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

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, puesto que normalmente se basan únicamente en su vocación. Muchos de ellos se equivocan en sus elecciones, lo cual se ve reflejado en el abandono del primer curso del grado, siendo este punto la motivación central para la realización de este TFG, pues mediante la creación de un Servicio Web RESTful especializado se garantiza al estudiante la orientación adecuada para la elección de su grado, a través de formularios estandarizados con bases psicológicas, psicotécnicas y pedagógicas, asegurando una mayor continuidad y evitando así el abandono universitario. El Servicio Web RESTful realizado en este TFG da un servicio óptimo mediante dos formularios; uno dónde se orienta al estudiante sobre los grados universitarios (ingeniería, ciencias sociales, artes y otras ciencias) y otro formulario que mide la capacidad de concentración para ver cuánto esfuerzo le puede suponer elegir la carrera que desea. Por último, se toman las notas del usuario, y se da así una respuesta final mediante sugerencias en la facilidad o no del grado que desea.

Palabras Clave

Estudiante, Servicios Web, test estandarizados, abandono universitario, elección, estudios, apoyo.

Abstract

Currently, young people who are in the process of accessing university face a problem that is the choice of their degree or university career, because they are normally based only on their vocation. Many of them make mistakes in their choices, which is reflected in the abandonment of the first year of the degree, this point being the central motivation for the completion of this TFG, since by creating a specialized RESTful Web Service the student is guaranteed adequate guidance for choosing your degree, through standardized forms with psychological, psychotechnical and pedagogical bases, ensuring greater continuity and thus avoiding university dropout. The RESTful Web Service carried out in this TFG provides an optimal service through two forms; one where the student is oriented on university degrees (engineering, social sciences, arts and other sciences) and another form that measures the ability to concentrate to see how much effort it may take to choose the career you want. Finally, the user's notes are taken, and a final answer is given by means of suggestions at the ease or not of the degree you want.

Keywords

Student, Web Services, standardized tests, university dropout, election, studies, support.

Índice de contenidos

Contenido

[1 Introducción 1](#_Toc107258983)

[1.1 Objetivos 2](#_Toc107258984)

[2 Gestión del proyecto 3](#_Toc107258985)

[2.1 Modelo de ciclo de vida 3](#_Toc107258986)

[2.2 Papeles desempeñados en el proyecto 5](#_Toc107258987)

**¡Error! Referencia de hipervínculo no válida.¡Error! Referencia de hipervínculo no válida.**[2.3 Planificación 5](#_Toc107258990)

[2.4 Presupuesto 6](#_Toc107258991)

**¡Error! Referencia de hipervínculo no válida.¡Error! Referencia de hipervínculo no válida.**[2.5 Ejecución 8](#_Toc107258994)

[3 Estado del arte 9](#_Toc107258995)

[3.1 ¿Qué son los Servicios Web tradicionales y cómo funcionan? 9](#_Toc107258996)

**¡Error! Referencia de hipervínculo no válida.¡Error! Referencia de hipervínculo no válida.**[4 Análisis 23](#_Toc107258999)

[4.1 Análisis de dominio 23](#_Toc107259000)

[4.2 Especificación de requisitos 29](#_Toc107259001)

[4.3 Análisis de seguridad 31](#_Toc107259002)

[5 Diseño 35](#_Toc107259003)

[5.1 Arquitectura del Sistema 35](#_Toc107259004)

[5.2 Diseño de subsistema backend 36](#_Toc107259005)

**¡Error! Referencia de hipervínculo no válida.¡Error! Referencia de hipervínculo no válida.**[5.3 Diseño del subsistema frontend 49](#_Toc107259008)

[6 Implementación 49](#_Toc107259009)

[6.1 Implementación de backend 49](#_Toc107259010)

**¡Error! Referencia de hipervínculo no válida.¡Error! Referencia de hipervínculo no válida.**[6.2 Referencia al repositorio de software 57](#_Toc107259013)

[6.3 Manuales 58](#_Toc107259014)

[7 Pruebas y validación 58](#_Toc107259015)

[8 Conclusiones y líneas futuras 65](#_Toc107259038)

[Bibliografía 62](#_Toc107259039)

[Anexos 71](#_Toc107259040)

[1. Definición de los recursos de la API 71](#_Toc107259041)

[2. Métodos HTTP de los recursos 74](#_Toc107259042)

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

Índice de ilustraciones

Ilustración 1. Metodología en cascada………………………………..……………………………….3

Ilustración 2. Planificación estimada……………………………………………………….…………..6

Ilustración 3. Planificación realizada…………………………………………………………………….8

Ilustración 4. Servicios Web SOAP………………………………………………………..……………..9

Ilustración 5. Discovery Process………………………………………………………………..……….10

Ilustración 6. Diagrama de una estructura REST……………………………………..………….12

Ilustración 7. Niveles de madurez de los Servicios Web REST………………..…………..15

Ilustración 8. Nivel 0 de Madurez del Servicio Web REST…………………………………..15

Ilustración 9. Nivel 1 de Madurez del Servicio Web REST……………………….………….17

Ilustración 10. Nivel 2 de Madurez del Servicio Web REST……………………………….…18

Ilustración 11. Nivel 3 de Madurez del Servicio Web REST………………………………….20

Ilustración 12. Población de matriculados en universidades………………………………22

Ilustración 13. Test de Toulouse……………………………………………………………….……….27

Ilustración 14. Versión final……………………………………………………………………..………..35

Ilustración 15. Diagrama E/R de la base de datos TFG……………………………..………..45

Ilustración 16. Diagrama E/R tipos de relaciones……………………………………………….46

Ilustración 17. Backend de la API……………………………………………………………….………49

Ilustración 18. Diagrama de clases UML…………………………………………………….………51

Ilustración 19. Método "create" de la clase "UsuarioController"……………………....52

Ilustración 20. Librería JSON de PLAY…………………………………………………………..…….53

Ilustración 21. Llamada al recurso "Usuario" mediante el método POST……………53

Ilustración 22. Respuesta de la API…………………………………………………………….………53

Ilustración 23. Uso del driver MySQL…………………………………….……………….…..……..54

Ilustración 24. Método "addUsuario"………………………………………………………………..55

Ilustración 25. Usuario insertado en la base de datos……………………………..………..56

Ilustración 26. Fichero "routes", recurso usuarios……………………………………………..56

Ilustración 27. Query creación BBDD……………………………………………………..………….57

Ilustración 28. Petición de creación de usuario……………………………………………..…..59

Ilustración 29. Respuesta de creación de usuario…………………………………..………….60

Ilustración 30. Respuesta de la petición del formulario CHASIDE……………..……….60

Ilustración 31. Respuesta de la petición del formulario Toulouse……………..……….61

Ilustración 32. Petición envió de respuesta CHASIDE………………………………..……….62

Ilustración 33. Petición envió de respuesta Toulouse……………………………….……….63

Ilustración 34. Petición envío de usuario………………………………………………….……….63

Ilustración 35. Petición ingreso de nota matemáticas……………………………………….64

Ilustración 36. Respuesta petición ingreso de nota matemáticas………………………64

**¡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.¡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.**

Índice de tablas

[Tabla 1. Coste de la luz por horas trabajadas…………………………………………………….…7](#_Toc106842494)

Tabla 2. Tasas de abandono el primer año universitario…………………………………….24

Tabla 3. Evaluación CHASIDE……………………………………………………………………………..25

Tabla 4. Análisis de seguridad…………………………………………………………………………….32

Tabla 5. Exposición del recurso: /usuarios ……………………………………………….……….39

Tabla 6. Exposición del recurso: /usuarios/:id…………………………………………………...41

Tabla 7. Exposición del recurso: /usuarios/:idUsuarios/notas/:id………………..…….42

Tabla 8. Exposición del recurso: /usuarios/:idUsuarios/formularios/:tipo………...44

# 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 realizarse 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:

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

# Gestión del proyecto

## Modelo de ciclo de vida

En este caso la metodología escogida fue la metodología en cascada ya que es la que más se ajusta a este TFG, dado que los requerimientos son fijos y el trabajo avanza en forma lineal hacia el final [Ilustración 1].

La versión original fue presentada por Royce en 1970, aunque son más conocidos los trabajos 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 (comunicación con el cliente) y avanza a través de planificación, modelado, construcción y despliegue, para concluir con el mantenimiento del software.

1. **Comunicación**. En esta etapa el analista se reúne con el cliente escuchando sus necesidades y, tras estas reuniones, genera el 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. **Planeación o planificación**. En esta etapa se realiza una planificación de los recursos y se estiman los tiempos para el desarrollo de cada una de las etapas.
3. **Modelado**. En esta etapa se realiza un análisis de los requisitos que darán como resultado el diseño del sistema y del programa:
   1. **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 y la funcionalidad de sus partes, así como la manera en que se combinan unas con otras.
   2. **Diseño del programa.** Se desarrollan los algoritmos necesarios para satisfacer los requerimientos del cliente, además del estudio necesario para saber qué herramientas son requeridas para la etapa de codificación.
4. **Construcción.** Se implementa el código del programa para que realice las funcionalidades detalladas en los algoritmos, y después se realizan un conjunto de pruebas y corrección de errores con el objetivo de revisar el cumplimiento de lo acordado con el cliente.
5. **Despliegue del software.** Se trata de la ejecución del sistema, donde el cliente revisa y valida si se han cubierto todas sus necesidades. Una vez revisadas, se realizan las correcciones oportunas para solucionar las necesidades no cubiertas por parte del cliente.

## Papeles desempeñados en el proyecto

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

### Roles del tutor

El tutor ha realizado dos roles. Por un lado, el rol de director del proyecto, ya que ha participado en la planificación y definición de objetivos; por otro lado, de analista de requisitos, ya que ayudó a establecer los requisitos de la aplicación.

### Roles del estudiante

El alumno ha ejercido cuatro roles. En primer lugar el rol de cliente, puesto que propuso la idea de la aplicación. En segundo lugar, de analista de requisitos dado que estableció los requisitos de la aplicación. En tercer lugar de desarrollador, ya que diseñó y escribió el código. 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.

Imagen que contiene Gráfico

Descripción generada automáticamente

Ilustración 2. Planificación estimada

Como se puede observar en la ilustración anterior [Ilustración 2], la parte que más tiempo ocupa del desarrollo del programa es la creación de las bases del proyecto, a saber: 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, ya que condicionarán el desarrollo de éste al ser dependiente 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

1. Luz (kW/h). Teniendo en cuenta que durante los días hábiles de destinan dos horas/día al TFG, los días de fin de semana cuatro 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 [Tabla 1].

Tabla 1. Coste de la luz por horas trabajadas

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

### Costes en la fase de producción

1. Amazon WebService:
   1. 50 TB/mes 🡪 0,77 euros por GB.
   2. t2.medium 🡪 0,93 euros por hora el API.
   3. Totales 🡪 670,37 euros/mes.
2. 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

En este apartado se muestran, mediante un diagrama de GANTT, los tiempos estimados que se dedican a las tareas una vez terminadas todas las etapas del proyecto.

Tabla

Descripción generada automáticamente

Ilustración 3. Planificación realizada

En comparación del mostrado en el apartado 2.3, en la ilustración anterior [Ilustración 3] se puede observar un aumento del tiempo en la extracción de los objetivos de las ideas del proyecto, en el análisis del dominio y en el estado del arte, esto se debe a la falta de bases en la rama de la psicología, necesarias para el proyecto, factor que se vio reflejado también en el diseño de los servicios.

# 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?

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.

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

Ilustración 4. Servicios Web SOAP

El intercambio de mensajes se encuentra estandarizado bajo una descripción de un Servicio Web (WSD), que 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*) [Ilustración 5] 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, como 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, además, 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.
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” [Ilustración 6]. 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 reside en que 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 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”. 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 restricciones que impone el estilo arquitectónico REST son:

1. **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 la lógica de la aplicación que se quiere diseñar (p.e: Formulario, Usuario, entre otros). El recurso debe ser identificable y único, usando un proceso estandarizado. En el caso de la web se usan las URI (*Unified Resource Identifier*) (p.e: /usuarios y /usuarios/:idUsuario). La URI actúa como nombre y dirección en cada recurso, lo cual permite que se pueda “navegar” entre los diferentes recursos.
2. **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: información 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) utilizando los formatos definidos por el estándar MIME (Multipurpose Internet Mail Extensions) que es el estándar para codificar representaciones.
3. **Hipermedia.** Son el vínculo y los enlaces a los distintos recursos, y esto es lo que permite la relación de recursos. Una restricción significativa de REST es denominada como HATEOAS (*Hypermedia as the Engine of the Application State*). Esta restricción admite que, mediante el uso del hiperenlace, se obtenga la representación del recurso, que contiene URIs de otros recursos con los que se puede interactuar.
4. **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 y así el servidor puede procesarlas, es decir, son 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.
5. **Interfaz homogénea.** Todos los recursos comparten la misma interfaz para poder trabajar mejor. 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 está, logrando así una representación (GET), crear un recurso (POST), actualizar la representación completa de un recurso (PUT), y parcialmente (PATCH) o eliminar un recurso (DELATE). Todos van a la URI de un recurso, (unas veces a la URI del recurso que va a ser cambiado, y otras veces la URI de recurso que toma el papel de “factoría” de recursos). De este modo, la API de una aplicación que se basa en el estilo REST, 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.

### Modelo de madurez de Richardson

Leonard Richardson propone una clasificación en diversos niveles llamados **Niveles de madurez de los Servicios Web REST** [Ilustración 7].

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 Web tradicionales, 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, normalmente basados en RIP (*Remote Procedure Invocation*) [Ilustración 8].

Escala de tiempo

Descripción generada automáticamente

Ilustración 8. Nivel 0 de Madurez del Servicio Web REST

Un clásico ejemplo 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” y 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, llega a secciones o documentos del sitio Web usando las distintas URIs.

Imagen que contiene Gráfico

Descripción generada automáticamente

Ilustración 9. Nivel 1 de Madurez del Servicio Web REST

Por ejemplo:

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

Siguiendo con el ejemplo del apartado del nivel 0 [Ilustración 9].

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 los tipos de contenidos que especifican el formato o formatos que sigue el recurso [Ilustración 10].

Imagen que contiene Diagrama

Descripción generada automáticamente

Ilustración 10. Nivel 2 de Madurez del Servicio Web REST

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. 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 “*HATEOAS*”, 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 fácil de entender al usuario, cuya la respuesta ofrece informacion adicional como enlaces a otros recursos ampliando las interacciones con estos [Ilustración 11].

Escala de tiempo

Descripción generada automáticamente

Ilustración 11. Nivel 3 de Madurez del Servicio Web REST

Siguiendo con el ejemplo, en el caso del método POST, mandará información adicional para exponer 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.

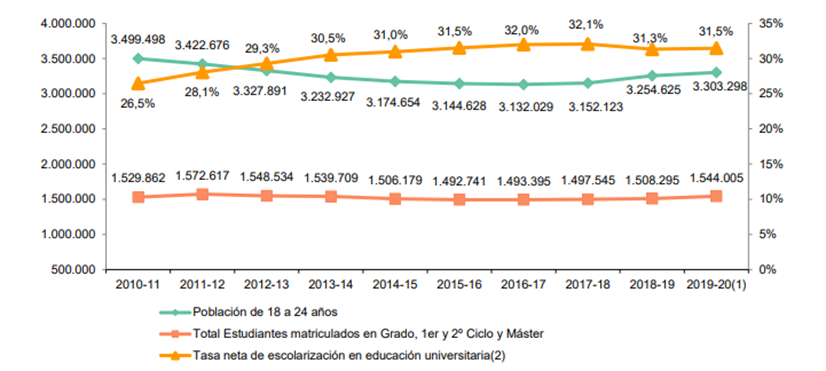


Ilustración 12. Población de matriculados en universidades.

Según la ilustración anterior [Ilustración 12], 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 2. Tasas de abandono en el primer año universitario.

Como se puede observar [Tabla 2], 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 realizarán dos tipos de test estandarizados: test de aptitudes y test de concentración; que serán informatizados.

El “*Test de Orientación Vocacional CHASIDE*” de Holland Ríase 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 [Tabla 3].

Imagen que contiene gabinete, reloj, diferente, pantalla

Descripción generada automáticamente

Tabla 3. Evaluación CHASIDE

El “*Test de Toulouse”* de E. Toulouse y H. Piéron [Ilustración 13] 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:

1. El Índice Global de Atención y Percepción (IGAP), constituye una medida de la capacidad perceptiva y atencional de los evaluados.
2. El Cociente de Concentración (CC), mide la capacidad de concentración que tiene el usuario.
3. 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 13. 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**

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

* **Capacidad y eficiencia**

1. Debe soportar por defecto como numero de 25, 125, 500 peticiones simultáneas por 1, 60, 360 segundos respectivamente con un tiempo de respuesta máximo de 4 segundos.

* **Interoperabilidad con otros sistemas**
  1. Debe ser accesible desde cualquier dispositivo (tablets, móviles, otras aplicaciones, etcétera).
  2. Debe ser ajena a como estén formados los sistemas que acceden a dichos recursos.
  3. Debe usar JSON[14] como formato para enviar y transcribir los datos.
  4. Debe subirse a AWS[15] para que sea 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**

1. Debe ser implementado como Servicio Web.
2. Debe utilizar los métodos HTTP para interactuar con los demás recursos.

## Análisis de seguridad

Este análisis es 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 [Tabla 4]:

* **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 |

Tabla 4. Análisis de seguridad

En base a la tabla anterior, 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 transformarlo 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.

Diagrama

Descripción generada automáticamente

Ilustración 14. Versión final

Como se puede ver en la imagen [Ilustración 14] la parte del backend se encuentra alojada en el servidor Web, dentro de AWS, el cual será accesible por los usuarios. Por otro lado, el frontend se encuentra administrado por el cliente, mediante una App o Web Forms, que se encargará de crear una interfaz para que los usuarios de los clientes ingresen la información requerida por la API.

Por último, la base de datos se encontrará alojada en el cliente, salvo en algunos casos determinados se encontrarán alojadas junto al servicio Web.

## Diseño de subsistema backend

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

### 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. Tras 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

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

**/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

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

Tabla 5. Exposición del recurso: /usuarios

#### 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 5] definido anteriormente, los otros métodos se encuentran en el anexo [1].



Como aparece en la anterior tabla, el recurso “**/usuario**” es utilizado principalmente para la creación del usuario en la base de datos, la información del usuario es enviada en formato JSON a la API mediante el método POST, si el formato se encuentra mal formado, la API devolverá el error 404, en caso de no estar funcionando el servicio devolverá el error 500, y en caso de poder hacerlo devolverá un mensaje de OK y un JSON con los datos del usuario ingresado, entre los cuales, estará la URI por la cual se puede acceder a este usuario creado.

Respecto al método GET, está enfocado a temas de administración de los usuarios, para ver todos los usuarios almacenados en la base de datos.

Además del recurso “**/usuario”** que es el ejemplo seguido durante todo este capítulo. A continuación se explicaran los siguientes recursos: “**/usuarios/:idUsuarios/notas”** , “**/usuarios/:idUsuarios/formularios/:id**” y “**/usuarios/:id**”, ya que estos tres recursos son los que llevan la carga de la funcionalidad de la API.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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 | - | - | 404-Not Found |
| 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 |

Tabla 6. Exposición del recurso: /usuarios/:id

Como muestra la tabla del recurso “**/usuarios/:id**”, [Tabla 6] sólo posee métodos GET, DELETE y PATCH, para poder acceder a ellos, debes poner en el parámetro id de la URI un id que pertenezca a un usuario ingresado anteriormente. El método GET envía al usuario que realizo la llamada, el usuario en formato JSON con toda la información asociada, muy útil tras realizar los formularios, ya que en este recurso se almacenaran sus aptitudes, intereses y nivel de concentración.

Respecto al método PATCH es utilizado en vez del PUT dado que sólo le es permitido al usuario cambiar la contraseña, puesto que el nombre es fijo y los otros datos son manipulados por la API. Y, por último, el método DELETE ya que es utilizado por el usuario, cuando este quiera eliminar todo registro de el en la BBDD.

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



Tabla 7. Exposición del recurso: /usuarios/:idUsuarios/notas/:id

Como se describe en la tabla del recurso “**/usuarios/:idUsuarios/notas**” [Tabla 7], sólo posee métodos GET, DELETE y PUT, para poder acceder a ellos, debes poner en el parámetro idUsuario de la URI un id que pertenezca a un usuario ingresado anteriormente, de igual forma que se ingresa el id de la nota previamente introducida. El método GET envía al usuario que realizo la llamada, la nota en formato JSON con toda la información asociada, muy útil tras realizar los formularios, ya que en este recurso se almacenaran sus recomendaciones, es decir, las horas de estudio recomendadas y el riesgo al suspenso de la asignatura. Respecto al método PUT es utilizado para cambiar el nombre a la asignatura, la nota y a las horas de estudio dedicadas, las cuales serán tratadas para obtener una recomendación. Y, por último, el método DELETE ya que es utilizado por el usuario, cuando este quiera eliminar todo registro de las notas en la BBDD.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Método | URI | Utilidad | Representación | Código Respuesta |
| POST | /usuarios/:idUsuarios/formularios/  :tipo |  |  | 404-Not Found |
| GET | /usuarios/:idUsuarios/formularios/  :tipo | Obtiene todas las notas de un resultado en concreto de un usuario en concreto | JSON | 200-OK |
| 500-Internal Server Error |
| PUT | /usuarios/:idUsuarios/formularios/  : tipo | Actualiza las respuestas el formulario escogido del usuario | JSON | 200-OK |
| 400-Bad Request |
| 500-Internal Server Error |
| DELETE | /usuarios/:idUsuarios/formularios/  : tipo | - | - | 404-Not Found |
| PATCH | /usuarios/:idUsuarios/formularios/  : tipo | - | - | 404-Not Found |

Tabla 8. Exposición del recurso: /usuarios/:idUsuarios/formularios/:tipo

Por último, queda por detallar el recurso “**/usuarios/:idUsuarios/formularios/:id**” [Tabla 8] que es el encargado de obtener uno de los dos test que posee la API, mediante el método GET, el cual posee las preguntas siempre, y en el caso de ser respondido, las respuestas también estarán incluidas. Respecto al método PUT es utilizado para dar un valor a las respuestas de los formularios. En ambos casos el formato de los datos es JSON.

#### 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

Para dar soporte a los datos recibidos de la API, se ha realizado un diseño que cubriera dichos datos, simplificándolo en la medida de lo posible, el cual se mostrará a continuación [Ilustración 15]:

Diagrama

Descripción generada automáticamente

Ilustración 15. 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 [Ilustración 16]:

Imagen que contiene Gráfico

Descripción generada automáticamente

Ilustración 16. Diagrama E/R tipos de relaciones

Una vez explicadas las figuras que aparecen en la Ilustración 14. A continuación se procederá a detallar las relaciones entre las tablas, sus funcionalidades y su estructura.

#### Usuario

Esta es la tabla principal, de la cual dependen las demás, dado que al usuario se le asignan los formularios y las notas.

La relación entre usuario y notas es 1-N, ya que un usuario puede tener N notas, mientras que unan nota pertenece solo a un usuario. Caso diferente al del formulario, ya que seria 1-1 donde solo existen dos formularios que son creados en el momento que se crea un usuario nuevo.

Respecto a la estructura, cabe destacar que solo tiene como clave primaria el id, que será utilizado para establecer la relación con las tablas “formulario” y “nota”.

#### Nota

En esta tabla se encuentra todo lo asociado a las notas del usuario y las recomendaciones asociadas a éstas. La relación es con la tabla “usuario”, la cual se ha detallado en el apartado anterior. Respecto a la estructura, cabe destacar que tiene una sola clave primaria por la cual se encuentra identificada y una clave foránea que pertenece a la tabla “usuario”.

#### Formulario

Esta tabla tiene la funcionalidad de relacionar el usuario, con las respuestas, creada por la orientación del diseño, es decir, se ha diseñado la base de datos vista desde el punto de vista del programa.

La relación es con la tabla “usuario”, la cual se ha detallado en el apartado anterior y con la tabla “respuesta”, de la cual sería 1-N dado que un formulario puede tener N respuestas mientras que cada respuesta se asocia a un solo formulario.

Respecto a la estructura, cabe destacar que tiene una sola clave primaria por la cual se encuentra identificada por la cual se encuentra identificada, que será utilizada como clave foránea en la tabla “respuesta” y una clave foránea que pertenece a la tabla “usuario”.

#### Respuesta

En ella se encuentran todas las respuestas del usuario relacionadas cada una de ellas a una pregunta en concreto.

La relación con la tabla “formulario” queda detallada en el apartado anterior, mientras que la relación con la tabla “pregunta” es de 1-N donde una pregunta puede tener N respuestas mientras que una respuesta solo tiene una pregunta.

Respecto a la estructura, cabe destacar que tiene una sola clave primaria por la cual se encuentra identificada y dos claves foráneas que pertenece una a la tabla “formulario” siendo esta “idFormulario” y otra pertenece a la tabla “pregunta” siendo ésta “idPregunta”.

#### Pregunta

En ella se encuentran todas las preguntas del formulario.

La relación con la tabla “respuesta” queda detallada en el apartado anterior. Respecto a la estructura, cabe destacar que tiene una sola clave primaria por la cual se encuentra identificada, que será utilizada como clave foránea en la tabla “respuesta”.

## Diseño del subsistema frontend

Al ser una API no es necesario que esta posea una interfaz gráfica, ya que está diseñada para interactuar con otros sistemas, los cuáles si se encargaran de crear una interfaz intuitiva para el usuario, la cual se encargara de recoger, mostrar y enviar los datos a la API.

# Implementación

En este apartado se recogerá todos los datos necesarios para la implementación de la API, además se detallarán los componentes, estructura y funciones del backend.

## Implementación de backend

En este apartado se hablará que tecnologías se han empleado para realizar el diseño, cuál es su estructura y cómo funcionan cada una de las partes que lo componen [Ilustración 17]:

Captura de pantalla de un celular con letras

Descripción generada automáticamente

Ilustración 17. Backend de la API

Como se puede observar en la ilustración el backend utiliza MySQL para 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; Y PLAY dado que 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.

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.

1. Beans: son las semillas/entidades que representan al recurso, necesarias para la transmisión y manipulación del recurso.
2. Controllers: son las clases encargadas de atender las peticiones HTTP.
3. Model: es la parte lógica de la API, en este caso, se encarga de realizar las sugerencias en base a los datos del usuario.
4. Services: clases encargadas de conectarse a la base de datos mediante Querys, además de aislar las llamadas de la base de datos.
5. Utils: encargada de crear el response adaptado para devolver JSON.
6. 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 18. Diagrama de clases UML

Como se puede observar, las clases “Formulario”, “Usuario” y “Nota” extienden de “RecursoWeb” (línea azul) [Ilustración 18]. 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 19. Método "create" de la clase "UsuarioController"

El método "create" [Ilustración 19]. 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 sí posea datos, se transcribirán a la clase “Usuario”, mediante la función que posee la librería nativa de PLAY [Ilustración 20].



Ilustración 20. 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. [Ilustración 21] [Ilustración 22].

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Ilustración 21. Llamada al recurso "Usuario" mediante el método POST

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Ilustración 22. 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 asociarán 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”. [Ilustración 23].

Texto

Descripción generada automáticamente

Ilustración 23. 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”. [Ilustración 24].

Texto

Descripción generada automáticamente

Ilustración 24. 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. [Ilustración 25].

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Ilustración 25. 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 26].

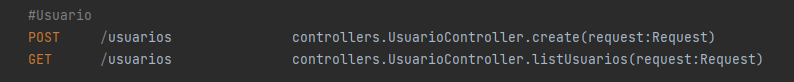


Ilustración 26. Fichero "routes", recurso usuarios

### Estructura e implementación de la BBDD

La estructura está representada en el apartado 5.2.3, para poder llevar la implementación se crearon Querys predefinidas encargadas de generar la base de datos, las tablas, las relaciones entre sí y los datos necesarios.

A continuación, se mostrará la creación de la base de datos y la tabla usuario. [Ilustración 27].

Texto

Descripción generada automáticamente

Ilustración 27. Query creación BBDD

## 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

Tras el desarrollo final del Servicio Web RESTful, cuyo funcionamiento se probó mediante Postman, ya que es una herramienta que consume muy pocos recursos y nos permite (a través de la emisión de solicitudes HTTP) probar los métodos utilizados para este Servicio Web como GET, POST, DELATE y PATCH, se ha podido comprobar el correcto funcionamiento de la API y su interconexión con la BD.

Una vez comprobado, se ha procedido a realizar la validación de la API en base a un caso que pase por todas las fases de interacción con la API, que será detallado a continuación.

Inicialmente el usuario ingresará a la aplicación del cliente con el nombre de usuario y la contraseña. Tras su registro, recibirá los formularios de Toulouse y CHASIDE los cuales tiene que responderlos en base a las preguntas que contenga. Una vez respondidos, la aplicación mostrará al usuario cuáles son sus aptitudes, intereses y nivel de concentración. Después de verlo detenidamente, el usuario introducirá en la aplicación las notas de las asignaturas, la asignatura de matemáticas y física, y las horas de estudio dedicadas a estas asignaturas, respectivamente. Tras hacerlo, se le mostrará la información introducida en matemáticas junto con la recomendación de horas de estudio y el riesgo de suspender la asignatura. Una vez finalizado el primer año de carrera, el usuario decide si quiere eliminar sus datos personales.

Para validar el caso descrito anteriormente y con ello validar el correcto funcionamiento de la API, se realizaron los siguientes pasos:

1. Primero el usuario debe ingresar en la aplicación del usuario e insertar su nombre y contraseña para registrarse. Una vez insertados los datos la aplicación del cliente enviara los datos a la API en formato JSON, mediante la siguiente URI: **POST** **“/usuarios”** [Ilustración 28].

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

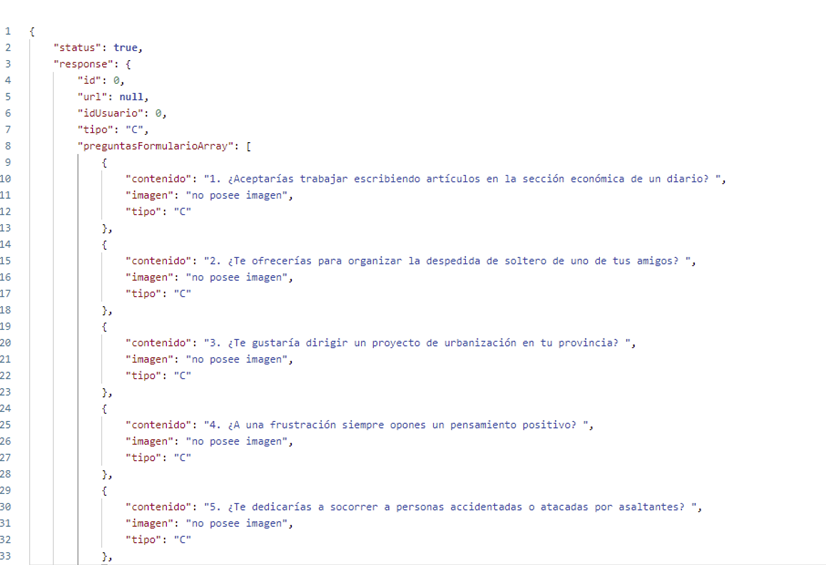
Ilustración 28. Petición de creación de usuario

Devolviendo como respuesta el objeto Usuario en formato JSON donde uno de los atributos del objeto es la URI para acceder al nuevo usuario [Ilustración 29].



*Ilustración 29. Respuesta de creación de usuario*

1. Tras el registro la aplicación del cliente debe obtener del JSON obtenido en el paso anterior la URI, dado que en ella tiene el id del usuario, después debe llamar al pedir las preguntas de cada formulario para mostrársela al usuario mediante la siguiente URI: GET “/usuarios/1/formularios/C” para el caso del test de CHASIDE y para el caso del test de Toulouse GET “/usuarios/1/formularios/T”



*Ilustración 30. Respuesta de la petición del formulario CHASIDE*

Como se puede observar [Ilustración 30] proporciona los títulos de las preguntas del test, sin imágenes.



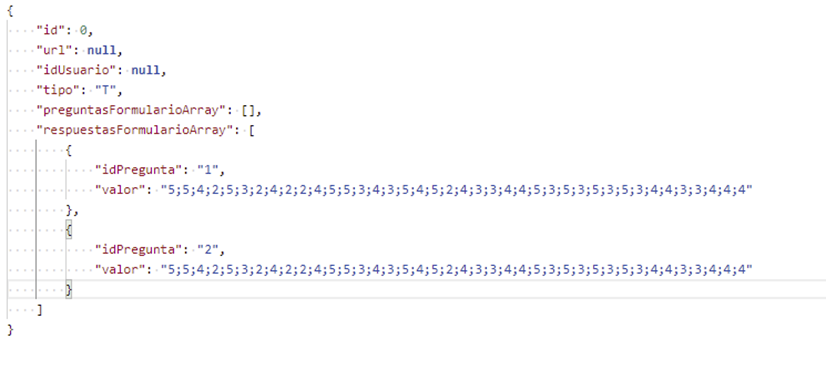
*Ilustración 31. Respuesta de la petición del formulario Toulouse*

En esta imagen [Ilustración 31] podemos observar que además de la pregunta se le asocia una imagen en base64 para que la aplicación del cliente lo asocie a una etiqueta img en el html.

1. Una vez tenga la app las preguntas, deberá mostrárselas para que el usuario ingrese las respuestas y luego enviar a la API en formato JSON las respuestas haciendo uso de la URI: PUT “/usuarios/1/formularios/C” y PUT “/usuarios/1/formularios/T” [Ilustración 32]



*Ilustración 32. Petición envió de respuesta CHASIDE*



*Ilustración 33. Petición envió de respuesta Toulouse*

Tras recibir la respuesta [Ilustración 33] automáticamente se actualiza el usuario con los resultados de los test para mostrarlos tiene que pedir el usuario usando la URI: GET “http://localhost:9000/usuarios/1” obteniendo como respuesta el siguiente JSON.



*Ilustración 34. Petición envío de usuario*

Como se puede observar en la ilustración [Ilustración 34], tiene aptitudes Humanísticas/sociales, no posee intereses y tiene un nivel de concentración bajo.

1. Tras revisar los resultados de los test, el usuario introduce en la aplicación la asignatura, la nota obtenida(puntuación) y las horas que ha dedicado o dedicará a dicha asignatura. Para ello la aplicación debe llamar a la siguiente URI: POST “/usuarios/1/notas” y enviar los datos en formato JSON a la API como se muestra a continuación: [Ilustración 35]



*Ilustración 35. Petición ingreso de nota matemáticas*

Como se puede observar el usuario introdujo en tiempo de estudio un 3 siendo el tiempo medido en minutos. Como ya realizó los test con anterioridad, esta misma llamada le devuelve la nota con la sugerencia incluida. [Ilustración 36]

**

*Ilustración 36. Respuesta petición ingreso de nota matemáticas*

Como se puede observar el usuario tiene un alto riesgo de suspender la asignatura y se le recomienda como mínimo 209 minutos en vez de 3.

# Conclusiones y líneas futuras

Este TFG parte de la demanda de estudiantes que quieren acceder a las universidades y de las universidades que quieren reducir el abandono universitario en el primer año de carrera, por ello es una herramienta que facilita la gestión en cuanto a la elección de la titulación.

En primer lugar, el usuario se tiene que registrar en la aplicación mediante un nombre de usuario y una contraseña. Una vez realizado el paso anterior, dispondrá de dos test estandarizados (Toulouse y CHASIDE) los cuales le medirán las aptitudes, los intereses vocacionales y la concentración. Tras realizar los test dispondrá de los resultados de los mismos en su perfil. Además, el usuario debe introducir sus notas de cada asignatura, junto con el nombre y las horas dedicadas a éstas. Al introducirlas recibirá en cada nota las horas recomendadas de estudio y el riesgo de cursar cada asignatura o, si bien lo prefiere, en vez de hacerlo individualmente, puede visualizarlas de forma genérica.

Por último, el usuario podrá relacionarse con los demás usuarios para poder ayudarse en aquellas asignaturas de riesgo alto, pudiendo acceder a sus resultados si los otros estudiantes así lo permiten.

Debido a la limitación de tiempo en el proyecto, no ha podido añadirse la implantación de la seguridad, como el token Bearer, muy útil para ofrecer una identificación segura del usuario, ya que en cada llamada salvo la de login o registro del usuario, sería requerido el token para acreditar que el usuario tiene esos permisos. Por otro lado, aunque no fue planteado, es necesario una pequeña interfaz gráfica que permita a los desarrolladores de la aplicación del cliente, tener un acceso más intuitivo a la API.

Una vez implementadas las mejoras anteriores, se podría indagar más en que factores promueven la intención del abandono o del fracaso universitario, por ejemplo: la cantidad de horas prácticas y teóricas, el estrés sufrido, etcétera, ya que son factores importantes asociados a la etapa universitaria de los estudiantes.

# Bibliografía

1. **¡Error! Referencia de hipervínculo no válida.**Martha Báez. Holland Ríase. Test de orientación vocacional CHASIDE (2007). Academia Edu. Consultado el 3 de Mayo de 2022. <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. Consultado el 3 de Mayo de 2022. <https://web.teaediciones.com/Ejemplos/Extracto_libro_TP-R.pdf>
3. DR. Winston W. Royce (1970) Managin the development of large software systems. Consultado el 5 de Mayo de 2022. <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. Consultado el 11 de Mayo de 2022. Consultado el 17 de Mayo de 2022. [https://www.w3.org/TR/ws-arch/#whatis](https://www.w3.org/TR/ws-arch/" \l "whatis)
5. Centro Criptológico Nacional. (mayo de 2020). Guía de Seguridad de las TIC CCN-STIC 803. Consultado el 17 de Mayo de 2022. <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. Consultado el 19 de Mayo de 2022.

<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. Consultado el 19 de Mayo de 2022. Consultado el 2 de Junio de 2022. <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. Consultado el 2 de Junio de 2022. <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. Consultado el 7 de Junio de 2022. <https://www.universidades.gob.es/stfls/universidades/Estadisticas/ficheros/Datos_y_Cifras_2020-21.pdf>
2. **¡Error! Referencia de hipervínculo no válida.**Remote Procedure Invocation. Enterprise Integration Patterns. Consultado el 9 de Junio de 2022. <https://www.enterpriseintegrationpatterns.com/EncapsulatedSynchronousIntegration.html>
3. Fielding R. (2000) Architectural Styles and the Design of Network-based Software Architectures. Thesis Doctoral. University of California, Irvine. Consultado el 9 de Junio de 2022.
4. 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. Consultado el 10 de Junio de 2022. <https://doi.org/10.1145/3106237.3121282>
5. Fielding, Taylor (2002) Principled Design of the Modern Web Architecture. ACM Transactions on Internet Technology, Vol. 2, No. 2Pages 115–150. Consultado el 10 de Junio de 2022.
6. JSON – Introduction.W3C.https://www.w3schools.com/js/js\_json\_intro.asp Consultado el 14 de Junio de 2022.
7. Amazon. Informática en la nube con AWS. Consultado el 14 de Junio de 2022. https://aws.amazon.com/es/what-is-aws/
8. Diario Oficial de la Unión Europea. REGLAMENTO (UE) 2016/679 DEL PARLAMENTO EUROPEO Y DEL CONSEJO  
   de 27 de abril de 2016. Consultado el 14 de Junio de 2022. 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.**

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

# Anexos

## Definición de los recursos de la API

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

## Métodos HTTP de los recursos

Recurso /notas

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

Recurso /usuarios/idUsuarios/notas

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