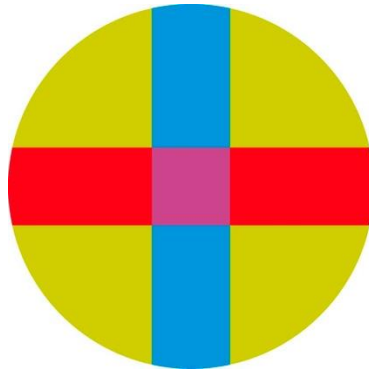


UNIVERSIDAD SAN PABLO - CEU
ESCUELA POLITÉCNICA SUPERIOR
GRADO EN INGENIERÍA DE SISTEMAS DE INFORMACIÓN



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

Autor: David Recio Arnés
Tutor: Sergio Saugar García

Junio 2023



Calificación del Trabajo Fin de Grado

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/2023, acuerda otorgar al Trabajo Fin de Grado presentado por D./Dña. 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 la disyuntiva de elegir un grado universitario basándose generalmente 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 Trabajo de Fin de Grado (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 mediante dos formularios; uno dónde se orienta al estudiante sobre los grados universitarios (ingeniería, ciencias sociales, artes y otras ciencias), y el 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 con relación a la dificultad de elegir el grado que desea.

Palabras Clave

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

Abstract

Currently, young people who are in the process of accessing the university face the dilemma of choosing a university degree, generally based on their vocation. Many of them are wrong in their choices, which is reflected in the abandonment of the first year of the degree, being this point the central motivation for the realization of this Final Degree Project (TFG), because through the creation of a specialized RESTful Web Service the student is guaranteed the proper guidance for the choice of their degree, through standardized forms with psychological, psycho-technical and pedagogical bases, ensuring greater continuity and thus avoiding university dropout. The RESTful Web Service developed in this TFG provides an optimal service through two forms; one where the student is oriented about university degrees (engineering, social sciences, arts and other sciences), and the other form that measures the student's concentration capacity to see how much effort it may take to choose the career he/she wants. Finally, the user's notes are taken, and a final answer is given with suggestions regarding the difficulty of choosing the desired degree.

Keywords

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

Índice de contenidos

Capítulo 1 Introducción.....	1
1.1 Objetivos.....	2
Capítulo 2 Gestión del proyecto	3
2.1 Modelo de ciclo de vida.....	3
2.2 Papeles desempeñados en el proyecto.....	5
2.2.1 Roles del tutor	5
2.2.2 Roles del estudiante	5
2.3 Planificación.....	5
Capítulo 3 Estado del arte	7
3.1 ¿Qué son los Servicios Web y cómo funcionan?	7
3.1.1 Servicios Web tradicionales.....	8
3.1.2 Servicios RESTful (estilo arquitectónico REST).....	9
3.1.3 Modelo de madurez de Richardson.....	11
Capítulo 4 Análisis	15
4.1 Análisis de dominio.....	15
4.2 Especificación de requisitos	20
4.3 Análisis de seguridad	21
Capítulo 5 Arquitectura y Diseño del sistema.....	23
5.1 Arquitectura del sistema	23
5.2 Diseño del subsistema Backend	24
5.2.1 Diseño de los servicios RESTful.....	25
5.2.2 Diseño de la base de datos	29
5.2.3 Diseño del Algoritmo	30
5.3 Diseño del subsistema Frontend	35
Capítulo 6 Implementación.....	38
6.1..... Implementación de Backend	38
6.1.1 Estructura e implementación de la lógica de negocio con .Net Core.....	39

6.1.2 Estructura e implementación de la BBDD.....	40
6.2 Implementación de Frontend	41
6.3 Referencia al software y despliegue	41
6.3.1 Despliegue	41
6.4 Manuales	43
Capítulo 7 Pruebas y validación.....	45
Capítulo 8 Conclusiones y líneas futuras	55
8.1 Líneas futuras.....	57
Bibliografía	59
Anexo	60
Tablas de los recursos	60

Índice de ilustraciones

Ilustración 1. Metodología en Cascada	3
Ilustración 2. Servicio Web.....	7
Ilustración 3. Mensaje SOAP	9
Ilustración 4. Diagrama de una estructura REST.....	9
Ilustración 5. Niveles de madurez de los Servicios Web REST	11
Ilustración 6. Arquitectura del proyecto	24
Ilustración 7. Diagrama entidad relación	29
Ilustración 8. Diagrama de flujo de Recomendaciones	33
Ilustración 9. Diagrama de diseño Registrarse	36
Ilustración 10. Diagrama de diseño páginas internas	36
Ilustración 11. Diagrama de Implementación.....	42
Ilustración 12. URI de Preguntas.....	45
Ilustración 13. Resultados de la llamada a Preguntas	46
Ilustración 14. URI de UsuarioAsignatura	46
Ilustración 15. Respuesta de la petición UsuarioAsignatura	47
Ilustración 16. Registro de la aplicación	47
Ilustración 17. JSON de inserción de usuario.....	48
Ilustración 18. Inserción en la base de datos.....	48
Ilustración 19. Página inicial dentro de la aplicación	48
Ilustración 20. Opciones del navegador lateral	49
Ilustración 21. Formulario CHASIDE en la aplicación	49

Ilustración 22. Guardado de los datos del formulario CHASIDE en la aplicación .	50
Ilustración 23. Desplegable del Perfil de Usuario.....	50
Ilustración 24. Intereses y aptitudes del alumno	51
Ilustración 25. Formulario de Toulouse en la aplicación.....	52
Ilustración 26. Página de asignaturas de la aplicación	52
Ilustración 27. Inserción de la asignatura por parte del Alumno	53
Ilustración 28. tabla asignaturas.....	53
Ilustración 29. Información de la asignatura previa a la actualización	53
Ilustración 30. Actualización de la asignatura	54
Ilustración 31. Información de la asignatura después de la actualización.....	54
Ilustración 32. Borrar la asignatura	54

Capítulo 1

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 lo condicionan, tales como: factores de comportamiento (hábitos de estudio), afectivos (nivel de satisfacción), y motivacionales (internos y externos). Antes de iniciar la universidad, el estudiante se encuentra con una serie de dudas. En primer lugar, la elección de la titulación, para facilitar esta tarea, las universidades realizan unas jornadas de orientación para que los estudiantes de segundo de bachillerato conozcan la universidad de la mano de algunos docentes y de la delegación de cada facultad, es decir, de algunos estudiantes.

Para la elección de la carrera también es fundamental conocer la vocación y las aptitudes del estudiante. La vocación tiene carácter intrínseco, por tanto, no puede evaluarse de la misma forma que las aptitudes.

Las aptitudes deberían alinearse con la carrera seleccionada para así obtener el mayor rendimiento posible. Para esto, se pueden realizar unos formularios estandarizados, cuyos resultados servirán de recomendación para elegir mejor una titulación. Esto es fundamental, dado que los estudiantes tienen que abordar una carga de trabajo, y una planificación a la cual no están acostumbrados.

1.1 Objetivos

Para abordar las dificultades que se encuentra el estudiante antes de comenzar su primer año de universidad se ha establecido los siguientes objetivos:

General:

Crear un Servicio Web RESTful que asesore y acompañe al estudiante mediante recomendaciones durante ese período.

Específicos:

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 él 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.

Capítulo 2

Gestión del proyecto

2.1 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 [1].

La versión original fue presentada por Royce en 1970, aunque son más conocidos los trabajos realizados por Boehm en 1981, Sommerville en 1985 y Sigwart 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.

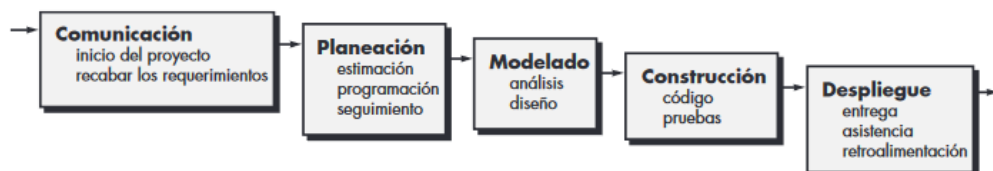


Ilustración 1. Metodología en Cascada

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:
 - i. **Diseño del sistema.** Se descompone y organiza el sistema en partes separadas, describiendo la estructura del sistema y la funcionalidad de sus partes, así como la manera en que se combinan unas con otras.
 - ii. **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.



2.2 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.

2.2.1 Roles del tutor

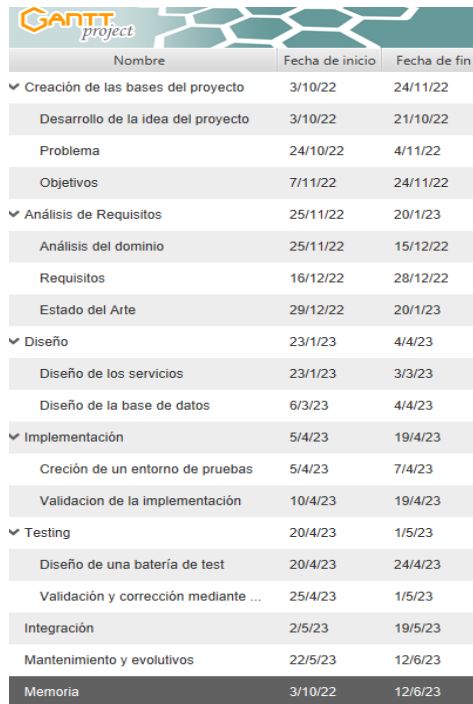
El tutor ha realizado dos roles. Por un lado, el rol de director del proyecto, ya que ha participado en la planificación y definición de objetivos; por otro lado, de analista de requisitos, ya que ayudó a establecer los requisitos de la aplicación.

2.2.2 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.

2.3 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.



Nombre	Fecha de inicio	Fecha de fin
✓ Creación de las bases del proyecto	3/10/22	24/11/22
Desarrollo de la idea del proyecto	3/10/22	21/10/22
Problema	24/10/22	4/11/22
Objetivos	7/11/22	24/11/22
✓ Análisis de Requisitos	25/11/22	20/1/23
Análisis del dominio	25/11/22	15/12/22
Requisitos	16/12/22	28/12/22
Estado del Arte	29/12/22	20/1/23
✓ Diseño	23/1/23	4/4/23
Diseño de los servicios	23/1/23	3/3/23
Diseño de la base de datos	6/3/23	4/4/23
✓ Implementación	5/4/23	19/4/23
Creación de un entorno de pruebas	5/4/23	7/4/23
Validación de la implementación	10/4/23	19/4/23
✓ Testing	20/4/23	1/5/23
Diseño de una batería de test	20/4/23	24/4/23
Validación y corrección mediante ...	25/4/23	1/5/23
Integración	2/5/23	19/5/23
Mantenimiento y evolutivos	22/5/23	12/6/23
Memoria	3/10/22	12/6/23

Ilustración 2. Planificación real

Como se puede observar en la ilustración anterior, el motivo de que aparezca a partir de 3/10/22 es debido a que se realizó una primera versión el anterior año, pero no hubo tiempo de terminarla, por ende, se eliminaron los elementos erróneos y se cambió de tecnología. Una vez dicho esto, se pueden observar tres grandes periodos de tiempo, siendo la creación de las bases del proyecto, se prolongó tanto en el tiempo debido a que se solventaron los elementos erróneos de la antigua versión ajustando los objetivos, sigue el periodo de análisis de requisitos, en este periodo se extrajo del proyecto anterior tanto el análisis del dominio y el estado del arte, pero se rediseñaron los requisitos en base a los nuevos objetivos, por último queda el periodo de diseño el cual posee la mayor duración, esto se debe a que se tuvo que rediseñar los servicios y la base de datos por completo, centrando en la funcionalidad en vez a la tecnología (error que tuve en la anterior versión).

Cabe destacar que el periodo de integración fue en parte alargado dado a un proceso de aprendizaje de las herramientas de AWS.

Capítulo 3

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.

3.1 ¿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 [2].

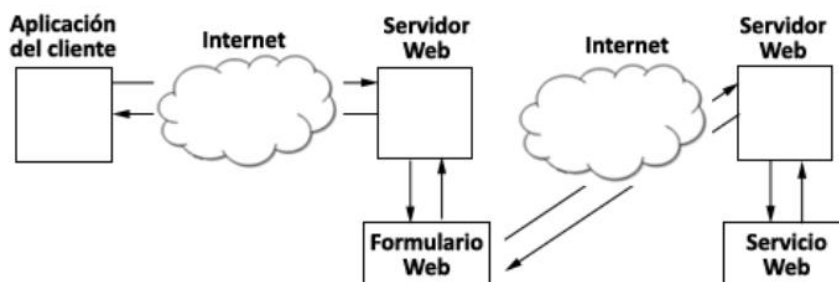


Ilustración 2. Servicio Web

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).

3.1.1 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[2].

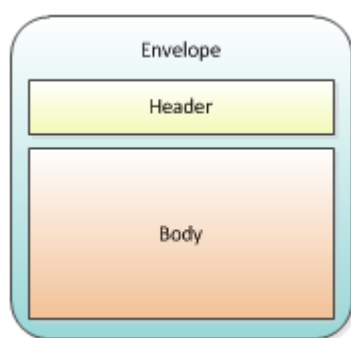


Ilustración 3. Mensaje SOAP

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 al Cliente que debe enviarse y, por otro lado, el cuerpo contendrá el mensaje propiamente dicho.

3.1.2 Servicios RESTful (estilo arquitectónico REST)

El estilo arquitectónico REST (*Representational State Transfer*) lo definió Roy Fielding como “una arquitectura 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.

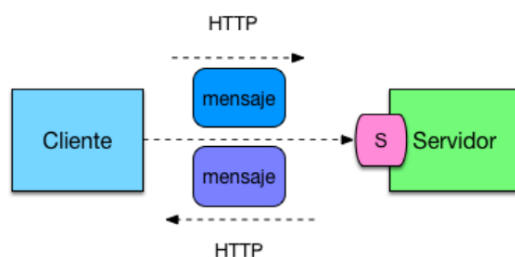


Ilustración 4. Diagrama de una estructura REST

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

1. **Recurso: Identificación:** 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 expone 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 sería 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 software que se utiliza para diseñar servicios web basados 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[1].

3.1.3 Modelo de madurez de Richardson

Para poder evaluar y clasificar la implementación de los Servicios Web RESTful Leonard Richardson en su artículo titulado "Maturity Model for REST Web Services" ofrece una diferenciación en niveles[3].

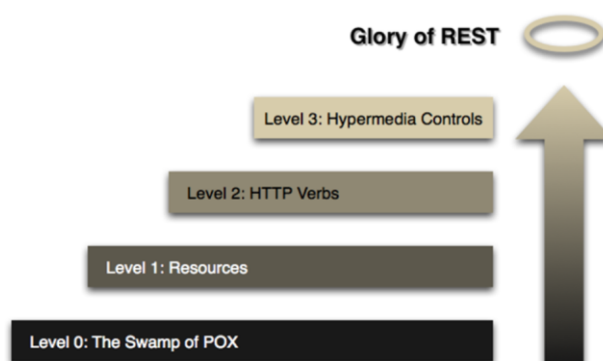


Ilustración 5. Niveles de madurez de los Servicios Web REST

3.1.3.1 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 transmitir las interacciones entre los propios mecanismos, normalmente basados en RIP (*Remote Procedure Invocation*).

3.1.3.2 Nivel 1

Es la primera etapa que 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.

3.1.3.3 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.



3.1.3.4 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.

Capítulo 4

Análisis

4.1 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.

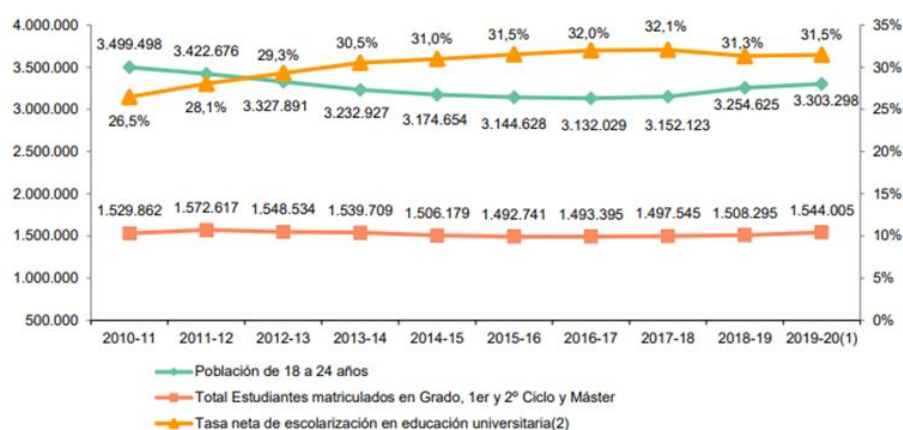


Ilustración 7. Población de matriculados en universidades.

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:

	Cohorte 2014-2015		Cohorte 2015-2016		Cohorte 2016-2017	
	Abandono del estudio en 1º año	Cambio del estudio en 1º año	Abandono del estudio en 1º año	Cambio del estudio en 1º año	Abandono del estudio en 1º año	Cambio del estudio en 1º año
Total	21,5%	8,2%	21,7%	8,6%	21,8%	8,7%
Rama de enseñanza						
Ciencias Sociales y Jurídicas	20,1%	7,2%	20,4%	7,4%	20,4%	7,6%
Ingeniería y Arquitectura	25,3%	10,7%	25,1%	11,8%	25,2%	11,0%
Artes y Humanidades	27,7%	9,2%	28,4%	9,1%	28,6%	9,3%
Ciencias de la Salud	17,1%	6,6%	17,4%	7,1%	17,7%	7,5%
Ciencias	22,2%	10,9%	22,1%	11,3%	23,2%	12,6%

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 [4].

C	H	A	S	I	D	E							
98	9	21	33	75	84	77	← Intereses						
12	34	45	92	6	31	42							
64	80	96	70	19	48	88							
53	25	57	8	38	73	17							
85	95	28	87	60	5	93							
1	67	11	62	27	65	32	C	H	A	S	I	D	E
78	41	5	23	83	14	68	15	63	22	69	26	13	94
20	74	3	44	54	37	49	51	30	39	40	59	66	7
71	56	81	16	47	58	35	2	72	76	29	90	18	79
91	89	36	52	97	24	61	46	86	82	4	10	43	55

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[5]:

$$\text{IGAP} = \text{ACIERTOS} - (\text{ERRORES} + \text{OMISIONES})$$

$$\text{CC} = \text{A} - \text{E} / \text{A} + \text{O}$$

$$\text{ICI} = \text{ACIERTOS} - \text{ERRORES} / \text{RESPUESTAS} \times 100$$

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

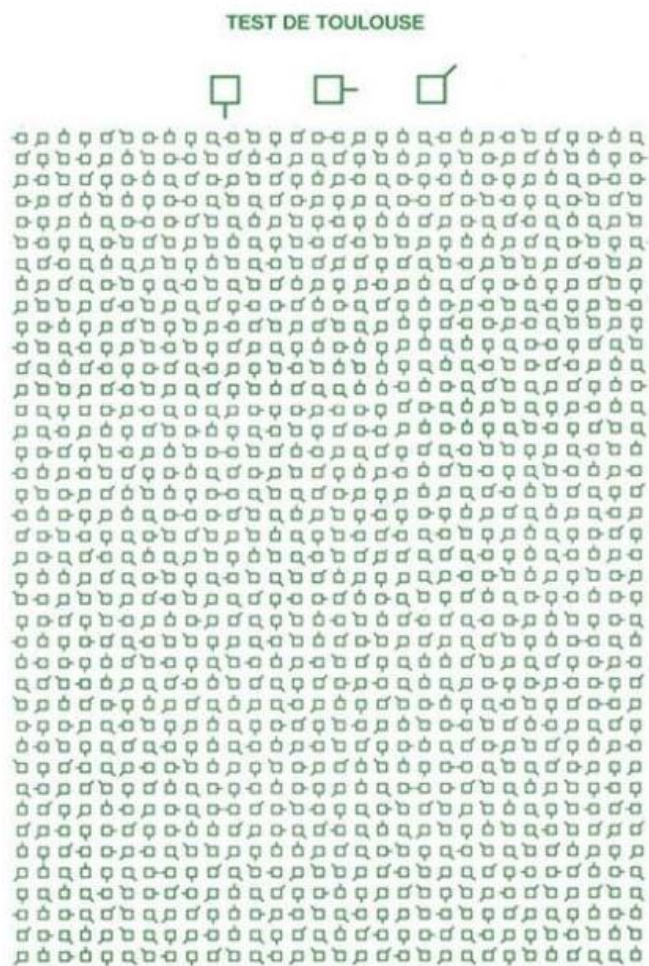


Ilustración 13. Test de Toulouse

4.2 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.

- **Interoperabilidad con otros sistemas**

- Debe ser accesible desde cualquier dispositivo (*tablets*, móviles, otras aplicaciones, etcétera).
- Debe ser ajena a como estén formados los sistemas que acceden a dichos recursos.
- Debe usar JSON como formato para enviar y transcribir los datos.

- **Protección de datos**
 - Deben estar las id de los usuarios en formato GUID.
 - Deberán transmitirse los datos a aquellas personas autorizadas.
- **Requisitos sobre entorno tecnológico y de comunicaciones**
 1. Debe utilizar los métodos HTTP para interactuar con los demás recursos.
 2. Debe haber una interfaz operable por el usuario.

4.3 Análisis de seguridad

En este análisis se buscará cubrir lo posible tratándose en el contexto que se realiza (plazo de tiempo, recursos y alcance del trabajo). Para ello primero se definirá las dimensiones que abarcará:

- **Integridad:**

La facilidad que un tercero no autorizado corrompa la información. La forma que el aplicativo tenga integridad se usa el Login previo para que en todo momento el usuario se encuentra identificado durante la sesión.
- **Autenticidad:**

Ser capaz de saber si la información no fuera la producida en el origen, es decir, que se pueda suplantar. Para saber el origen se pide en todo momento la autorización de acceso (Token Bearer), que es proporcionada en el momento que entra en la aplicación.

Capítulo 5

Arquitectura y Diseño del sistema

Para desarrollar la aplicación, en primer lugar, se ha elegido una arquitectura que permita cumplir los requisitos no funcionales que se han establecido. Esa arquitectura está basada en Servicios Web RESTful. Después, para desarrollar la lógica de la aplicación, se ha realizado un diseño de los Servicios RESTful (que se abordará en los siguientes subcapítulos).

Posteriormente se ha realizado el modelo de la base de datos de manera que recoja los datos necesarios para el funcionamiento de la aplicación, por último, se ha diseñado una interfaz amigable para la interacción con el usuario con la máquina.

5.1 Arquitectura del sistema

Para poder cubrir el análisis del capítulo 4, se ha elegido una arquitectura que se divide en dos partes. El Frontend que es el encargado de alojar la interfaz donde el usuario interactúa, mientras el Backend, compuesto por dos servidores, cumplirá con la función de aportar la lógica de la aplicación y la persistencia de los datos.

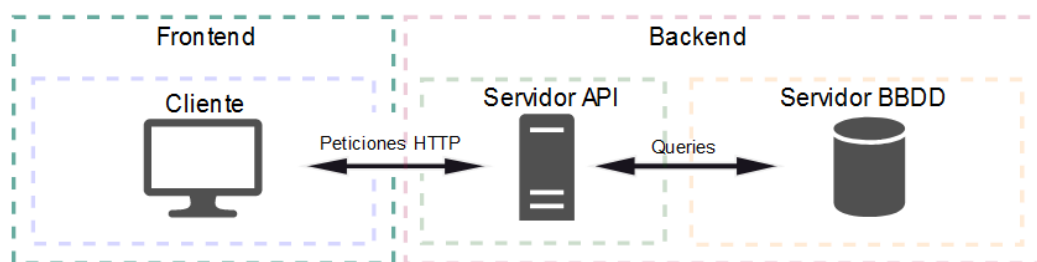


Ilustración 6. Arquitectura del proyecto

Como se puede ver en la Ilustración 6, las capas principales Frontend y Backend se comunican entre si mediante peticiones HTTP, siguiendo el modelo cliente-servidor, donde el cliente se constituye por una aplicación web en la que el usuario realizará acciones que desembocará en peticiones al servidor, el cual generará una respuesta visible en la interfaz del cliente. Respecto al Backend está compuesto por dos servidores, debido a la necesidad de la abstracción de la base de datos, permitiendo que en el caso de realizar un cambio sobre el servidor de BBDD o sobre el servidor API no exista una dependencia que obligue a realizar múltiples cambios en el otro servidor.

Para finalizar el Servidor API es el encargado de alojar la API RESTful, dedicada a recibir las peticiones del cliente y mediante el uso de las consultas obtener los datos necesarios que al procesarlos ofrecerá una respuesta acorde a la petición. Dicha respuesta será mostrada en la interfaz del cliente y en el caso de que sea necesario su persistencia, se almacenará en la base de datos.

5.2 Diseño del subsistema Backend

En este subcapítulo se detallará el diseño de la arquitectura del Servidor Web y de la base de datos, describiendo tanto la lógica de la API como de la estructura del sistema de almacenamiento de datos.



La API RESTful es la encargada de recibir las peticiones HTTP de la aplicación por lo tanto es necesario realizar una definición de los recursos a la vez que se definen los atributos de clases que soportaran dichos servicios.

Los diferentes recursos que se encuentren publicados realizaran diferentes interacciones con la base de datos para hacer que sean persistentes.

5.2.1 Diseño de los servicios RESTful

Se han diseñado los Servicios RESTful tras el análisis realizado en el subcapítulo 4.1, los cuales están expuestos de forma individual en diferentes URIs.

En los siguientes subcapítulos se representarán las tablas de los recursos más importantes, las restantes quedarán representadas en el anexo.

5.2.1.1 Definición de los recursos

Un recurso es una entidad que posee un conjunto de datos expuestos a través de una URI. Los recursos expuestos a continuación tienen la función de dotar toda la lógica necesaria a la aplicación siendo estos:

- **Respuestas ("/usuarios/{idUsuario}/formularios/{idFormulario}"):** almacena las respuestas de los formularios CHASIDE y Toulouse del usuario, para poder almacenarla hace uso del atributo **Valor**.
- **Preguntas("/formularios/{idFormulario}"):** almacena las preguntas de los formularios CHASIDE y Toulouse, para poder almacenarlas correctamente se usa el atributo **Tipo** para poder saber a qué formulario pertenece la pregunta, el atributo **Contenido**, el cual almacena la pregunta en sí y, por último, el atributo **imagen_url** utilizado para almacenar la URL de la imagen.
- **Usuario("/usuarios/{idUsuario}"):** Este es el recurso que identifica a cada usuario y almacena los resultados de los formularios. En él, se almacena su nombre en el atributo **Nombre**, el atributo **Pass** donde se almacena la contraseña, para poder guardar los datos del formulario CHASIDE se hace uso

de los atributos **Administrativas_Contables**, **Humanisticas_Sociales**, **Artísticas**, **Medicina_CsSalud**, **Ingenieria_Computacion**, **DefensaSeguridad** y **CienciasExactas_Agrarias** junto con el uso de un sufijo **_int** para los intereses y **_apt** para las aptitudes. Por último, almacena los resultados del Formulario de Toulouse en los atributos **CC** para la concentración, **IGAP** para la capacidad de atención y retención e **ICI** para la impulsividad.

- **Usuarios("/usuarios")**: se encarga de listar a los usuarios existentes y de añadir nuevos, para ello solo posee tres atributos, el atributo nombre, el atributo **Pass** y el atributo **Url**, este último es el que almacena la URL del usuario para poder acceder al recurso usuario
- **UsuarioAsignatura("/usuarios/{idUsuario}/asignaturas/{idAsignatura}")**: guarda los datos referentes a dicha relación insertados por parte del usuario como la **Nota** y el **TiempoEstudio**, además de los datos generados por las recomendaciones **Riesgo**, **TiempoRecomendado**.
- **UsuarioAsignaturas("/usuarios/{idUsuario}/asignaturas")**:
Se encarga de listar las asignaturas que tiene asociado el usuario, para realizarlo, este recurso posee el atributo nombre, el cual es el propio nombre de la asignatura y el atributo **Url**, utilizado para almacenar la URL de la asignatura asociada al usuario para poder acceder al recurso **"usuarioAsignatura"**.
- **Asignaturas("/asignaturas")**: permite listar y añadir asignaturas para que puedan ser asociadas a los usuarios, para ello almacenan el atributo nombre donde se guarda el nombre de la asignatura.

A continuación, se mostrará una tabla del recurso más importante del proyecto, las demás tablas se encontrarán en el anexo.



Recurso: UsuarioAsignatura				
URI: /usuarios/{idUserio}/asignaturas/{idAsignatura}				
Método	Descripción	Cuerpo de la solicitud	Código de respuestas	Cuerpo de la respuesta
GET	Solicitud de obtención de la asignatura escogida perteneciente al usuario	-	200 OK	Obtiene la asignatura escogida perteneciente al usuario
			500 Internal error	-
POST	Solicitud de asociar la asignatura escogida al usuario	Representación del UsuarioAsignatura para asociar en formato JSON	200 OK	Asocia la asignatura escogida al usuario
			500 Internal error	-
PUT	Solicitud de modificar la nota o el tiempo de estudio de la asignatura	Representación del UsuarioAsignatura para modificar en formato JSON	200 OK	Modifica la nota o el tiempo de estudio de la asignatura
			500 Internal error	-
DELETE	Solicitud de borrado la asociación	-	200 OK	Borra la asociación
			500 Internal error	-
PATCH	-	-	405: Method not Allowed	-

Como se puede observar en la tabla anterior los métodos por los cuales puedes realizar la petición al servicio son: GET, POST, PUT y DELETE que en el caso de realizar la petición correctamente y que no se produzca ningún error devolverán un código 200 junto con el resultado de la petición y en el caso de lo contrario, se obtendrá un error 500, para el caso de PATCH al intentar realizar una petición devolverá que ese método no se encuentra disponible, además tanto el POST como el PUT necesitan de un JSON aparte de la URI para realizar la petición correctamente, ya que los datos del JSON son necesarios para llevarla a cabo. Por último, en la columna de la descripción queda detallado lo que realiza el método y en cuerpo de la respuesta lo que devolverá según si se han producido errores o no, además el resultado n caso de ser satisfactorio devolverá la información en formato JSON para el caso del GET y el POST mientras que para el PUT y el DELETE solo enviara un mensaje de que se ha completado.



5.2.2 Diseño de la base de datos

Para dar soporte a los datos recibidos de la API, se ha procedido a la creación de tablas que se expondrán a continuación mediante el uso del diagrama entidad relación.

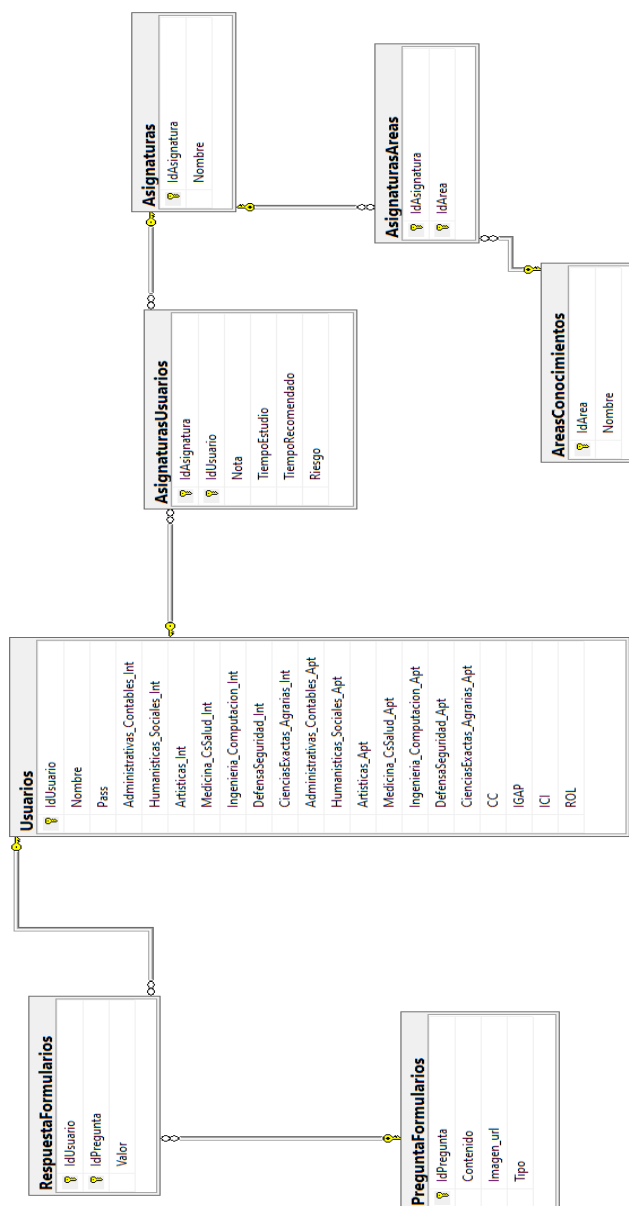


Ilustración 7. Diagrama entidad relación

En el diagrama destacan principalmente dos factores. El primero sería el diseño de la tabla “**PreguntaFormularios**”, es la única que posee un campo que almacena

URL, debido a que las imágenes pesan demasiado supondrían una penalización a la eficiencia tanto a la extracción de estas, como a la conversión de cadena de bits a imagen dentro de la aplicación, por ello se tomó la decisión, de cargar en la aplicación las imágenes alojadas en un servidor a través de una URL propia almacenada en la base de datos

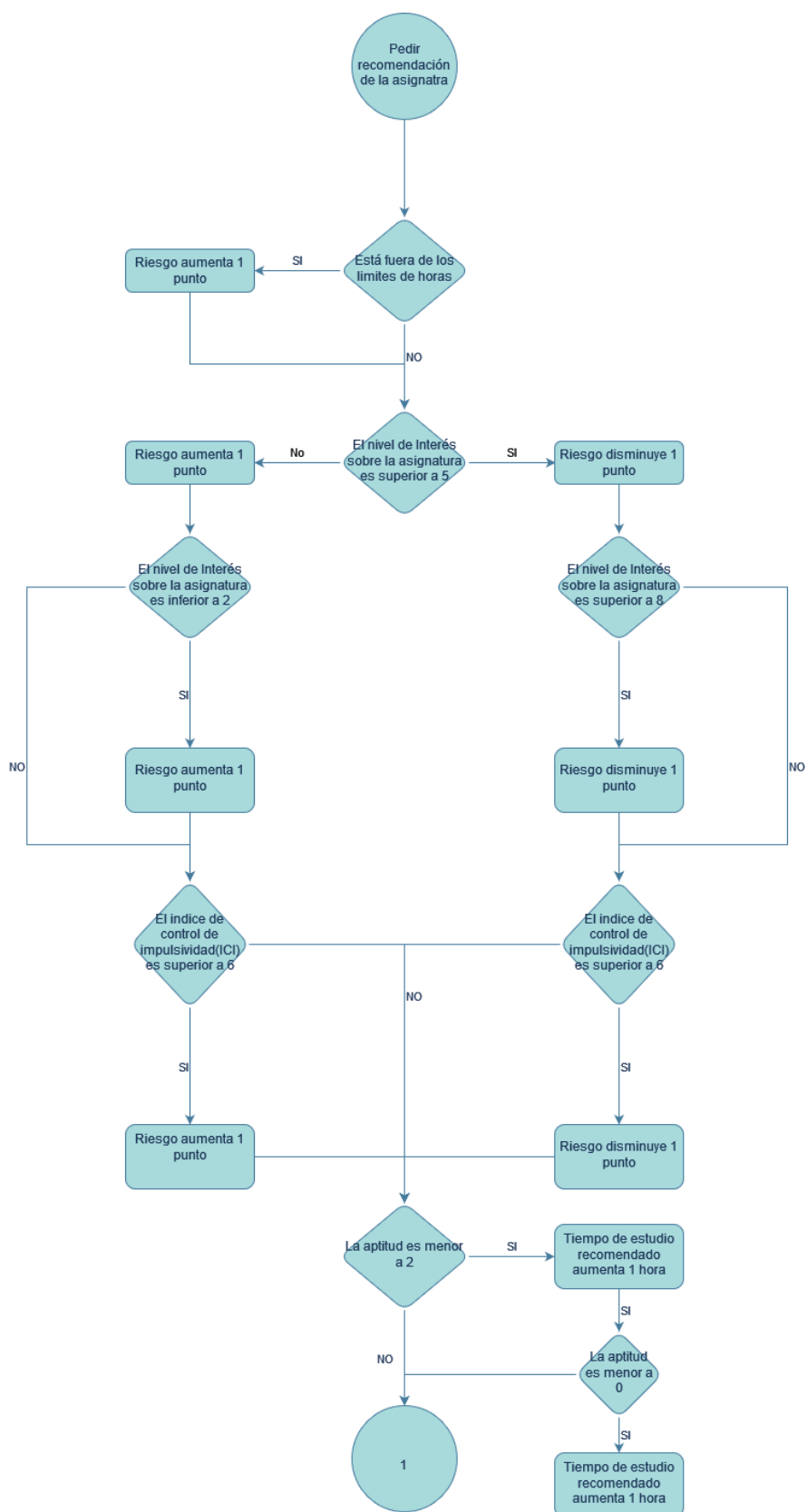
Por último, el otro factor hace referencia en la estructura de las relaciones entre **“Usuario”**, **“RespuestaFormularios”** y **“PreguntaFormularios”**, los cuales tienen esta relación debido al carácter invariable de la tabla **“PreguntaFormularios”**, expuesto anteriormente.

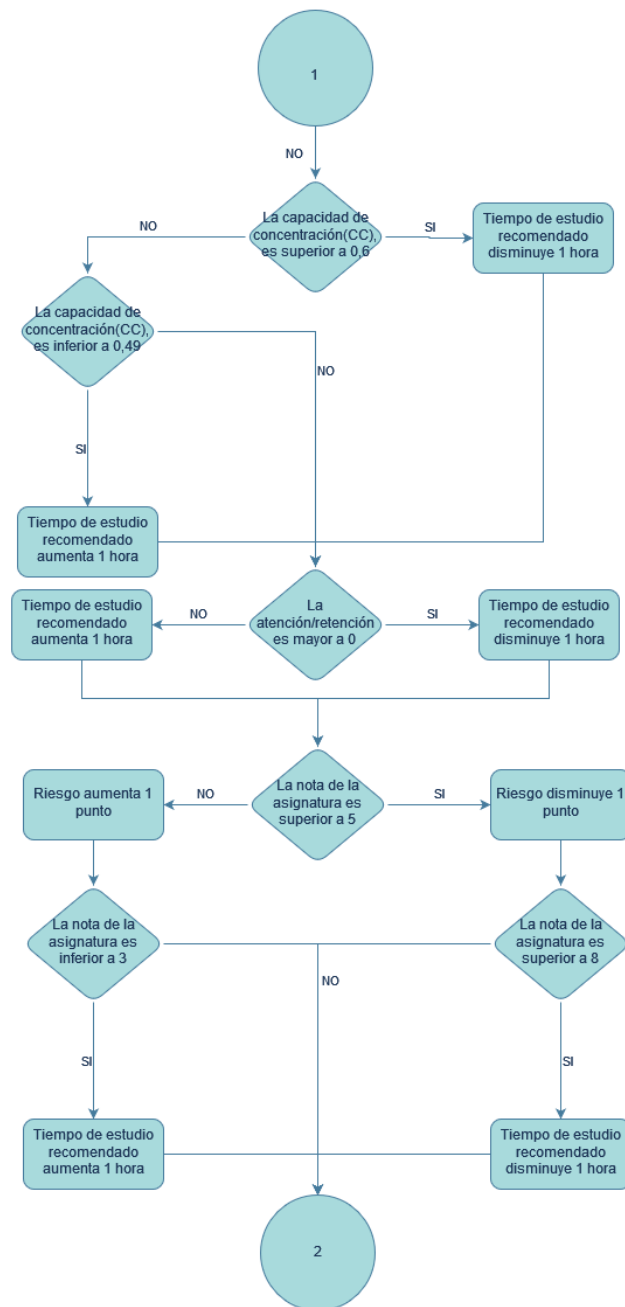
Otras tablas importantes son **“RespuestaFormularios”** y **“AsignaturasUsuarios”** que además de ser el nexo de unión entre dos tablas almacenan información del propio usuario (respuesta de la pregunta asociada, nota y tiempo de estudio del usuario) y en el caso de esta última, de la información elaborada por el algoritmo (tiempo de estudio recomendado y el riesgo).

5.2.3 Diseño del Algoritmo

En este subcapítulo se expondrá la lógica tras las recomendaciones generadas, explicando la entrada de los datos y el proceso de obtención de los resultados.

Los datos de entrada son las respuestas de los formularios CHASIDE y Toulouse, que por medio de las fórmulas detalladas en el subcapítulo 4.1 se obtienen las aptitudes, intereses, retención de la información, impulsividad y concentración, todos estos resultados serán utilizados junto con la nota y el tiempo de estudio de la asignatura, para la obtención de las recomendaciones siguiendo los siguientes pasos en orden.





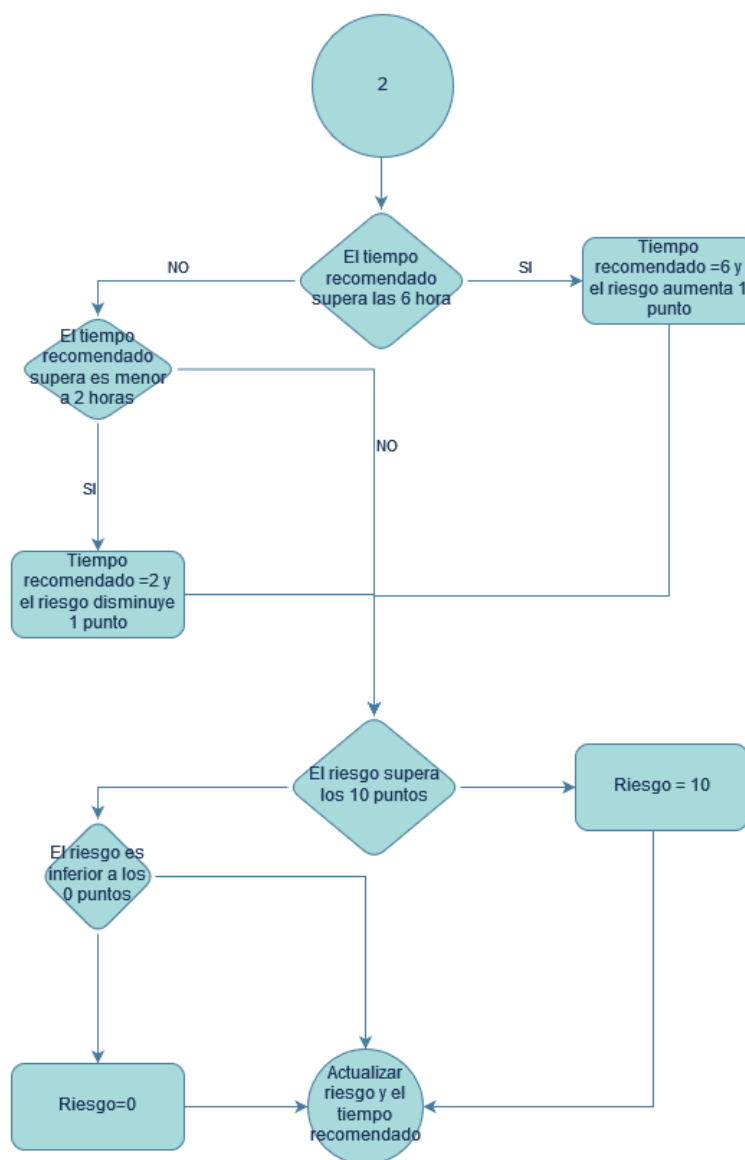


Ilustración 8. Diagrama de flujo de Recomendaciones

Primero se evaluará si el tiempo de estudio se encuentra entre las 2 y 6 horas semanales, en caso de no serlo el riesgo aumenta un punto, ya que el tiempo se encuentra fuera del rango recomendado.

Segundo, según el interés de las áreas a las que corresponda dicha asignatura aumentará o disminuirá el riesgo de suspender la asignatura[6] en base a la siguiente tabla.

Interés	Riesgo
Mayor a 6	Disminuye un punto
Mayor a 8	Disminuye un punto
Menor a 5	Aumenta un punto
Menor a 2	Aumenta un punto

Tabla 1. Riesgos e intereses

Tercero, si el alumno es impulsivo, es decir, su valor de impulsividad es superior a 0,6, y además no posee interés sobre la asignatura (menor a 4), el riesgo aumenta un punto dado que el alumno evitará ponerse a estudiar[11], pero si el interés es alto, el riesgo disminuye dado que el alumno estará interesado en profundizar en la asignatura[10]. Cuarto, según su aptitud, si es menor que dos, el tiempo de estudio recomendado aumenta una hora y en caso de ser menor a 0 la aptitud, el tiempo de estudio recomendado aumenta dos horas [7].

Quinto, Si la concentración es mayor a 0,6 significa que tiene una gran capacidad de concentración, por ende, necesita menos tiempo de estudio [9], desembocando en reducir las horas de estudio recomendadas en 1, en el caso de que la concentración sea menor a 0,49 el tiempo de estudio recomendado aumentara en 1. Sexto, si la retención es mayor a 1 tiene una capacidad de retención alta con lo que el tiempo recomendado que requiere el estudiar la asignatura disminuye 1 hora [8], mientras que si es inferior a 0 el tiempo recomendado aumenta en 1. Séptimo, según la nota obtenida tanto el tiempo de estudio recomendado como el riesgo puede aumentar o disminuir, para ello se han establecido los siguientes rangos. Si la nota se encuentra por debajo o igual a 3 el riesgo y el tiempo de estudio recomendado aumentan un punto, si está por debajo del 5 pero por encima del dos, solo aumenta el riesgo ya que es más próximo al aprobado, si la nota es un 5 no es relevante y se descarta, si es mayor a 5 e inferior o igual a 8 el riesgo disminuye un punto, pero se le mantiene el tiempo de estudio recomendado con la intención de que mejore, en cambio sí es



mayor que un 8 el tiempo de estudio recomendado disminuye, debido que, al sacar buena nota, puede dedicar parte del tiempo de estudio de esta asignatura en otra asignatura con mayor riesgo.

Por último, se rectificarán los resultados, es decir si el tiempo recomendado es menor al límite inferior, se le recomendará el tiempo mínimo y el riesgo disminuirá un punto, en caso de superar el límite superior, el riesgo aumentará un punto y se le asignará el tiempo máximo recomendado. Para evitar que el riesgo supere la franja entre 0 y 10, el caso de superarla se pondrá los límites respectivamente.

Tras generar los cambios con el algoritmo se le asignará las recomendaciones a la asignatura correspondiente.

5.3 Diseño del subsistema Frontend

Es el responsable de mostrar los datos de manera sencilla y fácil para la interacción con el usuario, de tal forma que el alumno pueda registrarse, añadirse asignaturas, realizar los formularios y así obtener recomendaciones.

Para poder realizarlo se han diseñado dos formatos principales:

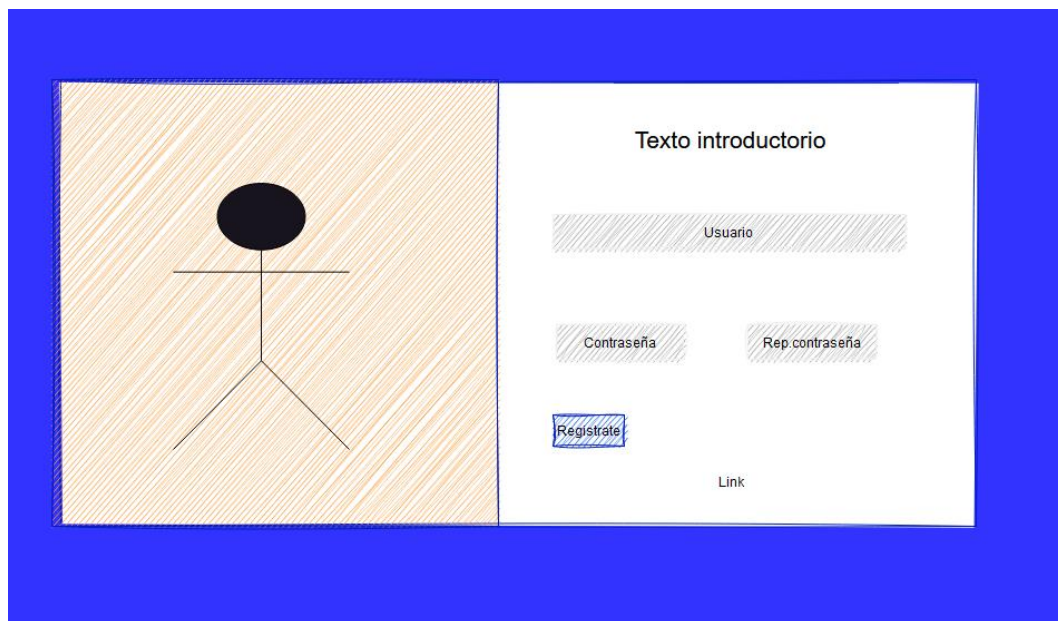


Ilustración 9. Diagrama de diseño Registrarse

Una vista limpia y sencilla con los parámetros de entrada, sin ningún navegador utilizado tanto en la pestaña de *login* como en la de crear una nueva cuenta.

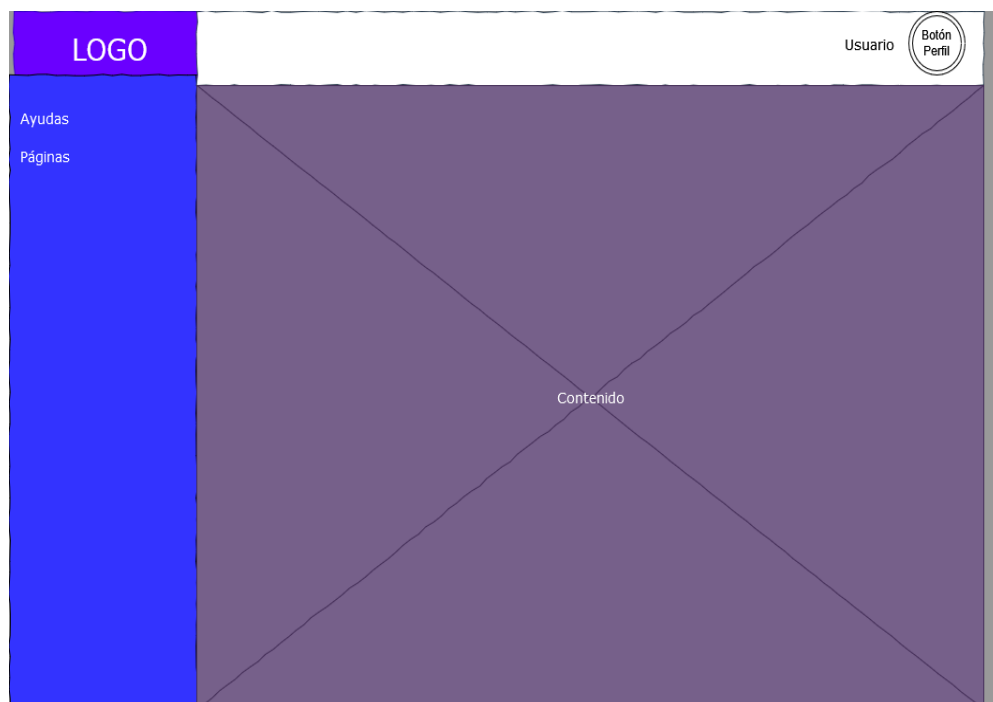


Ilustración 10. Diagrama de diseño páginas internas



En el segundo caso se ha optado con una vista más compleja, pero intuitiva en donde se dispone de dos elementos invariables, el navegador izquierdo donde se encuentran las diferentes paginas asociadas y el navegador superior que contiene el perfil y el cierre de sesión. Respecto al espacio comprendidos entre los dos navegadores, se ocupará con los diferentes recursos para cada página.

Para poder desarrollar la interfaz se realizará mediante el patrón de diseño MVC, compuesto por las Vistas que son las encargadas de mostrar la parte visual al usuario, el Modelo, encargado de albergar la lógica suficiente para llamar a la API y el Controlador, el responsable de atender a las acciones o métodos que requiere la vista y devolver un resultado tanto en forma de petición a la API, como un cambio de página o un evento en la vista.

Capítulo 6

Implementación

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

6.1 Implementación de Backend

Para la implementación de los Servicios RESTful se ha utilizado .Net Core. La estructura de la aplicación se encuentra dividida en tres partes:

- *Controllers*: Clases encargadas de atender las diferentes peticiones HTTP y donde se encuentran las rutas de los recursos expuestos.
- *Entities*: Son las entidades del modelo de la aplicación, cuya función es de obtener y generar en formato JSON los datos necesarios de las peticiones.
- *Services*: Son las clases destinadas a aislar las llamadas a la BBDD.

A la vez que se realizaba la implementación de los servicios, se realizaba la implementación de la BD en SQL Server para dar persistencia a los datos procedentes de la API.



6.1.1 Estructura e implementación de la lógica de negocio con .Net Core

En este subcapítulo se definirá la estructura e implementación realizada en .Net Core. Para poder entender mejor la estructura, durante todo este subcapítulo se hará uso como ejemplo del método **"GetAsignaturasUsuario"**.

```
[HttpGet]
[Route("usuarios/{idUsuario}/asignaturas/{idAsignatura}")]

public async Task<ActionResult> GetAsignaturasUsuario(Guid
idUsuario, Guid idAsignatura)
{

    AsignaturaUsuario nota = await
asignaturaUsuarioBBDD.AsignaturasUsuarios.FindAsync(idAsignatura, idUsuario);

    AsignaturaUsuarioGet asignaturaUsuarioGet = new() {
        UrlAsignatura = new
Uri(Request.GetEncodedUrl().Split("usuarios")[0] + "asignaturas/" +
idAsignatura),
        TiempoEstudio=nota.TiempoEstudio,
        TiempoRecomendado=nota.TiempoRecomendado,
        Riesgo=nota.Riesgo,
        Nota=nota.Nota,
    };

    return Ok(asignaturaUsuarioGet);
}
```

Esta función se encarga de obtener toda la información del usuario en referente a una asignatura, para ello, el usuario realiza una petición HTTP con el método GET a la URI detallada en el *Route* al controlador Asignatura. Elemento que se encarga de exponer los servicios mediante el uso de los patrones de las URIs, una vez accedida a la función, se hace uso de la *Entity* **AsignaturaUsuario** para almacenar los datos recogidos del *Service* **asignaturaUsuarioBBDD**, debido a que las *Entities* son las encargadas de estructurar los datos recibidos por los servicios, los cuales son los encargados de acceder a la base de datos.

Tras recoger los datos en la variable *nota*, se hace uso de la clase **asignaturaUsuarioGet** para enviar los datos al usuario en formato JSON.

6.1.2 Estructura e implementación de la BBDD

Gracias al diagrama E/R realizado en el diseño de la BD en el subcapítulo 5.2.2, se ha realizado su implementación en el servidor SQL gracias a la interfaz gráfica SQL Server Management Studio Management Studio 19.

Debido a que se ha empleado SQL Server como motor de SQL, se han utilizado en casi todas las tablas el tipo GUID como identificador, dado que asegura un valor irrepetible y no secuencial, el único caso donde no se hizo uso fue en la tabla **PreguntasFormulario**. Se realizó con el propósito de simplificar a nivel del desarrollo, la identificación de las preguntas, como caso excepcional, esta tabla tampoco es modificada, ya que contiene las preguntas de los formularios, las cuales son invariables.

Por último, se han creado unos scripts SQL que permiten inicializar la base de datos, con los datos necesarios para el correcto funcionamiento del proyecto. La ubicación de dichos ficheros se encuentra en el directorio del api, dentro de la carpeta SQL.

6.2 Implementación de Frontend

Para la implementación el Frontend, en este proyecto se ha hecho uso de una plantilla de software libre llamada **sb-admin2[12]**. Esto se debe a que es una plantilla intuitiva, fácil de adaptar a las necesidades y requerimientos de la aplicación, minimiza el uso del CSS al centrarse en la utilización de las clases y pose una interfaz orientada a la reutilización del *layout*.

6.3 Referencia al software y despliegue

Todo el software relacionado con este proyecto está disponible en Github a través de la siguiente URL: https://github.com/davidRecio/tfg_Contenedor

La URL para acceder a la aplicación es: <https://54.194.149.25/web>

La URL para acceder a la API es: <https://54.194.149.25/API/swagger/index.html>

6.3.1 Despliegue

Para el despliegue de los componentes del proyecto se ha usado la plataforma de AWS ya que permite el acceso desde cualquier ordenador sin tener que instalar ningún programa. Para desplegar las partes del proyecto se ha seguido el siguiente esquema:

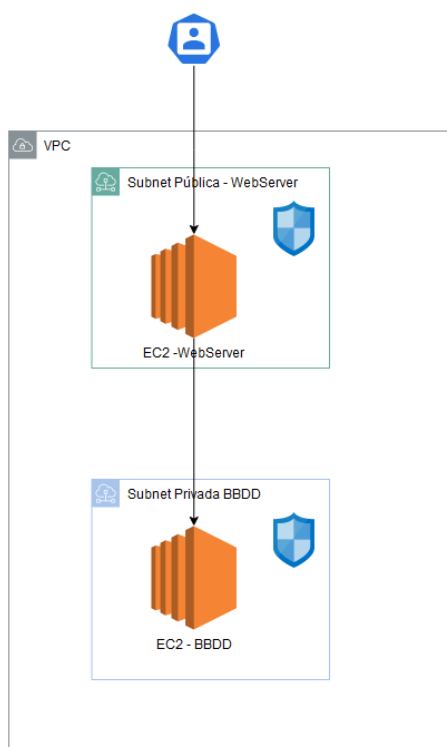


Ilustración 11. Diagrama de Implementación

Para crear el entorno detallado en el diagrama, primero se ha creado el VPC, mediante el servicio de *Cloud Formation*, ya que permite la creación de VPC mediante plantillas, facilitando así la creación de las redes y subredes donde se alojaran las EC2 del proyecto, el motivo por el cual se ha decidido separar en dos máquinas diferentes ha sido para asegurar un entorno aislado y restringido, evitando vulnerabilidades de acceso, ya que solo se puede acceder a ella a partir de EC2 donde se encuentra el IIS. Para la creación de las EC2 se ha mirado los requisitos de hardware que necesita tanto el SQL Server como el IIS, siendo en el caso del EC2 donde se alojara el IIS una máquina t2.medium y para el caso del SQL Server una máquina t3.xlarge. Además, el IIS necesita el sistema operativo Windows para funcionar, con lo que se le proporciono el Windows Server, en el caso de la BBDD se optó por el mismo sistema operativo para facilitar el despliegue. Una vez creadas las maquinas, se crearon los *Security Groups*

representados en el diagrama como “escudos”, encargados de delimitar el acceso a las maquinas.

Una vez levantado el escenario mostrado en el diagrama anterior, se configuro el IIS para que pudiese publicar la aplicación y la API, además de dotarle de un certificado SSL para habilitar las conexiones HTTPS, aunque al ser el mismo la entidad certificadora, sigue apareciendo el aviso de sitio inseguro.

Por último, el fichero de configuración de la VPS se encuentra alojado dentro del repositorio en la carpeta. /Doc/Despliegue.

6.4 Manuales

El manual de uso de la aplicación de este proyecto se encuentra alojado dentro del repositorio en la carpeta. /Doc/Manuales. También se puede acceder a este manual a través de la aplicación dentro del apartado “Ayuda”.

Respecto al manual de uso, está acompañado de imágenes para su facilitar su comprensión, e incluye un apartado de pruebas (extraídas del capítulo 7 que veremos a continuación).



Capítulo 7

Pruebas y validación

Para concluir con el desarrollo del software del proyecto, se han realizado pruebas individuales, haciendo uso de la interfaz que proporciona la herramienta de documentación de APIs Web *Swagger*, sobre cada uno de los recursos involucrados en las peticiones que realiza la aplicación a la API, permitiendo a su vez, comprobar la correcta conexión a la base de datos.

Para detallar el método utilizado, en este caso, se van a mostrar los resultados de la prueba del recurso **Preguntas** haciendo un GET hacia la siguiente URI:

Request URL

```
https://54.194.149.25/API/api/formularios/1
```

Ilustración 12. URI de Preguntas

Nos devuelve las preguntas del formulario Toulouse, para obtenerlo se introduce el tipo del formulario dado que el tipo es el identificador.

Response body

```
[
  {
    "idPregunta": 99,
    "contenido": "Figura1",
    "imagen_url": "https://i.postimg.cc/3RMJvLMZ/Figura1.webp",
    "tipo": "1"
  },
  {
    "idPregunta": 100,
    "contenido": "Figura2",
    "imagen_url": "https://i.postimg.cc/brpy6swD/Figura2.webp",
    "tipo": "1"
  },
  {
    "idPregunta": 101,
    "contenido": "Figura3",
    "imagen_url": "https://i.postimg.cc/d18Qm8wc/Figura3.webp",
    "tipo": "1"
  },
  {
    "idPregunta": 102,
    "contenido": "Figura4",
    "imagen_url": "https://i.postimg.cc/prC23Rm0/Figura4.webp",
    "tipo": "1"
  },
  {
    "idPregunta": 103,
    "contenido": "Figura5",
    "imagen_url": "https://i.postimg.cc/3RMJvLMZ/Figura1.webp",
    "tipo": "1"
  }
]
```

Ilustración 13. Resultados de la llamada a Preguntas

Una de las peticiones principales es el obtener la información necesaria de las recomendaciones, para ello se utiliza el recurso **UsuarioAsignatura** haciendo un GET a la URI:

Request URL

```
https://54.194.149.25/API/api/usuarios/7e17d9a9-9850-4def-b7ac-c78188783e05/asignaturas
```

Ilustración 14. URI de UsuarioAsignatura

Se obtienen los datos de las asignaturas en las que el alumno se ha relacionado, además de proporcionar una URI que mostrará los detalles de la asignatura en cuestión.



Response body

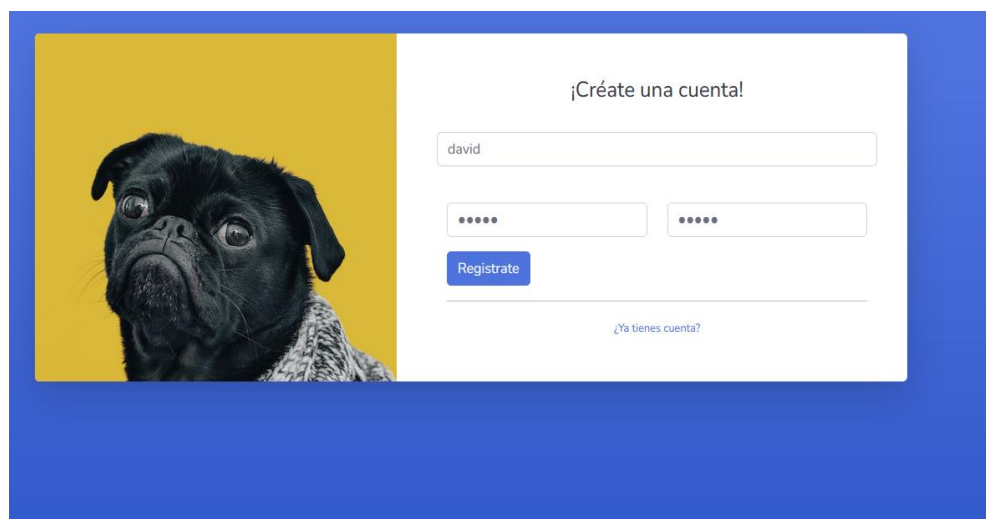
```
[
  {
    "urlAsignatura": "https://54.194.149.25/API/api/usuarios/7e17d9a9-9850-4def-b7a",
    "nombre": "Historia",
    "riesgo": 3,
    "nota": 10
  },
  {
    "urlAsignatura": "https://54.194.149.25/API/api/usuarios/7e17d9a9-9850-4def-b7a",
    "nombre": "Geografia",
    "riesgo": 0,
    "nota": 22
  }
]
```

Ilustración 15. Respuesta de la petición UsuarioAsignatura

Una vez comprobado lo anterior, se ha procedido a realizar la validación de la aplicación realizando todos los pasos relacionados al alumno, desde su registro hasta el cierre de sesión, pudiendo realizar los formularios, asociarle asignaturas y obteniendo recomendaciones de estas durante el camino.

Pasos para realizar la validación del correcto funcionamiento de la aplicación:

1. El alumno realiza el registro en la aplicación haciendo uso de la interfaz insertando su usuario y contraseña.



The image shows a web registration form titled "¡Créate una cuenta!". On the left, there is a yellow square containing a black pug dog. The form itself is white and contains a text input field with the value "david", two password input fields represented by dots, a blue "Register" button, and a link that says "¿Ya tienes cuenta?". The entire form is set against a blue background.

Ilustración 16. Registro de la aplicación

En el momento que el alumno pulsa sobre el botón la aplicación realiza una petición POST al recurso /api/usuarios enviándole el JSON que se muestra a continuación.

```
{
  "nombre": "string",
  "pass": "string"
}
```

Ilustración 17. JSON de inserción de usuario

Desembocando en el guardado del usuario en la base de datos

	IdUsuario	Nombre	Pass	Administrativas_Contables_Int	Humanisticas_Sociales_Int	Artisticas_Int	Medicina_CsSalud_Int	Ir
1	F12133DA-8001-EE11-A63C-54BEF7033D69	admin	admin	4	4	2	4	7
2	F09E72EB-8F7C-4741-B76F-B9FFE10AA211	david	david	NULL	NULL	NULL	NULL	7

Ilustración 18. Inserción en la base de datos

2. Una vez realizado el registro, el usuario podrá acceder a la página de inicio.

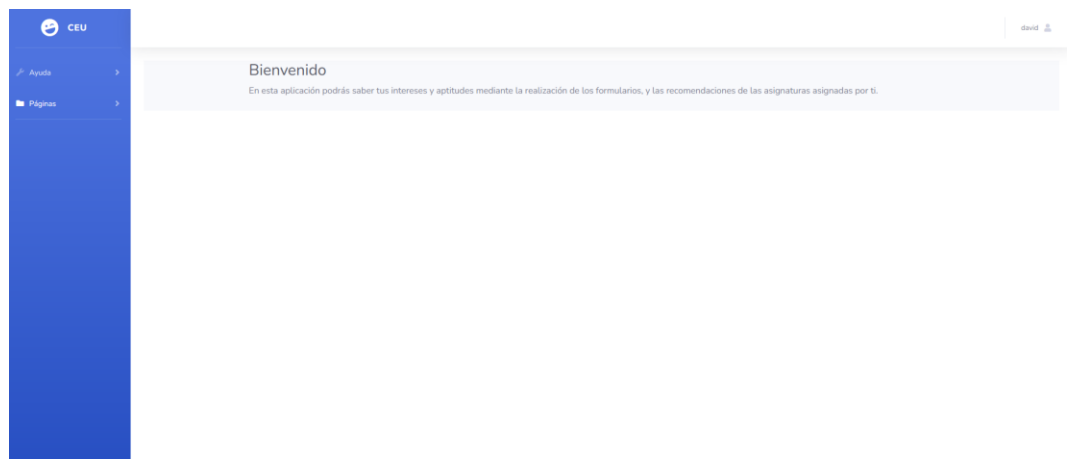


Ilustración 19. Página inicial dentro de la aplicación

En ella lo primero que deberá hacer son los formularios tanto de Toulouse como de CHASIDE, para ello hará uso del navegador izquierdo.

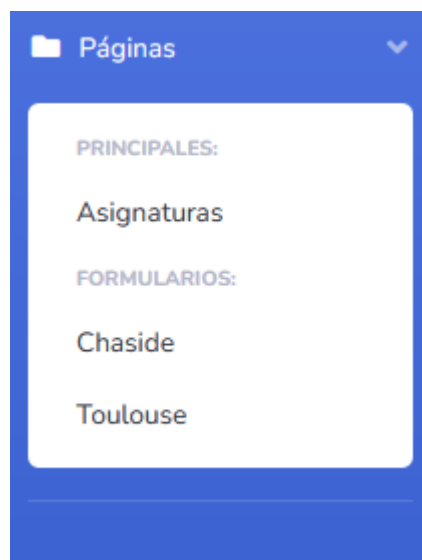


Ilustración 20. Opciones del navegador lateral

3. Una vez seleccionado el formulario de CHASIDE, el usuario leerá la información que se le proporciona sobre el formulario en la misma ventana.

Formulario de Chaside

Este formulario evaluará las aptitudes e intereses del estudiante en función de una serie de preguntas de Si o No. En caso de no contestar a una pregunta se tomará opción escogida el No

1. ¿Aceptarías trabajar escribiendo artículos en la sección económica de un diario?

☒ No
☐ Si

2. ¿Te ofrecerías para organizar la despedida de soltero de uno de tus amigos?

☒ No
☐ Si

Ilustración 21. Formulario CHASIDE en la aplicación

Tras finalizar el formulario el alumno pulsará sobre el botón guardar apareciendo por pantalla que los datos han sido guardados.

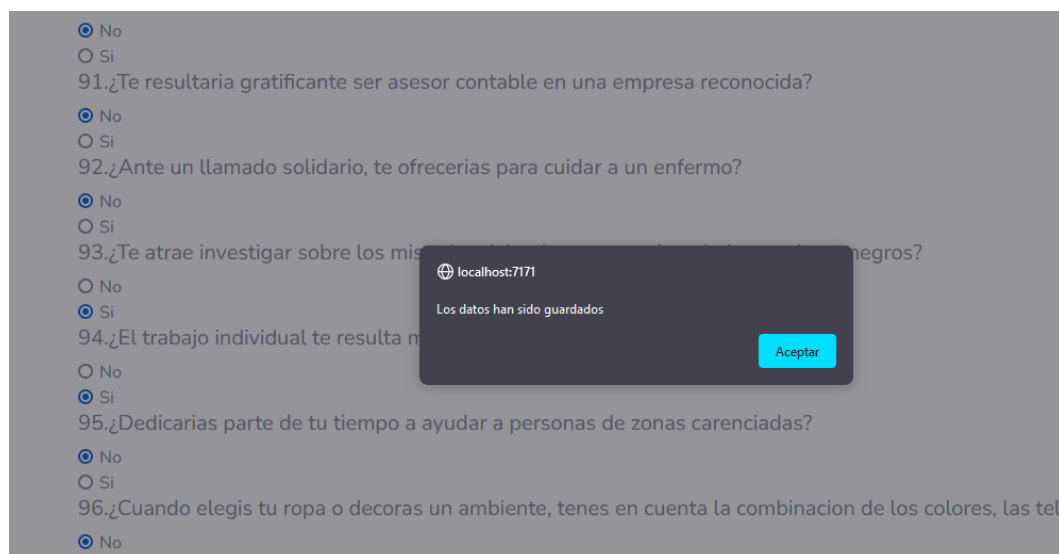


Ilustración 22. Guardado de los datos del formulario CHASIDE en la aplicación

A su vez en el momento que se guardan los datos ya puede acceder a la información de los resultados obtenidos a través de su perfil.

4. El alumno revisará los resultados de su perfil pulsando sobre la imagen de la persona desplegando el panel del Usuario, en el cual escogerá la opción perfil.

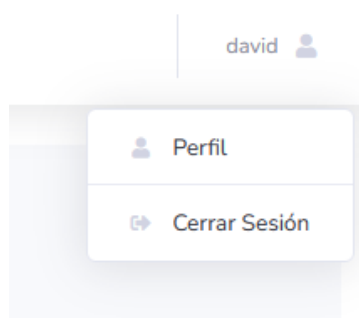


Ilustración 23. Desplegable del Perfil de Usuario

5. Una vez pulsada la opción “Perfil” este podrá ver sus aptitudes e intereses obtenidas tras la realización del formulario CHASIDE.



Intereses

Administrativas y Contables: 3

Humanísticas y Sociales: 5

Artísticas: 4

Medicina y Cs.Salud: 2

Ingeniería de Sistemas: 0

Defensa y Seguridad: 2

Ciencias Exactas y Agrarias: 3

Aptitudes

Administrativas y Contables: 2

Humanísticas y Sociales: 0

Artísticas: 0

Medicina y Cs.Salud: 2

Ingeniería de Sistemas: 0

Defensa y Seguridad: 0

Ciencias Exactas y Agrarias: 3

Ilustración 24. Intereses y aptitudes del alumno

6. Seguidamente el alumno realizara el formulario de Toulouse, siguiendo las indicaciones que aparecen en él.



Ilustración 25. Formulario de Toulouse en la aplicación

7. Una vez completados ambos formularios el alumno ya podrá asociarse las asignaturas disponibles por la herramienta, para obtener recomendaciones. Para poder ir a la ventana deberá pulsar la opción Asignaturas del navegador lateral
8. Una vez dentro de esta pestaña podrá observar una tabla vacía la cual se ira llenando a medida que introduzca asignaturas.

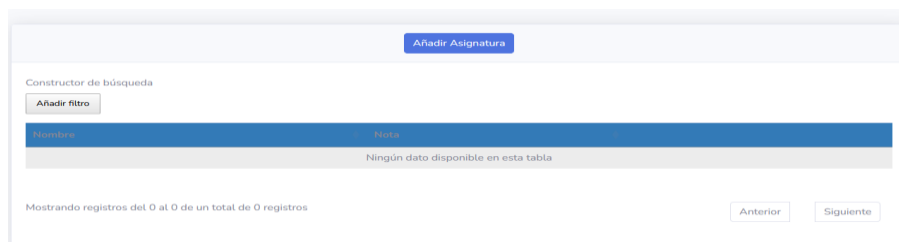


Ilustración 26. Página de asignaturas de la aplicación

9. Cuando el alumno pulsa el botón de añadir asignatura el alumno podrá mediante una ventana emergente, seleccionar una asignatura y rellenar los campos correspondientes y guardándolas al pulsar el botón crear.



Añadir asignatura

Asignatura:

Matematicas

Nota:

3

T.Estudio:

2

Volver

Crear

Ilustración 27. Inserción de la asignatura por parte del Alumno

10. El alumno podrá ver en cada una de las asignaturas el tiempo recomendado pulsando el botón con el icono de lupa, eliminarla con el icono de papelera o editarla con el icono de lápiz.

Nombre	Nota	
Matematicas	3	  

Ilustración 28. tabla asignaturas

Información asignatura

Asignatura:

Matematicas

Nota:

3

T.Estudio:

2

TR.Estudio:

6

Riesgo:

1

Ilustración 29. Información de la asignatura previa a la actualización

Actualizar asignatura

Asignatura:

Matematicas

Nota:

1

T.Estudio:

2

Volver

Actualizar

Ilustración 30. Actualización de la asignatura

Información asignatura

Asignatura:

Matematicas

Nota:

1

T.Estudio:

2

TR.Estudio:

6

Riesgo:

2

Ilustración 31. Información de la asignatura después de la actualización

Borrar asignatura

Estas seguro de borrar la asignatura

Volver

Borrar

Ilustración 32. Borrar la asignatura

11. Por último, el alumno cerrará la sesión mediante el botón de usuario y pulsando cerrar sesión.

Capítulo 8

Conclusiones y líneas futuras

El objetivo principal de este proyecto es la creación de un Servicio Web RESTful que asesore y proporcione un acompañamiento guiado al estudiante a la hora de gestionar y organizar sus tiempos de estudio, se han establecido cuatro objetivos específicos: el primero es la creación de un sistema 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, este objetivo se ha llevado a cabo mediante la implementación del algoritmo que utiliza las fórmulas descritas en el subcapítulo 4.1 junto con los datos recogidos de los formularios, obteniendo como resultado las aptitudes e intereses del alumno, además de la capacidad de concentración. El segundo objetivo se centra en el de la creación de un sistema que sea capaz de realizar una planificación de tiempos de generada mediante los datos obtenidos a través de los datos obtenidos hasta ahora del alumno. Para alcanzar este objetivo se realiza la implementación del algoritmo descrito en el subcapítulo 5.2.3 junto con los resultados obtenidos de los formularios y de los datos de la asignatura asignada al alumno, posteriormente genera una recomendación en los tiempos de estudio, además de un nivel de riesgo utilizado como la facilidad de suspender la asignatura, siendo el 0 casi improbable y el 10 casi seguro. El tercer objetivo es la creación de una aplicación para que el usuario pueda introducir datos y obtener las recomendaciones, este objetivo se ha cubierto con la creación de la aplicación descrita en los capítulos 5 y 6. El cuarto objetivo requiere de la creación de una interfaz con la que se pueda acceder a los recursos ofrecidos por la API, para cubrir este objetivo se ha realizado la

implementación de *Swagger* en la propia API.

Respecto a los requisitos detallados en el subcapítulo 4.2, se han abordado de la siguiente manera: se ha creado una ventana para poder registrarte con un usuario y una contraseña, una vez iniciada la sesión se puede acceder a las ventanas de los formularios, los cuales se nutren de servicios que poseen la lógica de las fórmulas de los formularios CHASIDE y Toulouse. De esta forma se abordan los objetivos de medir la concentración, las aptitudes y los intereses mediante formularios estandarizados. Dentro de la aplicación, existe un apartado de asignaturas donde el alumno puede ver las recomendaciones, además de gestionar las asignaturas disponibles, en este caso se ha cumplido con el requisito de mostrar las recomendaciones, pero no con el de ver las asignaturas cursadas del primer año, dado que al no contar con el rol de administrador, el alumno no puede crear esas asignaturas, ya que él solo puede asignarse a las mismas previamente creadas, una vez realizado los formularios el alumno puede ver los resultados de los formularios accediendo al perfil.

Debido al diseño de la arquitectura toda la funcionalidad de la aplicación se realiza gracias a los Servicios Web alojados en la API RESTful, encargada de exponer los recursos mediante URIs accesibles mediante los métodos HTTP los cuales traen como respuesta objetos de tipo JSON, por lo tanto, al poseer la API separada de la aplicación en otra EC2 (como queda detallado en el subcapítulo 6.3.1). La aplicación solo aporta la interfaz, siendo fácil la portabilidad a otros entornos como el móvil, además de que la aplicación es ajena a cómo están formados dichos recursos.

Por último, respecto a la protección de los datos, al tratarse de SQL Server los identificadores de los usuarios se encuentran en formato GUID, pero al no implementarse correctamente los roles de los usuarios, no se puede dar como completado el requisito de transmisión de la información a las personas autorizadas.



8.1 Líneas futuras

Con la creación del proyecto se han determinado una serie de líneas futuras que se podrían desarrollar para mejorar la calidad del proyecto:

- Añadir una serie de procedimientos que permitan automatizar copias de seguridad de los datos, con ello se aseguraría la continuidad de los datos permitiendo realizar estudios sobre los datos.
- Creación de un balanceador de carga en AWS. Esto permitiría mitigar los efectos de sobrecarga del servidor, garantizando la disponibilidad y rendimiento de las máquinas.
- Actualmente la aplicación no valida si los parámetros de entrada son correctos, ya que en la mayoría de los inputs permite cualquier carácter, pero en la funcionalidad que los consume necesita convertirlos en el tipo adecuado, al limitarlo directamente sobre la interfaz, aliviaría la carga en la funcionalidad mejorando el rendimiento de la aplicación.
- Creación del rol administrador para que se puedan añadir las asignaturas de cada carrera al listado, permitiendo a cada alumno asignarse las asignaturas que curse el primer año.
- Actualmente sí se aporta seguridad en los identificadores de las tablas, pero se guarda las contraseñas de los usuarios en claro, esto es una mala praxis de la programación, ya que, en cualquier momento en caso de conseguir acceder a la base de datos, la información personal de los usuarios estará en riesgo.

- El proyecto se ha desplegado de forma manual y no posee una batería de test automática, con lo que tras desplegarlo se han comprobado manualmente si funcionan los servicios implicados en las recomendaciones, esto provoca una ralentización tanto en el despliegue como en la validación, si se usar un mecanismo como *Azure Devops* y se crearán test unitarios o funcionales, se podría desplegar la aplicación y ejecutar los test de manera automática.

Bibliografía

- [1] Pressman, R. S. (2005). "Software Engineering: A Practitioner's Approach, 736(7) (33-36)
- [2] Pautasso, C., Zimmermann, O., & Leymann, F. (2008). "RESTful Web Services vs. 'Big' Web Services: Making the Right Architectural Decision. (805-812)
- [3] Fowler, M. (2010, March 18). Richardson Maturity Model.
- [4] Morales, M. F., & Gálvez, S. A. H. (2018). Adaptación y validación de la batería Chaside: estudio con estudiantes ecuatorianos. *Revista UNIANDES Episteme*, 5(1), 935-945.
- [5] E. Toulouse y H. Piéron. (2013). TP-R. Toulouse-Piéron-Revisado, prueba perceptiva y de atención (9-37).
- [6] Ainley, M., Hidi, S., & Berndorff, D. (2002). Interest, learning, and the psychological processes that mediate their relationship. *Journal of educational psychology*, 94(3), 545-561.
- [7] Poropat, A. E. (2009). A meta-analysis of the five-factor model of personality and academic performance. *Psychological bulletin*, 135(2), 322-338
- [8] Roediger III, H. L., & Butler, A. C. (2011). The critical role of retrieval practice in long-term retention. *Trends in Cognitive Sciences*, 15(1), 20-27.
- [9] Mrazek, M. D., Franklin, M. S., Phillips, D. T., Baird, B., & Schooler, J. W. (2013). Mindfulness training improves working memory capacity and GRE performance while reducing mind wandering. *Psychological Science*, 24(5), 776-781.
- [10] Eysenck, M. W., Derakshan, N., Santos, R., & Calvo, M. G. (2007). Anxiety and cognitive performance: Attentional control theory. *Emotion*, 7(2), 336-353
- [11] Steel, P., & König, C. J. (2006). Integrating theories of motivation. *Academy of Management Review*, 31(4), 889-913.
- [12] Startbootstrap. SB Admin 2. <https://startbootstrap.com/theme/sb-admin-2>. (Último acceso: mayo 2023).

Anexo

Tablas de los recursos

Recurso: Preguntas				
URI: /formularios/{idFormulario}				
Método	Descripción	Cuerpo de la solicitud	Código de respuestas	Cuerpo de la respuesta
GET	Solicitud de obtener las preguntas del formulario	-	200 OK	Obtiene la respuesta del usuario del formulario
			500 Internal error	-
POST	-	-	405: Method not Allowed	-
PUT	-	-	405: Method not Allowed	
DELETE	-	-	405: Method not Allowed	
PATCH	-	-	405: Method not Allowed	-

Como se puede observar en la tabla anterior solo se puede acceder con el método GET el cual sirve para obtener todas las preguntas del formulario indicado en la URI, en caso de poder realizar correctamente la petición te devolverá el conjunto de preguntas en formato JSON junto con el código 200, sino te devolverá un código de error 500. Para el resto de los métodos devolverá el código 405 ya que no están asociados al servicio.

Recurso: Respuestas				
URI: /usuarios/{idUserio}/formularios/{idFormulario}				
Método	Descripción	Cuerpo de la solicitud	Código de respuestas	Cuerpo de la respuesta
GET	Solicitud de obtener la respuesta del usuario del formulario	-	200 OK	Obtiene la respuesta del usuario del formulario
			500 Internal error	-
POST	-	-	405: Method not Allowed	-
PUT	Solicitud de modificar la respuesta del usuario del formulario	Representación del Respuesta para modificar en formato JSON	200 OK	Modifica la respuesta del usuario del formulario
			-	-
DELETE	Solicitud de borrar la respuesta del usuario del formulario	-	200 OK	Borra la respuesta del usuario del formulario
			-	
PATCH	-	-	405: Method not Allowed	-

Como se puede observar en la tabla anterior los métodos por los cuales puedes realizar la petición al servicio son: GET, PUT y DELETE que en el caso de realizar la petición correctamente y que no se produzca ningún error devolverán un código 200 junto con el resultado de la petición y en el caso de lo contrario, se obtendrá un error 500, para el caso de PATCH y el POST al intentar realizar una petición devolverá que ese método no se encuentra disponible, además el método PUT necesita de un JSON aparte de la URI para realizar la petición correctamente, ya que los datos del JSON son necesarios para llevarla a cabo. Por último, en la columna de la descripción queda detallado lo que realiza el método y en cuerpo de la respuesta lo que devolverá según si se han producido errores o no, además el resultado en caso de ser satisfactorio devolverá la información en formato JSON para el caso del GET mientras que para el PUT y el DELETE solo enviara un mensaje de que se ha completado.

Recurso:Usuario				
URI:/usuarios/{idUserio}				
Método	Descripción	Cuerpo de la solicitud	Código de respuestas	Cuerpo de la respuesta
GET	Solicitud de obtener la información del usuario	-	200 OK	obtiene la información del usuario
			500 Internal error	-
POST	-	-	405: Method not Allowed	-
PUT	Solicitud de actualizar la información del usuario	Representación del Usuario para crear en formato JSON	200 OK	obtiene la información del usuario
			500 Internal error	-
DELETE	Solicitud de borrar la información del usuario	-	200 OK	-
			500 Internal error	-
PATCH	-	-	405: Method not Allowed	-

Como se puede observar en la tabla anterior los métodos por los cuales puedes realizar la petición al servicio son: GET, PUT y DELETE que en el caso de realizar la petición correctamente y que no se produzca ningún error devolverán un código 200 junto con el resultado de la petición y en el caso de lo contrario, se obtendrá un error 500, para el caso de PATCH y el POST al intentar realizar una petición devolverá que ese método no se encuentra disponible, además el método PUT necesita de un JSON aparte de la URI para realizar la petición correctamente, ya que los datos del JSON son necesarios para llevarla a cabo. Por último, en la columna de la descripción queda detallado lo que realiza el método y en cuerpo de la respuesta lo que devolverá según si se han producido errores o no, además el resultado en caso de ser satisfactorio devolverá la información en formato JSON para el caso del GET mientras que para el PUT y el DELETE solo enviara un mensaje de que se ha completado.

Recurso: Usuarios				
URI: /usuarios				
Método	Descripción	Cuerpo de la solicitud	Código de respuestas	Cuerpo de la respuesta
GET	Solicitud de obtener los usuarios existentes	-	200 OK	Obtiene los usuarios existentes
			500 Internal error	-
POST	Solicitud de creación de usuario	Representación del Usuarios para crear en formato JSON	201 Created	Obtiene el usuario creado
			500 Internal error	
PUT	-	-	405: Method not Allowed	-
DELETE	-	-	405: Method not Allowed	-
PATCH	-	-	405: Method not Allowed	-

Como se puede observar en la tabla anterior solo se puede acceder con el método GET el cual sirve para obtener todos los usuarios creados, en caso de poder realizar correctamente la petición te devolverá el conjunto de los usuarios en formato JSON junto con el código 200, sino te devolverá un código de error 500, y el POST que además de la URI, necesita en formato JSON los datos del usuario para crearle, en caso de hacerlo devolverá un 501 y los datos del usuario, sino la consigue crear devolverá un código de error 500 como en el caso anterior. Para el resto de los métodos devolverá el código 405 ya que no están asociados al servicio.

Recurso: UsuarioAsignaturas				
URI: / usuarios/{idUserio}/asignaturas				
Método	Descripción	Cuerpo de la solicitud	Código de respuestas	Cuerpo de la respuesta
GET	Solicitud de obtener las asignaturas asociadas a un usuario	-	200 OK	Obtiene las asignaturas asociadas a un usuario
			500 Internal error	-
POST	-	-	405: Method not Allowed	-
PUT	-	-	405: Method not Allowed	-
DELETE	-	-	405: Method not Allowed	-
PATCH	-	-	405: Method not Allowed	-

Como se puede observar en la tabla anterior solo se puede acceder con el método GET el cual sirve para obtener todas las asignaturas del usuario indicado en la URI, en caso de poder realizar correctamente la petición te devolverá el conjunto de asignaturas (solo el nombre y la URI de la asignatura para obtener mayor detalle) en formato JSON junto con el código 200, sino te devolverá un código de error 500. Para el resto de los métodos devolverá el código 405 ya que no están asociados al servicio.

Recurso: Asignaturas				
URI: /asignaturas				
Método	Descripción	Cuerpo de la solicitud	Código de respuestas	Cuerpo de la respuesta
GET	Solicitud de obtener las asignaturas existentes	-	200 OK	Obtiene las asignaturas existentes
			500 Internal error	-
POST	Solicitud de creación de asignatura	Representación la Asignatura para crear en formato JSON	201 Created	Obtiene la asignatura creada
			500 Internal error	
PUT	-	-	405: Method not Allowed	-
DELETE	-	-	405: Method not Allowed	-
PATCH	-	-	405: Method not Allowed	-

Como se puede observar en la tabla anterior solo se puede acceder con el método GET el cual sirve para obtener todas las asignaturas insertadas, en caso de poder realizar correctamente la petición te devolverá el conjunto de las asignaturas en formato JSON junto con el código 200, sino te devolverá un código de error 500, y el POST que además de la URI, necesita en formato JSON los datos de la asignatura para crearla, en caso de hacerlo devolverá un 501 y los datos de la asignatura, sino la consigue crear devolverá un código de error 500 como en el caso anterior. Para el resto de los métodos devolverá el código 405 ya que no están asociados al servicio.