobar si se está identificado, en ese caso no habrá ningún problema, o si en cambio no se está identificado y se redirigirá a la ventana de login.

```
const appRoutes : Routes = [
    {path: '', component: InitialPageComponent},
    {path: 'myprojects', component: ProjectComponent},
    {path: 'myprojects/:page', component: ProjectComponent},
    {path: 'home', component: OtherProjectsComponent},
    {path: 'home/:page', component: OtherProjectsComponent},
    {path: 'login', component: LoginComponent},
    {path: 'logout/:sure', component: LoginComponent},
    {path: 'registro', component: RegisterComponent},
```

```
    {path: 'ajustes', component: UserEditComponent, canActivate: [IdentityGuar
d]},
    {path: 'project-
edit/:id', component: ProjectEditComponent, canActivate: [IdentityGuard]},
    {path: 'project-
detail/:id', component: ProjectDetailComponent, canActivate: [IdentityGuard]},
    {path: 'new-
project', component: ProjectNewComponent, canActivate: [IdentityGuard]},
    {path: 'requests', component: RequestsComponent},
    {path: 'myrequests', component: MyrequetsComponent},
    {path: 'user-
detail/:id', component: UserDetailComponent, canActivate: [IdentityGuard]}
];
```

4.2.2 Rutas del Backend

En este apartado se puede observar todas las rutas que se usan en la parte del Backend.

4.2.2.1 Usuario

```
create:
  path: /create
  controller: App\Controller\UserController::create
  methods: [POST]

login:
  path: /login
  controller: App\Controller\UserController::login
  methods: [POST]

user_edit:
  path: /user/edit
  controller: App\Controller\UserController::edit
  methods: [PUT]

user_detail:
  path: /user/detail/{id}
  controller: App\Controller\UserController::userDetail
  methods: [GET]
```

4.2.2.2 Proyecto

```yaml
project_new:
  path: /project/new
  controller: App\Controller\ProjectController::newProject
  methods: [POST]

project_edit:
  path: /project/edit/{id}
  controller: App\Controller\ProjectController::newProject
  methods: [PUT]

project_list:
  path: /project/list
  controller: App\Controller\ProjectController::projects
  methods: [GET]

other_project_list:
  path: /project/list_other
  controller: App\Controller\ProjectController::otherProjects
  methods: [GET]

project_detail:
  path: /project/detail/{id}
  controller: App\Controller\ProjectController::project
  methods: [GET]

project_remove:
  path: /project/remove/{id}
  controller: App\Controller\ProjectController::remove
  methods: [DELETE]
```

4.2.2.3 Follow

```yaml
follow:
  path: /follow
  controller: App\Controller\FollowController::index

follow_match:
  path: /follow/match/{id}
  controller: App\Controller\FollowController::onMatch
  methods: [GET]

follow_check_match:
  path: /follow/check
```

```
  controller: App\Controller\FollowController::checkMatch
  methods: [GET]

follow_my_check_match:
  path: /follow/mycheck
  controller: App\Controller\FollowController::checkMyMatchs
  methods: [GET]

follow_remove:
  path: /follow/remove/{id}
  controller: App\Controller\FollowController::remove
  methods: [DELETE]

mail:
  path: /follow/mail/{id}
  controller: App\Controller\FollowController::sendMail
  methods: [GET]
```

4.2.2.4 Categoría

```
category_all:
  path: /category/all
  controller: App\Controller\CategoryController::categories
  methods: [GET]
```

4.3 Modelos

Los modelos son la parte encargada de gestionar los datos. Definen como y que datos se van a guardar. En este caso, tenemos hasta cuatro tipos de modelos: Usuario, Proyecto, Categoría y Follow.

4.3.1 Modelos del Frontend

En este apartado se van a ver los modelos relacionados con la parte del Frontend.

4.3.1.1 Usuario

```
export class User{
    gettoken: string | undefined;
    constructor(
        public id: number,
```

```
        public name: string,
        public surname: string,
        public email: string,
        public image: string,
        public password: string,
        public role: string,
        public createdAt: string
    ){

    }
}
```

4.3.1.2 Proyecto

```
export class Project{
    public id: number;
    public name: string;
    public description: string;
    public imageUrl: string;
    public category: Category;
    public user: User;
    public createdAt: string;


    constructor(id:number, name:string, desc:string, imagePath:string, categor
y: Category, userId: User, createdAt: string){
        this.id= id;
        this.name = name;
        this.description= desc;
        this.imageUrl = imagePath;
        this.category = category;
        this.user = userId;
        this.createdAt = createdAt;

    }
}
```

4.3.1.3 Categoría

```
export class Category{
    public id: number;
    public name: string;
    public color: string;
    public createdAt: string;
```

```
    constructor(id:number, name:string, color:string,  createdAt: string){
        this.id= id;
        this.name = name;
        this.color = color;
        this.createdAt = createdAt;

    }
}
```

4.3.2 Modelos del Backend

En este apartado se van a ver los modelos relacionados con la parte del Backend.

Respecto a estos modelos hay que tener en consideración lo siguiente:

Todos los modelos tienen métodos getters y setters para poder obtener y modificar los valores de las variables desde otros componentes.

Todos los modelos tienen una función denominada jsonSerialize encargada de delimitar las características de los modelos que van a ser descompuestos cuando estos mismos modelos se usen en otros componentes.

4.3.2.1 Usuario

```
<?php

namespace App\Entity;

use Doctrine\ORM\Mapping as ORM;

/**
 * Users
 *
 * @ORM\Table(name="users")
 * @ORM\Entity
 */
class User implements \JsonSerializable
{
    /**
     * @var int
     *
     * @ORM\Column(name="id", type="integer", nullable=false)
     * @ORM\Id
```

```
     * @ORM\GeneratedValue(strategy="IDENTITY")
     */
    private $id;

    /**
     * @var string
     *
     * @ORM\Column(name="name", type="string", length=50, nullable=false)
     */
    private $name;

    /**
     * @var string|null
     *
     * @ORM\Column(name="surname", type="string", length=150, nullable=true)
     */
    private $surname;

    /**
     * @var string
     *
     * @ORM\Column(name="email", type="string", length=255, nullable=false)
     */
    private $email;

    /**
     * @var string|null
     *
     * @ORM\Column(name="image", type="string", length=255, nullable=true)
     */
    private $image;

    /**
     * @var string
     *
     * @ORM\Column(name="password", type="string", length=255, nullable=false)
     */
    private $password;

    /**
     * @var string|null
     *
     * @ORM\Column(name="role", type="string", length=20, nullable=true)
     */
    private $role;

    /**
```

```
     * @var \DateTime|null
     *
     * @ORM\Column(name="created_at", type="datetime", nullable=true, options=
{"default"="CURRENT_TIMESTAMP"})
     */
    private $createdAt = 'CURRENT_TIMESTAMP';


    public function jsonSerialize(): array
    {
        return [
            'id' => $this->id,
            'name' => $this->name,
            'surname' => $this->surname,
            'email' => $this->email,
            'image' => $this->image
        ];
    }
}
```

4.3.2.2 Proyecto

```
<?php

namespace App\Entity;

use Doctrine\ORM\Mapping as ORM;
use App\Entity\User;

/**
 * Project
 *
 * @ORM\Table(name="projects", indexes={@ORM\Index(name="fk_project_category",
 columns={"category_id"}), @ORM\Index(name="fk_project_user", columns={"user_i
d"})})
 * @ORM\Entity
 */
class Project implements \JsonSerializable
{
    /**
     * @var int
     *
     * @ORM\Column(name="id", type="integer", nullable=false)
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="IDENTITY")
```

```
     */
    private $id;

    /**
     * @var string
     *
     * @ORM\Column(name="name", type="string", length=50, nullable=false)
     */
    private $name;

    /**
     * @var string
     *
     * @ORM\Column(name="description", type="string", length=255, nullable=fal
se)
     */
    private $description;

    /**
     * @var string|null
     *
     * @ORM\Column(name="image", type="string", length=255, nullable=true)
     */
    private $image;

    /**
     * @var \DateTime|null

     *
     * @ORM\Column(name="created_at", type="datetime", nullable=true, options=
{"default"="CURRENT_TIMESTAMP"})
     */
    private $createdAt = 'CURRENT_TIMESTAMP';

    /**
     * @var \App\Entity\User
     *
     * @ORM\ManyToOne(targetEntity="User")
     * @ORM\JoinColumns({
     *   @ORM\JoinColumn(name="user_id", referencedColumnName="id")
     * })
     */
    private $user;

    /**
     * @var \App\Entity\Category
     *
```

```
     * @ORM\ManyToOne(targetEntity="Category")
     * @ORM\JoinColumns({
     *   @ORM\JoinColumn(name="category_id", referencedColumnName="id")
     * })
     */
    private $category;


    public function jsonSerialize(): array
    {
        return [
            'id' => $this->id,
            'name' => $this->name,
            'description' => $this->description,
            'imageUrl' => $this->image,
            'category' => $this->category,
            'user' => $this->user
        ];
    }
}
```

4.3.2.3 Follow

```
<?php

namespace App\Entity;

use Doctrine\ORM\Mapping as ORM;
use App\Entity\User;
use App\Entity\Project;

/**
 * Follows
 *
 * @ORM\Table(name="follows", indexes={@ORM\Index(name="fk_follow_project", co
lumns={"project_id"}), @ORM\Index(name="fk_follow_user", columns={"user_id"})}
)
 * @ORM\Entity
 */
class Follow implements \JsonSerializable
{
    /**
     * @var int
     *
```

```
     * @ORM\Column(name="id", type="integer", nullable=false)
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="IDENTITY")
     */
    private $id;

    /**
     * @var \DateTime|null
     *
     * @ORM\Column(name="created_at", type="datetime", nullable=true, options=
{"default"="CURRENT_TIMESTAMP"})
     */
    private $createdAt = 'CURRENT_TIMESTAMP';

    /**
     * @var \Users
     *
     * @ORM\ManyToOne(targetEntity="User")
     * @ORM\JoinColumns({
     *   @ORM\JoinColumn(name="user_id", referencedColumnName="id")
     * })
     */
    private $user;

    /**
     * @var \Projects
     *
     * @ORM\ManyToOne(targetEntity="Project")

     * @ORM\JoinColumns({
     *   @ORM\JoinColumn(name="project_id", referencedColumnName="id")
     * })
     */
    private $project;
```

```
    public function jsonSerialize(): array
    {
        return [
            'id' => $this->id,
            'project' => $this->project,
            'user' => $this->user
        ];
    }
}
```

4.3.2.4 Categoría

```php
<?php

namespace App\Entity;

use Doctrine\ORM\Mapping as ORM;

/**
 * Category
 *
 * @ORM\Table(name="categories")
 * @ORM\Entity
 */
class Category implements \JsonSerializable
{
    /**
     * @var int
     *
     * @ORM\Column(name="id", type="integer", nullable=false)
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="IDENTITY")
     */
    private $id;

    /**
     * @var string
     *
     * @ORM\Column(name="name", type="string", length=50, nullable=false)
     */
    private $name;

    /**
     * @var string|null
     *
     * @ORM\Column(name="color", type="string", length=50, nullable=true)
     */
    private $color;

    /**
     * @var \DateTime|null
     *
     * @ORM\Column(name="created_at", type="datetime", nullable=true, options=
{"default"="CURRENT_TIMESTAMP"})
```

```
     */
    private $createdAt = 'CURRENT_TIMESTAMP';


    public function jsonSerialize(): array
    {
        return [
            'id' => $this->id,
            'name' => $this->name,
            'color' => $this->color
        ];
    }
}
```

4.4 Funciones

En este apartado se va a ver en más detalle las funciones más importantes que se usan.

Se diferencian en dos grupos, Frontend y en Backend:

4.4.1 Funciones del Backend

Antes de empezar con las funciones, hay que hacer mención de la función resjson, ya que es muy usada en todas las funciones del Backend.

```
private function resjson($data)
    {
        // Serializar datos con serializer
        $json = $this->get('serializer')->serialize($data, 'json');

        // Response con httpfoundation
        $response = new Response();


        // Asignar contenido a la respuesta
        $response->setContent($json);

        // Indicar formato de respuesta
        $response->headers->set('Content-Type', 'application/json');


        // Devolver la respuesta
```

```
        return $response;
    }
```

También hay que hacer referencia a una función que se ejecuta cada vez que inicia sesión un usuario. Esta función genera un código que sirve para verificar la autenticidad del usuario. Este código se denomina en muchas funciones token y gracias a esto se puede mejorar en gran medida la seguridad de la plataforma web, protegiendo así la información de los distintos usuarios.

4.4.1.1 Funciones relacionadas con Usuario

```
    public function create(Request $request)
    {
        // Recoger la información que se recibe de la peticón
        $json = $request->get('json', null);
        // Decodificar el json
        $params = json_decode($json);
        // Respuesta por defecto
        $data = [
            'status' => 'error',
            'code' => 200,
            'message' => 'Error creating the user',
        ];

        // Comprobar y validar los datos
        if ($json != null) {
            $name = (!empty($params->name)) ? $params->name : null;
            $surname = (!empty($params->surname)) ? $params->surname : null;
            $email = (!empty($params->email)) ? $params->email : null;
            $password = (!empty($params->password)) ? $params-
>password : null;
        }


        // Crear el objeto usuario
        $user = new User();
        $user->setName($name);
        $user->setSurname($surname);
        $user->setEmail($email);
        $user->setRole('USER');
        $user->setCreatedAt(new \DateTime('now'));

        // Codificar la contraseña
        $pwd = hash('sha256', $password);
```

```
        $user->setPassword($pwd);
        $data = $user;

        // Comprobar duplicados
        $em = $this->getDoctrine()->getManager();

        $user_repo = $this->getDoctrine()->getRepository(User::class);
        $isset_user = $user_repo->findBy(array(
            'email' => $email
        ));

        // Si no hay duplicados, guardar en la bbdd
        if (count($isset_user) == 0) {
            $em->persist($user);
            $em->flush();
      // Devolver la respuesta

            $data = [
                'status' => 'succes',
                'code' => 200,
                'message' => 'The user has been created',
                'user' => $user
            ];
        } else {
            $data = [
                'status' => 'error',
                'code' => 400,
                'message' => 'The user already exist'
            ];
        }
        // Hacer la respuesta en json
        return $this->resjson($data);
    }


    public function login(Request $request, JwtAuth $jwt_auth)
    {
        // Recibir los datos por post
        $json = $request->get('json', null);
        $params = json_decode($json);


        // Respuesta por defecto
        $data = [
            'status' => 'error',
            'code' => 200,
            'message' => 'El usuario no se ha podido identificar'
```

```
        ];

        // Comprobar y validar los datos
        if ($json != null) {
            $email = (!empty($params->email)) ? $params->email : null;
            $password = (!empty($params->password)) ? $params-
>password : null;
            $gettoken = (!empty($params->gettoken)) ? $params-
>gettoken : null;

            $validator = Validation::createValidator();
            $validate_email = $validator->validate($email, [
                new Email()
            ]);

            if (!empty($email) && !empty($password) && count($validate_email)
== 0) {
                // Cifrar contraseña
                $pwd = hash('sha256', $password);
                // Llamar servicio para identificar usuario
                if ($gettoken) {
                    $signup = $jwt_auth->signup($email, $pwd, $gettoken);
                } else {
                    $signup = $jwt_auth->signup($email, $pwd);
                }

                return new JsonResponse($signup);
            }
        }

        // Devolver respuesta en json

        return $this->resjson($data);
    }

    public function edit(Request $request, JwtAuth $jwt_auth)
    {
        // Recoger la cabecera de autenticación
        $token = $request->headers->get('Authorization');
        // Comprobar si el token es correcto
        $authCheck = $jwt_auth->checkToken($token);
        // Respuesta por defecto
        $data = [
            'status' => 'error',
            'code' => 400,
            'message' => 'Usuario NO ACTUALIZADO'
```

```
        ];


        if ($authCheck) {


            // Conseguir Entity Manager
            $em = $this->getDoctrine()->getManager();
            // Conseguir datos usuarios
            $identity = $jwt_auth->checkToken($token, true);

            // Conseguir el usuario a actualizar
            $user_repo = $this->getDoctrine()->getRepository(User::class);
            $user = $user_repo->findOneBy([
                'id' => $identity->sub
            ]);
            // Recoger datos por POST
            $json = $request->get('json');
            $params = json_decode($json);
            // Comprobar y validar datos
            if (!empty($json)) {
                $name = (!empty($params->name)) ? $params->name : null;
                $surname = (!empty($params->surname)) ? $params-
>surname : null;
                $email = (!empty($params->email)) ? $params->email : null;
                $image = (!empty($params->image)) ? $params->image : null;
                // Asignar nuevos datos al objeto usuario
                $user->setEmail($email);
                $user->setName($name);
                $user->setSurname($surname);
                $user->setImage($image);
                // Comprobar duplicados
                $isset_user = $user_repo->findBy([
                    'email' => $email
                ]);

                if (count($isset_user) == 0 || $identity->email == $email) {
                    // Guardar cambios en la bbdd
                    $em->persist($user);
                    $em->flush();

                    $data = [
                        'status' => 'success',
                        'code' => 200,
                        'message' => 'Usuario ACTUALIZADO',
                        'user' => $user
                    ];
```

```
                } else {
                    $data = [
                        'status' => 'error',
                        'code' => 400,
                        'message' => 'No puedes usar ese email'
                    ];
                }
            }

        }
       // Devolver la respuesta en json

        return $this->resjson($data);
    }

    public function userDetail(Request $request, JwtAuth $jwt_auth, $id = null
)
    {
        $data = [
            'status' => 'error',
            'code' => 404,
            'message' => 'Usuario no encontrado',
            'id' => $id
        ];
        // Conseguir token
        $token = $request->headers->get('Authorization');
       // Comprobar token
        $authCheck = $jwt_auth->checkToken($token);
        if ($authCheck) {
       // Obtener usuario
            $user = $this->getDoctrine()->getRepository(User::class)-
>findOneBy([
                'id' => $id
            ]);


            $data = [
                'status' => 'success',
                'code' => 200,
                'user' => $user
            ];
        }

        // Devolver una respuesta en json

        return $this->resjson($data);
    }
```

4.4.1.2 Funciones relacionadas con Proyecto

```
   public function newProject(Request $request, JwtAuth $jwt_auth, $id = null)
    {
        $data = [
            'status' => 'error',
            'code' => 400,
            'message' => 'El proyecto no ha podido crearse'
        ];


        // Recoger el token
        $token = $request->headers->get('Authorization', null);

// Comprobar si es correcto
        $authCheck = $jwt_auth->checkToken($token);



        if ($authCheck) {
            // Recoger datos por post
            $json = $request->get('json', null);
            $params = json_decode($json);
            // Recoger el objeto del usuario identificado
            $identity = $jwt_auth->checkToken($token, true);

            // Comprobar y validar datos
            if (!empty($json)) {
                $user_id = ($identity->sub != null) ? $identity->sub : null;
                $title = (!empty($params->name)) ? $params->name : null;
                $description = (!empty($params->description)) ? $params-
>description : null;
                $category_id = (!empty($params->category)) ? $params-
>category : null;
                $image = (!empty($params->imageUrl)) ? $params-
>imageUrl : null;


                if (!empty($user_id) && !empty($title)) {


                    $em = $this->getDoctrine()->getManager();
                    $user = new User();
                  // Obtener usuario
                    $user = $this->getDoctrine()->getRepository(User::class)-
>findOneBy([
                        'id' => $user_id
```

```
                    ]);
                    $category = new Category();
                // Obtener categoria
                    $category = $this->getDoctrine()-
>getRepository(Category::class)->findOneBy([
                        'id' => $category_id
                    ]);

                    if ($id == null) {
                        // Rellenar el Objeto Proyecto
                        $project = new Project();
                        $project->setUser($user);
                        $project->setName($title);
                        $project->setDescription($description);
                        $project->setCategory($category);
                        $project->setImage($image);

                        $createdAt = new \DateTime('now');
                        $project->setCreatedAt($createdAt);



                // Guardar en bbd

                        $em->persist($project);
                        $em->flush();

                        $data = [
                            'status' => 'success',
                            'code' => 200,
                            'message' => 'El proyecto se ha guardado',
                            'proyecto' => $project
                        ];
                    } else {
                        // Actualizar proyecto
                        $project = $this->getDoctrine()-
>getRepository(Project::class)->findOneBy([
                            'id' => $id,
                            'user' => $identity->sub
                        ]);

                        if ($project && is_object($project)) {
                        // Modificar datos del proyecto
                            $project->setName($title);
                            $project->setDescription($description);
                        // Guardar bbdd
                            $em->persist($project);
```

```
                            $em->flush();

                            $data = [
                                'status' => 'success',
                                'code' => 200,
                                'message' => 'El proyecto se ha actualizado',
                                'proyecto' => $project
                            ];
                        }
                    }
                }
            }
        }


        // Devolver respuesta en json

        return $this->resjson($data);
    }









       public function projects(Request $request, JwtAuth $jwt_auth, Paginator
Interface $paginator)
    {
        $data = [
            'status' => 'error',
            'code' => 404,
            'message' => 'No se pueden listar los proyectos'
        ];
        // Recoger la cabecera de autenticación
        $token = $request->headers->get('Authorization');

        // Comprobar token
        $authCheck = $jwt_auth->checkToken($token);

        if ($authCheck) {
            // Conseguir identidad del usuario
            $identity = $jwt_auth->checkToken($token, true);

            $em = $this->getDoctrine()->getManager();

            // Configurar el bundle de paginación
```

```
            $dql = "SELECT v FROM App\Entity\Project v WHERE v.user = {$identi
ty->sub} ORDER BY v.id DESC";
            $query = $em->createQuery($dql);

            // Hacer consulta paginación
            $page = $request->query->getInt('page', 1);
            $items_per_page = 3;

            // Invocar paginación
            $pagination = $paginator-
>paginate($query, $page, $items_per_page);
            $total = $pagination->getTotalItemCount();
            // Preparar array para enviar
            $data = [
                'status' => 'successs',
                'code' => 200,
                'total_items_count' => $total,
                'page_actual' => $page,
                'items_per_page' => $items_per_page,
                'total_page' => ceil($total / $items_per_page),
                'projects' => $pagination,
                'user_id' => $identity->sub

            ];
        }
      // Devolver la respuesta en json
        return $this->resjson($data);
    }




      public function project(Request $request, JwtAuth $jwt_auth, $id = null)
    {
        $data = [
            'status' => 'error',
            'code' => 404,
            'message' => 'Proyecto no encontrado',
            'id' => $id
        ];
        // Obtener token
        $token = $request->headers->get('Authorization');
        $authCheck = $jwt_auth->checkToken($token);
        if ($authCheck) {
            // Obtener identidad
            $identity = $jwt_auth->checkToken($token, true);
            // Obtener el proyecto
```

```
            $project = $this->getDoctrine()->getRepository(Project::class)-
>findOneBy([
                'id' => $id,
                'user' => $identity->sub
            ]);
            // Obtener si el proyecto existe
            if ($project && is_object($project)) {
                $data = [
                    'status' => 'success',
                    'code' => 200,
                    'project' => $project
                ];
            }
        }

        // Devolver una respuesta en json

        return $this->resjson($data);
    }

    public function remove(Request $request, JwtAuth $jwt_auth, $id = null)
    {
        $data = [
            'status' => 'error',
            'code' => 404,
            'message' => 'Proyecto no encontrado'
        ];
      // Obtener token
        $token = $request->headers->get('Authorization');

        $authCheck = $jwt_auth->checkToken($token);

        if ($authCheck) {
            // Obtener identidad
            $identity = $jwt_auth->checkToken($token, true);


            $doctrine = $this->getDoctrine();
            $em = $doctrine->getManager();
            // Obtener proyecto
            $project = $doctrine->getRepository(Project::class)->findOneBy([
                'id' => $id
            ]);

            if ($project && is_object($project) && $identity->sub == $project-
>getUser()->getId()) {
                    // Eliminar de la bbdd
```

```
                $em->remove($project);
                $em->flush();

                $data = [
                    'status' => 'success',
                    'code' => 200,
                    'message' => 'Proyecto borrado'
                ];
            }
        }
      // Devolver respuesta en json
        return $this->resjson($data);
    }

    public function otherProjects(Request $request, JwtAuth $jwt_auth, Paginat
orInterface $paginator)
    {
        $data = [
            'status' => 'error',
            'code' => 404,
            'message' => 'No se pueden listar los proyectos'
        ];
        // Recoger la cabecera de autenticación
        $token = $request->headers->get('Authorization');

        // Comprobar token
        $authCheck = $jwt_auth->checkToken($token);

        if ($authCheck) {
            // Conseguir identidad del usuario
            $identity = $jwt_auth->checkToken($token, true);

            $em = $this->getDoctrine()->getManager();

            // Configurar el bundle de paginación
            $dql = "SELECT v FROM App\Entity\Project v WHERE v.user != {$ident
ity->sub} ORDER BY v.id DESC";
            $query = $em->createQuery($dql);

            // Hacer consulta paginación
            $page = $request->query->getInt('page', 1);

            $items_per_page = 6;


            // Invocar paginación
```

```
            $pagination = $paginator-
>paginate($query, $page, $items_per_page);
            $total = $pagination->getTotalItemCount();
            // Preparar array para enviar
            $data = [
                'status' => 'success',
                'code' => 200,
                'total_items_count' => $total,
                'page_actual' => $page,
                'items_per_page' => $items_per_page,
                'total_page' => ceil($total / $items_per_page),
                'projects' => $pagination,
                'user_id' => $identity->sub

            ];
        }
      // Devolver respuesta en json
        return $this->resjson($data);
    }
```

4.4.1.3 Funciones relacionadas con Follow

```
    public function onMatch(Request $request, JwtAuth $jwt_auth, $id = null)
    {
        $data = [
            'status' => 'error',
            'code' => 404,
            'message' => 'Error al dar match'
        ];
        // Recoger el token
        $token = $request->headers->get('Authorization', null);
        // Comprobar si es correcto
        $authCheck = $jwt_auth->checkToken($token);

        if ($authCheck) {
            // Obtener identidad del usuario
            $identity = $jwt_auth->checkToken($token, true);

            $em = $this->getDoctrine()->getManager();

            if ($id != null) {
            // Comprobar duplicados
                $user_id = ($identity->sub != null) ? $identity->sub : null;
```

```
                $follow_created = $this->getDoctrine()-
>getRepository(Follow::class)->findOneBy([
                    'user' => $user_id,
                    'project' => $id


                ]);

                if (!$follow_created) {
                   // Obtener usuario
                    $user = $this->getDoctrine()->getRepository(User::class)-
>findOneBy([
                        'id' => $user_id
                    ]);
                   // Obtener proyecto
                    $project = $this->getDoctrine()-
>getRepository(Project::class)->findOneBy([
                        'id' => $id
                    ]);
                   // Crear objeto follow

                    $follow = new Follow();
                    $follow->setProject($project);
                    $follow->setUser($user);
                    $createdAt = new \DateTime('now');
                    $follow->setCreatedAt($createdAt);

                   // Guardar en bbdd

                    $em->persist($follow);
                    $em->flush();

                    $data = [
                        'status' => 'success',
                        'code' => 200,
                        'message' => 'Match creado'
                    ];
                }
            }

            $data = [
                'status' => 'error',
                'code' => 404,
                'message' => 'Error al dar match'
            ];
        }
       // Devolver respuesta en json
```

```
        return $this->resjson($data);
    }




      public function checkMatch(Request $request, JwtAuth $jwt_auth, Paginato
rInterface $paginator)
    {
        $data = [
            'status' => 'error',
            'code' => 404,
            'message' => 'Error al buscar los match'
        ];

        // Recoger el token
        $token = $request->headers->get('Authorization', null);
        // Comprobar si es correcto
        $authCheck = $jwt_auth->checkToken($token);

        if ($authCheck) {
            // Obtener identidad del usuario
            $identity = $jwt_auth->checkToken($token, true);

            $em = $this->getDoctrine()->getManager();
            // Crear query
            $dql = "SELECT v FROM App\Entity\Follow v WHERE v.user != {$identi
ty->sub} ORDER BY v.id DESC";
            $query = $em->createQuery($dql);

            $pagination = $paginator->paginate($query, 1, 5);

            $data = [
                'status' => 'success',
                'code' => 200,
                'message' => 'Match traidos correctamente',
                'query' => $pagination
            ];
        }
            // Devolver datos en json
        return $this->resjson($data);
    }

    public function checkMyMatchs(Request $request, JwtAuth $jwt_auth, Paginat
orInterface $paginator)
```

```
    {
        $data = [
            'status' => 'error',
            'code' => 404,
            'message' => 'Error al buscar los match'
        ];

        // Recoger el token
        $token = $request->headers->get('Authorization', null);
        // Comprobar si es correcto
        $authCheck = $jwt_auth->checkToken($token);


        if ($authCheck) {
            // Obtener identidad del usuario
            $identity = $jwt_auth->checkToken($token, true);

            $em = $this->getDoctrine()->getManager();
            // Crear query
            $dql = "SELECT v FROM App\Entity\Follow v WHERE v.user = {$identit
y->sub} ORDER BY v.id DESC";
            $query = $em->createQuery($dql);

            $pagination = $paginator->paginate($query, 1, 5);


            $data = [
                'status' => 'success',
                'code' => 200,
                'message' => 'Match traídos correctamente',
                'query' => $pagination
            ];
        }
      // Devolver respuesta en json
        return $this->resjson($data);
    }

    public function remove(Request $request, JwtAuth $jwt_auth, $id = null)
    {
        $data = [
            'status' => 'error',
            'code' => 404,
            'message' => 'Follow no encontrado'
        ];
            // Recoger el token

        $token = $request->headers->get('Authorization');
```

```
            // Comprobar si es correcto

        $authCheck = $jwt_auth->checkToken($token);

        if ($authCheck) {
            // Obtener identidad del usuario
            $identity = $jwt_auth->checkToken($token, true);

            $doctrine = $this->getDoctrine();
            $em = $doctrine->getManager();
            // Obtener el follow
            $follow = $doctrine->getRepository(Follow::class)->findOneBy([
                'id' => $id
            ]);




            if ($follow && is_object($follow)) {
            // Eliminar de la bbdd
                $em->remove($follow);
                $em->flush();

                $data = [
                    'status' => 'success',
                    'code' => 200,
                    'message' => 'Follow borrado'
                ];
            }
        }
            // Devolver respuesta en json
        return $this->resjson($data);
    }

    public function sendMail(Request $request, JwtAuth $jwt_auth, $id = null,
MailerInterface $mailer)
    {

        $data = [
            'status' => 'error',
            'code' => 404,
            'message' => 'Follow no encontrado'
        ];
            // Recoger el token
        $token = $request->headers->get('Authorization');
            // Comprobar que es correcto
        $authCheck = $jwt_auth->checkToken($token);
```

```
        if ($authCheck) {
            // Obtener identidad del usuario
            $identity = $jwt_auth->checkToken($token, true);

            $doctrine = $this->getDoctrine();
            $em = $doctrine->getManager();
            // Obtener usuario
            $user = $doctrine->getRepository(User::class)->findOneBy([
                'id' => $id
            ]);


            $mail1 = $identity->email;

            $mail2 = $user->getEmail();
            // Crear el mail
            $email = (new Email())
                ->from('jjlltt75@gmail.com')
                ->to($mail2)
                ->subject('Match realizado en MBIP!')
                ->text('Sending emails is fun again!')

        >html('<p>El usuario</p>' . $mail1 . '<p>ha aceptado la solicitud por e
l proyecto Name</p>');


            try {
                  // Enviar mail
                $mailer->send($email);
            } catch (TransportExceptionInterface $e) {
                // En caso de error, que muestre el error
            }

            $data = [
                'status' => 'success',
                'code' => 200,
                'message' => 'Email enviado'
            ];
        }
      // Enviar respuesta en json
        return $this->resjson($data);
    }
```

Respecto a la última función sendMail, únicamente se ha usado para enviar correos a través de Gmail. Y para que estos se pudieran recibir, había que permitir dentro de la cuenta de Gmail, el uso de aplicaciones sospechosas.

4.4.1.4 Funciones relacionadas con Categoria

```
public function categories(Request $request, JwtAuth $jwt_auth,  PaginatorInte
rface $paginator)
    {
        $data = [
            'status' => 'error',
            'code' => 404,
            'message' => 'Categorías no encontradas'
        ];
        // Obtener token
        $token = $request->headers->get('Authorization');
        $authCheck = $jwt_auth->checkToken($token);

        if ($authCheck) {
            // Obtener el proyecto
            $categories = $this->getDoctrine()-
>getRepository(Category::class)->findAll();
            $pagination = $paginator->paginate($categories, 1, 10);


            // Comprobar si el proyecto existe
            if ($pagination && is_object($pagination)) {
                $data = [
                    'status' => 'success',
                    'code' => 200,
                    'categories' => $pagination
                ];

            }
        }

        // Devolver una respuesta en json

        return $this->resjson($data);
    }
```

4.4.2 Funciones del Frontend

Todas las funciones tienen una estructura muy parecida. Cada una de estas funciones se usan en distintos componentes.

4.4.2.1 Funciones relacionadas con Usuario

```
export var global = {
      // Url local donde se encuentra la API
    url: 'http://127.0.0.1:8080/Backend/public/'
}
```

```
this.url = global.url;
```

La variable url es una variable que hace referencia a la dirección de la API, para hacer las pruebas se usaba una dirección local.

```
 register(user: User):Observable<any>{
      // Convertir a json
        let json= JSON.stringify(user);
      // Crear variable params con el usuario en json
        let params = 'json='+json;
      // Crear cabeceras
        let headers = new HttpHeaders().set('Content-Type', 'application/x-
www-form-urlencoded');
      // Devolver por POST
        return this._http.post(this.url+'create', params, {headers: headers});
    }

    signup(user: User, gettoken: string):Observable<any>{
        if(gettoken == 'true'){
            user.gettoken = gettoken;
        }

        // Convertir a json
        let json= JSON.stringify(user);
      // Crear variable params con el usuario en json

        let params = 'json='+json;
      // Crear cabeceras
```

```
        let headers = new HttpHeaders().set('Content-Type', 'application/x-
www-form-urlencoded');
      // Devolver por POST
        return this._http.post(this.url+'login', params, {headers:headers});
    }

    update(user: User, token: string):Observable<any>{
      // Convertir a json
        let json= JSON.stringify(user);
      // Crear variable params con el usuario en json

        let params = 'json='+json;
      // Crear cabeceras
        let headers = new HttpHeaders().set('Content-Type', 'application/x-
www-form-urlencoded')
                                       .set('Authorization', token);
      // Devolver por PUT
        return this._http.put(this.url+'user/edit', params, {headers:headers})
;
    }

    getUser(token: string, id:number):Observable<any>{
            // Crear cabeceras
        let headers = new HttpHeaders().set('Content-Type', 'application/x-
www-form-urlencoded')
                                       .set('Authorization', token);
            // Devolver por GET
        return this._http.get(this.url+'user/detail/'+id, {headers:headers});
    }

    getIdentity(){
      // Obtener identity del LocalStorage
        let identity = localStorage.getItem('identity');

        if(identity && identity != 'undefined'){
            this.identity= JSON.parse(identity);
        }else{
            this.identity = null;
        }
```

```
      // Devolver identity
        return this.identity;
    }

    getToken(){
```

```
      // Obtener el token del LocalStorage
        let token = localStorage.getItem('token');

        if(token && token != 'undefined'){
            this.token= token;

        }else{
            this.token = null;
        }
      // Devolver el token
        return this.token;
    }
```

Como se ha comentado anteriormente, cada una de estas funciones se usa en distintos componentes con una estructura similar a la siguiente:

Este método se encuentra en el componente de register.component.ts

```
  onSubmit(form: NgForm){
    this._userService.register(this.user).subscribe(
       response => {
         console.log(response);
         if(response.status === 'error'){
          alert('El usuario ya existe');
         }
         else{
          this._router.navigate(['/login']);
         }
       },
       error => {
         console.log(error);
       }
    );
  }
}
```

Este tipo de componente se denomina Observable en Angular, se puede diferenciar fácilmente debido a su función suscribe.

Esta función que se observa (onSubmit) se ejecuta una vez se pulse el botón de Registrarse en la página de Registro. Dentro de la función, hacemos referencia a _userService, que es un Servicio dentro de Angular. Los servicios son formas de organizar métodos, por ejemplo, en el caso de Usuario, se han creado todos los métodos relacionados con Usuario en _userService.

Una vez se llama al método registre del _userService, podemos observar que se llama al método suscribe, el cual es el encargado de ejecutar alguna acción cuando haya algún cambio en el Observable. Se han planteado dos posibilidades, que el método retorne una respuesta, por lo que se llevara a cabo las acciones dentro de response, y que haya algún error, con lo que se llevara a cabo las acciones dentro de error.

4.4.2.2 Funciones relacionadas con Proyecto

```
 getProjects(token: string, page:number):Observable<any>{
      // Crear cabeceras
           let headers = new HttpHeaders().set('Content-
Type', 'application/x-www-form-urlencoded')
                                          .set('Authorization', token);
      // Devolver por GET
           return this._http.get(this.url+'project/list?page='+page, {headers
:headers});

        }

        getOtherProjects(token: string, page:number):Observable<any>{
      // Crear cabeceras
           let headers = new HttpHeaders().set('Content-
Type', 'application/x-www-form-urlencoded')
                                          .set('Authorization', token);
      // Devolver por GET
           return this._http.get(this.url+'project/list_other?page='+page, {h
eaders:headers});
        }

        getProject(token: string, id:number):Observable<any>{
           // Crear cabeceras
           let headers = new HttpHeaders().set('Content-
Type', 'application/x-www-form-urlencoded')
                                          .set('Authorization', token);
      // Devolver por GET
           return this._http.get(this.url+'project/detail/'+id, {headers:head
ers});
        }

        create(token: any, project: Project):Observable<any>{
      // Convertir a json
           let json= JSON.stringify(project);
      // Crear variable params con el usuario en json
           let params = 'json='+json;
      // Crear cabeceras
```

```
            let headers = new HttpHeaders().set('Content-
Type', 'application/x-www-form-urlencoded')
                                           .set('Authorization', token);
      // Devolver por POST
            return this._http.post(this.url+'project/new', params, {headers:he
aders});
        }

        update(project: Project, token: string, id:number):Observable<any>{
      // Convertir a json
            let json= JSON.stringify(project);
      // Crear variable params con el usuario en json
            let params = 'json='+json;




      // Crear cabeceras
            let headers = new HttpHeaders().set('Content-
Type', 'application/x-www-form-urlencoded')
                                           .set('Authorization', token);
      // Devolver por PUT
            return this._http.put(this.url+'project/edit/'+id, params, {header
s:headers});
        }

        delete(token: string, id:number):Observable<any>{
      // Crear cabeceras



         let headers = new HttpHeaders().set('Content-Type', 'application/x-
www-form-urlencoded')
                                           .set('Authorization', token);
      // Devolver por DELETE
            return this._http.delete(this.url+'project/remove/'+id, {headers:h
eaders});
        }

        onMatch(token: string, id: number){
      // Crear cabeceras
            let headers = new HttpHeaders().set('Authorization', token);
      // Devolver por GET
            return this._http.get(this.url+'follow/match/'+id, {headers:header
s});
        }
    }
```

Este método se encuentra en el componente project.component.ts

```
getProjects(page: number){
    this._projectService.getProjects(this.token, page).subscribe(
      response => {
        this.projects= response.projects;
        var number_pages= [];
        for(var i=1; i<=response.total_page; i++){
          number_pages.push(i);
        }
        this.number_page = number_pages;

        if(page >= 2){
          this.prev_page = page-1;
        }else{
          this.prev_page = 1;
        }



        if(page < response.total_page){
          this.next_page = page+1;
        }else{
          this.next_page = response.total_page;
        }
              },
      error => {
        console.log(error);

      }
    );
  }
```

Esta función que se observa (getProjects) se ejecuta una vez se accede a la página de myprojects. Dentro de la función, se hace referencia a _projectService, que es el Servicio encargado de almacenar todos los métodos relacionados con Proyecto.

La estructura es muy similar al método visto anteriormente. En caso de que haya una repuesta, se almacena en una variable projects todos los proyectos que se nos retornan de la petición. Además de esto, el siguiente trozo de código es el encargado de gestionar la paginación de la página, es decir saber qué proyectos tienen que ir en que página.

4.4.2.3 Funciones relacionadas con Follow

```
checkMatch(token: string):Observable<any>{
          // Crear cabeceras
        let headers = new HttpHeaders().set('Content-Type', 'application/x-
www-form-urlencoded')
                                       .set('Authorization', token);
      // Devolver por GET
        return this._http.get(this.url+'follow/check', {headers:headers});
    }

    checkMyMatchs(token: string):Observable<any>{
            // Crear cabeceras
        let headers = new HttpHeaders().set('Content-Type', 'application/x-
www-form-urlencoded')
                                       .set('Authorization', token);
      // Devolver por GET
        return this._http.get(this.url+'follow/mycheck', {headers:headers});
    }

    delete(token: string, id:number):Observable<any>{
    // Crear cabeceras
        let headers = new HttpHeaders().set('Content-Type', 'application/x-
www-form-urlencoded')
                                       .set('Authorization', token);

      // Devolver por DELETE
        return this._http.delete(this.url+'follow/remove/'+id, {headers:header
s});
    }

    sendMail(token: string, id:number):Observable<any>{
    // Crear cabeceras
        let headers = new HttpHeaders().set('Content-Type', 'application/x-
www-form-urlencoded')
                                       .set('Authorization', token);
      // Devolver por GET
        return this._http.get(this.url+'follow/mail/'+id, {headers:headers});
    }
```

Este método se encuentra en el componente other-projects.component.ts

```
checkMatch(){
    this._followService.checkMatch(this.token).subscribe(
      response => {
        console.log(response);
```

```
        this.follows= response.query;
      },
      error => {
        console.log(error);
      }

    )
  }
```

Esta función que se observa (checkMatch) se ejecuta una vez se accede a la página de other-projects. Dentro de la función, se hace referencia a _followService, que es el Servicio encargado de almacenar todos los métodos relacionados con Follow.

La estructura sigue siendo casi idéntica a los métodos que se han visto anteriormente. Si hay una respuesta, se almacena en la variable follows los follows que vienen con la respuesta, aunque en este caso se denominan query, ya que en el Backend hay que realizar una query para que te devuelvan estos valores.

4.4.2.4 Funciones relacionadas con Categoría

```
getCategories(token: string):Observable<any>{
            // Crear cabeceras
            let headers = new HttpHeaders().set('Content-
Type', 'application/x-www-form-urlencoded')
                                           .set('Authorization', token);

      // Devolver por GET
            return this._http.get(this.url+'category/all', {headers:headers});
          }
```

Este método se encuentra en el componente project-new.component.ts

```
getCategories(){
    this._categoryService.getCategories(this.token).subscribe(
      response => {
        this.categories= response.categories;
        console.log(this.categories);
      },
      error => {
        console.log(error);

      }
    );
```

Esta función que se observa (getCategories) se ejecuta una vez se accede a la página de project-new. Dentro de la función, se hace referencia a _categoryService, que es el Servicio encargado de almacenar todos los métodos relacionados con Categoria.

4.5 Error CORS

En este apartado se va a hablar de uno de los errores más frecuentes que surgieron al realizar este proyecto y como se solucionó.

El error CORS es un error que ocurre al intentar hacer peticiones de una url a otra distinta. Esto ocurre como medida de protección, siempre que no se indique lo contrario. La solución fue añadir las siguientes líneas de código en el Backend. Estas líneas añaden cabeceras a las distintas peticiones para que no haya ningún problema con el error CORS.

```
header('Access-Control-Allow-Origin: *');
header("Access-Control-Allow-Headers: X-API-KEY, Origin, X-Requested-
With, Content-Type, Accept, Access-Control-Request-Method, Authorization");
header("Access-Control-Allow-Methods: GET, POST, OPTIONS, PUT, DELETE");
header("Allow: GET, POST, OPTIONS, PUT, DELETE");
```

4.6 Diseño

En este apartado se va a ver cómo están hechas cada una de las pantallas de la aplicación

4.6.1 Página Inicial

La página inicial es una página sencilla donde hay dos enlaces, uno a la página de registro y el otro a la página de iniciar sesión.

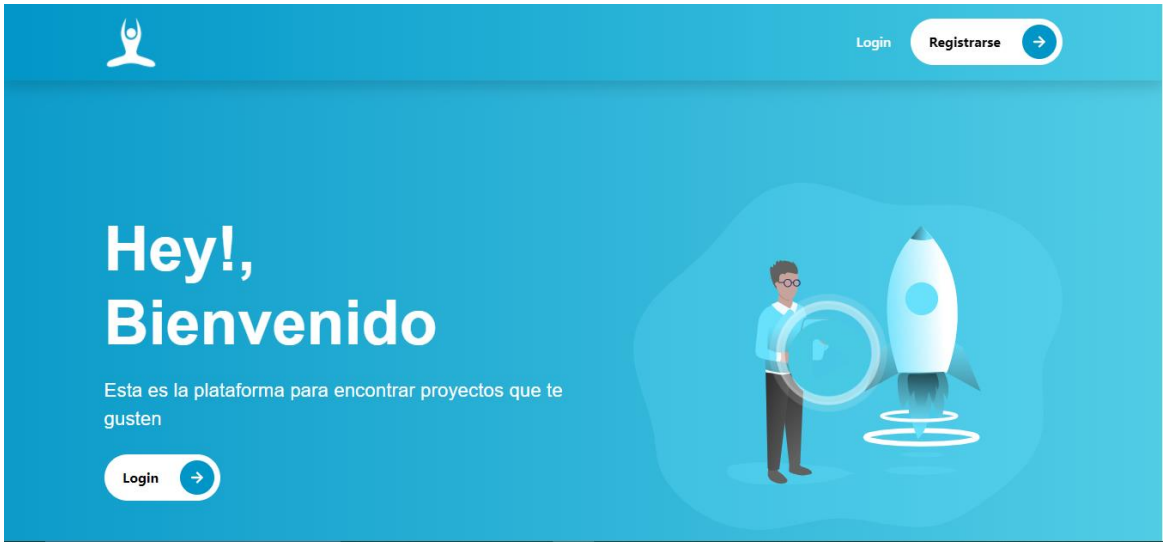

Figura 2. Página Inicial

4.6.2 Iniciar sesión

La página de iniciar sesión está compuesta por dos campos de tipo texto para introducir el correo electrónico y a contraseña y así identificarse.

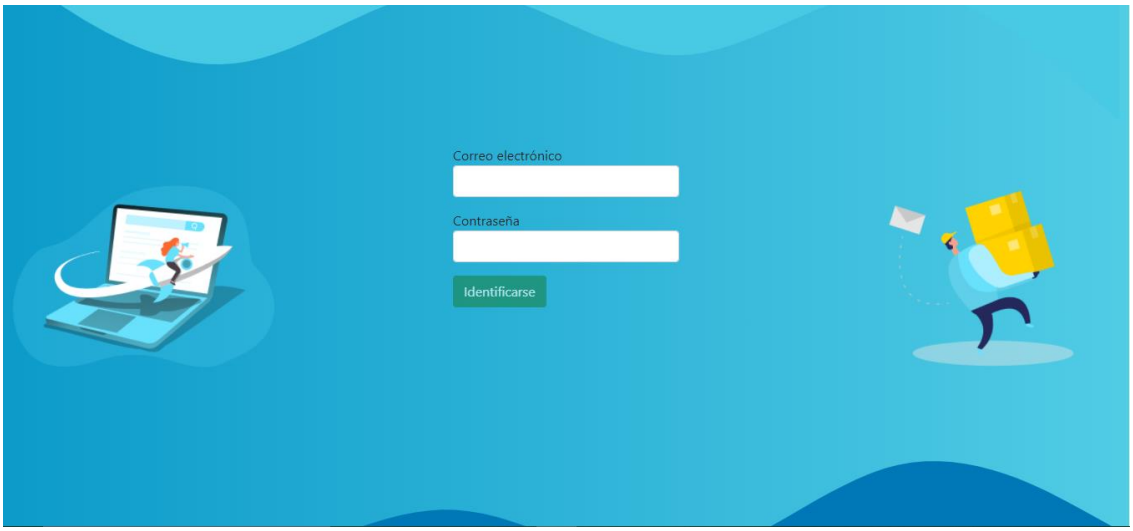

Figura 3. Página Iniciar Sesión

4.6.3 Registro

La página de Registro está compuesta por cuatro campos de tipo texto para introducir el nombre, el apellido, el correo electrónico y la contraseña.

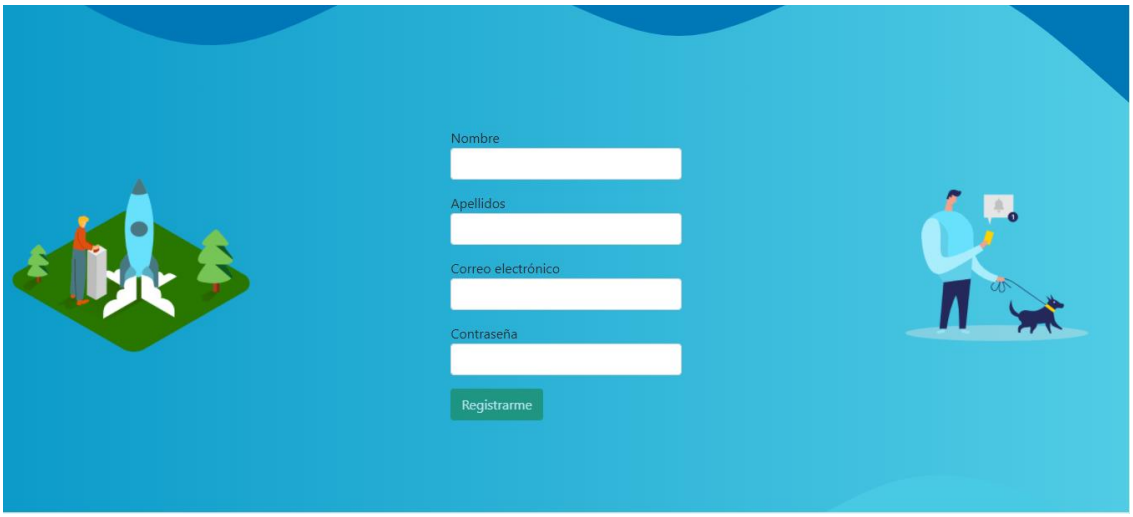

Figura 4. Página Registro

4.6.4 Otros proyectos

En esta página se podrá todos los proyectos creados por otros usuarios. Además, se tendrá la opción de filtrar por categoría y así aparecerán los proyectos que pertenezcan a dicha categoría. También hay una barra de navegación, que va a ser común en todas las siguientes páginas, que permite desplazarse entre las distintas páginas de la web.

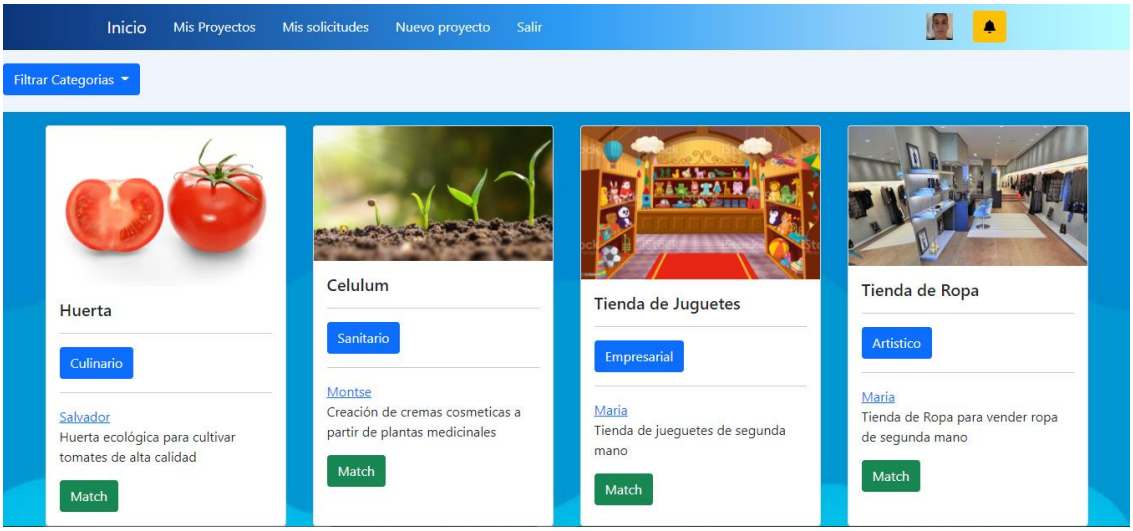

Figura 5. Página Otros Proyectos

4.6.5 Proyectos

En esta página se pueden ver los proyectos creados por el mismo usuario registrado, donde tendrá tres botones: Ver, que redirige a la página de detalles del proyecto, Editar que redirige a la página de editar proyecto, y Eliminar, que elimina el proyecto.

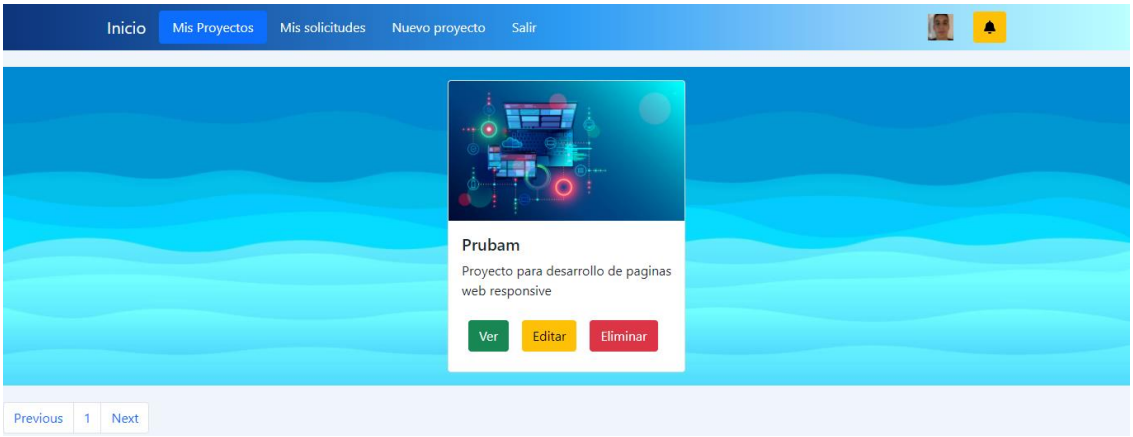

Figura 6. Página Proyectos

4.6.6 Ajustes del usuario

Esta es la página donde se puede cambiar los datos del usuario. Hay 4 campos de tipo texto, para el nombre, el apellido, el correo electrónico, y la imagen de perfil.

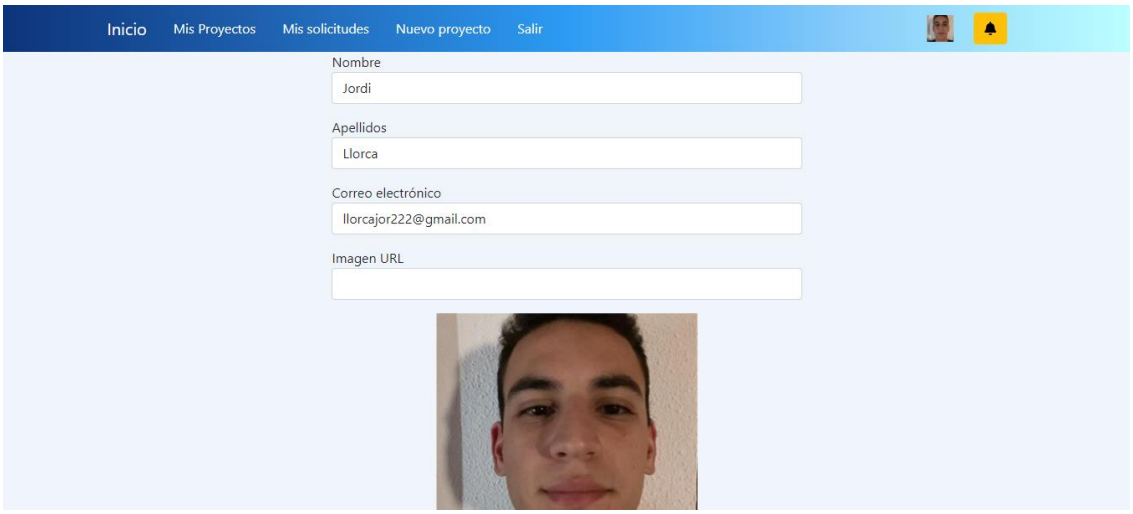

Figura 7. Página Ajustes del Usuario

4.6.7 Detalles del usuario

En esta página se podrá ver los datos de otros usuarios. Se puede ver el nombre, el apellido, la foto de perfil y el correo electrónico.

Figura 8. Página Detalles del Usuario

4.6.8 Detalles del proyecto

En esta página se podrá ver los detalles del proyecto. Se puede ver el nombre, la descripción, la foto y la categoría.

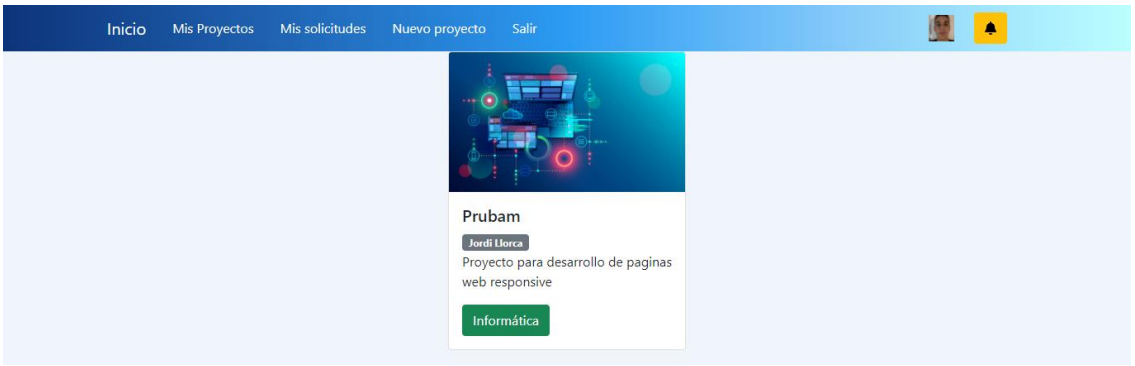

Figura 9. Página Detalles del proyecto

4.6.9 Editar proyecto

En esta página se podrá editar el proyecto. Habrá cuatro campos, uno para el nombre, otro para la descripción, otro para la imagen y otro para la categoría.

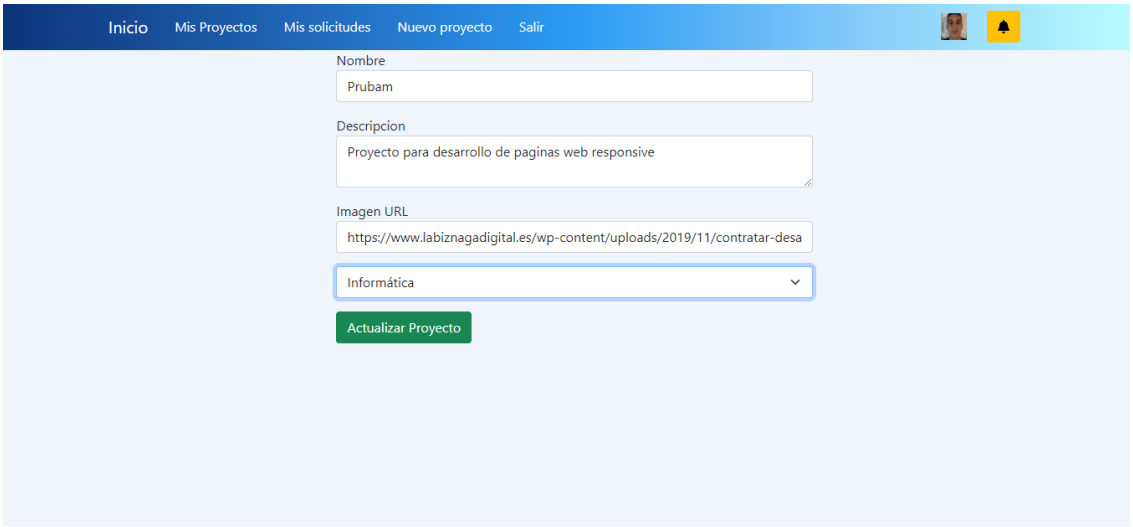

Figura 10. Página Editar Proyecto

4.6.10 Peticiones

En esta página se podrá ver las peticiones de unirse a un proyecto que se hayan realizado. Se podrá ver a que proyecto se ha realizado la petición, quien la ha realizado, y dos opciones, una de aceptar y otra de rechazar. La opción de aceptar hará que se mande un correo al usuario que ha realizado la petición avisándolo de que se ha aceptado su solicitud. La opción de rechazar borra la solicitud.

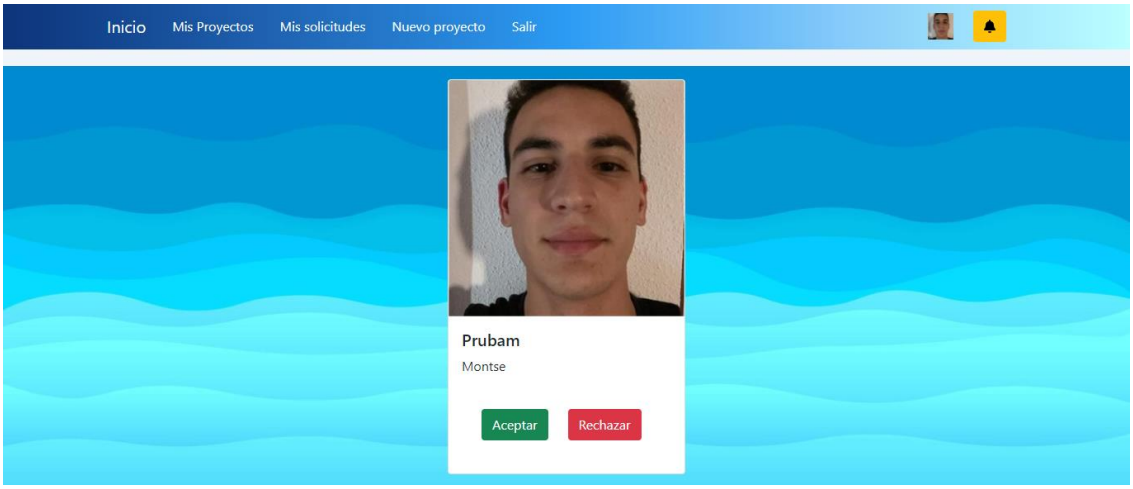

Figura 11. Página Peticiones

4.6.11 Mis peticiones

Esta página muestra las solicitudes que un usuario haya podido mandar. Se podrá ver el nombre del proyecto, el nombre del usuario que lo ha creado y la imagen del proyecto.

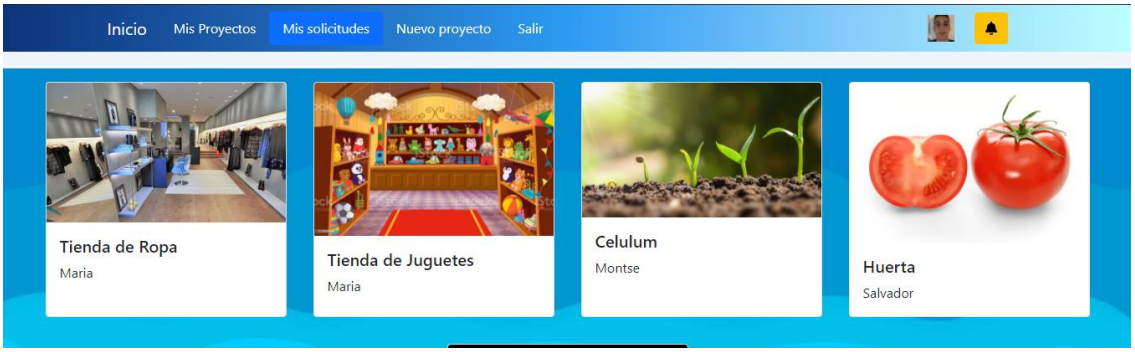

Figura 12. Página Mis Peticiones

4.6.12 Nuevo proyecto

Desde esta página se pueden crear nuevos proyectos. Tiene cuatro campos, el nombre, la descripción, la imagen y la categoría.

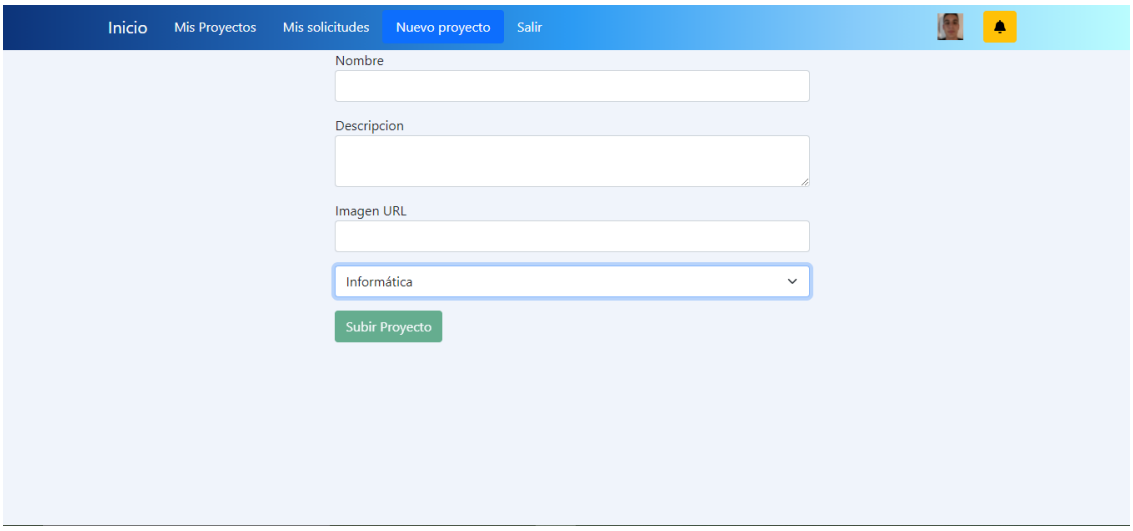

Figura 13. Página Nuevo Proyecto

4.7 Sistema de recomendación

En este apartado se va a hablar de la creación de un sistema de recomendación que permitirá sugerir proyectos afines al usuario en función de los proyectos a los que haya solucionado unirse anteriormente.

Cabe nombrar que este modelo no se ha implementado en la plataforma web, sin embargo, vamos a ver paso a paso como se debería de desarrollar dicho modelo.

4.7.1 Introducción

Un sistema de recomendación es un sistema que predice en función de la información histórica del usuario los productos, en el caso de este proyecto se trata de los distintos proyectos que podemos encontrar, con mayor posibilidad de gustar al usuario. Esta información histórica variará en función del sistema de recomendación, pero algunos de los datos más usados son: edad, proyectos que se hayan solicitado unirse previamente, valoraciones que los usuarios hayan dado, localización. La finalidad de este sistema no es más que ayudar a los usuarios a descubrir nuevos proyectos de interés para mejorar su experiencia en la plataforma.

El sistema de recomendación que se va a implementar usa procesamiento de lenguaje natural, que es el campo de la Inteligencia Artificial que estudia la forma de comunicar las maquinas con las personas mediante el uso de lenguajes humanos como el español o el inglés.

Hace unos años, hubo una revolución en el procesamiento de lenguaje natural al introducir algoritmos de aprendizaje automático, los cuales van a ser usados en este modelo.

En este sistema de recomendaciones se va a tener en cuenta cuatro factores: la categoría a la que pertenecen los proyectos que solicitemos unirse, el usuario que ha creado dichos proyectos, la descripción del proyecto y el nombre del proyecto.

Entrando más en detalle en los distintos tipos de sistemas de recomendaciones, se pueden distinguir dos clases de ellos: los filtros colaborativos y los filtros basados en contenido.

4.7.1.1 Filtros colaborativos

Los filtros colaborativos son aquellos que únicamente están basado en las interacciones pasadas entre usuarios y los proyectos para generar nuevas recomendaciones.

La siguiente figura muestra una matriz ficticia que podría representar las valoraciones que ciertos usuarios han dado a ciertos proyectos.

Figura 14. Matriz de filtros colaborativos

La idea es que mediante esta matriz de interacciones sea posible recomendar nuevos proyectos a los distintos usuarios.

La principal ventaja de este método es que no se requiere tener ninguna información acerca del usuario o del producto.

4.7.1.2 Filtros basados en contenido

Los filtros basados en contenido, a diferencia de los filtros colaborativos sí que requieren de información acerca del usuario y/o del proyecto.

La idea de este método es tratar de crear un modelo, basado en las características disponibles, que expliquen las interacciones observadas entre el usuario y los distintos proyectos. Es decir, utiliza las características del proyecto para predecir que otros proyectos con características similares o distintas pueden llegar a gustar a un usuario.

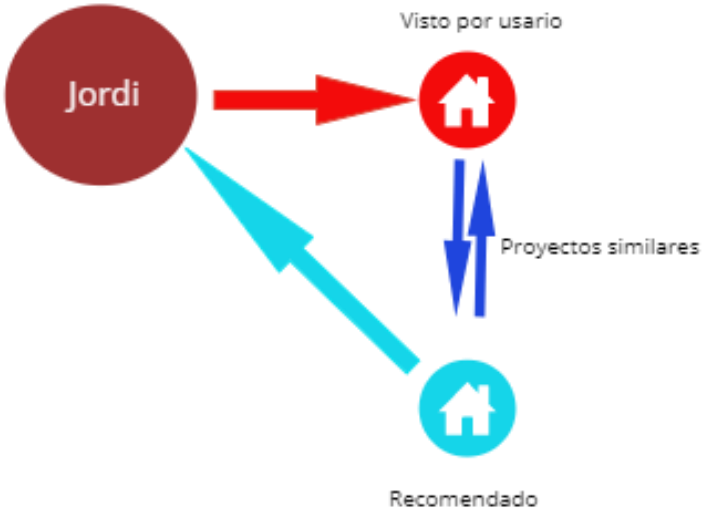

Figura 15. Filtro basado en contenido

Con este método se puede encontrar que personas que prefieran proyectos de distintas categorías pueden preferir proyectos de distinta índole, siendo la categoría una de estas características determinantes que permitirán al modelo recomendar proyectos más afines a cada una de los dos usuarios.

4.7.1.3 Hibrido

También existe una tercera opción, que es el filtro hibrido, este filtro es una combinación de los filtros nombrados anteriormente, el filtro colaborativo y el filtro basado en contenido.

El filtro hibrido suele ser el que mejor rendimiento da, sin embargo, hace falta más recursos para llevarlo a cabo.

4.7.1.4 Conclusiones

Para este proyecto se ha considerado que el filtro basado en contenido es la mejor opción, ya que no se cuenta con una matriz de interacción usuario-proyectos, con lo que el método colaborativo no sería posible de realizar.

4.7.2 Similitud

Antes de desarrollar el modelo, hay que hablar del concepto de similitud. La similitud entre dos proyectos, es la relación que tienen con sus características comunes. Mayor similitud, significa más características en común.

En los siguientes apartados se va a ver las distintas fórmulas usadas para calcular la similitud que existe entre distintos proyectos, ya que se recomendara un proyecto u otro dependiendo de este valor.

4.7.2.1 Similitud Coseno

La Similitud Coseno es una métrica usada para medir como de similar son los documentos independientemente de su tamaño. Matemáticamente mide el coseno del ángulo entre dos vectores proyectados en un espacio multidimensional. Es la métrica más usada para calcular la similitud entre distintos documentos.

$$sim(A, B) = \cos(\theta) = \frac{A * B}{|A| * |B|} \quad (6.1)$$

El valor del coseno varía entre -1 y 1, y los proyectos se organizan en orden descendente. Se utilizan uno de los dos siguientes enfoques para las recomendaciones:

Enfoque Top-n: Donde se recomiendan los n mejores proyectos

Enfoque de escala de calificación: Donde se establece un umbral y se recomiendan todas las películas por encima de ese umbral.

4.7.2.2 Otros enfoques

4.7.2.2.1 Distancia Euclidiana

Los elementos similares se encontrarán muy próximos entre sí si se trazan en un espacio de n dimensiones. Por lo tanto, podemos calcular la distancia entre elementos y, en función de esa distancia, recomendar elementos al usuario. La fórmula para la distancia euclidiana viene dada por:

La distancia euclidiana es la distancia entre dos puntos en un espacio euclídeo.

$$Distancia\ Euclidiana = \sqrt{(x_1 - y_1)^2 + \cdots + (x_n - y_n)^2} \tag{6.2}$$

4.7.2.2.2 Correlación de Pearson

Es una medida de dependencia lineal entre dos variables aleatorias cuantitativas.

Indica cuánto están correlacionados dos elementos. Cuanto mayor sea la correlación, mayor será la similitud. La correlación de Pearson se puede calcular mediante la siguiente fórmula:

$$sim(u, v) = \frac{\sum(r_{ui} - \bar{r}_u) * (r_{vi} - \bar{r}_v)}{\sqrt{\sum(r_{ui} - \bar{r}_u)^2 * (r_{vi} - \bar{r}_v)^2}} \tag{6.3}$$

4.7.2.3 Resumen

En resumen, los enfoques que se van a usar son: Filtros basados en contenidos, similitud del coseno, enfoque Top-n. Estas técnicas se utilizan dentro del campo del machine learning.

4.7.3 Manos a la obra

En este apartado se va a ver como se ha desarrollado el sistema de recomendaciones paso a paso.

4.7.3.1 Librerías

Primero se va a hablar de las librerías que se han usado para desarrollar el modelo.

4.7.3.1.1 Pandas

Pandas es una librería de Python especializada en el manejo y análisis de estructuras de datos.

```
import pandas as pd
```

4.7.3.1.2 Scikit-learn

Scikit-learn es una biblioteca para aprendizaje automático de software libre para el lenguaje de programación Python. Cuenta con algoritmos de clasificación, regresión, clustering y reducción de dimensionalidad.

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity
```

CountVectorizer se encarga de:

Eliminar palabras como 'el', 'la', 'los', etc. Ya que estas palabras no dan ninguna información útil.

Finalmente, construya la matriz TF-IDF sobre los datos, que es la encargada de medir la frecuencia de aparición de las distintas palabras.

Cosine_similarity se encarga de ejecutar los cálculos con la fórmula de la similitud del coseno.

4.7.3.1.3 NLTK

El kit de herramientas de lenguaje natural, o más comúnmente NLTK, es un conjunto de bibliotecas y programas para el procesamiento del lenguaje natural

```
import nltk
from nltk.corpus import stopwords
```

Stopwords son palabras como: de, la, que, etc… que no aportan ningún significado y por lo tanto pueden ser eliminadas sin alterar al sistema de recomendación.

4.7.3.2 Desarrollo

Primero de todo, importamos los datos:

```
datos = pd.read_csv('datos.csv', low_memory=False)
```

A continuación, definimos las características que queremos analizar:

```
caracteristicas = ['category', 'name', 'user']
```

El siguiente paso sería modificar las cadenas de texto que tenemos para pasarlas a minúscula y quitar el espacio de los nombres

```
def clean_data(x):
    if isinstance(x, list):
        return [str.lower(i.replace(" ", "")) for i in x]
    else:
        if isinstance(x, str):
            return str.lower(x.replace(" ", ""))
        else:
            return ''

for caracteristica in caracteristicas:
    datos[caracteristica] = datos[caracteristica].apply(clean_data)
```

Seguidamente crearíamos una nueva cadena de texto que contenga las cuatro características que vamos a analizar, 'category', 'name', 'description', 'user'

```
def create_soup(x):
    return x['category'] + ' ' + x['user'] + ' ' + x['name'] + ' ' + x['descri
ption']

datos['soup'] = datos.apply(create_soup, axis=1)
```

El siguiente paso es eliminar las 'stopwords' y calcular la similitud del coseno:

```
nltk.download('stopwords')
count = CountVectorizer(stop_words=stopwords.words('spanish'))
```

```
count_matrix = count.fit_transform(metadata['soup'])
coseno = cosine_similarity(count_matrix, count_matrix)
```

Después se tendría que cambiar el índice:

```
datos = datos.reset_index(drop=True)
indices = pd.Series(datos.index, index=datos['name'])
```

Para finalizar, se crea un método que permita obtener recomendaciones si se le pasa el nombre de un proyecto como parámetro y devuelve 5 proyectos similares:

```
def get_recommendations(name, cosseno=coseno):
    idx = indices[name]

    sim_scores = list(enumerate(coseno[idx]))

    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)

    sim_scores = sim_scores[1:6]

    movie_indices = [i[0] for i in sim_scores]

    return datos['name'].iloc[movie_indices]
```

4.8 Plan de negocio

Esta idea se ha planteado como proyecto final de carrera, pero en un futuro se buscaría poder seguir adelante con él y llegar a crear una empresa.

Se ha planteado una primera aproximación al plan de negocio de dicha empresa.

Para tener una pequeña idea del plan de negocio se va a proceder a diseñar un model business canvas analizando cada uno de sus puntos.

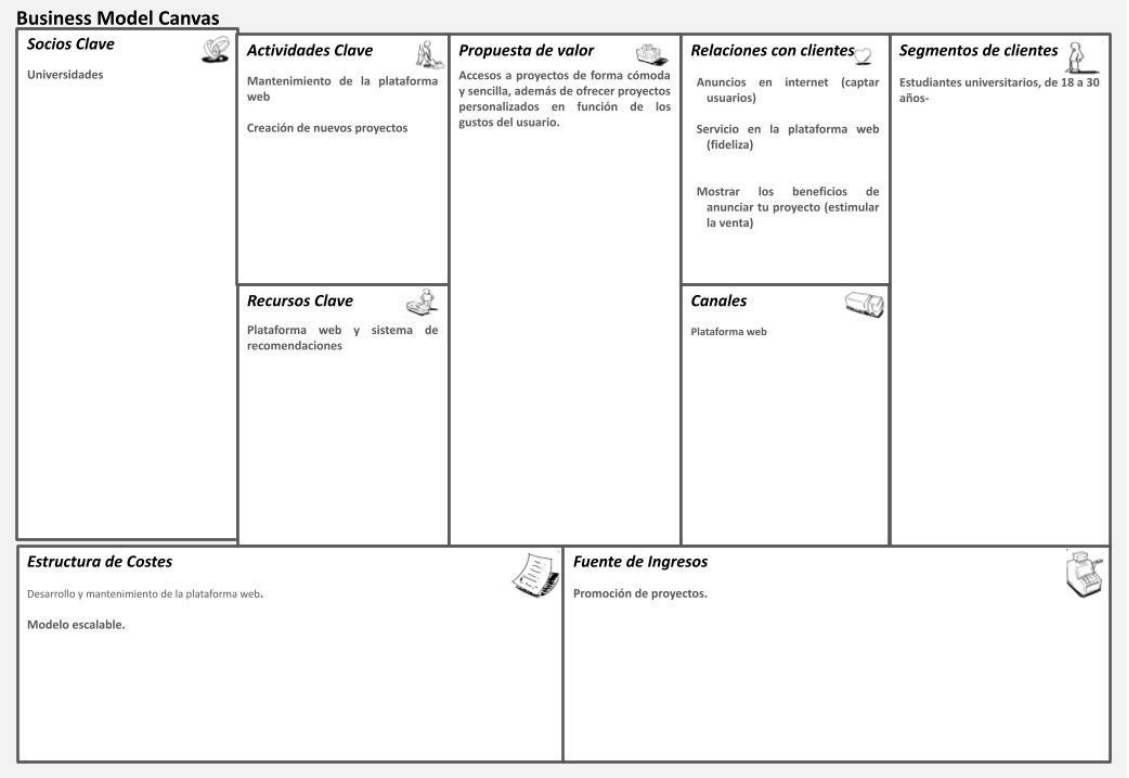

Figura 16. Business Model Canvas

4.8.1 Socios claves

Como Socios Claves se podrían destacar las distintas Universidades que promovieran esta plataforma a sus estudiantes. Además, todos los centros educativos como colegios públicos, privados, academias, etc.…

4.8.2 Actividades claves

Las actividades claves son: tener la web siempre operativa y el mantenimiento de creación de proyectos nuevos para siempre tener opciones recientes a los nuevos usuarios. Además, promover continuamente la plataforma web mediante un plan de marketing para que la cantidad de usuarios activos fuera creciendo y con lo cual aumentaría los proyectos nuevos creados.

4.8.3 Recursos clave

Como recursos claves se podrían destacar la web, el sistema de recomendación, el alojamiento web y los programadores que pudieran participar en el proyecto. A todo esto, se le podría sumar una aplicación móvil que podría ser usada como una red social.

4.8.4 Propuesta de valor

La propuesta de valor consiste en la facilidad y comodidad de encontrar ideas o proyectos afines al usuario y la facilidad de contactar con la persona al cargo del proyecto para poder crear un equipo o poder colaborar de alguna manera. Además, se facilitará enormemente el acceso a una comunidad donde se compartan valores de trabajo y una actitud de proactividad.

4.8.5 Relación con clientes

La relación con los clientes será de forma totalmente online y a través de la plataforma. Siempre se tendrá acceso a soporte en caso de necesitarlo y se podrá hacer sugerencias sobre algún aspecto de la plataforma de forma rápida y sencilla. Además, existirá una comunidad online donde se podrá interactuar con los distintos usuarios registrados.

4.8.6 Canales

El canal principal será la plataforma web, sin embargo, también se podrá acceder a la aplicación móvil.

4.8.7 Segmentos de clientes

Estudiantes de universidades de entre 18 a 30 años sin trabajo estable.

4.8.8 Estructura de costes

Puesta en marcha de la plataforma web y mantenimiento de la misma, así como su alojamiento web.

Desarrollo de todos los componentes de la plataforma web como de la aplicación móvil, el mantenimiento de las mismas y su alojamiento web.

4.8.9 Fuente de ingresos

La fuente de ingresos proviene de un sistema de publicidad donde se realzarán (aparecerán más) los proyectos que se deseen. El coste inicial será de mostrar durante 1 día un proyecto por 2€, durante una semana 5€ y durante un mes 15€. Además, se alojarían sitios publicitarios dentro de la plataforma web, permitiendo a empresas externas publicitar sus productos. Se podría destacar un sinfín de empresas interesadas en anunciar sus productos, ya que su público sería estudiantes. Se podría destacar como empresas interesadas: Portales de anuncios de empleos, empresas de software como Adobe, empresas dedicadas a ofrecer cursos, etc…

5 Conclusiones finales

5.1 Conclusiones

Respecto a las conclusiones finales podemos sacar las siguientes:

Si hubiera que volver a desarrollar la plataforma desde 0, optaría por desarrollarlo con las siguientes tecnologías: React para el Frontend, ya que considero que es una librería de Javascript más sencilla de usar que Angular y por tanto en proceso de aprendizaje y adaptación hubiera sido más corto. Además, al ser más popular que Angular podríamos encontrar más recursos informativos en internet que ayudarían enormemente al aprendizaje de dicha herramienta.

Figura 17. React vs Angular

Respecto al Backend, usaría Django, en vez de Symfony, ya que el framework de Python es más conocido y por tanto se pueden encontrar más recursos informativos en la web.

Figura 18. Django vs Symfony

Además, al tratarse de un framework de Python ayudaría enormemente a la incorporación del sistema de recomendación, ya que este está escrito en Python, lenguaje más popular a la hora de desarrollar Inteligencia Artificial.

La planificación del proyecto habría sido distinta. Habría desarrollado el diseño de la página web en un primer momento.

A nivel personal, me propuse realizar este proyecto con la finalidad de aprender nuevas tecnologías para desarrollo web. Durante todo este trayecto he aprendido mucho y descubierto nuevas cosas.

Aún queda mucho por hacer para poder crear una muy buena plataforma donde distintas personas puedan colaborar de forma conjunta para llevar a cabo sus ideas, sin embargo, este ha sido el primer paso en esta dirección.

Espero seguir creciendo y mejorando. Algún día, tal vez, esta plataforma de la que hablo dejará de ser una idea y se convertirá en un hecho. Hasta ese día, aún hay mucho trabajo que hacer.

5.2 Principales aportaciones

Se han alcanzado los objetivos que se propusieron en un primer momento. Se ha conseguido la creación de la plataforma web que permite la creación de distintos proyectos y la inscripción en proyectos de otros usuarios. También se ha conseguido desarrollar el modelo de inteligencia Artificial

con éxito, sin embargo, no se ha llegado a implementar dentro de la plataforma web. Además, se ha planteado una primera aproximación al plan de negocio que se pretende aplicar en un futuro.

5.3 Desarrollos futuros

Una de las mejoras a desarrollar, consistiría en la implementación de un sistema de recomendaciones dentro de la plataforma web, debido a que de esta manera se mejoraría la experiencia del usuario al encontrar más proyectos afines a sus gustos. Además, se buscaría mejorar este sistema añadiendo más parámetros como la ubicación y el idioma del usuario.

Otra de las mejoras, sería la implantación de apartados donde se pudieran publicitar los proyectos seleccionados y así generar ingresos.

La siguiente mejora a implementar, consistiría en la creación de un sistema de competiciones, donde cualquier usuario pudiera publicar un reto y los demás usuarios de la plataforma pudieran resolverlo, ya que de esta manera se incentivaría en gran medida la participación de los usuarios en distintos proyectos, ya que estos retos podrían dar cierto estatus a los usuarios que lograran completarlos.

Otra de las mejoras, consistiría en la mejora del sistema de mensajería, ya que actualmente, el sistema de envió de correos al aceptar una solicitud tiene bastantes defectos, como por ejemplo que al registrarse con un correo en la plataforma web, este no existiera y por lo tanto el correo jamás llegaría. Como propuesta a esto, se podría crear un sistema de mensajería dentro de la propia plataforma.

Alguna otra de las posibles mejoras, consistiría en añadir un sistema de rango a los usuarios, permitiendo así que los que hubieran participado en más proyectos tuvieran algunas insignias que los distinguieran. Con esto se perseguiría crear un sistema donde los distintos usuarios tuvieran una razón por la cual unirse a más proyectos y esforzarse.

La última de las mejoras, consistiría en la creación de una tabla en la base de datos para almacenar la información referente a si un usuario ha aceptado o rechazado la solicitud de formar parte de un proyecto, de esta manera se podría tener constancia de estas acciones, para posteriormente recomendar unos proyectos u otros.

Referencias

Eremenko, K. (s.f.). *https://www.udemy.com/*. Obtenido de https://www.udemy.com/course/machinelearning/learn/lecture/19229276?start=15#overview

Falk, K. (s.f.). Practical Recommender Systems.

González, A. (s.f.). *https://cleverdata.io/*. Obtenido de https://cleverdata.io/sistemas-recomendacion-machine-learning/

https://es.wikipedia.org. (s.f.). Obtenido de https://es.wikipedia.org/wiki/Distancia_euclidiana

https://es.wikipedia.org/. (s.f.). Obtenido de https://es.wikipedia.org/wiki/Similitud_coseno

https://es.wikipedia.org/. (s.f.). Obtenido de https://es.wikipedia.org/wiki/Coeficiente_de_correlaci%C3%B3n_de_Pearson

https://getbootstrap.com/. (s.f.). Obtenido de https://getbootstrap.com/docs/5.0/components/

https://symfony.com/. (s.f.). Obtenido de https://symfony.com/doc/current/index.html

https://www.emprendedores.es/. (s.f.). Obtenido de https://www.emprendedores.es/gestion/modelo-3/

https://www.grapheverywhere.com/. (s.f.). Obtenido de https://www.grapheverywhere.com/algoritmo-de-similitud-de-coseno/

Lucas, J. (s.f.). *https://openwebinars.net/*. Obtenido de https://openwebinars.net/blog/que-es-c/

Moreno, A. (s.f.). *https://www.iic.uam.es*. Obtenido de https://www.iic.uam.es/inteligencia/que-es-procesamiento-del-lenguaje-natural/

Muriung, P. (s.f.). *https://www.youtube.com/*. Obtenido de https://www.youtube.com/watch?v=iJKCj8uAHz8

Robles, V. (s.f.). *https://www.udemy.com/*. Obtenido de https://www.udemy.com/course/master-en-desarrollo-web-full-stack-angular-node-laravel-symfony/

Robles, V. (s.f.). *https://www.udemy.com/*. Obtenido de https://www.udemy.com/course/aprende-a-publicar-sitios-web-en-internet-hosting-vps/

Robles, V. (s.f.). *https://www.udemy.com/*. Obtenido de https://www.udemy.com/course/master-en-php-sql-poo-mvc-laravel-symfony-4-wordpress/

Schwarzmüller, M. (s.f.). *https://www.udemy.com/*. Obtenido de https://www.udemy.com/course/the-complete-guide-to-angular-2/

Sharma, A. (s.f.). *https://www.datacamp.com/*. Obtenido de https://www.datacamp.com/community/tutorials/recommender-systems-python

Visus, A. (s.f.). *https://www.esic.edu/*. Obtenido de https://www.esic.edu/rethink/tecnologia/para-que-sirve-python

DOCUMENTO II: PRESUPUESTOS

CUADRO Nº1: PRECIOS DE LA MANO DE OBRA.

Código	Descripción de la mano de obra	Precio (€/h)
MO1	Graduado en Ingeniería en Tecnologías Industriales	20, 00

CUADRO Nº2: PRECIOS DE LOS MATERIALES PUESTOS EN OBRA.

Código	Descripción del material	Precio (€)	Periodo de amortización (años)	Precio (€/h)
MA1	Ordenador	500	6	0,01
MA2	Servidor	576	6	0,01

CUADRO Nº3: PRECIOS DESCOMPUESTOS.

Nº de orden	Descripción de las unidades	Unidad		Precio(e)	Importe(e)
UO1	**Desarrollo plataforma web**	**h**			**25,77**
MO1	Graduado en Ingeniería de Tecnologías Industriales	h	1	20	20,00
MA1	Ordenador	h	1	0,01	0,01
MA2	Servidor	h	1	0,01	0,01
	Costes directos complementarios		0,03	20,02	0,60
	Costes directos				20,62
	Costes indirectos		0,25	20,62	5,15

CUADRO Nº3: PRECIOS DESCOMPUESTOS.

Nº de orden	Descripción de las unidades	Unidad		Precio(e)	Importe(e)
UO2	**Desarrollo modelo Inteligencia Artificial**	**h**			**25,76**
MO1	Graduado en Ingeniería de Tecnologias Industriales	h	1	20	20,00
MA1	Ordenador	h	1	0,01	0,01
	Costes directos complementarios		0,03	20,01	0,60
	Costes directos				20,61
	Costes indirectos		0,25	20,61	5,15

CUADRO Nº3: PRECIOS DESCOMPUESTOS.

Nº de orden	Descripción de las unidades	Unidad		Precio(e)	Importe(e)
UO3	**Redacción de la documentación**	**h**			**25,76**
MO1	Graduado en Ingeniería de Tecnologías Industriales	h	1	20	20,00
MA1	Ordenador	h	1	0,01	0,01
	Costes directos complementarios		0,03	20,01	0,60
	Costes directos				20,61
	Costes indirectos		0,25	20,61	5,15

CUADRO Nº4: PRECIOS UNITARIOS.

Nº de orden	**Descripción de la unidad de obra**	**Unidad**	**Precio (€)**
UO1	Desarrollo plataforma web	h	25,77
UO2	Desarrollo modelo Inteligencia Artificial	h	25,76
UO3	Redacción de la documentación	h	25,76

CUADRO Nº5: ESTADO DE MEDICIONES

Nº de orden	**Descripción de la unidad de obra**	**Unidad**	**Medición**
UO1	Desarrollo plataforma web	h	100,00
UO2	Desarrollo modelo Inteligencia Artificial	h	20,00
UO3	Redacción de la documentación	h	50,00

CUADRO Nº5: ESTADO DE MEDICIONES

Nº de orden	Descripción de la unidad de obra	Unidad	Precio (€)	Medición	Importe (€)
UO1	Desarrollo Plataforma web	h	25,77	100,00	2577,00
UO2	Desarrollo modelo Inteligencia Aritficial	h	25,76	20,00	515,20
UO3	Redacción de documentación	h	25,76	50,00	1288,00
	Total Presupuesto de Ejecución Material (PEM)				4380,20
	Gastos Generales (12 % del PEM)				525,62
	Beneficio Industrial (6 % del PEM)				262,81
	Total Presupuesto de Ejecución por Contrata (PEC)				5168,63
	IVA (21 % del PEC)				1085,41
	PRESUPUESTO BASE DE LICITACIÓN				6254,04

El coste total del proyecto asciende a **SEIS MIL DOSCIENTOS CINCUENTA Y CUATRO EUROS Y CUATRO CÉNTIMOS**