

***I.E.S. La Vereda***

*La Poble de Vallbona - València*



# **E-Empresas:**

## **Sistema de Valoración de Prácticas Empresariales**



**Módulo proyecto**

**Ciclo:** 2º Desarrollo de Aplicaciones Web

**Autor:** Alejandro Lorenzo Toledo

**Tutor de proyecto:** José Miguel Fajardo Asensi

## Introducción

La idea de este proyecto viene dada del profesorado del departamento de informática del IES La Vereda desde la necesidad de gestionar la retroalimentación de las empresas sobre las prácticas de los alumnos, si aceptan nuevos alumnos, o cuál es la situación de la empresa sobre acoger alumnos.

En muchas ocasiones, resulta difícil saber si la empresa está satisfecha con la experiencia, si aceptaría nuevos estudiantes o si ha cambiado su situación y no puede acoger más alumnos.

Por tanto, **E-Empresas** es una **plataforma web orientada a mantener el contacto entre tutores para coordinarse en la búsqueda de empresas para las prácticas de los alumnos.**

El sistema permite a los administradores registrar empresas, gestionar solicitudes de alta/baja por parte de usuarios, asignar estudiantes a empresas, registrar comentarios y valoraciones, y hacer seguimiento del progreso.

Los estudiantes también tienen un papel activo en la plataforma, de manera que pueden expresar su opinión y dejar retroalimentación para reflejar su experiencia mediante comentarios, mientras los tutores validan y supervisan dichas interacciones.

Toda esta información será guardada en una base de datos dedicada en los servidores del centro, de manera que el proyecto perdure en el tiempo y en el ecosistema del IES La Vereda, además de que será accesible para todo el mundo conectado a la red del instituto.

# ÍNDICE DE CONTENIDOS

Introducción .....	2
Módulos implicados .....	4
Tecnologías utilizadas en el proyecto.....	5
Timeline o plan de trabajo .....	6
Documentación del Modelo de la Base de Datos .....	9
Documentación del Backend – Laravel .....	14
Documentación del Frontend - React .....	22
Manual de usuario.....	24
Instrucciones de instalación .....	30
Protección legal .....	33
Plan de rentabilidad .....	33
GitHub .....	34
Conclusiones .....	35
Anexo y webgrafía .....	36

## Módulos implicados

**Base de Datos** - Diseño de la BBDD. Desde el diseño en modelo entidad - relación hasta el uso de SQL durante el desarrollo.

- Javier García Blasco

**Programación** - La lógica de programación y la organización del código que se trabaja en esta asignatura ha sido clave para el desarrollo.

- Joaquín Vicente Alonso Saiz

**Entornos de desarrollo** – GitHub como herramienta principal de control de versiones de este proyecto.

- Roberto González Cardenete

**Lenguaje de Marcas y Diseño de Interfaces** – Todo el diseño y desarrollo de los estilos de la web vienen dados de este módulo.

- José Miguel Fajardo Asensi (Lenguaje de Marcas)
- Francisco Hernández Lara (Diseño de Interfaces)

**Cliente y Servidor** – Los principales encargados de dar vida a la aplicación, estos módulos han sido cruciales para el correcto funcionamiento de esta.

- Amador Gramage Borrás (Cliente)
- Javier García Blasco (Servidor)

**FOL y Empresa** – La intervención de este módulo con el proyecto desemboca en la monetización de este.

- Itziar de Oteiza Galdón

**Despliegue de Aplicaciones web** – Crucial para el despliegue de la aplicación en los servidores del centro.

- Borja Cleries Rodríguez

## Tecnologías utilizadas en el proyecto

Para el desarrollo del proyecto se han seleccionado una serie de tecnologías, las cuales todas han sido enseñadas y estudiadas en el centro, tanto lenguajes como frameworks.

### Frontend:

En el frontend se encuentra la experiencia del usuario que va a utilizar la aplicación, para ello se ha utilizado:

- **React** como framework principal
- **JavaScript** para las llamadas a la API y dinamismo de la página
- **HTML** para la representación de la información
- **SASS** como preprocesador para los estilos
- **CSS** como lenguaje en el que se basa sass

### Backend:

El backend se encarga de toda la lógica del servidor, gestión de datos y seguridad de la aplicación, donde se gestionan las llamadas del front y se devuelven los resultados deseados. Las tecnologías utilizadas son las siguientes:

- **Laravel** como framework principal
- **PHP** como lenguaje en el que se basa Laravel
- **Artisan** para lanzar el servidor
- **Middleware** para filtrar las peticiones HTTP
- **Routing** para definir las rutas de llamadas de la API
- **XAMPP** para desplegar el proyecto en local y poder desarrollar el proyecto

### Control de versiones

Para gestionar el control de versiones del proyecto se ha utilizado **Git**, facilitando mantener un historial completo de los cambios realizados, además de servir de alojamiento de repositorio remoto

## Timeline o plan de trabajo

### Primera semana

Durante la primera semana estuve alternando entre el desarrollo de la base de datos y la creación de los mockups de la web. Empecé por crear el diagrama ER para consultar con el tutor si los conceptos del proyecto han quedado claros.

	MARZO						
TAREAS	24	25	26	27	28	29	30
Desarrollo de la BD							
Mockups							

### Segunda semana

Durante la segunda semana empecé con la creación del SQL de la base de datos, incluyendo DDL y DML. Continuando también con mockups.

	ABRIL						
TAREAS	31	1	2	3	4	5	6
Desarrollo de la BD							
Mockups							
Creación proyecto Laravel							

### Tercera semana

Durante la tercera semana, el día 9 envié la base de datos al tutor para que se revisara en el departamento de informática y darme algo de feedback. También empecé con el proyecto en React, creando algunos componentes y rutas base.

	ABRIL						
TAREAS	7	8	9	10	11	12	13
Creación proyecto React							
Desarrollo proyecto Laravel							

### Cuarta semana

En esta cuarta semana, el día 17 recibí feedback sobre la base de datos del proyecto. Había entendido mal el enunciado del proyecto así que tocaba rehacer y retocar algunas de las tablas y el proyecto de Laravel.

	ABRIL						
TAREAS	14	15	16	17	18	19	20
Desarrollo de la BD							
Desarrollo proyecto Laravel							

### Quinta semana

Durante la quinta semana recibí el acceso al GitHub del instituto para poder subir los proyectos y continuar con el desarrollo, ahora con control de versiones. Los desarrollos de ambos proyectos los llevaba a cabo al mismo tiempo, implementando y testeando.

	ABRIL						
TAREAS	21	22	23	24	25	26	27
Subida a GitHub							
Desarrollo React							
Desarrollo Laravel							

### Sexta semana

En esta sexta semana continué un poco con el desarrollo de cada proyecto y el día 2 recibí el Excel con los datos de empresas y tutores del SAÓ para importar directamente y empecé a jugar un poco con cómo se pueden insertar estos datos.

	ABRIL						
TAREAS	28	29	30	1	2	3	4
Excel							
Desarrollo React							
Desarrollo Laravel							

### Séptima semana

En esta séptima semana me dediqué a buscar información sobre cómo implementar la inserción de datos a través de un archivo .xls. Debo reconocer que ha sido un proceso lleno de estrés, pero gratificante al mismo tiempo.

	MAYO						
TAREAS	5	6	7	8	9	10	11
Excel							

### Octava semana

Durante esta semana el proyecto ya empezó a tomar forma, con la importación de tutores y empresas a través de Excel finalizada, comencé con el responsive del proyecto.

	MAYO						
TAREAS	12	13	14	15	16	17	18
Excel							
Desarrollo React							
Responsive							
Desarrollo Laravel							

### Novena semana

Durante esta semana, con todas las funcionalidades principales terminadas, solo queda asegurarse de que cada funcionalidad está restringida a un rol o restringe otros roles.

	MAYO						
TAREAS	19	20	21	22	23	24	25
Funcionalidad de roles							



## Documentación del Modelo de la Base de Datos

A continuación, comentaré el modelo que va a seguir la base de datos, explicando cada tabla, una pequeña descripción y los campos que la componen. Finalmente, un enlace al diagrama ER para verlo representado gráficamente que también estará adjunto en el anexo.

### 1. comments

Almacena la retroalimentación que dejan los tutores sobre las empresas a lo largo del tiempo.

Campos:

**id (PK, int(11))**: Identificador único de cada comentario.

**id\_company (int(11))**: Clave foránea a la empresa (companies.id).

**id\_tutor (int(11))**: Clave foránea al tutor autor del comentario (users.id).

**comment (varchar(300))**: Texto del comentario.

### 2. companies

Contiene los datos de las empresas registradas en el programa de prácticas.

Campos:

**id (PK, int(11))**: Identificador único de la empresa.

**name (varchar(255))**: Nombre de la empresa.

**manager (varchar(255))**: Persona responsable.

**phone (varchar(50)), email (varchar(255))**: Datos de contacto.

**address (varchar(255)), website (varchar(255))**: Ubicación y web.

**is\_private (tinyint(1))**: Indica si la entidad es pública o privada.

**allows\_erasmus (tinyint(1))**: Indica si acepta estudiantes Erasmus.

### 3. *company\_deregister\_request*

*La funcionalidad de esta tabla no está reflejada aún en el proyecto.*

Registra las solicitudes de baja de empresas en la plataforma.

Campos:

**id (PK, int(11))**: Identificador de la solicitud.

**id\_company (int(11))**: Clave foránea a la empresa solicitada (companies.id).

**id\_requestor (int(11))**: Clave foránea al usuario que pide la baja (users.id).

**reason (text)**: Motivo de la solicitud.

### 4. *company\_register\_request*

*La funcionalidad de esta tabla no está reflejada aún en el proyecto.*

Recoge las solicitudes de alta de nuevas empresas en la plataforma.

Campos:

**id (PK, int(11))**: Identificador de la solicitud.

**id\_company (int(11))**: Clave foránea a la empresa sugerida (companies.id).

**id\_requestor (int(11))**: Clave foránea al usuario que pide el alta (users.id).

**name (varchar(255))**: Nombre de la empresa.

**manager (varchar(255))**: Persona responsable.

**phone (varchar(50)), email (varchar(255))**: Datos de contacto.

**address (varchar(255)), website (varchar(255))**: Ubicación y web.

**is\_private (tinyint(1))**: Indica si la entidad es pública o privada.

**allows\_erasmus (tinyint(1))**: Indica si acepta estudiantes Erasmus.

### 5. *company\_reviews*

Esta tabla almacena las reseñas de los alumnos hacia las empresas.

Campos:

**id (PK, int(11))**: Identificador de la reseña.

**id\_company (int(11))**: Clave foránea a la empresa reseñada (companies.id).

**id\_student (int(11))**: Clave foránea del estudiante que comenta (companies.id).

**comment (text)**: Cuerpo de la reseña.

**rating (tinyint(1))**: Valoración numérica de la reseña.

**would\_recommend(tinyint(1))**: Si el usuario recomienda la empresa.

**approved(tinyint(1))**: Indica si el comentario ha sido aprobado por admin o tutor.

### 6. *notices*

Esta tabla recoge las tareas que los tutores se asignan entre sí.

**id (int(11))**: identificador único de cada tarea.

**id\_user\_emitter (int(11))**: Identificador del usuario que envía la tarea.

**id\_user\_receiver (int(11))**: Identificador del usuario que recibe la tarea.

**text (text)**: Cuerpo de la tarea.

**seen (tinyint(1))**: Indica si la tarea se ha visto o no.

### 7. *users*

Almacena los usuarios del sistema: estudiantes, tutores y administradores.

Campos:

**id (PK, int)**: Identificador único de usuario.

**name, email, password (varchar(255))**: Datos de acceso y contacto.

**is\_admin, is\_student, is\_tutor (tinyint(1))**: Roles de usuario.

**nia (int), nif (varchar(20))**: Identificadores académicos y fiscales.

**gender (varchar(20))**: Género del usuario.

### **Relaciones entre tablas**

**users.id\_company - companies.id** (1 empresa puede tener muchos usuarios asignados)

**company\_reviews.id\_company - companies.id** (1 empresa puede tener muchas valoraciones)

**company\_reviews.id\_student - users.id** (1 estudiante puede dejar muchas valoraciones, solo 1 por empresa)

**comments.id\_company - companies.id** (1 empresa puede tener muchos comentarios de tutores)

**comments.id\_tutor - users.id** (1 tutor puede dejar muchos comentarios)

**notices.id\_user\_emitter - users.id** (1 usuario puede enviar muchas tareas)

**notices.id\_user\_receiver - users.id** (1 usuario puede recibir muchas tareas)

**company\_register\_request.id\_requestor - users.id** (1 usuario puede solicitar registrar muchas empresas)

**company\_deregister\_request.id\_company - companies.id** (1 empresa puede recibir varias solicitudes de baja)

**company\_deregister\_request.id\_requestor - users.id** (1 usuario puede solicitar la baja de varias empresas)

### **DDL**

El documento DDL recoge la relación de una base de datos definiendo su estructura y las características de sus elementos según el modelo relacional.

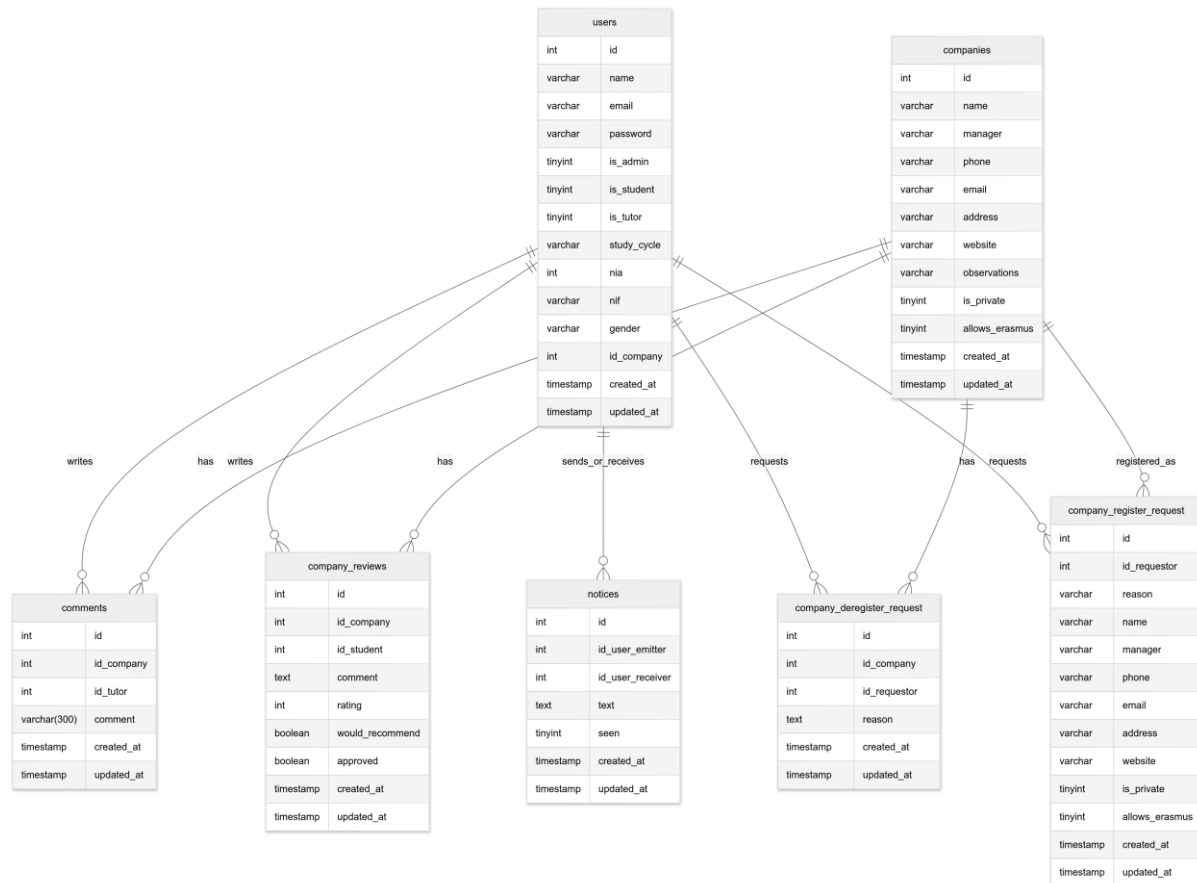
El archivo .sql se encontrará adjuntado en la misma carpeta que este documento en GitHub.

### **DML**

El archivo .sql adjunto a la carpeta del proyecto incluye algunos datos de prueba junto al DDL. Estas inserciones se pueden utilizar para poder utilizar la aplicación sin tener que agregar algún usuario a través de código, de manera que tampoco incluye la información personal privada de los tutores o las empresas, que deberán ser importados en la misma aplicación desde el menú de importación por Excel.

## Modelo ER

El modelo entidad-relación representa el diseño de la base datos, sus tablas y sus relaciones.



Haz clic en este [enlace](#) para ver el diagrama en una nueva ventana.

## Documentación del Backend – Laravel

Para este proyecto, usaremos Laravel para crear una API a la que se le harán llamadas desde React.

### Modelos

Los modelos son clases que representan una tabla de la base de datos y permite acceder sus registros de una forma sencilla sin hacer uso de SQL manualmente. Cada modelo estará vinculado a una tabla de la base de datos y, por tanto, tendrán el mismo nombre de la tabla a la que hacen referencia.

Por tanto, tenemos:

- Comment.php: Observaciones de los tutores:
  - o tutor() - Clave foránea de tutor
  - o company() - clave foránea de compañía
- Company.php: Compañías registradas:
  - o reviews() - recoge todas las reviews en las que aparece
  - o tutorComments() - recoge todas las observaciones de la compañía
- CompanyDeregisterRequest.php: Solicitudes de baja de empresa (funcionalidad aún no implementada):
  - o company() - Clave foránea de la empresa
  - o requestor() - El usuario que solicita la baja
- CompanyRegisterRequest.php: Solicitudes de alta de empresa (funcionalidad aún no implementada):
  - o company() - Clave foránea de la empresa
  - o requestor() - El usuario que solicita la baja
- CompanyReview.php: Reseñas de estudiante usuarios:
  - o company() - Clave foránea de la empresa
  - o student() - Clave foránea del estudiante
- Notice.php: Tareas entre tutores:
  - o emitter() - tutor emisor
  - o receiver() - tutor receptor
- User.php: Usuarios registrados:
  - o comments() - Recoge todas las observaciones del usuario

- reviews() - recoge todas las reseñas del usuario
- noticesSent() - tareas enviadas
- noticesReceived() - tareas recibidas

## Controladores

Los controladores son clases que organizan la lógica para manejar las solicitudes HTTP, actuando como intermediario entre las rutas y los modelos.

A continuación, listaré todos los controladores del proyecto junto a las funciones que implementan, algunas por defecto de Laravel y otras propias.

Tenemos:

- CommentController.php: gestión de las observaciones de tutor.
- CompanyController.php: gestión de las empresas.
- CompanyDeregisterRequest: gestión de las solicitudes de baja.
- CompanyRegisterRequest: gestión de las solicitudes de alta.
- CompanyImportController: importación de empresas por excel.

```
class CompanyImportController extends Controller
{
    public function import(Request $request)
    {
        $request->validate([
            'file' => 'required|mimes:xlsx',
        ]);

        logger('Inicio de importación de empresas');
        Excel::import(new CompaniesImport, $request->file('file'));
        logger('Importación completada');

        return response()->json(['message' => 'Empresas importadas correctamente']);
    }
}
```

En la imagen adjunta se muestra la clase de importación por Excel de empresas. El validate verifica que el archivo es .xls. Luego creo un log para indicar el inicio de la importación. Con la función import de Excel, creamos un nuevo import con el archivo. Debe de haberse creado una clase para el import de empresas, como se muestra en la imagen (new CompaniesImport).

- CompanyReviewController.php: gestiona las reseñas de usuarios.

```
public function allReviews(Request $request)
{
    $reviews = CompanyReview::with(['company', 'user'])
        ->where('approved', 0)
        ->orderByDesc('created_at')
        ->paginate(10);

    return response()->json($reviews);
}

public function approve($id)
{
    $review = CompanyReview::findOrFail($id);
    $review->approved = true;
    $review->save();

    return response()->json($review);
}
```

La función `allReviews` busca todas las reseñas de la empresa, que estén aprobadas, las ordena por más reciente y las pagina de 10 en 10.

La función `approve` simplemente aprueba la reseña.

- `NoticeController.php`: controlador de las tareas.
- `TutorsImprotController.php`: similar al import de empresas, pero para tutores.
- `UserController.php`: gestión de usuarios.



## Imports

Los imports los vamos a utilizar para poder inyectar datos de Excel a la base de datos.

### CompaniesImport.php:

```
class CompaniesImport implements ToCollection
{
    public function collection(Collection $rows)
    {
        $rows = $rows->skip(2);

        $groups = $rows->groupBy(function ($row) {
            return mb_strtolower(trim($row[14] ?? ''));
        });

        Log::info("Grupos detectados: " . $groups->count());

        foreach ($groups as $key => $group) {
            $firstIndex = $group->keys()->first() + 3;

            $name = trim($group->first()[14] ?? '');
            if (!$name) {
                Log::warning("Fila $firstIndex: nombre vacío, se ignora grupo.");
                continue;
            }
        }
    }
}
```

En esta primera parte del código lo primero que se hace es saltar dos filas, ya que no contienen información relevante. La variable groups agrupa todos los datos en los que coincida el nombre de la empresa (fila 14 del excel), ya que aparecen varias veces.

Luego, por cada grupo (empresa) se verifica si el campo de nombre está vacío (columna 14 del excel). Si es así lo ignora. El firstIndex se refiere a la posición de la fila de la empresa en el excel. Se le suma 3 porque en excel se empieza a contar en 1 y 2 por las filas saltadas al principio.

```
$firstRow = $group->first();
$data = [
    'manager'      => trim(($firstRow[23] ?? '') . ' ' . ($firstRow[24] ?? '') . ' ' . ($firstRow[25] ?? '')),
    'phone'         => trim($firstRow[18] ?? ''),
    'email'         => filter_var(trim($firstRow[19] ?? ''), FILTER_VALIDATE_EMAIL) ?: null,
    'address'       => trim($firstRow[15] ?? ''),
    'website'       => '',
    'observations'  => '',
    'allows_erasmus' => strtoupper(trim($firstRow[49] ?? '')) === 'N' ? 0 : 1,
    'is_private'    => is_numeric($firstRow[20] ?? null) ? (int) $firstRow[20] : 0,
];

Log::info("Fila $firstIndex: importando empresa \"$name\", $data");

Company::updateOrCreate(
    ['name' => $name],
    $data
);

Log::info("Importación completada.");
}
```

En esta segunda parte del código, teniendo la posición de la fila de excel (firstRow) creamos el \$data en el que iremos recogiendo todos los datos de la empresa para guardarlos en la base de datos. Una vez terminado lanzamos un Log indicando que se va a importar y finalmente creamos la empresa o actualizamos si ya existe. Recalco que el campo clave de una empresa es el nombre.

## TutorsImport.php

```

class TutorsImport implements ToCollection
{
    public function collection(Collection $rows)
    {
        $rows = $rows->skip(2);

        $groups = $rows->groupBy(fn($row) => trim($row[27] ?? ''));

        Log::info("Tutores únicos detectados: " . count($groups));

        foreach ($groups as $key => $group) {
            $firstIndex = $group->keys()->first() + 3;
            $nifTutor = trim($group->first()[27] ?? '');

            if (!$nifTutor) {
                Log::warning("Fila $firstIndex: NIF de tutor vacío, se ignora grupo.");
                continue;
            }

            $row = $group->first();
            $name = trim(($row[28] ?? '') . ' ' . ($row[29] ?? '') . ' ' . ($row[30] ?? ''));
            $phone = trim($row[31] ?? '');
            $email = filter_var(trim($row[32] ?? ''), FILTER_VALIDATE_EMAIL) ?: null;
            $department = trim($row[33] ?? '');

            $data = [
                'name'      => $name ?: 'Desconocido',
                'nif'       => $nifTutor,
                'email'     => $email ?: strtolower(str_replace(' ', '', $name)) . '@default.com',
                'study_cycle' => $department ?: 'No asignado',
                'password'  => bcrypt($phone ?: '1234'),
                'is_tutor'  => true,
            ];

            Log::info("Fila $firstIndex: importando tutor NIF={$nifTutor}", $data);

            User::updateOrCreate(
                ['nif' => $nifTutor],
                $data
            );
        }

        Log::info("Importación de tutores completada.");
    }
}

```

Similar al de empresas, pero selecciona la columna 27, que es donde empiezan los tutores. Como apunte, si el correo no se incluye en el excel, se creará un correo con formato: "nombrecompletodeltutor@default.com". La contraseña será el número de teléfono que aparezca en el excel, de no tener, por defecto será 1234.

## Rutas

Como el proyecto va a funcionar como API, usaremos el archivo api.php para definir las rutas a las que se le harán peticiones desde el front.

Las rutas son las siguientes:

```
Route::apiResource('comments', CommentController::class);
Route::apiResource('companies', CompanyController::class);
Route::apiResource('company-deregister-requests', CompanyDeregisterRequestController::class);
Route::apiResource('company-register-requests', CompanyRegisterRequestController::class);
Route::apiResource('company-reviews', CompanyReviewController::class);
Route::apiResource('notices', NoticeController::class);
Route::apiResource('users', UserController::class);
```

La ruta /login:

```
Route::post('/login', function (Request $request) {
    $user = User::where('email', $request->email)->first();

    if (!$user || !Hash::check($request->password, $user->password)) {
        return response()->json(['message' => 'Credenciales inválidas'], 401);
    }

    $token = $user->createToken('api-token')->plainTextToken;
    return response()->json([
        'token' => $token,
        'user' => [
            'id' => $user->id,
            'email' => $user->email,
            'name' => $user->name,
        ]
    ]);
});
```

La ruta /login comprueba que el usuario existe y que su contraseña esté codificada. Si es así, crea el token.

Las rutas protegidas:

```
Route::middleware('auth:sanctum')->group(function () {

    Route::post('/logout', function (Request $request) {
        $request->user()?->currentAccessToken()->delete();
        return response()->json(['message' => 'Sesión cerrada correctamente']);
    });

    Route::post('/tutors/import', [TutorsImportController::class, 'import']);
    Route::post('/companies/import', [CompanyImportController::class, 'import']);

    Route::get('/user', fn(Request $request) => $request->user());

    Route::get('/notices', [NoticeController::class, 'index']);
    Route::post('/notices', [NoticeController::class, 'store']);

    Route::put('/tutor-comments/{comment}', [CommentController::class, 'update']);
    Route::delete('/tutor-comments/{comment}', [CommentController::class, 'destroy']);
});
```

A estas rutas solo se puede acceder si tienes token. El front gestiona los roles.

Rutas públicas:

```
Route::get('/companies/{company}/reviews', [CompanyReviewController::class, 'index']);
Route::post('/companies/{company}/reviews', [CompanyReviewController::class, 'store']);
Route::delete('/companies/{company}/reviews/{review}', [CompanyReviewController::class, 'destroy']);
Route::get('/reviews', [CompanyReviewController::class, 'allReviews']);
Route::put('/reviews/{id}', [CompanyReviewController::class, 'approve']);

Route::get('/companies/{company}/tutor-comments', [CommentController::class, 'index']);
Route::post('/companies/{company}/tutor-comments', [CommentController::class, 'store']);
```

Se debe configurar el archivo .env de la siguiente manera:

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=e_empresas
DB_USERNAME=root
DB_PASSWORD=
```

## Documentación del Frontend - React

Para la creación del entorno cliente se usará React, que además de haber sido el framework estudiado en el centro, aporta dinamismo a las páginas web.

### Rutas

La web mostrará según qué componentes dependiendo de la ruta a la que se redirija y de los roles asociados.

Para proteger las rutas se hace uso del componente `ProtectedRoute.jsx` y se usa así:

```
<Route
  path="/profile"
  element={
    <ProtectedRoute
      requiredRoles={["admin", "tutor"]}
      element={<Profile />}
    />
  }
/>
```

Recibe por parámetro un array de roles permitidos y el elemento a mostrar.

### Componentes

Haremos usos de los componentes para evitar la redundancia de código.

- **ImportCompanies y ImportTutors:** Componentes para importar el archivo xls
- **Logout:** Botón para cerrar sesión.
- **Navbar:** Barra de navegación con enlaces a todas las rutas dependiendo del rol de usuario.
- **ProtectedRoute:** Protección de rutas por rol.
- **ScrollToTopButton:** Botón para volver al principio de la página.
- **StarRating:** Muestra con estrellas un número sobre 5, usada para la puntuación de la empresa.

## Páginas

Las páginas son mostradas al ser llamadas en las rutas. Cada página hace uso de los componentes según conveniencia.

- **CompaniesList:** Muestra un listado con todas las empresas con una barra de búsqueda.
- **CompanyDetail:** Página de detalle de empresa que incluye:
  - Comentarios de alumnos y gestión de estos.
  - Observaciones de tutores y gestión de estos.
  - Datos de la empresa.
- **CompanyManager:** Menú de gestión de empresas para los administradores, se usa el componente de importación de empresa.
- **Home:** Una página de inicio con algunos enlaces de la página.
- **LoginPage:** Página de inicio de sesión.
- **Profile:** Zona personal de tutores y administradores en la que se pueden poner tareas entre ellos y aprobar solicitudes de reseña para los alumnos, asegurando comentarios constructivos y no ofensivos.
- **UserManager:** Gestión de usuarios para administradores. Usa el componente de ImportTutors.

También se ha hecho uso de algunas **librerías** de React para llevar a cabo el proyecto:

- **Axios** para la gestión de llamadas a la API
- **Bootstrap** como framework de estilos
- **React-router** para crear rutas y asociarlas a componentes
- **SASS** como preprocesador de estilos

# Manual de usuario

## ¿Qué es E-Empresas?

E-Empresas es una plataforma web de gestión de prácticas empresariales para facilitar la coordinación entre tutores, la asignación de alumnos a empresas y el seguimiento de su experiencia durante las prácticas.

## Funcionamiento general de la aplicación

Primeramente, se debe estar previamente registrado por un administrador para poder acceder a la aplicación en su totalidad.

Además, el sistema tendrá distintas funcionalidades dependiendo del rol que tenga el usuario activo.

La aplicación permite registrar empresas para que se lleve un registro de las observaciones que tienen los tutores con respecto a las prácticas de los alumnos.

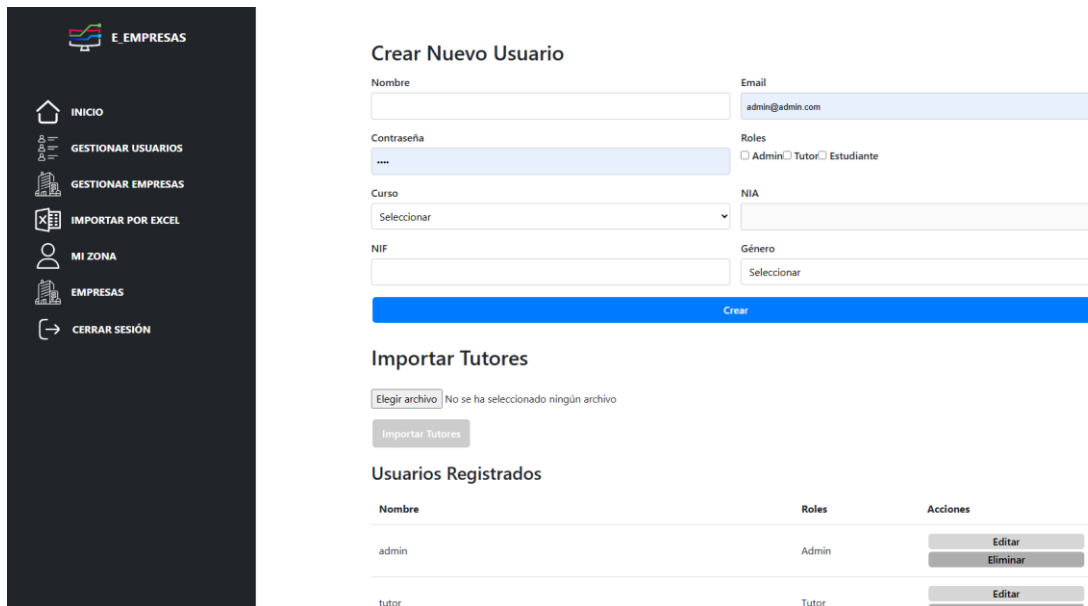
## Primer contacto



Lo primero que veremos tras iniciar sesión será la página de inicio, con algunos enlaces de lo que puedes hacer en la web. La barra de navegación también incluye los enlaces a los que se puede acceder según tus permisos.



## Gestión de datos



**Crear Nuevo Usuario**

Nombre:

Email:

Contraseña:

Roles: ☐ Admin ☐ Tutor ☐ Estudiante

Curso:

NIA:

NIF:

Género:

**Importar Tutores**

No se ha seleccionado ningún archivo

**Usuarios Registrados**

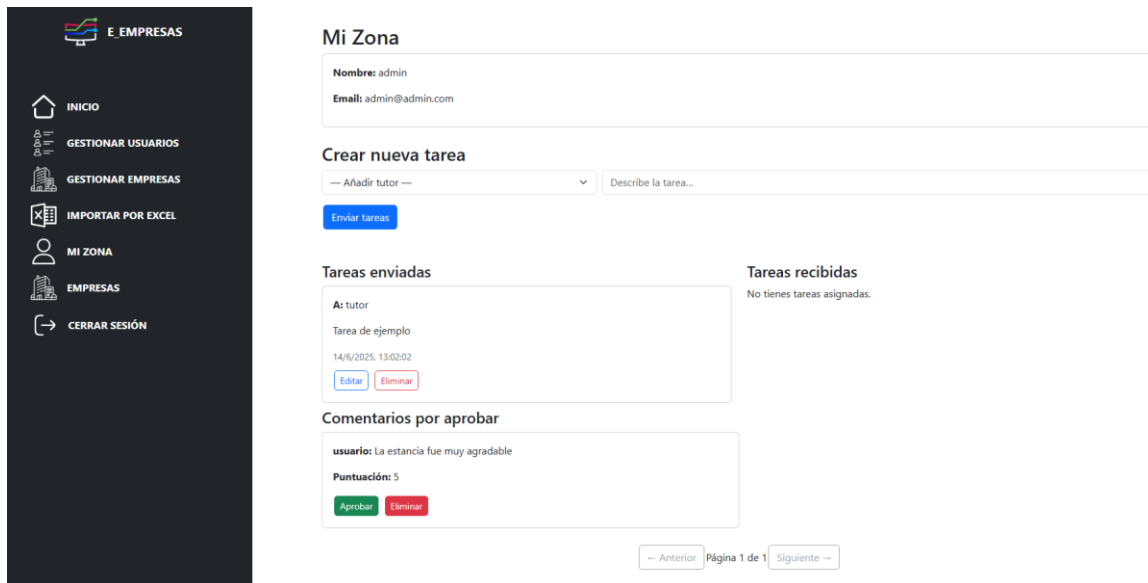
Nombre	Roles	Acciones
admin	Admin	<input type="button" value="Editar"/> <input type="button" value="Eliminar"/>
tutor	Tutor	<input type="button" value="Editar"/>

Este es el menú de gestión de usuarios, donde en la sección de creación aparecerán campos para rellenar y registrar. Al hacer clic al editar el formulario de creación pasa a ser de edición y se rellena con los campos del usuario seleccionado.

Justo debajo aparece el componente de inserción por excel, el cual también es accesible desde la página dedicada (importar por excel en la barra de navegación).

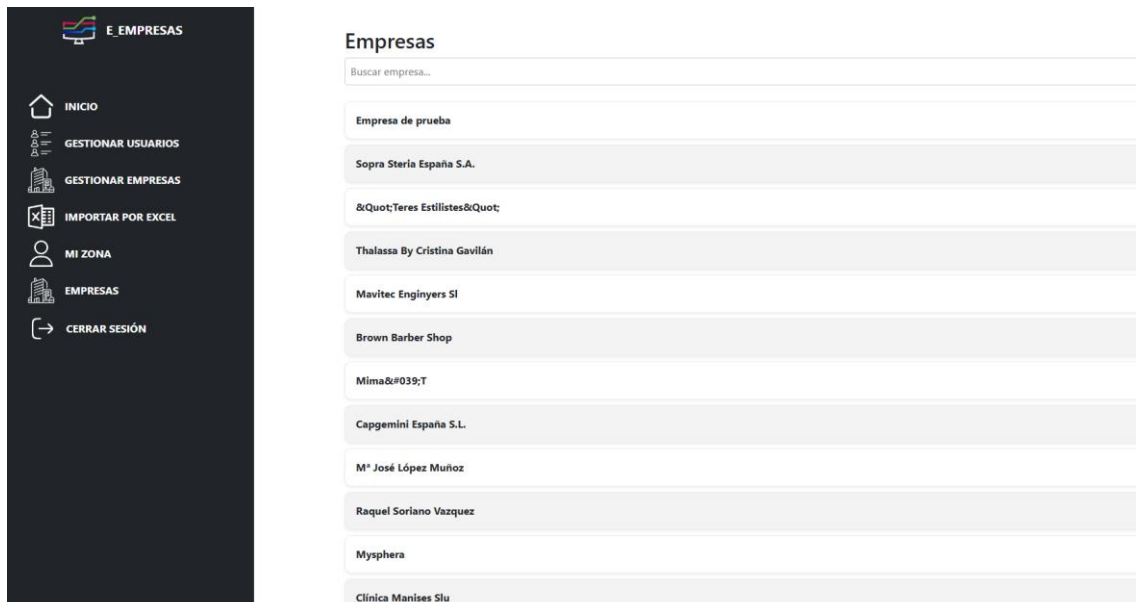
La página de gestión de empresas es igual que la de usuario, pero con los datos de empresa.

## Espacio personal



Esto es la interfaz de la zona personal de administradores y tutores. Pueden crear tareas a otros usuarios y ver las tareas enviadas y recibidas. Además, también pueden aprobar o eliminar las reseñas de empresa hechas por los alumnos.

## Listado de empresas



En el apartado de empresas se muestra una lista de todas las empresas registradas que al hacer clic sobre una de ellas te llevará a la página de detalle.

## Vista de detalle de empresa

### Empresa de prueba

#### Datos de la empresa:

**Jefe:** Manager  
**Teléfono:** 123456789  
**Correo:** eprueba@gmail.com  
**Privada:** NO

#### Comentarios de tutores

?

**Publicar**

La empresa acepta alumnos

14/6/2025

Editar

Eliminar

#### Comentarios

No hay comentarios aún.

Esto es lo que ve un tutor, un formulario para crear observaciones de la empresa y ver los datos de esta.

A continuación, la vista de estudiante:

## Empresa de prueba

Puntuación media: Aún no hay puntuaciones

### Añade un comentario

¿Cómo ha sido tu experiencia?

Puntuación general:

Muy malo

☒ Lo recomendaría

Enviar

### Comentarios

Comentario de prueba

14/6/2025

★★★★★

Recomendaría: Sí

Eliminar comentario

Como aparece en la imagen habrá justo debajo del nombre de la empresa su calificación, pero esta no será actualizada hasta que el comentario se apruebe. Mostrará una media sobre cinco de todas las reseñas de usuario.

Debajo hay un formulario para las reseñas y un listado de comentarios ordenados por el más reciente y el comentario del usuario el primero, con opción de borrado. En caso de ser administrador se podrá borrar cualquier comentario.

## Diferenciación de roles y sus acciones

En esta sección comentaré los roles que intervienen en las acciones que pueden realizar los usuarios.

### Alumno:

Vista general de las empresas registradas con sus valoraciones de otros alumnos y la reseña de su estancia en la empresa durante el contrato de prácticas.

### Profesor:

Acceso a importación de usuarios y empresas por Excel (archivo extraído del SAÓ) y a su espacio personal, donde podrá asignar tareas a otros tutores o administradores.

En la vista de empresa no podrá añadir reseñas, pero sí observaciones de la empresa (solo visible para tutores)

### **Administrador:**

Acceso a la interfaz CRUD de usuarios y empresas y acceso al espacio personal, igual que los tutores.

### **Principales funciones por página**

A continuación, un listado de todas las páginas disponibles en el entorno cliente junto a una descripción de su contenido.

#### **Página de inicio** *Accesible para todos los usuarios.*

Se muestra un pequeño grupo de enlaces para acceder a las diferentes páginas de la web.

#### **Menú de gestión de usuarios** *Accesible para administradores*

Aparece un bloque para crear usuarios, que al editar un usuario este rellenará sus campos y cambiará a estado de edición. También aparece la importación de los usuarios a través del Excel del saó. Finalmente, un listado con todos los usuarios con sus roles y botones para editar o eliminar.

#### **Menú de gestión de empresas** *Accesible para administradores*

Exactamente igual que el de usuarios, pero relacionado con las empresas.

#### **Importación por Excel** *Accesible para administradores y tutores*

Una página en la que aparecen dos bloques en los que se puede subir el archivo **(solo en formato .xls)** para importar tutores y empresas.

#### **Mi zona** *Accesible para administradores y tutores*

Un espacio personal en el que se puede asignar otras tareas a otros tutores y administradores de la aplicación, marcarlas como leído o editarlas. Como añadido, se puede validar los comentarios hechos por alumnos, para evitar comentarios inadecuados.

#### **Empresas** *Accesible para todos los usuarios*

Se muestra un listado con buscador de todas las empresas registradas en la aplicación. Al hacer clic sobre una de ellas aparecerá un menú distinto según el tipo de usuario:

Estudiantes: Añadir comentarios a modo de reseña sobre su estancia en las prácticas.

Tutores: Añadir observaciones de la empresa a modo de anotaciones sobre la situación de la empresa en la aceptación de alumnos de prácticas. Las observaciones pueden ser sobre cualquier tema que considere el usuario.

Administradores: Controlar los comentarios subidos por los alumnos.

## Instrucciones de instalación

E-Empresas corre bajo un proyecto de Laravel como backend y React para frontend. Para instalarlo, es necesario que el back esté desplegado (en una máquina o en un contenedor), que esté enlazado a la base de datos (archivo adjunto en GitHub), y que el front esté también desplegado, puede estar desplegado en el mismo contenedor que el back, pero es recomendable al menos separar front y back.

Una vez desplegados, habrá que ajustar las direcciones de las llamadas y de CORS. Para el front, hay que cambiar el proxy del archivo package.json a la url destino del back, y en el back, dentro de la carpeta config, en el archivo de cors.php, editar la línea de allowed origins y añadir la url final del front.

### Instalación en un único sistema operativo

En este apartado se explicará como desplegar el proyecto para usarlo en local o en una misma red. Puede estar desplegado con el front en el sistema anfitrión con una máquina virtual con el back o todo junto en el mismo sistema.

#### Requisitos previos:

- Tener instalado PHP (v8.x), Composer y Node.js
- Tener XAMPP para MySQL y Apache
- Tener instalado Git

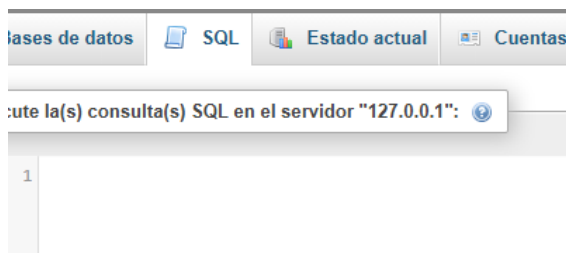
#### Preparar la base de datos:

En el panel de control de XAMPP tendremos que iniciar Apache y MySQL

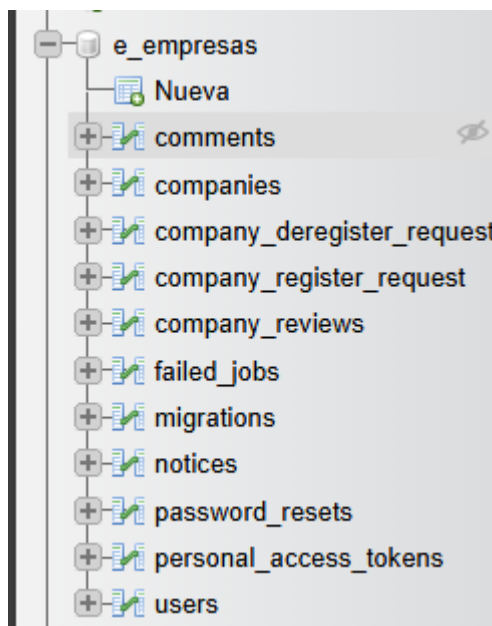
Modules				
Service	Module	PID(s)	Port(s)	Actions
<input type="checkbox"/>	Apache	1832 18516	80, 443	Stop
<input type="checkbox"/>	MySQL	8788	3306	Stop

Una vez dentro, acceder a la URL - [localhost / 127.0.0.1 | phpMyAdmin 5.2.1](http://localhost/127.0.0.1/phpMyAdmin/5.2.1)

Se nos mostrará la página de phpMyAdmin, y accederemos al apartado de SQL para pegar el código de la base de datos.



Una vez pegado, clic en continuar y ya tendríamos la base de datos importada.



### Clonar repositorio:

```
git clone https://github.com/ieslavereda-projects/24_25_Alejandro_Lorenzo_DAW_AppEmpreses_IESLaVereda
cd .\24_25_Alejandro_Lorenzo_DAW_AppEmpreses_IESLaVereda
```

### Instalación del backend en Laravel

```
>> cd e_empresas_api
>> composer install
>> cp .env.example .env
>> php artisan key:generate
>> php artisan migrate --seed
```

Habr  que configurar el .env como la captura adjunta en la secci n de la documentaci n del Backend.

### Ejecuci n del proyecto:

Lo primero de todo es haber iniciado el servidor de XAMPP, si no es as  no funcionara. Luego ejecutamos

```
php artisan serve
```

```
INFO Server running on [http://127.0.0.1:8000].
Press Ctrl+C to stop the server
```

### Instalaci n del frontend en React:

```
cd .\e_empresas_front\
npm install
```

Comprobar la URL del proxy en package.json, debe ser la direcci n donde corra el backend:

```
"proxy": "http://localhost:8000",
```

Lanzamos el proyecto:

```
npm start
```

```
Compiled successfully!

You can now view e_empresas_front in the browser.

Local:            http://localhost:3000
On Your Network:  http://192.168.1.65:3000
```

Ya tenemos front y back desplegados en local. Con ambos servidores en ejecuci n, se puede acceder a la aplicaci n desde un navegador. Si se desea acceder desde otros dispositivos en la misma red, basta con sustituir localhost por la IP local de la m quina donde se ejecuta el servidor.



## Protección legal

Si queremos escalar el proyecto a un servicio, debemos tener en cuenta los siguientes puntos:

- Nombre comercial y dominio: definir un nombre a la aplicación para ser reconocida y tener un dominio y registrarlos.
- Derechos de autor: todo el código, diseño y documentación del proyecto deberían estar protegidos por derechos de autor desde el momento en que se cree.

## Plan de rentabilidad

### Monetización

Para sacar rentabilidad económica del proyecto, he ideado los siguientes planes:

Plan	Usuarios	Funcionalidades	Precio anual (€)
<b>Starter</b>	Hasta 20	Registro manual, visualización de empresas, sin importación de datos	190€
<b>Pro</b>	Hasta 100	Importación de datos, estadísticas básicas y gestión avanzada de empresas	490€
<b>Advanced</b>	Hasta 500	Roles avanzados y soporte de correos	990€
<b>Enterprise</b>	Sin límite	API, informes personalizados, integración con sistemas externos (calendario)	Desde 1.990€

Incluso algunos extras opcionales:

- 1€ extra por usuario adicional
- Informe PDF personalizado: 10€ por descarga
- Soporte técnico personalizado: +20€/mes

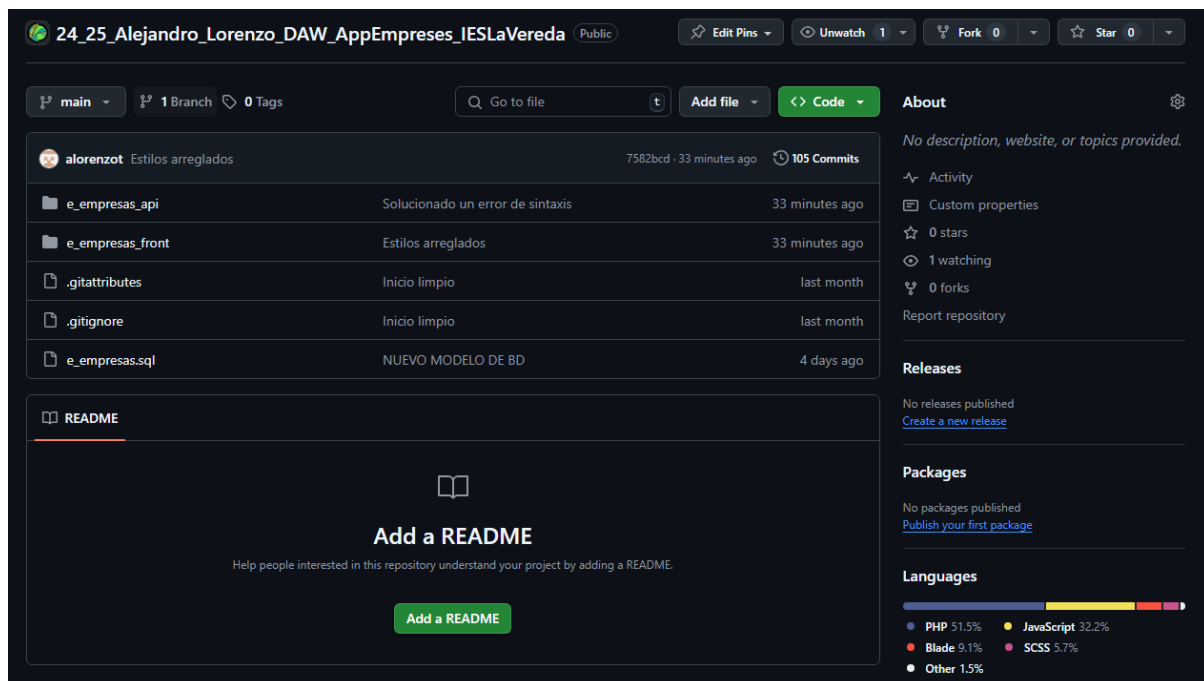
Otra forma de monetización sería a través de anuncios, ofreciendo a las empresas que paguen por que su marca sea visible en la web y destacar su perfil.

## Promoción

Para asegurar que la plataforma alcance a sus usuarios objetivo y tenga éxito, habrá que hacer uso de estrategias de promoción efectivas:

- Redes Sociales: Creando perfiles activos de LinkedIn, Instagram y Facebook, en los que se comparta contenido relevante, novedades y casos de éxito
- Marketing de contenido: Publicando artículos relacionados con la gestión educativa y empresarial.

## GitHub



En GitHub se encuentran las carpetas con el entorno cliente y el entorno servidor de la aplicación, además de la base de datos, la presentación y la documentación.

## Conclusiones

### Grado de consecución de los objetivos

He quedado bastante satisfecho con el resultado del proyecto final, sobre todo el hecho de combinar todas las asignaturas en una y reafirmar todo lo aprendido durante el curso.

En cuanto a la totalidad de lo que me he propuesto hacer, por problemas de tiempo no he podido conseguirlo.

No obstante, he quedado muy contento con los estilos de la página, que además de ser responsive, encuentro que la disposición de los elementos es bastante correcta y los colores no molestan a la vista. Un estilo sencillo que aporta claridad.

### Problemas encontrados

Realmente no he tenido ningún problema a la hora de desarrollar la aplicación. No obstante, toda la investigación de la importación a través de Excel me ha consumido mucho tiempo y ha resultado en algunas carencias en la aplicación. Como:

- No se puede exportar a Excel
- Solo admins pueden agregar usuarios
- Relacionar alumnos con empresas

### Mejoras

A pesar del gran trabajo realizado, creo que aún se pueden implementar más funcionalidades, y que algunas de ellas tienen las tablas preparadas, pero no están implementadas:

- Los tutores agreguen alumnos y los asocien a empresas
- Función de mensajes entre tutores
- Solicitudes de alta/baja de empresas (tablas incluidas)
- Solicitudes de alta/baja de usuarios
- Sistema de envío de correos y recordatorios
- Recuperación de contraseña
- Historial de cambios
- Archivado en vez de eliminación de datos

## Anexo y webgrafía

- Proyecto en GitHub:  
[24 25 Alejandro Lorenzo DAW AppEmpresas IESLaVereda](#)
- [Mermaid](#): Generación del Diagrama ER
- [Enlace al Diagrama ER](#)
- [Importación de datos de Excel](#)
- [Bootstrap](#)
- [React](#)
- [Laravel](#)
- [Composer](#)
- [XAMPP](#)