ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

DESARROLLO DE UN SISTEMA WEB PARA CAPACITACIÓN DE CIBERSEGURIDAD CON ESTRATEGIA DE GAMIFICACIÓN

DESARROLLO DEL BACKEND

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

JORGE WASHINGTON ALBA VENEGAS

DIRECTOR: DR. RICHARD PAUL RIVERA GUEVARA

DMQ, febrero 2022

CERTIFICACIONES

Yo, Jorge Washington Alba Venegas 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.

JORGE WASHINGTON ALBA VENEGAS

jorge.alba@epn.edu.ec

jorge-ctps@outlook.com

Certifico que el presente trabajo de integración curricular fue desarrollado por Jorge Washington Alba Venegas, bajo mi supervisión.

DR. RICHARD PAUL RIVERA GUEVARA
DIRECTOR

richard.rivera01@epn.edu.ec

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el producto resultante 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.

Jorge Washington Alba Venegas

DEDICATORIA

Dedico este proyecto a mi familia, debido a que, me han brindado un apoyo total durante mi carrera.

ALBA VENEGAS JORGE WASHINGTON

AGRADECIMIENTO

Agradezco a mi familia por estar siempre a mi lado apoyándome en mis estudios y brindándome su eterna confianza y ánimos.

ALBA VENEGAS JORGE WASHINGTON

ÍNDICE DE CONTENIDO

CERTIFICACIONES I
DECLARACIÓN DE AUTORÍA II
DEDICATORIA III
AGRADECIMIENTO IV
ÍNDICE DE CONTENIDO V
RESUMEN VII
ABSTRACT VIII
1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO 1
1.1 Objetivo general 2
1.2 Objetivos específicos 2
1.3 Alcance 2
1.4 Marco Teórico 2
Metodología 2
Metodología ágil 3
Backend 3
2 METODOLOGÍA 4
2.1 Metodología de Desarrollo 4
Roles 4
Artefactos 5
2.2 Diseño de la arquitectura 7
Arquitectura de Datos 7
Patrón arquitectónico Modelo Vista Controlador 7
2.3 Herramientas de desarrollo 8
3 RESULTADOS 10
Sprint 0. Configuración del ambiente de desarrollo 10
Sprint 1. Implementación de modelo usuario y autenticación. 12
Sprint 2. Implementación de modelo temas y progreso del curso. 17
Sprint 3. Implementación de campos para la gamificación del curso. 24
Sprint 4. Implementación de modelos contenidos, logros y políticas. 27
Sprint 5. Despliegue en Heroku y pruebas de aceptación. 34
4 Conclusiones 38
5 Recomendaciones 39

6 REFERENCIAS BIBLIOGRÁFICAS 40
7 ANEXOS 42

RESUMEN

Los problemas de ciberseguridad se han visto incrementados debido a la pandemia, puesto que la modalidad de teletrabajo trajo consigo un mayor número ataques en Ecuador. Por lo tanto, tener conocimiento básico de la ciberseguridad es una necesidad en estos tiempos; las empresas deben capacitar continuamente a sus empleados y de forma general, es relevante incrementar el conocimiento sobre las amenazas de ciberseguridad que afectan a los usuarios de Internet.

En relación con la falencia de conocimiento sobre ciberseguridad en la población, se ha desarrollado una *API REST* la cual proporciona un acceso, obtención, transformación y devolución de la información que el *frontend* del sistema web para capacitación en ciberseguridad con técnicas de gamificación requiera presentar para una correcta interacción con el usuario, por medio de consultas robustas y complejas.

Este proyecto se ha desarrollado con el *framework* de *Laravel* el cual posee una interacción eficiente con la base de datos *MySQL*. Se ha usado la metodología ágil S*crum* debido a que brinda una correcta flexibilidad y adaptación a cambios, como también una interacción constante con el usuario.

En el presente documento se expone el procedimiento del desarrollo de la *API REST* como también los resultados obtenidos, el despliegue en producción y la capacidad de ser consumida por el cliente.

PALABRAS CLAVE: *API REST*, *MySQL*, *Laravel*, S*crum*.

ABSTRACT

Cybersecurity problems have increased due to the pandemic, since the teleworking modality brought with it greater number of attacks in Ecuador. Therefore, having basic knowledge of cybersecurity is a necessity in these times; companies must continuously train their employees and in general it is relevant to increase the knowledge about cybersecurity threats that affect Internet users.

In relation to the lack of knowledge about cybersecurity in the population, a *REST API* has been developed to provide access, obtain, transform and return the information that the frontend of the web system for cybersecurity training with gamification techniques requires, to present for a correct interaction with the user, through robust and complex queries.

This project has been developed with *Laravel* framework which has an efficient interaction with the *MySQL* database. The agile methodology S*crum* has been used because it provides a correct flexibility and adaptation to changes, as well as a constant interaction with the user.

In this document the procedure of the development of the *REST API* is exposed as well as the results obtained, the deployment in production and the capacity to be consumed by the client.

KEYWORDS: *API REST*, *MySQL*, *Laravel*, S*crum.*

1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

La ciberseguridad es una temática de gran importancia para cualquier tipo de usuario, es decir, para usuarios casuales como también para empresas u organizaciones; debido a que estas últimas poseen un tratamiento de datos más complejo. Los datos generados por dichas entidades se encuentran en constante peligro de ser interceptados y ser tratados para fines delictivos por atacantes cibernéticos, generando consecuencias negativas para el afectado, estos casos se producen mayoritariamente cuando se presentan medidas deficientes, conocimiento sobre ciberseguridad exiguo o malas prácticas de ciberseguridad [1].

A medida que el tiempo transcurre y con el avance de la tecnología aparecen técnicas o prácticas mejoradas para brindar una mayor protección a los datos, dispositivos inteligentes y equipos informáticos [2]. En este sentido, obtener conocimientos básicos sobre la temática de ciberseguridad es una necesidad que se debe efectuar de forma eficiente [3]. En la actualidad existe una gran cantidad de cursos en línea para adquirir conocimiento sobre cualquier temática de interés del usuario, varios de estos cursos brindan un buen catálogo de temas, sin embargo, no ofrecen una experiencia interactiva durante el progreso del curso al usuario.

Los cursos en línea deben adaptar sus características a las nuevas necesidades, para que de este modo el usuario genere cada vez más interés en el curso que está desarrollando; este objetivo se lo puede lograr con el uso de técnicas que permitan potenciar las habilidades del usuario a través una experiencia interactiva como lo es la gamificación [4]; la gamificación es un concepto el cual está relacionado con la meta de alcanzar los objetivos propuestos, mientras la experiencia del usuario durante el desarrollo de las actividades ofrecidas se las realiza con contenido dinámico, el cual tiene una influencia en el usuario a proseguir realizando dichas actividades hasta conseguir el objetivo principal.

En relación con lo mencionado con anterioridad, es necesario el desarrollo del *backend* que consiste en una *API REST* que brinda la posibilidad al *frontend* de generar un sistema interactivo con el uso de varias características como lo son: el generar una interfaz con contenidos dinámicos con la implementación de técnicas de gamificación como también, el ofrecer conocimientos variados y sólidos sobre la ciberseguridad o cualquier otra temática de interés, que posibiliten al usuario al terminar el curso cumplir el objetivo de obtener los conocimientos necesarios para posteriormente aplicarlos en la vida diaria.

1.1 Objetivo general

Desarrollar una *API REST* para el sistema web de capacitación en ciberseguridad con estrategia de gamificación.

1.2 Objetivos específicos

1. OE1 Levantar los requerimientos funcionales y no funcionales del sistema.
2. OE2 Diseñar la arquitectura y base de datos del sistema.
3. OE3 Implementar el *API REST* para que sea accedida por cualquier cliente.
4. OE4 Implementar pruebas de aceptación del *API REST*.
5. OE5 Desplegar el API REST en un servidor público.

1.3 Alcance

El presente proyecto propone el desarrollo de una *API REST* con el objetivo de aprovechar los beneficios ofrecidos por las herramientas de desarrollo y de esta manera brindar respuestas correctas a la interfaz de usuario, como también tener un almacenamiento de datos simple pero eficiente. Los clientes podrán tener acceso a visualizar, almacenar, editar, crear y eliminar datos correspondientes a la base de datos del sistema. Además, el *API REST* brindará la autenticación de usuarios; como también la seguridad necesaria en el acceso y manejo de la información disponible.

1.4 Marco Teórico

Metodología

Una metodología se refiere a un grupo de métodos o técnicas que son usadas para cumplir con un objetivo o meta propuestos dentro de una indagación científica o actividades relacionadas con el uso de destrezas o entendimientos. También, se define a una metodología como el proceso de optar por un método adecuado en relación a la aplicación de un objeto en concreto [5].

Una metodología de desarrollo hace referencia a una gama de actividades las cuales proporcionan la gestión o administración correcta a un proyecto de software para conseguir

un producto con probabilidades altas de éxito [6]. Dentro del desarrollo de software se hacen presentes dos tipos de metodologías de desarrollo las cuales se denominan como tradicionales y ágiles. Las metodologías tradicionales se caracterizan por presentar una notoria rigidez, es decir, no trabaja en base al dinamismo; en esta metodología también se debe generar documentación profunda. Estas metodologías a medida del tiempo han presentado varias deficiencias que no aportan aspectos positivos al desarrollo del proyecto como a las personas inmersas en este, puesto que, en este tipo de metodologías la participación del cliente es casi nula, además de que el objetivo o meta principal se encuentra centrado y definido únicamente en la fase inaugural de dicho proyecto [5].

Metodología ágil

Las metodologías designadas como ágiles poseen una base fundamental en el aspecto relacionado con las capacidades de respuestas a cambios del equipo de desarrollo y también en la continua retroalimentación con el cliente para obtener un producto centrado en la calidad; estos aspectos son temas tratados en las reuniones constantes que se realizan durante todo el desarrollo del proyecto. Este tipo de metodologías destacan por ser implementadas en proyectos que necesiten ser desarrollados en un tiempo corto [5]. Sin embargo, estas metodologías también exhiben ciertos conflictos, entre el cual destaca la tardía detección de algún error o cambio, debido a que esto puede afectar a la calidad del producto como también, al costo de este.

Backend

El *backend* dentro de un sistema web hace referencia al manejo de todos los procedimientos que no son visibles de dicho sistema, pero que poseen una relevancia significativa en relación con el correcto funcionamiento de un sistema web: una de las actividades esenciales que caracterizan a un *backend* son la comunicación con la base de datos o la conexión con el servidor hosting [7].

La importancia de un correcto desarrollo de *backend* radica en varios aspectos, entre los cuales destacan: el intercambio de información, debido a que los usuarios interactúan con el sistema y generan datos que posteriormente son almacenados en la base de datos; mejora de la experiencia del usuario, una rapidez de carga mínima es de gran relevancia para que los usuarios obtengan una buena experiencia del sistema, por último la seguridad es un aspecto fundamental en los sistemas actuales por lo cual el *backend* debe proporcionar seguridad en el acceso a las búsquedas que se ofrecen [7].

2 METODOLOGÍA

Un caso de estudio es un término que engloba varios métodos de investigación que poseen como base la indagación alrededor de una ejemplificación; un caso de estudio se caracteriza principalmente por la existencia de una correcta comprensión de la realidad del objeto de estudio. Por otro lado, su objetivo se relaciona con el alcance de la comprensión de su actividad en situaciones relevantes, con bases en el estudio de particularidades y complejidades de dicho caso [8]. El caso de estudio correspondiente para este proyecto es la ciberseguridad, basada en el desarrollo de un *backend* que proporciona los métodos correspondientes para generar un curso de temáticas de interés aplicando técnicas de gamificación.

2.1 Metodología de Desarrollo

S*crum* es una metodología ágil, la cual permite generar iteraciones conocidas como *Sprints,* que facilitan el desarrollo de todas las actividades propuestas para el proyecto. S*crum* brinda la posibilidad de encajar cambios que se presenten durante las etapas de desarrollo de los *Sprints* las cuales tendrán una duración de aproximadamente 3 semanas [9]. Además, con el uso de esta metodología ágil se logra mantener contacto constante entre el cliente y el equipo de desarrollo.

Roles

S*crum* define varios roles con sus respectivas responsabilidades, donde cada rol debe ser cumplido con su respectiva dedicación para lograr todos los objetivos propuestos y por lo tanto generar software de calidad [9]. Es de gran importancia tener como prioridad el cumplimiento de los requerimientos del cliente, lo cual conlleva a que todo el proceso sea correctamente manejado y cumplir con las expectativas del cliente.

Product Owner

La responsabilidad principal del *Product Owner* es el manejo del *Product Backlog*, para lo cual es necesario realizar las historias de usuario y proporcionar la priorización correspondiente. Además, la persona con este rol necesita generar una comunicación eficiente en relación con los objetivos y elementos del *Product Backlog* [9]. Este rol está asignado al Dr. Richard Rivera puesto que posee gran conocimiento del tema y proporciona la información necesaria al resto del equipo.

S*crum Master*

La persona con la asignación de este rol debe tener la capacidad de transmitir el conocimiento teórico y práctico de S*crum*, [9] también debe asegurar el cumplimiento de las características correspondientes a los eventos de S*crum* como lo son los plazos y el nivel de producción. En este aspecto este rol fue asignado al Dr. Richard Rivera.

Development Team

El *Development Team* debe estar en capacidad de aplicar habilidades técnicas las cuales se desarrollan en la implementación del proyecto [9], tienen la responsabilidad de desempeñar los *Sprints* y de cumplir con el *Sprint Goal*.

En la **TABLA I** se visualiza las asignaciones correspondientes al equipo S*crum*, con sus respectivos nombres y cargos.

TABLA I Asignación de roles

Rol	**Integrante**
Product Owner	Dr. Richard Rivera
Scrum Master	Dr. Richard Rivera
Development Team	Sr. Jorge Alba Sr. Juan Dávila

Artefactos

La documentación dentro de S*crum* permite que el proceso de desarrollo del proyecto posea una organización adecuada, de tal manera que, se obtenga un proyecto de calidad y que proporcione valor al cliente [9].

Recopilación de Requerimientos

Los requerimientos corresponden a las funcionalidades que el usuario final necesita en el proyecto, por lo cual, deben ser tratadas correctamente para obtener resultados satisfactorios para el cliente final [10].

La recopilación de requerimientos fue obtenida por medio del *Product Owner*, quien planteo y detalló con claridad los requerimientos del sistema web para después realizar un análisis

y proponer una posible solución. La información completa con respecto a la recopilación de requerimientos se encuentra en el **ANEXO II**.

Historias de usuario

Una historia de usuario hace referencia a un escrito en el cual se encuentra detallado un requerimiento propuesto por el usuario final, el cual se encuentra escrito desde un criterio netamente del cliente [11]. Cada historia de usuario corresponde a un requerimiento del sistema como se lo visualiza en la **TABLA II**. Las historias de usuario restantes se encuentran en el **ANEXO II.**

TABLA II Historia de Usuario HU001

HISTORIA DE USUARIO	
Identificador (ID): HU001	**Usuario:** Administrador
Nombre Historia: Registrar usuarios	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alta
Iteración Asignada: 2	
Responsable: Jorge Alba	
Descripción: El administrador deberá proporcionar la funcionalidad de registro en el *backend*, para que el usuario pueda registrarse en el sistema.	
Observación: El registro deberá contener los siguientes campos: nombre, email, contraseña, confirmar contraseña.	

Product Backlog

Corresponde a un listado de los elementos del producto o proyecto, el cual debe estar ordenado en relación con la priorización propuesta. La persona encargada del *Product Backlog* es el *Product Owner* quien tiene la responsabilidad de proporciona una gestión total del *Product Backlog* para alcanzar los objetivos planteados. El detalle del *Product Backlog* se encuentra en el **ANEXO II**.

Sprint Backlog

El desarrollo del proyecto se lo realiza por medio de iteraciones las cuales a su vez contienen tareas correspondientes a cada *Sprint*, una vez culminadas todas las actividades del *Sprint* se asigna el estado de finalizado a modo de demostración del trabajo realizado. El equipo de desarrollo son los encargados de realizar estas tareas con ayuda del S*crum* Master quien se encarga de tomar la decisión con respecto a que tareas son asignadas del

Product Backlog al *Sprint Backlog*. El *Sprint Backlog* detallado se encuentra en el **ANEXO II**.

2.2 Diseño de la arquitectura

El problema presentado debe ser resuelto por medio de un modelo o patrón arquitectónico el cual debe generar una solución que se basa en la eficiencia y satisfacción de las necesidades del cliente. En la siguiente sección se detalla la elección de modelo arquitectónico que fue usado para el desarrollo del proyecto.

Arquitectura de Datos

El *framework Laravel* hace uso de Eloquent el cual es un mapeador relacional de objetos; esta herramienta proporciona una interacción apacible con la base de datos. La interacción se da gracias a la generación de componentes denominados como modelos, los cuales interactúan directamente con la base de datos. Además, esta herramienta también ofrece más funcionalidades importantes y necesarias para el desarrollo de un *backend* como lo son: inserción, actualización, y eliminación de registros en la base de datos por medio de comandos sencillos de ejecutar [12].

En la **Fig. 1** se visualiza las herramientas usadas para el desarrollo del proyecto como también el flujo del Modelo-Vista-Controlador (MVC).

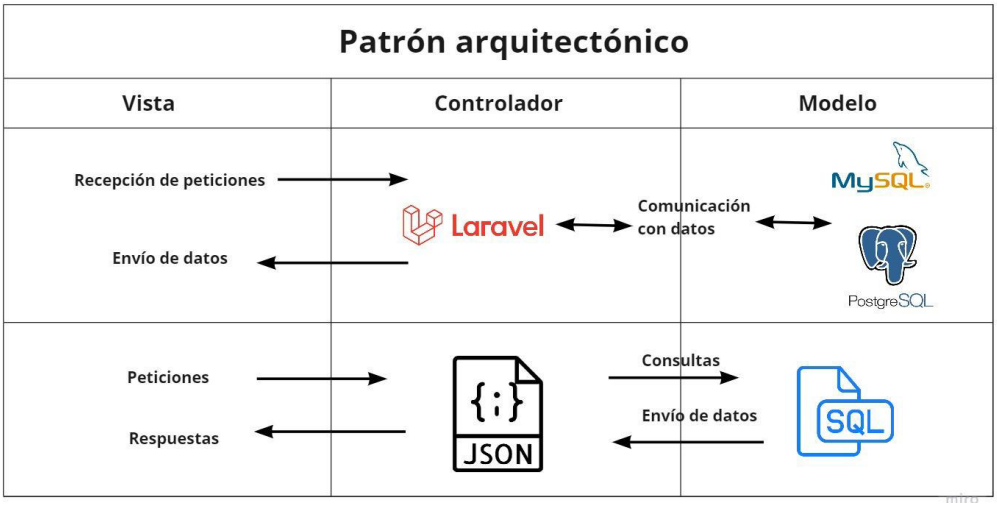


Fig. 1: Patrón arquitectónico correspondiente a la *API REST*

Patrón arquitectónico Modelo Vista Controlador

El uso del patrón arquitectónico MVC, permite el manejo de una buena organización y estructuración de los elementos o componentes de un proyecto de software; con el uso de

este modelo se puede gestionar de mejor manera las responsabilidades y las relaciones entre los elementos que componen el sistema de software. Este modelo cuenta con varias capas que se describen a continuación:

Modelo: hace referencia a las consideraciones o entidades que poseen la función de almacenamiento de la información del sistema [13].

Vista: esta capa es la encargada de todos los elementos de visualización, es decir, es la responsable de la generación de la interfaz que se presenta al usuario final [13].

Controlador: la capa de controlador posee como objetivo principal el posibilitar un enlace entre el usuario y el sistema, es decir, actúa como intercesor entre dichas entidades [13].

2.3 Herramientas de desarrollo

En la **TABLA III** se puede observar las herramientas que fueron elegidas, debido a la estructura del proyecto y los requerimientos de este; estas herramientas son necesarias para un correcto desarrollo del proyecto.

TABLA III Herramientas usadas para el desarrollo del proyecto

Herramienta	Descripción	Justificación
Laravel	Laravel es un framework PHP que cuenta con sintaxis sencilla y que por consiguiente es fácil de entender, este *framework* permite el desarrollo de proyectos en tiempos cortos y eficientemente [12].	Gracias a su arquitectura Modelo Vista Controlador proporciona funcionalidades que permiten adaptar y desarrollar todas las partes de una aplicación de *backend* de manera clara.
Composer	Es un gestor de dependencias de PHP, permite usar librerías de terceros que son necesarias para un desarrollo completo de un *API* [12].	Composer permite declarar, descargar y conservar actualizados los paquetes de software necesarios para el proyecto.
XAMPP	Distribución de Apache de software libre que ofrece los servicios de Apache, MSQL/MariaDB, PHP y Perl [14].	XAMPP ofrece los servicios necesarios para la implementación del proyecto puesto que todas sus funcionalidades se relacionan correctamente con el *framework* de *Laravel*.
PostgreSQL	Sistema de base de datos relacional, que permite escribir código desde diferentes leguajes obviando una nueva compilación de base de datos [15].	Permite dar una correcta integridad a los datos, generar entornos tolerantes fallos y ofrece una administración adecuada para cualquier tipo de proyecto.

Postman	Herramienta que ofrece el servicio de testear cualquier *API* mediante el uso de peticiones a esta [16].	*Postman* permite que se lleve un control eficiente en las peticiones del proyecto y visualizar si las respuestas obtenidas de la *API* son las correctas.
Heroku	Plataforma que ofrece servicios de servidores, a la vez que también se puede obtener una administración sencilla de dichos servicios [17].	Esta plataforma permite un despliegue de *backend* sencillo debido a su gratuidad y correcta administración.
Git	Herramienta orientada al control de versionamiento de proyectos de software [18].	*Git* permite durante el desarrollo del proyecto trabajar en ramas para obtener un desarrollo más eficiente y tolerante a fallos por medio de comandos sencillos.
GitHub	*GitHub* es una plataforma que ofrece el servicio de repositorios para proyectos de software que son de vital importancia para una correcta gestión de dichos proyectos [18].	Esta plataforma permite gestionar el proyecto de manera sencilla durante la etapa de desarrollo del proyecto y despliegue del *backend* en producción.

3 RESULTADOS

En esta sección se presentan los resultados obtenidos de forma detallada de las actividades realizadas en cada *Sprint*.

Sprint 0. Configuración del ambiente de desarrollo

El *Sprint Backlog* 0 corresponde a las tareas relacionadas con la preparación y configuración del ambiente de trabajo, para que posterior a ella sea posible trabajar de forma correcta las funcionalidades del proyecto. Resultados obtenidos en el *Sprint*:

- Creación del diseño de la base de datos en PowerDesigner.
- Configuración de XAMPP.
- Configuración del IDE PHPStorm.
- Creación del proyecto en *Laravel*.

Creación del diseño de la base de datos en PowerDesigner.

Para el diseño de la base de datos que será implementada en el *backend* del sistema, es necesario realizarlo con las entidades correspondientes y con sus respectivas relaciones, además, de contar con todos los campos necesarios que permiten que el sistema cumpla con los requerimientos propuestos por el cliente como se muestra en la **Fig. 2**.

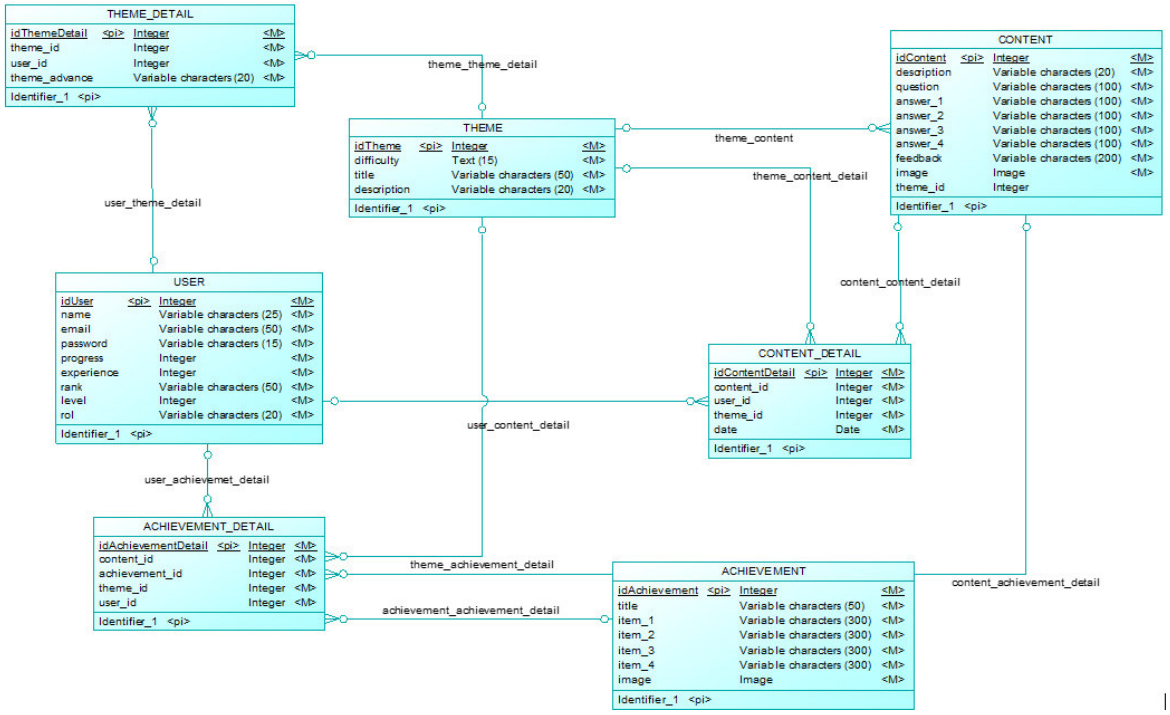


Fig. 2: Modelo de la base de datos de la *API REST*

Configuración del IDE PHPStorm.

PHPStorm es un IDE orientado al lenguaje PHP el cual brinda un editor completo para PHP, HTML Y JavaScript [19]. Además, también brinda otras funcionalidades como prevención de errores y una conexión con XAMPP sencilla que permite que el desarrollo del *backend* sea desarrollado con herramientas con buena comunicación y compatibilidad como se visualiza en la **Fig. 3**.

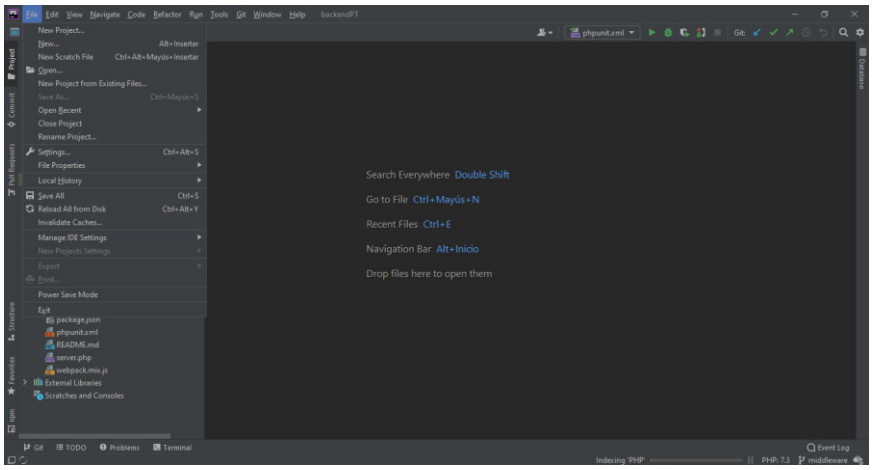


Fig. 3: Configuración del IDE de desarrollo PHPStorm

Configuración de XAMPP.

Es un software que ofrece un sistema de gestión de base de datos MySQL, se encuentra soportado por Apache el cual permite el uso del interprete para leguaje PHP que será el usado en el desarrollo del backend del proyecto como se muestra en la **Fig. 4**.

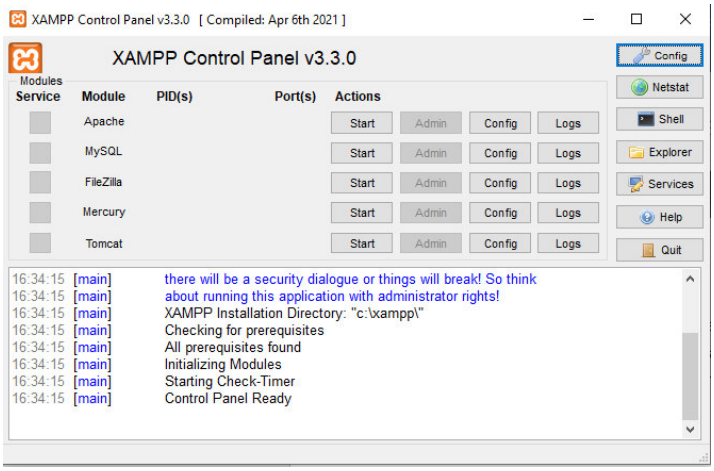


Fig. 4: Configuración de la herramienta XAMPP

Creación del proyecto en *Laravel.*

En la **Fig. 5** se visualiza la implementación del *backend* para el sistema, este es desarrollado en el *framework* de *Laravel* por lo cual es necesario la creación del proyecto para empezar con la implementación de los requerimientos del cliente y de este modo cumplir con los objetivos propuestos durante cada etapa de desarrollo del proyecto.

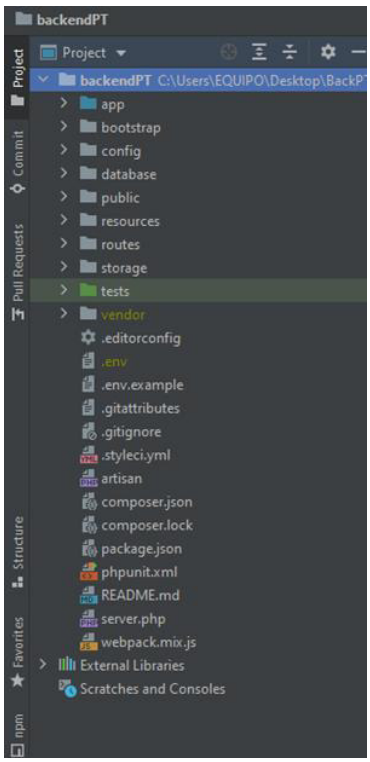


Fig. 5: Inicialización del proyecto en *Laravel*

Sprint 1. Implementación de modelo usuario y autenticación.

En concordancia con el *Sprint Backlog*, para realizar a cabalidad las actividades correspondientes al *Sprint* 1, se implementó la tabla o también denominado modelo con los campos necesarios para realizar un registro, inicio de sesión y cierre de sesión para los usuarios del sistema. Estas actividades corresponden desde la implementación de las migraciones hasta la creación de rutas las cuales pueden ser accedidas por el cliente. Resultados obtenidos del *Sprint*:

- Elaboración de migración y *seeder* para el modelo de usuario.
- Implementación función de registro, inicio de sesión y cierre de sesión con validación de campos.
- Elaboración de ruta de registro, inicio de sesión y cierre de sesión de usuarios.

- Elaboración de funciones de petición para visualización de datos del modelo de usuario.

Elaboración de migración y *seeder* para el modelo de usuario.

Laravel hace uso de la herramienta denominada como JWT (*JSON Web Token*) la cual permite generar token para los usuarios que se registren o inicien sesión en el sistema con el motivo u objetivo del correcto envió de datos *JSON*, a la vez que proporciona las características de validación y seguridad de dichos datos.

Para hacer uso total de JWT se necesita realizar la instalación de esta en el proyecto que está siendo desarrollado, posterior a ello se debe modificar el archivo de configuración de la app implementando esta herramienta en los *providers* y en *aliases*. El siguiente paso es publicar el archivo de configuración y generar un *jwt-auth secret*.

Con la herramienta de JWT lista para ser usada se procede a la implementación de las migraciones en *Laravel*, las cuales permiten que en la base de datos se genere la tabla correspondiente. En la **Fig. 6** se muestra la migración correspondiente al modelo de usuario.

```php
public function up()
{
    // Creación de tabla usuarios con sus respectivos atributos y valores por defecto

    Schema::create( table: 'users', function (Blueprint $table) {
        $table->bigIncrements( column: 'id');
        $table->string( column: 'name');
        $table->string( column: 'email')->unique();
        $table->string( column: 'password');
        $table->rememberToken();
        $table->timestamps();
    });
}
```

Fig. 6: Migración de modelo usuario

En la **Fig. 7** se observa de la implementación del *seeder* correspondiente con el fin de poseer dos cuentas para el administrador y usuarios ficticios los cuales permiten comprobar el funcionamiento del almacenamiento en la base de datos en relación con el modelo usuario.

```php
// Creación de 2 administardores
$password = Hash::make( value: '123123');

User::create([
    'name' => 'Administrador1',
    'email' => 'admin1@gmail.com',
    'password' => $password,
]);
User::create([
    'name' => 'Administrador2',
    'email' => 'admin2@gmail.com',
    'password' => $password,
]);

// Generar usuarios ficticios
for ($i = 0; $i < 10; $i++) {
    User::create([
        'name' => $faker->name,
        'email' => $faker->email,
        'password' => $password,
    ]);
}
```

Fig. 7: *Seeder* del modelo usuario

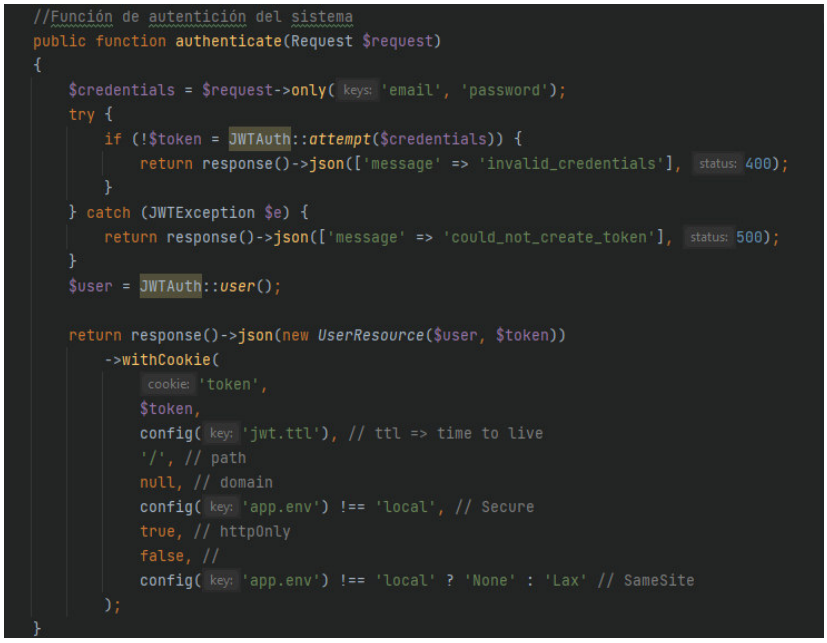
Implementación función de registro, inicio de sesión y cierre de sesión.

Para realizar la implementación de las funciones de autenticación es necesario colocar en el modelo de usuario creado que hará uso de la herramienta de JWT, en la **Fig. 8** se observa el código correspondiente en dicho modelo.

```php
use Tymon\JWTAuth\Contracts\JWTSubject;

class User extends Authenticatable implements JWTSubject
{
```

Fig. 8: Uso de JWT el modelo de usuario

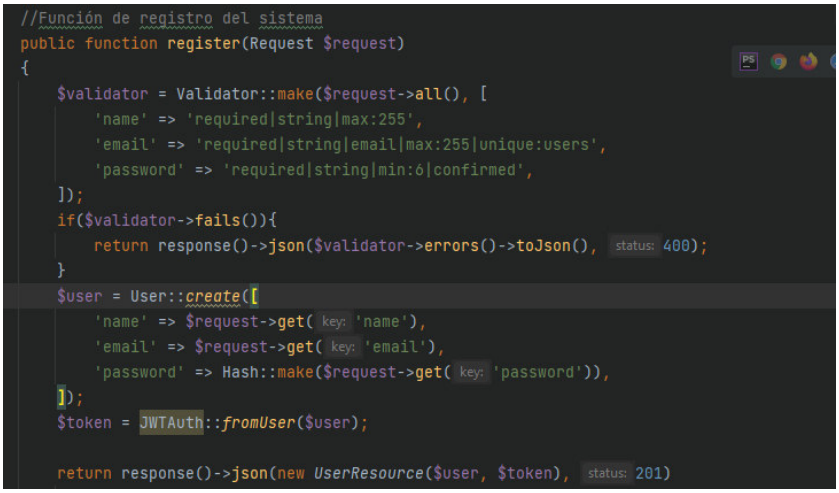
Laravel permite crear controladores los cuales brindan la posibilidad de generar la lógica de las funciones que realizarán las rutas que posteriormente serán creadas y accedidas por el cliente. En la **Fig. 9** se muestra la implementación del controlador del modelo de usuario el cual cuenta con las funciones de autenticación correspondientes teniendo en cuenta la validación de los campos que son necesarios para cada formulario, para la función de inicio de sesión se requiere el correo y contraseña por lo cual la función valida que esos campos existan en la base de datos y que sean los correctos.



```php
//Función de autentición del sistema
public function authenticate(Request $request)
{
    $credentials = $request->only( keys: 'email', 'password');
    try {
        if (!$token = JWTAuth::attempt($credentials)) {
            return response()->json(['message' => 'invalid_credentials'], status: 400);
        }
    } catch (JWTException $e) {
        return response()->json(['message' => 'could_not_create_token'], status: 500);
    }
    $user = JWTAuth::user();

    return response()->json(new UserResource($user, $token))
        ->withCookie(
            cookie: 'token',
            $token,
            config( key: 'jwt.ttl'), // ttl => time to live
            '/', // path
            null, // domain
            config( key: 'app.env') !== 'local', // Secure
            true, // httpOnly
            false, //
            config( key: 'app.env') !== 'local' ? 'None' : 'Lax' // SameSite
        );
}
```

Fig. 9: Función de autenticación en el controlador del modelo usuario

Por otro lado, en la **Fig. 10** se visualiza la función de registro que realiza la validación de los campos como nombre, correo, contraseña y confirmación de contraseña.



```php
//Función de registro del sistema
public function register(Request $request)
{
    $validator = Validator::make($request->all(), [
        'name' => 'required|string|max:255',
        'email' => 'required|string|email|max:255|unique:users',
        'password' => 'required|string|min:6|confirmed',
    ]);
    if($validator->fails()){
        return response()->json($validator->errors()->toJson(), status: 400);
    }
    $user = User::create([
        'name' => $request->get( key: 'name'),
        'email' => $request->get( key: 'email'),
        'password' => Hash::make($request->get( key: 'password')),
    ]);
    $token = JWTAuth::fromUser($user);

    return response()->json(new UserResource($user, $token), status: 201)
```

Fig. 10: Función de registro en el controlador del modelo usuario

Por último, la función de cierre de sesión mostrada en la **Fig. 11** implementa la funcionalidad de invalidar el *token* siempre y cuando este activo previamente y corresponda al usuario con esa sesión activa.

Fig. 11: Función de cierre de sesión en el controlador del modelo usuario

Se genera una función visualizada en la **Fig. 12** con el objetivo de obtener los datos del usuario con sesión activa, la cual devuelve todos los datos del usuario con dicha sesión siempre y cuando el *token* sea válido y corresponda al usuario, esta función también se encarga de devolver el mensaje correspondiente del error que puede llegar a producirse.

Fig. 12: Función de obtención del usuario con sesión activa

Elaboración de ruta de registro, inicio de sesión y cierre de sesión de usuarios.

Los elementos intermediarios entre el requerimiento del usuario y la devolución de los datos almacenados en la base de datos son las rutas con sus respectivos tipos de petición. Para realizar la implementación de dichas rutas se las deben colocar en el archivo api.php como se muestra en la **Fig. 13**, cada ruta debe especificar el tipo de petición de la ruta, el nombre

con el cual será llamada desde el cliente y la referencia del controlador y nombre de función a la cual corresponde su lógica.

```
//Rutas de autenticacion
Route::post( uri: 'register', [UserController::class, 'register']);
Route::post( uri: 'login', [UserController::class, 'authenticate']);
Route::post( uri: 'logout', [UserController::class, 'logout']);
```

Fig. 13: Rutas correspondientes al modelo de usuario

Elaboración de funciones de petición para visualización de datos del modelo de usuario.

Las peticiones GET son de gran importancia para los clientes puesto que estas devuelven información importante, como se menciona con anterioridad, es necesario una ruta la cual devuelva los datos del usuario **Fig. 14**, para lo cual se creó en el controlador la lógica correspondiente a esta solución, una vez generada dicha lógica se crea una nueva ruta denominada *user* la cual devolverá los datos del usuario con sesión activa.

```
Route::get( uri: '/user', [UserController::class, 'getAuthenticatedUser']);
```

Fig. 14: Ruta de obtención de usuario con sesión activa

En el caso del usuario administrador puede acceder a la lista de usuarios devolviendo los datos estrictamente necesarios por lo cual también se deben generar la lógica en el controlador y sus respectivas rutas.

En el caso del usuario administrador puede acceder a la lista de usuarios devolviendo los datos estrictamente necesarios por lo cual también se deben generar la lógica en el controlador y sus respectivas rutas como se visualiza en la **Fig. 15**.

```
Route::get( uri: '/allUsers', [UserController::class, 'allUsers']);
Route::get( uri: '/users/{user}', [UserController::class, 'show']);
```

Fig. 15: Rutas para administradores

Sprint 2. Implementación de modelo temas y progreso del curso.

Realizando el seguimiento al *Sprint Backlog* se realizan las actividades descritas en el *Sprint* 2, las cuales están relacionadas con la implementación del modelo de temas con

sus respectivas funcionalidades y rutas, hasta generar un nuevo campo para el usuario que corresponde al progreso del curso y los progresos de los temas. Resultados obtenidos del *Sprint*:

- Elaboración de migración, *seeder*, controlador y rutas para la creación, visualización, actualización y eliminación del modelo de temas.
- Elaboración de migración, controlador y rutas para la creación, visualización, actualización y eliminación del modelo de detalles de tema.
- Modificación de la migración y *seeder* del modelo de usuario para generar el campo de progreso del curso.

Elaboración de migración, *seeder*, controlador y rutas para la creación, visualización, actualización y eliminación del modelo de temas.

La **Fig. 16** muestra las migraciones correspondientes al modelo de temas las cuales cuentan con los siguientes atributos: id, titulo, descripción y dificultad. Esta actividad se la realizo con un solo comando de PHP el cual creo dicha tabla en la base de datos con sus atributos correspondientes.

```php
public function up()
{
    //Creación de tabla temas con sus respectivos atributos

    Schema::create( table: 'themes', function (Blueprint $table) {
        $table->bigIncrements( column: 'id');
        $table->string( column: 'title');
        $table->string( column: 'description');
        $table->string( column: 'difficulty');
        $table->timestamps();
    });
}
```

Fig. 16: Migración del modelo de temas

Con el objetivo de obtener datos ficticios en el modelo de temas se creó un *seeder* visualizado en la **Fig. 17** para dicho modelo en el cual se crean los datos ficticios gracias a la herramienta de *faker* la cual proporciona una gran cantidad de opciones de datos ficticios.

```php
//Vaciar tabla
Theme::Truncate();

//Llamada a la herramienta faker
$faker = \Faker\Factory::create();


//Creación de temas ficticios
$num_themes = 6;
for ($i = 0; $i <$num_themes; $i++) {
    Theme::create([
        'title' => $faker->sentence,
        'description' => $faker->sentence(),
        'difficulty' => $faker->word,
    ]);
}
```

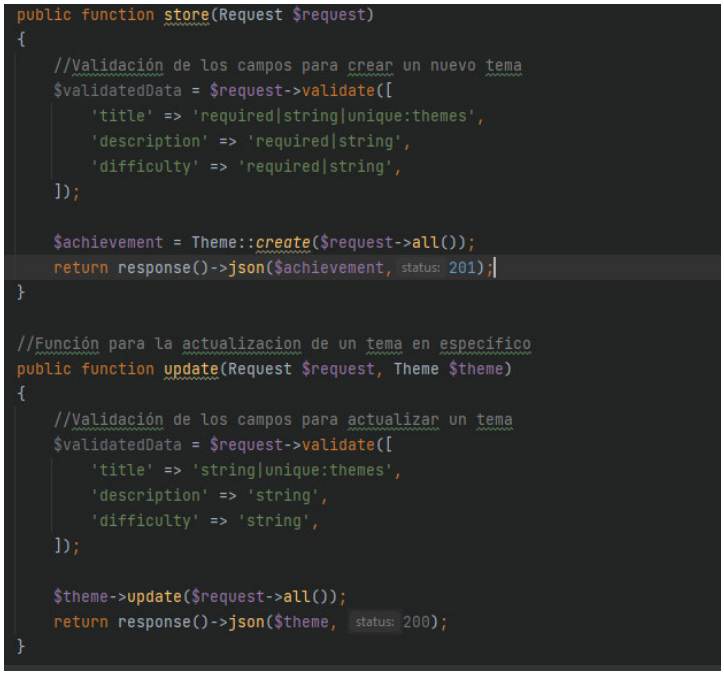
Fig. 17. *Seeder* del modelo de temas

Con las migraciones y datos ficticios en la base de datos se procede a crear la lógica en el controlador correspondiente al modelo de temas **Fig. 18**, es decir, se crean las funciones de visualización, creación, actualización y eliminación de temas. Para las funciones de visualización es necesario que se devuelva la información en formato *JSON* y que el cliente obtenga la lista de temas o un tema en específico por medio de su respectiva ruta.

```php
//Función para la obtención la lista de temas
public function index()
{
    return Theme::all();
}

//Funcion para obtener un tema en específico
public function show(Theme $theme)
{
    return $theme;
}
```

Fig. 18: Funciones de visualización de datos del modelo de temas

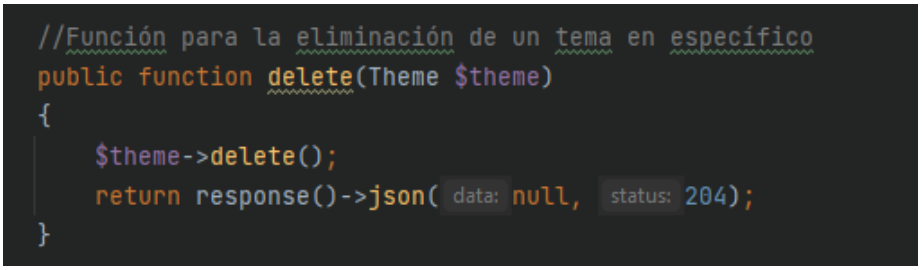
Con respecto a la lógica de las funciones de creación y actualización de teas es necesario realizar la respectiva validación de datos en cada función **Fig. 19**, esto permitirá que se los datos enviados por el cliente sean almacenados solo si cumple con las validaciones correspondientes, esto permite evitar que los datos que se almacenen sean erróneos.



```php
public function store(Request $request)
{
    //Validación de los campos para crear un nuevo tema
    $validatedData = $request->validate([
        'title' => 'required|string|unique:themes',
        'description' => 'required|string',
        'difficulty' => 'required|string',
    ]);

    $achievement = Theme::create($request->all());
    return response()->json($achievement, status: 201);
}

//Función para la actualizacion de un tema en especifico
public function update(Request $request, Theme $theme)
{
    //Validación de los campos para actualizar un tema
    $validatedData = $request->validate([
        'title' => 'string|unique:themes',
        'description' => 'string',
        'difficulty' => 'string',
    ]);

    $theme->update($request->all());
    return response()->json($theme, status: 200);
}
```

Fig. 19: Funciones de creación y actualización del modelo de temas

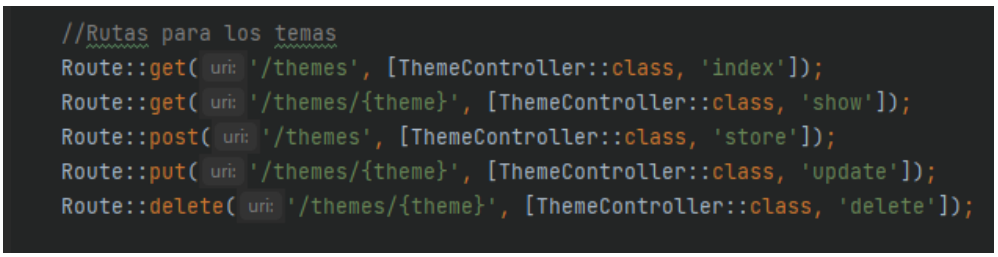
Por último, como se muestra en la **Fig. 20** se implementa la lógica de eliminación de un tema para lo cual se debe tener en cuenta que solo se podrá eliminar dicho tema siempre y cuando exista dentro de la base de datos.



```php
//Función para la eliminación de un tema en específico
public function delete(Theme $theme)
{
    $theme->delete();
    return response()->json( data: null, status: 204);
}
```

Fig. 20: Función para eliminar un tema en especifico

Una vez creadas las funciones en el controlador se prosigue con la implementación de las respectivas rutas **Fig. 21** que permitirán al cliente acceder e interactuar con la información que se encuentra en la base de datos.



```php
//Rutas para los temas
Route::get( uri: '/themes', [ThemeController::class, 'index']);
Route::get( uri: '/themes/{theme}', [ThemeController::class, 'show']);
Route::post( uri: '/themes', [ThemeController::class, 'store']);
Route::put( uri: '/themes/{theme}', [ThemeController::class, 'update']);
Route::delete( uri: '/themes/{theme}', [ThemeController::class, 'delete']);
```

Fig. 21: Rutas para el modelo de temas

Elaboración de migración, controlador y rutas para la creación, visualización, actualización y eliminación del modelo de detalles de tema.

Las migraciones que pueden ser implementadas en *Laravel* permite crear tablas o modelos fácilmente con los atributos necesarios e incluso si es necesario con valores por defecto. En el modelo de detalles de temas sus atributos están relacionados con otras tablas por lo cual son claves foráneas y un atributo de avance del tema los cuales permitirán al cliente conocer el estado de los temas de cada usuario **Fig. 22**, es decir, a que usuario pertenece, en que tema se encuentra y en qué estado está el avance de cada tema disponible.

```php
public function up()
{
    Schema::create( table: 'theme_details', function (Blueprint $table) {
        $table->bigIncrements( column: 'id');
        $table->foreignId( column: 'user_id')
            ->references( column: 'id')
            ->on( table: 'users')
            ->onDelete( action: 'restrict');
        $table->foreignId( column: 'theme_id')
            ->references( column: 'id')
            ->on( table: 'themes')
            ->onDelete( action: 'restrict');
        $table->string( column: 'theme_advance');
        $table->timestamps();
    });
}
```

Fig. 22: Migración del modelo de detalles de tema

Dentro del controlador del modelo de detalles de tema como se muestra en la **Fig. 23** se implementa la lógica respectiva a cada ruta que se creará, es decir, el cliente tendrá la posibilidad de visualizar, crear, actualizar y eliminar un detalle de tema. Las funciones de visualización permiten devolver en formato *JSON* la lista de los detalles de tema del usuario con sesión activa o uno en específico.

```php
//Función para la obtención de el detalle de tema del usuario con sesión activa
public function index()
{
    $detailList = ThemeDetail::all();
    $details = [];

    foreach ($detailList as $detail) {
        if($detail->user_id === auth()->user()->id){
            $details[] = $detail;
        }
    }
    return response()-> json($details, status: 200);
}

//Función para la obtención de el detalle especifico de tema del usuario con sesión activa
public function show(ThemeDetail $theme)
{
    return $theme;
}
```

Fig. 23: Función de visualización de información en modelo detalles de tema

La creación y actualización de detalles de tema deben contar con sus validaciones **Fig. 24**, las cuales brinden el control de que los datos enviados sean correctos para posterior a ello guardarlos en la base de datos.

```php
//Función para la creación de un detalle de tema
public function store(Request $request)
{
    //Validación de los campos para crear un nuevo detalle de tema
    $validatedData = $request->validate([
        'user_id' => 'required|exists:users,id',
        'theme_id' => 'required|exists:themes,id',
        'theme_advance' => 'required|string',
    ]);

    return ThemeDetail::create($request->all());
}

//Función para la actualización de un detalle de tema
public function update(Request $request, ThemeDetail $theme)
{
    //Validación de los campos para actualizar un detalle de tema
    $validatedData = $request->validate([
        'user_id' => 'exists:users,id',
        'theme_id' => 'exists:themes,id',
        'theme_advance' => 'string',
    ]);

    $theme->update($request->all());
    return response()->json($theme, status: 200);
}
```

Fig. 24: Funciones de creación y actualización del modelo detalles de tema

La función de eliminar un detalle de tema se ejecuta si se verifica que ese registro existe en la base de datos **Fig. 25**, caso contrario retornará un mensaje que mencione que ese registro no existe.

```php
//Función para la eliminación de un detalle de tema
public function delete(ThemeDetail $theme)
{
    $theme->delete();
    return response()->json( data: null, status: 204);
}
```

Fig. 25: Función de eliminar un detalle de tema en específico

Con toda la lógica de este modelo implementado en el controlador se elaboran las rutas mostradas en la **Fig. 26** las cuales pueden ser accedidas por el cliente.

```php
//Rutas para detalles de tema
Route::get( uri: '/themesDetails', [ThemeDetailController::class, 'index']);
Route::get( uri: '/themesDetails/{theme}', [ThemeDetailController::class, 'show']);
Route::post( uri: '/themesDetails', [ThemeDetailController::class, 'store']);
Route::put( uri: '/themesDetails/{theme}', [ThemeDetailController::class, 'update']);
Route::delete( uri: '/themes/{theme}', [ThemeDetailController::class, 'delete']);
```

Fig. 26: Rutas correspondientes al modelo de temas

Modificación de la migración y *seeder* modelo de usuario para generar el campo de progreso del curso.

El progreso del curso es implementado como un atributo más del modelo de usuario en su correspondiente migración **Fig. 27**, por lo cual, se debe realizar una modificación en la migración para añadir este campo en dicho modelo; el campo de progreso cuenta con un valor por defecto de 0 puesto que al inicio del curso este será el valor correspondiente.

```php
public function up()
{
    // Creación de tabla usuarios con sus respectivos atributos y valores por defecto

    Schema::create( table: 'users', function (Blueprint $table) {
        $table->bigIncrements( column: 'id');
        $table->string( column: 'name');
        $table->string( column: 'email')->unique();
        $table->string( column: 'password');
        $table->integer( column: 'progress')->nullable()->default( value: 0);
        $table->rememberToken();
        $table->timestamps();
    });
}
```

Fig. 27: Adición del atributo progreso en el modelo de usuario

Una vez realizada la modificación en la migración se debe modificar también el *seeder* como se muestra en la **Fig. 28** para que cree este campo a los usuarios ficticios que se almacenaran en la base de datos con el motivo de probar las rutas.

```php
// Generar usuarios ficticios
for ($i = 0; $i < 10; $i++) {
    User::create([
        'name' => $faker->name,
        'email' => $faker->email,
        'password' => $password,
        'progress'=> $faker->numerify( string: '###'),
    ]);
}
```

Fig. 28: Modificación del *seeder* del modelo usuario

Sprint 3. Implementación de campos para la gamificación del curso.

En relación con el *Sprint Backlog* se ejecutan las actividades descritas en el *Sprint* 3, las cuales están relacionadas con la actualización del campo de progreso, hasta la creación de nuevos campos que permitan usar gamificación en el sistema. Resultados obtenidos del *Sprint*:

- Modificación del controlador de usuario para validación de datos.
- Implementación de roles en el modelo de usuario.
- Implementación de campos de gamificación en modelo de usuario.

Modificación del controlador de usuario para validación de datos.

Debido al nuevo campo añadido de progreso del curso en el modelo de usuario, se debe controlar que este sea del tipo correcto cuando se actualice su valor por lo cual en la validación de sus datos correspondientes a la actualización del modelo de usuario se genera una revisión para dicho campo como se visualiza en la **Fig. 29**.

```
$validatedData = $request->validate([
    'progress' => 'numeric',
]);
```

Fig. 29: Validación del atributo de progreso en el modelo de usuario

Implementación de roles en el modelo de usuario.

Los roles dentro de un sistema permiten controlan las acciones que cada usuario dependiendo de su rol puede realizar, en el caso de este sistema existen varios roles entre los cuales destacan el rol de usuario y el rol de super administrador. Por un lado, el rol de super administrador tiene los permisos de realizar cualquier tipo de acción dentro del sistema, por otro lado, el rol de usuario posee permisos limitados que le permitan interactuar con el curso de la mejor forma posible.

La implementación de roles empieza con la definición de que roles posee el sistema, por lo cual dentro del modelo de usuario se coloca dicha implementación como se muestra en la **Fig. 30**.

```
//Definición de roles del sistema
const ROLE_SUPERADMIN = 'ROLE_SUPERADMIN';
const ROLE_ADMIN = 'ROLE_ADMIN';
const ROLE_USER = 'ROLE_USER';

private const ROLES_HIERARCHY = [
    self::ROLE_SUPERADMIN => [self::ROLE_ADMIN],
    self::ROLE_ADMIN => [self::ROLE_USER],
    self::ROLE_USER => []
];
```

Fig. 30: Definición de roles en el modelo de usuario

La lógica de verificación de que rol posee cada usuario se encuentra en una función que se coloca al final del modelo de usuario **Fig. 31**, es decir, recibe el rol con el que el usuario cuenta y proporcionará los permisos correspondientes debido a su jerarquía.

```
public function isGranted($role)
{
    if ($role === $this->role) {
        return true;
    }
    return self::isRoleInHierarchy($role, self::ROLES_HIERARCHY[$this->role]);
}
private static function isRoleInHierarchy($role, $role_hierarchy)
{
    if (in_array($role, $role_hierarchy)) {
        return true;
    }
    foreach ($role_hierarchy as $role_included) {
        if(self::isRoleInHierarchy($role,self::ROLES_HIERARCHY[$role_included]))
        {
            return true;
        }
    }
    return false;
}
```

Fig. 31: Función de jerarquía de roles

Con el objetivo de mejorar el control de roles en el sistema se añade un nuevo atributo en la migración del modelo de usuario, el cual tiene por defecto el rol de usuario **Fig. 32**, es decir, cada vez que un usuario se registre en el sistema el rol será el de un usuario.

```php
public function up()
{
    // Creación de tabla usuarios con sus respectivos atributos y valores por defecto

    Schema::create( table: 'users', function (Blueprint $table) {
        $table->bigIncrements( column: 'id');
        $table->string( column: 'name');
        $table->string( column: 'email')->unique();
        $table->string( column: 'password');
        $table->integer( column: 'progress')->nullable()->default( value: 0);
        $table->string( column: 'role')->default( value: \App\Models\User::ROLE_USER);
        $table->rememberToken();
        $table->timestamps();
    });
}
```

Fig. 32: Adición del atributo de rol en el modelo de usuario

Implementación de campos de gamificación en modelo de usuario.

La gamificación es una técnica comúnmente usada para mejorar la experiencia del usuario en una página web o a su vez inducir al usuario que use con más frecuencia dicha página. Para el caso en concreto de este curso se generan varios campos en el modelo usuario: experiencia, rango y nivel. Los campos permitirán que el curso use la gamificación para obtener los beneficios mencionados.

La implementación comienza con el aumento de los campos correspondientes en la migración del modelo usuario como se muestra en la **Fig. 33**; los campos cuentan con valores por defecto los cuales brindan una ayuda al cliente para que al momento de que un usuario se registre ya cuenta con los campos por defecto.

```php
// Creación de tabla usuarios con sus respectivos atributos y valores por defecto

Schema::create( table: 'users', function (Blueprint $table) {
    $table->bigIncrements( column: 'id');
    $table->string( column: 'name');
    $table->string( column: 'email')->unique();
    $table->string( column: 'password');
    $table->integer( column: 'experience')->nullable()->default( value: 0);
    $table->integer( column: 'progress')->nullable()->default( value: 0);
    $table->string( column: 'rank')->nullable()->default( value: 'Cabo en ciberseguridad');
    $table->integer( column: 'level')->nullable()->default( value: 0);
    $table->string( column: 'role')->default( value: \App\Models\User::ROLE_USER);
    $table->rememberToken();
    $table->timestamps();
});
```

Fig. 33: Adición de campos de gamificación en el modelo de usuario

Los *seeder* del modelo respectivo se implementan con el uso de la herramienta *faker* la cual permite generar campos ficticios de forma sencilla y que brindan la posibilidad de realizar pruebas con los datos almacenados como se plasma en la **Fig. 34**.

```php
// Generar usuarios ficticios
for ($i = 0; $i < 10; $i++) {
    User::create([
        'name' => $faker->name,
        'email' => $faker->email,
        'password' => $password,
        'experience' => $faker->numerify( string: '####'),
        'progress'=> $faker->numerify( string: '###'),
        'rank' => $faker->word,
        'level' => $faker->numerify( string: '##')
    ]);
}
```

Fig. 34: Modificación de *seeder* del modelo usuario

Por último, como se visualiza en la **Fig. 35**, dentro de la implementación de los campos de gamificación se debe validar que al momento de la actualización de cada uno de ellos los datos sean verídicos, para lo cual en la lógica de la función de actualización del controlador del modelo de usuario se colocan las restricciones correspondientes.

```php
$validatedData = $request->validate([
    'experience' => 'numeric',
    'progress' => 'numeric',
    'rank' => 'string',
    'level' => 'numeric',
]);
```

Fig. 35: Validación de campos de gamificación

Sprint 4. Implementación de modelos contenidos, logros y políticas.

Con el objetivo de cumplir las actividades descritas en el *Sprint* 4 se ejecutan dichas tareas, las cuales están relacionadas con la elaboración de toda la lógica correspondiente a los modelos de contenidos y de logros. Resultados obtenidos del *Sprint*:

- Elaboración de migración, *seeder*, controlador y rutas para la creación, visualización, actualización y eliminación del modelo de contenidos.
- Elaboración de migración, *seeder*, controlador y rutas para la creación, visualización, actualización y eliminación del modelo de logros.
- Implementación de relaciones entre los modelos de la base de datos.
- Implementación de políticas.

Elaboración de migración, *seeder*, controlador y rutas para la creación, visualización, actualización y eliminación del modelo de contenidos.

El modelo de contenidos posee varios campos en la migración correspondiente los cuales permiten generar contenidos para los temas **Fig. 36**, los cuales posean información concreta y relevante para los usuarios del curso.

```php
//Craci\u00f3n de tabla contenidos con sus respectivos atributos

Schema::create( table: 'contents', function (Blueprint $table) {
    $table->bigIncrements( column: 'id');
    $table->string( column: 'description');
    $table->string( column: 'question');
    $table->string( column: 'answer_1');
    $table->string( column: 'answer_2');
    $table->string( column: 'answer_3');
    $table->string( column: 'answer_4');
    $table->string( column: 'feedback');
    $table->string( column: 'image')->nullable();
    $table->foreignId( column: 'theme_id')
        ->references( column: 'id')
        ->on( table: 'themes')
        ->onDelete( action: 'restrict');
    $table->timestamps();
});
```

Fig. 36: Migración del modelo de contenidos

Los *seeder* de dicho modelo permiten generar datos ficticios lo cuales brindan ayuda a la visualización de los datos que se encuentran almacenados en la base de datos de los contenidos como se muestra en la **Fig. 37**.

```php
//Vaciar la tabla
Content::truncate();

//Llamada a la herramienta faker
$faker = \Faker\Factory::create();

// Obtenemos todos los temas de la base de datos
$themes = Theme::all();

//Creación de un contenido para cada tema
foreach ($themes as $theme) {
    Content::create([
        'description' => $faker->sentence,
        'question' => $faker->sentence,
        'answer_1' => $faker->sentence,
        'answer_2' => $faker->sentence,
        'answer_3' => $faker->sentence,
        'answer_4' => $faker->sentence,
        'feedback' => $faker->sentence,
        'theme_id' => $theme->id,
    ]);
}
```

Fig. 37: *Seeder* del modelo de contenidos

El controlador cuenta con la lógica de las funcionalidades de visualización, creación, actualización y eliminación de contenidos en la siguiente **Fig. 38** se muestra un ejemplo de la implementación de la creación de nuevos contenidos con sus respectivas validaciones.

```php
//Función para la actualización de un contenido
public function update(Request $request, Content $content)
{

    //Validación de los campos para actualizar un contenido
    $validatedData = $request->validate([
        'description' => 'string',
        'question' => 'string',
        'answer_1' => 'string',
        'answer_2' => 'string',
        'answer_3' => 'string',
        'answer_4' => 'string',
        'feedback' => 'string',
        'image' => 'image|dimensions:min_width=200,min_height=200',
        'theme_id'=> 'exists:themes,id',
    ]);

    $content->update($request->all());
    return response()->json($content, status: 200);
}
```

Fig. 38: Validación de datos en función de actualización del modelo de contenidos

La creación de rutas correspondientes al modelo de contenidos se las implementa en el archivo api.php **Fig. 39**, las cuales permitirán realizar todas las acciones al cliente correspondientes a dicho modelo.

```php
//Rutas para contenidos
Route::get( uri: '/contents', [ContentController::class, 'index']);
Route::get( uri: '/contents/{content}', [ContentController::class, 'show']);
Route::post( uri: '/contents', [ContentController::class, 'store']);
Route::put( uri: '/contents/{content}', [ContentController::class, 'update']);
Route::delete( uri: '/contents/{content}', [ContentController::class, 'delete']);
Route::get( uri: '/themes/{theme}/contents', [ContentController::class, 'contents']);
```

Fig. 39: Rutas correspondientes al modelo de contenidos

Elaboración de migración, *seeder*, controlador y rutas para la creación, visualización, actualización y eliminación del modelo de logros.

El modelo de logros cuenta con varios campos en su migración los cuales ofrecen al usuario una interfaz llamativa cuando un tema sea completado **Fig. 40**, estos campos corresponden a una imagen, una lista de ítems que corresponden a los conocimientos obtenidos y un título que incentive al usuario a proseguir con el curso.

```php
// Creación de tabla logros con sus respectivos atributos

Schema::create( table: 'achievements', function (Blueprint $table) {
    $table->bigIncrements( column: 'id');
    $table->string( column: 'title');
    $table->string( column: 'item_1');
    $table->string( column: 'item_2');
    $table->string( column: 'item_3');
    $table->string( column: 'item_4');
    $table->string( column: 'image')->nullable();
    $table->timestamps();
});
```

Fig. 40: migración del modelo de logros

Los *seeder* del modelo de logros proporcionan datos ficticios los cuales permiten al cliente hacer uso de ellos con el objetivo de generar la interfaz antes de usar datos reales y de este modo trabajar con más eficiencia como se visualiza en la **Fig. 41**.

```php
//Vaciar la tabla
Achievement::truncate();

//Llamada a la herramienta faker
$faker = \Faker\Factory::create();

//Creación de logros ficticios
$num_achievements = 6;
for ($j = 0; $j < $num_achievements; $j++) {
    Achievement::create([
        'title' => $faker->sentence,
        'item_1' => $faker->sentence,
        'item_2' => $faker->sentence,
        'item_3' => $faker->sentence,
        'item_4' => $faker->sentence,
    ]);
}
```

Fig. 41: *Seeder* del modelo de logros

Dentro del controlador correspondiente a este modelo se genera la lógica para las rutas que son accedidas por el cliente, en la **Fig. 42** se muestra un ejemplo de la lógica de actualización de un logro con sus respectivas validaciones.

```php
//Función para la actualizacion de un logro en específico
public function update(Request $request, Achievement $achievement)
{

    //Validación de los campos para actualizar un logro
    $validatedData = $request->validate([
        'title' => 'string|unique:achievements',
        'item_1' => 'string',
        'item_2' => 'string',
        'item_3' => 'string',
        'item_4' => 'string',
        'image' => 'image|dimensions:min_width=200,min_height=200',
    ]);

    $achievement->update($request->all());
    return response()->json($achievement, status: 200);
}
```

Fig. 42: Validación de datos en función de actualización del modelo de logros

Una vez concretadas las funciones del controlador se implementan las rutas correspondientes a cada función como se visualiza en la **Fig. 43**, teniendo en cuenta la referencia al controlador y el nombre correspondiente de estas.

```php
//Rutas para logros
Route::get( uri: '/achievements', [AchievementController::class, 'index']);
Route::get( uri: '/achievements/{achievement}', [AchievementController::class, 'show']);
Route::post( uri: '/achievements', [AchievementController::class, 'store']);
Route::put( uri: '/achievements/{achievement}', [AchievementController::class, 'update']);
Route::delete( uri: '/achievements/{achievement}', [AchievementController::class, 'delete']);
```

Fig. 43: Rutas correspondientes al modelo de logros

Implementación de relaciones entre los modelos de la base de datos.

Para la implementación de relaciones es necesario tener en claro el modelo de la base de datos, con un conocimiento concreto de la base de datos se generan las relaciones en cada modelo dándole relevancia a la cardinalidad de dichas relaciones. Como resultado de las relaciones entre todas las tablas de la base de datos se generan tablas de detalles de contenido, la cual brindara información relacionada con el usuario perteneciente a ese detalle, el tema correspondiente al contenido que está cursando, el contenido correspondiente al contenido que está cursando y la fecha correspondiente a la última interacción con el curso; esta tabla también podrá ser accedida por medio de rutas que radican de lógica en el controlador del modelo correspondiente como se visualiza en la **Fig. 44**.

```php
//Rutas para detalle de contenidos
Route::get( uri: '/contentsDetails', [ContentDetailController::class, 'index']);
Route::get( uri: '/contentsDetails/{content}', [ContentDetailController::class, 'show']);
Route::post( uri: '/contentsDetails', [ContentDetailController::class, 'store']);
Route::put( uri: '/contentsDetails/{content}', [ContentDetailController::class, 'update']);
Route::delete( uri: '/achievements/{content}', [ContentDetailController::class, 'delete']);
```

Fig. 44: Rutas del modelo detalles de contenido

Por otro lado, se genera también la tabla de detalle de logro, la cual proporciona la información de último logro conseguido por el usuario, a que usuario pertenece dicho detalle, a que tema corresponde el logro obtenido y el contenido en el que se encuentra el usuario **Fig. 45**, este modelo también cuenta con sus respectivas rutas las cuales permiten usar la lógica implementada en los controladores del modelo de detalle de logro.

```php
//Rutas para detalle de logros
Route::get( uri: '/achievementsDetails', [AchievementDetailController::class, 'index']);
Route::get( uri: '/achievementsDetails/{achievement}', [AchievementDetailController::class, 'show']);
Route::post( uri: '/achievementsDetails', [AchievementDetailController::class, 'store']);
Route::put( uri: '/achievementsDetails/{achievement}', [AchievementDetailController::class, 'update']);
Route::delete( uri: '/achievementsDetails/{achievement}', [AchievementController::class, 'delete']);
```

Fig. 45: Rutas del modelo de detalles de logro

Implementación de políticas.

Dentro de un sistema web el *backend* es el encargado de la implementación de políticas, es decir, proporcionar los permisos correspondientes a cada función o lógica presente en el *backend* en relación con el rol que tenga el usuario con la sesión activa.

Como primer paso de debe implementar un *middleware* el cual permitirá dentro del sistema realizar una validación entre las rutas públicas y rutas privadas **Fig. 46**, es decir, que rutas puede hacer uso un usuario con y sin sesión activa.

```php
Route::group(['middleware' => ['jwt.verify']], function() {
```

Fig. 46: Implementación del *middleware* en archivo api.php

Laravel permite que la implementación de políticas sea sencilla, puesto que se debe ejecutar un comando para crear las políticas del modelo requerido en donde se coloca dependiendo de la función que validaciones de deben ejecutar para permitirle al usuario efectuar correctamente una acción o enviar el mensaje de que no está autorizado a realizar dicha acción. En la **Fig. 47** se muestra el ejemplo de la implementación de políticas en el

modelo de temas. Estas políticas dictaminan que se pueden crear o actualizar temas en el sistema solo si el usuario posee un rol de super administrador.

```php
/**
 * Determine whether the user can create models.
 *
 * @param  \App\Models\User  $user
 * @return \Illuminate\Auth\Access\Response|bool
 */
public function create(User $user)
{
    //Puede realizar esta accion si posee el rol de SUPERADMIN
    return $user->isGranted( role: User::ROLE_SUPERADMIN);
}

/**
 * Determine whether the user can update the model.
 *
 * @param  \App\Models\User  $user
 * @param  \App\Models\Theme  $theme
 * @return \Illuminate\Auth\Access\Response|bool
 */
public function update(User $user, Theme $theme)
{
    //Puede realizar esta accion si posee el rol de SUPERADMIN
    return $user->isGranted( role: User::ROLE_SUPERADMIN);
}
```

Fig. 47: Implementación de políticas en el modelo de temas

Una vez implementadas las políticas en el archivo correspondiente se debe hacer referencia a la política en cada una de las funciones del controlador correspondiente, en la **Fig. 48** se muestra la implementación de la referencia de las funciones de creación de temas.

```php
//Funcion para la creación de temas
public function store(Request $request)
{
    //Comprobación de permisos
    $this->authorize( ability: 'create',  arguments: Theme::class);

    //Validación de los campos para crear un nuevo tema
    $validatedData = $request->validate([
        'title' => 'required|string|unique:themes',
        'description' => 'required|string',
        'difficulty' => 'required|string',
    ]);

    $achievement = Theme::create($request->all());
    return response()->json($achievement, status: 201);

}
```

Fig. 48: Referencia de políticas en el controlador del modelo de temas

Sprint 5. Despliegue en Heroku y pruebas de aceptación.

Con el objetivo de realizar a cabalidad las tareas del Sprint Backlog, se ejecutan todas las actividades correspondientes al *Sprint* 5. Este Sprint contiene las tareas relacionadas con el despliegue en la plataforma de Heroku y la realización de pruebas de aceptación. Resultados del *Sprint*:

- Configuración inicial en Heroku.
- Configuración de variables de entorno y ejecución de migraciones en Heroku.
- Realización de pruebas de aceptación.

Configuración inicial en Heroku.

Heroku permite realizar el despliegue de un proyecto de manera rápido por medio de la línea de comandos, es necesario en primer lugar tener instalado Heroku CLI la cual es una herramienta que permite realizar dicho despliegue. Como primer paso se debe generar un archivo el cual contenga la referencia de la configuración del servidor web de Apache en la **Fig. 49** se puede observar el archivo con su correspondiente código.

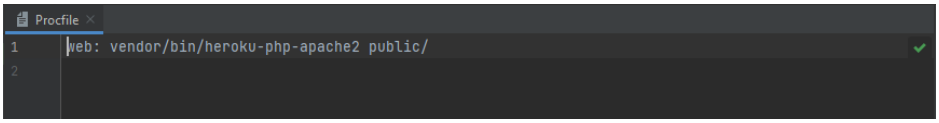


Fig. 49: Archivo Procfile

Una vez el archivo se encuentre en la raíz del proyecto de proceder a ejecutar los comandos de *git* que permiten guardar los últimos cambios realizados en la rama correspondiente. Heroku permite crear una nueva aplicación con un solo comando al cual se le puede asignar un nombre en específico, en este caso es cyberscourse, se ejecuta dicho comando; como se visualiza en la **Fig. 50** en la plataforma de Heroku se genera dicha aplicación.

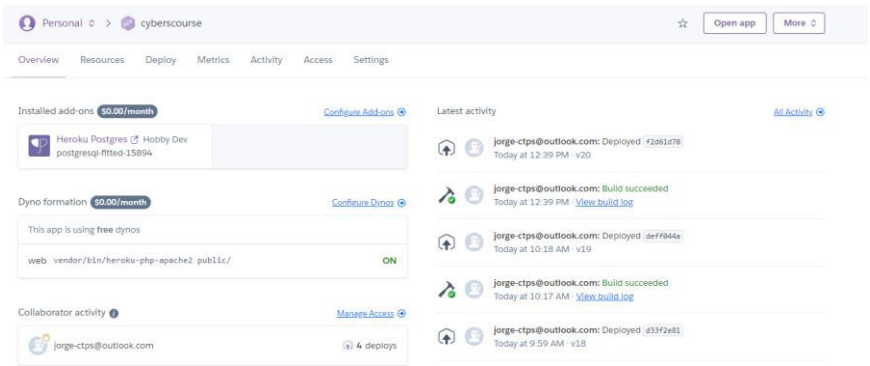


Fig. 50: Visualización de la aplicación desplegada en Heroku

Configuración de variables de entorno y ejecución de migraciones en Heroku.

Heroku es una plataforma que permite realizar un despliegue de un proyecto de software de forma gratuita, sin embargo, posee ciertas limitantes como lo es el almacenamiento de imágenes; esta plataforma no permite un almacenamiento correcto de imágenes debido a que posee in sistema de archivos efímero. Como solución se hace uso de Cloudinary la cual es una plataforma de servicios de gestión de imágenes y videos en la nube gratuita; la integración se la realiza de forma sencilla haciendo referencia a que ahora las imágenes se almacenarán en Cloudinary y configurando las variables de entorno,

El proyecto en producción hará uso de la base de datos de *PostgreSQL* puesto que Heroku permite hacer uso de esta herramienta de forma gratuita. Para conseguir un correcto despliegue también, es necesario ingresar todas las variables de entorno correspondientes y necesarias del proyecto. En la **Fig. 51** se visualizan todas las variables de entorno que son necesarias para el correcto funcionamiento del proyecto en producción.

Fig. 51: Variables de entorno en Heroku

Una vez todo este correctamente configurado se ejecutan las migraciones en producción, cabe recalcar que también se ejecuta un *seeder* puesto que corresponde a las credenciales de los usuarios super administradores. Para esto, se debe abrir la terminal de *bash* proporcionada por la plataforma de Heroku y ejecutar el comando correspondiente a las migraciones y el *seeder*, la **Fig. 52** muestra el resultado de ejecución sin fallos de dicho comando.

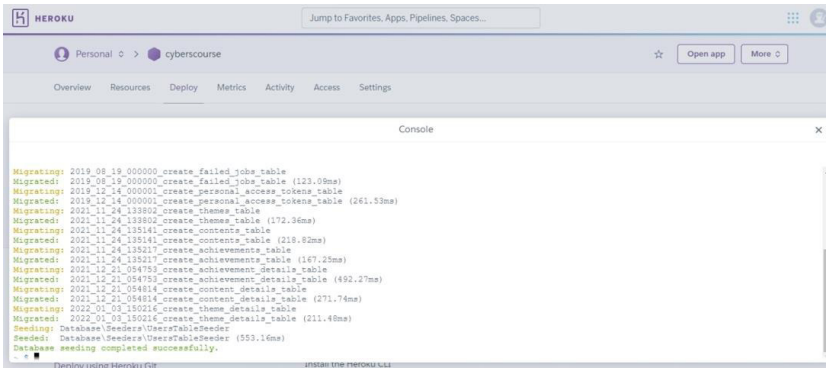


Fig. 52: Ejecución de migraciones y *seeder* en producción

Realización de pruebas de aceptación.

Las pruebas de aceptación permiten generar un listado de tablas las cuales mencionan de manera descriptiva si la tarea correspondiente a cada historia de usuario fue completada y cumple con los requerimientos del cliente. En la siguiente sección se presenta como ejemplo la ejecución de la prueba de aceptación correspondiente al requerimiento de inicio de sesión, el listado de las pruebas de aceptación completo se encuentra en el ANEXO V.

La tarea correspondiente al requerimiento de inicio de sesión se describe como la funcionalidad de permitir la autenticación en el sistema web por medio de las credenciales del correo electrónico y la contraseña. En el caso de que estos parámetros sean correctos el sistema dará dicha autenticación, por otro lado, si alguno de estos parámetros es incorrecto se debe generar un mensaje de credenciales no validas. En la **Fig. 53** se muestra la ejecución en producción de la ruta correspondiente al inicio de sesión cuando las credenciales son correctas.

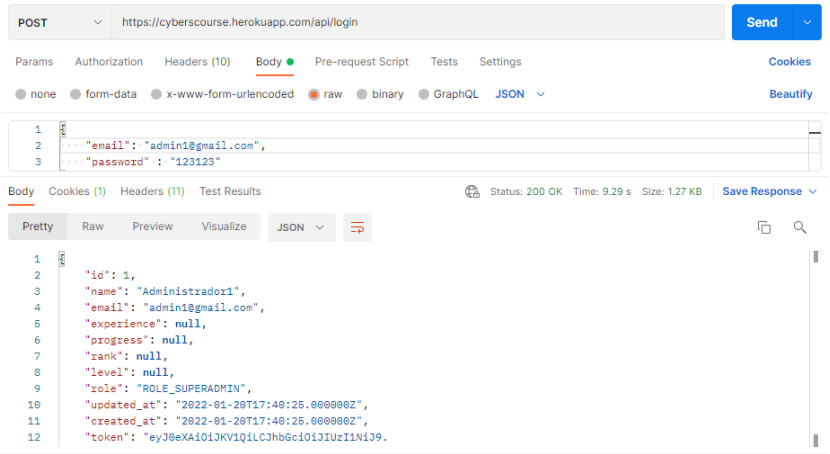


Fig. 53: Inicio de sesión con Heroku

En el caso de que una de las dos credenciales no sea la correcta, es decir, el sistema realiza el análisis y validación de los datos y no poseen coincidencia con los datos almacenados en la base de datos se mostrará un mensaje de error. La **Fig. 54** muestra el ejemplo pertinente cuando un usuario intenta iniciar sesión con una credencial de contraseña no válida.

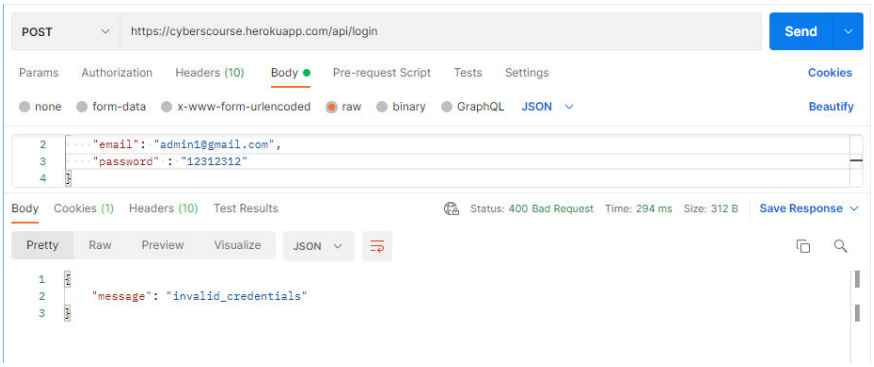


Fig. 54: Respuesta de credenciales no válidas

La **TABLA IV** muestra la prueba de aceptación correspondiente a la tarea de inicio de sesión o autenticación de usuarios. El detalle de todas las pruebas de aceptación se encuentra en el **ANEXO II**.

TABLA IV: Prueba de Aceptación PA002

	Prueba de Aceptación
Identificador: PA002	**Identificador de Historia de Usuario:** HU002
Título de la prueba de aceptación: Autenticar usuarios	
Descripción: El usuario tendrá la posibilidad de iniciar sesión completando los campos requeridos en el sistema conservando los cambios que haya realizado en el desarrollo del curso.	
Entrada/Pasos de ejecución: Ingresar a la URL del sistema web Seleccionar el botón de iniciar sesión Completar los campos: - Correo electrónico - Contraseña Seleccionar el botón de iniciar sesión	
Resultado deseado: El *backend* realiza la revisión de los datos ingresados y verifica si dichos daros se encuentran en el sistema, en el caso de ser correctos los valida y proporciona el inicio de sesión al usuario.	
Evaluación de prueba: Resultado obtenido con éxito. El *backend* valida los datos e inicia sesión en el sistema. Aprobación total del cliente.	

4 CONCLUSIONES

- El uso de una metodología ágil permite que el proyecto tenga una organización correcta de tareas a realizar, durante el desarrollo del proyecto se presentaron varios cambios los cuales fueron incluidos y ejecutados de la mejor forma posible ya que el propósito desde el inicio del proyecto fue elaborar software de calidad que proporcione un valor significativo al cliente.
- El uso del *framework* de Laravel permite que el desarrollo del proyecto se acople de buena manera a los requerimientos del cliente, debido a que, este *framework* cuenta con todas las funcionalidades necesarias para ejecutar las tareas que se generaron a partir de dichos requerimientos.
- Los objetivos planteados en el inicio del desarrollo se cumplen a cabalidad, puesto que, el proyecto desarrollado cuenta con las características que definen a una API REST; la misma que puede ser accedida por cualquier cliente que requiera desarrollar un *frontend* interactivo de un curso de una temática de interés con el uso de técnicas de gamificación.
- Durante el desarrollo del *sprint* correspondiente a las pruebas, es de gran importancia el ejecutar pruebas de aceptación; las cuales permiten verificar que cada requerimiento del usuario o cliente sea validado y cumplido por completo.
- El despliegue realizado en la plataforma de Heroku permite llevar a cabo un despliegue del proyecto de forma sencilla y rápida para que de esta forma cualquier cliente pueda hacer uso de esta. Además, el poseer el código fuente en una herramienta de control de versiones como *GitHub* permite que el entendimiento de las funcionalidades de la *API REST* sean mejor entendidas por el cliente.

5 RECOMENDACIONES

- Es de gran importancia el tener un conocimiento y entendimiento total del modelo de la base de datos y comprobar que dicho modelo trabaje de forma eficiente a la vez que cumple con totalidad los requerimientos del cliente.
- La seguridad que proporcione una *API REST* es una temática que debe estar siempre presente ya que se debe controlar tanto los datos que ingresan para ser almacenados en la base de datos, como también, que usuarios tienen los permisos necesarios para acceder a la información.
- El contacto continuo con el cliente proporciona la posibilidad de mejorar el proyecto por medio de la retroalimentación, puesto que esta situación facilita la incorporación de nuevos requerimientos que pueden generarse en etapas avanzadas.
- Es relevante para un correcto desarrollo de una *API REST* el manejar la documentación oficial de cada herramienta usada para el desarrollo de este. Los errores o dificultades con las herramientas siempre estarán presentes, sin embargo, la mejor fuente para resolver dichos errores se encuentra en sus respectivas documentaciones oficiales.

6 REFERENCIAS BIBLIOGRÁFICAS

[1] L. Abril, «Ecuador está entre los países con más ciberataques en América Latina,» EL COMERCIO, 29 Julio 2021. [En línea]. Available: https://www.elcomercio.com/tendencias/tecnologia/ecuador-ciberataques-america-latina-hacker.html. [Último acceso: 17 Diciembre 2021].

[2] L. Pazmiño, F. Flores, L. Ponce, J. Zaldumbide, V. Parraga, B. Loarte, G. Cevallos, I. Maldonado and R. Rivera, "Challenges and Opportunities of IoT Deployment in Ecuador," in *2019 International Conference on Information Systems and Software Technologies (ICI2ST)*, Quito, 2019.

[3] «Informe sobre ciberseguridad en Ecuador y Latinoamérica,» Ecuador Today, 9 Septiembre 2021. [En línea]. Available: https://ecuadortoday.media/2021/09/09/informe-sobre-ciberseguridad-en-ecuador-y-latinoamerica/. [Último acceso: 17 Diciembre 2021].

[4] C. Bleich, « Gamification In eLearning | What Works And What Doesn't?,» edgepoint, [En línea]. Available: https://www.edgepointlearning.com/blog/gamification-in-elearning/. [Último acceso: 28 Diciembre 2021].

[5] G. Velásquez, «METODOLOGÍA DE DESARROLLO DE SOFTWARE,» Universidad Católica de los Ángeles Chimbote, 19 Diciemnre 2017. [En línea]. Available: https://www.uladech.edu.pe/images/stories/universidad/documentos/2018/metodologia-desarrollo-software-v001.pdf. [Último acceso: 2 Enero 2022].

[6] J. F. Pareja Quinaluisa, «Evaluación de procesos de software utilizando EvalProSoft Aplicado a un caso de estudio,» 08 02 2012. [En línea]. Available: http://bibdigital.epn.edu.ec/handle/15000/4491 .

[7] M. Mejia, «¿Qué es el Backend y cómo usarlo?,» Crehana, 16 Febrero 2021. [En línea]. Available: https://www.crehana.com/ec/blog/desarrollo-web/que-es-el-backend-y-como-usarlo/. [Último acceso: 2 Enero 2022].

[8] C. Alvarez y F. Maroto, «La elección del estudio de caso en investigación educativa,» Gazeta de Antropología, 19 Abril 2012. [En línea]. Available: http://www.ugr.es/~pwlac/G28_14Carmen_Alvarez-JoseLuis_SanFabian. [Último acceso: 2 Enero 2022].

[9] K. Schwaber y J. Sutherland, «Las Reglas del Juego,» La Guía de Scrum, Julio 2013. [En línea]. Available: https://scrumguides.org/docs/scrumguide/v1/Scrum-Guide-ES.pdf. [Último acceso: 5 Enero 2022].

[10] C. Guerra, «Obtención de Requerimientos. Técnicas y Estrategia,» SG, [En línea]. Available: https://sg.com.mx/revista/17/obtencion-requerimientos-tecnicas-y-estrategia. [Último acceso: 5 Enero 2022].

[11] A. Álvarez, «Historias de Usuario: qué son, reglas y consejos,» netmind, 6 Julio 2020. [En línea]. Available: https://netmind.net/es/historias-de-usuario-reglas/. [Último acceso: 5 Enero 2022].

[12] «Laravel Documentation,» Laravel, [En línea]. Available: https://laravel.com/docs/8.x/readme. [Último acceso: 7 Enero 2022].

[13] J. M. Aguilar, «¿Qué es el patrón MVC en programación y por qué es útil?,» campusMVP, 15 Octubre 2019. [En línea]. Available: https://www.campusmvp.es/recursos/post/que-es-el-patron-mvc-en-programacion-y-por-que-es-util.aspx. [Último acceso: 5 Enero 2022].

[14] «Windows Preguntas frecuentes,» Apache Friends, [En línea]. Available: https://www.apachefriends.org/es/faq_windows.html. [Último acceso: 9 Enero 2022].

[15] «PostgreSQL 14.1 Documentation,» PostgreSQL, 11 Noviembre 2021. [En línea]. Available: https://www.postgresql.org/docs/. [Último acceso: 9 Enero 2022].

[16] «Create API Documentation with Postman,» API PLATFORM, [En línea]. Available: https://www.postman.com/api-documentation-tool/. [Último acceso: 9 Enero 2022].

[17] «Documentation,» Heroku Dev Center, [En línea]. Available: https://devcenter.heroku.com/categories/reference. [Último acceso: 9 Enero 2022].

[18] Y. Fernandez, «Qué es Github y qué es lo que le ofrece a los desarrolladores,» xataka, 30 Octubre 2019. [En línea]. Available: https://www.xataka.com/basics/que-github-que-que-le-ofrece-a-desarrolladores. [Último acceso: 9 Enero 2022].

[19] «Quick Start Guide,» Jetbrains, 3 Agosto 2021. [En línea]. Available: https://www.jetbrains.com/es-es/phpstorm/documentation/. [Último acceso: 12 Enero 2022].

[20] R. Rivera, L. Pazmiño, F. Becerra y J. Barriga, «An Analysis of Cyber Espionage Process,» de *Developments and Advances in Defense and Security. Proceedings of MICRADS 2021*, Cartagena, 2021.

7 ANEXOS

ANEXO I. Certificado de originalidad

ANEXO II. Manual técnico

ANEXO III. Manual de usuario

ANEXO III. Manual de instalación

Anexo I. Certificado de Originalidad

CERTIFICADO DE ORIGINALIDAD

Quito, D.M. 21 de febrero de 2022

De mi consideración:

Yo, Richard Paúl Rivera Guevara, en calidad de director del Trabajo de Integración Curricular titulado DESARROLLO DEL BACKEND asociado al proyecto DESARROLLO DE UN SISTEMA WEB PARA CAPACITACIÓN DE CIBERSEGURIDAD CON ESTRATEGIA DE GAMIFICACIÓN, elaborado por el estudiante, JORGE WASHINGTON ALBA VENEGAS 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 completo, producto del Trabajo de Integración Curricular indicado.

El documento escrito tiene un índice de similitud del 7%.

Es todo cuanto puedo certificar en honor a la verdad, pudiendo el interesado hacer uso del presente documento para los trámites de titulación.

NOTA: Se adjunta el informe generado por la herramienta Turnitin en el Quipux correspondiente.

Atentamente,

RICHARD PAUL RIVERA GUEVARA

Firmado digitalmente por RICHARD PAUL RIVERA GUEVARA
Fecha: 2022.02.21 18:12:41 -05'00'

Dr. Richard Rivera.
Profesor EPN-ESFOT

ANEXO II Manual técnico

En la siguiente sección se detallan los componentes del manual técnico del proyecto.

1. RECOLIPACION DE REQUERIMIENTOS

En esta sección se presenta la recopilación de requerimientos del proyecto visualizado en la **TABLA V**.

TABLA V: Recopilación de requerimientos

RECOPILACIÓN DE REQUERIMIENTOS	
ID - RR	**ENUNCIADO DEL ÍTEM**
RR001	Como usuario administrador necesita actualizar el contenido de los temas del por medio de una interfaz.
RR002	Como usuario administrador, necesita registrar a los usuarios finales.
RR003	Como usuario administrador, necesita autenticar a los usuarios finales.
RR004	Como usuario administrador, necesita crear los temas del curso.
RR005	Como usuario administrador, necesita actualizar la información de los temas del curso.
RR006	Como usuario administrador, necesita crear los progresos de los temas del curso.
RR007	Como usuario administrador, necesita crear los progresos del curso.
RR008	Como usuario administrador, necesita actualizar los progresos de los temas del curso.
RR009	Como usuario administrador, necesita actualizar los progresos del curso.
RR010	Como usuario administrador, necesita crear roles para el sistema web.
RR011	Como usuario administrador, necesita crear la experiencia de los usuarios finales.
RR012	Como usuario administrador, necesita actualizar la experiencia de los usuarios finales.
RR013	Como administrador, necesita crear el contenido de los temas del curso.
RR014	Como administrador, necesita actualizar el contenido de los temas del curso.

2. HISTORIAS DE USUARIO

En esta sección se presenta la lista de historias de usuario del proyecto desde la **TABLA VI** hasta la **TABLA XIX**.

TABLA VI: Historia de Usuario HU002

HISTORIA DE USUARIO	
Identificador (ID): HU002	**Usuario:** Administrador
Nombre Historia: Autenticar usuarios	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alta
Iteración Asignada: 2	
Responsable: Jorge Alba	
Descripción: El administrador deberá brindar la funcionalidad de autenticar a los usuarios finales en el *backend* que deseen iniciar sesión en el sistema.	
Observación: El formulario de inicio de sesión deberá contener los campos de email y de contraseña para efectuar la autenticación en el sistema.	

TABLA VII: Historia de Usuario HU001

HISTORIA DE USUARIO	
Identificador (ID): HU003	**Usuario:** Administrador
Nombre Historia: Crear temas del curso	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alta
Iteración Asignada: 1	
Responsable: Jorge Alba	
Descripción: El administrador deberá proporcionar la funcionalidad de crear los temas del curso en el *backend*, para que el usuario final pueda interactuar con dichos temas.	
Observación: Los cursos contaran con los campos: titulo, dificultad, imagen y progreso del tema. La dificultad de los temas está dada por: fácil, medio y difícil.	

TABLA VIII: Historia de Usuario HU004

HISTORIA DE USUARIO	
Identificador (ID): HU004	**Usuario:** Administrador
Nombre Historia: Actualizar la información de los temas del curso	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Media
Iteración Asignada: 1	
Responsable: Jorge Alba	
Descripción: El administrador deberá ofrecer la funcionalidad en el *backend* de actualizar la información de los temas del curso en el sistema.	
Observación: Las actualizaciones de los temas del curso deben ser únicamente realizadas por el administrador del sistema.	

TABLA IX: Historia de Usuario HU005

HISTORIA DE USUARIO	
Identificador (ID): HU005	**Usuario:** Administrador
Nombre Historia: Crear progresos de los temas del curso	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alta
Iteración Asignada: 2	
Responsable: Jorge Alba	
Descripción: El administrador deberá realizar en el *backend* la funcionalidad que permita mostrar al usuario los avances de los temas del curso en el sistema.	
Observación: El progreso del tema puede tener tres estados: iniciado, bloqueado y terminado.	

TABLA X: Historia de Usuario HU006

HISTORIA DE USUARIO	
Identificador (ID): HU006	**Usuario:** Administrador
Nombre Historia: Crear progresos del curso	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alta
Iteración Asignada: 2	
Responsable: Jorge Alba	
Descripción: El administrador debe proporcionar en el *backend* la funcionalidad de registrar el progreso del curso por medio de un porcentaje de avance.	
Observación: El porcentaje de avance del curso indicara al usuario final cuanto ha completado del curso y cuanto le falta para culminarlo.	

TABLA XI: Historia de Usuario HU007

HISTORIA DE USUARIO	
Identificador (ID): HU007	**Usuario:** Administrador
Nombre Historia: Actualizar el progreso de los temas del curso	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Media
Iteración Asignada: 2	
Responsable: Jorge Alba	
Descripción: El administrador debe proporcionar la funcionalidad de actualizar el progreso de los temas del curso a medida que el usuario interactúe con los mismos.	
Observación: Cuando un usuario sea nuevo en el sistema únicamente tendrá disponibles los temas de dificultad fácil habilitados, una vez que sean completados estos se habilitaran los temas de dificultad media y, por último, una vez culminados los temas de dificultad media se habilitaran los temas de dificultad difícil.	

TABLA XII: Historia de Usuario HU008

HISTORIA DE USUARIO	
Identificador (ID): HU008	**Usuario:** Administrador
Nombre Historia: Actualizar el progreso del curso	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Media
Iteración Asignada: 2	
Responsable: Jorge Alba	
Descripción: El administrador debe efectuar la funcionalidad de actualizar el progreso del curso en relación con los temas que se completen por parte del usuario final en el curso.	
Observación: Cada curso proporcionara un porcentaje el cual se ira adjuntando al porcentaje general de progreso del curso para que de este modo el usuario sea capaz de visualizar su progreso total del curso.	

TABLA XIII: Historia de Usuario HU009

HISTORIA DE USUARIO	
Identificador (ID): HU009	**Usuario:** Administrador
Nombre Historia: Crear roles	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alta
Iteración Asignada: 3	
Responsable: Jorge Alba	
Descripción: El administrador deberá generar roles de administrador y de usuario final en el *backend* para proporcionar un correcto manejo de usuarios en el sistema.	
Observación: El usuario administrador tendrá la capacidad de realizar acciones relacionadas con el manejo del sistema, mientras que el usuario final podrá interactuar con el curso y visualizar toda la información correspondiente.	

TABLA XIV: Historia de Usuario HU010

HISTORIA DE USUARIO	
Identificador (ID): HU010	**Usuario:** Administrador
Nombre Historia: Crear experiencia	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alta
Iteración Asignada: 3	
Responsable: Jorge Alba	
Descripción: El administrador deberá crear un apartado de experiencia en cada usuario el cual será proporcionado a medida que el usuario final vaya completando el curso.	
Observación: La experiencia al iniciar el curso será de 0, dicha experiencia se incrementará a medida que el usuario final vaya completando los temas de cada curso.	

TABLA XV: Historia de Usuario HU011

HISTORIA DE USUARIO	
Identificador (ID): HU011	**Usuario:** Administrador
Nombre Historia: Actualizar experiencia	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Media
Iteración Asignada: 3	
Responsable: Jorge Alba	
Descripción: El administrador deberá brindar una funcionalidad en el *backend* de actualización de experiencia de cada usuario.	
Observación: Cada tema del curso ofrecerá distinta experiencia dependiendo del nivel de dificultad.	

TABLA XVI. Historia de Usuario HU012

HISTORIA DE USUARIO	
Identificador (ID): HU012	**Usuario:** Administrador
Nombre Historia: Crear contenidos de los temas	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alta
Iteración Asignada: 1	
Responsable: Jorge Alba	
Descripción: El administrador deberá proporcionar la funcionalidad en el *backend* de crear el contenido de los temas del curso.	
Observación: El contenido de cada tema del curso cuenta con los siguientes campos: descripción, preguntas y respuestas.	

TABLA XVII: Historia de Usuario HU013

HISTORIA DE USUARIO	
Identificador (ID): HU013	**Usuario:** Administrador
Nombre Historia: Actualizar contenido de los temas	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Media
Iteración Asignada: 1	
Responsable: Jorge Alba	
Descripción: El administrador deberá ofrecer la funcionalidad de actualización de los campos correspondientes a el contenido de cada tema del curso.	
Observación: Las actualizaciones del contenido de los temas del curso deben ser únicamente realizadas por el administrador del sistema.	

TABLA XVIII: Historia de Usuario HU014

HISTORIA DE USUARIO	
Identificador (ID): HU014	**Usuario:** Administrador
Nombre Historia: Crear logros	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Media
Iteración Asignada: 1	
Responsable: Jorge Alba	
Descripción: El administrador deberá ofrecer la funcionalidad de actualización de los campos correspondientes a los logros del curso.	
Observación: Las actualizaciones los logros del curso deben ser únicamente realizadas por el administrador del sistema.	

TABLA XIX: Historia de Usuario HU015

HISTORIA DE USUARIO	
Identificador (ID): HU015	**Usuario:** Administrador
Nombre Historia: Actualizar logros	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Media
Iteración Asignada: 1	
Responsable: Jorge Alba	
Descripción: El administrador deberá ofrecer la funcionalidad de actualización de los campos correspondientes a los logros del curso.	
Observación: Las actualizaciones del contenido de los logros del curso deben ser únicamente realizadas por el administrador del sistema.	

3. PRODUCT BACKLOG

En esta sección se presenta el *Product Backlog* del proyecto como se muestra en la **TABLA XX**.

TABLA XX: *Product Backlog*

ELABORACIÓN DEL *PRODUCT BACKLOG*				
ID-HU	**HISTORIA DE USUARIO**	**ITERACIÓN**	**ESTADO**	**PRIORIDAD**
HU001	Registrar usuarios	1	Finalizado	Media
HU002	Autenticar usuarios	1	Finalizado	Media
HU003	Crear temas del curso	2	Finalizado	Alta
HU004	Actualizar la información de los temas del curso	2	Finalizado	Alta
HU005	Crear progresos de los temas del curso	2	Finalizado	Alta
HU006	Crear progresos del curso	2	Finalizado	Alta
HU007	Actualizar el progreso de los temas del curso	2	Finalizado	Alta
HU008	Actualizar el progreso del curso	3	Finalizado	Alta
HU009	Crear roles	3	Finalizado	Media
HU010	Crear experiencia	3	Finalizado	Baja
HU011	Actualizar experiencia	3	Finalizado	Baja
HU012	Crear contenido de los temas	4	Finalizado	Alta
HU013	Actualizar contenido de los temas	4	Finalizado	Alta
HU014	Crear logros	4	Finalizado	Alta
HU015	Actualizar logros	4	Finalizado	Alta

4. SPRINT BACKLOG

En esta sección se presenta el Sprint Backlog correspondiente al proyecto como se muestra en la **TABLA XXI**.

TABLA XXI: *Sprint Backlog*

ELABORACIÓN DEL *SPRINT BACKLOG*						
ID-SB	**NOMBRE**	**MÓDULO**	**ID-HU**	**HISTORIA DE USUARIO**	**TAREAS**	**TIEMPO ESTIMADO**
SB00	Acondicionamiento del ambiente de trabajo				• Realizar la recopilación de requerimientos por medio de una reunión con el cliente. • Analizar los requerimientos obtenidos de la reunión con el cliente. • Realizar las historias de usuario basándose en los requerimientos. • Instalación y actualización de herramientas para el desarrollo del proyecto. • Realización del modelo entidad relación.	50 H
SB01	Registrar usuarios		HU001	Registrar usuarios	• Elaboración de migración y modelo de usuario. • Elaboración de *seeder* para modelo de Usuario. • Implementación función de registro en controlador para modelo de usuario. • Validación de campos para registro de usuarios. • Elaboración de ruta de registro de usuarios.	60 H

					• Elaboración de funciones para visualización de datos del modelo de usuario. • Elaboración de rutas para la visualización de datos del modelo de usuario.	
	Autenticar usuarios		HU002	Autenticar usuarios	• Elaboración de función de autenticación en controlador para modelo de usuario. • Validación de campos para la autenticación de usuarios. • Implementación de funciones para actualizar, eliminar usuarios. • Elaboración de ruta de inicio de sesión de usuarios.	
SB02	Implementación de CRUD del modelo de temas.	Crear temas del curso	HU003	Crear temas del curso	• Elaboración de migraciones para el modelo de temas. • Elaboración de *seeder* para modelo de temas. • Implementación de función para la creación de temas en el controlador de modelo de temas. • Validación de campos para creación de temas en modelo de temas. • Elaboración de ruta para creación de temas en el modelo de temas. • Elaboración de funciones para visualización de datos del modelo de temas.	70 H

					• Elaboración de rutas para la visualización de datos del modelo de temas.	
		Actualizar los temas del curso	HU004	Actualizar los temas del curso	• Implementación de funciones para la actualización y eliminación de temas en el controlador de modelo de temas. • Validación de campos para la actualización de un tema. • Elaboración de rutas para actualización y eliminación de temas en el modelo de temas.	
	Implementación de CRUD del modelo de detalle de tema.	Crear progresos de los temas del curso.	HU005	Crear progresos de los temas del curso.	• Elaboración de migración y modelo de detalle de temas. • Implementación funciones de visualizar y crear detalles de tema en el controlador para modelo de detalle de temas. • Validación de campos para creación el modelo detalle de temas. • Elaboración de rutas para visualizar y crear datos del modelo detalle de temas.	
	Implementación de CRUD del modelo de Usuario.	Crear progresos del curso.	HU006	Crear progresos del curso.	• Modificación de la migración de modelo de usuario para incluir el campo de progreso del curso. • Modificación del *seeder* de usuario para el campo progreso del curso.	

					• Validación de nuevo campo en función de resgistro del controlador de modelo de usuario.	
	Implementación de CRUD del modelo de detalle de temas.	Actualizar el progreso de los temas del curso.	HU007	Actualizar el progreso de los temas del curso.	• Implementación de función para la actualización y eliminación de detalle de temas en el controlador de modelo de detalle de temas. • Validación de campos para la actualización de un detalle de tema. • Elaboración de ruta para actualizar y eliminar detalles de temas en el modelo de detalle de temas.	
SB03	Implementación de CRUD del modelo de usuario.	Actualizar el progreso del curso.	HU008	Actualizar el progreso del curso.	• Validación de nuevo campo en función de actualización del controlador de modelo de usuario.	40 H
		Crear roles.	HU09	Crear roles.	• Modificación de la migración de modelo de usuario para incluir el campo de rol. • Modificación del *seeder* de usuario para el campo de rol. • Validación de nuevo campo en función de registro del controlador de modelo de usuario.	
		Crear experiencia.	HU010	Crear experiencia.	• Modificación de la migración de modelo de usuario para incluir los campos de experiencia, rango y nivel. • Modificación del *seeder* de usuario para los campos de experiencia, rango y nivel.	

					• Validación de nuevos campos en función de registro el campo de rol del controlador de modelo de usuario.	
		Actualizar experiencia	HU011	Actualizar experiencia	• Validación de nuevo campo en función de actualización del controlador de modelo de usuario.	
SB04	Implementación de CRUD del modelo de contenido.	Crear contenido de los temas.	HU012	Crear contenido de los temas.	• Elaboración de migraciones para el modelo de contenidos. • Elaboración de *seeder* para modelo de contenidos. • Implementación de función para la creación de contenidos en el controlador de modelo de contenidos. • Validación de campos para creación de contenidos en modelo de contenidos. • Elaboración de ruta para creación de contenidos en el modelo de contenidos. • Elaboración de funciones para visualización de datos del modelo de contenidos. • Elaboración de rutas para la visualización de datos del modelo de contenidos.	100 H
		Actualizar contenido de los temas.	HU013	Actualizar contenido de los temas.	• Implementación de funciones para la actualización y eliminación de contenidos en el controlador de modelo de contenidos.	

					• Validación de campos para la actualización de un contenido. • Elaboración de rutas para actualización y eliminación de contenidos en el modelo de contenidos.	
	Implementación de CRUD del modelo de logro.	Crear logros	HU014	Crear logros	• Elaboración de migraciones para el modelo de logros. • Elaboración de *seeder* para modelo de logros. • Implementación de función para la creación de logros en el controlador de modelo de logros. • Validación de campos para creación de logros en modelo de logros. • Elaboración de ruta para creación de logros en el modelo de logros. • Elaboración de funciones para visualización de datos del modelo de logros. Elaboración de rutas para la visualización de datos del modelo de logros.	
		Actualizar logros	HU015	Actualizar logros	• Implementación de funciones para la actualización y eliminación de logros en el controlador de modelo de logros. • Validación de campos para la actualización de un logro.	

					• Elaboración de rutas para actualización y eliminación de logros en el modelo de logros.	
	Implementación de relaciones entre modelos de la base de datos.				• Elaboración de migraciones para modelos detalle de logro y detalle de contenido. • Implementación de funcionalidades CRUD de los modelos en sus respectivos controladores • Validación de campos los modelos. • Elaboración de rutas CRUD para los modelos.	
	Implementación de políticas en modelos de la base de datos.				• Implementación de políticas en todos los modelos de la base de datos con sus respectivos permisos. • Implementación de uso de las políticas creadas en los controladores de cada modelo de la base de datos.	
SB05	Pruebas y despliegue.				• Realizar pruebas de aceptación. • Desplegar la *API* en Heroku.	90 H

5. PRUEBAS DE ACEPTACIÓN

En la siguiente sección se detalla la lista de pruebas de aceptación del proyecto desde la **TABLA XXII** hasta la **TABLA XXXV**.

TABLA XXII: Prueba de Aceptación PA001

Prueba de Aceptación	
Identificador: PA001	**Identificador de Historia de Usuario:** HU001
Título de la prueba de aceptación: Registrar usuarios	
Descripción: El usuario tendrá la posibilidad de registrarse en el sistema por medio del envío de los datos correspondientes al formulario de registro.	
Entrada/Pasos de ejecución: Ingresar a la URL del sistema web Seleccionar el botón de registrarse Completar los campos: - Nombre - Correo electrónico - Contraseña - Confirmación de contraseña Seleccionar el botón de registrarse	
Resultado esperado: El *backend* analiza los datos que se enviarán y valida que cada uno de ellos sea del tipo correcto, en el caso de que sean datos validos estos se almacenarán en la base de datos y el usuario quedará registrado en el sistema.	
Evaluación de prueba: Resultado obtenido con éxito. El backend valida los datos y registra al usuario en el sistema. Aprobación total del cliente.	

TABLA XXIII: Prueba de Aceptación PA003

Prueba de Aceptación	
Identificador: PA003	**Identificador de Historia de Usuario:** HU003
Título de la prueba de aceptación: Crear temas del curso	
Descripción: El usuario con la sesión activa correspondiente al rol de super administrador podrá enviar los datos correspondientes a la creación de temas.	
Entrada/Pasos de ejecución: Iniciar sesión como super administrador Generar un *JSON* con los campos: - Titulo - Dificultad - Descripción Enviar los datos	
Resultado deseado: El sistema de *backend* verifica los datos correspondientes, si los datos son válidos almacena el tema creado en la base de datos.	
Evaluación de prueba: Resultado obtenido con éxito. El *backend* valida los datos y almacena los temas creados. Aprobación total del cliente.	

TABLA XXIV: Prueba de Aceptación PA004

Prueba de Aceptación	
Identificador: PA004	**Identificador de Historia de Usuario:** HU004
Título de la prueba de aceptación: Actualizar la información de los temas del curso	
Descripción: El usuario con la sesión activa correspondiente al rol de super administrador podrá actualizar la información de un tema en específico.	
Entrada/Pasos de ejecución: Iniciar sesión como super administrador Generar un *JSON* con los campos que se desee actualizar Enviar los datos	
Resultado deseado: El sistema de *backend* verifica los datos correspondientes, si los datos son válidos almacena los cambios del tema seleccionado en la base de datos.	
Evaluación de prueba: Resultado obtenido con éxito. El *backend* valida los datos y almacena los cambios en el registro del tema seleccionado Aprobación total del cliente.	

TABLA XXV: Prueba de Aceptación PA005

Prueba de Aceptación	
Identificador: PA005	**Identificador de Historia de Usuario:** HU005
Título de la prueba de aceptación: Crear progresos de los temas del curso	
Descripción: Cada usuario que se registre en el sistema tendrá un campo correspondiente al avance de cada tema del curso el cual mostrara el estado de cada tema con la característica de iniciado, bloqueado o terminado.	
Entrada/Pasos de ejecución: Ingresar a la URL del sistema web Seleccionar el botón de registrarse Completar los campos: - Nombre - Correo electrónico - Contraseña - Confirmar contraseña Seleccionar el botón de registrarse	
Resultado deseado: El sistema de *backend* proporciona la tabla de detalles de tema en la base de datos la cual brinda la información de a que usuario corresponde, el tema correspondiente y el progreso de cada tema.	
Evaluación de prueba: Resultado obtenido con éxito. El *backend* valida los datos y almacena los datos actualizados del tema seleccionado. Aprobación total del cliente.	

TABLA XXVI: Prueba de Aceptación PA006

Prueba de Aceptación	
Identificador: PA006	**Identificador de Historia de Usuario:** HU006
Título de la prueba de aceptación: Crear los progresos del curso	
Descripción: Cada usuario registrado obtendrá un atributo por defecto denominado como progreso del curso el cual muestra de forma numérica que porcentaje del curso ha completado.	
Entrada/Pasos de ejecución: Ingresar a la URL del sistema web Seleccionar el botón de registrarse Completar los campos: - Nombre - Correo electrónico - Contraseña - Confirmar contraseña Seleccionar el botón de registrarse	
Resultado deseado: El sistema de *backend* se encarga de generar cada vez que se registre un nuevo usuario el atributo de progreso del curso en la tabla del modelo de usuario en la base de datos con un valor por defecto de 0.	
Evaluación de prueba: Resultado obtenido con éxito. El *backend* genera de forma automática el nuevo atributo. Aprobación total del cliente.	

TABLA XXVII: Prueba de Aceptación PA007

Prueba de Aceptación	
Identificador: PA007	**Identificador de Historia de Usuario:** HU007
Título de la prueba de aceptación: Actualizar el progreso de los temas del curso	
Descripción: Cuando un usuario avance en el curso por medio de la culminación de temas, los avances de cada curso se actualizarán y cambiarán de iniciado a terminado.	
Entrada/Pasos de ejecución: Ingresar a la URL del sistema web Seleccionar el botón de iniciar sesión Completar los campos: - Correo electrónico - Contraseña Seleccionar el tema disponible Completar los contenidos del tema Seleccionar el botón de conseguir logro	
Resultado deseado: El momento que un usuario culmina un tema del curso el *backend* posibilita la opción de actualizar el avance siempre y cuando el valor enviado sea válido y lo almacena en la base de datos cambiándolo de iniciado a culminado.	
Evaluación de prueba: Resultado obtenido con éxito. El *backend* almacena la actualización del avance del tema culminado. Aprobación total del cliente.	

TABLA XXVIII: Prueba de Aceptación PA008

Prueba de Aceptación	
Identificador: PA008	**Identificador de Historia de Usuario:** HU008
Título de la prueba de aceptación: Actualizar el progreso del curso	
Descripción: A medida que el usuario del curso avance el atributo de progreso del curso debe actualizarse en relación con la lógica de la cantidad de temas.	
Entrada/Pasos de ejecución: Ingresar a la URL del sistema web Seleccionar el botón de iniciar sesión Completar los campos: - Correo electrónico - Contraseña Seleccionar el tema disponible Completar los contenidos del tema Seleccionar el botón de conseguir logro	
Resultado deseado: El *backend* permite que realizar la actualización del atributo de progreso del curso cada siempre si es un dato válido.	
Evaluación de prueba: Resultado obtenido con éxito. El *backend* valida los datos y almacena en el registro correspondiente. Aprobación total del cliente.	

TABLA XXIX: Prueba de Aceptación PA009

Prueba de Aceptación	
Identificador: PA009	**Identificador de Historia de Usuario:** HU009
Título de la prueba de aceptación: Crear roles	
Descripción: El sistema cuenta con los roles de: super administrador y usuario.	
Entrada/Pasos de ejecución: Rol de super administrador: Modificar el *seeder* del modelo de usuarios Generar los usuarios super administradores Correr el *seeder* Rol de usuario: Seleccionar el botón de registrarse Completar los campos: - Nombre - Correo electrónico - Contraseña - Confirmar contraseña	
Resultado deseado: Los usuarios con rol de super administrador se pueden generar en el *seeder* correspondiente, por otro lado, a los usuarios que se registren en el sistema se les proporcionará por defecto el rol de usuario.	
Evaluación de prueba: Resultado obtenido con éxito. El *backend* permite generar usuarios con los diferentes roles. Aprobación total del cliente.	

TABLA XXX: Prueba de Aceptación PA010

Prueba de Aceptación	
Identificador: PA010	**Identificador de Historia de Usuario:** HU010
Título de la prueba de aceptación: Crear experiencia	
Descripción: El sistema hace uso de técnicas de gamificación por lo cual los usuarios registrados contaran con varios atributos: experiencia, rango y nivel que justifiquen la gamificación del sistema.	
Entrada/Pasos de ejecución: Ingresar a la URL del sistema web Seleccionar el botón de registrarse Completar los campos: - Nombre - Correo electrónico - Contraseña - Confirmar contraseña Seleccionar el botón de registrarse	
Resultado deseado: El *backend* proporciona los atributos de gamificación una vez que un usuario se registre en el sistema y proporciona sus valores por defecto correspondientes.	
Evaluación de prueba: Resultado obtenido con éxito. El *backend* genera los atributos de gamificación con sus valores por defecto. Aprobación total del cliente.	

TABLA XXXI: Prueba de Aceptación PA011

Prueba de Aceptación	
Identificador: PA011	**Identificador de Historia de Usuario:** HU011
Título de la prueba de aceptación: Actualizar experiencia	
Descripción: Los atributos de gamificación se actualizarán en la base de datos en relación a la lógica implementada por el cliente de la API REST.	
Entrada/Pasos de ejecución: Calcular los datos de gamificación Enviar los datos	
Resultado deseado: De acuerdo a la lógica del cliente de la *API REST* el *backend* permite actualizar dichos datos, si son válidos, y los almacenan en la base de datos.	
Evaluación de prueba: Resultado obtenido con éxito. El *backend* valida los nuevos datos y los almacena en el registro del usuario correspondiente. Aprobación total del cliente.	

TABLA XXXII: Prueba de Aceptación PA012

Prueba de Aceptación	
Identificador: PA012	**Identificador de Historia de Usuario:** HU012
Título de la prueba de aceptación: Actualizar contenido de los temas	
Descripción: El *backend* recibirá los datos correspondientes a los contenidos de los temas desde el cliente y deberá almacenar los datos siempre y cuando sean válidos.	
Entrada/Pasos de ejecución: Iniciar sesión como super administrador Generar un *JSON* con los campos: - Descripción - Pregunta - Respuesta 1 - Respuesta 2 - Respuesta 3 - Respuesta 4 - *Feedback* - Imagen - Id del tema al que pertenece Enviar los datos	
Resultado deseado: El *backend* realiza la verificación de los datos enviados, en el caso de ser correctos y validos los almacena en la base de datos.	
Evaluación de prueba: Resultado obtenido con éxito. El *backend* valida los datos y almacena los contenidos creados. Aprobación total del cliente.	

TABLA XXXIII Prueba de Aceptación PA013

Prueba de Aceptación	
Identificador: PA013	**Identificador de Historia de Usuario:** HU013
Título de la prueba de aceptación: Actualizar contenido de los temas	
Descripción: En el caso de que el usuario con rol de super administrador dese actualizar un contenido en específico enviará los datos seleccionados para actualizar y deberán almacenarse en la base de datos.	
Entrada/Pasos de ejecución: Iniciar sesión como super administrador Generar un *JSON* con los campos que se desee actualizar Enviar los datos	
Resultado deseado: El *backend* realiza la validación de los campos del contenido en específico que se actualizará y en el caso que sean correctos se almacenaran los cambios en la base de datos.	
Evaluación de prueba: Resultado obtenido con éxito. El *backend* valida los datos y almacena los cambios en el registro del tema seleccionado. Aprobación total del cliente.	

TABLA XXXIV: Prueba de Aceptación PA014

Prueba de Aceptación	
Identificador: PA014	**Identificador de Historia de Usuario:** HU014
Título de la prueba de aceptación: Crear logros	
Descripción: El *backend* recibirá los datos correspondientes a los logros de los temas desde el cliente y deberá almacenar los datos siempre y cuando sean válidos.	
Entrada/Pasos de ejecución: Iniciar sesión como super administrador Generar un *JSON* con los campos: - Titulo - *Item* 1 - *Item* 2 - *Item* 3 - *Item* 4 - Imagen Enviar los datos	
Resultado deseado: El *backend* realiza la verificación de los datos enviados, en el caso de ser correctos y validos los almacena en la base de datos	
Evaluación de prueba: Resultado obtenido con éxito. El *backend* valida los datos y almacena los logros creados. Aprobación total del cliente.	

TABLA XXXV: Prueba de Aceptación PA015

Prueba de Aceptación	
Identificador: PA015	**Identificador de Historia de Usuario:** HU015
Título de la prueba de aceptación: Actualizar logros	
Descripción: En el caso de que el usuario con rol de super administrador dese actualizar un logro en específico enviará los datos seleccionados para actualizar y deberán almacenarse en la base de datos.	
Entrada/Pasos de ejecución: Iniciar sesión como super administrador Generar un *JSON* con los campos que se desee actualizar Enviar los datos	
Resultado deseado: El *backend* realiza la validación de los campos del logro en específico que se actualizará y en el caso que sean correctos se almacenaran los cambios en la base de datos.	
Evaluación de prueba: Resultado obtenido con éxito. El *backend* valida los datos y almacena los cambios en el registro del logro seleccionado. Aprobación total del cliente.	

ANEXO III Manual de usuario

El video correspondiente al manual de usuario del sistema puede ser accedido desde el siguiente enlace: https://www.youtube.com/watch?v=WdfkKlwNySw

ANEXO IV Manual de instalación

El código fuente generado a lo largo del desarrollo de este proyecto, así como las instrucciones de instalación del proyecto se encuentra en la plataforma de GitHub. A estos se puede acceder a través del siguiente enlace: https://github.com/jorgew1998/Backend-PT.