ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

DESARROLLO DE UN SISTEMA PARA LA INTERACCIÓN DE LOS MIEMBROS DE LA COMUNIDAD DE LA IGLESIA IFGF

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE TECNÓLOGO SUPERIOR EN DESARROLLO DE SOFTWARE

MICHAEL ANDRIK GUANOLUISA QUIROZ

DIRECTOR: ING. BYRON GUSTAVO LOARTE CAJAMARCA, MSc.

DMQ, febrero 2022

CERTIFICACIONES

Yo, Michael Andrik Guanoluisa Quiroz declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

MICHAEL GUANOLUISA

michael.guanoluisa@epn.edu.ec

michaelandrik10@gmail.com

Certifico que el presente trabajo de integración curricular fue desarrollado por Michael Andrik Guanoluisa Quiroz, bajo mi supervisión.

ING. BYRON LOARTE, MSc.
DIRECTOR

byron.loarteb@epn.edu.ec

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

GUANOLUISA QUIROZ MICHAEL ANDRIK

DEDICATORIA

Este trabajo está dedicado primero a Dios, que sin él no podría haberlo hecho, segundo a mi mamá que siempre me brindó su apoyo incondicional, fue la fortaleza que necesité en todo momento, mujer valiente y honorable que me enseña a ser mejor persona cada día.

MICHAEL ANDRIK GUANOLUISA QUIROZ

AGRADECIMIENTO

Agradezco primero a Dios, por siempre guiarme, mostrarme su amor y ser lo principal en mi vida. También a mi mamá, que siempre me apoya en todo momento, ella me mostró que con esfuerzo se llegan a conseguir las cosas por eso le agradezco a Dios por tener en mi vida a alguien como ella. Mujer que se esforzó hasta este momento para que yo pudiera estudiar, sin duda es lo más valioso que tengo en mi vida y es un agradecimiento infinitamente.

Agradezco a todos los profesores de la Escuela Politécnica Nacional, quienes me han enseñado y me han motivado a aprender de la carrera que escogí, todos han sido una muestra para mí de que escogí la mejor carrera, me encanta aprender de ellos y les agradezco por tener esa actitud de buena enseñanza con los estudiantes. También de forma especial agradecer al Ing. Byron Loarte quien me ha guiado no solo en este proyecto y en la culminación de este trabajo de titulación sino también como un amigo que en cualquier momento lo necesite, así como la Ing. Luz Marina quien también me ha guiado en la culminación de este trabajo de titulación.

MICHAEL ANDRIK GUANOLUISA QUIROZ

ÍNDICE DE CONTENIDO

RESUMEN

Actualmente, la pandemia del COVID-19 ha afectado en gran medida al Ecuador, siendo esto en lo sanitario, social, económico y político. Ante las dificultades actuales se ha impedido la realización de todo tipo de actividades económicas y sociales en diferentes grados. Un claro ejemplo de ello son las Iglesias, las cuales se han visto afectadas por estos cambios y como organismos sociales cuya misión se realiza a través del contacto y la interacción. El espacio virtual a través del uso de Internet y medios informáticos se han convertido en algunas de las herramientas más apropiadas, para encontrar alternativas y dar continuidad a la vida ministerial de la Iglesia.

Para poder ayudar a la comunidad cristiana en el presente trabajo se ha desarrollado un *backend* para la Iglesia Fraternidad Internacional del Evangelio Completo GISI (IFGF) ubicada en la ciudad de Quito, permitiendo de esta manera la interacción de los miembros de la Iglesia con la comunidad cristiana por medio de vídeos, eventos, artículos, comentarios y noticias que sean relevantes para la comunidad, con el objetivo de aprovechar el espacio virtual y mejorar el desarrollo de las actividades por medio de la tecnología.

El presente documento se encuentra estructurado de la siguiente manera: en la primera parte se específica los antecedentes, objetivos, alcance y el respectivo marco metodológico de este proyecto. En la segunda parte se especifica cómo se ha implementado la metodología ágil *Scrum*, modelo de la base de datos, diseño de la arquitectura y herramientas en el desarrollo del *backend*. En la tercera parte se especifican cada una de las actividades que han sido ejecutadas por cada iteración, el resultado que se ha obtenido, conclusiones y recomendaciones que se han obtenido en el desarrollo de este trabajo de integración.

PALABRAS CLAVE: COVID-19, *Scrum, Node, Express, MongoDB*, API *RESTful*.

ABSTRACT

Currently, the COVID-19 pandemic has affected Ecuador to a great extent, this being health, social, economic, and political. Given the current difficulties, all kinds of economic and social activities have been prevented to varying degrees. A clear example of this is the Churches, which have been affected by these changes and as social organizations whose mission is carried out through contact and interaction. The virtual space through the use of the Internet and computer media have become some of the most useful tools to find alternatives and give continuity to the ministerial life of the Church.

In order to help the Christian community in the present work, a backend has been developed for the “Iglesia Fraternidad Internacional del Evangelio Completo GISI (IFGF)” located in the city of Quito, thus allowing the interaction of the members of the Church with the community. Christian through vídeos, events, articles, comments, and news that are relevant to the community, with the aim of taking advantage of the virtual space and improving the development of activities through technology.

This report is structured as follows: the first part specifies the background, objectives, scope, and the respective methodological framework of this project. The second part specifies how the agile Scrum methodology, database model, architecture design and tools have been implemented in the development of the backend. The third part specifies each of the activities that have been executed by each iteration, the result that has been obtained, conclusions and recommendations that have been obtained in the development of this curricular integration work.

KEYWORDS: COVID-19, *Scrum*, *Node*, *Express*, *MongoDB*, API *RESTful*.

1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

Las Iglesias siendo organismos sociales donde la mayor parte de sus actividades son realizadas de forma presencial debido al COVID-19, se han visto afectadas por las medidas tomadas por el gobierno del Ecuador. Según el acuerdo Interministerial N°. 00010 el aforo permitido para las actividades presenciales en las Iglesias de todo el país es del 30% de su capacidad, esto último ha llevado a que las Iglesias implementen nuevas estrategias que eviten las actividades presenciales y un contagio masivo ya que la pandemia aún no ha terminado [1]. Ante las dificultades actuales las Iglesias han optado por nuevas formas de llevar sus actividades de forma diferente a como normalmente eran realizadas. Por otra parte, cumpliendo con los acuerdos del Comité de Operaciones de Emergencia (COE), las medidas de prevención solicitadas, entre otras; encontraron en la tecnología una alternativa para poder continuar con sus servicios ministeriales, como lo menciona el Sr. Walter Hidalgo pastor de la Iglesia IFGF [2].

En la actualidad, es inevitable encontrar múltiples sistemas *web* o aplicaciones móviles interconectadas entre sí que ayuden en las tareas cotidianas de las personas. De hecho, múltiples aplicaciones como *Twitter*, *YouTube*, *Facebook*, *Uber*, entre otras; hacen uso de varios servidores para el desarrollo de un *backend* que les permita brindar y consumir una serie de API's a través de varios *endpoints*. Permitiendo de esta manera un crecimiento horizontal en las empresas y que varias aplicaciones se puedan comunicar de una manera mucho más organizada y eficiente [3].

Por lo citado anteriormente, en el presente proyecto de integración curricular se ha desarrollado un *backend* utilizando el lenguaje de programación *JavaScript* para la Iglesia Cristiana (IFGF) ubicada en la ciudad de Quito, permitiendo de esta manera crear una serie de *endpoints* públicos y protegidos para que cualquier aplicación del lado del cliente o aplicación móvil pueda consumir esta información sin importar el lenguaje de programación, Tecnología o *Framework*. Logrando de esta manera la interacción de los miembros de la Iglesia con la comunidad cristiana por medio de vídeos, eventos, artículos, comentarios y noticias que sean relevantes para la comunidad, con el objetivo de aprovechar el espacio virtual y mejorar el desarrollo de las actividades por medio de la tecnología.

1.1 Objetivo general

Desarrollar un sistema para la interacción de los miembros de la comunidad de la Iglesia (IFGF).

1.2 Objetivos específicos

1. Determinar los requerimientos funcionales y no funcionales para el *backend*.
2. Diseñar el modelo de la Base de datos y arquitectura *REST* para el *backend* en base a los requerimientos.
3. Codificar el *backend* en base al lenguaje de programación y requerimientos.
4. Verificar el funcionamiento del *backend* y el despliegue a producción.

1.3 Alcance

Las últimas tendencias tecnológicas, promueven el uso y creación de API's ya que permiten obtener una serie de ventajas no solo dentro de las empresas si no la escalabilidad para la integración y el consumo de servicios de otras aplicaciones. Es por esta razón que grandes empresas no solo de tecnología disponen de una serie de API's que sean públicas y/o protegidas para que la comunicación y transferencia de datos sea mucho más fácil y segura; sin importar el lenguaje de programación [4].

Con el objetivo de implementar nuevas estrategias tecnológicas y gracias a las ventajas que ofrecen las API's, en este trabajo se ha desarrollado un *backend* dirigido a los miembros de la Iglesia Cristiana (IFGF) de la ciudad de Quito, con el fin de evitar los contagios masivos y solucionar algunos problemas que en sus actividades presenciales se han generado debido a la pandemia.

Por último, sin dejar de lado la integridad, consistencia y seguridad de los datos con el *frontend* y base de datos. El *backend* se ha desarrollado utilizando una arquitectura *REST*, proporcionando una serie de *endpoints* que se describen a continuación:

Información general

- Creación de varios *endpoints* para presentar toda la información de la Iglesia (noticias, mensajes bíblicos y álbum de fotos).
- Creación de varios *endpoints* para gestionar la información general (noticias, mensajes bíblicos y álbum de fotos).

Donaciones

- Creación de un *endpoint* para registrar donación.
- Creación de un *endpoint* para ver el registro de donaciones.

Eventos

- Creación de varios *endpoints* para gestionar eventos.
- Creación de un *endpoint* para inscribirse en un evento.
- Creación de un *endpoint* protegido para visualizar las personas inscritas en los eventos.

Vídeos

- Creación de varios *endpoints* para gestionar vídeos.
- Creación de un *endpoint* para visualizar los vídeos publicados.

Juegos

- Creación de un *endpoint* para registrar cuestionarios.
- Creación de un *endpoint* para visualizar cuestionarios.
- Creación de un *endpoint* para registrar datos del jugador.
- Creación de un *endpoint* para visualizar los puntajes del juego.

1.4 Marco Teórico

El *backend* es el medio que permite realizar una conexión entre una aplicación por el lado del cliente y la Base de datos, a esto se llama conexión por el lado del servidor. El *backend* debe ser desarrollado con un proceso ordenado ya que es el que mantiene la lógica del negocio, la seguridad y la integridad de los datos [5].

Las últimas tendencias tecnológicas, promueven el uso y creación de API's ya que permiten obtener una serie de ventajas no solo dentro de la empresa si no la escalabilidad para la integración y el consumo de servicios de otras aplicaciones. Es por ello por lo que, grandes empresas disponen de una serie de API's que sean públicas o protegidas logrando que la comunicación y transferencia de datos sea mucho más fácil y segura sin importar el lenguaje de programación [6].

API como sus siglas lo mencionan Interfaz de Programación de Aplicaciones, se describe como el grupo de conceptos y protocolos que se requieren para el proceso de desarrollo e implementación de un *software*, esto es lo que permite que se realice una comunicación entre aplicaciones. Definiendo a una API como un modelo de *software* que se establece

para la comunicación con otro en base a solicitudes y respuestas [7]. Además, existen 2 tipos de API's:

- **Privadas:** son internas, usadas por cierto grupo de personas pertenecientes a una organización, con acceso restringido.
- **Públicas:** son usados de forma libre con total acceso a estos.

Actualmente el uso de las API's es cada vez mayor, llegando a un punto donde todo se encuentre implementado bajo esta tecnología. Además, permite a las empresas comunicarse de una forma fácil y que el traslado de la información sea cada vez en tiempo real. Como otro punto a favor sobre el uso de las API's es que las mismas siempre mantendrán la información 100% segura, garantizando de esta manera la integridad de datos dentro de la comunicación que se disponga, sino también funcionalidad, eficiencia y sin dejar de lado la rapidez con la que se disponga la información, sin duda una característica esencial para una buena comunicación [8].

Una Base de datos no relacional (NoSQL) es un tipo de Base de datos la cual no contiene relaciones con otras tablas como si lo dispone una Base de datos relacional. En este sentido, las bases de datos NoSQL están formadas por colecciones las cuales almacenan los datos dentro de documentos [9].

MongoDB Atlas, es una plataforma que proporciona una serie de servicios para el alojamiento de datos. En ese sentido, para la gestión y el almacenamiento de la información se dispone de un *clúster* para poder configurar todo el Sistema Gestor de Base de Datos NoSQL logrando evitar tediosas configuraciones y a la vez disponer de un panel administrativo que es fácil de utilizar para los desarrolladores [10].

Node.js es un entorno de desarrollo basado en *JavaScript*, el cual permite ejecutar programas en el lado del servidor. Además, funciona en base a un patrón de solicitudes y respuestas permitiendo la creación de API's de tipo *RESTful*. La finalidad de Node.js es desarrollar sistemas basados en tiempo real de una forma mucho más eficiente y escalable [11].

Express es un *Framework* que permite crear aplicaciones del lado del servidor para la gestión de peticiones HTTP, contiene una serie de paquetes y librerías para una alta compatibilidad con *Node.js*. Además, al ser un *Framework* de código abierto cada vez se integran nuevas funcionalidades complejas, como por ejemplo el uso de *middlewares* para el manejo de inicio de sesión, autenticación, autorización, cabeceras de seguridad, etc. Siendo uno de los *Frameworks* preferidos en la actualidad por la comunidad de

desarrolladores para la integración y comunicación con aplicaciones del lado del cliente [12].

2 METODOLOGÍA

Un estudio de casos se entiende como una serie de sucesos a los cuales se analizan en base a una investigación o búsqueda, para esto el estudio de casos se define como la obtención de un conocimiento concreto sobre un tema a profundidad. Este tipo de investigación se diferencia de otros por ser cualitativa es decir, que su estudio o análisis no está dado por algo estadístico sino al estudio de un suceso donde su objetivo es crear hipótesis o teorías de los casos que pueden concretar en otras hipótesis nuevas [13].

Debido a esto, el presente trabajo de integración curricular mantiene un estudio de casos, ya que se parte de una investigación sobre la actual pandemia y la dificultad que esto conlleva en las actividades diarias. Todo esto, permite llevar a cabo el desarrollo de un *backend* para para que los miembros Iglesia IFGF puedan interactuar y comunicarse con toda la comunidad cristiana por medio de la tecnología.

2.1 Metodología de Desarrollo

Las metodologías de desarrollo de *software* son estructuras de trabajo que sirven para ordenar, planificar y dirigir el desarrollo de un producto de manera ordenada y colaborativa. Es así como, en los proyectos de *software* las metodologías permiten determinar tareas, roles, tiempos, iteraciones, entre otros desde un enfoque totalmente ágil.

Las metodologías ágiles dentro del desarrollo de *software* sirven para encapsular el proceso del desarrollo en una serie de roles y artefactos que permitan mostrar resultados inmediatos al cliente, logrando de esta manera que el proyecto sea más fácil de adaptarse a nuevos cambios que surjan sin dejar de lado la calidad del producto final [14]. En ese sentido, en las próximas secciones se describe como se ha implementado cada una de las fases de la metodología *Scrum* en el desarrollo del *backend*.

Roles

Los roles que se han definido para este proyecto son: *Product Owner, Scrum Master* y *Development Team.* Estos roles operan de forma colaborativa en todo el desarrollo logrando una participación más activa y organizada.

Product Owner

Este rol es el encargado de alinear las tareas que el equipo tiene en dirección de sus prioridades, también es el encargado de mostrar estas tareas al *Development Team* para la creación de nuevos productos [15]. En ese sentido, en la **TABLA I**: Asignación de roles se evidencia la persona encargada para este rol.

Scrum Máster

Este rol lo ocupa la persona que hace el papel de líder, el cual trabaja de la mano con el *Product Owner* para mantener y manejar las tareas previstas por este. Además, es el comunicador directo con el *Development Team* para planificar y supervisar cada una de las tareas que son ejecutadas en cada *Sprint* [15]. Por tal razón, en la **TABLA I**: Asignación de roles se evidencia la persona encargada para este rol.

Development Team

Este rol esta principalmente conformado por un grupo de personas que se encargan de cumplir o realizar cada una de las tareas proporcionadas por el *Product Owner*, por otra parte, son los encargados del avance del desarrollo por medio de entregables funcionales por cada *Sprint* previamente planificado [15]. Muestra de ello, en la **TABLA I** se evidencia a la persona encargada para este rol.

TABLA I: Asignación de roles

ROLES	NOMBRES
Product Owner	Pastor Walter Hidalgo
Scrum Master	Ing. Byron Loarte, MSc.
Development Team	Michael Guanoluisa

Artefactos

En la metodología *Scrum* se definen varios artefactos que se han utilizado para este proyecto. No obstante, estos artefactos permiten administrar el proceso de desarrollo y el manejo de la información en base al tiempo y entregables funcionales, dando como resultado el cumplimiento de los objetivos que se han plateado [15].

Recopilación de Requerimientos

Se describe como la actividad para definir las necesidades del cliente, las cuales son las que definen el alcance del proyecto [16]. A continuación, en la **TABLA II** se muestra el formato que se ha utilizado para la elaboración de la Recopilación de los requerimientos manteniendo de forma ordenada la información que se ha obtenido. Sin embargo, la tabla completa con toda la información se detalla en el **ANEXO II** del presente documento.

TABLA II: Formato para la Recopilación de requerimientos

RECOPILACIÓN DE REQUERIMIENTOS		
TIPO DEL SISTEMA	**ID - RR**	**ENUNCIADO DEL ÍTEM**
BACKEND	**RR001**	Como usuario invitado, administrador y/o miembro de la Iglesia, necesitan generar varios *endpoints* para presentar la información general de la Iglesia.
	RR002	Como usuario administrador y miembro de la Iglesia necesitan generar varios *endpoints* para poder iniciar y cerrar sesión.

Historias de Usuario

Se definen como una pequeña descripción de las necesidades del cliente, esto permite que dichas historias se puedan asignar ordenadamente en cada una de las iteraciones que deben ser desarrolladas, probadas e implementadas [15]. A continuación, la **TABLA III** representa una de las Historias de Usuario que se ha utilizado para el desarrollo del *backend*, mientras que las demás Historias de usuario se encuentran detalladas en el **ANEXO II** del presente documento.

TABLA III: Historia de usuario Nro.1

Historia de Usuario	
Identificador: HU001	**Usuario:** Invitado, administrador y miembro de la Iglesia.
Nombre historia: Generar varios *endpoints* para mostrar información de la Iglesia.	
Prioridad en Negocio (Alto/Medio/Bajo): Bajo	**Riesgo en Negocio (Alto/Medio/Bajo):** Bajo
Iteración asignada: 1	
Responsable: Michael Guanoluisa	

Descripción: El *backend* por medio del perfil asignado permite generar varios *endpoints* para proporcionar información general de la Iglesia. No obstante, el *backend* retorna *endpoints* que pueden ser consumidos por cualquier aplicación por el lado del cliente.
Observación: Los *endpoints* mencionados anteriormente son de acceso público.

Product Backlog

Es una lista de tareas definidas o establecidas por el *Scrum Master* el cual las ordena por el tipo de prioridad que tengan cada una de estas, permitiendo que el equipo de desarrollo pueda desarrollar los módulos que tengan más prioridad para la entrega [15]. A continuación, en la **TABLA IV**, se muestra el formato que se ha utilizado para listar el *Product Backlog*, mientras que la tabla completa se detalla en el **ANEXO II** del presente documento.

TABLA IV: Formato del *Product Backlog*

ID-PB	ID-HU	HISTORIA DE USUARIO	ITERACIÓN	PRIORIDAD
PB-001	HU001	Generar varios *endpoints* para mostrar información de la Iglesia.	1	Bajo
PB-002	HU002	Generar varios *endpoints* para iniciar y cerrar sesión.	1	Alta

Sprint Backlog

Representa una tabla que muestra las tareas del *Product Backlog* con mucho más detalle, las mismas que van a formar parte de cada uno de los *Sprints*. Además, cada una de estas tareas cuentan con un tiempo requerido para su cumplimiento [15]. A continuación, se muestra la **TABLA V**, la misma que representa el formato que se ha utilizado en el desarrollo de este proyecto para listar los 5 *Sprints*, mientras que la tabla completa con todas las tareas se detalla en el **ANEXO II** del presente documento.

TABLA V: Formato del *Sprint Backlog*

ELABORACIÓN DEL *SPRINT BACKLOG*						
ID-SB	**NOMBRE**	**MÓDULO**	**ID-HU**	**HISTORIA DE USUARIO**	**TAREAS**	**TIEMPO ESTIMADO**
SB000	Configuración del ambiente de desarrollo	-------	-----	----------	• Definición de requerimientos. • Elaboración de la Base de Datos en MongoDB. • Arquitectura *REST*. • Roles de usuarios.	20 H

2.2 Diseño de la arquitectura

El definir una arquitectura de *software* permite que el equipo de desarrollo pueda trabajar correctamente en cada uno de los módulos y la integración con nuevas herramientas y librerías. Asegurando de esta manera la calidad del código, facilitando el mantenimiento, evolución e implementación de nuevas funcionalidades [15]. Para ello es necesario en este proyecto definir una arquitectura ideal para la etapa de codificación del *backend* y su integración con otras herramientas o tecnologías.

Patrón arquitectónico

Para el desarrollo del *backend* se ha utilizado la arquitectura Modelo Vista Controlador (MVC) la cual emplea la separación del *software* en 3 componentes. Además, esta arquitectura ha mostrado en la actualidad ser un patrón arquitectónico ampliamente utilizado en el campo del desarrollo y su utilización en grandes empresas [17].

- **Modelo:** representa la información de la Base de datos utilizada por medio de consultas.
- **Vista:** representa la parte visual con la que llega a interactuar el usuario final.

- **Controlador:** representa el intermediario entre el Modelo y la Vista, la cual se encarga de comunicar y transportar la información de una manera íntegra para el usuario final.

La **Fig. 1**, muestra el patrón arquitectónico que se ha implementado para el desarrollo del *backend*, el mismo que permite tener una serie de ventajas como: estructura organizada, facilita el mantenimiento y la mejora continua si se lo requiere.

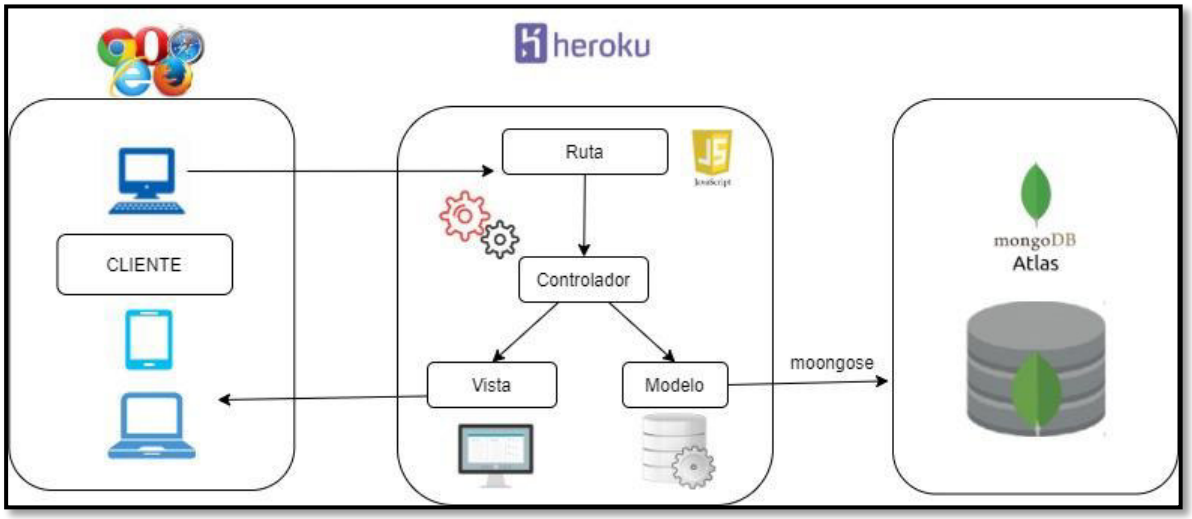

Fig. 1: Patrón Arquitectónico – *backend*

2.3 Herramientas de desarrollo

En el proceso de codificación se utilizan herramientas de desarrollo de *software*, las cuales permiten el diseño, modelado, codificación, integración, pruebas, etc., dependiendo del tipo de proyecto que se va a desarrollar. Sin embargo, el proceso de desarrollo conlleva algunas etapas importantes en donde se requiere de su utilización e integración con otras más [18]. Es por esta razón, que en la **TABLA VI** se muestra las herramientas que se han utilizado durante el desarrollo del *backend* y la comunicación con la base de datos no relacional (NoSQL).

TABLA VI: Herramientas para el desarrollo del *backend*

HERRAMIENTA	JUSTIFICACIÓN
Node.js	Es un entorno de JavaScript que permite crear y ejecutar el *backend* en el lado del servidor [11].

MongoDB	Es un Sistema Gestor de Base de datos no relacional NoSQL, que permite almacenar la información y realizar consultas o peticiones en tiempo real ya que no se necesita relaciones [10].
Heroku	Es una plataforma en la nube que ayuda a llevar un proyecto a producción de manera rápida, eficaz y óptima [19].
MongoDB Atlas	Es un servicio *cloud* de Base de datos NoSQL, destinado a la gestión de tareas en tiempo real por medio de consultas optimizadas [10].
Express	Este *Framework*, permite la creación de API's, *backends*, sistemas *web*, entre otros de una manera óptima y rápida [12].

Librerías

La **TABLA VII** muestra un conjunto de librerías que se han empleado para la codificación del *backend* y la integración con las herramientas listadas en la tabla anterior.

TABLA VII: Librerías para el desarrollo del *backend*

LIBRERÍA	DESCRIPCIÓN
Moongose	Es una librería de Node.js, que se encarga de la gestión de consultas para una base de datos NoSQL y aplicaciones *web* con el *Framework Express* [20].
bcrypt	Es una biblioteca que ayuda a codificar las contraseñas [21].
Jwt	Es un formato o estándar de codificación abierto basado en JSON para la creación de *tokens* de autorización [22].

3 RESULTADOS

En esta sección, se detallan cada uno de los resultados que se han obtenido en la implementación de los diferentes *endpoints* para el *backend,* así como el resultado de las pruebas y el proceso de despliegue a producción. No obstante, cada uno de los resultados se presenta por medio de *Sprints*.

3.1 *Sprint* 0. Configuración del ambiente de desarrollo

En base a lo estipulado en el *Sprint Backlog* este *Sprint* contiene las siguientes tareas:

- Definición de requerimientos.
- Elaboración de la Base de datos en *MongoDB*.
- Arquitectura *REST* para el *backend*.
- Roles de usuario.

Definición de requerimientos

Generar *endpoints* de información general

El *backend* crea un método y una ruta que permita obtener toda la información general de la Iglesia para los usuarios con perfil administrador, miembros de la Iglesia e invitados.

El *backend* crea un método y una ruta que permita gestionar toda la información general de la Iglesia para los usuarios con perfil administrador.

Generar *endpoints* para el registro de usuarios

El *backend* crea un método y una ruta que permita el registro de usuarios con perfil invitado.

Generar *endpoints* para la autenticación de usuarios

El *backend* crea un método y una ruta que permita el inicio de sesión de usuarios con perfil administrador y miembro de la Iglesia. No obstante, los endpoints que se han generado devuelven un *token* para un inicio de sesión seguro y que no afecte la integridad de la Base de datos.

Generar *endpoints* para gestionar eventos

El *backend* crea un método y una ruta que permita gestionar toda la información de eventos para los usuarios con perfil administrador. Mientras que, para los usuarios con perfil miembro de la Iglesia el *backend* crea un método y una ruta para visualizar, inscribirse y de ser el caso borrar la inscripción al evento.

Generar *endpoints* para gestionar vídeos

El *backend* crea un método y una ruta que permita gestionar toda la información de vídeos para los usuarios con perfil administrador. Mientras que, para los usuarios con perfil miembro de la Iglesia el *backend* crea un método y una ruta para visualizar aquellos vídeos que han sido registrados.

Generar *endpoints* para gestionar donaciones

El *backend* crea un método y una ruta que permita gestionar toda la información de donaciones para los usuarios con perfil administrador. Mientras que, para los usuarios con perfil miembro de la Iglesia el *backend* crea un método y una ruta para visualizar y realizar donaciones que estén disponibles.

Generar *endpoints* para gestionar cuestionarios

El *backend* crea un método y una ruta que permita gestionar toda la información de cuestionarios para los usuarios con perfil administrador. Mientras que, para los usuarios con perfil miembro de la Iglesia el *backend* crea un método y una ruta para visualizar y responder a los cuestionarios de la Iglesia.

A continuación, la **Fig. 2**, **Fig. 3** y **Fig. 4** muestran los diferentes *endpoints* a los que pueden interactuar los usuarios del *backend*.

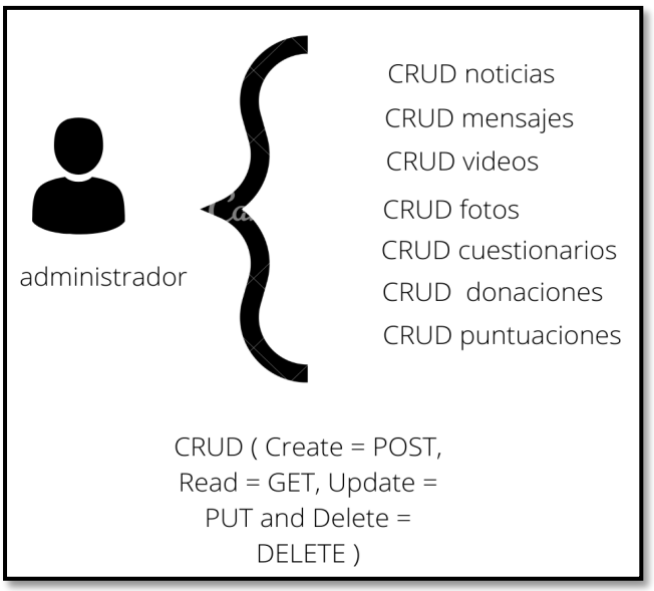

Fig. 2: Usuario con perfil administrador

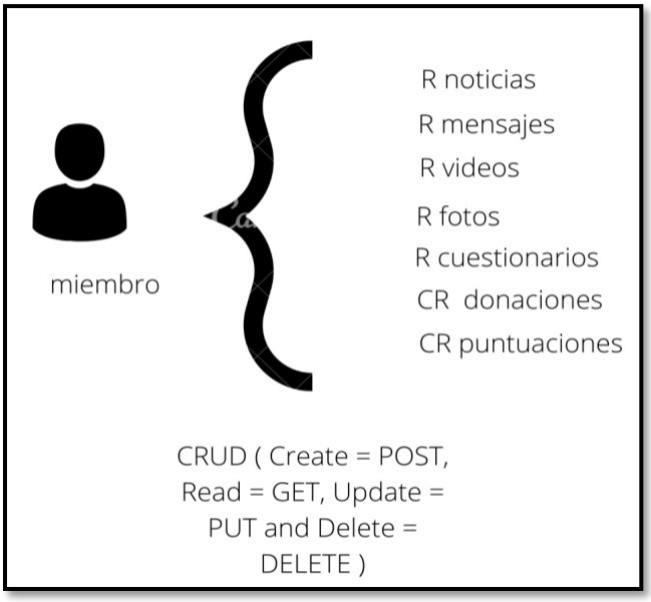

Fig. 3: Usuario con perfil miembro de la Iglesia

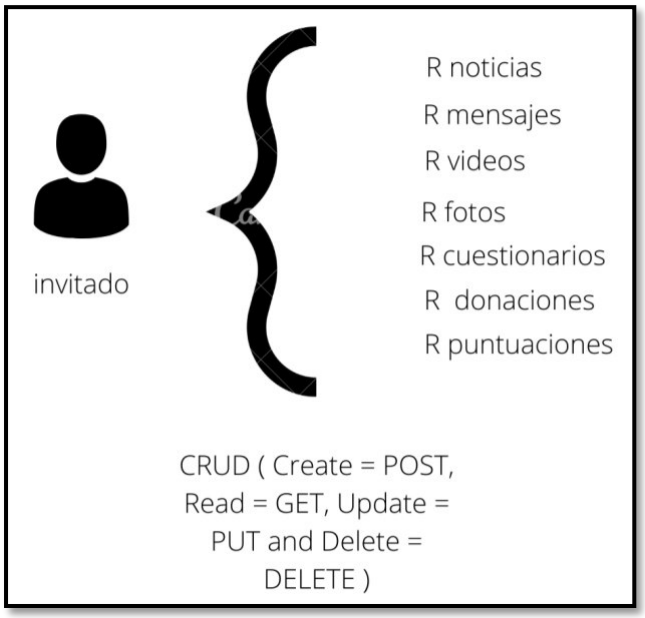

Fig. 4: Usuario con perfil invitado

Elaboración de la Base de datos en *MongoDB*

Para el almacenamiento de los datos que se requieren en este proyecto se ha optado por utilizar un Sistema Gestor de Base de Datos NoSQL llamado *MongoDB*, el cual permite sincronizar y almacenar toda la información en tiempo real por medio de colecciones y documentos. No obstante, la **Fig. 5** ilustra la base de datos *NoSQL* que se ha utilizado para

la comunicación con el *backend*, mientras que el esquema completo se lo puede apreciar en el **ANEXO II** del presente documento.

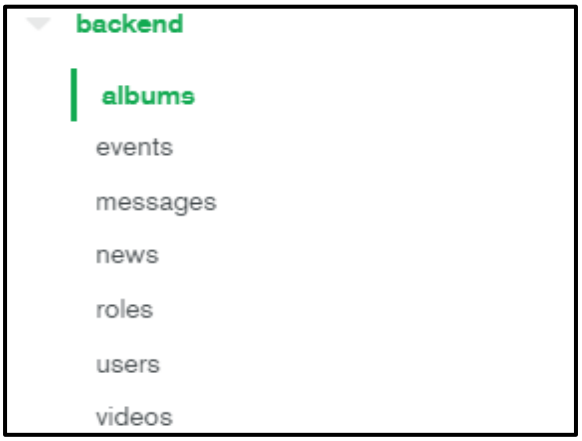

Fig. 5: Colecciones de la Base de datos *NoSQL*

Arquitectura *REST* para el *backend*

En el desarrollo del *backend* se ha utilizado una arquitectura *REST*, el cual permite utilizar una serie de principios y buenas prácticas al momento de crear una serie de *endpoints* ya sean privados o protegidos para garantizar en todo momento la integridad y disponibilidad de la información. Además, para la estructura de directorios, archivos y patrón de arquitectura se ha utilizado *Visual Studio Code*, el cual es un *IDE* para el desarrollo de aplicaciones de *software* siendo muy versátil y fácil de usar [23]. A continuación, la **Fig. 6** ilustra la estructura del proyecto.

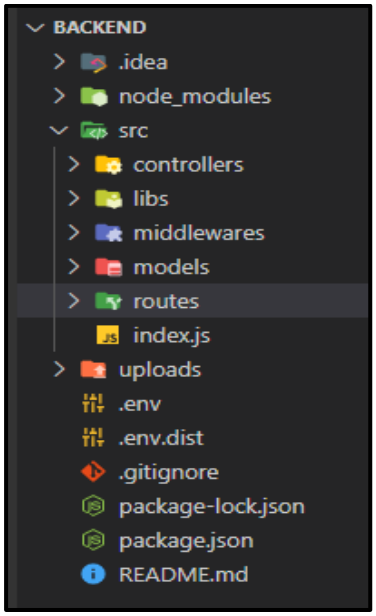

Fig. 6: Estructura del *backend*

Roles de usuarios

A continuación, la **Fig. 7** ilustra a los tres usuarios y a los módulos a los cuales tienen acceso dependiendo del rol que tengan asignado.

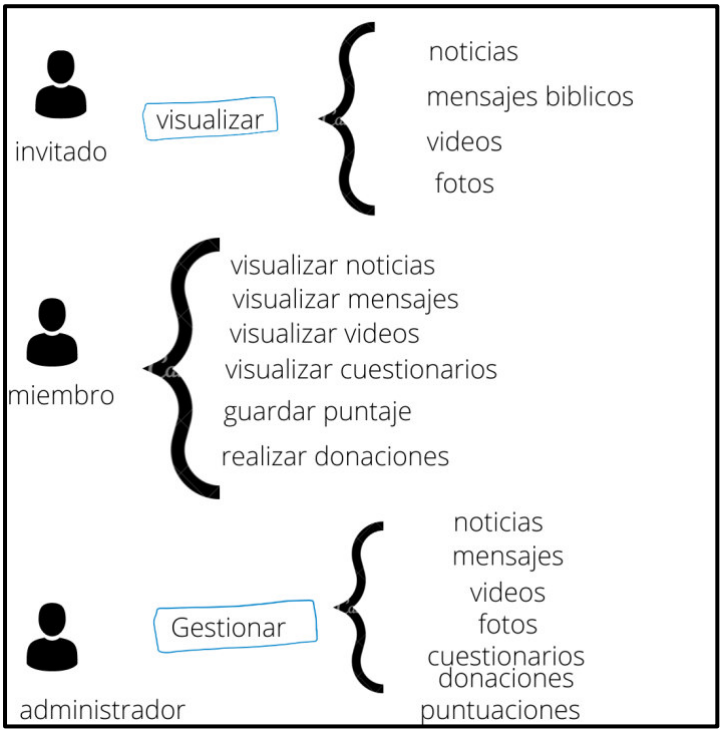

Fig. 7: Usuario y roles para el *backend*

3.2 *Sprint* 1. Resultado de la implementación de los *endpoints* para los usuarios con perfil invitado y administrador

En base a lo estipulado en el *Sprint Backlog* este *Sprint* contiene las siguientes tareas:

- Generar un *endpoint* para visualizar la información general de la Iglesia.
- Generar *endpoints* para gestionar la información general de la Iglesia.
- Generar *endpoints* para el registro y autenticación de usuarios.
- Generar *endpoints* para gestionar eventos.
- Generar *endpoint* para gestionar donaciones.
- Generar *endpoint* para visualizar y aprobar donaciones.
- Generar *endpoints* para gestionar vídeos.
- Generar *endpoints* para gestionar cuestionarios.

Generar un *endpoint* para visualizar la información general de la Iglesia

Toda la información referente a la Iglesia se encuentra almacenada en una colección de la base de datos NoSQL la cual posee información como: noticias, eventos, personal, entre otros. Además, por medio del *backend* se ha creado un método y una ruta pública de tipo *GET* el cual permite obtener toda la información para que pueda ser consumida por parte de una aplicación del lado del cliente o móvil como se ilustra en la **Fig. 8**. Por otra parte, el proceso para realizar el consumo de la información y las validaciones respectivas se lo puede apreciar de mejor manera en el **ANEXO III** del presente documento.

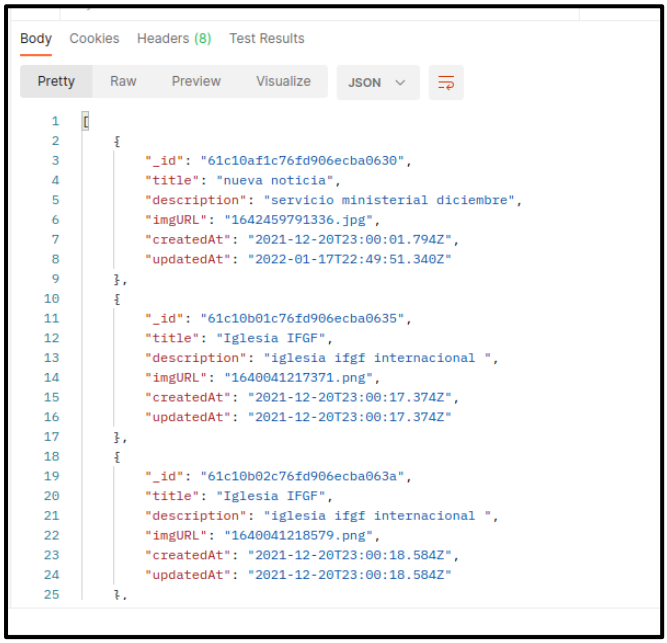

Fig. 8: Método *GET* de información general

Generar *endpoints* para gestionar la información general de la Iglesia

Para la gestión de la información de la Iglesia se han creado varios métodos, rutas públicas y privadas los cuales permiten que el usuario con perfil administrador pueda gestionar toda la información referente a la Iglesia como: noticias, álbum de fotos, mensajes bíblicos, eventos, entre otros. Además, por medio del *backend* se ha creado una ruta pública de tipo *GET* para obtener toda la información, una ruta privada de tipo *POST* para ingresar la información a través de un formulario, una ruta privada de tipo *PUT* para la actualización de la información y una ruta privada de tipo *DELETE* para la eliminación de la información si fuera el caso como se ilustra en la **Fig. 9**. Por otra parte, el proceso para realizar el consumo de la información y las validaciones respectivas se lo puede apreciar de mejor manera en el **ANEXO III** del presente documento.

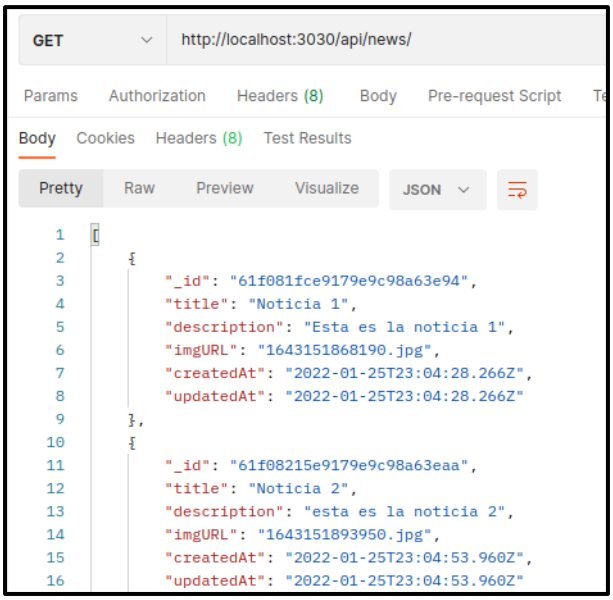

Fig. 9: Método *GET* de noticias

Generar *endpoints* para el registro y autenticación de usuarios

Para el inicio de sesión y registro de usuarios se han creado varios métodos y rutas privadas los cuales han permitido llevar un mejor control al momento de que cualquier usuario quiera consumir la información del *backend.* En ese sentido por medio del *backend* se ha creado un método y ruta privada de tipo *POST* para ingresar la información a través de un formulario y el registro en la Base de Datos NoSQL. Adicional a ello, se ha creado un método y ruta privada de tipo *POST* para ingresar la información a través de un formulario y luego realizar el inicio de sesión respectivo como se ilustra en la **Fig. 10**. Además, el proceso para realizar el consumo de la información y las validaciones respectivas se lo puede apreciar de mejor manera en el **ANEXO III** del presente documento.

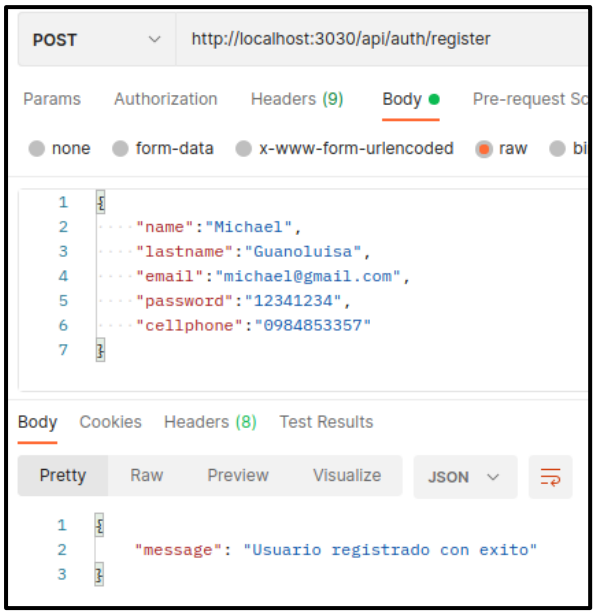

Fig. 10: Método *POST* para registrarse

Generar *endpoints* para gestionar eventos

Para la gestión de eventos de la Iglesia se han creado varios métodos, rutas públicas y privadas los cuales permiten que el usuario con perfil administrador pueda gestionar toda la información de una manera mucho más organizada. Además, por medio del *backend* se ha creado una ruta pública de tipo *GET* para obtener toda la información del evento, una ruta privada de tipo *POST* para ingresar la información del evento a través de un formulario, una ruta privada de tipo *PUT* para la actualización de la información del evento y una ruta privada de tipo *DELETE* para la eliminación de la información del evento si fuera el caso como se ilustra en la **Fig. 11**. Por otra parte, el proceso para realizar el consumo de la información y las validaciones respectivas se lo puede apreciar de mejor manera en el **ANEXO III** del presente documento.

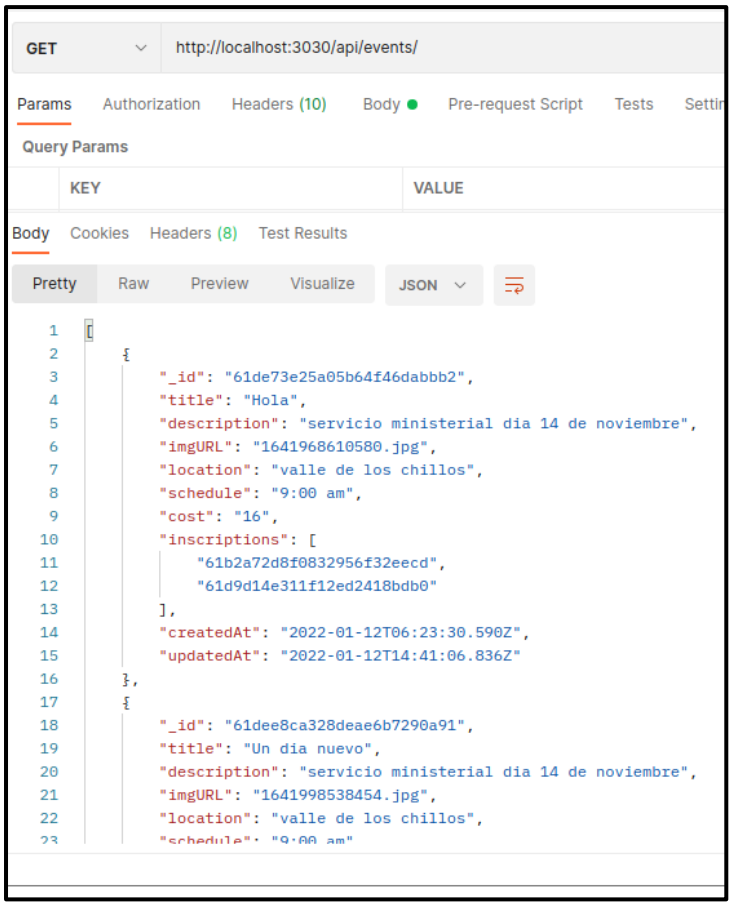

Fig. 11: Método *GET* para visualizar los eventos

Generar *endpoint* para gestionar donaciones

Para la gestión de donaciones de la Iglesia se han creado varios métodos, rutas públicas y privadas los cuales permiten que el usuario con perfil administrador pueda gestionar toda la información de una manera mucho más organizada. Además, por medio del *backend* se ha creado una ruta pública de tipo *GET* para obtener toda la información de la donación, una ruta privada de tipo *POST* para ingresar la información de la donación a través de un formulario, una ruta privada de tipo *PUT* para la actualización de la información de la donación y una ruta privada de tipo *DELETE* para la eliminación de la información de la donación si fuera el caso como se ilustra en la **Fig. 12**. Por otra parte, el proceso para realizar el consumo de la información y las validaciones respectivas se lo puede apreciar de mejor manera en el **ANEXO III** del presente documento.

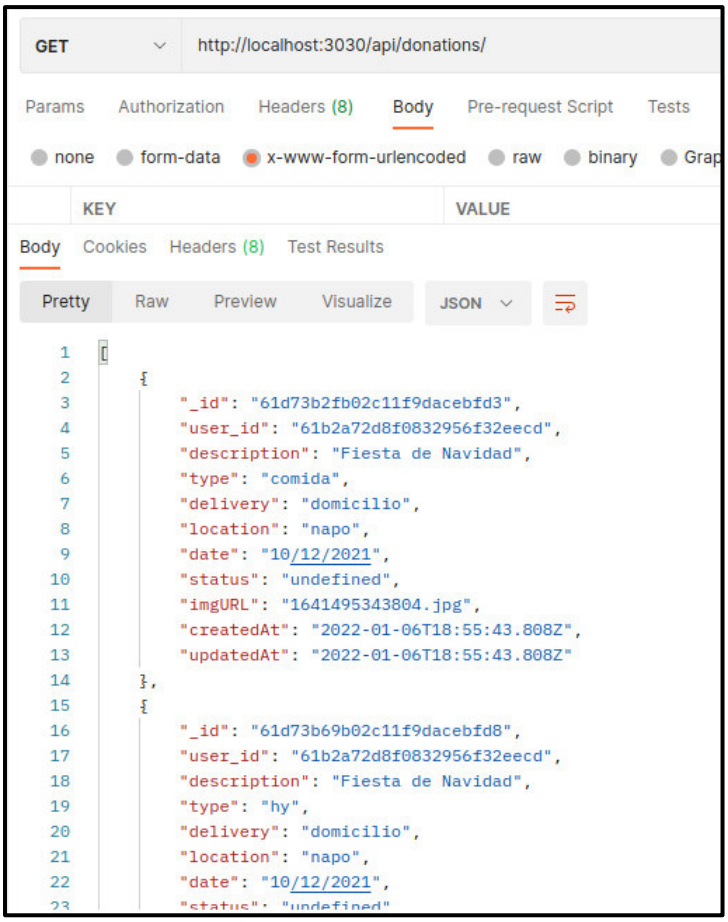

Fig. 12: Método *GET* para visualizar las donaciones

Generar *endpoint* para visualizar y aprobar donaciones

Toda la información referente a las donaciones que se han realizado por parte de los miembros de la Iglesia se encuentra almacenada en una colección de la base de datos NoSQL. Además, por medio del *backend* se ha creado un método y una ruta pública de tipo *GET* el cual permite obtener toda la información y aprobar la donación como se ilustra en la **Fig. 13**. Por otra parte, el proceso para realizar el consumo de la información y las validaciones respectivas se lo puede apreciar de mejor manera en el **ANEXO III** del presente documento.

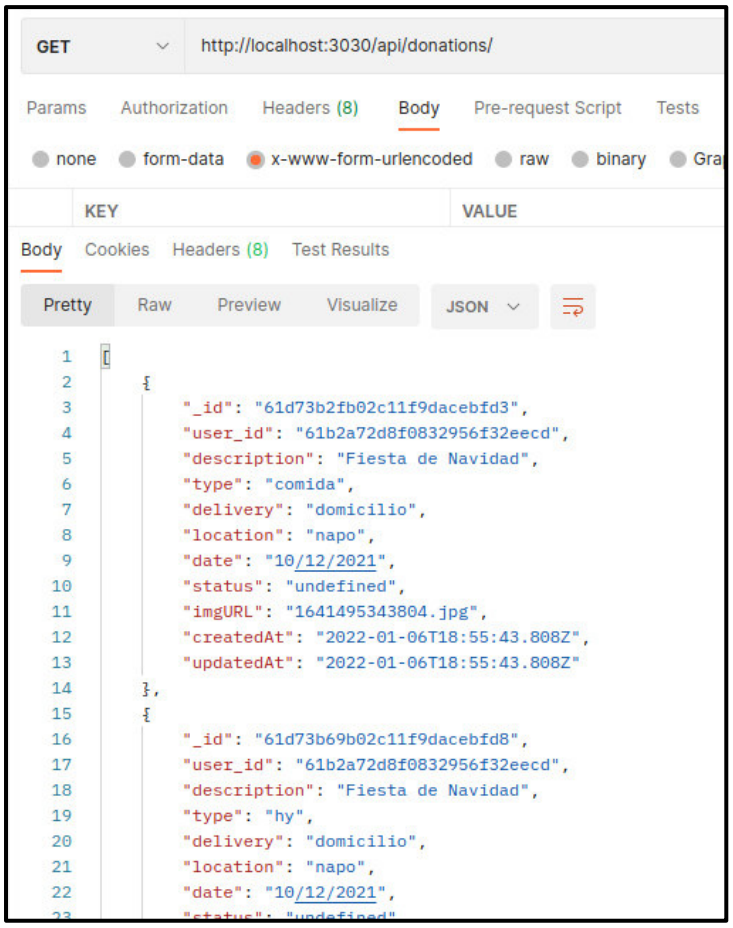

Fig. 13: Método *GET* obtener las donaciones

Generar *endpoints* para gestionar vídeos

Para la gestión de vídeos de la Iglesia se han creado varios métodos, rutas públicas y privadas los cuales permiten que el usuario con perfil administrador pueda gestionar toda la información de una manera mucho más organizada. Además, por medio del *backend* se ha creado una ruta pública de tipo *GET* para obtener toda la información del video, una ruta privada de tipo *POST* para ingresar la información del video a través de un formulario, una ruta privada de tipo *PUT* para la actualización de la información del video y una ruta privada de tipo *DELETE* para la eliminación de la información del video si fuera el caso como se ilustra en la **Fig. 14**. Por otra parte, el proceso para realizar el consumo de la información y las validaciones respectivas se lo puede apreciar de mejor manera en el **ANEXO III** del presente documento.

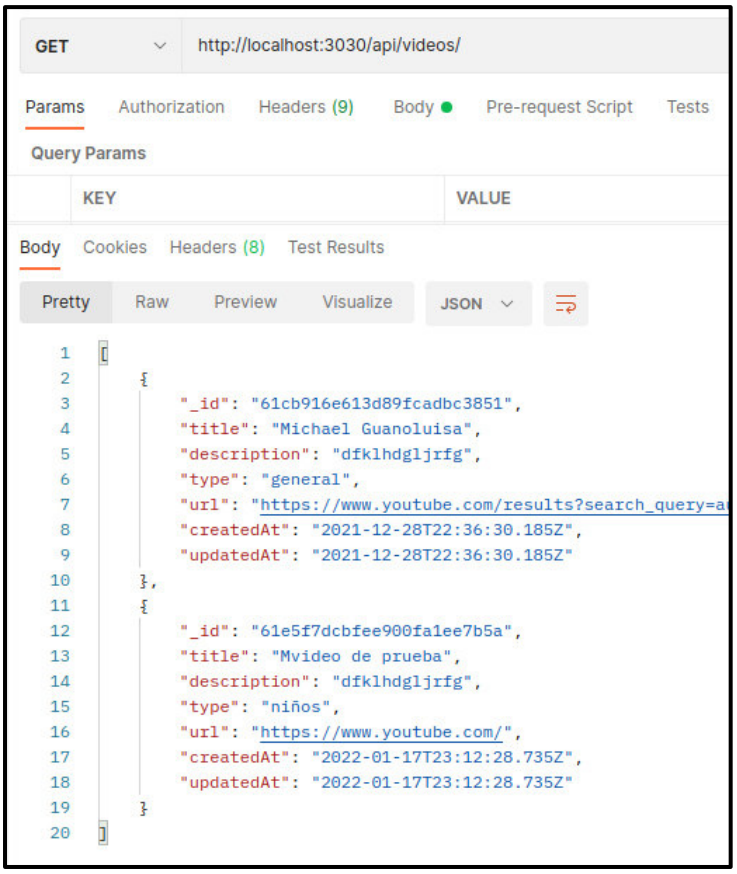

Fig. 14: Método *GET* para obtener vídeos

Generar *endpoints* para gestionar cuestionarios

Para la gestión de cuestionarios de la Iglesia se han creado varios métodos, rutas públicas y privadas los cuales permiten que el usuario con perfil administrador pueda gestionar toda la información de una manera mucho más organizada. Además, por medio del *backend* se ha creado una ruta pública de tipo *GET* para obtener toda la información del cuestionario, una ruta privada de tipo *POST* para ingresar la información del cuestionario a través de un formulario, una ruta privada de tipo *PUT* para la actualización de la información del cuestionario y una ruta privada de tipo *DELETE* para la eliminación de la información del cuestionario si fuera el caso como se ilustra en la **Fig. 15**. Además, el proceso para realizar el consumo de la información y las validaciones respectivas se lo puede apreciar de mejor manera en el **ANEXO III** del presente documento.

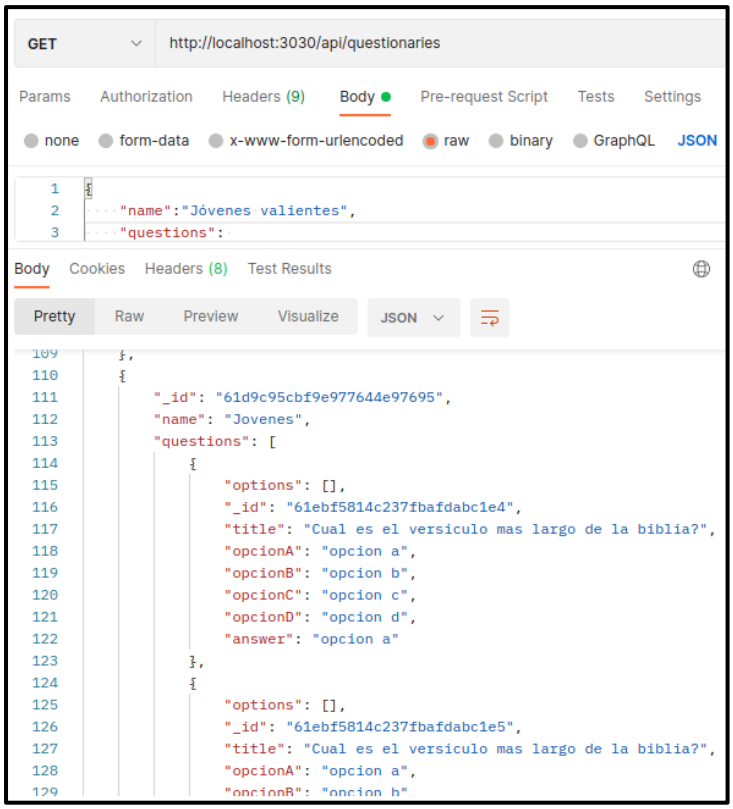

Fig. 15: Método *GET* para visualizar cuestionarios

3.3 *Sprint* 2. Resultado de la implementación de los *endpoints* para el usuario con perfil miembro de la Iglesia

En base a lo estipulado en el *Sprint Backlog* este *Sprint* contiene las siguientes tareas:

- Generar un *endpoint* para interactuar con donaciones.
- Generar *endpoints* para interactuar con eventos.
- Generar un *endpoint* para interactuar con cuestionarios.
- Generar un endpoint para visualizar vídeos.

Generar un *endpoint* para interactuar con donaciones

Por medio del *backend* se han creado varios métodos, rutas públicas y privadas que permiten al usuario con perfil miembro de la Iglesia interactuar con cada donación que ha publicado la Iglesia. En ese sentido, se ha creado una ruta pública de tipo *GET* para obtener toda la información de la donación, una ruta privada de tipo *POST* para realizar una donación a través de un formulario y una ruta privada de tipo *DELETE* para cancelar la donación si fuera el caso como se ilustra en la **Fig. 16**. Además, el proceso para realizar el

consumo de la información y las validaciones respectivas se lo puede apreciar de mejor manera en el **ANEXO III** del presente documento.

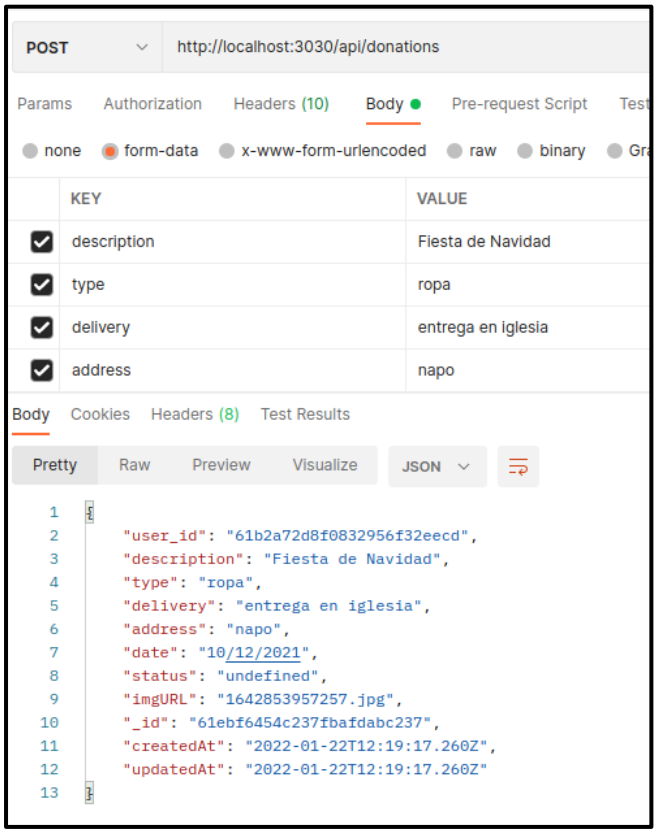

Fig. 16: Método *POST* para crear donaciones

Generar *endpoints* para interactuar con eventos

Por medio del *backend* se han creado varios métodos, rutas públicas y privadas que permiten al usuario con perfil miembro de la Iglesia interactuar con cada evento que ha publicado la Iglesia. En ese sentido, se ha creado una ruta pública de tipo *GET* para obtener toda la información del evento, una ruta privada de tipo *PUT* para registrarse al evento a través de un formulario y una ruta privada de tipo *DELETE* para cancelar la inscripción al evento si fuera el caso como se ilustra en la **Fig. 17**. Además, el proceso para realizar el consumo de la información y las validaciones respectivas se lo puede apreciar de mejor manera en el **ANEXO III** del presente documento.

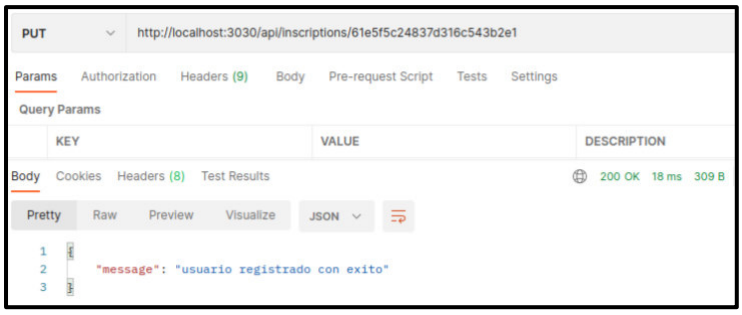

Fig. 17: Método *PUT* para inscribirse a un evento

Generar un *endpoint* para interactuar con cuestionarios

Por medio del *backend* se han creado varios métodos, rutas públicas y privadas que permiten al usuario con perfil miembro de la Iglesia interactuar con cada cuestionario que ha publicado la Iglesia. En ese sentido, se ha creado una ruta pública de tipo *GET* para obtener toda la información del cuestionario, una ruta privada de tipo *POST* para realizar un cuestionario a través de un formulario y un método para cancelar el cuestionario si fuera el caso como se ilustra en la **Fig. 18**. Además, el proceso para realizar el consumo de la información y las validaciones respectivas se lo puede apreciar de mejor manera en el **ANEXO III** del presente documento.

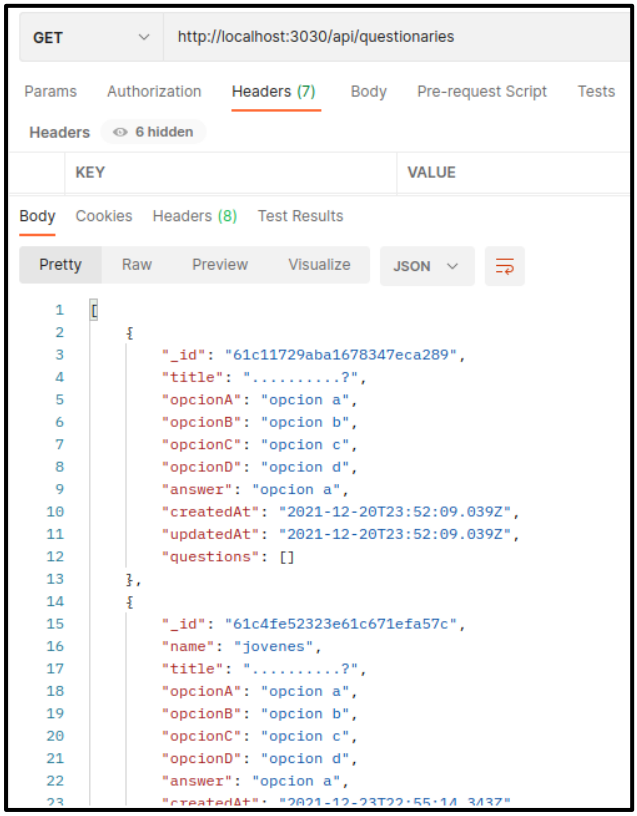

Fig. 18: Método *GET* para visualizar cuestionarios

Generar un *endpoint* que le permita visualizar vídeos

Toda la información referente a los vídeos por parte de la Iglesia se encuentra almacenada en una colección de la base de datos NoSQL. No obstante, por medio del *backend* se ha creado un método y una ruta pública de tipo *GET* el cual permite obtener toda la información de los vídeos como se ilustra en la **Fig. 19**. Además, el proceso para realizar el consumo de la información y las validaciones respectivas se lo puede apreciar de mejor manera en el **ANEXO III** del presente documento.

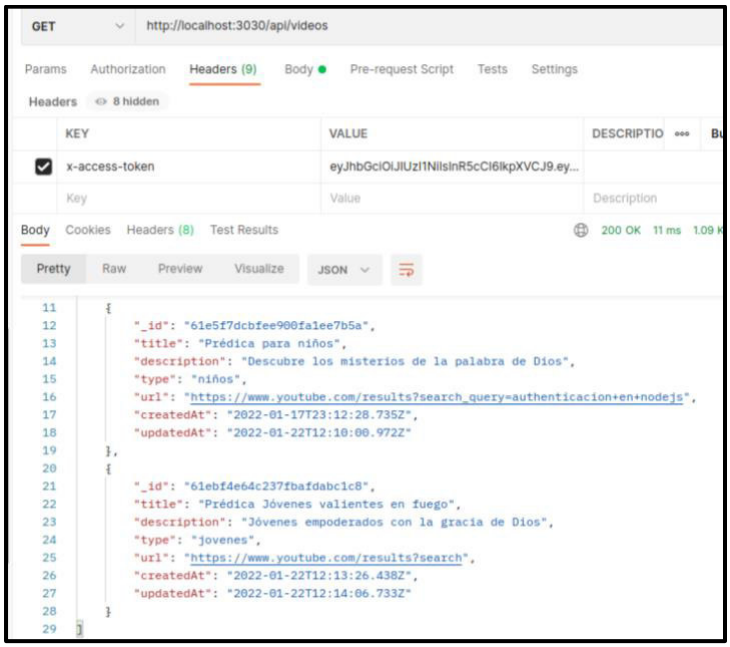

Fig. 19: Método *GET* visualizar vídeos

3.4 *Sprint* 3. Pruebas en el *backend*

Una vez que se ha finalizado la etapa de codificación de cada uno de los *endpoints* en el *backend* y en base a lo estipulado en el *Sprint Backlog* este *Sprint* contiene las siguientes tareas:

- Ejecución de pruebas unitarias y resultados.
- Ejecución de pruebas de rendimiento y resultados.
- Ejecución de pruebas de aceptación y resultados.

Ejecución de pruebas unitarias y resultados

Al finalizar el proceso de codificación y siguiendo la planificación que se ha establecido, en esta sección se procede a realizar las pruebas unitarias al *backend*, permitiendo conocer si existe algún fallo en alguna de las funcionalidades. En ese sentido, para realizar dichas

pruebas se ha utilizado *Mocha* y *Chai*, las cuales son herramientas que permiten realizar pruebas unitarias completas para proporcionar calidad al proyecto [24].

La **Fig. 20** muestra un fragmento del código que se ha implementado para el inicio de sesión de los usuarios, por otra parte, en la **Fig. 21** se puede apreciar el resultado que se ha obtenido después de haber realizado la respectiva prueba. Mientras que el detalle completo de la ejecución y resultados de las demás pruebas se lo puede apreciar en el **ANEXO II** del presente documento.

```
it("Post login respond a json with contain a user", (done) => {
  data = {
    email: "admin@hotmail.com",
    password: "12341234",
  };
  request(app)
    .post("/api/auth/login")
    .send(data)
    .set("Accept", "application/json")
    .expect("Content-Type", /json/)
    .expect(200)
    .end(done);
});
```

Fig. 20: Porción de código de prueba para el inicio de sesión

```
DB esta conectada
POST /api/auth/login 200 115.182 ms - 300
  ✔ Post login respond a json with contain a user (129ms)
POST /api/auth/register 200 81.921 ms - 42
  ✔ Post register respond a json with message user registered (85ms)
GET /api/news 200 6.236 ms - 5792
  ✔ Get News respond a json with contain news
GET /api/news/61c10af1c76fd906ecba0634 404 2.379 ms - 2
  ✔ Get a news by id respond 404 not found
POST /api/news 403 0.761 ms - 28
  ✔ Post news with a json
POST /api/news 201 11.774 ms - 180
  ✔ Post news with a json
```

Fig. 21: Resultado de la prueba unitaria de inicio de sesión

Tras la ejecución de esta prueba y en base a los resultados que se han obtenido, se determina que cada uno de los módulos del *backend* no presenta ningún fallo a nivel de funcionalidad o validación respectivamente.

Resultados de pruebas de compatibilidad

Esta prueba permite verificar la compatibilidad del contenido en varios dispositivos [8], en el caso de este proyecto que pertenece al desarrollo de un *backend* las pruebas de compatibilidad han sido ejecutadas por clientes HTTP, los cuales son: *Postman* y *Thunder Client,* estos clientes han permitido verificar que el contenido se presente de forma correcta

en la invocación de cada uno de los *endpoints* ya sean públicos o protegidos. En ese sentido, la **TABLA VIII** presenta los clientes HTTP en donde se han ejecutado las respectivas pruebas, mientras que el detalle completo de la ejecución y resultados de esta prueba se lo puede apreciar en el **ANEXO II** del presente documento.

TABLA VIII: Clientes HTTP para la prueba de compatibilidad

NOMBRE	VERSIÓN
Postman	V 9.9.3
Thunder Client	V 1.10.0

Tras la ejecución de estas pruebas y en base a los resultados que se han obtenido, se determina que el *backend* no presenta ningún fallo a nivel de tiempo de respuesta o presentación de la información con las solicitudes solicitadas respectivamente.

Resultados de pruebas de aceptación

El propósito de estas pruebas es que el usuario que pruebe el *software* pueda verificar y comprobar que los requerimientos que se han establecido al principio del proyecto se hayan desarrollado con éxito. Además, estas pruebas deben realizarse antes de que un sistema *software* sea puesto a producción [25]. En ese sentido, la **TABLA IX** presenta una prueba de aceptación y el resultado que se ha obtenido, mientras que el detalle completo de la ejecución y resultados de esta prueba se lo puede apreciar en **ANEXO II** del presente documento.

TABLA IX: Prueba de aceptación Nro. 1

Prueba de Aceptación	
Identificador: PA001	**Identificador historia de Usuario:** HU001
Nombre: Generar varios *endpoints* para mostrar información de la Iglesia.	

Descripción: El *backend* por medio del perfil asignado permite generar varios *endpoints* para proporcionar información general de la Iglesia. Además, el *backend* retorna *endpoints* que pueden ser consumidos por cualquier aplicación por el lado del cliente.
Pasos de ejecución: • Acceder a las rutas generadas por el *backend*, las mismas que son públicas. • Llamar al método *GET* de *Albums*, *News* y *Messages*. • El *backend* genera la respuesta con los datos de la Base de datos.
Resultado deseado: El *backend* le permite al usuario generar *endpoints* con los datos de visualización de informacion general de la Iglesia.
Evaluación de la prueba: El resultado es el esperado y se tiene la aceptación completa del cliente.

Tras la ejecución de estas pruebas y en base a los resultados que se han obtenido, se determina que los usuarios del *backend* están conformes con todas las funcionalidades logrando la aprobación del *Product Owner* y con ello pasar a la siguiente etapa.

3.5 Sprint 4. Despliegue del *backend*

Una vez que se ha finalizado la etapa de codificación en el *backend* y en base a los resultados que se han obtenido por cada una de las pruebas, en esta sección se presenta las siguientes tareas a realizar:

- Despliegue de la Base de datos en *MongoDB Atlas*.
- Despliegue del *backend* en *Heroku*.

Despliegue de la Base de datos en *MongoDB Atlas*

Una vez que se ha finalizado el proceso de codificación y las pruebas respectivas, se realiza el despliegue de la Base de datos NoSQL en *MongoDB Atlas*, como se ilustra en la **Fig. 22** cabe recalcar que este procedimiento de despliegue se encuentra detallado de mejor manera en el en el **ANEXO I** del presente documento.

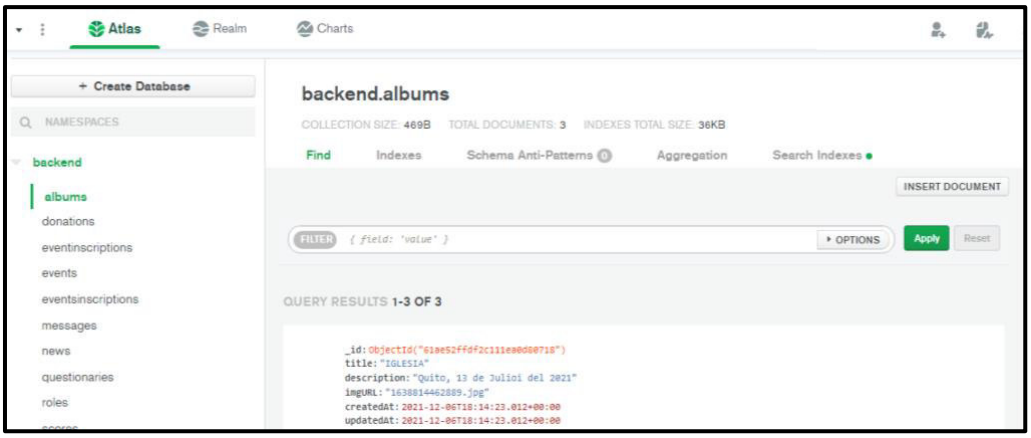

Fig. 22: Base de datos NoSQL alojada en *MongoDB Atlas*

Despliegue del *backend* en *Heroku*

Una vez que se ha finalizado el proceso de codificación y las pruebas respectivas, se realiza el despliegue del backend en la plataforma *Heroku*, como se ilustra en la **Fig. 23** cabe recalcar que este procedimiento de despliegue se encuentra detallado de mejor manera en el en **ANEXO I** el del presente documento. Por último, para acceder al *backend*, se debe ingresar en el navegador la siguiente url:

https://backend-ifgf.herokuapp.com/api/

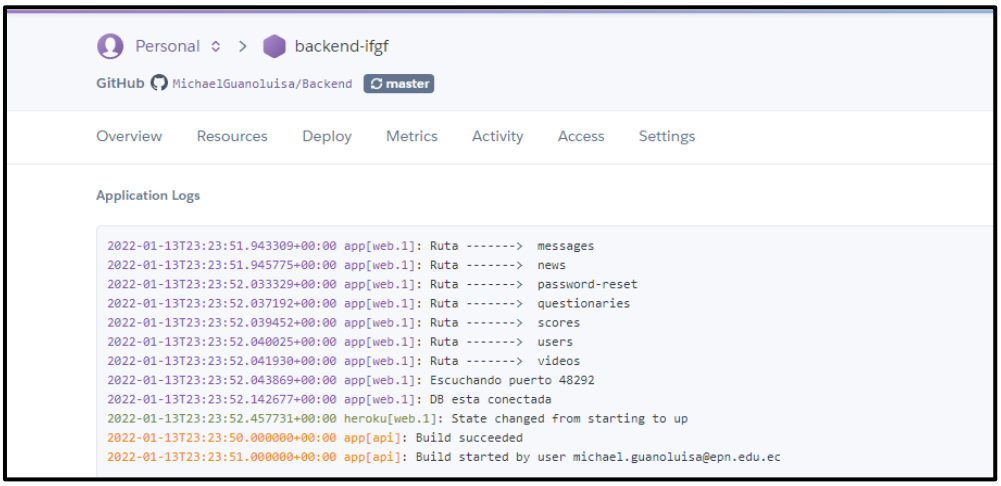

Fig. 23: *backend* alojado en *Heroku*

Finalmente, el pastor Walter Hidalgo de la Iglesia Fraternidad Internacional del Evangelio Completo GISI (IFGF) ha otorgado un certificado, el mismo que ratifica el cumplimiento de todos los requerimientos y funcionalidades del backend que han sido solicitados al comienzo del proyecto. El certificado se adjunta en el **ANEXO II.**

4 CONCLUSIONES

En esta sección se muestran las conclusiones que se han obtenido durante el desarrollo de este trabajo de integración curricular.

- El listado de requerimientos que se han obtenido al comienzo del proyecto han hecho que la implementación del *backend* juntamente con las API's *RESTful* se hayan desarrollado de una forma estructurada y escalable si a un futuro se requiere agregar más perfiles y módulos.
- El uso de la metodología ágil *Scrum* ha permitido que el proyecto tenga avances y correcciones en cada *Sprint*, logrando de esta manera obtener un producto de calidad en los tiempos que se han establecido al inicio del proyecto.
- El uso de la arquitectura Modelo-Vista-Controlador en el desarrollo del *backend* ha permitido estructurar de forma correcta cada uno de los *endpoints* y que sea de fácil entendimiento para cualquier desarrollador si en el caso de que alguna persona se integre como parte del equipo de desarrollo en un futuro.
- El uso de la Base de datos NoSQL, ha permitido que la gestión de la información sea de una forma mucho más organizada y que el acceso sea mucho más rápido ya que al no existir relaciones permite acceder a las mismas en tiempo real.
- Actualmente el *backend* está desplegado y listo para ser usado por cualquier aplicación por parte del cliente con el que desee consumirse las rutas que se han generado por el *backend*.
- Las pruebas que se han ejecutado son una buena forma de probar todas las funcionalidades en porciones de código para que funcionen de forma exacta.

5 RECOMENDACIONES

En esta sección se muestran las recomendaciones que se han obtenido durante el desarrollo de este trabajo de integración curricular.

- Es importante verificar el uso de las versiones de Node.js que sean estables o que tengan soporte, ya que puede afectar considerablemente al momento de hacer el despliegue en cualquier plataforma de la nube.
- Se recomienda que el administrador establezca nuevas políticas de privacidad de información para nuevos *endpoints* que se vayan desarrollando.
- Se recomienda realizar *backups* de la Base de Datos cuando en ciertos períodos de tiempo programados, con el objetivo de salvaguardar toda la información y requerir cuando sea necesario.
- Se recomienda que en una siguiente versión del *backend* se pueda integrar a todas las Iglesias que pertenecen a la Ciudad de Quito.

6 REFERENCIAS BIBLIOGRÁFICAS

[1] P. Romo, R. Prado y J. Zevallos, «ACUERDO INTERMINISTERIAL NO. 00010,» 18 09 2020. [En línea]. Available: https://www.gestionderiesgos.gob.ec/wp-content/uploads/2020/10/acuerdo_interministerial_no__00010_MSP_TURISMO_MDG.pdf. [Último acceso: 12 07 2021].

[2] P. W. Hidalgo, Interviewee, *Pandemia e Iglesia.* [Entrevista]. 12 07 2021.

[3] APIS, BANKING AS A SERVICE, DESARROLLADORES, «BBVA,» 23 03 2016. [En línea]. Available: https://www.bbvaapimarket.com/es/mundo-api/api-rest-que-es-y-cuales-son-sus-ventajas-en-el-desarrollo-de-proyectos/. [Último acceso: 10 2021].

[4] I. d. Souza, «rockcontent,» 17 03 2020. [En línea]. Available: https://rockcontent.com/es/blog/api-rest/. [Último acceso: 10 2021].

[5] M. Mejia, «¿Qué es el Backend y cómo usarlo?,» 16 02 2021. [En línea]. Available: https://www.crehana.com/ec/blog/desarrollo-web/que-es-el-backend-y-como-usarlo/?gclid=EAIaIQobChMIpuSP4YrG9QIVmWSLCh0Shw4oEAAYASAAEgJtY_D_BwE. [Último acceso: 12 2021].

[6] Analitica web, «{ida BLOG,» 13 10 2020. [En línea]. Available: https://blog.ida.cl/desarrollo/mongodb-atlas-el-salto-a-la-nube/. [Último acceso: 12 2021].

[7] Red hat, «Red hat,» 08 05 2020. [En línea]. Available: https://www.redhat.com/es/topics/api/what-is-a-rest-api. [Último acceso: 12 2021].

[8] Ebooks online, «Ebooks online,» 13 04 2021. [En línea]. Available: https://ebooksonline.es/que-es-una-prueba-de-compatibilidad-prueba-hacia-adelante-y-hacia-atras-ejemplo/#:~:text=La%20prueba%20de%20compatibilidad%20es,de%20red%20o%20M%C3%B3vil%20gairis.. [Último acceso: 12 2021].

[9] ayudaley, [En línea]. Available: https://ayudaleyprotecciondatos.es/bases-de-datos/no-relacional/. [Último acceso: 12 2021].

[10] MongoDB, «MongoDB,» [En línea]. Available: https://www.mongodb.com/es/what-is-mongodb. [Último acceso: 12 2021].

[11] J. Rivera, «¿Qué es NodeJS? Tu mejor aliado para crear aplicaciones web,» 08 02 2021. [En línea]. Available: https://www.crehana.com/ec/blog/desarrollo-web/que-es-nodejs/?gclid=EAIaIQobChMI7df-z4bT9QIVqwaICR2RZQ9REAAYASAAEgJLzvD_BwE. [Último acceso: 12 2021].

[12] MSN web docs, «Introducción a Express/Node,» [En línea]. Available: https://developer.mozilla.org/es/docs/Learn/Server-side/Express_Nodejs/Introduction. [Último acceso: 12 2021].

[13] UNIVERSIDAD CATOLICA LOS ANGELES CHIMBOTE, «METODOLOGIA DE DESARROLLO DE SOFTWARE,» www.uladech.edu.pe, CHIMBOTE-PERÚ, 2017.

[14] D. d. Silva, «Blog de Zendesk,» Web Content & SEO Associate, 19 02 2021. [En línea]. Available: https://www.zendesk.com.mx/blog/metodologia-agil-que-es/. [Último acceso: 11 2021].

[15] E. ABELLÁN, «Scrum: qué es y cómo funciona esta metodología,» [En línea]. Available: https://www.wearemarketing.com/es/blog/metodologia-scrum-que-es-y-como-funciona.html. [Último acceso: 12 2021].

[16] I. Zabala, «Como recopilar requisitos para un proyecto,» 2019. [En línea]. Available: https://enredandoproyectos.com/recopilar-los-requisitos-de-un-proyecto/#:~:text=al%20estado%20futuro.-,La%20recopilaci%C3%B3n%20de%20Requisitos,proyecto%20para%20cumplir%20los%20objetivos.. [Último acceso: 12 2021].

[17] Universidad de Alicante, [En línea]. Available: https://si.ua.es/es/documentacion/asp-net-mvc-3/1-dia/modelo-vista-controlador-mvc.html#:~:text=Modelo%20Vista%20Controlador%20(MVC)%20es,control%20en%20tres%20componentes%20distintos.&text=La%20Vista%2C%20o%20interfaz%20de,los%20mecanismos%20interacci%C3. [Último acceso: 12 2021].

[18] okhosting, «Herramientas de Desarrollo de Software,» [En línea]. Available: https://okhosting.com/blog/herramientas-de-desarrollo-de-software/. [Último acceso: 12 2021].

[19] Capterra, «HEROKU,» [En línea]. Available: https://www.capterra.ec/software/158191/heroku. [Último acceso: 12 2021].

[20] Codigo facilito, «Curso de Nodejs,» [En línea]. Available: https://codigofacilito.com/articulos/que-es-mongoose. [Último acceso: 12 2021].

[21] izertis, [En línea]. Available: https://www.izertis.com/es/-/blog/encriptacion-de-password-en-nodejs-y-mongodb-bcrypt. [Último acceso: 12 2021].

[22] L. M. L. Magaña, «Qué es Json Web Token y cómo funciona,» 2020. [En línea]. Available: https://openwebinars.net/blog/que-es-json-web-token-y-como-funciona/. [Último acceso: 12 2021].

[23] C. Cano, «La Arquitectura REST,» [En línea]. Available: http://www.tsgroup.com.co/wps/portal/tsg/blog/detalle-blog/la-arquitectura-rest#:~:text=El%20estilo%20REST%20hace%20%C3%A9nfasis,recursos%20universales%20%C3%BAnicos%20(URI)..

[24] yeeply, «¿Qué son las pruebas unitarias y cómo llevar una a cabo?,» [En línea]. Available: https://www.yeeply.com/blog/que-son-pruebas-unitarias/. [Último acceso: 12 2021].

[25] ISTQB, «Pruebas de aceptación de software según el ISTQB,» [En línea]. Available: http://www.pmoinformatica.com/2016/08/pruebas-aceptacion-software-istqb.html. [Último acceso: 12 2021].

[26] L. Carvajal, Metodología de la Investgación Científica. Curso general y aplicado, 28 ed., Santiago de Cali: U.S.C., 2006, p. 139.

[27] C. d. D. OCDE, Impacto macroeconómico del COVID-19 en Ecuador: desafíos y respuestas, MAKING DEVELOPMENT HAPPEN volumen 5, 2020.

[28] R. G. Vasquez, «Noticias MU,» 24 Julio 2020. [En línea]. Available: https://www.umnews.org/es/news/la-pandemia-ha-impactado-hasta-lo-mas-sagrado. [Último acceso: 11 07 2021].

[29] I. R. Salvador, «Psicologia y mente,» [En línea]. Available: https://psicologiaymente.com/psicologia/estudio-de-caso. [Último acceso: 12 2021].

[30] SMARTBEAR, «SMARTBEAR,» [En línea]. Available: https://smartbear.com/learn/performance-monitoring/api-endpoints/. [Último acceso: 12 2021].

[31] J. Lucas, 04 09 2019. [En línea]. Available: https://openwebinars.net/blog/que-es-nodejs/. [Último acceso: 12 2021].

[32] MDN Web docs, «MDN Web docs,» [En línea]. Available: https://developer.mozilla.org/es/docs/Learn/Server-side/Express_Nodejs/Introduction. [Último acceso: 12 2021].

ANEXOS

A continuación, se presentan todos los Anexos que se han utilizado en el desarrollo del *backend*, los cuales se encuentran divididos de la siguiente manera:

- **ANEXO I.** Resultado del programa antiplagio *Turnitin***.**
- **ANEXO II.** Manual Técnico.
- **ANEXO III.** Manual de Usuario.
- **ANEXO IV.** Manual de Instalación.

ANEXO I

A continuación, se presenta el certificado que el Director de Tesis ha emitido y en donde se evidencia el resultado que se ha obtenido en la herramienta antiplagio *Turnitin*.

ESCUELA POLITÉCNICA NACIONAL
ESCUELA DE FORMACIÓN DE TECNÓLOGOS
CAMPUS POLITÉCNICO "ING. JOSÉ RUBÉN ORELLANA"

CERTIFICADO DE ORIGINALIDAD

Quito, de 14 febrero de 2022

De mi consideración:

Yo, Loarte Cajamarca Byron Gustavo, en calidad de Director del Trabajo de Integración Curricular titulado Desarrollo de un backend asociado al ***DESARROLLO DE UN SISTEMA PARA LA INTERACCIÓN DE LOS MIEMBROS DE LA COMUNIDAD DE LA IGLESIA IFGF*** elaborado por el estudiante ***MICHAEL ANDRIK GUANOLUISA QUIROZ*** de la carrera en ***TECNOLOGÍA SUPERIOR EN DESARROLLO DE SOFTWARE***, certifico que he empleado la herramienta Turnitin para la revisión de originalidad del documento escrito sesiones: Descripción del componente desarrollado, Metodología, Resultados, Conclusiones y Recomendaciones, del Trabajo de Integración Curricular indicado.

El documento escrito tiene un índice de similitud del 07%.

Es todo cuanto puedo certificar en honor a la verdad, pudiendo el interesado hacer uso del presente

documento para los trámites de titulación.

Atentamente,

Ing. Byron Loarte, MSc.
Profesor Ocasional
Escuela de Formación de Tecnólogos

ANEXO II

Recopilación de Requerimientos

A continuación, la **TABLA** presenta la Recopilación de requerimientos que se han obtenido al inicio del proyecto de acuerdo con lo solicitado por el dueño del producto.

TABLA I: Recopilación de requerimientos

RECOPILACIÓN DE REQUERIMIENTOS		
TIPO DEL SISTEMA	**ID - RR**	**ENUNCIADO DEL ÍTEM**
BACKEND	**RR003**	Como usuario administrador necesita generar a varios *endpoints* para: ● Ingresar información de la Iglesia (noticias, mensajes bíblicos y álbum de fotos). ● Visualizar información de la Iglesia (noticias, mensajes bíblicos y álbum de fotos). ● Modificar información de la Iglesia (noticias, mensajes bíblicos y álbum de fotos). ● Eliminar información de la Iglesia (noticias, mensajes bíblicos y álbum de fotos).
	RR004	Como usuario administrador necesita generar varios *endpoints* para realizar lo siguiente: ● Registrar eventos. ● Visualizar eventos. ● Modificar eventos. ● Eliminar eventos.
	RR005	Como usuario administrador necesita generar a un *endpoint* visualizar los miembros inscritos a un evento.
	RR006	Como usuario administrador necesita generar un *endpoint* para visualizar las donaciones realizadas.

	RR007	Como usuario administrador necesita generar varios *endpoints* para realizar lo siguiente: ● Registrar vídeos. ● Visualizar vídeos. ● Modificar vídeos. ● Eliminar vídeos.
	RR008	Como usuario administrador necesita generar varios *endpoints* para realizar lo siguiente: ● Registrar cuestionarios. ● Visualizar cuestionarios. ● Modificar cuestionarios. ● Eliminar cuestionarios.
	RR009	Como usuario invitado necesita generar un *endpoint* para registrarse en el sistema *web*.
	RR010	Como usuario miembro de la Iglesia necesita generar varios *endpoints* para realizar lo siguiente: ● Realizar donación de dinero. ● Realizar donación de ropa. ● Realizar donación de alimentos.
	RR011	Como usuario miembro de la Iglesia necesita generar varios *endpoints* para realizar lo siguiente: ● Visualizar eventos creados. ● Inscribirse a un evento. ● Eliminar inscripción al evento.
	RR012	Como usuario miembro de la Iglesia necesita generar a varios *endpoints* para realizar lo siguiente: ● Visualizar cuestionarios. ● Realizar cuestionarios. ● Visualizar puntaje.

	RR013	Como usuario miembro de la Iglesia necesita generar un *endpoint* para realizar lo siguiente: ● Visualizar vídeos.

Historias de Usuario

Culminada la etapa de recopilación de requerimientos, se procede a desarrollar las Historias de Usuario, para el *backend*. A continuación, se presentan las 12 Historias de Usuario escritas en base a los requerimientos del proyecto que va desde la **TABLA X**: Historia de usuario Nro.2 a la **TABLA XX**: Historia de usuario Nro.12**.**

TABLA X: Historia de usuario Nro.2

Historia de Usuario	
Identificador: HU002	**Usuario:** Administrador y miembros de la Iglesia
Nombre historia: Generar varios *endpoints* para iniciar y cerrar sesión.	
Prioridad en Negocio: Medio	**Riesgo en Negocio:** Medio
Iteración asignada: 1	
Responsable: Michael Guanoluisa	
Descripción: El *backend* por medio del perfil asignado permite generar varios *endpoints* para iniciar y cerrar sesión. Además, estos usuarios se pueden identificar por medio de un usuario y contraseña a través de cualquier aplicación por el lado del cliente para gestionar los módulos a su cargo.	
Observación: Los usuarios tienen que iniciar sesión para poder acceder al resto de *endpoints* protegidos.	

TABLA XI: Historia de usuario Nro.3

Historia de Usuario	
Identificador: HU003	**Usuario:** Administrador
Nombre historia: Generar varios *endpoints* para gestionar la información de la Iglesia.	
Prioridad en Negocio: Medio	**Riesgo en Negocio:** Medio
Iteración asignada: 1	
Responsable: Michael Guanoluisa	
Descripción: El usuario administrador en el *backend* puede generar varios *endpoints* para ingresar, visualizar, modificar y eliminar información respecto a: • Noticias. • Mensajes bíblicos. • Álbum de fotos La estructura de esta información esta dada por la siguiente: • Título. • Descripción. • Imagen	
Observación: El usuario administrador es el único que puede acceder a los *endpoints* mencionados anteriormente, es decir necesita iniciar sesión para gestionar la información.	

TABLA XII: Historia de usuario Nro.4

Historia de Usuario	
Identificador: HU004	**Usuario:** Administrador
Nombre historia: Generar varios *endpoints* para la gestión de eventos.	

Prioridad en Negocio (Alto/Medio/Bajo): Medio	**Riesgo en Negocio (Alto/Medio/Bajo):** Medio
Iteración asignada: 1	
Responsable: Michael Guanoluisa	
Descripción: El *backend* por medio del perfil asignado permite generar varios *endpoints* que le permiten gestionar los eventos de la Iglesia como: ● Registrar eventos. ● Visualizar eventos. ● Modificar eventos. ● Eliminar eventos. sin embargo, toda esta información puede ser gestionada a través de cualquier aplicación por el lado del cliente.	
Observación: El usuario administrador es el único que puede acceder a los *endpoints* mencionados anteriormente, es decir necesita iniciar sesión para gestionar la información.	

TABLA XIII: Historia de usuario Nro.5

	Historia de Usuario
Identificador: HU005	**Usuario:** Administrador
Nombre historia: Generar un *endpoint* para visualizar las donaciones realizadas.	
Prioridad en Negocio (Alto/Medio/Bajo): Medio	**Riesgo en Negocio (Alto/Medio/Bajo):** Bajo
Iteración asignada: 1	
Responsable: Michael Guanoluisa	

Descripción: El *backend* por medio del perfil asignado permite generar un *endpoint* que le permite visualizar las donaciones realizadas como: ● Dinero ● Ropa ● Alimentos. sin embargo, toda esta información puede ser gestionada a través de cualquier aplicación por el lado del cliente.
Observación: El usuario administrador identificado es el único que puede acceder al *endpoint* mencionado anteriormente.

TABLA XIV: Historia de usuario Nro.6

	Historia de Usuario
Identificador: HU006	**Usuario:** Administrador
Nombre historia: Generar a varios *endpoints* para gestionar vídeos.	
Prioridad en Negocio (Alto/Medio/Bajo): Medio	**Riesgo en Negocio (Alto/Medio/Bajo):** Medio
Iteración asignada: 1	
Responsable: Michael Guanoluisa	
Descripción: El *backend* por medio del perfil asignado permite generar varios *endpoints* que le permiten gestionar los vídeos de la Iglesia como: ● Registrar vídeos. ● Visualizar vídeos. ● Modificar vídeos. ● Eliminar vídeos.	

sin embargo, toda esta información puede ser gestionada a través de cualquier aplicación por el lado del cliente.
Observación: El usuario administrador es el único que puede acceder a los *endpoints* mencionados anteriormente.

TABLA XV: Historia de usuario Nro.7

Historia de Usuario	
Identificador: HU007	**Usuario:** Administrador
Nombre historia: Generar varios *endpoints* para gestionar cuestionarios.	
Prioridad en Negocio (Alto/Medio/Bajo): Medio	**Riesgo en Negocio (Alto/Medio/Bajo):** Medio
Iteración asignada: 1	
Responsable: Michael Guanoluisa	
Descripción: El *backend* por medio del perfil asignado permite generar varios *endpoints* que le permiten gestionar los cuestionarios de la Iglesia como: ● Registrar cuestionarios. ● Visualizar cuestionarios. ● Modificar cuestionarios. ● Eliminar cuestionarios. sin embargo, toda esta información puede ser gestionada a través de cualquier aplicación por el lado del cliente.	
Observación: El usuario administrador identificado es el único que puede acceder a los *endpoints* mencionados anteriormente.	

TABLA XVI: Historia de usuario Nro.8

Historia de Usuario	
Identificador: HU008	**Usuario:** Invitado
Nombre historia: Generar un *endpoint* para registrarse.	
Prioridad en Negocio (Alto/Medio/Bajo): Medio	**Riesgo en Negocio (Alto/Medio/Bajo):** Alto
Iteración asignada: 1	
Responsable: Michael Guanoluisa	
Descripción: El *backend* por medio del perfil asignado permite generar un *endpoint* que le permita iniciar sesión el usuario invitado por medio de los siguientes campos: ● Nombre. ● Apellido. ● Edad. ● Correo. ● Celular. sin embargo, toda esta información puede ser gestionada a través de cualquier aplicación por el lado del cliente.	
Observación: El backend le permite un registro que le identifica como usuario miembro de la Iglesia.	

TABLA XVII: Historia de usuario Nro.9

Historia de Usuario	
Identificador: HU009	**Usuario:** Miembros de la Iglesia
Nombre historia: Generar varios *endpoints* para realizar donaciones.	

Prioridad en Negocio (Alto/Medio/Bajo): Medio	**Riesgo en Negocio (Alto/Medio/Bajo):** Alto
Iteración asignada: 2	
Responsable: Michael Guanoluisa	
Descripción: El *backend* por medio del perfil asignado permite generar varios *endpoints* que le permiten gestionar las donaciones de la Iglesia como: ● Donación de dinero. ● Donación de ropa. ● Donación de alimentos. sin embargo, toda esta información puede ser gestionada a través de cualquier aplicación por el lado del cliente.	
Observación: El usuario miembro de la Iglesia identificado es el único que puede acceder a los *endpoints* mencionados anteriormente.	

TABLA XVIII: Historia de usuario Nro.10

	Historia de Usuario
Identificador: HU010	**Usuario:** Miembros de la Iglesia
Nombre historia: Generar varios *endpoints* para la gestión de eventos.	
Prioridad en Negocio (Alto/Medio/Bajo): Medio	**Riesgo en Negocio (Alto/Medio/Bajo):** Alto
Iteración asignada: 2	
Responsable: Michael Guanoluisa	
Descripción:	

El *backend* por medio del perfil asignado permite generar varios *endpoints* que le permiten gestionar los eventos de la Iglesia como: ● Visualizar eventos creados. ● Inscribirse a un evento. ● Eliminar inscripción al evento. sin embargo, toda esta información puede ser gestionada a través de cualquier aplicación por el lado del cliente.
Observación: El usuario miembro de la Iglesia identificado es el único que puede acceder a los *endpoints* mencionados anteriormente.

TABLA XIX: Historia de usuario Nro.11

Historia de Usuario	
Identificador: HU011	**Usuario:** Miembros de la Iglesia
Nombre historia: Generar varios endpoints para la gestión de cuestionarios.	
Prioridad en Negocio (Alto/Medio/Bajo): Medio	**Riesgo en Negocio (Alto/Medio/Bajo):** Alto
Iteración asignada: 2	
Responsable: Michael Guanoluisa	
Descripción: El *backend* por medio del perfil asignado permite generar varios *endpoints* que le permitan gestionar los cuestionarios de la Iglesia como: ● Visualizar cuestionarios. ● Realizar cuestionarios. ● Visualizar puntaje. sin embargo, toda esta información puede ser gestionada a través de cualquier aplicación por el lado del cliente.	

Observación: El usuario miembro de la Iglesia es el único que puede acceder a los *endpoints* mencionados anteriormente.

TABLA XX: Historia de usuario Nro.12

Historia de Usuario	
Identificador: HU012	**Usuario:** Miembros de la Iglesia
Nombre historia: Generar un *endpoint* para visualizar vídeos.	
Prioridad en Negocio (Alto/Medio/Bajo): Medio	**Riesgo en Negocio (Alto/Medio/Bajo):** Alto
Iteración asignada: 2	
Responsable: Michael Guanoluisa	
Descripción: El *backend* por medio del perfil asignado permite generar un *endpoint* que le permitan enviar información de los vídeos de la Iglesia. Sin embargo, toda esta información puede ser gestionada a través de cualquier aplicación por el lado del cliente.	
Observación: El usuario miembro de la Iglesia es el único que puede acceder a los *endpoints* mencionados anteriormente.	

Product Backlog

En la **TABLA XXI** enumera la prioridad de cada requisito que se ha implementado en el *backend*. Estos requisitos se clasifican de acuerdo con las necesidades del dueño del producto y la complejidad del desarrollo.

TABLA XXI: *Product Backlog*

ELABORACIÓN DEL *PRODUCT BACKLOG*				
ID-HU	**Historia de Usuario**	**Iteración**	**Prioridad**	**Estado**
HU003	Generar varios *endpoints* para gestionar información general de la Iglesia.	1	Medio	Terminado
HU004	Generar varios *endpoints* para la gestión de eventos.	1	Medio	Terminado
HU005	Generar un *endpoint* para visualizar las donaciones realizadas.	1	Medio	Terminado
HU006	Generar varios *endpoints* para gestionar vídeos.	1	Medio	Terminado
HU007	Generar varios *endpoints* para gestionar cuestionarios.	1	Medio	Terminado
HU008	Generar un *endpoint* para registrarse.	1	Alta	Terminado
HU009	Generar varios *endpoints* para realizar donaciones.	2	Alta	Terminado
HU010	Generar varios *endpoints* para interactuar con eventos.	2	Medio	Terminado
HU011	Generar varios *endpoints* para interactuar con cuestionarios.	2	Medio	Terminado
HU012	Generar un *endpoint* para visualizar vídeos.	2	Medio	Terminado

SPRINT BACKLOG

La **TABLA XXII** presenta los cinco *Sprints* en los que se ha desarrollo el *backend*, listando las actividades y el tiempo determinado para cumplir con los entregables que se han establecido con el dueño del producto.

TABLA XXII: *Sprint Backlog*

ELABORACIÓN DEL *SPRINT BACKLOG*						
ID – SB	**NOMBRE**	**Recursos**	**ID-HU**	**HISTORIA DE USUARIO**	**TAREAS**	**TIEMPO ESTIMADO**
SB001	Diseño e implementación del *endpoint* para los usuarios con perfil invitado y administrador	Sección informativa	HU001	Generar varios *endpoints* para mostrar información de la Iglesia.	• Implementación de *endpoints* de tipo público que devuelven la información como: noticias, mensajes bíblicos y álbumes de fotos.	70 H
		Inicio de sesión	HU002	Generar varios *endpoints* para iniciar y cerrar sesión.	• Diseño e implementación del *endpoint* para el inicio de sesión. • Diseño e implementación del *endpoint* para el cierre de sesión.	

					● Verificación del consumo por parte del cliente REST. ● Consulta a la base de datos y autorización. ● Cargar los módulos asignados.	
		Gestionar información general	HU003	Generar varios *endpoints* para gestionar información general de la Iglesia.	● Diseño e implementación de *endpoints* para registrar, visualizar, actualizar y eliminar información. ● Consulta en la Base de datos. ● requeridos.	
		Eventos	HU004	Generar varios *endpoints* para la gestión de eventos.	● Diseño e implementación de *endpoints* para registrar, visualizar, actualizar y eliminar eventos. ● Consulta en la Base de datos. ● Validación de los datos requeridos.	

		Donaciones	HU005	Generar un *endpoint* para visualizar las donaciones realizadas.	● Diseño e implementación de *endpoint* para visualizar donaciones realizadas. ● Consulta en la Base de datos. ● Validación de los datos requeridos.	
		Vídeos	HU006	Generar varios *endpoints* para gestionar vídeos.	● Diseño e implementación de *endpoints* para registrar, visualizar, actualizar y eliminar vídeos. ● Consulta en la Base de datos. ● Validación de los datos requeridos.	
		Cuestionarios	HU007	Generar varios *endpoints* para gestionar cuestionarios.	● Diseño e implementación de *endpoints* para registrar, visualizar, actualizar y eliminar cuestionarios. ● Consulta en la Base de datos. ● Validación de los datos requeridos.	

		Registro de usuario	HU008	Generar un *endpoint* para registrarse.	● Diseño e implementación de *endpoint* para el registro de usuario. ● Validación de los datos requeridos. ● Verificación que el registro sea único.	
SB002	Diseño e implementación del *endpoint* para los usuarios con perfil miembro de la Iglesia	Módulo – donaciones	HU009	Generar varios *endpoints* para realizar donaciones.	● Diseño e implementación de *endpoints* para realizar donaciones. ● Validación de los datos requeridos. ● Verificación del tipo de donación.	70 H
		Módulo – Eventos	HU010	Generar varios *endpoints* para interactuar con eventos.	● Diseño e implementación de *endpoints* para registrar, visualizar y eliminar inscripción en un evento. ● Validación de datos requeridos.	

		Módulo – Cuestionarios	HU011	Generar varios *endpoints* para interactuar con cuestionarios.	• Diseño e implementación de un *endpoint* para visualizar cuestionario. • Diseño e implementación de un *endpoint* para almacenar nombre y puntaje de los jugadores. • Validación de los datos requeridos.	
		Módulo - Reportes de acoso	HU012	Generar un *endpoint* para visualizar vídeos.	• Diseño e implementación de un *endpoint* para visualizar información de vídeos.	
SB003	Pruebas en el *backend*				• Pruebas unitarias • Pruebas de rendimiento. • Prueba de aceptación.	20 H
SB004	Despliegue del *backend* y Base de Datos NoSQL				• Despliegue de la Base de Datos NoSQL en *MongoDB Atlas*	10 H

		• Despliegue del *backend* en *Heroku*.	
Documentación		• Informe Técnico. • Anexos.	50 H
TOTAL			**240 H**

Diseño de la Base de datos NoSQL

La **Fig. 24**, ilustra las siete colecciones que han sido necesarias en el desarrollo de cada uno de los *endpoints* del *backend*, manteniendo así la información mejor organizada y permitiendo que las consultas sean más eficientes.

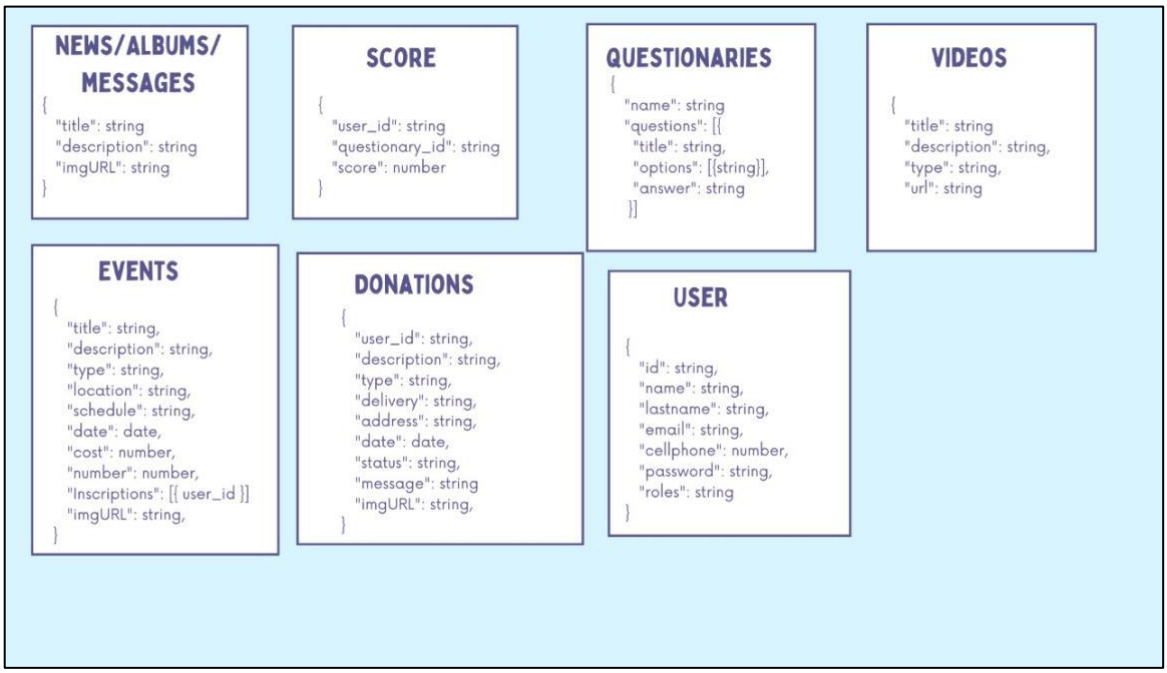

Fig. 24: Diseño de la Base de datos no Relacional

Pruebas

Terminada la etapa de codificación se ha desarrollado la ejecución de pruebas unitarias, de compatibilidad y de aceptación para comprobar la calidad del código del *backend* y cada uno de los módulos respectivamente.

Pruebas unitarias

A continuación, en la **Fig. 25** se muestra el código para probar el registro de un nuevo usuario. Por otra parte, la **Fig. 26** se prueba la respuesta del *backend* para la creación de un dato con respuesta aceptada. En cambio, la **Fig. 27** se prueba el método de creación con una respuesta de *token* invalido, esto para las rutas que son protegidas. De la misma manera, en la **Fig. 28** se prueba el método *GET* para obtener los datos almacenados, en este caso con una respuesta *OK* o 200 que devuelve el *backend*. Por último, la **Fig. 29** se prueba el método *GET* para obtener los datos almacenados, en este caso con una respuesta 404 de que no encontró el dato en el momento de que se envíe un id de manera incorrecta.

```
});

it("Post register respond a json with message user registered", (done) => {
  data = {
    name: "michael",
    lastname: "guanoluisa",
    email: `${faker.name.firstName()}+ @hotmail.com`,
    cellphone: "0961393801",
    password: "12341234",
  };
  request(app)
    .post("/api/auth/register")
    .send(data)
    .set("Accept", "application/json")
    .expect("Content-Type", /json/)
    .expect(200)
    .expect('{"message":"Usuario registrado con exito"}')
    .end(done);
});
```

Fig. 25: Código de prueba al método *register*

```
it("Post news with a json", (done) => {
  data = {
    title: faker.datatype.string(),
    description: faker.datatype.string(),
    imgURL: "ifgf.png",
  };
  request(app)
    .post("/api/news")
    .send(data)
    .set("Accept", "application/json")
    .set(
      "x-access-token",
      "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI
    )
    .expect("Content-Type", /json/)
    .expect(201)
    .end(done);
});
```

Fig. 26: Código de respuesta para la creación de datos, con respuesta 201

```
it("Post news with a json", (done) => {
  data = {
    title: faker.datatype.string(),
    description: faker.datatype.string(),
    imgURL: "ifgf.png",
  };
  request(app)
    .post("/api/news")
    .send(data)
    .set("Accept", "application/json")
    .expect("Content-Type", /json/)
    .expect(403)
    .expect('{"message":"Inicie sesión"}')
    .end(done);
});
```

Fig. 27: Código de prueba a la creación con respuesta de error por *token* incorrecto

```
it("Get News respond a json with contain news", (done) => {
  request(app)
    .get("/api/news")
    .set("Accept", "application/json")
    .expect("Content-Type", /json/)
    .expect(200, done);
});
```

Fig. 28: Código prueba al método obtener los datos, con respuesta 200

```
it("Get a news by id respond 404 not found", (done) => {
  id = "61c10af1c76fd906ecba0634";
  request(app)
    .get(`/api/news/${id}`)
    .set("Accept", "application/json")
    .expect("Content-Type", /json/)
    .expect(404, done);
});
```

Fig. 29: Código prueba al método obtener datos con un id incorrecto

Una vez que se han establecido el conjunto de pruebas y el código necesario se ejecuta todas las pruebas y obtener el resultado de cada una de ellas, como se ilustra en la **Fig. 30** en donde se puede apreciar la ejecución de forma exitosa de cada una de ellas.

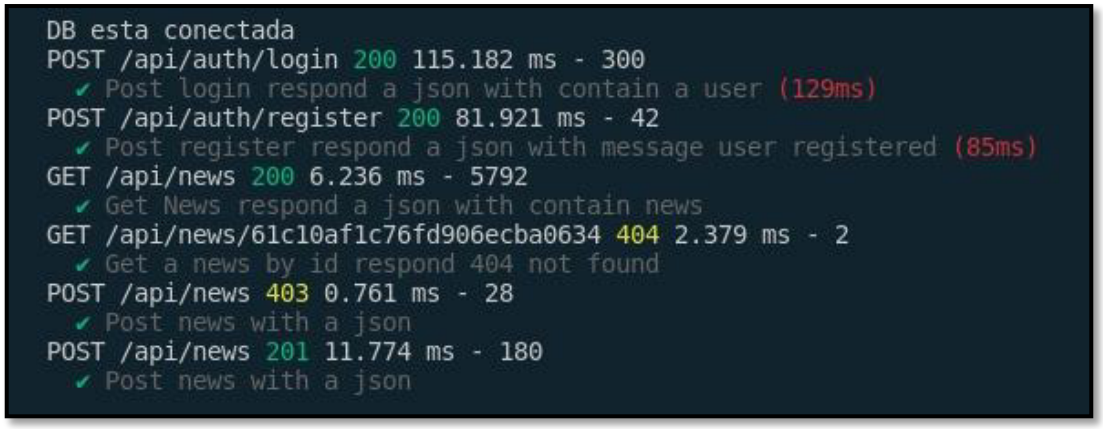

Fig. 30: Resultado exitoso de cada prueba

Prueba de Compatibilidad

A continuación, se presentan las pruebas de compatibilidad, en este caso utilizando los clientes HTTP que son *Postman* y *Thunder Client*. Cabe resaltar que en el presente proyecto específicamente para la sección de Resultados del presente documento se ha presentado cada uno de los resultados de los endpoints con Postman, pero para evidenciar esta prueba se ejecuta el método de iniciar sesión como se ilustra en la **Fig. 31.**

Por otra parte, *Thunder Client* se ha utilizado como extensión de *Visual Studio Code, para* ejecutar cada una de las pruebas, en ese sentido se ha probado el método *GET* para obtener noticias como se presenta en la **Fig. 32**, y de igual forma el inicio de sesión como se muestra en la **Fig. 33.**

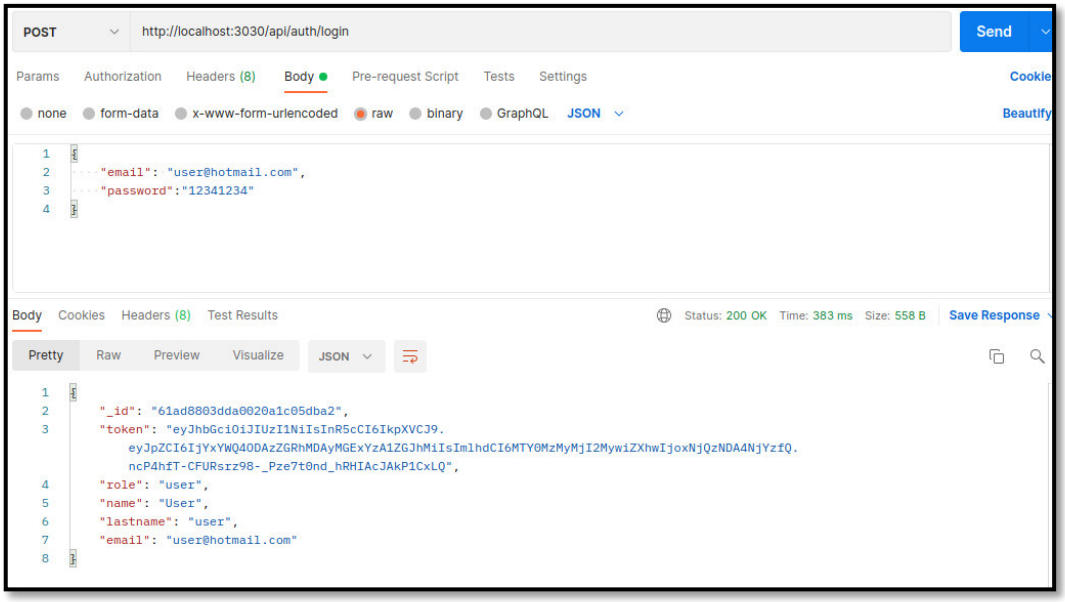

Fig. 31: E*ndpoint* para probar el inicio de sesión

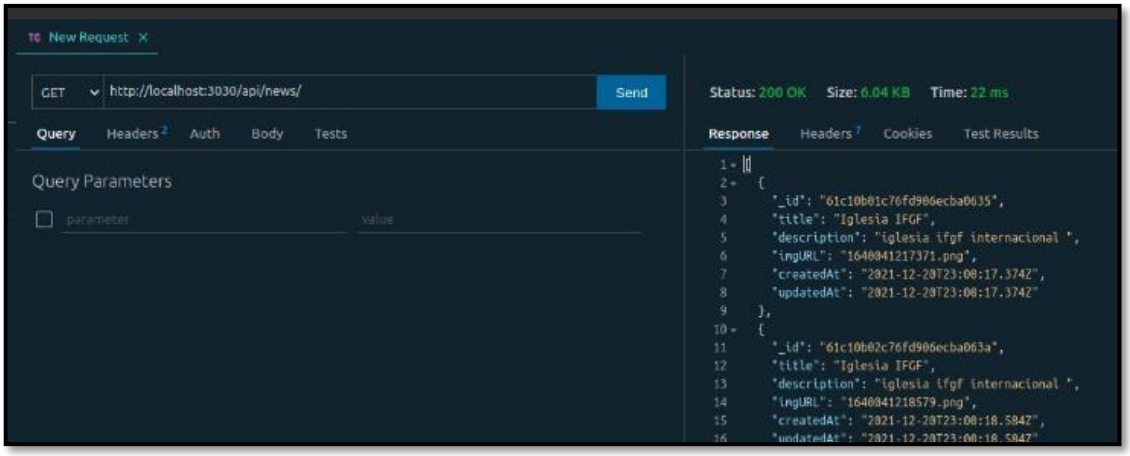

Fig. 32: *Endpoint* para probar la obtención de noticias

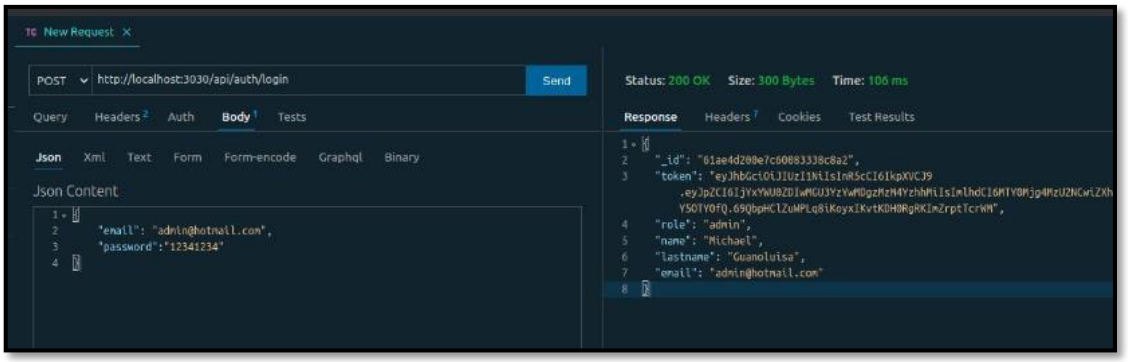

Fig. 33: *Endpoint* para probar el inicio de sesión

Pruebas de aceptación

A continuación, se muestran las 11 Pruebas de aceptación que van desde la **TABLA XXIII**: Prueba de aceptación Nro. 2 hasta **TABLA XXXIII**: Prueba de aceptación 12 – Generar *endpoints* para visualizar vídeos. Además, cada prueba describe el procedimiento que se le ha asignado a los diferentes tipos de usuario para la correcta ejecución, comprobación y a su vez la aprobación de cada una.

TABLA XXIII: Prueba de aceptación Nro. 2

Prueba de Aceptación	
Identificador: PA002	**Identificador historia de Usuario:** HU002
Nombre: Generar varios *endpoints* para iniciar y cerrar sesión.	
Descripción: El *backend* por medio del perfil asignado permite generar varios *endpoints* para iniciar y cerrar sesión. Además, estos usuarios se pueden identificar por medio de un usuario y contraseña a través de cualquier aplicación por el lado del cliente para gestionar los módulos a su cargo.	
Pasos de ejecución: • Acceder a las rutas generadas por el backend, las mismas que son públicas. • Llamar al método *POST* de *login*, enviando sus credenciales en formato *JSON*. • El backend genera la respuesta con los datos del usuario autenticado.	
Resultado deseado: El *backend* le permite al usuario generar *endpoints* con credenciales de usuario, que le permitan autenticarse.	
Evaluación de la prueba: El resultado es el esperado y se tiene la aceptación completa del cliente.	

TABLA XXIV: Prueba de aceptación 3 – Generar *endpoints* para gestionar información de la Iglesia

	Prueba de Aceptación
Identificador: PA003	**Identificador historia de Usuario:** HU003.
Nombre: Generar varios *endpoints* para gestionar la información de la Iglesia.	
Descripción: El *backend* por medio del perfil asignado permite generar varios *endpoints* que le permiten gestionar la información general de la Iglesia como: ● Ingresar información de la Iglesia. ● Visualizar información de la Iglesia. ● Modificar información de la Iglesia. ● Eliminar información de la Iglesia. sin embargo, toda esta información puede ser gestionada a través de cualquier aplicación por el lado del cliente.	
Pasos de ejecución: • Acceder a las rutas generadas por el *backend*, las mismas que son protegidas. • Llamar al método *POST* de *login* para poder autenticarse y acceder a estas rutas protegidas. • Llamar a los métodos CRUD para la gestión de la información como: *POST, PUT, GET y DELETE.* • El *backend* genera la respuesta con los datos de la base de datos.	
Resultado deseado: El *backend* le permite al usuario administrador generar *endpoints* para gestionar la informacion de la Iglesia, los mismos que le permiten crear, visualizar, actualizar y borrar estos datos.	
Evaluación de la prueba: El resultado es el esperado y se tiene la aceptación completa del cliente.	

TABLA XXV: Prueba de aceptación 4 – Generar *endpoints* para gestionar eventos

Prueba de Aceptación	
Identificador: PA004	**Identificador historia de Usuario:** HU004.
Nombre: Generar varios *endpoints* para la gestión de eventos.	
Descripción: El *backend* por medio del perfil asignado permite generar varios *endpoints* que le permiten gestionar los eventos de la Iglesia como: ● Registrar eventos. ● Visualizar eventos. ● Modificar eventos. ● Eliminar eventos. Sin embargo, toda esta información puede ser gestionada a través de cualquier aplicación por el lado del cliente.	
Pasos de ejecución: • Acceder a las rutas generadas por el *backend*, las mismas que son protegidas. • Llamar al método *POST* de *login* para poder autenticarse y acceder a estas rutas protegidas. • Llamar a los métodos CRUD para la gestión de los eventos como: *POST, PUT, GET y DELETE.* • El *backend* genera la respuesta con los datos de la base de datos.	
Resultado deseado: El *backend* le permite al usuario administrador generar *endpoints* para gestionar los eventos, los mismos que le permiten crear, visualizar, actualizar y borrar estos datos.	
Evaluación de la prueba: El resultado es el esperado y se tiene la aceptación completa del cliente.	

TABLA XXVI: Prueba de aceptación 5 – Generar *endpoints* para iniciar sesión

Prueba de Aceptación	
Identificador: PA005	**Identificador historia de Usuario:** HU005.
Nombre: Generar un *endpoint* para visualizar las donaciones realizadas.	
Descripción: El *backend* por medio del perfil asignado permite generar un *endpoint* que le permite visualizar las donaciones realizadas como: ● Dinero ● Ropa ● Alimentos. sin embargo, toda esta información puede ser gestionada a través de cualquier aplicación por el lado del cliente.	
Pasos de ejecución: • Acceder a la ruta generada por el *backend*, la misma que es protegida. • Llamar al método *POST* de *login* para poder autenticarse y acceder a esta ruta protegida. • Llamar al método *GET y PUT,* los mismos que le permiten ver las donaciones y actualizar su estado de aceptado ha rechazado, permitiendo postear un mensaje hacia el usuario que realizo esta donación. • El *backend* genera la respuesta con los datos de la base de datos.	
Resultado deseado: El *backend* le permite al usuario administrador generar *endpoints* con los datos de visualización y actualizar de donaciones de la Iglesia, permitiéndole aceptar o rechazar una donación.	
Evaluación de la prueba: El resultado es el esperado y se tiene la aceptación completa del cliente.	

TABLA XXVII: Prueba de aceptación 6 – Generar *endpoints* para gestionar vídeos

Prueba de Aceptación	
Identificador: PA006	**Identificador historia de Usuario:** HU006.
Nombre: Generar a varios *endpoints* para gestionar vídeos.	
Descripción: El *backend* por medio del perfil asignado permite generar varios *endpoints* que le permiten gestionar los vídeos de la Iglesia como: ● Registrar vídeos. ● Visualizar vídeos. ● Modificar vídeos. ● Eliminar vídeos. sin embargo, toda esta información puede ser gestionada a través de cualquier aplicación por el lado del cliente.	
Pasos de ejecución: • Acceder a las rutas generadas por el *backend*, las mismas que son protegidas. • Llamar al método *POST* de *login* para poder autenticarse y acceder a estas rutas protegidas. • Llamar a los métodos CRUD para la gestión de los vídeos como: *POST, PUT, GET y DELETE.* • El *backend* genera la respuesta con los datos de la base de datos.	
Resultado deseado: El *backend* le permite al usuario administrador generar *endpoints* para gestionar los vídeos, los mismos que le permiten crear, visualizar, actualizar y borrar estos datos.	
Evaluación de la prueba: El resultado es el esperado y se tiene la aceptación completa del cliente.	

TABLA XXVIII: Prueba de aceptación 7 – Generar *endpoints* para gestionar cuestionarios

<table>
<tr><th colspan="2">Prueba de Aceptación</th></tr>
<tr><td>Identificador: PA007</td><td>Identificador historia de Usuario: HU007.</td></tr>
<tr><td colspan="2">Nombre: Generar varios endpoints para gestionar cuestionarios.</td></tr>
<tr><td colspan="2">Descripción:

El backend por medio del perfil asignado permite generar varios endpoints que le permiten gestionar los cuestionarios de la Iglesia como:

● Registrar cuestionarios.
● Visualizar cuestionarios.
● Modificar cuestionarios.
● Eliminar cuestionarios.

sin embargo, toda esta información puede ser gestionada a través de cualquier aplicación por el lado del cliente.</td></tr>
<tr><td colspan="2">Pasos de ejecución:

• Acceder a las rutas generadas por el backend, las mismas que son protegidas.

• Llamar al método POST de login para poder autenticarse y acceder a estas rutas protegidas.

• Llamar a los métodos CRUD para la gestión de los cuestionarios como: post, PUT, GET y DELETE.

• El backend genera la respuesta con los datos de la base de datos.</td></tr>
<tr><td colspan="2">Resultado deseado:

El backend le permite al usuario administrador generar endpoints para gestionar los cuestionarios, los mismos que le permiten crear, visualizar, actualizar y borrar estos datos.</td></tr>
<tr><td colspan="2">Evaluación de la prueba:</td></tr>
</table>

El resultado es el esperado y se tiene la aceptación completa del cliente.

TABLA XXIX: Prueba de aceptación 8 – Generar *endpoints* para registrarse

Prueba de Aceptación	
Identificador: PA008	**Identificador historia de Usuario:** HU008.
Nombre: Generar un *endpoint* para registrarse.	
Descripción: El *backend* por medio del perfil asignado permite generar un *endpoint* que le permita iniciar sesión el usuario invitado por medio de los siguientes campos: ● Nombre. ● Apellido. ● Edad. ● Correo. ● Celular. sin embargo, toda esta información puede ser gestionada a través de cualquier aplicación por el lado del cliente.	
Pasos de ejecución: • Acceder a las rutas generadas por el *backend*, las mismas que son públicas. • Llamar al método *POST* de *register,* el mismo que le permite al usuario por medio del envío de un *json* con los datos personales requeridos registrarse en la base de datos. • El *backend* genera la respuesta con los datos del usuario registrado.	
Resultado deseado: El *backend* le permite al usuario generar *endpoints* que le permite registrarse con el envío de sus datos personales.	
Evaluación de la prueba: El resultado es el esperado y se tiene la aceptación completa del cliente.	

TABLA XXX: Prueba de aceptación 9 – Generar *endpoints* para realizar donaciones

	Prueba de Aceptación
Identificador: PA009	**Identificador historia de Usuario:** HU009.
Nombre: Generar varios *endpoints* para realizar donaciones.	
Descripción: El *backend* por medio del perfil asignado permite generar varios *endpoints* que le permiten gestionar las donaciones de la Iglesia como: • Donación de dinero. • Donación de ropa. • Donación de alimentos. sin embargo, toda esta información puede ser gestionada a través de cualquier aplicación por el lado del cliente.	
Pasos de ejecución: • Acceder a las rutas generadas por el *backend*, las mismas que son protegidos. • Llamar al método *POST* de *login* para poder autenticarse y acceder a estas rutas protegidas. • Llamar al método *POST* de donaciones, que le permite al usuario registrado y autenticado crear una donación. • El *backend* genera la respuesta con los datos de la base de datos.	
Resultado deseado: El *backend* le permite al usuario miembro de la Iglesia generar un *endpoint* para la creación de donaciones.	
Evaluación de la prueba: El resultado es el esperado y se tiene la aceptación completa del cliente.	

TABLA XXXI: Prueba de aceptación 10 – Generar *endpoints* para la gestión de eventos

Prueba de Aceptación	
Identificador: PA010	**Identificador historia de Usuario:** HU010
Nombre: Generar varios *endpoints* para la gestión de eventos.	
Descripción: El *backend* por medio del perfil asignado permite generar varios *endpoints* que le permiten gestionar los eventos de la Iglesia como: ● Visualizar eventos creados. ● Inscribirse a un evento. ● Eliminar inscripción al evento. sin embargo, toda esta información puede ser gestionada a través de cualquier aplicación por el lado del cliente.	
Pasos de ejecución: • Acceder a las rutas generadas por el *backend*, las mismas que son protegidas. • Llamar al método *POST* de *login* para poder autenticarse y acceder a estas rutas protegidas. • Llamar a los métodos *GET, PUT y DELETE* de inscripciones a eventos. • El *backend* genera la respuesta con los datos de la base de datos.	
Resultado deseado: El *backend* le permite al usuario miembro de la Iglesia generar *endpoints* para visualizar, inscribirse y borrar inscripción a un evento que haya registrado el usuario administrador.	

TABLA XXXII: Prueba de aceptación 11 – Generar *endpoints* para la gestión de cuestionarios

Prueba de Aceptación	
Identificador: PA011	**Identificador historia de Usuario:** HU011
Nombre: Generar varios *endpoints* para la gestión de cuestionarios.	
Descripción: El backend por medio del perfil asignado permite generar varios *endpoints* que le permitan gestionar los cuestionarios de la Iglesia como: ● Visualizar cuestionarios. ● Realizar cuestionarios. ● Visualizar puntaje. sin embargo, toda esta información puede ser gestionada a través de cualquier aplicación por el lado del cliente.	
Pasos de ejecución: • Acceder a las rutas generadas por el *backend*, las mismas que son protegidas. • Llamar al método *POST* de *login* para poder autenticarse y acceder a estas rutas protegidas. • Llamar a los métodos *GET y post* de cuestionario y puntaje, para visualizar cuestionarios y guardar puntaje al responder. • El *backend* genera la respuesta con los datos de la base de datos.	
Resultado deseado: El *backend* le permite al usuario miembro de la Iglesia generar *endpoints* para visualizar cuestionarios y almacenar su puntaje al haber contestado tales cuestionarios.	
Evaluación de la prueba: El resultado es el esperado y se tiene la aceptación completa del cliente.	

TABLA XXXIII: Prueba de aceptación 12 – Generar *endpoints* para visualizar vídeos

Prueba de Aceptación	
Identificador: PA012	**Identificador historia de Usuario:** HU012
Nombre: Generar un *endpoint* para visualizar vídeos.	
Descripción: El *backend* por medio del perfil miembro de la Iglesia le permite visualizar vídeos, llamando al método *GET* el mismo que genera el *endpoint* con los vídeos almacenados en la base de datos.	
Pasos de ejecución: • Acceder a las rutas generadas por el *backend*, las mismas que son protegidas. • Llamar al método *POST* de *login* para poder autenticarse y acceder a estas rutas protegidas. • Llamar a los métodos *GET* de vídeos, para visualizar vídeos. • El *backend* genera la respuesta con los datos de la base de datos.	
Resultado deseado: El *backend* le permite al usuario miembro de la Iglesia generar un *endpoint* con los datos de visualización de vídeos.	
Evaluación de la prueba: El resultado es el esperado y se tiene la aceptación completa del cliente.	

Certificado de Iglesia IFGF

IFGF

Iglesia Fraternidad Internacional del Evangelio Completo GISI (IFGF)

Quito, 10 de febrero del 2022

CERTIFICADO

Yo, Walter Hidalgo, con CC. 1707795173 como pastor de Iglesia Fraternidad Internacional del Evangelio Completo IFGF GISI de Quito – Ecuador.

Por medio del presente certifico:

Que el Sr. **Michael Andrik Guanoluisa Quiroz** con cédula de ciudadanía **131535684-4,** estudiante de la Carrera de Tecnología Superior en Desarrollo de Software, realizó para su trabajo de integración curricular el desarrollo de un sistema (*backend)* para la interacción de los miembros de la comunidad de la Iglesia Fraternidad Internacional del Evangelio Completo GISI (IFGF), mismo que cumple con todos los requerimientos y funcionalidades definidas en las reuniones mantenidas.

Es todo en cuanto puedo certificar en honor a la verdad, el interesado puede hacer uso del mismo como bien tuviere.

Atentamente

Pastor Walter Hidalgo
0984575828

Av. Ilaló y Río Corrientes 170804 San Rafael
Quito-Ecuador

ANEXO III

A continuación, para acceder al enlace del Manual de Usuario se debe ingresar a la siguiente URL:

https://youtu.be/-ZNg8xB_4zs

en el que se menciona de forma clara y detallada cada una de las funcionalidades del *backend* así como los perfiles que intervienen en la misma.

ANEXO IV

A continuación, se procede a definir las credenciales de acceso para el *backend*, así como el enlace al repositorio en *GitHub* en donde se encuentra el código fuente y en el apartado de *README* los pasos para realizar la instalación de forma local.

Credenciales de acceso para el *backend*

Para acceder al *backend* en producción, ingresar a la siguiente URL:

https://backend-ifgf.herokuapp.com/api

Credenciales para el perfil administrador:

- **Correo del usuario:**
- **Contraseña:**

Credenciales para el perfil miembro de la Iglesia:

- **Correo del usuario:**
- **Contraseña:**

Repositorio del código fuente del *backend*

El código fuente de todo el proyecto, se encuentra alojado en el repositorio *GitHub*, el cual se puede acceder a través de la siguiente URL:

https://github.com/MichaelGuanoluisa/Backend

Documentación de las rutas del *backend*

La documentación de cada ruta del *backend* con sus datos de entrada y posibles respuestas se presenta por un servicio de *Swagger*, el cual se ha desplegado en conjunto con el *backend*, al cual se puede acceder a través de la siguiente URL:

https://backend-ifgf.herokuapp.com/api-docs/