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 2022

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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

Texto con, a lo sumo, 200 palabras.

Palabras Clave

Abstract

Resumen anterior, en inglés.

Keywords

Índice de contenidos

[Capítulo 1 Introducción 1](#_Toc106906157)

[Capítulo 2 Estado del arte 5](#_Toc106906158)

[2.1 Servicios Web 5](#_Toc106906159)

[2.2 Algoritmo de recomendación 8](#_Toc106906160)

[2.2.1 Introducción a la ciencia de los datos [1] 8](#_Toc106906161)

[2.2.2 Machine Learning: métodos de aprendizaje [2] 8](#_Toc106906162)

[2.2.3 Técnicas de análisis de datos prescriptivas: sistemas de recomendación [4] 11](#_Toc106906163)

[2.2.4 Medidas de similitud o distancia [6] 13](#_Toc106906164)

[2.2.5 Lenguaje utilizado: R 14](#_Toc106906165)

[Capítulo 3 Gestión del Proyecto 15](#_Toc106906166)

[3.1 Modelo de ciclo de vida 15](#_Toc106906167)

[3.2 Papeles desempeñados en el proyecto 16](#_Toc106906168)

[3.3 Planificación 16](#_Toc106906169)

[3.4 Presupuesto 18](#_Toc106906170)

[3.5 Ejecución 18](#_Toc106906171)

[Capítulo 4 Análisis y arquitectura 19](#_Toc106906172)

[4.1 Análisis de dominio 19](#_Toc106906173)

[4.2 Especificación de requisitos 19](#_Toc106906174)

[4.3 Análisis de los casos de uso y de las clases de análisis 23](#_Toc106906175)

[4.4 Análisis de seguridad 24](#_Toc106906176)

[4.5 Análisis desde la perspectiva del RGPD (si procede) 24](#_Toc106906177)

[4.6 Arquitectura del sistema 24](#_Toc106906178)

[Capítulo 5 Diseño 25](#_Toc106906179)

[5.1 Diseño del Backend 25](#_Toc106906180)

[5.1.1 Diseño del Servicio Web RESTful 25](#_Toc106906181)

[5.1.2 Diseño del algoritmo de recomendación 39](#_Toc106906182)

[5.1.3 Diseño de la base de datos 39](#_Toc106906183)

[5.2 Diseño del Frontend 45](#_Toc106906184)

[5.2.1 Diseño de la interfaz de usuario 45](#_Toc106906185)

[5.3 Modelo de clases de diseño 55](#_Toc106906186)

[5.4 Entorno de construcción 55](#_Toc106906187)

[5.5 Plan de pruebas 55](#_Toc106906188)

[5.6 Diagrama de infraestructuras de nivel 3 55](#_Toc106906189)

[5.7 Diagrama de infraestructuras de nivel 2 (si procede) 56](#_Toc106906190)

[Capítulo 6 Implementación 57](#_Toc106906191)

[6.1 Implementación Backend 57](#_Toc106906192)

[6.1.1 Implementación Servicio REST 57](#_Toc106906193)

[6.1.2 Implementación Algoritmo recomendación 59](#_Toc106906194)

[6.1.3 Implementación Base de datos 59](#_Toc106906195)

[6.2 Implementación Frontend 59](#_Toc106906196)

[6.2.1 XXX 60](#_Toc106906197)

[6.2.2 XXX 66](#_Toc106906198)

[Capítulo 7 Construcción 67](#_Toc106906199)

[7.1 Referencia al repositorio de software 67](#_Toc106906200)

[7.2 Manuales 67](#_Toc106906201)

[Capítulo 8 Algoritmo de Recomendación 68](#_Toc106906202)

[8.1 Sistema de recomendación de JOBS 68](#_Toc106906203)

[8.1.1 Estructura del sistema de recomendación 68](#_Toc106906204)

[8.1.2 API “ofertas\_nuevas.R” 69](#_Toc106906205)

[8.1.3 API “alumno\_nuevo.R” 70](#_Toc106906206)

[8.1.4 Llamada de las APIs 71](#_Toc106906207)

[8.1.5 Intercambio de ficheros Web-APIs 71](#_Toc106906208)

[Capítulo 9 Conclusiones y líneas futuras 73](#_Toc106906209)

[Referencias 75](#_Toc106906210)

[Anexo I 76](#_Toc106906211)

[Otras posibilidades para realizar el análisis y el diseño 76](#_Toc106906212)

[Anexo II 77](#_Toc106906213)

[Fuente de inspiración del presente documento 77](#_Toc106906214)

[Glosario de términos 79](#_Toc106906215)

[Anexos 81](#_Toc106906216)

Índice de ilustraciones

Ilustración 1

Ilustración 2

Índice de tablas

Tabla 1

Tabla 2

# Introducción

Se explica en qué consiste el proyecto y se enumeran los capítulos. Debe incluir sus objetivos.

# Estado del arte

En este capítulo se van a identificar el estado del arte de los Servicios Web y de los Algoritmos de recomendación.

## Servicios Web

Los servicios web son una tecnología que facilita un servicio a través de internet. 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:

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

Estos servicios con una arquitectura cliente/servidor, en la que se puede identificar al servidor como las maquinas que almacenan la información y los recursos, y al cliente cualquier dispositivo que se conecte.

Existen dos tipos de servicios web:

* SOAP (Simple Object Access Protocol): Es un protocolo estándar de mensajes basado en XML que permite la comunicación para intercambiar información entre diferentes máquinas. Al ser un protocolo, cuenta con reglas que aumentan la complejidad y sobrecarga, lo cual perjudica el tiempo que tardan las páginas en cargarse.

El envio de la solicitud de datos se basa en HTTP, mientras que los mensajes SOAP que se reciben necesitan ser únicamente documentos XML.

* REST (Representacional State Transfer): Es un estilo de arquitectura software que facilita el intercambio de recursos basados en HTTP. Representa un conjunto de principios que permiten crear, leer, actualizar y eliminar los datos.

Una de las principales características de REST es que es *stateless*, es decir, no almacena los estados, debe de ser el cliente el que introduce el estado en cada llamada realizada.

El modelo de Richardson propone una clasificación de los servicios web mediante el nivel de madurez de cada uno:

* + Nivel 0: The Swamp of POX

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

Escala de tiempo

Descripción generada automáticamente con confianza baja

*Ilustración 1: Nivel 0 del Modelo Madurez Richardson*

* + Nivel 1: Resources

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

Gráfico

Descripción generada automáticamente con confianza baja

*Ilustración 2: Nivel 1 del Modelo Madurez Richardson*

* + Nivel 2: HTTP Verbs

Distingue entre diferentes recursos y diferentes métodos HTTP.

Imagen que contiene Escala de tiempo

Descripción generada automáticamente

*Ilustración 3: Nivel 2 del Modelo Madurez Richardson*

* + Nivel 3: Hypermedia Controls

Hace uso de HATEOAS (Hypermedia As The Engine Of Application State). Al hacer un request nos devolverá información adicional (link) sobre otro recurso.

Escala de tiempo

Descripción generada automáticamente

*Ilustración 4: Nivel 3 del Modelo Madurez Richardson*

## Algoritmo de recomendación

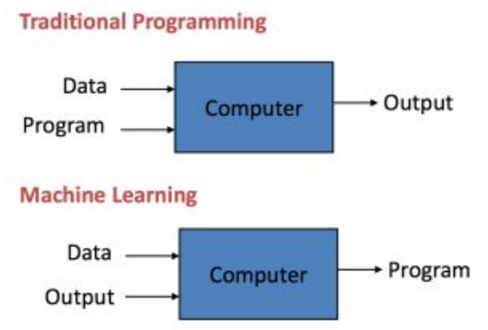
### Introducción a la ciencia de los datos

La automatización de tareas y la capacidad de extraer valor de los datos es el presente y el futuro cercano de cualquier tipo de actividad/función/negocio que implique análisis y toma de decisiones. Otras eras de la historia de la humanidad se han caracterizado por manipular materias primas físicas: cobre, hierro, bronce, etc. Nosotros nos ha tocado vivir la “era de los datos”, y es el dato el nuevo material del que tenemos que extraer valor.

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.

### Machine Learning: métodos de aprendizaje

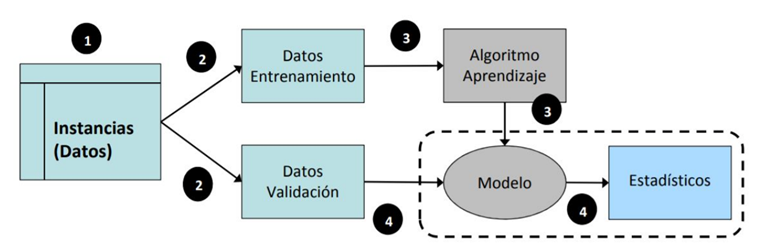
Machine Learning son un conjunto de métodos capaces de detectar automáticamente patrones en los datos con el objetivo de definir modelos que permitan representar de una forma simplificada la realidad. Machine Learning implica definir algoritmos que se encarga de que el modelo aprenda de los datos (se ajuste a los datos), es decir, el algoritmo se encarga de crear una función matemática para representar la realidad a partir de los datos. La diferencia fundamental entre la programación tradicional y la de Machine Learning radica en los elementos de entrada y de salida que tiene un modelo y el otro:



*Ilustración 5: Diferencia entre la programación tradicional y el ML*

Los métodos de aprendizaje [2] implican la consecución de una serie de pasos:

* Los datos iniciales son preprocesados, se presentan al sistema de aprendizaje.
* Los datos se dividen en dos grupos. Los datos de entrenamiento servirán para generar el modelo. Los datos de validación servirán para valorar la calidad del modelo.
* El algoritmo de aprendizaje genera el patrón que mejor se adecua a los datos.
* Se valora la calidad del modelo utilizando los datos validación, para lo cual se utilizan diferentes estadísticos/métricas.



*Ilustración 6: Pasos de un modelo de aprendizaje*

Las técnicas y algoritmos de Machine Learning analizan datos con dos objetivos:

* Predecir: intentan responder a la pregunta, ¿Qué podría ocurrir? Implica ser capaz de predecir qué valores de salida se darán para unos valores de entrada determinados.
* Informar: intentan responder a la pregunta, ¿Qué ha ocurrido? Implica extraer información acerca de cómo el modelo, para unas variables de entrada, produce unas variables de salida.

En función del objetivo que persiguen hay dos familias de métodos de aprendizaje:

* 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) o 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 que nos permiten estimar lo que no conocemos y las descriptivas que nos permiten explorar, analizar y entender mejor los datos. 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.

La gran cantidad de sitios especializados en Internet, ofertando millones de productos y/o servicios para su consumo, se ha convertido en un caos de información. Los sistemas de recomendación surgen como solución a este problema. Un sistema de recomendación recibe información del usuario acerca de productos y/o servicios en los que el usuario se encuentra interesado y le recomienda aquéllos cercanos a sus necesidades. Los sistemas de recomendación utilizan métodos matemáticos y estadísticos para explorar los datos y crear recomendaciones adaptadas a cada usuario. Los algoritmos de recomendación son utilizados por Netflix para remendarnos películas, por Spotify para recomendarnos canciones y por Amazon para recomendarnos productos.

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:

* Filtrado colaborativo [5]: 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. Se utilizan las valoraciones que el propio usuario A ha hecho de los ítems similares como predicción de su valoración sobre el ítem X. Este sistema puede parecer similar al basado en contenido, la diferencia se encuentra en que cada ítem no está definido por sus atributos sino por el perfil de valoraciones que ha recibido.

He optado para diseñar el sistema de recomendación de Jobs por un enfoque de filtrado colaborativo. Mi objetivo es recomendar a los usuarios de Jobs ofertas de trabajo o puestos en prácticas a las que son más afines en función de su perfil, es decir, su CV es afín a los requerimientos de la oferta.

### Medidas de similitud o distancia

Todos los sistemas de recomendación tienen una cosa en común, para poder llevar a cabo las predicciones, necesitan definir y cuantificar la similitud entre ítems o usuarios.

Los sistemas de recomendación utilizan el término distancia como cuantificación de la similitud o diferencia entre observaciones. Si se representan las observaciones en un espacio p dimensional, siendo p el número de variables asociadas a cada observación (ítem o usuario), cuando más se asemejen dos observaciones, más próximas estarán, de ahí que se emplee el término distancia. Hay diferentes medidas de similitud:

* Correlación de Pearson: se emplea para calcular la correlación lineal entre dos vectores. Es la medida más apropiada si hay datos nulos.
* Correlación restringida de Pearson: utilizada valores medianos en lugar de medias de rating de dos usuarios.
* Coseno: es una de las medidas de similitud más utilizada. Trata de encontrar cómo dos vectores están relacionados entre ellos midiendo el coseno del ángulo entre los dos vectores. Considera los datos nulos cómo válidos.
* Jaccard: considera el número de preferencias compartidas entre dos usuarios y asume que dos usuarios serán más parecidos cuando tengan valores informados (ratings informados) similares.
* Diferencias de medias cuadradas: considera los ratings absolutos, en lugar de qué ítems han sido valorados. Normalmente se suelen combinar Jaccard.
* PIP: Proximity calcula la diferencia aritmética entre dos ratings. Impact calcula si un ítem le gusta o no al usuario. Popularity calcula la media aritmética del rating de ítems del usuario.
* Correlación de Spearman: Considera el ranking en lugar del rating para calcular la similitud.
* Correlación Tau de Kendall: considera los rankings relativos en lugar del rating para calcular la similitud.

Para el sistema de recomendación de Jobs en optado por utilizar el coseno como medida de similitud. 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.

### Lenguaje utilizado: R

De entre los diferentes lenguajes de programación que se pueden utilizar para el análisis de datos he decidido utilizar R para programar el algoritmo de selección.

He elegido R ya que he considerado que presenta ventajas significativas: excelente gama de paquetes de código abierto y de alta calidad, dispone funciones y métodos estadísticos integrales muy completos y permite una muy buena visualización de datos con el uso de bibliotecas como ggplot2. También he 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.

# Gestión del Proyecto

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

## Modelo de ciclo de vida

El ciclo de vida de un software es una secuencia estructurada y bien definida de las etapas necesarias para desarrollar el software deseado. Para el desarrollo del proyecto se ha decidido utilizar un Modelo de Cascada ya que es una metodología simple y las actividades (requisitos) están divididas en fases secuenciales contando con bucles por si es necesario volver hacia atrás para modificar/mejorar una fase. Esta metodología fue propuesta por Winston W. Royce en 1970, también conocido como el modelo lineal o modelo de ciclo de vida. Dicho modelo se divide en las siguientes fases:

* Requisitos: realizar una lista detallada de tareas, gráficos de dependencia, establecer hitos, ir mejorando los planes a medida que vamos sabiendo más, gráficos de GANTT y PERT.
* Análisis: consiste en la especificación del producto, describir el sistema desde la perspectiva del usuario, descripción detallada de los datos de entrada/salida, preparación del proyecto, ver las necesidades y objetivos (análisis del dominio).
* Diseño: está dividida en el diseño de la arquitectura (definición de los grandes bloques funcionales de la aplicación y su comunicación, además de elegir el estilo arquitectónico) y diseño detallado (diseño detallado de cada uno de los componentes introducidos en la fase de diseño de la arquitectura).
* Implementación: Traducción de toda la fase de diseño.
* Verificación: realización de pruebas (unitarias, integración, …) para comprobar el correcto funcionamiento del software.
* Mantenimiento: Puesta en producción del software y mantenimiento de este.

## Papeles desempeñados en el proyecto

Los dos tutores de este TFG, Sergio Saugar García y Alex Rayón Jerez, son los promotores y propietarios de este proyecto y el alumno, Carlos Soler Martín es el director del proyecto, siendo también el analista-programador que a lo largo de esta memoria se explicará el diseñado y desarrollo que se ha llevado a cabo para la aplicación JOBS.

## Planificación

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 de la arquitectura.
* Pruebas y validación del proyecto.
* Documentar el proyecto.

El diagrama Gantt muestra el tiempo estimado que llevara para realización de cada una de las tareas que se han mencionado, siendo el inicio el 14 de enero de 2022 y la finalización el 22 de junio de 2022. Este diagrama como se ha comentado se ha realizado.

Gráfico

Descripción generada automáticamente con confianza media

*Ilustración 6: Diagrama Gantt de la planificación del proyecto*

Se identificarán las tareas a realizar, su descripción, sus fechas estimadas de inicio y de fin, sus precedencias, los recursos que requieren, etc. Se podrán utilizar diagramas de Gantt, PERT / CPM, etc. Deberán incluirse también los diferentes hitos a lo largo del proyecto. Para que el estudiante tenga la experiencia real de planificar y controlar su cumplimiento, es **importante** que la planificación se realice con anterioridad a la finalización del proyecto.

Se pueden encontrar explicaciones tanto sobre los diagramas que se indican en esta sección como los que se mencionan en los siguientes apartados en la [documentación sobre técnicas de la metodología Métrica 3](https://administracionelectronica.gob.es/pae_Home/dam/jcr:da7d91fa-d6bd-467c-be32-a72e27c603b3/METRICA_V3_Tecnicas.pdf).

## Presupuesto

El desarrollo del proyecto de este TFG no ha supuesto incurrir en ningún coste, ya que el software utilizado no ha implicado pagar ninguna licencia y todas las horas de desarrollo han sido asumidas por el alumno. Como el alumno cursa el doble grado de ISI y ADE, su intención es continuar con la elaboración de este proyecto desde un punto de vista empresarial, con objeto de realizar una propuesta de emprendimiento empresarial donde el proyecto software pueda ser la semilla de una futura empresa emergente.

## Ejecución

Se mostrará cómo ha transcurrido realmente el proyecto. Se podrán utilizar los mismos diagramas que en apartado anterior (diagramas de Gantt, PERT / CPM, etc.).

Realizar otro GANTT con timepos mayores

# Análisis y arquitectura

## Análisis de dominio

La función principal de la plataforma Jobs es simplificar, eficientar y automatizar el proceso de asignación de prácticas y primeros empleos a los estudiantes universitarios y recién graduados. He analizado cómo funciona actualmente en la Universidad CEU San Pablo (en adelante CEU) este proceso con el fin de identificar los problemas y buscar soluciones.

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

Actualmente el CEU cuentan con 5 aplicaciones para la búsqueda de trabajo 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. Una vez realizado lo anterior, el estudiante/recién graduado debe de buscar de manera manual las empresas a las que le gustaría aplicar, una vez encuentra una empresa de trabajo que le interesa debe de enviar a través de la aplicación una pequeña carta de motivación o a veces la empresa te redirige a su propia página web en la sección de recruiting. Por parte de las empresas también deben de registrarse en la aplicación y publicar su oferta de trabajo, o normalmente el equipo de carreras profesionales reciben ofertas de las empresas por email y 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.

Adicionalmente, existe una aplicación para la búsqueda de prácticas curriculares dentro del campus virtual. Las empresas envían una ficha con las características de las prácticas al equipo de carreras profesionales. Si al estudiante le interesa realiza una solicitud e incluye su CV que le llega al equipo de carreras profesionales que analiza si el CV se adapta a lo requerido por la empresa y si es así se lo mandan a la empresa, todo el proceso es 100% manual. Esta aplicación también permite firmar el anexo de las practicas, realizar una memoria sobre las practicas realizadas, realizar un seguimiento del trámite de las prácticas y realizar una encuesta.

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 y funcionan de forma 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 matching entre CVs y ofertas.

La solución que se propone en este TFG permite unificar las aplicaciones anteriormente mencionadas en una única aplicación, digitalizar y homogeneizar los CVs y las ofertas de prácticas/empleo de una forma muy eficiente (tiempo de imputación 3-5 minutos), automatizar el proceso de matching entre CVs y ofertas y simplificar los procesos del equipo de carreras profesionales del CEU.

## Especificación de requisitos

En un escenario habitual, se deberán especificar, al menos, los siguientes tipos de requisitos:

* *Funcionalidades del sistema,* que podrán detallarse, por ejemplo, mediante diagramas de casos de uso con su especificación. Alternativamente se pueden mostrar historias de usuario.
* *Interfaz de usuario* (si procede). Puede incluirse, por ejemplo, el nivel de cumplimiento de la norma de accesibilidad [WCAG](https://www.w3.org/WAI/standards-guidelines/wcag/).
* *Rendimiento*, incluyendo tiempos de respuesta.
* *Capacidad*, incluyendo el número máximo de peticiones que se pueden procesar por unidad de tiempo.
* *Seguridad*. Se indicará el nivel de seguridad que ha de satisfacer el sistema, por ejemplo, según el Esquema Nacional de Seguridad (RD 3/2010) o según alguna norma de la familia ISO 27000. Este nivel de seguridad se establecerá a partir de un análisis de riesgos (véase la sección de análisis de seguridad).
* *Interoperabilidad con otros sistemas* (si procede).
* *Protección de datos* (si procede), en concreto, el cumplimiento del Reglamento General de Protección de Datos ([UE] 2016/679) y de la Ley Orgánica 3/2018 de Protección de Datos Personales y Garantía de los Derechos Digitales (véase la sección de análisis desde la perspectiva del RGPD).
* *Requisitos sobre entorno tecnológico y de comunicaciones*, es decir, requisitos sobre gestores de bases de datos, sistema o los sistemas operativos, hardware, protocolos de red, etc.

Al comienzo del desarrollo del proyecto se definieron una serie de requisitos que para facilitar el desarrollo de la aplicación:

Requisitos funcionales

* Los alumnos pueden registrarse, hacer login en la aplicación y cerrar sesión.
* Los alumnos pueden añadir su “CV” (skills y capacidades), que posteriormente podrá ser modificado por dicho alumno si lo desea.
* Los alumnos pueden modificar su perfil.
* Los alumnos pueden consultar su CV.
* Los alumnos pueden ver las ofertas no asignadas de empresas registradas en la aplicación.
* Los alumnos pueden solicitar una recomendación de las ofertas no asignadas cuyos requisitos se asimilen más a su CV.
* Las empresas pueden registrarse, hacer login en la aplicación y cerrar sesión.
* Las empresas pueden modificar su perfil.
* Las empresas pueden crear nuevas ofertas, que posteriormente podrá ser modificado por dicha empresa si lo desea.
* Las empresas pueden consultar los CV de los alumnos sin ofertas asignadas.
* Las empresas pueden ver todas las ofertas que han creado.
* El administrador puede asignar ofertas no asignadas a alumnos sin oferta asignada.

Requisitos no funcionales:

* Mejora de rendimiento.
* Mayor escalabilidad. La API puede responder a otro tipo de operaciones o realizar nuevas versiones tanto como se desee.
* Mayor flexibilidad, ya que se puede se pueden alojar en diferentes servidores las páginas del front y la API.
* Se requieren menos recursos del servidor, ya que no se mantiene el estado de la petición y por tanto no requiere memoria.

**Diagrama

Descripción generada automáticamente**

*Ilustración 7: Diagrama de la casos de uso*

## Análisis de los casos de uso y de las clases de análisis

A partir de los casos de uso, se obtendrá el modelo de clases de análisis, así como la realización de los casos de uso (diagramas de interacción entre objetos). En el diagrama de clases, para cada una de ellas, se identificarán sus atributos, responsabilidades (funcionalidades), asociaciones, agregaciones y generalizaciones. El comportamiento de las clases podrá mostrarse mediante diagramas de transición de estados.

Los diagramas elaborados no estarán condicionados por la tecnología utilizada, sino que estarán centrados en el problema en sí a resolver.

Xxx

## Análisis de seguridad

Se llevará a cabo un análisis de riesgos de acuerdo con las dimensiones de autenticidad, confidencialidad, integridad, disponibilidad y trazabilidad.

Xxx

## Análisis desde la perspectiva del RGPD (si procede)

En caso de que sea necesario, se llevará a cabo una gestión del riesgo y evaluación de impacto en tratamientos de datos personales.

Xxx

## Arquitectura del sistema

La arquitectura propuesta para la aplicación es una arquitectura REST. Se trata de una arquitectura cliente-servidor, en la que el cliente es el responsable de realizar las peticiones mediante el protocolo HTTP y el servidor responde con códigos de estado y la información solicitada.

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

* Fontend: 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 encargado de manipular y analizar los datos de entrada para devolver las recomendaciones solicitadas por un alumno y, por último, el núcleo del backend encargado de procesar los datos y conectar todos los componentes de la arquitectura.

Para facilitar el entendimiento de la arquitectura del sistema se ha elaborado un ejemplo: Pablo Sotelo, recién graduado en la Universidad CEU San Pablo, está buscando su primer empleo, para ello accede con su cuenta a la aplicación JOBS para ver que empresas le recomiendan en función de su CV. Cuando Pablo accede a la ventana donde se encuentra el recomendador y solicita recomendación, el núcleo del backend envía una petición al sistema de recomendación para que calcule las ofertas que más se ajustan a Pablo en función de su CV. Una vez el sistema de recomendación tenga una respuesta devuelve al núcleo del backend que a su vez se lo envía al frontend para que le muestre las ofertas que le recomienda el algoritmo de manera más visual.

Diagrama

Descripción generada automáticamente

*Ilustración 8: Arquitectura del sistema*

# Diseño

## Diseño del Backend

### Diseño del Servicio Web RESTful

#### Definición de un recurso

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

* Alumno: El recurso Alumno representa a aquellos usuarios, que, desempeñando el rol de Alumno, se conectan a la aplicación. Además, un alumno está asociado a un CV, se le pueden asignar una única oferta y puede modificar sus atributos. Los atributos del alumno son:
  + Nombre: Nombre completo, incluyendo el segundo nombre en caso de que existiera.
  + Apellido: Primer apellido.
  + Correo electrónico: Correo electrónico del alumno.
  + Número de teléfono: Número de teléfono de contacto del alumno
  + Nombre de usuario: Nombre de usuario para identificar al alumno.
  + Contraseña: Contraseña para verificar al alumno.
* Alumnos: Permite listar todos los alumnos registrados en la aplicación y también permite registrar nuevos alumnos.
* Empresa: El recurso Empresa representa a aquellas empresas, que, desempeñando el rol de Empresa, se conectan a la aplicación. Además, una empresa puede crear ofertas de trabajo, modificar sus ofertas de trabajo y modificar sus atributos. Los atributos de la empresa son:
  + Nombre de la empresa: Nombre completo de la empresa.
  + Correo electrónico: Correo electrónico de la empresa.
  + Número de teléfono: Número de teléfono de contacto de la empresa.
  + Nombre de usuario: Nombre de usuario para identificar a la empresa.
  + Contraseña: Contraseña para verificar a la empresa.
* Empresas: Permite listar todas las empresas registradas en la aplicación y también permite registrar una nueva empresa.
* Administrador: El recurso Administrador representa a aquellos administradores, que, desempeñando el rol de Administrador, se conectan a la aplicación. Además, un administrador puede asignar ofertas de trabajo a alumnos. Los atributos del administrador son:
  + Nombre de usuario: Nombre de usuario para identificar al administrador.
  + Contraseña: Contraseña para verificar al administrador.
* Administradores: Permite listar todos los administradores registrados en la aplicación y también permite registrar un nuevo administrador.
* Oferta: El recurso Oferta representa a aquellas ofertas, que, las crean las empresas y se les asignan a los alumnos. Los atributos de la oferta son:
  + Nombre de la empresa: Nombre de la empresa que realiza la oferta.
  + Título de la oferta: Titulo y descripción de la oferta, ej: Analista Jr. data science.
  + Alumno: alumno asociado a esa oferta de trabajo (en el caso de que la oferta esté asignada a un alumno).
  + Ciudad: Ciudad donde se realizará el trabajo.
  + Teléfono de contacto: Teléfono de contacto de la persona responsable de la oferta.
  + Nombre de contacto: Nombre de contacto de la persona responsable de la oferta.
  + Grado: Grado requerido para optar por la oferta de trabajo. Los grados disponibles son: Ingeniería Informática, Ingeniería Industrial, Ingeniería de Telecomunicaciones, Data scientist, Matemáticas, Física, ADE, Derecho, Marketing y Sociología.
  + Nota media: Nota media mínima para optar por la oferta de trabajo.
  + Niveles: Las ofertas requieren ciertos niveles de idiomas, de soft-skills, de 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.
  + Estado: Indica si la oferta está asignada o no.
* Ofertas: Permite listar todas las ofertas y también permite registrar una nueva oferta.
* CV: El recurso CV representa los Curriculum Vitae creados por cada alumno, dicho CV se crea por el propio alumno y también puede ser modificado. Los atributos del CV son:
  + Alumno: Alumno asociado al CV.
  + Grado: Grado cursado/estudiado por el alumno. Los grados disponibles son: Ingeniería Informática, Ingeniería Industrial, Ingeniería de Telecomunicaciones, Data scientist, Matemáticas, Física, ADE, Derecho, Marketing y Sociología.
  + Nota media: Nota media obtenida por el alumno en sus estudios.
  + Niveles: EL CV requiere ciertos niveles de idiomas, de soft-skills, de 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 esta sección se exponen los patrones de URI asignados a los recursos comentados anteriormente.

* + **/alumnos**: Reúne todas las peticiones HTTP relacionadas con los alumnos.
  + **/alumnos/:id**: Reúne todas las peticiones HTTP relacionadas un alumno especifico.
  + **/alumnos/:id/CV**: Reúne todas las peticiones HTTP relacionadas con el CV de un alumno especifico.
  + **/alumnos/CV/oferta\_asignada**: Reúne todas las peticiones HTTP relacionadas con los CV de los alumnos con una oferta asignada.
  + **/empresas**: Reúne todas las peticiones HTTP relacionadas con las empresas.
  + **/empresas/:id**: Reúne todas las peticiones HTTP relacionadas una empresa especifica.
  + **/empresas/ofertas**: Reúne todas las peticiones HTTP relacionadas con todas las ofertas.
  + **/empresas/:id/ofertas**: Reúne todas las peticiones HTTP relacionadas con todas las ofertas de una empresa especifica.
  + **/empresas/:id/ofertas/:id**: Reúne todas las peticiones HTTP relacionadas con una oferta especifica de una empresa especifica.
  + **/empresas/ofertas/:alumno\_id**:Reúne todas las peticiones HTTP relacionadas con la oferta asignada a un alumno específico.
  + **/ofertas/:id**: recibe todas las peticiones HTTP relacionadas con una oferta especifica.

#### API de cada recurso

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Método | URI | Utilidad | Semántica | Cuerpo Solicitud | Códigos de respuesta |
| POST | /alumnos | Añade un nuevo alumno. | JSON | Atributos de un nuevo alumno | 201 – Created. Location Header  400 – Bad Request  500 – Internal Server Error |
| GET | / alumnos | Se obtienen un listado de todos los alumnos. | JSON |  | 200 – OK  400 – Bad request  500 – Internal Server Error |
| PUT, DELETE, PATCH | - | - | - | - | 405 – Method not Allowed |

*Tabla 1: Recurso /alumnos*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Método | URI | Utilidad | Semántica | Cuerpo Solicitud | Códigos de respuesta |
| GET | /alumnos/id | Se obtiene todos los atributos 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. Location Header  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 |
| GET | /alumnos/CV/oferta\_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, PUT, DELETE, PATCH | - | - | - | - | 405 – Method not Allowed |

*Tabla 4: Recurso /alumnos/CV/oferta\_asignada*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Método | URI | Utilidad | Semántica | Cuerpo Solicitud | Códigos de respuesta |
| POST | /empresas | Añade una nueva empresa. | JSON | Atributos de una nueva empresa | 201 – Created. Location Header  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 5: 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 6: Recurso /empresas/id*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Método | URI | Utilidad | Semántica | Cuerpo Solicitud | Códigos de respuesta |
| GET | /empresas/ofertas | Se obtiene todos los atributos de todas las ofertas | JSON | - | 200 – OK  404 – Not Found  400 – Bad request  500 – Internal Server Error |
| GET | /empresas/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 |
| POST, PATCH, DELETE | - | - | - | - | 405 – Method not Allowed |

*Tabla 7: Recurso /empresas/ofertas*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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. Location Header  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 8: 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 para verificar si la oferta esta asignada | 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 9: Recurso /empresas/id/ofertas/id*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Método | URI | Utilidad | Semántica | Cuerpo Solicitud | Códigos de respuesta |
| GET | /empresas/ofertas/alumno\_id | Se obtiene la oferta asignada de un alumno específico | JSON | - | 200 – OK  404 – Not Found  400 – Bad request  500 – Internal Server Error |
| POST, PUT, DELETE, PATCH | - | - | - | - | 405 – Method not Allowed |

*Tabla 10: Recurso /empresas/ofertas/alumno\_id*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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 |
| PUT | /ofertas/id | Modificación del campo de alumno asignado y del campo para verificar si la oferta esta asignada | JSON | Atributo alumno y asignado de una oferta | 200 – OK  404 – Not Found  400 – Bad request  500 – Internal |
| POST, DELET, PATCH | - | - | - | - | 405 – Method not Allowed |

*Tabla 11: Recurso /ofertas/id*

### Diseño del algoritmo de recomendación

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

El diseño del algoritmo de recomendación se basa en el cálculo de similitud, pero no directamente entre el CV del alumno que solicita la recomendación y los requerimientos de las ofertas sin asignar, y esto es así para las empresas únicamente tengan que imputar 13 parámetros para definir las ofertas y no 36 parámetros que son los que contienen los CV de los alumnos. El algoritmo de recomendación se estructura en 2 partes:

1. Calculador similaridad ofertas sin asignar: 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. Recomendador ofertas alumno sin 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

En este apartado se mostrará el diseño de la base de datos que se ha llevado a cabo para almacenar los atributos de los recursos definidos en el punto anterior. A continuación, se mostrarán las tablas que contiene la base de datos de la aplicación:

* Alumnos: tabla que almacena todos los alumnos registrados con las siguientes columnas:
  + alumno\_id (Primary Key): Integer.
  + username: String.
  + password: String.
  + nombre: String.
  + apellido: String.
  + telefono: String.
  + email: String.
* Admin: tabla que almacena todos los administradores registrados con las siguientes columnas:
  + admin\_id (Primary Key): Integer.
  + username: String.
  + password: String.
* Empresas: tabla que almacena todas las empresas registradas con las siguientes columnas:
  + empresa\_id (Primary Key): Integer.
  + username: String.
  + password: String.
  + empresa\_nombre: String.
  + teléfono: String.
  + email: String.
* Cv: tabla que almacena todos los cv de cada alumno con las siguientes columnas.
  + Id (Primary Key): Integer.
  + alumno\_id (Foreign Key): Integer.
  + grado: Integer.
  + nota\_media: Integer.
  + Niveles de idiomas, de soft-skills, de competencias y lenguajes de programación: Integer.
    - 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): Integer.
  + alumno\_id (Foreign Key): Integer.
  + empresa\_id (Foreign Key): Integer.
  + empresa\_nombre: String.
  + job\_title: String.
  + ciudad: String.
  + telefono: String.
  + nombre\_contacto: String.
  + estado: String.
  + nota\_media: Integer.
  + grado: Integer.
  + Niveles de idiomas, de soft-skills, de competencias y lenguajes de programación: Integer.
    - Idiomas: inglés, alemán y francés.
    - Soft-Skills: trabajo en equipo y comunicación
    - Competencias: matemáticas, estadística, gestión de proyectos, sostenibilidad, y big data.
    - Programación.

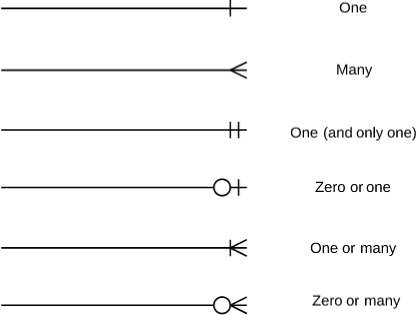
Cada una de las tablas mostradas contiene un PRIMARY KEY (PK). Otras tablas, como cv y ofertas, contienen una o varias FOREIGN KEY (FK), las cuales hacen referencia a la PK de otras tablas. En la siguiente ilustración se muestra el diagrama Entidad Relación de la base de datos diseñada:

Imagen que contiene texto, estacionamiento, mucho, tabla

Descripción generada automáticamente

*Ilustración X: Diagrama E/R de la base de datos*

El diagrama E/R de la base de datos muestra cómo se relacionan las diferentes tablas mencionadas anteriormente. Las cardinalidades entre tablas son las siguientes:



*Ilustración X: Cardinalidades Diagrama E/R de la base de datos*

#### Diseño físico de datos

Se llevará un proceso de normalización y de optimización para obtener las tablas de la base de datos.

#### Migración y carga inicial de los datos (si procede)

En caso de que sea necesario, se elaborará un plan de migración y carga inicial de datos.

## Diseño del Frontend

En este apartado se definirá el diseño llevado a cabo en la parte front end de la aplicación.

### Diseño de la interfaz de usuario

La interfaz de usuario está formada por varias páginas con diferentes funcionalidades, para ello se han diseñado las siguientes ventanas:

Interfaz de usuario gráfica

Descripción generada automáticamente

*Ilustración X: Ventana Login*

*Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente*



*Ilustración X: Ventana creación nueva cuenta*

*Imagen que contiene Diagrama

Descripción generada automáticamente*

*Ilustración X: Ventana principal*

*Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente*

*Ilustración X: Ventana perfil*

*Interfaz de usuario gráfica

Descripción generada automáticamente*

*Ilustración X: Ventana introducir/modificar CV*

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

*Ilustración X: Ventana mostrar CV*

Tabla

Descripción generada automáticamente

*Ilustración X: Ventana ver ofertas*

Interfaz de usuario gráfica

Descripción generada automáticamente

*Ilustración X: Ventana Recomendador de ofertas*

*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*

## Modelo de clases de diseño

El diagrama de clases de análisis se transformará en otro en el que se incluyan aspectos más tecnológicos (accesos a bibliotecas, API de persistencia, etc.).

Xxx

## Entorno de construcción

Para la elaboración del proyecto se ha utilizado el IDE Pycharm de Jetbrains por su facilidad de uso. El IDE utilizado proporciona múltiples frameworks, pero se ha utilizado Flask para la creación de la aplicación web y el servicio Restful.

Se indicará el IDE, *frameworks*, simuladores, etc. que se han utilizado.

## Plan de pruebas

Se establecerán los criterios para codificar las pruebas realizadas a través de *test driven development* (TDD). Asimismo, se diseñarán el resto de pruebas que sean necesarias: estrés, carga, evaluación de la accesibilidad, a través de *scripting* (p.ej., con Selenium), con usuarios reales, etc.

Xxx

## Diagrama de infraestructuras de nivel 3

Se mostrarán las diferentes zonas de seguridad, *firewalls*, VLANs, servidores, etc. Los componentes de la infraestructura podrán ser tanto materiales como virtuales.

Xxx

## Diagrama de infraestructuras de nivel 2 (si procede)

Se mostrarán los *routers*, *switches*, conexiones entre ellos, etc. Para cada uno de los dispositivos se indicarán sus características técnicas.

Xxx

# Implementación

En este capítulo se explicará la implementación llevada a cabo para el diseño explicado en el punto anterior, se ha dividido en Backend y Frontend.

## Implementación Backend

### Implementación Servicio REST

Como se ha comentado en el apartado 5.4, para la implementación del servicio web RESTful se ha utilizado el framework Flask de Python. El programa se ha organizado en los siguientes directorios:

Texto

Descripción generada automáticamente con confianza media

*Ilustración X: Directorios del proyecto*

* .idea: Contiene las configuraciones de ficheros del proyecto.
* migrations y model: Implementación de la base de datos, para más información consultar apartado 6.1.3.
* motorR: Contiene los ficheros del sistema de recomendación, para más información consultar apartado 6.1.2.
* static y templates: Implementación de frontend, para más información consultar apartado 6.2.
* venv\_TFG: contiene el entorno virtual del proyecto donde están todas las bibliotecas y scripts necesarios para la ejecución del proyecto.
* requirements.txt: Contiene los nombres de las bibliotecas y paquetes a instalar en el entorno virtual.
* venv\_TFG.sh: Es un script ejecutable para la instalación del entorno virtual.
* execute\_TFG.sh y R.py: Son scripts un script para ejecutar la aplicación y el sistema de recomendación, para más información consultar apartado xxx
* app.py: 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 expone una porción de código del método GET para la obtención de todas las ofertas que están sin asignar, es decir, disponibles, otras dos porciones para mostrar cómo se realiza la petición del método GET mediante REST para ver todas las ofertas disponibles y para ver si un alumno tiene una oferta asignada:

Texto

Descripción generada automáticamente

*Ilustración C: Porción código para ver ofertas disponibles*

Texto

Descripción generada automáticamente

*Ilustración Z: Porción código REST para ver ofertas disponibles*

*Texto

Descripción generada automáticamente*

*Ilustración T: Porción código REST para comprobar el estado de la oferta*

Continuando el ejemplo proporcionado en el epígrafe 4.6: El alumno Pablo Sotelo decide ver que ofertas hay disponibles en la aplicación, para ello accede a la página correspondiente. Cuando accede a la página se ejecuta el código de la ilustración C, el cual realiza dos peticiones GET para mostrarlas en el front-end:

* 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 Z 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 T pueda realizar una consulta a la base de datos comprobando si el alumno Pablo Sotelo tiene una oferta asignada o no.

### Implementación Algoritmo recomendación

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

De entre los diferentes lenguajes de programación que se pueden utilizar para el análisis de datos he decidido utilizar R para programar el algoritmo de selección.

He elegido R ya que he considerado que presenta ventajas significativas: excelente gama de paquetes de código abierto y de alta calidad, dispone funciones y métodos estadísticos integrales muy completos y permite una muy buena visualización de datos con el uso de bibliotecas como ggplot2. También he 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”: calcula la similitud, en función de los atributos de las ofertas, 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.
  + “Recomendador ofertas alumno sin oferta asignada”: calcula la similitud, en función del CV, 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.
* llamadaAPIS.R: llama al archivo “apis.R” mediante la librería plumber y lo ejecuta en el puerto 8015.
* vecinos\_ofertas\_sin\_asignar.xslx: Resultado en un archivo Excel obtenido de la API “Calculador similaridad ofertas sin asignar” que será necesario para 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 ‘/empresas/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 ‘/empresas/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 ‘/alumnos/CV/oferta\_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.

### Implementación Base de datos

Para almacenar toda la información de los recursos de la aplicación se ha decidido utilizar la base de datos PostGresql. La base de datos está dividida en tablas y cada uno de los datos están almacenados en las diferentes columnas apreciadas en el apartado 5.1.3. A continuación, se mostrará un ejemplo de la tabla alumnos:

Tabla

Descripción generada automáticamente

*Ilustración X: tabla alumnos 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:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

*Ilustración X: directorios Base de datos*

* 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 y 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. A continuación, se mostrará la creación de la clase Alumno:

class Alumno(db.Model):  
 \_\_tablename\_\_ = 'alumnos'  
 alumno\_id = db.Column(db.Integer, primary\_key=True, nullable=False)  
 username = db.Column(db.String, unique=True)  
 password = db.Column(db.String)  
 nombre = db.Column(db.String)  
 apellido = db.Column(db.String)  
 telefono = db.Column(db.String, unique=True)  
 email = db.Column(db.String, unique=True)  
  
 def \_\_init\_\_(self, username, password, nombre, apellido, telefono, email):  
 self.alumno\_id = get\_user\_id()  
 self.username = username  
 self.password = password  
 self.nombre = nombre  
 self.apellido = apellido  
 self.telefono = telefono  
 self.email = email

## 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 fichero CSS y JavaScript. A continuación, se mostrará un ejemplo del código de la tabla para ver todas las ofertas disponibles:

<div class="table100 ver1 m-b-110">  
 <div class="table100-head">  
 <table>  
 <thead>  
 <tr class="row100 head">  
 <th class="cell100 column1">Empresa </th>  
 <th class="cell100 column2">Titulo trabajo</th>  
 <th class="cell100 column3">Ciudad</th>  
 </tr>  
 </thead>  
 </table>  
 </div>  
 <div class="table100-body js-pscroll">  
 <table>  
 <tbody>  
 {% for oferta in ofertas %}  
 <tr class="row100 body">  
 <td class="cell100 column1">{{ oferta.empresa\_nombre }}</td>  
 <td class="cell100 column2">{{ oferta.job\_tittle }}</td>  
 <td class="cell100 column3">{{ oferta.ciudad }}</td>  
 </tr>  
 {% endfor %}  
 </tbody>  
 </table>  
 </div>  
</div>

La variable ofertas de en el código anterior se obtiene a través de una petición GET que se ha explicado previamente en el epígrafe 6.1.1. A continuación, se mostrará como se ve el código anterior desde un navegador:

Interfaz de usuario gráfica, Aplicación, Teams

Descripción generada automáticamente

*Ilustración X: Visualización tabla ofertas disponibles*

En el proyecto se han organizado las plantillas HTML en un directorio llamado templates como se puede apreciar a continuación:

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

*Ilustración X: Directorio templates*

Las plantillas css, archivos json y las imagenes se encuentran en el directorio static:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

*Ilustración X: Directorio static*

# Construcción

## Referencia al repositorio de software

Se indicará la URL del repositorio en que está el software para, si procede, que el tribunal pueda probarlo en sus propias máquinas.

EXPLICAR AWS

## Manuales

En caso de que no tenga sentido incluir en el fichero README.md del repositorio alguno de los siguientes manuales: instalación, operación o usuario, se incluirá en la memoria del TFG.

## 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. A continuación, se muestra un ejemplo en el que se crea al alumno Pablo Sotelo a través de una petición POST:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Ilustración X: Petición POST en Postman

Como se puede observar en la ilustración X, hay que pasar los parámetros necesarios del alumno dentro del body de la petición. En la siguiente imagen se muestra la respuesta que da el servidor a la petición POST de la ilustración X:

Interfaz de usuario gráfica, Aplicación, Teams

Descripción generada automáticamente

Ilustración Y: Respuesta de petición POST en Postman

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

# Conclusiones y líneas futuras

Ideas a las que se llega después del desarrollo del proyecto, así como las líneas posibles de trabajo posterior.

Xxx

# Referencias

1. CIENCIA DE LOS DATOS Kelleher, D. J., Tierney, B. (2021). Ciencia de datos (MIT Press / Conocimientos Esenciales) (Spanish Edition). Ediciones UC.
2. MACHINE LEARNING Roman, V. (2019, 6 de febrero). Introducción al Machine Learning: una guía desde cero. Ciencia y Datos. <https://medium.com/datos-y-ciencia/introduccion-al-machine-learning-una-gu%C3%ADa-desde-cero-b696a2ead359>
3. METODOS DE APRENDIZAJE Cañadas, R. (2021, 7 de septiembre). Algoritmos de machine learning. ABdatum. <https://abdatum.com/machine-learning/algoritmos-machine-learning>.
4. SISTEMAS DE RECOMENDACIÓN González, A. (2020). Sistemas de recomendación de contenido con Machine Learning. Cleverdata. <https://cleverdata.io/sistemas-recomendacion-machine-learning/>
5. FILTRADO COLABORATIVO Bagnato, J.I. (2019, 27 de agosto). Sistemas de Recomendación. Crea en Python un motor de recomendación con Collaborative Filtering. Aprende Machine Learning. <https://www.aprendemachinelearning.com/sistemas-de-recomendacion/>
6. MEDIDAS DE SIMILITUD Mendoza Olguín, G. E., Laureano De Jesús, Y., & Pérez De Celis Herrero, M. D. L. C. (2019). Métricas de similaridad y evaluación para sistemas de recomendación de filtrado colaborativo. Revista de Investigación en Tecnologías de la Información, 7(14), 224–240. <https://doi.org/10.36825/riti.07.14.019>
7. <https://martinfowler.com/articles/richardsonMaturityModel.html>
8. Lucidchart. Notación y símbolos de diagramas entidad-relación. <https://www.lucidchart.com/pages/es/simbolos-de-diagramas-entidad-relacion>
9. <https://www.ceupe.com/blog/que-son-los-servicios-web.html>
10. <https://atvise.vesterbusiness.com/news/tipos-web-service-en-la-industria/>
11. https://www.marcpampols.net/es/arquitectura-rest-que-es-para-que-sirve/

# Anexo I

# Otras posibilidades para realizar el análisis y el diseño

A lo largo de este documento se ha asumido un desarrollo orientado a objetos. No obstante, se pueden seguir otros enfoques, por ejemplo, un desarrollo estructurado con diagrama de contexto, diagramas de flujos de datos, diagrama entidad-relación, etc.

Asimismo, es posible utilizar el diagrama entidad-relación de forma complementaria a los diagramas de clases.

Por otra parte, el tutor del proyecto puede dar sus propias orientaciones si lo considera oportuno.

# Anexo II

# Fuente de inspiración del presente documento

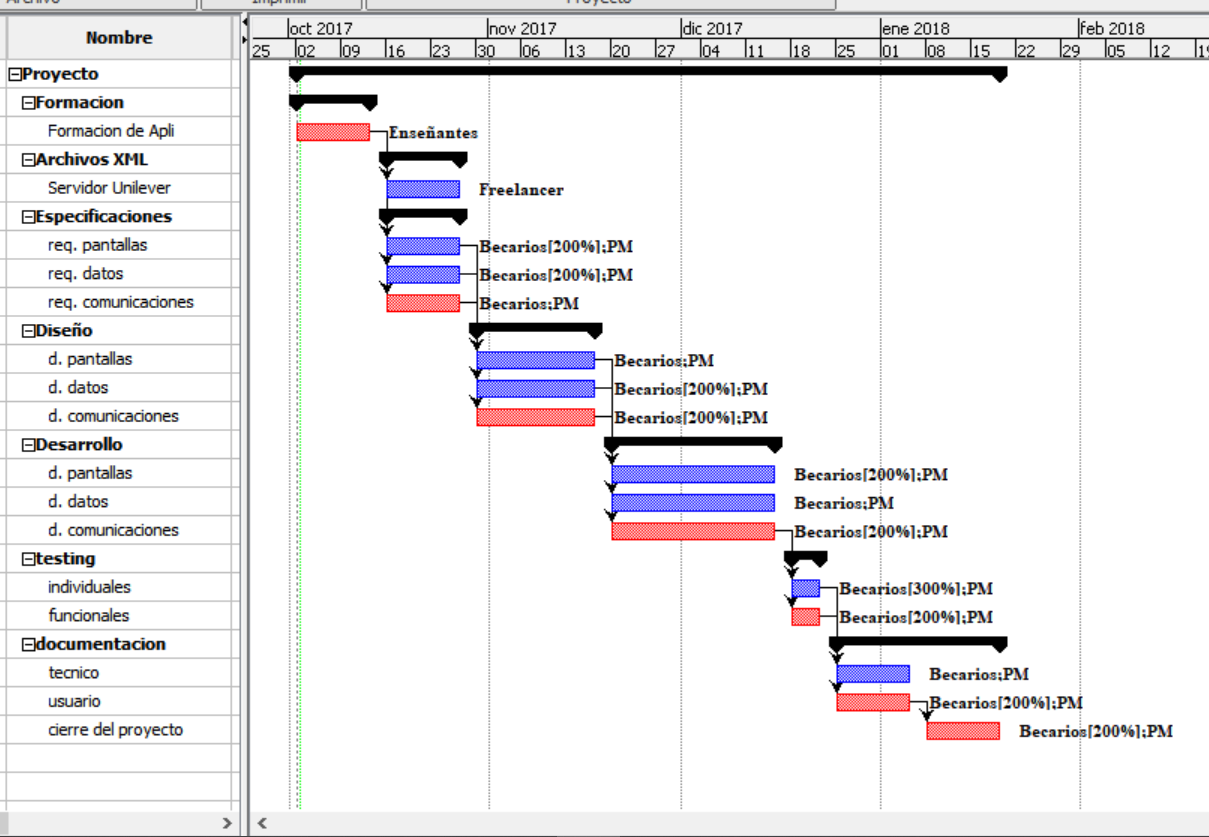
Independiente de que su uso que esté actualmente más o menos extendido, se ha considerado que la metodología [Métrica 3](https://administracionelectronica.gob.es/pae_Home/pae_Documentacion/pae_Metodolog/pae_Metrica_v3.html) es adecuada como marco de referencia en la elaboración de la documentación. En consecuencia, ha sido una fuente de inspiración.

# Glosario de términos

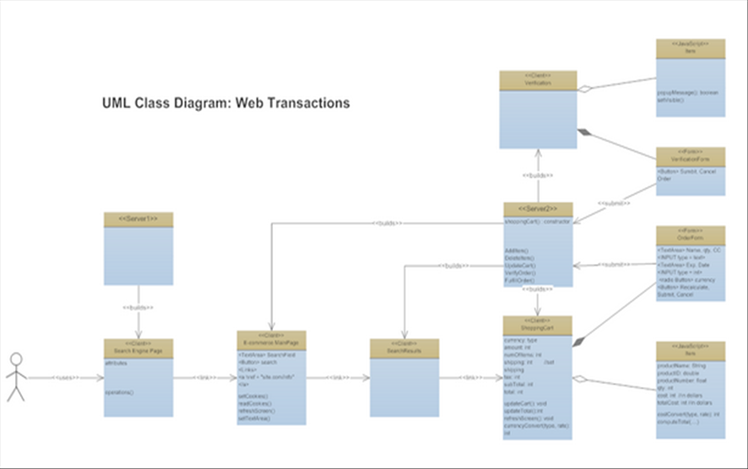
Si es necesario.

# Anexos

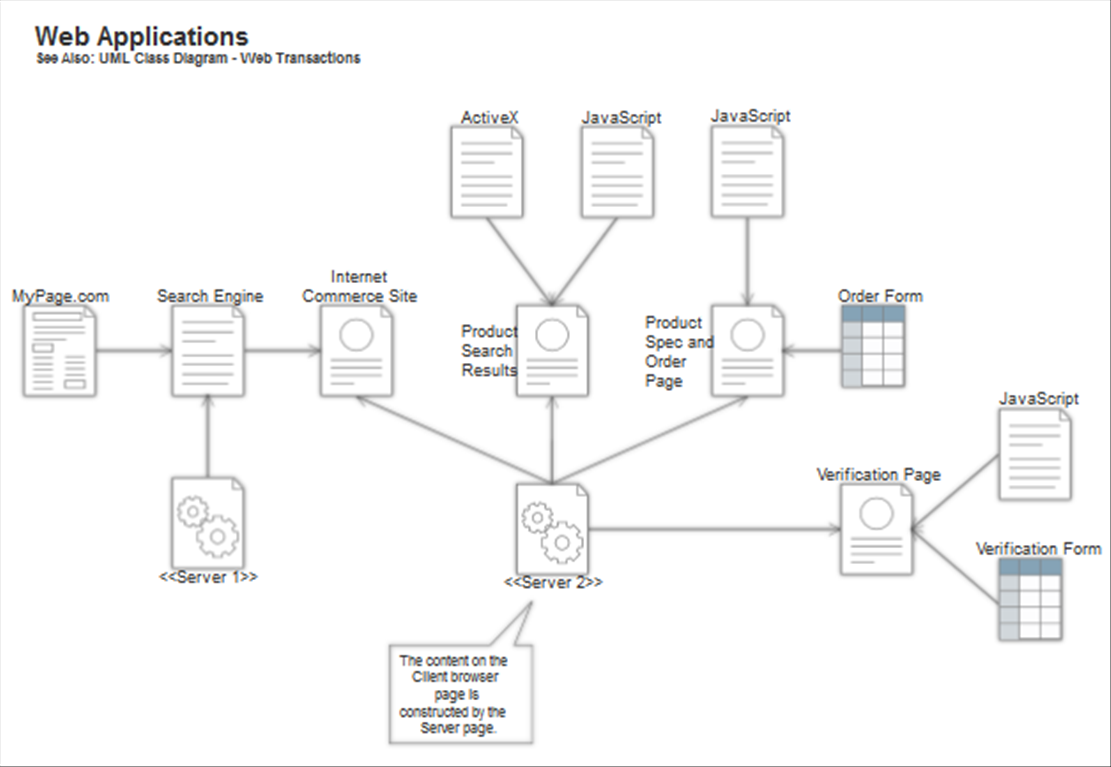
Ejemplo de planificación de un TFG



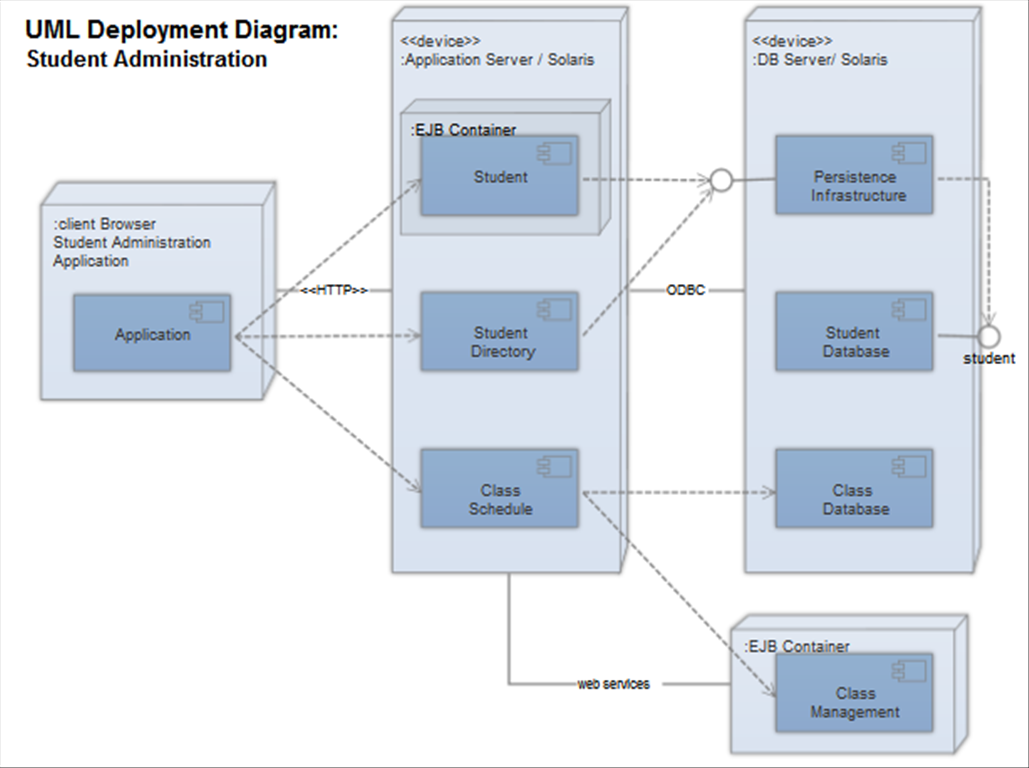
Ejemplo de diagrama de clases



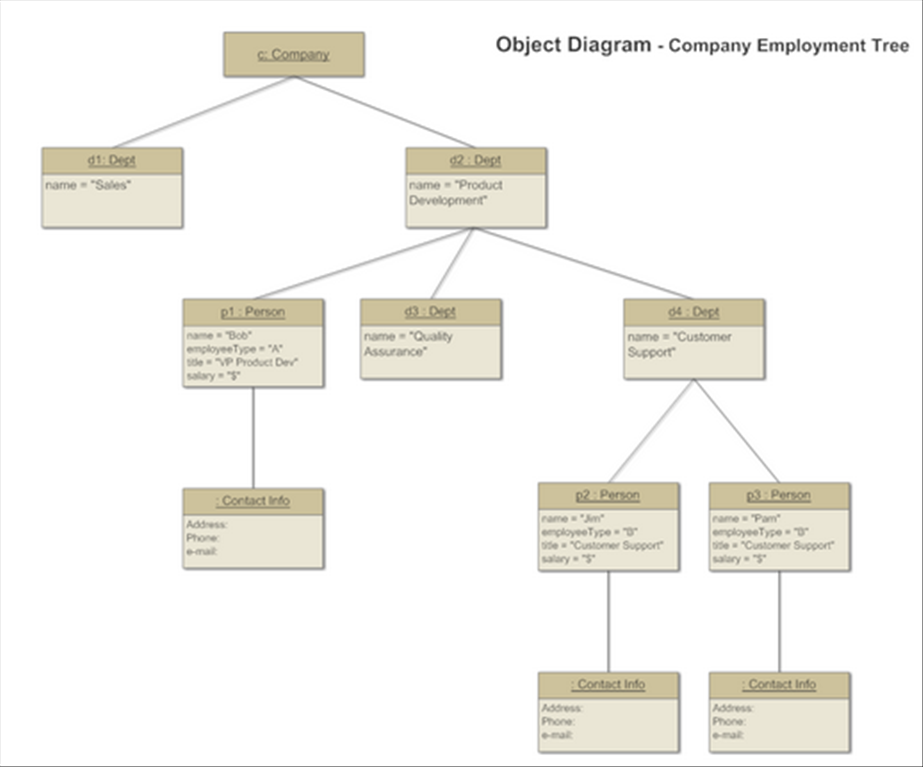
Ejemplo de diagrama de componentes



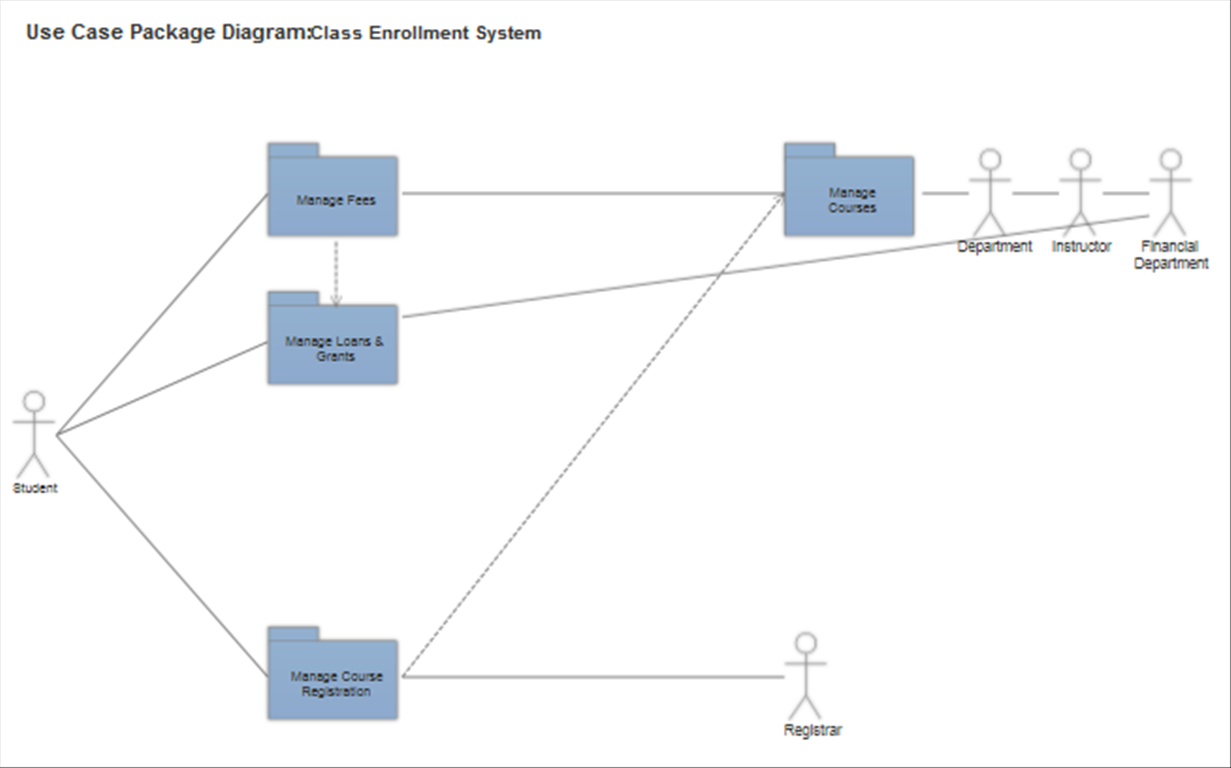
Ejemplo de diagrama de despliegue



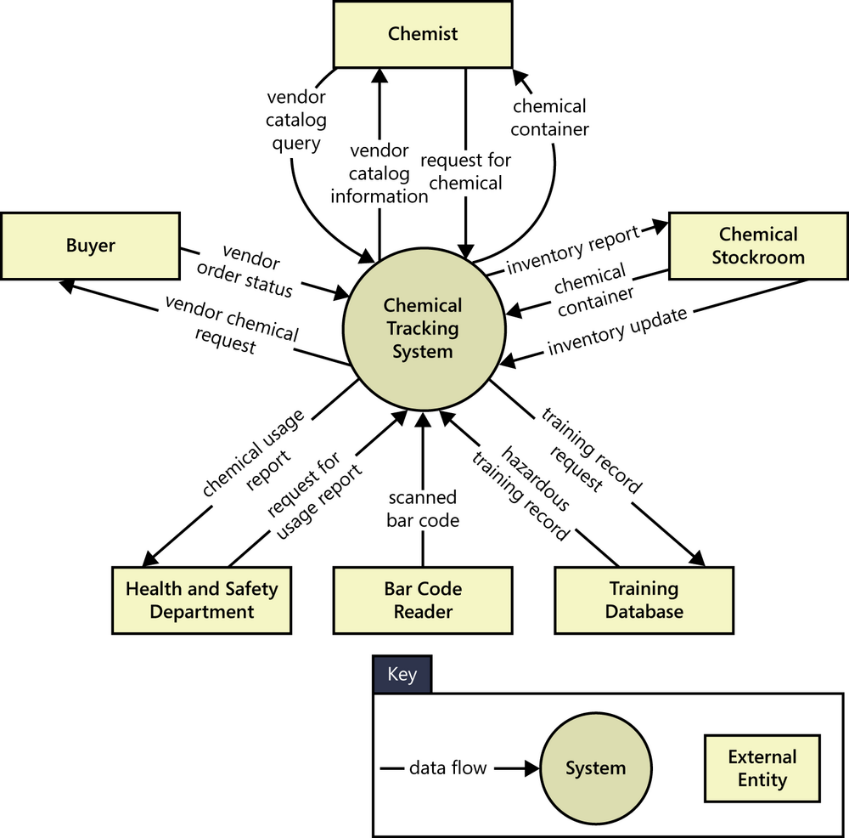
Ejemplo de diagrama de objetos



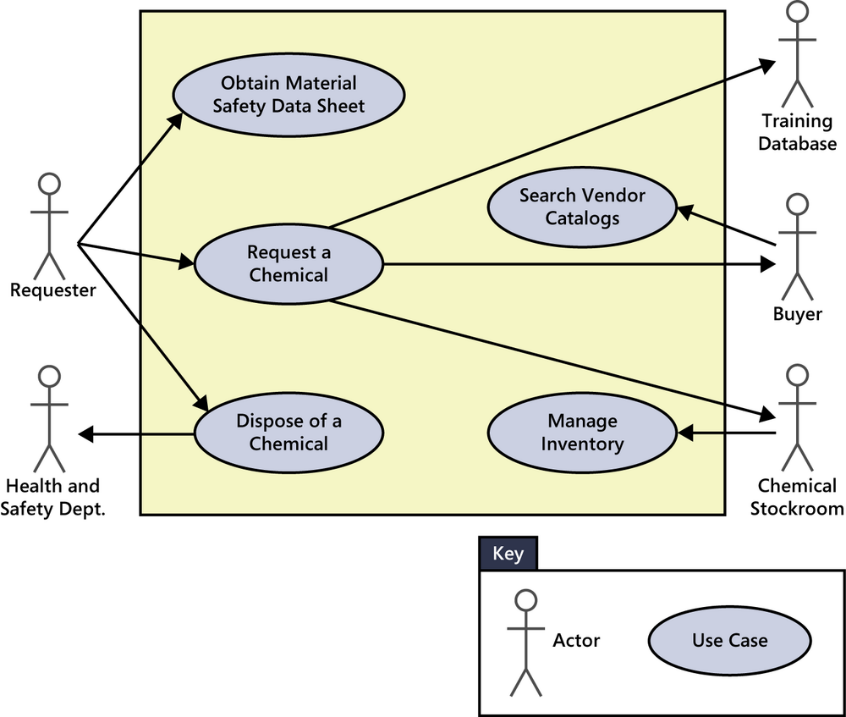
Ejemplo de diagrama de paquetes



Ejemplo de diagrama de contexto



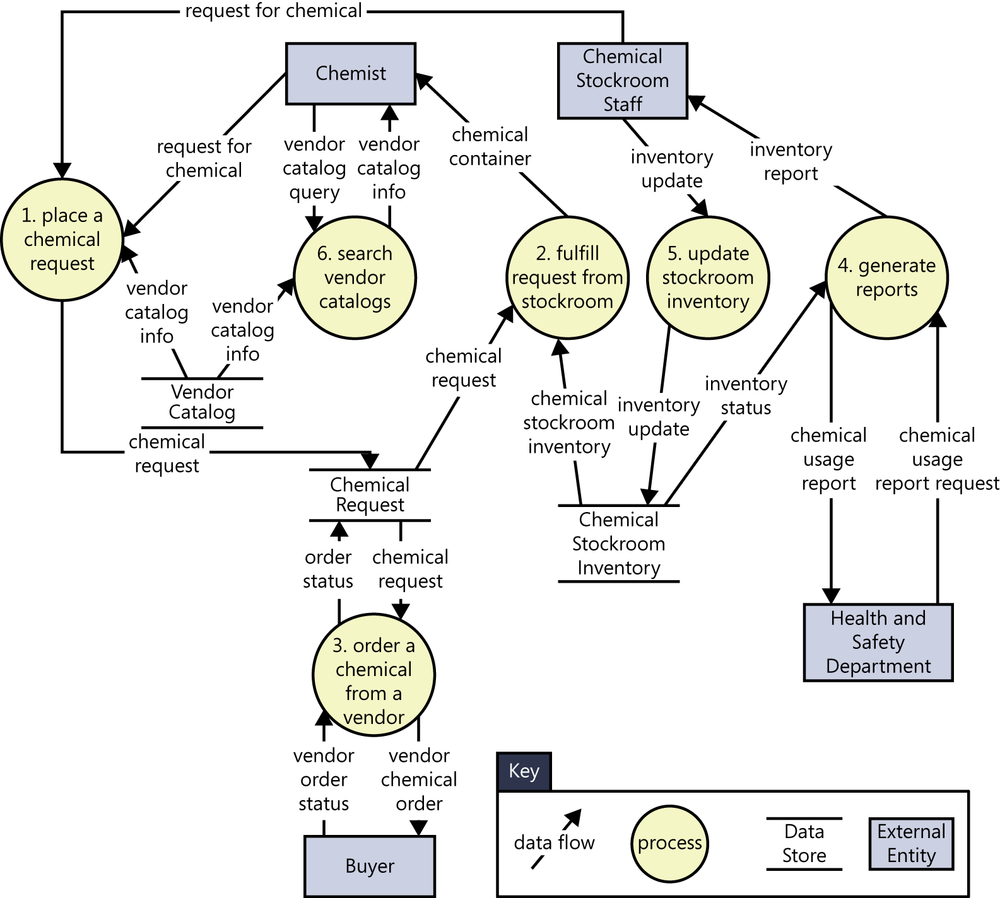
Ejemplo de caso de uso



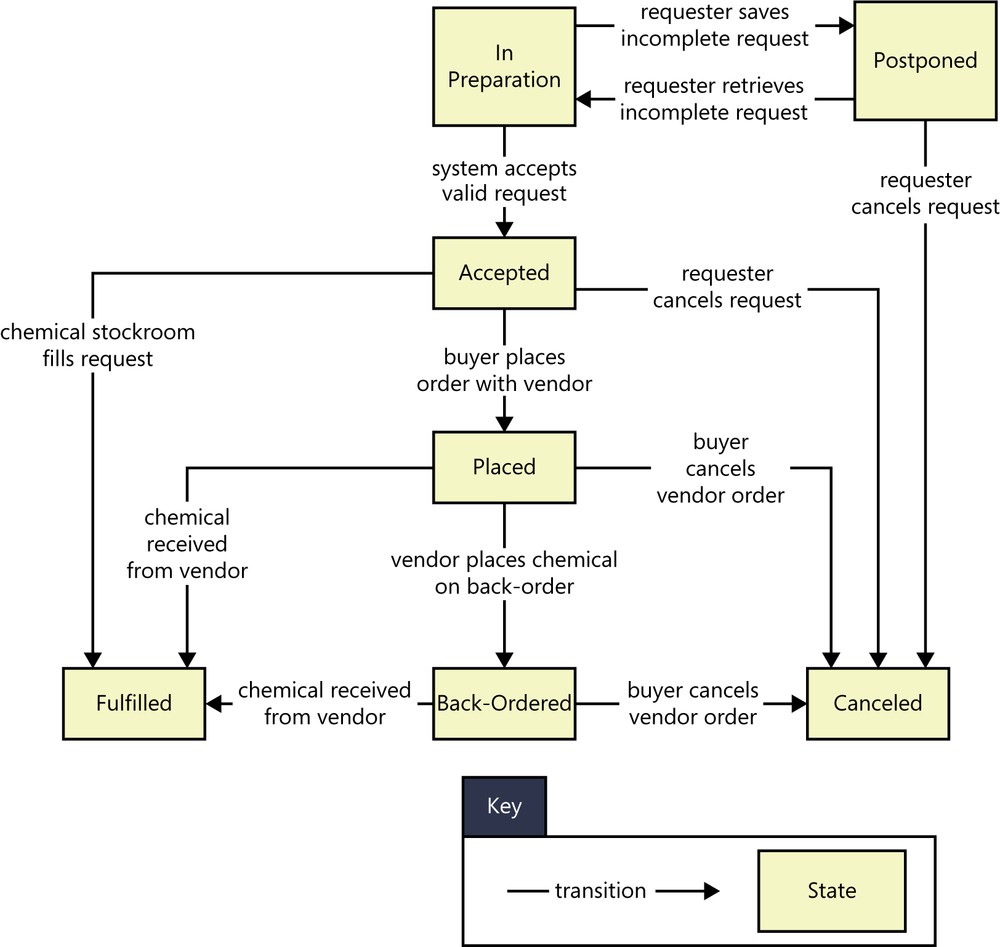
Ejemplo de diagrama de actividades

An illustration showing the normal flow of a use
                  case in the left part of the figure. Starting with a solid
                  dot that represents the use case preconditions, arrows lead
                  downward through a series of steps in rectangles to the use
                  case postconditions. In the middle of the stack of
                  rectangles is a diamond representing a branch point
                  condition. An arrow goes from the diamond to the right,
                  where there is another series of vertical boxes and arrows
                  representing steps in the alternative flow. The last step in
                  the alternative flow has an arrow going back to the left to
                  rejoin the last step in the normal flow.

Ejemplo de DFD Diagrama de Flujo de datos



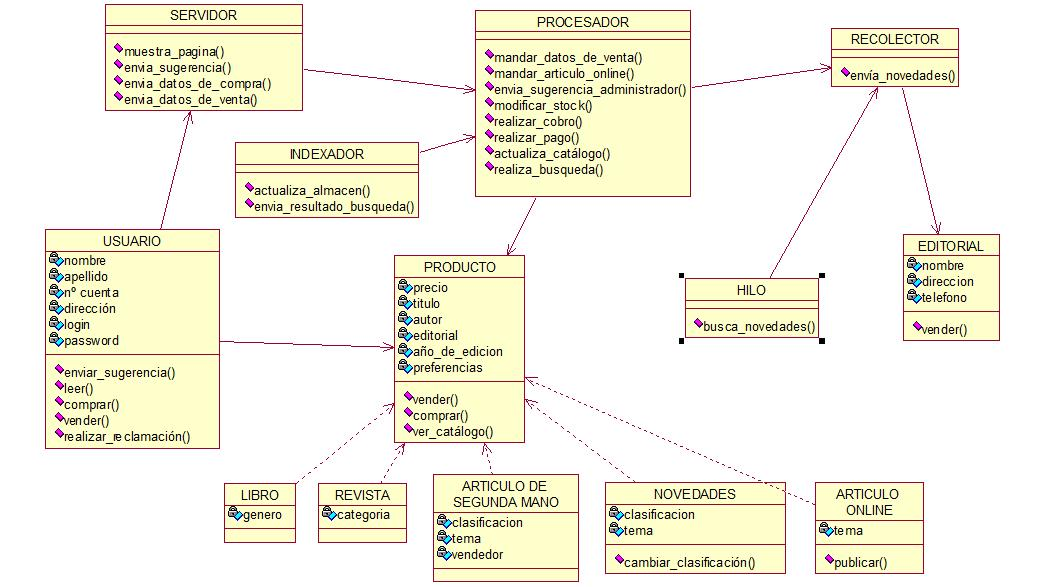
Ejemplo de diagrama de transición



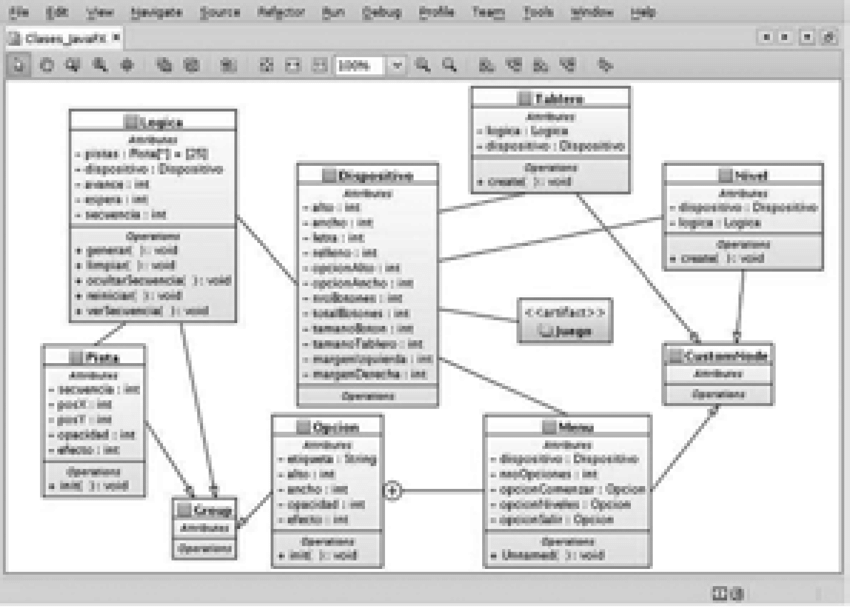
Ejemplo de diagrama lógico de una vpn



Ejemplo de Diagrama E/R



Ejemplo de diagrama de clases de usuarios



Ejemplo de interfaz HMI

