Gestión De Prácticas

Arquitectura C/S y Diseño Web - Grupo 15

Miembros del equipo:

- Carlos López
- Carlos Cercos
- Gonzalo López
- Nicolae-Alexandru Molnar
- Rubén Minero

Gestión de Prácticas

Análisis del problema

Se nos solicita la creación de un sistema de asignación de prácticas para la Universidad de Alcalá de Henares. Para la realización de este proyecto se va a desarrollar un diseño Modelo-Vista-Controlador que sea capaz de seguir las indicaciones realizadas por la universidad.

La universidad nos ha informado de que serán necesarios 3 tipos de usuarios siendo estos el alumno, aquel que será capaz de hacer una solicitud de prácticas y a quien se le asignará la práctica en cuestión en función de sus preferencias, el responsable de prácticas, el encargado de las gestiones de las prácticas de parte de la empresa, y, por último, el tutor de prácticas, el encargado de las gestiones de las prácticas desde el lado de la universidad.

Al inicio del curso, los tutores de prácticas darán de alta a su empresa asignándose, a sí mismo u otro, como tutor de la misma. Además, definirán las especificaciones de cada práctica: número de alumnos a recibir, turnos de trabajo y semanas de contrato.

En cuanto al acceso a la aplicación, los únicos que pueden darse de alta o baja son los tutores externos. El responsable y los alumnos ya disponen de cuentas en el sistema, asignadas por la universidad.

Una vez entren los alumnos al sistema tendrán la opción de revisar si ya se les han asignado unas prácticas o realizar la selección de 10 empresas, en las que realizar las prácticas, ordenadas por orden de preferencia.

Tras el cierre del plazo de solicitud de prácticas el responsable lanzará el proceso de asignación de prácticas mediante al cual se le envía un correo a los alumnos para que sepan que ya se podrá observar la resolución en la aplicación. Esta asignación se realizará en función de la nota media del expediente del alumno.

Una vez finalizadas las prácticas de un alumno, su tutor asignado evaluará su desempeño en las prácticas y le asignará una nota, redactando un informe que describa el proceso de prácticas y las competencias involucradas en la evaluación.

Nuestra aplicación también permitirá la creación de tres informes, siendo estos:

- **Informe de prácticas asignadas:** Solicitado por el responsable, contiene datos sobre la resolución del proceso de asignación de prácticas: Número de alumnos registrados, número de alumnos asignados a cada empresa, total y separados por oferta, listado de empresas sin alumnos asignados y listado de alumnos sin prácticas asignadas.
- **Informe de evaluación de prácticas:** Solicitado por el responsable, consiste en un listado de alumnos junto con sus notas de prácticas, y un enlace de descarga del informe generado por el tutor al evaluar a cada alumno.
- **Informe de prácticas completadas:** Solicitado por un alumno, consiste en un informe que contiene los datos personales del alumno, la empresa y el tutor que ha

gestionado la práctica, junto con las fechas de inicio y fin de contrato, que son útiles para su Curriculum Vitae. En este informe quedan excluidas la evaluación y la nota de prácticas, del informe del tutor.

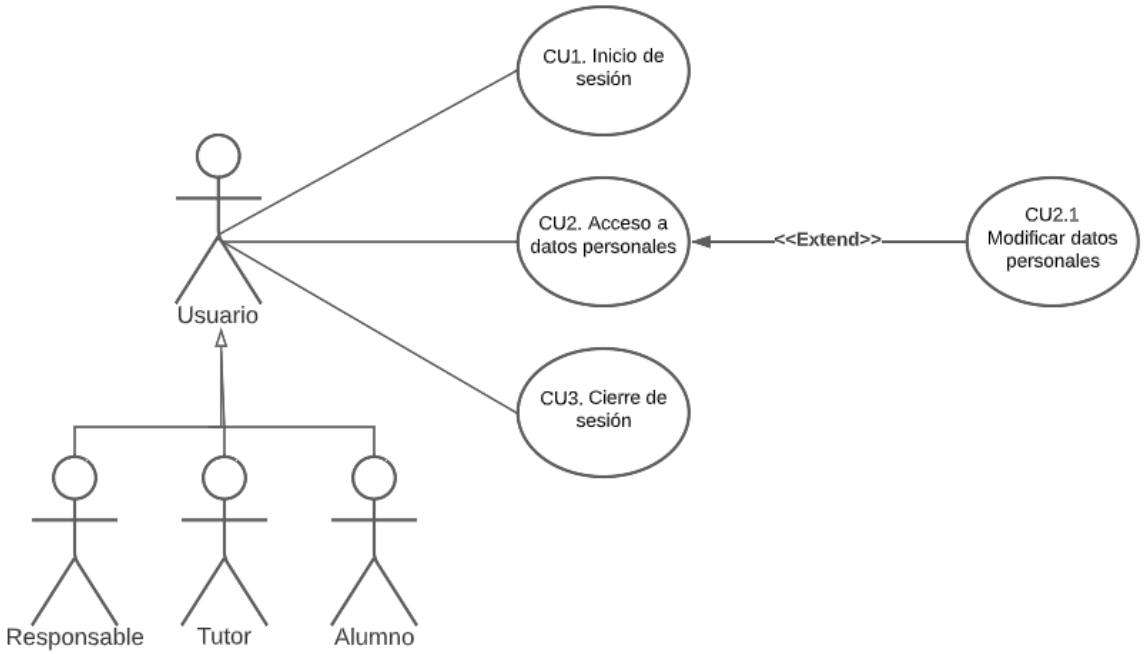
Solución propuesta

Teniendo en cuenta los requisitos recogidos en el análisis del problema, hemos planteado una solución, que consiste en una aplicación compuesta por dos servidores, front-end y back-end, cuya funcionalidad describimos con los siguientes diagramas.

Casos de uso

A continuación, diseñamos un diagrama de casos de uso en los que identificamos cuatro actores, usuario, tutor, responsable y alumno, donde los tres últimos heredan del primero.

Usuario



El actor “Usuario”, del cual heredan los actores tutor, alumno y responsable, puede realizar los siguientes casos de uso:

- **CU1:** El usuario accede a un formulario que le solicita sus credenciales de inicio de sesión: Nombre de usuario y contraseña.
    - Precondición: El usuario tiene que estar dado de alta en el sistema.
    - Postcondición: El usuario recibe una cookie de sesión en el sistema.
- **CU2:** El usuario solicita a la aplicación un resumen de los datos registrados en la misma.

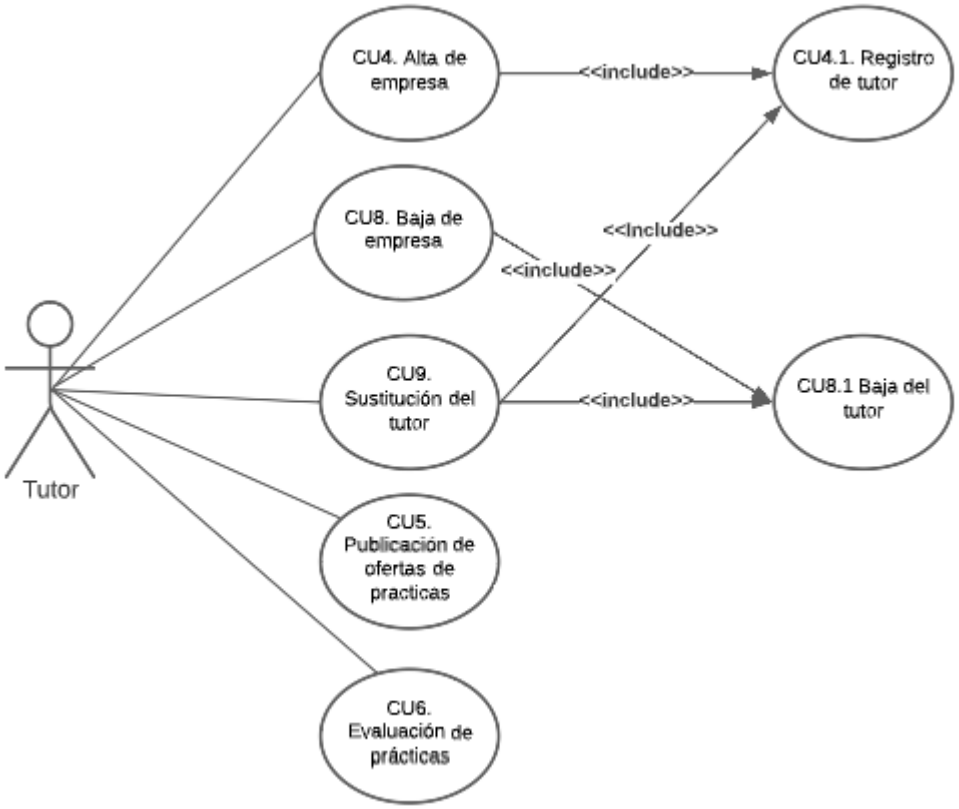
- Precondición: El usuario debe tener una sesión válida y activa.

- **CU2.1:** En la ventana de acceso a datos personales, el usuario puede modificar los datos registrados, dentro de los límites definidos por la aplicación, y guardar o descartar los cambios por medio de botones.
  - Precondición: El usuario ha solicitado un informe de sus datos personales.
  - Postcondición: Si se producen cambios, se reflejan en la base de datos.

- **CU3:** El usuario cierra su sesión actual y vuelve al menú de inicio de sesión (CU1).
  - Precondición: El usuario debe tener una sesión válida y activa.
  - Postcondición: La sesión activa deja de ser válida para autenticar al usuario.

El resto de actores heredan estos casos de uso del actor “Usuario”, dado que estos casos de uso son independientes del rol del actor en el sistema.

Tutor

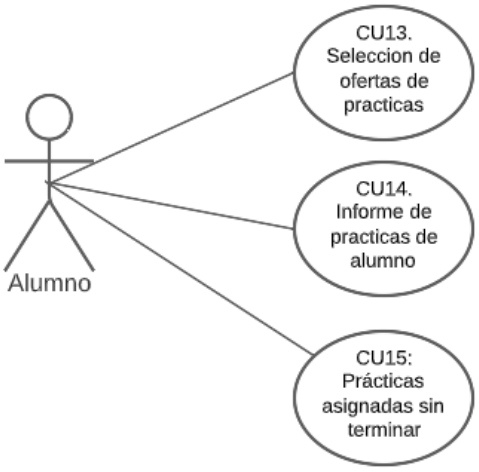


El actor tutor puede realizar los siguientes casos de uso:

- **CU4:** El tutor incluye una empresa al sistema.
  - Precondición: El tutor tiene que estar dado de alta en el sistema.
  - Precondición: Tiene que haber un tutor en el sistema al que se le será asignada la empresa.
- **CU4.1:** El tutor registra otro tutor en el sistema para asignarlo a una empresa nueva o para sustituirlo a él mismo en esta empresa.
  - Precondición: El tutor tiene que estar dado de alta en el sistema.

    - Poscondición: El tutor creado deberá ser asignado a una empresa.
- **CU5:** El tutor crea una nueva oferta de prácticas definiendo los datos de la oferta y publicándola.
    - Precondición: El tutor tiene que estar dado de alta en el sistema.
    - Precondición: La oferta de prácticas tiene que pertenecer a la empresa a la que ha sido asignado el tutor que crea la oferta.
- **CU6:** El tutor realiza la evaluación de las prácticas de un alumno.
    - Precondición: El tutor tiene que estar dado de alta en el sistema.
    - Precondición: El tutor solo es capaz de realizar evaluación de las prácticas que estén bajo su tutelaje.
    - Precondición: El alumno tiene que haber terminado sus prácticas.
- **CU8:** El tutor da de baja a una empresa que ya no vaya a ofertar prácticas.
    - Precondición: El tutor tiene que estar dado de alta en el sistema.
    - Precondición: El tutor dará de baja al tutor que tiene la empresa asignada.
- **CU8.1:** El tutor da de baja a un tutor que ya no vaya trabajar con nuestro sistema.
    - Precondición: El tutor tiene que estar dado de alta en el sistema.
    - Precondición: La empresa está dada de baja o existe otro tutor activo.
- **CU9:** El tutor buscará un sustituto para si mismo.
    - Precondición: El tutor tiene que estar dado de alta en el sistema.
    - Precondición: El tutor tendrá que darse de baja y haber añadido otro tutor que sirva de sustituto.

Alumno



El actor alumno, puede realizar los siguientes casos de uso:

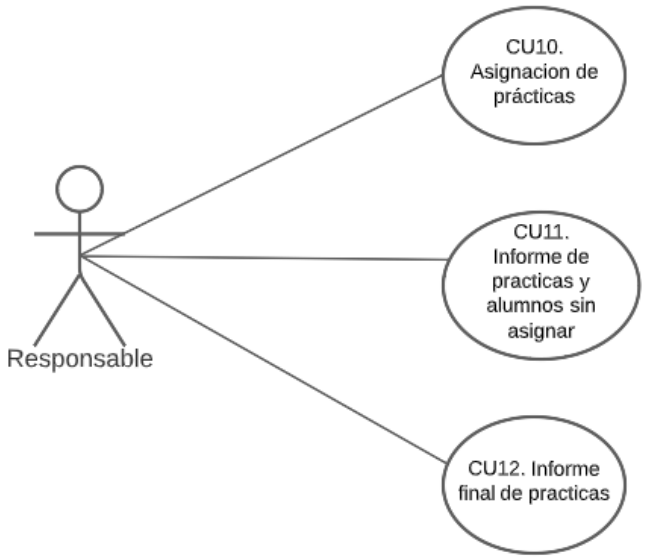
- **CU13:** El alumno accede a un ventana que le muestra las prácticas que podrá seleccionar con un orden de prioridad.
    - Precondición: El alumno tiene que estar dado de alta en el sistema.
    - Postcondición: El usuario genera su orden de selección de prácticas.

- **CU14:** El alumno accede a un ventana que le muestra un informe de sus datos relacionados con la práctica que ha realizado.

- Precondición: El alumno tiene que estar dado de alta en el sistema.
- Precondición: El alumno debe haber finalizado sus practicas.
- Postcondición: El usuario recibe su informe de datos.

- **CU15:** El alumno accede a un ventana que le muestra los datos de la práctica en curso que tenga.
  - Precondición: El alumno tiene que estar dado de alta en el sistema.
  - Precondición: El alumno debe haber empezado sus prácticas.

Responsable



El actor responsable, puede realizar los siguientes casos de uso:

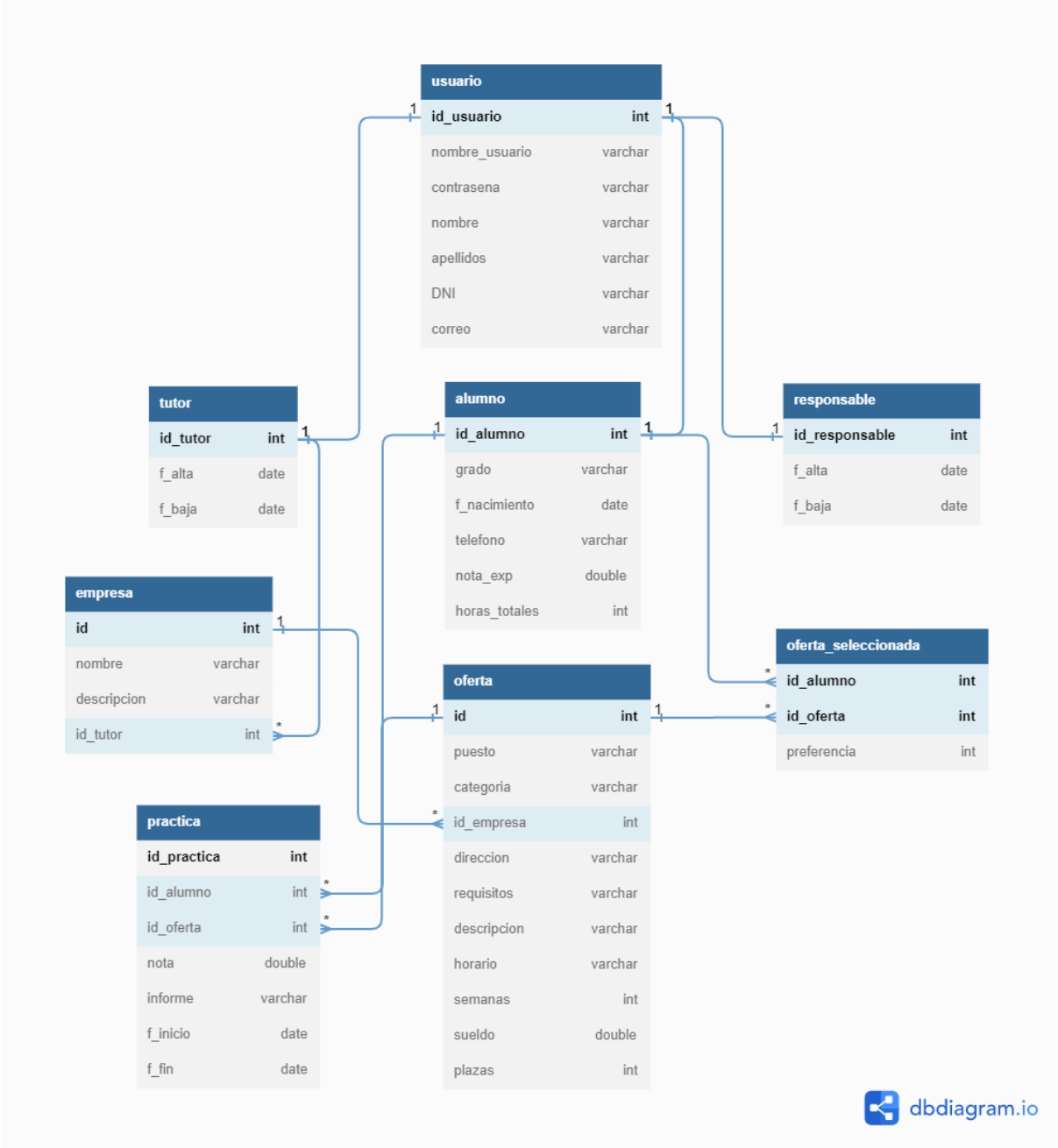
- **CU10:** El responsable realiza la asignación de prácticas, recibiendo en una ventana los datos de esa asignación.
  - Precondición: Los alumnos deben de haber realizado sus selecciones.
  - Postcondición: El responsable recibe los datos tras la asignación.

- **CU11:** El responsable accede a una ventana que le muestra los informes en curso de los alumnos que están realizando sus prácticas.
  - Precondición: Debe de haber alumnos con sus prácticas ya empezadas o finalizadas.
  - Postcondición: El responsable recibe los datos de prácticas de todos los alumnos.

- **CU12:** El responsable accede a la ventana que le muestra los informes en curso de los alumnos que están realizando sus prácticas, pero esta vez genera el informe final.
  - Precondición: Las prácticas de los alumnos deben de estar finalizadas.
  - Precondición: Los tutores deben de haber generado los informes.
  - Postcondición: El responsable genera el acta final de las prácticas.

Diseño de la BBDD (E/R)

La base de datos está organizada según la estructura del siguiente modelo E/R. Se representan las entidades como tablas, sus claves principales en negrita y sus relaciones con líneas azules.



```
ref: usuario.id_usuario - alumno.id_alumno
ref: usuario.id_usuario - tutor.id_tutor
ref: usuario.id_usuario - responsable.id_responsable

ref: empresa.id < practica.id_empresa
ref: tutor.id_tutor < empresa.id_tutor

ref: practica_asignada.id_alumno > alumno.id_alumno
ref: practica_asignada.id_practica > practica.id

ref: practica_seleccionada.id_alumno > alumno.id_alumno
ref: practica_seleccionada.id_practica > practica.id
```

En el diagrama E/R adjunto podemos observar las siguientes tablas:

- **Tabla usuario:** Representa al actor usuario, de él heredan las tablas tutor, alumno y responsable, y contiene los datos comunes a los tres roles de la aplicación:
    - nombre_usuario: Cadena de caracteres única para cada usuario que sirve como primera parte de sus credenciales.
    - contrasena: Cadena de 256 caracteres alfanúmerica que sirve como segunda parte de las credenciales de un usuario. Contiene la contraseña definida por el usuario al registrarse, después de aplicar el método de encriptación SHA256.
    - nombre: Nombre, simple o compuesto, del usuario, como cadena de caracteres alfabéticos.
    - apellidos: Cadena alfabética compuesta por uno o varios apellidos.
    - DNI: Cadena alfanumérica por 8 dígitos y un carácter alfabético de control, único para cada usuario.
    - correo: Cadena alfanumérica que sigue la expresión regular “[a-z0-9]+@[a-z]+\.[a-z]{2,3}”. Será único para cada usuario.
- **Tabla tutor:** Representa al actor tutor. Se relaciona con la tabla usuario para completar los campos generales que necesita cualquier perfil y con la tabla empresa, ya que existe un tutor para cada una. Hereda sus campos de la tabla usuario y añade los siguientes:
    - f_alta: Fecha de tipo *date* para representar el comienzo de las funciones del tutor. Se especifica como *not null* para asegurar que se introduce la fecha al dar de alta a un tutor.
    - f_baja: Fecha de tipo *date* para representar el fin de las funciones del tutor. Cuando el tutor deje de ejercer, no se borran sus datos de la BD, sino que se rellena este campo y deja de ser nulo.
- **Tabla alumno:** Representa al actor alumno, que hereda los campos de la tabla usuario y añade los siguientes:
    - grado: Cadena de caracteres alfabéticos que representa qué grado, dentro de la universidad, está cursando el alumno.

    - f_nacimiento: Tipo Date que representa la fecha de nacimiento del alumno, con la que podemos obtener su edad.
    - teléfono: Cadena alfanumérica que sigue la expresión regular "^\+?(\d{2})|\d{6}$", que permite números con el formato "+dd ddd ddd ddd", donde "d" es un dígito del 0-9.
    - nota_exp: Número real entre 0.0 y 10.0 que representa la nota media de todas asignaturas del expediente del alumno.
- **Tabla responsable:** Representa al actor responsable. Se relaciona únicamente con la tabla usuario para heredar sus campos y añadir los mismos que la tabla tutor:
    - f_alta: Fecha de tipo *date* para representar el comienzo de las funciones del responsable. Se especifica como *not null* para asegurar que se introduce la fecha al dar de alta al responsable.
    - f_baja: Fecha de tipo *date* para representar el fin de las funciones del responsable. Cuando el único responsable de la aplicación deja de ejercer, se añade su fecha de baja.
- **Tabla empresa:** Representa a la entidad que ofrece prácticas de empleo a los alumnos. Una empresa tiene los siguientes campos:
    - nombre: Cadena de caracteres alfanuméricos que representa el nombre de la empresa.
    - descripcion: Cadena de caracteres alfanuméricos que representa una descripción de las funciones que desempeña la empresa.
    - id_tutor: Identificador de números enteros que representa al tutor de la empresa. Solo puede existir un único tutor al mismo tiempo para cada empresa, pero un tutor puede aparecer en varias empresas.
- **Tabla oferta:** Representa una oferta de práctica ofrecida por una empresa, que contiene los siguientes campos:
    - puesto: Cadena alfanumérica que da nombre a la oferta de prácticas. Representa el puesto de prácticas que ofrece la empresa.
    - categoria: Cadena alfabética que representa un campo o ámbito de trabajo, utilizado para filtrar ofertas más fácilmente.
    - direccion: Cadena alfanumérica que representa la dirección física de la oficina o edificio en el que se llevarán a cabo las prácticas.
    - requisitos: Cadena alfanumérica que representa un listado de requisitos mínimos que un alumno necesita cumplir para solicitar dicho puesto de prácticas.
    - descripcion: Cadena de caracteres alfanumérica que describe las tareas realizadas durante las prácticas y detalles relevantes sobre la oferta.
    - horario: Cadena alfanumérica, típicamente con el formato "Lunes a Viernes de 8 a 14h" o "Lunes: 8-14h,Martes:15-17h,...", que representa el horario de trabajo definido en el contrato de prácticas.
    - semanas: Número de semanas disponibles para completar las horas de prácticas.
    - sueldo: Sueldo mensual, en bruto, que recibirá el alumno durante el período de prácticas. Al tratarse de una universidad española, la moneda es el euro.

    - plazas: Número de alumnos que pueden ser asignados a una oferta de prácticas.
- **Tabla oferta_seleccionada:** Representa la relación entre un alumno y las prácticas que solicita realizar. Una solicitud puede tener hasta 10 prácticas, las claves de la tabla son *id_alumno* e *id_practica*, ya que un mismo alumno puede tener 10 entradas. Contiene los identificadores para relacionarse con los alumnos y las prácticas, y un campo extra:
    - preferencia: Número entero para representar del 1 al 10 el grado de preferencia de una práctica, siendo el valor 1 el preferido.
- **Tabla practica**: Representa la entidad practica, la relación entre un alumno y la oferta de prácticas que se le asigna. Las claves de la tabla son el identificador de la práctica, el alumno que la realiza y los datos de la oferta, ya que un alumno puede realizar varias prácticas y una oferta puede dar lugar a varias prácticas. Contiene los campos:
    - nota: Número real entre 0.0 y 10.0 que representa la calificación del desempeño del alumno durante la práctica. Será nulo hasta que el tutor lleve a cabo la evaluación.
    - informe: Texto alfanumérico redactado por el tutor al evaluar las prácticas, que describe las tareas realizadas por el alumno durante el proceso de prácticas y una evaluación del mismo.
    - f_inicio: Fecha de tipo *date* que indica el comienzo de la realización de una práctica por parte de un alumno. Se establece como *not null* para asegurar que se introduce al asignarse a un alumno.
    - f_fin: Fecha de tipo *date* que indica la finalización del proceso de prácticas. Las prácticas cuya fecha de fin sea nula se consideran en curso, de lo contrario, se consideran terminadas.

Problemas encontrados y soluciones aportadas

Durante el análisis de requisitos de la aplicación, hemos encontrado detalles sin definir en los requisitos del enunciado:

- **Usuarios de la aplicación:** Los usuarios de la aplicación se clasifican en 3 tipos: Responsables, uno por universidad, encargado de gestionar a todos los alumnos de la misma; tutor, uno por cada empresa, encargado de evaluar y gestionar a todos los alumnos de prácticas en su empresa; y alumno, asociado a una única universidad y relacionado con su tutor por medio de la empresa que oferta la práctica.
- **Mensajes de correo electrónico:** Una vez terminada la asignación de prácticas, el correo de notificación se realizará por medio de la plataforma de la universidad, de forma ajena a nuestra aplicación.
- **Alumnos participando en varias prácticas:** Suponiendo el caso de que un alumno suspenda unas prácticas, dicho alumno deberá repetir todo el proceso de selección de prácticas, quedando registrada su actividad en más de una práctica.

Por otro lado, durante el diseño del diagrama de casos de uso, hemos decidido soluciones para los siguientes problemas:

- **Alta de un tutor:** Hemos considerado el caso en el que un tutor se da de alta en el

sistema para, seguidamente, dar de alta su empresa como ofertadora de prácticas. No hemos permitido la existencia de tutores no asociados a una empresa, por lo que el proceso de alta consiste en dar de alta a una empresa, proceso que incluye el alta de un tutor.

- **Baja de un tutor:** Dado que una empresa no puede estar dada de alta sin un tutor activo, el tutor no puede darse de baja directamente. Para hacerlo, deberá de dar de baja a su empresa, o sustituirse por otro tutor, dejando ambas acciones a dicho tutor dado de baja en el sistema.

Al diseñar el modelo E/R hemos tomado las siguientes medidas:

- **Usuarios activos:** Los usuarios de tipo Responsable y Tutor se mantendrán en la base de datos aún después de su baja, lo que supone una necesidad de identificar qué usuarios están activos y cuales están dados de baja. Para ello, las tablas respectivas a cada uno guardan dos fechas, inicio y fin de actividad, que representan el período durante el cual estuvieron dados de alta. Para identificar usuarios dados de alta, basta con filtrar aquellos tutores o responsables cuya fecha de fin aún sea nula.
- **Oferta seleccionada:** Para evitar que un alumno seleccione varias veces una oferta como preferida, los identificadores de alumno y oferta son claves, para qué sólo exista un valor de preferencia para cada oferta.
- **Prácticas sin terminar:** Para identificar prácticas en curso utilizaremos el campo f_fin, que comienza nulo y se mantiene en ese estado hasta que un alumno haya finalizado sus horas de prácticas curriculares.
- **Identificador de prácticas**. A pesar de asignar como claves el identificador de un alumno y el de una oferta para distinguir una práctica, facilita la comunicación entre backend y frontend asignar un identificador a la práctica, que también será clave. Se utiliza para el envío de peticiones a la API.
- **Ofertas reutilizables:** Cuando una oferta se queda sin plazas tras la asignación de prácticas, no se elimina de la BBDD, si no que deja de mostrarse en el listado de ofertas disponibles. De este modo, si una empresa quiere volver a ofertar el mismo contrato de prácticas, basta con que vuelva a asignar un número de plazas mayor a cero para que los alumnos reciban la oferta.

En cuanto al MockUp de la aplicación front-end:

- **Datos personales para diferentes roles:** La presencia de diferentes roles en la BBDD y las diferencias entre ellos provocan la necesidad de una representación distinta de sus datos personales. Por ello, la página de front-end tendrá más o menos campos en función del rol, dando como resultado tres páginas de datos personales diferentes.

Por último, durante el diseño del MockUp de la aplicación back-end, hemos aplicado las siguientes soluciones:

- **Asignación de prácticas:** Cuando el responsable da comienzo al caso de uso de asignación de prácticas, en el front-end, se produce una llamada a la API REST. Hemos decidido implementar el nodo “/practice/assign” que, recibiendo por POST un listado de alumnos, y un listado de ofertas, devuelve un listado de ofertas asignadas a cada alumno.
- **Permisos sobre la API:** Conociendo el identificador de una empresa o de un

usuario podemos obtener información sobre sus prácticas, nota de expediente o calificación de la práctica actual. Por lo cual, el acceso a la API debe estar restringido a los usuarios con acceso a dichos recursos, y nadie más.

Tecnologías a utilizar

Para llevar a cabo la implementación del proyecto *GestionDePracticas* vamos a hacer uso de las siguientes herramientas software, que clasificaremos en Backend, Frontend y DevOps:

Backend

Para el desarrollo de la API REST del proyecto, encargada de servir funciones de consulta y modificación del estado de la aplicación, utilizaremos:

- Lenguajes de Programación
  - **Java**: Lenguaje de programación orientado a objetos utilizado para implementar la lógica del servidor.
  - **SQL**: Lenguaje de consulta a BBDD estándar utilizado por bases de datos relacionales.
- Lenguajes de Configuración
  - **YAML**: Es un lenguaje de declaración de datos que facilita la legibilidad y la capacidad de escritura del usuario, utilizado en nuestro proyecto para almacenar pares clave-valor que representan configuración del servidor. Por ejemplo: url de la BBDD, nombre y contraseña de la misma, etc.
- Frameworks
  - **Spring Boot**: Es un framework que nos permite crear aplicaciones autocontenidas, desviando la atención del programador de la arquitectura y la configuración, centrándose únicamente en el desarrollo de la lógica.
  - **JUnit**: Es un Framework Open Source para la automatización de las pruebas, tanto unitarias, como de integración.
- IDEs
  - **Apache Netbeans**: Entorno de desarrollo integrado libre, para el desarrollo de aplicaciones web, móviles o de escritorio, principalmente en Java.
  - **Postman**: Plataforma de API que permite a los desarrolladores construir y probar llamadas a sus APIs. Útil para depurar.
- Persistencia
  - **MariaDB**: Sistema de gestión de bases de datos relacional, de código abierto. Utiliza la misma sintaxis que el estándar SQL, pero difiere de otras bases de datos en la manera en la que almacena los datos y procesa las funciones.
  - **JDBC**: API que define una librería estándar para el acceso a fuentes de datos, principalmente orientado a bases de datos relacionales basadas en SQL, sobre Java.

Frontend

La arquitectura de nuestro proyecto dispone de un servidor frontend, enfocado a la interacción con los usuarios y la representación de los datos para el mismo. Este servidor se comunica con la REST API ofrecida por el servidor backend y hace uso de su interfaz para consultar o modificar el estado de la aplicación. Para llevar a cabo el desarrollo de este servidor utilizamos:

- Lenguajes de Programación
  - **TypeScript**: Es un superconjunto de JavaScript, que añade tipos estáticos y objetos basados en clases al lenguaje de programación, dialecto de ECMAScript, que nos permite generar contenido dinámico en el lado del cliente.
- Configuración y comunicación con backend
  - **JSON**: JavaScript Object Notation, formato de intercambio de datos basado en pares clave-valor, nativo de JavaScript, aplicado sobre texto plano. Este formato se utilizará tanto para configurar las propiedades del servidor frontend como para intercambiar objetos con la API de SpringBoot.
- Frameworks
  - **Angular.js**: Framework MVC basado en TypeScript muy útil para desarrollar tanto aplicaciones web SPA, Single-Page Applications, como aplicaciones con diferentes rutas. Utiliza una estructura basada en componentes y la capacidad de TypeScript para generar contenido dinámico para ofrecer nuevas etiquetas HTML, reutilizables y que se renderizan de forma independiente, sin tener que recargar la página.
- IDEs
  - **Visual Studio Code**: Entorno de desarrollo integrado muy potente, que da soporte a una gran cantidad de lenguajes de programación y proyectos gracias a su mercado de extensiones y la integración con la terminal. Por esta razón, utilizamos VS Code para front en lugar de Netbeans.

DevOps

Los diferentes servidores que componen la aplicación, frontend, backend y sistema gestor de base de datos, están desplegados cada uno en un contenedor diferente. Para la gestión de la infraestructura y la configuración de los contenedores utilizamos:

- Lenguajes de Programación
  - **Shell Script**: Lenguaje utilizado para la automatización de tareas y procesos complejos sobre el intérprete de comandos Bash.
- Lenguajes de Configuración
  - **YAML**: Sigue la misma definición que el punto explicado en Backend, utilizado en archivos de configuración de infraestructura.

- IDEs
  - **Vi editor**: Editor de texto para terminal, sin GUI, muy potente gracias a un conjunto de atajos y sistemas de automatización.
  - **Visual Studio Code (Remote)**: Versión remota, conectada por ssh a la máquina virtual, de VS Code que facilita la programación en remoto.
  - **Netbeans (Remote)**: Versión remota, conectada por ssh a la máquina virtual, de Netbeans que facilita la programación en remoto.
- Servidores web
  - **Tomcat**: Contenedor de servlets de código abierto, desarrollado bajo el proyecto Jakarta, para la implementación de JavaServlet, JavaServlet Pages (JSP) o Java Sockets. Es el servidor sobre el que trabaja Spring Boot y cuya configuración y gestión nos abstrae.
  - **Node.js**: Entorno de ejecución de JavaScript, multiplataforma, de código abierto, orientado a eventos asíncronos. Es el servidor sobre el que trabaja Angular.js, abstrayendo los detalles de configuración y gestión.
- Contenerización
  - **Docker**: Proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores, proporcionando una capa adicional de abstracción y automatización de la virtualización.
    - **Docker-Compose**: Herramienta basada en YAML que permite definir, desplegar y compartir aplicaciones multi-contenedor fácilmente.

Flujo de trabajo

El equipo de cinco miembros se dividirá en dos equipos de desarrollo, de tres miembros cada uno, donde el tercer miembro del equipo estará presente en ambos. Este último miembro es el encargado de gestionar la infraestructura de la aplicación, además de participar en el proceso de desarrollo de ambos equipos, asumiendo el rol de DevOps.

Para facilitar la comunicación y el trabajo en equipo, el flujo de trabajo seguirá las siguientes pautas:

Github

Para desarrollar el proyecto utilizaremos la plataforma de control de versiones github, junto con la herramienta git. El flujo de trabajo se organiza y asigna en proyectos de github, como explicamos en adelante, y el versionado del código se realizará mediante repositorios.

Control de Versiones

Al ser un proyecto académico, vamos a agrupar todos los servidores de la aplicación en un único repositorio. Dicho repositorio contendrá una rama “main” con la última versión “puesta en producción” de la aplicación en cada momento, en paralelo, tendremos dos ramas distintas: “front” y “back”.

Los equipos de desarrollo de cada servidor trabajarán en su respectiva rama, propiciando un trabajo limpio, independiente de fallos en otro servidor y aislando los esfuerzos de ambos equipos cuando sea necesario.

Dentro de cada rama, los miembros de cada equipo crearán las subramas que consideren necesarias siguiendo el formato “front-NombreDeIssue” o “back-NombreDeIssue”. De este modo, las ramas de front y back representan la última versión, en desarrollo, estable de cada servidor.

La integración de subramas en sus ramas superiores se hará mediante Pull Requests (PR), que en el caso de cada equipo implica una revisión por parte de otro integrante del equipo, y en el caso de la rama main implica la revisión por parte del miembro encargado de la infraestructura.

Reparto de tareas

El reparto de tareas de la aplicación se llevará a cabo mediante Github Projects. Un proyecto está compuesto por cuatro fases: *Todo*, *In Progress*, *Done* y *Accepted*. Las tareas del proyecto se representan con *issues* de github, y se distribuirán en cada una de las fases en función de su estado: Sin empezar, en progreso, lista para revisión y aceptada.

Transferir una tarea de una fase a otra es responsabilidad del integrante que tenga la tarea asignada para las fases Todo, In Progress y Done. Sin embargo, debe ser otro miembro del equipo el que acepte la tarea, dejando un comentario en la misma, “Tarea aceptada”, o devolverla a la fase In Progress si no es aceptada, dejando un comentario con el motivo del rechazo.

Excluyendo el proyecto creado para realizar este documento, el repositorio estará asociado a tres proyectos: General, Frontend y Backend. El proyecto general contendrá las tareas globales del proyecto, junto a las relativas a la infraestructura y la integración de ambos servidores. Los proyectos Frontend y Backend contendrán las tareas específicas de cada equipo de desarrollo, relativas a la funcionalidad a implementar o probar.

Desarrollo de código

El desarrollo de soluciones software para ambos servidores deberá seguir una serie de pautas acordadas entre los miembros del equipo.

Implementación de soluciones de código

Durante el desarrollo del proyecto, independientemente del equipo, la implementación del código fuente del mismo se hará siguiendo la siguiente metodología:

1. Diseño de la función que satisface el requisito que requiere una solución.
2. Modularización de la misma en aras de su reusabilidad y legibilidad.
3. Implementación de la solución diseñada.
4. Documentación de todas las funciones involucradas siguiendo el estándar Javadoc en backend, y Dgeni, o similares, para frontend.

5. Diseño de test unitarios válidos y suficientes para poder detectar errores o bugs.
6. Reunión con todos los miembros del equipo, back o front, para el diseño de test de integración.
7. Una vez superados todos los test, creación de una Pull Request para integrar la solución en la rama estable.

Este proceso, llevado a cabo en una subrama de las ramas back o front, asegura la correcta integración de nuevo código a la aplicación resultante, que permite a todos los miembros del equipo trabajar con un número mínimo de bloqueos, por trabajo de sus compañeros, y hacer “rollbacks” a versiones estables del proyecto.

Diseño de test unitarios

Una vez finalizado el desarrollo de una primera versión de una solución software, el responsable de la misma debe diseñar test unitarios, en la tecnología de desarrollo de tests correspondiente, suficientes para comprobar en una sola ejecución el correcto funcionamiento de las soluciones.

Un test unitario es una prueba que permite determinar el correcto funcionamiento de las unidades más pequeñas de nuestro software, las funciones. Esto implica la prueba de funciones de forma independiente, “mockeando” sus dependencias, asumiendo que estas funcionan correctamente.

El uso de objetos “Mock” en los test unitarios sirve para simular el comportamiento de dependencias, sin necesidad de inyectar. Un ejemplo de esto son las llamadas a base de datos, utilizaremos un objeto “Mock” donde definiremos respuestas programadas a determinadas consultas de interés, sin realizar una conexión real a la base de datos.

Diseño de test de integración

El otro tipo de test que utilizamos para el desarrollo del proyecto son los test de integración. A diferencia de los test unitarios, los test de integración intentan demostrar el correcto funcionamiento de nuestras soluciones, en conjunto con las demás soluciones que forman el servidor.

En este tipo de tests sí inyectamos las dependencias de cada función, asumiendo que el comportamiento unitario de las funciones involucradas es correcto. Siguiendo el ejemplo anterior y suponiendo que una función de registro de usuario y la función que guarda datos en la base de datos funcionan correctamente por separado, un test de integración comprueba que, en efecto, la base de datos contiene el usuario generado por la función de registro.

Por lo tanto, a la hora de diseñar los tests de integración, debemos tener en cuenta que estos se ejecutan únicamente una vez superados los test unitarios.

Mockup de la aplicación

El mockup de la aplicación consiste en un boceto gráfico de la interfaz de usuario, en el caso del servidor front-end, o del acceso a la API REST, en el caso del servidor back-end.

Frontend

La primera versión en el apartado de frontend tiene la siguiente estructura:

- La primera pantalla que se observa es la de acceso a la app, visible por todos los usuarios (actores) de la aplicación, con una apariencia similar a la siguiente:

Alumno

- A continuación, se muestran los menús específicos para cada tipo de usuario, empezando por la apariencia que tendría el menú del alumno:

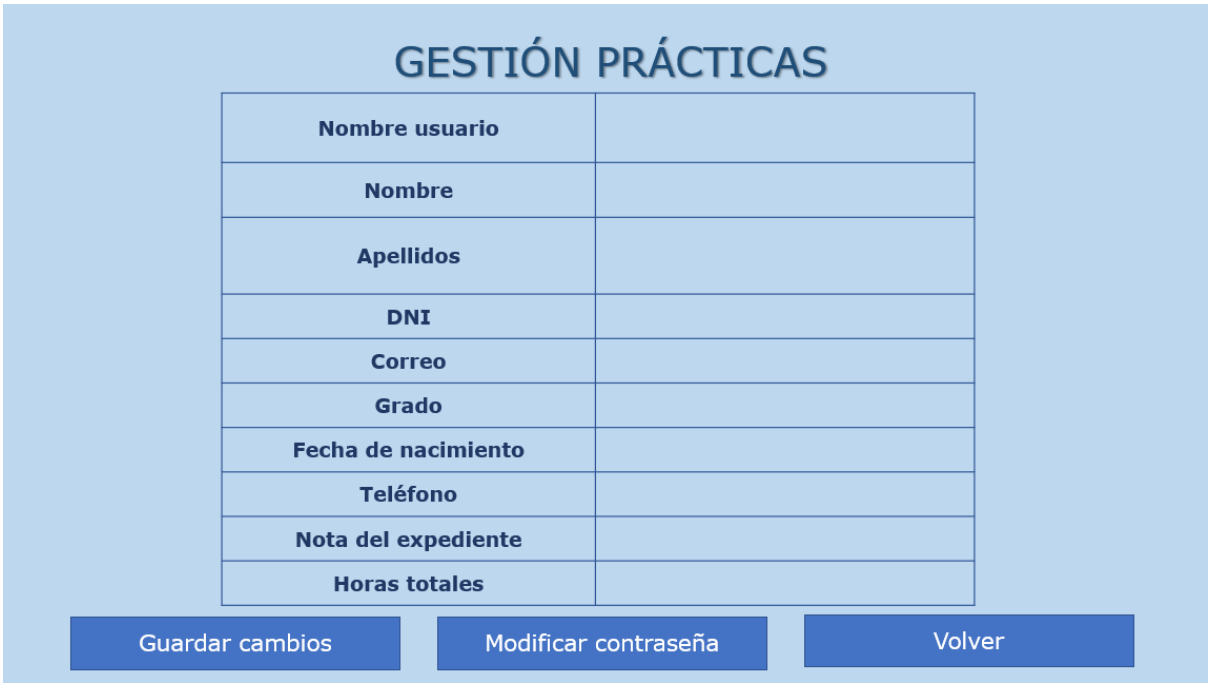
Como se ve en el menú, existen 4 opciones, que tendrán una apariencia similar a la siguiente:

- Página que verán los alumnos al elegir la opción de seleccionar sus preferencias (corresponde al caso de uso número 13).

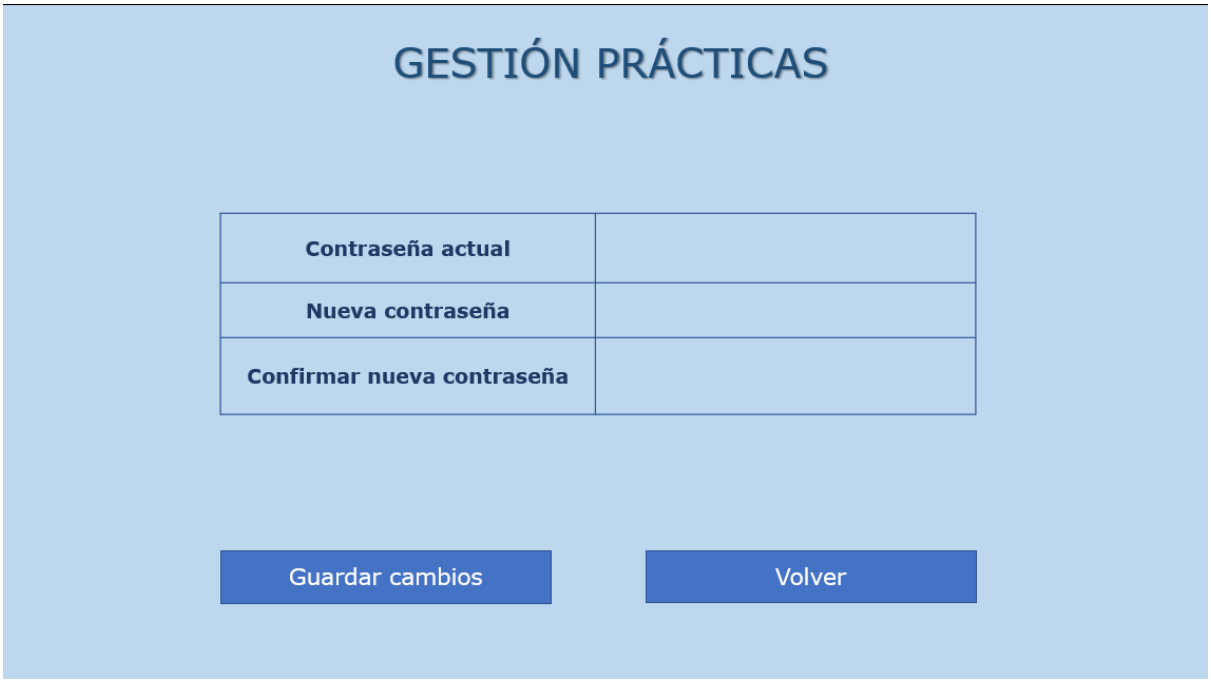
- Página que verán los alumnos al seleccionar la opción de ver sus prácticas en curso (corresponde al caso de uso número 15).

- Página que verán los alumnos al seleccionar la opción de ver sus datos (corresponde al caso de uso número 14).

- Página que verán los alumnos al seleccionar la opción de ver datos personales (corresponde al caso de uso número 2.1).

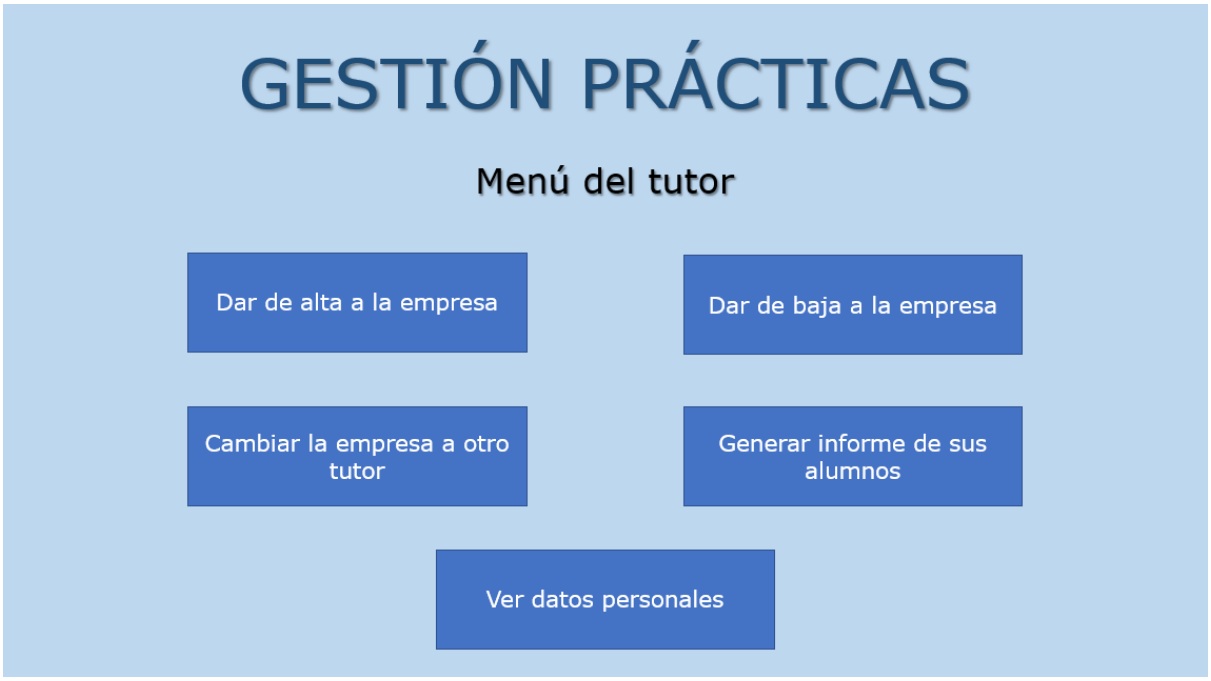


- Página que ven todos los usuarios al seleccionar *modificar contraseña*. Deben introducir su clave actual y escribir dos veces la nueva contraseña.



Tutor

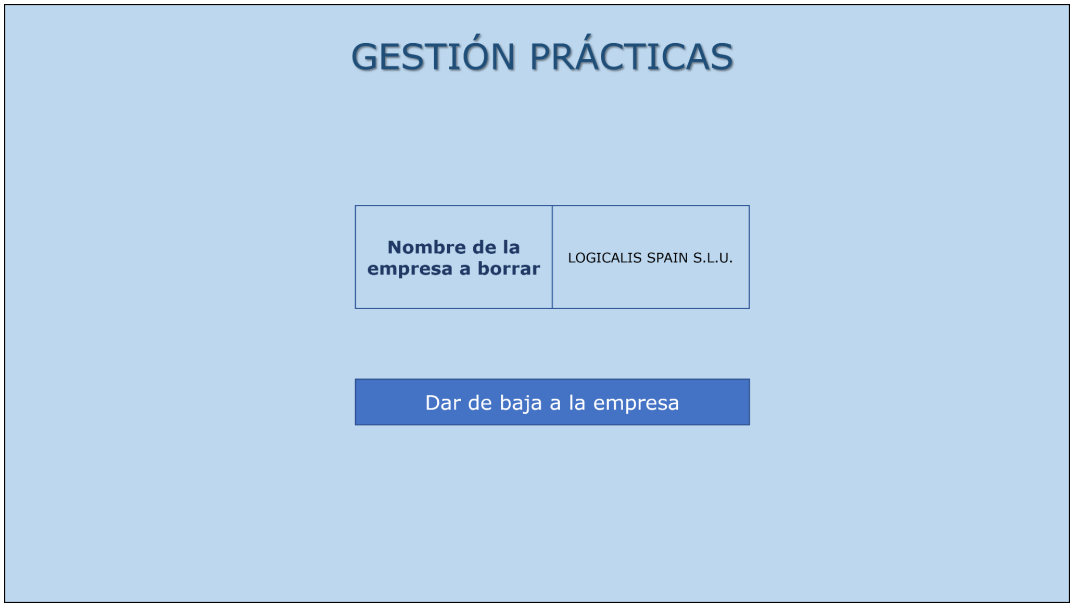
- El diseño del menú del tutor al acceder a la aplicación es el siguiente:



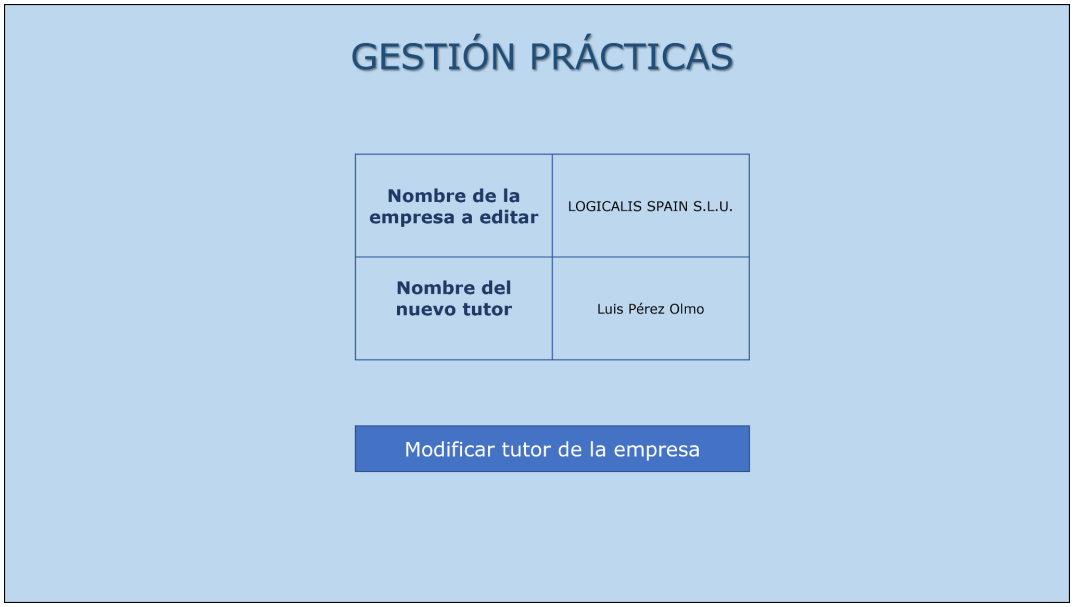
Como se ve en el menú, contiene 5 opciones, que tendrán una apariencia similar a la siguiente:

- Página que verán los tutores al seleccionar la opción de dar de alta una empresa en el sistema (corresponde al caso de uso número 4).

- Página que verán los tutores al seleccionar la opción de dar de baja una empresa en el sistema (corresponde al caso de uso número 8).



- Página que verán los tutores al seleccionar la opción de cambiar un tutor de una empresa en el sistema (corresponde al caso de uso número 9).

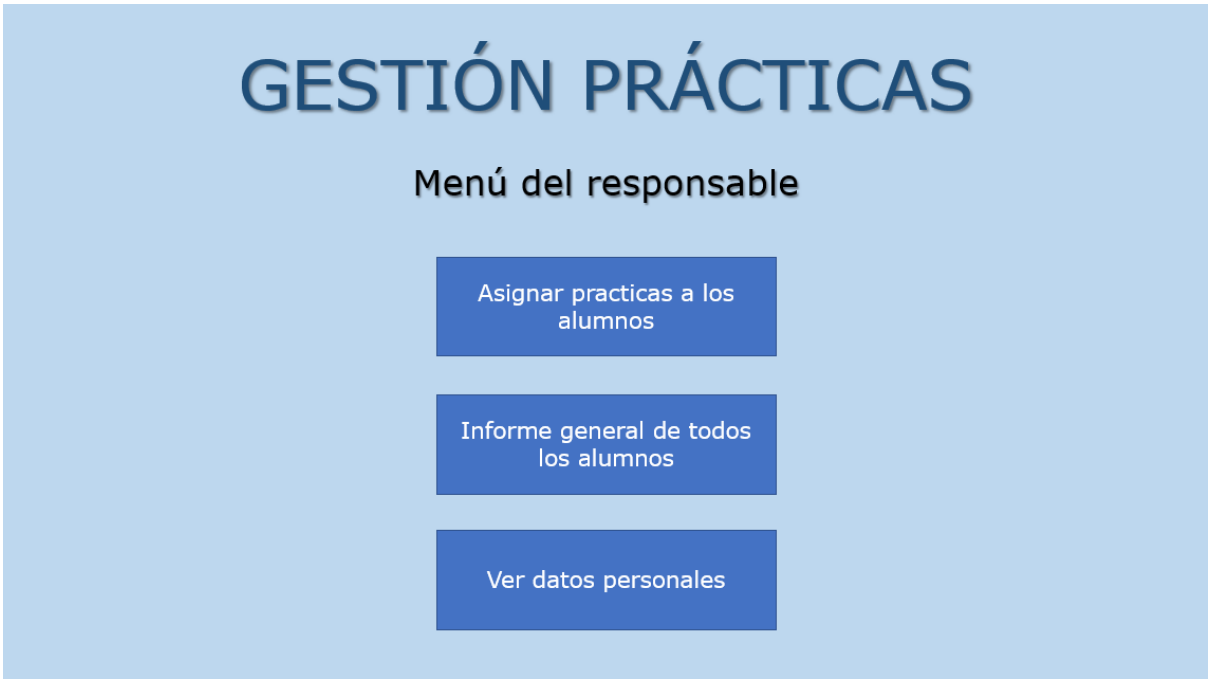


- Página que verán los tutores al seleccionar la opción de generar los informes de los alumnos (corresponde al caso de uso número 6).

- Página que verán los tutores al seleccionar la opción de modificar sus datos (corresponde al caso de uso número 2.1).

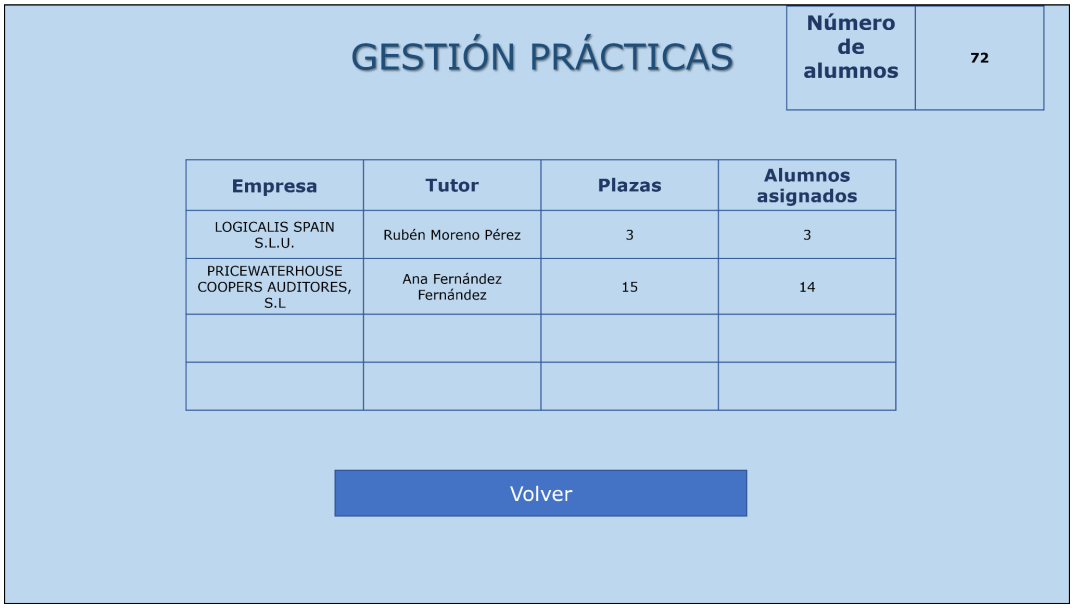
Responsable

- El diseño del menú del responsable al acceder a la aplicación es el siguiente:

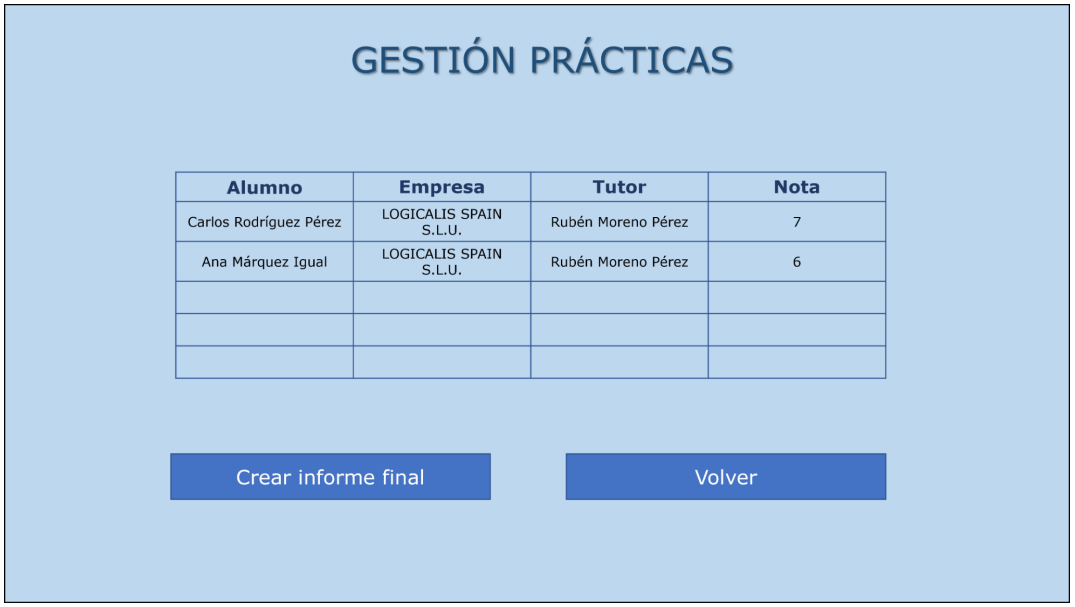


Como se ve en el menú, existen 3 opciones, que tendrán una apariencia similar a la siguiente:

- Página que observa el responsable al seleccionar la opción de asignar las prácticas a los alumnos. Se muestra para visualizar las estadísticas tras lanzar el proceso de asignación (corresponde al caso de uso número 10).



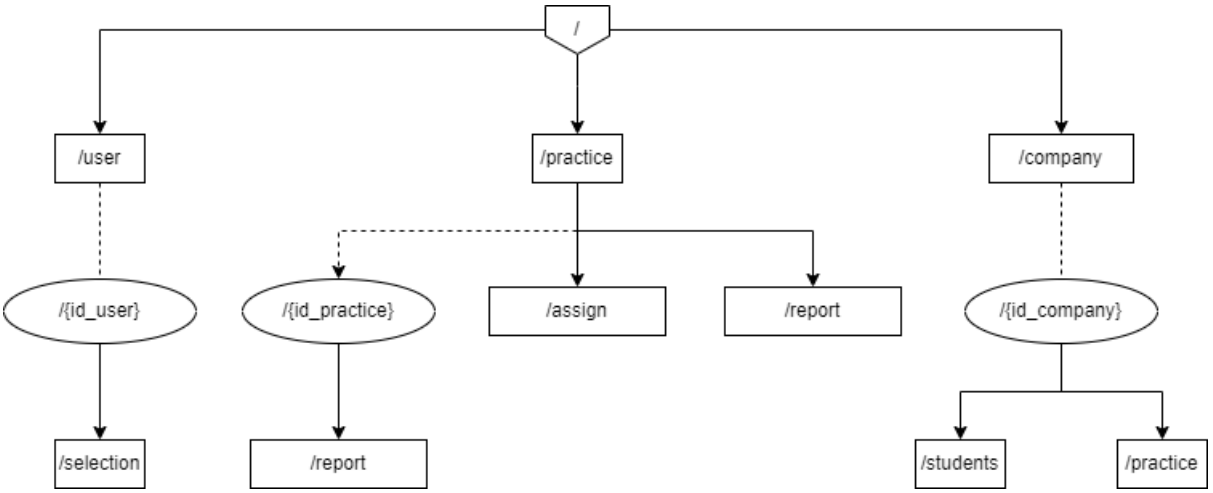
- Página que verán los responsables al seleccionar la opción de crear el informe general de todos los alumnos (corresponde a los casos de uso número 11 y 12).



- Página que verán los responsables al seleccionar la opción de modificar sus datos (corresponde al caso de uso número 2.1).

Backend

El mockup del servidor backend consiste en un diagrama que representa los diferentes "endpoints" de nuestra aplicación.



- **/user**

POST → Inicio de sesión de cualquier usuario.

GET → El responsable obtiene el número total de alumnos.

- **/user/{id_user}**

GET → Obtener información del usuario.

POST → Modificar información del usuario.

- **/user/{id_user}/selection**

GET → Obtener selección realizada por el alumno.

POST → Publicación de selección de prácticas según preferencias realizada por el alumno.

- **/practice**

GET → Obtener listado de todas las prácticas ofertadas en la aplicación.

POST → Alta de una práctica por parte de un tutor.

- **/practice/assign**

POST → El responsable lanza el proceso de asignación de prácticas a alumnos según su selección por preferencia.

- **/practice/report**

GET → El responsable obtiene un informe de todos los alumnos de todas las prácticas con sus notas para poder rellenar las actas oficiales.

- **/practice/{id_practice}**

GET → Obtener información completa de una práctica.

POST → Modificar información o dejar de ofertar una práctica.

- **/practice/{id_practice}/report**

GET → Alumno obtiene su informe de realización de la práctica.

- **/company**

POST → Alta de una empresa por parte de un tutor.

GET → El responsable obtiene un resumen sobre el número de alumnos por empresa y si alguna empresa no va a recibir alumnos.

- **/company/{id_company}**

GET → Obtener información de la empresa.

POST → Modificar información de la empresa.

- **/company/{id_company}/students**

GET → El tutor obtiene la lista de estudiantes de las prácticas de su empresa.

- **/company/{id_company}/practice**

POST → El tutor publica el informe y nota de los alumnos de una práctica.

GET → El tutor obtiene la lista de todas las prácticas realizadas o en curso de su empresa.

Bibliografía

- Generador de expresiones regulares
- Diagrama de casos de uso online
- Modelo E/R online
- Documentación de Github Projects
- Mockup back-end online
- Documentación de Spring Boot
- Documentación de JUnit
- API JDBC Java
- Documentación Angular.js
- Documentación Docker
- Ejemplo de aplicación similar al propuesto
- Testing en Angular.js
- Testing de Spring Boot y Spring Security