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](#_Toc135865823)

[1.1 Objetivos 2](#_Toc135865824)

[2 Gestión del proyecto 3](#_Toc135865825)

[2.1 Modelo de ciclo de vida 3](#_Toc135865826)

[2.2 Papeles desempeñados en el proyecto 5](#_Toc135865827)

[2.2.1 Roles del tutor 5](#_Toc135865828)

[2.2.2 Roles del estudiante 5](#_Toc135865829)

[2.3 Planificación 5](#_Toc135865830)

[2.4 Presupuesto 6](#_Toc135865831)

[2.4.1 Costes en la fase de desarrollo 7](#_Toc135865832)

[2.4.2 Costes en la fase de producción 7](#_Toc135865833)

[2.5 Ejecución 8](#_Toc135865834)

[3 Estado del arte 9](#_Toc135865835)

[3.1 ¿Qué son los Servicios Web y cómo funcionan? 9](#_Toc135865836)

[3.1.1 Servicios Web tradicionales 10](#_Toc135865837)

[3.1.2 Servicios RESTful (estilo arquitectónico REST) 10](#_Toc135865838)

[3.1.3 Modelo de madurez de Richardson 12](#_Toc135865839)

[4 Análisis 13](#_Toc135865840)

[4.1 Análisis de dominio 13](#_Toc135865841)

[4.2 Especificación de requisitos 19](#_Toc135865842)

[4.3 Análisis de seguridad 20](#_Toc135865843)

[5 Diseño 21](#_Toc135865844)

[5.1 Arquitectura del Sistema 21](#_Toc135865845)

[5.2 Diseño de subsistema backend 21](#_Toc135865846)

[5.2.2 Diseño de la base de datos 22](#_Toc135865847)

[5.3 Diseño del subsistema frontend 22](#_Toc135865848)

[6 Implementación 22](#_Toc135865849)

[6.1 Implementación de backend 23](#_Toc135865850)

[6.1.1 Estructura e implementación de la lógica de negocio con Play Framework 23](#_Toc135865851)

[6.1.2 Estructura e implementación de la BBDD 23](#_Toc135865852)

[6.2 Referencia al repositorio de software 23](#_Toc135865853)

[6.3 Manuales 23](#_Toc135865854)

[7 Pruebas y validación 23](#_Toc135865855)

[8 Conclusiones y líneas futuras 23](#_Toc135865856)

[Bibliografía 24](#_Toc135865857)

[Anexos 27](#_Toc135865858)

[1. Definición de los recursos de la API 27](#_Toc135865859)

[2. Métodos HTTP de los recursos 29](#_Toc135865860)

Índice de ilustraciones

[Ilustración 1. Metodología en Cascada 3](#_Toc135865803)

[Ilustración 2. Planificación realizada 8](file:///C:\Users\david\Documents\GitHub\tfg_Contenedor\Doc\david-recio-memoria.docx#_Toc135865804)

[Ilustración 3. Servicio Web 9](#_Toc135865805)

[Ilustración 4. Servicio Web Tradicional 10](#_Toc135865806)

[Ilustración 5. Diagrama de una estructura REST 11](file:///C:\Users\david\Documents\GitHub\tfg_Contenedor\Doc\david-recio-memoria.docx#_Toc135865807)

[Ilustración 6. Niveles de madurez de los Servicios Web REST 12](#_Toc135865808)

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 ofrecerá una parte de aplicación para operar con el mediante una interfaz
4. El sistema por parte de la API tendrá una interfaz en la que expondrá los recursos que ofrece, además esta interfaz permitirá probarlos individualmente

# 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 [6].

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 [6], 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 [6].



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 2. Planificación realizada

En comparación del mostrado en el apartado 2.3, en la ilustración anterior [6] 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 y cómo funcionan?

Los Servicios Web son un medio estandarizado para permitir la interacción máquina a máquina a través de una red (cliente, servidor). Está diseñado de forma modular para realizar una serie de tareas concretas, estos pueden buscarse a través de la red e invocarse en consecuencia, proporcionando un servicio concreto al cliente que realizó la petición[4].

Diagrama

Descripción generada automáticamente

Ilustración 3. Servicio Web

En el diagrama anterior se puede ver el funcionamiento de un Servicio Web.

Primero, la aplicación del cliente realizaría una serie de peticiones al Servidor Web donde se encuentran alojados los servicios. Estas peticiones se realizan a través de lo que se conoce como llamadas a procedimientos remotos. Las llamadas a procedimientos remotos son métodos alojados en el Servicio Web correspondiente.

Por ejemplo, Amazon ofrece un servicio web que proporciona los precios de los productos vendidos en línea a través de amazon.com. La aplicación del cliente( la interfaz de usuario de la página web de Amazon) podría estar en un lenguaje completamente diferente al Servidor Web y aun así se pueden comunicar ya que son independientes.

Para diseñar un Servicio Web hay que tener en cuenta que el principal componente son los datos que se transfieren entre el cliente y el servidor, para transferirlos se hace uso de lenguajes de marcado como XML (Extensible markup language) ya que proporciona una plataforma común para que las aplicaciones desarrolladas en varios lenguajes de programación se comuniquen entre sí.

En la actualidad, existen dos estilos para la construcción de Servicios Web:

* Servicios basados en Arquitectura Orientada a Servicios (SOA, Servicios Web tradicionales, basados en protocolo WS-\*)
* Servicios basados en una Arquitectura Orientada a Recursos (Servicios Web RESTful).

### Servicios Web tradicionales

Los Servicios Web tradicionales que se basan en la pila de protocolos de WS-\*,

Utilizan SOAP*(Simple Object Access Protocol)*,que se basa en la transferencia de datos XML como mensajes SOAP[4].

Diagrama

Descripción generada automáticamente

Ilustración 4. Servicio Web Tradicional

Tal como muestra la imagen, el mensaje SOAP necesita un elemento raíz (<Envelope>, siendo este el primer elemento del documento XML) y dentro de este se divide el contenido en dos partes, por un lado, la cabecera, que indica a

qué cliente debe enviarse y, por otro lado, el cuerpo contendrá el mensaje propiamente dicho.

### Servicios RESTful (estilo arquitectónico REST)

El estilo arquitectónico REST (*Representational State Transfer*) lo definió Roy Fielding como “una arquitectura de software para sistemas hipermedia distribuidos”. 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 5. Diagrama de una estructura REST

Las restricciones que impone el estilo arquitectónico REST para poder seguirlo son:

1. **Recurso: Identificación, estado:** Los recursos son la abstracción principal en la que se basa REST, los cuales deben ser únicos e identificables para ello, en el caso de la web, se utilizan las URIs. Las URIs se encargan de exponer al recurso, es decir , actúan como si fueran el nombre y la dirección del recurso, para que sea accesible e identificable entre los distintos recursos. Las URIs al actuar como nombre, no pueden incluir acciones ya que sería filtrar la información del recurso que lo expones en vez de identificarlo claramente .
2. **Representación de un recurso:**  Son los datos y metadatos del recurso que poseen toda la información de este. En el caso de la Web, se pueden solicitar en formato HTML (utilizado para el navegador y por ende seria consumido por el usuario) o en JSON (orientado a que se ha consumido por máquinas).
3. **Hipermedia:** Es una de las restricciones más importante del estilo arquitectónico de REST es *HATEOAS*(*Hypermedia as the Engine of the Application State*), La cual dice que mediante el uso de un hiperenlace (URI), se obtenga la representación de un recurso, que, a su vez, contiene URIs a los diferentes recursos con los que se puede interactuar.
4. **Comunicación:** REST establece una comunicación entre cliente-servidor, dicha comunicación es síncrona, donde el cliente es el encargado de iniciar la comunicación mediante solicitudes (que contienen toda la información necesaria para que el servidor pueda procesarla) a los recursos del servidor, procesará cada solicitud y le devolverá al cliente la respuesta por cada una de ellas.
5. **Interfaz homogénea:** Todos los recursos deben seguir un estándar para que sea más fácil su manipulación, por ello, en el caso de la Web, se utilizan los verbos que proporciona el protocolo HTTP, donde cada uno de ellos nos permiten realizar diferentes cambios en el estado del recurso.

Para una representación del recurso se utiliza el verbo (GET), para la creación de este se usa (POST), para actualizar la información del recurso se usa (PUT), para borrarle se usa (DELETE).

Para resumir, un servicio RESTful es un estilo de arquitectura de software que se utiliza para diseñar servicios web basado en un conjunto de principios y restricciones que permiten a los sistemas comunicarse entre sí a través de HTTP de manera eficiente, escalable y sin estado[5].

### Modelo de madurez de Richardson

Para poder evaluar y clasificar la implementación de los servicios RESTful Leonard Richardson en su artículo titulado "Maturity Model for REST Web Services" ofrece una diferenciación en niveles[6].

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

Descripción generada automáticamente

Ilustración 6. 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*).

#### **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.

#### **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.

#### **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.

# 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.

Escala de tiempo

Descripción generada automáticamente

Ilustración 7. 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 2. Tasas de abandono en el primer año universitario.

Como se puede observar, tanto el porcentaje de abandono como el porcentaje de cambio de estudios sigue una progresión creciente. Con 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, el grado de concentración, la atención/retención e impulsividad 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 formularios utilizados en el estudio psicológico entorno al estudiante, pero dado el alcance del proyecto, solo se realizarán dos tipos de formularios estandarizados: test de aptitudes y test de concentración; que serán informatizados.

El “*Test de Orientación Vocacional CHASIDE*” de Holland Ríase es un formulario 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[1].

Imagen que contiene gabinete, reloj, diferente, pantalla

Descripción generada automáticamente

Tabla 3. Evaluación CHASIDE

El “*Test de Toulouse”* de E. Toulouse y H. Piéron es un formulario 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[2]:

**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

# Diseño

## Arquitectura del Sistema

## Diseño de subsistema backend

#### **Definición de los recursos**

#### **Atributos de los recursos**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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 | - | - | 404-Not Found |
| PATCH | /usuarios | - | - | 404-Not Found |

Tabla 5. 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

## Diseño del subsistema frontend

# 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

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

### Estructura e implementación de la 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, 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

# Bibliografía

1. 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. Guru99. What are Web Services? Architecture, Types, Example [https://www.guru99.com/Web-service-architecture.html](https://www.guru99.com/Web-service-architecture.html%20) (Último acceso: mayo 2023).
5. Fielding, R. T. (2000). Architectural Styles and the Design of Network-based Software Architectures. Tesis de doctorado. University of California, Irvine. Disponible en: <https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>
6. Fowler, M. (2010, March 18). Richardson Maturity Model.<https://martinfowler.com/articles/richardsonMaturityModel.html>. (último acceso mayo 2023)
7. Instinto Programado.com ¿Qué es el modelo de madurez de Richardson? <https://www.instintoprogramador.com.mx/2021/01/que-es-el-modelo-de-madurez-de.html> (último accesomayo 2023)

# 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 |