eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

GRADO EN INGENIERÍA INFORMÁTICA DE GESTIÓN Y SISTEMAS DE INFORMACIÓN

TRABAJO FIN DE GRADO

SISTEMA DE GESTIÓN DOCUMENTAL DE PRÁCTICAS VOLUNTARIAS UTILIZANDO FIRMA DIGITAL

Alumna: Gutierrez Beitia, Ariane
Directora: Armendariz Leunda, Ana Jesus

Curso: 2018-2019

Fecha: 20 de julio del 2019

Resumen

El proyecto “Sistema de gestión documental de prácticas voluntarias utilizando firma digital” consiste en el diseño e implementación de un sistema web accesible por alumnos, profesores y empresas, para realizar la gestión de selección y documentación de prácticas empresariales. La aplicación está diseñada para llevar a cabo la firma del contrato lo más fácil y cómoda posible, haciendo uso de la firma digital, y así contribuir al desarrollo sostenible.

En este documento se detalla todo el proceso de ejecución del proyecto, empezando por el planteamiento inicial, que es la fase previa a la creación, seguido por los requisitos originales, el análisis, diseño y desarrollo. A continuación, se especifican las pruebas finales del sistema y, para finalizar, se presenta la evaluación de todo el proceso realizando la valoración del proyecto acabado.

Palabras clave

Firma digital, Angular, servicios REST, servidor, prácticas, componente

Laburpena

“Sinadura digitalaren bidezko borondatezko praktiken dokumentazio kudeaketa sistema” proiektua, ikasle, irakasle eta empresentzak eskuragarri dagoen web aplikazio baten diseinu eta inplementazioan datza, non empresa-praktiken aukeraketa eta dokumentazioa kudeatzen den. Aplikazioa diseinatuta dago hitzarmenaren sinadura ahalik eta errezena eta erosoena egiteko sinadura digitala erabiliz eta, horrela, garapen iraunkorrari laguntzeko.

Dokumentu honek proiektuaren exekuzio prozesu osoa zehazten du, hasierako planteamenduarekin hasita, aurre-sorkuntza fasea dena, eta jatorrizko eskakizunak, analisia, diseinua eta garapenarekin jarraituta. Ondoren, sistemako azken probak zehazten dira eta, azkenik, prozesu osoaren ebaluazioa aurkezten da, amaitutako proiektua ebaluatuz.

Hitz gakoak

Sinadura digitala, Angular, REST zerbitzuak, zerbitzaria, praktikak, konponentea

Summary

The project "Documentary management system of voluntary practices using digital signature" consists of the design and implementation of a web system accessible by students, teachers and companies, to carry out the management of selection and documentation of business practices. The application is designed to carry out the signing of the contract as easy and comfortable as possible, using the digital signature, and this contributes to a sustainable development.

This document details the entire project execution process, starting with the initial approach, which is the pre-creation phase, followed by the original requirements, analysis, design and development. Then, the final tests of the system are specified and, at the end, the evaluation of the whole process is presented, carrying out the evaluation of the finished project.

Keywords

Digital signature, Angular, REST services, server, practices, component

Índice de contenido

Índice de ilustraciones

Índice de tablas

1. Introducción

En este apartado se realizará una introducción al proyecto, donde se mostrará una descripción global del proyecto y la razón de su creación.

1.1. Descripción del proyecto

Algunos centros educativos ofrecen la oportunidad de realizar prácticas voluntarias en una empresa para convalidar los créditos de la facultad, por las horas invertidas. De esta forma, el alumno se integra por unos meses en un lugar de trabajo donde poder llevar a la práctica y desarrollar los conocimientos adquiridos en el transcurso de su educación. Durante el proceso previo, el estudiante debe seleccionar entre las diferentes ofertas de las empresas, las que le resulten más atractivas teniendo en cuenta las funciones a desarrollar, accesibilidad, horario, etc… Una vez seleccionadas las ofertas que mejor se adecúen al perfil presentado, comienza el proceso de selección, mediante el cual, y tras una entrevista con el alumno, la empresa dará el visto bueno definitivo a su solicitud. Después de que el alumno se haya decidido por una, elegirá un profesor para que las tutorice y, tanto la instructor encargado de tutorizar sus prácticas en la empresa como el profesor, deben firmar el documento donde se detalla toda la información relacionada con las prácticas.

Embecas es una aplicación web que está pensada y diseñada para hacer la gestión lo más fácil e intuitiva posible. Esta aplicación posibilita la utilización de la firma digital para validar el documento sin la necesidad de conseguir la firma física de los participantes en el proceso. De esta manera, el proceso de firma será más rápido y cómodo.

Este sistema es accesible para todos los participantes del proceso de selección de prácticas, es decir, el alumno, el profesor encargado de tutorizar las prácticas en representación del centro, la empresa en la que se realiza el trabajo y el instructor que se encarga de supervisar al alumno.

Esta aplicación está diseñada tanto para ordenadores como para dispositivos móviles, para que se pueda realizar todo el proceso de selección en cualquier aparato. Así, si el alumno quiere consultar cual es el estado de su solicitud, no tiene porque encender el ordenador, puede realizar la consulta desde su teléfono. El único inconveniente, es que no podrá realizar la firma mediante el móvil, ya que es necesario tener la aplicación Idazki, que es el sistema que se encarga de la firma electrónica, descargada en la computadora.

1.2. Motivación

Recientemente he estado algo más de cinco meses haciendo prácticas voluntarias en una empresa, por lo que he podido comprobar cómo funciona tanto la aplicación de G.A.U.R., como el proceso de firmar el contrato una vez adjudicadas las prácticas. Tuve que seguir el siguiente procedimiento: una vez que la empresa me adjudicó el puesto, la profesora me firmó el documento impreso con los detalles de las prácticas, entregué el documento en secretaría, donde se me facilitaron tres copias donde se incluía la información de la tutora. Después, fui a la empresa a que el jefe del departamento firmara los tres acuerdos, para volver a la universidad y entregar uno de los contratos en secretaria, otro se lo quedó la empresa y la tercera copia fue para mí.

Todo el proceso necesario para la firma del documento me pareció muy largo y farragoso, teniendo en cuenta además, la necesidad de la disponibilidad de todos los participantes en el proceso. Entregar el documento y firmarlo no conlleva más de 5 minutos, pero la secretaría, la profesora y la empresa, tienen unos horarios que no permiten realizar todo el proceso de forma continuada, por lo que es posible que se necesiten varios días para llevar a cabo las gestiones necesarias. Además, muchas de las empresas no están cerca de la universidad y se necesita transporte público o coche para desplazarse de un sitio a otro, suponiendo una pérdida de tiempo y dinero.

Las circunstancias que he descrito tras mi experiencia personal, me llevaron a pensar que la firma digital era una alternativa óptima y eficaz para mejorar la lentitud del proceso. Esta tecnología se utiliza en muchas empresas y organizaciones, ofreciendo seguridad y confianza en el uso de documentos electrónicos. Asimismo, reduce el uso del papel, contribuyendo a la mejora del medio ambiente.

Por otro lado, en las prácticas formé parte de un grupo que está utilizando Angular para el desarrollo de una aplicación web, por lo que me gustaría aplicar todo lo aprendido para la realización del proyecto.

Por todo esto, mi mayor motivación para llevar a cabo este proyecto ha sido crear una aplicación web para hacer más cómodo el proceso de gestión de las prácticas voluntarias en la empresa, utilizando nuevas tecnologías.

2. Planteamiento inicial

En este capítulo se presentan los objetivos del proyecto, así como todas las fases y tareas que forman parte del desarrollo y su planificación inicial. Además, se incluyen todas las herramientas que se utilizarán, los riesgos que pueden ocurrir durante el proceso y el coste total que supone la creación de la aplicación.

2.1. Objetivos

Este proyecto tiene tres objetivos principales. En primer lugar desarrollar una aplicación web accesible para las partes involucradas en el proceso de las prácticas, es decir, el alumno, el profesor y la empresa. Además, incluir funcionalidades que pueden ser útiles para cada participante, como pueden ser el historial o la opción de subir el currículum del alumno. De esta forma, cada uno tendrá acceso a diferente información relacionada con su implicación en las prácticas en cualquier momento. También, teniendo los tres usuarios en la plataforma, se facilita el proceso de firma del contrato, ya que todos tendrán acceso al documento.

Por otra parte, será necesaria la autorización de todos los implicados para la firma del contrato, en el cual se especificarán todos los detalles relacionados con las prácticas. Para ello, el objetivo es utilizar la firma digital, agilizando mucho el proceso y simplificando al alumno la tarea de llevar al profesor y a la empresa el contrato para firmar.

Por último, y no menos importante, está el diseño de la página. Hay que tener en cuenta que la página debe tener un buen diseño para que sea fácil de utilizar y sea rápido determinar qué función tiene cada elemento de la pantalla. Por todo esto, el objetivo final es conseguir que la interacción con la aplicación sea sencilla y se haga de forma intuitiva. Además, que sea accesible tanto para ordenadores como dispositivos móviles.

2.2. Alcance

En este apartado se definen todas las etapas en las que se ha dividido el desarrollo del proyecto y sus tareas correspondientes. A continuación, se pueden ver todas ellas en la Estructura de Descomposición de Trabajo o EDT.

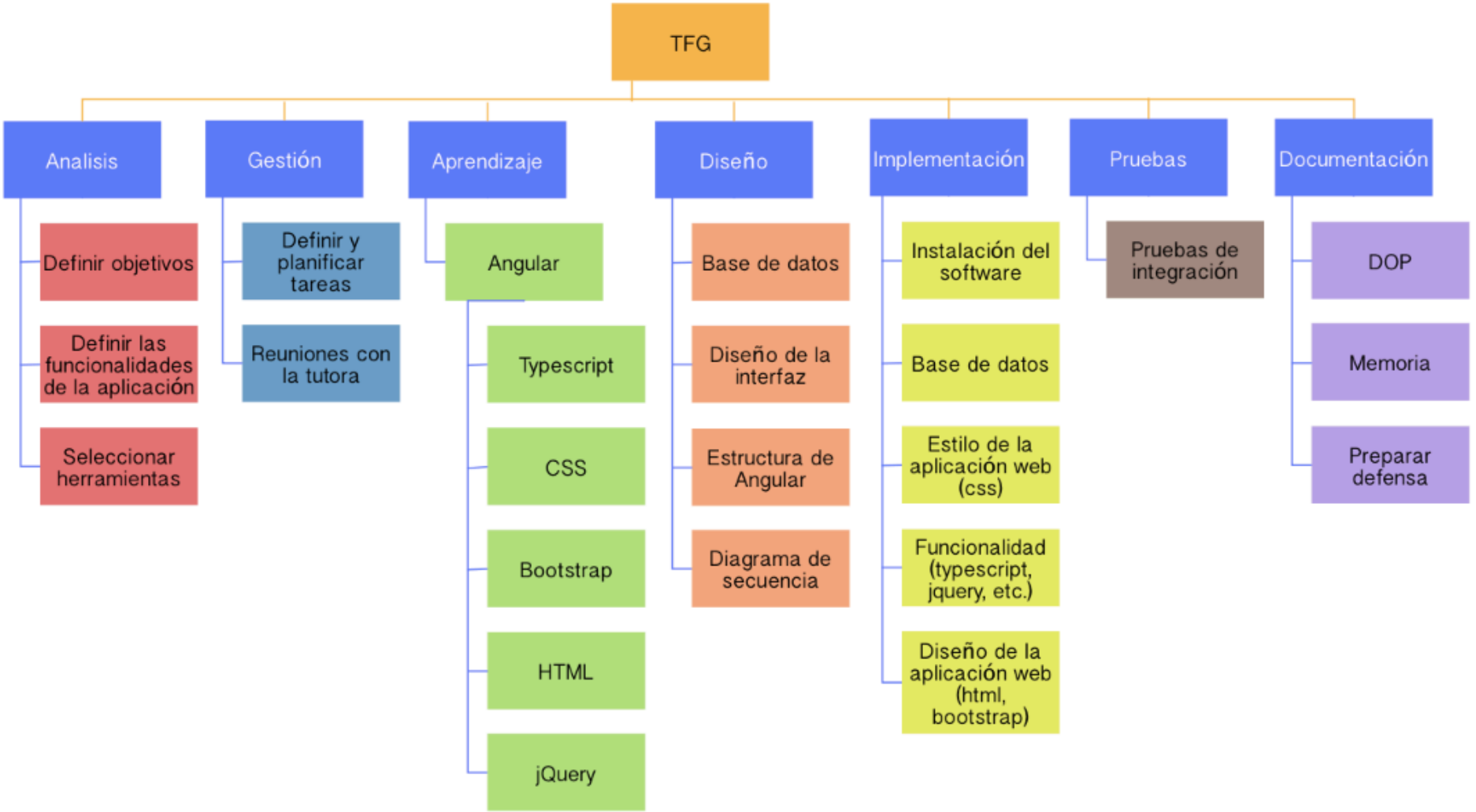

Ilustración 1. EDT del proyecto.

Estas son las fases por las que ha pasado este proyecto:

- **Análisis:** En esta etapa se define la aplicación, es decir, las herramientas que se utilizarán para su creación, todas las funcionalidades que tendrá y sus objetivos principales.

- **Gestión:** Esta es una de las fases más importantes del proyecto, ya que es la encargada de definir y gestionar todas las tareas a realizar. Cuanto más detallada sea la planificación, más fácil será su correcto desarrollo. Aunque sea una de las primeras fases que se lleve a cabo, se alargará durante todo el proceso, puesto que existe la posibilidad de que la planificación inicial y las tareas se modifiquen a medida que avance el trabajo. Además, las reuniones con el tutor se harán durante todo el proceso.

- **Aprendizaje:** Es la encargada del estudio de todas las herramientas que se utilizarán a lo largo del proyecto. Puesto que es la primera vez que crearé una aplicación web con Angular, es importante conocer bien su alcance y todas las opciones que ofrece.

- **Diseño:** Una vez llegados a esta fase, hay que especificar cuál será el diseño de la aplicación, tanto la parte visible que serán las pantallas, como las estructuras de datos.

- **Implementación:** Después de determinar cuál será el diseño de la aplicación, comienza el proceso de implementación, donde se llevará a cabo toda la escritura del código. Esta es la parte más larga y complicada del proyecto, ya que consiste en darle vida a todas las ideas y funcionalidades que la definen.

- **Pruebas:** El plan de pruebas consiste en comprobar que todas las funcionalidades implementadas funcionan correctamente.

- **Documentación:** Está fase se desarrollará durante todo el proyecto, trabajando en paralelo tanto en el desarrollo como en la documentación. De esta forma, todo lo que se vaya resolviendo, quedará plasmado en la memoria. En esta fase también se incluirá la preparación de la defensa.

A continuación, se detallan las tareas de cada grupo y el tiempo estimado para cada una de ellas:

- **ANÁLISIS:**

Definir objetivos

Paquete de trabajo: Análisis.
Duración: 5 horas.
Descripción: Se define los objetivos principales que quieren obtenerse con la realización del proyecto.
Salidas / entregables: Planteamiento de los objetivos que se quieren obtener.
Recursos necesarios: Ordenador y conexión a internet.

Definir las funcionalidades de la aplicación

Paquete de trabajo: Análisis.
Duración: 4 horas.
Descripción: Se definen las funcionalidades que tendrá la aplicación, en este caso, todas las acciones que podrán realizar los alumnos, profesores y la empresa.
Salidas / entregables: Listado de acciones que podrá realizar cada participante.
Recursos necesarios: Ordenador con conexión a internet, papel y boli.

Seleccionar herramientas

Paquete de trabajo: Análisis.
Duración: 10 horas.
Descripción: Investigar a cerca de Angular, cómo puede conectarse a una base de datos y cómo puede implementarse la firma digital.
Salidas / entregables: Conocimiento sobre todas las herramientas a utilizar.
Recursos necesarios: Ordenador y conexión a internet.

- **GESTIÓN:**

Definir y planificar tareas

Paquete de trabajo: Gestión.
Duración: 5 horas.
Descripción: Se creará la planificación del proyecto, definiendo las diferentes fases por las que pasará y las tareas a realizar en cada una de ellas.
Salidas / entregables: Diagrama de EDT y gantt
Recursos necesarios: Cacoo y GanttProject

Reuniones con la tutora

Paquete de trabajo: Gestión.
Duración: 8 horas.
Descripción: Se realizarán reuniones con la tutora para llevar un control sobre la planificación y realización del proyecto y establecer los cambios que sean necesarios.
Salidas / entregables: Posibles cambios a realizar sobre el proyecto.
Recursos necesarios: Despacho de la tutora, email o video llamada.

- **APRENDIZAJE:**

Typescript

Paquete de trabajo: Aprendizaje.
Duración: 18 horas.
Descripción: Aprender Typescript para las funcionalidades de la aplicación.
Salidas / entregables: Conocimiento sobre TypeScript.
Recursos necesarios: Ordenador con conexión a internet para consultar foros, videos, cursos,...

CSS

Paquete de trabajo: Aprendizaje.
Duración: 12 horas.
Descripción: Aprender CSS para los estilos de la aplicación web.
Salidas / entregables: Conocimiento sobre CSS.
Recursos necesarios: Ordenador con conexión a internet para consultar foros, videos, cursos,...

Bootstrap[3]

Paquete de trabajo: Aprendizaje.
Duración: 11 horas.
Descripción: Aprender bootstrap para el diseño y la estructura de la parte front-end [**1]** de la aplicación.
Salidas / entregables: Conocimiento sobre bootstrap **[8].**
Recursos necesarios: Ordenador con conexión a internet para consultar foros, videos, cursos,...

HTML

Paquete de trabajo: Aprendizaje.
Duración: 15 horas.
Descripción: Aprender HTML para el diseño de la página web.
Salidas / entregables: Conocimiento sobre HTML **[10]**.
Recursos necesarios: Ordenador con conexión a internet para consultar foros, videos, cursos,...

jQuery

Paquete de trabajo: Aprendizaje.
Duración: 14 horas.
Descripción: Aprender jQuery para interactuar con los elementos HTML.
Salidas / entregables: Conocimiento sobre jQuery.
Recursos necesarios: Ordenador con conexión a internet para consultar foros, videos, cursos,...

- **DISEÑO:**

Base de datos

Paquete de trabajo: Diseño.
Duración: 3 horas.
Descripción: Diseñar la base de datos con los datos y relaciones adecuadas. Es importante tener claro cuáles serán esos datos, ya que puede conllevar mucho tiempo y esfuerzo tener que rehacerla a mitad del proyecto por no haberla diseñado bien desde el principio.
Salidas / entregables: Planteamiento de todas las clases, datos y relaciones necesarios.
Recursos necesarios: Papel y bolígrafo.

Diseño de la interfaz

Paquete de trabajo: Diseño.
Duración: 10 horas.
Descripción: Pensar cuáles serán las pantallas que tendrá la aplicación, su diseño, que datos aparecerán en cada una y cuáles serán las conexiones que tendrán entre ellas.
Salidas / entregables: Boceto de todas las pantallas.
Recursos necesarios: Ordenador con conexión a internet, papel y bolígrafo.

Estructura en Angular

Paquete de trabajo: Diseño.
Duración: 12 horas.
Descripción: Diseñar qué componentes y servicios tendrá el programa, donde se guardarán los datos comunes para tenerlos accesibles, etc.
Salidas / entregables: Una buena estructura para empezar a trabajar.
Recursos necesarios: Visual Studio Code.

Diagrama de secuencia

Paquete de trabajo: Diseño.
Duración: 15 horas.
Descripción: Realizar el diagrama de secuencia correspondiente a la aplicación teniendo en cuenta la estructura de Angular y las funcionalidades implementadas.
Salidas / entregables: Diagrama de secuencia.
Recursos necesarios: Cacoo.

- **IMPLEMENTACIÓN:**

Instalación del software

Paquete de trabajo: Implementación.
Duración: 4 horas.
Descripción: Instalar todo el software necesario para realizar el proyecto.
Salidas / entregables: Instalación de las herramientas necesarias.
Recursos necesarios: Ordenador y conexión a internet.

Base de datos

Paquete de trabajo: Implementación.
Duración: 5 horas.
Descripción: Implementar la base de datos, con todas las clases, datos y relaciones correspondientes.
Salidas / entregables: La base de datos.
Recursos necesarios: MySQLWorkbench

Estilo de la aplicación web (css)

Paquete de trabajo: Implementación.
Duración: 40 horas.
Descripción: Implementar una hoja de estilos con los colores, tamaños, márgenes, etc. que se le quieran aplicar a los elementos de las pantallas.
Salidas / entregables: Una hoja de estilos.
Recursos necesarios: Visual Studio Code.

Funcionalidades (typescript y jQuery)

Paquete de trabajo: Implementación.
Duración: 100 horas.
Descripción: Implementar todas las funcionalidades de la aplicación.
Salidas / entregables: Todas las funcionalidades implementadas.
Recursos necesarios: Visual Studio Code.

Diseño de la aplicación web (html y bootstrap)

Paquete de trabajo: Implementación.
Duración: 100 horas.
Descripción: Implementar todas las estructuras de datos (formularios, botones, checkbox,...) para mostrar la información.
Salidas / entregables: Archivos .html.
Recursos necesarios: Visual Studio Code.

- **PRUEBAS:**

Pruebas de integración

Paquete de trabajo: Pruebas.
Duración: 8 horas.
Descripción: Durante toda la implementación de la aplicación se realizarán pruebas para comprobar que todo funciona como debería y no surgen errores inesperados o no tratados.
Salidas / entregables: Tablas de pruebas.
Recursos necesarios: Visual Studio Code.

- **DOCUMENTACIÓN:**

DOP

Paquete de trabajo: Documentación.
Duración: 16 horas.
Descripción: Escribir el DOP, es decir, definir los objetivos del proyecto. Esta tarea se realiza nada más empezar el proyecto.
Salidas / entregables: DOP.
Recursos necesarios: Microsoft Word.

Memoria

Paquete de trabajo: Documentación.
Duración: 110 horas.
Descripción: Escribir la memoria del proyecto, donde se recopile todo el trabajo realizado durante el desarrollo de la aplicación, incluyendo toda la documentación reunida, así como todas las tareas, experiencias, diagramas,...
Salidas / entregables: La memoria.
Recursos necesarios: Microsoft Word.

Preparar defensa

Paquete de trabajo: Documentación.
Duración: 20 horas.
Descripción: Preparar defensa del proyecto.
Salidas / entregables: La defensa.
Recursos necesarios: La memoria, el proyecto y PowerPoint.

2.3. Planificación temporal

En esta sección se puede ver cuál será la duración total del proyecto sumando el tiempo estimado para cada tarea y una planificación semanal plasmado en un diagrama de Gantt.

Tarea	Duración estimada (horas)
Análisis	**19**
Definir objetivos	5
Definir las funcionalidades de la aplicación	4
Seleccionar herramientas	10
Gestión	**13**
Definir y planificar tareas	5
Reuniones con la tutora	8
Aprendizaje	**70**
Typescript	18
Css	12
Bootstrap	11
Html	15
jQuery	14

Diseño	**40**
Base de datos	3
Diseño de la interfaz	10
Estructura en Angular / diagrama de clases	12
Diagrama de secuencia	15
Implementación	**249**
Instalación del software	4
Base de datos	5
Estilo de la aplicación web (css)	40
Funcionalidades (Typescript y jquery)	100
Diseño de la aplicación web (html y bootstrap)	100
Pruebas	8
Pruebas finales	8
Documentación	**146**
DOP	16
Memoria	110
Preparar defensa	20
TOTAL	**532 horas**

Tabla 1. Planificación temporal

Estimando que se tardará un total de **532 horas** que equivalen a **15 semanas** de trabajo, suponiendo que se trabajará una media de 7 horas diarias, el proyecto debería de estar finalizado para el 12 de Julio.

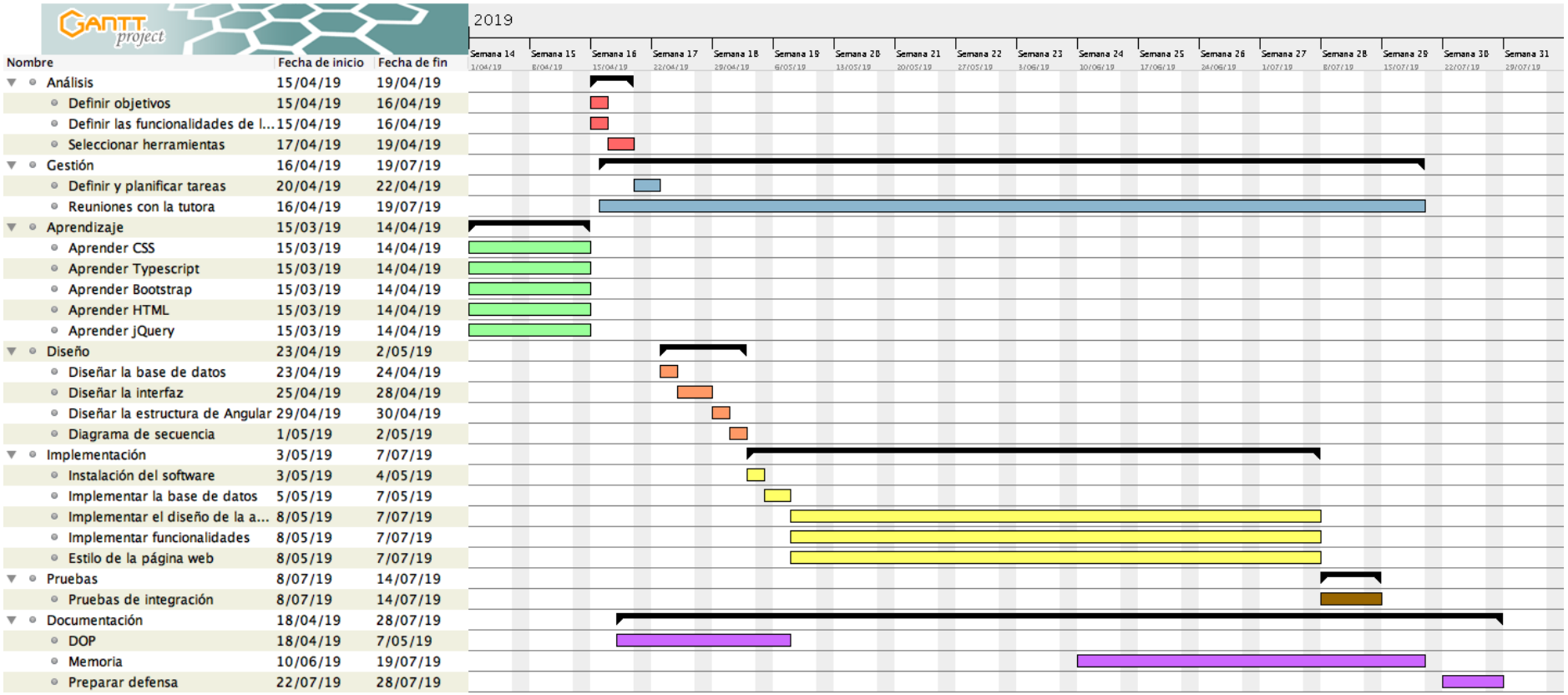

Ilustración 2. Diagrama de Gant

2.4. Arquitectura

Este proyecto tendrá una arquitectura de tipo cliente-servidor. El trabajo se divide en dos tareas independientes: Por un lado se encuentra el servidor [5] que se encarga de conectar con la base de datos y exponer recursos o servicios y, por otro lado, la aplicación que consume los servicios proporcionados por el servidor. Gracias a esta arquitectura se consigue mantener separada la interfaz del usuario de la parte lógica.

El cliente ejecutará la aplicación desde el navegador y usará la interfaz gráfica para navegar por la web y solicitar cierta información. Para conseguir esos datos, la aplicación accederá a los datos a través de unos servicios expuestos en el servidor.

Ilustración 3. Arquitectura del proyecto

2.5. Herramientas

En este apartado se explican todas las herramientas utilizadas para realizar este proyecto:

- **Visual Studio Code:** Visual Studio Code es un editor de código que se usa, entre otras cosas, para la programación web con HTML, CSS y JavaScript. Esta herramienta se utilizará para implementar la aplicación [7].

Ilustración 4. Logo de Visual Studio Code

- **Cacoo:** Cacoo es una aplicación en la nube que permite crear diagramas online y poder elaborarlos con otros usuarios, pudiendo editarlo al mismo tiempo. Esta herramienta se usará para crear el EDT, el diagrama de la base de datos, el diagrama de secuencia y todos los diagramas personalizados.

Ilustración 5. Logo de Cacoo

- **GanttProject:** Es una aplicación que permite organizar proyectos a través de diagramas de Gantt, así como la carga horaria diaria. Es una aplicación open source, por lo que se puede descargar de forma gratuita y además es una

aplicación íntegramente escrita en Java por lo que lo hace multiplataforma. Esta herramienta se utilizará para crear el diagrama de Gantt.

Ilustración 6. Logo de GanttProject

- **Node.js:** Es un entorno de ejecución para JavaScript que contiene npm **[9]** como sistema de gestor de paquetes, que sirve para gestionar los paquetes y dependencias en el lado del cliente, por lo que es indispensable para el desarrollo web Front-end **[1]**. Es el entorno en el que se programará la aplicación web.

Ilustración 7. Logo de node.js

- **Apache Tomcat:** Es un servidor web gratuito y de código abierto que permite a los propietarios de sitios web servir contenido en la web.

Ilustración 8. Logo de Apache Tomcat

- **Postman:** Es una aplicación que permite realizar diferentes tareas dentro del mundo API REST, como la creación de peticiones a APIs o elaboración de test para validar su comportamiento. Gracias a esta herramienta, se creará una colección con todas las peticiones API del proyecto, para poder probarlas de manera rápida y sencilla.

Ilustración 9. Logo de Postman

- **Eclipse:** Es un entorno de desarrollo integrado (IDE), de código abierto y multiplataforma. Se utiliza mayormente para el desarrollo de sitios web, programas en C++ o aplicaciones Java. Esta herramienta se utilizará para crear los servicios API REST, creando una conexión entre la base de datos.

Ilustración 10. Logo de Eclipse

- **MySQL Workbench:** Es una herramienta visual de diseño de bases de datos que permite diseñar visualmente, modelar, generar y administrar bases de datos. Esta aplicación que utilizará para el diseño e implementación de la base de datos.

Ilustración 11. Logo de MySQL Workbench

- **Google drive:** Es un servidor de documentos que permite guardar ficheros en internet y acceder a ellos desde cualquier dispositivo conectado a internet. Esta herramienta permite crear presentaciones, documentos, hojas de cálculo, etc. y compartirlos con otros usuarios. Además, admite la edición de varios usuarios al mismo tiempo. Esta herramienta se utilizará para crear copias de seguridad tanto de la implementación como de la documentación para tener el código actualizado en todo momento y tenerlo accesible desde diferentes dispositivos. También se creará la presentación usando esta aplicación.

Ilustración 12. Logo de Google Drive

- **Microsoft Word:** Es un procesador de texto, es decir, es un software que permite la creación y edición de documentos de texto. Este programa se usará para crear la memoria del proyecto.

Ilustración 13. Logo de Microsoft Word

- **Microsoft PowerPoint:** Es un software que permite realizar presentaciones a través de diapositivas. El programa contempla la posibilidad de utilizar textos, imágenes, música y animaciones. El programa se usará para crear la presentación para la defensa del proyecto.

Ilustración 14. Logo de Microsoft PowerPoint

- **Idazki Desktop:** Esta aplicación de Izenpe (empresa de certificados) permite la firma electrónica de documentos. Se utilizará la librería **idazki.js** para poder interactuar con esta plataforma a través de nuestra aplicación. Esta herramienta se usará para firmar el contrato final mediante certificados personales **[11]**.

Ilustración 15. Logo de Idazki Desktop

2.6. Gestión de riesgos

Todo proyecto conlleva una serie de riesgos que hay que identificar y saber gestionar, para asegurarse de que afectan lo menos posible a su desarrollo. En este apartado se evalúan algunos de esos peligros que pueden surgir durante la ejecución del proyecto, evaluando en qué medida afectarían negativamente, como preverlos, cuál sería la forma de actuar si esto ocurriera y cuál es la posibilidad de que pasen.

Antes de empezar, se clasificarán los riesgos en función de la probabilidad de que ocurran:

Probabilidad	Porcentaje
Baja	0% - 30%
Media	30% - 70%
Alta	70% - 100%

Tabla 2. Clasificación de probabilidades

Además, hay que tener en cuenta el impacto que tendrán estos riesgos en el proyecto, ya que, mientras que algunos no afectarán a la planificación, otros pueden suponen un gran retraso.

Impacto	Retraso
Bajo	1 - 2 días
Medio	3 - 5 días
Alto	5 - 10 días
Muy alto	Varias semanas

Tabla 3. Clasificación del impacto

A continuación, de detallan algunos de los riesgos más significativos:

Enfermedad o indisponibilidad

Descripción	Cualquier enfermedad o razón personal que pueda hacer que estés indispuesto para realizar el trabajo planeado para ese día.
Prevención	Intentar llevar a cabo la planificación inicial en todo lo posible y dejar unos días de margen por si ocurriera alguna incidencia inesperada. Además, evitar cualquier riesgo innecesario que pueda afectar a la salud o al estado físico.
Plan de contingencia	Recuperarse de la enfermedad o indisponibilidad y aumentar el tiempo de trabajo cada día para recompensar esas horas perdidas.
Probabilidad	Baja
Impacto	Bajo

Tabla 4. Enfermedad o indisponibilidad

Pérdida de documentación o código

Descripción	Debido a un fallo en el equipo o por un error no premeditado, se puede perder parte del trabajo realizado.
Prevención	Hacer copias de seguridad diariamente tanto de la documentación como del código.
Plan de contingencia	Rehacer el código o la documentación lo antes posible para tener más reciente el trabajo. Y, en caso de que la pérdida fuese lo suficientemente grande como para no poder solucionarlo en un plazo de tiempo corto, rehacer la planificación. .
Probabilidad	Media
Impacto	Alto

Tabla 5. Pérdida de documentación o código

Problemas a la hora de la implementación y el diseño

Descripción	En caso de tener problemas a la hora de diseñar una parte del código y no conseguir sacarlo adelante a pesar de llevar mucho tiempo intentándolo.
Prevención	Llevar a cabo la tarea en un lugar tranquilo, sin distracciones, e intentar lo dejarse llevar por el estrés o el bloqueo momentáneo.
Plan de contingencia	En caso de quedarse bloqueado, aplazar esa tarea para otro momento y descansar. En caso de querer continuar trabajando, continuar con una tarea de menor tensión.
Probabilidad	Media
Impacto	Medio

Tabla 6. Bloqueo a la hora de la implementación y el diseño

Error con la compilación de Angular

Descripción	Al ser una tecnología nueva que no he usado hasta ahora, puede resultar más complicado identificar los errores de compilación que puedan surgir durante la implementación. Además, emplear librerías externas de Angular puede ocasionar problemas por no saber cómo utilizarlas.
Prevención	Prestar atención a la implementación de todos los módulos y componentes de la aplicación para prevenir los errores de compilación e investigar la utilización de las librerías antes de usarlas sin ningún conocimiento.
Plan de contingencia	Buscar toda la información posible sobre el problema con el que nos encontremos durante la ejecución de la aplicación.
Probabilidad	Alta
Impacto	Medio

Tabla 7. Error con la compilación de Angular

Mala planificación por falta de conocimiento

Descripción	Es importante hacer una buena planificación de todas las tareas que se deben llevar a cabo y el tiempo necesario para ellas, porque puede que algunas requieran más tiempo de lo esperado inicialmente. Además, al trabajar con nueva tecnología, de la que se va aprendiendo a medida que avanza el proyecto, es posible que nos cueste más desarrollar ciertas partes del código, por lo que esa falta de conocimiento puede afectar a la hora de implementar ciertas partes del proyecto. Todo ello puede ocasionar el aumento del tiempo estimado para la formación, lo cual afectaría al tiempo total del proyecto y podría retrasarlo.
Prevención	Aprovechar al máximo todo el tiempo destinado al aprendizaje y estudiar en las horas libres que no están incluidas en el desarrollo del proyecto. También, detallar bien todas las tareas y subtareas y ser generoso a la hora de estimar el tiempo necesario para cada una.
Plan de contingencia	En caso de necesitar más tiempo para ciertas tareas, de volverá a planificar el tiempo, siempre intentando estar dentro de la fecha de entrega pensada.
Probabilidad	Alta
Impacto	Muy alto

Tabla 8. Mala planificación por falta de conocimiento

2.7. Evaluación económica

Al tratarse de un trabajo de fin de grado para la universidad, no se espera ningún beneficio económico por su realización. Sin embargo, todo proyecto supone un coste económico que podemos calcular teniendo en cuenta los costes fijos como la mano de obra del programador y el software y hardware utilizados y otros costes variables como la luz y el acceso a internet.

- **Mano de obra:**

Para calcular el coste de mano de obra de este proyecto, se ha estimado que la media salarial de un programador junior es de 25€/hora por jornada completa.

Este proyecto lo llevará a cabo una sola persona y, teniendo en cuenta que está realizando prácticas en empresa por las mañanas, le dedicará una media de 30 horas semanales, incluyendo también los fines de semana.

Considerando el precio por hora y las horas semanales calculadas, se estima un salario de 3.000€ mensuales. Se estima que se necesitan 4 meses para realizar este proyecto, por lo que la mano de obra final será de 1**2.000€**.

- **Gastos de software:**

Todas las aplicaciones necesarias para este proyecto son de software libre, por lo que no añadiremos gastos adicionales por esta parte. El sistema operativo del ordenador en cambio, es de pago, pero se incluirá en el precio del ordenador, especificado en el siguiente apartado.

- **Gastos de hardware:**

El ordenador utilizado es un MacBook Air, con un precio de 1147€ el día de su compra. Estimando que la duración media del ordenador es de 5 años, calcularemos el precio de amortización del equipo:

1147€ / 5 años / 12 meses = 19,12€ / mes

Teniendo en cuenta que la duración del proyecto es de 4 meses, el precio del portátil serían **76,48€.**

- **Gastos de luz:**

Suponiendo que trabajaremos en casa con la compañía eléctrica de Iberdrola, el precio de la luz es de 0,18€/kWh **[12]**. Según lo arriba especificado, trabajaremos un total de 20 horas semanales, que equivalen a 320 horas en los 4 meses de trabajo, por lo que el precio de la luz ascendería a **57,6€**.

- **Gastos totales:**

A continuación, se mostrará una tabla con los costes calculados en cada apartado y el precio total del proyecto:

Tipo de gasto	Gasto ocasionado
Mano de obra	12.000€
Gastos de software	0€
Gastos de hardware	76,48€
Luz	57,60€
Gastos totales:	**12.134,08€**

Tabla 9. Evaluación económica

Por lo que el precio total por el desarrollo de la aplicación sería de **12.134,08€**.

3. Análisis de antecedentes

La idea de esta aplicación surgió como posible mejora del sistema que se utiliza actualmente en el proceso de selección de las prácticas en la universidad de la UPV, Praktiges. Después de conocer y utilizar dicha aplicación, se detectaron algunas carencias y vulnerabilidades que se trata de mejorar mediante este proyecto. La aplicación existente en la actualidad es similar a la que se presenta en este proyecto en varias características, por ejemplo, filtrar las prácticas ofertadas por provincia y localidad o ver los datos de la adjudicación, aunque no en algunas otras, por ejemplo, la oportunidad de firmar el documento de forma digital sin necesidad de la firma física o que los profesores tengan acceso a la información de las ofertas. En esta sección se van a detallar las características de la aplicación que está actualmente en funcionamiento.

3.1. Praktiges

Praktiges es una aplicación desarrollada para la universidad de la UPV que gestiona el procedimiento de selección y adjudicación de las prácticas en empresa. Tanto el alumno como la empresa y el profesor tienen acceso a la información relacionada con el acuerdo realizado.

Por su parte, el alumno puede acceder a su perfil de la universidad y crear una especie de currículum vitae. Para ello, debe rellenar los apartados correspondientes al currículum a partir de una plantilla preseleccionada con los campos que debe rellenar. Esto, dependiendo de la información que se quiera facilitar, ofrece muy poco margen de maniobra, ya que no se puede incluir ningún detalle que no esté ya especificado en la tabla para completar, por ejemplo, proyectos realizados dentro y fuera del centro.

Por otro lado, según la experiencia que tuve con la aplicación, estas son algunas de las características que tiene el proceso de selección: El alumno puede seleccionar varias ofertas de prácticas y a partir de ese momento, la empresa se pone en contacto con el alumno. Existe una lista de preselecciones, en la cual no figura ninguna entrada hasta

que el alumno no ha sido aceptado en alguna de las ofertas elegidas. Esto conlleva a la incertidumbre de no saber el estado en el que se encuentra dicha oferta, para saber si la opción de optar a ese puesto sigue en pie o si ya has sido rechazado. En lo referente a las adjudicaciones, la empresa tiene la opción de asignarle al alumno la oferta directamente, lo que supone que el alumno no tiene la última palabra en lo relativo a la adjudicación. A partir de ese momento, en el apartado de adjudicaciones se muestra una entrada nueva y, como consecuencia, no le permite al alumno apuntarse a nuevas ofertas. Esto puede confundir al alumno, ya que él se espera ser el que tenga la decisión final a través de la aplicación y no la empresa. En mi caso no llegué a ver ningún apartado en la lista de preselecciones, a pesar de que me habían aceptado en dos ofertas, y me asignaron una de ellas sin que yo les comunicase mi decisión final.

En lo relacionado con el profesor, puede acceder a la información de las tutorizaciones a partir de la búsqueda por cursos académicos y ver toda la información del acuerdo. El profesor no tiene decisión sobre la tutorización de las prácticas a través de la aplicación, lo hace de forma personal cuando firma el acuerdo con la empresa, y secretaria realiza el trámite. A partir de ese momento, la aplicación tiene todo lo relacionado con el acuerdo.

Desde otro punto de vista, la empresa puede añadir nuevas ofertas y ver todos los datos relacionados con la selección de alumnos, tanto las peticiones, como las asignaciones y rechazos. Además, puede añadir nuevos instructores y ver los detalles de ofertas realizadas en otros cursos académicos.

En lo correspondiente al proceso de firmar el contrato con los detalles del acuerdo final, hay que seguir el siguiente procedimiento: una vez que la empresa adjudica el puesto al alumno, el profesor tiene que firmar el documento impreso con los detalles de las prácticas, después hay que entregar el documento en secretaría, donde se facilitan tres copias donde se incluye la información de la tutora. A continuación, en la empresa, el jefe del departamento firma los tres acuerdos, y luego hay que volver a la universidad y entregar uno de los contratos en secretaria, mientras que el segundo documento se lo queda la empresa y el otro el alumno.

Como particularidad de la aplicación, decir que no está adaptada a dispositivos móviles, por lo que se dificulta su uso a través de teléfonos, ya que la vista es la misma que en el ordenador y al ampliarla se pierde visibilidad y la interacción con los botones y entradas se complica.

4. Captura de requisitos y análisis

En esta sección, se van a especificar los diferentes requisitos a tener en cuenta para un buen funcionamiento de la aplicación y se identificarán también todos los usuarios que podrán acceder al sistema. Es un paso fundamental en el desarrollo del proyecto, ya que consiste en identificar y satisfacer en todo lo posible las necesidades de todos los tipos de usuarios.

4.1. Requisitos previos

Al analizar el funcionamiento de la aplicación actual para la gestión de las prácticas, se detectaron varias funciones que debía tener este proyecto:

- Una aplicación que identifique a todos los usuarios del sistema. En este caso serán los alumnos, profesores tutores y, de parte de la empresa, el representante de recursos humanos y los instructores.

- Una aplicación accesible tanto para los alumnos como para profesores y empresas, con la que puedan interactuar y tener acceso a la información relacionada con las prácticas en cualquier momento.

- Un historial de prácticas, tanto para el profesor como para la empresa. En el caso del profesor, para llevar un registro de todos los alumnos a los que les ha tutorizado las prácticas y, en el caso de la empresa, todos los alumnos que han participado en la experiencia y que han formado parte de la empresa en algún momento.

- Una forma de firmar el contrato final sin tener que necesitar la firma física de cada involucrado. Así, el proceso será mucho más rápido y cómodo.

- Un apartado para el alumno donde pueda adjuntar su currículum, así la empresa podrá tener información personalizada sobre el tipo de alumno que quiere realizar las prácticas.

- Una aplicación que sea intuitiva, en la que sea fácil saber cómo funciona y los pasos a dar para conseguir lo que se desea. Además, estará adaptada tanto para ordenadores como para dispositivos móviles.

- Un procedimiento de adjudicación donde el alumno esté al corriente en todo momento del estado en el que se encuentran sus selecciones de ofertas, para que sepa si todavía se encuentra con posibilidades de optar al puesto, o de lo contrario, ya ha sido rechazado.

- Un procedimiento de adjudicación donde el alumno sea el que tenga la última palabra y que, a pesar de que haya sido aceptado por una empresa, sea el estudiante el que dé su visto bueno para la adjudicación. Porque, puede ocurrir que el alumno sea aceptado en varias empresas, por lo que ninguna empresa puede adjudicar el puesto hasta que el alumno sea quien decida la oferta final que desea realizar.

4.2. Jerarquía de actores

A continuación, se puede observar la jerarquía de actores, que demuestra todos los actores que pueden actuar sobre el sistema. Se presenta en la Ilustración 16. Jerarquía de actores.

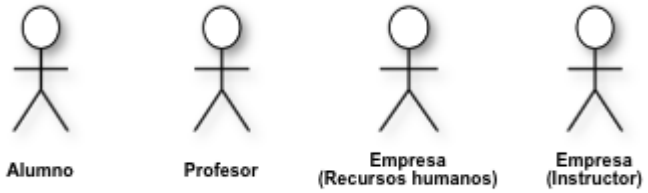

Ilustración 16. Jerarquía de actores

- **Alumno:** Todo aquel alumno que tenga acceso a la plataforma del centro educativo, por lo tanto un usuario y contraseña para acceder a dicha plataforma, y esté interesado en realizar prácticas en una empresa.

- **Profesor:** Todos los profesores con usuario y contraseña con acceso a la plataforma del centro educativo, que hayan tutorizado algunas prácticas o tengan intención de hacerlo.

- **Empresa (Recursos humanos):** Toda aquella empresa que tenga relación con el centro educativo y quiera registrarse en la aplicación para ofrecer un puesto de prácticas a alumnos del centro.

- **Empresa (instructor):** Todo aquel trabajador de las empresas, que tenga la autoridad para ejercer de instructor de alumnos que realicen las prácticas.

5. Diseño

Procedemos a describir la sección de diseño. El objetivo es utilizar las ideas recogidas en la captura de requisitos y diseñar un funcionamiento y flujo de trabajo aplicables al entorno de programación. Esta fase es fundamental, debido a que es la fase previa al desarrollo, y sirve como base para construir una estructura de trabajo con la que se pueda llevar a cabo todo lo planteado inicialmente de una forma organizada. Teniendo en cuenta los actores identificados en el apartado anterior, se explicarán los casos de uso para cada uno se ellos. Además, se explicará en modelo de dominio correspondiente a este proyecto y se diseñará una base de datos con todas las entidades necesarias, la correspondiente información de cada una y sus relaciones. Por último, se planteará un diagrama de secuencia donde se podrá observar el flujo de llamadas entre las diferentes clases que componen el sistema.

5.1. Diagrama de estados

En la siguiente Ilustración 17. Diagrama de estados, se puede ver el diagrama de estados, que representa todas las condiciones por las que puede pasar la preselección de una oferta de prácticas.

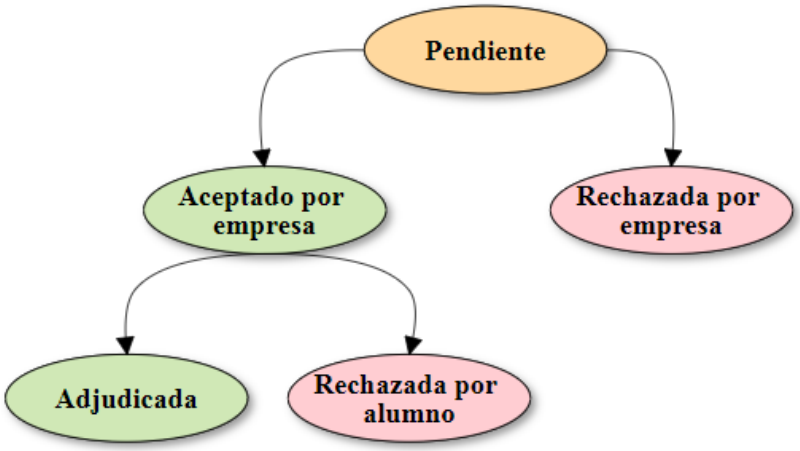

Ilustración 17. Diagrama de estados

Desde el momento en el que un alumno selecciona una oferta que le atrae y se incluye a la lista de preselecciones, el estado de esta pasa a ser “pendiente”, ya que se encuentra en trámite de decisión. Después, cuando se le notifica a la empresa la información del alumno que está interesado en su oferta, esta tiene la opción de rechazar o aceptar a un alumno en dicha práctica. En caso de tomar la decisión de rechazar, el estado pasa a ser “rechazado por empresa”. Esa decisión no tiene vuelta atrás. Si le gustaría contratar a ese alumno, en cambio, el estado cambiaría a “aceptado por empresa”. Desde este momento, el testigo se le pasa al alumno, puesto que tendrá la última palabra en cuanto a la elección de la práctica. El estudiante solo puede elegir una de las ofertas en las que está aceptado por la empresa. Una vez tome su decisión final, el estado de esa oferta pasa a “adjudicada”. Todas las demás ofertas de ese alumno pasarán a ser “rechazado por alumno”.

5.2. Casos de uso

Tras detectar los actores que tendrán acceso al sistema, los casos de uso representan todas las actividades que podrán realizar dichos usuarios dentro de la aplicación. Esas actividades son extraídas de los requisitos detectados anteriormente.

Para el alumno existen los casos de uso que se muestran en la Ilustración 18. Casos de uso alumno:

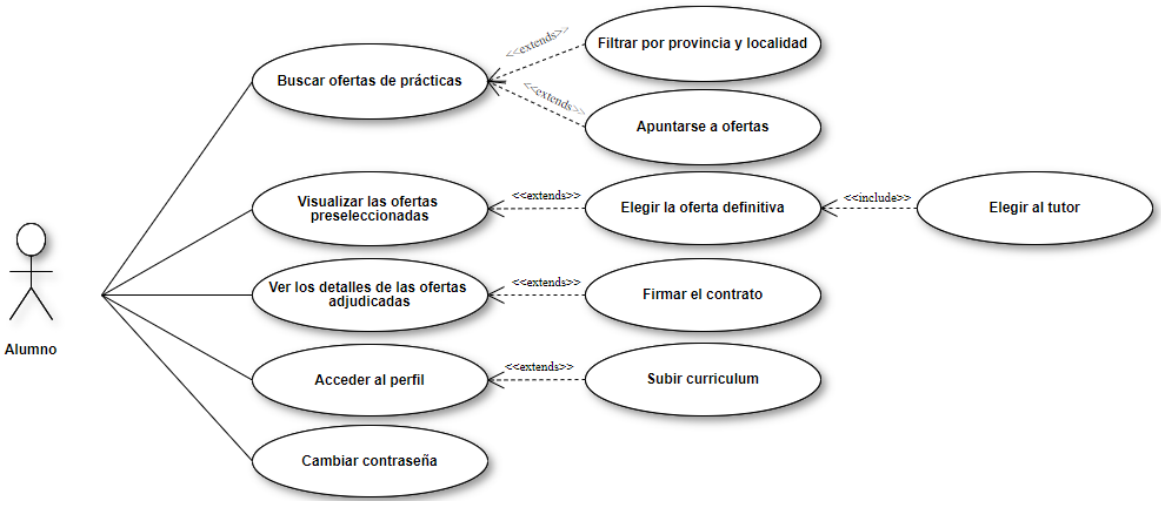

Ilustración 18. Casos de uso alumno

- **Buscar ofertas de prácticas:** Al entrar en el sistema, lo primero que debe realizar el alumno para comenzar el proceso de selección es buscar las ofertas de prácticas y apuntarse a ellas. En esa pantalla le aparecerán todas las ofertas disponibles según la titulación que esté estudiando y podrá apuntarse a todas las que se adecuen a sus requisitos personales. Además, la pantalla ofrece la posibilidad de filtrar esas ofertas por provincia y localidad.

- **Visualizar las ofertas preseleccionadas:** Tras elegir las ofertas que más se ajusten a sus gustos y necesidades, el alumno tendrá acceso a otra pantalla para visualizar las prácticas en las que está en proceso de selección y ver el estado en el que se encuentra en cada una de ellas. A partir de esa pantalla, tendrá la autoridad de seleccionar, entre las ofertas en las que ha sido aceptado, la definitiva. Después de escogerla, tendrá que seleccionar el profesor que le gustaría tener como tutor.

- **Ver los detalles de las ofertas adjudicadas:** Una vez el alumno haya seleccionado las prácticas definitivas, puede ver las especificaciones del contrato final, así como la información del tutor, el instructor y los detalles de la oferta. Posteriormente, podrá proceder a la firma del acuerdo.

- **Acceder al perfil:** El alumno puede acceder a su perfil para ver los datos personales obtenidos de su centro educativo. También, podrá adjuntar un documento PDF con su currículum para que las empresas puedan ver su perfil.

- **Cambiar contraseña:** A través de la cabecera de la aplicación, donde se muestra su inicio de sesión, el alumno podrá acceder a la pantalla de cambio de contraseña. En esa pantalla, deberá incluir su contraseña actual, la nueva contraseña y una confirmación de la nueva contraseña. Si esa información es correcta, el cambio de clave se realizará inmediatamente.

Para el profesor existen los casos de uso que se muestran en la Ilustración 19. Casos de uso profesor:

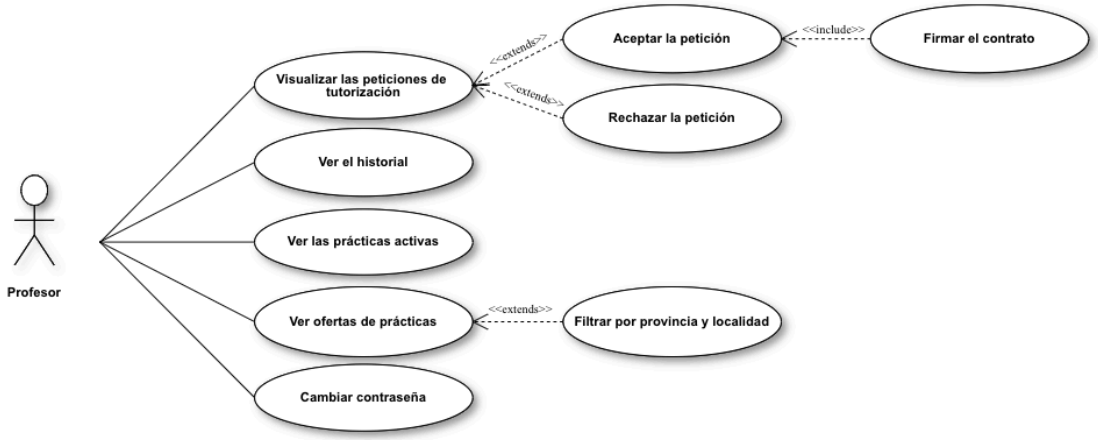

Ilustración 19. Casos de uso profesor

- **Visualizar las peticiones de tutorización:** El alumno debe hacer la petición de tutorización de sus prácticas. El profesor tendrá que decidir si acepta ser tutor o no en esas prácticas. Para tomar esta decisión, el profesor tendrá acceso a una pantalla donde se mostrarán todas las peticiones de tutorización, con los detalles del alumno y la oferta. Desde esta pantalla, el profesor podrá aceptar o rechazar dicha petición. Y, en el caso de aceptarla, firmar el contrato que formalice tu implicación.

- **Ver el historial:** El profesor podrá ver el historial de prácticas que ha tutorizado, ordenadas cronológicamente.

- **Ver las prácticas activas:** El profesor podrá ver el listado de prácticas que están activas en ese momento.

- **Ver las ofertas de prácticas:** Muchos alumnos acuden a los educadores pidiendo recomendaciones sobre las ofertas disponibles por su implicación en otras tutorizaciones anteriores, por lo que los profesores también tendrán una pantalla donde poder ver todas esas ofertas y poder aconsejar al alumno. Esa pantalla tendrá la opción de filtrar las ofertas por provincia y localidad.

- **Cambiar contraseña:** A través de la cabecera de la aplicación, donde se muestra su inicio de sesión, el profesor podrá acceder a la pantalla de cambio de contraseña. En esa pantalla, deberá incluir su contraseña actual, la nueva contraseña y una confirmación de la nueva contraseña. Si esa información es correcta, el cambio de clave se realizará inmediatamente.

Para la empresa (representada por el encargado de recursos humanos) existen los casos de uso que se muestran en la Ilustración 20. Casos de uso empresa (recursos humanos):

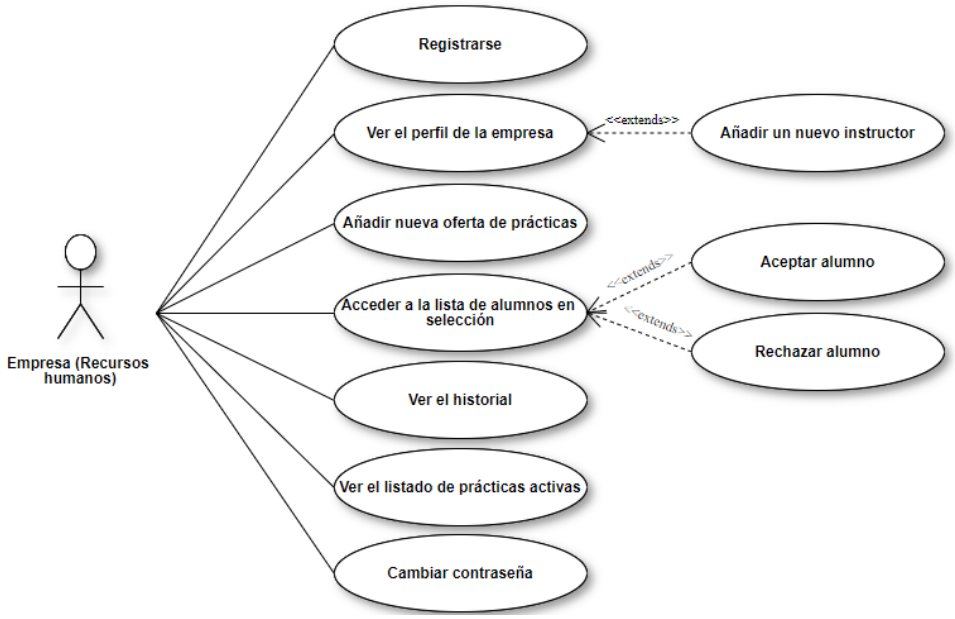

Ilustración 20. Casos de uso empresa (recursos humanos)

- **Registrarse:** Para que la empresa pueda ofrecer puestos de trabajo a estudiantes, primero tendrá que registrarse en la aplicación. Para ello la persona encargada de recursos humanos deberá proporcionar la información especificada.

- **Ver el perfil de la empresa:** El responsable de recursos humanos podrá acceder a la información de la empresa, así como a la lista de instructores. Además, podrá añadir nuevos instructores a la lista, proporcionando su información personal.

- **Añadir nueva oferta de prácticas:** La empresa puede añadir una nueva oferta de prácticas, especificando los detalles y requisitos del trabajo. Deberá incluir una nueva oferta por cada puesto de trabajo que quiera proporcionar.

- **Acceder a la lista de alumnos en selección:** La empresa podrá ver el listado de alumnos que se han apuntado a cada oferta. Asimismo, tendrá acceso al currículum de cada alumno para así, poder conocerlos más personalmente. En esta pantalla, tendrá la opción de rechazar a los alumnos que no les interese contratar y seleccionar al que más le guste.

- **Ver el historial:** La empresa puede ver todos los alumnos que han realizado las prácticas en su empresa y cuál ha sido dicho contrato. Esta lista se proporcionará ordenada cronológicamente.

- **Ver el listado de prácticas activas:** La empresa podrá ver el listado de prácticas que están activas en ese momento.

- **Cambiar contraseña:** A través de la cabecera de la aplicación, donde se muestra su inicio de sesión, la empresa podrá acceder a la pantalla de cambio de contraseña. En esa pantalla, deberá incluir su contraseña actual, la nueva contraseña y una confirmación de la nueva contraseña. Si esa información es correcta, el cambio de clave se realizará inmediatamente.

Para el instructor existen los casos de uso que se muestran en la Ilustración 21. Casos de uso instructores:

Ilustración 21. Casos de uso instructores

- **Pendientes de firma:** Ver el listado de prácticas que debe firmar el instructor.

- **Ver el listado de prácticas activas:** El instructor podrá ver la lista de alumnos aceptados para realizar las prácticas. El instructor deberá firmar el contrato para finalizar la adjudicación y, una vez terminado, realizar una pequeña evaluación.

- **Ver el historial:** El instructor podrá ver el historial de las prácticas que ha tutorizado. Esta lista se proporcionará ordenada cronológicamente.

- **Cambiar contraseña:** A través de la cabecera de la aplicación, donde se muestra su inicio de sesión, el instructor podrá acceder a la pantalla de cambio de contraseña. En esa pantalla, deberá incluir su contraseña actual, la nueva contraseña y una confirmación de la nueva contraseña. Si esa información es correcta, el cambio de clave se realizará inmediatamente.

5.3. Flujo de trabajo

En esta sección se muestra un diagrama de flujo de trabajo (Ilustración 22. Flujo de trabajo), que consiste en conectar a los participantes del proceso de selección mediante todas las tareas implicadas. Todas esas tareas están ordenadas, de forma que todas tendrán un antecedente que tendrá que ejecutarse para poder pasar a la siguiente.

Todos los usuarios de esta aplicación tienen menor o mayor implicación en el proceso de selección de las prácticas, por lo que todos deberán realizar su papel para que la adjudicación se realice.

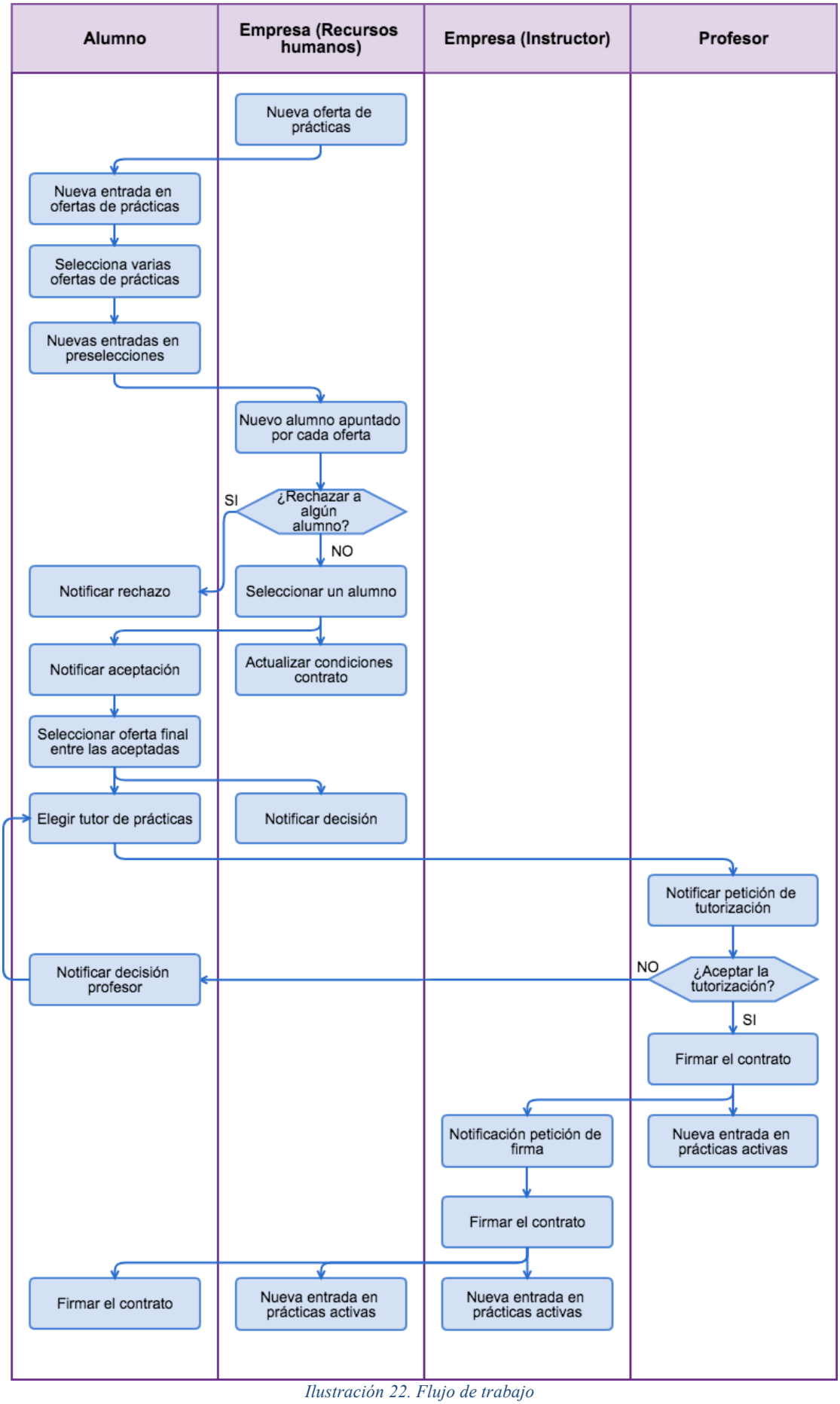

Ilustración 22. Flujo de trabajo

Esta es una breve explicación del flujo de trabajo:

En primer lugar, las empresas deben añadir nuevas ofertas para que los alumnos puedan optar a esos puestos. Una vez que esas nuevas ofertas estén incluidas, a los alumnos se les mostrará en la búsqueda de la pantalla “ofertas de prácticas”. A continuación, el alumno debe seleccionar las ofertas que más se adecuen a sus requisitos y se verán en “preselecciones”. Cada vez que un alumno se apunte a una oferta, en la pantalla de “alumnos en selección” la empresa tendrá una nueva entrada con la información del alumno.

Después de realizar la preselección, se le pasa el testigo a la empresa, que deberá decidir qué alumno se adapta mejor al puesto y rechazar a los que no lo hagan. En el caso de rechazar a alguno de ellos, se le notificará la decisión de la empresa. De lo contrario, si la organización se decide por un estudiante, deberá modificar las condiciones del contrato, actualizando sus valores con datos que se ajusten a la fecha en la que se encuentran y elegir al trabajador que será el instructor del alumno dentro de la empresa. Se le notificará al alumno la aprobación de la empresa.

A continuación, el alumno deberá tomar la decisión y elegir una de las ofertas en las que ha sido aceptado. Hecho esto, deberá seleccionar al profesor que realizará la función de tutorizar las prácticas y, a su vez, se le comunicará a la empresa la elección del alumno.

Por parte del profesor, se le informará de la petición de tutorización y deberá aceptar o rechazar dicha solicitud. En caso de rechazarla, se le avisará al alumno, quien tendrá que elegir a otro profesor. Si, de lo contrario, acepta la solicitud, tendrá que firmar el contrato y se le incluirán esas prácticas a la pantalla de “prácticas activas”.

En cuanto el profesor haya firmado el contrato, al instructor se le avisará de la nueva práctica y será su turno para firmar el acuerdo.

Para finalizar, una vez la oferta esté firmada tanto por el profesor como por el instructor, el alumno también deberá firmarlo y, una vez obtenidas las tres firmas, el acuerdo se hará oficial.

5.4. Modelo de dominio

En el modelo de dominio se representan todas las entidades que forman parte del proyecto y los datos guardados de cada una de ellas. Su finalidad es comprender y visualizar los diferentes participantes del sistema y cuál es la relación entre ellos (Se puede ver en la Ilustración 23. Modelo de dominio).

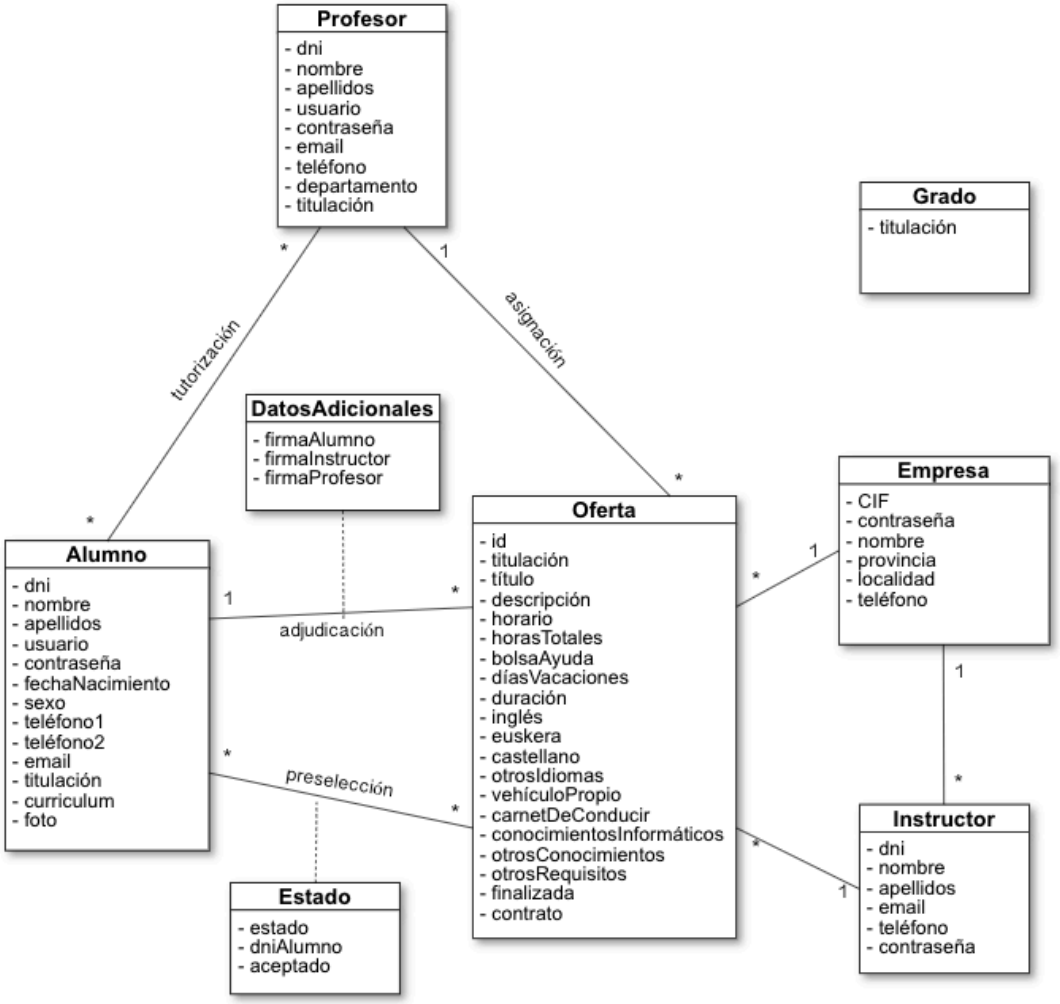

Ilustración 23. Modelo de dominio

A continuación, se describen cada una de las entidades:

Para este proyecto existen 4 tipos de usuario: el alumno, el profesor, la empresa y el instructor.
El **alumno** representa a todos los estudiantes que están interesados en realizar prácticas en una empresa y contaremos con su información personal como son el número de DNI, nombre, apellidos, fecha de nacimiento, sexo, teléfonos, email, currículum y foto, para que las empresas tengan oportunidad de conocer al alumno y poder contactar con él. Además, guardaremos su usuario y contraseña para poder validar su inicio de sesión en la aplicación.

Por otro lado tenemos al **profesor**, del cual guardaremos sus datos más relevantes como el número de DNI, nombre, apellidos, email, teléfono y departamento al que pertenece. También, guardaremos el usuario y contraseña para poder corroborarlos en el inicio de sesión.

La **empresa** simboliza a la compañía que quiera ofrecer un puesto de prácticas a alumnos de la universidad. De esta guardaremos el nombre de la empresa, provincia, localidad y teléfono, además de su CIF y contraseña.

Para terminar con los tipos de usuario, el **instructor** es aquel trabajador que tutorizará las prácticas del alumno dentro de la empresa. Se guardarán su número de DNI, nombre, apellidos, email, teléfono y contraseña. Cada instructor solo podrá tutorizar las prácticas asociadas a una empresa, pero la empresa tendrá varios trabajadores que puedan ejercer de instructor. Así mismo, la empresa adjudicará una o varias ofertas a cada instructor, pero cada oferta tendrá un solo trabajador asignado.

Las empresas tienen la oportunidad de incluir nuevas **ofertas** de prácticas, tantas ofertas como puestos de trabajo quiera ofrecer. A la hora de añadir nuevas ofertas, deberá rellenar los campos correspondientes a la descripción de las prácticas y los requisitos iniciales para poder seleccionarlas. Estos serán los datos que deberá rellenar: la titulación a la que ofrece el trabajo, el título de las prácticas, una descripción más detallada del puesto, el horario, las horas totales, la ayuda económica que ofrece, los días de vacaciones correspondientes por las horas invertidas y la duración de las prácticas. Además, tiene que completar los requisitos que debe cumplir cada alumno para poder optar al puesto: si debe tener nivel de inglés, euskera y castellano o algún otro idioma, vehículo propio y carnet de conducir, conocimientos informáticos, otros conocimientos o algún otro requisito no mencionado anteriormente. Además, incluiremos un atributo para saber si la práctica a finalizado o no y otro para, una vez el profesor acepte la tutorización y firme el acuerdo, guardar dicho contrato.

Los alumnos podrán apuntarse a todas las ofertas de trabajo que deseen para la **preselección** y a cada oferta se apuntarán uno o varios alumnos. Se guardará un atributo extra para saber el estado en el que se encuentra cada asignación y otros dos para saber si el profesor elegido por el alumno ha aceptado o no la solicitud de tutorización. En la decisión final, solo se le adjudicará un único alumno a cada oferta. En esa adjudicación, hay que saber si la oferta ya ha sido firmada por todos los participantes del proceso, que son el alumno, el profesor y el instructor en representación de la empresa.

A la hora de que el alumno realice su decisión final al seleccionar una oferta, debe elegir un profesor que se las tutorice. Un alumno podrá realizar más de unas prácticas y podrá escoger a un profesor diferente para cada una de ellas, pero a cada oferta sólo se le asignará un educador.

Por último, tendremos un registro de todos los **grados** en los que se ofrecen puestos de prácticas.

5.5. Base de datos

Este proyecto tendrá una única base de datos, que contendrá todos los datos necesarios, recogidos en sus correspondientes tablas. En la Ilustración 24. Base de datos, se pueden apreciar todas las tablas, con sus atributos, claves primarias y secundarias.

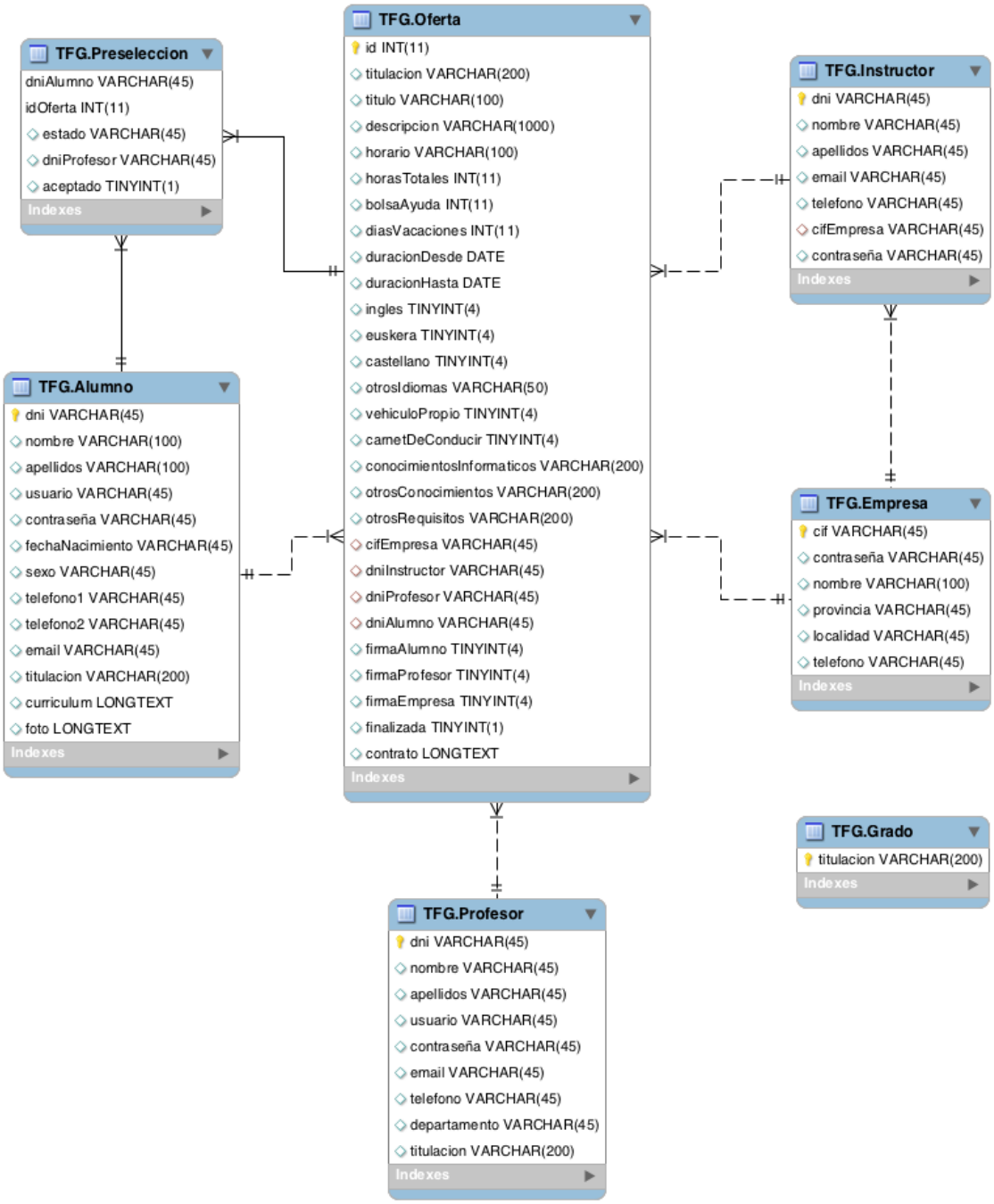

Ilustración 24. Base de datos

A continuación se detallan todas las tablas de la base de datos:

- **Alumno**

  Esta tabla guarda todos los datos relacionados con la información personal de los alumnos. A excepción del currículum, la información no cambiará en ningún punto de la aplicación. Por otro lado, los alumnos tendrán la posibilidad de incluir su

currículum para que las empresas puedan tener más información sobre ellos, por lo que es el único atributo dinámico de la tabla. Tanto la foto como el currículum se guardan en formato base64 y la clave primaria de cada alumno es el número de DNI.

- **Profesor**

Esta tabla tiene los datos personales del profesor. Todos los atributo son estáticos, ya que solo se utilizan para visualizarlos por pantalla y no se modifican en la aplicación. El número de DNI es el atributo por el que se identifica a cada profesor, por lo que es su clave primaria.

- **Empresa**

Esta tabla es la encargada de guardar toda la información relacionada con las empresas. Los atributos de esta tabla no cambian, debido a que son datos proporcionados por la empresa cuando se registra y permanecen estáticos en el sistema. Se utiliza el CIF de la empresa como clave primaria para su identificación.

- **Instructor**

En esta tabla se guardan todos los datos personales de los instructores de todas las empresas. Estas personas son asignadas a una oferta de prácticas y son las encargadas de firmar el contrato final, dando así el último visto bueno a la adjudicación. Cada instructor solo puede tutorizar las prácticas asociadas a una empresa, por lo que cada tupla tendrá una clave secundaria llamada *cifEmpresa* para guardar el CIF de la empresa a la que pertenece. La clave primaria de esta tabla es el número de DNI.

- **Oferta**

En esta tabla están todas las prácticas ofertadas por las diferentes empresas que participan en el sistema. Cada oferta es un puesto de trabajo diferente, por lo que cada una de ellas corresponde a un solo alumno. Esta tabla es la más importante del proyecto, ya que todos los datos que contiene son dinámicos y van cambiando a medida que avanza el proceso de adjudicación del alumno.

A continuación se detallan todos los atributos de la tabla:

- **id:** Es la clave primaria de la tabla. Se autoincrementa, por lo que la propia base de datos le asigna una cada vez que se incluye una nueva a la tabla.
- **titulación:** La empresa debe asignar sus ofertas de prácticas a un grado en concreto dependiendo del tipo de trabajo que se ofrezca.
- **título:** Este es el título de cada oferta, por el que el alumno se guiará para
- **descripción:** Esta es la descripción ofrecida por las empresas para poder explicar en qué consiste el trabajo que realizarán los alumnos dentro de la organización.
- **horario:** Este es el horario que ofrece cada práctica. Puede ser un horario concreto, como un texto explicativo donde se describa la disponibilidad de empresa.
- **horasTotales:** Las horas totales que trabajará el alumno durante el tiempo que esté en la empresa.

- **bolsaAyuda:** Este atributo indica la posible ayuda económica mensual que ofrezca la empresa por el trabajo realizado.
- **diasVacaciones:** El número de días que corresponden de vacaciones por la duración de las prácticas.
- **duración:** La duración de la oferta de prácticas, es decir, la fecha de inicio y final.
- **inglés:** Este atributo booleano indica si es un requisito o no el conocimiento de inglés para la adjudicación de la oferta.
- **euskera:** Este atributo booleano indica si es un requisito o no el conocimiento de euskera para la adjudicación de la oferta.
- **castellano:** Este atributo booleano indica si es un requisito o no el conocimiento de castellano para la adjudicación de la oferta.
- **otrosIdiomas:** Este atributo indica si sería necesario el conocimiento de algún otro idioma no mencionado anteriormente.
- **vehículoPropio:** Este atributo booleano indica el requisito de si es necesario o no tener vehículo propio.
- **carnetDeConducir:** Este atributo booleano indica el requisito de si es necesario o no tener carnet de conducir.
- **conocimientosInformaticos:** Los conocimientos informáticos necesarios para realizar las prácticas.
- **otrosConocimientos:** Los conocimientos previos necesarios para realizar las prácticas.
- **otrosRequisitos:** Los requisitos previos necesarios que no se hayan mencionado con anterioridad.
- **cifEmpresa:** La clave secundaria que hace referencia al CIF de la empresa a la que pertenece dicha oferta.
- **dniInstructor:** La clave secundaria que hace referencia al DNI del instructor seleccionado para tutorizar las prácticas.
- **dniProfesor:** La clave secundaria que hace referencia al DNI del profesor seleccionado por el alumno para tutorizar las prácticas.
- **firmaAlumno:** Este atributo booleano indica si ha sido firmado o no por el alumno el contrato final cuando se haya finalizado el proceso de adjudicación.
- **firmaProfesor:** Este atributo booleano indica si ha sido firmado o no por el profesor el contrato final cuando se haya finalizado el proceso de adjudicación.
- **firmaEmpresa:** Este atributo booleano indica si ha sido firmado o no por la empresa el contrato final cuando se haya finalizado el proceso de adjudicación.
- **finalizada:** Este atributo booleano determina si las prácticas están finalizadas o, de lo contrario, siguen activas.
- **contrato:** Este atributo guardará el valor del contrato que deben firmar todos los participantes. Se actualizará cada vez que se realice una firma.

- **Preselección**

Los alumnos pueden seleccionar más de una oferta a la hora de apuntarse a las prácticas, y a cada oferta puede apuntarse más de un alumno, por lo que esta tabla guarda un registro de las ofertas a las que se apunta cada alumno. Por todo esto, la clave primaria de esta tabla la forman el DNI del alumno y el id de la oferta. Como datos adicionales de la selección, se guarda el estado en el que se encuentra dicha selección, es decir, para

saber si ha sido aceptada o rechazada, por ejemplo, el DNI del profesor y un atributo booleano de "aceptado". Estos dos últimos atributos se utilizarán para conocer si el profesor que a elegido el alumno para la tutorización, ha aceptado esa solicitud.

5.6. Diagrama de secuencia

En esta sección se va a explicar el flujo de llamadas que realiza la aplicación para conseguir los datos del servidor. Todas las peticiones a los servicios REST tienen la misma estructura y siguen el mismo hilo de llamadas para conseguir la respuesta final. La aplicación está programada en TypeScript, lo que supone que la programación es asíncrona. Esto significa que cuando se realiza una llamada a una función dentro de otra función, el método secundario devuelve el control al programa llamante antes de que haya terminado, mientras sigue operando en segundo plano, lo que implica que cuando le devuelve la ejecución al programa principal, puede que el secundario no haya finalizado. Esto agiliza el proceso de ejecución, pero complica el razonamiento sobre el programa. Existen varias formas de controlar esta ejecución, sin embargo, para este proyecto se han utilizado los Observables, que emitirán un evento cuando el programa secundario haya finalizado. De esta manera, la función principal sabrá cuando la respuesta está completa.

Existen otros servicios de Angular llamados **Resolvers**. Estos inyectables están asociados a componentes, y su funcionalidad es ejecutarse antes de que se cargue dicho componente. La mayoría de las llamadas a los servicios REST se iniciarán mediante el método *resolver()* de esta clase, que se ejecutará en cuento la aplicación redirija al componente al que está asociado, para asegurarse de que los datos se cargan antes de mostrarse en la pantalla.

Usaremos como ejemplo la petición al servicio *listaOfertasDePracticas* para la demostración que se puede ver en la Ilustración 25. Diagrama de secuencia.

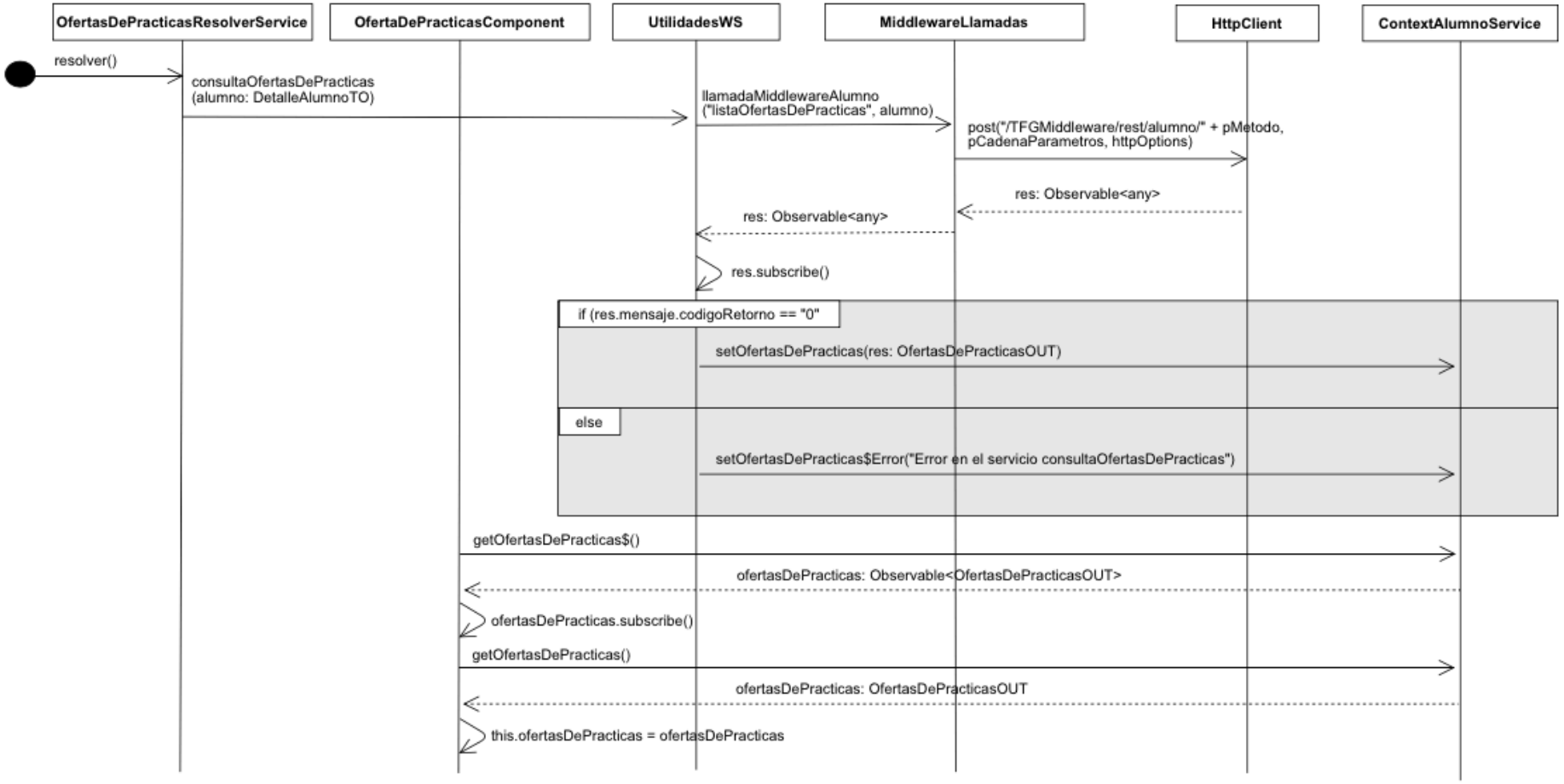

Ilustración 25. Diagrama de secuencia

La llamada al resto de servicios REST se ejecutan de la misma manera pero, dependiendo del usuario que esté realizando la petición, cambiará en contexto y el método de la clase *MiddlewareLlamadas*, que efectuará la llamada a la función del usuario correspondiente.

Por otra parte, si el componente que realiza la llamada no tiene ningún **Resolver** que se ejecute antes de que este se inicie, el mismo componente será el que haga la llamada a *UtilidadesWS*. El resto de pasos serán los mismos.

A continuación, se explicará el proceso de la llamada más detalladamente:

- El servicio *OfertaDePracticasResolve* ejecutará su método *resolver()*, realizando la llamada al método *consultaOfertasDePracticas()* de *UtilidadesWS*, pasándole como parámetro la información del alumno.
- Una vez acabe la ejecución del método *resolver()*, que será en cuento realice la llamada a *UtilidadesWS*, se ejecutará la clase *OfertasDePracticasComponent* que se subscribirá a la respuesta del método *getOfertasDePracticas$()* (que devuelve un observable) de la clase *ContextAlumnoService*, el cual en ese momento no emitirá ningún evento puesto que su valor no ha cambiado todavía. Entonces, el inicio del componente quedará a la espera de que el observable emita un evento.
- La clase ContextAlumnoService tendrá dos atributos, uno con la respuesta del servicio y otro con el Observable a esa respuesta.
- Mientras tanto, *UtilidadesWS* ejecutará el método *llamadaMiddlewareAlumno()* del servicio *MiddlewareLlamadas* pasándole como parámetros el nombre del método del servidor al que quiere realizar la llamada y el detalle del alumno.
- El servicio *MiddlewareLlamadas* le devolverá a UtilidadesWS un Observable, por lo que, este último se quedará a la espera de que dicho Observable emita un evento cuando obtenga una respuesta del servidor.
- Entretanto, el método *llamadaMiddlewareAlumno()* realizará una petición http al servidor, pasándole como parámetros la URL donde está expuesto el servicio requerido, el detalle del alumno y las opciones de configuración de la petición, que en este caso sirven para especificar que el contenido de la petición será de tipo JSON.
- Una vez la petición se realiza y se obtiene una respuesta que será un Observable, se emitirá un evento indicando que se ha recogido la respuesta, por lo que *UtilidadesWS*, que se había quedado a la espera, recibe la emisión del evento. En ese momento, se ejecuta el código que hay dentro de la subscripción al Observable, y la respuesta se le envía a *ContextAlumnoService*.
- En el momento en el que se le notifica la respuesta a *ContextAlumnoService*, el Observable emite un evento de que el atributo a cambiado y es aquí cuando la clase *OfertasDePracticasComponent* recibe el aviso.
- A partir de ese momento el componente ya tiene la lista de ofertas antes de que el componente haya acabado de cargarse, y podrá cargar el HTML con los datos recogidos.

6. Desarrollo

Este apartado se encarga de describir paso a paso como a sido el desarrollo de este proyecto. También se explicarán el funcionamiento y las estructuras propias de las herramientas utilizadas y la configuración realizada a partir de esa base.

El desarrollo de esta aplicación está dividido en dos partes independientes. Por una parte está la aplicación, que será accesible mediante el navegador y, por otro lado, el servidor encargado de comunicarse con la base de datos y exponer los servicios a los que se realizarán las peticiones.

Cabe destacar que, a pesar de tener una estructura diferente, comparten los modelos de las clases que sirven como entrada y salida de los servicios, es decir, los parámetros de entrada y salida son iguales en la parte Angular y en la parte REST, de este modo, cuando se envían los parámetros de una aplicación a otra, convergen perfectamente y no hay problemas con sus atributos. Estas clases sólo sirven como prototipos para guardar atributos, no tendrán ninguna funcionalidad.

6.1. Modelos

Para entender mejor cuál es la estructura de los modelos y facilitar la comprensión de los parámetros de entrada y salida de los métodos que se van a explicar más adelante, se procederá a aclarar la función y los datos que guarda cada uno de ellos. Los colores usados para distinguir cada clase, ayudarán a identificar las clases que incluyen otras.

En la Ilustración 26. Modelos de entrada, se encuentran los parámetros de entrada de algunos servicios, que se diferencian por el "IN" al final del nombre de la clase:

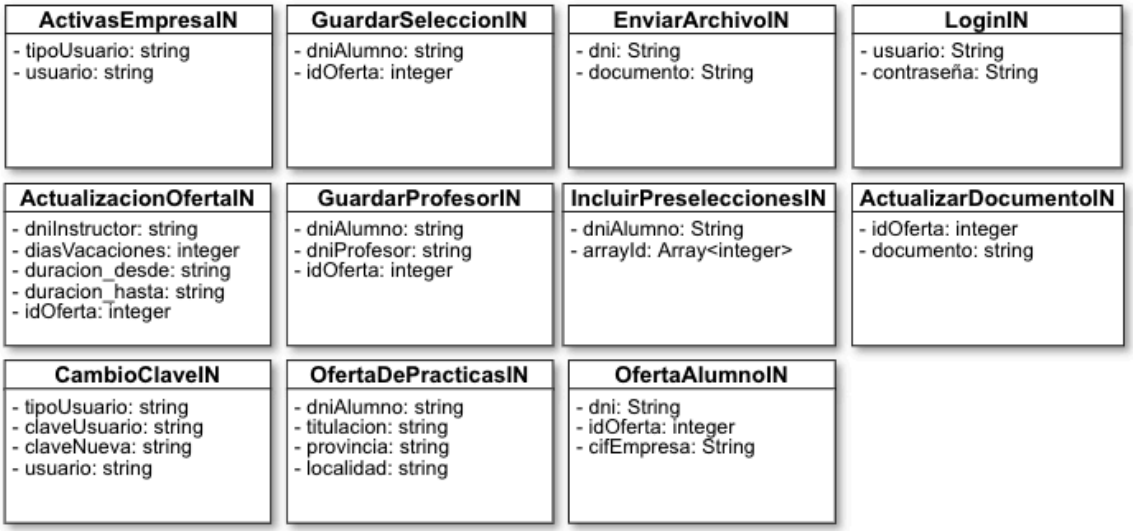

Ilustración 26. Modelos de entrada

A continuación se explicarán los más importantes:

- **LoginIN:** Esta clase servirá de entrada para el servicio de validación del usuario.

- **OfertaDePracticasIN:** En esta clase definiremos los filtros que tiene la búsqueda de ofertas de prácticas.

- **ActualizarOfertaIN:** En esta clase se guardarán los datos a modificar de una oferta. Para identificar de que oferta se trata, se incluye también el id de la oferta.

- **OfertaAlumnoIN:** Este modelo servirá para relacionar los alumnos y las ofertas, pasándole el DNI del alumno, el id de la oferta y el CIF de la empresa, podremos vincular directamente esas tres entidades.

- **EnviarArchivoIN:** En esta clase le enviaremos al servicio el documento que desea vincular el alumno a su perfil, es decir, su currículum. Le enviaremos también el DNI del alumno para identificar quien está realizando la petición.

- **IncluirPreseleccionesIN:** Esta clase nos servirá para añadir al alumno con el DNI "dniAlumno" todas las ofertas seleccionadas para la preselección. Para ello, le pasaremos un array con todos los id de las ofertas escogidas.

Por otro lado, tenemos las clases que sirven como base para los parámetros de entrada y salida, distinguidos por el "TO" que llevan al final del nombre. Estas clases se muestran en la Ilustración 27. Modelos de detalles:

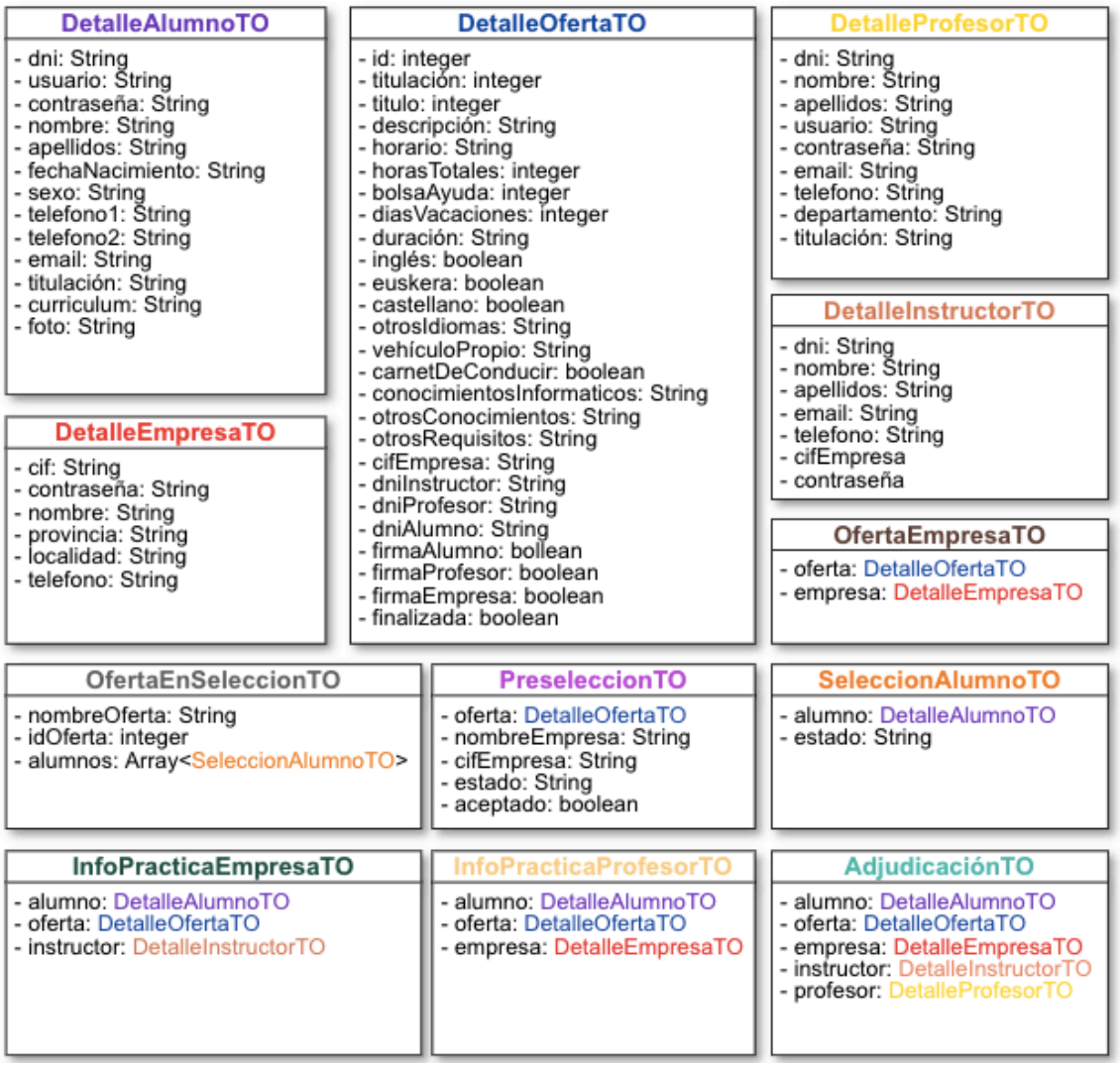

Ilustración 27. Modelos de detalles

A continuación se muestran los más relevantes:

- **DetalleAlumnoTO:** Esta clase recogerá todos los datos personales del alumno.
- **DetalleOfertaTO:** Este modelo recogerá todos los detalles de la oferta de prácticas.
- **DetalleProfesorTO:** Esta clase agrupará toda la información personal del alumno.
- **DetalleEmpresaTO:** Esta clase devolverá los datos personales de la empresa.
- **DetalleInstructorTO:** Este modelo revuelve toda la información personal del instructor.
- **AdjudicaciónTO:** Esta clase sirve para mantener agrupada toda la información de todos los participantes de cada práctica.
- **OfertaEmpresaTO:** Esta clase sirve para recoger la información de la oferta de prácticas y la empresa a la que pertenece.
- **OfertaEnSeleccionTO:** Este modelo sirve para recoger todos los alumnos que están en la lista de selección de una oferta.
- **PreseleccionTO:** Este modelo le servirá a la parte del alumno para recoger la información de la oferta que tiene en preselección, estado en el que se encuentra y si el profesor a aceptado la tutorización.

Para finaliza, en la Ilustración 28. Modelos de salida, se muestran las clases que sirven como respuesta de los servicios, identificadas por el "OUT" del final del nombre. Todas las salidas tendrán el atributo *mensaje* para que la aplicación sepa si el servicio se ha ejecutado correctamente o no. De esta manera, la aplicación podrá actuar dependiendo de ese resultado.

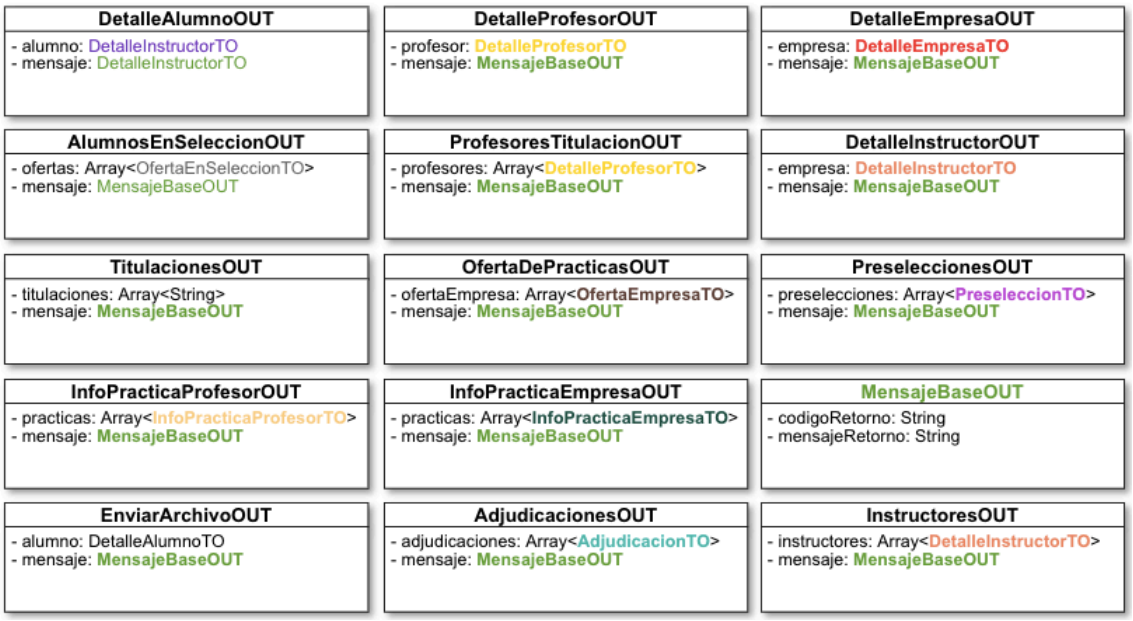

Ilustración 28. Modelos de salida

A continuación, se explican los más significativos:

- **MensajeBaseOUT:** El atributo *códigoRetorno* sirve para identificar mediante un código el resultado obtenido del servicio. Los valores más comunes son "0", en el caso de que el servicio se haya ejecutado correctamente, y "1", en el caso de que haya ocurrido algún error durante la ejecución. También puede existir algún otro valor que sirve para identificar diferentes comportamientos de un servicio. Por otro lado, está el atributo *mensajeRetorno* que será un reflejo del código en forma de frase escrita.

- **AdjudicacionesOUT:** Esta clase recoge en una lista todas las adjudicaciones de un alumno, teniendo cada una de esas entradas de la lista la información relevante a la adjudicación.

- **EnviarArchivoOUT:** Este modelo devuelve la información del alumno actualizada, después de que se haya modificado su currículum.

- **InstructoresOUT:** Esta clase devuelve una lista con todos los instructores que pertenecen a la empresa que ha realizado la petición de los trabajadores.

6.1. Estructura de Angular

A continuación, se describen, por una parte, la estructura básica de Angular, la cual se establece cuando se crea un nuevo proyecto y, por otro lado, la estructura propia creada a partir de esa base.

6.1.1. Base de Angular

Como se ha mencionado anteriormente, se utilizará Angular como framework para la implementación del código, el cual crea su propia estructura al crear un nuevo proyecto, que se puede ver en la ilustración 29.

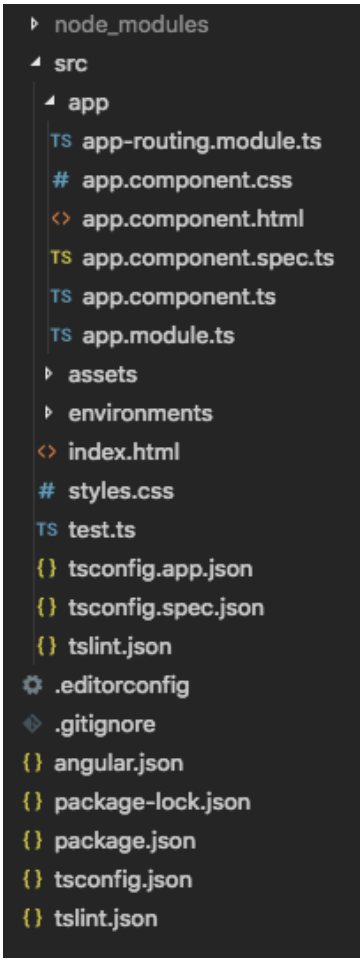

Ilustración 29. Base de Angular

A continuación se explicarán los archivos y carpetas más relevantes de esa configuración:

En la carpeta de **app** se encontrará todo el código fuente de la aplicación, los componentes, servicios y todos aquellos elementos que tengan que ver con la vista de la aplicación y en **assets**, en cambio, se encontrarán todos aquellos archivos estáticos del proyecto, como imágenes, vídeos o ficheros similares.

Angular se estructura mediante **módulos**, que son contenedores para almacenar componentes y servicios dentro de la aplicación. Se organizan jerárquicamente, a partir de un módulo raíz, se enlazan otros. Estas clases usan el decorador *@NgModule()* y como parámetro, se les pasa un solo objeto que la configuración de este. Dentro de esa configuración, se declara un array llamado *imports* con los punteros a otros módulos que forman la aplicación. Además, hay otras dos propiedades, *declarations* y *providers,* que hacen referencia a los componentes y servicios respectivamente que son utilizados en esa clase. *app.module.ts* es el módulo raíz, por lo que los demás deben ser declarados allí. A continuación, se muestra un ejemplo de este proyecto con la configuración de un módulo:

```
@NgModule({
 declarations: [
   ErrorGeneralComponent,
   ErrorAplicacionComponent
 ],
 imports: [
   CommonModule,
   TranslateModule,
   RouterModule
 ],
 exports: [
   ErrorGeneralComponent,
   ErrorAplicacionComponent
 ]
})
```

Uno de los módulos más importantes de la aplicación es el **app-routing.module.ts**, que se encarga del enrutamiento de la página, es decir, sirve para definir qué componente se visualizará con cada ruta. Si algún componente tiene un *resolver* asignado para ejecutarse antes de cargar la vista, se declarará en esta clase. En las siguientes líneas se muestra un ejemplo de enrutamiento:

```
{path: 'alumnosEnSeleccion', component: AlumnosEnSeleccionComponent,
   resolve: { AlumnosEnSeleccionResolverService}
 },
```

Por otra parte se encuentran los **componentes**, que son los bloques que conforman una aplicación web en Angular. Son pequeñas partes lógicas que representan diferentes partes de la pantalla. Cada componente lo forman cuatro archivos diferentes: el fichero .ts es el encargado de la funcionalidad del componente, el fichero .html con la plantilla, el .css con los estilos y el .spec.ts que sirve para el testing de cada componente. Estos componentes usan el decorador @Component() que, al igual que los módulos, reciben como parámetro las propiedades que lo configuran. Los componentes definen nuevas etiquetas HTML con las que se permite el uso de ese componente dentro de otro. El nombre de esa nueva etiqueta se define dentro de la configuración como valor del parámetro *selector*. Por ejemplo, si el selector tiene el valor app-root, se podrá visualizar en la pantalla usando la etiqueta <app-root></app-root>. Aunque la plantilla del componente puede escribirse directamente usando la propiedad *template*, es más frecuente encontrar la plantilla en otro fichero independiente y hacerle referencia con el parámetro *templateUrl*. Por último, la propiedad *styles* hace referencia al CSS que contiene los estilos que utiliza ese componente. A continuación se puede ver la configuración de un componente:

```
@Component({
 selector: 'app-root',
 templateUrl: './app.component.html',
 styleUrls: ['./app.component.css']
})
```

El gestor de paquetes npm genera un archivo **package.json** que podría definirse como la declaración de la aplicación, donde se definen características como el nombre del proyecto, la versión, las dependencias, los autores, etc. Asimismo, se encuentra el archivo **package-lock.json**, que realiza un seguimiento de la versión exacta de cada paquete que se instala. Para finalizar con la gestión de las dependencias, nos encontramos con la carpeta **node_modules**, que se encarga de administrar las dependencias de la aplicación. Por tanto, todas las librerías que se declaren como dependencias en el archivo package.json, deben estar descargadas en esta carpeta.

Dentro de la carpeta *src* también se encuentra **index.html** que es el archivo HTML principal, es decir, donde se va a alojar nuestra aplicación y **styles.css** son los estilos generales del proyecto, que se podrán usar en cualquier componente, aunque luego cada componente puede definir sus propios estilos.

Existen otros muchos ficheros de configuración que no se modificarán para llevar a cabo este proyecto.

6.1.2. Estructura de Angular para este proyecto

Partiendo de la propia base establecida cuando se crea un nuevo proyecto en Angular, en esta sección se explica cuál ha sido la estructura creada para el desarrollo de este proyecto.

Dentro de la carpeta **app**, donde se guarda todo el código de este proyecto, la aplicación está estructurada en 3 bloques diferentes (ver Ilustración 30. Bloques de Angular para este proyecto).

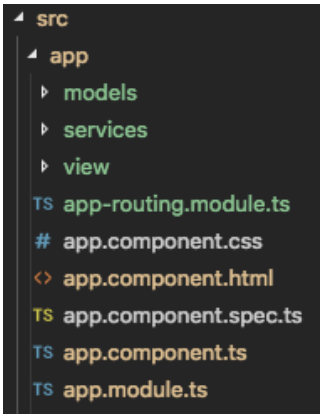

Ilustración 30. Bloques de Angular para este proyecto

Models

En este bloque se guardan todas las clases modelo que tiene la aplicación, que representan las estructuras de datos que sirven como entrada y salida de los diferentes servicios. Estos modelos se han dividido en 3 bloques:

- **models-in:** Estos modelos son los parámetros de entrada para los servicios API, que estarán compuestos dependiendo de los datos que sean requeridos por los servicios.
- **models-to:** Estos modelos son los encargados de guardar los datos obtenidos de la base de datos, así como la información personal del usuario, los datos de las prácticas ofertadas, etc.
- **models-out:** Estos modelos representan los parámetros de salida de los servicios API.

Services

En este bloque están accesibles todos los servicios de la aplicación. Estos servicios usan el decorador **@Injectable()** ya que son inyectables desde cualquier parte del sistema y sirven para compartir funcionalidades y datos entre los componentes. En la Ilustración 31. Esquema bloque services, se puede ver un esquema de este bloque.

Ilustración 31. Esquema bloque services

A continuación se detalla la funcionalidad de cada servicio y los métodos o atributos más importantes de cada uno de ellos:

ContextAlumnoService
En este servicio se guardan todos aquellos datos que son accesibles en toda la aplicación cuando el usuario que accede a la aplicación es un alumno, los datos del alumno y las ofertas de prácticas en las que está apuntado, por ejemplo. Los atributos pertenecientes a esta clase son las salidas de los servicios y sus observables, que serán los responsables de emitir el cambio en la salida de los servicios. Los métodos, por otro lado, son los getters y setters de dichos atributos.

ContextEmpresaService
En este servicio se guardan todos los datos relacionados con el inicio de sesión de una empresa. Es decir, toda la información accesible de la empresa, como la información personal de la empresa o las prácticas que tiene ofertadas, por ejemplo. Los atributos pertenecientes a esta clase son las salidas de los servicios y sus observables, que serán los responsables de emitir el cambio en la salida de los servicios. Los métodos, por otro lado, son los getters y setters de dichos atributos.

ContextEmpresaService
En este servicio se guardan todos los datos que se obtienen de las diferentes funcionalidades que puede realizar el profesor cuando inicia sesión. Los atributos pertenecientes a esta clase son las salidas de los servicios y sus observables, que serán los responsables de emitir el cambio en la salida de los servicios. Los métodos, por otro lado, son los getters y setters de dichos atributos.

ContextService
Todos los datos comunes relacionados con la funcionalidad de la aplicación que sirven para todos los tipos de usuarios se guardan en este servicio.

- **tipoUsuario:** Este atributo guarda el tipo de usuario que ha accedido a la aplicación.

- **trazas:** Este array guarda las trazas del método anadirTrazaACola() que se explicará más adelante.

- **idSesion:** Este atributo guarda el id de sesión creado a partir de un algoritmo criptográfico SHA-256. Cada inicio de sesión tendrá uno diferente.

- **moduloActual:** Este atributo guarda el número de la página principal para mostrar en el breadcrumb.

- **pasoActual:** Este atributo guarda el número que hace referencia a cada pantalla de la aplicación. Se utiliza para visualizar el nombre de la pantalla actual en el breadcrumb.

- **idioma:** Este atributo guarda el idioma en el que se están mostrando la información.

LimpiarSesionService
Cuando el usuario cierra sesión o hay algún fallo en el sistema, este servicio se encarga de inicializar todos los datos provenientes de los servicios. Todos los atributos los inicializa a vacío o nulo.

MiddlewareLlamadas
Este servicio es el encargado de realizar las llamadas a los servicios REST. Todos estos servicios están divididos por el tipo de usuario que los requiere, por lo que en este inyectable están las 4 funciones a las que se debe llamar para ejecutar un servicio.

- llamadaMiddlewareOtros(pMetodo:string, pParametro:any): Observable<any>
  Esta función realiza la llamada a http://localhost:8080/TFGMiddleware/rest/otros/ donde están todos los servicios comunes que son accesibles para cualquier tipo de usuarios. Como parámetros recibe el nombre del método y el parámetro correspondiente al método invocado. La respuesta es un observable de la respuesta.

- llamadaMiddlewareAlumno(pMetodo:string, pParametros:any): Observable<any>
  Esta función realiza la llamada a http://localhost:8080/TFGMiddleware/rest/alumno/ donde están todos los servicios que son accesibles para cuando el tipo de usuario que accede a la aplicación es un alumno. Como parámetros recibe el nombre del método y el parámetro correspondiente al método invocado. La respuesta es un observable de la respuesta.

- llamadaMiddlewareProfesor(pMetodo:string, pParametros:any): Observable<any>
  Esta función realiza la llamada a http://localhost:8080/TFGMiddleware/rest/profesor/ donde están todos los servicios que son accesibles para cuando el tipo de usuario que accede a la aplicación es un profesor. Como parámetros recibe el nombre del método y el parámetro correspondiente al método invocado. La respuesta es un observable de la respuesta.

- llamadaMiddlewareEmpresa(pMetodo:string, pParametros:any): Observable<any>
  Esta función realiza la llamada a http://localhost:8080/TFGMiddleware/rest/empresa/ donde están todos los servicios que son accesibles para cuando el tipo de usuario que accede a la aplicación pertenece a una empresa. Como parámetros recibe el nombre del método y el parámetro correspondiente al método invocado. La respuesta es un observable de la respuesta.

StateService
Este servicio es el delegado de notificar a los diferentes componentes mediante observables, si cambia el estado de los flags que se encargan de saber si es necesario pintar en la pantalla la cabecera, la miga de pan (el breadcrumb, que ayuda al usuario a ubicarse dentro de la aplicación) y el loader. Los atributos y métodos de esta clase son los observables y sus correspondientes getters y setters.

UtilidadesWS
Este inyectable es el encargado de llamar al middleware que realiza las llamadas a los servicios. A su vez, es el gestor de las respuestas de estos servicio, recoge su solución y determina si es correcta o, de lo contrario, la función no se ha ejecutado correctamente. Es importante tener diferenciadas las llamadas a los servicios de los componentes, para que, si hay que realizar algún cambio relacionado con los servicios, todos los cambios que sean necesarios estén en la misma clase, y así, no tener que buscarlos en toda la aplicación. Todas las funciones de esta clase corresponden a los métodos de los servicios REST, por lo que se procederá a explicarlos más adelante.

Utilidades
En este servicio están accesibles todas las funciones que son comunes para todos los componentes, de esta forma, no es necesario tener los mismos métodos en diferentes partes del código.

- anadirTrazaACola(pMensaje: string)
  Esta función crea una clase modelo con el mensaje facilitado como parámetro y la fecha y hora del día que se crea. Las funciones más relevantes la usan y de esta manera, se puede realizar un seguimiento del flujo de llamadas que está ejecutando cada usuario y, en el caso de que la aplicación falle, poder detectar más fácilmente donde ha ocurrido el error.

- lanzarErrorGeneral()
  Esta función se ejecuta cuando ocurre un fallo durante la llamada a los servicios. Su función es redirigir la aplicación a una página de error.

- lanzarErrorAplicacion()
  Esta función se ejecuta cuando ocurre algún fallo inesperado durante la ejecución de la aplicación. Su función es redirigir a una página de error.

- lanzarModal(modal)
  Pasándole como parámetro la instancia del popup que se quiere mostrar, esta función se encarga de abrirlo.

- comprobarFechaMayorIgual(fechaComprobar)
  Esta función compara la fecha incluida como parámetro con la fecha de hoy y devuelve un booleano indicando si es mayor o no. Este método se utiliza para comprobar que las fechas proporcionadas por el usuario son correctas.
- esFechaFinMayorIgual(fechaInicial,fechaFinal)
  Esta función compara dos fechas y devuelve un booleano indicando si el segundo parámetro es mayor que el primero. Este método es utilizado para comprobar que las fechas proporcionadas por el usuario son correctas.

- obtenerIdSesion()
  Esta función utiliza el algoritmo criptográfico SHA-256 para obtener un id de sesión con la fecha y hora en la que el usuario a realizado el login. De esta forma, si hay más de un usuario al mismo tiempo interactuando con la aplicación, ese id nos servirá para localizar las trazas de dichos usuarios y saber que está ejecutando cada uno de ellos.

Dentro de los servicios, hay otro grupo llamado **Resolvers**. La funcionalidad de estos servicios es ejecutarse antes de que se cargue dicho componente. De esta manera, se encargan de proporcionar valores (ya sean observables, listas o cualquier otro tipo) a los componentes, antes de que estos se inicien y se visualicen. Así, si algún componente debe mostrar por pantalla datos antes de lanzar el HTML, el resolver se los retornará.

ActivasEmpresaResolverService
Este servicio asociado a *ActivasEmpresaComponent* consigue una lista con todas las prácticas que están actualmente activas para la empresa.

ActivasProfesorResolverService
Este servicio asociado a *ActivasProfesorComponent* consigue una lista con todas las prácticas que están actualmente activas para el profesor.

AdjudicacionesResolverService
Este servicio asociado a *AdjudicacionesComponent* recoge toda la información de los participantes de cada adjudicación para mostrarle al alumno los detalles del acuerdo final.

AlumnosEnSeleccionResolverService
Este servicio se encarga de obtener la lista de alumnos apuntamos a cada oferta de una empresa en concreto. Este resolver corresponde a AlumnosEnSeleccionComponent, por lo que cuando ese componente se carga, tiene accesible esa lista en ContextEmpresaService para poder pintarla por pantalla.

HistorialEmpresaResolverService
La función de este resolver consiste en recopilar los datos de todas las prácticas realizadas en la empresa. El componente asociado es *HistorialEmpresaComponent*.

HistorialProfesorResolverService
La función de este resolver consiste en recopilar los datos de todas las prácticas tutorizadas por un profesor. El componente asociado es *HistorialProfesorComponent*.

NuevaOfertaResolverService
Cuando la empresa incluye una nueva oferta de prácticas debe elegir para qué titulación está disponible. Para ello, se le da una lista con todas las titulaciones que puede seleccionar. Este servicio se encarga de conseguir esa lista.

OfertasDePracticasProfesorResolverService
Este resolver se encarga de recopilar todas las ofertas disponibles para una titulación. El componente asociado es *OfertasDePracticasProfesorComponent*.

OfertasDePracticasResolverService
Este resolver se encarga de recopilar todas las ofertas disponibles para la titulación que está cursando el alumno, teniendo en cuenta en las que no está apuntado previamente. El componente asociado es *OfertasDePracticasComponent*.

PerfilEmpresaResolverService
El componente *PerfilEmpresaComponent* visualiza en el perfil de la empresa una tabla con todos sus instructores. Este resolver consigue una lista con todos esos trabajadores.

PeticionesFirmaInstructorResolverService
Este servicio recoge una lista con todas las peticiones de tutorización que tiene un profesor. Está asociado a la pantalla *PeticionesFirmaInstructorComponent*.

PeticionesTutorizacionResolverService
Este servicio devuelve una lista con todas las peticiones de tutorización que tiene pendientes. Este resolver está asociado a *PeticionesTutorizacionComponent.*

PreseleccionesResolverService
Este servicio recoge todas las ofertas que el alumno tiene preseleccionadas y el estado de selección en el que se encuentran. *PreseleccionesComponent* es el componente asociado.

View

Angular está basado en componentes, que son pequeñas partes lógicas que representan diferentes partes de la pantalla. Cada componente tiene un archivo .ts que es el responsable de la funcionalidad, un archivo .html que se ocupa de la parte visual y un archivo .css que se encarga de dar estilo a ese componente.

En este bloque se encuentran todos los componentes que son la parte visible del sistema, es decir, todo lo que se visualiza durante la navegación de la aplicación. A continuación se muestra su organización en Angular (ver Ilustración 32. Estructura view Angular).

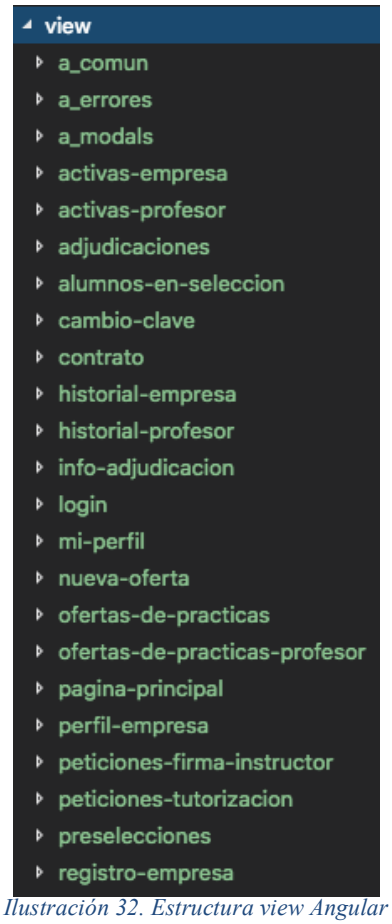

Ilustración 32. Estructura view Angular

En la Ilustración 33. Esquema Angular proyecto, se puede ver un esquema de este bloque y todos sus componentes:

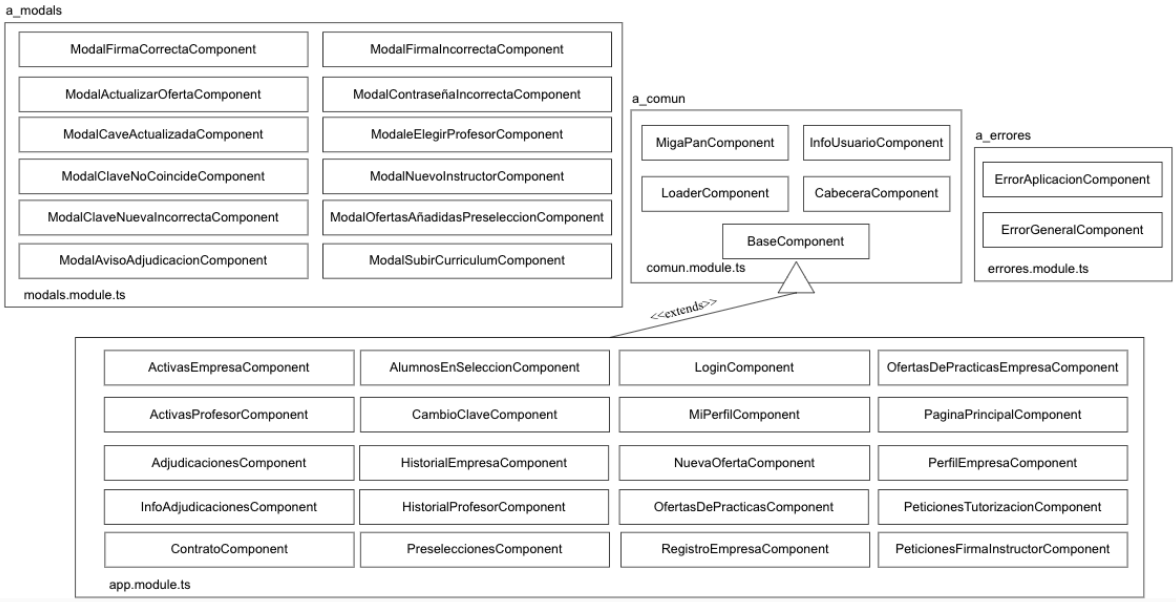

Ilustración 33. Esquema Angular proyecto

Como se puede ver en la ilustración anterior, todas las pantallas de la aplicación (bloque de abajo) están dentro del módulo principal app.module.ts y existen otros tres módulos secundarios (en la parte superior) que se incluyen dentro del módulo principal.

comun.module.ts

En este módulo se encuentran los componentes comunes que comparten todas las pantallas:

- **BaseComponent:** Este componente no contiene parte visual en la aplicación, pero todos los componentes de la aplicación que se encargan de las funcionalidades de las pantallas, extienden de él, puesto que se encarga de facilitar todos los servicios y métodos necesarios para su implementación.

- **CabeceraComponent:** Este componente representa la cabecera de la aplicación, donde se presentan todas las funcionalidades a las que tiene acceso el usuario.

- **InfoUsuarioComponent:** Este componente se visualiza dentro de la cabecera, para indicar el nombre del usuario que está dentro de la aplicación. Además, se cerrar sesión, cambiar la contraseña y el idioma gracias a este elemento.

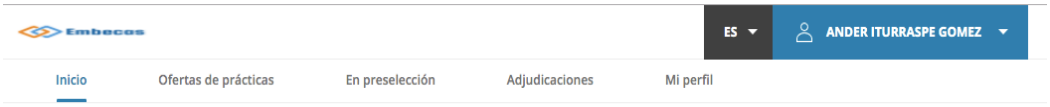

Ilustración 34. Visualización cabecera e infoUsuario

- **LoaderComponent:** Este componente no tiene parte funcional, pero sirve para notificar al usuario el proceso de carga de la aplicación, por si un recurso tarda mucho en cargarse, para que el usuario esté al corriente de que la petición se está procesando.

Ilustración 35. Visualización loader

- **MigaPanComponent:** Este elemento le comunica a usuario en que parte de la aplicación se encuentra.

Ilustración 36. Visualización miga de pan

errores.module.ts

En este módulo se encuentran las pantallas de error que pueden saltar durante la navegación en caso de que ocurra algún problema.

- **ErrorAplicaciónComponent:** Este componente representa una pantalla de error que salta cuando la aplicación recibe un error inesperado que no se ha contemplado dentro de la funcionalidad del sistema.

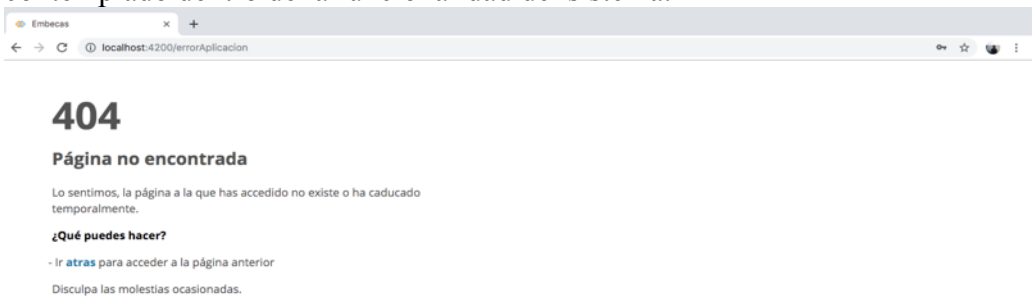

Ilustración 37. Pantalla error 404

- **ErrorGeneralComponent:** Este componente representa una pantalla de error que salta cuando la aplicación recibe un error inesperado que no se puede tratar por parte del servidor.

Ilustración 38. Pantalla error 500

modals.module.ts

Este módulo contiene popups que se muestran durante la navegación con contenido complementario, para que el usuario esté al tanto de lo que está ocurriendo. Además, algunos de ellos contienen campos de entrada que el usuario debe rellenar para finalizar una acción correctamente.

- **ModalActualizarOfertaComponent:** Una vez la empresa selecciona un alumno para realizar las prácticas, debe actualizar la fecha de inicio y fin de las prácticas, los días de vacaciones que corresponden a ese periodo y el empleado que realizará el papel de instructor. Todo eso se decidirá en esta ventana emergente.

Ilustración 39. Modal actualizar oferta

- **ModalClaveActualizadaComponent:** Una vez el usuario haya proporcionado la contraseña actual y la nueva, y la modificación se haya ejecutado correctamente, se muestra esta ventana informando al usuario de que el cambio ha sido realizado correctamente.

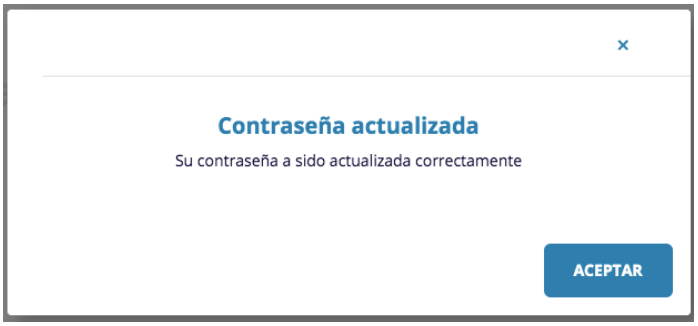

Ilustración 40. Modal contraseña actualizada

- **ModalClaveNoCoincideComponent:** Esta ventana se muestra cuando el usuario quiere cambiar su contraseña, pero la clave actual introducida no coincide con la que está en la base de datos.

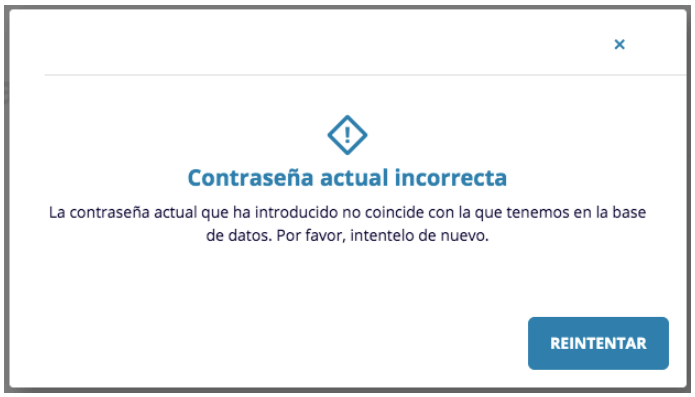

Ilustración 41. Modal contraseña actual incorrecta

- **ModalClaveNuevaIncorrectaComponent:** Cuando el usuario quiere modificar su contraseña, debe introducir la nueva dos veces. Si esos dos campos no coinciden, se le notifica al usuario que debe volver a introducirlas de nuevo.

Ilustración 42. Modal nueva clave incorrecta

- **ModalContraseñaIncorrectaComponent:** Al inicar sesión, puede que el usuario introduzca el usuario o contraseña incorrectos. Este popup se mostrará cuando esto suceda.

Ilustración 43. Modal contraseña incorrecta

- **ModalElegirProfesorComponent:** Una vez un alumno selecciona la oferta final que quiere realizar, debe elegir un profesor para que las tutorice. Para ello, se le mostrará esta ventana con las opciones disponibles para que escoja.

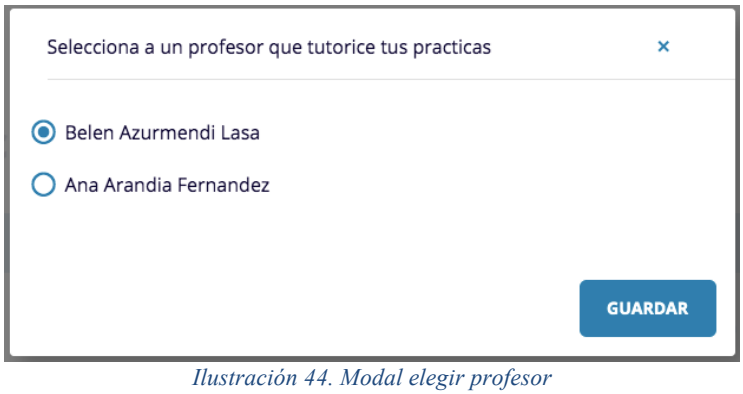

Ilustración 44. Modal elegir profesor

- **ModalNuevoInstructorComponent:** El representante de recursos dentro de la empresa que tiene acceso a la lista de instructores, también tiene la oportunidad de incluir uno nuevo a esa lista. Cuando quiera realizar esa operación, se

mostrará esta ventana con todos los campos que debe rellenar sobre la información personal del trabajador.

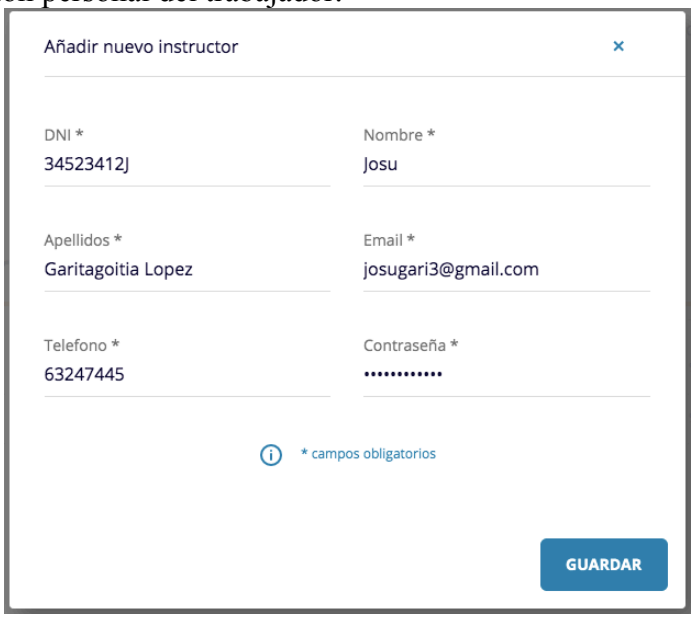

Ilustración 41. Modal nuevo instructor

- **ModalOfertasAñadidasPreseleccionComponent:** Este popup se visualizará para notificar al alumno que las ofertas que ha seleccionado en la ventana "ofertas de prácticas", se han incluido a lista de preselecciones correctamente.

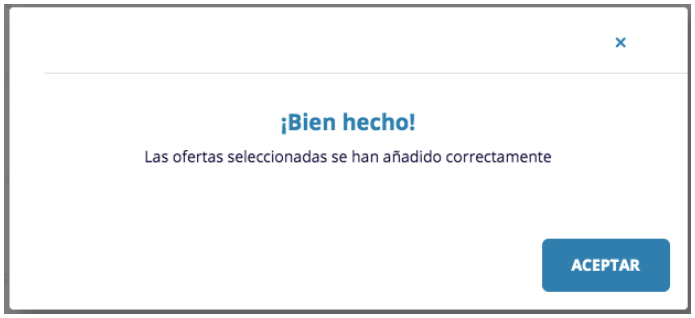

Ilustración 45. Modal ofertas añadidas

- **ModalSubirCurriculumComponent:** El alumno tiene la posibilidad de subir un documento PDF con su currículum. Para realizar esa acción, se mostrará esta ventana.

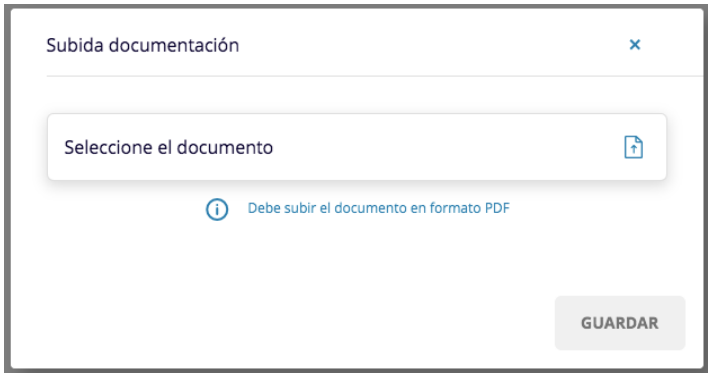

Ilustración 46. Modal subir currículum

- **ModalFirmaCorrectaComponent:** Este modal sirve para notificar al usuario que la firma del contrato se ha realizado correctamente.

Ilustración 47. Modal firma correcta

- **ModalFirmaIncorrectaComponent:** Esta ventana sirve para notificar al usuario que la firma del contrato no ha podido llevarse a cabo de forma exitosa.

Ilustración 48. Modal firma incorrecta

- **ModalAvisoAdjudicacionComponent:** Si el alumno ya tiene unas prácticas adjudicadas que no ha finalizado, no podrá apuntarse a alguna nueva. Esta ventana sirve para avisarle de eso en el caso de que intente seleccionar una nueva oferta.

Ilustración 49. Modal aviso adjudicación

app.component.ts
Como se puede apreciar en la Ilustración 33. Esquema Angular proyecto, todos los componentes de las pantallas extienden de *BaseComponent* el cual se encarga de facilitar todos los servicios necesarios.

El método *ngOnInit()* es lo primero que se ejecuta cuando se inicia un componente, antes de que se cargue la vista. En esta función indicaremos los componentes que queremos que se visualicen en la vista y se recoge el observable encargado de cargar los datos que se mostrarán en la pantalla. En la *Ilustración 50. Método ngOnInit()* se ve un ejemplo del código de tienen todos los componentes que cargan observables.

```
ngOnInit() {
  this.utilidades.anadirTrazaACola("PerfilEmpresaComponent.ngOnInit - " + this.contextService.idSesion + " - INI");
  this.inicializarValores();
  this.stateService.setHeader(true);
  this.stateService.setBreadcumb(true);
  this.contextService.moduloActual = 1;
  this.contextService.pasoActual = 4;

  this.instructores$ = this.contextEmpresaService.getInstructoresEmpresa$();
  this.instructores$.subscribe(
    () => {
      this.instructores = this.contextEmpresaService.getInstructoresEmpresa();
      this.stateService.setLoader(false);
    },
    error => {
      this.utilidades.anadirTrazaACola("PerfilEmpresaComponent.ngOnInit - " + this.contextService.idSesion + " - Error - " + JSON.stringify(error));
      this.stateService.setLoader(false);
      this.utilidades.lanzarErrorGeneral();
    }
  );
  this.utilidades.anadirTrazaACola("PerfilEmpresaComponent.ngOnInit - " + this.contextService.idSesion + " - FIN");
}
```

Ilustración 50. Método ngOnInit()

Todas las funciones tienen trazas para indicar el inicio y el final de dicha función y los elementos importantes que ocurran mientras se esté ejecutando.

PANTALLAS COMUNES

- **LoginComponent:** Esta es la primera pantalla que aparece cuando se inicia la aplicación. En el caso del alumno y el profesor, la pantalla será igual, pero en el caso de la empresa, los componentes de la pantalla cambian. La persona que desee acceder tiene que seleccionar el tipo de usuario que es y rellenar los datos de login.

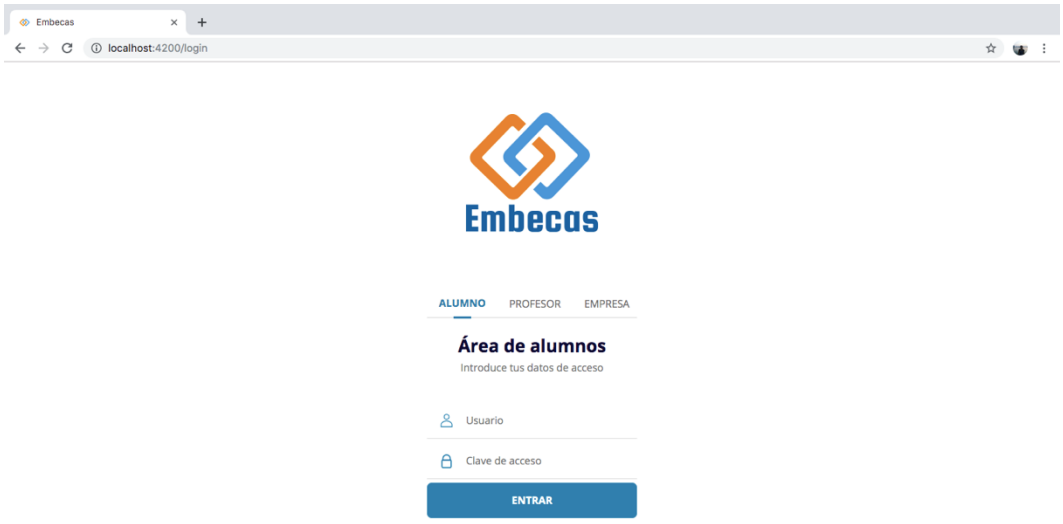

Ilustración 51. Pantalla login Alumno (ordenador)

A continuación, se muestra la apariencia del login en el caso de la empresa, donde podrá seleccionar el tipo de papel que representa el usuario dentro de la organización:

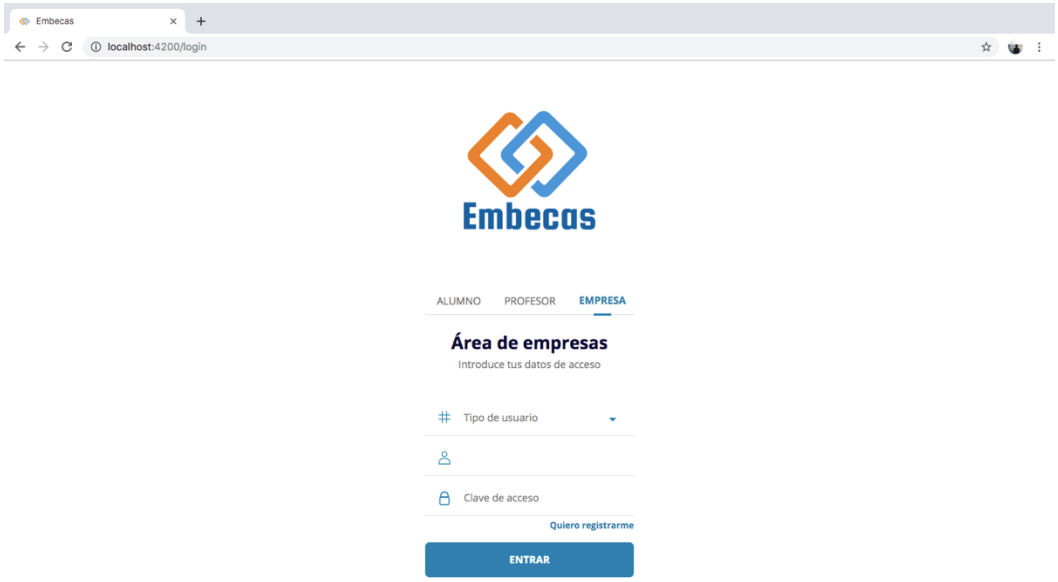

Ilustración 52. Pantalla login empresa (ordenador)

- **CambioClaveComponent:** A través de esta pantalla los usuarios podrán cambiar su contraseña. Para ello deben introducir la contraseña actual, la nueva y una confirmación de la nueva clave.

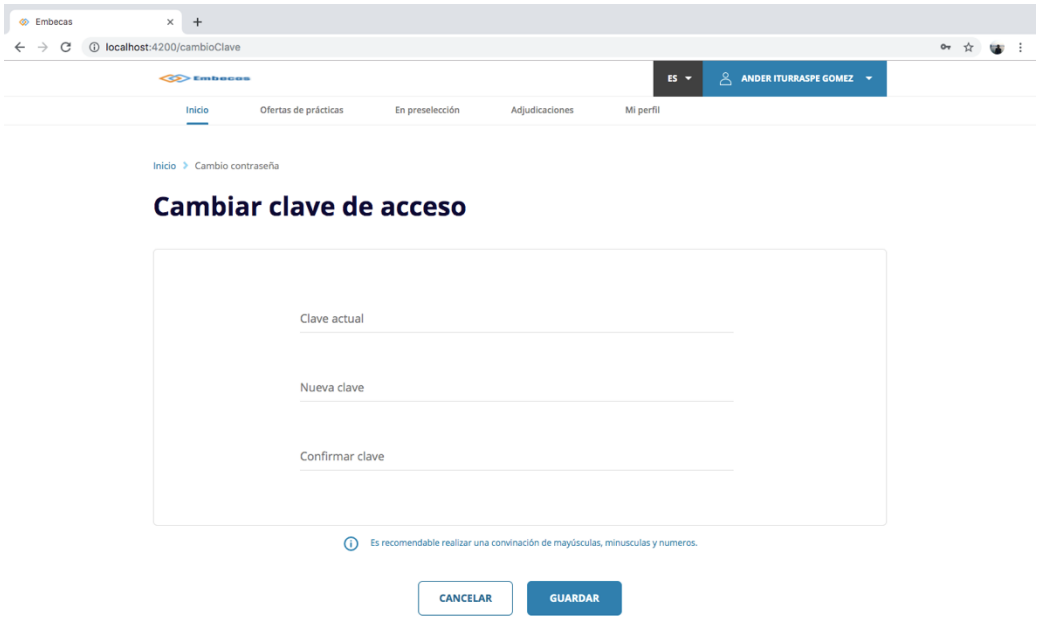

Ilustración 53. Pantalla cambiar clave (ordenador)

- **PaginaPrincipalComponent:** Esta será la primera pantalla que le aparezca al usuario cuando inicie sesión, donde aparecerán todas las opciones que tiene dentro de la aplicación. A cada usuario se le mostrarán las que corresponden al grupo de usuarios al que pertenece.

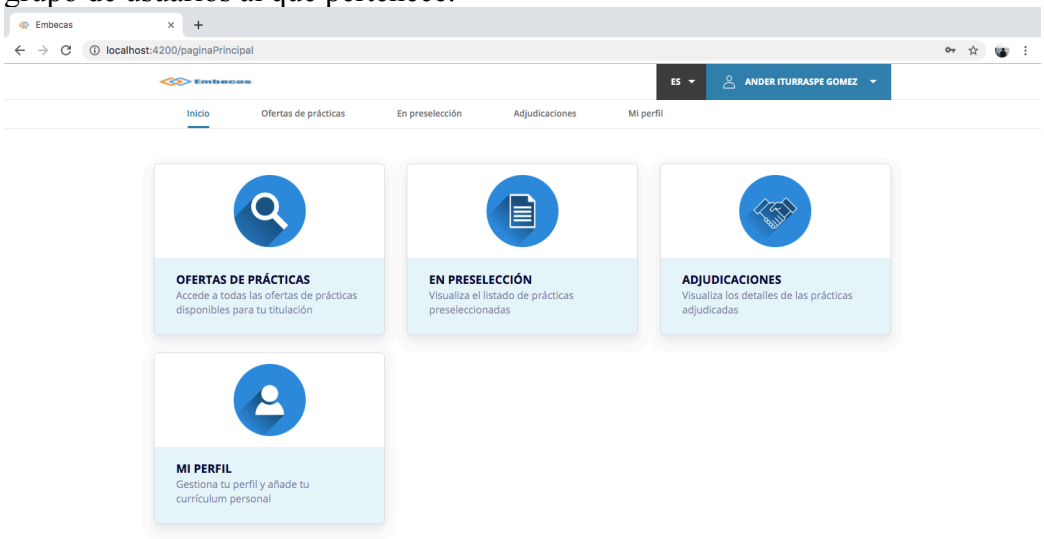

Ilustración 54. Pantalla principal alumno (ordenador)

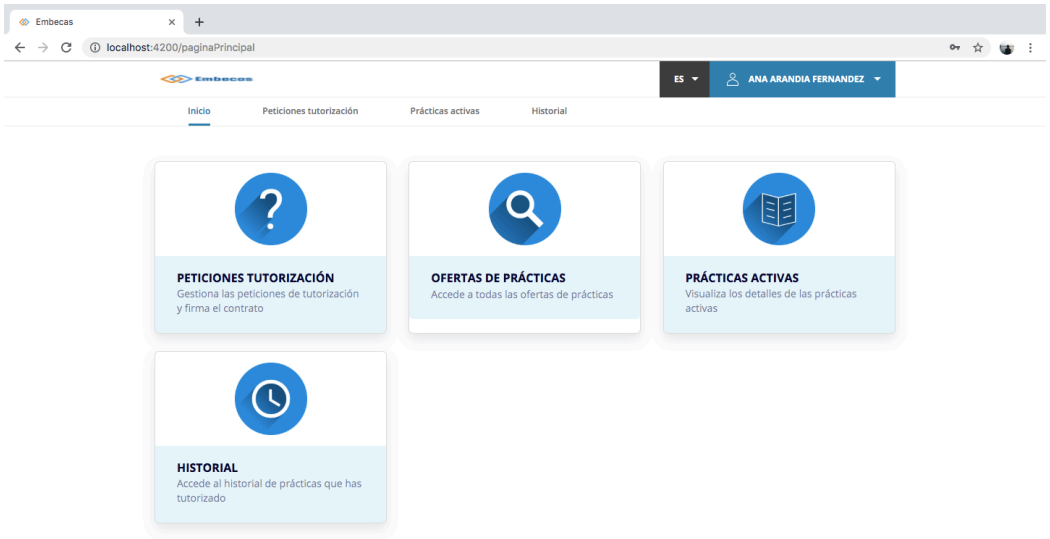

Ilustración 55. Pantalla principal profesor (ordenador)

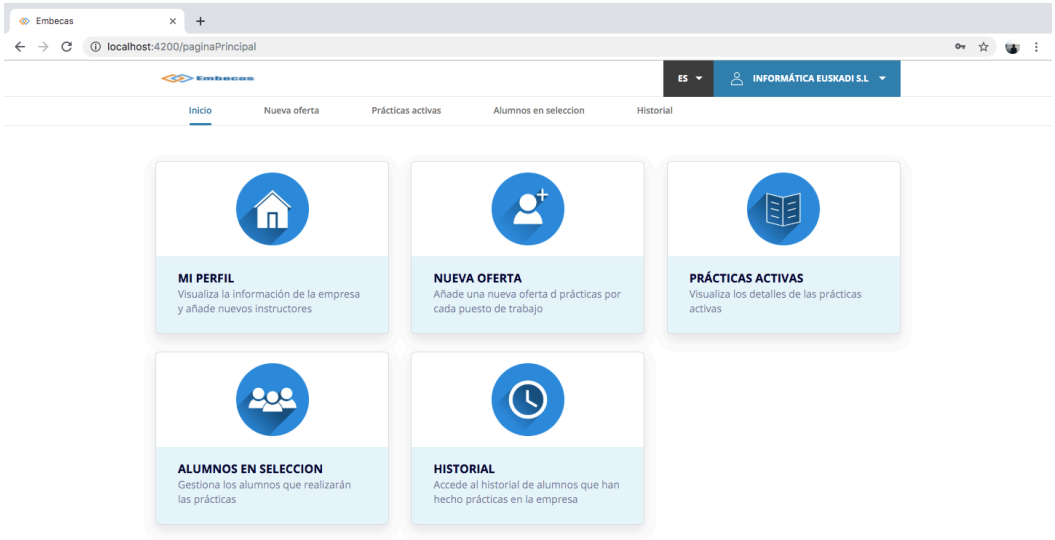

Ilustración 56. Pantalla principal recursos humanos (ordenador)

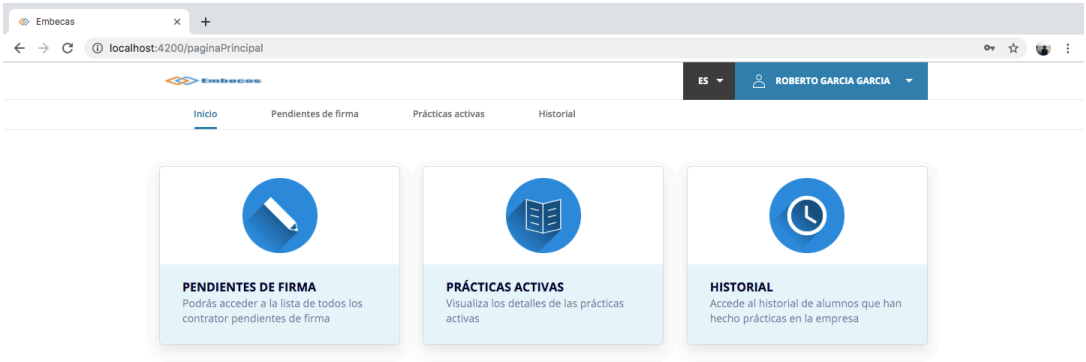

Ilustración 57. Pantalla principal instructor (ordenador)

PANTALLAS DEL ALUMNO

- **AdjudicacionesComponent:** En esta pantalla se visualizaran todas las prácticas que ha realizado el alumno, con su información más descriptiva.

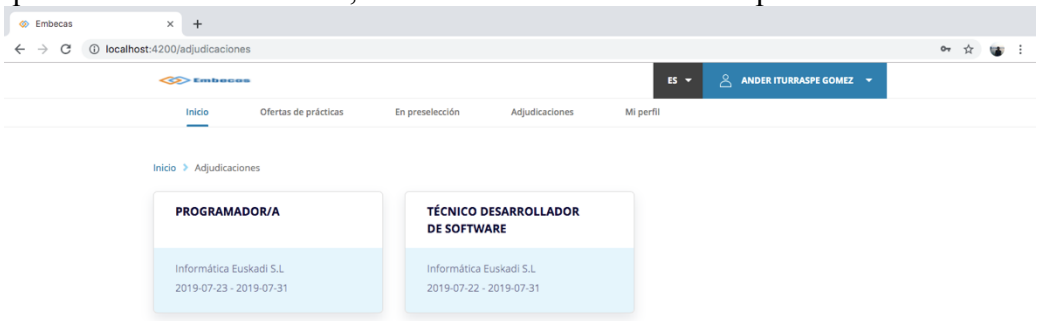

Ilustración 58. Pantalla adjudicaciones (ordenador)

- **InfoAdjudicacionesComponent:** Esta pantalla se mostrará una vez el alumno selecciona una de las adjudicaciones. En ella se ven los datos de todos los implicados en las prácticas, así como la información de la empresa, el instructor, el profesor, el alumno y la oferta.

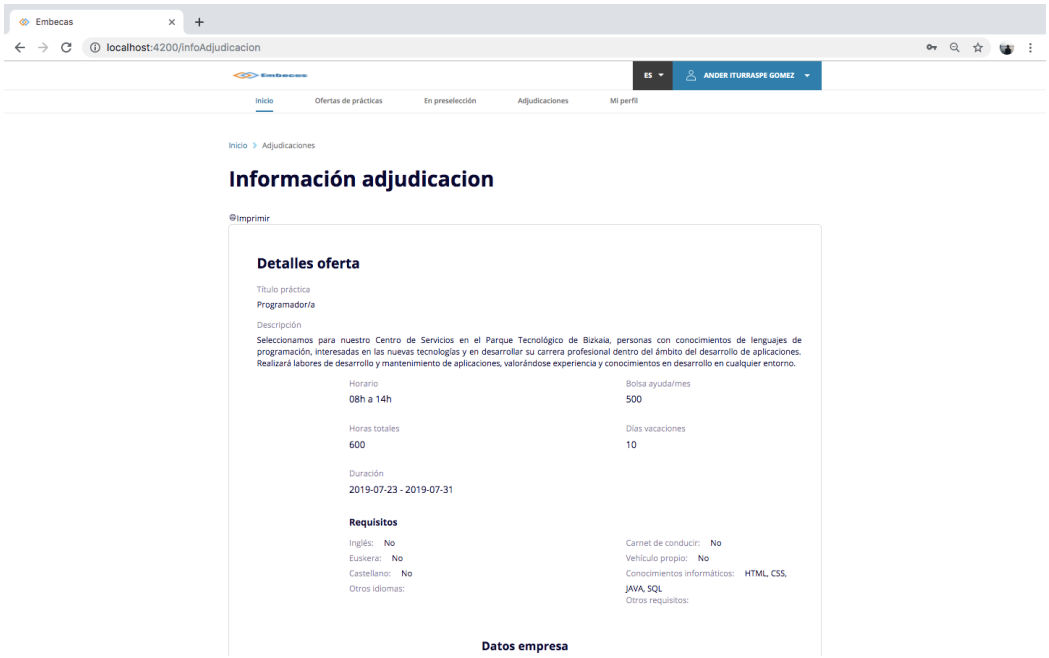

Ilustración 59. Pantalla información adjudicación (ordenador)

- **MiPerfilComponent:** En esta pantalla el alumno puede ver sus datos personales y el currículum, en el caso de tenerlo subido. De lo contrario, puede adjuntar uno pulsando el botón "subir currículum" que se redirecciona a la subida de documentación.

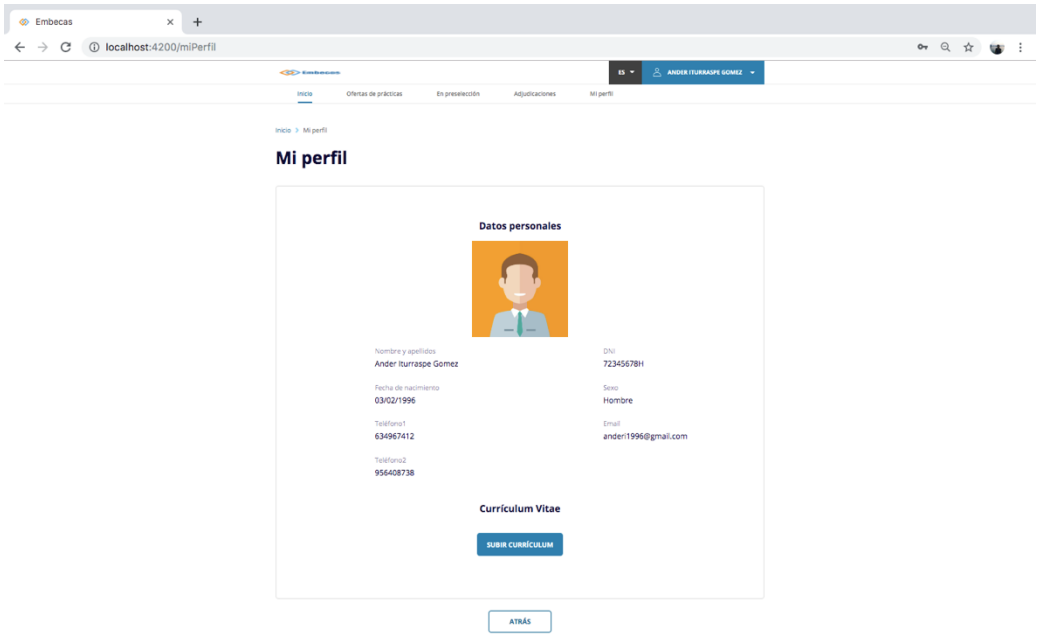

Ilustración 60. Pantalla perfil alumno (ordenador)

- **OfertasDePracticasComponent:** En esta pantalla el alumno puede ver el listado de prácticas ofertadas de empresas para su titulación. Lo primero que se ve de cada una de ellas es la información más relevante, como el título de las prácticas, la empresa, la duración y la ayuda económica. Una vez el alumno siente interés en una oferta, puede pinchar encima y acceder a la información más detallada. El estudiante puede apuntarse a todas las ofertas deseadas, seleccionando el checkbox y pulsando el botón "Guardar". Para realizar una

búsqueda filtrada, puede seleccionar la provincia e incluso la localidad en la que desea trabajar.

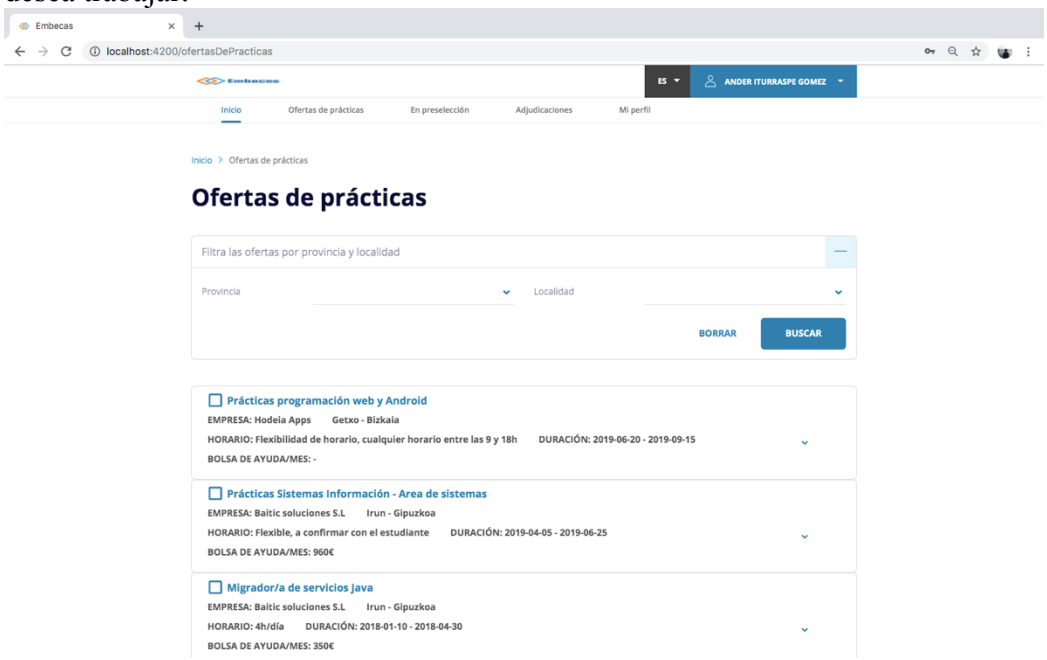

Ilustración 61. Pantalla ofertas de prácticas 1 (ordenador)

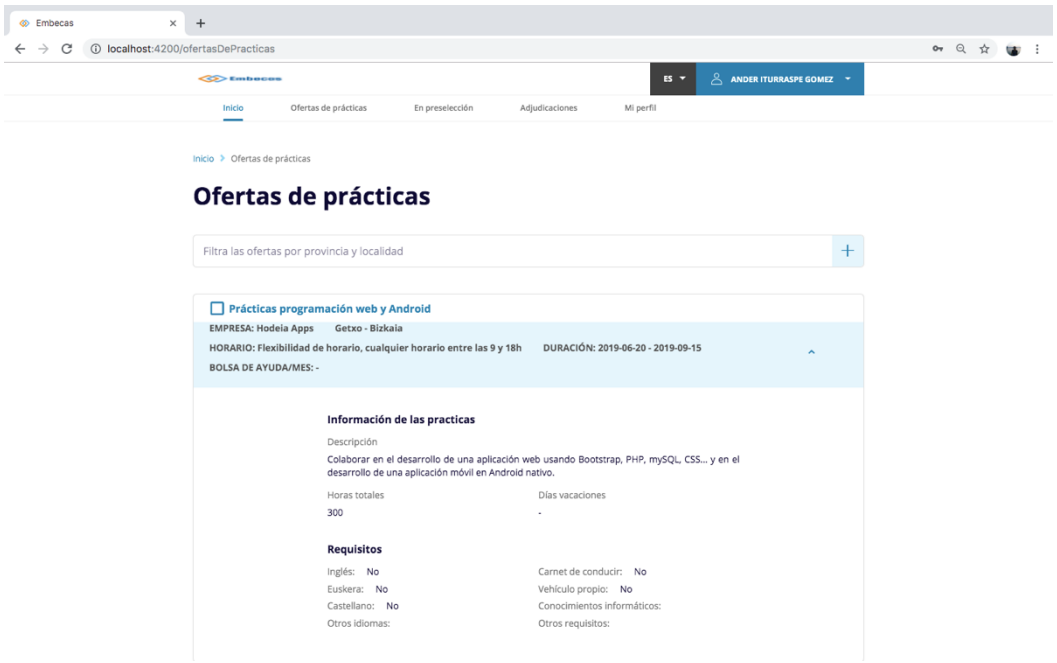

Ilustración 62. Pantalla ofertas de prácticas 2 (ordenador)

- **PreseleccionesComponent:** En esta pantalla el alumno tiene una lista con todas las ofertas a las que se ha apuntado y el estado en el que se encuentra cada una de ellas. La decisión de la oferta final que quiera realizar la tomará en esta pantalla, seleccionando el radioButton correspondiente a esa oferta.

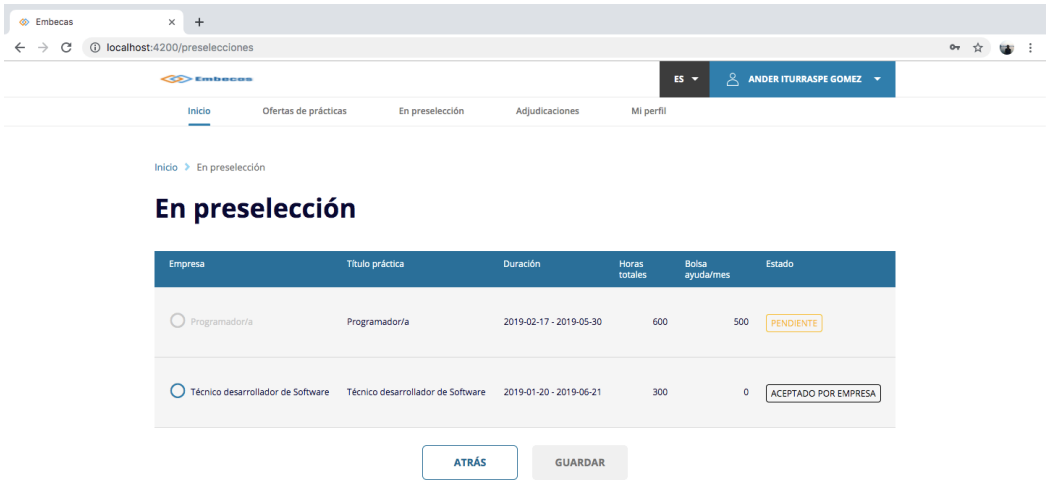

Ilustración 63. Pantalla preselecciones (ordenador)

PANTALLAS DE LA EMPRESA

- **ActivasEmpresaComponent:** Esta pantalla muestra una tabla con todas las prácticas que están actualmente activas en la empresa. En la tabla se visualizan tanto los datos de la oferta como los del alumno. La pantalla no tiene ninguna funcionalidad, excepto la de mostrar los datos más relevantes de las ofertas actuales.

Ilustración 64. Pantalla prácticas activas empresa (ordenador)

- **HistorialEmpresaComponent:** Esta pantalla muestra una tabla con un historial de todas las prácticas realizadas en la empresa, las cuales ya no están activas.

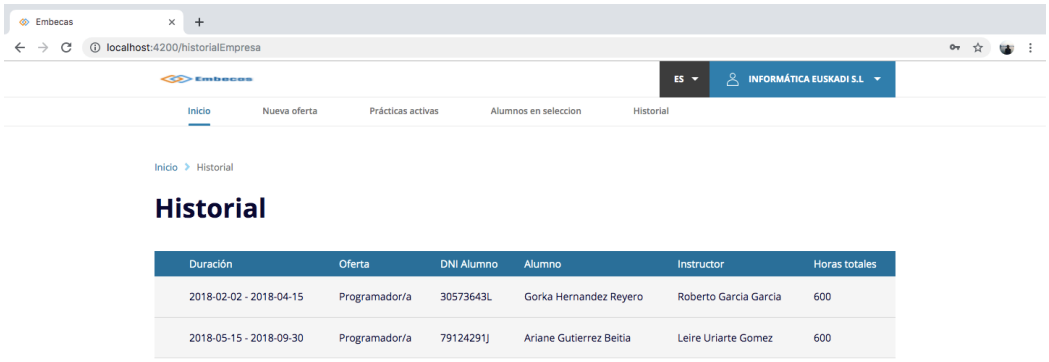

Ilustración 65. Pantalla historial empresa (ordenador)

- **NuevaOfertaComponent:** Si la empresa quiere añadir una nueva oferta de prácticas, deberá rellenar todos los campos que se muestran en esta pantalla con la información del trabajo. Dentro de la funcionalidad de la pantalla, hay validaciones que disminuyen la probabilidad de que ocurran fallos en el servicio o de que el usuario introduzca datos sin sentido, como letras o negativos en los valores numéricos. Además comprueba que las fechas introducidas como inicio y fin de las prácticas sean superiores al día de hoy, y que la de inicio sea inferior a la de fin.

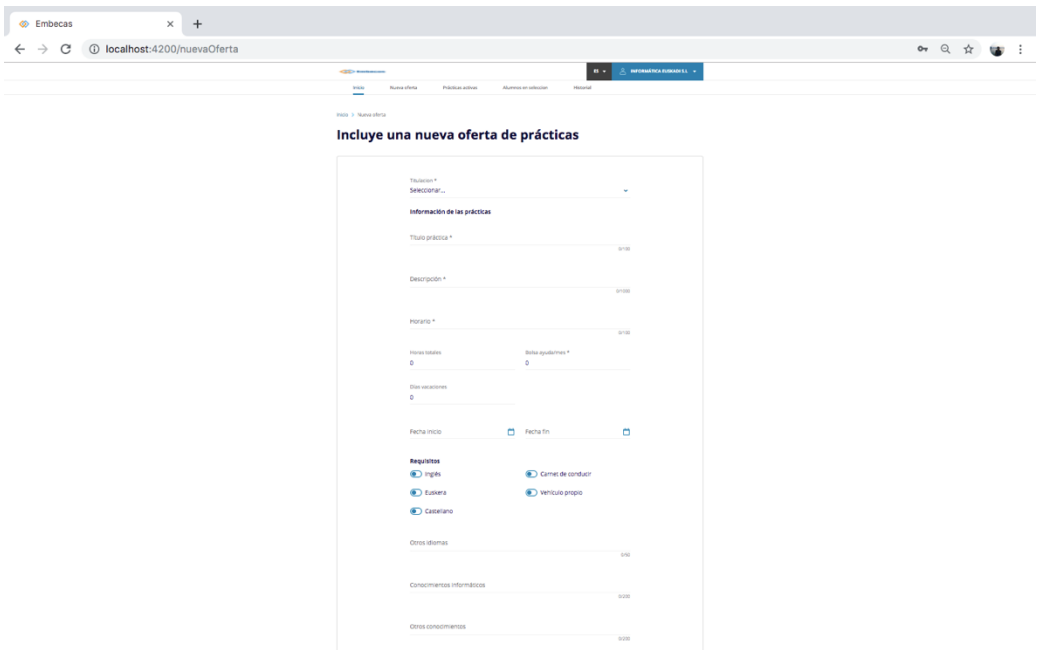

Ilustración 66. Pantalla nueva oferta (ordenador)

- **AlumnosEnSeleccionComponent:** El representante de recursos humanos tendrá acceso a esta pantalla, donde podrá ver todos los alumnos que se han apuntado a cada práctica ofertada de la empresa. En ella, podrá rechazar a los

alumnos que no se adecuen al trabajo y seleccionar al más conveniente. También podrán ver el currículum de un si pinchan encima de su DNI.

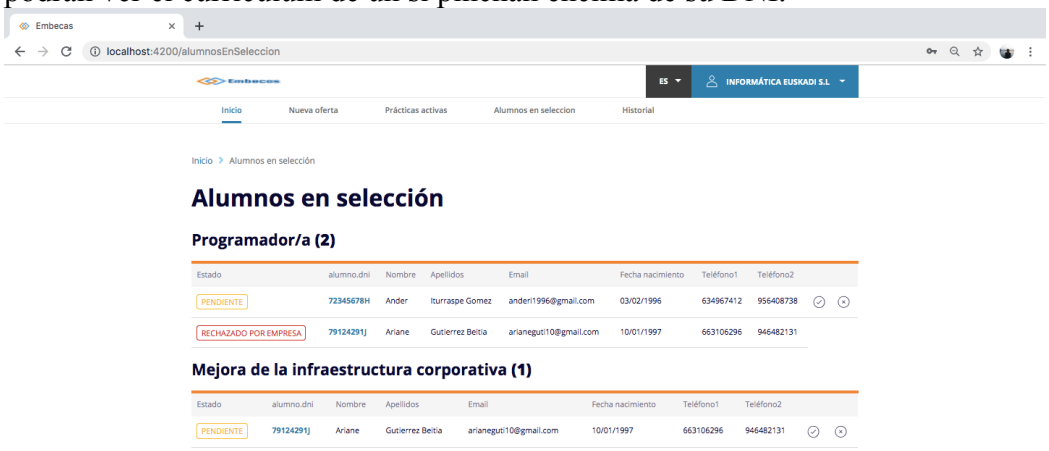

Ilustración 67. Pantalla alumnos en selección (ordenador)

- **PerfilEmpresaComponent:** En esta pantalla recursos humanos puede ver la información de la empresa y una tabla con todos sus instructores. Además, podrá añadir uno nuevo a la lista mediante el botón "añadir instructor".

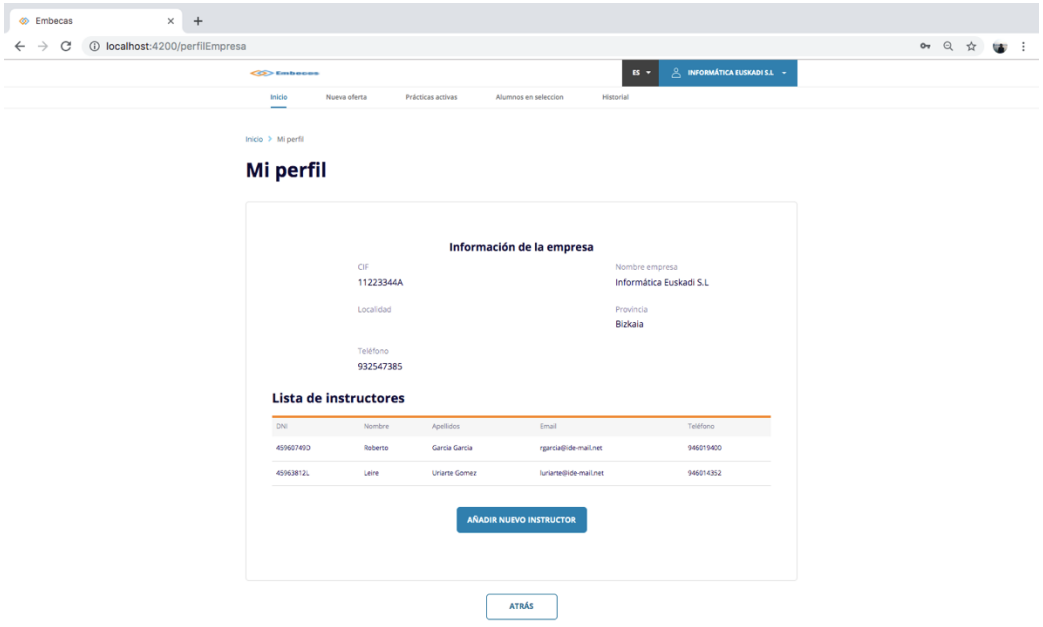

Ilustración 68. Pantalla perfil empresa (ordenador)

- **RegistroEmpresaComponent:** Mediante esta pantalla una empresa tendrá la opción de registrarse en la aplicación para ofrecer puestos de prácticas para los alumnos del centro.

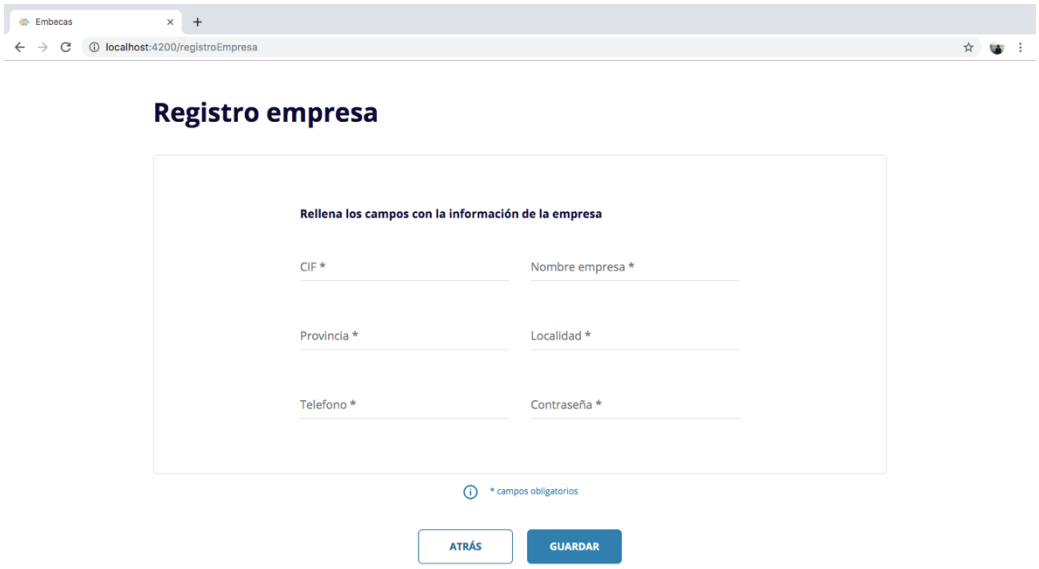

Ilustración 69. Pantalla registro empresa (ordenador)

PANTALLAS DEL PROFESOR

- **ActivasProfesorComponent:** Esta pantalla tiene una tabla con todas las prácticas activas en las que está formando parte el profesor. La funcionalidad de esta sección es mantener informado al profesor de los alumnos a los que tutoriza y poder realizar la evaluación final dando por finalizado su trabajo.

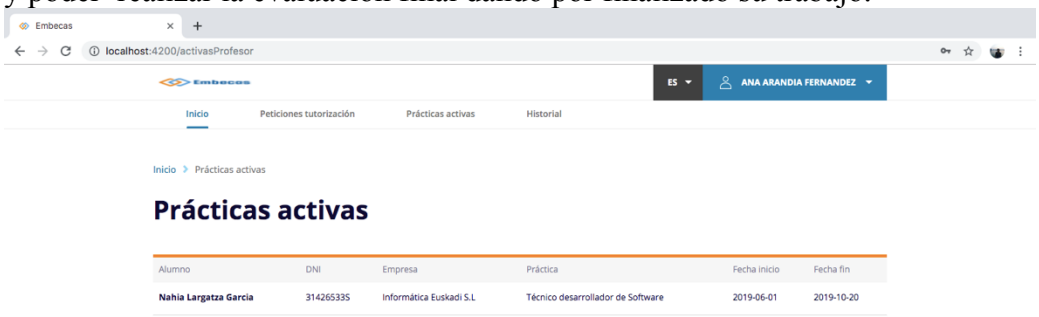

Ilustración 70. Pantalla peticiones tutorización profesor (ordenador)

- **HistorialProfesorComponent:** Esta pantalla muestra una tabla con un historial de todas las prácticas tutorizadas por el profesor, las cuales ya no están activas.

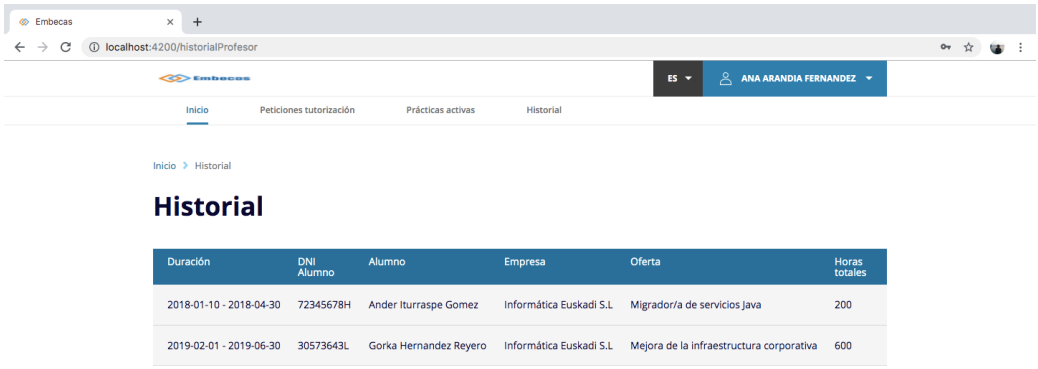

Ilustración 71. Pantalla historial profesor (ordenador)

- **OfertasDePracticasProfesorComponent:** En esta pantalla el profesor puede acceder a la información de las prácticas ofertadas para la titulación a la que está asignado. También tendrá una sección de búsqueda para que pueda filtrar esas ofertas por provincia y localidad.

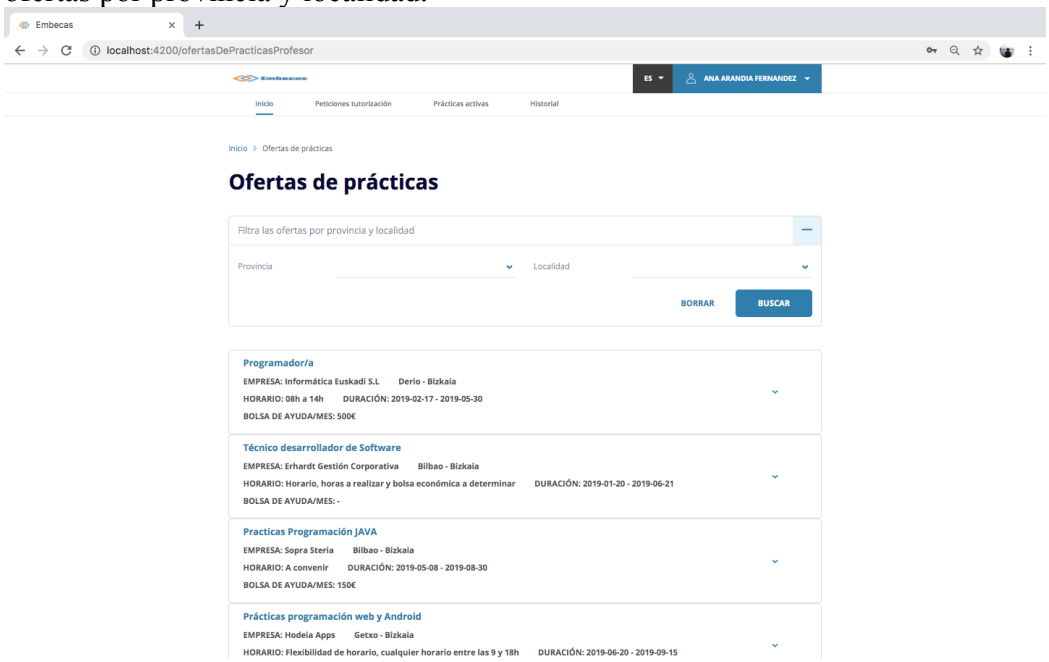

Ilustración 72. Pantalla ofertas de prácticas profesor (ordenador)

- **PeticionesTutorizaciónComponent:** Los profesores pueden ver las peticiones de tutorización en esta pantalla. Tendrá dos botones con los que podrá aceptar o rechazar dicho trabajo.

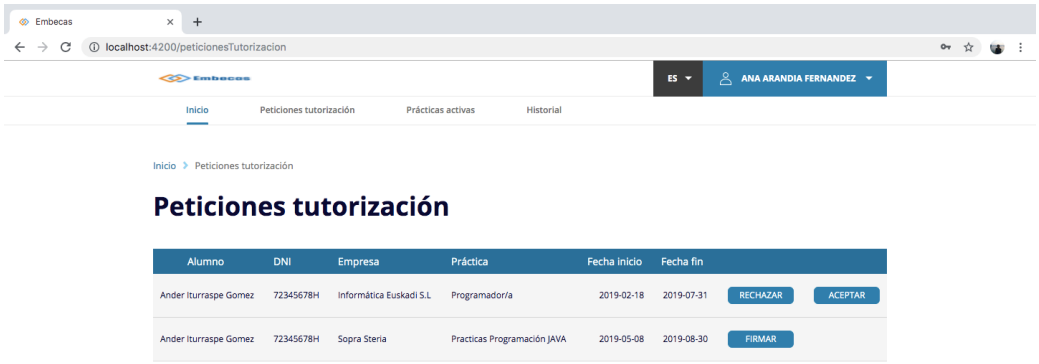

Ilustración 73. Pantalla peticiones tutorización profesor (ordenador)

PANTALLAS DISPOSITIVOS MÓVILES

En este pequeño apartado se mostrarán las pantallas adaptadas a dispositivos móviles que más cambian en cuanto al diseño.

Ilustración 74. Pantalla peticiones tutorización (móvil)

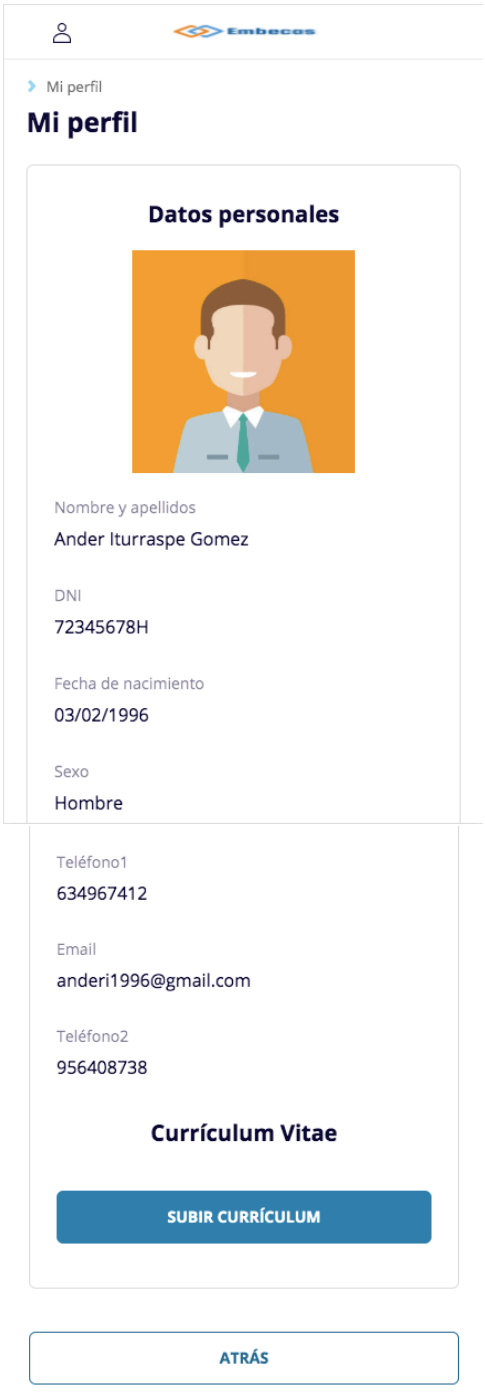

Ilustración 75. Pantalla perfil alumno (móvil)

Ilustración 76. Pantalla preselecciones (móvil)

Ilustración 77. Pantalla ofertas de prácticas (móvil)

6.2. Estructura de Eclipse

En cuanto a la segunda estructura, se ha usado dynamic web project para crear y exponer los servicios mediante un servidor Tomcat. En la siguiente ilustración se puede ver la organización de esta parte de la aplicación.

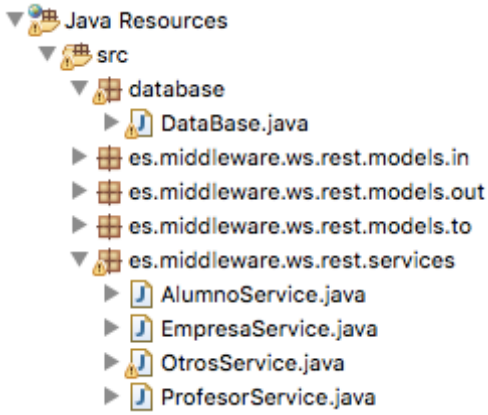

Ilustración 78. Estructura de Eclipse

A continuación, se explican los paquetes, clases, atributos y métodos necesarios para la programación de la parte REST que está implementada en Java usando la herramienta de Eclipse.

6.2.1. Database

En este paquete se encuentra la clase *singleton* **DataBase**, que gestiona todas las llamadas a la base de datos.

6.2.2. es.middleware.ws.rest.models

En este paquete se encuentran todas las clases que sirven como modelos de estructuras para guardar datos, que sirven como entrada y salida de los diferentes servicios. Estas clases guardan los atributos necesarios y sus debidos getters y setters.

Cada uno de los servicios tendrá un parámetro de entrada y otro de salida. Desde la aplicación Angular, estos parámetros se envían en formato JSON [6], y después, se transforman en un objeto de la clase correspondiente con el siguiente ejemplo de código, siendo "alumno" el parámetro JSON de entrada y queriendo transformar en la clase "DetalleAlumnoTO":

```
DetalleAlumnoTO detalleAlumno  = new DetalleAlumnoTO();
detalleAlumno = new Gson().fromJson(alumno, DetalleAlumnoTO.class);
```

Estos modelos se han dividido en 3 paquetes:

- **in:** Estos modelos son los parámetros de entrada, que estarán compuestos por los atributos necesarios para llevar a cabo la petición en la parte REST.
- **to:** Estos modelos son los encargados de guardar los datos obtenidos de la base de datos, así como la información personal del usuario, los datos de las prácticas ofertadas, etc.
- **out:** Estos modelos representan los parámetros de salida.

6.2.3. es.middleware.ws.rest.services

En este paquete se guardan las cuatro clases que poseen todos los servicios.

OtrosService

Esta clase contiene todos los servicios oportunos para validar a todo tipo de usuarios.

validarAlumno

URL: http://localhost:8080/TFGMiddleware/rest/otros/validarAlumno
Descripción: Este servicio sirve para identificar al alumno que accede a la aplicación, comprobando a quien corresponden el usuario y contraseña introducidos.
Parámetro entrada: LoginIN
Parámetro salida: DetalleAlumnoOUT

validarProfesor

URL: http://localhost:8080/TFGMiddleware/rest/otros/validarProfesor
Descripción: Este servicio sirve para identificar al profesor que accede a la aplicación, comprobando si el usuario y contraseña introducidos son correctos y corresponden a algún usuario.
Parámetro entrada: LoginIN
Parámetro salida: DetalleProfesorOUT

validarEmpresa

URL: http://localhost:8080/TFGMiddleware/rest/otros/validarEmpresa
Descripción: Este servicio sirve para identificar a la empresa que accede a la aplicación, comprobando si el usuario y contraseña introducidos son correctos. El servicio nos devuelve toda la información sobre la empresa que ha iniciado sesión y un mensaje para saber si no ha habido ningún problema para realizar la comprobación de autenticación.
Parámetro entrada: LoginIN
Parámetro salida: DetalleEmpresaOUT

validarInstructor

URL: http://localhost:8080/TFGMiddleware/rest/otros/validarInstructor
Descripción: Este servicio sirve para identificar al instructor que accede a la aplicación, comprobando si el usuario y contraseña introducidos son correctos. El servicio nos devuelve toda la información sobre el instructor que ha iniciado sesión y un mensaje para saber si no ha habido ningún problema para realizar la comprobación de autenticación.
Parámetro entrada: LoginIN
Parámetro salida: DetalleInstructorOUT

cambiarClave

URL: http://localhost:8080/TFGMiddleware/rest/login/cambiarClave
Descripción: Este servicio sirve para cambiar la clave del usuario que lo requiere.
Parámetro entrada: CambioClaveIN
Parámetro salida: MensajeBaseOUT

actualizarDocumentoContrato

URL: http://localhost:8080/TFGMiddleware/rest/otros/actualizarDocumentoContrato
Descripción: Este servicio sirve para actualizar el contrato cada vez que un usuario lo firme.
Parámetro entrada: ActualizarDocumentoIN
Parámetro salida: MensajeBaseOUT

AlumnoService

Esta clase contiene todos los servicios relacionados con la sesión del alumno, por lo que sólo están accesibles si el tipo de usuario que accede a la aplicación es un estudiante.

listaOfertasDePracticas

URL: http://localhost:8080/TFGMiddleware/rest/alumno/listaOfertasDePracticas
Descripción: Este servicio devuelve todas las prácticas que están ofertadas para la titulación correspondiente al alumno que ha solicitado la información, sin tener en cuenta las que ya ha escogido para la preselección.
Parámetro entrada: OfertasDePracticasIN
Parámetro salida: OfertaDePracticasOUT

incluirPreselecciones

URL: http://localhost:8080/TFGMiddleware/rest/alumno/incluirPreselecciones
Descripción: Este servicio sirve para incluir en la base de datos las ofertas seleccionadas por el alumno. Es decir, todas las ofertas que le interesen de la lista de "ofertas de prácticas" se añadirán a la lista de preselecciones. La respuesta del servicio es un mensaje para informar si se ha podido llevar a cabo la tarea.
Parámetro entrada: IncluirPreseleccionesIN
Parámetro salida: MensajeBaseOUT

listaPreselecciones

URL: http://localhost:8080/TFGMiddleware/rest/alumno/listaPreselecciones
Descripción: Gracias a este servicio, recuperamos la lista de ofertas que el alumno a preseleccionado y el estado en el que se encuentra cada una de ellas.
Parámetro entrada: DetalleAlumnoTO
Parámetro salida: PreseleccionesOUT

enviarArchivo

URL: http://localhost:8080/TFGMiddleware/rest/alumno/enviarArchivo
Descripción: Este servicio sirve para actualizar el currículum del alumno. La respuesta del servicio es la información del alumno actualizada y un mensaje para informar si se ha podido llevar a cabo la tarea.
Parámetro entrada: EnviarArchivoIN
Parámetro salida: DetalleAlumnoOUT

profesoresTitulación

URL: http://localhost:8080/TFGMiddleware/rest/alumno/profesoresTitulacion
Descripción: El servicio devuelve la lista con todos los profesores que pertenecen a la titulación enviada como parámetro de entrada.
Parámetro entrada: titulación: String
Parámetro salida: ProfesoresTitulacionOUT

adjudicaciones

URL: http://localhost:8080/TFGMiddleware/rest/alumno/adjudicaciones
Descripción: El servicio devuelve una lista con todas las prácticas que ha realizado el alumno.
Parámetro entrada: dniAlumno: String
Parámetro salida: AdjudicacionesOUT

guardarSeleccion

URL: http://localhost:8080/TFGMiddleware/rest/empresa/guardarSeleccion
Descripción: Este servicio realiza todas las operaciones necesarias cuando el alumno selecciona la oferta final que quiere realizar, como por ejemplo, actualizar el estado de la oferta que ha elegido y de las que ha rechazado. El servicio devuelve un mensaje para informar si todo se ha ejecutado correctamente.
Parámetro entrada: GuardarSeleccionIN
Parámetro salida: MensajeBaseOUT

elegirProfesor

URL: http://localhost:8080/TFGMiddleware/rest/empresa/elegirProfesor
Descripción: Como consecuencia de este servicio se actualiza el profesor seleccionado para realizar las prácticas. Este ajuste se realiza en la preselección, ya que el profesor todavía no ha validado tu decisión.
Parámetro entrada: GuardarProfesorIN
Parámetro salida: MensajeBaseOUT

firmaAlumno

URL: http://localhost:8080/TFGMiddleware/rest/empresa/firmaAlumno
Descripción: Este servicio actualiza el atributo de confirmación de la firma del alumno.
Parámetro entrada: id: integer
Parámetro salida: MensajeBaseOUT

EmpresaService

Esta clase contiene todos los servicios relacionados con la sesión de la empresa, tanto para el representante de recursos humanos como para los instructores.

listaTitulaciones

URL: http://localhost:8080/TFGMiddleware/rest/empresa/listaTitulaciones
Descripción: Este servicio devuelve un listado con todas las titulaciones en las que hay ofertas de prácticas, junto con un mensaje para describir si todo ha ido bien.
Parámetro entrada: -
Parámetro salida: TitulacionesOUT

nuevaOferta

URL: http://localhost:8080/TFGMiddleware/rest/empresa/nuevaOferta
Descripción: Este servicio incluye una nueva oferta de prácticas en la tabla de ofertas. Recibe como parámetro de entrada toda la información relacionada con las condiciones y descripción del trabajo a realizar y devuelve un mensaje de error para anunciar si la oferta se ha añadido correctamente.
Parámetro entrada: DetalleOfertaTO
Parámetro salida: MensajeBaseOUT

alumnosEnSeleccion

URL: http://localhost:8080/TFGMiddleware/rest/empresa/alumnosEnSeleccion
Descripción: Este servicio devuelve un listado con todos los alumnos que están en la lista de selección de cada oferta realizada por la empresa, además de un mensaje para informar del estado de la petición a la base de datos, para saber si todo se ha recuperado bien.
Parámetro entrada: cifEmpresa: String
Parámetro salida: AlumnosEnSeleccionOUT

rechazarAlumno

URL: http://localhost:8080/TFGMiddleware/rest/empresa/rechazarAlumno
Descripción: Este servicio sirve para rechazar al alumno que tiene la empresa en proceso de selección. Utilizando el dni del alumno y el id de la oferta de la que se trata, al alumno se le notifica el nuevo estado de la oferta. La respuesta de este servicio es una lista actualizada con el estado y los alumnos que están apuntados a cada oferta, más un mensaje para notificar si la actualización se ha realizado correctamente.
Parámetro entrada: OfertaAlumnoIN
Parámetro salida: AlumnosEnSeleccionOUT

historialEmpresa

URL: http://localhost:8080/TFGMiddleware/rest/empresa/historialEmpresa
Descripción: Este servicio sirve para recuperar una lista con el historial de alumnos y prácticas que se han realizado en la empresa, ordenadas cronológicamente.
Parámetro entrada: cifEmpresa: String
Parámetro salida: InfoPracticaEmpresaOUT

aceptarAlumno

URL: http://localhost:8080/TFGMiddleware/rest/empresa/aceptarAlumno
Descripción: Este servicio sirve para aceptar al alumno que tiene la empresa en proceso de selección. Utilizando el dni del alumno y el id de la oferta en la que se le acepta, al alumno se le notifica el nuevo estado de la oferta. La respuesta de este servicio es una lista actualizada con el estado y los alumnos que están apuntados a cada oferta, más un mensaje para notificar si la actualización se ha realizado correctamente.
Parámetro entrada: OfertaAlumnoIN
Parámetro salida: AlumnosEnSeleccionOUT

añadirNuevaEmpresa

URL: http://localhost:8080/TFGMiddleware/rest/empresa/añadirNuevaEmpresa
Descripción: Este servicio sirve para registrar a una nueva empresa en la aplicación. Para ello, la representante de recursos humanos deberá proporcionar la información más relevante acerca de la empresa.
Parámetro entrada: DetalleEmpresaTO
Parámetro salida: MensajeBaseOUT

instructoresEmpresa

URL: http://localhost:8080/TFGMiddleware/rest/empresa/instructoresEmpresa
Descripción: Este servicio sirve para recoger una lista con toda la información de los instructores que pertenecen a la empresa.
Parámetro entrada: cifEmpresa: String
Parámetro salida: InstructoresOUT

añadirInstructor

URL: http://localhost:8080/TFGMiddleware/rest/empresa/añadirInstructor
Descripción: Este servicio añade un nuevo instructor a la base de datos. Para ello, se debe facilitar toda la información que se solicita a cerca del instructor. Este servicio devuelve la lista de instructores actualizada.
Parámetro entrada: DetalleInstructorTO
Parámetro salida: InstructoresOUT

actualizarOferta

URL: http://localhost:8080/TFGMiddleware/rest/empresa/actualizarOferta
Descripción: Cuando la empresa elige a un alumno para realizar las prácticas, tiene que actualizar las condiciones de esta, adecuando la duración, los días de vacaciones que corresponden a ese tiempo y el instructor. Este servicio sirve realizar esos cambios.
Parámetro entrada: ActualizaciónOfertaIN
Parámetro salida: MensajeBaseOUT

activasEmpresa

URL: http://localhost:8080/TFGMiddleware/rest/empresa/activasEmpresa
Descripción: Este servicio devuelve todas las ofertas que la empresa tiene activas en el momento, las que no han finalizado todavía, ya sea porque la fecha final no ha llegado o

porque no han realizado el informe final que da por finalizada la relación entre práctica y alumno.
Parámetro entrada: cifEmpresa: String
Parámetro salida: InfoPracticaEmpresaOUT

peticionesFirmaInstructor

URL: http://localhost:8080/TFGMiddleware/rest/empres/peticionesFirmaInstructor
Descripción: Este servicio devuelve un listado con todas las prácticas que el instructor debe firmar para dar su aprobación y que la selección siga adelante.
Parámetro entrada: dniInstructor: String
Parámetro salida: InfoPracticaEmpresaOUT

firmaInstructor

URL: http://localhost:8080/TFGMiddleware/rest/empresa/firmaInstructor
Descripción: Este servicio actualiza el atributo de confirmación de la firma de la empresa.
Parámetro entrada: id: integer
Parámetro salida: MensajeBaseOUT

ProfesorService

historialProfesor

URL: http://localhost:8080/TFGMiddleware/rest/profesor/historialProfesor
Descripción: Este servicio sirve para recuperar una lista con el historial de alumnos y prácticas que ha tutorizado el profesor que corresponde al DNI de entrada, ordenadas cronológicamente.
Parámetro entrada: dniInstructor: String
Parámetro salida: InfoPracticaProfesorOUT

listaOfertasDePracticas

URL: http://localhost:8080/TFGMiddleware/rest/profesor/listaOfertasDePracticas
Descripción: Este servicio devuelve todas las práctica ofertadas para la titulación relacionada con el profesor.
Parámetro entrada: OfertasDePracticasIN
Parámetro salida: OfertasDePracticasOUT

firmaProfesor

URL: http://localhost:8080/TFGMiddleware/rest/profesor/firmaProfesor
Descripción: Este servicio actualiza el atributo de confirmación de la firma de la empresa.
Parámetro entrada: id: integer
Parámetro salida: MensajeBaseOUT

peticionesTutorizacion

URL: http://localhost:8080/TFGMiddleware/rest/profesor/peticionesTutorizacion
Descripción: Este servicio recoge todas las peticiones de tutorización que tiene un profesor.
Parámetro entrada: dniProfesor: String
Parámetro salida: InfoPracticasProfesorOUT

aceptarPeticion

URL: http://localhost:8080/TFGMiddleware/rest/profesor/aceptarPeticion
Descripción: Este servicio notifica la decisión del profesor con respecto a la tutorización de un alumno.
Parámetro entrada: GuardarProfesorIN
Parámetro salida: InfoPracticasProfesorOUT

activasProfesor

URL: http://localhost:8080/TFGMiddleware/rest/profesor/activasProfesor
Descripción: Este servicio devuelve una lista con todas las ofertas que el profesor está tutorizando actualmente, es decir, que no han finalizado todavía, ya sea porque la fecha final no ha llegado o porque no realizado la evaluación final.
Parámetro entrada: dniProfesor: String
Parámetro salida: InfoPracticasProfesorOUT

6.5. Conexión a la base de datos

Para que Eclipse tenga acceso a la base de datos, hay que realizar cierta configuración que se explica a continuación.

En primer lugar se necesita un conector, que se comporte como capa intérprete. El objetivo de este tipo de conectores es hacer posible el acceder a las bases de datos desde cualquier aplicación. En este proyecto se utiliza la versión 5.1.47 del mysql-connector (mysql-connector-java-5.1.47.jar)

Para finalizar, hay que crear la conexión. Para ello se necesitan la localización donde se encuentra instalada la base de datos, en este caso en el puerto 3306 de localhost, el nombre de la base de datos y el usuario y contraseña en el caso de necesitarlas para acceder. En la siguiente ilustración se puede ver esa configuración:

```
String url = "jdbc:mysql://localhost:3306/TFG?useSSL=true";
String user = "root";
String password = "*******";
String driver = "com.mysql.jdbc.Driver";
private static DataBase nDataBase = null;
Connection conn;

/********************* CONSTRUCTOR **********************/
public DataBase() {
    try {
        Class.forName(driver).newInstance();
        conn = (Connection) DriverManager.getConnection(url, user, password);
        System.out.println("Conexion a la base de datos establecida!");
    } catch (Exception e) {
        System.out.println("MySQL exception: " + e.getMessage());
    }
}
```

Ilustración 79. Conexión base de datos

6.6. Creación de los servicios REST

Para el desarrollo de los servicios de este proyecto, se ha utilizado el framework de Jersey, de código abierto, que proporciona soporte para la API de JAX-RS.

JAX-RS (Java API for RESTful Web Services) es una API de Java que proporciona soporte para la creación de servicios web de acuerdo con la arquitectura de REST. JAX-RS usa anotaciones para simplificar el progreso de creación de los servicios, las cuales se explicarán más adelante.

Lo primero que se necesita para preparar el entorno es proporcionar al proyecto las dependencias o librerías de Jersey necesarias. En este caso, se ha generado un Dynamic Web Project llamado "TFGMiddleware" por lo que se importaron los archivos **jersey-client.jar**, **jersey-server.jar** y **javax.ws.rs-api-2.1.1.jar**.

En segundo lugar, hay que crear la clase que proporciona el servicio REST. En este caso habrá cuatro clases distintas en un mismo paquete y cada clase corresponderá a una URI diferente.

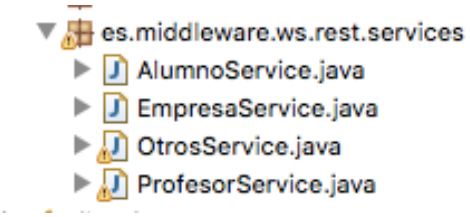

Ilustración 80. Servicios de Eclipse

Se puede ver la estructura de la clase *AlumnoService* en la siguiente ilustración.

```
@Path("/alumno/")
@Consumes({MediaType.APPLICATION_JSON})
@Produces({MediaType.APPLICATION_JSON})
public class AlumnoService {


    @POST
    @Path("/listaOfertasDePracticas")

    public OfertaDePracticasOUT listaOfertasDePracticas(String ofertaDePracticaIN) {

        OfertasDePracticasIN ofertaIN  = new OfertasDePracticasIN();
        ofertaIN = new Gson().fromJson(ofertaDePracticaIN, OfertasDePracticasIN.class);

        return DataBase.getDataBase().listaOfertas(ofertaIN);

    }
```

Ilustración 81. Clase servicio REST

Este es el significado de cada anotación:

- **@Path:** Identifica la URI de una clase o método que sirve las peticiones. A nivel de clase, indica que para acceder a los métodos de la clase por REST, la petición debe parecerse a http://.../**alumno**/… En el método indica que, para acceder al recurso, se debe dirigir la petición a http://.../alumnos/**listaOfertasDePracticas**

- **@POST:** Indica que el método anotado corresponde a una petición HTTP POST.

- **@Consumes:** Se utiliza para especificar qué tipo de representaciones puede aceptar o consumir del cliente. Si se aplica a nivel de clase, todos los métodos consumirán los tipos especificados de manera predeterminada. Si se aplica a nivel de método, anula cualquier anotación @Consumes aplicada a nivel de clase.

- **@Produces:** Se utiliza para especificar el tipo de respuesta que puede reproducir y enviar al cliente. Si se aplica a nivel de clase, todos los métodos producirán los tipos especificados de manera predeterminada. Si se aplica a nivel de método, anula cualquier anotación @Produces aplicada a nivel de clase.

Para finalizar, se debe configurar el archivo web.xml para indicar que, cuando la URL cumpla determinado patrón, la petición será redirigida al paquete que acabamos de crear.

```
<servlet>
    <servlet-name>Jersey REST Service</servlet-name>
    <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-class>
    <init-param>
        <param-name>jersey.config.server.provider.packages</param-name>
        <param-value>es.middleware.ws.rest.services,
        com.jersey.jaxb,
        com.fasterxml.jackson.jaxrs.json</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
</servlet>


<servlet-mapping>
    <servlet-name>Jersey REST Service</servlet-name>
    <url-pattern>/rest/*</url-pattern>
</servlet-mapping>
```

Ilustración 82. Configuración web.xml para exponer servicios REST

Para esta configuración, las etiquetas que se deben personalizar son <param-value> y <url-pattern>. En este caso, <param-value> recoge el nombre del paquete donde se encuentran todos los servicios y <url-pattern> es un filtro que asocia a los servicios REST, por lo que todos las URL deben tener "/rest/". Este es el ejemplo de la url necesaria para realizar la petición al servicio mencionado anteriormente:
`http://localhost:8080/TFGMiddleware/rest/alumno/listaOfertasDePracticas`

6.7. Problemas de CORS

El Intercambio de Recursos de Origen Cruzado (CORS) es un mecanismo que utiliza la cabecera HTTP para permitir que un user-agent (en este caso el navegador) obtenga permiso para acceder a recursos desde un servidor, en un origen distinto al que pertenece. Un agente crea una petición HTTP de origen cruzado cuando solicita un servicio desde un dominio, protocolo o puerto diferente al que lo generó.

En este proyecto, la aplicación está instalada en http://localhost:4200, pero el servidor está en http://localhost:8080, por lo que, cuando la aplicación realiza una petición al servidor estando este en un puerto diferente, se crea una petición HTTP de origen cruzado y no ejecuta la solicitud.

Para solucionar este inconveniente, se ha configurado un proxy para que haga de punto intermedio entre la aplicación y el servidor. Por lo que, cuando el navegador realice una petición, no estará accediendo directamente al servidor, sino que realizará una solicitud sobre el proxy y será este quien se conecte con el servidor y nos devuelva el resultado.

Para realizar la configuración, se crea un archivo *proxy.conf.json* en la carpeta raíz del proyecto (en la misma donde se encuentra *package.json*), con el código que se puede ver en la siguiente ilustración.

```
"/TFGMiddleware/rest/*": {
    "target": "http://localhost:8080",
    "secure": false,
    "changeOrigin": true
},
```

Ilustración 83. Configuración proxy

En esa estructura se detalla que todas las peticiones que se realicen a "/TFGMiddleware/res/" corresponden al punto de proxy "http://localhost:8080/".

Para terminar esta configuración se debe modificar el archivo package.json de esta manera:

```
"scripts": {
  "ng": "ng",
  "start": "ng serve --proxy-config proxy.conf.json",
  "build": "ng build",
  "test": "ng test",
  "lint": "ng lint",
  "e2e": "ng e2e"
},
```

Ilustración 84. Configuración package.json para el proxy

De esta forma, con el comando npm start, iniciaremos la aplicación utilizando la configuración del proxy realizada previamente.

6.8. Aplicación multi idioma

Esta aplicación puede ser utilizada en cualquier tipo de centro educativo que ofrezca la oportunidad de realizar prácticas en una empresa. Esto implica que los usuarios pueden tener diferente idioma materno o simplemente que quieran realizar la consulta en un idioma determinado. Por ello, es importante que la aplicación sea capaz de mostrar y gestionar datos en cualquier idioma. Asimismo, dependiendo de la localización en la que se esté usando la aplicación, el formato del contenido no siempre es el mismo. El formato de la fecha, por ejemplo, en Estados Unidos se utiliza MM/DD/AA mientras que en la mayor parte de los países europeos utiliza el formato DD/MM/AA.

Para configurar lo mencionado previamente, se emplea la librería **ngx-translate**, específicamente sus dependencias ngx-translate/core y ngx-translate/http-loader.

- **ngx-translate/core:** Dependencia con la funcionalidad principal para aplicar internacionalización en aplicaciones Angular.

- **ngx-translate/http-loader:** Dependencia encargada de cargar los archivos de traducción desde el directorio src/assets/i18n utilizando HttpClient.

Estas dependencias se descargan mediante el siguiente comando:

```
Npm install @ngx-translate/core @ngx-translate/http-loader
```

El segundo paso para llevar a cabo la configuración consiste en modificar el módulo principal app.module.ts.

```
import { NgModule } from '@angular/core';

import { TranslateLoader, TranslateModule } from '@ngx-translate/core';
import { TranslateHttpLoader } from '@ngx-translate/http-loader';
import { HttpClient, HttpClientModule } from '@angular/common/http';

import { registerLocaleData } from '@angular/common';
import localeES from '@angular/common/locales/es';
import { AppComponent } from '../app.component';


registerLocaleData(localeES);

@NgModule({
    declarations: [
        AppComponent,
    ],
    imports: [
        BrowserModule,
        HttpClientModule,
        TranslateModule.forRoot({
            loader: {
                provide: TranslateLoader,
                useFactory: (http: HttpClient) => {
                    return new TranslateHttpLoader(http, 'assets/i18n/', '.json');
                },
                deps: [HttpClient]
            }
        })
    ],
    bootstrap: [AppComponent]
})
export class AppModule { }
```

Ilustración 85. Configuración AppModule para el multi idioma

Las partes más destacables son las siguientes:

- **Imports de TranslateLoader, TranslateModule y TranslateHttpLoader:** Corresponden a los módulos y servicios necesarios para que ngx-translate funcione correctamente.
- **Configuración de TranslateModule:** Se debe utilizar un loader personalizado para este módulo, que carga los archivos de idioma utilizando HttpClient.

En el directorio src/assets/i18n/ se incluyen los archivos .json referentes al idioma, uno por cada idioma al que se quiera traducir. Todos los textos que se utilicen en la aplicación se recogen de estos ficheros, por lo que cada vez que la aplicación cambia de idioma, los textos se cogen del fichero correspondiente al idioma seleccionado. En este proyecto, la aplicación se ha traducido a dos idiomas, por lo que se han creado dos archivos, es.json con las traducciones en castellano, y eu.json con las de euskera.

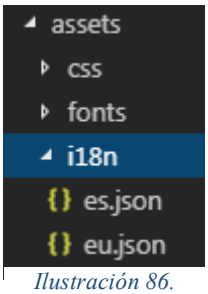

Ilustración 86. Archivos multi idioma

Existen dos formas de realizar la petición a esos ficheros desde el archivo HTML. A continuación se puede ver un ejemplo de cómo mostrar el mismo texto de las dos maneras distintas.

```
<label class="form-label">{{login.texto' | translate}}</label>
<label class="form-label" [translate]="login.texto'"></label>
```

Para finalizar con los ajustes de la traducción, hay que especificar el idioma que se quiere utilizar.
Por una parte, en el componente principal app.component.ts se determina el idioma por defecto (ver Ilustración 87. Inicialización idioma AppComponent).

```
import { Component } from '@angular/core';
import { TranslateService } from '@ngx-translate/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent{

  constructor(protected translate: TranslateService){
      this.translate.setDefaultLang('es');
  }
}
```

Ilustración 87. Inicialización idioma AppComponent

Y, por último, en el componente que dé la opción al usuario de cambiar el idioma, una función que cambie el lenguaje de la aplicación.

```
cambiarIdioma(idioma: string){
    this.translateService.setDefaultLang(idioma);
}
```

Ilustración 88. Cambio idioma Angular

6.9. Idazki Desktop

Para poder utilizar Idazki como método de firma, necesitamos tener previamente instalados los certificados de Izenpe, el middleware y la propia aplicación de escritorio.

Idazki Desktop facilita la firma del contrato usando la firma digital [4] mediante los certificados personales que tenemos en el ordenador. Desde nuestra aplicación se abre la aplicación de escritorio de Idazki, que procede a realizar la firma del documento adjuntado.

Estos son los pasos a seguir desde nuestro sistema, para realizar la llamada:

1. Inicializar el componente:

```
this.idazki = new Idazki('http://servicios.izenpe.com/idazkiweb');
```

2. Configurar las opciones generales y el método de carga de los certificados

```
this.idazki.setOption("dlgcertsel-enableocsp", "0");
this.idazki.signSetAdESLevel("bes");
this.idazki.setCryptoStoreAuto();
```

3. Configurar el comportamiento para el tratamiento de errores

```
this.idazki.setErrorCallback(function (errorCode) {
       this.error = errorCode;
      this.idazki.endTransaction();
    });
```

4. Configurar la entrada pasando el texto a firmar

```
this.idazki.addInput("inline-binary", document, "inline", null);
```

5. Iniciar el proceso de firma indicando el tipo

```
this.idazki.sign("pades", function () {
```

6. Obtener el resultado de la firma

```
       this.token = this.idazki.getToken();
       this.idazki.getOutputContent(0, true, function (result) {
       this.idazki.endTransaction();
       });
});
```

Esta es la secuencia de pasos a dar con Idazki Desktop:

- Seleccionar el certificado (En este caso se ha probado la firma con el certificado de la fabrica de moneda y timbre):

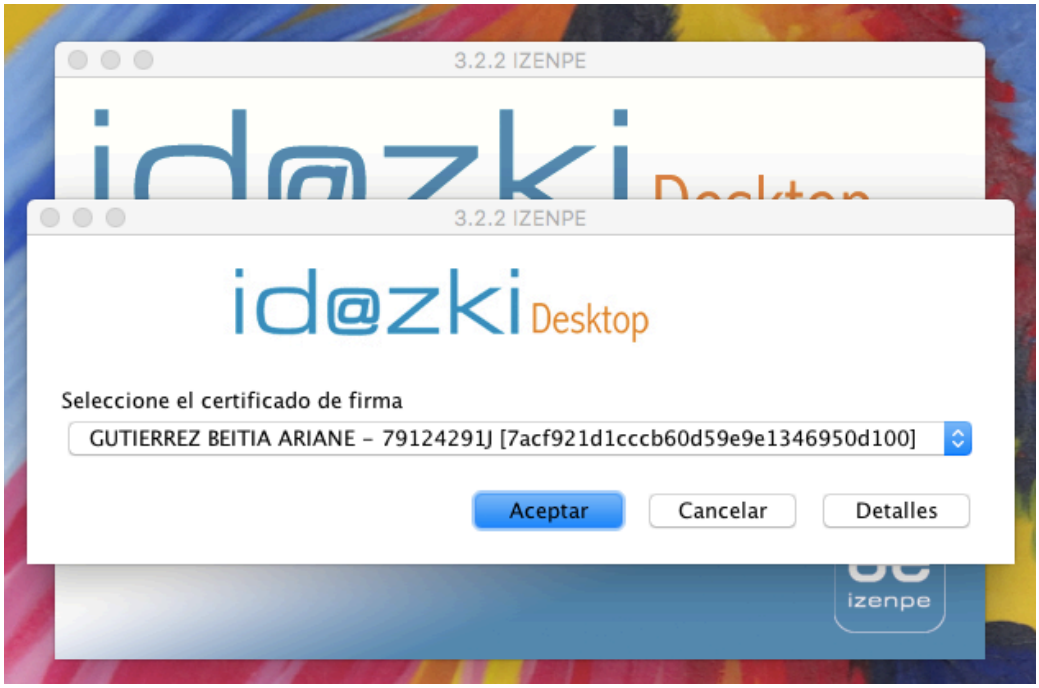

Ilustración 89. Selección certificado en Izenpe Desktop

7. Verificación y evaluación

En este apartado se detallan las pruebas realizadas para la comprobación del correcto desarrollo de la interfaz y de las funcionalidades. Las pruebas se clasifican en base a la pantalla en la que se han efectuado y se calificarán en verde, en el caso de se hayan completado satisfactoriamente, o en rojo, en el caso de que no se hayan obtenido los resultados esperados.

La mayoría de tiempo invertido en realizar las pruebas durante el desarrollo del proyecto fue a nivel de funcionalidad de la aplicación, las cuales no están reflejadas en este documento. Puesto que la tecnología usada para la implementación de la parte front-end era totalmente nueva, todas esas pruebas son respectivas al proceso de aprendizaje.

7.1. Pruebas interfaz

En esta parte se realizarán pruebas en el Front-end que es la parte visual del sistema, recolectando los datos de entrada del usuario.

A continuación se muestran las pruebas realizadas en la página de login.

Acción	Resultado esperado	Resultado obtenido
Rellenar erróneamente el usuario o contraseña	Mensaje de usuario o contraseña incorrectos	Muestra un mensaje de error de usuario o contraseña incorrectos.
Dejar en blanco el usuario o contraseña	El botón de “entrar” no ejecuta nada.	El botón de “entrar” no se realiza ninguna acción.
Cambiar de tipo de usuario	Cambio de la pantalla de login dependiendo del usuario activado	Cambia la pantalla de login
Iniciar sesión pulsando el botón de “entrar”	Mostrar loader mientras se carga la página principal	Se muestra el loader y después accede a la página principal
Iniciar sesión pulsando la tecla “enter”	Iniciar sesión	Inicia sesión

Tabla 10. Pruebas interfaz pantalla login

Al probar la pantalla de Login, se puede apreciar que todo ha salido como se esperaba.

En el caso de la cabecera de la aplicación, estas son las pruebas realizadas:

Acción	Resultado esperado	Resultado obtenido
Cambiar de pantalla usando las opciones de la cabecera	Navegación a la pantalla correspondiente	Navega a la pantalla esperada
Cambiar de pantalla usando las opciones de la cabecera	Cambia de color para mostrar la pantalla actual	No cambia de color

Usar la info del usuario para usar sus funciones	La info del usuario despliega las opciones y realiza la navegación	Acciona el desplegable y la navegación correctamente
Cambiar el idioma	Cambiar el idioma de los textos y mostrar el idioma seleccionado	Se traducen los textos y aparece el idioma elegido.

Tabla 11. Pruebas interfaz cabecera

En las pruebas se puede observar que no se ha logrado implementar el marcado de la pantalla actual.

Estas son las pruebas realizadas en el popup de subir el currículum:

Acción	Resultado esperado	Resultado obtenido
Subir un documento PDF	Mostrar carga del documento y después la subida correcta.	Se muestra el estado de carga durante 5 segundos y después la carga completada
Subir un documento que no sea PDF	Mostrar error de documento	Se muestra de color rojo error de documento e icono de volver a cargar.
Pulsar el botón guardar sin haber seleccionado ningún documento	El botón no se acciona	El botón no se habilita

Tabla 12. Pruebas interfaz modal subir curriculum

Por las pruebas se puede observar el correcto funcionamiento de la ventana emergente de subir el currículum.

Esas son las pruebas realizadas en el popup de elegir el profesor:

Acción	Resultado esperado	Resultado obtenido
Hacer click fuera de la ventana	No se cierra el modal	No se cierra el modal
Seleccionar un profesor	Habilitar el botón de “guardar”	Se habilita el botón “guardar”
Click en el botón “guardar”	Se cierra el popup	No se cierra el popup

Tabla 13. Pruebas interfaz modal elegir profesor

El modal de elegir el profesor no funciona como se esperaba a la hora de cerrar la ventana.

Esas son las pruebas realizadas en el popup de añadir un instructor:

Acción	Resultado esperado	Resultado obtenido
Introducir letras o símbolos en el campo del teléfono	No se rellena el input	No se escribe nada en el campo
Pulsar el botón sin haber rellenado todos los campos	No ocurre nada	El modal no hace nada hasta que no se rellenen todos los campos
Escribir el email	Muestra un error por el formato del email	Muestra un mensaje de formato incorrecto
Escribir el teléfono	Muestra un error por el formato del teléfono	Muestra un mensaje de error de formato incorrecto
Click en el botón “gurdar”	No se activa si el email y el teléfono están mal	El botón no se habilita hasta que el teléfono y el email no tienen el formato correcto

Tabla 14. Pruebas modal añadir instructor

Según las pruebas, se puede comprobar que la función de añadir un nuevo instructor no se ejecuta si los datos introducidos no tienen el formato correcto.

Esas son las pruebas realizadas en el popup de actualizar oferta:

Acción	Resultado esperado	Resultado obtenido
Pulsar el botón guardar sin haber rellenado todos los campos	No ocurre nada	El botón no acciona ninguna función hasta que no se rellenan todos los campos
Escribir los días de vacaciones	No permite valor no numérico	El campo no recoge valores negativos, caracteres o símbolos
Introducir las fechas	Mostrar mensaje de error en el caso de que las fechas sean inferiores al día actual y la fecha de inicio sea superior a la final	Se muestra mensaje de error

Tabla 15. Pruebas interfaz modal actualizar oferta

Por lo que se puede apreciar en las pruebas, el modal cumple con los resultados esperados.

En la pantalla de preselecciones para el alumno se han realizado las siguientes pruebas:

Acción	Resultado esperado	Resultado obtenido
Selección de una oferta	Se habilita el botón de "guardar"	Se habilita el botón de "guardar"
Carga de los estados de las ofertas	Los estados se cargan del color correspondiente	Los estados se visualizan del color correcto

Tabla 16. Pruebas interfaz pantalla preselecciones

La parte front-end de la pantalla preselecciones cumple lo esperado.

En la página del perfil del alumno se han realizado la siguiente prueba:

Acción	Resultado esperado	Resultado obtenido
Carga del currículum	Visualizar todas las páginas del currículum en un tamaño correcto y con buena visibilidad.	El currículum se visualiza de forma correcta y con un buen tamaño.

Tabla 17. Pruebas pantalla perfil alumno

Estas son la pruebas que corresponden a la pantalla de alumnos en selección de la parte de la empresa:

Acción	Resultado esperado	Resultado obtenido
Carga de los estados	Visualizar los estados del color correspondiente	Los estados se visualizan del color que les corresponde
Click en el DNI	Se abre una nueva ventana del navegador con el documento	Se abre una nueva ventana del navegador con el currículum adecuado

Tabla 18. Pruebas interfaz pantalla alumnos en selección

A continuación se presentan las pruebas de front-end realizadas en la página de añadir nueva oferta:

Acción	Resultado esperado	Resultado obtenido
Click en los switch	Los campos cambian de color según la selección	Los switch se cambian a color azul en el caso de estar accionado.
Click en botón "guardar" sin rellenar todos los campos requeridos	No se acciona si los campos obligatorios no están rellenados	No se ejecuta ninguna función

Rellenar los campos de las fechas de forma incorrecta	Visualizar error en el caso de seleccionar una fecha anterior al día actual o de meter la fecha de inicio superior a la final	Se visualiza un mensaje de error
Rellenar campos numéricos	No permitir números negativos	No se permiten los números negativos

Tabla 19. Pruebas interfaz añadir oferta

7.2. Pruebas funcionales

En este apartado se analizarán las pruebas relacionadas con el back-end [2] de la aplicación, que es el área que gestiona la parte lógica, es decir, la parte del sistema que no es visible para el usuario, donde están programadas todas las funcionalidades que tendrá el proyecto.

Esas pruebas se pueden observar en la siguiente tabla:

Acción	Resultado esperado	Resultado obtenido
Iniciar sesión con el servidor inactivo	Visualización de la pantalla de error 500	Se visualiza la página de error
Hacer referencia a usuarios que no existen al insertar en la base de datos	La sentencia no se ejecuta y envía un mensaje de error	La base de datos no realiza el cambio y el servidor devuelve un mensaje de error
Hacer referencia a usuarios que no existen al insertar en la base de datos	La aplicación visualiza la pantalla de error	La aplicación redirecciona a la página de error 500
Conexión aplicación - servidor	Cargar los datos de manera correcta	La conexión de la aplicación con el servidor se realiza de forma exitosa
Conexión servidor - base de datos	Recoger y modificar los datos de forma correcta	La conexión del servidor a la base de datos se realiza de forma exitosa
Iniciar sesión con diferentes usuarios	Identificar correctamente al tipo de usuario	El tipo de usuario que inicia sesión se recoge correctamente
Enviar y recibir los parámetros de la aplicación al servidor	No se crea ningún problema con la conversión de los datos	Cuando se transforma de un string a JSON no ocurre ningún problema y los parámetros convergen perfectamente

Equivocarse en el nombre del servicio REST	Redirección a la página de error	Se visualiza la página de error 500
Mandar de la aplicación al servidor toda la información del alumno	Convertir a JSON sin ningún problema	La conversión a JSON falla
Ejecución de los observables	Los datos se consiguen antes de cargar el HTML	El HTML se carga un poco antes que los datos. La consola muestra error por unos milisegundos.
Cambiar idioma	Todos los textos de la aplicación se traducen	La aplicación traduce todos los textos correctamente

Tabla 20. Pruebas funcionales

8. Conclusiones

En este apartado se evaluarán las conclusiones finales tras finalizar el proyecto. Para ello, se realizará una comparación de lo planeado inicialmente con los cambios que han ido surgiendo a lo largo del proyecto. Además, es la hora de hacer una reflexión y valorar que cosas han podido salido mal a lo largo de estos 4 meses, pero también de destacar todo aquello que ha salido como se esperaba o incluso mejor.

Para finalizar, se realizará una reflexión personal del proyecto de una forma subjetiva. Puesto que es la primera vez que he llevado a cabo un trabajo tan grande por mi cuenta, ha supuesto un gran reto y he descubierto cosas tanto positivas como negativas qu han afectado al proyecto.

8.1. Cambios

En primer lugar, se van a analizar los cambios principales que han sucedido a lo largo del proyecto. Algunos de esos cambios han ido surgiendo a medida que avanzada el proyecto y otros, en cambio, por una mala planificación.

8.1.1. De MongoDB a MySQL

Antes de empezar las prácticas de empresa, Angular era una tecnología completamente desconocida para mi, por lo que cuando me decidí a utilizar esta herramienta para realizar el trabajo de fin de grado, una de las cosas sobre la que tuve que aprender fue la manera de implementar la base de datos y sobre cómo recoger y modificar su información. Todo lo que encontraba sobre Angular y base de datos mencionaba una herramienta llamada MongoDB.

MongoDB es un sistema de base de datos orientada a documentos, lo que significa que cada entrada o registro puede tener un esquema de datos diferente, con atributos y "columnas" que no tienen porque repetirse de un registro a otro. Los datos se guardan en BSON, que es una versión modificada de JSON. Además, es una base de datos NoSQL, lo que significa que no usa SQL como lenguaje principal de consultas y, por consecuencia, no soporta operaciones JOIN.

A pesar de no haber trabajado nunca con este tipo de estructuras, me pareció interesante y decidí investigar a cerca del tema y de cómo poder implementarlo en mi aplicación. Descubrí la librería de npm *mongoose*, que permite realizar la conexión con la base de datos mongoDB de tu ordenador. Ahí encontré el primer obstáculo. No conseguía de ninguna manera conectar la aplicación con la base de datos y, según pasaba el tiempo y no veía resultados decidí cambiar de estrategia.

Además, el no poder usar sentencias SQL con JOIN para recoger y modificar los datos, rompía los esquemas que tenía en la cabeza con respecto al tratamiento de las bases de datos, puesto que todo lo aprendido en la universidad sobre bases de datos ha sido con SQL.

Por todo esto y, teniendo en cuenta que utilizar Angular también iba a suponer un reto, pensé que no podía dedicarle tanto tiempo al aprendizaje de dos tecnologías nuevas y decidí implementar un middleware usando MySQL y Eclipse, ya que conocía bien como realizar la conexión entre ambas, debido a que ya había trabajado antes con ello. Por otra parte, me gustaba la idea de mantener separados la lógica de la parte visible de la aplicación.

8.1.2. Cambios en la planificación y costes

Todas las veces que se empieza a trabajar en un nuevo proyecto, se es consciente de la importancia que tiene realizar una buena planificación e intentar cumplirla en la medida de lo posible. Pero, también es verdad que, a medida que avanza, se nos puede ir olvidando seguir esa planificación, tanto porque no conseguimos llevar el ritmo que habíamos previsto al principio o por consecuencias que se van dando durante el desarrollo.

Al inicio de proyecto, como se ha mencionado en el apartado anterior, invertí mucho tiempo en investigar como poder realizar la conexión con la base de datos de la forma más eficiente posible. Esto lo hacia a la vez que escribía la primera parte de la documentación, el DOP, por lo que no le dediqué tanto tiempo como esperaba a la definición de objetivos del proyecto. Como consecuencia, tuve que retrasar la implementación de la aplicación. Ese retraso unido a mi falta de experiencia con Angular y servicios REST, que fue la alternativa que encontré a la recogida y modificación de los datos, han hecho que haya tenido que dedicarle más horas de las calculadas a la implementación en el último mes y medio, teniendo en cuenta además, que compartía las horas del día para escribir la memoria y el código.

Esas horas extra provocan un cambio en la evaluación económica inicial. En el apartado 2.7, inicialmente se calcularon 30 horas semanales dedicadas al trabajo. En el último mes y medio, ese número ha ascendido a 48 horas, por lo que, si se vuelve a calcular, el total de mano de obra subiría a 14.700€, 2.700€ más de lo estimado.

8.1.3. Cambio de herramientas

Cuando surgió la idea de incorporar la firma digital para la validación del contrato en la selección de las prácticas de empresa, todavía no estaba segura de cómo hacerlo. Investigando un poco las librerías que ofrece Angular, ninguna me convenció, porque algunas de ellas no estaban muy experimentadas o no eran lo suficientemente acreditativas.

Entonces, mi tutora me enseñó Idazki. Esta aplicación reconoce los certificados digitales, así como las credenciales de tarjetas como la que tienen los profesores en la universidad. Además, la empresa ofrece una librería JavaScript libre que facilita la incorporación de la herramienta a la aplicación creada en este proyecto.

8.1.4. Cambio de planes

Parece que cuando se acaba la grado, el próximo paso es hacer un master. En mi caso, ese no era un plan decisivo, porque no quería meterme a un master sin conocer bien el alcance de mi carrera y las posibles salidas que tiene. Durante los cuatro años de estudio, se dan muchas asignaturas sobre diferentes aspectos de la informática, pero no se profundiza demasiado y, en mi caso, me interesaban varios de ellos. Por todo esto, no estaba segura de cual podría ser una buena opción para hacer un master, por lo que decidí no realizar ninguno el próximo año.

Cuando se acabó el primer cuatrimestre, tomé la iniciativa de apuntarme a ofertas de prácticas en empresa para introducirme en un entorno laboral y conocer lo que podía ser mi futuro en un trabajo. Así, podría conocer otros aspectos de la informática, formarme más y ver en que podría enfocar mis futuros estudios.

Por todo esto, no tenía prisa en entregar el trabajo de fin de grado este curso académico. Incluso tenía la idea de entregarlo en septiembre/octubre, por lo que no tenía mucha urgencia en elegir el tema y empezar a trabajar en el.

Al de poco de elegir el tema del proyecto, en la empresa me ofrecieron la oportunidad de seguir trabajando con ellos una vez tuviera el título académico, lo que supuso grandes cambios en mi planificación personal, ya que entregarlo en las fechas previstas suponía empezar muy tarde en el trabajo, por lo que me propuse entregarlo en julio, para volver a la empresa lo antes posible. Esto me creo gran estrés, ya que quería dedicarle más tiempo al estudio de Angular, y suponía tener menos tiempo para realizar el proyecto.

8.2. Trabajo futuro

Todos los requisitos planteados inicialmente se han llevado a cabo de manera exitosa. Aun así, al igual que en muchos sistemas, se podrían realizar mejoras que ampliaran la calidad del trabajo. En este apartado veremos algunas de esas mejoras.

8.2.1. Añadir funcionalidades

Todas las pantallas de la aplicación cumplen con su funcionalidad principal, pero también es cierto, que se pueden añadir nueva funciones que faciliten al usuario su utilización. Por ejemplo, en las pantallas de historiales, se podría añadir un filtro de búsqueda por fecha o alumno, de está manera, si el usuario está buscando una entrada concreta de la tabla, lo podría hacer de un modo más sencillo y no tener que registrarla entera.

Por otro lado, se podrían incluir más mensajes o textos informativos, para que el usuario esté al tanto en todo momento de lo que está ocurriendo dentro de la aplicación o del estado en el que se encuentra cada asignación.

8.2.2. Contenido de las pantallas

En el caso de las pantallas que contienen tablas, se ofrece poca información visible, solo la necesaria para identificar correctamente cada entrada. Por ello, convendría añadir toda la información relevante para cada oferta de prácticas, y que se mostrara cuando el usuario clicara encima de alguna de ellas.

8.2.3. Diseño de la página web

Respecto a la estructura de la página web, creo que ha quedado bastante intuitiva y sencilla, pero se podrían mejorar las formas y secciones de algunas pantallas para que tengan un aspecto más profesional y serio.

8.2.4. Añadir nuevos idiomas

Los idiomas disponibles para esta aplicación son el euskera y el castellano, lo cual serviría para centros donde esos fueran los lenguajes más importantes. Pero la idea es que está aplicación sea útil para diferentes centros educativos donde podría haber más diversidad de idiomas, por ello, convendría contratar a un traductor experto que realizará las traducciones.

8.2.5. Dispositivos móviles

Adaptar todas las funcionalidades a los dispositivos móviles, así como optimizar la visualización de todas las pantallas en dichos dispositivos.

8.3. Reflexión personal

Para empezar he de decir que ha sido la primera vez que he hecho un trabajo tan grande yo sola. Durante los años en la universidad hemos realizado varios trabajos bastante grandes, que requerían de muchas horas de trabajo para sacarlos adelante. La diferencia es que siempre han sido en grupo y nunca tan largos, o implicaban solo la parte de la aplicación o la de documentación, no las dos juntas.

Para cuando empecé con este proyecto, ya llevaba unas semanas aprendiendo y practicando Angular con el grupo de trabajo al que me asignaron en las prácticas de empresa, y no se me hizo muy difícil acostumbrarme a ello, aunque ayudar en un proyecto y crear uno propio nada tienen que ver. Asique, cuando lo elegí como herramienta para el desarrollo de mi aplicación, realmente no sabía todo lo que eso conllevaría, ya que ninguno de los lenguajes que usa (TypeScript, HTML y CSS) los habíamos dado durante la carrera. También, una de las cosas que me hizo lanzarme a utilizar Angular fue la iniciativa que tuvieron dos compañeros de la empresa en ayudarme en todo lo que podrían.

Por esa parte, debo decir que creo que he crecido mucho como programadora, ya que, a pesar de que me ofrecieron su ayuda, cuando llegaba a casa y me disponía a implementar, no estaban allí para ayudarme y he tenido que valerme por mi misma para encontrar soluciones a problemas que iban surgiendo. Una ventaja o desventaja del TFG, depende de cómo se mire, es que nadie te aplica fechas de entrega ni te vigila si estás trabajando o no, pero esa es una de las partes más importantes de este trabajo a mi parecer, ser capaz de sacar adelante una idea desarrollada por ti, imponerte tu propio horario y, con la base obtenida en la universidad y el auto-aprendizaje, hacer un proyecto del que sentirte orgulloso.

Debo decir que este trabajo a tenido sus más y sus menos. Un día estás muy contenta porque consigues implementar el observable para visualizar los datos, luego te pasas tres días implementando el diseño de una sola pantalla y ves que no avanzas, al día siguiente hablas con la tutora y te das cuenta de que la documentación no va mal, pero pasa una semana, solo te queda un mes para acabar y te agobias de todo lo que te queda por hacer,… Ese ha sido mi estado emocional durante los cuatro meses del proyecto.

Quitando el primer mes de trabajo, donde quizá no le dediqué tanto tiempo, he sido muy constante durante todo el proceso, incluso le he destinado más horas que al estudio de las asignaturas durante toda la carrera. Por lo que, a pesar de todas las dificultades, creo que al final ha tenido buen resultado y me siento satisfecha del trabajo logrado. Echando la vista atrás y viendo todo lo que he conseguido en estos meses, crea en mi más ganas de aprender y seguir formándome en esta área.

Glosario

[1] **Front-end:** El front-end es la parte del software que se encarga de la interacción son el usuario.

[2] **Back-end:** El back-end es la parte del software que se encarga de todas las funcionalidades de la aplicación, así como las consultas a la base de datos, la conexión con el servidor, etc.

[3] **Bootstrap:** Es un framework basado el HTML, CSS y JavaScript que facilita el diseño web.

[4] **Firma digital:** Es un mecanismo criptográfico que permite dar validez a un documento a través de certificados oficiales que valida la firma y la identidad de la persona que la realiza.

[5] **Servidor:** Es una aplicación en ejecución capaz de atender las peticiones de un cliente y devolverle una respuesta en concordancia.

[6] **JSON:** Es un formato de intercambio de datos.

Bibliografía

[7] Documentation for Visual Studio Code. Recuperado de http://code.visualstudio.com/docs

[8] Bootstrap. Recuperado de https://getbootstrap.com

[9] npm | build amazing things. Recuperado de https://www.npmjs.com

[10] HTML Tutorial. Recuperado de https://www.w3schools.com/default.asp

[11] Idazki Desktop, firma por protocolo. Recuperado de http://www.izenpe.eus/contenidos/informacion/idazki_izenpe/es_def/adjuntos/idazki-desktop-protocolo_v3.2.pdf

[12] ¿Cuánto vale el kilovatio hora de luz en España? Recuperado de https://tarifasgasluz.com/faq/precio-kwh

[13] Angular. Recuperado de https://angular.io