UNIVERSIDAD SAN PABLO - CEU

ESCUELA POLITÉCNICA SUPERIOR

GRADO EN INGENIERÍA DE SISTEMAS DE INFORMACIÓN



TRABAJO FIN DE GRADO

**JOBS : diseño e implementación de una aplicación RESTful para búsqueda de empleo para estudiantes/graduados**

Autor: Carlos Soler Martín

Tutores: Sergio Saugar García (U. San Pablo)

Alex Rayón Jerez (U. Deusto)

Junio 2023

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Datos del alumno   |  | | --- | | Nombre: Carlos Soler Martín |   Datos del Trabajo   |  | | --- | | TÍTULO DEL PROYECTO: JOBS - diseño e implementación de una aplicación RESTful para búsqueda de empleo para estudiantes/graduados |   Tribunal calificador   |  |  | | --- | --- | | Presidente: | Fdo.: |  |  |  | | --- | --- | | Secretario: | Fdo.: |  |  |  | | --- | --- | | Vocal: | Fdo.: |  |  | | --- | | Reunido este tribunal el \_\_\_ /\_\_\_ /\_\_\_\_\_\_\_, acuerda otorgar al Trabajo Fin de Grado presentado por D./Dña. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ la calificación de \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | |

Resumen

El propósito de este Trabajo de Fin de Grado consiste en el diseño e implementación de una aplicación que aborde los problemas identificados en relación con el proceso de asignación de prácticas y primeros empleos a los alumnos universitarios y recién graduados en el CEU. La solución planteada se centra en unificar las aplicaciones existentes, digitalizar y homogeneizar los CVs de los alumnos y las ofertas de prácticas/empleo proporcionadas por las empresas, y automatizar de manera integral todo el proceso, incluyendo la recomendación de ofertas a los alumnos. Se parte del análisis del estado del arte de los servicios Web y algoritmos de recomendación. Se expone la metodología de gestión del proyecto. Se realiza el análisis de dominio, especificando requisitos y definiendo la arquitectura del sistema. Posteriormente se diseña e implementa la arquitectura del sistema. Por último, se expone el despliegue de la aplicación junto con las pruebas realizadas y las conclusiones del proyecto.

Palabras Clave

Servicios Web RESTful, Aplicaciones Web, Flask, Postman, HTTP, algoritmo de recomendación, Python, despliegue AWS.

Abstract

The purpose of this TFG is to design and implement an application that addresses the identified problems regarding the process of assigning internships and first jobs to university students and recent graduates at CEU. The proposed solution focuses on unifying the existing applications, digitize, and standardize the students' CVs and the internship/job offers provided by companies, and fully automating the entire process, including offering recommendations to the students. The project begins with an analysis of the state of the art in Web services and recommendation algorithms. The project management methodology is presented, followed by the domain analysis, specifying requirements, and defining the system architecture. Subsequently, the system architecture is designed and implemented. Finally, the deployment of the application, along with the conducted tests and project conclusions, are presented.

Keywords

RESTful Web Services, Web Applications, Flask, Postman, HTTP, recommendation algorithm, Python, AWS deployment.

Índice de contenidos

[Capítulo 1 Introducción 1](#_Toc136020065)

[Capítulo 2 Estado del arte 5](#_Toc136020066)

[2.1 Servicios Web 5](#_Toc136020067)

[2.2 Algoritmo de recomendación 10](#_Toc136020068)

[2.2.1 Introducción a la ciencia de los datos 11](#_Toc136020070)

[2.2.2 *Machine Learning*: métodos de aprendizaje 11](#_Toc136020073)

[2.2.3 Técnicas de análisis de datos prescriptivas: sistemas de recomendación 12](#_Toc136020092)

[2.2.4 Medidas de similitud o distancia 13](#_Toc136020095)

[Capítulo 3 Gestión del Proyecto 15](#_Toc136020096)

[3.1 Modelo de ciclo de vida 15](#_Toc136020097)

[3.2 Papeles desempeñados en el proyecto 16](#_Toc136020098)

[3.3 Planificación y ejecución 16](#_Toc136020099)

[Capítulo 4 Análisis y arquitectura 19](#_Toc136020100)

[4.1 Análisis de dominio 19](#_Toc136020101)

[4.2 Especificación de requisitos 21](#_Toc136020108)

[4.3 Análisis de los casos de uso 22](#_Toc136020111)

[4.4 Arquitectura del sistema 24](#_Toc136020120)

[Capítulo 5 Diseño 26](#_Toc136020121)

[5.1 Diseño del *backend* 26](#_Toc136020122)

[5.1.1 Diseño del Servicio Web RESTful 26](#_Toc136020124)

[5.1.2 Diseño del algoritmo de recomendación 41](#_Toc136020125)

[5.1.3 Diseño de la base de datos 42](#_Toc136020126)

[5.2 Diseño del *frontend* 45](#_Toc136020136)

[Capítulo 6 Implementación 47](#_Toc136020137)

[6.1 Implementación *backend* 47](#_Toc136020138)

[6.1.1 Implementación Servicio REST 47](#_Toc136020139)

[6.1.2 Implementación Algoritmo recomendación 50](#_Toc136020140)

[6.1.3 Implementación Base de datos 54](#_Toc136020141)

[6.2 Implementación Frontend 55](#_Toc136020142)

[Capítulo 7 Despliegue y pruebas 56](#_Toc136020155)

[7.1 Despliegue 56](#_Toc136020156)

[7.2 Pruebas 57](#_Toc136020157)

[Capítulo 8 Conclusiones y líneas futuras 61](#_Toc136020158)

[Bibliografía 63](#_Toc136020159)

[Anexo A 66](#_Toc136020160)

[Diseño de las pantallas del *frontend* 66](#_Toc136020161)

[Anexo B 74](#_Toc136020162)

[JSON del banco de pruebas 74](#_Toc136020163)

[Colección Alumnos: 74](#_Toc136020164)

[Colección Empresas 76](#_Toc136020165)

[Colección CV 79](#_Toc136020166)

Índice de ilustraciones

[Ilustración 1: Nivel 0 del Modelo Madurez Richardson 7](#_Toc136012045)

[Ilustración 2: Nivel 1 del Modelo Madurez Richardson 7](#_Toc136012046)

[Ilustración 3: Nivel 2 del Modelo Madurez Richardson 8](#_Toc136012047)

[Ilustración 4: Nivel 3 del Modelo Madurez Richardson 8](#_Toc136012048)

[Ilustración 5: Diagrama Gantt de la planificación del proyecto 16](file:////Users/carlosoler/Documents/GitHub/TFGWord/TFGCarlosSoler/docs/gisi-tfg-csoler-memoria-curso-2022%20(vLimpio).docx#_Toc136012049)

[Ilustración 6: Diagrama Gantt de la ejecución del proyecto 16](file:////Users/carlosoler/Documents/GitHub/TFGWord/TFGCarlosSoler/docs/gisi-tfg-csoler-memoria-curso-2022%20(vLimpio).docx#_Toc136012050)

[Ilustración 7: Diagrama de casos de uso 21](#_Toc136012051)

[Ilustración 8: Arquitectura del sistema 23](#_Toc136012052)

[Ilustración 9: Diagrama E/R de la base de datos 41](#_Toc136012053)

[Ilustración 10: Ventana Recomendador de ofertas 44](#_Toc136012054)

[Ilustración 11: Parte del código para ver ofertas disponibles 46](#_Toc136012055)

[Ilustración 12: Parte del código REST del método GET de las ofertas 47](#_Toc136012056)

[Ilustración 13: Clase Alumno de la base de datos en python 53](#_Toc136012057)

[Ilustración 14: Código del archivo R.py 54](#_Toc136012058)

[Ilustración 15: Código del archivo ExecuteTFG.sh 55](#_Toc136012059)

[Ilustración 16: Petición POST en Postman 56](#_Toc136012060)

[Ilustración 17: Respuesta de petición POST en Postman 56](#_Toc136012061)

Índice de tablas

[Tabla 1: Recurso /alumnos 26](#_Toc135561725)

[Tabla 2: Recurso /alumnos/id 27](#_Toc135561726)

[Tabla 3: Recurso /alumnos/id/CV 29](#_Toc135561727)

[Tabla 4: Recurso /empresas 30](#_Toc135561728)

[Tabla 5: /empresas/id 31](#_Toc135561729)

[Tabla 6: Recurso /empresas/id/ofertas 32](#_Toc135561730)

[Tabla 7: Recurso /empresas/id/ofertas/id 33](#_Toc135561731)

[Tabla 8: Recurso /ofertas 35](#_Toc135561732)

[Tabla 9: Recurso /ofertas/id 35](#_Toc135561733)

[Tabla 10: Recurso /ofertas/cvs 36](#_Toc135561734)

# Introducción

Se ha llevado a cabo un análisis del proceso de asignación de prácticas y primeros empleos a los alumnos universitarios y recién graduados en la Universidad CEU San Pablo (CEU), con el propósito de identificar los problemas existentes y encontrar soluciones adecuadas. Se ha observado que los procesos y aplicaciones utilizados para la búsqueda y asignación de prácticas y ofertas de empleo presentan una serie de problemas, entre ellos la existencia de funcionalidades duplicadas, disparidad en las formas de operar, procesos completamente manuales, falta de digitalización en los CVs y las ofertas de prácticas/empleo, contenido heterogéneo y ausencia de un proceso automatizado de recomendación de oferta a los alumnos.

El objetivo de este Trabajo de Fin de Grado (TFG) es diseñar e implementar una aplicación que resuelva los problemas mencionados anteriormente. Se busca unificar las aplicaciones existentes, digitalizar y homogeneizar los CVs de los alumnos y las ofertas de prácticas/empleo proporcionadas por las empresas, y automatizar por completo todo el proceso, incluyendo la recomendación de ofertas a los alumnos. A través de este enfoque, se espera mejorar significativamente la eficiencia y la calidad del proceso de asignación, beneficiando a los alumnos, al CEU y a las empresas.

La memoria de este TFG está estructurada en los siguientes capítulos:

* Capítulo 2: se analiza el estado del arte de dos tecnologías fundamentales en el ámbito digital: los Servicios Web y los algoritmos de recomendación. Se examina la evolución de los Servicios Web, diferenciando entre los tradicionales y los basados en REST. También se estudia el funcionamiento y la tipología de los algoritmos de recomendación.
* Capítulo 3: se explica la metodología de desarrollo de software utilizada en el proyecto. Se detalla la planificación inicial del proyecto, incluyendo estimaciones de tiempo, y se muestra la ejecución real del proyecto con sus tiempos correspondientes.
* Capítulo 4: se abordan aspectos cruciales del desarrollo de software, como el análisis de dominio, la especificación de requisitos, el análisis de casos de uso y la arquitectura del sistema. Se exploran mejores prácticas, herramientas y metodologías relacionadas con cada uno de estos temas, con el objetivo de comprender a fondo el entorno del problema, capturar las necesidades de los usuarios, identificar casos de uso clave y establecer una arquitectura sólida para el sistema.
* Capítulo 5: se detalla el diseño de la plataforma basado en la arquitectura REST. La plataforma se divide en dos componentes principales: el *backend* y el *frontend*. Se describe el diseño de cada uno de ellos, teniendo en cuenta sus funcionalidades y requisitos específicos.
* Capítulo 6: se describe la implementación de la plataforma, basándose en el diseño presentado en el capítulo anterior. Se aborda la implementación de los dos componentes principales de la arquitectura del sistema: el *backend* y el *frontend*.
* Capítulo 7: se presenta una descripción detallada del despliegue de la aplicación en la plataforma de servicios en la nube de *Amazon Web Services* (AWS), junto con las pruebas realizadas durante su implementación y mantenimiento.
* Capítulo 8: se exponen las conclusiones obtenidas a lo largo de todo el TFG, así como posibles líneas futuras para el desarrollo y mejora de la aplicación.

# Estado del arte

En este capítulo se van a identificar el estado del arte sobre dos tecnologías muy importantes en el mundo digital: los Servicios Web y los algoritmos de recomendación. En particular, se examinará la evolución de los Servicios Web, diferenciando los tradicionales de los basados en REST. Además, se analizará el funcionamiento y la tipología de los algoritmos de recomendación.

## Servicios Web

W3C define un Servicio Web como: “Un sistema software diseñado para admitir la interacción interoperable de máquina a máquina a través de una red “ (Web Services Architecture Working Group, 2004). Hacen referencia al conjunto de protocolos, procesos, estándares entre dos máquinas o aplicaciones para comunicarse entre sí. Los Servicios Web cuentan con dos características fundamentales:

* Son multiplataforma: el cliente y servidor no tienen por qué contar con la misma configuración del sistema, lenguaje soportado o plataforma.
* Son sistemas distribuidos: un servicio web no está dirigido para un único cliente, sino que son varios los que pueden acceder al mismo servicio.

Los Servicios Web presentan una arquitectura cliente/servidor. El servidor es el sistema encargado del almacenamiento de datos y recursos, mientras que los clientes solicitan los servicios y datos a los servidores. (IBM, 2023)

Una Aplicación Web se define como un servicio en línea que proporciona una aplicación interactiva accesible a través de un navegador web. En otras palabras, es una herramienta que permite a los usuarios utilizar una variedad de servicios y funciones al conectarse a un servidor web a través de Internet (ictea, s.f.). Las Aplicaciones Web se estructuran de la siguiente manera: (Coppola, 2023)

* *Frontend*: es la parte visual y funcional con la que interactúa el cliente. Sus componentes principales son las imágenes, colores, logotipos, botones e información.
* *Backend*: es la parte encargada de procesar y administrar la información que alimenta al *frontend*. A diferencia del *frontend*, el *backend* no es visible ni accesible por los usuarios finales. Está formado por la base de datos y el software.

Existen dos tipos de estilos arquitectónicos que sirven para implementar Servicios Web:

* Servicios Web tradicionales: basados en una arquitectura orientada a servicios (SOA) y en la pila de protocolos WS-\*. La arquitectura SOA es un concepto de arquitectura software que utiliza una serie de componentes, denominados servicios, para crear aplicaciones. Cada uno de estos servicios de manera independiente intercambian datos y funcionalidades a los consumidores (AWS, s.f.). Los dos componentes más destacables de los Servicios Web tradicionales son: (IBM, 2022)
  + SOAP (*Simple Object Access Protocol*): es un protocolo estándar de mensajes basado en XML que permite la comunicación para intercambiar información entre diferentes máquinas.
  + WSDL (*Web Services Description Language*): es un estándar basado en XML utilizado para describir un Servicio Web.
* Servicios Web basados en el estilo arquitectónico REST: denominados comúnmente Servicios Web RESTful. REST (*Representacional State Transfer*) es un conjunto de criterios de diseños para Servicios Web que se centran en los recursos. No es una arquitectura en sí misma, pero se ha utilizado para el diseño de Servicios Web. (Richardson & Ruby, 2007)

Una de las principales características de REST es que es *stateless*, es decir, las peticiones entre cliente y servidor son “autocontenidas”, por lo tanto, el servidor no almacena el estado de la comunicación entre él y todos sus clientes. Debido a que el peso del almacenamiento del estado de la comunicación se encuentra en el cliente, permite que el servicio sea escalable.

Las restricciones de REST (*Representational State Transfer*) son un conjunto de principios arquitectónicos que definen las reglas y limitaciones para el diseño y desarrollo de Servicios Web. Estas restricciones son fundamentales para garantizar la escalabilidad, interoperabilidad y simplicidad de los sistemas basados en REST. Las restricciones se resumen en los siguientes aspectos:

* Definición e identificación de un recurso: la definición precisa de los recursos establece una base sólida para el diseño de servicios web RESTful, donde los recursos representan entidades o funcionalidades específicas. Mediante la asignación de URIs, cada recurso puede ser identificado y accedido de manera consistente. Esta claridad en la definición e identificación de los recursos facilita la interoperabilidad y la comunicación efectiva entre los componentes del sistema. (IBM, 2023)
* Representación de un recurso: cada recurso debe tener una representación específica que se envía al cliente al acceder a él. Esta representación, en formatos como JSON o XML, contiene toda la información necesaria para comprender y manipular el recurso. Al establecer una representación clara y completa, se fomenta la interoperabilidad y la reutilización de los servicios RESTful (Ekuan, 2023).
* HATEOAS (*Hypermedia As The Engine Of Application State*): implica el uso de enlaces hipermedia en las respuestas de la API, lo que permite a los clientes explorar de manera dinámica los recursos relacionados. Esta funcionalidad se asemeja a la navegación web, donde se siguen enlaces para alcanzar objetivos específicos (Gupta, 2022).
* Interfaz homogénea: para garantizar una manipulación sencilla de los recursos, todos ellos deben seguir una interfaz consistente. Esto se logra mediante el uso de un estándar bien definido. En el caso de la Web, esta interfaz se basa en el protocolo HTTP, que establece un conjunto de métodos para la manipulación de recursos.

El Modelo de Madurez de Richardson (*Richardson Maturity Model*) fue desarrollado por Leonard Richardson en 2008 con el propósito de clasificar los Servicios Web según su grado de adecuación al estilo arquitectónico de la Web. Este modelo utiliza cuatro niveles o capas que permiten evaluar la conformidad de un Servicio Web en términos de su uso de URI, HTTP e inclusión de hipermedia. El modelo de Richardson propone una clasificación de los servicios web mediante el nivel de madurez de cada uno: (Fowler, 2010)

* + Nivel 0: *The Swamp of POX*

Es el más básico. Se utiliza para servicios con un único recurso y método HTTP.

Ilustración 1: Nivel 0 del Modelo Madurez Richardson

Escala de tiempo

Descripción generada automáticamente con confianza baja

*Fuente: Richardson Maturity Model - Martin Fowler*

* + Nivel 1: *Resources*

Distingue entre diferentes recursos, pero con un único método HTTP.

Ilustración 2: Nivel 1 del Modelo Madurez Richardson

Gráfico

Descripción generada automáticamente con confianza baja

*Fuente: Richardson Maturity Model - Martin Fowler*

* + Nivel 2: HTTP *Verbs*

Distingue entre diferentes recursos y diferentes métodos HTTP.

Ilustración 3: Nivel 2 del Modelo Madurez Richardson

Imagen que contiene Escala de tiempo

Descripción generada automáticamente

*Fuente: Richardson Maturity Model - Martin Fowler*

* + Nivel 3: *Hypermedia Controls*

Hace uso de HATEOAS. Al hacer una solicitud (*request*) nos devolverá información adicional (*link*) sobre otro recurso.

Ilustración 4: Nivel 3 del Modelo Madurez Richardson

Escala de tiempo

Descripción generada automáticamente

*Fuente: Richardson Maturity Model - Martin Fowler*

## Algoritmo de recomendación

En este epígrafe se va a describir es estado del arte necesario para comprender el algoritmo de recomendación utilizado en el proyecto. Se expondrán los principales métodos de aprendizaje utilizados en la construcción de sistemas de recomendación. Además, se describirán las técnicas de análisis de datos prescriptivas y medidas de similitud utilizadas en la generación de recomendaciones personalizadas.

### Introducción a la ciencia de los datos

La ciencia de datos es un campo apasionante que combina habilidades estadísticas y cuantitativas avanzadas con programación con el objetivo de sacar valor a la inmensa cantidad de datos con los que nos encontramos en nuestro entorno académico o profesional, que es a lo que nos referimos cuando hablamos de “*Big Data*”. En general estas técnicas de análisis y explotación de datos es lo que popularmente se ha llamado “Inteligencia Artificial” y, más en concreto, “*Machine Learning*”, que son algoritmos con la habilidad de aprender sin haber sido explícitamente programados. (Kelleher & Tierney, 2021)

### *Machine Learning*: métodos de aprendizaje

*Machine Learning* son un conjunto de métodos capaces de detectar automáticamente patrones en los datos con el objetivo de definir modelos que permitan representar de una forma simplificada la realidad. *Machine Learning* implica definir algoritmos que se encargan de que el modelo aprenda de los datos (se ajuste a los datos), es decir, el algoritmo se encarga de crear una función matemática para representar la realidad a partir de los datos. (Roman, 2019)

En función del objetivo que persiguen hay dos familias de métodos de aprendizaje: (Cañadas, 2021)

* Métodos de aprendizaje predictivos o supervisados: el objetivo es estimar una variable a futuro, en función del tipo de variable distinguimos entre:
  + Clasificación: la variable es categórica pudiendo ser binaria (ej. sí/no, paga/no paga…), múltiple (ej. compra a, b o c) u ordenada (ej. riesgo alto, medio, bajo)
  + Regresión: la variable es numérica (ej. precio, cantidad, tiempo…)
* Métodos de aprendizaje descriptivos o no supervisados: el objetivo es explorar los datos
  + Agrupamiento o “*Clustering*”: el objetivo es agrupar los datos en conjuntos (“*clusters*”) que se parezcan dentro de un mismo conjunto y que se diferencien de otros conjuntos (Ej. clientes con hábitos de compra similares)
  + Reglas de asociación: buscan reglas (reglas de co-ocurrencia) que describen la mayor parte posible de los datos (ej. productos que se compran juntos)
  + Análisis de Componentes Principales (ACP)

### Técnicas de análisis de datos prescriptivas: sistemas de recomendación

Como acabamos de ver hay dos tipos de técnicas de análisis de datos, las predictivas y las descriptivas. Pero existe un tercer nivel de análisis de datos como extensión natural a los dos anteriores, la analítica prescriptiva que tiene como objetivo realizar recomendaciones sobre las acciones que se han de seguir, por ejemplo, para reducir costes o mejorar los beneficios. Algunos casos de uso típicos de la analítica prescriptiva son: generar planificaciones, establecer estrategias de precios, anticipar la afluencia de público a comercios, etc. (González, s.f.)

La finalidad de un sistema de recomendación es predecir la valoración que un usuario va a hacer de un ítem (libros, películas, páginas, productos, anuncios, etc.) que todavía no ha evaluado. Esta valoración se genera al analizar o las características de cada ítem o las valoraciones de cada usuario a cada ítem. Los sistemas de recomendación se utilizan para recomendar contenido personalizado a los usuarios en base a su perfil, preferencias o intereses.

A la hora de definir y diseñar un sistema de recomendación podemos optar por tres enfoques: (Bagnato, 2019)

* Filtrado colaborativo: las recomendaciones a un usuario se basan en las preferencias de “usuarios similares”.
* Filtrado basado en contenido: las recomendaciones se basan en sus propias preferencias, buscando elementos similares por los que ha mostrado interés.
* Filtrado basado en reglas: las recomendaciones se basan en reglas previamente definidas como, por ejemplo, en características sociodemográficas.

Hay dos tipos de filtrados colaborativos:

* Basados en usuario (*user-based*): para predecir la valoración que un usuario *A* hará de un ítem *X* que todavía no ha visto, se buscan usuarios con perfiles similares a *A* y se utilizan las valoraciones de estos otros usuarios sobre el ítem *X* como estimación de la valoración de *A*.
* Basados en ítems (*item-based*): para predecir la valoración que un usuario *A* hará de un ítem *X* que todavía no ha visto, se buscan otros ítems similares (en función del perfil de valoraciones que han recibido) y que el usuario *A* también haya valorado.

### Medidas de similitud o distancia

Los sistemas de recomendación utilizan el término distancia como cuantificación de la similitud o diferencia entre observaciones. Hay diferentes medidas de similitud: correlación de Pearson, coseno, Jaccard, diferencias de medias cuadradas, correlación de Spearman, etc. (Mendoza Olguín, Laureano De Jesús, & Pérez de Celis Herrero, 2019).

Una de las medidas de similitud más utilizada es el coseno que consiste en encontrar cómo dos vectores están relacionados entre ellos midiendo el coseno del ángulo entre los dos vectores. El coseno del ángulo que forman dos vectores puede interpretarse como una medida de similitud de sus orientaciones, independientemente de sus magnitudes. Si dos vectores tienen exactamente la misma orientación (el ángulo que forman es 0º) su coseno toma el valor de 1, si son perpendiculares (forman un ángulo de 90º) su coseno es 0 y si tienen orientaciones opuestas (ángulo de 180º) su coseno es de -1.

# Gestión del Proyecto

En este capítulo se explica la metodología software utilizada para el desarrollo del proyecto. Posteriormente, se expone la planificación inicial del proyecto con los tiempos estimados y la ejecución del proyecto con los tiempos reales.

## Modelo de ciclo de vida

Para el desarrollo del proyecto se ha decidido utilizar un modelo de cascada ya que es la metodología más adecuada en proyectos de pequeño tamaño con requisitos que no van a variar. Este modelo se divide en actividades (requisitos) están divididas en fases secuenciales contando con bucles por si es necesario volver hacia atrás para modificar/mejorar una fase (Risso, 2022). En función de la propuesta de Winston W. Royce en 1970, el modelo se divide en las siguientes fases: (Sommerville, 2005)

* Análisis y definición de los requerimientos: se elabora una lista detallada de tareas, gráficos de dependencia e hitos. Para lograr la definición de servicios, restricciones y metas del sistema, es importante especificar el producto y describir el sistema desde la perspectiva del usuario, identificando sus necesidades y objetivos. Para ello, se llevan a cabo consultas con los usuarios y se establecen los aspectos en detalle, los cuales sirven como especificación del sistema.
* Diseño del sistema y del *software*: incluye el diseño de la arquitectura (definición de los grandes bloques funcionales de la aplicación y su comunicación, además de elegir el estilo arquitectónico) y el diseño detallado (diseño detallado de cada uno de los componentes introducidos en la fase de diseño de la arquitectura).
* Implementación y prueba de unidades: implementación de la fase de diseño a través de pruebas de unidades, verificado que cada unidad cumpla su especificación.
* Integración y prueba del sistema: las unidades individuales se integran y se realizan pruebas del sistema completo.
* Funcionamiento y mantenimiento: puesta en producción del software y mantenimiento del mismo.

## Papeles desempeñados en el proyecto

Los dos tutores de este TFG, Sergio Saugar García y Alex Rayón Jerez, son los promotores y el alumno, Carlos Soler Martín es el director del proyecto, siendo también el analista-programador.

## Planificación y ejecución

A continuación, se listarán las tareas a realizar que se deberán llevar a cabo durante la realización del proyecto:

* Análisis de dominio de la aplicación, donde se definirán los procesos y actores que pertenecen al dominio analizado.
* Especificación de los requisitos obtenidos a través del análisis del dominio.
* Definición de una arquitectura de software que cumpla con los requisitos definidos.
* Diseño de la arquitectura (*backend* y *frontend*).
* Implementación.
* Pruebas y validación del proyecto.
* Documentar el proyecto.

El diagrama Gantt de la Ilustración 5 muestra una estimación del tiempo que llevaría la realización de cada una de las tareas mencionadas, siendo el inicio el 14 de enero de 2022 y la finalización el 22 de junio de 2022. Este diagrama se ha realizado antes de comenzar el proyecto para planificar y estimar los tiempos de cada tarea.

La Ilustración 6 muestra el tiempo real que se ha empleado en cada una de una de las tareas mencionadas, siendo el inicio el 14 de enero de 2022 y la finalización el 31 de mayo de 2023. El proyecto ha excedido la duración estimada, prolongándose exactamente por un año adicional.

Ilustración 5: Diagrama Gantt de la planificación del proyecto

Imagen que contiene Gráfico

Descripción generada automáticamente

*Fuente: Elaboración propia*

Ilustración 6: Diagrama Gantt de la ejecución del proyecto

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

*Fuente: Elaboración propia*

# Análisis y arquitectura

En este capítulo, se van a analizar aspectos cruciales del desarrollo de software, en concreto: el análisis de dominio, la especificación de requisitos, el análisis de casos de uso y la arquitectura del sistema. A medida que se exploran cada uno de estos temas, se examinan mejores prácticas, herramientas y metodologías que permiten comprender a fondo el entorno del problema, capturar las necesidades y deseos de los usuarios, identificar los casos de uso clave y establecer una arquitectura sólida que sustente el sistema.

## Análisis de dominio

Se ha analizado cómo funciona actualmente en el CEU el proceso de asignación de prácticas y primeros empleos a los alumnos universitarios y recién graduados, con el fin de identificar los problemas y buscar soluciones.

En el proceso intervienen tres actores: el departamento de carreras profesionales del CEU (administrador del proceso), los alumnos/recién graduados del CEU y las empresas que ofertan las practicas curriculares para alumnos y empleos para recién graduados.

Actualmente el CEU cuentan con 5 aplicaciones para la búsqueda de trabajo/prácticas de las cuales se utilizan dos: *GetHighered* (ofertas internacionales) y *JobTeaser* (ofertas nacionales). La funcionalidad en ambas aplicaciones es la misma. El estudiante/recién graduado crea su perfil y adjunta su CV. El CV debe contener toda la información relevante sobre el alumno que sea de interés para la empresa, como su nombre, email, teléfono, los idiomas que habla y con qué fluidez, los estudios y su nota media, sus hard *skills* (ej. marketing, matemáticas, estadística, lenguajes de programación, etc.) y sus *soft skills* (ej. capacidad analítica, trabajo en equipo, comunicación, liderazgo, etc.), en todos los casos con su nivel correspondiente. El alumno/recién graduado busca de manera manual las empresas a las que le gustaría aplicar. Cuando encuentra una oferta que le interesa, envía a través de la aplicación una pequeña carta de motivación a la empresa y se lo comunica al departamento de carreras profesionales del CEU, que son los encargados de asignar esa oferta al alumno una vez que la empresa le ha aceptado. Los alumnos con ofertas asignadas ya no reciben notificaciones de nuevas ofertas.

Por su parte, las empresas también deben de registrarse en la aplicación y publicar sus ofertas de trabajo y/o prácticas, aunque también es posible que el equipo de carreras profesionales reciba ofertas de las empresas por email, siendo ellos los que las publican en las plataformas de empleo. Cuando son las propias empresas las que publican la oferta de empleo es necesario que alguien del equipo de carreras profesionales valide la solicitud. Todos estos procesos son 100% manuales. Las ofertas de trabajo y/o prácticas contienen el nombre de la empresa, la definición del puesto, nombre de la persona de contacto y su número de teléfono, así como los requerimientos de: idiomas y nivel, estudios y nota media, *hard skills* (ej. marketing, matemáticas, estadística, lenguajes de programación, etc.) y *soft skills* (ej. capacidad analítica, trabajo en equipo, comunicación, liderazgo, etc.), en todos los casos con su nivel requerido correspondiente.

Los procesos y las aplicaciones para la búsqueda y asignación de prácticas y ofertas de primer empleo presentan los siguientes problemas: las aplicaciones tienen funcionalidades duplicadas, la forma de operar es diferente, los procesos son totalmente manuales, los CVs y las ofertas de prácticas/empleo no están digitalizadas y tienen contenidos heterogéneos y no existe un proceso automático de casación entre CVs y ofertas.

## Especificación de requisitos

El objetivo principal de este TFG es crear una aplicación para simplificar, hacer eficiente y automatizar el proceso de asignación de prácticas y primeros empleos a los alumnos universitarios y recién graduados. Los requisitos derivados del análisis de dominio son los siguientes:

**Requisitos funcionales**

* Desarrollo de 2 tipos de usuarios: alumno, empresa, y el administrador de la aplicación.
* Los usuarios podrán registrase en la aplicación.
* Los usuarios y el administrador podrán hacer *login* en la aplicación.
* Los usuarios y el administrador podrán cerrar sesión en la aplicación.
* Los usuarios podrán modificar sus datos personales.
* Estandarizar el CV y las ofertas: idiomas más hablados con el nivel del alumno, competencias, *soft* *skills* y *hard* *skills* más habituales en el mundo profesional y el nivel correspondiente del alumno.
* Los usuarios del tipo alumno podrán añadir su CV.
* Los usuarios del tipo alumno podrán modificar su CV.
* Los usuarios del tipo alumno podrán consultar su CV.
* Los usuarios del tipo alumno podrán ver las ofertas no asignadas de empresas registradas en la aplicación.
* Los usuarios del tipo alumno que no tengan ninguna oferta asignada podrán solicitar una recomendación de las ofertas sin asignar.
* Los usuarios del tipo empresa podrán crear nuevas ofertas.
* Los usuarios del tipo empresa podrán modificar sus ofertas.
* Los usuarios del tipo empresa podrán consultar los CV de los alumnos sin ofertas asignadas.
* Los usuarios del tipo empresa podrán consultar todas las ofertas que han creado.
* El administrador podrá asignar ofertas no asignadas a alumnos sin oferta asignada.

**Requisitos no funcionales:**

* Software escalable: la API puede responder a otro tipo de operaciones o realizar nuevas versiones tanto como se desee.
* Se requieren menos recursos del servidor, ya que no se mantiene el estado de la petición y por tanto no requiere memoria.
* Software interoperable: implementación del patrón Modelo-Vista-Controlador, el cual permite el desacoplamiento la vista del cliente de la lógica de la aplicación

## Análisis de los casos de uso

Con el objetivo de exponer de forma más detallada los procesos que intervienen en la aplicación, en la Ilustración 7 se presenta un diagrama de casos de uso.

Ilustración 7: Diagrama de casos de uso

*Diagrama

Descripción generada automáticamente*

*Fuente: Elaboración propia*

Como se puede observar en la Ilustración 7, existen 2 tipos de usuarios: alumno (con y sin oferta asignada) y empresa, además del administrador. Los usuarios necesitan estar registrados en el sistema para poder acceder al mismo, en caso de que no estén registrados deberán de crear una nueva cuenta.

El usuario del tipo alumno podrá añadir/modificar su CV, consultar su CV, ver todas las ofertas que no están asignadas a ningún otro alumno y modificar su perfil. Únicamente los alumnos que no tienen una oferta asignada podrán solicitar la recomendación de las ofertas.

El usuario del tipo empresa podrá crear y modificar sus ofertas, ver el CV de alumnos que no tienen ofertas asignadas, ver todas sus ofertas publicadas y modificar su perfil.

El administrador es el encargado de asignar las ofertas disponibles a los alumnos que no tengan ninguna oferta asignada.

## Arquitectura del sistema

La arquitectura propuesta para la aplicación es una arquitectura REST. Se trata de una arquitectura cliente-servidor (ver epígrafe 2.1).

Como se muestra en la Ilustración 8, la arquitectura se divide en dos partes:

* *Frontend*: parte visual con la que interactúa el cliente.
* *Backend*: contiene la lógica del programa. Está formado por tres componentes: la base de datos donde se almacena la información del sistema, el sistema de recomendación y, por último, el servicio web encargado de procesar los datos y conectar todos los componentes de la arquitectura.

Ilustración 8: Arquitectura del sistema

Diagrama

Descripción generada automáticamente

*Fuente: Elaboración propia*

La comunicación entre el cliente y el servidor se lleva a cabo utilizando el protocolo HTTP. Este proceso se realiza mediante peticiones HTTP. En el flujo de interacción, el cliente envía solicitudes al servidor utilizando los métodos definidos por HTTP, como GET, POST, PUT y DELETE, según el tipo de acción que se desea realizar en los recursos. Estas solicitudes contienen información relevante, como parámetros, encabezados y, en algunos casos, datos que se desean enviar al servidor. El servidor, por su parte, procesa las solicitudes recibidas y realiza las operaciones correspondientes en los recursos solicitados. Una vez que el servidor ha completado la acción solicitada, responde al cliente con una respuesta HTTP que incluye un código de estado (como 200 OK para una solicitud exitosa) y, cuando corresponde, el resultado de la operación en formato JSON. En las siguientes secciones se profundizará en el diseño de cada uno de los componentes descritos en la arquitectura del sistema.

# Diseño

En este capítulo, se detalla el diseño de la plataforma basado en la arquitectura propuesta. La arquitectura divide la plataforma en dos componentes principales: el *backend* y el *frontend*. Siguiendo esta división, se describirá el diseño de cada uno de ellos, teniendo en cuenta sus funcionalidades y requisitos particulares.

## Diseño del *backend*

En este epígrafe, se abordará el diseño del *backend* de la aplicación, centrándonos en tres aspectos clave: el diseño del servicio web RESTful, el diseño del algoritmo de recomendación y el diseño de la base de datos. Estos componentes son fundamentales para el funcionamiento eficiente y efectivo de la aplicación, permitiendo la interacción con los usuarios, la generación de recomendaciones personalizadas y el almacenamiento de datos de manera estructurada. A lo largo de los siguientes epígrafes, se analizará cada uno de estos aspectos en detalle.

### Diseño del Servicio Web RESTful

En este epígrafe, se abordará la definición y diseño de un servicio RESTful, centrándonos en los elementos clave que lo componen. Se explorarán conceptos como recursos, URIs, representaciones y protocolo HTTP. Estos componentes son fundamentales para la implementación y el funcionamiento adecuado de un servicio RESTful.

#### Definición de un recurso

Tras el análisis de dominio realizado se han obtenido los siguientes recursos:

* Alumno: el recurso Alumno recolecta el nombre, los apellidos, el correo electrónico, el número de teléfono, el nombre de usuario y la contraseña de un alumno.
* Alumnos: representa al conjunto de alumnos existentes dentro de la aplicación.
* Empresa: el recurso Empresa recolecta el nombre de la empresa, el correo electrónico, el número de teléfono, el nombre de usuario y la contraseña de una empresa.
* Empresas: representa al conjunto de empresas existentes dentro de la aplicación.
* Administrador: el recurso Administrador recolecta el nombre de usuario y la contraseña de un administrador.
* Administradores: permite listar todos los administradores registrados en la aplicación.
* Oferta: este recurso representa las diferentes ofertas que existen en el sistema. Una oferta contiene el nombre la empresa que la crea, un título significativo, la ciudad donde realizará el trabajo, un nombre y teléfono de contacto. Por otro lado, contiene la descripción de la oferta que incluye, entre otros, la titulación requerida para la misma y la nota media. También establece los niveles que el candidato debe tener en relación con los idiomas, soft-skills, competencias y lenguajes de programación, cuyo valor asignado es: no aplica, nivel básico, nivel medio y nivel avanzado. Los niveles necesarios por cada oferta son:
  + - Idiomas: inglés, alemán y francés.
    - Soft-Skills: trabajo en equipo y comunicación.
    - Competencias: matemáticas, estadística, gestión de proyectos, sostenibilidad y big data.
    - Programación: nivel de programación en general.

Eventualmente la oferta podrá ser asignada y, en ese caso, contendrá una referencia al recurso Alumno al que se le ha asignado la oferta.

* Ofertas: permite listar todas las ofertas y también permite registrar una nueva oferta.
* CV: este recurso representa los Curriculum Vitae creados por cada alumno que existen en el sistema. Una CV contiene una referencia alumno que lo ha creado, el grado que ha cursado/estudiado el alumno y la nota media obtenida durante sus estudios. También establece los niveles que el candidato debe tener en relación con los idiomas, soft-skills, competencias y lenguajes de programación, cuyo valor asignado es: no aplica, nivel básico, nivel medio y nivel avanzado. Los niveles necesarios por cada CV son:
  + - Idiomas: inglés, alemán y francés.
    - Soft-Skills: capacidad analítica, trabajo en equipo, comunicación, pensamiento crítico, innovación, liderazgo, toma de decisiones y resolución a problemas.
    - Competencias: marketing, e-commerce, diseño gráfico, matemáticas, estadística, gestión de proyectos, redes sociales, sostenibilidad, inteligencia artificial, big data, machine learning, análisis de datos, bases de datos, cloud, IoT, redes, sistemas operativos, desarrollo web y diseño web.
    - Lenguajes Programación: R, Java, Pascal y Python.

#### Patrones de las URIs

En todo Servicio Web RESTful los recursos deben estar identificados mediante URIS. A continuación, se exponen los patrones definidos para los distintos recursos definidos:

* + **/alumnos**: se identifican todas las peticiones HTTP relacionadas con los alumnos.
  + **/alumnos/:id**: se identifican todas las peticiones HTTP relacionadas un alumno especifico.
  + **/alumnos/:id/CV**: se identifican todas las peticiones HTTP relacionadas con el CV de un alumno especifico.
  + **/empresas**: se identifican todas las peticiones HTTP relacionadas con las empresas.
  + **/empresas/:id**: se identifican todas las peticiones HTTP relacionadas una empresa especifica.
  + **/empresas/:id/ofertas**: se identifican todas las peticiones HTTP relacionadas con todas las ofertas de una empresa especifica.
  + **/empresas/:id/ofertas/:id**: se identifican todas las peticiones HTTP relacionadas con una oferta especifica de una empresa especifica.
  + **/ofertas**: se identifican todas las peticiones HTTP relacionadas con las ofertas.
  + **/ofertas/:id**: se identifican todas las peticiones HTTP relacionadas con una oferta especifica.
  + **/ofertas/cvs**: se identifican todas las peticiones HTTP relacionadas con los CV de los alumnos en relación con la oferta asignada a dicho alumno.

#### API de cada recurso

A continuación, se muestran las APIs de cada recurso recogidas desde la Tabla 1 hasta la Tabla 10:



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Método | URI | Utilidad | Semántica | Cuerpo Solicitud | Códigos de respuesta |
| POST | /alumnos | Añade un nuevo alumno. | JSON | Representación de un alumno | 201 – Created  400 – Bad Request  500 – Internal Server |
| GET | / alumnos | Se obtiene la representación | JSON |  | 200 – OK  400 – Bad request  500 – Internal Server Error |
| PUT, PATCH, DELETE | - | - | - | - | 405 – Method not Allowed |

Tabla 1: Recurso /alumnos

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Método | URI | Utilidad | Semántica | Cuerpo Solicitud | Códigos de respuesta |
| GET | /alumnos/:id | Se obtiene la representación de un alumno por su id | JSON | - | 200 – OK  404 – Not Found  400 – Bad request  500 – Internal Server Error |
| PUT | /alumnos/id | Modificar atributos del alumno | JSON | - | 200 – OK  404 – Not Found  400 – Bad request  500 – Internal Server Error |
| POST, PATCH, DELETE | - | - | - | - | 405 – Method not Allowed |

Tabla 2: Recurso /alumnos/id

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Método | URI | Utilidad | Semántica | Cuerpo Solicitud | Códigos de respuesta |
| POST | /alumnos/:id/CV | Creación del CV de un alumno especifico mediante su id | JSON | Atributos del CV de un alumno | 201 – Created  400 – Bad Request  500 – Internal Server Error |
| GET | /alumnos/id/CV | Se obtiene el CV de un alumno por su id | JSON | - | 200 – OK  404 – Not Found  400 – Bad request  500 – Internal Server Error |
| PUT | /alumnos/id/CV | Creación y modificación del CV un alumno especifico mediante su id | JSON | Atributos del CV de un alumno | 200 – OK  404 – Not Found  400 – Bad request  500 – Internal |
| DELETE, PATCH | - | - | - | - | 405 – Method not Allowed |

Tabla 3: Recurso /alumnos/id/CV

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Método | URI | Utilidad | Semántica | Cuerpo Solicitud | Códigos de respuesta |
| POST | /empresas | Añade una nueva empresa. | JSON | Atributos de una nueva empresa | 201 – Created  400 – Bad Request  500 – Internal Server Error |
| GET | /empresas | Se obtienen un listado de todas las empresas. | JSON |  | 200 – OK  404 – Not Found  400 – Bad request  500 – Internal Server Error |
| PUT, DELETE, PATCH | - | - | - | - | 405 – Method not Allowed |

Tabla 4: Recurso /empresas

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Método | URI | Utilidad | Semántica | Cuerpo Solicitud | Códigos de respuesta |
| GET | /empresas/id | Se obtiene todos los atributos de una empresa por su id | JSON | - | 200 – OK  404 – Not Found  400 – Bad request  500 – Internal Server Error |
| PUT | /empresas/id | Modificar un atributo de una empresa | JSON | Atributos de una empresa | 200 – OK  404 – Not Found  400 – Bad Request  500 – Internal Server Error |
| POST, PATCH, DELETE | - | - | - | - | 405 – Method not Allowed |

Tabla 5: /empresas/id



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Método | URI | Utilidad | Semántica | Cuerpo Solicitud | Códigos de respuesta |
| POST | /empresas/id/ofertas | Una empresa especifica crea una nueva oferta | JSON | Atributos de una oferta | 201 – Created  400 – Bad Request  500 – Internal Server Error |
| GET | /empresas/id/ofertas | Se obtiene un listado de todas las ofertas de una empresa especifica | JSON |  | 200 – OK  404 – Not Found  400 – Bad request  500 – Internal Server Error |
| PUT, DELETE, PATCH | - | - | - | - | 405 – Method not Allowed |

Tabla 6: Recurso /empresas/id/ofertas

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Método | URI | Utilidad | Semántica | Cuerpo Solicitud | Códigos de respuesta |
| GET | /empresas/id/ofertas/id | Se obtiene una oferta especifica de una empresa especifica | JSON |  | 200 – OK  404 – Not Found  400 – Bad request  500 – Internal Server Error |
| PUT | /empresas/id/ofertas/id | Modificación del campo de alumno asignado y del campo estado de la oferta | JSON | Atributo alumno y asignado de una oferta | 200 – OK  404 – Not Found  400 – Bad request  500 – Internal |
| POST, DELETE, PATCH | - | - | - | - | 405 – Method not Allowed |

Tabla 7: Recurso /empresas/id/ofertas/id

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Método | URI | Utilidad | Semántica | Cuerpo Solicitud | Códigos de respuesta |
| GET | /ofertas | Se obtiene todos los atributos de todas las ofertas | JSON |  | 200 – OK  404 – Not Found  400 – Bad request  500 – Internal Server Error |
| GET | /ofertas?estado=”SIN ASIGNAR” | Se obtiene todos los atributos de las ofertas que están sin asignar. | JSON | - | 200 – OK  404 – Not Found  400 – Bad Request  500 – Internal Server Error |
| GET | /ofertas?alumno=alumno\_id | Se obtiene la oferta asignada de un alumno especifico. | JSON | - | 200 – OK  404 – Not Found  400 – Bad Request  500 – Internal Server Error |
| POST, DELETE, PATCH | - | - | - | - | 405 – Method not Allowed |

Tabla 8: Recurso /ofertas

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Método | URI | Utilidad | Semántica | Cuerpo Solicitud | Códigos de respuesta |
| GET | /ofertas/id | Se obtiene una oferta especifica | JSON |  | 200 – OK  404 – Not Found  400 – Bad request  500 – Internal Server Error |
| POST, DELET, PATCH, PUT | - | - | - | - | 405 – Method not Allowed |

Tabla 9: Recurso /ofertas/id

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Método | URI | Utilidad | Semántica | Cuerpo Solicitud | Códigos de respuesta |
| GET | /ofertas/cvs?estado=”ASIGNADA” | Se obtiene todos los CV de los alumnos con una oferta asignada | JSON | - | 200 – OK  404 – Not Found  400 – Bad request  500 – Internal Server Error |
| POST, DELET, PATCH, PUT | - | - | - | - | 405 – Method not Allowed |

Tabla 10: Recurso /ofertas/cvs

### Diseño del algoritmo de recomendación

El objetivo del algoritmo de recomendación es, cuando un alumno sin ofertas asignadas lo solicita, recomendarle las ofertas no asignadas que más se adaptan a su CV.

De entre las alternativas para desarrollar un algoritmo de recomendación expuestas en el epígrafe 2.2.3, se ha optado por un de filtrado colaborativo. Este diseño permite recomendar a los alumnos sin oferta asignada ofertas de trabajo o puesto de prácticas a las que son más afines en función de su perfil, es decir, su CV es afín a los requerimientos de la oferta. De entre las medidas de similitud expuestas en el epígrafe 2.2.4, se ha optado por utilizar el coseno como medida de similitud.

El diseño del algoritmo de recomendación se fundamenta en el cálculo de similitud. Sin embargo, en este caso, la similitud no se calcula directamente entre el CV del alumno que solicita la recomendación y los requisitos de las ofertas sin asignar. Esta decisión se toma con el objetivo de simplificar el proceso de creación de las ofertas por parte de las empresas, ya que solo necesitarán ingresar 13 parámetros para definir las ofertas, en contraste con los 36 parámetros que se encuentran en los CV de los alumnos. El algoritmo de recomendación estructura el cálculo de la similitud en 2 APIs cuya implementación se verá en los epígrafes 6.1.2.2. y 6.1.2.3.:

1. Cálculo de similitud de ofertas sin asignar y ofertas que están asignadas: calcula la similitud, en función de los atributos de las ofertas (13 parámetros), entre las ofertas sin asignar que las empresas han imputado en el sistema y las ofertas que ya han sido asignadas a los alumnos por el administrador.
2. Cálculo de similitud entre alumnos sin oferta asignada y alumnos con oferta asignada: calcula la similitud, en función del CV (36 parámetros), de un alumno sin oferta asignada con los alumnos que tienen una oferta asignada. Cuando un alumno sin oferta asignada solicita recomendación, el sistema le recomienda la/s oferta/s sin asignar que son similares a las ofertas asignadas a alumnos similares a él. Se considera que si dos alumnos son similares pueden recibir ofertas similares, es decir, tienen afinidad a ofertas similares.

### Diseño de la base de datos

Como se ha descrito detalladamente en el epígrafe 4.4 de la arquitectura propuesta, resulta crucial contar con un sistema de almacenamiento de datos para asegurar la coherencia y consistencia de la información. En este sentido, se ha tomado la decisión de emplear una base de datos para llevar a cabo dicha tarea. El diseño de la base de datos ha sido elaborado tomando en cuenta los recursos y procesos identificados durante el análisis del dominio, como se detalla en el epígrafe 4.1. A continuación, se muestra en la Ilustración 9 el diagrama Entidad/Relación propuesto que refleja las relaciones entre las diferentes tablas o entidades:

Ilustración 9: Diagrama E/R de la base de datos

Imagen que contiene texto, estacionamiento, mucho, tabla

Descripción generada automáticamente

*Fuente: Elaboración propia*

En el diagrama E/R presentado, se encuentran definidos todos los recursos que forman parte de la lógica de la aplicación, junto con las relaciones que existen entre ellos. Como se puede ver en la Ilustración 9, cada una de las tablas mostradas contiene un PRIMARY KEY (PK). Otras tablas, como cv y ofertas, contienen una o varias FOREIGN KEY (FK), las cuales hacen referencia a la PK de otras tablas. A continuación, se describen, con más detalle, las tablas y los atributos de cada de cada una de ellas:

* Alumnos: tabla que almacena todos los alumnos registrados con las siguientes columnas: alumno\_id (*Primary Key*), *username*, *password*, nombre, apellido, teléfono y email.
* Admin: tabla que almacena todos los administradores registrados con las siguientes columnas: admin\_id (*Primary Key*), username y *password.*
* Empresas: tabla que almacena todas las empresas registradas con las siguientes columnas: empresa\_id (*Primary Key*), *username*, *password*, empresa\_nombre, teléfono y email.
* Cv: tabla que almacena todos los cv de cada alumno con las siguientes columnas: Id (*Primary Key*), alumno\_id (*Foreign Key*), grado, nota\_media, niveles de idiomas, de *soft-skills*, de competencias y lenguajes de programación:
  + Idiomas: inglés, alemán y francés.
  + Soft-Skills: capacidad analítica, trabajo en equipo, comunicación, pensamiento crítico, innovación, liderazgo, toma de decisiones y resolución a problemas.
  + Competencias: marketing, e-commerce, diseño gráfico, matemáticas, estadística, gestión de proyectos, redes sociales, sostenibilidad, inteligencia artificial, big data, machine learning, análisis de datos, bases de datos, cloud, IoT, redes, sistemas operativos, desarrollo web y diseño web.
  + Lenguajes Programación: R, Java, Pascal y Python.
* Ofertas: tabla que almacena todas las ofertas de cada empresa en las siguientes columnas: job\_id: (*Primary* *Key*), alumno\_id (*Foreign* *Key*), empresa\_id (Foreign Key), empresa\_nombre, job\_title, ciudad, teléfono, nombre\_contacto, estado, nota\_media, grado, niveles de idiomas, de *soft*-*skills*, de competencias y lenguajes de programación:
  + Idiomas: inglés, alemán y francés.
  + Soft-Skills: trabajo en equipo y comunicación
  + Competencias: matemáticas, estadística, gestión de proyectos, sostenibilidad, y big data.
  + Programación.

## Diseño del *frontend*

Como se ha mencionado en el epígrafe 4.4, la arquitectura planteada se divide en dos partes: el *backend, expuesto en el* epígrafe*; y* el *frontend* (interfaz de usuario), cuyo diseño va a ser presentado en este epígrafe.

La interfaz de usuario está formada por varias páginas con diferentes funcionalidades, para ello se ha realizado un diseño de cada una de las pantallas de la aplicación. A continuación, en la Ilustración 10 se muestra como ejemplo la pantalla de recomendación de ofertas, el resto de las pantallas diseñadas se incluyen en el Anexo A.

Ilustración 10: Ventana Recomendador de ofertas

Interfaz de usuario gráfica

Descripción generada automáticamente

*Fuente: Elaboración propia*

Como se puede ver en la Ilustración 10, la pantalla cuenta con una barra de menú en la parte superior con el logo de la aplicación y con botones para poder navegar entre las otras páginas de la aplicación. En la zona central se encuentra el título de la página con una breve descripción sobre el sistema de recomendación de las ofertas. Además, cuenta con un botón situado debajo de la descripción, que al ser pulsado mostraría una tabla con todas las ofertas que se le recomiendan al alumno y la información relevante de las ofertas (Empresa, Trabajo, Ciudad, …).

# Implementación

En este capítulo, se describe la implementación de la plataforma basada en el diseño propuesto en el capítulo anterior. A continuación, se aborda la implementación de los dos componentes principales de la arquitectura del sistema: el *backend* y el *frontend*. Para facilitar el entendimiento de la implementación de la aplicación, se ha elaborado un ejemplo: Pablo Sotelo, alumno del CEU, está buscando prácticas, para ello accede con su cuenta a la aplicación para ver que empresas le recomiendan en función de su CV.

## Implementación *backend*

En este epígrafe, se abordará la implementación del *backend* de la aplicación, centrándose en la implementación de tres aspectos clave: del servicio web RESTful, del algoritmo de recomendación y de la base de datos. A lo largo de los siguientes epígrafes, se analizará cada uno de estos aspectos en detalle.

### Implementación Servicio REST

El RWS (*Representational Web Service*) ha sido implementado utilizando el Flask, que es un *framework* de desarrollo web en Python que proporciona una forma sencilla y flexible de construir aplicaciones web. Para mapear el diseño del RWS en Flask, se utilizan varias características clave del *framework*:

* Los recursos del RWS se pueden representar mediante clases en Flask. Cada recurso se define como una clase que hereda de “*flask.Resource*”. Dentro de esta clase, se pueden implementar los diferentes métodos HTTP, como GET, POST, PUT o DELETE, que representan las acciones que se pueden realizar en el recurso.
* Las URIs del RWS se mapean en Flask utilizando las rutas definidas en las rutas de la aplicación. Flask proporciona un decorador “@app.route” que permite asociar una función o método de una clase con una determinada URI. Al definir estas rutas, se especifica el patrón de la URI y se asigna a la función o método correspondiente en Flask.
* El protocolo HTTP se maneja automáticamente en Flask. Por ejemplo, cuando se realiza una solicitud GET a una determinada URI definida en Flask, el *framework* se encarga de enrutar la solicitud a la función o método correspondiente para manejarla adecuadamente. Del mismo modo, Flask se encarga de manejar las respuestas HTTP y generar las respuestas adecuadas, ya sea en formato JSON, XML u otros formatos, según lo especificado en el diseño del RWS.

La implementación del servicio RESTful del programa se ha realizado en el fichero “app.py”, el cual contiene toda la lógica del programa y los patrones de las URIS con los métodos definidos en el apartado 5.1.1. A continuación, se exponen 2 partes de código del fichero “*app.py”*: en la Ilustración 11 se muestra parte del código del método GET para la obtención de todas las ofertas que están sin asignar y en la Ilustración 12 se muestra una parte del código donde se define el método GET mediante REST para ver todas las ofertas.

Ilustración : Parte del código para ver ofertas disponibles

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

*Fuente: Elaboración propia*

Ilustración : Parte del código REST del método GET de las ofertas

Texto

Descripción generada automáticamente

*Fuente: Elaboración propia*

Para ilustrarlo con el ejemplo del alumno Pablo Sotelo, cuando este decide ver que ofertas hay disponibles en la aplicación, debe acceder a la página donde se ven todas las ofertas disponibles, ejecutándose el código de la Ilustración 11, el cual realiza dos peticiones GET para mostrarlas en el *frontend*:

* Una primera petición GET para obtener todas las ofertas sin asignar. Cuando se realiza esta petición se le pasa el parámetro “SIN ASIGNAR”, el cual, va a ser necesario para que el código de la Ilustración 12 pueda filtrar mediante la variable estado y así poder realizar la consulta a la base de datos de todas las ofertas que no tienen ningún alumno asignado.
* Otra petición GET para comprobar si Pablo Sotelo tiene ya una oferta asignada, en la que se le pasa el id del alumno para que así la parte de código de la Ilustración 12 pueda realizar una consulta a la base de datos comprobando si el alumno Pablo Sotelo tiene una oferta asignada o no.

### Implementación Algoritmo recomendación

En este epígrafe, se abordará la implementación del algoritmo de recomendación de la aplicación, centrándose en la estructura de dicho algoritmo y la implementación de las APIs necesarias para el funcionamiento del algoritmo de recomendación.

#### **Estructura del algoritmo de recomendación**

Como lenguaje de programación del algoritmo se ha elegido R ya que presenta las siguientes ventajas: excelente gama de paquetes de código abierto y de alta calidad, dispone funciones y métodos estadísticos integrales muy completos. También se ha considerado que R tiene como principal desventaja el rendimiento, que no es el mejor, es decir, R es lento en comparación con otros lenguajes de programación, pero para programar el algoritmo de recomendación es lo suficientemente rápido.

El algoritmo de recomendación se estructura de la siguiente manera:

* apis.R: contiene el código de las siguientes APIs:
  + “Calculador similaridad ofertas sin asignar” (ver epígrafe 6.1.2.2): se ha implementado siguiendo el diseño del cálculo de similitud de ofertas sin asignar y ofertas que están asignadas descrito en el epígrafe 5.1.2.
  + “Recomendador ofertas alumno sin oferta asignada” (ver epígrafe 6.1.2.3): se ha implementado siguiendo el diseño del cálculo de la similitud entre alumnos sin oferta asignada y alumnos con oferta asignada descrito en el epígrafe 5.1.2.
* llamadaAPIS.R: llama al fichero “apis.R” mediante la librería *plumber* de R y lo ejecuta en el puerto 8015.
* vecinos\_ofertas\_sin\_asignar.xslx: es un fichero en formato excel que relaciona las dos APIs y contiene los vecinos de las ofertas sin asignar. Es el output de la API “Calculador similaridad ofertas sin asignar” y a su vez el input de la API “Recomendador ofertas alumno sin oferta asignada”.

#### **API “Calculador similaridad ofertas sin asignar”**

Está API se ejecutará únicamente cuando una empresa cree una oferta nueva o cuando el administrador asigne una oferta sin asignar a un alumno sin oferta asignada.

Esta API realiza un GET de la URI ‘/ofertas’ para poder obtener todas las ofertas que hay registradas en la base de datos, tanto las asignadas como las no asignadas. El resultado de la petición GET se transforma de un JSON a una matriz para poder tratar los datos en R. Las columnas de la matriz de ofertas obtenida se ordenan y se eliminan las columnas que no son relevantes para el análisis.

Se transpone la matriz de ofertas y, utilizando el método del coseno, se calcula la similitud (por columnas) entre todas las ofertas. Cada oferta es un vector de 13 variables (requerimientos), si dos vectores son iguales el ángulo que forman es 0º y su coseno toma el valor de 1 (máxima similitud), mientras que si son totalmente diferentes forman un ángulo de 90º y su coseno es 0 (similitud nula).

La matriz de similitud entre ofertas muestra la similitud de cada oferta con el resto, evidentemente cada oferta tiene similitud 1 consigo misma. Por ejemplo, si una oferta tiene una similitud de 0,9348 con otra, implica que los vectores de dichas ofertas forman un ángulo de 20,80º y su coseno es 0,9348. Lo cual implica qué de los 13 requerimientos de dichas ofertas, 10 son iguales y 3 (20,80º/90º x 13) son diferentes.

Se genera la matriz de vecinos que contiene las 6 ofertas más similares a cada oferta. La primera oferta más similar a una determinada oferta es ella misma con una similitud de 1. Una vez eliminada la columna del primer vecino, se vuelve añadir la columna estado de la oferta para poder quedarse únicamente con una matriz que contiene los vecinos de las ofertas que están sin asignar. Finalmente se salva el archivo "vecinos\_ofertas\_sin\_asignar.xlsx" que se convierte en un input de la API “Recomendador ofertas alumno sin oferta asignada”

#### **API “Recomendador ofertas alumno sin oferta asignada”**

Está API se ejecutará únicamente cuando un alumno que no tenga ninguna oferta asignada acceda a la página ‘Recomendador’ y pulse el botón para obtener las ofertas no asignadas que le recomienda el algoritmo.

Primero se realizan varias peticiones GET para recoger los datos con los que va a trabajar:

* URI ‘/alumnos/id/CV’ para poder obtener el CV del alumno que ha solicitado la recomendación. El parámetro ‘id’ (id del alumno que solicita recomendación) de esta URI es el parámetro que solicita la función que ejecuta esta API. El resultado de la petición GET se transforma de un JSON a una matriz para poder tratar con datos en R.
* URI ‘/ofertas’ para poder obtener todas las ofertas registradas en la base de datos. El resultado de la petición GET se transforma de un JSON a una matriz para poder tratar los datos en R, que posteriormente se filtra a partir de la columna estado, para obtener una única matriz con las ofertas sin asignar.
* URI ‘/ofertas/cvs?estado=”ASIGNADA”’ para poder obtener el CV de todos los alumnos que tienen una oferta asignada. El resultado de la petición GET se transforma de un JSON a una matriz para poder tratar los datos en R.

Esta API parte con cuatro inputs: las tres matrices obtenidas de cada petición GET descritas anteriormente y el archivo “vecinos\_ofertas\_sin\_asignar.xlsx" que se ha obtenido como resultado de la API “Calculador similaridad ofertas sin asignar”.

Se genera la matriz de alumnos que es la unión de la matriz de alumnos con ofertas asignadas y la matriz del alumno que solicita la recomendación. Se eliminan las variables (columnas) no relevantes y se obtiene una matriz de alumnos que contiene únicamente el CV de cada alumno estructurado en 36 skills.

Se transpone la matriz de alumnos y, utilizando el método del coseno, se calcula la similitud (por columnas) entre los todos los alumnos. Cada alumno es un vector de 36 variables (skills). La matriz de similitud entre alumnos muestra la similitud de cada alumno con el resto, evidentemente cada alumno tiene similitud 1 consigo mismo. Por ejemplo, si un alumno tiene una similitud de 0,9059 con otro alumno, implica que los vectores de dichos alumnos forman un ángulo de 25,05º y su coseno es 0,9059. Lo cual implica que de las 36 skills, 26 son iguales y 10 (25,05º/90º x 36) son diferentes.

Se genera la matriz de vecinos que contiene los 6 alumnos más similares a cada alumno. El primer alumno más similar a un determinado alumno es él mismo con una similitud de 1. Una vez eliminada la columna del primer vecino se vuelve a añadir la columna ‘alumno\_id’ para poder filtrar únicamente los 5 vecinos del alumno que ha solicitado la recomendación.

Se identifican los vecinos comunes entre el vector de los 5 vecinos del alumno que ha solicitado la recomendación y la matriz de los 5 vecinos de las ofertas sin asignar (archivo “vecinos\_ofertas\_sin\_asignar.xlsx"). Una vez identificado los vecinos comunes, se crea una matriz que contiene los datos importantes de cada oferta sin asignar (título de la oferta, empresa, ciudad, …). Finalmente, la API devuelve un output es un JSON que muestra al alumno que ha solicitado la recomendación la/s oferta/s sin asignar que son similares a las ofertas asignadas a alumnos similares a él.

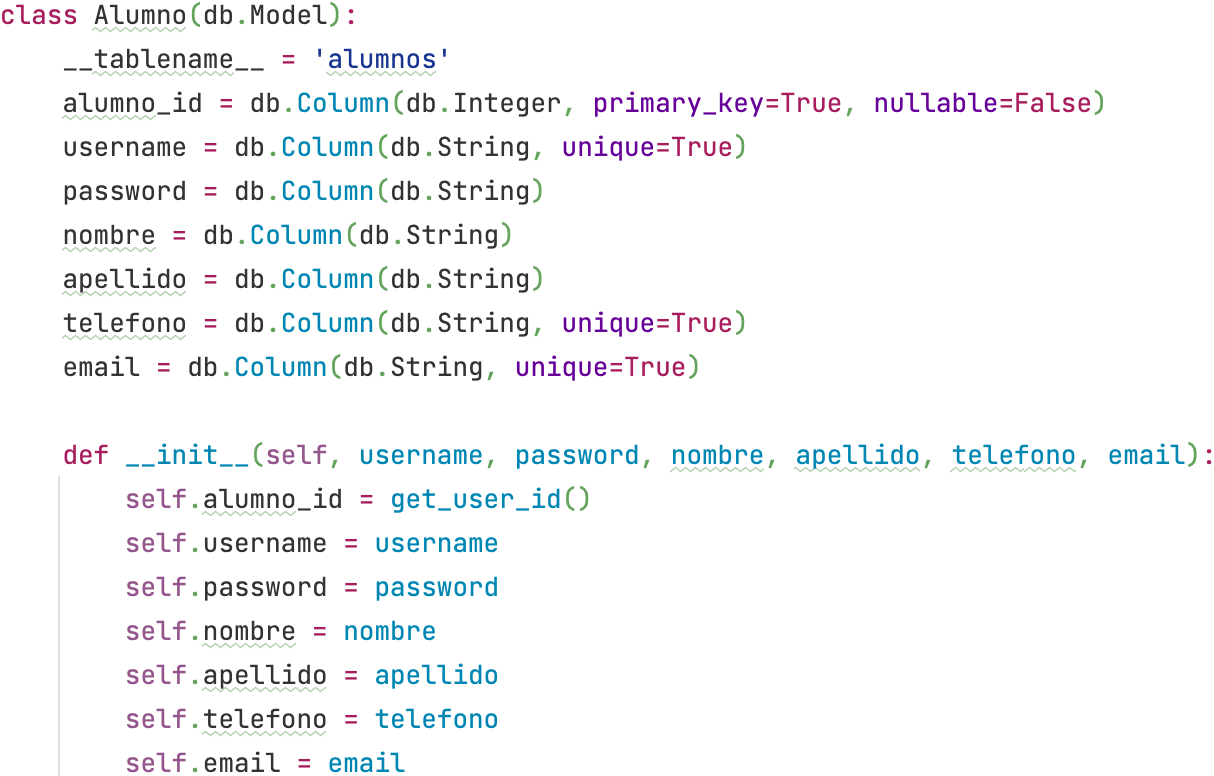
Para ilustrarlo con el ejemplo del alumno Pablo Sotelo. Imaginemos que Accenture ha creado una nueva oferta de trabajo, la API “Calculador similaridad ofertas sin asignar” se ejecutará y devolverá una nueva versión del output ‘vecinos\_ofertas\_sin\_asignar.xslx’ que contiene cual son los vecinos (5 ofertas asignadas más similares) de la nueva oferta de Accenture. Cuando el Pablo Sotelo accede a la página donde se encuentra el recomendador y solicita recomendación de las ofertas no asignadas, se ejecutará la API “Recomendador ofertas alumno sin oferta asignada” la cual devolverá como output un JSON que mostrará a Pablo la/s oferta/s sin asignar que son similares a las ofertas asignadas a alumnos similares a él.

### Implementación Base de datos

Para implementar el diseño de la base de datos propuesto en el epígrafe 5.1.3. se ha utilizado la base de datos PostGresql.La conexión y configuración de la base de datos se ha realizado con el ORM SQLAlchemy. Dentro de la aplicación la implementación de la base de datos se organiza en los siguientes directorios:

* migrations: es una extensión que se genera cada vez que se hace una nueva migración y actualiza de la base de datos.
* model: contiene las clases: Alumno, CV, Ofertas, Empresa y Admin, además de los métodos para realizar las consultas a la base de datos. Cada vez que se actualiza una clase hay que realizar una migración para que se actualice la base de datos. En la Ilustración 13 se muestra un ejemplo de la clase Alumno:

Ilustración : Clase Alumno de la base de datos en python



*Fuente: Elaboración propia*

Como se observa en la Ilustración 13, primero se definen cada uno de los atributos del Alumno y que tipo de dato son, posteriormente se inicializan los atributos del alumno mediante el método init.

## Implementación Frontend

Se han utilizado plantillas HTML para representar y mostrar todos los recursos de una manera más visual y atractiva para el usuario final. Además de HTML, se ha utilizado para complementar y mejorar el atractivo los ficheros CSS y JavaScript.

# Despliegue y pruebas

En este capítulo, se presenta una descripción detallada del despliegue de la aplicación en la plataforma de servicios de la nube de *Amazon Web Services* (AWS), así como las pruebas realizadas para su implementación y mantenimiento. Se abordarán los pasos y configuraciones necesarios para llevar a cabo el despliegue exitoso de la aplicación en un entorno de AWS, garantizando su escalabilidad, seguridad y rendimiento óptimo. Además, se explicarán las pruebas y validaciones llevadas a cabo durante el proceso de implementación para asegurar el correcto funcionamiento de la aplicación en el entorno de producción.

## Despliegue

Para que la aplicación sea accesible desde cualquier dispositivo y desde cualquier lugar, se ha desplegado en la plataforma AWS Cloud Para ello, se ha elegido una instancia EC2 de la capa gratuita con la distribución de Linux, Ubuntu. Dicha instancia ofrece un almacenamiento óptimo para la carga de trabajo, configuraciones de CPU optimizadas y alto rendimiento de las redes. Para el despliegue de la aplicación, en esta instancia se ha clonado el repositorio del proyecto. Dentro de la aplicación, se han creado dos scripts para facilitar el despliegue:

* R.py: encargado de ejecutar el archivo “*apis.R”* (ver el epígrafe 6.1.2.1) a través de Python con la librería rpy2. Como se observa en la Ilustración 14, este fichero se ha creado para poder ejecutar el código que escrito en R en Python.

Ilustración : Código del archivo R.py

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

*Fuente: Elaboración propia*

* ExecuteTFG.sh: como se observa en la Ilustración 15 es un fichero ejecutable que se encarga de ejecutar el archivo “*R.py”* y “*app.py*” (ver epígrafe 6.1.1.) en paralelo.

Ilustración 15: Código del archivo ExecuteTFG.sh

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

*Fuente: Elaboración propia*

Por último, se ha implementado un Daemon (proceso demonio) que ejecuta de forma continua el archivo executeTFG.sh. Esto asegura que la aplicación y el algoritmo de recomendación estén siempre en funcionamiento mientras la máquina EC2 esté encendida. Además, cuenta con la siguiente IP elástica: [http://13.37.90.252:5000](http://13.37.90.252:5000/), para poder acceder a la aplicación desde una dirección fija.

## Pruebas

Para probar el funcionamiento del servicio REST se ha utilizado la herramienta *Postman*. Es una plataforma que actúa como un cliente HTTP y te permite realizar peticiones HTTP para poder analizar los resultados obtenidos de cada una de las peticiones. En la Ilustración 16 se muestra un ejemplo en el que se crea al alumno Pablo Sotelo a través de una petición POST:

Ilustración 16: Petición POST en Postman

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

*Fuente: Elaboración propia*

Como se puede observar en la Ilustración 16, hay que pasar los parámetros necesarios del alumno dentro del cuerpo de la petición. En la Ilustración 17 se muestra la respuesta que da el servidor a la petición POST de la Ilustración 16:

Ilustración 17: Respuesta de petición POST en Postman

Interfaz de usuario gráfica, Aplicación, Teams

Descripción generada automáticamente

*Fuente: Elaboración propia*

Observando la Ilustración 17, la API nos devuelve el código 201, lo cual significa que el alumno Pablo Sotelo ha sido creado correctamente.

Adicionalmente se ha creado un banco de pruebas en POSTMAN para realizar pruebas sobre todos los métodos mencionados en el epígrafe 5.1.1.3. Para más información consultar el Anexo B, donde se encuentran adjuntados todos los JSON del banco de pruebas realizado.

# Conclusiones y líneas futuras

Se ha desarrollado una aplicación, denominada "JOBS", que cumple con los objetivos de este TFG. Mediante el diseño e implementación de esta aplicación, se ha logrado resolver los problemas identificados en análisis de dominio en relación con el proceso del CEU de asignación de prácticas y primeros empleos a los alumnos universitarios y recién graduados.

La aplicación “JOBS” permite digitalizar y homogeneizar los CVs de los alumnos y las ofertas de prácticas/empleo proporcionadas por las empresas, y automatiza por completo todos los procesos, incluyendo la recomendación de ofertas a los alumnos.

Las futuras líneas de actuación consistirían en la implantación real de la aplicación “JOBS” en el departamento de carreras profesionales del CEU. Esta implantación permitiría unificar las aplicaciones existentes, mejorando significativamente la eficiencia y la calidad del proceso de asignación, beneficiando a los alumnos, al CEU y a las empresas involucradas.

En conclusión, este TFG ha logrado desarrollar una solución integral y automatizada que transforma y mejora el proceso de asignación de prácticas y primeros empleos en el CEU, brindando beneficios tangibles a todas las partes involucradas.

# Bibliografía

atvise. (2021). *Tipos De Web Service En La Industria 4.0*. Recuperado el 11 de Enero de 2023, de atvise: https://atvise.vesterbusiness.com/news/tipos-web-service-en-la-industria/

AWS. (s.f.). *¿Qué es la arquitectura orientada a servicios?* Recuperado el 6 de Febrero de 2023, de AWS: https://aws.amazon.com/what-is/service-oriented-architecture/

Bagnato, J. (2019). *Sistemas de Recomendación*. Recuperado el 4 de Abril de 2022, de aprendemachinelearning: https://www.aprendemachinelearning.com/sistemas-de-recomendacion/

Cañadas, R. (2021). *Algoritmos de machine learning*. Recuperado el 4 de Abril de 2022, de ABdatum: https://abdatum.com/machine-learning/algoritmos-machine-learning.

Coppola, M. (2023). *Frontend y backend: qué son, en qué se diferencian y ejemplos*. Recuperado el 24 de Abril de 2023, de Hubspot: https://blog.hubspot.es/website/frontend-y-backend#:~:text=Por%20sus%20aplicaciones%20y%20caracter%C3%ADsticas,programación%20de%20sus%20funcionalidades%20principales.

Cortés Pérez, D. (s.f.). *¿Qué son los servicios web?* Recuperado el 15 de Enero de 2023, de CEUPE: https://www.ceupe.com/blog/que-son-los-servicios-web.html

Ekuan, M. (Marzo de 2023). *Diseño de API web RESTful*. Recuperado el 20 de Abril de 2023, de Microsoft: https://learn.microsoft.com/es-es/azure/architecture/best-practices/api-design

Fowler, M. (2010). *Richardson Maturity Model*. Recuperado el 20 de Febrero de 2023, de martinFowler: https://martinfowler.com/articles/richardsonMaturityModel.html

González, A. (s.f.). *Sistemas de recomendación de contenido con Machine Learning*. Recuperado el 25 de Abril de 2022, de cleverdata: https://cleverdata.io/sistemas-recomendacion-machine-learning/

Gupta, L. (2022). *HATEOAS Driven REST APIs*. Recuperado el 20 de Abril de 2023, de restfulapi: https://restfulapi.net/hateoas/

IBM. (2022). *WebSphere Application Server: Overview*. Recuperado el 6 de Febrero de 2023, de IBM: https://www.ibm.com/docs/en/was/9.0.5?topic=websphere-application-server-overview

IBM. (2023). *Definición de recursos en aplicaciones RESTful*. Recuperado el 20 de Abril de 2023, de IBM: https://www.ibm.com/docs/es/was-nd/8.5.5?topic=applications-defining-resources-in-restful

IBM. (2023). *IBM Spectrum Scale Overview*. Recuperado el 6 de Febrero de 2023, de IBM: https://www.ibm.com/docs/en/spectrum-scale?topic=STXKQY/gpfsclustersfaq.html

ictea. (s.f.). *¿Qué es una aplicación web?* Recuperado el 23 de Enero de 2023, de ictea: https://www.ictea.com/cs/index.php?rp=/knowledgebase/4205/iQue-es-una-aplicacion-web.html

Kelleher, J., & Tierney, B. (2021). *Ciencia de Datos.* Ediciones UC.

Lucidchart. (s.f.). Obtenido de Lucidchart: https://www.lucidchart.com/pages/es/simbolos-de-diagramas-entidad-relacion

Mendoza Olguín, G., Laureano De Jesús, Y., & Pérez de Celis Herrero, M. C. (2019). Métricas de similaridad y evaluación para sistemas de recomendación de filtrado colaborativo. *Revista de Investigación en Tecnología de la Información*, 224–240.

Pàmpols, M. (2020). *Arquitectura REST. Qué es y para qué sirve*. Recuperado el 11 de Diciembre de 2022, de marcpampols: https://www.marcpampols.net/es/arquitectura-rest-que-es-para-que-sirve/

Richardson, L., & Ruby, S. (2007). *RESTful Web Services.* O´REILLY.

Risso, I. (2022). *Domina el modelo en cascada y potencia al máximo tus proyectos de software*. Recuperado el 21 de Abril de 2022, de crehana: https://www.crehana.com/blog/transformacion-digital/modelo-en-cascada/

Roman, V. (2019). *Introducción al Machine Learning: Una Guía Desde Cero*. Recuperado el 10 de Abril de 2022, de Medium: https://medium.com/datos-y-ciencia/introduccion-al-machine-learning-una-gu%C3%ADa-desde-cero-b696a2ead359

Sommerville, I. (2005). *Ingeniería del software.* Pearson Educación.

Web Services Architecture Working Group. (2004). *Web Services Architecture.* Recuperado el 10 de Enero de 2023, de W3C: https://www.w3.org/TR/ws-arch/

# Anexo A

# Diseño de las pantallas del *frontend*

*Interfaz de usuario gráfica

Descripción generada automáticamente*

*Ilustración X: Ventana crear nueva oferta trabajo*

*Tabla

Descripción generada automáticamente*

*Ilustración X: Ventana ver alumnos sin ofertas asignadas*

*Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente*

*Ilustración X: Ventana asignar oferta de trabajo*

*Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente*

*Ilustración X: Ventana Modificar Alumno*

*Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente*

*Ilustración X: Ventana Modificar Empresa*

*Tabla

Descripción generada automáticamente*

*Ilustración X: Ventana ver ofertas de una empresa*

# Anexo B

# JSON del banco de pruebas

## Colección Alumnos:

{

"info": {

"\_postman\_id": "fd541ac4-2e5d-49b1-a896-fc994e0a6570",

"name": "Alumnos",

"schema": "https://schema.getpostman.com/json/collection/v2.1.0/collection.json",

"\_exporter\_id": "17512099"

},

"item": [

{

"name": "GET todos los alumnos",

"request": {

"method": "GET",

"header": [],

"url": {

"raw": "13.37.90.252:5000/alumnos",

"host": [

"13",

"37",

"90",

"252"

],

"port": "5000",

"path": [

"alumnos"

]

}

},

"response": []

},

{

"name": "GET alumno by id",

"request": {

"method": "GET",

"header": [],

"url": {

"raw": "13.37.90.252:5000/alumnos/73",

"host": [

"13",

"37",

"90",

"252"

],

"port": "5000",

"path": [

"alumnos",

"73"

]

}

},

"response": []

},

{

"name": "POST alumno",

"request": {

"method": "POST",

"header": [],

"body": {

"mode": "raw",

"raw": "{\n \"username\": \"dsfads\", \n \"password\": \"dsd\", \n \"nombre\": \"Pedro\",\n \"apellido\": \"jijiji\", \n \"telefono\": \"737373738\", \n \"email\": \"jiijij@gmail.com\"\n}",

"options": {

"raw": {

"language": "json"

}

}

},

"url": {

"raw": "13.37.90.252:5000/alumnos",

"host": [

"13",

"37",

"90",

"252"

],

"port": "5000",

"path": [

"alumnos"

]

}

},

"response": []

},

{

"name": "PUT alumno",

"request": {

"method": "PUT",

"header": [],

"body": {

"mode": "raw",

"raw": "{\n \"nombre\": \"xxx\",\n \"apellido\": \"zzz\",\n \"email\": \"xxx@gmail.com\",\n \"telefono\": 666666666\n}",

"options": {

"raw": {

"language": "json"

}

}

},

"url": {

"raw": "13.37.90.252:5000/alumnos/71",

"host": [

"13",

"37",

"90",

"252"

],

"port": "5000",

"path": [

"alumnos",

"71"

]

}

},

"response": []

}

]

}

## Colección Empresas

{

"info": {

"\_postman\_id": "cdd285f4-6825-44a0-9493-e9522546e4bd",

"name": "Empresas",

"schema": "https://schema.getpostman.com/json/collection/v2.1.0/collection.json",

"\_exporter\_id": "17512099"

},

"item": [

{

"name": "GET todas las empresas",

"request": {

"method": "GET",

"header": [],

"url": {

"raw": "13.37.90.252:5000/empresas",

"host": [

"13",

"37",

"90",

"252"

],

"port": "5000",

"path": [

"empresas"

]

}

},

"response": []

},

{

"name": "GET empresa by id",

"request": {

"method": "GET",

"header": [],

"url": {

"raw": "13.37.90.252:5000/empresas/31",

"host": [

"13",

"37",

"90",

"252"

],

"port": "5000",

"path": [

"empresas",

"31"

]

}

},

"response": []

},

{

"name": "POST empresa",

"request": {

"method": "POST",

"header": [],

"body": {

"mode": "raw",

"raw": "{\n \"username\": \"Holaquetal\", \n \"password\": \"Holaquetal\",\n \"empresa\_nombre\": \"Holaquetal\", \n \"telefono\": \"666666666\", \n \"email\": \"carlo22@gmail.com\"\n}",

"options": {

"raw": {

"language": "json"

}

}

},

"url": {

"raw": "13.37.90.252:5000/empresas",

"host": [

"13",

"37",

"90",

"252"

],

"port": "5000",

"path": [

"empresas"

]

}

},

"response": []

},

{

"name": "PUT empresa",

"request": {

"method": "PUT",

"header": [],

"body": {

"mode": "raw",

"raw": "{\n \"empresa\_nombre\": \"Holaquetal\", \n \"telefono\": \"666666666\", \n \"email\": \"carlo22@gmail.com\"\n}",

"options": {

"raw": {

"language": "json"

}

}

},

"url": {

"raw": "13.37.90.252:5000/empresas/32",

"host": [

"13",

"37",

"90",

"252"

],

"port": "5000",

"path": [

"empresas",

"32"

]

}

},

"response": []

}

]

}

## Colección CV

{

"info": {

"\_postman\_id": "91d71c74-5d68-421d-b75b-524ab9362f07",

"name": "CV",

"schema": "https://schema.getpostman.com/json/collection/v2.1.0/collection.json",

"\_exporter\_id": "17512099"

},

"item": [

{

"name": "GET CV by id",

"request": {

"method": "GET",

"header": [],

"url": {

"raw": "13.37.90.252:5000/alumnos/71/CV",

"host": [

"13",

"37",

"90",

"252"

],

"port": "5000",

"path": [

"alumnos",

"71",

"CV"

]

}

},

"response": []

},

{

"name": "POST CV para un usuario",

"request": {

"method": "POST",

"header": [],

"body": {

"mode": "raw",

"raw": "{\n \"grado\": 2, \n \"nota\_media\": 5,\n \"ingles\": 1, \n \"aleman\": 1, \n \"frances\": 0, \n \"capacidad\_analitica\": 2,\n \"trabajo\_equipo\": 3, \n \"comunicacion\": 1, \n \"pensamiento\_critico\": 3,\n \"inovacion\": 0, \n \"liderazgo\": 0, \n \"decision\_making\": 1,\n \"problem\_solving\": 1, \n \"marketing\": 1, \n \"e\_commerce\": 1, \n \"diseno\_grafico\": 0,\n \"matematicas\": 2, \n \"estadistica\": 2, \n \"gestion\_proyectos\": 0, \n \"redes\_sociales\": 0,\n \"sostenibilidad\": 1, \n \"inteligencia\_artificial\": 1, \n \"big\_data\": 0, \n \"machine\_learning\": 1,\n \"analisis\_datos\": 0, \n \"bases\_datos\": 1, \n \"cloud\": 0, \n \"intenet\_of\_things\": 0,\n \"networks\": 0, \n \"sistemas\_operativos\": 0, \n \"web\_desarrollo\": 2,\n \"web\_diseno\": 1, \n \"r\": 1, \n \"java\": 0, \n \"pascal\": 0, \n \"python\": 0 \n}",

"options": {

"raw": {

"language": "json"

}

}

},

"url": {

"raw": "13.37.90.252:5000/alumnos/71/CV",

"host": [

"13",

"37",

"90",

"252"

],

"port": "5000",

"path": [

"alumnos",

"71",

"CV"

]

}

},

"response": []

},

{

"name": "PUT CV para un usuario",

"request": {

"method": "PUT",

"header": [],

"body": {

"mode": "raw",

"raw": "{\n \"grado\": 0, \n \"nota\_media\": 8,\n \"ingles\": 2, \n \"aleman\": 1, \n \"frances\": 0, \n \"capacidad\_analitica\": 3,\n \"trabajo\_equipo\": 2, \n \"comunicacion\": 1, \n \"pensamiento\_critico\": 0,\n \"inovacion\": 2, \n \"liderazgo\": 0, \n \"decision\_making\": 1,\n \"problem\_solving\": 2, \n \"marketing\": 3, \n \"e\_commerce\": 2, \n \"diseno\_grafico\": 1,\n \"matematicas\": 3, \n \"estadistica\": 2, \n \"gestion\_proyectos\": 0, \n \"redes\_sociales\": 2,\n \"sostenibilidad\": 2, \n \"inteligencia\_artificial\": 1, \n \"big\_data\": 3, \n \"machine\_learning\": 1,\n \"analisis\_datos\": 0, \n \"bases\_datos\": 3, \n \"cloud\": 1, \n \"intenet\_of\_things\": 3,\n \"networks\": 2, \n \"sistemas\_operativos\": 3, \n \"web\_desarrollo\": 1,\n \"web\_diseno\": 1, \n \"r\": 3, \n \"java\": 2, \n \"pascal\": 2, \n \"python\": 2 \n}",

"options": {

"raw": {

"language": "json"

}

}

},

"url": {

"raw": "13.37.90.252:5000/alumnos/71/CV",

"host": [

"13",

"37",

"90",

"252"

],

"port": "5000",

"path": [

"alumnos",

"71",

"CV"

]

}

},

"response": []

},

{

"name": "GET CV de alumnos con oferta asignada",

"request": {

"method": "GET",

"header": [],

"url": {

"raw": "13.37.90.252:5000/ofertas/cvs?estado=ASIGNADA",

"host": [

"13",

"37",

"90",

"252"

],

"port": "5000",

"path": [

"ofertas",

"cvs"

],

"query": [

{

"key": "estado",

"value": "ASIGNADA"

}

]

}

},

"response": []

}

]

}

Colección Ofertas

{

"info": {

"\_postman\_id": "a3435433-765d-4177-a5ab-20d16edb21d4",

"name": "Ofertas",

"schema": "https://schema.getpostman.com/json/collection/v2.1.0/collection.json",

"\_exporter\_id": "17512099"

},

"item": [

{

"name": "GET ofertas by emp id",

"protocolProfileBehavior": {

"disableBodyPruning": true

},

"request": {

"method": "GET",

"header": [],

"body": {

"mode": "raw",

"raw": "",

"options": {

"raw": {

"language": "json"

}

}

},

"url": {

"raw": "13.37.90.252:5000/empresas/1/ofertas",

"host": [

"13",

"37",

"90",

"252"

],

"port": "5000",

"path": [

"empresas",

"1",

"ofertas"

]

}

},

"response": []

},

{

"name": "POST ofertas",

"request": {

"method": "POST",

"header": [],

"body": {

"mode": "raw",

"raw": "{ \n \"job\_title\": \"xxxx\", \n \"ciudad\": \"Madrid\", \n \"grado\": 3, \n \"nota\_media\": 6,\n \"ingles\": 2, \n \"aleman\": 2, \n \"frances\": 2,\n \"trabajo\_equipo\": 2, \n \"comunicacion\": 2, \n \"matematicas\": 2,\n \"estadistica\": 2, \n \"gestion\_proyectos\": 2, \n \"sostenibilidad\": 2,\n \"big\_data\": 2, \n \"programacion\": 2,\n \"telefono\": 676567556,\n \"nombre\_contacto\":\"pedro\"\n}",

"options": {

"raw": {

"language": "json"

}

}

},

"url": {

"raw": "13.37.90.252:5000/empresas/1/ofertas",

"host": [

"13",

"37",

"90",

"252"

],

"port": "5000",

"path": [

"empresas",

"1",

"ofertas"

]

}

},

"response": []

},

{

"name": "GET oferta by emp id and id",

"request": {

"method": "GET",

"header": [],

"url": {

"raw": "13.37.90.252:5000/empresas/1/ofertas/82",

"host": [

"13",

"37",

"90",

"252"

],

"port": "5000",

"path": [

"empresas",

"1",

"ofertas",

"82"

]

}

},

"response": []

},

{

"name": "PUT oferta",

"request": {

"method": "PUT",

"header": [],

"body": {

"mode": "raw",

"raw": "{\n \"alumno\_id\": 73,\n \"estado\":\"ASIGNADA\"\n}",

"options": {

"raw": {

"language": "json"

}

}

},

"url": {

"raw": "13.37.90.252:5000/empresas/1/ofertas/71",

"host": [

"13",

"37",

"90",

"252"

],

"port": "5000",

"path": [

"empresas",

"1",

"ofertas",

"71"

]

}

},

"response": []

},

{

"name": "GET ofertas sin asignar",

"request": {

"method": "GET",

"header": [],

"url": {

"raw": "13.37.90.252:5000/ofertas?estado=SIN ASIGNAR",

"host": [

"13",

"37",

"90",

"252"

],

"port": "5000",

"path": [

"ofertas"

],

"query": [

{

"key": "estado",

"value": "SIN ASIGNAR"

}

]

}

},

"response": []

},

{

"name": "GET oferta asignada de un alumno by id alum",

"request": {

"method": "GET",

"header": [],

"url": {

"raw": "13.37.90.252:5000/ofertas?alumno=72",

"host": [

"13",

"37",

"90",

"252"

],

"port": "5000",

"path": [

"ofertas"

],

"query": [

{

"key": "alumno",

"value": "72"

}

]

}

},

"response": []

},

{

"name": "get ofertas by id",

"request": {

"method": "GET",

"header": [],

"url": {

"raw": "13.37.90.252:5000/ofertas/2",

"host": [

"13",

"37",

"90",

"252"

],

"port": "5000",

"path": [

"ofertas",

"2"

]

}

},

"response": []

}

]

}