UNIVERSIDAD SAN PABLO - CEU

ESCUELA POLITÉCNICA SUPERIOR

GRADO EN INGENIERÍA DE SISTEMAS DE INFORMACIÓN



TRABAJO FIN DE GRADO

Diseño e Implementación de una

aplicación RESTful para la disminución del abandono en el primer año universitario

Design and Implementation of a

RESTful application to reduce dropout in the first year of university



Junio 2022

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Datos del alumno   |  | | --- | | Nombre: DAVID RECIO ARNÉS |   Datos del Trabajo   |  | | --- | | TÍTULO DEL PROYECTO:  Diseño e Implementación de una aplicación RESTful  para la disminución del abandono en el primer año universitario. |   Tribunal calificador   |  |  | | --- | --- | | Presidente: | Fdo.: |  |  |  | | --- | --- | | Secretario: | Fdo.: |  |  |  | | --- | --- | | Vocal: | Fdo.: |  |  | | --- | | Reunido este tribunal el \_\_\_ /Junio/2022, acuerda otorgar al Trabajo Fin de Grado presentado por D. David Recio Arnés la calificación de \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | |

Resumen

Texto con, a lo sumo, 200 palabras.

En la actualidad, los jóvenes que están en el proceso de acceder a la universidad se enfrentan a una problemática que es la elección de su grado o carrera universitaria, por falta de orientación vocacional, y muchos de ellos se equivocan en sus elecciones lo cual se ve reflejado en el abandono del primer curso del grado, siendo estos puntos la motivación central para la realización de este TFG, pues mediante la creación de un servicio Web RESTful especializado, la orientación vocacional afín a sus gustos personales, garantizan al estudiante la orientación adecuada para la elección de su grado a través de formularios estandarizados con bases psicológicos, psicotécnicos y pedagógicos para así garantizar una mayor continuidad y evitar la deserción universitaria. El servicio Web RESTful realizado en este TFG da un servicio optimo mediante dos formularios uno donde se orienta en los grados universitarios (ingeniería, ciencias sociales, artes y ciencias) y consiguiente el otro formulario que ve la capacidad de concentración para así ver cuánto le puede afectar la carrera que desea entrar y por último se toma las notas del usuario y da así una respuesta final mediante sugerencias en la facilidad o no del grado que desea.

Palabras Clave

Abstract

Resumen anterior, en inglés.

Keywords

Índice de contenidos

[Capítulo 1 Introducción 1](#_Toc106131005)

[1.1 Objetivos 2](#_Toc106131006)

[Capítulo 2 Gestión del proyecto 3](#_Toc106131007)

[2.1 Modelo de ciclo de vida 3](#_Toc106131008)

[2.1.1 El diseño del programa es lo primero (“*program design come first”*) 4](#_Toc106131009)

[2.1.2 Documentar el diseño (“*document the design”*) 5](#_Toc106131010)

[2.1.3 Hazlo dos veces (“*do it twice”*) 6](#_Toc106131011)

[2.1.4 Planificación y pruebas de control (“*Plan, control and monitor testing*”) 6](#_Toc106131012)

[2.2 Papeles desempeñados en el proyecto 7](#_Toc106131013)

[2.2.1 Roles del tutor 7](#_Toc106131014)

[2.2.2 Roles del estudiante 7](#_Toc106131015)

[2.3 Planificación 7](#_Toc106131016)

[2.4 Presupuesto 8](#_Toc106131017)

[2.4.1 Costes en la fase de desarrollo 9](#_Toc106131018)

[2.4.2 Costes en la fase de producción 9](#_Toc106131019)

[2.5 Ejecución 10](#_Toc106131020)

[Capítulo 3 Estado del arte 11](#_Toc106131021)

[3.1 ¿Qué son los Servicios Web tradicionales y cómo funcionan? 11](#_Toc106131022)

[3.1.1 Servicios Web RESTful (estilo arquitectónico REST) 13](#_Toc106131023)

[3.1.2 Modelo de madurez de Richardson 16](#_Toc106131024)

[Capítulo 4 Análisis 19](#_Toc106131025)

[4.1 Análisis de dominio 19](#_Toc106131026)

[4.2 Especificación de requisitos 23](#_Toc106131027)

[ *Requisitos funcionales* 26](#_Toc106131028)

[4.3 Análisis de los casos de uso y de las clases de análisis 27](#_Toc106131029)

[4.4 Análisis de seguridad 28](#_Toc106131030)

[Capítulo 5 Diseño e implementación 33](#_Toc106131031)

[5.1 Arquitectura del sistema 33](#_Toc106131032)

[5.2 Modelo de clases de diseño 33](#_Toc106131033)

[5.3 Diseño físico de datos 33](#_Toc106131034)

[5.4 Migración y carga inicial de datos (si procede) 33](#_Toc106131035)

[5.5 Diseño de la interfaz de usuario 33](#_Toc106131036)

[5.6 Entorno de construcción 34](#_Toc106131037)

[5.7 Plan de pruebas 34](#_Toc106131038)

[5.8 Diagrama de infraestructuras de nivel 3 34](#_Toc106131039)

[5.9 Diagrama de infraestructuras de nivel 2 (si procede) 34](#_Toc106131040)

[Capítulo 6 Construcción 34](#_Toc106131041)

[6.1 Referencia al repositorio de software 34](#_Toc106131042)

[6.2 Manuales 35](#_Toc106131043)

[Capítulo 7 Conclusiones y líneas futuras 37](#_Toc106131044)

[Bibliografía 39](#_Toc106131045)

[Anexo I 41](#_Toc106131046)

[Otras posibilidades para realizar el análisis y el diseño 41](#_Toc106131047)

[Anexo II 42](#_Toc106131048)

[Fuente de inspiración del presente documento 42](#_Toc106131049)

[Glosario de términos 43](#_Toc106131050)

[Anexos 45](#_Toc106131051)

Índice de ilustraciones

[Ilustración 1. Metodología en Cascada 3](#_Toc106842458)

[Ilustración 2. Variante Metodología en Cascada 4](#_Toc106842459)

[Ilustración 3. Planificación estimada 6](C:\\Users\\david\\Documents\\GitHub\\tfgFinal\\Doc\\Memoria.docx" \l "_Toc106842460)

[Ilustración 4. Servicios Web Tradicionales 9](#_Toc106842461)

[Ilustración 5. Discovery Process 10](#_Toc106842462)

[Ilustración 6. Diagrama de una estructura REST 11](C:\\Users\\david\\Documents\\GitHub\\tfgFinal\\Doc\\Memoria.docx" \l "_Toc106842463)

[Ilustración 7. Niveles de madurez de los Servicios Web REST 14](#_Toc106842464)

[Ilustración 8. Nivel 0 de Madurez del Servicio Web REST 15](#_Toc106842465)

[Ilustración 9. Nivel 1 de Madurez del Servicio Web REST 16](#_Toc106842466)

[Ilustración 11. Nivel 3 de Madurez del Servicio Web REST 19](#_Toc106842467)

[Ilustración 11.Población de matriculados en universidades. 21](#_Toc106842468)

[Ilustración 12. Test de Toulouse 25](#_Toc106842469)

[13 Funcionalidades del sistema 27](#_Toc106842470)

Índice de tablas

[Tabla 1. Tasas de abandono en el primer año 24](#_Toc106842494)

[Tabla 2. Evaluación CHASIDE 25](#_Toc106842495)

# Introducción

Desde hace algunos años los problemas más importantes que se encuentran las universidades durante el primer año universitario son la tasa de abandono y el fracaso académico. Para evitar este fracaso, hay que centrarse en el estudio de los factores que determinan el éxito o fracaso de un estudiante, tales como: factores comportamentales (hábitos de estudio), factores afectivos (nivel de satisfacción), y factores motivacionales (internos y externos).

Por su parte, el estudiante al iniciar la universidad se encuentra con una serie de dificultades. En primer lugar, la elección de la titulación, para ello existen unas jornadas que facilitan los colegios el último año donde visitan distintas universidades y carreras, a modo de orientación. Para la elección de la carrera también es fundamental conocer la vocación y las aptitudes del estudiante; la vocación tiene carácter intrínseco que no puede evaluarse de la misma forma que las aptitudes, en cambio, las aptitudes deberían alinearse con la carrera que seleccionara el alumno para que obtuviera un mejor rendimiento, para ello, pueden evaluarse mediante unos formularios estandarizados que ofrecen unos resultados que sirven a modo de recomendación para elegir mejor una titulación.

Una vez que el estudiante conoce sus aptitudes, puede determinar qué asignaturas requieren de una mayor o menor concentración de estudio, y así poder realizar una mejor planificación de su tiempo. Esto es fundamental, dado que muchos estudiantes tienen que abordar toda la carga de trabajo que conllevan unos estudios universitarios, y una planificación que hasta ese momento de sus vidas no han tenido que hacer. Por último, no hay que olvidar que otra de las dificultades que se encuentra el estudiante es establecer relaciones entre compañeros, que le servirán en un futuro para facilitar el trabajo en equipo.

## Objetivos

Para abordar las dificultades que se encuentra el estudiante antes de comenzar su primer año de universidad y durante el desarrollo mismo (mencionadas en el apartado anterior), se va a crear un Servicio Web que asesora y acompaña al estudiante mediante recomendaciones durante ese período. Para ello se han establecido los siguientes objetivos:

* El sistema será capaz de realizar una valoración de las aptitudes del estudiante, y de su concentración mediante el análisis de los resultados de unos formularios estandarizados, para realizar recomendaciones sobre la elección de la titulación.
* El sistema será capaz de realizar una planificación de tiempos de estudio, mediante recomendaciones de los datos obtenidos anteriormente.
* El sistema buscará fomentar la colaboración entre estudiantes y el aprendizaje grupal, para reforzar las bases de conocimiento de los compañeros de segundo, mediante el apoyo a los estudiantes de primero.

# Gestión del proyecto

## Modelo de ciclo de vida

En este caso la metodología escogida fue la metodología en cascada dado que se ajusta a la este, dado que los requerimientos son fijos y el trabajo avanza en forma lineal hacia el final.[6]

La versión original fue presentada por Royce en 1970, aunque son más conocidos los refinamientos realizados por Boehm en 1981, Sommerville en 1985y Sigwart y col. En 1990. Esta metodología, se basa en la evolución del producto a través de una secuencia de fases de forma lineal mediante iteraciones del estado anterior.

Diagrama

Descripción generada automáticamente con confianza media

Ilustración 1. Metodología en Cascada

Esta metodología comienza con la especificación de los requerimientos por parte del cliente y avanza a través de planificación , modelado, construcción y despliegue, para concluir con el mantenimiento del software.

Para que se adapte mejor al proyecto se ha realizado una variante, la cual simplemente disgrega las partes, dando lugar a un total de siete etapas diferentes:

Diagrama

Descripción generada automáticamente

Ilustración 2. Variante Metodología en Cascada

1. **Análisis de requisitos del software**. En esta etapa el analista se sienta junto al cliente escuchando las necesidades de este y tras estas reuniones genera SRD (documento de especificación de requisitos), que contiene la especificación completa de lo que debe hacer el sistema sin detalles técnicos, dicho documento debe estar consensuado con el cliente para delimitar el alcance del proyecto.
2. **Diseño del sistema.** Se descompone y organiza el sistema en partes separadas, generando el SDD (Descripción del diseño del software), que contiene la descripción de la estructura del sistema la funcionalidad de sus partes, así como la manera en que se combinan unas con otras.
3. **Diseño del programa.** Se desarrollan los algoritmos necesarios para satisfacer los requerimientos del cliente , además del estudio necesario para saber que herramientas son requeridas para la etapa de codificación
4. **Codificación.** Se implementa el código del programa para que realice las funcionalidades detalladas en los algoritmos.
5. **Testing.** Realización de pruebas y corrección de errores procedentes de la etapa anterior teniendo como objetivo revisar que se satisfaga las necesidades del cliente, previamente declaradas en el SDD
6. **Despliegue del software.** Ejecución del sistema, donde el cliente revisa y valida si es cierto que se consiguieron cubrir todas sus necesidades.
7. **Mantenimiento.** Adaptación y corrección de las necesidades no cubiertas por parte del cliente.

## Papeles desempeñados en el proyecto

Según la naturaleza del proyecto, nos encontramos 2 entidades, siendo estas el tutor del trabajo fin de grado (TFG) y el estudiante.

### Roles del tutor

El tutor ha realizado tres roles: el rol de director del proyecto, ya que ha participado en la planificación y definición de objetivos; de analista de requisitos ayudó a establecer los requisitos de la aplicación.

### Roles del estudiante

El alumno ha ejercido cuatro roles: el rol de cliente, puesto que propuso la idea de la aplicación; analista de requisitos dado que estableció los requisitos de la aplicación; de desarrollador, diseñó y escribió el código; y, finalmente, de *tester*, ya que realizó las pruebas necesarias para validar el correcto funcionamiento de la aplicación.

## Planificación

En este apartado se muestran, mediante un diagrama de GANTT, los tiempos estimados que se dedican a las tareas, antes y durante el desarrollo del programa, para extraer una visión más amplia del recorrido, arrojando aún más luz en el apartado de conclusiones.

Imagen que contiene Gráfico

Descripción generada automáticamente

Ilustración 3. Planificación estimada

Como se puede observar en la ilustración anterior las tres partes que más tiempo ocupan del desarrollo del programa son: la creación de las bases del proyecto que son: el desarrollo de la idea del proyecto, análisis del problema y los objetivos que se van a abordar en el proyecto.. Esto se debe a la relevancia que tienen, puesto que condicionarán el desarrollo del mismo al ser dependientes uno del otro. Presupuesto

En este apartado hay que diferenciar entre dos tipos de costes, según la fase del desarrollo del programa en que nos encontremos: costes en la fase de desarrollo y costes en la fase de producción.

### Costes en la fase de desarrollo

* Luz (kW/h). Teniendo en cuenta que durante los días hábiles de destinan 2 horas/día al TFG, los días de fin de semana 4 horas/día, y los días festivos no se incluyen (puesto que no se trabaja en el TFG). El precio medio de la luz actualmente en el mercado regulado es de 0.21846 euros/kWh. Por tanto, tras los cálculos, supone 73, 4 euros/336 horas totales destinadas al desarrollo del programa.



* Raspberry Pi 4 Modelo B 8GB HeatsinkSet. Este modelo en PC Componentes tiene un precio de 159,28 euros.
* Equipo Acer Predator g3620 con i7-3770, 16GB RAM, 2TB. Tiene un precio en MediaMarkt de 849 euros IVA incluido.
* Licencia Microsoft Windows 10 Home 64 Bits OEM. Tiene un precio en PC Componentes de 127,07 euros.
* Licencias de Software (Sublimetext, Intellijide). Gratis al ser estudiante

### Costes en la fase de producción

* Amazon WebService:
  + 50 TB/mes 🡪 0,77 euros por GB.
  + t2.medium 🡪 0,93 euros por hora el API.
  + Totales 🡪 670,37 euros/mes.
* Salario de los empleados: desarrollador, dedicado al mantenimiento y experto técnico en casos puntuales (externo) 🡪 19.000 euros por el desarrollador y 2200 euros brutos/mes por el experto.

## 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.).

# Estado del arte

En este capítulo se expondrá el contexto y definición de los servicios Web RESTful, las tecnologías utilizadas y una comparativa entre las aplicaciones ya existentes.

## ¿Qué son los Servicios Web tradicionales y cómo funcionan?

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

Ilustración 4. Servicios Web Tradicionales

Según W3C  [4]un servicio Web es un sistema de software diseñado para permitir la interacción máquina a máquina a través de una red, y tiene una interfaz descrita y comprensible por las máquinas (específicamente WSDL). Además, la forma de comunicarse con el Servicio Web es mediante el uso de mensajes SOAP, normalmente transmitidos mediante HTTP, con una serialización XML, junto con otros estándares relacionados con la web.

El intercambio de mensajes se encuentra estandarizado bajo una descripción de un Servicio Web (WSD), qué a su vez, se trata de una especificación que es capaz de procesar la máquina, escrito en WSDL, que define los formatos, protocolos y tipos de datos. “*En esencia, la descripción del servicio representa un acuerdo que rige la mecánica de interacción con ese servicio”.*

Los mensajes SOAP (Protocolo de Arquitectura Orientada a Servicios) [7]proporcionan un marco estándar, extensible y componible para empaquetar e intercambiar mensajes XML mediante el uso de encabezados. Pueden ser transportados por una variedad de protocolos de red, cómo HTTP, SMTP, FTP, RMI/IIOP o un protocolo de mensajería propietario, cuya finalidad es representar la información necesaria para invocar un servicio o reflejar los resultados de una invocación de servicio, y contiene la información especificada en la definición de interfaz de servicio.

Diagrama

Descripción generada automáticamente

Ilustración 5. Discovery Process

Para finalizar, es importante definir cómo funcionan los Servicios Web. En este caso, se muestra el proceso de descubrimiento, el cual sigue los distintos pasos:

1. Las entidades implicadas (receptor y proveedor) se “conocen” entre sí.
   1. Este proceso se realiza mediante la obtención del WSD y una descripción funcional asociada a éste (FD). La FD puede ser tan simple como los metadatos o una URI, o puede ser más compleja como una colección de declaraciones.
   2. El solicitante es el que proporciona los criterios al servicio de descubrimiento con el fin de seleccionar un WSD basada en una FD.
2. Una vez realizada la primera etapa, ambas entidades acuerdan una semántica para realizar la interacción deseada. Esta semántica puede ser ofrecida por parte de la entidad proveedora al solicitante mediante el régimen “Tómalo o déjalo”.
3. En esta etapa, ambas entidades incorporan tanto el WSD como la semántica para poder comunicarse.
4. Por último, una vez incorporado el paso anterior, ambas entidades empiezan a comunicarse mediante mensajes SOAP.

### Servicios Web RESTful (estilo arquitectónico REST)

El estilo arquitectónico REST ( Representational State Transfer) lo definió Roy Fielding como “una arquitectura de software para sistemas hipermedia distribuidos” [11].REST se fundamenta en el sistema cliente - servidor, en el que el cliente ingresa en los servicios a través de un puerto (socket), usando el protocolo HTTP como fuente de comunicación de los mensajes.

Diagrama

Descripción generada automáticamente

Ilustración 6. Diagrama de una estructura REST

El éxito de la arquitectura REST es porque da resolución a las necesidades de un sistema hipermedia: “diseñada para ajustarse a las necesidades de un sistema hipermedia distribuido de gran escala: escalabilidad en las interacciones 10 entre componentes, interfaces genéricas, despliegue independiente de componentes y diseño de componentes intermedios para reducir la latencia de las interacciones y reforzar la seguridad y encapsular sistemas heredados”[12]. El uso más importante del estilo arquitectónico lo encontramos en la Web (Es un sistema hipermedia elaborado utilizando este estilo). Para alcanzar sus objetivos, las limitaciones que impone el estilo arquitectónico REST son:

**Recurso: Identificación, estado.** La conceptualización principal en la que se fundamenta REST son los recursos. Por definición un recurso es todo elemento dinámico clave para el orden de la aplicación que se quiere crear (p.e: Formulario, Usuario, entre otros). El recurso debe ser identificable y único de modo evidente usando un proceso estandarizado. En el ejemplo de la web se usan las URI( Unified Resource Identifier); p.e: /usuario/:idUsuario/formulario. La URI ejecuta como nombre y dirección en cada recurso. Lo cual admite que se pueda “navegar” entre los diferentes recursos. La URI no admite acciones y no se debe filtrar información del recurso.

***Representación de un recurso.*** La representación del recurso hace referencia a la información de un recurso en un momento en el tiempo (una” imagen del estado del recurso”). Es la unión de metadatos y datos específicos que dan información sobre los datos incluidos (p.e: como desarrollarlos, informacion extra, entre otros.). En la Web, se solicita un recurso en HTML (para el navegador) o JSON (para ser utilizado por otros programas) intercambiando varios tipos de archivos ( multimedia, imágenes, textos, entre otros) por medio del MIME (Multipurpose Internet Mail Extensions ) que es un modelo para codificar representaciones.

***Hipermedia.*** Son vínculo y enlaces a los distintos recursos. Y esto es lo que permite la relación de recursos. Una restricción significativa de REST es la denominado como HATEOAS( Hypermedia as the Engine of the Application State ). Esta restricción admite que, mediante el uso del hiperenlace, se obtenga la representación de 11 recurso, que este contiene URIs de otros recursos con los que se puede interactuar.

***Comunicación***. La arquitectura de REST establece una estructura cliente-servidor, donde se obtiene una comunicación sincronizada. El componente cliente es el que empieza la comunicación, a través de una solicitud a los recursos del servidor. Esta solicitud incluye toda la información solo así el servidor puede procesarlas es decir peticiones autocontenidas. Estas solicitudes abarcan datos de control, metadatos y una representación para atender el contenido de la solicitud. Finalmente, el servidor recibe, procesa y devuelve una respuesta a su solicitud al cliente.

***Interfaz homogénea.*** Todos los recursos comparten la misma interfaz para así permitir su manejo. En el ejemplo de la Web, se usa como interfaz homogénea el protocolo HTTP, en el cual los métodos usados permiten hacer cambios en el estado de un recurso como: conocer el recurso empleado y en qué estado esta, y así logrando una representación (GET), crear un recurso (POST), actualizar la representación completa de un recurso (PUT), y parcialmente (PATCH) o eliminar un recurso (DELATE). Todos estos van a la URI de un recurso ( unas veces a la URI del recurso que va a ser cambiado, y otras a la URI de recurso que toma el papel de “ factoría” de recursos. De este modo, la API de una aplicación que se basa en un estilo arquitectónico está dividida en diversos recursos, donde cada uno son puntos de entrada a posibles peticiones de clientes (lo que permite la división de solicitudes entre diferentes recursos, que se encuentran en distintas localizaciones, todo esto de forma evidente mediante la utilización y descubrimiento de las URIs utilizando un interfaz homogéneo simple y común con todos los recursos). En general la arquitectura REST indica cómo se comporta una aplicación Web bien definida: un conjunto de páginas Web son los recursos que conforman una aplicación web (máquina de estados) donde los clientes (usuarios) navegan utilizando enlaces( transiciones de estado). La elaboración de las transiciones da la siguiente página al cliente (estado de la aplicación) que la procesa y la interpreta [13].

### Modelo de madurez de Richardson

Leonard Richardson propone un sistema de **Niveles de madurez de los Servicios Web REST** [10]

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Ilustración 7. Niveles de madurez de los Servicios Web REST

#### **Nivel 0**

Este nivel corresponde a los Servicios de Web tradicionales y gran parte, siendo el punto de partida en el uso de HTTP como sistema de transporte de las interacciones de forma remota, pero sin usar ningún tipo de mecanismo web. Fundamentalmente aquí se usa HTTP como un canal para trasmitir las interacciones entre los propios mecanismos, usualmente están basados en RIP(Remote Procedure Invocation).

Escala de tiempo

Descripción generada automáticamente

Ilustración 8. Nivel 0 de Madurez del Servicio Web REST

De este modo, se utiliza el protocolo de transporte HTTP transmitir documentos XML, siendo usados únicamente GET y POST para las interacciones remotas, sin una estructura de mensajes tipo SOAP y por ello una gran parte de estos servicios se denominan “RESTful”.

Un ejemplo clásico es la solicitud de cita

médica obteniendo como respuesta los huecos disponibles o un mensaje de error:

Texto, Carta

Descripción generada automáticamente

Texto, Carta

Descripción generada automáticamente

#### **Nivel 1**

Es la primera etapa para llegar a “the Glory of Rest”, para lograrlo se tiene como objetivo la identificación de los recursos a través de una URI, permitiendo lanzar peticiones a "recursos” (en REST, se llama así la información con la que se interactúa, sin importar en el formato en la que esté) individuales,en vez de usar un único punto de entrada,llegando a poder acceder a secciones o documentos del sitio Web usando las URIs.

En la denominación de una URI ha de cumplirse:

• No deben implicar acciones.

• Deben ser inequívocas. Una única para un mismo recurso.

• Deben de ser independientes del formato

• Deben mantener una lógica en la jerarquía.

• No deben de hacer filtrados de información de un recurso.

Imagen que contiene Gráfico

Descripción generada automáticamente

Ilustración 9. Nivel 1 de Madurez del Servicio Web REST

Por ejemplo:

URI “http://www.clinicaBertrana.com/doctores/mjones” representa a un doctor concreto

Siguiendo con el ejemplo del apartado del nivel 0

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

#### **Nivel 2**

En este nivel los servicios utilizan todos los métodos que ofrece HTTP, siguiendo de forma rigurosa el estándar creado por los desarrolladores REST donde se acordó: GET (accede a los datos de un recurso), POST(creando el recurso), PUT (modifica un recurso) y DELETE (elimina un recurso); Además están los códigos de estado(para poder saber la situación de la solución) y kos tipos de contenidos( que especifican el formato o formatos que sigue el recurso).

Imagen que contiene Diagrama

Descripción generada automáticamente

Ilustración 10. Nivel 2 de Madurez del Servicio Web REST

**)**

Ejemplos de los métodos:

* GET http://www. clinicaBertrana.com/ usuarios/jsmith
* DELETE http://www. clinicaBertrana.com/ usuarios/jsmith
* POST http://www. clinicaBertrana.com/ usuarios/jsmith
* PUT http://www. clinicaBertrana.com/ usuarios/jsmith

Siguiendo con ejemplo del apartado nivel 1, se intercambia GET por el POST, escondiendo los datos, ya que en vez de enviar los datos por la URL se envían en el cuerpo, pudiendo además ser estos cifrados y estructurados; También usa para actualizar el método PUT, dado que solo se actualiza un dato entero.

Imagen que contiene Logotipo

Descripción generada automáticamente Texto

Descripción generada automáticamente con confianza media Texto

Descripción generada automáticamente Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

#### **Nivel 3**

Es el último nivel, donde se habla del término “*Hypermedia controls*”, según el cual, tras al realizar una petición, la misma respuesta nos ofrece la información necesaria para comprender cómo utilizar el recurso. Para poder llegar a ese punto es necesario que los enlaces de los recursos presenten un “tipado” que le sea facil de entender al usuario donde la respuesta ofrece informacion adicional como enlaces a otros recursos ampliando las interacciones con estos.

Escala de tiempo

Descripción generada automáticamente

Ilustración 11. Nivel 3 de Madurez del Servicio Web REST

Siguiendo con el ejemplo que ha evolucionado pasando por los niveles, ahora en el caso del método POST, mandara información adicional que expone los recursos con sus URIs

Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

# Análisis

Investigación, análisis y requerimientos psicológicos del problema para la creación de la API.

## Análisis de dominio

La etapa universitaria es una de las experiencias más enriquecedoras de la vida de una persona, no sólo a nivel de formación en vista a un futuro laboral, sino también de crecimiento personal (madurez, independencia, etcétera). Muchos alumnos ingresan el primer año, pero su número se reduce considerablemente en el segundo año de carrera. Esto se debe al abandono universitario tras el primer año cursado.

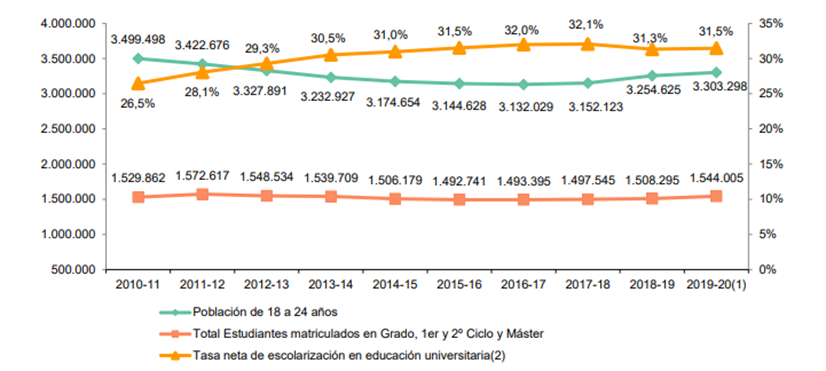


Ilustración 11.Población de matriculados en universidades.

Según la ilustración anterior, la tasa neta de escolarización en la educación universitaria sobre la población comprendida entre 18 y 24 años, se puede ver un interés creciente en el estudio de carreras universitarias, sin embargo, si nos fijamos en el año escolar 2016-17 se produce un leve descenso y estancamiento en el interés de los estudiantes, a pesar de ser más numerosa la franja de edad. Este dato se puede relacionar con la tabla que se mostrará a continuación.

Tabla

Descripción generada automáticamente

Tabla 1. Tasas de abandono en el primer año

*.*

Como se puede observar, tanto el porcentaje de abandono como el porcentaje de cambio de estudios sigue una progresión creciente. En relación al punto anterior, ese descenso del interés universitario se puede provocar por el aumento de la frustración o desinterés de esta, dando lugar a tan altas tasas de abandono o de cambio de estudios.

Tal y cómo se muestra en las dos tablas anteriores, la tasa de abandono en el primer año es un problema real. Esto se debe a muchos factores, como la falta de motivación, de tiempo, de planificación, etcétera. Del conjunto de causas principales este proyecto se centrará en el caso donde el estudiante no reúne las aptitudes y el grado de concentración necesarios para lograr obtener la titulación, ya que generan una frustración o desmotivación que le inducen a tomar la decisión de abandonar los estudios superiores, lo que podría solventarse con una buena elección académica antes de comenzar el primer año ofreciendo a los estudiantes la posibilidad de realizar una serie de test estandarizados sobre sus aptitudes, motivaciones, planificación, concentración, etcétera, que le sirvan como recomendaciones para elegir mejor la titulación que deberían estudiar.

Hay muchos tipos de instrumentos utilizados en el estudio psicológico entorno al estudiante, pero dado el alcance del proyecto, solo se realizaran dos tipos de test estandarizados : test de aptitudes y test de concentración; que serán informatizados.

El “*Test de Orientación Vocacional CHASIDE*” de Holland Ríase [1] es un test muy utilizado para evaluar las aptitudes que se basa en el psicoanálisis vocacional, y permite tomar una decisión según las aptitudes y los intereses del estudiante. Se trata de contestar a preguntas sencillas con Sí/No, donde a las respuestas afirmativas se le asigna 1 punto y las negativas 0 puntos, para después contabilizar todos los puntos mediante la tabla de valores que se presenta a continuación [2].

Imagen que contiene gabinete, reloj, diferente, pantalla

Descripción generada automáticamente

Tabla 2. Evaluación CHASIDE

El “*Test de Toulouse”* de E. Toulouse y H. Piéron [7] es un test muy utilizado para evaluar las aptitudes perceptivas y atencionales. Consiste en localizar una serie de figuras en un conjunto extenso de figuras similares, con el objetivo de medir la cantidad de aciertos, errores y omisiones. Una vez recogidos los datos, se pueden obtener:

* El Índice Global de Atención y Percepción (IGAP), constituye una medida de la capacidad perceptiva y atencional de los evaluados.
* El Cociente de Concentración (CC), mide la capacidad de concentración que tiene el usuario.
* El Índice de Control de la Impulsividad (ICI), informa sobre el nivel de impulsividad que tiene el usuario a encontrar las figuras.

Para la elaboración del proyecto se analizarán tanto los dos índices anteriores como el cociente de concentración

**IGAP= ACIERTOS – (ERRORES + OMISIONES)**

**CC = A – E / A + O**

**ICI = ACIERTOS – ERRORES / RESPUESTAS X 100**

Donde: A es acierto, E es error y O es omisión.

Patrón de fondo

Descripción generada automáticamente

Ilustración 12. Test de Toulouse

## Especificación de requisitos

En este apartado se detallarán los diferentes tipos de requisitos que cubrirá la aplicación, que son: funcionalidades del sistema, rendimiento, capacidad, seguridad, interoperabilidad con otros sistemas, protección de datos, requisitos sobre entorno tecnológico y de comunicaciones, requisitos funciones y no funcionales.

* *Funcionalidades del sistema:*

Mediante una interfaz proporcionada por la universidad, el estudiante accede, se registra con un usuario y contraseña. Comienza rellenando el primer formulario de CHASIDE que este clasifica por gustos e interés a los estudiantes a las carreras óptimas, después este estudiante puede seguir para una respuesta más específica en cada caso, mediante el formulario TOULOSE obtiene el nivel de concentración. Apoyándose en el análisis de ambos resultados se puede estimar con que facilidad va a poder llevar las materias académicas de cada grado.

Finalmente, el usuario podrá poner sus calificaciones obtenidas en el bachillerato para que la API pueda analizar todos estos datos dando los resultados con mayor precisión objetivamente , proporcionando como resultado si tiene alto o bajo rendimiento o aptitudes para dichos grados.

Una vez ingresan a la universidad, pueden ir insertando los resultados en los exámenes para seguir obteniendo recomendaciones, cada vez más precisas. *Rendimiento*

Los tiempos de respuesta de la aplicación son

* *Capacidad y eficiencia*

La API por defecto puede soportar como numero de 25, 125, 500 peticiones simultáneas por 1, 60, 360 segundos respectivamente con un tiempo de respuesta promedio de 1 segundo.

* *Interoperabilidad con otros sistemas*

Al tratarse de una API REST, posee una gran capacidad de comunicación entre distintos sistemas, ya que se encuentra almacenada en un servidor externo donde solo expone los recursos a través de las URIs, por lo tanto es ajena a como estén formados los sistemas que acceden a dichos recursos, siempre y cuando envíen y transcriban los datos en formato correcto( en este caso es JSON[14], ya que es usado en el intercambio de información entre ordenadores),y por ultimo como esta subida a AWS[15] es accesible desde distintos entornos.

* *Protección de datos*.

Según el reglamento (UE) 2016/679 [16], establece una serie de puntos donde los que más afectan a la API son:

1. Determinar las responsabilidades. En el caso de la API, contará con un responsable como con un encargado, que se encargaran de velar por el cumplimiento de la normativa.
2. Deber de informar. El responsable del tratamiento de los datos debe informar la duración y el modo en el que se van a emplear los datos del cliente al cliente.
3. Consentimiento inequívoco. El tratamiento de los datos se debe hacer mediante un consentimiento voluntario por parte del cliente, por ende, la empresa que haga uso de la API, debe establecer un contrato con los clientes para asegurar el cumplimiento del punto.
4. Medidas de seguridad y organizativas. El responsable del tratamiento de los datos debe establecer una serie de medidas para asegurar que los datos no se encuentran comprometidos, este apartado se verá más en profundidad en el apartado 4.3.
5. Evaluación del impacto. En base a la criticidad de los datos, como en el caso de la API no guarda datos sensibles, este apartado no es un problema.

* *Requisitos sobre entorno tecnológico y de comunicaciones.*

Los requisitos definidos por el cliente especifican que debe ser implementado como un servicio web, y que la comunicación se establezca por los métodos que ofrece HTTP, por ello, este proyecto se enfoca en la realización de una API RESTful, la cual se sirve de los métodos HTTP para interactuar con los recursos expuestos, y de la capacidad de interoperabilidad con otros sistemas que posee.

### *Requisitos funcionales*

1. El sistema permitirá registrarse mediante un usuario y contraseña.
2. Debe medir las aptitudes mediante un formulario estandarizado.
3. Debe medir la concentración con un formulario estandarizado.
4. Establecerá relaciones entre los resultados de los formularios estandarizados (test de aptitudes, de concentración, etcétera) para dar consejos en la planificación.
5. Debe tener un servicio donde se muestren las recomendaciones acerca de las elecciones del estudiante en cuanto a los estudios.
6. Debe tener un servicio para mostrar las materias cursadas el primer año.
7. Debe implementarse como un servicio web

* *Requisitos no funcionales*
* Los formularios deben estar estandarizados y con una base probada para aumentar su probabilidad de éxito.
* Debe ser accesible desde cualquier dispositivo (tablets, móviles, otras aplicaciones, etcétera).

## Análisis de seguridad

Este análisis uno de los primeros pasos a la hora de implantar el ENS, pues de esta categorización dependerán muchas de las medidas a implantar tanto del marco operacional como de las medidas de protección. Para ello, es importante definir cuáles son las dimensiones que abarca este análisis, que son, según la Guía de la Seguridad de las TIC [ver tabla 3]:

* **Integridad**: Las consecuencias asociadas a que un tercero no autorizado corrompa la información.
* **Confidencialidad**: Las consecuencias de revelar información a personas que no se encuentran dentro de los destinatarios, por ende, que no estén autorizadas a recibir dicha información.
* **Trazabilidad**: Las consecuencias de no poder descubrir qué persona ha accedido a un sistema o ha corrompido la información.
* **Disponibilidad**: Las consecuencias de que una persona autorizada no sea capaz de acceder a la información cuándo ésta la necesite.
* **Autenticidad**: Las consecuencias de que la información no fuera la producida en el origen, es decir, que se pueda suplantar.

Una vez definidas las dimensiones es importante destacar que existen diferentes niveles de impacto: bajo, medio y alto. Esto niveles son diferenciados por la criticidad, sensibilidad, etcétera, del sistema o servicio.

Los pasos seguidos para analizar los servicios son los siguientes:

1. Identificar los activos a proteger. En este caso sería la información sensible que se puede extraer en cada servicio.
2. Determinar cuál sería el impacto en función de cada dimensión.
3. Selección de las medidas de seguridad de acuerdo con las dimensiones de seguridad según el nivel de respuesta y su naturaleza.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Servicio/sistema | Integridad | Confidencialidad | Trazabilidad | Disponibilidad | Autenticidad |
| Login | ALTO | ALTO | ALTO | ALTO | ALTO |
| Petición de los formularios | MEDIO | BAJO | BAJO | MEDIO | MEDIO |
| Envió de respuestas formularios | ALTO | MEDIO | MEDIO | MEDIO | ALTO |
| Registro del progreso de estudios | MEDIO | MEDIO | ALTO | MEDIO | ALTO |
| Recomendaciones del estudio | ALTO | ALTO | BAJO | BAJO | MEDIO |
| Administración de credenciales | ALTO | ALTO | MEDIO | BAJO | ALTO |
| Administración de la información de los estudios del estudiante | ALTO | ALTO | ALTO | MEDIO | ALTO |

En base a la tabla anterior [ver tabla 3], se analizará la prevención y respuesta en cada nivel:

1. **Nivel alto**

Se puede ver una clara importancia de la integridad, autenticidad y confidencialidad.

* **Prevención**

Para abordar esta problemática se ha implementado el uso de tokens, que hacen la labor de transmitir datos sensibles. En esta aplicación se utiliza el token bearer, que es el encargado de cifrar una serie de datos sensibles, a los cuales sólo tiene acceso el origen, que pueden ser un dato oculto pactado previamente y único por cada cliente. De esta forma se asegura que el token es autentico e incorrupto, ya que al destransformarlo y compararlo con el dato almacenado no coincidiría. Este sistema también ofrece esa autenticidad, ya que al ser enlazado a un único cliente sólo puede conocerlo este mismo cliente, además del típico acceso mediante una contraseña segura, compuesta por caracteres numéricos, símbolos y no numéricos.

Por último, cabe destacar que toda información personal que se almacene en la base de datos estará cifrada.

* **Respuesta**

Debido a la criticidad del nivel, se optará por el apagado o la redirección del servicio en otro puerto según el nivel de riesgo, independientemente de la medida que se tomará anteriormente, y se procederá a un análisis y recuperación de datos, para determinar que mejora aplicar para reforzar la prevención.

1. **Nivel medio**

En este apartado sólo se encuentra la trazabilidad ya que, aunque se produzca una fuga de datos, al estar cifrados no serán de mucha utilidad al atacante.

* **Prevención**

Para prevenir esa sustracción de información, se establece una medida de doble autentificación (contraseña y token), y a su vez se proporcionará una cookie en el navegador del cliente, que expondrá quien ha usado el servicio, hora y día que fueron realizadas las operaciones.

* **Respuesta**

En el momento que se detecte una filtración se bloqueará el acceso a la persona implicada y al usuario afectado, y se le mandará un mensaje de cambio de credenciales.

1. **Nivel bajo**

Aunque sea un API REST y necesite estar disponible todo el tiempo, en la gran mayoría de casos no es necesario interactuar constantemente con ella, por eso en comparación con las demás partes, ésta es la menos importante.

* **Prevención**

La prevención de la desconexión de los clientes suscritos se encuentra alojada en un servidor de AWS, que es el encargado de establecer un entorno seguro, flexible e independiente al país de origen.

* **Respuesta**

En caso de haber picos de peticiones, se puede ampliar la capacidad del servidor de forma rápida, para adaptarse a dicha afluencia.

# Diseño

## Arquitectura del Sistema

Para poder cubrir el análisis del capítulo 4, se ha elegido una arquitectura que se divide en dos partes, la primera parte es el Backend donde se desarrolla la parte funcional de la API y la segunda parte un ligero Frontend donde se encuentra una interfaz simple que sirve como herramienta para acceder a la API.

Icono

Descripción generada automáticamenteLogotipo, nombre de la empresa

Descripción generada automáticamenteLogotipo

Descripción generada automáticamenteImagen que contiene dibujo, luz

Descripción generada automáticamente

Ilustración 13. Modelo de arquitectura Servicio Web RESTful

Respuestas HTTP

Solicitudes HTTP

Queries SQL

Datos

Datos

**BACKEND**

**API**

**BBDD**

**FRONTEND**

## Diseño de subsistema backend

Tomando como referencia la ilustración 13 , mostrada en el punto 1 de este capítulo, se definirá las tecnologías usadas en el Backend y las partes más importantes del mismo encargadas de realizar la funcionalidad de la API.

### Tecnologías utilizadas en el Backend

* **MySQL:** Usada como BD es la encargada de la gestión de los datos, dada su gran fiabilidad y agilidad. Además, cuenta con una diversidad de interfaces que soportan y permiten al usuario un entorno más amigable para gestionar los datos.
* **Java:** Se ha utilizado el lenguaje Java para el desarrollo de la API, debido a su difusión en los entornos informáticos (por lo tanto, se encuentra muy bien documentado), seguridad, fiabilidad y compatibilidad con Play Framework.
* **PLAY FRAMEWOK:** Play es un Framework de desarrollo Web para Java y Scala, por lo tanto, es muy útil para la finalidad de este proyecto, ya que facilita el desarrollo de la API, proporcionando un esqueleto y una configuración de URIs muy útiles y sencillas de comprender para el desarrollador.

### Diseño del servicio web RESTful

Para el diseño de este apartado, primero se han definido los recursos y sus atributos, obtenidos tras el análisis del dominio realizado en el capítulo 4, que son necesarios para el funcionamiento de la API, seguidamente de su definición, se han estructurado los patrones de las URIs para poder exponer cada recurso, terminando en el diseño de las tablas que indican los métodos, URIs, utilidad y códigos de respuesta para cada uno de los recursos, estas tablas se encuentran en el anexo [1].

#### Definición de los recursos

Sirviéndose del análisis del dominio y de los requisitos, se han identificado los siguientes recursos:

* Usuario: permite eliminar a un usuario en concreto, acceder a su información y actualizar la contraseña u otros datos.
* Usuarios: permite añadir un usuario nuevo, eliminar todos los usuarios como caso de emergencia u obtener todos los usuarios, estas dos últimas acciones solo se le permiten al responsable de la gestión de los datos.
* Nota: permite obtener la asignatura, el riesgo, la puntuación obtenida en ella, el tiempo de estudio recomendado y el tiempo de estudio dedicado, actualizar los datos de la nota.
* Notas: permite añadir una nota nueva, eliminar todas las notas como caso de emergencia u obtener todas las notas, estas dos últimas acciones solo se le permiten al responsable de la gestión de los datos.
* Formularios: permite añadir las respuestas de un formulario
* Formulario: permite obtener las preguntas de un formulario en concreto.

#### Atributos de los recursos

En este apartado se detallará a cada recurso definido anteriormente, explicando sus atributos y características más importantes.

#### Usuario:

La API requiere la creación de uno, para poder identificar y enlazar los datos necesarios parar realizar las recomendaciones personalizadas.

Los atributos que posee el usuario son:

* Nombre: es el identificador para el usuario, que este tendrá que añadir para poder realizar las recomendaciones.
* Contraseña: un mecanismo de seguridad, para que los datos de los usuarios sean individuales.
* Aptitudes: guarda los tipos de carreras eres más compatible
* Intereses: guarda los tipos de carreras eres más te pueden gustar
* Nivel de concentración: mide la capacidad de concentración

#### Nota:

Se encarga de almacenar el tiempo dedicado a cada asignatura y la puntuación obtenida.

Los atributos que posee la nota son:

* Asignatura: guarda el nombre de la asignatura cursada.
* Puntuación: almacena la nota de la asignatura.
* Tiempo de estudio: recoge el tiempo dedicado a la asignatura.
* Tiempo de estudio recomendado: guarda el tiempo sugerido.
* Riesgo: el nivel de riesgo para suspender
* Tipo: si es primer parcial, segundo o bachiller

#### Formulario:

Son los test estandarizados, encargados de analizar las aptitudes, capacidades y concentración del usuario, como ya se ha explicado anteriormente en mayor profundidad en capítulo 4 en el apartado de Análisis del dominio.

Los atributos que posee el formulario son:

* Tipo: guarda la información necesaria para identificar cuál de los dos formularios es.
* Respuestas: almacena las respuestas ingresadas por el usuario.
* Preguntas: recoge las preguntas del formulario según el tipo.

#### Patrones de las URIs

Gracias a la definición de las URIs se pueden mandar peticiones HTTP determinadas por los métodos y los recursos:

**/usuarios** -> Representa el conjunto de Usuarios

**/resultados** -> Representa el conjunto de resultados

**/notas** -> Representa el conjunto de notas

**/formularios** -> Representa el conjunto de formularios

**/usuarios/:id** -> Representa un usuario determinado

**/usuarios/:idUsuarios/notas** ->Representa el conjunto de notas de un resultado determinado de un usuario determinado

**/usuarios/:idUsuarios/notas/:id** -> Representa una nota determinada de un resultado determinado de un usuario determinado

**/usuarios/:idUsuarios/formularios** -> Representa un conjunto de formularios de un usuario determinado

**/formularios/:id** -> Representa un formulario determinado

#### API de cada recurso

En este apartado se mostrará un ejemplo de URIs definidas, métodos, utilidad del método, representación (formato de la información), código de respuesta, todo ello representado en formato tabla, del recuso Usuario[Tabla 3] definido anteriormente, los otros métodos se encuentran en el anexo [1].

Tabla

Descripción generada automáticamente

Tabla 3. Exposición del recurso: usuarios

#### Representaciones utilizadas

En este proyecto se ha optado por usar JSON[14] para las representaciones de los recursos, dado que permite ver los atributos asociado a este con claridad y sencillez.

### Diseño de la base de datos

Diagrama

Descripción generada automáticamente

Ilustración 14. Diagrama E/R de la base de datos tfg

Para poder entender el diagrama de la Ilustración 14 a continuación se definirán el significado cada elemento

• Llave amarilla: indica PRIMARY KEY.

• Rombo rojo: indica FOREIGN KEY.

• Rombo azul: indica que la columna no acepta valores NULL.

• Rombo blanco: indica que la columna si acepta valores NULL.

La relación entre columnas significa:

Imagen que contiene Gráfico

Descripción generada automáticamente

Ilustración 15. tDiagrama E/R tipos de relaciones

Las tablas “nota” y “formulario” poseen la columna “url” con la intención de acercarse al nivel 3 del ***Modelo de madurez de Richardson***explicado en el capítulo 3, un ejemplo sería: /usuario/1/formulario/1 esta URI quedaría almacenada en la tabla de formulario haciendo referencia al recurso concreto.

En algunos casos se ha procedido a la creación de tablas, cuya función es solventar el problema de que un objeto lleva como atributo otro objeto , véase el caso del formulario, que en sus atributos posee una lista de preguntas y respuestas, para abordar este problema en concreto se ha creado la tabla preguntas y otra para respuestas, donde respuesta está asociada al formulario y la pregunta está asociada a la respuesta .

A continuación, se detallará las tablas creadas y sus funciones:

* Usuario: permite almacenar a los usuarios registrados con todos sus atributos
* Nota: almacena tanto el nombre de la asignatura, como la puntuación, el tiempo de estudio, tiempo de estudio sugerido, riesgo
* Formulario: almacena solo el tipo y la url siendo el caso especial mencionado anteriormente
* Respuesta: almacena las respuestas del formulario
* Pregunta: almacena las preguntas del formulario

## Diseño del subsistema frontend

# Implementación

## Implementación de backend

Para poder implementar el diseño del servicio Web RESTful se ha utilizado PLAY Framework y Java descritos en el punto 5.2.1. A su vez la estructura de La API se encuentra dividida en seis partes: Beans, Controllers, Model, Services, Utils, Routes.

* Beans: son las semillas/entidades que representan al recurso, necesarias para la transmisión y manipulación del recurso.
* Controllers: son las clases encargadas de atender las peticiones HTTP.
* Model: es la parte lógica de la API, en este caso, se encarga de realizar las sugerencias en base a los datos del usuario.
* Services: clases encargadas de conectarse a la base de datos mediante Querys, además de aislar las llamadas de la base de datos.
* Utils: encargada de crear el response adaptado para devolver JSON.
* Routes: es el archivo donde se declaran las URIs y los métodos a los que están asociados.

### Estructura e implementación de la lógica de negocio con Play Framework

En este apartado se expondrá la estructura e implementación de cada una de las partes mencionadas en el apartado anterior y un ejemplo asociado para facilitar la comprensión .

#### Beans

En el paquete Beans se encuentran los recursos (definidos en el punto 5.2.2.1) definidos como clases Java. Estas clases cumplen con la funcionalidad de convertir esos recursos en objetos, pudiendo ser tratados y gestionados desde la API.

Pantalla de computadora

Descripción generada automáticamente

Ilustración 16, Diagrama de clases UML

Como se puede observar, las clases “Formulario”, “Usuario” y “Nota” extienden de “RecursoWeb” (línea azul). Esto se debe a que “RecursoWeb” tiene los dos atributos principales para que pueda llegarse al nivel 3 del ***Modelo de madurez de Richardson*** , ya que posee el atributo “url” que es el encargado de gestionar la ruta de acceso al recurso, que posteriormente será usado en las peticiones HTTP para que el usuario pueda a través de un recurso navegar entre sus relaciones.

También es necesario mencionar a las clases “RespuestaFormulario” y “PreguntaFormulario”, que surgieron como casos especiales y esenciales ya que se encargaran de representar y por ende administrar las respuestas y preguntas respectivamente del formulario.

#### Controllers

Las clases dentro del paquete “controllers” se encargan de recibir las llamadas que realiza el usuario sobre la API, para que mediante el uso de los “Beans” descritos anteriormente, se conviertan en objetos que los datos puedan ser administrados, analizados o transmitidos.

En este paquete se encuentran las clases análogas a los “Beans” que se encuentran expuestos en las URIs, es decir, las clases dentro del paquete son: “FormularioControler”, “NotaControler” y “UsuarioControler”.

A continuación, mostraré como se trata la llamada al recurso usuario mediante el método POST de HTTP, esta llamada lanzara el método créate dentro de la clase “UsuarioControler”.

Texto

Descripción generada automáticamente

Ilustración 17. Método "create" de la clase "UsuarioController"

El método "create" recibe como parámetro de entrada una request HTTP que representan los datos enviados por parte del usuario en formato JSON, en caso de que el JSON se encuentre vacío, se mandará al cliente un mensaje de que el JSON no posee datos, en el caso de que si posee datos, se transcribirán a la clase “Usuario”, mediante la función que posee la librería nativa de PLAY.



Ilustración 18. libreria JSON de PLAY

Una vez transcritos a la clase “Usuario”, se utilizará el método “addUsuario” de la clase “UsuarioBBDD” (que se expondrá posteriormente en el apartado 6.2.1.4 ), añadiendo este usuario a la base de datos.

Como respuesta enviará dicho usuario al cliente que realizo la llamada, en el cual contendrá el url para que este sea identificado y accesible para el cliente.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Ilustración 19. Llamada al recurso "Usuario" mediante el método POST

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Ilustración 20. Respuesta de la API

#### Model

En esta clase se encuentra el algoritmo encargado de realizar las sugerencias a través de las respuestas de los test estandarizados y las notas, recibidas por parte del usuario, estas sugerencias se almacenarán junto con la nota del usuario, y los resultados de los formularios se asociaran al usuario.

#### Services

Dentro de este punto nos encontramos las clases incluidas en el paquete “services”, las cuales se encargarán gestionar de las conexiones a la BBDD, realizando acciones tales como: inserción, actualización, consulta o borrado de los datos.

Estas conexiones se realizan a través de la clase “ConexionBBDD” encargada de abrir o cerrar las conexiones para poder realizar las acciones mencionadas anteriormente, mediante el uso del driver de MySQL[5.2.1] “com.mysql.jdbc.Driver”.

Texto

Descripción generada automáticamente

Ilustración 21. Uso del driver MySQL

Cabe destacar que la clase “ConexionBBDD”, también se encarga de crear la base de datos y de la información necesaria para que la API pueda realizar sus funciones correctamente.

Continuando con el ejemplo expuesto en el apartado 6.2.1.1, se mostrará a continuación el método “addUsuario”.

Texto

Descripción generada automáticamente

Ilustración 22. Método "addUsuario"

Este método recibe como entrada el objeto “Usuario” que contiene toda la información del usuario, necesaria para la inserción en la base de datos.

Haciendo uso de la función “createStatement.executeUpdate” puedo mandar a la base de datos la Query encargada de insertar los datos del objeto usuario en la base de datos, a su vez este método devolverá el “idUsuario” autogenerado por ella, que servirá para completar la “url” que posteriormente se introducen mediante otra Query a la base de datos.

Además, cuando se genera el usuario, ya se le asocian sus dos test estandarizados vacíos, donde solo se encuentra asociados a ellos las preguntas de los mismos.

Tras la llamada al recurso "Usuario" mediante el método POST el usuario se encuentra en la base de datos como se muestra en la siguiente ilustración.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Ilustración 23. Usuario insertado en la base de datos

#### Routes

Este es un fichero de configuración proporcionado por PLAY donde se describen las URIs, los métodos HTTP y el controlador asociado a cada llamada.

Siguiendo con el ejemplo de Usuario, se mostrará en la siguiente ilustración la configuración del recurso usuario, donde se encuentra detallada la llamada analizada anteriormente.

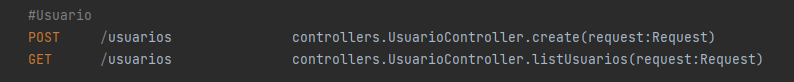


Ilustración 24. Fichero "routes", recurso usuarios

### Estructura e implementación de la BBDD

## Implementación de frontend

No posee una implementación al uso al tratarse de una API como queda detallado en el apartado de diseño de frontend.

## Entornos de despliegue

El proyecto se apoya en dos infraestructuras principalmente: el ordenador personal el cual se encarga de alojar la base de datos y la raspberry la cual almacena es servidor de servicios, que en su interior se encuentra el api REST encargada de exponer mediante el navegador los diferentes recursos que son gestionados por los diferentes servicios.

Diagrama

Descripción generada automáticamente

Ilustración 16. Primera versión de la arquitectura

Este entorno solo fué utilizado para el entorno de pruebas dado que la versión final será similar al entorno que se encontrará una vez que lo utilice el cliente.

Diagrama

Descripción generada automáticamente

Ilustración 17. Versión final

El segundo esquema muestra que el servidor se encuentra alojado en los servidores de Amazon, donde en ellos se encontrará ejecutándose la API a la espera de recibir las peticiones de los recursos, para poder ofrecer los datos, la API los obtendrá de las propias bases de datos del cliente, como se puede observar a continuación.

## Referencia al repositorio de software

El repositorio donde se encuentra el trabajo de fin de grado es:

<https://github.com/davidRecio/tfgFinal>

## Manuales

Los manuales del usuario se encuentran en el directorio /Doc, bajo el nombre “manualUsuario” y los JSON de ejemplo se encuentran en el directorio /JSON, por último, el manual de instalación se encuentra en el fichero README.md .

# Pruebas y validación

# Conclusiones y líneas futuras

## Conclusiones

Este TFG parte de la demanda de estudiantes que quieren acceder a las universidades, pero no tienen suficiente orientación vocacional y de la deserción en el primer año de carrera para ello es una herramienta que facilita la gestión para ver cual la mejor opción de grado.

Para ello, en primer lugar, se realizó un estudio para ver la viabilidad del proyecto y el impacto que tendría en los nuevos ingresos universitarios, con datos estadísticos de los estudiantes del primer año que desertaron de la carrera siendo el factor de escoger el grado equivocado el principal motivo. Por ende, este proyecto facilita a los estudiantes mediante sus gustos y sus notas académicas ver que carrera les vendría mejor y se les facilitara hacerla.

Después, se hizo un análisis de dominio y se vio los requisitos necesarios que debía cumplir la API, se analizaron los formularios principales y necesarios testeados anteriormente y con datos asertivos, para así poder cubrir todas las necesidades del usuario que requiera al ingresar a la API y buscar sus decisiones más asertivas.

Para la realización del proyecto se escogió por desarrollar un servicio Web RESTful ya que es un sistema de fácil manejo de los recursos remotos y es óptimamente eficiente en su uso. Fue la mejor opción ya que permite una buena portabilidad entre distintas plataformas y escalar aplicaciones sin mucha complejidad en los desarrolladores.

La parte de la lógica de la aplicación se realizó por medio de Play Framework, ya que esta permite trabajar sobre una estructura que ayuda a simplificar la creación y la ejecución de APIs en el servicio Web.

Se utilizó una BD para el almacenamiento de todos los registros y datos relacionados con los formularios. La implementación del proyecto se hizo a través de MySQL, más detallado en el capítulo 5.

El correcto funcionamiento del Servicio Web que fue creado se probó mediante Postman, ya que es una herramienta que consumen muy pocos recursos y nos permite probar los métodos utilizados para este Servicio Web como GET, POST, DELATE y PATCH.

Durante el desarrollo del proyecto se dificultó por varios problemas tal como la conexión inicial a la base de datos a través de los drives que ofrece Framework.

## Líneas futuras

En un futuro, este programa puede llegar a implementarse en un servidor de la universidad, utilizando sus propios formularios específicos. La universidad se haría cargo de la seguridad y disponibilidad de la API, mientras que yo sólo me debería encargar del mantenimiento y soporte, en caso de introducir nuevas funcionalidades. Otra opción sería implementar el programa dentro de un contenedor Docker, simplificando la instalación y el mantenimiento, para adaptarse a cualquier entorno en la nube, como los que ofrece Amazon Web Service o cualquier entorno propio, por tanto, de esa manera la aplicación no quedaría ligada a un solo cliente.

Respecto a funcionalidades nuevas, se podría hacer hincapié en nuevas áreas del estudiante, no únicamente centrado en la elección de una carrera. Podrían analizarse otras vías, como la parte económica (para aquellos que viven en residencias de estudiantes), o incluso fomentar la enseñanza, mediante tutorías entre estudiantes o cursos de apoyo.

## Cumplimiento de objetivos

El proyecto tiene como finalidad cumplir los objetivos planteados anteriormente, los cuales volveremos a mostrar en este apartado, junto con los requisitos asociados a ellos:

**Objetivo 1:** El sistema será capaz de realizar una valoración de las aptitudes del estudiante, y de su concentración mediante el análisis de los resultados de unos formularios estandarizados, para realizar recomendaciones sobre la elección de la titulación.

Requisitos asociados:

### *Requisitos funcionales:*

1. El sistema permitirá registrarse mediante un usuario y contraseña.
2. Debe medir las aptitudes mediante un formulario estandarizado.
3. Debe medir la concentración con un formulario estandarizado.

### *Requisitos no funcionales:*

* Los formularios deben estar estandarizados y con una base probada para aumentar su probabilidad de éxito.

Primero, es necesario que el usuario se registre para poder crear un perfil al que se asociar los datos que requiere la API para su correcto funcionamiento, para ello se creó el recurso “usuario” el cual recoge el usuario y la contraseña para identificarle.

Por otro lado, como ya se explicó en el punto de ***Análisis de dominio*** los test son estandarizados y probados, se encuentran asociados al recurso “formulario” el cual gestionara tanto las preguntas como las respuestas del mismo.

**Objetivo 2:** El sistema será capaz de realizar una planificación de tiempos de estudio, mediante recomendaciones de los datos obtenidos anteriormente.

Requisitos asociados:

### *Requisitos funcionales:*

1. Establecerá relaciones entre los resultados de los formularios estandarizados (test de aptitudes, de concentración, etcétera) para dar consejos en la planificación.
2. Debe tener un servicio para mostrar las materias cursadas el primer año.
3. Debe medir la concentración con un formulario estandarizado.
4. Debe tener un servicio donde se muestren las recomendaciones acerca de las elecciones del estudiante en cuanto a los estudios.

* *Requisitos no funcionales*
* Los formularios deben estar estandarizados y con una base probada para aumentar su probabilidad de éxito.

Para poder establecer dichas recomendaciones el usuario primero debe realizar los test, una vez realizados se le asignará al usuario sus aptitudes, intereses y el nivel de concentración, datos que se emplearan en el momento que ingrese las notas el usuario, terminando por generar las sugerencias de horas de estudio y el riesgo de la asignatura.

Por último, queda el objetivo 3 y unos pocos requisitos que, a pesar de no estar relacionados intrínsecamente con los objetivos, fueron requeridos por parte del cliente.

**Objetivo 3:** El sistema buscará fomentar la colaboración entre estudiantes y el aprendizaje grupal, para reforzar las bases de conocimiento de los compañeros de segundo, mediante el apoyo a los estudiantes de primero.

Dado que mostrará los niveles de riesgo en las diferentes asignaturas, los usuarios pueden buscar a otros usuarios entre los compañeros para ayudarse entre ellos para superar las dificultades de la carrera universitaria.

### *Requisitos funcionales*

1. Debe implementarse como un servicio Web

* *Requisitos no funcionales*
* Debe ser accesible desde cualquier dispositivo (tablets, móviles, otras aplicaciones, etcétera).

Respecto a los requisitos funcionales y no funcionales, están realizados desde un principio pues la idea principal es el desarrollo de un servicio Web y por la misma definición de servicio Web, este es independiente a los sistemas que consumen sus recursos.

# Bibliografía

1. **¡Error! Referencia de hipervínculo no válida.**Martha Báez. Holland Ríase. Test de orientación vocacional CHASIDE (2007). Academia Edu <https://www.academia.edu/10367175/Test_De_Orientaci%C3%B3n_Vocacional_Chaside>
2. E. Toulouse y H. Piéron. TEA Ediciones (1978, 2004, 2013). TP-R. Toulouse-Piéron-Revisado, prueba perceptiva y de atención. <https://web.teaediciones.com/Ejemplos/Extracto_libro_TP-R.pdf>
3. DR. Winston W. Royce (1970) Managin the development of large software systems. <https://www.praxisframework.org/files/royce1970.pdf>
4. David Booth, Hugo Haas, Francis McCabe, Eric Newcomer, Michael Champion, Chris Ferris, David Orchard (W3C Working Group Note 11 February 2004) Web Services Architecture <https://www.w3.org/TR/ws-arch/#whatis>
5. Centro Criptológico Nacional. (mayo de 2020). Guía de Seguridad de las TIC CCN-STIC 803. <https://www.ccn-cert.cni.es/series-ccn-stic/800-guia-esquema-nacional-de-seguridad/682-ccn-stic-803-valoracion-de-sistemas-en-el-ens-1/file.html>
6. Roger S. Pressman, Ph.D.University of Connecticut. Ingeniería del software, Un enfoque práctico , 33–35.

<http://cotana.informatica.edu.bo/downloads/ld-Ingenieria.de.software.enfoque.practico.7ed.Pressman.PDF>

1. David Booth, Hugo Haas, Francis McCabe, Eric Newcomer, Michael Champion, Chris Ferris, David Orchard (W3C Working Group Note 11 February 2004) Web Services Architecture <https://www.w3.org/TR/ws-arch/#SOAP> **¡Error! Referencia de hipervínculo no válida.**
2. Ministerio de Universidades (Publicación 2020-2021).

Datos y cifras del Sistema Universitario Español. Página 45 <https://www.universidades.gob.es/stfls/universidades/Estadisticas/ficheros/Datos_y_Cifras_2020-21.pdf>

1. Ministerio de Universidades (Publicación 2020-2021).

Datos y cifras del Sistema Universitario Español. Página 26 <https://www.universidades.gob.es/stfls/universidades/Estadisticas/ficheros/Datos_y_Cifras_2020-21.pdf>

1. **¡Error! Referencia de hipervínculo no válida.**Remote Procedure Invocation. Enterprise Integration Patterns https://www.enterpriseintegrationpatterns.com/EncapsulatedSynchronousIntegration.html
2. Fielding R. (2000) Architectural Styles and the Design of Network-based Software Architectures. Thesis Doctoral. University of California, Irvine
3. Fielding, Taylor (2017) Reflections on the REST architectural style and "principled design of the modern web architecture" (impact paper award) Proceeding’s 11th Joint Meeting on Foundations of Software Engineering. <https://doi.org/10.1145/3106237.3121282>
4. Fielding, Taylor (2002) Principled Design of the Modern Web Architecture. ACM Transactions on Internet Technology, Vol. 2, No. 2Pages 115–150.
5. JSON – Introduction.W3C.https://www.w3schools.com/js/js\_json\_intro.asp
6. Amazon. Informática en la nube con AWS. https://aws.amazon.com/es/what-is-aws/
7. Diario Oficial de la Unión Europea. REGLAMENTO (UE) 2016/679 DEL PARLAMENTO EUROPEO Y DEL CONSEJO  
   de 27 de abril de 2016.https://www.boe.es/doue/2016/119/L00001-00088.pdf

**¡Error! Referencia de hipervínculo no válida.¡Error! Referencia de hipervínculo no válida.¡Error! Referencia de hipervínculo no válida.¡Error! Referencia de hipervínculo no válida.¡Error! Referencia de hipervínculo no válida.¡Error! Referencia de hipervínculo no válida.¡Error! Referencia de hipervínculo no válida.**

# 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

## Definición de los recursos de la API

|  |  |  |  |
| --- | --- | --- | --- |
| **Recurso** | **URI** | **Funcionalidad** | **Métodos soportados** |
| Usuario | /usuarios | Representa el conjunto de Usuarios | POST,GET,DELETE |
| Resultados | /resultados | Representa el conjunto de resultados | GET,DELETE |
| Notas | /notas | Representa el conjunto de notas | GET,DELETE |
| Recomendaciones | /recomendaciones | Representa el conjunto de recomendaciones | GET,DELETE |
| Formularios | /formularios | Representa el conjunto de formularios | DELETE |
| Usuarios/id | /usuarios/:id | Representa un usuario determinado | GET,PUT,DELETE,PATCH |
| Usuarios/idUsuarios/resultados | /usuarios/:idUsuarios/resultados | Representa el conjunto de resultados de un usuario determinado | POST,GET,DELETE |
| Usuarios/idUsuarios/resultados /id | /usuarios/:idUsuarios/resultados/:id | Representa un resultado determinado de un usuario determinado | GET, DELETE,PATCH |
| Usuarios/idUsuarios/resultados/  idResultados/notas | /usuarios/:idUsuarios/resultados/  :idResultados/notas | Representa el conjunto de notas de un resultado determinado de un usuario determinado | POST,GET,DELETE |
| Usuarios/idUsuarios/resultados  /idResultados/notas/id | /usuarios/:idUsuarios/resultados/  :idResultados/notas/:id | Representa una nota determinada de un resultado determinado de un usuario determinado | GET, DELETE,PATCH |
| Usuarios/idUsuarios/resultados  /idResultados/recomendaciones | /usuarios/:idUsuarios/resultados/  :idResultados/recomendaciones | Representa el conjunto de recomendaciones de un resultado determinado de un usuario determinado | GET,DELETE |
| Usuarios/idUsuarios/formularios | /usuarios/:idUsuarios/formularios | Representa un conjunto de formularios de un usuario determinado | POST |
| Formularios/id | /formularios/:id | Representa un formulario determinado | GET |

## Métodos HTTP de los recursos

Recurso /usuarios

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Método | URI | Utilidad | Representación | Código Respuesta |
| POST | /usuarios | Añade los usuarios | JSON | 200-OK |
| 400-Bad Request |
| 500-Internal Server Error |
| GET | /usuarios | Obtiene todos los usuarios | JSON | 200-OK |
| 500-Internal Server Error |
| PUT | /usuarios | - | - | 404- Not Found |
| DELETE | /usuarios | Borra todos los usuarios | - | 200-OK |
| 500-Internal Server Error |
| PATCH | /usuarios | - | - | 404-Not Found |

Recurso /resultados

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Método | URI | Utilidad | Representación | Código Respuesta |
| POST | /resultados | - | - | 404-Not Found |
| GET | /resultados | Obtiene todos los resultados | JSON | 200-OK |
| 500-Internal Server Error |
| PUT | /resultados | - | - | 404-Not Found |
| DELETE | /resultados | Borra todos los resultados | JSON | 200-OK |
| 500-Internal Server Error |
| PATCH | /resultados | - | - | 404-Not Found |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Método | URI | Utilidad | Representación | Código Respuesta |
| POST | /notas | - | - | 404-Not Found |
| GET | /notas | Obtiene las notas de todos | JSON | 200-OK |
| 500-Internal Server Error |
| PUT | /notas | - | - | 404-Not Found |
| DELETE | /notas | Borra todas las notas de todos | - | 200-OK |
| 500-Internal Server Error |
| PATCH | /notas | - | - | 404-Not Found |

Recurso /notas

Recurso /recomendaciones

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Método | URI | Utilidad | Representación | Código Respuesta |
| POST | /recomendaciones | - | - | 404-Not Found |
| GET | /recomendaciones | Obtiene las recomendaciones de todos | JSON | 200-OK |
| 500-Internal Server Error |
| PUT | /recomendaciones | - | - | 404-Not Found |
| DELETE | /recomendaciones | Borra todas las recomendaciones de todos | - | 200-OK |
| 500-Internal Server Error |
| PATCH | /recomendaciones | - | - | 404-Not Found |

Recurso /formularios

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Método | URI | Utilidad | Representación | Código Respuesta |
| POST | /formularios | - | - | 404-Not Found |
| GET | /formularios | - | - | 404-Not Found |
| PUT | /formularios | - | - | 404-Not Found |
| DELETE | /formularios | Borra todos los formularios de todos | - | 200-OK |
| 500-Internal Server Error |
| PATCH | /formularios | - | - | 404-Not Found |

Recurso /usuarios/id

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Método | URI | Utilidad | Representación | Código Respuesta |
| POST | /usuarios/:id | - | - | 404-Not Found |
| GET | /usuarios/:id | Obtiene todos los datos de un usuario determinado | JSON | 200-OK |
| 500-Internal Server Error |
| PUT | /usuarios/:id | Actualiza todos los datos del usuario | JSON | 200-OK |
| 400-Bad Request |
| 500-Internal Server Error |
| DELETE | /usuarios/:id | Borra al usuario | - | 200-OK |
| 500-Internal Server Error |
| PATCH | /usuarios/:id | Actualiza la contraseña del usuario | JSON | 200-OK |
| 400-Bad Request |
| 500-Internal Server Error |

Recurso /usuarios/idUsuarios/resultados

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Método | URI | Utilidad | Representación | Código Respuesta |
| POST | /usuarios/:idUsuarios/resultados | Añade un resultado con una temática determinada, P.ej “parcial” el cual recibirá las notas y recomendaciones | JSON | 200-OK |
| 400-Bad Request |
| 500-Internal Server Error |
| GET | /usuarios/:idUsuarios/resultados | Obtiene todos los resultados de un usuario | JSON | 200-OK |
| 500-Internal Server Error |
| PUT | /usuarios/:idUsuarios/resultados | - | - | 404-Not Found |
| DELETE | /usuarios/:idUsuarios/resultados | Borra todos los resultados de un usuario en concreto | JSON | 200-OK |
| 400-Bad Request |
| 500-Internal Server Error |
| PATCH | /usuarios/:idUsuarios/resultados | - | - | 404-Not Found |

Recurso /usuarios/idUsuarios/resultados/id

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Método | URI | Utilidad | Representación | Código Respuesta |
| POST | /usuarios/:idUsuarios/resultados/:id | - | - | 404-Not Found |
| GET | /usuarios/:idUsuarios/resultados/:id | Obtiene el tipo de un resultado en concreto | JSON | 200-OK |
| 500-Internal Server Error |
| PUT | /usuarios/:idUsuarios/resultados/:id | - | - | 404-Not Found |
| DELETE | /usuarios/:idUsuarios/resultados/:id | Borra un resultado en concreto | - | 200-OK |
| 500-Internal Server Error |
| PATCH | /usuarios/:idUsuarios/resultados/:id | - | JSON | 200-OK |
| 400-Bad Request |
| 500-Internal Server Error |

Recurso /usuarios/idUsuarios/resultados/idResultados/notas

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Método | URI | Utilidad | Representación | Código Respuesta |
| POST | /usuarios/:idUsuarios/resultados/  :idResultados/notas | Añade las notas para un usuario en concreto | JSON | 200-OK |
| 400-Bad Request |
| 500-Internal Server Error |
| GET | /usuarios/:idUsuarios/resultados/  :idResultados/notas | Obtiene todas las notas de un resultado en concreto de un usuario en concreto | JSON | 200-OK |
| 500-Internal Server Error |
| PUT | /usuarios/:idUsuarios/resultados/  :idResultados/notas | - | - | 404-Not Found |
| DELETE | /usuarios/:idUsuarios/resultados/  :idResultados/notas | Borra las notas de un resultado en concreto para un usuario en concreto | - | 200-OK |
| 500-Internal Server Error |
| PATCH | /usuarios/:idUsuarios/resultados/  :idResultados/notas | - | - | 404-Not Found |

Recurso /usuarios/idUsuarios/resultados/idResultados/notas/id

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Método | URI | Utilidad | Representación | Código Respuesta |
| POST | /usuarios/:idUsuarios/resultados/  :idResultados/notas/:id | - | - | 404-Not Found |
| GET | /usuarios/:idUsuarios/resultados/  :idResultados/notas/:id | Obtiene una nota en concreto | JSON | 200-OK |
| 500-Internal Server Error |
| PUT | /usuarios/:idUsuarios/resultados/  :idResultados/notas/:id | - | - | 404-Not Found |
| DELETE | /usuarios/:idUsuarios/resultados/  :idResultados/notas/:id | Borra una nota en concreto | - | 200-OK |
| 500-Internal Server Error |
| PATCH | /usuarios/:idUsuarios/resultados/  :idResultados/notas/:id | Actualiza el valor de la nota | JSON | 200-OK |
| 400-Bad Request |
| 500-Internal Server Error |

Recurso /usuarios/idUsuarios/resultados/idResultados/recomendaciones

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Método | URI | Utilidad | Representación | Código Respuesta |
| POST | /usuarios/:idUsuarios/resultados/  :idResultados/recomendaciones | - | - | 404-Not Found |
| GET | /usuarios/:idUsuarios/resultados/  :idResultados/recomendaciones | Obtiene todas las recomendaciones de un usuario | JSON | 200-OK |
| 500-Internal Server Error |
| PUT | /usuarios/:idUsuarios/resultados/  :idResultados/recomendaciones | - | - | 404-Not Found |
| DELETE | /usuarios/:idUsuarios/resultados/  :idResultados/recomendaciones | Borra todas las recomendaciones de un usuario | JSON | 200-OK |
| 500-Internal Server Error |
| PATCH | /usuarios/:idUsuarios/resultados/  :idResultados/recomendaciones | - | - | 404-Not Found |

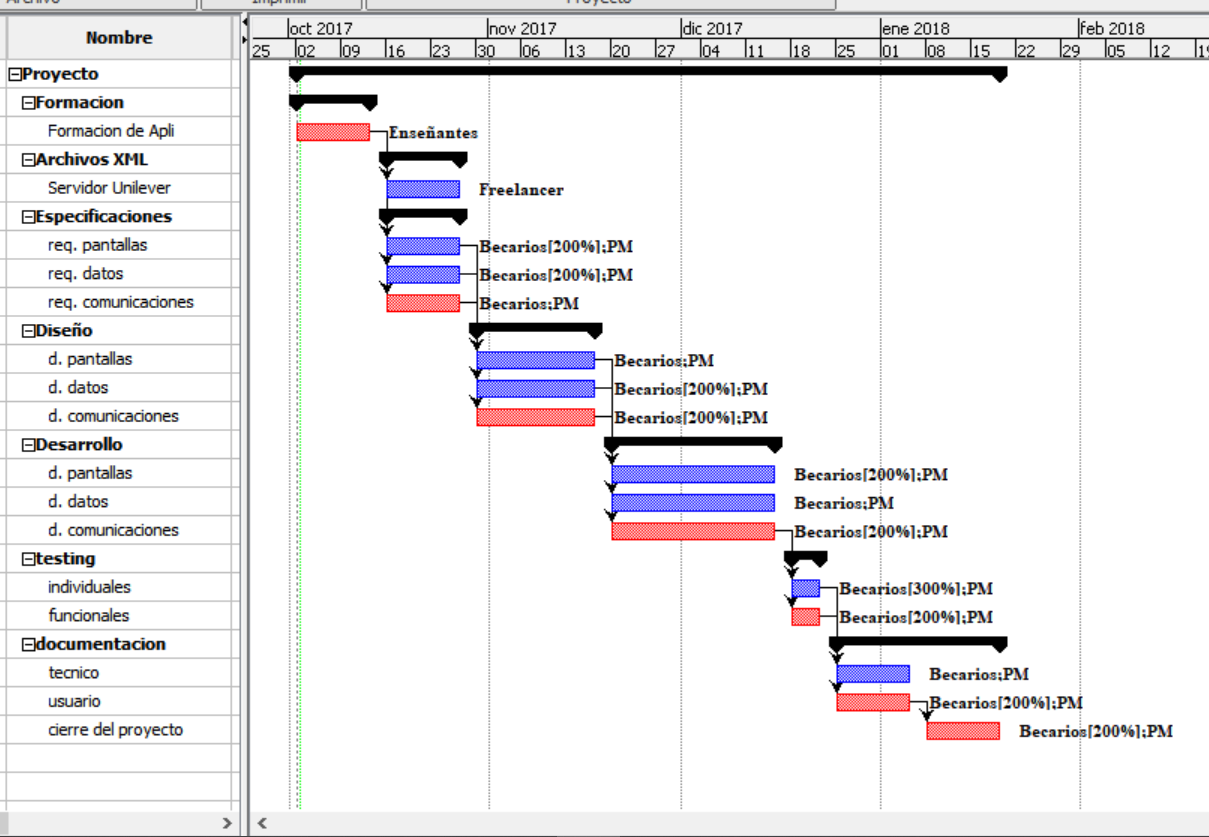
Recurso /usuarios/idUsuarios/formularios

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Método | URI | Utilidad | Representación | Código Respuesta |
| POST | /usuarios/:idUsuarios/formularios | Crea un nuevo formulario completo | JSON | 200-OK |
| 400-Bad Request |
| 500-Internal Server Error |
| GET | /usuarios/:idUsuarios/formularios | - | - | 404-Not Found |
| PUT | /usuarios/:idUsuarios/formularios | - | - | 404-Not Found |
| DELETE | /usuarios/:idUsuarios/formularios | - | - | 404-Not Found |
| PATCH | /usuarios/:idUsuarios/formularios | - | - | 404-Not Found |

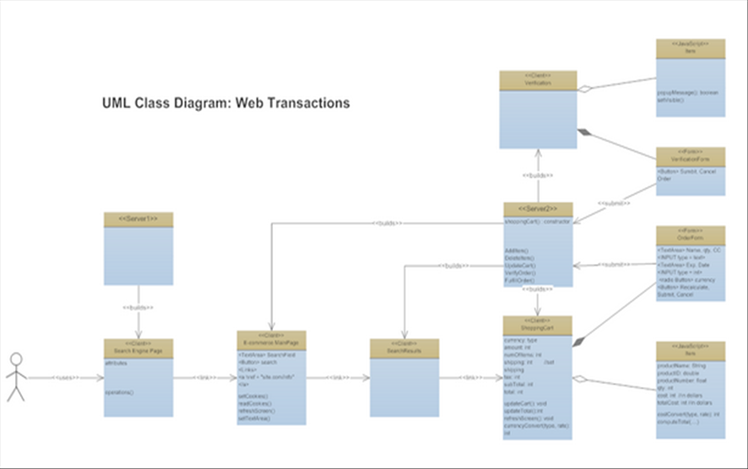
Recurso /formularios/id

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Método | URI | Utilidad | Representación | Código Respuesta |
| POST | /formularios/:id | - | JSON | 404-Not Found |
| GET | /formularios/:id | Obtiene las preguntas de un formulario en concreto | - | 200-OK |
| 500-Internal Server Error |
| PUT | /formularios/:id | - | - | 404-Not Found |
| DELETE | /formularios/:id | - | - | 404-Not Found |
| PATCH | /formularios/:id | - | - | 404-Not Found |

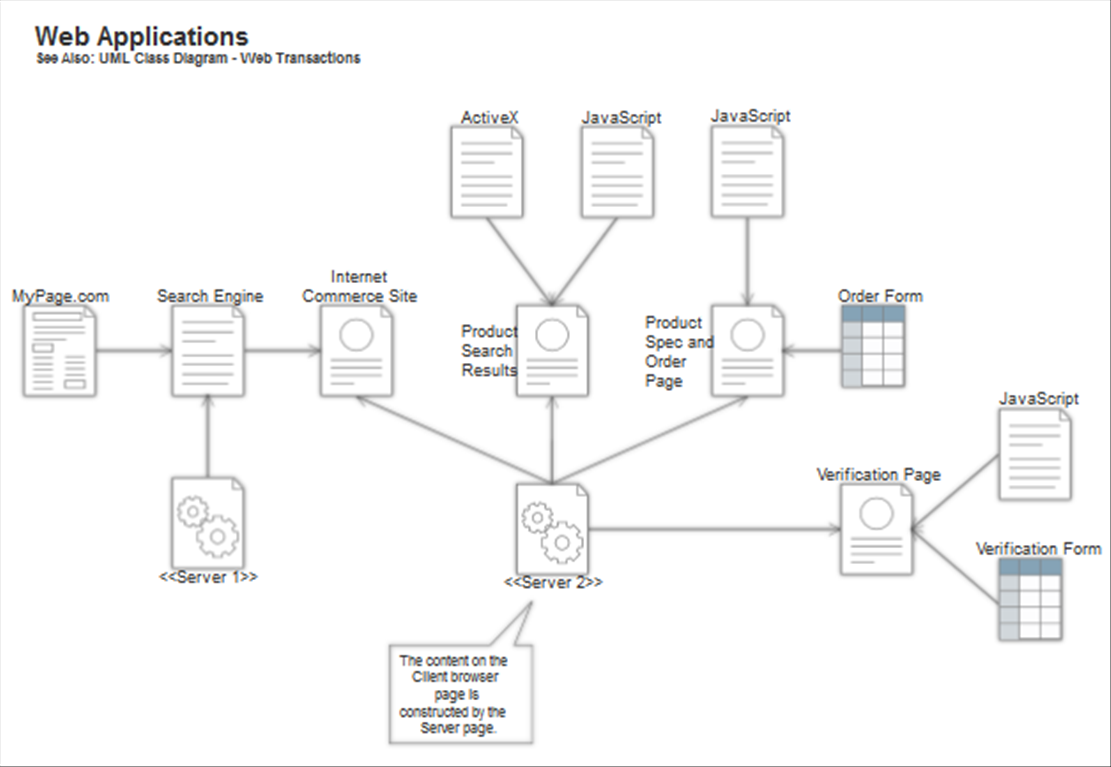
Ejemplo de planificación de un TFG



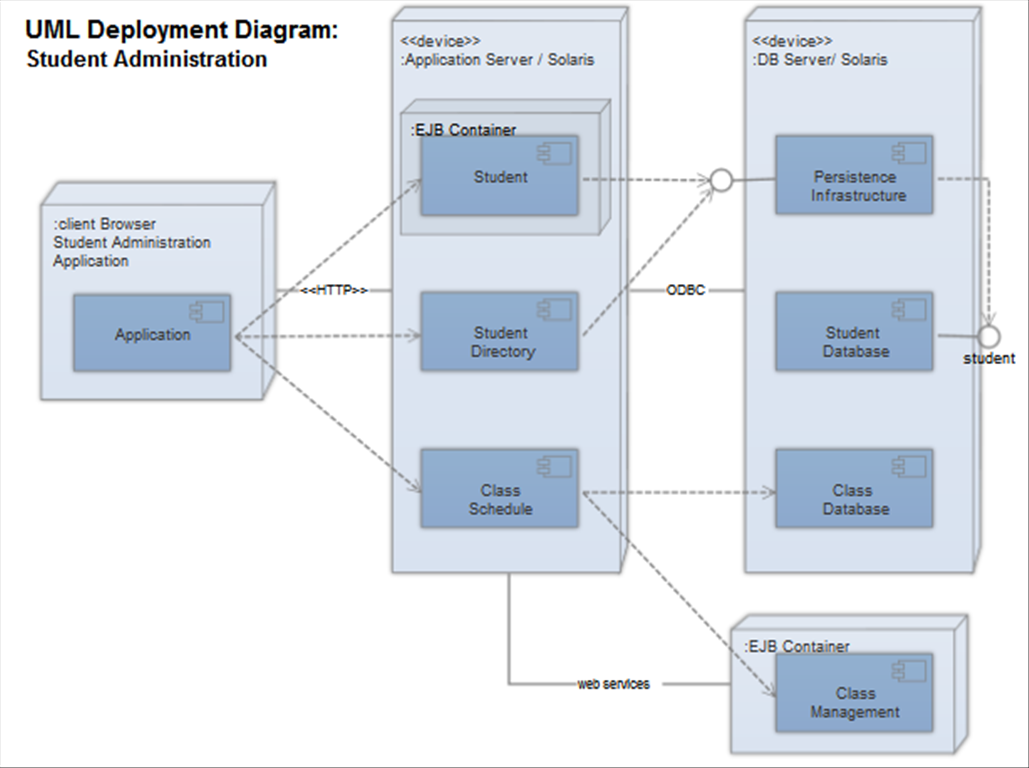
Ejemplo de diagrama de clases



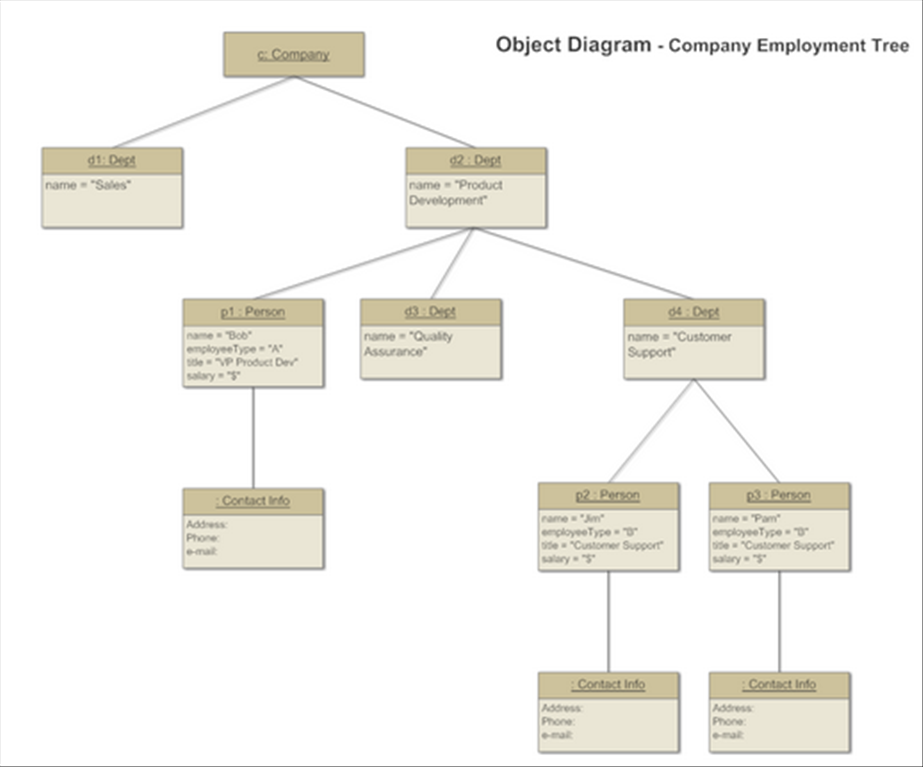
Ejemplo de diagrama de componentes



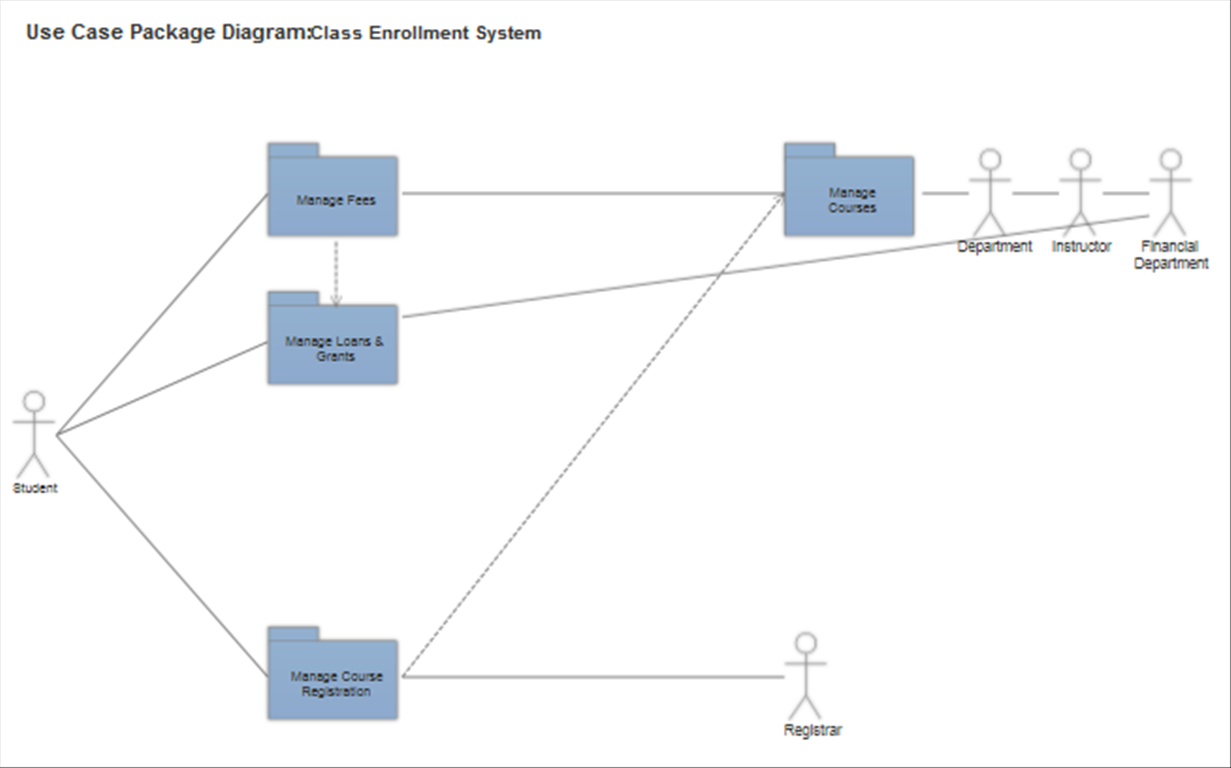
Ejemplo de diagrama de despliegue



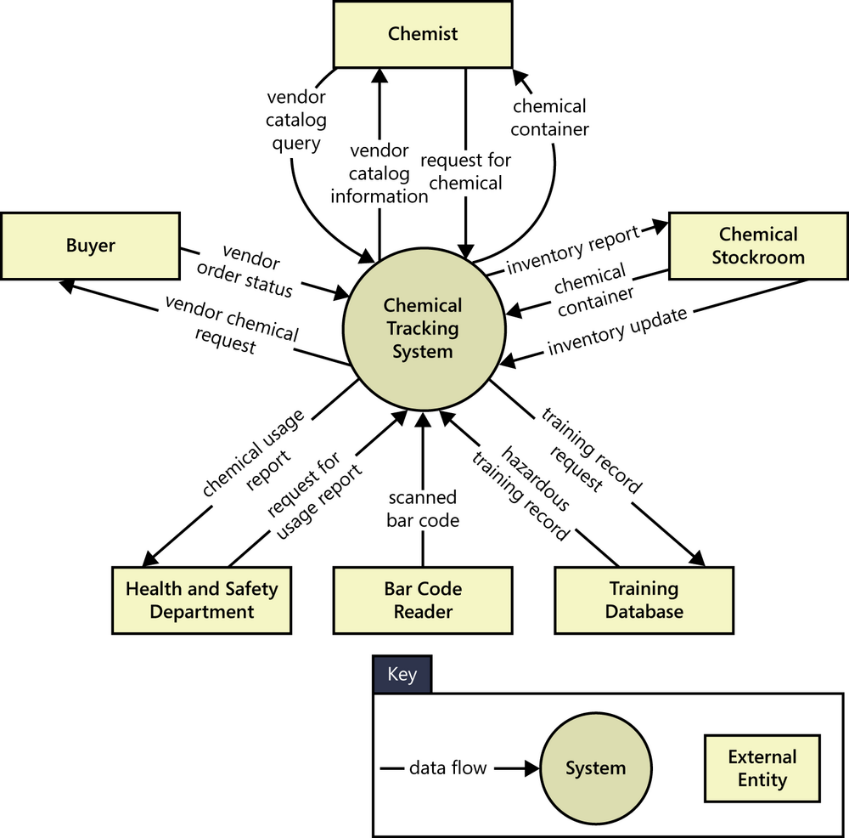
Ejemplo de diagrama de objetos



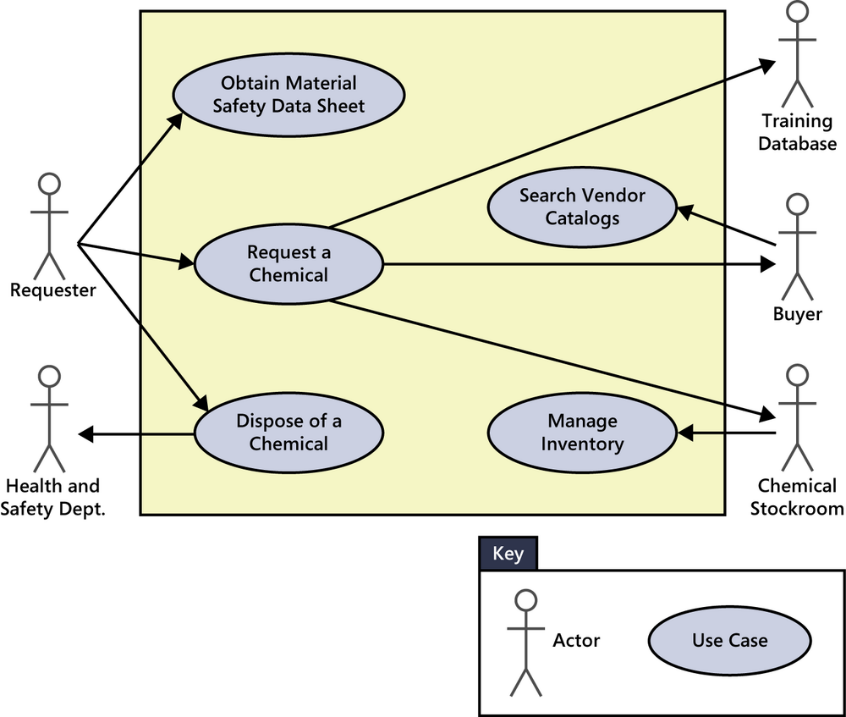
Ejemplo de diagrama de paquetes



Ejemplo de diagrama de contexto



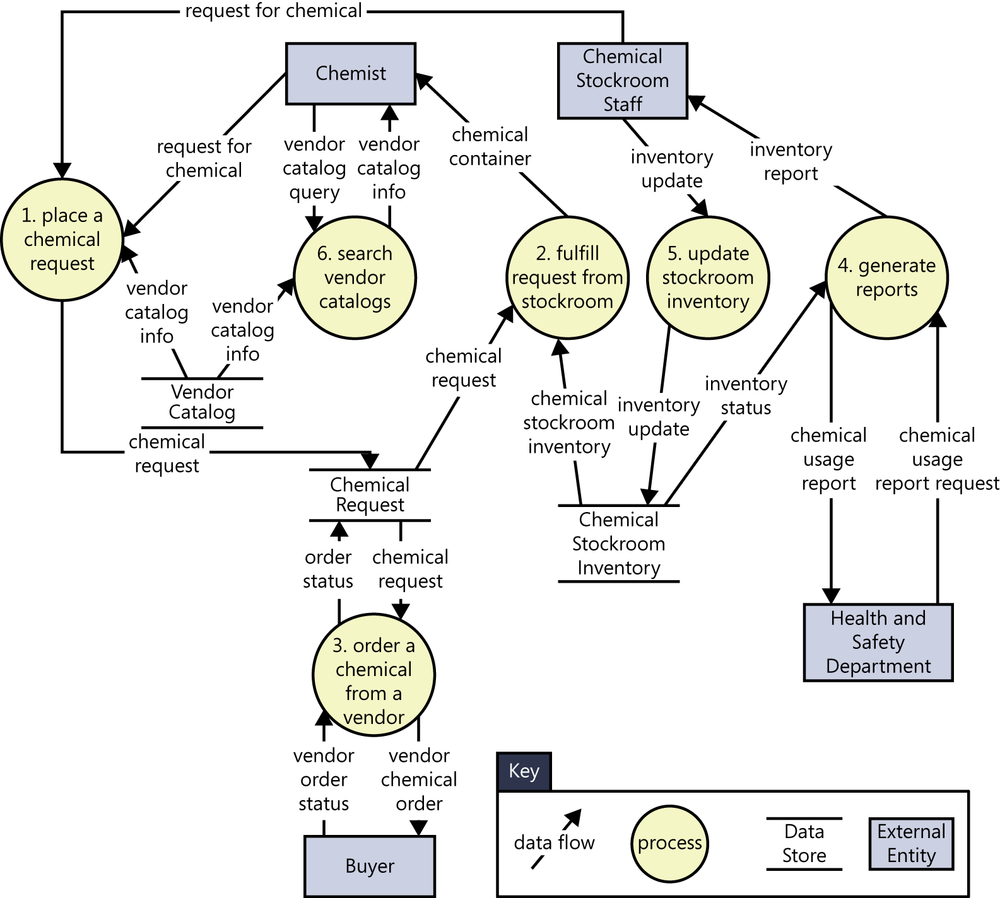
Ejemplo de caso de uso



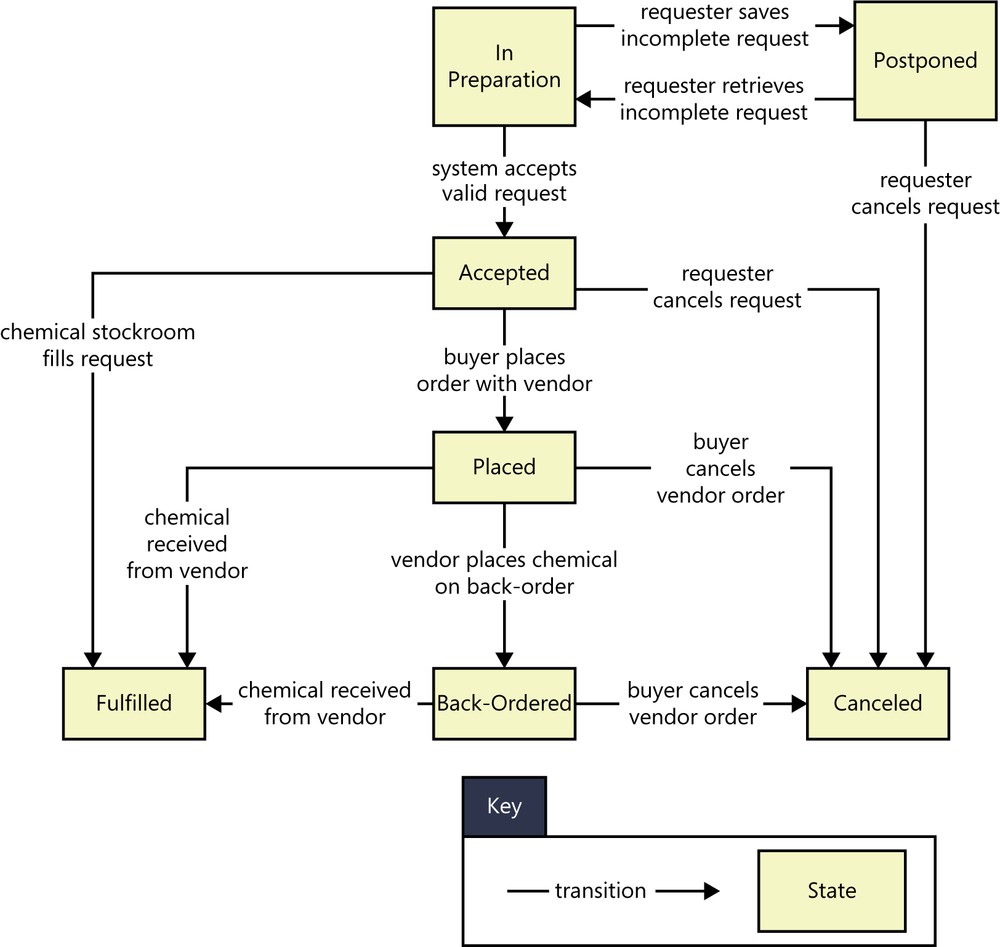
Ejemplo de diagrama de actividades

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



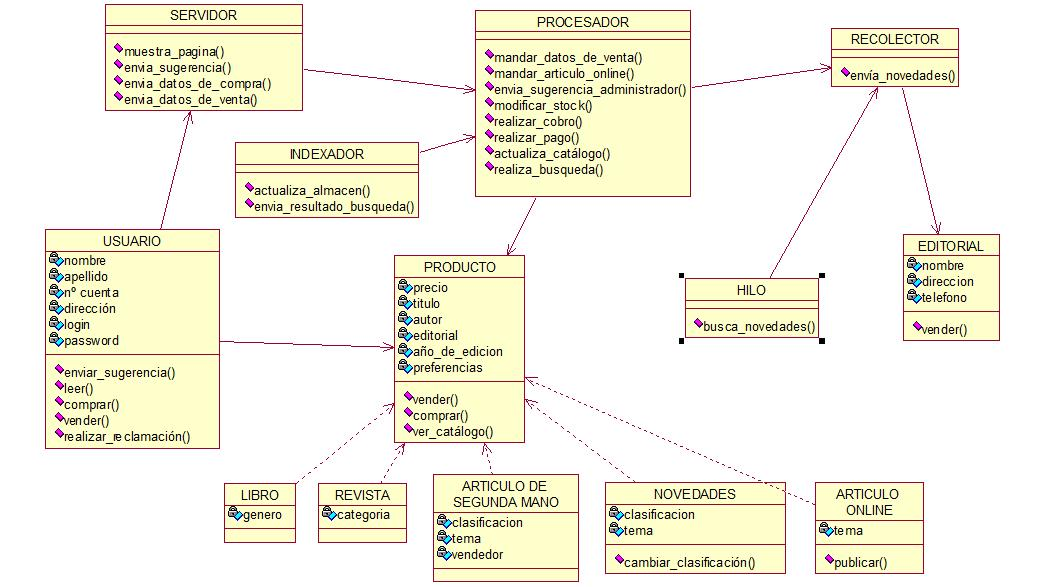
Ejemplo de diagrama de transición



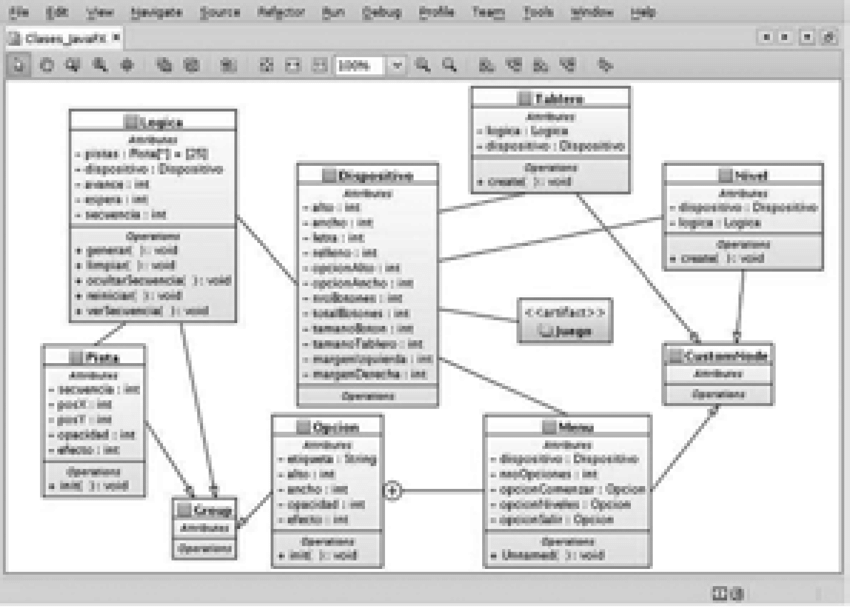
Ejemplo de diagrama lógico de una vpn



Ejemplo de Diagrama E/R



Ejemplo de diagrama de clases de usuarios



Ejemplo de interfaz HMI

