Universitat Oberta de Catalunya

uoc.edu

Banco del tiempo

Néstor Ibáñez Polo
Grado de Ingeniería Informática
Desarrollo Web

Gregorio Robles Martínez
Santi Caballe Llobet

26 de junio de 2023

Agradecimientos

A mi padre y a su mujer por apoyarme y animarme en los momentos duros. Pero en especial a Laura García, ya que ella es la persona que más paciencia, apoyo, ánimos, y sobre todo tiempo me ha dado. Esto no podría haber sido posible con ninguna otra persona a mi lado. Gracias.

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Banco del tiempo*
Nombre del autor:	*Néstor Ibáñez Polo*
Nombre del consultor/a:	*Gregorio Robles Martínez*
Nombre del PRA:	*Santi Caballe Llobet*
Fecha de entrega (mm/aaaa):	06/2023
Titulación:	*Grado de ingeniería Informática*
Área del Trabajo Final:	*Desarrollo web*
Idioma del trabajo:	*Castellano*
Palabras clave	*Compartir, aprender, enseñar, banco del tiempo.*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados y conclusiones del trabajo.*

La finalidad del trabajo es la creación de un banco del tiempo, como parte final de un trabajo de fin de grado, es decir, una plataforma web donde las personas puedan compartir conocimiento, sin ningún ánimo de lucro. Se ha creado una bolsa de horas, las cuales se obtienen (10 horas) al registrarse, y la única manera posible de obtener más horas es compartir tus conocimientos con más usuarios, los cuales podrán, o bien “pagarte” con las horas que tengan en su bolsa, o bien compartiendo sus conocimientos contigo.

El proyecto ha sido desarrollado principalmente en HTML, JavaScript, PHP y MySQL. Se le ha dado forma con el framework Bootstrap5.0. Se han separado cada una de las vistas en un archivo independiente, relacionado con un controlador el cual proporcionara los datos obtenidos de la base de datos, para que dicha vista muestre correctamente su contenido.

Abstract (in English, 250 words or less):

The purpose of the work is the creation of a time bank, as a final part of a final degree project, that is, a web platform where people can share knowledge, without any profit motive. A bag of hours has been created, which are obtained (10 hours) by registering, and the only possible way to obtain more hours is to share your knowledge with other users, who can either "pay you" with the hours they have in their bag, or by sharing their knowledge with you.

The project has been developed mainly in HTML, JavaScript, PHP and MySQL. It has been shaped with the Bootstrap5.0 framework. Each of the views has been separated in an independent file, related to a controller which will provide the data obtained from the database, so that the view displays its content correctly.

Índice

Lista de figuras

1. Introducción

1.1 Contexto y justificación del Trabajo

Hoy más que nunca, el aprendizaje e-learning, las redes sociales, las comunicaciones vía videoconferencia, están en auge. Esto me lleva al desarrollo de mi proyecto enfocado en el aprendizaje online. Su funcionamiento es sencillo: lo primero que hay que hacer es darse de alta a través del formulario de la propia web. En este momento el usuario registrado recibe una bolsa de 10 horas. Estas horas se convertirán en la moneda de cambio. Es decir, Pepe enseñara 1 hora de guitarra a Manolo, Manolo puede pagarle con una clase de 1 hora de inglés o bien darle una de sus horas. De este modo se obliga a, o bien aprendes a cambio de enseñar, o no podrás aprender más si no te quedan horas para pagar, ya que la única manera de adquirir tiempo será compartiendo tus conocimientos con otros.

1.2 Objetivos del Trabajo

1.2.1 Objetivos principales

El objetivo del proyecto es poner en contacto a las personas, con la intención de generar un intercambio de conocimientos y habilidades entre ellos sin ningún tipo de coste económico, simplemente una inversión de horas.

1.2.2 Subobjetivos

El proyecto irá destinado a cualquier persona interesada en enseña y/o aprender una serie de habilidades o intereses propios. Por otro lado, está la parte de las ofertas/demandas. Un usuario (Pepe) puede crear tantas ofertas como quiera, indicando la categoría (música, informática, lenguas, etc.) y otro usuario (Manolo) interesado en sus ofertas, contactara con él vía mail. Si se ponen de acuerdo y finalmente Manolo está satisfecho con las clases obtenidas, deberá o bien dar una clase de algo a Pepe, o bien pagar 1 hora de su bolsa de horas. Además, Manolo podrá valorar la clase recibida.

1.3 Enfoque y método seguido

Primero de todo, y puesto que tras un proyecto siempre hay un equipo, se ha de aclarar que el desarrollo de este proyecto ha sido realizado exclusivamente por mí. Yo he hecho todo, la planificación, el frontend, el backend, los manuales e instrucciones, etc.

Además de los conocimientos previos, ya sean obtenidos en formaciones anteriores o la propia experiencia laboral, he intentado, y creo que, conseguido, seguir las metodologías aprendidas en otra asignaturas, como “*Proyecto de desarrollo del software*”.

1.4 Planificación del Trabajo

1.4.1 Hitos principales

Hito	Fecha
Pec1 Planificación	Miércoles 01/03/2023
Pec2: Alcance y objetivos	Lunes 13/03/2023
Pec3: Implementación	Lunes 10/04/2023
Pec4: Memoria	Sábado 03/06/2023
Defensa	Lunes 03/07/2320

1.4.2 Planificación

Actividad	Fecha Inicio	Duracion en días	Fecha Fin
PEC1	**01/03/2023**	**10**	**11/03/2023**
Idea principal	01/03/2023	2	02/03/2023
Tegnologías a usar	03/03/2023	2	04/03/2023
Objetivos	05/03/2023	3	07/03/2023
Planificación	08/03/2023	5	12/03/2023
PEC2 Alcance y objetivos	**13/03/2023**	**22**	**03/04/2023**
Modelo de casos de uso	13/03/2023	4	16/03/2023
Historias de usuario	17/03/2023	2	18/03/2023
Modelo de pantallas	19/03/2023	3	21/03/2023
Arquitectura de la base de datos	22/03/2023	4	25/03/2023
Diseño relacional de la base de datos	26/03/2023	5	30/03/2023
Diagrama de clases principales	31/03/2023	2	01/04/2023
Diagrama de arquitectura	02/04/2023	2	03/04/2023
PEC3 Implementación	**10/04/2023**	**55**	**03/06/2023**
Creacion de la base de datos (mysql)	10/04/2023	3	12/04/2023
Conexión a la base de datos	13/04/2023	1	13/04/2023
Funcionalidades de usuario	14/04/2023	17	30/04/2023
Funcionalidades de administrador	01/05/2023	25	25/05/2023
Despliegue	26/05/2023	1	26/05/2023
Finalización de la documentación	27/05/2023	4	30/05/2023
PEC4 Memoria y presentación	**03/06/2023**	**24**	**26/06/2023**
Documentacion de memoria	03/06/2023	16	18/06/2023
Video de presentación	19/06/2023	6	24/06/2023
Informe de Autoevaluación	25/06/2023	2	26/06/2023
Defensa	03/07/2023	5	07/07/2023

Ilustración 1. Planificación del trabajo

1.4.3 Diagrama de Gantt

El correspondiente diagrama de Gantt [1] referente a la planificación.

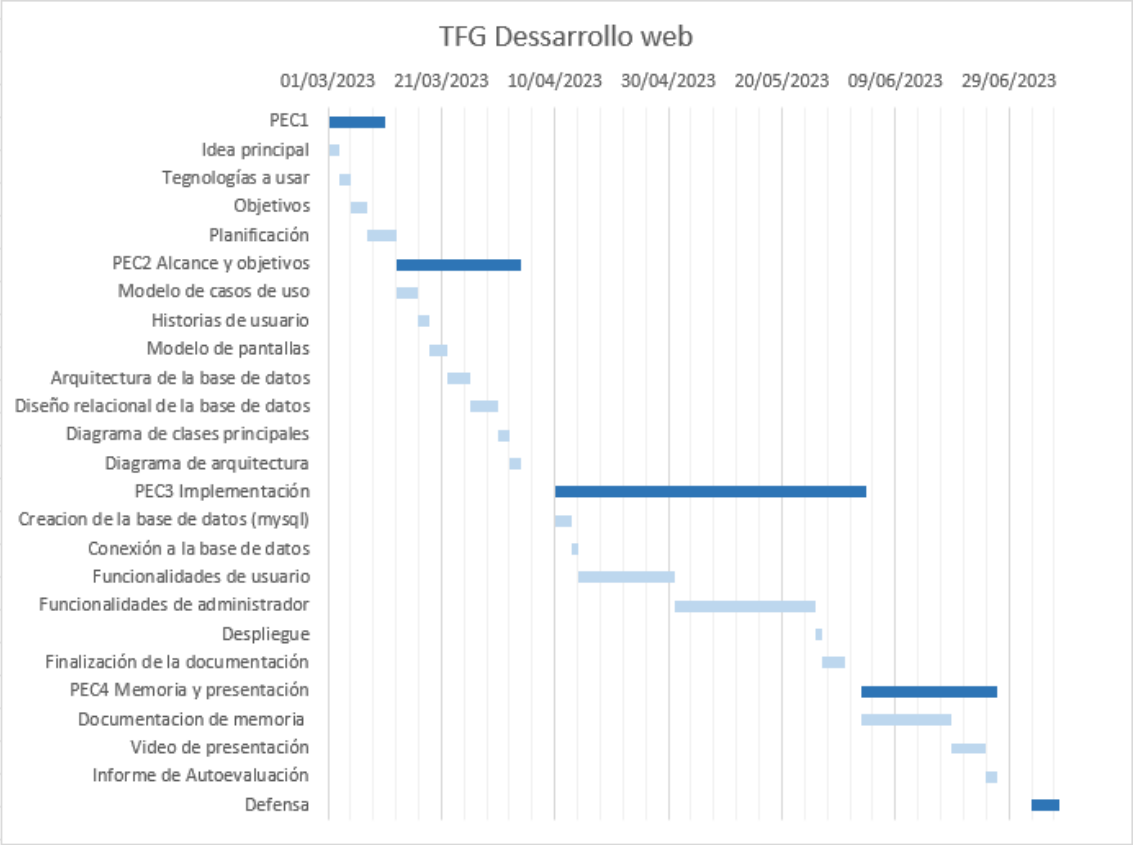

Ilustración 2. Diagrama de Gantt

1.4.4 Análisis de riesgos

El mayor riesgo es el cálculo de tiempo que se va a necesitar para la realización de cada apartado, ya que esto podría retrasarnos en el desarrollo, con lo que podría dar lugar a un producto de menoro calidad, o lo que es peor, no entregar a tiempo alguno de los hitos principales, cosa que es de obligado cumplimiento.

Otras causas que nos llevaran a un posible retraso serian:

- Problemas de salud.
- Problemas técnicos con el hardware necesario.
- Problemas de investigación por falta de conocimientos de los recursos lógicos seleccionados.
- Un exceso de trabajo que nos obligue a hacer horas extra, quitándonos así tiempo para realizar el proyecto.

1.5 Breve sumario de productos obtenidos

En el desarrollo web de este proyecto, para la creación de una plataforma donde compartir conocimientos sin ningún ánimo de lucro, se han obtenido los siguientes productos:

· Todo el código fuente para la creación de dicha plataforma, tanto los scripts para la creación de la base de datos, como el propio código que da forma y lógica a la web. Por lo tanto, se obtiene una versión funcional del producto.
· Documento científico-técnico de la planificación, desarrollo y finalización del trabajo.
· Anexos de la documentación de la memoria.
· Vídeo explicativo.
· Informe de autoevaluación
· Presentación del producto.

1.6 Breve descripción de los otros capítulos de la memoria

Capítulo 2: Análisis.

· Requisitos funcionales.
· Requisitos no funcionales.
· Modelo de casos de uso.
· Casos de uso.

Capítulo 3: Diseño.

· Diagrama de arquitectura.
· Diagrama de clases.
· Diseño relacional de la base de datos.
· Diseño lógico de la base de datos.
· Listado de ficheros y procedimientos.
· Pantallas.
· Otros ficheros y carpetas.

2. Análisis

A continuación, se presentan las consideraciones para tener en cuenta en forma de requisitos mínimos que queremos obtener, una serie de funcionalidades en forma de diagrama de caso de uso (ilustración 3) y los propios casos de uso.

2.1 Requisitos funcionales

Estas son las actividades mínimas que el sistema deberá realizar.

Requisitos funcionales	
Requerimiento	**Descripción del requerimiento**
RF1	Constará de un sistema de login, el cual permitirá o denegara el acceso a los usuarios en función de si estos están registrados o no.
RF2	Constará de un sistema de encriptación de contraseñas, con lo que las contraseñas solo “viajaran” a través de la red de manera encriptada.
RF3	El usuario ha de poder recuperar su contraseña.
RF4	El usuario ha de poder realizar búsquedas, ya sea por interés o cualquier texto.
RF5	Intercambio de tiempo (unidad de medida de pago), se realiza una vez la persona que aprende finaliza “la clase”.
RF6	Cada oferta debe tener un identificador, necesario para la manipulación interna de datos.
RF7	Configuración de seguridad mediante un sistema de certificado[3] de cifrado, para que la navegación por la plataforma sea segura.
RF8	Necesitará una base de datos para almacenar la información.
RF9	La plataforma constará de un método de contacto usuario-administración (email)
RF10	Los campos de datos deben ser coherentes con lo que se espera recibir en ellos, es decir, por ejemplo: un campo que espera una dirección de correo electrónico ha de impedir el avance de la aplicación si el formato no es correcto.

2.3 Requisitos no funcionales

Es importante no olvidarnos de los requisitos no funcionales, ya que son los que van a detallar la arquitectura del sistema.

Requisitos NO funcionales	
Requerimiento	**Descripción del requerimiento**
RNF1	La aplicación necesitara un mínimo de X Megas para su instalación. Esto dependerá del sistema operativo utilizado para el servidor y la versión de librerías instaladas (la aplicación en sí no ocupa más de 10 megas).
RNF2	La aplicación debe de proporcionar tiempos de respuesta rápidos.
RNF3	La aplicación debe mantener los datos almacenados seguros y protegidos.
RNF4	La aplicación debe ser fácil de descargar e instalar e intuitiva.
RNF5	La base de datos ha de ser relacional.
RNF6	La manipulación de datos se realizará mediante un CRUD.
RNF7	El sistema debe gestionar los posibles errores.
RNF8	Si una interacción con la base de datos da algún error o excepción, dicho error ha de quedar reflejado en un control del log.
RNF9	El sistema debe ser intuitivo y amigable para el usuario.

2.3 Modelo de casos de uso

Como veremos continuación en la ilustración 3, un administrador puede hacer lo mismo que un usuario. El usuario, aunque tiene las mismas historias que un administrador, se verá limitado dado su rol. Por ejemplo, un usuario no podrá modificar su rol, sin embargo, un administrador podrá modificar el rol de cualquier usuario. Esto sucede en más ocasiones y se ve mejor en el apartado "Modelo de pantallas" donde se diferencian los campos que puede modificar un usuario y los que únicamente puede un administrador.

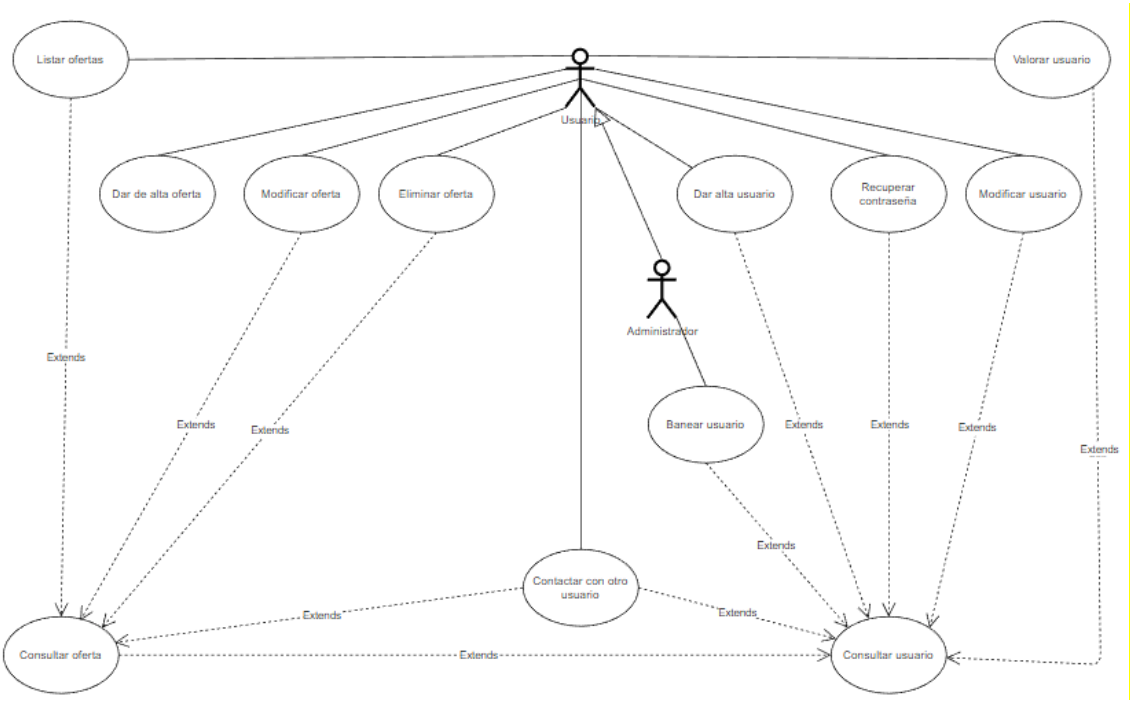

Ilustración 3. Casos de uso.

2.4 Casos de uso

Detalle de los casos de uso acorde a la ilustración 3 expuesta en el apartado 2.3 Modelo de casos de uso.

Caso de uso	Consultar usuario *Ilustración 17. Pantalla de información personal y modificar usuario.*	
Actor	Usuario y administrador	
Propósito	Comprobar si el usuario existe en la BD	
Resumen	USUARIO:	SISTEMA: 1.El caso de uso recibe un username en forma de String 2.Buscar username en la BD. EX1 sí existe: 2.1 devolver el usuario Si no existe: 2.1 Devolver FALSE
Precondición	Id de usuario., rol de usuario	
Postcondición		
Excepciones	EX1: Error al leer la BD	

Caso de uso	Dar alta usuario *Ilustración 17. Pantalla de información personal y modificar usuario*.	
Actor	Usuario	
Propósito	Dar de alta un usuario, hacer un registro (Sing-up)	
Resumen	USUARIO: 2.Rellenar campos con datos personales: - Username, Email, contraseña, nombre, apellidos - Aceptar condiciones de uso.	SISTEMA: 1.Mostrar pantalla “registrar usuario” 3.Se activa el botón “Aceptar” 4.Consulta si el username y email ya están registrado. EX1 5. Si el username y/o email existen: 5.1 Muestra un mensaje y vuelve al paso 2. Si ni el username ni el email existen: 5.1.1 Insertar datos en la BD. EX2 6.Mostrar pantalla de login
Precondición		
Postcondición	El usuario queda registrado en la BD con un id numérico auto genérico	
Excepciones	EX1: Error al leer la BD EX2: Error al escribir a la BD	

Caso de uso	Modificar usuario *Ilustración 17. Pantalla de información personal y modificar usuario*	
Actor	Usuario y administrador	
Propósito	Dar de alta un usuario, hacer un registro (Sing-up)	
Resumen	USUARIO: 2.Rellenar campos con datos personales: - Username (este campo solo lo puede cambiar un administrador), Email, contraseña, nombre, apellidos No hay un “Eliminar usuario” ya que he preferido usar la variante “Baneado”, el usuario tendrá 3 posibles roles, usuario, administrador o baneado. Un usuario baneado no podrá hacer login en la plataforma. Un usuario no puede cambiar su rol, esto solo puede hacerlo un administrador.	SISTEMA: 1.Muestra pantalla “Mi perfil” 3.Consulta si el username y email ya están registrado. EX1 5.Si el username y/o email existen: 5.1 Muestra un mensaje y vuelve al paso 2. Si ni el username ni el email existen: 5.1. Insertar datos en la BD. EX2 6.Mostrar mensaje “Datos modificados correctamente”
Precondición	Id, nombre, apellidos, password, username, horas, votos, valoración	
Postcondición	Los datos son actualizados en la BD	
Excepciones	EX1: Error al leer la BD EX2: Error al escribir a la BD	

Caso de uso	Recuperar contraseña. *Ilustración 14. Pantalla de recuperación de contraseña*.	
Actor	Usuario	
Propósito	Dar de alta un usuario, hacer un registro (Sing-up)	
Resumen	USUARIO: 2.El usuario rellena el campo email	SISTEMA: 1.Mostrar pantalla de recuperar contraseña 3.Consulta si el email existe. EX1 Si existe: 3.1. Se envía la contraseña por mail a la dirección indicada. EX3 3.2. Se muestra la pantalla de login Si no existe: 3.1. Se muestra un mensaje al usuario 3.2. repetimos paso 1
Precondición	Email del usuario que quiere recuperar su contraseña	
Postcondición	Se envía un mail a la dirección indicada con la pass del usuario correspondiente	
Excepciones	EX1: Error al leer la BD EX3: El servidor de correo no está disponible	

Caso de uso	Valorar usuario *Ilustración 19. Pantalla de valoraciones*..	
Actor	Usuario	
Propósito	Dar de alta un usuario, hacer un registro (Sing-up)	
Resumen	USUARIO: 1.Selecciona el icono para votar a otro usuario 3.El usuario introduce la valoración que crea conveniente y paga las horas que correspondan.	SISTEMA: 2.El sistema muestra la pantalla para votar al usuario. 4.El sistema lee de la BD la cantidad de votos que tiene, y la valoración que tiene, con estos datos y la valoración dada por el usuario, se calcula la nueva valoración media. EX1 5.Se guarda la nueva valoración media en la BD y se le suma 1 a la cantidad de votos y se le suman las horas correspondientes. EX2
Precondición	Id, puntuación	
Postcondición	El usuario queda actualizado con la nueva puntuación	
Excepciones	EX1: Error al leer la BD EX2: Error al escribir a la BD	

Caso de uso	Contactar con otro usuario (Botón de contactar en cada row)	
Actor	Usuario	
Propósito	Dar de alta un usuario, hacer un registro (Sing-up)	
Resumen	USUARIO: 1.Selecciona el icono para contactar con otro usuario	SISTEMA: 2.El sistema manda un email al usuario que ha creado la oferta indicando que hay alguien con email xxx@xxx.xx interesado en su oferta para que se ponga en contacto con él. EX3
Precondición	Id del usuario que quiere contactar, y email del usuario con el que se quiere contactar	
Postcondición	El usuario que intenta contactar debe esperar la respuesta del ofertante.	
Excepciones	EX3: El servidor de correo no está disponible	

Caso de uso	Consultar oferta	
Actor	Usuario	
Propósito	Recoger una oferta concreta para, posiblemente, mostrarla o manipularla posteriormente.	
Resumen	USUARIO:	SISTEMA: 1.Se hace una petición a la BD con el id de la oferta 2.Se devuelve la oferta. EX1 Si no existe se devuelve False. EX1
Precondición	Id de la oferta a consultar	
Postcondición	Se devuelve la oferta buscada	
Excepciones	EX1: Error al leer la BD	

Caso de uso	Listar ofertas y Mis ofertas *Ilustración 16. Pantalla de misOfertas.*	
Actor	Usuario	
Propósito	Dar de alta un usuario, hacer un registro (Sing-up)	
Resumen	USUARIO: 2.Si el usuario logueado quiere ver sus ofertas, ha de hacer click en el enlace a sus ofertas (Mis ofertas)	SISTEMA: 1.Se muestra la lista total de ofertas, excluyendo las que son del usuario logueado EX1. (Listado) 3.Se devuelve una lista con únicamente las ofertas del usuario logueado EX1. (Mis ofertas))
Precondición		
Postcondición	Devolver una lista de ofertas en función de donde haya hecho click el usuario.	
Excepciones	EX1: Error al leer la BD	

Caso de uso	Dar de alta oferta *Ilustración 16. Pantalla de misOfertas.*	
Actor	Usuario	
Propósito	Dar de alta un usuario, hacer un registro (Sing-up)	
Resumen	USUARIO: 2.Selecciona una categoría, una subcategoría e introduce una breve descripción, aquí además puede poner horarios de disponibilidad, etc.	SISTEMA: 1.Muestra el formulario para añadir ofertas 3.Se inserta la oferta en la BD. EX2 4.Se actualiza la lista de ofertas, es decir, se hace un reload de la página para que el usuario vea la tabla actualizada. EX1
Precondición	categoría, subcategoría y descripción.	
Postcondición	La oferta queda almacenada en la BD.	
Excepciones	EX1: Error al leer la BD EX2: Error al escribir a la BD	

Caso de uso	Modificar ofertas (Botón de modificar en cada row)	
Actor	Usuario	
Propósito	Dar de alta un usuario, hacer un registro (Sing-up)	
Resumen	USUARIO: 1.El usuario hace click en “Mis ofertas” 3.El usuario hace click en la oferta que quiere modificar. 5.El usuario modifica los valores deseados. 6.Se aceptan los cambios.	SISTEMA: 2.Se muestra la lista de ofertas pertenecientes al usuario logueado. EX1 4.Se rellenan los campos con los valores de la oferta. 7.Se actualiza la BD con los cambios realizados. EX2 8. Se actualiza la lista de ofertas, es decir, se hace un reload de la página para que el usuario vea la tabla actualizada. EX1
Precondición	Id de la oferta a consultar	
Postcondición	La oferta se guarda en la BD con los nuevos valores.	
Excepciones	EX1: Error al leer la BD EX2: Error al escribir a la BD	

Caso de uso	Eliminar ofertas (Botón de eliminar en cada row)	
Actor	Usuario	
Propósito	Dar de alta un usuario, hacer un registro (Sing-up)	
Resumen	USUARIO: 1.El usuario hace click en “Mis ofertas” 3.El usuario hace click en la oferta que quiere eliminar.	SISTEMA: 2.Se muestra la lista de ofertas pertenecientes al usuario logueado. EX1 4.Se elimina la oferta de la BD. EX2 5. Se actualiza la lista de ofertas, es decir, se hace un reload de la página para que el usuario vea la tabla actualizada. EX1
Precondición	Id de la oferta a consultar	
Postcondición	Se elimina el registro correspondiente a la oferta eliminada.	
Excepciones	EX1: Error al leer la BD EX2: Error al escribir a la BD	

Caso de uso	Banear usuario Pantalla de Listado de usuarios → botón modificar → Pantalla modificar Usuario)	
Actor	Administrador	
Propósito	Dar de alta un usuario, hacer un registro (Sing-up)	
Resumen	USUARIO: 1.El administrador pide que se muestre una lista de usuarios. 3.Se selecciona el usuario que se pretende banear. 4.Se selecciona el rol “Baneado” 5.Se acepta el cambio.	SISTEMA: 2.Se muestra la lista de todos los usuarios. EX1 6.Se modifica en la BD el rol del usuario a Baneado. EX2
Precondición	Id del usuario a banear y rol que se le asigna	
Postcondición	Se actualiza el rol del usuario a Baneado, este usuario no podrá hacer login hasta que se le vuelva a cambiar el rol.	
Excepciones	EX1: Error al leer la BD EX2: Error al escribir a la BD	

3. Diseño

El diseño lo he realizado basándome tanto en los conocimientos adquiridos en los CFGS, como en la asignatura de *ingeniería del software*.

3.1 Diagrama de arquitectura

Como se puede ver en la ilustración 4, el usuario hace una petición a través de la vista, esta solicita la información al controlador, el cual se la pide al modelo, este se la devuelve al controlador, el controlador se encargara de la gestión de posibles errores, o de dar formato a la información según las necesidades, una vez todo esto esté gestionado, devolverá la información a la vista, la cual se la mostrara al usuario.

Ilustración 4. Diagrama de arquitectura.

3.2 Diagrama de clases

Para la elaboración del diagrama de clases, me he baso un diseño ER y el lenguaje UML, como se puede ver en la ilustración 5. Con esto no solo se consigue una representación gráfica, la cual para mayor entendimiento deberá ser leída con los requisitos funcionales y no funcionales que se adjuntan en el punto 2.1 Análisis de requisitos.

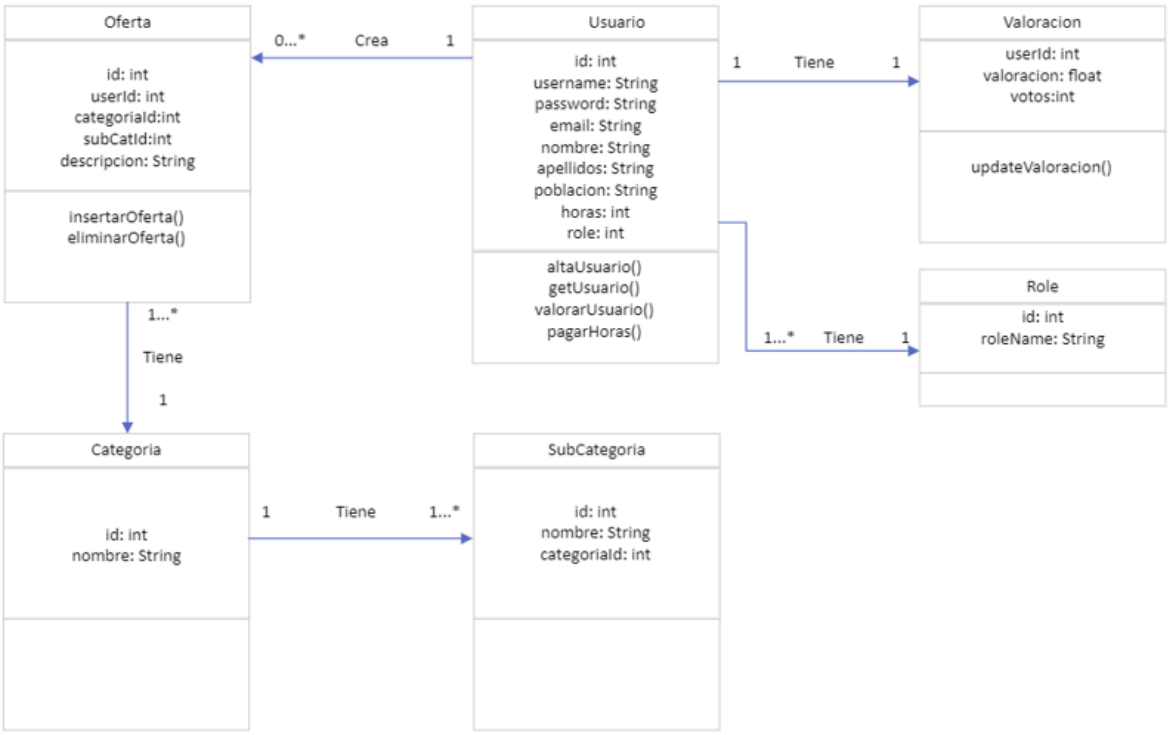

Ilustración 5. Diagrama de clases.

3.3 Diseño relacional de base de datos

A partir del diseño lógico, se ha creado un conjunto de tablas, vistas y triggers en MySql. Para una mayor agilidad, las tablas han sido indexadas, y se han generado dos vistas, una para usuarios y otra para ofertas, ya que he considerado que podrían llegar a almacenar un gran volumen de datos, lo resultaría en lentas consultas si se hiciera directamente a las tablas.

A continuación, en la ilustración 6 se muestra la estructura básica de dicha base de datos.

Ilustración 6. Diseño relacional de base de datos.

3.4 Diseño lógico de la base de datos

El diseño lógico de la base de datos es bastante sencillo, ya que la mayor parte de código se implementa en el frontend, por ser un proyecto web y no un proyecto de base de datos. Pero voy a explicar el trigger, que me ahorro gran parte de trabajo si lo hubiera hecho en el frontend:

Dado que cuando un usuario se registra se deben inicializar tanto su valoración como la cantidad de veces que se ha sido votado o puntuado, y como se observa en el apartado anterior (Diseño relacional de base de datos), las valoraciones y la cantidad de votos que tiene el usuario no están directamente en la tabla de usuarios (users). Inicialmente esto lo hacía en el frontend, primero se insertaba el usuario, acto seguido se preguntaba a la base de datos por el ultimo id insertado, y con ello se hacía un UPDATE a la tabla valoraciones, pero es mucho más limpio si, directamente, al hacer el INSERT de un alta de usuario, se actualiza de forma automática la tabla valoraciones.

3.4.1 Tablas

En este apartado se muestran los scripts utilizados para la creación de las tablas de la base de datos.

Tabla de roles de usuario: contendrá el identificador único de cada tipo de rol.

```
CREATE TABLE `role` (
`id` int NOT NULL,
`roleName` varchar(45) DEFAULT NULL,
PRIMARY KEY (`id`),
UNIQUE KEY `id_UNIQUE` (`id`),
UNIQUE KEY `role_name_UNIQUE` (`roleName`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;
```

Instrucción INSERT que nos crea tres registros en la tabla ‘role’, ejecutar esta instrucción es necesario, ya que en el código fuente se hacen comparaciones y se habilitan unas funcionalidades u otras en función del rol de usuario conectado.

```
INSERT INTO role (id, roleName)
VALUES(1,'ADMIN'), (2,'USER'), (3,'BANNED');
```

Tabla de categorías: contiene el identificador y el nombre de cada categoría:

```
CREATE TABLE `categorias` (
   `id` int NOT NULL AUTO_INCREMENT,
   `nombre` varchar(45) NOT NULL,
   PRIMARY KEY (`id`),
   UNIQUE KEY `nombre_UNIQUE` (`nombre`),
   UNIQUE KEY `id_UNIQUE` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT
CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

Tabla de subcategorías: contiene el identificador y el nombre de cada subcategoría, pero, además, queda enlazada a la tabla “categorías” mediante una clave foránea.

```
CREATE TABLE `sub_categorias` (
`id` int NOT NULL AUTO_INCREMENT,
`nombre` varchar(45) NOT NULL,
`categoriasId` int NOT NULL,
PRIMARY KEY (`id`),
UNIQUE KEY `nombre_UNIQUE` (`nombre`),
UNIQUE KEY `id_UNIQUE` (`id`),
```

```
KEY `FK_subcat_categorias_idx` (`categoriasId`),
CONSTRAINT `FK_subcat_categorias` FOREIGN KEY (`categoriasId`)
REFERENCES `categorias` (`id`) ON DELETE CASCADE ON UPDATE
CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT
CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

Tabla de usuarios: guardara un registro por cada nuevo usuario que se dé de alta en la plataforma, contendrá todos sus datos personales, queda enlazada a la tabla “role” mediante una clave foránea:

```
CREATE TABLE `users` (
  `id` int NOT NULL AUTO_INCREMENT,
  `username` varchar(45) NOT NULL,
  `passw` varchar(100) DEFAULT NULL,
  `email` varchar(100) NOT NULL,
  `nombre` varchar(45) DEFAULT NULL,
  `apellidos` varchar(45) DEFAULT NULL,
  `poblacion` varchar(100) DEFAULT NULL,
  `horas` int DEFAULT '10',
  `role` int NOT NULL DEFAULT '2',
  PRIMARY KEY (`id`),
  UNIQUE KEY `username_UNIQUE` (`username`) /*!80000 INVISIBLE
*/,
  UNIQUE KEY `email_UNIQUE` (`email`),
  KEY `FK_users_role_idx` (`role`),
  CONSTRAINT `FK_users_role` FOREIGN KEY (`role`)
REFERENCES `role` (`id`) ON DELETE RESTRICT ON UPDATE
RESTRICT
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT
CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

Tabla ofertas: en esta tabla se insertará un nuevo registro cada vez que un usuario cree una oferta nueva. Cara registro constara de un identificador, el id del usuario que la inserta, la categoría y subcategoría a la que pertenece dicha oferta, y una breve descripción proporcionada por el usuario.

```
CREATE TABLE `ofertas` (
  `id` int NOT NULL AUTO_INCREMENT,
  `userId` int NOT NULL,
  `categoriaId` int NOT NULL,
  `subCatId` int NOT NULL,
  `descripcion` varchar(240) DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `FK_oferta_users_idx` (`userId`),
  KEY `FK_oferta_categorias_idx` (`categoriaId`,`subCatId`),
```

```
  KEY `FK_oferta_subCategoria_idx` (`subCatId`),
  CONSTRAINT `FK_oferta_subCategoria` FOREIGN KEY (`subCatId`)
REFERENCES `sub_categorias` (`id`),
  CONSTRAINT `FK_oferta_users` FOREIGN KEY (`userId`)
REFERENCES `users` (`id`) ON DELETE CASCADE ON UPDATE
CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT
CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

Tabla de valoraciones: cada vez que un usuario valora a otro se actualiza esta tabla, es decir, supongamos un usuario X que a de valorar a un usuario Y el cual le ha enseñado N horas de programación PHP, el usuario X valorara dicha "clase" con la nota que vea conveniente, y mediante un calculo sencillo se actualizara el campo "valoración" y se incrementara en 1 el campo "votos".

```
CREATE TABLE `valoraciones` (
  `userId` int NOT NULL,
  `valoracion` decimal(4,2) DEFAULT '0.00',
  `votos` int DEFAULT '0',
  PRIMARY KEY (`userId`),
  UNIQUE KEY `userId_UNIQUE` (`userId`),
  KEY `idUser_idx` (`userId`),
  CONSTRAINT `FK_valoraciones_users` FOREIGN KEY (`userId`)
REFERENCES `users` (`id`) ON DELETE CASCADE ON UPDATE
CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;
```

3.4.2 Triggers o disparadores

¿Qué son los triggers o disparadores?

Bien un trigger o disparador es un script el cual se ejecuta en función de algún suceso previamente programado en la base de datos. El siguiente trigger se ejecutará tras la inserción de un nuevo registro en la tabla "users", y lo que hará será crear un nuevo registro en la tabla "valoraciones" con el identificado del usuario registrado y los campos "valoración" y "votos" inicializados a 0:

Dependiendo de la versión de MySql que se nos haya instalado deberemos usar una de las 2 opciones:

Opción 1:

```
CREATE DEFINER=`root`@`localhost` TRIGGER
`users_AFTER_INSERT` AFTER INSERT ON `users` FOR EACH ROW
INSERT INTO valoraciones (userId,  valoracion, votos) VALUES(new.id,
0, 0);
```

Opción 2:

```
CREATE DEFINER=`root`@`localhost` TRIGGER
`users_AFTER_INSERT` AFTER INSERT ON `users` FOR EACH
ROW BEGIN
   INSERT INTO valoraciones (userId,  valoracion, votos)
VALUES(new.id, 0, 0);
END;
```

3.4.3 Vistas

Las vistas en una base de datos no son más que la unificación o combinación de campos de algunas tablas mediante consulta SQL. Se utilizan para agilizar la lectura de datos cuando hablamos de grandes cantidades o simplemente para obtener un conjunto de datos de una manera rápida y sencilla.

Vista v_ofertas: esta vista está diseñada para unificar en una sola consulta las tablas *ofertas, categorías, subcategorías, users* y *valoraciones*.

```
CREATE ALGORITHM=UNDEFINED DEFINER=`root`@`localhost` SQL
SECURITY DEFINER VIEW `bdt`.`v_ofertas` AS select
   `bdt`.`ofertas`.`id` AS `id`,
   `bdt`.`ofertas`.`userId` AS `userId`,
   `bdt`.`ofertas`.`categoriaId` AS `categoriaId`,
   `bdt`.`categorias`.`nombre` AS `catNombre`,
   `bdt`.`ofertas`.`descripcion` AS `descripcion`,
   `bdt`.`users`.`nombre` AS `nombre`,
   `bdt`.`users`.`apellidos` AS `apellidos`,
   `bdt`.`users`.`username` AS `username`,
   `bdt`.`users`.`email` AS `email`,
   `bdt`.`users`.`poblacion` AS `poblacion`,
   `bdt`.`valoraciones`.`valoracion` AS `valoracion`,
   `bdt`.`valoraciones`.`votos` AS `votos`,
   `bdt`.`sub_categorias`.`id` AS `subCatId`,
   `bdt`.`sub_categorias`.`nombre` AS `subCatNombre`
  from ((((`bdt`.`ofertas`
      join `bdt`.`users` on((`bdt`.`ofertas`.`userId` = `bdt`.`users`.`id`)))
      join `bdt`.`categorias` on((`bdt`.`ofertas`.`categoriaId` =
`bdt`.`categorias`.`id`)))
      join `bdt`.`sub_categorias` on((`bdt`.`ofertas`.`subCatId` =
`bdt`.`sub_categorias`.`id`)))
      join `bdt`.`valoraciones` on((`bdt`.`users`.`id` =
`bdt`.`valoraciones`.`userId`)));
```

Vista v_ users: esta vista está diseñada para unificar en una sola consulta las tablas *users, role* y *valoraciones*.

```
CREATE ALGORITHM=UNDEFINED DEFINER=`root`@`localhost` SQL
SECURITY DEFINER VIEW `bdt`.`v_users` AS select
   `bdt`.`users`.`id` AS `id`,
   `bdt`.`users`.`username` AS `username`,
   `bdt`.`users`.`passw` AS `passw`,
   `bdt`.`users`.`email` AS `email`,
   `bdt`.`users`.`nombre` AS `nombre`,
   `bdt`.`users`.`apellidos` AS `apellidos`,
   `bdt`.`users`.`poblacion` AS `poblacion`,
   `bdt`.`users`.`horas` AS `horas`,
   `bdt`.`role`.`id` AS `roleId`,
   `bdt`.`role`.`roleName` AS `roleName`,
   `bdt`.`valoraciones`.`userId` AS `userId`,
   `bdt`.`valoraciones`.`valoracion` AS `valoracion`,
   `bdt`.`valoraciones`.`votos` AS `votos`
   from ((`bdt`.`users`
      join `bdt`.`role` on((`bdt`.`users`.`role` = `bdt`.`role`.`id`)))
      join `bdt`.`valoraciones` on((`bdt`.`users`.`id` =
`bdt`.`valoraciones`.`userId`)));
```

3.5 Listado de ficheros y procedimientos

A continuación, se describen los fichero, su funcionamiento, y tras cada fichero los procedimientos más relevantes de cada uno.
Se identifican los ficheros con un tono verde y los procedimientos o funciones con un tono azul.

Fichero	index.php
Descripción	Este fichero se encargará de la redirección de las url a las que se deben llamar en cada momento.
Parámetros	$action → acción a realizar, es decir, si redirecciona hacia un controlador (ctl) o hacia una vista (view). $option → controlador o vista a la que se debe redireccionar.
Funcionamiento	El administrador de la plataforma debe rellenar estos valores para que luego el controlador cree la conexión a la base de datos, desde la cual se harán todas las consultas.

3.5.1 Modelos

A continuación, se explican los modelos utilizados, la capa que “dará forma” a los datos.

Fichero	/bdt/model/role.php
Descripción	Este fichero contiene el modelo de los roles de usuario en la base de datos.
Parámetros	$id → id del rol $roleName → nombre del rol.
Funcionamiento	Se encarga de la gestión de roles.

Fichero	/bdt/model/valoraciones.php
Descripción	Este fichero contiene el modelo de las subcategorias
Parámetros	$valUserId → id del usuario. $valoracion → valoración del usuario. $votos → cantidad de veces que el usuario ha sido votado.
Funcionamiento	Se encarga de la gestión de las valoraciones.

Fichero	/bdt/model/Categoria.php
Descripción	Este fichero contiene el modelo de las categorías
Parámetros	$id → id de la categoría. $catNombre → nombre de la categoría.
Funcionamiento	Se encarga de la gestión de las categorías. Es decir, crea categorías, y manipula sus datos.

Función	getCategorias($orderBy = "nombre", $logSql=false): bool|array|null
Descripción	función que devuelve las categorías.
Parámetros	$ orderBy → Columna por la que se desean ordenar las categorías. $logSql → Boleano para grabar el log del resultado de la ejecución. $query → Sentencia que se va a ejecutar, en este caso: *"SELECT * FROM categorias ORDER BY {$orderBy};";*
Funcionamiento	Se crea la sentencia indicada, y se envía a la función *getRows*, la cual la ejecutará, retornará el resultado a esta, y esta devolverá un lista de categorías.
Salida	Resultado de la sentencia sql en formato: [[{key, value}, {key, value}, {key, value}], [{key, value}, {key, value}, {key, value}]] False si algo ha fallado. Graba un log si se lo indicamos o si ha fallado algo.

Fichero	/bdt/model/subCategoria.php
Descripción	Este fichero contiene el modelo de las subcategorías.
Parámetros	$id → id de la subcategoría. $nombre → nombre de la subcategoría. $catId → id de la categoría a la que pertenece esta subcategoría.
Funcionamiento	Se encarga de la gestión de las subcategorías.

Función	getSubCategorias ($orderBy = "nombre", $logSql=false): bool|array|null
Descripción	función que devuelve las categorías.
Parámetros	$ orderBy → Columna por la que se desean ordenar las categorías. $logSql → Boleano para grabar el log del resultado de la ejecución. $query → Sentencia que se va a ejecutar, en este caso: *"SELECT * FROM sub_categorias ORDER BY {$orderBy};";*
Funcionamiento	Se crea la sentencia indicada, y se envía a la función *getRows()*, la cual la ejecutará, retornará el resultado a esta, y esta devolverá un lista de sub_categorias.
Salida	Resultado de la sentencia sql en formato: [[{key, value}, {key, value}, {key, value}], [{key, value}, {key, value}, {key, value}]] False si algo ha fallado. Graba un log si se lo indicamos o si ha fallado algo.

Fichero	/bdt/model/Usuario.php
Descripción	Este fichero contiene el modelo del usuario.
Parámetros	use Role → herencia de Role. use Valoraciones → herencia de Valoraciones. $id → id del usuario. $username → username del usuario. $password → contraseña del usuario. $email → correo electrónico del usuario. $nombre → nombre del usuario. $apellidos → apellidos del usuario. $poblacion → población del usuario. $horas → horas que le quedan disponibles al usuario para aprender de alguien.
Funcionamiento	Se encarga de la gestión de las categorías. Es decir, crea categorías, y manipula sus datos.

Función	altaUsuario()
Descripción	Inserta un usuario de la base de datos.
Parámetros	$sql → sentencia INSERT a ejecutar, en este caso: *"INSERT INTO users (username, passw, email, nombre, apellidos, poblacion, horas) VALUES ('{$this->username}', '{$this->password}', '{$this->email}', '{$this->nombre}', '{$this->apellidos}', '{$this->poblacion}', 10);";*
Funcionamiento	Se crea la sentencia indicada, y se envía a la función *executeSql*, la cual la ejecutará
Salida	true si todo ha ido bien. false si algo ha fallado.

Función	getUserByUserId ($userId)
Descripción	Busca un usuario por su id.
Parámetros	$ userId → id del usuario a buscar $sql → sentencia SELECT a ejecutar, en este caso: *"SELECT * FROM v_users WHERE username='{$* userId*}';"*
Funcionamiento	Se crea la sentencia indicada, y se envía a la función *executeSql*, la cual la ejecutará, y rellenaremos los campos del propio modelo Usuario
Salida	true si todo ha ido bien. false si algo ha fallado.

Función	modificarUsuario()
Descripción	Modifica un usuario de la base de datos.
Parámetros	$sql → sentencia UPDATE a ejecutar, en este caso 2 sentencias: La primera UPDATEARA la tabla users con los nuevos valores, y la segunda UPDATEARA la tabla valoraciones con los valores correspondientes a dicho usuario.
Funcionamiento	Se crean las sentencia indicadas, y se envían a la función *executeSql*, la cual la ejecutará
Salida	true si todo ha ido bien. false si algo ha fallado.

Función	getUserByUserName ($userName)
Descripción	Busca un usuario por su nombre.
Parámetros	$userName→ username del usuario a buscar $sql → sentencia SELECT a ejecutar, en este caso: *"SELECT * FROM v_users WHERE username='{$userName}';"*
Funcionamiento	Se crea la sentencia indicada, y se envía a la función *executeSql*, la cual la ejecutará, y rellenaremos los campos del propio modelo Usuario
Salida	true si todo ha ido bien. false si algo ha fallado.

Función	eliminarUsuario($userId)
Descripción	Elimina una oferta de la base de datos
Parámetros	$userId→ id del usuario a eliminar. $sql → sentencia DELETE a ejecutar, en este caso: *"DELETE FROM users WHERE id=$userId;"*
Funcionamiento	Se crea la sentencia indicada, y se envía a la función *executeSql*, la cual la ejecutará
Salida	true si todo ha ido bien. false si algo ha fallado.

Función	updateValoracion ($userId, $valoracion, $voto)
Descripción	Elimina una oferta de la base de datos
Parámetros	$userId→ id del usuario a valorar. $valoracion → nueva valoración que se le asigna. $votos → nuvos votos que se le asignan. $sql → sentencia UPDATE a ejecutar, en este caso: *"UPDATE valoraciones* *SET valoracion = '".$valoracion."', votos = '".$voto."'* *WHERE userId=".$userId.";"*
Funcionamiento	Se crea la sentencia indicada, y se envía a la función *executeSql*, la cual la ejecutará
Salida	true si todo ha ido bien. false si algo ha fallado.

Función	pagarHoras($selectedUserId, $horas)
Descripción	Sirve para que un usuario X pague n horas a otro usuario Z
Parámetros	$selectedUserId → id del usuario al que se le van a pagar las horas. $horas → cantidad de horas a pagar Esta función ejecuta 2 sentencias, la primera sumara las horas al usuario Z y la segunda restara las horas al usuario X $sql → sentencia UPDATE a ejecutar, en este caso: *"UPDATE users SET horas = horas -".$horas." WHERE id='".$this->id."';"* "UPDATE users SET horas = horas +".$horas." WHERE id ='".$selectedUserId."';"
Funcionamiento	Se crea la sentencia indicada, y se envía a la función *executeSql*, la cual la ejecutará
Salida	true si todo ha ido bien. false si algo ha fallado.

Fichero	/bdt/model/ Oferta.php
Descripción	Este fichero contiene el modelo de las ofertas
Parámetros	use Categoria → herencia de Categoria. use subCategoria → herencia de subCategoria. $id → id de la oferta. $userId → id del usuario propietario de la oferta. $categoriaId → id de la categoría a la que pertenece la oferta. $subCatId → id de la subcategoría a la que pertenece la oferta. $descripcion → texto descriptivo de la oferta.
Funcionamiento	Se encarga de la gestión de las categorías. Es decir, crea ofertas, y manipula sus datos. Esta clase tiene distintas funciones que detallamos a continuación.

Función	insertarOferta ()
Descripción	Inserta una oferta en la base de datos
Parámetros	$sql → sentencia INSERT a ejecutar, en este caso: *"INSERT INTO ofertas (userId, categoriaId, subCatId, descripcion)* *VALUES ('$this->userId', '{$this->categoriaId}', '{$this->subCatId}', '{$this->descripcion}');"*
Funcionamiento	Se crea la sentencia indicada, y se envía a la función *executeSql*, la cual la ejecutará
Salida	true si todo ha ido bien. false si algo ha fallado.

Función	modificarOferta ()
Descripción	modifica una oferta existente en la base de datos
Parámetros	$ofertaId → id de la oferta a modificar. $categoriaId → id de la nueva categoría que se le asignara a la oferta. $subCatId → id de la nueva subcategoría que se le asignara a la oferta. $descripcion → texto descriptivo que se le asignara a la oferta. $sql → sentencia INSERT a ejecutar, en este caso: *"UPDATE ofertas* *SET categoriaId = '".$categoriaId."',* *subCatId = '".$subCatId."',* *descripcion = '".$descripcion."'* *WHERE id=".$ofertaId.";"*
Funcionamiento	Se crea la sentencia indicada, y se envía a la función *executeSql*, la cual la ejecutará
Salida	true si todo ha ido bien. false si algo ha fallado.

Función	eliminarOferta ()
Descripción	Elimina una oferta de la base de datos
Parámetros	$ofertaId → id de la oferta a eliminar. $sql → sentencia DELETE a ejecutar, en este caso: *"DELETE FROM ofertas WHERE id=$ofertaId;"*
Funcionamiento	Se crea la sentencia indicada, y se envía a la función *executeSql*, la cual la ejecutará
Salida	true si todo ha ido bien. false si algo ha fallado.

Función	autoFillOferta ()
Descripción	Genera una variable oferta
Parámetros	$ofertaId → id de la oferta a generar. $sql → sentencia SELECT a ejecutar, en este caso: *"SELECT * FROM ofertas WHERE id='{$ofertaId}';";*
Funcionamiento	Se crea la sentencia indicada, y se envía a la función *getRow*, la cual la ejecutará
Salida	false si algo ha fallado.

3.5.2 Controladores

Los controladores son el intermediario entre las vistas y el modelo. El controlador recibe una petición a través de la vista, seguramente pedida por un usuario. El controlador la pide al modelo y la devuelve a la vista.

Fichero	/bdt/controller/config.php
Descripción	En este fichero se parametriza la configuración de acceso a la base de datos
Parámetros	$hostCfg, $usernameCfg, $passwordCfg, $dbnameCfg, $portCfg
Funcionamiento	El administrador de la plataforma debe rellenar estos valores para que luego el controlador cree la conexión a la base de datos, desde la cual se harán todas las consultas.

Fichero	/bdt/controller/DAOController.php
Descripción	En él se crea la conexión a la base de datos, todas las consultas realizadas en la plataforma pasan por aquí.
Parámetros	$host, $username, $password, $dbname, $port
Funcionamiento	Recibe una consulta a la base de datos, la realiza, gestiona errores, y devuelve el resultado. Al final del fichero se crean 2 conexiones, una “directa” ($conn = new DAOController()), la cual ha de ser referenciada en cada uno de los ficheros que se usa, algo que resulta un tanto incomodo y por eso he decidido usarla así solo en 3 ficheros (/bdt/controller/ajax_ctl/)a modos de ejemplo, para el resto de consultas he puesto la conexión en una variable de sesión y es usada desde dicha variable.
Salida	----------

Función	initDb(): void
Descripción	Establece la conexiona a la base de datos.
Parámetros	$host, $username, $password, $dbname, $port
Funcionamiento	Recibe los parámetros indicados, y establece la conexión
Salida	----------

Función	getConn()
Descripción	Un getter que devuelve la conexión
Parámetros	$this->conn
Salida	$this->conn

Función	getRow($sql, $logSql=false): bool|array|null
Descripción	función que recibe una sentencia sql, la ejecuta y devuelve el resultado
Parámetros	$sql → sentencia SELECT a ejecutar $logSql → Boleano para grabar el log del resultado de la ejecución.
Salida	Resultado de la sentencia sql en formato {key, value} False si algo ha fallado. Graba un log si se lo indicamos o si ha fallado algo.

Función	executeSql($sql, $logSql=false): mysqli_result|bool|Exception
Descripción	función que recibe una sentencia sql y la ejecuta
Parámetros	$sql → sentencia INSERT, UPDATE, DELETE a ejecutar $logSql → Boleano para grabar el log del resultado de la ejecución
Salida	Resultado de la sentencia sql. False si algo ha fallado. Graba un log si se lo indicamos o si ha fallado algo.

Función	getRows($sql, $logSql=false): bool|array|null
Descripción	Función que recibe una sentencia sql, la ejecuta y devuelve el resultado
Parámetros	$sql → sentencia SELECT a ejecutar $logSql → Boleano para grabar el log del resultado de la ejecución.
Salida	Resultado de la sentencia sql en formato: [[{key, value}, {key, value}, {key, value}], [{key, value}, {key, value}, {key, value}]] False si algo ha fallado. Graba un log si se lo indicamos o si ha fallado algo.

Función	getLastInsertedId(): bool|int|null
Descripción	Función que devuelve el ultimo id generado en la base de datos
Parámetros	----------
Salida	Ultimo id insertado en la base de datos

Función	showAlert($msg): void
Descripción	Función que mostrara una alerta (en formato JavaScript) por pantalla.
Parámetros	$msg → Mensaje que se mostrara en la alerta.
Salida	----------

Fichero	/bdt/controller/contactar_ctl.php
Descripción	Controlador encargado del envío de correos cuando un usuario X desea hacerle saber a otro usuario Z que está interesado en una de sus ofertas.
Parámetros	$userId → usuario que quiere contactar $to → usuario con el que se quiere contactar
Funcionamiento	El fichero enviara un email avisando al usuario $to que el usuario $userId está interesado en una de sus ofertas.
Salida	Email enviado o una alerta advirtiendo de lo contrario.

Fichero	/bdt/controller/eliminarUsuario_ctl.php
Descripción	Controlador que recibe el id de una oferta y elimina dicha oferta de la base de datos.
Parámetros	$ofertaId → id del usuario que se desea eliminar
Funcionamiento	El fichero recibe el identificador de un usuario, se crea una variable de tipo Usuario, la cual es un modelo de *Usuario* que contiene la función de eliminación de usuarios (eliminarUsuario($userId)), esta función es llamada pasándole el identificador correspondiente y se realiza la eliminación.
Salida	Nos devolverá a la página de origen, el listado manejado por un administrador dado que solo un administrador puede eliminar usuarios.

Fichero	/bdt/controller/insertarOferta_ctl.php
Descripción	Controlador que recibe una nueva categoría, una subcategoría y una descripción de la oferta y inserta un nuevo registro en la base de datos.
Parámetros	$selCat → id de la categoría a la que pertenecerá la nueva oferta. $subCatId → id de la subcategoría a la que pertenecerá la nueva oferta. $desc → descripción de la nueva oferta.
Funcionamiento	Se comprueba que se hayan seleccionado una categoría y una subcategoría (de lo contrario se muestra un mensaje en formato JS alert) y se introduce un nuevo registro en la base de datos mediante la función insertarOferta().
Salida	----------

Fichero	/bdt/controller/insertarCategoria_ctl.php
Descripción	Controlador que recibe una nueva categoría y/o una nueva subcategoría y la inserta en la base de datos
Parámetros	$categoria → nombre de la nueva categoría $subCat → nombre de la nueva subcategoría
Funcionamiento	Se comprueba que la $categoria no sea null, ya que cualquier subcategoría ha de tener una categoría asociada. Se comprueba si la categoría existe, de ser así, se entiende que lo que se va a insertar será una subcategoría, se toma el id de la categoría, de lo contrario se hace el insert de la categoría. Si la categoría existe, se comprueba que la subcategoría no exista, de ser así se hace el insert de la nueva subcategoría
Salida	----------

Fichero	/bdt/controller/logged_ctl.php
Descripción	Controlador que se añade a todas las vistas para evitar que *"se nos cuelen por detrás"*
Parámetros	$_SESSION['logged'] → variable de sesión que nos indica si hay un usuario logueado o no.
Funcionamiento	Se comprueba si la variable $_SESSION['logged'] NO existe y si es null, de ser así, significa que no hay ningún usuario logueado y no podrán acceder a páginas que no se deba desde F12. Cualquier intento de acceder mediante la inspección de elementos de la plataforma, lo devolverá a la página de bienvenida.
Salida	----------

Fichero	/bdt/controller/login_ctl.php
Descripción	Controlador encargado de la gestión de log-in de usuarios.
Parámetros	$usuario_req → username del usuario que intenta hacer log-in. $password → contraseña del usuario que intenta hacer log-in.
Funcionamiento	Se recogen el username y la contraseña proporcionada, mediante la función validarUserPassword($usuario_req, $password), se comprueba que el usuario existe y que rol tiene, en caso que el rol sea 1 o 2 (admin o user) se le da paso a la plataforma, en caso que el rol sea 3 (baneado) se le avisa mediante una alerta y no se le da paso.
Salida	----------

Fichero	/bdt/controller/personalInfo_ctl.php
Descripción	Controlador encargado de la información personal de los usuarios.
Parámetros	$enc → variable de tipo Encriptador. $userId → id del usuario a modificar. $userName → userName del usuario a modificar. $emailReg → email del usuario a modificar. $password → contraseña actual del usuario a modificar. $newPass = → nueva contraseña del usuario a modificar. $repeatNewPass → comprobación de la nueva contraseña $nombreReg → nombre del usuario a modificar. $apellidos → apellidos del usuario a modificar. $poblacion → población del usuario a modificar. $valoracion → valoración del usuario a modificar. $votos → votos del usuario a modificar. $horas → horas del usuario a modificar. $role → rol del usuario a modificar.
Funcionamiento	En caso de que el usuario desee cambiar su contraseña, se le hace introducir dos veces la nueva contraseña y que ambas coincidan. Si el usuario no quiere cambiar su contraseña, se comprueba que la antigua no sea nula. El resto de los campos que necesitan algún tipo de verificación, como el formato del email, se verifican en la propia vista (/bdt/view/personalInfo_view.php) Finalmente se llama a la función modificarUsuario(), la cual realizara las modificaciones pertinentes.
Salida	----------

Fichero	/bdt/controller/validarUserPassword_ctl.php
Descripción	Controlador encargado de la comprobación de la existencia de un usuario con la correspondiente contraseña.
Parámetros	$userName→ email del usuario que intenta hacer log-in. $password → contraseña del usuario que intenta hacer log-in.
Funcionamiento	La función recibe los dos parámetros arriba indicados, mediante el $userName busca el usuario correspondiente, si lo encuentra
Salida	----------

Fichero	/bdt/controller/register_ctl.php
Descripción	ontrolador encargado del registro de usuarios.
Parámetros	$nombreReg → nombre del nuevo usuario. $apellidosReg → apellidos del nuevo usuario. $userNameReg → username del nuevo usuario. $passwordReg → contraseña del nuevo usuario. $poblacionReg → población del nuevo usuario. $emailReg → email del nuevo usuario.
Funcionamiento	El fichero recibe los datos del usuario que se quiere registrar en la plataforma, con estos datos se crea una variable de tipo Usuario, se llama a la función altaUsuario() y a su vez esta ejecuta la sentencia mediante executeSql($sql) la cual contiene un try/catch que comprobara si existe o no dicho usuario, ya que al ser campos únicos el email y el username, no puede haber coincidencia y la propia base de datos devuelve el error tratado en dicho try/catch
Salida	Alerta si la base de datos ha devuelto algún error.

Fichero	/bdt/controller/valorar_ctl.php
Descripción	Controlador encargado de la actualización de datos referente a votos y horas, cuando un usuario X quiere valorar y/o gratificar con N horas a un usuario Z.
Parámetros	$usuario → el usuario logueado. $selectedUserId → id del usuario que se va a valorar. $valoracion → valoración del usuario seleccionado. $votosSelectedUser → votos del usuario seleccionado $voto → voto que el usuario logueado quiere dar al usuario seleccionado. $pHoras → horas que el usuario logueado da al usuario seleccionado.
Funcionamiento	Se busca el usuario seleccionado mediante la función getUserByUserId(id), se coge su valoración actual y se calcula la nueva valoración en función de: *(($valoracion * $votosSelectedUser) + $voto) / ($votosSelectedUser + 1);* Acto seguido se actualiza la valoración mediante la función updateValoracion(selectedUserId, $newValoracion, $votosSelectedUser+1) y se le suman las horas, si es necesario, mediante la función pagarHoras($selectedUserId, $pHoras)
Salida	true si todo ha ido bien. false si algo ha fallado.

Fichero	/bdt/controller/recuperarPass_ctl.php
Descripción	Desde este controlador se genera el cuerpo de un mensaje, que posteriormente se enviará por correo electrónico incluyendo la contraseña.
Parámetros	$to→ email del usuario que quiere recuperar la contraseña.
Funcionamiento	El fichero recibe un email, comprueba si existencia, en la base de dados. Si dicho email existe, se envía un email con la contraseña perteneciente a dicho email
Salida	----------

Fichero	/bdt/controller/loguot_ctl.php
Descripción	Este controlador nos devuelve a la página de inicia de la plataforma, e inicializa las variables logged y usuario.
Parámetros	$_SESSION['logged'] → Variable que nos indica si hay alguien logueado. $_SESSION['usuario'] → Variable que contiene el Usuario logueado.
Funcionamiento	Cuando un usuario se desloguea de la plataforma, se inicializan las variables arriba indicadas a *false* y *null* respectivamente y nos devuelve a la página de bienvenida.
Salida	----------

Fichero	/bdt/controller/modificarOferta_ctl.php
Descripción	Controlador encargado de la modificación de ofertas.
Parámetros	$ofertaId → id de la oferta a modificar. $categoriaId → id de la categoría que se le asignara a la oferta. $subCatId → id de la subcategoría que se le asignara a la oferta. $descripcion → descripción de la oferta.
Funcionamiento	Se recogen los parámetros arriba indicados, y se envían a la función modificarOferta(), para que esta la modifique.
Salida	----------

3.5.3 Controladores Ajax

Es un controlador, pero este funciona de forma asíncrona, es decir, es capaz de hacer las peticiones en segundo plano. Esto se ha aplicado para las consultas que podían tener una respuesta con un mayor volumen de datos, como por ejemplo rellenar el listado de ofertas, como se puede ver en la ilustración 15, o el listado de usuarios como se puede ver en la ilustración 16.

Fichero	/bdt/controller/ajax_ctl/reloadTable_ctl.php
Descripción	Controlador encargado de devolver la lista de ofertas existentes en la base de datos.
Parámetros	$categoria → categoría a la que pertenecen las ofertas que se quieren buscar. $subcategoria →subcategoría a la que pertenecen las ofertas que se quieren buscar. $userId → Id del usuario que realiza la petición $origen → página desde la que se realiza la petición $sql → Sentencia que se va a ejecutar, en este caso: "SELECT * FROM v_ofertas WHERE categoriaId LIKE '%{$categoria}' AND subCatId LIKE '%{$subcategoria}' ";
Funcionamiento	Se crea la sentencia indicada, además se tiene en cuenta si la petición proviene de *origen=listado* u *origen=misOfertas*, ya que si proviene de *listado*, el usuario lo que pretende es ver las ofertas que no son suyas, por el contrario, si proviene de *misOfertas*, el usuario lo que quiere es ver sus ofertas, por lo tanto a $sql se le concatena *" AND userId != '{$userId}' "* para el caso *listado* o *" AND userId = '{$userId}' "* para el caso *misOfertas*. Todo esto se envía a la función *getRows,* la cual la ejecutará, retornará el resultado a esta, y esta devolverá un lista de usuarios.
Salida	Resultado de la sentencia sql en formato: [[{key, value}, {key, value}, {key, value}], [{key, value}, {key, value}, {key, value}]] False si algo ha fallado.

Fichero	/bdt/controller/ajax_ctl/listarUsuarios_ctl.php
Descripción	Controlador encargado de devolver la lista de usuarios existentes en la base de datos.
Parámetros	$sql → Sentencia que se va a ejecutar, en este caso: *"SELECT * FROM v_users ORDER BY userId, username;";*
Funcionamiento	Se crea la sentencia indicada, y se envía a la función *getRows*, la cual la ejecutará, retornará el resultado a esta, y esta devolverá un lista de usuarios.
Salida	Resultado de la sentencia sql en formato: [[{key, value}, {key, value}, {key, value}], [{key, value}, {key, value}, {key, value}]] False si algo ha fallado.

Fichero	/bdt/controller/ajax_ctl/reloadSubCats_ctl.php
Descripción	Controlador encargado de devolver la lista de subcategorias existentes en la base de datos.
Parámetros	$categoria → categoría a la que pertenecen las subcategorías que se quieren buscar. $sql → Sentencia que se va a ejecutar, en este caso: *" SELECT * FROM sub_categorias WHERE categoriasId LIKE '{$categoria}';";*
Funcionamiento	Se crea la sentencia indicada, y se envía a la función *getRows*, la cual la ejecutará, retornará el resultado a esta, y esta devolverá un lista de usuarios.
Salida	Resultado de la sentencia sql en formato: [[{key, value}, {key, value}, {key, value}], [{key, value}, {key, value}, {key, value}]] False si algo ha fallado.

3.5.4 Utilities

Los utilities, son un conjunto de algoritmos creados por mí, como por ejemplo funciones JavaScript o encriptación y desencriptación de contraseñas.

Fichero	/bdt/utilities/sripts.js
Descripción	Este fichero contiene la mayoría de las funciones JS utilizadas en la plataforma.
Parámetros	El fichero como tal ni tiene ni recibe parámetros, pero cada una de las funciones tiene o recibe los parámetros necesarios.
Funcionamiento	----------
Salida	----------

Función	checker()
Descripción	Función que muestra u oculta la contraseña del usuario
Parámetros	----------
Funcionamiento	Comprueba si el switch de la contraseña está marcado o no, y en función de cómo esté, llamara a la función showPasword para mostrar la contraseña, o la función hidePassword para ocultarla.
Salida	----------

Función	showPassword()
Descripción	Función que cambia el tipo de objeto a "text", mostrando así la contraseña del usuario
Parámetros	----------
Funcionamiento	Esta función busca el elemento password, y lo convierte en un elemento de tipo text
Salida	----------

Función	hidePassword()
Descripción	Función que cambia el tipo de objeto a "password", ocultando así la contraseña del usuario
Parámetros	----------
Funcionamiento	Esta función busca el elemento *password*, y lo convierte en un elemento de tipo password
Salida	----------

Función	checkerAceptar()
Descripción	Función que activa el botón de aceptar, para poder registrarse, al aceptar los términos y condiciones de uso.
Parámetros	obj → es el checkbox que hay que marcar para aceptar las condiciones de uso.
Funcionamiento	Recibe el obj, se mira si está marcado o no, y cuando se marca, se habilita el botón *Aceptar*
Salida	----------

Función	volverAtras()
Descripción	Función que nos devolverá a la página de la que venimos
Parámetros	origen → es la vista de la que venimos.
Funcionamiento	Genera un texto tal que: ***'?action=view&option='+origen;*** y lo asigna a window.location, de este modo nos redirigirá a fichero *index.php*, el cual gestionara la redirección a donde tenemos que ir según *origen*.
Salida	----------

Función	populateDataTable(tableName, pageLength= 5)
Descripción	Función que recibe el nombre de una table y genera un DataTable con el formato indicado.
Parámetros	tableName → nombre de la tabla a la que se le ha de dar formato. pageLength → cantidad de registros a mostrar por página. Por defecto 5. order → Columnas por las que queremos ordenar. ordering → Permite que las columnas sean ordenadas haciendo clic en la cabecera (true/false) paging → Muestra(true) u oculta(false) el select con las cantidades lengthMenu → Menu desplegable "Show entries" ([5, 10, 15]) pageLength → Valor inicial del menu desplegable "Show entries" oLanguage → Texto a mostrar en la sección info. Info → Muestra(true) u oculta(false) la sección info: "Showing 1 to 10 of 16 entries. scrollY → Altura de la tabla a partir de la cual queremos que nos muestre el scroll vertical. scrollX → Anchura de la tabla a partir de la cual queremos que nos muestre el scroll horizontal. pagingType → muestra los botones *First, Previous, Next y Last* ('full_numbers'). language: { lengthMenu: 'Mostrar _MENU_ registros por página', zeroRecords: 'Nothing found - sorry', infoEmpty: 'No records available', } → conjunto de mensaje a mostrar.
Funcionamiento	Recibe el nombre de la tabla que tiene que buscar, y le da el formato en función de todas las variable arriba mencionadas.
Salida	----------

Función	confirmarEliminarOferta(ofertaId)
Descripción	Función que muestra una alerta pidiendo confirmación al eliminar una oferta
Parámetros	ofertaId → identificador de la oferta que queremos eliminar.
Funcionamiento	Mostrará una alerta de confirmación o cancelación preguntando que si realmente se desea eliminar la oferta seleccionada.
Salida	Mensaje de alerta.

Función	confirmarEliminarUsuario(userId)
Descripción	Función que muestra una alerta pidiendo confirmación al eliminar un usuario.
Parámetros	userId → identificador del usuario que queremos eliminar.
Funcionamiento	Mostrará una alerta de confirmación o cancelación preguntando que si realmente se desea eliminar el usuario seleccionado.
Salida	Mensaje de alerta.

Función	reloadSubCat(subCatList)
Descripción	Función que recibe una lista de subcategorías y rellena el select correspondiente.
Parámetros	subCatList → lista de subcategorías.
Funcionamiento	Recibe una lista de subcategorías, con ella montara un *select*. Una vez montado se lanza el trigger 'change', propio del select para que se actualice la lista en función de categoría seleccionada.
Salida	----------

Función	populateUsuarios(usuarios)
Descripción	Función que crear y rellena la tabla de usuarios
Parámetros	usuarios → lista de usuarios a mostrar.
Funcionamiento	Función que recibe una lista de usuarios, y con ella crea y rellena una tabla. Y finalmente llamara a la función *popuateDataTable(tableName)*, para que esta de formato a la tabla.
Salida	----------

Función	populateTable(ofertas, role)
Descripción	Función que crear y rellena las tablas de misOfertas y listado
Parámetros	ofertas → lista de ofertas a mostrar. role → rol del usuario conectado.
Funcionamiento	Función que recibe una lista de ofertas, y con ella crea y rellena una tabla, además, tendrá en cuenta el rol del usuario conectado, dependiendo del rol mostrara las ofertas propias del administrador (rol == 1), o las ofertas propias del usuario (rol != 1). La función también busca el origen del que procedemos para saber si tenemos que mostrar opciones de eliminar y modificar ofertas o no. Y finalmente llamara a la función *popuateDataTable(tableName)*, para que esta de formato a la tabla.
Salida	----------

Función	topMenuLogged(role)
Descripción	Función que mostrara la parte del menú superior.
Parámetros	role → rol del usuario conectado.
Funcionamiento	Es función muestra las opciones del menú superior, en función de si el usuario ya ha hecho login o no, y si es un usuario normal o por el contrario es un administrador.
Salida	----------

Función	topMenuLogout()
Descripción	Función que ocultara la parte del menú superior.
Parámetros	role → rol del usuario conectado.
Funcionamiento	Es función muestra las opciones del menú superior cuando aún no se ha hecho login.
Salida	----------

Fichero	/bdt/utilities/encriptador.php
Descripción	Clase para encriptar y desencriptar las contraseñas
Parámetros	SUBS_ALPHABET → alfabeto con el que se encriptara el texto correspondiente.
Funcionamiento	Esta clase tiene dos funciones, *encrypt($text)* y *decrypt($text)*, encargadas de encriptar o desencriptar el texto que reciban.
Salida	----------

Función	encrypt($text)
Descripción	Función que recibe un texto lo encripta
Parámetros	$text → texto a encriptar.
Funcionamiento	Recibe un texto y mediante el algoritmo "AES-256-ECB" y el alfabeto arriba indicado, encriptara el texto recibido.
Salida	Texto encriptado

Función	decrypt($text)
Descripción	Función que recibe un texto lo desencripta
Parámetros	$text → texto a desencriptar.
Funcionamiento	Recibe un texto y mediante el algoritmo "AES-256-ECB" y el alfabeto arriba indicado, desencriptara el texto recibido.
Salida	Texto desencriptado.

Fichero	/bdt/utilities/sendMail.php
Descripción	Clase para el manejo de correos
Parámetros	$to → destinatario del correo. $subject → título del correo. $body → cuerpo del correo.
Funcionamiento	Recibirá los parámetros arriba indicados, y enviará un mail, desde la cuenta de correo *bancodeltiempo2023@gmail.com* al destinatario correspondiente. Además, se ha generado un template con el fondo de pantalla de la plataforma.
Salida	Email.

2.2.5.5 Vistas

Las vistas son los ficheros que, mediante la combinación de código HTML, CSS, JavaScript y PHP, dan firma a lo que finalmente ve el usuario

Parámetros	----------
Funcionamiento	Este fichero contiene el head de la página, ya que siempre va a ser el mismo así será más fácil de importar. Este mismo fichero, importara el header del html. Desde él, también se llaman a las librerías necesarias para la realización de la plataforma (jquery, datables, bootstrap5, etc) además de cargar los estilos propios.
Salida	----------

Fichero	/bdt/view/header.php
Descripción	Menú superior de la página.
Parámetros	$usuario → variable de tipo usuario.
Funcionamiento	Este fichero carga y muestra el menú superior de la página. La variable usuario se utiliza para saber que usuario está conectado, y de este modo saber su información personal.
Salida	----------

Fichero	/bdt/view/footer.php
Descripción	Footer de la página.
Parámetros	----------
Funcionamiento	Este fichero contiene el footer de la página, ya que siempre va a ser el mismo, el cual contiene 4 botones, que nos llevaran al Facebook, al LinkedIn, y a la propia página de la UOC, respectivamente, y el 4º botón abrirá la venta de correo, perada para enviar un email a la administración de la plataforma.
Salida	----------

Fichero	/bdt/view/filtros_view.php
Descripción	Este fichero contiene los filtros de categorías y subcategorías
Parámetros	$categorias → lista de categorías. $usuario → usuario conectado. $origen → es la vista de la que venimos.
Funcionamiento	El fichero recibe una lista con todas las categorías existentes, con ella monta un select, y una vez montado, ejecutara su propio trigger 'change', el cual llamará al controlador reloadSubCats_ctl, este a su vez devolverá un lista de subcategorías, en la respuesta se llamará la función reloadSubCats y montara el select de subcategorías
Salida	----------

Fichero	/bdt/view/listado_view.php
Descripción	Este fichero muestra la pantalla de listado.
Parámetros	$usuario → el usuario conectado. $_SESSION['origen'] → Dado que una vez se ha hecho login se redirige a esta página, se asigna por primera vez un origen, en este caso *listado*.
Funcionamiento	El usuario ve el listado de todas las ofertas, menos las suyas, desde aquí puede filtrar, buscar, contactar o redirigirse a la pantalla de valoraciones.
Salida	----------

Fichero	/bdt/view/insertarCategoria_view.php
Descripción	Este fichero muestra la pantalla de inserción de categorías.
Parámetros	$usuario → el usuario conectado. $categorias → lista de todas las categorías existentes. $subCategorias → lista de todas las subcategorías existentes.
Funcionamiento	Recogerá una subcategoría i/o una categoría, y llamará al controlador de insertarCategoria_ctl, el cual tras las comprobaciones pertinentes decidirá si inserta o no los valores recogidos.
Salida	Texto de la categoría. Texto de la subcategoría.

Fichero	/bdt/view/misOfertas_view.php
Descripción	Este fichero muestra la pantalla de las ofertas propias del usuario logueado.
Parámetros	$usuario → el usuario conectado. $selCat → id de la categoría seleccionada. $subCatId → id de la subcategoría seleccionada. $desc → descripción dada por el usuario.
Funcionamiento	El usuario puede listar, modificar o insertar nuevas ofertas. Para insertar una nueva oferta debe seleccionar una categoría, una subcategoría y opcionalmente da una descripción de la oferta. El fichero recoge los datos y los envía al controlador insertarOferta_ctl, el cual tras las comprobaciones pertinentes decidirá si inserta o no los valores recogidos.
Salida	----------

Fichero	/bdt/view/listarUsuarios_view.php
Descripción	Este fichero muestra la pantalla de listado de usuarios (solo visible por administradores).
Parámetros	$usuario → el usuario conectado. $listaUsuarios → la lista completa de los usuarios registrados en la plataforma.
Funcionamiento	Se monta una tabla con todos los usuarios registrados en la plataforma. Desde aquí los administradores pueden eliminar usuarios o acceder a su información personal.
Salida	----------

Fichero	/bdt/view/login_view.php
Descripción	Este fichero muestra la pantalla de login.
Parámetros	$_SESSION['logged'] = false; → Esta variable controla si el usuario esta logueado o no $_SESSION['usuario'] = null; → esta variable contendrá el usuario logueado.
Funcionamiento	Pantalla que recoge un email y una contraseña y lo envía al controlador login_ctl.php, el cual dará paso, o no, a la plataforma.
Salida	Username del usuario. Contraseña del usuario.

Fichero	/bdt/view/modificarOfertas_view.php
Descripción	Este fichero muestra la pantalla que permite al usuario modificar sus ofertas.
Parámetros	$usuario → el usuario conectado. $ofertaId → Identificador de la oferta a modificar. $oferta → oferta que se va a modificar.
Funcionamiento	El usuario llega a esta pantalla desde su listado de ofertas, aquí puede elegir nueva categoría, nueva subcategoría y modificar la descripción, estos datos son recogidos y enviados al controlador modificarOferta_ctl, el cual hará la modificación correspondiente en la oferta seleccionada.
Salida	$ofertaId → Identificador de la oferta a modificar. $categoriaId → categoría de la oferta a modificar. $subCatId → subcategoría de la oferta a modificar. $descripcion → descripción de la oferta a modificar.

Fichero	/bdt/view/recuperarPass_view.php
Descripción	Es fichero muestra la pantalla de recuperación de contraseña.
Parámetros	----------
Funcionamiento	El usuario rellena el campo email con su correo electrónico y le da a *enviar*. Este envia la dirección al controlador recuperarPass_ctl.php el cual comprobara si existe dicha dirección o en la base de datos y realizara las acciones comentadas.
Salida	Email por comprobar.

Fichero	/bdt/view/ofertante_view.php
Descripción	Este fichero permite al usuario conectado, ver los datos detallados de otro usuario.
Parámetros	$selectedUser → el usuario que queremos consultar. $oferta → oferta que se va a consultar. $ofertaId → identificador de la oferta correspondiente. $usuario → usuario conectado.
Funcionamiento	El usuario llega a esta pantalla desde su listado de ofertas, haciendo click, en cualquier Nick de la columna *ofertante.* Al hacer click en el Nick, se envían la $ofertaId, y el $userId, con esto se monta la vista que permite ver los datos del usuario deseado. Una vez se valora y paga con horas al usuario estos datos son eviados al controlador valorar_ctl.php el cual realizara los cambios necesarios.
Salida	id del usuario a valorar. Valoración que se le da al usuario a valorar(id). Horas que se le tienen que sumar al usuario seleccionado (id).

Fichero	/bdt/view/register_view.php
Descripción	Es fichero muestra la pantalla de registro.
Parámetros	$_SESSION['logged'] = false; → Esta variable controla si el usuario esta logueado o no $_SESSION['usuario'] = null; → esta variable contendrá el usuario logueado.
Funcionamiento	El usuario rellena los campos y acepta las condiciones de uso, estos datos son enviados al controlador register_ctl.php y este realiza las acciones necesarias.
Salida	$nombreReg → nombre del usuario que se va a registrar. $apellidosReg → apellidos del usuario que se va a registrar. $userName → username del usuario que se va a registrar. $passwordReg → contraseña del usuario que se va a registrar. $poblacionReg → población del usuario que se va a registrar. $emailReg → dirección de correo del usuario que se va a registrar.

Fichero	/bdt/view/personalInfo_view.php
Descripción	Este fichero permite al usuario conectado, ver sus propios datos y modificarlos. Dicho usuario accedera desde el menú superior haciendo click en *Información Personal*. Un administrador también accedera a esta pantalla, a través del menú desplegable que se encuentra en la parte superior izquierda (*Admin options*)
Parámetros	$enc → variable de tipo Encriptador. $selectedUser → el usuario que queremos consultar. $ofertaId → identificador de la oferta correspondiente. $usuario → usuario conectado. $role → rol del usuario conectado.
Funcionamiento	Se reciben los datos arriba indicados, y se monta la vista. En caso de que algún campo sea modificado, el botón *aceptar* enviara los datos al controlador personalInfo_ctl.php y este realizara los cambios pertinentes.
Salida	$userId → identificador del usuario consultado. $userName → username del usuario consultado. $emailReg → email del usuario consultado. $password → contraseña del usuario consultado. $newPass → nueva contraseña del usuario consultado. $repeatNew → repetición de la nueva contraseña (han de coincidir) $nombreReg → nombre del usuario consultado. $apellidos → apellidos del usuario consultado. $poblacion → población del usuario consultado. $valoracio → valoración del usuario consultado. $votos → votos del usuario consultado. $horas → horas del usuario consultado. $role → rol del usuario consultado.

Fichero	/bdt/view/welcome.php
Descripción	Es fichero contiene el <body> de la pantalla de bienvenida.
Parámetros	$_SESSION['logged'] = false; → Esta variable controla si el usuario esta logueado o no $_SESSION['usuario'] = null; → esta variable contendrá el usuario logueado.
Funcionamiento	Solo contiene el <body> de la primera pantalla que vera cualquier usuario al acceder a la plataforma.
Salida	----------

Fichero	/bdt/view/welcome_view.php
Descripción	Es fichero muestra la pantalla de bienvenida.
Parámetros	----------
Funcionamiento	Se cargan el head, el welcome y el footer, con todo ello se muestra la pantalla de bienvenida al usuario
Salida	----------

3.6 Pantallas

A continuación, se muestra el resultado visible final, es decir, lo que vera el usuario, el cual, gracias al framework Bootstrap 5, ha resultado bastante cómodo de realizar, en especial el diseño de tablas,

La ilustración 7 nos muestra la página de bienvenida del usuario, es decir, el index.php.

Ilustración 7. Página de bienvenida.

A continuación, en la ilustración 8 vemos el header o cabecera del administrador, es el header que tiene todas las opciones y funcionalidades.

Ilustración 8. Header de administrador.

La ilustración 9 nos muestra la cabecera que ve el usuario, como se puede observar, en ella falta el desplegable en el lado izquierdo, ya que este está solo accesible a usuarios con el rol de administrador.

Ilustración 9. Header de usuario.

En la ilustración 10 vemos el footer o pie de página, consta de cuatro iconos, los cuales, si los enumeramos de izquierda a derecha nos llevan a Facebook, LindekIn, a la página de la UOC, y el ultimo nos abre nuestro gestor de correo listo para contactar con la administración de la plataforma.

Ilustración 10. Footer.

La ilustración 11 nos muestra el filtro de búsqueda, consta de dos partes:

· La primera, dos desplegables con las categorías y subcategorías existentes en la base de datos.

· La segunda, un desplegable el cual hará que se muestre una cantidad de registros igual a la cantidad seleccionada. Y a su derecha una caja de texto la cual filtrara, mostrando solo los registros que contengan el texto escrito en cualquiera de sus campos coincidente con el texto en dicha caja de texto.

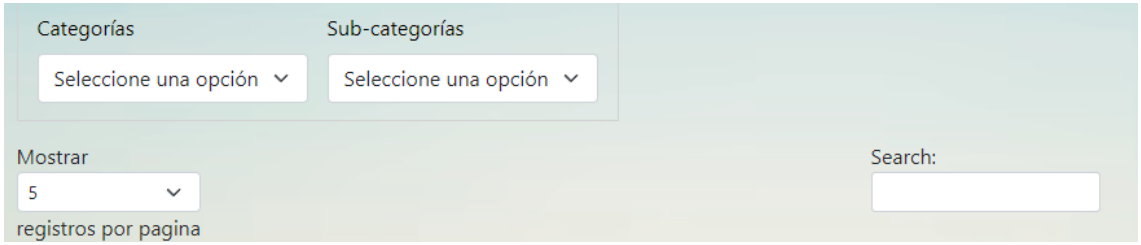

Ilustración 11. Filtros de búsqueda.

La ilustración 12 nos muestra la pantalla de registro de usuario, donde el usuario debe aceptar que sus datos puedan ser compartidos con otros usuarios para poder registrarse, ya que la manera de contactar entre usuarios es mediante su dirección de correo electrónico.

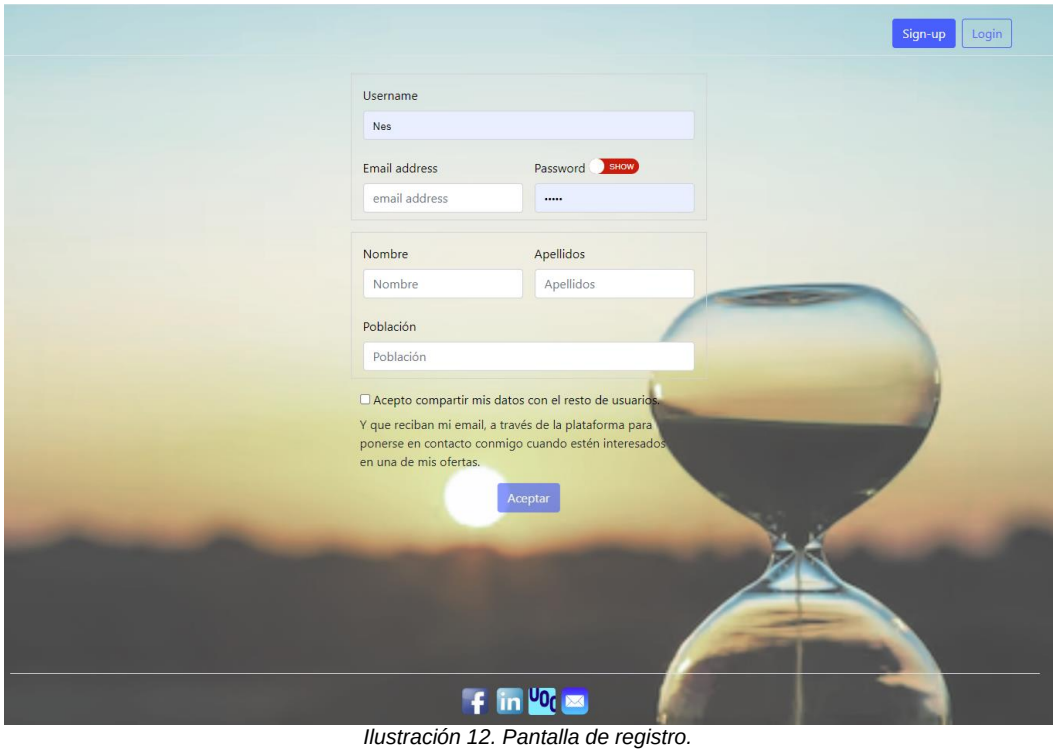

Ilustración 12. Pantalla de registro.

La ilustración 13 nos muestra la pantalla de login, es decir el acceso a la plataforma una vez el usuario se ha registrado en ella.

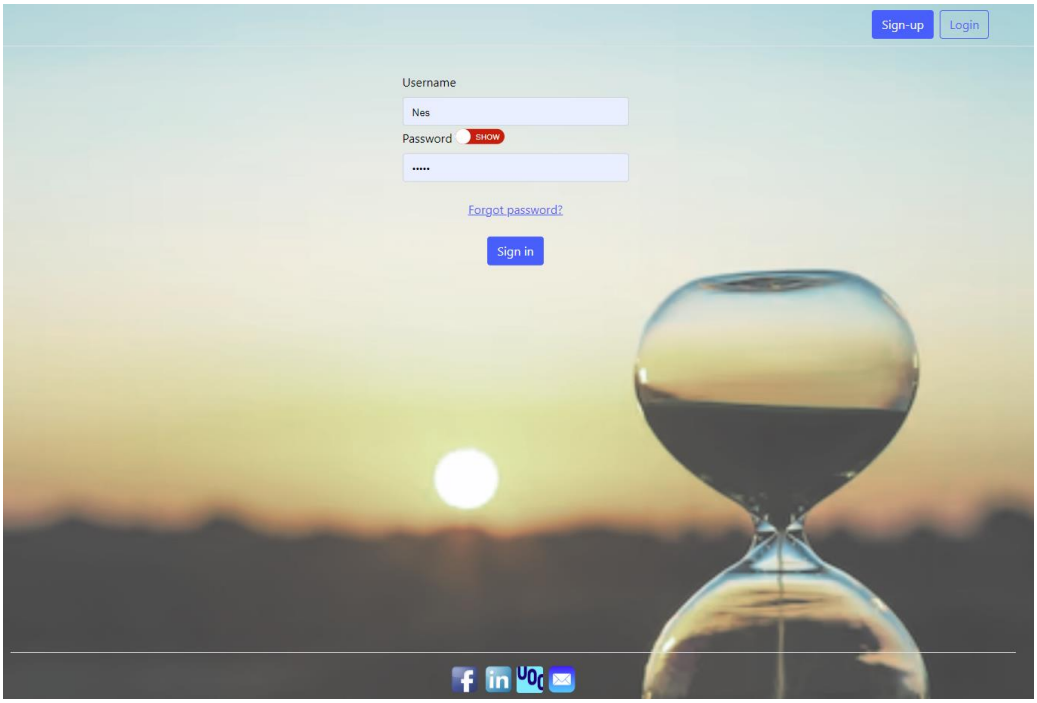

Ilustración 13. Pantalla de login.

La ilustración 14 nos muestra la pantalla desde la cual un usuario que no recuerda su contraseña, puede recuperarla mediante la dirección de correo con la cual se registró.

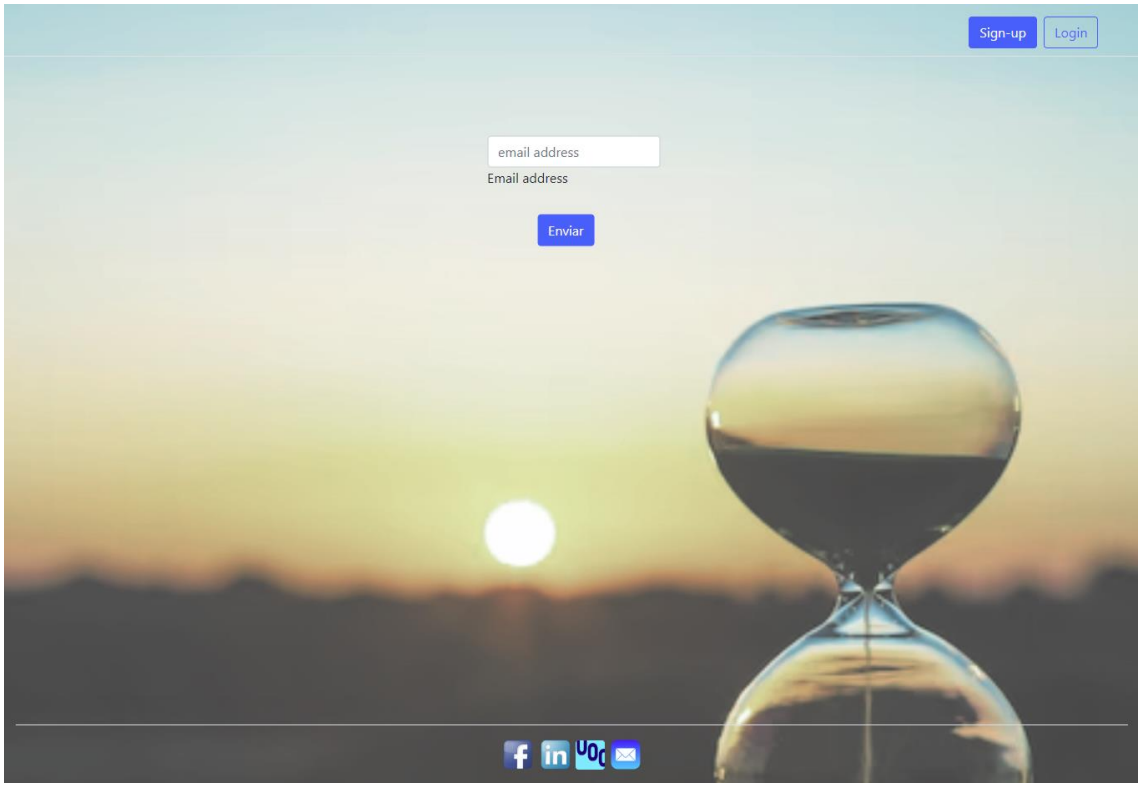

Ilustración 14. Pantalla de recuperación de contraseña.

En la ilustración 15 podemos ver la pantalla de listado de ofertas, la cual muestra al usuario todas las ofertas que no son suyas.

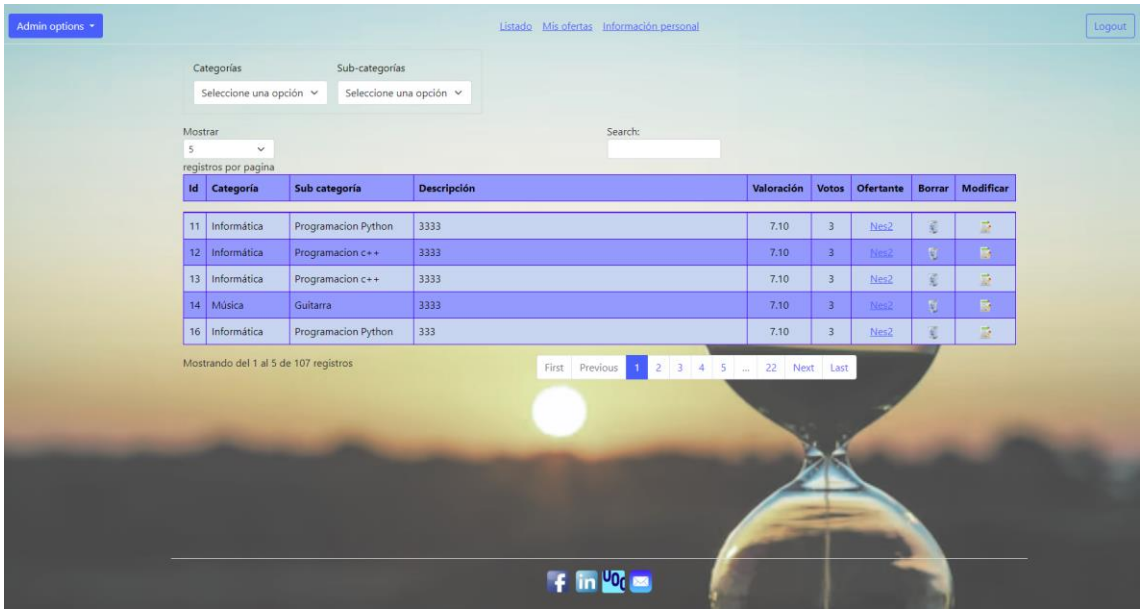

Ilustración 15. Pantalla de listado de ofertas.

En la ilustración 16 vemos la posibilidad de inserción de nuevas ofertas, además del listado de las ofertas propias únicamente del usuario.

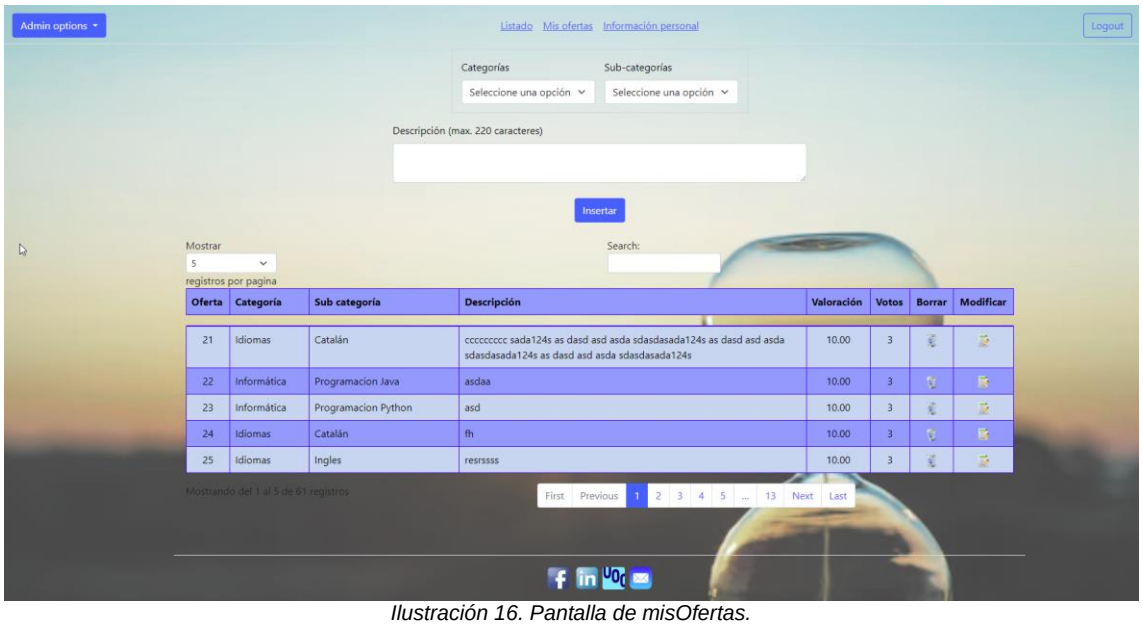

Ilustración 16. Pantalla de misOfertas.

La ilustración 17, nos muestra la pantalla donde o bien un administrador puede modificar cualquier campo referente a un usuario, o bien un usuario puede modificar únicamente los campos referentes a sus datos personales (campos del lado izquierdo)

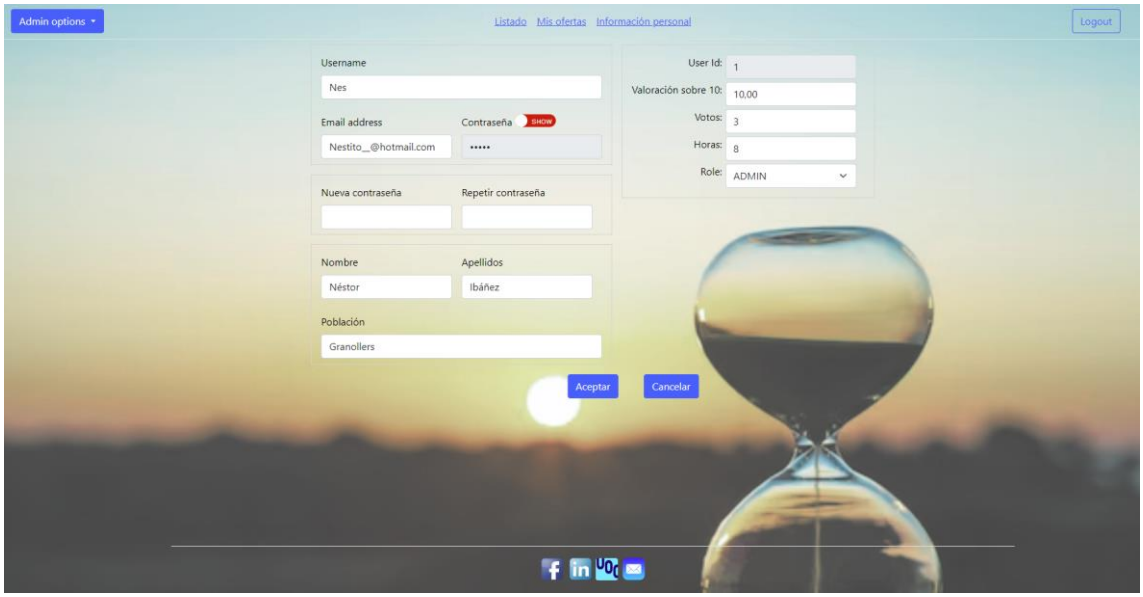

Ilustración 17. Pantalla de información personal y modificar usuario.

La ilustración 18 nos muestra el listado de usuarios registrados en la base de datos, solo accesible para administradores. Desde aquí un administrador puede eliminar un usuario, o modificar sus campos como se ha explicado en anteriormente en la ilustración 17.

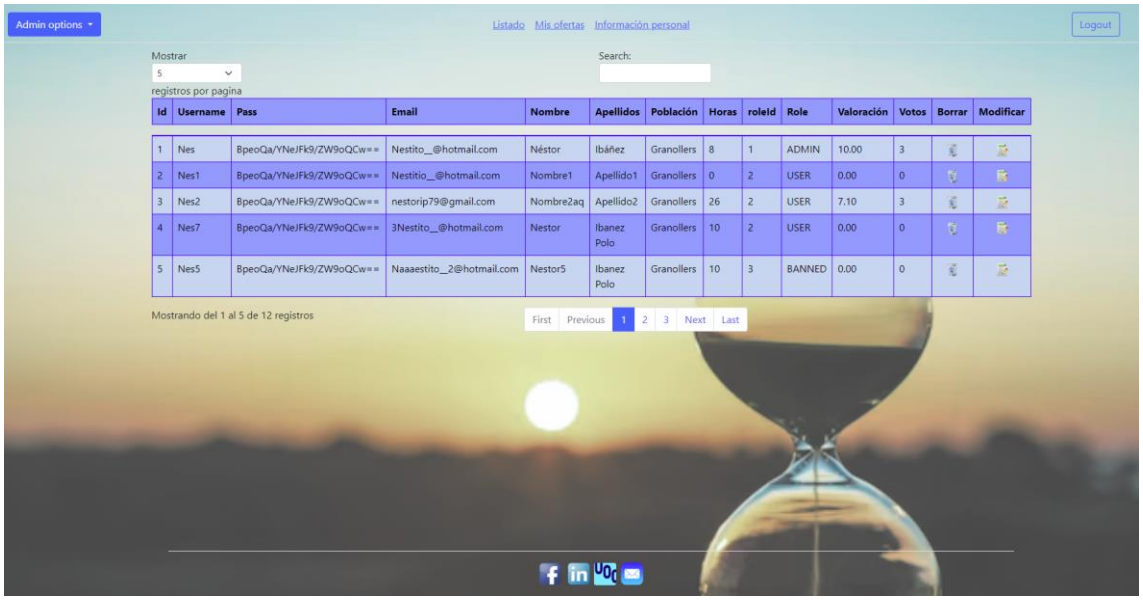

Ilustración 18. Pantalla de listado de usuarios.

La ilustración 19 muestra la información de un ofertante, se accede a ella seleccionado el usuario, que queremos ver, en la tabla de listado de ofertas (ilustración 15), en la columna “ofertante”. Esta pantalla también nos permite valorar y votar al usuario seleccionado.

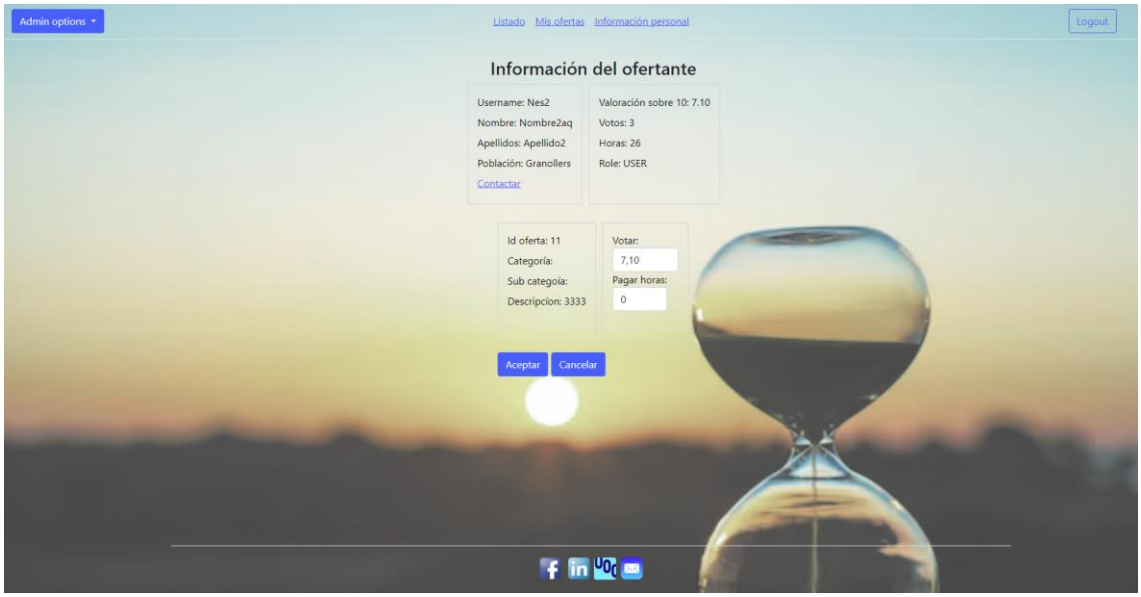

Ilustración 19. Pantalla de valoraciones.

La ilustración 20, permite al administrador la inserción de nuevas categorías y subcategorías.

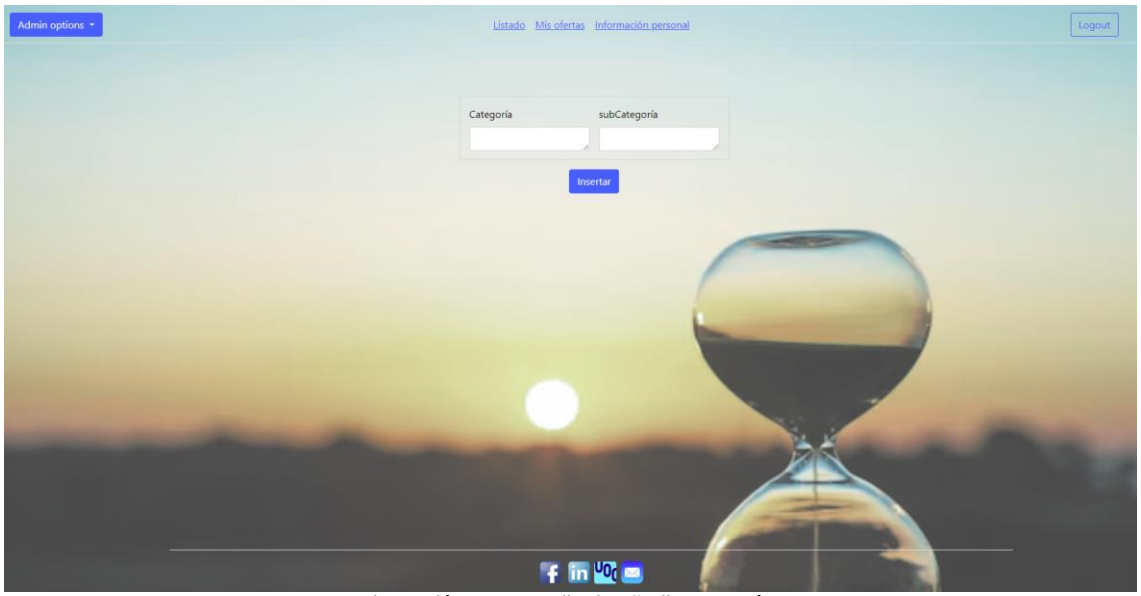

Ilustración 20. Pantalla de añadir categoría.

3.7 Otros ficheros y carpetas

/bdt/Schema.sql este fichero contiene las sentencias SQL que hay que ejecutar para montar la base de datos, con sus tablas, vista, trigger y lo necesario para el buen funcionamiento de la plataforma.

/bdt/view/css/estilos.css este fichero contiene los estilos personalizados que se han aplicado a la plataforma, algunos de ellos han sido obtenidos de las propias librerías de Boostrap y modificados al gusto.

/bdt/PHPMailer es la librería que hace posible él envió de correos electrónicos.

/bdt/logs/ en esta carpeta se generan los log de las funciones que ejecutan sentencias SQL.

4. Conclusiones

Para ser sincero, he decir que, eligiendo el Desarrollo Web como trabajo de fin de grado, fue como lanzarse a la piscina sin saber si habría agua o no, ya que, ciertamente mis conocimientos para el diseño de una plataforma web eran escasos y olvidados, pues desde que finalice los estudios del CFGS de Desarrollo de aplicaciones web, no había vuelto a hacer nada. Ciertamente, hay cosas que me han costado más de lo esperado y otras que aparte de ser muy interesantes, me han costado menos de lo que pensaba que lo harían cuando inicialmente plantee la propuesta de mi proyecto. Pero a pesar de todo, he conseguido alcanzar los objetivos planificados inicialmente.

Tras la finalización del proyecto, siento un gran orgullo, de poder haberlo finalizado sin la ayuda de nadie (a nivel de desarrollo, ya que para cualquier otro tema siempre he tenido el apoyo del profesorado). Siento que he adquirido muchos y nuevos conocimientos a través de la investigación necesaria a causa de la falta de conocimientos inicial.

En resumen, he disfrutado muchísimo aprendiendo y realizando un proyecto de desarrollo web y aplicando los conocimientos que he ido adquiriendo a lo largo del desarrollo.

Se ha de matizar que, aunque ha habido cambios respecto a la planificación inicial, esta se ha podido llevar a cabo con bastante exactitud. Por lo tanto, puedo decir con certeza que he aprendido como planificar un proyecto real, hacer un buen análisis de este y generar la documentación necesaria para su elaboración.

Algo que me ha quedado pendiente, y aunque no estaba en la planificación, pero me hubiera gustado desarrollar, hubiera sido un sistema de token o validación, para el registro en la plataforma, mediante respuesta a un link o código recibido en el correo electrónico del usuario que quiere registrarse. Establecer un sistema de horarios de disponibilidad mediante checkbox. También un sistema de chat donde los usuarios pudieran contactar entre ellos de una manera directa e inmediata.

5. Glosario

BACKEND	Encargado de conectar el servidor y/o base de datos con el frontend.
BD	Base de datos. *Conjunto de datos estructurados que pertenecen a un mismo contexto* [9].
CFGS	Ciclo formativo de grado superior.
ER	*Entity Relationship*. Modelo entidad relación, utilizado para el diseño conceptual de la base de datos.
FRAMEWORK	Conjunto de herramientas, desarrollada por terceros, que incluyéndolas en nuestro proyecto facilitan mucho las tareas.
FRONTEND	Parte visible, donde interactúan los usuarios, de una web.
INSERT	Sentencia que se utiliza para la inserción de un nuevo registro en la base de datos.
LOG-IN	Acción de entrar en la plataforma usando un nombre de usuario y una contraseña.
PASS	Abreviatura de password del inglés, que significa contraseña.
PHP	Lenguaje utilizado para el desarrollo web [2].
SQL	*Structured Query Language.* Lenguaje de consulta estructurada. Este lenguaje se utiliza para la gestión y manipulación de bases de datos.
UML	*Unified Modeling Language*. Lenguaje Unificado de Modelado. Diagrama que describe las clases de un sistema. [10]
UPDATE	Sentencia que se utiliza para la actualización de registros en la base de datos.

6. Bibliografía

[1] *Cómo crear un DIAGRAMA de GANTT en Excel [Cronograma usando los gráficos]* Disponible en: https://www.youtube.com/watch?v=chR6kx4btDQ [consulta: 06 de febrero de 2023].

[2] *Manual de PHP*. Disponible en: https://www.php.net/manual/es/ [consulta: en múltiples ocasiones entre el 17 de abril y el 30 de mayo de 2023].

[3] Instalar *certificado SSL GRATIS y activar HTTPS en tu Servidor WEB* https://www.youtube.com/watch?v=eM8tU9ZuRC0 [consulta: 20 de mayo de 2023] Disponible en:

[4] *Preparación del entorno en Linux* Disponible en: https://www.digitalocean.com/community/tutorials/how-to-install-linux-apache-mysql-php-lamp-stack-on-ubuntu-20-04-es [consulta: 10 de mayo de 2023]

[5] *Como instalar múltiples versiones de php* Disponible en: https://help.clouding.io/hc/es/articles/360021630059-C%C3%B3mo-Instalar-m%C3%BAltiples-versiones-de-PHP-7-2-7-4-and-8-0-en-Ubuntu-20-04 [consulta: 10 de mayo de 2023]

[6] *Como instalar PHP 8.2 en Ubuntu 22.04* Disponible en: https://techvblogs.com/blog/install-php-8-2-ubuntu-22-04 [consulta: 10 de mayo de 2023]

[7] *¿Cómo arreglar Apache2 no ejecutando archivos PHP?* Disponible en: https://support.hostinger.es/es/articles/3963793-como-arreglar-apache2-no-ejecutando-archivos-php [consulta: 10 de mayo de 2023]

[8] Instalar certificado SSL GRATIS y activar HTTPS en tu Servidor WEB Disponible en: https://www.youtube.com/watch?v=eM8tU9ZuRC0 [consulta: 20 de mayo de 2023]

[9] *Base de datos.* Disponible en: https://es.wikipedia.org/wiki/Base_de_datos [consulta: 8 de julio de 2023]

[10] *UML Diagrama de clases*. Disponible en: https://es.wikipedia.org/wiki/Diagrama_de_clases [consulta: 9 de julio de 2023]

7. Anexos

7.1 Manual y puesta en marcha

Partiendo de la base que la plataforma va a ser instalada en un sistema Ubuntu20, en la carpeta /var/www/bdt, a continuación, pongo todas las instrucciones que hay que seguir para su puesta en marcha y un breve resumen de lo que se está haciendo:

7.1.1 Instalación Apache y MySQL[4]

Se actualiza el repositorio de aplicaciones y se instalan los paquetes de apache2 y mysql-server

```
sudo apt update
sudo apt install apache2
sudo ufw allow in "Apache"
sudo apt install mysql-server
```

Entramos en la consola de MySQL y cambiamos la contraseña del usuario root a admin (puede ser cualquier otra siempre y cuando se configure correctamente en el fichero config.php)

```
mysql -u root -p
ALTER USER 'root'@'localhost' IDENTIFIED WITH
mysql_native_password BY 'admin';
FLUSH PRIVILEGES;
exit;
```

7.1.2 Instalación PHP[5] [6]

Se instalan paquetes necesarios:

```
sudo apt install pgp libapache2-mod-php php-mysql
sudo apt update
sudo apt install software-properties-common gnupg2 -y
sudo add-apt-repository ppa:ondrej/php
sudo update-alternatives --config php
sudo apt install php8.2-mysql
```

Se configura el host virtual en Ubuntu:

```
sudo mkdir/var/www/bdt
sudo chown -R $USER:$USER /var/www/bdt
```

Y se copia el siguiente codigo:

```
<VirtualHost *:80>
ServerName bdt
ServerAlias www.bdt.com
ServerAdmin webmaster@localhost
DocumentRoot /var/www/bdt
ErrorLog ${APACHE_LOG_DIR}/error.log
CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Guardamos y salimos

```
sudo a2ensite bdt
sudo a2dissite 000-default
sudo systemctl reload apache2
sudo chmod 777 -R /var/www/bdt
```

7.1.3 Configuración Apache2 cuando no ejecuta archivos PHP[7]

```
sudo nano /etc/apache2/apache2.conf
```

Y al final del fichero añadimos el siguiente codigo:

```
<FilesMatch \.php$>
SetHandler application/x-httpd-php
</FilesMatch>
```

Guardamos y salimos

Se habilitan y deshabilitan los módulos necesarios:

```
sudo a2dismod mpm_event && sudo a2enmod mpm_prefork && sudo a2enmod php8.2
sudo service apache2 restart
```

7.1.4 Instalar certificado de seguridad [8]

Se instala un certificado de seguridad y se autoconfigura:

```
sudo apt install snapd
sedo apt remove certbot
sudo snap install –classic certbot
sudo ln -s /snap/bin/certbot /usr/bin/certbot
```

Entramos en edición y modificamos el fichero bdt.conf (marco en rojo la línea a modificar), debemos poner la url donde tenemos alojado nuestro server

```
sudo nano /etc/apache2/sites-available/bdt.conf
```

```
<VirtualHost *:80>
    ServerName bdt
    ServerAlias bancodeltiempo2023.westeurope.cloudapp.azure.com
    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/bdt
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Guardamos y salimos

Configuramos de manera automática apache:

```
sudo certbot –apache
```

Se nos pedirá un correo electrónico, yo he puesto el de la plataforma (bancodeltiempo2023@gmail.com),

Nos preguntara si aceptamos los términos y condiciones de uso, le diremos que si (Y), a continuación, nos pregunta si queremos enviar al correo los certificados, podemos decirle que no (N). Y el último paso es elegir en cuál de los dominios queremos activar, seleccionaremos el que diga:
bancodeltiempo2023.westeurope.cloudapp.azure.com

Y ya tenemos una navegación segura.

7.2 Inserts para la base de datos

Adjunto una serie de INSERTS en la base de datos, para, en caso de pruebas, no tener que insertar valores manualmente a través de la plataforma, dichos inserts, deberán insertarse en el mismo orden que están puestos a continuación y prestar especial atención a los identificadores de cada registro, ya que estas sentencias están pensadas para una base de datos creada justo con las instrucciones dadas en el apartado 3.4 Diseño lógico de la base de datos. En caso de que ya hubiera registros añadidos con posterioridad, se deberán cambiar los identificadores de cada sentencia en función de los ya existentes en las correspondientes tablas.

```
INSERT INTO users (id, username, passw, email, nombre, apellidos, poblacion, horas, role)
VALUES (1, "Nes", "BpeoQa/YNeJFk9/ZW9oQCw==", "nestito__@hotmail.com", "Nestor", "ibanez", "Grn", 10, 1),
       (2, "admin", "BpeoQa/YNeJFk9/ZW9oQCw==", "admin@gmail.com", "nombre", "Apellido", "Bcn", 10, 1),
       (3, "user1", "BpeoQa/YNeJFk9/ZW9oQCw==", "user1@gmail.com", "nombre1", "Apellido1", "Bcn", 10, 2),
       (4, "user2", "BpeoQa/YNeJFk9/ZW9oQCw==", "user2@gmail.com", "nombre2", "Apellido2", "Bcn", 10, 2),
       (5, "user3", "BpeoQa/YNeJFk9/ZW9oQCw==", "user3@gmail.com", "nombre3", "Apellido3", "Bcn", 10, 3);
```

```
INSERT INTO categorias(id, nombre) VALUES (1, "Informática"), (2, "Música"), (3, "Idiomas");
```

```
INSERT INTO sub_categorias (id, nombre, categoriasId)
VALUES (1, " Programación c++",1), (2, " Programación Java", 1), (3, "Programación Python", 1),
       (4, "Guitarra", 2), (5, "Piano", 2), (6, "Flauta", 2),
       (7, "Catalán", 3), (8, "Inglés", 3), (9, "Francés", 3);
```

```
INSERT INTO ofertas (userId, categoriaId, subCatId, descripcion)
VALUES (1, 1, 1, "Clases de programación c++"),
    (1, 1, 2, "Clases de programación Java"),
    (1, 1, 3, "Clases de programación Python"),
    (1, 2, 4, "Clases de Guitarra"),
    (1, 2, 5, "Clases de Piano"),
    (1, 2, 6, "Clases de Flauta"),
    (1, 3, 7, "Clases de Catalán"),
    (1, 3, 8, "Clases de Inglés"),
    (1, 3, 9, "Clases de Frances"),
    (2, 1, 1, "Clases de programación c++"),
    (2, 1, 2, "Clases de programación Java"),
    (2, 1, 3, "Clases de programación Python"),
    (2, 2, 4, "Clases de Guitarra"),
    (2, 2, 5, "Clases de Piano"),
    (2, 2, 6, "Clases de Flauta"),
    (2, 3, 8, "Clases de Inglés"),
    (2, 3, 9, "Clases de Frances"),
    (3, 1, 1, "Clases de programación c++"),
    (3, 1, 3, "Clases de programación Python"),
    (3, 2, 4, "Clases de Guitarra"),
    (3, 2, 6, "Clases de Flauta"),
    (3, 3, 7, "Clases de Catalán"),
    (4, 1, 2, "Clases de programación Java"),
    (4, 2, 5, "Clases de Piano"),
    (4, 3, 7, "Clases de Catalán"),
    (5, 3, 9, "Clases de Frances");
```

7.3 Enlaces de interés

El proyecto el de código abierto y está disponible en:
https://github.com/nibanezp/bdt.git

También podemos acceder a la plataforma a través de:
https://bancodeltiempo2023.westeurope.cloudapp.azure.com/