Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Desarrollo back-end en .NET de una aplicación para la obtención de opiniones mediante gamificación.

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Josep Ferrandis Jorge

Tutor: Antonio Martí Campoy

2017-2018

Resumen

Este proyecto se centra en el diseño y desarrollo de un servidor (*back-end*) que proporcionará soporte mediante un catálogo de servicios a una aplicación llamada Nitpicky, la cual permitirá obtener opiniones mediante gamificación.

La gamificación permite al usuario realizar tareas de una forma lúdica. Mediante esta técnica, se pretende aumentar el interés de los usuarios a la hora de contestar distintas preguntas.

El sistema gestionará un ranking para generar todavía más interés en los usuarios, ya que estarán compitiendo entre ellos. Además, el sistema proporcionará soporte a la gestión de las propias preguntas, pudiendo crearlas, eliminarlas, modificarlas e incluso obtener estadísticas de estas.

Cabe destacar que este proyecto se ha desarrollado en base a una propuesta de la empresa en la cual se han realizado las prácticas del grado.

Para el desarrollo de este proyecto se han utilizado principalmente los lenguajes y tecnologías C#, SQL, .NET. Además, se ha utilizado la metodología de desarrollo TDD (*Test Driven Development*).

Palabras clave: *Test Driven Development*, API, obtención de opiniones, gamificación, C#, SQL, .NET, back-end

Resum

Aquest projecte se centra en el disseny i desenvolupament d'un servidor (back-end) que proporcionarà suport mitjançant un catàleg de serveis a una aplicació anomenada Nitpicky, la qual permetrà obtenir opinions mitjançant gamificació.

La gamificació permet a l'usuari realitzar tasques d'una forma lúdica. Mitjançant aquesta tècnica, es pretén augmentar l'interès dels usuaris a l'hora de contestar diferents preguntes.

El sistema gestionarà un rànquing per a generar encara més interès en els usuaris, ja que estaran competint entre ells. A més, el sistema proporcionarà suport a la gestió de les pròpies preguntes, podent crear-les, eliminar-les, modificar-les i fins i tot obtenir estadístiques d'aquestes.

Cal destacar que aquest projecte s'ha desenvolupat sobre la base d'una proposta de l'empresa en la qual s'han realitzat les practiques del grau.

Per al desenvolupament d'aquest projecte s'han utilitzat principalment els llenguatges i tecnologies C#, SQL, .NET. A més, s'ha utilitzat la metodologia de desenvolupament TDD (*Test Driven Development*).

Paraules clau: *Test Driven Development*, API, obtenció d'opinions, gamificació, C#, SQL, .NET, back-end

Abstract

This project focuses on the design and development of a server (back-end) that will provide support through a catalogue of services to an application called Nitpicky, which will allow opinions to be obtained through gamification.

Gamification allows the user to perform tasks in a playful way and through this technique, it is intended to increase the interest of users when answering different questions.

The system will manage a ranking to generate even more interest from users, as they will be competing. In addition, the system will provide support to the management of the questions themselves, being able to create them, delete them, modify them and even obtain statistics from them.

It should be noted that this project has been developed based on a proposal from the company in which the internships of the degree have been carried out.

For the development of this project, C#, SQL, .NET languages and technologies have been used mainly. In addition, the TDD (Test Driven Development) development methodology has been used.

Key words: Test Driven Development, API, feedback, gamification, C#, SQL, .NET, back-end

Glosario

- **Back-end:** La parte de una aplicación que almacena y manipula datos.
- **Front-end:** Es una interfaz a través de la cual los usuarios comunes pueden acceder a un programa.
- **Test unitario:** es un fragmento de código (normalmente un método) que invoca otro fragmento de código y comprueba posteriormente la exactitud de algunas suposiciones.
- **Http (*Hypertext Transfer Protocol*):** es el conjunto de reglas para la transferencia de archivos (texto, imágenes gráficas, sonido, vídeo y otros archivos multimedia) en internet.
- **Framework:** es una plataforma para el desarrollo de aplicaciones de software. Proporciona una base sobre la cual los desarrolladores de software pueden construir programas para una plataforma específica.
- **Arquitectura software:** es la organización fundamental de un sistema, encarnada en sus componentes, sus relaciones entre sí y los principios que guían su diseño y evolución.
- **Interfaz:** es un espacio compartido a través del cual dos o más componentes separados de un sistema informático intercambian información.
- **Arquitectura de Capas:** es la organización donde los componentes funcionales están separados jerárquicamente por capas. Cada capa solo está conectada con su superior y su inferior mediante interfaces.
- **API:** se refiere a la Interfaz de Programación de Aplicaciones. Es la plataforma que utiliza un programa para acceder a diferentes servicios en el sistema informático.
- **Dto (Data transfer object):** es un objeto que transporta datos entre procesos. Se utilizan cuando se recurre a interfaces remotas (por ejemplo, servicios web), donde cada llamada es una operación costosa.
- **Entidades:** es una unidad de datos que puede clasificarse y que tiene relaciones declaradas con otras entidades**.**
- **Diagrama entidad-relación:** Es un diagrama que representa las entidades y cómo se relacionan entre sí.

Tabla de contenidos

Tabla de figuras

1. Introducción

Este TFG se va a desarrollar basándose en una propuesta de la empresa en la cual se han realizado las practicas del grado. Se va a desarrollar una aplicación llamada Nitpicky la cual se dividirá en dos partes. La parte cliente o front-end, encargada de las interfaces gráficas de usuario y la parte servidor o back-end, encargada de almacenar y manipular los datos de la aplicación. Este TFG se centrará en el diseño y la implementación de esta última parte, la parte servidor. Cabe destacar que para la implementación de la aplicación se va a utilizar la metodología TDD [17] (*Test Driven Development*) explicada posteriormente.

En esta aplicación se distinguirán tres tipos de usuarios. Por una parte, el usuario que contesta a las preguntas, llamado jugador. Por otra parte, el usuario administrador que gestionará las preguntas y obtendrá el resultado de las mismas. El tercer tipo de usuario, llamado super usuario, será el encargado de crear y gestionar a los administradores.

Nitpicky es una aplicación que solicita retroalimentación del jugador sobre diversos temas importantes. Dicha retroalimentación consistirá en unas preguntas que los jugadores deberán elegir entre diversas opciones, como las siguientes: bueno, malo, neutral, etc.

Con el fin de estimular al jugador a la participación de dichas encuestas, la aplicación se diseñará mediante un proceso de gamificación, es decir, contestar a las diversas preguntas utilizando juegos. Se ha decidido utilizar la gamificación puesto que hay diversos estudios que demuestran que, mediante juegos, los niveles de participación aumentan notablemente [1]. Además, se diseñará un sistema de puntos con el fin de poder obtener una puntuación en cada juego y, de esta manera, poder crear un ranking. Con esta idea, se intenta motivar a los jugadores para que tengan como objetivo contestar a las preguntas de una forma rápida y, así, poder conseguir una buena posición en el ranking. Dicho ranking podrá ser utilizado por parte de la empresa u organización que hace uso de esta aplicación para recompensar a las personas con mejor destreza (que a la vez habrán tenido una gran participación en las encuestas), pudiéndose otorgar premios o recompensas.

Nitpicky proporcionará a la organización que utilice esta aplicación la funcionalidad para sondear la opinión de sus usuarios, clientes o empleados sobre temas específicos de forma

continua, es decir, recibiendo *feedback* constantemente. Dicha información será de gran utilidad para la empresa ya que les permitirá mejorar los temas tratados.

Las preguntas que realizará la aplicación serán siempre relacionadas con el jugador, por ejemplo, sobre un producto comprado o un proyecto en el que participó. Como respuesta a las preguntas, en lugar de recibir una respuesta directa, Nitpicky mostrará un juego que permitirá al jugador expresar su opinión.

1.1 Objetivos

El objetivo principal de este proyecto es diseñar e implementar la parte servidor de la aplicación Nitpicky. Esta parte servidor, desacoplada de las interfaces dedicadas a los usuarios, se encargará de almacenar y gestionar las preguntas que se les ofrecerán y de recoger las respuestas. Se generarán informes para los administradores y rankings para los jugadores.

Será necesario definir unos objetivos más específicos con el fin de poder cumplir el principal objetivo del proyecto. Estos objetivos son los siguientes:

- Definir el propósito exacto de la aplicación.
- Analizar y estudiar los requisitos necesarios.
- Estudiar los distintos casos de uso que presenta la aplicación.
- Utilizar y dominar la metodología TDD en el desarrollo de la aplicación.
- Diseñar tanto la arquitectura como el modelo de datos que se utilizará a la hora de la implementación.

No es un objetivo de este proyecto el diseño y el desarrollo de la parte cliente (*front-end*) donde se encontrarán las interfaces gráficas de usuario con su correspondiente funcionalidad. No obstante, el servidor estará diseñado para dar soporte a cualquier tipo de aplicación que pueda realizar peticiones HTTP con el fin de comunicarse con el servidor.

1.2 Estructura

Esta memoria está estructurada en nueve puntos. El primer punto es la introducción, donde se definirán los objetivos generales y la estructura de la memoria.

En el siguiente punto se establece una estimación de tiempo necesario para desarrollar las diferentes tareas en que se divide el proyecto.

En el tercer punto se definen los requisitos software de la aplicación siguiendo el estándar IEEE 830/1998.

En el cuarto punto se detallan los diferentes casos de uso mediante diagramas de caso de uso y diagramas de secuencia. También se define la información relativa a cada caso de uso mediante una tabla.

A continuación, en la siguiente fase del proyecto se desarrolla la explicación referente al diseño de la propia aplicación, ya sea de la parte de la arquitectura del servidor como el diseño de la base de datos con su diagrama de entidad-relación correspondiente.

Una vez terminada esta etapa de diseño, pasamos a la fase de implementación. En este apartado se detallarán tanto las tecnologías como las herramientas que se han realizado para poder desarrollar la aplicación.

Tras la etapa de implementación de la aplicación, se describen todas las pruebas realizadas que garantizan el correcto funcionamiento de todos los requisitos definidos en los puntos anteriores.

Para finalizar la memoria, se incluye un apartado de conclusiones donde se reflexiona sobre los problemas surgidos y diversos aspectos a tener en cuenta. Junto a este apartado encontraremos los apéndices y la bibliografía que se ha consultado para la obtención de la información necesaria que ha hecho posible la realización de este proyecto.

2. Estimación de tiempos

En este punto se va a realizar una estimación de los tiempos necesarios para la realización de las diferentes etapas del proyecto. La planificación para identificar los requisitos y funcionalidades es de dos semanas. En este periodo se pensará detenidamente cada uno de los requisitos y las funcionalidades del proyecto.

A continuación, se pasará a la fase de casos de uso, en la cual se planea utilizar un total de una semana para estudiar los posibles casos de uso y realizar diferentes diagramas de secuencia.

La siguiente parte del proyecto es el diseño, donde se estudiarán todas las tecnologías necesarias, que tipo de arquitectura es la adecuada, qué tipo de base de datos necesitaremos, etc. Dado que esta parte es muy importante en el desarrollo de la aplicación debido a que es la base del proyecto, se dedicarán dos semanas.

A continuación, se pasará a la implementación, la parte donde se desarrollará la aplicación en sí. Esta es la etapa más costosa, así que se destinará un total de seis semanas. Además, se empezará al mismo tiempo que la implementación de las pruebas de la aplicación, debido a que se va a utilizar la metodología TDD y va a ser necesario realizar las pruebas conforme se va desarrollando la aplicación. Cabe destacar que estas pruebas tendrán una semana más de duración que la propia implementación, ya que será necesario probar el correcto funcionamiento una vez la aplicación esté terminada.

Como se muestra en la tabla inferior, el proyecto tiene una duración de doce semanas, que serían unos tres meses de trabajo.

				Semanas											
Tareas	Inicio	Fin	Duración	1	2	3	4	5	6	7	8	9	10	11	12
Requisitos y funcionalidades	03/06/2018	03/20/2018	14d												
Casos de uso	03/21/2018	03/28/2018	7d												
Diseño	03/29/2018	04/12/2018	14d												
Implementación	04/13/2018	05/25/2018	42d												
Pruebas	04/13/2018	06/01/2018	49d												

3. Requisitos

En este apartado se van a definir todos los requisitos que deberá cumplir la aplicación. Se va a seguir el estándar IEEE 830/1998 [2] para la especificación de requisitos.

Mediante este estándar se podrá confeccionar un apartado donde se definan los requisitos de una manera muy completa.

3.1 Introducción

Como se ha comentado anteriormente se va a hacer uso del estándar IEE 830/1998 para la definición de los requisitos de la aplicación. Mediante este estándar se sabrán los pasos a seguir y qué estructura debe tener el análisis de requisitos. A continuación, se seguirán las directrices impuestas por este estándar.

3.1.1 Propósito

El objetivo de un buen análisis de especificación de requisitos es definir de una manera concreta y correcta todas y cada una de las funcionalidades que deberá realizar la aplicación. Además, también se especificarán ciertas funcionalidades que la aplicación no hará, dado que quedan fuera del alcance del proyecto.

3.1.2 Ámbito del sistema

El objetivo de este proyecto es desarrollar una API que proporcionará soporte a una aplicación que permitirá la obtención de opiniones de los jugadores mediante la gamificación. La aplicación que se desarrollará recibirá el nombre de Nitpicky.

Un jugador podrá acceder a un listado de todos los temas disponibles de los cuales tiene preguntas para responder, así pues, pulsando sobre el tema le aparecerán todas las preguntas referentes a ese tema que tenga pendientes de contestar.

El sistema permitirá al jugador descartar una pregunta, siendo está descartada del recuento final de resultados de cada pregunta.

El jugador podrá responder a la pregunta mediante un juego y una vez terminado el juego se guardarán tanto el resultado de la opinión del jugador como la eficiencia con la que el jugador ha respondido a dicha pregunta. Esta eficiencia será la puntuación del jugador.

Además, la aplicación permitirá a un jugador observar su puntuación, su posición en el ranking y el ranking de los tres mejores jugadores.

Para la parte del administrador, el sistema proporcionará soporte para generar y gestionar los temas, las preguntas y a que jugadores les corresponden contestarlas. Así que este administrador será capaz de generar estadísticas y llevar a cabo un seguimiento de las respuestas a las preguntas.

Por último, el super usuario, podrá transformar usuarios jugadores en usuarios administradores y viceversa. Los administradores como se ha comentado, serán los encargados de gestionar tanto la creación de las preguntas como las estadísticas de éstas.

El principal beneficio de esta aplicación es que mediante la gamificación y la competitividad aumentará notablemente la participación de los jugadores a la hora de contestar las preguntas propuestas por el sistema.

3.1.3 Referencias

IEEE Recommended Practice for Software Requirements Specification. ANSI/IEEE 830/1998

3.1.4 Visión general del documento

Este capítulo consta de la introducción, una descripción general y por último presenta los requisitos específicos. En la introducción, se ha proporcionado una visión general del capítulo. En el apartado de descripción general se encontrarán todos los aspectos que tienen relevancia en la aplicación y en los requisitos. Para finalizar, en la sección de requisitos específicos, se definirá cada uno de los requisitos con el suficiente detalle para que se pueda llevar a cabo su implementación. Además, mediante estos requisitos también se podrán planificar todas las pruebas que indicarán si se cumplen todos los requisitos expuestos.

3.2 Descripción general

En el siguiente apartado se describirán todos los aspectos que puedan afectar a los requisitos de la aplicación. Primero se describirá el marco en el que se va a desarrollar, para posteriormente establecer todos los requisitos de una manera más detallada.

3.2.1 Perspectiva del producto

El API que se va a desarrollar permitirá a un *front-end* que se suscriba a los servicios ofertados por éste mediante las llamadas pertinentes. Este *front-end* podrá ser tanto una aplicación móvil como una web.

3.2.2 Funciones del producto

En este apartado se definirán las funciones principales que el servidor deberá ser capaz de realizar.

Antes de empezar con la definición de funcionalidades, es necesario definir la distinción de los tres roles de usuarios, Ilustración1, que se podrán encontrar en la aplicación. Por una parte, está el usuario jugador. Este tipo de usuario será el encargado de contestar a las preguntas mediante juegos, dando así su opinión respecto a un producto, evento, servicio, etc. Por otra parte, se encuentra al usuario administrador, quien tendrá acceso a todas las tareas de gestión de las preguntas. Además, este tipo de usuario podrá obtener todos los resultados de las preguntas. El rol super usuario, será el encargado de gestionar a los administradores en el sistema. Por último el rol usuario no registrado será capaz de registrarse en el sistema.

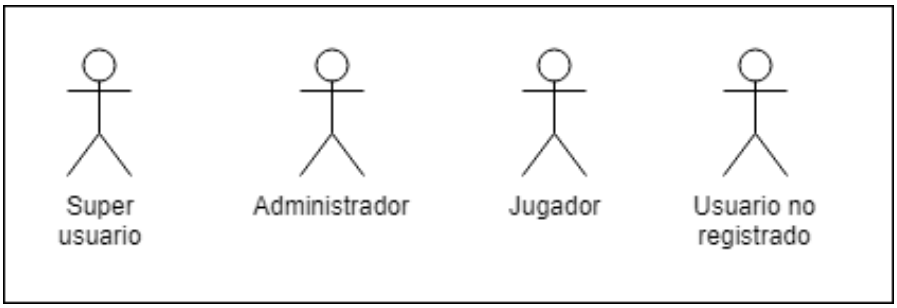


Ilustración 1 Roles de la aplicación

A continuación, se van a definir a grandes rasgos las funciones y servicios que va a proporcionar la API:

- Registro de usuario: Cualquier usuario nuevo en el sistema podrá registrarse. El sistema le asignará automáticamente el rol de jugador.

- Inicio de sesión: Cualquier usuario podrá iniciar sesión en el sistema.

- Crear, modificar y eliminar temas: El administrador podrá crear un tema, que más tarde estará compuesto por preguntas. Dicho administrador podrá modificar el enunciado del tema y también borrar lo, eliminando a su vez todas las preguntas dependientes del tema asociado.

- Crear, modificar y eliminar preguntas: El administrador podrá crear preguntas, debiendo especificar a qué tema pertenecerá cada una. El tema debe de estar creado anteriormente. Además, el administrador, podrá modificar el enunciado y también eliminar la propia pregunta.

- Crear y eliminar juegos: El administrador podrá crear un juego en el sistema, debiendo especificar el título del mismo. Además, también podrá eliminar un juego del sistema.

- Asignar pregunta a jugador: El administrador podrá asignar una pregunta a uno o más jugadores. Es necesario que existan tanto la pregunta, como el jugador, como el juego para responderla. Dicho juegos será aleatorio entre todos los juegos existentes en el sistema.

- Obtener resultados de las preguntas: El administrador podrá obtener distintos resultados como, por ejemplo, la respuesta media de cada pregunta, o de cada tema, la puntuación media de cada jugador, etc. Mediante todos estos resultados podrá crear estadísticas.

- Ver temas a contestar: Tanto el administrador como el jugador podrán listar los temas, con la diferencia de que el administrador podrá ver todos los temas y el jugador solo los temas en los que tenga preguntas pendientes.

- Ver preguntas a contestar: El administrador podrá visualizar todas las preguntas y el jugador solo las que tiene pendientes de contestar.
- Ver ranking y puntuación: El jugador y el administrador podrán ver el ranking de los mejores jugadores, pero solo el usuario jugador puede ver su propia puntuación y posición en la clasificación.
- Gestión de administradores: El super usuario gestionará a los administradores, pudiendo transformar a un usuario jugador en administrador y viceversa.

3.2.3 Características de los usuarios

Esta aplicación está pensada para un uso empresarial, en el que el administrador gestiona la información sobre las preguntas y los temas que deben contestar los jugadores. La gestión de los usuarios estará controlada por la propia aplicación, siendo esta la encargada de crear a los nuevos usuarios y almacenar su información en la base de datos. A la hora de iniciar sesión, se comprobará si el usuario existe o no y si sus credenciales son correctas.

En el sistema puede existir más de un administrador, siendo los encargados de gestionar las preguntas y resultados de la aplicación.

3.2.4 Restricciones

No existen restricciones más allá de la propia funcionalidad de la aplicación.

3.2.5 Suposiciones y Dependencias

La API está diseñada para dar soporte a cualquier tipo de cliente que pueda comunicarse mediante peticiones HTTP.

El rendimiento de respuesta de la API puede variar dependiendo del servidor web que la aloje.

3.2.6 Requisitos Futuros

Sería interesante implementar los siguientes requisitos en un futuro:

- Notificar al jugador cada cierto tiempo que tiene preguntas pendientes por contestar.

- Añadir respuestas abiertas, ya que hasta ahora solo se implementará el tipo de respuesta cerrada.

- Poder generar grupos de jugadores para facilitar al administrador el hecho de asignar preguntas a jugadores.

3.3 Requisitos específicos

En esta parte de la especificación de requisitos siguiendo el estándar IEEE 830/1998 se explicará de una manera mucho más detallada y completa cada uno de los requisitos.

3.3.1 Interfaces Externas

Para el desarrollo de este proyecto no serán necesarias interfaces gráficas puesto que se trata de una API donde ofreceremos servicios. No obstante, dichos servicios están pensados para la implementación de dos interfaces gráficas de usuarios diferentes. Por una parte, la interfaz para el rol jugador y por otra parte la interfaz para el rol administrador. Cada interfaz permitirá el acceso a las distintas funcionalidades, dependiendo del tipo de rol.

3.3.2 Funciones

En este apartado se van a explicar todas las tareas que el sistema va a llevar a cabo. Para la explicación se enumerará una lista con cada funcionalidad indicando qué rol debe desempeñar dicha función. Serán explicadas con el suficiente detalle para hacer posible su posterior implementación.

Primero se empezará con las funcionalidades del rol administrador:

Número	1
Requisito	Crear tema
Prioridad	Alta
Descripción	El administrador crea un nuevo tema al que posteriormente se asignarán las preguntas correspondientes.

Número	2
Requisito	Crear pregunta
Prioridad	Alta
Descripción	El administrador crea una pregunta nueva indicando a que jugadores y tema va dirigida la pregunta.

Número	3
Requisito	Crear juego
Prioridad	Alta
Descripción	El administrador crea un juego nuevo en el sistema

Número	4
Requisito	Modificar tema
Prioridad	Alta
Descripción	El administrador modifica el título del tema.

Número	5
Requisito	Modificar pregunta
Prioridad	Alta
Descripción	El administrador modifica la formulación de la pregunta.

Número	6
Requisito	Eliminar tema
Prioridad	Alta
Descripción	El administrador elimina un tema del sistema, eliminando todas las preguntas relacionadas con ese tema.

Número	7
Requisito	Eliminar pregunta
Prioridad	Alta
Descripción	El administrador elimina una pregunta del sistema, eliminando así la puntuación de dicha pregunta para todos los jugadores afectados.

Número	8
Requisito	Eliminar juego
Prioridad	Alta
Descripción	El administrador elimina un juego del sistema.

Número	9
Requisito	Ver lista de temas
Prioridad	Alta
Descripción	El administrador lista todos los temas que el sistema tiene almacenados.

Número	10
Requisito	Ver lista de preguntas
Prioridad	Alta
Descripción	El administrador lista todas las preguntas que pertenecen a un tema concreto.

Número	11
Requisito	Resultado pregunta concreta
Prioridad	Alta
Descripción	El administrador puede observar el resultado de una pregunta y la media de puntuación de los jugadores.

Número	12
Requisito	Resultado tema concreto
Prioridad	Alta
Descripción	El administrador puede observar el resultado de un tema concreto, que es la suma de las medias de cada pregunta. También se obtendrá la media total de puntuación en todas las preguntas de dicho tema.

Número	13
Requisito	Ver pregunta mejor valorada
Prioridad	Alta
Descripción	El administrador puede saber que pregunta ha sido la mejor valorada en un tema concreto.

Número	14
Requisito	Ver pregunta peor valorada
Prioridad	Alta
Descripción	El administrador puede saber que pregunta ha sido la peor valorada en un tema concreto.

Número	15
Requisito	Ver pregunta con mejor puntuación media
Prioridad	Alta
Descripción	El administrador puede saber que pregunta ha obtenido mayor puntuación media por los jugadores en un tema concreto.

Número	16
Requisito	Ver pregunta con peor puntuación media
Prioridad	Alta
Descripción	El administrador puede saber que pregunta ha obtenido menor puntuación media por los jugadores en un tema concreto.

Número	17
Requisito	Ver pregunta con mayor participación
Prioridad	Alta
Descripción	El administrador puede saber que pregunta ha sido la más contestada en un tema concreto.

Número	18
Requisito	Ver pregunta con menor participación
Prioridad	Alta
Descripción	El administrador puede saber que pregunta ha sido la menos contestada en un tema concreto.

Número	19
Requisito	Ver pregunta más descartada
Prioridad	Alta
Descripción	El administrador puede saber que pregunta ha sido la más descartada en un tema concreto.

Número	20
Requisito	Ver tema mejor valorado
Prioridad	Alta
Descripción	El administrador puede saber qué tema ha sido el mejor valorado.

Número	21
Requisito	Ver tema peor valorado
Prioridad	Alta
Descripción	El administrador puede saber qué tema ha sido el peor valorado.

Número	22
Requisito	Ver tema con mejor puntuación media
Prioridad	Alta
Descripción	El administrador puede saber el tema con mejor puntuación media.

Número	23
Requisito	Ver tema con peor puntuación media
Prioridad	Alta
Descripción	El administrador puede saber el tema con peor puntuación media.

Número	24
Requisito	Ver tema con más participación
Prioridad	Alta
Descripción	El administrador puede saber qué tema ha sido el más contestado.

Número	25
Requisito	Ver tema con menos participación
Prioridad	Alta
Descripción	El administrador puede saber qué tema ha sido el menos contestado.

Número	26
Requisito	Ver tema con más preguntas descartadas
Prioridad	Alta
Descripción	El administrador puede saber qué tema ha sido el más descartado.

Número	27
Requisito	Ver mejores tres jugadores
Prioridad	Alta
Descripción	El administrador puede listar a los tres jugadores con la puntuación más alta.

Ahora se van a definir las funcionalidades del rol Jugador:

Número	28
Requisito	Ver ranking
Prioridad	Alta
Descripción	El sistema proporciona al jugador la lista de los cinco mejores jugadores con sus correspondientes puntuaciones.

Número	29
Requisito	Ver puntuación
Prioridad	Alta
Descripción	El sistema proporciona al jugador su propia puntuación con su nombre.

Número	30
Requisito	Ver lista de temas disponibles
Prioridad	Alta
Descripción	El sistema proporciona al jugador un listado de temas de los cuales el jugador tiene preguntas pendientes de contestar.

Número	31
Requisito	Ver lista de preguntas disponibles
Prioridad	Alta
Descripción	El sistema proporciona al jugador una lista de preguntas disponibles correspondientes al tema seleccionado.

Número	32
Requisito	Descartar pregunta
Prioridad	Alta
Descripción	El jugador descarta una pregunta, dicha pregunta no tiene validez a la hora de calcular las estadísticas de cada pregunta y tema.

Número	33
Requisito	Responder pregunta
Prioridad	Alta
Descripción	El jugador responde a la pregunta seleccionada. Automáticamente se guarda toda la información necesaria. (puntuación de la pregunta y la respuesta a la pregunta)

Ahora, se van a definir las funcionalidades del rol Super Usuario:

Número	34
Requisito	Cambiar el rol de un jugador a administrador
Prioridad	Alta
Descripción	El super usuario cambia el rol de un usuario jugador al rol administrador.

Número	35
Requisito	Cambiar el rol de un administrador a jugador.
Prioridad	Alta
Descripción	El super usuario cambia el rol de un usuario administrador al rol jugador.

Por último, se van a definir las funcionalidades que van a poder ser usadas por todos los usuarios de la aplicación.

Número	36
Requisito	Iniciar sesión de usuario
Prioridad	Alta
Descripción	Cualquier usuario puede iniciar sesión con el fin de acceder a las funcionalidades de la aplicación.

Número	37
Requisito	Registrar nuevo usuario
Prioridad	Alta
Descripción	Cualquier usuario puede registrarse como jugador con el fin de poder crear una nueva cuenta que le de acceso a las funcionalidades de la aplicación.

3.3.3 Requisitos de rendimiento

En cuanto a los requisitos de rendimiento es necesario decir que estarán sujetos al rendimiento que pueda ofrecer el servidor donde se aloje la API.

La carga que tenga que soportar el servidor no será demasiado grande, aunque deberá tener una buena escalabilidad para poder soportar cargas mayores en el futuro.

3.3.4 Atributos del sistema

En este apartado se van a detallar que atributos de calidad del sistema serán necesarios. Estos atributos son fiabilidad, anonimidad, escalabilidad, seguridad y portabilidad

El sistema tiene que ser fiable, tiene que ofrecer todas las funcionalidades descritas en el apartado anterior. Todas estas funcionalidades tienen que funcionar correctamente.

El sistema tiene que ser escalable, tiene que ser capaz de adaptarse a la demanda de usuarios, pudiendo dar soporte al número de usuarios que estén registrados simultáneamente.

La aplicación tiene que garantizar la anonimidad de los jugadores, de esta manera el administrador no podrá saber la opinión de un jugador concreto sobre una pregunta.

El sistema debe de ser seguro, se debe proteger de accesos no autorizados y garantizar la persistencia de los datos.

El sistema tiene que ser portable, es decir, que será relativamente sencillo de instalar en otros servidores.

4. Casos de uso

Las técnicas de casos de uso capturan los requisitos funcionales de más alto nivel del sistema a desarrollar. Mediante la aplicación de esta técnica se consigue especificar el comportamiento del sistema. [3]

Los casos de uso pretenden definir qué hará el sistema, con un enfoque desde los diferentes roles existentes en la aplicación. Su propósito es realizar un seguimiento del proyecto y poder ofrecer una estructura a la aplicación.

Estas técnicas utilizan los llamados diagramas de caso de uso. En estos diagramas se encuentran los diferentes actores y las diferentes funciones que pueden realizar en el sistema.

Un actor especifica el rol del usuario que será una entidad que intercambiará información con el sistema. A cada actor se le asignarán actividades que son relevantes en la especificación de sus casos de uso.

A continuación, se mostrará el diagrama de casos de uso del sistema que posteriormente se desarrollará y después se hará una descripción detallada de cada uno.

4.1 Diagrama de casos de uso

A continuación, se muestra en la ilustración 2, el diagrama de casos de uso de la aplicación.

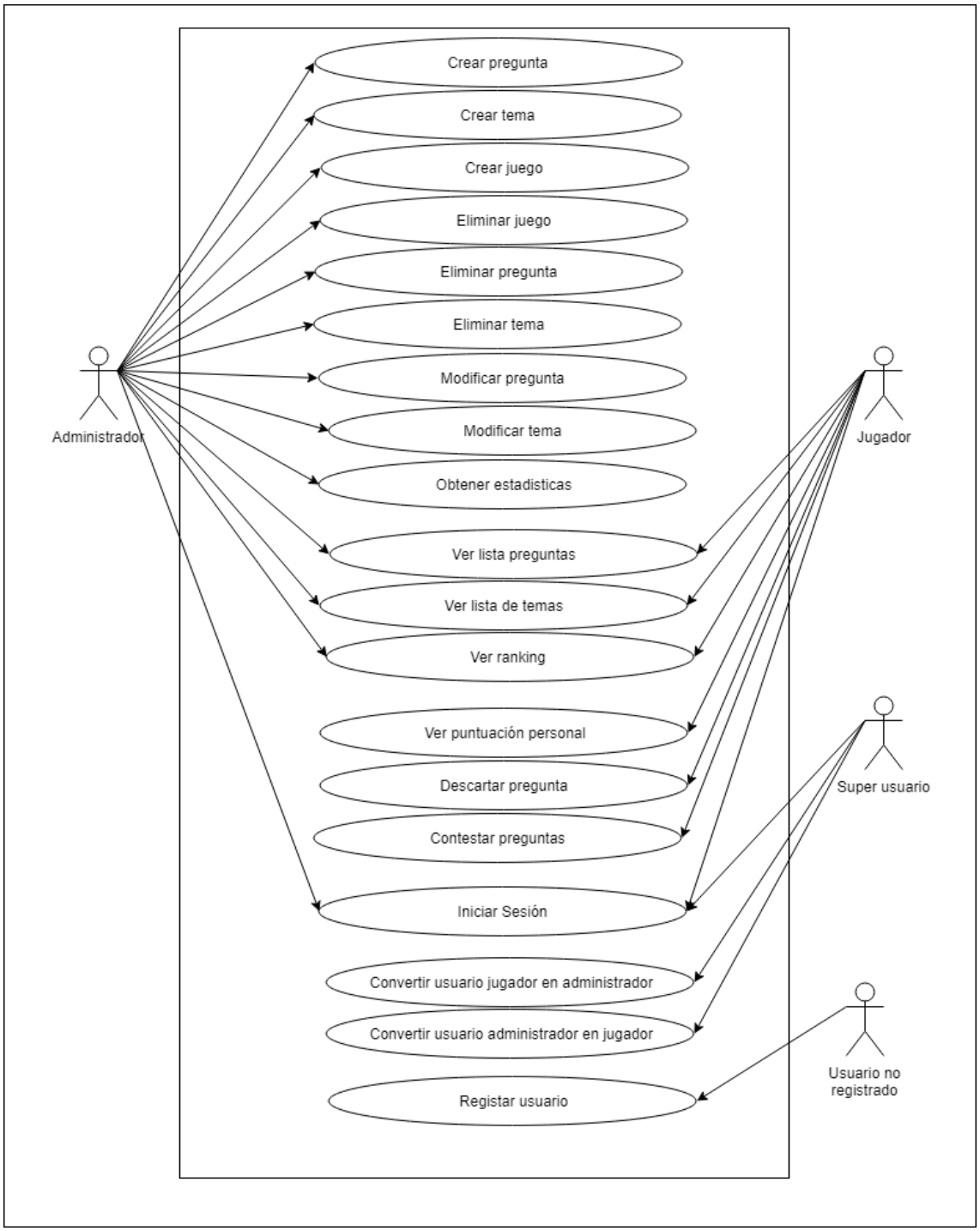


Ilustración 2 Casos de uso

4.2 Descripción casos de uso

A continuación, se describirán los casos de uso para posteriormente realizar la implementación de cada caso de uso.

Caso de uso	Crear tema
Actores	Administrador
Propósito	Añadir un nuevo tema al sistema
Resumen	El administrador añade un tema en el que posteriormente creará preguntas.
Precondiciones	El administrador debe de estar autentificado en el sistema

Caso de uso	Crear pregunta
Actores	Administrador
Propósito	Añadir una pregunta al sistema para poder obtener la opinión de los jugadores
Resumen	El administrador añade una nueva pregunta al sistema, debe introducir qué jugadores podrán contestar a dicha pregunta y a que tema pertenece. El sistema asignará dicha pregunta a un juego, eligiendo aleatoriamente entre los distintos juegos existentes en la base de datos.
Precondiciones	El administrador debe de estar autentificado en el sistema. Deben de estar creados los jugadores y el tema.

Caso de uso	Crear juego
Actores	Administrador
Propósito	Añadir un juego nuevo al sistema
Resumen	El administrador añade un nuevo juego al sistema. Debe introducir el título del juego.
Precondiciones	El administrador debe de estar autentificado en el sistema.

Caso de uso	Modificar pregunta
Actores	Administrador
Propósito	Modificar una pregunta del sistema
Resumen	El administrador puede modificar una pregunta cambiando la formulación de esta.
Precondiciones	El administrador debe de estar autentificado en el sistema y la pregunta debe de haber sido creada.

Caso de uso	Modificar tema
Actores	Administrador
Propósito	Modificar un tema del sistema
Resumen	El administrador puede modificar el título que recibe el tema.
Precondiciones	El administrador debe de estar autentificado en el sistema y el tema debe de haber sido creado.

Caso de uso	Eliminar tema
Actores	Administrador
Propósito	Eliminar un tema del sistema
Resumen	El administrador puede eliminar un tema en concreto, eliminando a su vez todas las preguntas dependientes de este tema.
Precondiciones	El administrador debe de estar autentificado en el sistema y el tema debe de haber sido creado.

Caso de uso	Eliminar pregunta
Actores	Administrador
Propósito	Eliminar una pregunta del sistema
Resumen	El administrador puede eliminar una pregunta en concreto del sistema.
Precondiciones	El administrador debe de estar autentificado en el sistema y la pregunta debe de haber sido creada.

Caso de uso	Eliminar juego
Actores	Administrador
Propósito	Eliminar un juego del sistema
Resumen	El administrador puede eliminar un juego en concreto.
Precondiciones	El administrador debe de estar autentificado en el sistema y el juego debe de haber sido creado.

Caso de uso	Obtener estadísticas
Actores	Administrador
Propósito	Obtener estadísticas del sistema
Resumen	El administrador puede obtener diferente información referente a los temas, preguntas y jugadores con la finalidad de generar unas estadísticas.
Precondiciones	El administrador debe de estar autentificado en el sistema.

Caso de uso	Ver lista preguntas
Actores	Administrador y Jugador
Propósito	Listar las preguntas disponibles sobre un tema
Resumen	El administrador lista todas las preguntas existentes referentes a un tema. El jugador lista todas las preguntas que tiene pendientes de contestar de un tema concreto.
Precondiciones	Tanto el administrador como el jugador han de estar autentificados y el tema debe de haber sido creado.

Caso de uso	Ver lista temas
Actores	Administrador y Jugador
Propósito	Listar los temas disponibles del sistema
Resumen	El administrador lista todos los temas que están registrados en el sistema. El jugador lista todos los temas en los que tiene preguntas pendientes de contestar.
Precondiciones	Tanto el administrador como el jugador han de estar autentificados.

Caso de uso	Ver ranking
Actores	Administrador y Jugador
Propósito	Ver los mejores jugadores
Resumen	Tanto el administrador como el jugador ven el listado de los mejores jugadores con su nombre y su puntuación.
Precondiciones	Tanto el administrador como el jugador han de estar autentificados.

Caso de uso	Ver puntuación personal
Actores	Jugador
Propósito	Ver la puntuación actual de un jugador
Resumen	El jugador puede ver su puntuación actual y su posición en el ranking.
Precondiciones	El jugador debe de estar autentificado en el sistema.

Caso de uso	Descartar pregunta
Actores	Jugador
Propósito	Se descartará una pregunta concreta para un jugador concreto
Resumen	El jugador descarta una pregunta para que no le aparezca más en la lista de preguntas
Precondiciones	El jugador debe de estar autentificado en el sistema y la pregunta debe de haber sido creada.

Caso de uso	Contestar pregunta
Actores	Jugador
Propósito	Contestar la pregunta sobre un tema del sistema
Resumen	El jugador responde a la pregunta, de esta forma se obtiene simultáneamente la opinión y la puntuación de dicha pregunta. Estos datos serán almacenados con la finalidad de poder generar estadísticas.
Precondiciones	El jugador debe de estar autentificado en el sistema y la pregunta debe de haber sido creada.

Caso de uso	Convertir usuario jugador en administrador
Actores	Super usuario
Propósito	Cambiar jugador a rol administrador
Resumen	El super usuario convierte a un usuario de rol administrador en un usuario de rol jugador
Precondiciones	El super usuario debe de estar autentificado en el sistema y el jugador al que se le desea cambiar el rol debe existir.

Caso de uso	Convertir usuario administrador en jugador
Actores	Super usuario
Propósito	Cambiar administrador a rol jugador
Resumen	El super usuario convierte a un usuario de rol administrador en un usuario de rol jugador.
Precondiciones	El super usuario debe de estar autentificado en el sistema y el administrador al que se le desea cambiar el rol debe existir.

Caso de uso	Iniciar sesión
Actores	Administrador, Jugador, Super usuario
Propósito	Cambiar administrador a rol jugador
Resumen	El usuario inicia sesión en el sistema mediante su nombre de usuario y su contraseña.
Precondiciones	El usuario debe existir en el sistema

Caso de uso	Registrarse
Actores	Todos los usuarios
Propósito	Registrarse en el sistema
Resumen	Un usuario puede registrarse en el sistema introduciendo los datos requeridos por el sistema.
Precondiciones	El nombre de usuario no debe de existir en el sistema

4.3 Diagramas de secuencia

En este apartado se van a mostrar dos diagramas de secuencia con la finalidad de mostrar el flujo de la aplicación en general, tanto en el lado cliente como en el lado servidor.

En primer lugar, se mostrará en la ilustración 3 un diagrama de secuencia que ilustra la interacción entre el usuario jugador, cliente y servidor para realizar un caso de uso concreto.

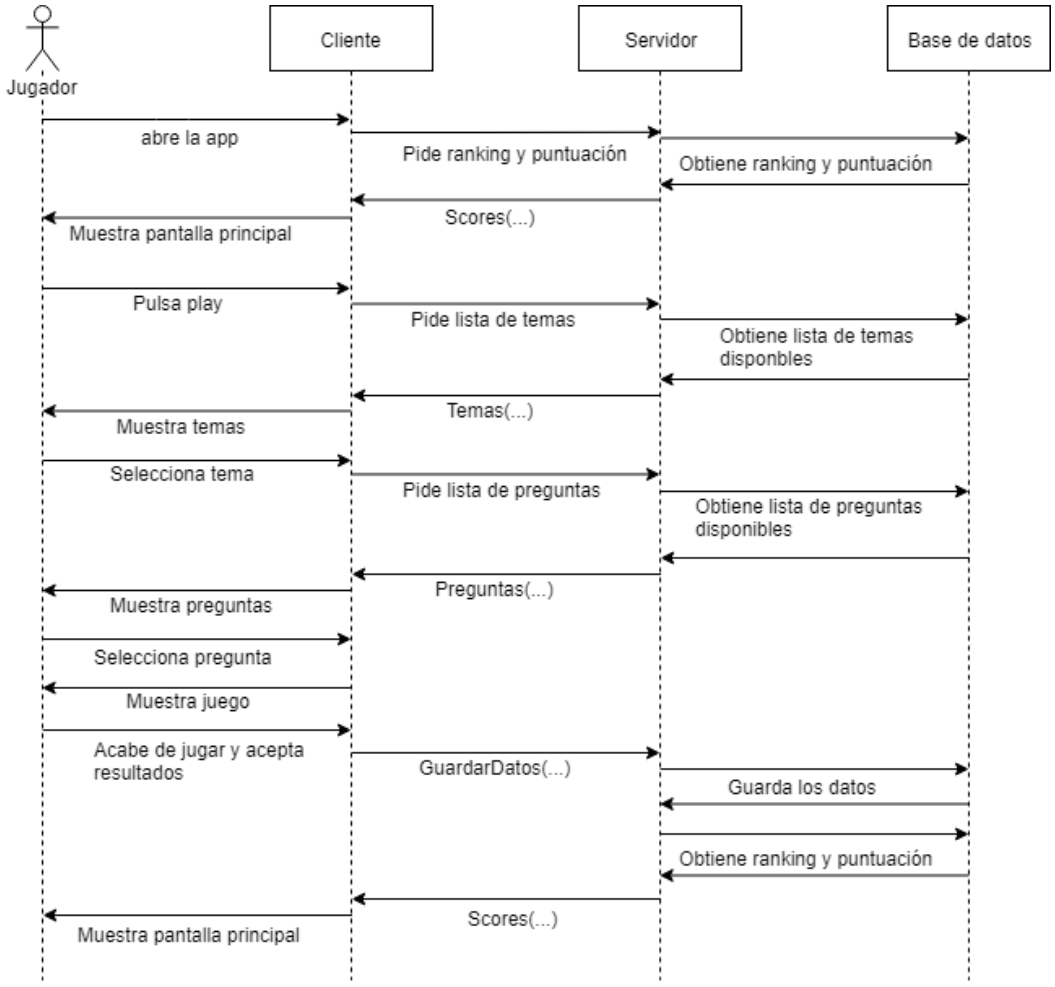


Ilustración 3 Diagrama de secuencia sobre caso de uso del rol jugador

La ilustración 3 muestra las siguientes acciones: el jugador entra a la aplicación (se supone que ya se ha identificado), obtiene el listado de los mejores jugadores y su propia puntuación. El jugador quiere contestar preguntas así que pulsará *play*, el sistema le muestra el listado de temas donde tiene preguntas que contestar, y el jugador elige un

tema. Una vez el jugador ha elegido el tema sobre el cual quiere contestar preguntas, el sistema muestra una lista de las preguntas disponibles, de nuevo el jugador elige la pregunta que desea contestar. Al elegir una pregunta el sistema muestra el juego con el que se obtendrán los datos necesarios para saber la opinión del jugador y la puntuación obtenida en el juego. El sistema almacena estos datos y finalmente devuelve al jugador a la pantalla inicial donde se muestra el ranking de los mejores jugadores y su puntuación.

A continuación, en la ilustración 4 se muestra un diagrama de secuencia referente a un caso de uso concreto del usuario administrador.

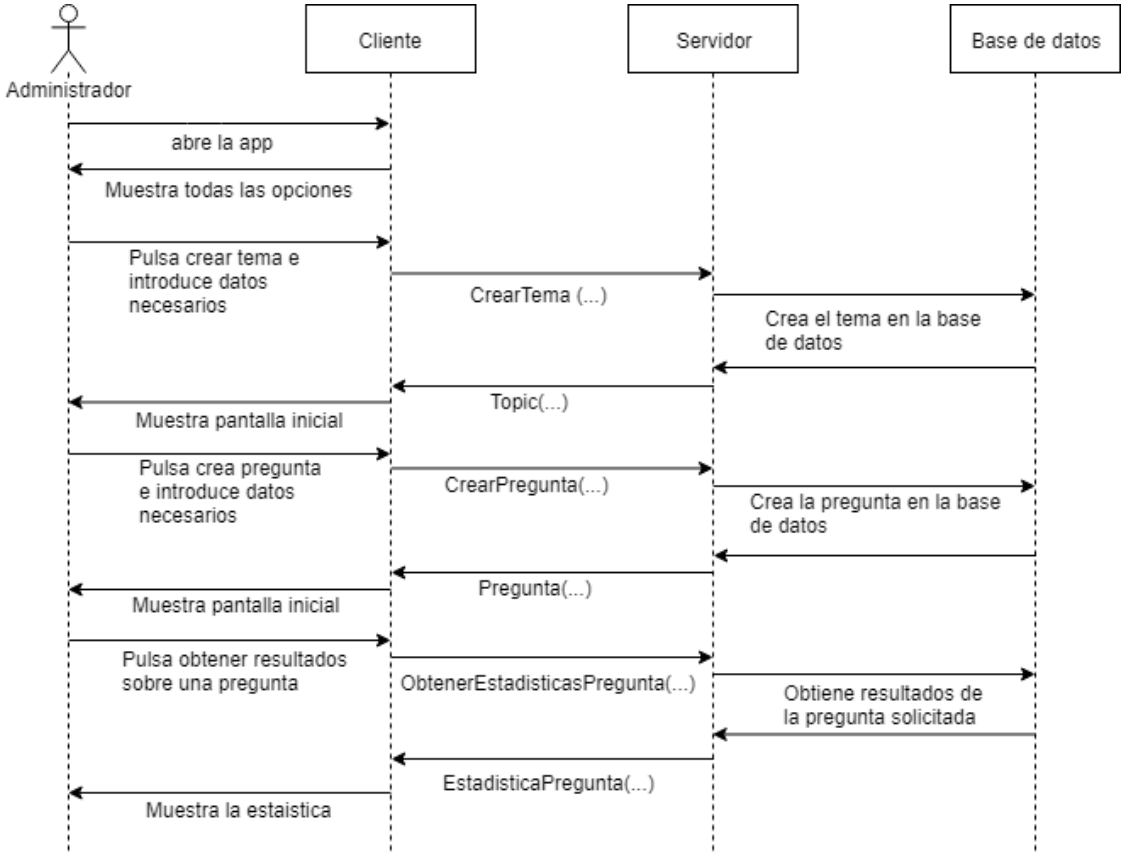


Ilustración 4 Diagrama de secuencia sobre caso de uso del rol administrador

La ilustración 4 muestra las siguientes acciones: el administrador entra en la aplicación (se supone que ya ha sido identificado), el cliente muestra el menú principal y el administrador pulsa en la opción de crear un tema nuevo, este introduce los datos necesarios y el cliente se encarga de comunicarse con el servidor para la creación de dicho tema. Una vez creado el tema el administrador se encuentra en el menú principal y esta vez selecciona la opción de crear una pregunta nueva, así pues, introduce todos los datos necesarios. Éste último envía los datos al servidor creando y asignando dicha pregunta a

los diferentes jugadores elegidos por el administrador. Una vez creada la pregunta el administrador pulsa la opción de obtener las estadísticas de una pregunta concreta, de esta manera el administrador selecciona la pregunta de la cual desea obtener las estadísticas, el cliente al igual que las otras veces, se comunica con el servidor para recuperar dichos datos. El servidor procesa la petición y devuelve las estadísticas de la pregunta seleccionada y el cliente es el encargado de mostrar dichos resultados.

5. Diseño

Como se ha comentado en la introducción, la aplicación Nitpicky se ha dividido en dos partes, por un lado, el *back-end* o API, donde se realizarán todas las funciones de mayor carga computacional y, por otro lado, el *front-end*, el cual es el responsable de las interfaces gráficas de usuario y algunas funcionalidades de poca carga computacional. De este modo se observa que Nitpicky utiliza el modelo cliente-servidor, donde el cliente se comunicará mediante peticiones HTTP con el servidor. Cabe desatar que en este TFG únicamente se va a diseñar e implementar el lado servidor.

Además, para la propia estructura del servidor se va a utilizar una arquitectura por capas. Cada una de las capas tendrá su propia funcionalidad. Esta arquitectura viene definida por el *framework* de la empresa, Devonfw.

En cuanto a la hora de almacenar todos los datos, se va a utilizar una base de datos relacional, con la cual se gestionará toda la información.

5.1 Modelo Cliente Servidor

La arquitectura cliente/servidor [4] se basa en la repartición de tareas. Se distribuye de una forma cooperativa el procesamiento de la información, donde los demandantes de recursos, llamados clientes, solicitan a los servidores el trabajo y la información deseada.

Mediante este modelo se logra que las dos partes, cliente y servidor, trabajen de forma desacoplada, lo que facilita su desarrollo y mantenimiento. El cliente es el encargado de realizar las llamadas al servidor con el fin de obtener la información requerida por el usuario. Además, también es el encargado de gestionar todas las interfaces gráficas de usuario, ya que el usuario interactuará directamente con dicho cliente.

Por otro lado, el servidor es el encargado de procesar las peticiones del cliente. Este realizará todas las tareas de mayor carga computacional, ya que la potencia y capacidad del servidor puede ser establecida e incluso distribuida en múltiples equipos, mientras que sobre la potencia del cliente sólo se puede indicar, pero no establecer, la potencia mínima

requerida. Además, el servidor expondrá unos determinados servicios, los cuales serán visibles y utilizados por el cliente con el fin de gestionar toda la información necesaria.

Un sistema que emplee este tipo de arquitectura sigue el siguiente esquema:

1- El cliente solicita la información necesaria al servidor.

2- La petición es enviada mediante HTTP y es recibida por el servidor.

3- El servidor procesa dicha solicitud.

4- El servidor envía mediante HTTP el resultado al cliente.

5- El cliente procesa la información recibida.

A continuación, se muestra la ilustración 5, donde se observa el esquema descrito anteriormente.

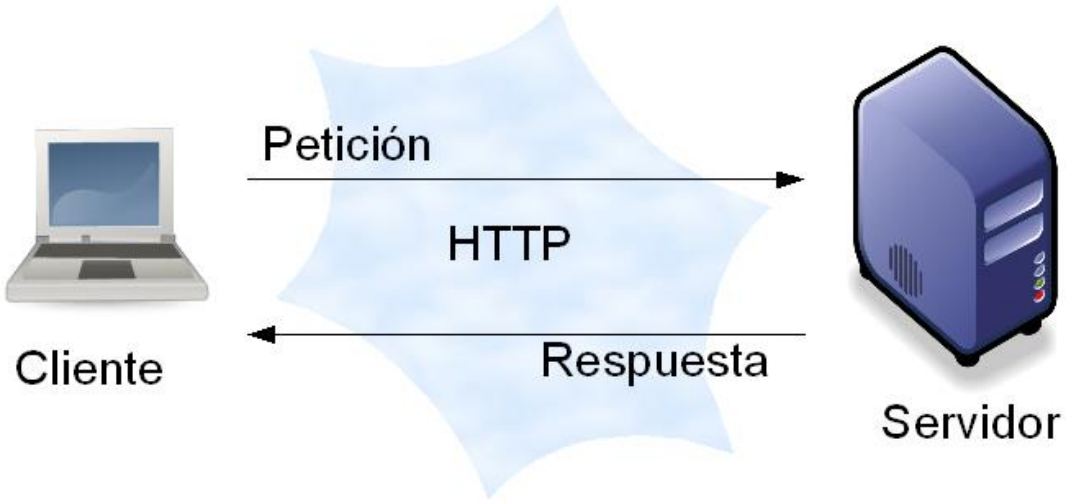


Ilustración 5 Modelo cliente/servidor

Mediante el uso de este modelo se consigue un sistema cuyo control es totalmente centralizado, dado que los accesos y los recursos son controlados por el servidor. Además, proporciona muchas facilidades para el mantenimiento e integración de nuevas tecnologías. De esta manera el sistema puede escalar con gran facilidad.

5.2 Arquitectura por capas

En este apartado se comentará la arquitectura que tendrá el propio servidor y el porqué de la utilización de dicha arquitectura. Se ha utilizado el patrón de desarrollo de NCapas, aplicando *Onion architecture* [5]. Mediante esta arquitectura se puede dividir el proyecto en diferentes capas, facilitando así la reutilización del código y separación de responsabilidades. La arquitectura propuesta utiliza interfaces para la cooperación entre capas. Cada capa contiene un conjunto de componentes independientes capaces de desarrollar una funcionalidad específica. Este patrón proporciona flexibilidad a la hora de implementar diferentes tipos de proyectos maximizando la eficiencia, el rendimiento y estrategias de despliegue.

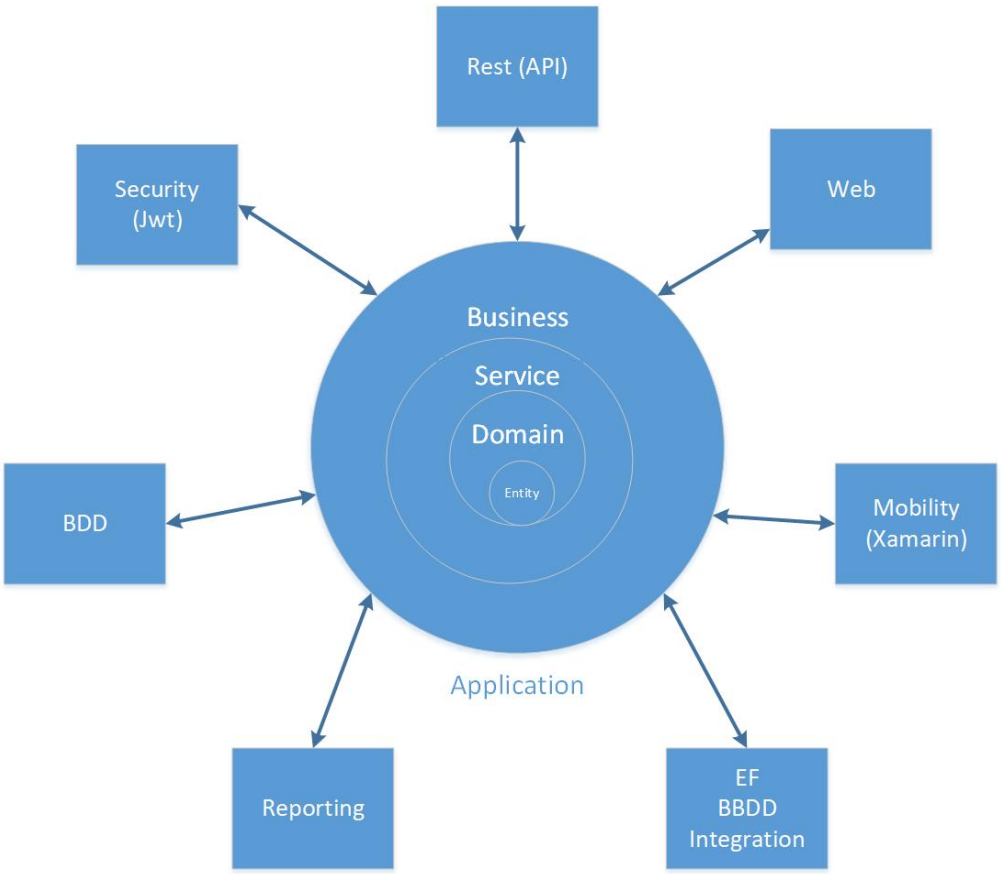


Ilustración 6 Arquitectura de n capas

5.2.1 Capas de la arquitectura

En esta arquitectura encontramos las siguientes capas (ilustración 6):

Capa de aplicación:

La capa de aplicación gestiona la interacción con el cliente, recibiendo las peticiones y comunicándose con la capa directamente inferior. Esta capa expone todos los servicios que puede utilizar el cliente con el fin de obtener la información necesaria.

Capa de negocio:

La capa de negocio implementa la funcionalidad central de la aplicación y encapsula la lógica de la aplicación. Esta capa hace posible que los datos estén optimizados y listos para diferentes consumidores de datos.

Capa de servicio:

La capa de servicio gestiona los datos obtenidos entre la capa de dominio y la capa de negocio. También transforma los datos para ser usados más eficientemente entre capas.

Capa de dominio:

La capa de dominio proporciona acceso a datos directamente expuestos desde otros sistemas. La fuente principal suele ser un sistema de base de datos.

5.3 Base de datos relacional

En este punto se va a explicar qué tipo de base de datos se va a utilizar en este proyecto comentando qué ventajas proporcionará a dicho proyecto.

Puesto que la aplicación va a necesitar una base de datos para poder almacenar toda la información, se diseñará una base de datos siguiendo el modelo relacional.

El modelo relacional es un modelo de organización y gestión de base de datos inventado por IBM [6]. Estos datos están almacenados en tablas compuestas por filas y columnas. Una de las mayores ventajas de este modelo es la facilidad de comprensión por el desarrollador inexperto. Además, este modelo se basa en la lógica de predicados para establecer relaciones entre distintos datos.

Para diseñar la tabla de datos se utiliza un diagrama de entidad-relación. Mediante este esquema se definen todas las entidades y las relaciones que van a existir en la aplicación. De esta manera el desarrollador puede comprender fácilmente el sistema y puede realizar la implementación, ya que este diagrama muestra la base de toda aplicación, que son los datos y las relaciones que existen entre ellos.

A continuación, la ilustración 7 muestra el diagrama de entidad-relación utilizado para la aplicación Nitpicky.

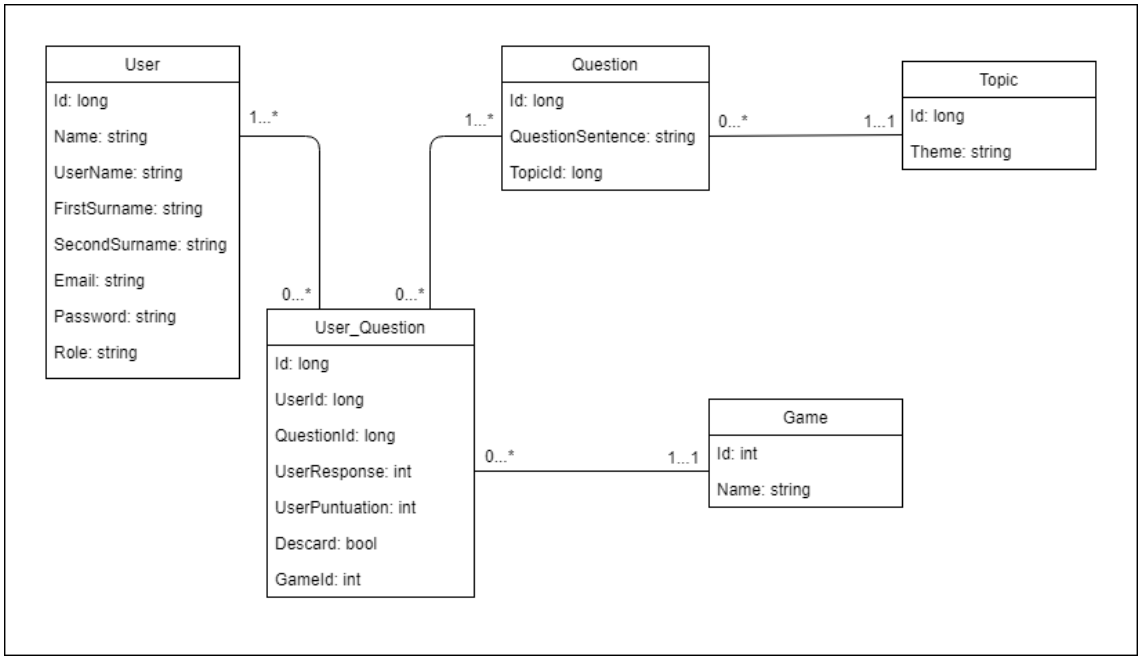


Ilustración 7 Diagrama entidad-relación

Como se observa en este diagrama, un usuario puede tener muchas preguntas que contestar y una pregunta puede tener muchos usuarios asociados. Esta relación entre usuario y pregunta se lleva a cabo gracias a la tabla User_Question, donde es necesario que exista tanto el usuario y la pregunta como el juego que se utilizará para responder la pregunta, en cada entrada de la tabla. Una pregunta solo puede y debe pertenecer a un tema, pero un tema puede tener o no preguntas.

Para hacer posible las relaciones entre tablas, se han definido como claves primarias todos los identificadores únicos de cada tabla, y como claves ajenas todos los atributos con nombre de tablas con terminación de Id. Como ejemplo, el atributo UserId de la tabla User_Question es clave ajena y está relacionada con la clave primaria de la tabla User que es el atributo Id.

A continuación, se comentará la función de las diferentes tablas:

- User: esta tabla almacenará toda la información de los usuarios con su identificador.
- Question: esta tabla almacenará el enunciado de la pregunta junto con el identificador del tema al que pertenece y su propio identificador.
- User_Question: esta tabla será la encargada de almacenar toda la información relacionada con la pregunta que debe contestar cada usuario.
- Topic: esta tabla almacenará todos los temas junto con su identificador propio.
- Game: se encargará de guardar el nombre de los diferentes juegos existentes en el sistema junto con su propio identificador.

6. Implementación

Una vez terminado el diseño de todos los elementos necesarios para el desarrollo de la aplicación, se va a proceder a la implementación de esta. En primer lugar, se comentarán las tecnologías y herramientas utilizadas. Posteriormente se describirá la metodología utilizada para implementar la aplicación entre otros aspectos.

6.1 Tecnologías y herramientas

Se van a describir a continuación todas las tecnologías y herramientas que se han utilizado para la implementación del proyecto, justificando la elección de cada una de ellas.

6.1.1 Devonfw

Devonfw [7] es una plataforma estándar con un enfoque industrializado con el fin de agilizar el proceso de desarrollo de proyectos software. Para ello, se integran diferentes tecnologías (java, node, typescript, .Net, serverless). Además, esta plataforma está constituida por productos de código abierto y de estándares definidos (OASP [8]). Para hacer uso de esta plataforma, el desarrollador parte de los planos de implementación para desarrollar una correcta aplicación.

Devonfw ha sido desarrollado por la empresa en la cual se ha llevado a cabo este TFG.

6.1.2 .NET y C#

.NET [9] es una plataforma desarrollada por Microsoft que ofrece un conjunto de herramientas, tecnologías y servicios que facilita el desarrollo de aplicaciones software.

.NET proporciona soporte para diferentes lenguajes, como F#, Visual Basic .Net, C++, C#, entre otros.

Para concretar, en este proyecto se ha utilizado .NET Core [10], ya que proporciona una serie de características como son:

Multiplataforma: permite desarrollar una aplicación que podrá ser ejecutada en los principales sistemas operativos de hoy en día, Windows, Linux y MacOS.

Código abierto: al ser código abierto, .NET Core favorece que el proceso de desarrollo sea más transparente.

Modular: .NET Core es modular, ya que se publica en paquetes de ensamblado más reducidos llamados *NuGet*.

Para desarrollar una aplicación en .NET Core es necesario utilizar C#, así que éste es el principal lenguaje del proyecto.

6.1.3 SQL

El lenguaje SQL permite gestionar de forma completa la base de datos de la aplicación: controlar el acceso a los datos, crear, borrar y modificar tablas, insertar, consultar, borrar y actualizar datos de las diferentes tablas. [11]

6.1.4 Postgres

Postgres [19] es un sistema de gestión de bases de datos relacional orientado a objetos y es de código abierto. Como servidor de base de datos, sus funciones principales son almacenar los datos de forma segura y devolverlos en respuesta a las solicitudes de otras aplicaciones de software. Puede manejar cargas de trabajo que van desde pequeñas aplicaciones de una sola máquina hasta grandes aplicaciones orientadas a Internet con muchos usuarios simultáneos.

6.1.5 Visual Studio 2017

Visual Studio es un entorno de desarrollo integrado desarrollado por Microsoft. Este entorno proporciona soporte a una gran variedad de lenguajes de programación, al igual que a entornos de desarrollo web. [12]

Se ha elegido este editor porque ha sido impuesto por la propia empresa. Éste editor ofrece una serie de herramientas muy útiles para el entorno .NET. Se ha trabajado con la última versión de este editor para poder disponer de todas las funcionalidades que éste ofrece.

6.1.6 REST (*REpresentational State Transfer*)

REST es una tecnología que permite la intercomunicación entre aplicaciones utilizando métodos HTTP. Algunas de sus características más importantes son:

- JSON es el formato en el que se transporta la información.
- También devuelve códigos de respuesta HTML. Los códigos de respuesta pueden ser 200(*Ok*), 403(*Forbidden*), 404(*Not found*), entre otros.
- Es necesario una URL y un método HTTP para utilizar esta tecnología. Los métodos HTTP son GET, para listar y leer; POST, para crear; DELETE, para eliminar; PATCH, para actualizar y PUT para reemplazar.

6.1.7 Swagger

Swagger [13] es una implementación del estándar OpenAPI [18] que proporciona un gran número de herramientas que ayudan a los desarrolladores a diseñar, construir, documentar y consumir servicios web REST proporcionando el contrato de los servicios. Este contrato se puede compartir entre desarrolladores de distintos equipos permitiendo el desarrollo en paralelo *front-end* y *back-end*. Además, dichas herramientas dan soporte a la realización de documentación automatizada, generación de código y generación de casos de prueba.

Swagger permite definir un contrato de comunicación entre el futuro consumidor del servicio y el proprio servidor. Este servidor, haciendo uso de Swagger, genera un cliente html permitiendo la publicación de cada servicio ofertado con el fin de que los consumidores conozcan todos los elementos necesarios para la correcta realización de la llamada al servicio. En el caso de esta aplicación, se ha utilizado especialmente esta funcionalidad. De esta forma el futuro cliente del servidor podrá saber qué elementos son los necesarios para cada servicio.

6.1.8 JSON WEB TOKEN

Json Web Token o JWT [14] es un conjunto de medios de seguridad basado en JSON donde se crea un token que sirve para enviar datos entre aplicaciones o servicios garantizando que los datos sean válidos y seguros.

Al igual que la herramienta Swagger, JWT también es una herramienta integrada en Devonfw, esta herramienta en concreto permite a la aplicación proteger a los usuarios de accesos no autorizados, ya que dispondrán de un token único cada uno.

6.1.9 Postman

Postman [15] es una herramienta que permite al desarrollador crear peticiones a APIs internas o de terceros con la finalidad de poder gestionarlas y probarlas.

Se ha utilizado Postman a la hora de realizar las pruebas de uso de la aplicación. Esta ha permitido simular las llamadas HTTP necesarias para probar todas las funcionalidades que se estudiaron en el análisis de requisitos y en el análisis de casos de uso.

6.1.10 FIDDLER

Fiddler [16] es un servidor proxy que sirve para depurar código HTTP. Fiddler se encarga de capturar tráfico HTTP y HTTPS y lo registra para que se pueda revisar por el usuario. Además, también tiene la capacidad de modificar el tráfico HTTP con el fin de solucionar problemas mientras se están enviando o recibiendo estas peticiones.

Se ha utilizado combinado con Postman (que es capaz de simular llamadas al servidor) para comprobar si las llamadas al servidor funcionan correctamente, ya que se ha podido observar el tráfico de la red y más concretamente las llamadas realizadas.

6.1.11 Chrome y Firefox

Google Chrome y Mozilla Firefox son dos de los navegadores más utilizados hoy en día. En este proyecto han sido utilizados para mostrar los contratos generados por la herramienta Swagger.

6.2 TDD (*Test-driven-development*)

TDD es una práctica de programación que se centra en el desarrollo dirigido por test. Se ha utilizado el libro *The Art of Unit Testing* [17], para la adquisición del conocimiento necesario para poder hacer uso de esta metodología. Para llevar a cabo esta metodología, se han diseñado e implementado los test antes que la implementación del código fuente. Esta técnica se divide en tres partes.

La primera parte es el desarrollo del test del requisito o funcionalidad específica que se desee cumplir. Para esta parte se utilizan test unitarios, esto significa que todos los test que se realicen tienen que cumplir una serie de aspectos: debe ser automático y repetible,

su implementación debe ser simple, fácil y rápida, su ejecución debe ser rápida, debe devolver siempre el mismo resultado ante la misma entrada, cada test debe de estar aislado del resto de test y por último cada test debe de ser relevante en el futuro.

Una vez implementados los test, se procede a la implementación del código fuente con el único objetivo de superar los test desarrollados anteriormente. En esta parte se implementa la funcionalidad estrictamente definida por el test, de esta forma es muy importante que los test de la aplicación recojan todas y cada una de las funcionalidades y requisitos especificados

Por último, se realiza la refactorización, esto es, una revisión del código que ya supera el test satisfactoriamente, para su optimización o limpieza. Esta parte es muy importante, ya que aquí es donde se asegura que el código fuente implementado es óptimo y leíble. También es necesario analizar el código realizado y hacerse unas preguntas como: ¿Dentro de un año o dos se entenderá él código? o ¿Hay alguna función que se pueda optimizar?

La ilustración 8 muestra un diagrama de flujo del desarrollo de una aplicación utilizando TDD.

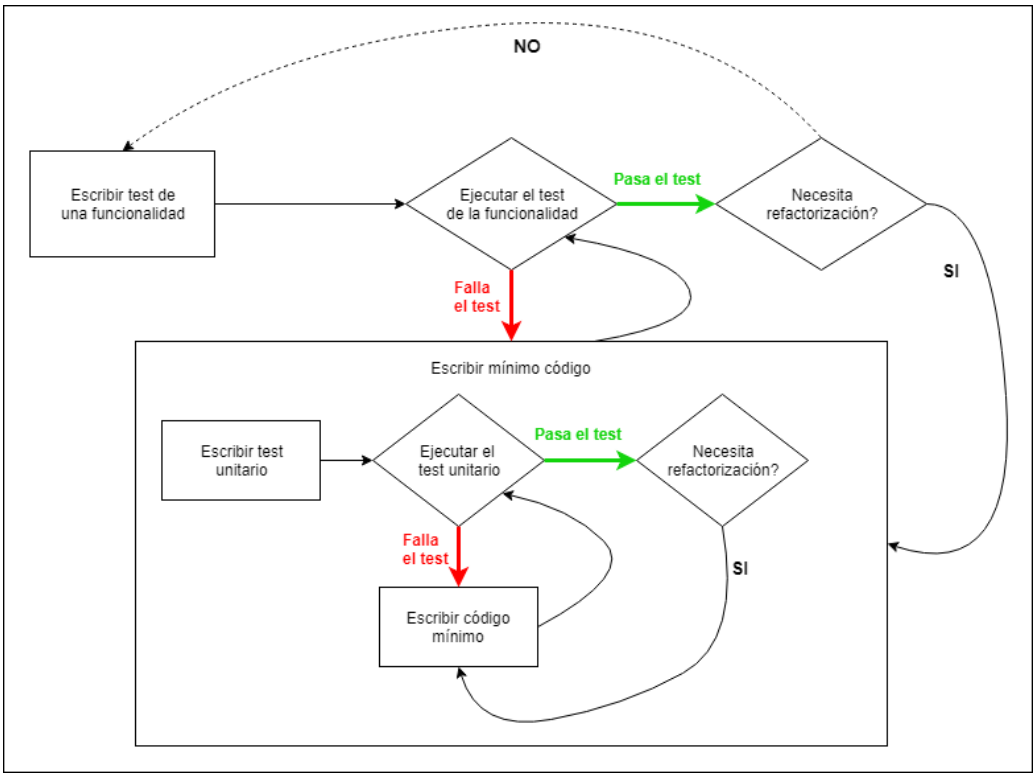


Ilustración 8 Esquema metodología TDD

Mediante esta técnica el programador se asegura que su código fuente supera satisfactoriamente cada uno de los test y a su vez que cumple todas y cada una de las funcionalidades requeridas.

Este proyecto se ha desarrollado implementando dicha técnica, con ello se ha conseguido una lista con test que comprueban todas las funcionalidades que se han descrito en el análisis de requisitos. Además, se ha logrado optimizar el código fuente, ya que después de implementar cada funcionalidad se analizó el código con el fin de hacerlo óptimo y leíble.

A continuación, en la ilustración 9 se muestra la implementación de uno de los test realizados junto con una breve descripción del mismo.

```
[Test]
public async Task Player_Get_Scores()
{
    RankingDto[] ranking = new RankingDto[3];
    ranking[0] = new RankingDto("antonio", 50);
    ranking[1] = new RankingDto("juan", 25);
    ranking[2] = new RankingDto("josep", 2);
    PlayerDto playerDto = new PlayerDto("josep", 2, 3);

    ScoresDto scoresExpected = new ScoresDto(ranking, playerDto);

    var scoresResult = await player.GetScores("josep");

    Assert.True(AreScoreDtoEquals(scoresExpected, scoresResult));
}
```

Ilustración 9 Ejemplo de test realizado

El test se divide principalmente en tres partes, una es la creación del objeto que se espera obtener al realizar el servicio, dicha creación se muestra en el cuadro amarillo. Se utiliza la nomenclatura nombre del servicio más la terminación *Expected* con la finalidad de que el código sea más leíble y quede claro que éste es el objeto esperado de dicho servicio. Por otra parte, está la sentencia enmarcada en azul, aquí es donde se realiza la llamada al servicio implementado. El objeto resultante recibe el nombre a partir de dos partes, la primera el nombre del servicio y la segunda parte *Result*. Al igual que en la parte anterior recibe este nombre con el fin de hacer más leíble el código. Por último, la sentencia enmarcada en rojo comprueba si el objeto resultante es igual que el esperado. Cabe destacar que el título que recibe el test es muy importante, ya que debe de seguir una

buena nomenclatura para facilitar la tarea de comprensión del código a futuros desarrolladores.

6.3 Seguridad

En este punto se va a detallar la seguridad que se ha implementado en la aplicación. Como ya se ha comentado en las tecnologías, se ha utilizado JWT. Esta es una herramienta que se encuentra en una gran variedad de tecnologías dedicadas a comunicaciones y transporte de datos en la red. Esta herramienta permite la creación de *tokens* únicos con los cuales se puede cifrar información sensible de la aplicación.

En el caso de este proyecto se ha utilizado esta herramienta para generar un *token* que encapsula el identificador de usuario junto con su rol, con el fin de proteger los datos de las personas que utilicen la aplicación. Mediante JWT ya no será necesario enviar ningún dato de usuario en el cuerpo del mensaje, ya que se enviará el *token* en la cabecera del mismo. Dicho token se generará al inicio de la sesión del usuario y contendrá toda la información relevante de éste.

Además, se ha utilizado un sistema de roles para que los usuarios solo puedan realizar las acciones que su rol les permite.

Esta comprobación de seguridad se realiza en los controladores, donde se evalúa si el usuario pertenece a ese rol dando permiso para entrar o no. En la ilustración 10 se puede observar cómo se realiza dicha comprobación.

```
[Route("/nitpicky/services/rest/playermanagement/scores")]
[EnableCors("CorsPolicy")]
[Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme, Roles = "player")]
public async Task<IActionResult> GetScores()
```

Ilustración 10 Comprobación de rol en el controlador

De esta forma la aplicación es capaz de identificar el rol del usuario, ya que utiliza el *token* descifrado para obtener toda la información necesaria.

Para descifrar el token se ha utilizado la función mostrada en la ilustración 11 con la cual se obtiene el nombre de usuario con el que posteriormente, mediante una consulta a la base de datos, se obtiene el resto de la información, como puede ser el identificador o el rol.

```
private string GetUserName(HttpContext httpContext)
{
    var headerValue = httpContext.Request.Headers["Authorization"].ToString().Replace("Bearer", string.Empty).Trim();
    var handler = new JwtSecurityTokenHandler();
    var jwtUser = handler.ReadJwtToken(headerValue);

    return jwtUser.Claims.FirstOrDefault(x => x.Type == "UserName").Value;
}
```

Ilustración 11 Método para obtener información del token

De esta forma se garantiza la autenticidad de los usuarios, ya que cada usuario tendrá su *token* único y además se garantiza que cada funcionalidad del sistema es realizada por el usuario con el rol especificado, denegando así el permiso a aquellos usuarios que no pertenezcan a dicho rol.

6.4 Importancia de las llamadas asíncronas

Como ya se ha comentado, la aplicación servidor recibirá una gran cantidad de llamadas provenientes de los clientes que utilicen dicho servidor. Estas llamadas podrían bloquear la aplicación en algunos puntos críticos donde la información pueda tardar en procesarse y enviarse, ya sea por el volumen de datos solicitados o por la velocidad de la red. La solución a este problema es hacer llamadas asíncronas, es decir hacer llamadas que se ejecutarán en segundo plano evitando así el bloqueo de la aplicación tanto en el lado cliente como en el lado servidor. La ilustración 12 muestra uno de los servicios implementados en el servidor, donde todos los métodos son asíncronos, esto nos permitirá obtener los datos solicitados sin bloquear la aplicación.

```
public async Task<ScoresDto> GetScores(long userId)
{
    RankingDto[] ranking = await GetUsersRanking();
    PlayerDto player = await GetUserInformation(userId);

    ScoresDto scores = new ScoresDto(userId, ranking, player);

    return scores;
}

public async Task<UserQuestionDto> SaveFeedbackResults(long userId, long questionId, int response, int puntuation)...

public async Task<TopicsAvaliableDto> GetTopicsAvaliable(long userId)...

public async Task<QuestionsAvaliableDto> GetQuestionsAvaliable(long userId, long topicId)...

public async Task<UserQuestionDto> DescardQuestion(long userId, long questionId)...

private async Task<PlayerDto> GetUserInformation(long userId)...

public async Task<RankingDto[]> GetUsersRanking()...
```

Ilustración 12 Ejemplo métodos asíncronos en un servicio

Como se observa en la imagen, en la declaración de los métodos es necesario utilizar la palabra reservada *async* para poder hacer llamadas asíncronas a otros métodos asíncronos.

También se puede observar en la primera línea del método desplegado que para realizar una llamada a un método asíncrono es necesaria la palabra reservada *await*.

La clase *Task<TResult>* representa una única operación que devuelve un valor y que se ejecuta asincrónicamente. Esta clase permite utilizar una gran variedad de métodos que permitirán saber por ejemplo si una tarea esta completada, cancelada, bloqueada, etc.

6.5 Estructura y entidades de la aplicación

6.4.1 Estructura del proyecto

En este apartado se va a describir la estructura de carpetas que se ha utilizado a la hora de implementar y organizar el proyecto. En primer lugar, es necesario comentar que se ha seguido una arquitectura por capas como se ha explicado anteriormente, esto significa que se ha dividido la aplicación en tres capas bien diferenciadas, la capa de negocio, la capa de servicios y la capa de dominio y entidades.

Es necesario decir que la capa de negocios y servicios se ha implementado mediante carpetas de gestiones, las cuales contienen todos los elementos que hacen posible la correcta gestión de cada entidad. Cabe destacar que se ha generado una carpeta nueva por cada entidad y por cada rol de la aplicación. Dentro de las carpetas de gestión de entidades tendremos solamente los servicios que afectarán directamente a estas entidades, es decir tendremos los métodos de CRUD (*Create*, *Read*, *Update*, *Delete*), los cuales permitirán crear, leer, modificar y borrar dichas entidades, además se encontrarán algunas funcionalidades específicas requeridas. Además, dentro de estas carpetas también están los conversores, los cuales nos permiten convertir entidades de la base de datos en objetos de transferencia de datos, que son usados para las comunicaciones con el cliente. A continuación, en la ilustración 13, se muestra un gestor para comprender la organización descrita:

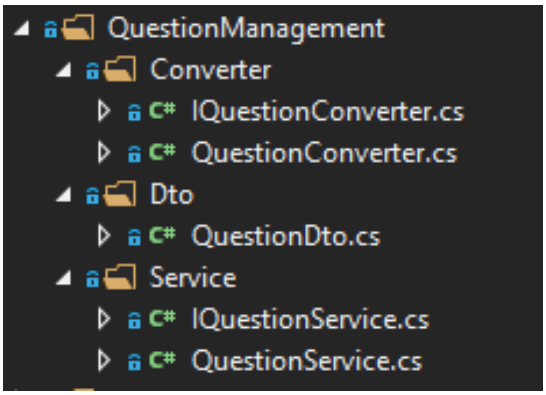


Ilustración 13 Estructura ejemplo carpeta de gestión

Por otra parte, cabe destacar las carpetas donde se gestionan los dos roles de la aplicación. Estas carpetas están subdivididas por otras tres carpetas. Una de ellas, *Controller*, aloja el controlador del rol, éste será el encargado de recibir o responder las peticiones del cliente. Además, el controlador se comunicará con la capa de servicio, alojada en otra subcarpeta que recibe el nombre de *Services*. Esta carpeta contendrá tanto la interfaz como la implementación del servicio, que contendrá toda la lógica necesaria para cumplir todas las funcionalidades del rol y también se comunicará con los servicios de las entidades con el fin de obtener los datos necesarios de ellos. Por último, la subcarpeta *Dto*, contiene los objetos de negocio, los cuales serán recibidos y enviados por parte del cliente. A continuación, se muestra en la ilustración 14 una imagen con el fin de entender mejor la organización descrita.

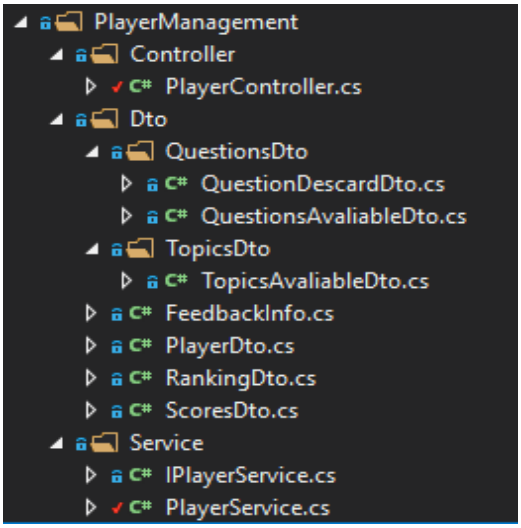


Ilustración 14 Carpeta gestión rol Jugador

Por último, en cuanto a la capa de dominio y entidades se refiere y como ya se ha comentado anteriormente, se ha generado una clase por cada entidad y posteriormente un

contexto donde se establecen las relaciones entre entidades. Cabe destacar que la capa de dominio está implementada en dos proyectos diferentes, en el proyecto *Entities* es donde tenemos las entidades y el contexto, mientras que el otro proyecto, llamado *UnitOfWork*, es el encargado de realizar todas las gestiones de la base de datos, siendo una capa intermedia entre la capa de servicios y la capa de dominio y entidades.

6.4.2 Entidades del proyecto

En este apartado se va a hablar de cómo se han organizado las entidades que confeccionan la aplicación. Como se ha comentado en el capítulo anterior, el diagrama de entidad-relación es la base de todo el proyecto, así que para la implementación de las entidades se ha seguido estrictamente dicho esquema.

Se ha generado una clase por cada tabla del diagrama diseñado. Y posteriormente se ha implementado un contexto donde se establecen todas las relaciones entre entidades, llamado ModelContext. Esta estructura mencionada se puede observar en la ilustración 15.

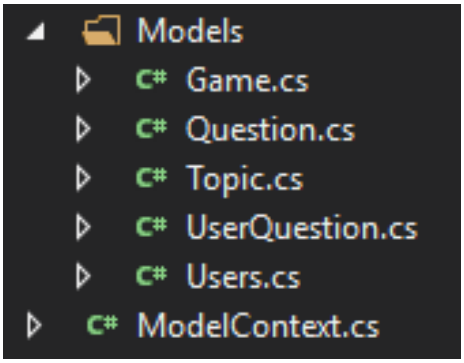


Ilustración 15 Modelo de datos

6.6 Servicios implementados

En este punto se van a describir todos los servicios implementados que va a ofrecer la aplicación. Se va a hacer una clasificación donde primero se mostrarán todos los servicios que pueden ser utilizados por los diferentes roles que existen en la aplicación. Todas las ilustraciones que se muestran en este apartado han sido generadas automáticamente por el servidor, haciendo uso de la herramienta Swagger.

En primer lugar, se muestran en la ilustración 16 los servicios implementados para la gestión de usuarios, los cuales proporcionan soporte para la autenticación y registro de usuarios:

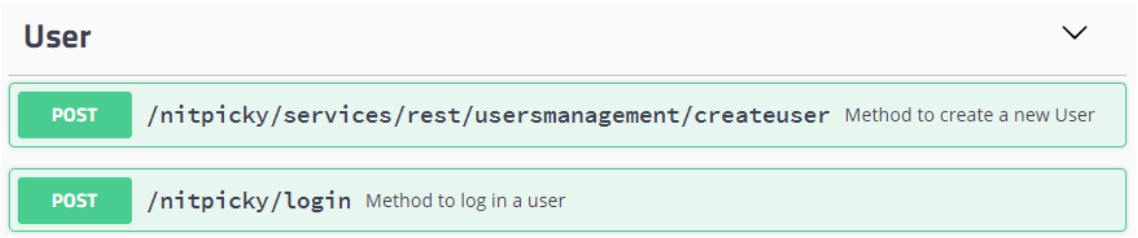


Ilustración 16 Servicios de cualquier usuario

- CreateUser: El sistema creará un nuevo usuario con la información introducida por el usuario.
- Login: El sistema comprobará las credenciales del usuario y generará el token de acceso en caso de ser correctas

Como se puede observar en la ilustración 17, los servicios implementados para el rol jugador son:

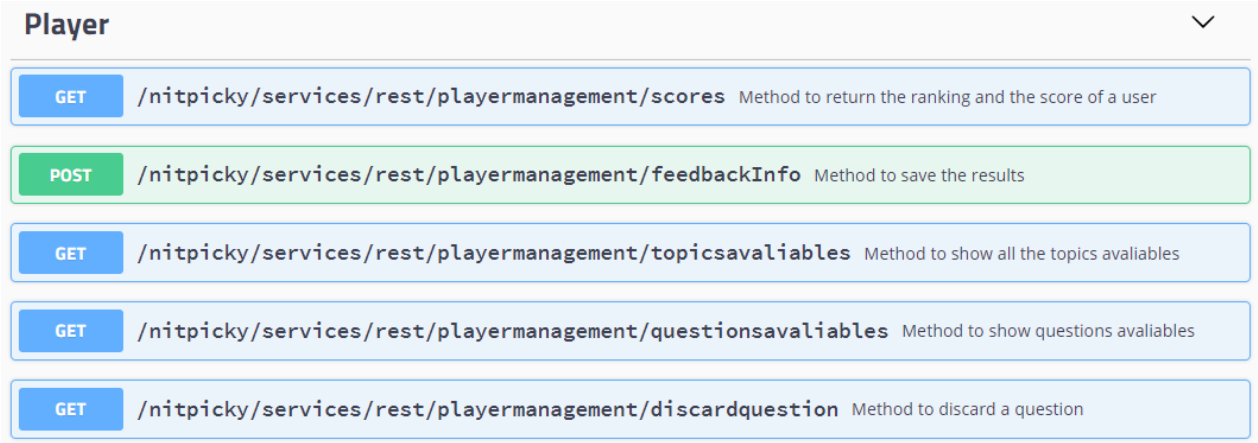


Ilustración 17 Servicios rol Jugador

- Scores: Este servicio devuelve el ranking de los mejores jugadores, junto con el nombre del usuario que solicita la información, su puntuación y su posición en el ranking.
- TopicsAvaliables: Este servicio devuelve todos los temas donde el jugador tiene preguntas pendientes por contestar.

- QuestionsAvaliable: Este servicio devuelve todas las preguntas de un tema específico que el jugador tiene pendientes de contestar.

- DescardQuestion: Este servicio permite al jugador descartar una pregunta de las cuales tenía pendientes de contestar. Una vez descartada una pregunta ya no aparecerá en preguntas disponibles para ese jugador.

- FeedbackInfo: Este servicio recoge los datos obtenidos del jugador con su puntuación en el juego y su opinión. Estos datos serán guardados en la base de datos para la obtención de las estadísticas.

A continuación, se muestra en la ilustración 18 todos los servicios implementados del rol administrador:

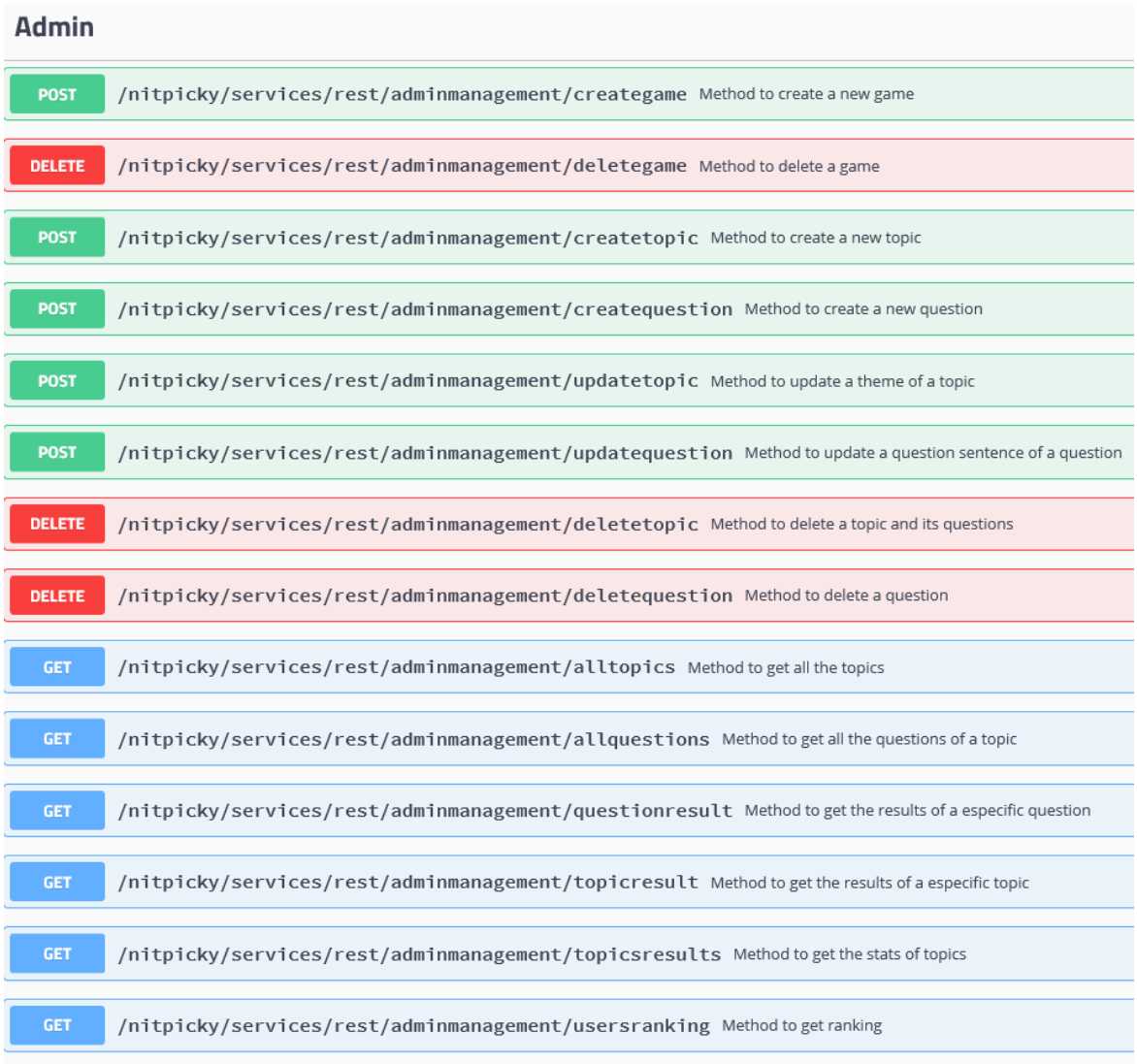


Ilustración 18 Servicios rol Administrador

- CreateGame: El servicio crea un juego.
- DeleteGame: El servicio elimina un juego.
- CreateTopic: El servicio crea un tema.
- CreateQuestion: El servicio crea una pregunta perteneciente a un tema asignando dicha pregunta a todos los jugadores que el administrador haya seleccionado.
- UpdateTopic: El sistema modifica el título del tema por el nuevo título que ha enviado el administrador.
- UpdateQuestion: El sistema modifica el texto de la pregunta al nuevo texto enviado por el administrador.
- DeleteTopic: El sistema borra la el tema seleccionado por el administrador y las preguntas asociadas a dicho tema.
- DeleteQuestion: El sistema borra la pregunta seleccionada por el administrador.
- AllTopics: El sistema devuelve todos los temas que están registrados en la aplicación.
- AllQuestions: El sistema devuelve todas las preguntas asociadas a un tema específico.
- QuestionResult: El sistema devuelve las estadísticas de la pregunta que el administrador ha solicitado.
- TopicResult: El sistema devuelve las estadísticas del tema que el administrador ha solicitado.
- TopicsResults: El sistema devuelve las estadísticas de todos los temas a nivel general.

Por último, se muestran en la ilustración 19 todos los servicios implementados para el rol super usuario.

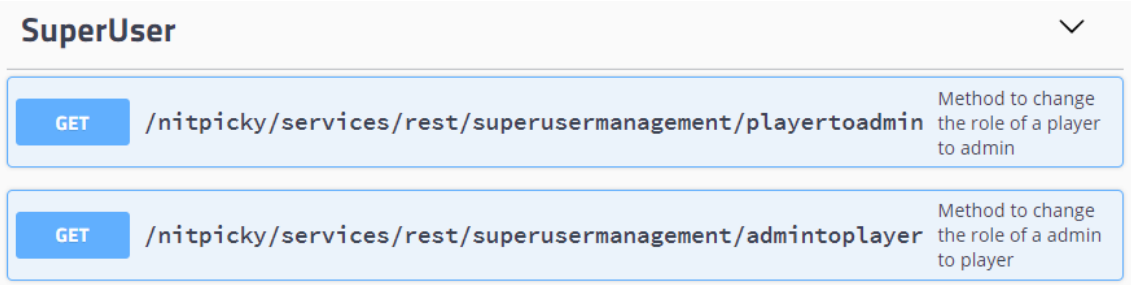


Ilustración 19 Servicios rol Super Usuario

- PlayerToAdmin: El sistema cambiara el rol de un usuario jugador a un usuario administrador.

- AdminToPlayer: El sistema cambiara el rol de un administrador jugador a un usuario jugador.

7. Pruebas

Este capítulo se dedicará a estudiar el comportamiento de la aplicación, verificando que todas las salidas se corresponden con las salidas esperadas. Este punto se focalizará en el estudio de los test realizados durante la implementación y de las pruebas de uso, el cual se dividirá en las funciones que puede realizar cada uno de los roles existentes en el sistema, el rol administrador, el rol jugador y el rol super usuario.

7.1 Pruebas de test

En este trabajo se ha desarrollado una aplicación que sigue estrictamente todos los requisitos especificados en el análisis de requisitos y casos de uso. Como la implementación se ha realizado utilizando la metodología TDD, explicada anteriormente, se han realizado los test que comprueban toda la funcionalidad y como se puede ver en la ilustración 20, al finalizar la implementación de la aplicación el resultado de todos los test ha sido satisfactorio.

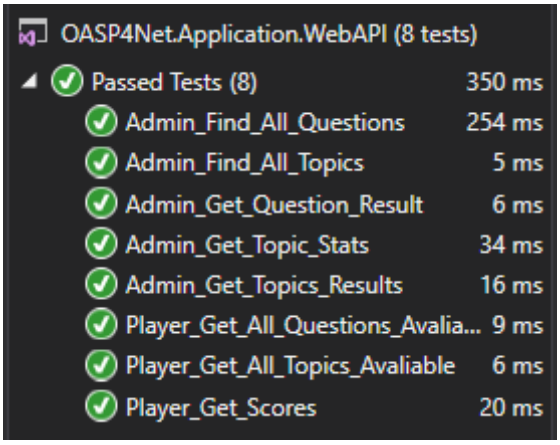


Ilustración 20 Test satisfactorios

7.2 Pruebas de uso

Después de realizar la comprobación de los test unitarios, con el fin de encontrar un comportamiento inusual o incorrecto debido a algún caso que no se haya analizado, se ha realizado una prueba de uso utilizando Postman, herramienta que ha permitido realizar peticiones al servidor, las cuales han servido para analizar las diferentes respuestas del servidor.

A continuación, se van a mostrar unas imágenes mediante las cuales se va a explicar el uso de la herramienta Postman. Posteriormente se mostrará la referencia a los anexos donde se encuentran todas las imágenes que muestran el correcto funcionamiento de todos los servicios implementados.

En primer lugar, la ilustración 21 muestra la pantalla principal de la herramienta Postman. En el cuadro naranja, se elige el tipo de llamada que se va a realizar, en el caso de esta aplicación se utilizan llamadas de tipo GET, POST y DELETE.

En el cuadro verde se encuentra el campo donde se especifica la url del servicio al que se quiere acceder, y a la derecha el botón de enviar. Justo debajo de estos campos, cuadro rojo, tenemos diferentes opciones para modificar las autorizaciones, las cabeceras y el cuerpo de la llamada.

En el campo de color azul encontramos un espacio donde escribir el cuerpo del mensaje, siempre en formato JSON. Cabe destacar que este campo solo se rellenará en llamadas de tipo POST. Seguidamente, se encuentra el cuadro negro, en el cual se podrá observar tanto las cabeceras como el cuerpo de la respuesta de la llamada.

Postman ofrece, en el cuadro amarillo, información del estatus de la respuesta, del tiempo transcurrido entre que lanza la llamada y recupera el resultado del servidor y el tamaño de este. Por último, en el campo de color rosa es donde se va a mostrar el resultado de la llamada.

Ilustración 21 Pantalla principal Postman

En segundo lugar, se muestra la imagen 22. Esta imagen corresponde a la opción de autorización del cuadro rojo de la imagen anterior. En este apartado se definen las autorizaciones necesarias que necesita la llamada para ser respondida por el servidor. El tipo de autorización se puede elegir en el campo de color rosa.

En este proyecto, como ya se ha explicado en el apartado de seguridad, se ha utilizado JWT que genera un token para cada usuario con el cual se podrá identificar. Así pues, para introducir dicho token en la llamada, se utilizará el tipo *bearer token*, y se introducirá en el campo de color verde donde se puede leer *Token*.

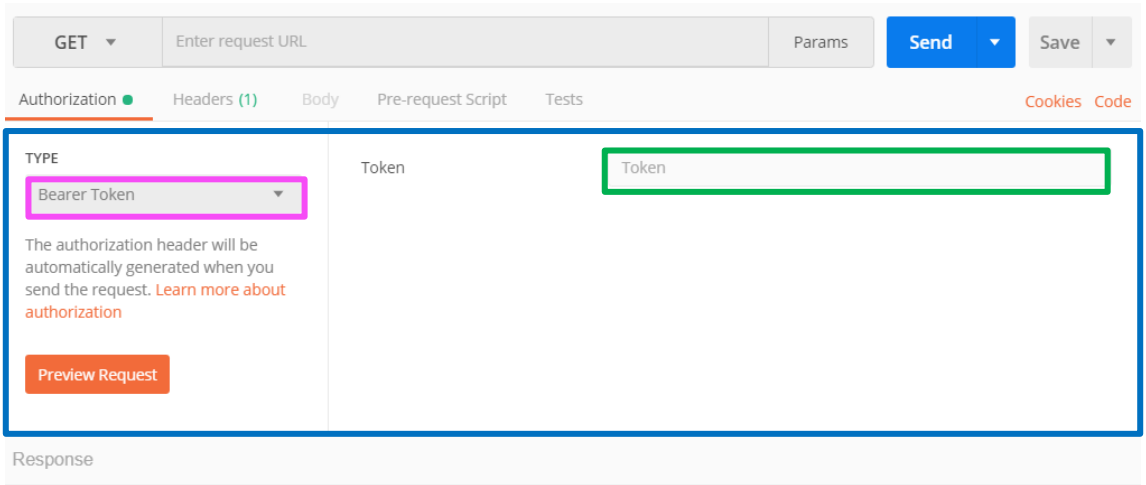


Ilustración 22 Pestaña seguridad Postman

El correcto funcionamiento de todos los servicios implementados se refleja en cada una de las imágenes mostradas en el Anexo 1, ya que la respuesta del servidor ha sido la esperada en todo momento. Dicho anexo se ha estructurado en tres partes: pruebas de uso del rol jugador, pruebas de uso del rol administrador y pruebas de uso del rol super usuario.

8. Conclusiones

Este trabajo tenía como objetivo principal diseñar e implementar la parte servidor de una aplicación web que permitía obtener la opinión de los usuarios respecto a temas propuestos por los administradores.

Tal y como se ha mostrado en el capítulo de pruebas, se puede decir que se ha conseguido alcanzar dicho objetivo. Además, en el punto 1.1, se muestran los objetivos secundarios del proyecto. Tras todo el trabajo, se puede concluir que finalmente se ha logrado abarcar satisfactoriamente todos estos objetivos. Aunque cabe destacar que alguno de ellos ha resultado más costoso de lo pensado. Uno de los más difíciles de conseguir ha sido el uso de la metodología TDD, ya que ha sido necesario leer mucha documentación al respecto y comprenderla de un modo adecuado.

Por otra parte, centrándonos en los requisitos, es necesario comentar que la aplicación cumple todos y cada uno de los establecidos al inicio del proyecto. Debido a este logro de objetivos y requisitos, se ha conseguido un elevado grado de satisfacción con la realización del proyecto.

Respecto al desarrollo de las diferentes fases del proyecto, cabe destacar que, por lo general, se han llevado a cabo en el tiempo estimado en la planificación previa. El diseño de la aplicación, más concretamente el modelo de datos, ha sido una gran sorpresa en cuanto a tiempo porque ha requerido de mucho más tiempo del esperado. En cambio, la fase de análisis de requisitos ha sido más rápida de lo previsto.

En cuanto a la implementación, se debe comentar que era una de las primeras veces que trabajaba en un entorno de estas características. Debido a esto, se necesitó un periodo de adaptación. No obstante, una vez pasado este periodo, el trabajo se llevó a cabo de una manera ágil y satisfactoria.

A continuación, se hablará de la relación que existe entre este TFG y los estudios cursados durante el grado. En general han sido muchas las asignaturas las que han ayudado en algún momento del proyecto, pero se van a destacar las más significativas. Una de las asignaturas que más útil ha resultado para la realización de este proyecto ha sido la asignatura de Bases de datos y sistemas de la información. Esta asignatura fue cursada en

tercero y gracias a ella, durante la realización del proyecto, se ha podido gestionar de manera correcta la base de datos. También ha sido útil retroceder a dicha asignatura para diseñar el modelo de datos. Otra de las asignaturas que a destacar es la de Gestión de proyectos. Gracias a esta se ha podido gestionar el tiempo de forma óptima y satisfactoria. Por último, cabe destacar que la experiencia Erasmus ha sido de gran ayuda, ya que durante esta se realizaron diversos proyectos que proporcionaron soltura y experiencia a la hora de desarrollar aplicaciones.

Para finalizar este TFG se va a hablar del aprendizaje que este ha supuesto. En primer lugar, destacar que ha sido la primera experiencia profesional la cual ha proporcionado una visión de la realidad fuera de la facultad. Por otro lado, a nivel de conocimientos técnicos, se ha experimentado una mejora notable y un gran aprendizaje debido a que se ha trabajado con diferentes tecnologías. Por último, mirando hacia el futuro, la realización de este TFG ha permitido ponerse en la piel de nuestro yo del futuro, haciéndonos ver que ese es el futuro que nos gusta.

9. Referencias

[1] Ana Ruiz Caballero. Estudio de la gamificación de una empresa para incentivar la motivación [En línea] [fecha de consulta: 24 de marzo de 2018]. Disponible en: https://uvadoc.uva.es/bitstream/10324/18243/1/TFG-I-417.pdf

[2] Especificación de Requisitos según el estándar de IEEE 830 (22 de octubre de 2008) [En línea] [fecha de consulta: 6 de abril de 2018]. Disponible en https://www.fdi.ucm.es/profesor/gmendez/docs/is0809/ieee830.pdf

[3] Diagrama de Casos de uso [En línea]. Polimedia, realizado por Penades Gramage, maría Carmen, 28 de junio de 2016 [fecha consulta 8 de abril de 2018]. Disponible en: http://hdl.handle.net/10251/66596

[4] CIUBOTARU, B. MUNTEAN, G. *Advanced Network Programming-Principles and TechniquesTítulo*. Londres: Springer, 2013, pp. 89-91. ISBN 978-1-4471-5291-0.

[5] Sandeep Singh Shekhawat, *Onion Architecture in ASP.NET Core MVC* [en línea] [fecha de consulta: 10 abril 2018]. Disponible en https://www.c-sharpcorner.com/article/onion-architecture-in-asp-net-core-mvc/

[6] C.J. Date, *EDGAR F. ("TED") CODD* [en línea] [fecha de consulta: 16 abril 2018]. Disponible en https://amturing.acm.org/award_winners/codd_1000892.cfm

[7] Devonfw [En línea] [fecha consulta: 14 de abril de 2018]. Disponible en https://troom.capgemini.com/sites/vcc/devon/overview.aspx

[8] OASP [En línea] [fecha consulta: 20 de junio de 2018]. Disponible en https://oasp.github.io/oasp

[9] .NET: Free. Cross-platform. Open source. A developer platform for building apps [En línia] [fecha de consulta 22 de junio de 2018]. Disponible en: https://www.microsoft.com/net/

[10] .NET Core Guide [En línia] [fecha de consulta 22 de junio de 2018] Disponible en: https://docs.microsoft.com/en-us/dotnet/core/index

[11] The world's largest web developer site. SQL tutorial [En línea] [fecha de consulta 5 de junio de 2018] Disponible en: https://www.w3schools.com/sql/

[12] Visual Studio 2017 [En línea] [fecha de consulta 20 de abril de 2018]. Disponible en: https://www.visualstudio.com/

[13] Swagger [En línea] [fecha de consulta 2 de mayo de 2018]. Disponible en: https://swagger.io/

[14] Json Web Token [En línea] [fecha de consulta 3 de mayo de 2018]. Disponible en: https://jwt.io/

[15] Postman [En línea] [fecha de consulta 3 de mayo de 2018]. Disponible en: https://www.getpostman.com/

[16] Fiddler [En línea] [fecha de consulta 4 de mayo de 2018]. Disponible en: https://www.telerik.com/fiddler

[17] FEATHERS, M. MARTIN, R. *The art of Unit Testing*. Segunda edición. Shelter Island: Manning, 2014. ISBN 9781617290893.

[18] OpenAPI [En línea] [fecha de consulta 16 de mayo de 2018]. Disponible en: https://www.openapis.org/

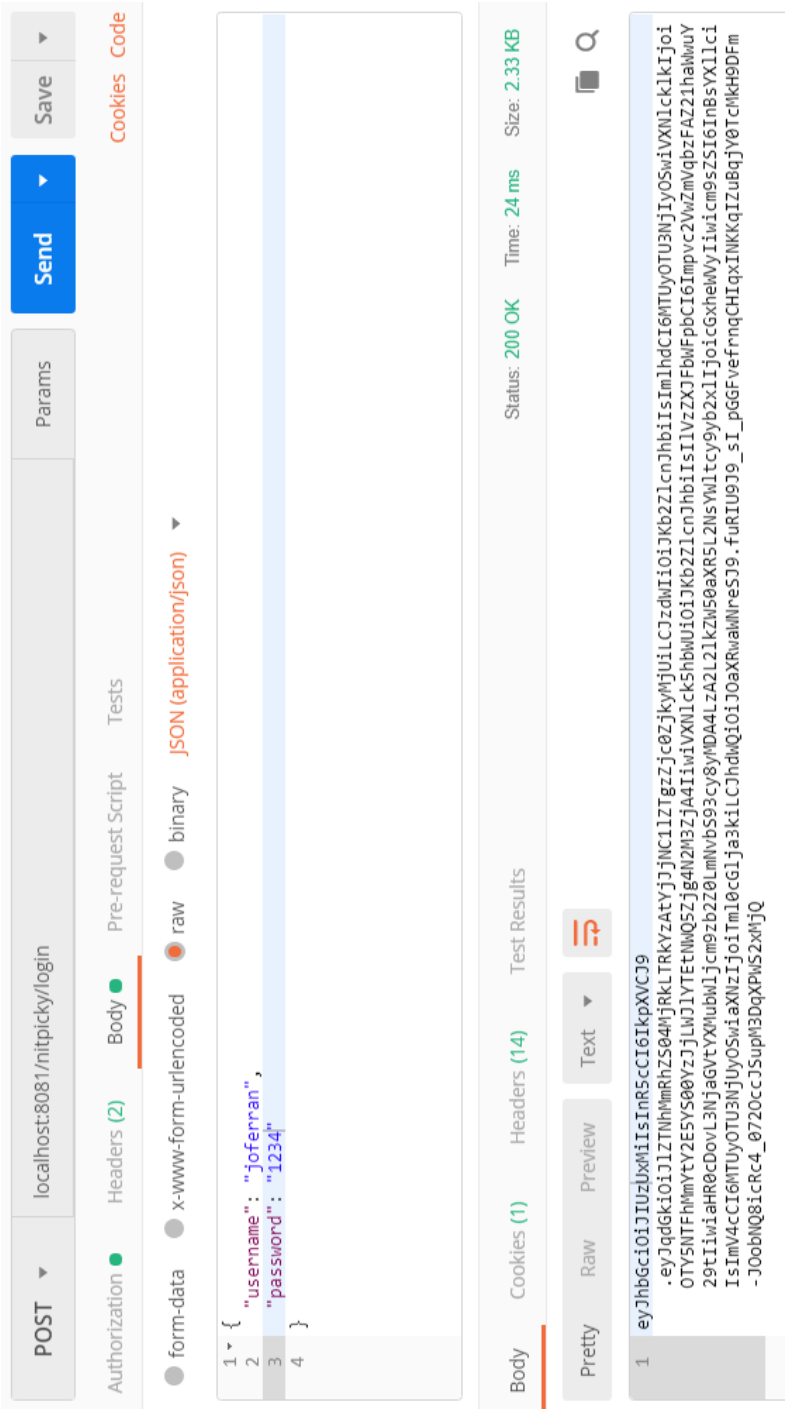
[19] Postgres [En línea] [fecha de consulta 15 de junio de 2018]. Disponible en: https://www.postgresql.org/

10. Anexos

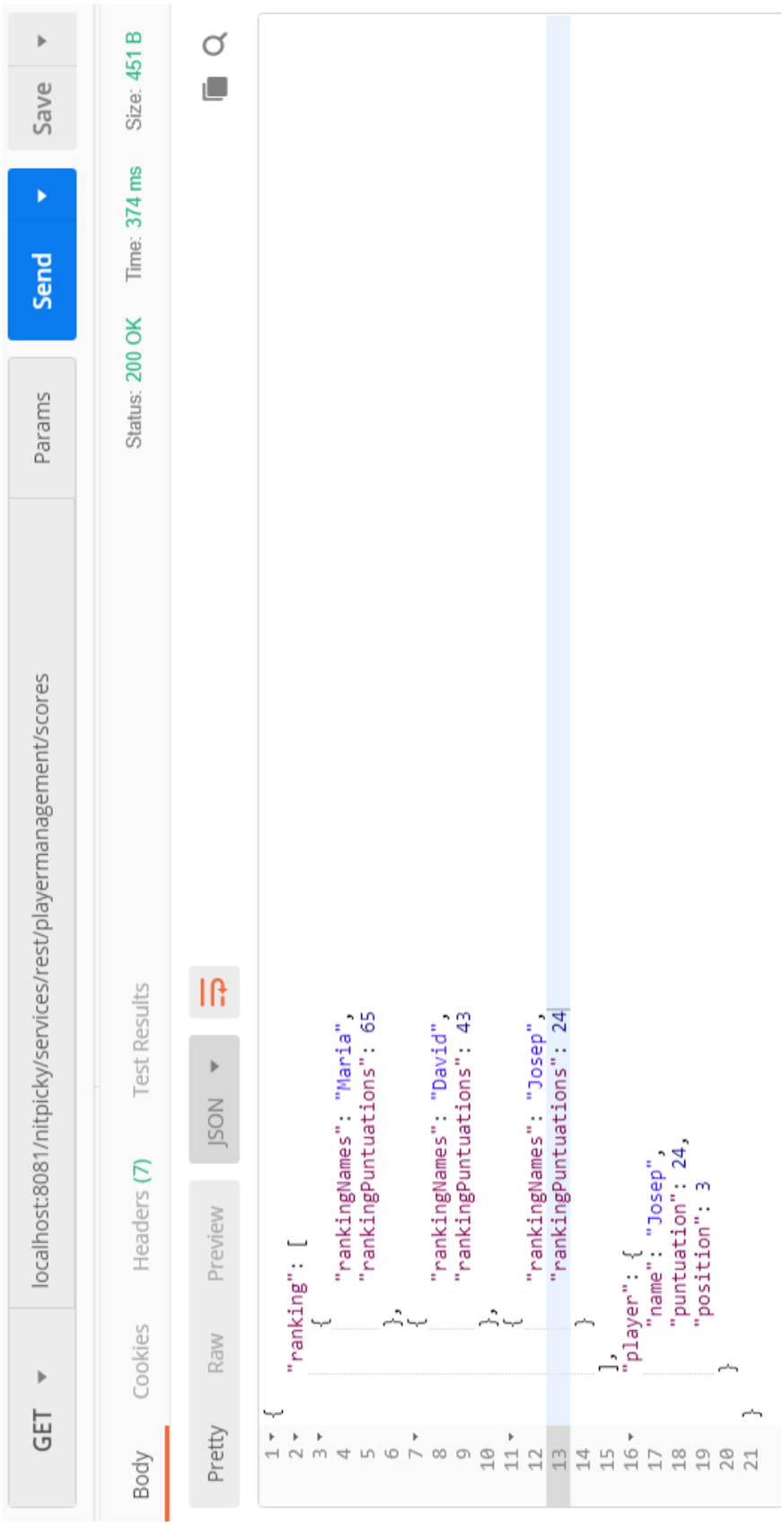
10.1 Anexo 1 Pruebas unitarias

ROL JUGADOR:

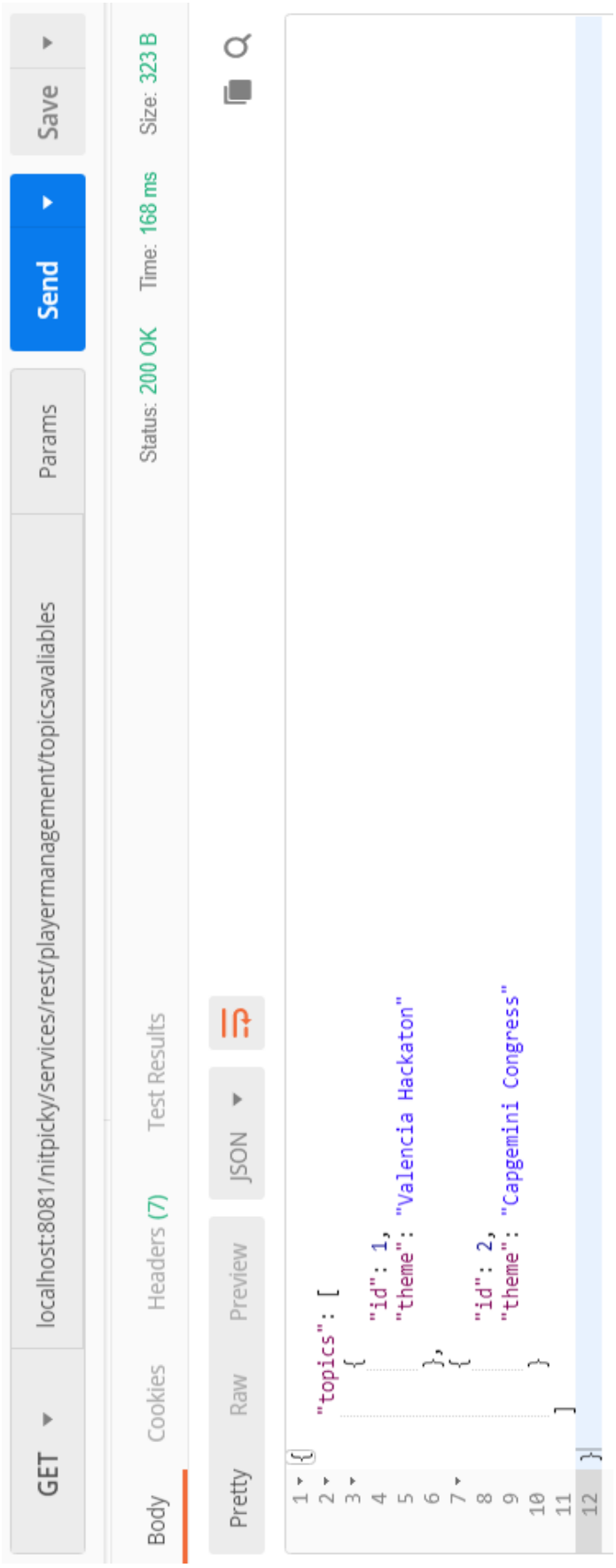
Inicio sesión del jugador



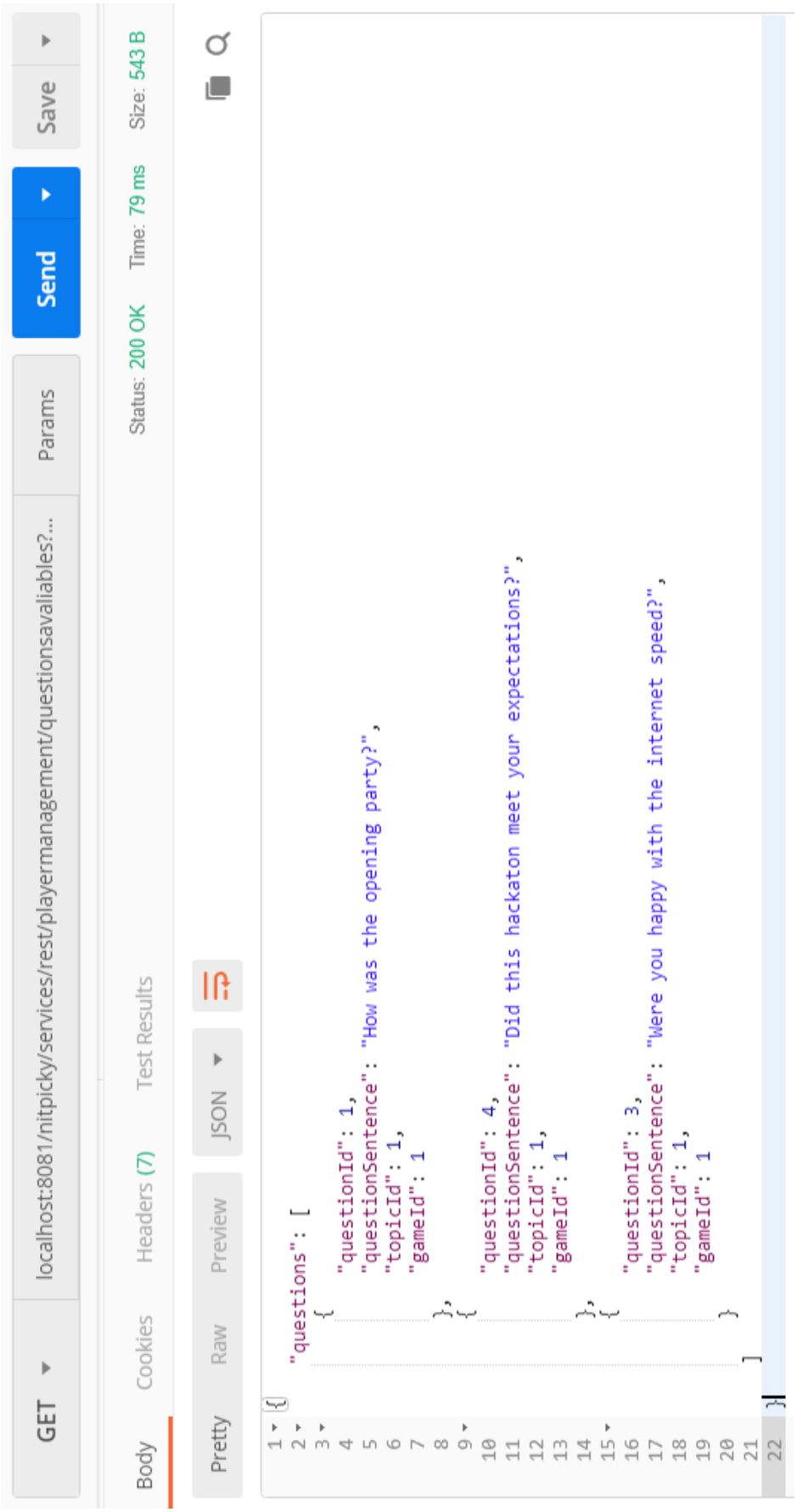
Obtener la lista de los mejores jugadores junto con su puntuación y posición en el ranking



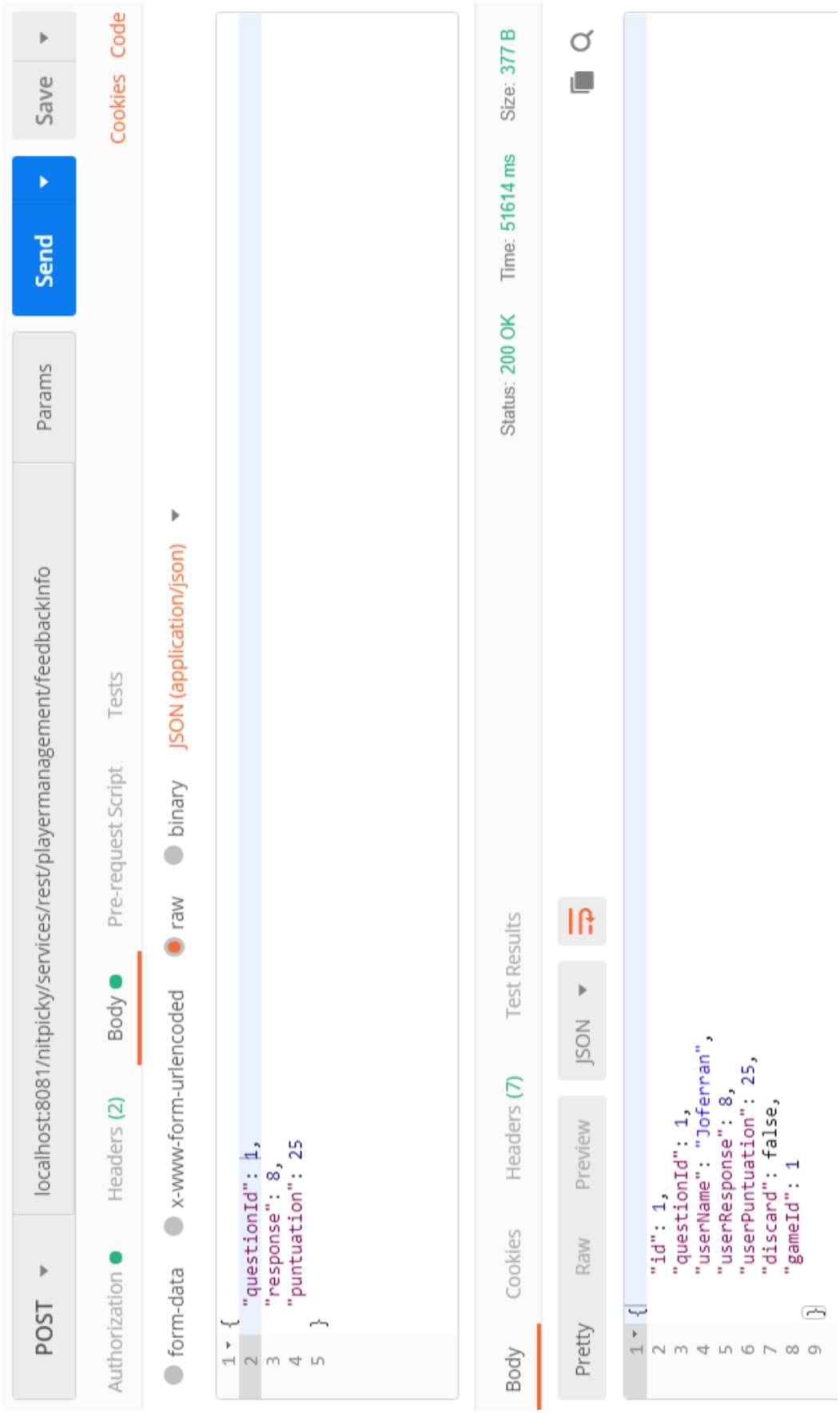
Obtener todos los temas disponibles



Obtener todas las preguntas disponibles de un tema concreto



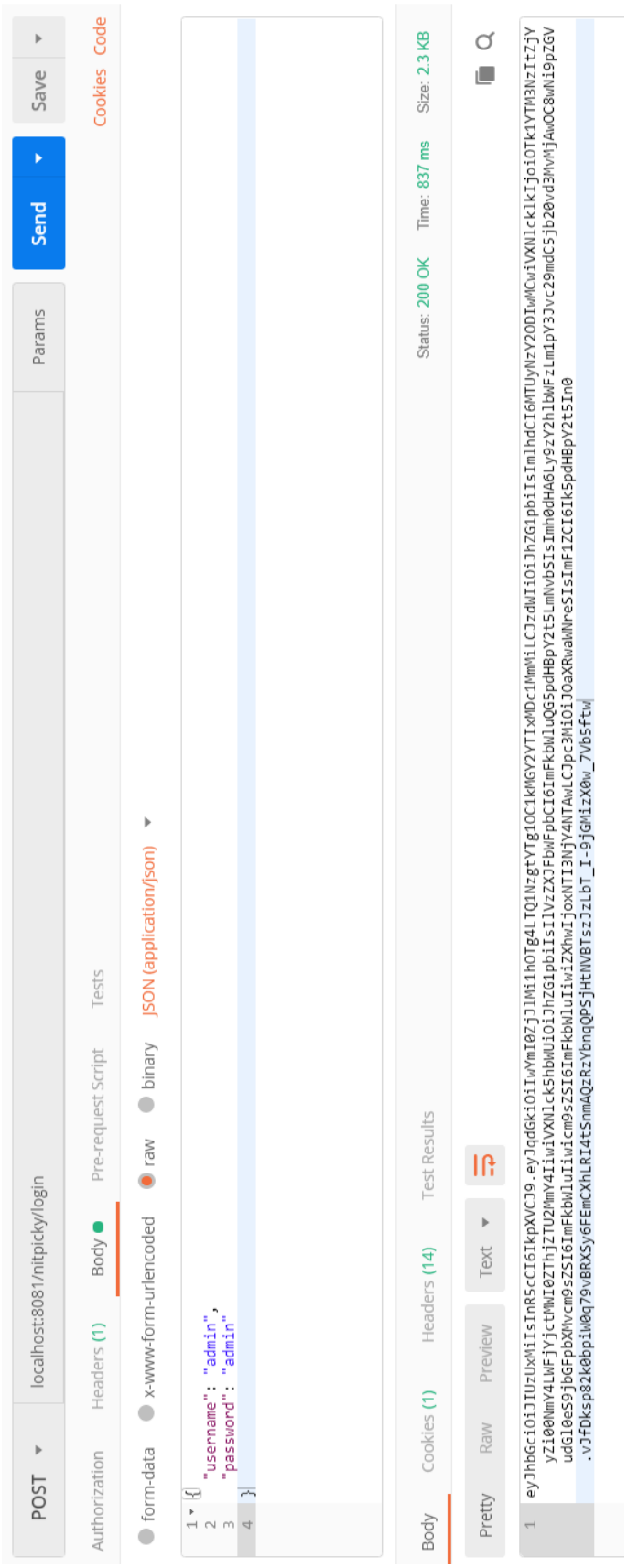
Guardar el resultado de una pregunta contestada



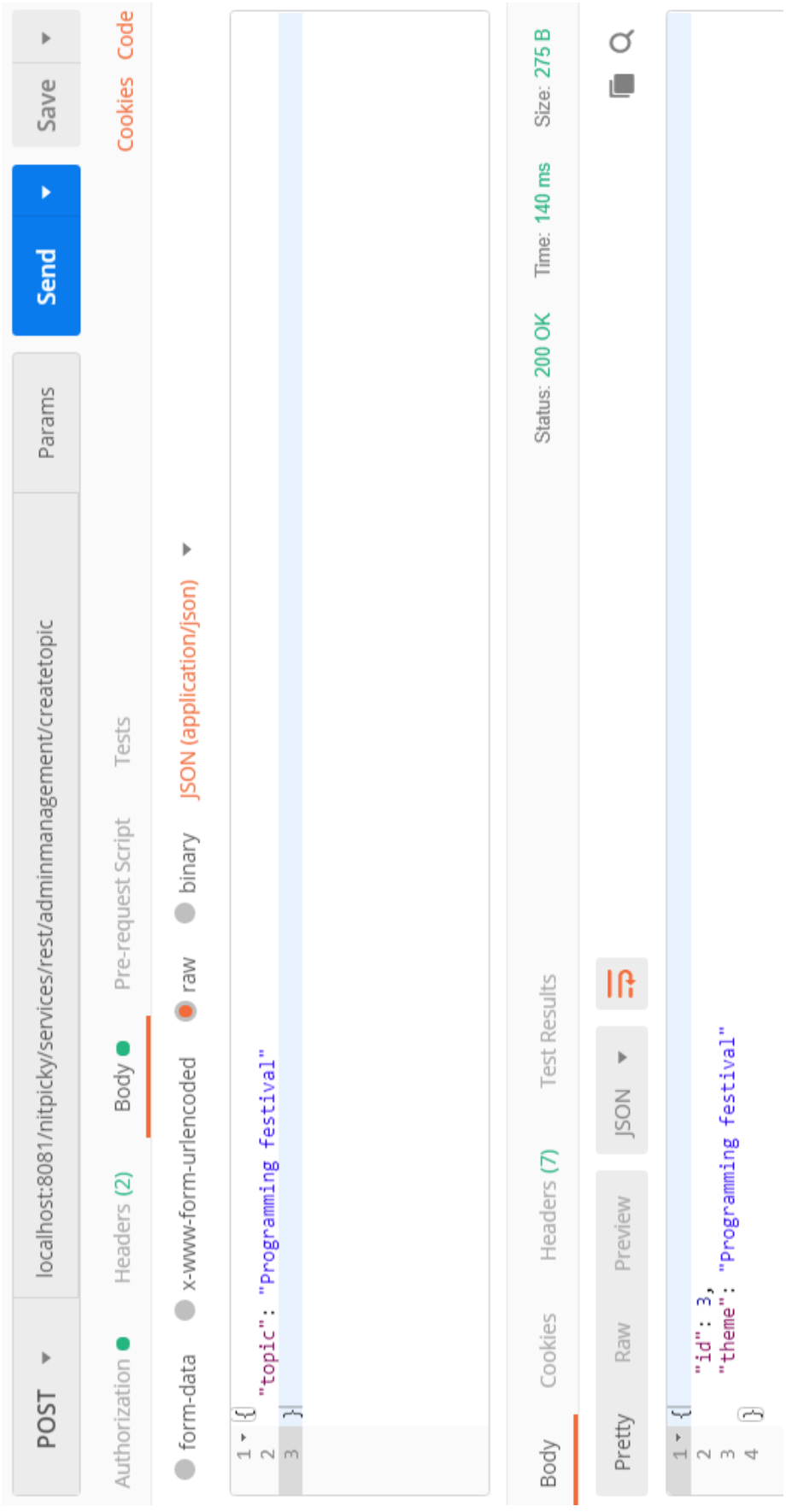
Descartar una pregunta determinada

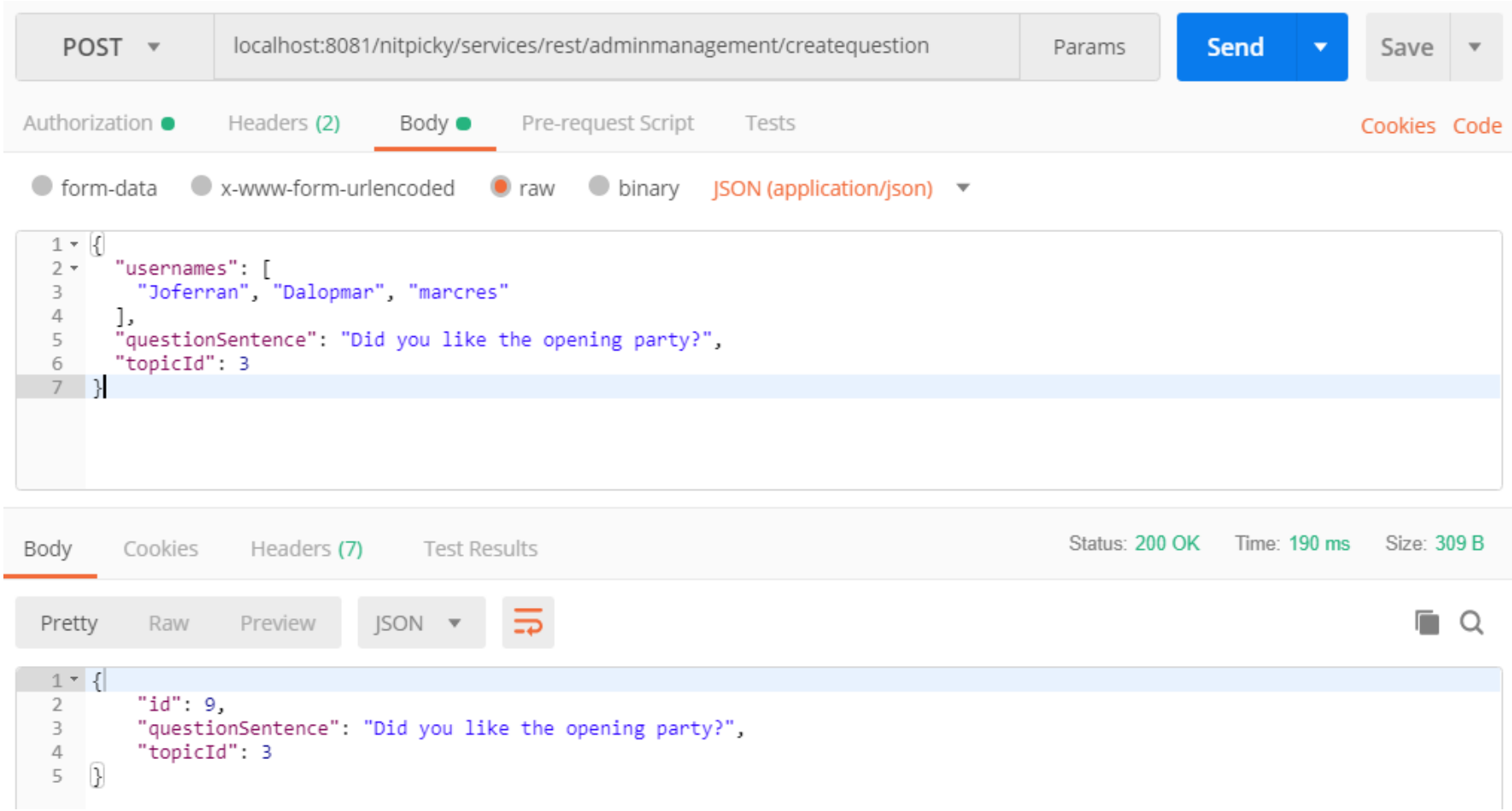
ROL ADMINISTRADOR:

Inicio sesión del administrador

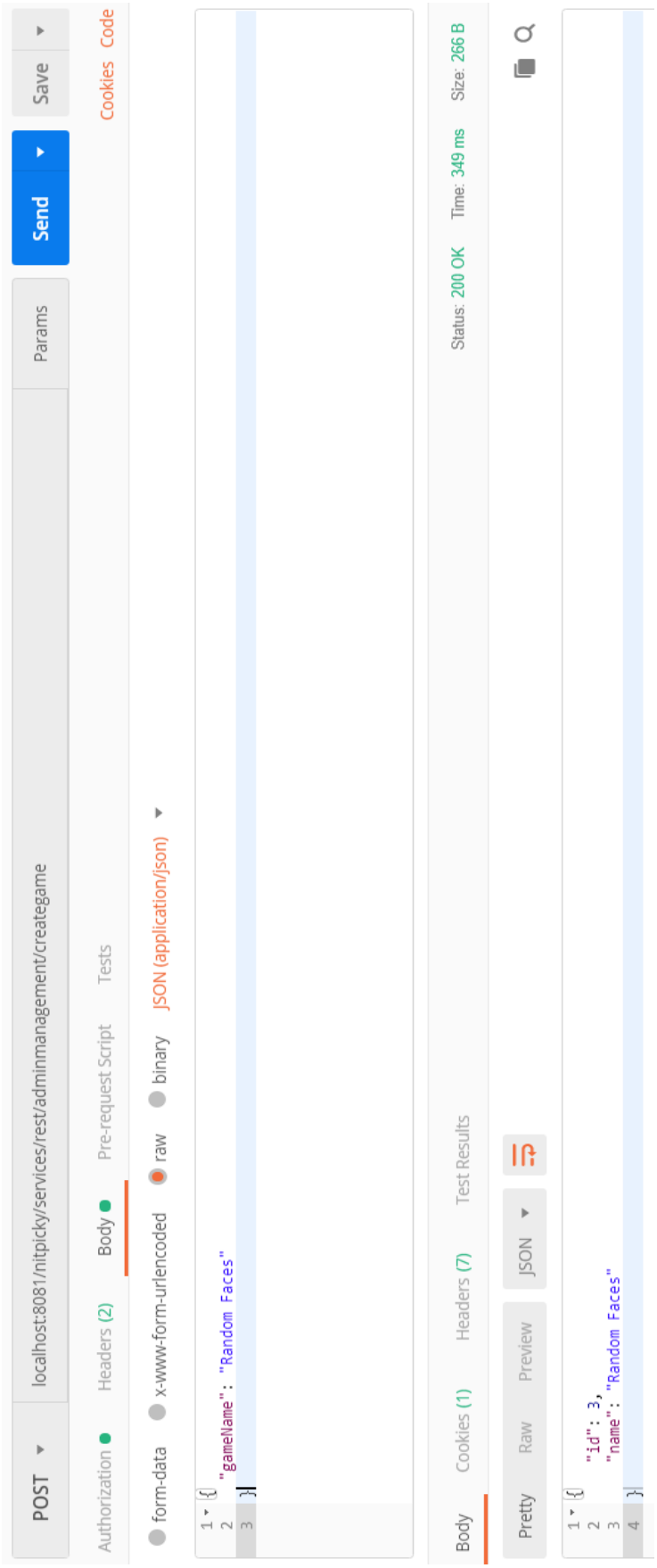


Crear tema

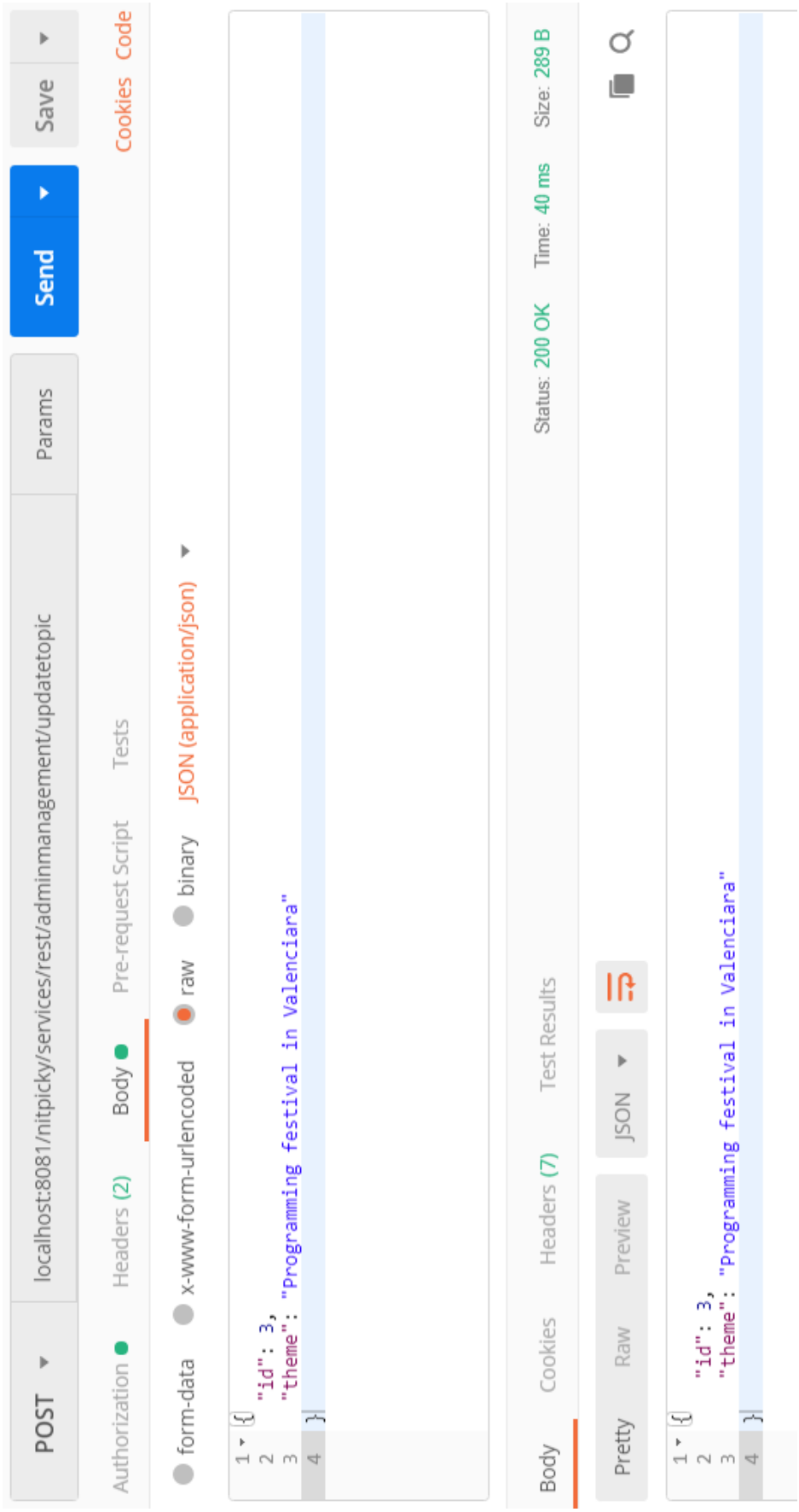

POST
localhost:8081/nitpicky/services/rest/adminmanagement/createtopic
Params
Send
Save
Authorization
Headers (2)
Body
Pre-request Script
Tests
Cookies
Code
form-data
x-www-form-urlencoded
raw
binary
JSON (application/json)
{
"topic": "Programming festival"
}
Status: 200 OK
Time: 140 ms
Size: 275 B
Body
Cookies
Headers (7)
Test Results
Pretty
Raw
Preview
JSON
{
"id": 3,
"theme": "Programming festival"
}


POST
localhost:8081/nitpicky/services/rest/adminmanagement/createquestion
Params
Send
Save
Authorization
Headers (2)
Body
Pre-request Script
Tests
Cookies
Code
form-data
x-www-form-urlencoded
raw
binary
JSON (application/json)
{
"usernames": [
"Joferran", "Dalopmar", "marcres"
],
"questionSentence": "Did you like the opening party?",
"topicId": 3
}
Body
Cookies
Headers (7)
Test Results
Status: 200 OK
Time: 190 ms
Size: 309 B
Pretty
Raw
Preview
JSON
{
"id": 9,
"questionSentence": "Did you like the opening party?",
"topicId": 3
}

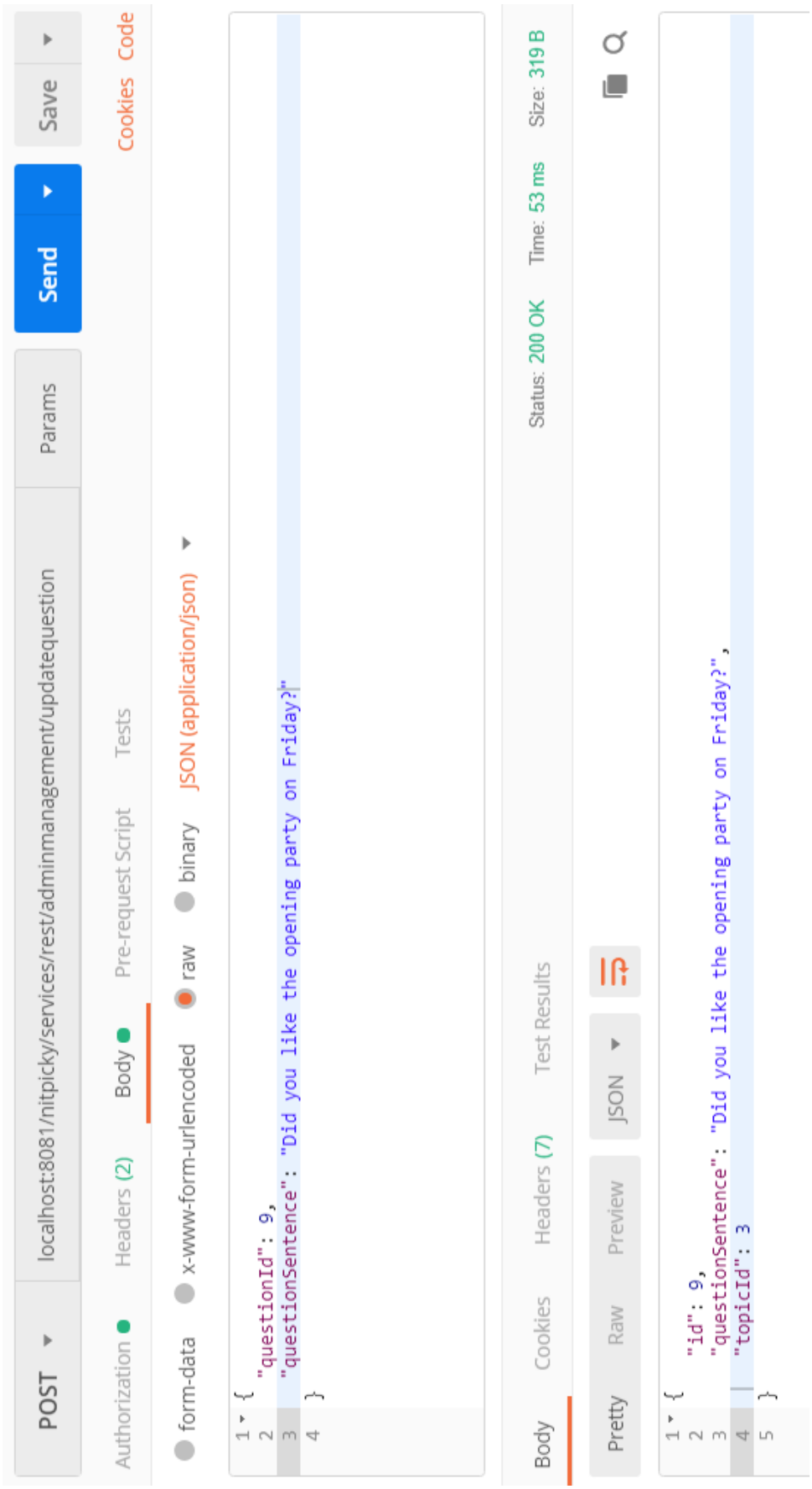
Crear juego


POST
localhost:8081/nitpicky/services/rest/adminmanagement/creategame
Params
Send
Save
Cookies
Code
Authorization
Headers (2)
Body
Pre-request Script
Tests
form-data
x-www-form-urlencoded
raw
binary
JSON (application/json)
{
"gameName": "Random Faces"
}
Status: 200 OK
Time: 349 ms
Size: 266 B
Body
Cookies (1)
Headers (7)
Test Results
Pretty
Raw
Preview
JSON
{
"id": 3,
"name": "Random Faces"
}

Modificar enunciado de un tema



Modificar enunciado de una pregunta

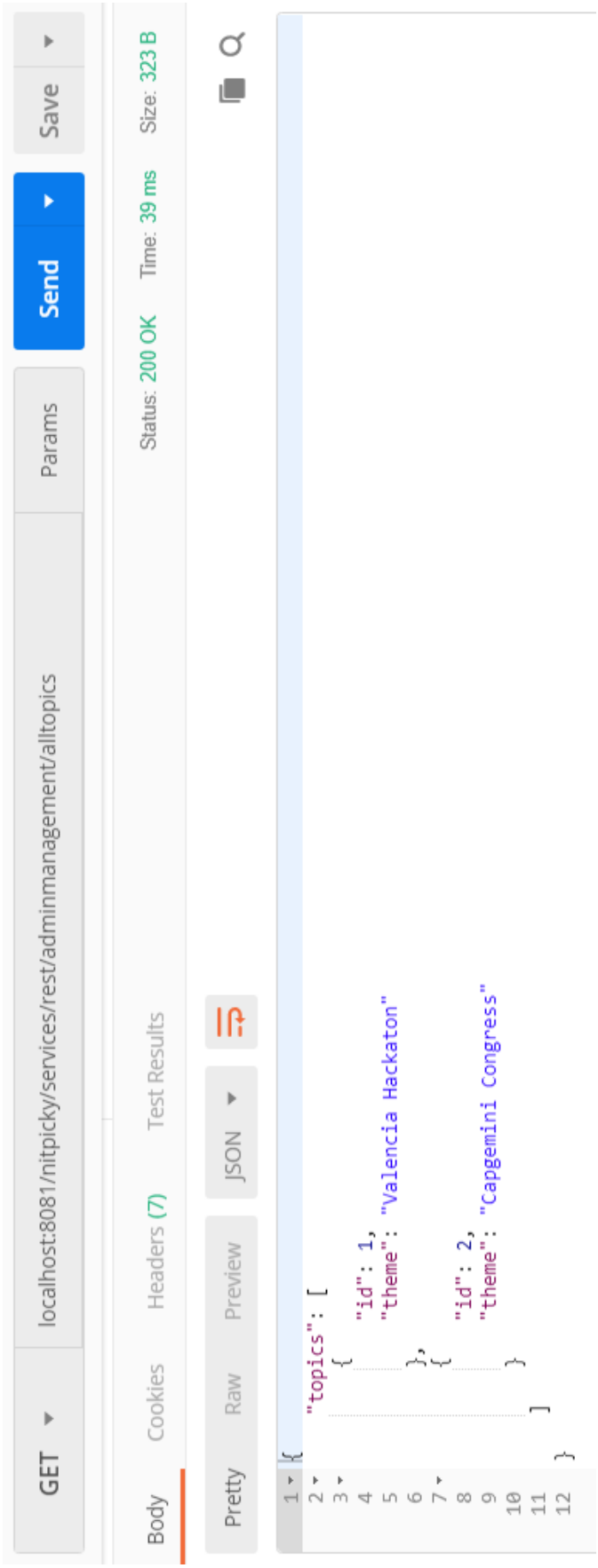


Eliminar un tema

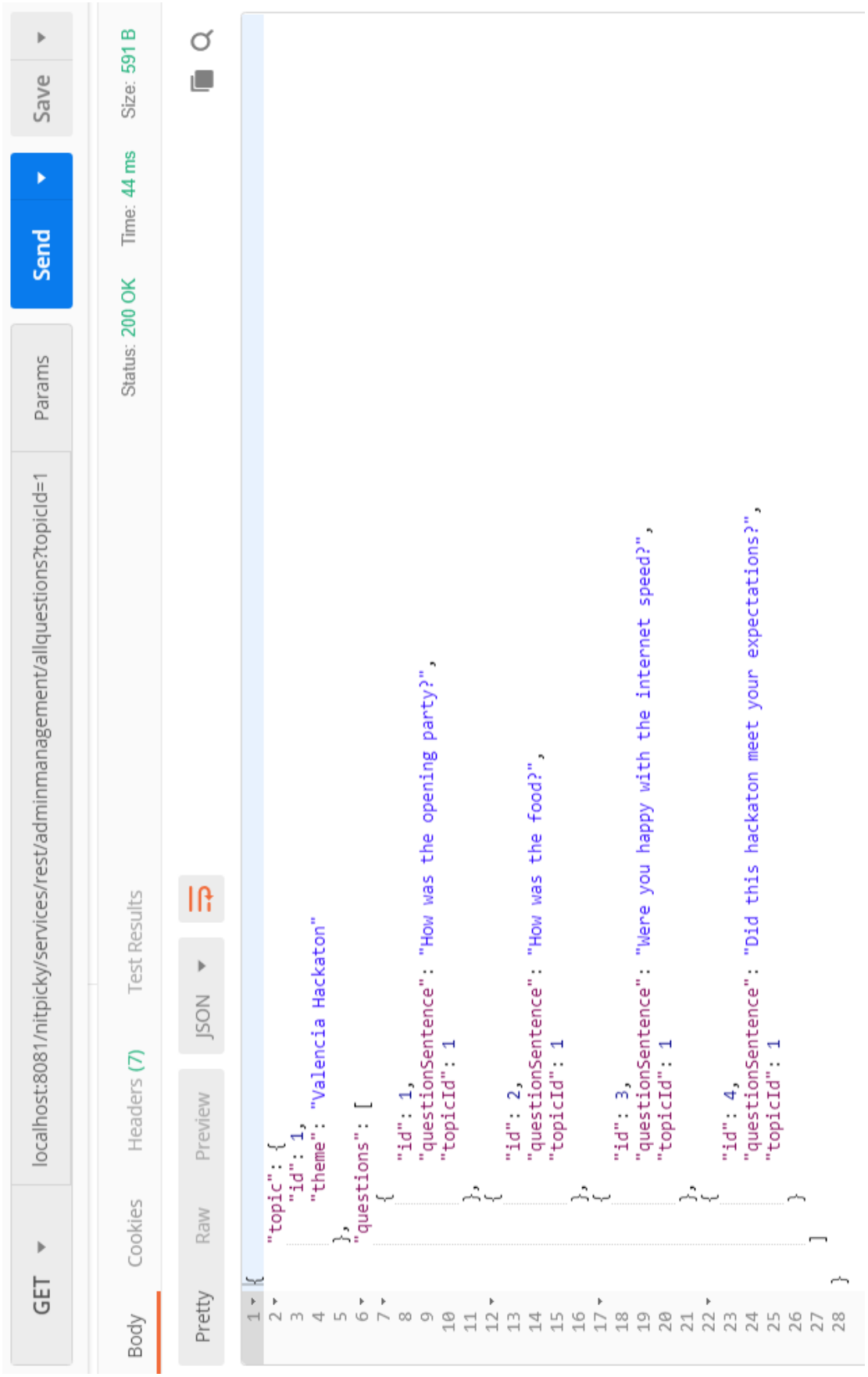
Eliminar una pregunta

Eliminar juego

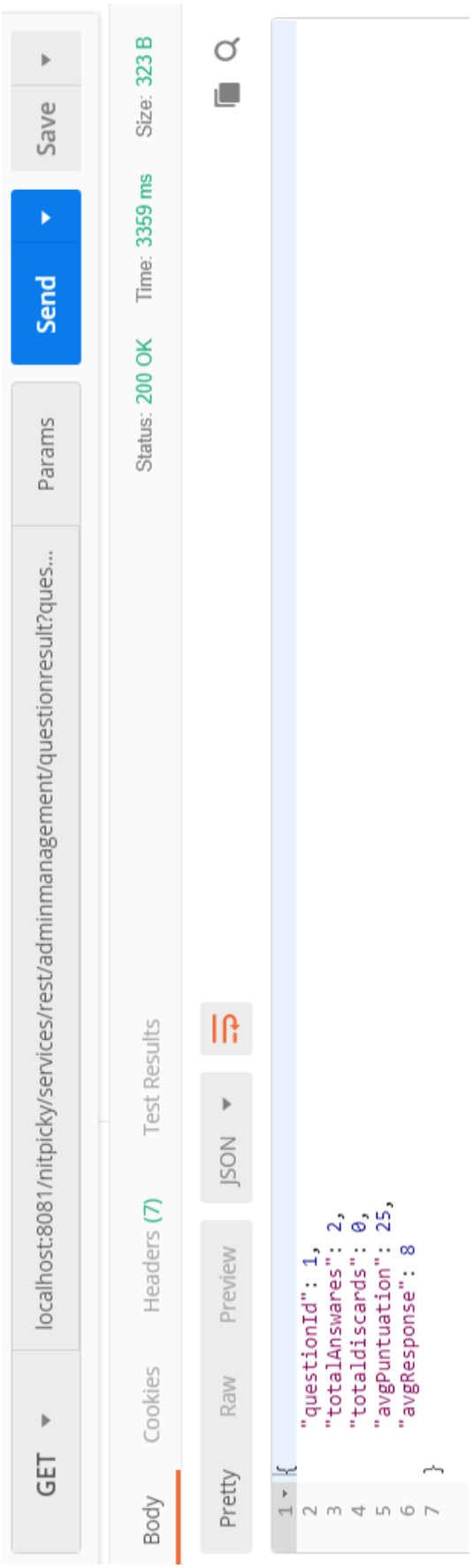
Obtener todos los temas del sistema



Obtener todas las preguntas de un tema concreto



Obtener los resultados de una pregunta concreta



Obtener los resultados de un tema concreto

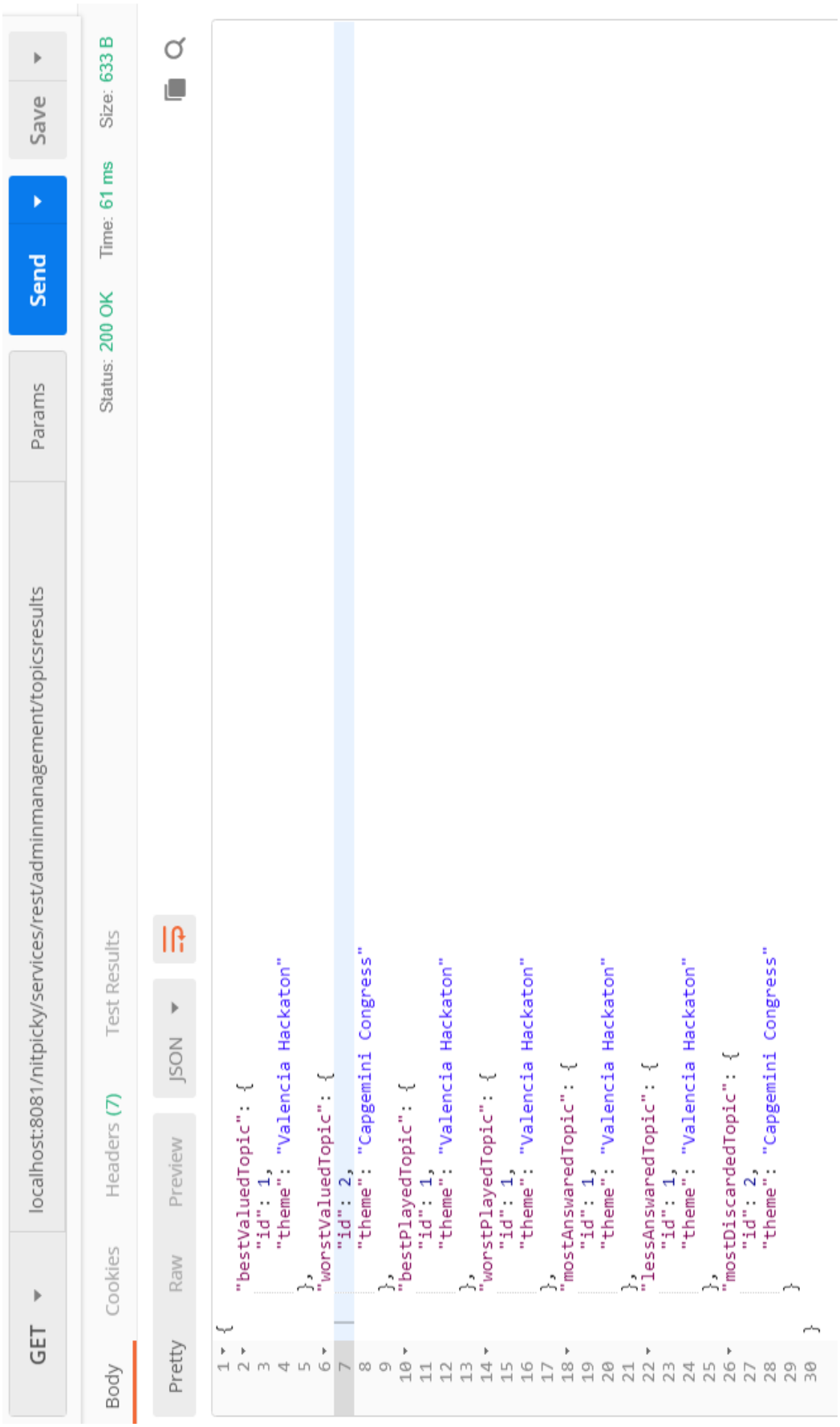
GET | localhost:8081/nitpicky/services/rest/adminmanagement/topicresult?topicId=1 | Params | Send | Save

Body | Cookies | Headers (7) | Test Results | Status: 200 OK | Time: 165 ms | Size: 1.01 KB

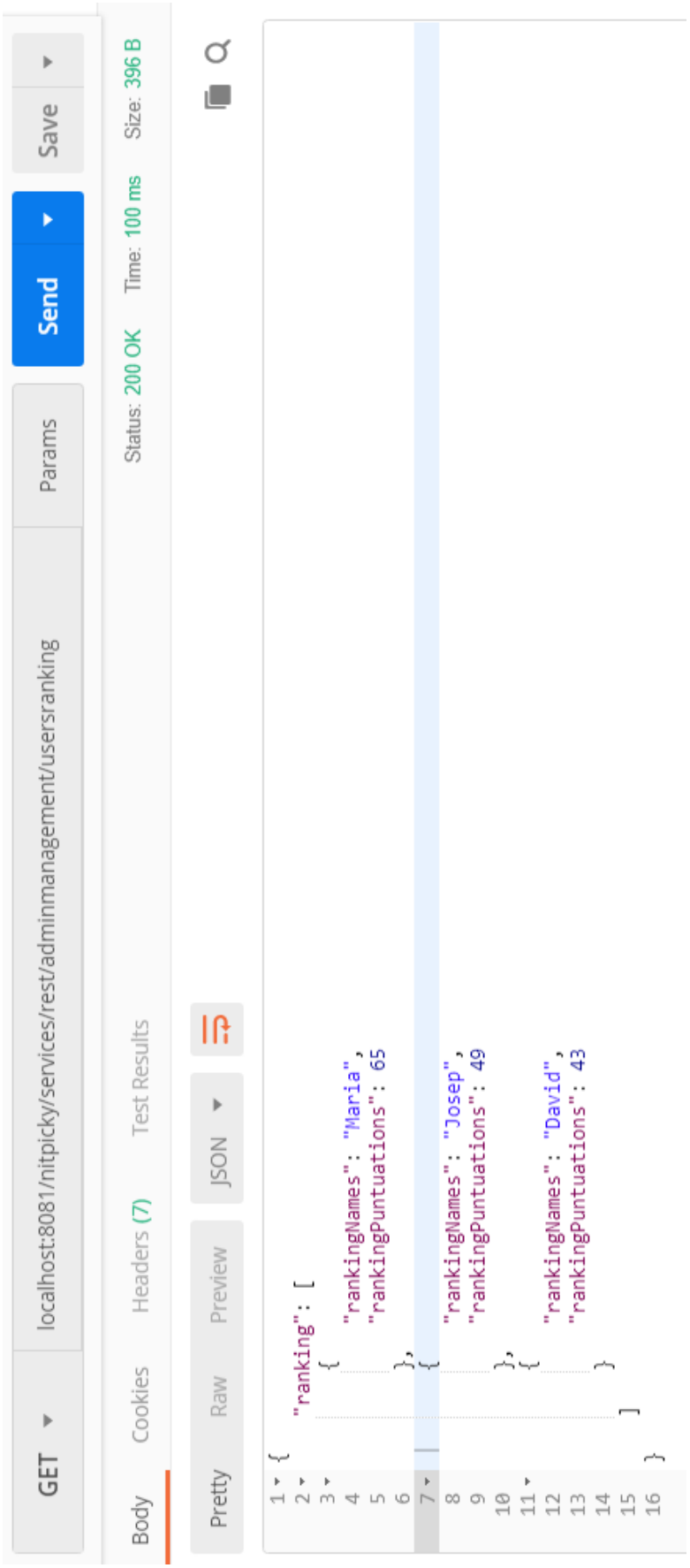
Pretty | Raw | Preview | JSON


```
        "topicId": 1,
        "totalAnswares": 4,
        "totaldiscards": 1,
        "avgPuntuation": 22,
        "avgResponse": 8
    },
    "bestQuestionValued": {
        "questionId": 1,
        "questionSentence": "How was the opening party?",
        "value": 8
    },
    "worstQuestionValued": {
        "questionId": 1,
        "questionSentence": "How was the opening party?",
        "value": 8
    },
    "bestQuestionPlayed": {
        "questionId": 1,
        "questionSentence": "How was the opening party?",
        "value": 25
    },
    "worstQuestionPlayed": {
        "questionId": 2,
        "questionSentence": "How was the food?",
        "value": 20
    },
    "mostQuestionAnswered": {
        "questionId": 1,
        "questionSentence": "How was the opening party?",
        "value": 2
    },
    "lessQuestionAnswered": {
        "questionId": 3,
        "questionSentence": "Were you happy with the internet speed?",
        "value": 0
    },
    "mostQuestiondiscarded": {
        "questionId": 4,
        "questionSentence": "Did this hackaton meet your expectations?",
        "value": 1
```

Obtener los resultados de todos los temas

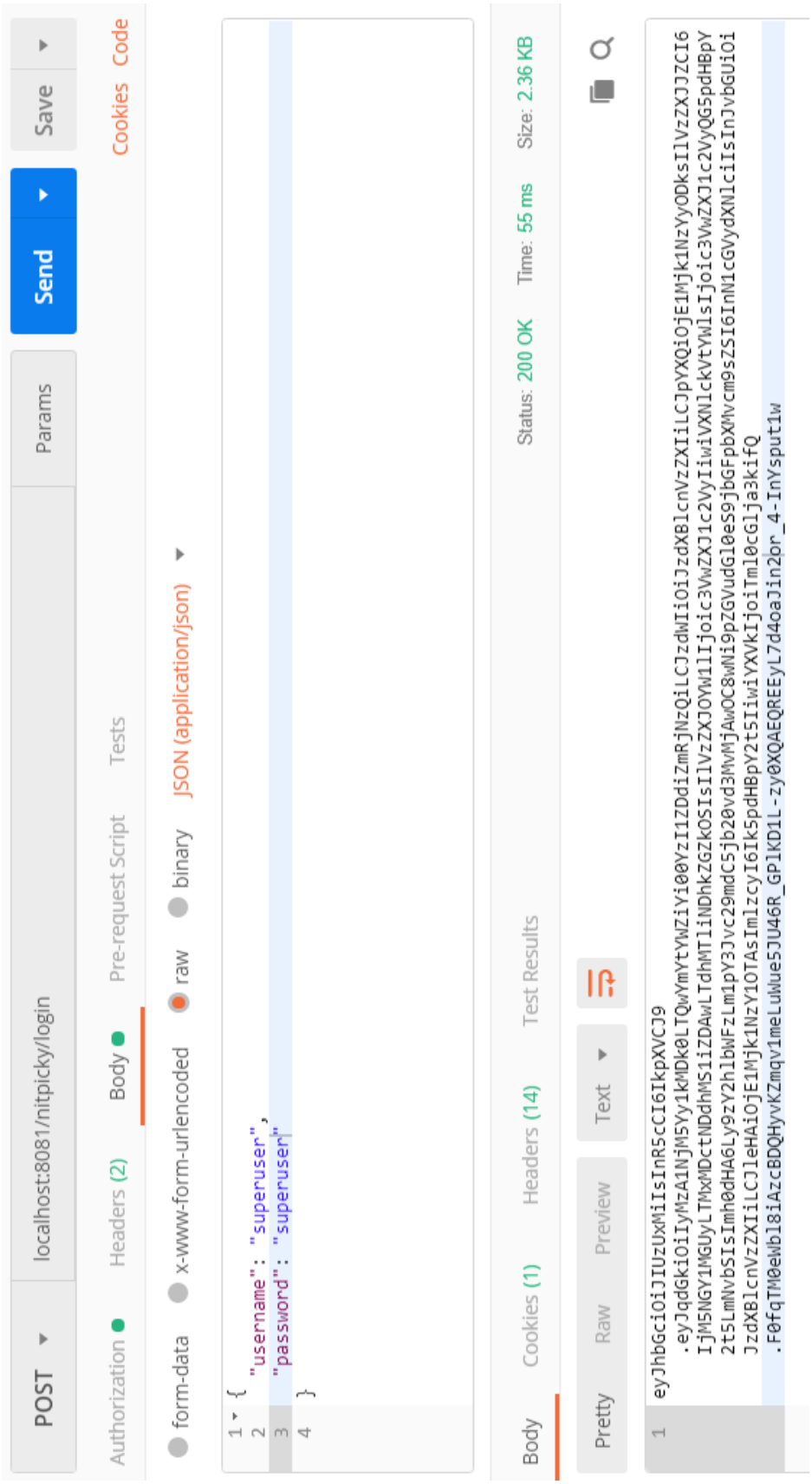


Obtener los tres mejores jugadores



ROL SUPER ADMINISTRADOR:

Inicio sesión super usuario



Modificar rol de jugador a administrador

Modificar rol de administrador a usuario