**UNIVERSIDAD CATÓLICA DE EL SALVADOR**

**FACULTAD DE INGENIERÍA Y ARQUITECTURA**



**INFORME FINAL DE PROYECTO**

**DESARROLLO DE API-REST PARA LA GESTIÓN DE USUARIOS Y EVALUACIÓN DE DESEMPEÑO DE PERSONAL, SISTEMAS PLIRET**

**PERÍODO DE REALIZACIÓN:**

**DEL 28 DE AGOSTO DE 2023 AL 31 DE ENERO DE 2024**

**PRESENTADO POR:**

**ROBERTO ANTONIO MORENO ZEPEDA**

**PARA OPTAR AL TÍTULO DE:**

**INGENIERO EN DESARROLLO DE SOFTWARE**

**SANTA ANA, FEBRERO DE 2024**

**AUTORIDADES UNIVERSITARIAS**

**MONSEÑOR Y LICENCIADO MIGUEL ÁNGEL MORÁN AQUINO**

**RECTOR**

**DOCTOR MOISÉS ANTONIO MARTÍNEZ ZALDÍVAR**

**VICERRECTOR GENERAL**

**MAESTRO MOISÉS ULISES GARCÍA PERDIDO**

**SECRETARIO GENERAL**

**MAESTRO MAURICIO ERNESTO VELÁSQUEZ SORIANO**

**DECANO DE LA FACULTAD DE INGENIERÍA Y ARQUITECTURA**

**SANTA ANA, FEBRERO 2024**

**AGRADECIMIENTOS**

En primer lugar, le agradezco a Dios por ser mi guía y faro durante el transcurso de mi vida y más pertinentemente a mi periodo universitario, por nunca dejarme atrás y siempre cuidar de mí.

A mi madre: Ana Carolina de Moreno Por siempre estar junto en todo momento en mi vida y más especialmente durante esta etapa de aprendizaje y crecimiento profesional. Por transmitirme sus valores, su cariño, su amor y por la fe que ha puesto en mí.

A mi padre: José Luis Moreno Chacón (Q.E.P.D.) por proveerme de todo y cuando necesite estando en vida y por todas las enseñanzas que me dejo, y por velar por mi luego de su partida.

A mi hermano mayor: Por el apoyo brindado durante mi vida y especialmente agradecido por esta etapa que estoy por terminar, su apoyo incondicional y paciencia para orientarme.

A mi hermano menor: Toby por las incontables noches que paso en desvelo junto a mí, por brindarme el mejor estado de ánimo durante mi vida y más especialmente durante esta etapa.

A la familia que me apoyó de una u otra forma queriendo hacer énfasis en mi tía Ligia por siempre ayudarme en los favores que así lo necesite, su paciencia y su fe puesta en mí.

A mis compañeros que me han acompañado durante estos 5 años especialmente a Balmore Alejandro Landaverde Romero por orientarme durante el transcurso de toda la carrera y aportarme sus conocimientos para llegar a ser un mejor profesional.

Finalmente agradezco a cada una de las personas que directa o indirectamente me han ayudado en esta etapa de mi vida y confiaron en mí, estaré eternamente agradecido.

**Roberto Antonio Moreno Zepeda**

**Contenido**

[INTRODUCCIÓN 10](#_Toc159609954)

[Ubicación geográfica 11](#_Toc159609955)

[Misión 11](#_Toc159609956)

[Visión 12](#_Toc159609957)

[Valores 12](#_Toc159609958)

[Objetivo General 17](#_Toc159609959)

[Objetivos Específicos 17](#_Toc159609960)

[DESCRIPCIÓN NARRATIVA DE LOS OBJETIVOS 17](#_Toc159609961)

[CRONOGRAMA 24](#_Toc159609962)

[Cronograma en formato texto 24](#_Toc159609963)

[Fase 1: 24](#_Toc159609964)

[Fase 2: 24](#_Toc159609965)

[Fase 3: 25](#_Toc159609966)

[Fase 4: 25](#_Toc159609967)

[Cronograma por presupuesto 26](#_Toc159609968)

[Tabla de herramientas y costos 28](#_Toc159609969)

[Recursos 29](#_Toc159609970)

[Recursos humanos 29](#_Toc159609971)

[Recursos de hardware. 30](#_Toc159609972)

[Recursos de software para el desarrollo del sistema 30](#_Toc159609973)

[Requerimientos funcionales: 31](#_Toc159609974)

[Requisitos no funcionales 32](#_Toc159609975)

[DESCRIPCCIÓN DE LA API 32](#_Toc159609976)

[Manager: 34](#_Toc159609977)

[Developer: 35](#_Toc159609978)

[Trainer: 35](#_Toc159609979)

[Trainee: 35](#_Toc159609980)

[Generales: 36](#_Toc159609981)

[Tecnologías utilizadas: 36](#_Toc159609982)

[Ingles 37](#_Toc159609983)

[Laravel 37](#_Toc159609984)

[MySQL 39](#_Toc159609985)

[Control de versiones 39](#_Toc159609986)

[Diagrama Entidad Relación 41](#_Toc159609987)

[Tablas 42](#_Toc159609988)

[Entidades 42](#_Toc159609989)

[Usuario 42](#_Toc159609990)

[Rol-Usuario 43](#_Toc159609991)

[Roles 43](#_Toc159609992)

[Habilidad-Rol 43](#_Toc159609993)

[Etiquetas 43](#_Toc159609994)

[Programa-Etiqueta 44](#_Toc159609995)

[Programas 44](#_Toc159609996)

[Módulos 44](#_Toc159609997)

[Temas 44](#_Toc159609998)

[Preguntas 45](#_Toc159609999)

[Criterios de Evaluación 45](#_Toc159610000)

[Notas 46](#_Toc159610001)

[Desarrolladores 46](#_Toc159610002)

[Entrenadores 46](#_Toc159610003)

[Ejecuciones 47](#_Toc159610004)

[Asignaciones 47](#_Toc159610005)

[Enrolamientos 48](#_Toc159610006)

[Llaves primarias 48](#_Toc159610007)

[Cardinalidad 49](#_Toc159610008)

[Endpoints disponibles para el usuario de tipo manager 50](#_Toc159610009)

[Obtener lista de programas: /api/v1/programs/{id?} 50](#_Toc159610010)

[Crear un programa: /api/v1/programs 50](#_Toc159610011)

[Crear una ejecución: /api/v1/executions 51](#_Toc159610012)

[Enrolar un trainee a una ejecución: /api/v1/executions/{id}/enroll-trainee/{id} 52](#_Toc159610013)

[Obtener lista de ejecuciones: /api/v1/executions/{id?} 52](#_Toc159610014)

[Asignar un módulo a una ejecución: /api/v1/executions/assign-trainee-module 53](#_Toc159610015)

[Crear un empleado: /api/v1/users/create-employee-account 54](#_Toc159610016)

[Obtener lista de programas: /api/v1/programs/{id?} 55](#_Toc159610017)

[Obtener lista de programas: /api/v1/programs/{id?} 55](#_Toc159610018)

[Obtener lista de programas: /api/v1/programs/{id?} 56](#_Toc159610019)

[Obtener lista de programas: /api/v1/programs/{id?} 56](#_Toc159610020)

[Obtener lista de programas: /api/v1/programs/{id?} 57](#_Toc159610021)

[Obtener lista de programas: /api/v1/programs/{id?} 57](#_Toc159610022)

[Obtener lista de programas: /api/v1/programs/{id?} 58](#_Toc159610023)

[Glosario 63](#_Toc159610024)

**INDICE DE IMÁGENES**

[Ilustración 1: Cronograma de actividades 24](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159610025)

[*Ilustración 2: Cronograma de por presupuesto agosto a septiembre* 26](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159610026)

[Ilustración 3: Cronograma de por presupuesto octubre a noviembre 27](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159610027)

[Ilustración 4: Cronograma de por presupuesto diciembre a enero 27](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159610028)

[Ilustración 5: Diagrama ERD 41](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159610029)

[Ilustración 6: Respuesta del endpoint /api/v1/programs/{id?} para manager 50](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159610030)

[Ilustración 7: Respuesta del endpoint /api/v1/programs para manager 51](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159610031)

[Ilustración 8: Respuesta del endpoint /api/v1/executions para manager 51](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159610032)

[Ilustración 9: Respuesta del endpoint /api/v1/executions/{id}/enroll-trainee/{id} para manager 52](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159610033)

[Ilustración 10: Respuesta del endpoint /api/v1/executions/{id?} para manager 53](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159610034)

[Ilustración 11: Respuesta del endpoint /api/v1/executions/assign-trainee-module 54](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159610035)

**Titulo**

[Tabla 1: Definición y análisis de herramientas con sus costos 30](#_Toc159610036)

[Tabla 2: Recursos humanos - Recursos 31](#_Toc159610037)

[Tabla 3: Recursos de hardware - Recursos 31](#_Toc159610038)

[Tabla 4: Recursos de software - Recursos 32](#_Toc159610039)

[Tabla 5: Tablas 43](#_Toc159610040)

[Tabla 6: Usuarios – Entidades 44](#_Toc159610041)

[Tabla 7: Rol-Usuario – Entidades 44](#_Toc159610042)

[Tabla 8: Roles – Entidades 44](#_Toc159610043)

[Tabla 9:Habilidad-Rol -Entidades 44](#_Toc159610044)

[Tabla 10: Etiquetas – Entidades 44](#_Toc159610045)

[Tabla 11: Programa-Etiqueta – Entidades 45](#_Toc159610046)

[Tabla 12: Programas – Entidades 45](#_Toc159610047)

[Tabla 13: Módulos – Entidades 45](#_Toc159610048)

[Tabla 14: Temas – Entidades 46](#_Toc159610049)

[Tabla 15: Preguntas – Entidades 46](#_Toc159610050)

[Tabla 16: Criterios de Evaluación – Entidades 46](#_Toc159610051)

[Tabla 17: Notas – Entidades 47](#_Toc159610052)

[Tabla 18: Desarrolladores – Entidades 47](#_Toc159610053)

[Tabla 19: Entrenadores – Entidades 47](#_Toc159610054)

[Tabla 20: Ejecuciones – Entidades 48](#_Toc159610055)

[Tabla 21: Asignaciones – Entidades 48](#_Toc159610056)

[Tabla 22: Enrolamientos - Entidades 49](#_Toc159610057)

[Tabla 23: Llaves primarias - Diccionario de datos 49](#_Toc159610058)

[Tabla 24: Cardinalidad - Diccionario de datos 50](#_Toc159610059)

[Tabla 25: Obtener lista de programas (manager) - Endpoints 51](#_Toc159610060)

[Tabla 26: Crear un programa (manager) - Endpoints 51](#_Toc159610061)

[Tabla 27: Crear una ejecución (manager) - Endpoints 52](#_Toc159610062)

[Tabla 28: Enrolar un trainee a una ejecución (manager) - Endpoints 53](#_Toc159610063)

[Tabla 29: Obtener lista de ejecuciones (manager) - Endpoints 53](#_Toc159610064)

[Tabla 30: Asignar un módulo a una ejecución (manager) - Endpoints 54](#_Toc159610065)

# INTRODUCCIÓN

En el mundo empresarial los cambios están en constante evolución, la necesidad de contar con herramientas eficaces y modernas para la gestión y evaluación de empleados se vuelve cada vez más imperante. La tecnología se ha convertido en un aliado estratégico en la optimización de procesos y en la toma de decisiones informadas.

Actualmente las empresas requieren una gestión de recursos humanos ágil, transparente y eficiente, que sustituya los métodos que tradicionalmente se han empleado para la evaluación del desempeño y competencias de los empleados, los cuales son llevados de forma manual o con sistemas obsoletos, lo que resulta en procesos lentos y subjetivos. Además, la falta de una plataforma centralizada y digitalizada para la recopilación y análisis de datos puede dificultar la identificación de fortalezas y áreas de mejora en el rendimiento del personal.

Con este panorama, surge la iniciativa de desarrollar y diseñar una Interfaz de Programación de Aplicaciones (API por sus siglas en inglés), en la cual se pueda implementar un sistema de evaluación de empleados, moderna y actualizada, que sea una solución innovadora diseñada para potenciar la efectividad de las estrategias de gestión de talento humano en las organizaciones.

El desarrollo de esta interfaz (API) se presenta aquí como una solución viable para abordar los desafíos descritos e impulsar la eficiencia y la equidad en la evaluación de los trabajadores, la API brindará a la empresa Sistemas Pliret una herramienta integral y de vanguardia para llevar a cabo evaluaciones de desempeño, seguimiento de plan de estudios y retroalimentación para los usuarios y personal que labora en ella.

Esta plataforma tecnológica permitirá a la empresa automatizar y personalizar los procesos de evaluación según el área técnica establecida por la empresa, promoviendo la estandarización de criterios, la transparencia y la trazabilidad. La API no solo facilitará la recopilación y análisis de datos, sino que también ofrecerá un marco flexible para adaptarse a las necesidades específicas de la organización siendo está quien establezca los módulos y temas a evaluar, impulsando así un enfoque objetivo en la toma de decisiones relacionadas con la gestión de empleados.

ANTECEDENTES DE LA INSTITUCIÓN

Fundada en el año # por el por entonces aun estudiante en ingeniería en sistemas el joven José Benjamín Zepeda Portillo. Siendo la empresa conocida inicialmente como Soluciones Aplicativas S.A de C.V. El ahora actual ingeniero José Benjamín Zepeda Portillo, que además de dueño y fundador dentro de la empresa su puesto dentro de la empresa oficialmente es el de Gerente de Tecnología.

Hoy en día Sistemas Pliret se dedica principalmente al outsourcing, es decir a proveer de personal altamente capacitado en conocimiento y experiencia tecnológica, a entidades bancarias y en menor medida al desarrollo de proyectos para empresas externas principalmente enfocados a la índole financiera.

# Ubicación geográfica

Corría a inicios del año 2020 cuando Sistemas Pliret pudo por fin cambiar de ubicación a la Torre Insigne, Colonia San Benito, San Salvador. Habían llegado a ser más 100 empleados y la empresa había dejado de estar ubicada en la casa de residencia de su fundador para cada ves ir convirtiéndose en una institución más profesional, sin embargo, ese mismo año llego la pandemia y sus clientes bajaron por lo que rápidamente debieron dejar ese nuevo sitio que tanto les había costado conseguir y hasta tuvieron que recurrir a dejar ir a la gran mayoría de sus empleados quedando únicamente con 15 empleados en su peor momento.

Desde entonces poco a poco se ha ido levantando, incluso estando hoy mejor que a inicios del 2020 teniendo actualmente a la fecha de escrito este informe más de 120 empleados y estando ubicados en Torre Futura, Colonia Escalón, San Salvador.

# Misión

Somos una empresa líder en desarrollo de software que ofrece soluciones innovadoras y personalizadas para impulsar el éxito de nuestros clientes a nivel nacional e internacional. Contamos con un equipo altamente calificado de ingenieros de software que utilizan las últimas tecnologías para crear software escalable, adaptable y eficiente que satisfaga las necesidades específicas de cada cliente. Nos comprometemos a brindar un servicio de excelencia con una comunicación constante y un enfoque centrado en la satisfacción del cliente. Nuestro objetivo es ayudar a nuestros clientes a alcanzar sus metas de negocio a través de la implementación de soluciones tecnológicas de vanguardia.

# Visión

Convertirnos en la mejor opción para el desarrollo de software y posicionamiento del personal de TI a nivel nacional e internacional. Aspiramos a lograrlo a través de un enfoque centrado en las necesidades y el presupuesto de nuestros clientes. Nos comprometemos a ofrecer soluciones innovadoras y personalizadas, utilizando las últimas tecnologías y un equipo altamente calificado. Buscamos establecer relaciones duraderas con nuestros clientes, brindándoles un servicio de excelencia y acompañándolos en su camino hacia el éxito.

# Valores

En nuestra empresa, los valores de cooperación, respeto, integridad y pasión por la excelencia son los pilares fundamentales que guían nuestra estrategia y operativa. Creemos en la fuerza del trabajo en equipo y la colaboración para alcanzar objetivos comunes. Promovemos un ambiente de respeto mutuo donde cada miembro del equipo se sienta valorado y escuchado. La integridad es innegociable en todas nuestras acciones, y nos comprometemos a actuar con honestidad y transparencia en todo momento. Finalmente, nuestra pasión por la excelencia nos impulsa a buscar la mejora continua en todo lo que hacemos, y a ofrecer a nuestros clientes un servicio y productos de la más alta calidad.

DESCRIPCIÓN DEL PROBLEMA

La empresa Sistemas Pliret, especializada en servicios de subcontratación de personal para empresas que requieren talento en el ámbito de desarrollo de software, cuenta con un equipo de entrenamiento y evaluación de nuevos empleados y demás personal. Esto se lleva a cabo a través de un proceso de seguimiento de una entrevista generalmente acompañada de una prueba técnica. Actualmente, este proceso genera desafíos relacionados con la falta de estandarización debido a la diversidad de requerimientos técnicos. Por lo tanto, surge la necesidad de desarrollar una herramienta que agilice y reduzca los tiempos de evaluación, al mismo tiempo que ofrezca una capacitación más efectiva.

Tomando en consideración la problemática descrita, se plantea el desarrollo de una Interfaz de Programación de Aplicaciones o API por sus siglas en inglés, que proporcione la integración con otras aplicaciones para la gestión de usuarios, entrenamiento de empleados, capacitaciones en línea y mejora de las habilidades y conocimiento tanto de los empleados actuales como del personal de nuevo ingreso.

Esta herramienta estará destinada hacia cuatro tipos de usuarios quienes podrán interactuar con la API de acuerdo con su nivel de autorización y según la gestión hacia la cual sea requerida, así esos niveles de estratificación serían:

* Manager: Quien se encarga de monitorizar y controlar y administrar la plataforma al resto de usuarios.
* Developer: Quien es responsable de la creación de los programas de entrenamiento para el personal de la empresa.
* Trainer: Usuario designado para llevar a cabo las evaluaciones del personal y nuevos empleados.
* Trainee: Todos los usuarios que ingresan al programa de capacitaciones.

La implementación de esta herramienta permitirá brindar una solución integral y eficiente en la gestión de los procesos de capacitación y evaluación del personal actual y de nuevo ingreso.

Con el desarrollo de la Interfaz de Programación de Aplicaciones, la empresa podrá centralizar y automatizar las diversas tareas relacionadas con la formación de sus empleados en las diferentes áreas que lo requiera.

Así mismo, la API permitirá al Manager hacer un seguimiento más preciso del progreso individual y colectivo de forma eficaz y oportuna. Por otro lado, la herramienta podrá impulsar el desarrollo profesional y técnico de los empleados que conforman Sistemas Pliret, permitiéndole ajustar los programas de estudio de acuerdo con las necesidades y objetivos de la organización.

Esta API impulsará a Sistemas Pliret para ser más competitivos y ofrecer al mercado laboral personal altamente capacitado y actualizado con las herramientas tecnológicas y de mayor tendencia en la sociedad actual.

JUSTIFICACIÓN

Para el ámbito empresarial actual, la evaluación del desempeño de sus cooperadores es fundamental, poder dar seguimiento a los parámetros de crecimiento técnico de sus empleados es un punto muy importante para el crecimiento y éxito de la misma organización. Muchas empresas se enfrentan a desafíos cada vez más exigentes y requieren de herramientas adecuadas para llevar a cabo estos procesos de manera efectiva y eficiente. Dichas herramientas les permiten entre otras cosas, recopilar, analizar y dar seguimiento a datos relevantes y relacionados con el desempeño de sus empleados, que ayudan en la toma de decisiones estratégicas y a la identificación de oportunidades de crecimiento y mejora del personal.

Como respuesta a la problemática planteada, surge la necesidad de contar con una solución tecnológica que simplifique y optimice los procesos de evaluación del personal. De aquí la importancia de poder contar con una herramienta adecuada que permita integrar los procesos requeridos para capacitar a todo empleado en la empresa Sistemas Pliret. El desarrollo de una Interfaz de Programación de Aplicaciones (API), permitirá crear un registro eficiente y seguro de los usuarios que hagan uso de la plataforma, así como de proporcionar los datos necesarios para el posterior análisis del desempeño de cada empleado de una forma sistemática y eficaz.

Conjuntamente con los beneficios que la empresa Sistemas Pliret obtendrá con la implementación de esta herramienta, el desarrollo de este proyecto ofrece una oportunidad de crecimiento profesional y técnico, al poder trabajar con nuevas tecnologías y diferentes metodologías modernas del desarrollo de software, entre estas la arquitectura REST, seguridad de datos, diseño y configuración de bases de datos.

Para Sistemas Pliret, la API permitirá un crecimiento importante en la gestión de las capacitaciones de su personal, así como poder contar con una herramienta con la que podrá estandarizar y automatizar los procesos de evaluación y capacitación de sus empleados, optimizar y agilizar la gestión de recursos humanos.

De igual manera la plataforma conllevará al uso más eficiente de los recursos, ahorrando tiempos en la ejecución de actividades y mejorando procesos en la administración de las evaluaciones.

La API brindará una mayor precisión en la recopilación y análisis de los datos, ofreciendo una comprensión más detallada del desempeño individual y colectivo del personal que forma parte de Sistemas Pliret.

La información obtenida a través de la plataforma facilitara la identificación de áreas de mejora, puntos fuertes y oportunidades para el desarrollo profesional del personal. Con estos datos en mano, la toma de decisiones estratégicas estará respaldada por información concreta y confiable, lo que fortalecerá el proceso de planificación y definición de objetivos corporativos.

Otro beneficio significativo de la API es la mejora en la comunicación y transparencia entre los distintos niveles jerárquicos de la empresa. La plataforma permitirá un acceso controlado y seguro a los datos de la evaluación, lo que fomentará una cultura organizacional basada en la confianza y retroalimentación constructiva. Los empleados podrán acceder a sus propias evaluaciones y tener una visión clara de sus fortalezas y áreas de mejora, lo que a su vez estimulará su motivación y compromiso con la empresa.

La plataforma facilitará la creación de informes y análisis de tendencias en el tiempo, esto significa poder hacer una revisión histórica del desempeño y seguimiento de la evolución de cada empleado. Este historial será invaluable para la identificación de patrones y evaluación del impacto de las estrategias de mejora implementadas.

Al ser una herramienta web, la API puede ser implementada como un servicio de backend en una aplicación monolítica o bien implementarse con un framework frontend, con el lenguaje que la empresa desee utilizar en el futuro esto puede ser por ejemplo con un frontend utilizando Next.JS, React.JS, Vue.JS, Java Server Faces, etc.

Esta API no requerirá de una dependencia técnica por parte del hosting ni lógica por parte de la implementación frontend, por lo que Sistemas Pliret podrá gestionar el despliegue para producción según sean sus requerimientos en el futuro.

ALCANCE

Para este proyecto se ha planteado la necesidad de desarrollar una API que permita la evaluación de personal, llevar un registro de usuarios, almacenamiento y análisis de datos relevantes sobre el progreso del desempeño para cada empleado de la empresa Sistemas Pliret.

Por tanto, el alcance del proyecto será diseñar y desarrollar una Interfaz de Programación de Aplicaciones, que permita al gerente de personal, el control, gestión y análisis de los datos almacenados, sobre el desempeño obtenido por los empleados de la empresa Sistemas Pliret, durante sus evaluaciones y capacitaciones.

Para el éxito de este proyecto, el alcance estará delimitado por las siguientes actividades:

1. Análisis de requerimientos: Consistirá en la revisión conjunta de los requerimientos solicitados por Sistemas Pliret, para asegurar que todas las necesidades y expectativas se han comprendido de forma clara y objetiva.
2. Selección de tecnologías y herramientas: Evaluar las diferentes tecnologías y herramientas disponibles para el desarrollo de la Interfaz de Programación de Aplicaciones, y hacer una selección apropiada basada en los criterios de eficiencia y seguridad que se requieran.
3. Diseño y desarrollo de la API: Tomando en consideración los requerimientos solicitados y luego de haber seleccionado las herramientas idóneas, se hará el diseño y el desarrollo, implementando los endpoints necesarios para la gestión de datos relacionados con el desempeño del personal.
4. Pruebas y depuración: Para garantizar el correcto funcionamiento de la API, se harán las pruebas necesarias que permitan evaluar en cada etapa el cumplimiento con los requerimientos solicitados.
5. Entrega de producto: Finalizada la etapa de desarrollo y pruebas, se hará entrega de la API para su evaluación y uso interno en la empresa Sistemas Pliret, de la misma forma se proporcionará la documentación con los procesos detallados para su implementación y uso.

OBJETIVOS

# Objetivo General

Desarrollar una Interfaz de Programación de Aplicaciones (API), para la evaluación y capacitación de empleados de la empresa Sistemas Pliret, S.A. de C.V., que le permita optimizar y simplificar el proceso de registro, análisis y seguimiento de datos para todo su personal.

# Objetivos Específicos

1. Diseñar una API Restful que permita al área de recursos humanos, crear programas de contenido y realizar evaluaciones a todo el personal antiguo y de nuevo ingreso.
2. Desarrollar módulos que permitan el registro de resultados de las evaluaciones en tiempo real para su control y posterior análisis.
3. Garantizar la seguridad de los datos de evaluación por medio de la encriptación, requerimiento de Json Web Tokens y autenticación por roles a los distintos endpoints de la API.

# DESCRIPCIÓN NARRATIVA DE LOS OBJETIVOS

Objetivo 1: Diseñar una API Restful que permita crear programas de contenido y realizar evaluaciones a todo el personal.

Consiste en diseñar y desarrollar una API Restful utilizando los lenguajes de programación Dotenv para la configuración de la API, un lenguaje a definir para el desarrollo de esta y SQL para la base de datos, que permitan a los responsables del rendimiento de los empleados antiguos y recién contratados, para crear programas de contenido y realizar evaluaciones. Con el desarrollo de la API, se mejorará el conocimiento técnico de cada empleado según los estándares de puntuación mínimos establecidos por la empresa en diferentes áreas técnicas.

Objetivo 2: Desarrollar módulos que permitan el registro de resultados de las evaluaciones en tiempo real para su control y posterior análisis.

Consiste en el desarrollo de los diferentes módulos con las funcionalidades necesarias que permitan a los evaluadores registrar en tiempo real resultados de las pruebas hechas a los empleados. Esto permitirá a Sistemas Pliret evaluar el impacto y progreso en la mejora continua de todo su personal, preparándose para una implementación prevista para el primer trimestre de 2024.

Objetivo 3: Garantizar la seguridad de los datos de evaluación.

Se enfoca en la implementación de medidas de seguridad desde el desarrollo de los diferentes módulos, que garanticen la encriptación de los datos de evaluación. La API incluirá medidas de seguridad que estarán orientadas a salvaguardar la confidencialidad y privacidad de la información, evitando el acceso no autorizado y asegurando la integridad de los datos.

CONTRIBUCION CIENTIFICA Y TECNICA

Con el desarrollo de este proyecto el alumno aportará una significativa contribución científica y tecnológica en el ámbito de la gestión de recursos humanos y optimización de procesos empresariales que se verán implementados en la plataforma a implementar. Esta iniciativa busca abordar desafíos cruciales en la gestión de talento humano y aportar soluciones innovadoras para mejorar la toma de decisiones informadas y estratégicas en la organización.

Entre las áreas de contribución en que se aportarán conocimientos adquiridos por el alumno, se pueden destacar:

* **Eficiencia y automatización de la evaluación de desempeño**: A través de sus conocimientos en desarrollo de software, diseño de sistemas, infraestructura de servidores, medidas de seguridad, desarrollo en la programación MVC y orientada a objetos, se desarrollará la API que estandariza el proceso de evaluación de desempeño de los empleados de Sistemas Pliret. Esto se traduce en una mayor eficiencia en los procesos de evaluación, al reducir el tiempo y los recursos necesarios para realizarlos.
* **Transparencia y equidad en la evaluación**: Utilizando sus conocimientos en desarrollo de software adquiridos durante la carrera, el estudiante desarrollará una API que estandariza la evaluación de desempeño de los empleados de Sistemas Pliret.

Esto se traduce en una mayor transparencia y equidad en la evaluación, al establecer criterios y métricas claros y consistentes para todos los empleados. Además, la API permite que los empleados puedan acceder a sus resultados de evaluación de manera transparente, lo que les permite identificar sus áreas de mejora.

* **Toma de decisiones basadas en datos**: El estudiante, utilizando sus conocimientos en desarrollo de software, ha desarrollará la API que recopila datos relevantes sobre el conocimiento técnico de los nuevos empleados de Sistemas Pliret. Estos datos pueden ser utilizados por los líderes y responsables de toma de decisiones para identificar las áreas de mejora del equipo y tomar decisiones informadas y estratégicas.
* **Desarrollo de la práctica y conocimiento**: Al desarrollar una API que estandariza el proceso de evaluación de desempeño, ha contribuirá al desarrollo de la práctica y conocimiento en el campo de la gestión de recursos humanos. El trabajo podrá ser utilizado por futuros trabajadores, investigadores y profesionales para mejorar los procesos de evaluación de desempeño en otras empresas.

DESCRIPCCIÓN NARRATIVA DE LOS OBJETIVOS

Objetivo 1: Diseñar una API Restful que permita crear programas de contenido y realizar evaluaciones a todo el personal.

Consiste en diseñar y desarrollar una API Restful utilizando los lenguajes de programación Dotenv para la configuración de la API, el lenguaje a definir para el desarrollo de esta y SQL para la base de datos, que permitan a los responsables del rendimiento de los empleados antiguos y recién contratados, para crear programas de contenido y realizar evaluaciones. Con el desarrollo de la API, se mejorará el conocimiento técnico de cada empleado según los estándares de puntuación mínimos establecidos por la empresa en diferentes áreas técnicas.

Objetivo 2: Desarrollar módulos que permitan el registro de resultados de las evaluaciones en tiempo real para su control y posterior análisis.

Consiste en el desarrollo de los diferentes módulos con las funcionalidades necesarias que permitan a los evaluadores registrar en tiempo real resultados de las pruebas hechas a los empleados. Esto permitirá a Sistemas Pliret evaluar el impacto y progreso en la mejora continua de todo su personal, preparándose para una implementación prevista para el primer trimestre de 2024.

Objetivo 3: Garantizar la seguridad de los datos de evaluación.

Se enfoca en la implementación de medidas de seguridad desde el desarrollo de los diferentes módulos, que garanticen la encriptación de los datos de evaluación. La API incluirá medidas de seguridad que estarán orientadas a salvaguardar la confidencialidad y privacidad de la información, evitando el acceso no autorizado y asegurando la integridad de los datos.

ACTORES INERVINIENTES EN EL PROYECTO

A continuación, se hace una descripción de los roles y su aporte, de cada uno de los actores principales (partes interesadas), que serán parte en el proyecto de desarrollo de la Interfaz de Programación de Aplicaciones (API), para la evaluación y capacitación del personal de la empresa Sistemas Pliret S.A. de C.V.

**Facilitador:**

Como facilitador del proyecto se reconoce al Ing. José Benjamín Zepeda Portillo, Gerente de Tecnología de la empresa Sistemas Pliret S.A. de C.V., quien ha solicitado y aprobado la ejecución del proyecto.

**Beneficiarios:**

Ente los beneficiarios se puede mencionar el mismo Gerente de Tecnología, el Gerente de Recursos Humanos, los Managers, como también los futuros y actuales empleados que conforman la empresa Sistemas Pliret S.A. de C.V., a quienes la implementación exitosa de la API permitirá mejorar la evaluación y desarrollo del personal en áreas técnicas. Los resultados esperados, como la optimización del proceso de evaluación, la generación de informes detallados y la seguridad de los datos, beneficiará directamente a la organización al impulsar la toma de decisiones informadas y la mejora continua.

**Desarrollador:**

El desarrollador para el proyecto es el estudiante Roberto Antonio Moreno Zepeda con carnet 2018-MZ-601, egresado de la carrera Ingeniería en Desarrollo de Software. Su responsabilidad es diseñar y desarrollar la API de evaluación de empleados siguiendo los requerimientos establecidos por Sistemas Pliret S.A. de C.V.

**Patrocinador**:

El proyecto se sustentará a través de los recursos del estudiante Roberto Antonio Moreno Zepeda, quien pondrá a disposición para el desarrollo de la API: tiempo, recursos de hardware y software (equipo de cómputo personal), y cualquier otro recurso tecnológico que fuere necesario. Además de los gastos de transporte cuando sea requerido el desplazamiento hacia la empresa para presentar los respectivos avances.

**Proyectos y labores realizadas:**

Gestión de las diferentes bases de datos en labores de tuning para aumentar el rendimiento de las bases de datos con los sistemas existentes, monitoreo de actividad y consumo de recursos en la base de datos, migraciones desde PostgreSQL Plus, mantenimiento preventivo de objetos de base de datos, optimización de kernel Linux para un funcionamiento adecuado del clúster de PostgreSQL y Jboss AS.

**Tecnologías y conceptos aplicados:** Balsamiq Mockups (desarrollo de prototipos), Toad Data Modeler, PGAdmin III, psql (linea de comandos), bash (Linux), análisis y diseño de sistemas.

HERRAMIENTAS PARA EL DESARROLLO DEL SISTEMA

Para que el software se pueda usar sin restricciones, tiene que seguir unas normas legales que las fijan la universidad o entidades del ámbito jurídico.

**Legales**

Para usar el software sin limitaciones, debe seguir unas condiciones legales que las determinan los organismos o entidades del ámbito jurídico.

* El reglamento interno de la Universidad Católica de El Salvador establece en su art. 19, Capítulo VI; que los derechos patrimoniales de autor de investigaciones realizadas por los estudiantes ordinariamente pertenecen a la Universidad, la cual, tendrá a su disposición para compartir los productos de la investigación en casos determinados.
* El art. 32 de la Ley de Fomento y Protección de la Propiedad Intelectual en El Salvador garantiza a los titulares de derecho de propiedad intelectual la posibilidad de ejercer acciones legales y solicitar medidas cautelares para proteger los derechos frente a cualquier tipo de infracción, estableciendo así, indemnización por daños y perjuicios causados por infracciones.

**Tiempo del desarrollo del sistema**

El periodo de realización del proyecto abarcara:

* Duración del proyecto: 5 meses y 4 días.
* Fecha de inicio: 28 de agosto de 2023.
* Fecha de finalización: 31 de enero de 2024

La cual constará de las siguientes fases:

Fase 1: Investigación y análisis de los requerimientos (28 de agosto de 2023 – 8 de septiembre de 2023): En esta fase se llevará a cabo una investigación de las necesidades de entrenamiento de la empresa Sistemas Pliret y se recopilarán los requerimientos específicos para la Interfaz de Programación de Aplicaciones.

Fase 2: Diseño de base de datos y lógica de negocios (11 de septiembre de 2023 – 22 de septiembre de 2023): Durante esta etapa, se diseñará la estructura de la base de datos que respaldará la API, así como la lógica de negocios necesaria para gestionar los usuarios, programas, módulos y temas.

Fase 3: Desarrollo de funcionalidades (25 de septiembre de 2023 – 29 de diciembre de 2023): Esta fase consiste en el desarrollo de las funcionalidades específicas para cada tipo de usuario y las interacciones necesarias entre ellos.

Fase 4: Periodo de pruebas unitarias (enero de 2024): Durante esta fase se realizarán pruebas unitarias para garantizar que la API funcione de manera óptima y cumpla con los requerimientos establecido.

Se hará la entrega de la API funcional y la documentación necesaria a través del compromiso del desarrollador a proporcionar la API a Sistemas Pliret en el plazo acordado con fecha límite del 31 de enero de 2024, permitiendo que la empresa pueda empezar a utilizarla.

Cuyos avances serán entregados a la empresa el último viernes de cada mes en el periodo de desarrollo y depuración el cual comprende del 25 de septiembre de 2023 al 31 de enero de 2024, permitiendo que la empresa Sistemas Pliret pueda empezar a utilizarla y pueda evaluar el impacto en la mejora continua de sus empleados para febrero de 2024.

Actividades y elementos no serán parte del alcance para este proyecto:

1. Despliegue para entorno de producción: el proyecto se limitará a la fase de diseño y desarrollo de la API en modo desarrollo. El despliegue para el entorno de producción no será considerado en este alcance.
2. Soporte del sistema: el desarrollo del proyecto no abarcará el soporte continuo de la API una vez se haga la entrega satisfactoria del mismo. Sistemas Pliret evaluará los resultados y beneficios obtenidos antes de considerar el soporte a largo plazo.

Dichas actividades se consolidan en el siguiente cronograma, el cual fue aprobado formalmente por las partes involucradas.

******CRONOGRAMA**

Ilustración 1: Cronograma de actividades

# Cronograma en formato texto

# Fase 1:

Investigación y toma de requerimientos (28 de agosto de 2023 - 8 de septiembre de 2023).

* Toma de requerimientos (28 de agosto al 4 de septiembre).
* Definición de tecnologías a utilizar (5 de septiembre al 8 de septiembre).

# Fase 2:

Diseño de base de datos y lógica de negocios (11 de septiembre de 2023 - 22 de septiembre de 2023).

* Diseño del esquema de la base de datos (11 de septiembre al 15 de septiembre).
* Definición de relaciones y tablas (18 de septiembre al 19 de septiembre).
* Diseño de la lógica de evaluación y generación de informes (20 de septiembre al 22 de septiembre).

# Fase 3:

Desarrollo de funcionalidades (25 de septiembre de 2023 - 29 de diciembre de 2023).

* Desarrollo del módulo de mánager (25 de septiembre al 13 de octubre).
* Desarrollo del módulo de developer (16 de octubre al 3 de noviembre).
* Desarrollo del módulo de trainer (6 de noviembre al 24 de noviembre).
* Desarrollo del módulo de trainee (27 de noviembre al 15 de diciembre).
* Integración de módulos (18 de diciembre al 29 de diciembre).

# Fase 4:

Periodo de pruebas unitarias y entrega (1 de enero de 2024 – 31 de enero de 2024)

* Unit Testing (1 de enero al 31 de enero).
* Depuración (1 de enero al 31 de enero).
* Creación de manual de usuario (1 de enero al 31 de enero).
* Entrega final (1 de enero al 31 de enero).

# Cronograma por presupuesto



*Ilustración 2: Cronograma de por presupuesto agosto a septiembre*



Ilustración 3: Cronograma de por presupuesto octubre a noviembre



Ilustración 4: Cronograma de por presupuesto diciembre a enero

# Tabla de herramientas y costos

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Concepto** | **Meses a usar** | **Costo por mes** | **Observación** | **Patrocinador** | **Costo Total** |
| Lenguajes de programación: A definir MySQL. | 5 | $0.00 | Sea cual sea el lenguaje a definir, ningún lenguaje mencionado es de pago porque son de código abierto | - | $0.00 |
| Computador tipo laptop | - | $700 | Costo estimado | - | $700 |
| Instagantt | 5 | $6.00 | Comercial | Estudiante | $30.00 |
| Licencia Windows 10 Home | - | $49.59 | Comercial | - | $49.59 |
| Salario Programador Junior | 5 | $500 | Suponiéndose que hubiese pago para el programador | - | $2500 |
| Internet | 5 | $33 | Contratado por el estudiante | Estudiante | $165 |
| Energía eléctrica | 5 | $65 | Contratado por el estudiante | Estudiante | $325 |
| Visual Studio | 5 | $4 | Open Source | Estudiante | $20 |
| StarUML | - | $55.30 | Comercial | Estudiante | $55.30 |
| Postman | 5 | $0.00 | Open Source | - | $0.00 |
| Git | 5 | $0.00 | Open Source | - | $0.00 |
| Github/Gitlab | 5 | $0.00 | Open Source | - | $0.00 |
| HeidySQL | 5 | $0.00 | Open Source | - | $0.00 |
| Google OAuth 2.0 | 5 | $0.00 | Open Source | - | $0.00 |
| Libre Office | 5 | $0.00 | Open Source | - | $0.00 |
| Navegador web | 5 | $0.00 | Open Source | - | $0.00 |

Tabla 1: Definición y análisis de herramientas con sus costos

# Recursos

Recursos humanos.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Rol** | **Cantidad** | **Tiempo** | **Fecha Inicio** | **Fecha Fin** |
| Analista - Programador | 1 | 5 meses y 4 días | 28/08/2023 | 31/01/2024 |
| Asesor | 1 | 5 meses y 4 días | 28/08/2023 | 31/01/2024 |

Tabla 2: Recursos humanos - Recursos

## Recursos de hardware.

|  |  |
| --- | --- |
| Computadora de escritorio | |
| Procesador | AMD Ryzen 7 5800X3D |
| RAM | 32.0GB |
| Almacenamiento | SSD 1024GB + HDD 1024GB |
| Monitor | Resolución: 1920 x 1080 |
| Teclado | Mecánico (genérico) |
| Mouse | Mecánico (genérico) |
| Cámara | HD Web Camera (genérica) |
| Micrófono | Razer BlackShark V2 |

Tabla 3: Recursos de hardware - Recursos

## Recursos de software para el desarrollo del sistema

|  |  |
| --- | --- |
| **Recurso** | **Software** |
| Navegador Web | Opera GX, Microsoft Edge, Google Chrome |
| Editor de texto | Visual Studio |
| Cliente HTTP | Postman |
| Software de ofimática | LibreOffice |
| Repositorio de archivos | Git |

Tabla 4: Recursos de software - Recursos

API RESTful

Para llevar a cabo el desarrollo de la API fue necesario definir primeramente que iba a tratar el sistema a desarrollar, para ello la primera fase del desarrollo se enfocó en llevar a cabo reuniones con los interesados para así a partir de las historias de usuario obtener los requerimientos funcionales y no funcionales. Seguidamente una vez definido eso se debía llevar a cabo la decisión de las tecnologías a implementar acorde a las necesidades de la empresa. A continuación, se detallan las primeras etapas del desarrollo.

Primeramente, de las entrevistas realizadas se acordó que los requerimientos acordes a su funcionalidad iban a ser los siguientes:

# Requerimientos funcionales:

* Los usuarios de tipo manager pueden crear, editar y eliminar usuarios.
* Los usuarios pueden iniciar sesión y acceder a la API.
* Los managers pueden crear programas.
* Los managers pueden iniciar ejecuciones de programas.
* Los managers pueden asignar developers a programas.
* Los managers pueden crear asignar trainers a ejecuciones.
* Los managers pueden crear asignar trainees a ejecuciones.
* Los developers pueden crear, editar y eliminar temas de módulos de programas.
* Los developers pueden crear, editar y eliminar temas de módulos.
* Los developers pueden crear, editar y eliminar preguntas de los temas.
* Los developers pueden crear, editar y eliminar los criterios a evaluar de los módulos.
* Los trainers pueden tomaran las notas de los trainees.
* Los trainers pueden tomaran las respuestas de los trainees en los cuestionarios.
* Los trainers pueden evaluar las soluciones de los trainees en sus actividades.
* Los trainers pueden exportar a Excel el reporte de sus trainees.
* Los trainers pueden ver las ejecuciones en los que participan.
* El sistema calcula el promedio de las calificaciones de los trainees a disposición de los trainers.
* Los trainees pueden enviar soluciones a las tareas.
* Los trainees pueden ver las ejecuciones en los que participan.

# Requisitos no funcionales

* El sistema seguirá los estándares de api resource.
* El sistema tendrá métodos de autorización con login con correo y contraseña, y Google.
* El sistema tendrá autentificación para la información de cada tipo de usuario.
* El sistema cifrara las contraseñas en la base de datos.
* Tiempo de respuesta rápido.
* Podrán existir más de 1 trainee por ejecución.

# DESCRIPCCIÓN DE LA API

La API funciona de la siguiente manera. Existen cuatro tipos de usuarios: *Manager*, *developer*, *trainer* y *trainee* que son los que pueden acceder para autentificarse en el sistema y dependiendo de los permisos de cada uno de estos serán las acciones que podrán llevar a cabo. Cada una de estas acciones en el contexto de la API se traduce a peticiones por medio de un cliente HTTP que accede a las URLs de la API que son escuchadas por el servidor. En el encabezado de cada petición se anexa el token de acceso que contiene los permisos de acceso del usuario, que a su vez es obtenido una vez realizado el inicio de sesión, y si los permisos necesarios para realizar la acción de la petición que contiene el token del usuario que la envía es permitido entonces se procede con la petición del usuario o de lo contrario la petición es denegada. Cada respuesta independientemente del resultado de la petición viene acompañada de un código HTTP para ser entendido por una futura implementación de un frontend por parte de la empresa.

Existen los programas que son agrupaciones de módulos que a su vez son agrupaciones de temas. Los programas tienen etiquetas para un futuro filtro en la plataforma interna de Pliret, por ejemplo, un programa puede ser de backend, bancos, java sprint boot, etc. Temas que requiera le empresa evaluar a su personal antiguo y de nuevo ingreso para filtrar a los mejores candidatos para las plazas de trabajo que manejan y poder enviarlos. Los módulos son agrupaciones de contenido orientado a ser el contenido de una semana y los temas son los distintos temas que contendrán todo un programa pero que van agrupados en los módulos.

Para cada módulo de un programa se realizará una evaluación al código de la semana a revisar y una entrevista sobre el tema. Cada uno de estos puntos a evaluar se registra y por cada módulo se obtiene un promedio y al final del programa se obtiene el promedio general.

Todo este apartado se refiere únicamente a la planeación del contenido a evaluar, la puesta en práctica del programa de contenido se llama ejecuciones que contienen enrolamientos que hace referencia al módulo puesto en marcha. Se puede entender los programas como las clases y las ejecuciones como la instancia de esas clases para un mejor entendimiento.

Los usuarios *manager* son los encargados de monitorear los programas y ejecuciones además de la creación de los otros tipos de usuarios en el programa; también son los responsables de crear las ejecuciones de los programas y los enrolamientos de los trainees en las ejecuciones.

Los usuarios *developer* son los encargados de generar el contenido de un programa, los manager asignan a usuarios de tipo *developer* para ser los encargados de agregar el contenido que se seguirá durante toda la ejecución de un programa como los temas, criterios de evaluación y las preguntas de cada módulo.

Los *trainers* son los que supervisan a los *trainees* que son asignados por el manager en su proceso a través de la ejecución, son los que toman el código de estos y los puntúan según los criterios de evaluación establecidos por el *developer* del programa además de ser los responsables de realizar la entrevistas al *trainee* durante la ejecución.

Los usuarios *trainees*, es decir los usuarios en periodo de evaluación, son los que se someten a la ejecución de un programa y cada módulo van obteniendo una puntuación por entrevista, que es el promedio de las preguntas de las entrevistas, y una puntuación del código que se les evaluó, que a su vez es un promedio de los criterios de evaluación asignados por el developer a su modulo, estos promedios a su vez son promediados para obtener la nota final al finalizar la ejecución.

Lo que sintetizado en una estructura orientada para una API es en la planeación de los siguientes *endpoints*, después de todo el desarrollo de este proyecto abarca únicamente la programación de la API por lo que no llevara ningún tipo de *frontend*, sino únicamente el *backend* así que las funcionalidades finales se representaran a través de los *endpoints* del programa. Las funcionalidades que tendrán los *endpoints* son los siguientes agrupados por los cuatro tipos de usuario que harán peticiones a la API:

# Manager:

* Crear una nueva ejecución.
* Listar todas las ejecuciones.
* Listar todos los programas.
* Crear un nuevo programa.
* Crear un empleado.
* Crear un *trainee*.
* Listar usuarios.
* Ver detalle de usuarios.
* Enrolar trainees a ejecuciones.
* Asignar modulo a una ejecución.
* Editar usuarios.
* Eliminar usuarios.
* Asignar *developer* para que cree contenido para el programa.
* Asignar *trainer* a una ejecución.

# Developer:

* Listar todos los programas.
* Ver detalle de programas.
* Crear un módulo.
* Editar módulos.
* Eliminar módulos.
* Crear un tema.
* Editar temas.
* Eliminar temas.
* Crear una pregunta.
* Editar preguntas.
* Eliminar preguntas.
* Crear un criterio de evaluación.
* Editar criterios de evaluación.
* Eliminar criterios de evaluación.

# Trainer:

* Listar todas las ejecuciones.
* Ver detalle de ejecuciones.
* Iniciar entrevista.
* Guardar respuestas.
* Terminar entrevistas.
* Iniciar evaluación de código.
* Guardar notas de la evaluación de código.
* Terminar evaluación de código.
* Crear reporte de trainee.

# Trainee:

* Enviar código a evaluar.
* Listar todas las ejecuciones.
* Ver detalle de ejecuciones.

# Generales:

* Login básico
* Google login

Hay que tener en cuenta que, aunque puedan compartir algunos *endpoints* algunos tipos de usuarios la información a mostrar variara según el tipo de usuario que reciba en la petición de la *API*, por lo que, aunque pueda ser el mismo *endpoint*, dependiendo el usuario la respuesta variara en función a sus permisos siendo la única excepción los endpoints para iniciar sesión.

# Tecnologías utilizadas:

Se decidió que el lenguaje debía ser de código abierto, no estar atado a una implementación especifica de alguna empresa y debería tener una gran comunidad entorno a ella para una documentación extensa y de fácil acceso. Se determinó que la mejor selección para el proyecto sería el lenguaje PHP porque además de estar orientado exclusivamente al desarrollo web ofrece varias maneras de implementación en servicios de hosting y se eligió el framework Laravel para la maquetación y estructura de la API que se utiliza junto al lenguaje de programación, así mismo se estableció como gestor de la base de datos el motor de MySQL. Entre las ventajas que ofrecen las tecnologías seleccionadas se detallan los siguientes:

* Amplia disponibilidad a la hora de ser montada en un servidor: Esto se refiere a que, una vez finalizado el desarrollo del proyecto, la API no estará restringida a ser desplegada en un servicio en específico de la compañía propietaria de la tecnología como en el caso de Azure (C#, .NET), Oracle Cloud Infrastructure (Java SE, Java EE, Spring, Hibernate, etc.). Por otro lado, PHP no requiere de una plataforma específica para montarse en internet y puede basar todas sus versiones de la API en código abierto lo que no ata al proyecto a costos extra, lo que reduce costos tanto en el desarrollo como si se lleva a producción, pudiendo desplegarla en la plataforma que más le convenga a la empresa, además de no requerir licencias para los IDEs de programación o del lenguaje mismo o framework.
* Comunidad activa y vigente: El soporte tanto del lenguaje PHP como de su framework Laravel que se utilizó en el desarrollo cuentan con una documentación frecuentemente actualizada por una comunidad vigente, lista para ser utilizada en todo momento, además de que siguen teniendo actualizaciones tanto de funcionalidades como de seguridad constantemente.

# Ingles

Al ser el framework Laravel la herramienta principal de desarrollo se trabajaron los archivos y toda la configuración interna de la API en el idioma inglés, esto debido a que el framework automatiza varias configuraciones porque internamente este interconecta los archivos de *migrations*, *models*, *resources* y la base de datos entre si guiándose únicamente por el nombre. Por ejemplo:

Un *model* de nombre “User” el *framework* automáticamente lo relacionara con la tabla de la base de datos llamada “users”, es decir el diminutivo en plural, y ese a su vez con la *migration* “table\_users” que luego la conecta con el *resource* “UserResource” para el despliegue de datos; otro ejemplo es: Enlazar el *model* con los atributos que este está persistiendo de la base de datos al lenguaje de programación. Todo esto únicamente lo hará si se usan palabras en inglés para el desarrollo para que sean detectadas por Laravel, si se usa el idioma español todas estas configuraciones deben hacerse manualmente por cada modelo a utilizar.

# Laravel

Laravel es un *framework* del lenguaje de programación PHP por lo que, aunque el *framework* utilice varios tipos de archivos para hacer un proyecto funcional (models, migrations, resources, etc.) en la práctica todos son archivos PHP (.php) por lo que el nombre del tipo de archivo hace referencia a la función que este tiene y no a la extensión de archivo que maneja. A continuación, se listarán los tipos de archivos del framework más relevantes y su función del que hace uso el proyecto:

* Models: Son los archivos que representan una entidad en la base de datos. Por cada tabla de la base de datos existirá un archivo model que lo representará en tiempo de ejecución y hará la persistencia en el proyecto. Es decir que si se accede al objeto de tipo model de usuarios este contiene todos los registros que existen de usuarios en la tabla usuarios de la base de datos.
* Controler: Es el archivo que procesa la petición que se le hace a la API y en base a los parámetros requeridos por el request, el header de la petición y el método HTTP empleado enviara la respuesta, es donde se ejecuta la lógica de programación como tal pero no literalmente donde esta plasmada en el código.
* Services: Estos tipos de archivos contienen toda la lógica de programación que luego es instanciada en los controladores para llevarlo a cabo, por ejemplo, en este proyecto es en uno de estos tipos de archivos donde se promedia la nota por modulo y por ejecución de los trainees, pero es hasta su instanciación en un controlador donde es llevado a cabo este proceso.
* Migration: Laravel nos permite crear el script de la base de datos en el mismo lenguaje PHP por medio de clases llamadas *migrations*. Cada una de estas clases hace referencia a una entidad de la base de datos que luego es creada en la base de datos. Esto es una ventaja porque permite manejar la estructura de la base de datos desde el mismo proyecto con estos tipos de archivos y no con un archivo externo con extensión SQL que contenga el script de la base de datos.
* Resource: Se encarga de maquetar la respuesta de la petición, el orden de los datos y si es requerido de llamar a otros *resources* en su respuesta. Es el que contiene la estructura de la respuesta. Por ejemplo, si se quiere ver el detalle de un usuario el archivo “UserResource” tendrá en su maquetación de respuesta todos los detalles del usuario a mostrar (nombre, apellido, tipo de usuario).
* Events: Son las clases encargadas de llevar procesos en segundo plano, como su nombre lo indica son activadas luego de ciertos eventos según se defina durante el desarrollo. Procesos en segundo plano se refiere a que se utiliza otro hilo del procesador para no sobrecargar el principal.
* Listeners: Son los que están a la escucha que se cumpla las condiciones para detonar la acción del evento, cada *listener* esta vinculado a un evento.
* Requests: Son las clases encargadas de definir los parámetros de cada petición donde se instancien estos archivos. Por ejemplo, el endpoint para crear un empleado requiere un input para el nombre y otro para el tipo de empleado, este tipo de lógica es lo que manejan estos archivos.
* Tests: Son los archivos encargados de crear pruebas que examinen el correcto funcionamiento de los *endpoints* y de la lógica de las clases *services*. Se usan para depurar código y encontrar errores.

# MySQL

El gestor de base de datos relaciones utilizado para este proyecto fue MySQL por ser ampliamente utilizado por su facilidad de uso, flexibilidad y escalabilidad. Desarrollado por Oracle, se basa en el lenguaje de consulta estructurado (SQL) para la creación, gestión y consulta de datos.

Sus principales características son:

Código abierto: Permite la libre distribución, modificación y redistribución del software, fomentando una comunidad activa de desarrolladores.

Alto rendimiento: Ofrece un procesamiento rápido de consultas y transacciones, ideal para aplicaciones web y de alto volumen de datos.

Escalabilidad: Soporta desde pequeñas bases de datos personales hasta grandes sistemas empresariales, adaptándose a las necesidades cambiantes.

Flexibilidad: Funciona en diversos sistemas operativos (Windows, Linux, macOS) y se integra con una amplia gama de lenguajes de programación (PHP, Java, Python).

Seguridad: Brinda mecanismos robustos para la protección de datos, incluyendo autenticación de usuarios, control de acceso y encriptación.

# Control de versiones

Esta fue una decisión únicamente llevada a cabo por el desarrollador el de utilizar un sistema de control de versiones para asegurar una disponibilidad total de los archivos como forma de seguro para evitar perdida de trabajo si se hubiese presentado algún inconveniente.

Específicamente se eligió el gestor llamado github que también sirve como plataforma online que permite alojar, gestionar y colaborar en proyectos de software utilizando el sistema de control de versiones Git. Para este proyecto se utilizó la modalidad gratuita y el único integrante fue el desarrollador del proyecto, se integra con diversas herramientas de desarrollo y cuenta con una comunidad activa de desarrolladores que comparten código, conocimiento y mejores prácticas, aunque en esta ocasión todo el código fue desarrollado en su totalidad para esta ocasión durante el transcurso del proyecto.

DISEÑO DEL SISTEMA

Antes de empezar con la programación teniendo los endpoints definidos para ser desarrollados el primer paso fue diseñar con el programa StarUML el diagrama de la base de datos para luego plasmar ese esquema en las migraciones de Laravel y finalmente crearlas en MySQL.

# Diagrama Entidad Relación

Ilustración 5: Diagrama ERD

DICCIONARIO DE DATOS

A continuación, se presentan las entidades que se crearon a manera de tablas en la base de datos, se muestran de manera detallada junto con los atributos que estas presentan y que se utilizó para crear la API.

# Tablas

|  |  |
| --- | --- |
| **Número** | **Nombre** |
| 1 | users |
| 2 | role\_user |
| 3 | roles |
| 4 | ability\_role |
| 5 | abilities |
| 6 | developers |
| 7 | trainers |
| 8 | tags |
| 9 | program\_tags |
| 10 | programs |
| 11 | modules |
| 12 | topics |
| 13 | questions |
| 14 | evaluation\_criterias |
| 15 | gradings |
| 16 | assignments |
| 17 | enrollments |
| 18 | executions |

Tabla 5: Tablas

# Entidades

## Usuario

|  |  |  |  |
| --- | --- | --- | --- |
| **Atributo** | **Tipo** | **Nulo** | **PK** |
| id | int | 0 | 1 |
| name | varchar | 1 | 0 |
| email | varchar | 0 | 0 |
| password | varchar | 0 | 0 |
| created\_at | timestamp | 0 | 0 |
| updated\_at | timestamp | 1 | 0 |
| deleted\_at | timestamp | 1 | 0 |

Tabla 6: Usuarios – Entidades

## Rol-Usuario

|  |  |  |  |
| --- | --- | --- | --- |
| **Atributo** | **Tipo** | **Nulo** | **PK** |
| id | int | 0 | 1 |
| role\_id | int | 0 | 0 |
| user\_id | int | 0 | 0 |

Tabla 7: Rol-Usuario – Entidades

## Roles

|  |  |  |  |
| --- | --- | --- | --- |
| **Atributo** | **Tipo** | **Nulo** | **PK** |
| id | int | 0 | 1 |
| name | varchar | 0 | 0 |

Tabla 8: Roles – Entidades

## Habilidad-Rol

|  |  |  |  |
| --- | --- | --- | --- |
| **Atributo** | **Tipo** | **Nulo** | **PK** |
| id | int | 0 | 1 |
| ability\_id | int | 0 | 0 |
| role\_id | int | 0 | 0 |

Tabla 9:Habilidad-Rol -Entidades

## Etiquetas

|  |  |  |  |
| --- | --- | --- | --- |
| **Atributo** | **Tipo** | **Nulo** | **PK** |
| id | int | 0 | 1 |
| name | varchar | 0 | 0 |
| created\_at | timestamp | 1 | 0 |
| updated\_at | timestamp | 1 | 0 |

Tabla 10: Etiquetas – Entidades

## Programa-Etiqueta

|  |  |  |  |
| --- | --- | --- | --- |
| **Atributo** | **Tipo** | **Nulo** | **PK** |
| id | int | 0 | 1 |
| name | varchar | 0 | 0 |
| program\_id | int | 0 | 0 |
| tag\_id | int | 0 | 0 |

Tabla 11: Programa-Etiqueta – Entidades

## Programas

|  |  |  |  |
| --- | --- | --- | --- |
| **Atributo** | **Tipo** | **Nulo** | **PK** |
| id | int | 0 | 1 |
| title | varchar | 0 | 0 |
| description | varchar | 0 | 0 |
| content | text | 0 | 0 |
| created\_at | timestamp | 0 | 0 |
| updated\_at | timestamp | 1 | 0 |
| deleted\_at | timestamp | 1 | 0 |

Tabla 12: Programas – Entidades

## Módulos

|  |  |  |  |
| --- | --- | --- | --- |
| **Atributo** | **Tipo** | **Nulo** | **PK** |
| id | int | 0 | 1 |
| title | varchar | 0 | 0 |
| descripcion | varchar | 0 | 0 |
| content | text | 0 | 0 |
| homework\_content | timestamp | 0 | 0 |
| program\_id | int | 0 | 0 |
| created\_at | timestamp | 0 | 0 |
| updated\_at | timestamp | 1 | 0 |
| deleted\_at | timestamp | 1 | 0 |

Tabla 13: Módulos – Entidades

## Temas

|  |  |  |  |
| --- | --- | --- | --- |
| **Atributo** | **Tipo** | **Nulo** | **PK** |
| id | int | 0 | 1 |
| index | int | 0 | 0 |
| title | varchar | 0 | 0 |
| descripcion | varchar | 0 | 0 |
| content | text | 0 | 0 |
| module\_id | int | 0 | 0 |
| created\_at | timestamp | 0 | 0 |
| updated\_at | timestamp | 1 | 0 |
| deleted\_at | timestamp | 1 | 0 |

Tabla 14: Temas – Entidades

## Preguntas

|  |  |  |  |
| --- | --- | --- | --- |
| **Atributo** | **Tipo** | **Nulo** | **PK** |
| id | int | 0 | 1 |
| question | varchar | 0 | 0 |
| grade\_definition | varchar | 0 | 0 |
| topic\_id | int | 0 | 0 |
| created\_at | timestamp | 0 | 0 |
| updated\_at | timestamp | 1 | 0 |
| deleted\_at | timestamp | 1 | 0 |

Tabla 15: Preguntas – Entidades

## Criterios de Evaluación

|  |  |  |  |
| --- | --- | --- | --- |
| **Atributo** | **Tipo** | **Nulo** | **PK** |
| id | int | 0 | 1 |
| objetive | varchar | 0 | 0 |
| grade\_definition | varchar | 0 | 0 |
| module\_id | int | 0 | 0 |
| created\_at | timestamp | 0 | 0 |
| updated\_at | timestamp | 1 | 0 |
| deleted\_at | timestamp | 1 | 0 |

Tabla 16: Criterios de Evaluación – Entidades

## Notas

|  |  |  |  |
| --- | --- | --- | --- |
| **Atributo** | **Tipo** | **Nulo** | **PK** |
| id | int | 0 | 1 |
| assignment\_id | int | 0 | 0 |
| gradable\_id | int | 0 | 0 |
| gradable\_type | varchar | 0 | 0 |
| comments | varchar | 0 | 0 |
| grade | float | 0 | 0 |
| created\_at | timestamp | 0 | 0 |
| updated\_at | timestamp | 1 | 0 |
| deleted\_at | timestamp | 1 | 0 |

Tabla 17: Notas – Entidades

## Desarrolladores

|  |  |  |  |
| --- | --- | --- | --- |
| **Atributo** | **Tipo** | **Nulo** | **PK** |
| id | int | 0 | 1 |
| program\_id | int | 0 | 0 |
| user\_id | int | 0 | 0 |
| active | boolean | 0 | 0 |
| created\_at | timestamp | 0 | 0 |
| updated\_at | timestamp | 1 | 0 |
| deleted\_at | timestamp | 1 | 0 |

Tabla 18: Desarrolladores – Entidades

## Entrenadores

|  |  |  |  |
| --- | --- | --- | --- |
| **Atributo** | **Tipo** | **Nulo** | **PK** |
| id | int | 0 | 1 |
| execution\_id | varchar | 0 | 0 |
| user\_id | varchar | 0 | 0 |
| active | varchar | 0 | 0 |
| created\_at | timestamp | 0 | 0 |
| updated\_at | timestamp | 1 | 0 |
| deleted\_at | timestamp | 1 | 0 |

Tabla 19: Entrenadores – Entidades

## Ejecuciones

|  |  |  |  |
| --- | --- | --- | --- |
| **Atributo** | **Tipo** | **Nulo** | **PK** |
| id | int | 0 | 1 |
| program\_id | int | 0 | 0 |
| start\_date | date | 0 | 0 |
| end\_date | date | 0 | 0 |
| program\_execution\_content | json | 1 | 0 |
| finished | timestamp | 1 | 0 |
| created\_at | timestamp | 0 | 0 |
| updated\_at | timestamp | 1 | 0 |
| deleted\_at | timestamp | 1 | 0 |

Tabla 20: Ejecuciones – Entidades

## Asignaciones

|  |  |  |  |
| --- | --- | --- | --- |
| **Atributo** | **Tipo** | **Nulo** | **PK** |
| id | int | 0 | 1 |
| execution\_id | int | 0 | 0 |
| module\_id | int | 0 | 0 |
| user\_id | int | 0 | 0 |
| interview\_grade | float | 0 | 0 |
| interview\_start\_at | timestamp | 1 | 0 |
| interview\_finish\_at | timestamp | 1 | 0 |
| interview\_snapshot | json | 1 | 0 |
| homework\_grade | float | 0 | 0 |
| homework\_start\_at | timestamp | 1 | 0 |
| homework\_finish\_at | timestamp | 1 | 0 |
| homework\_solution | text | 1 | 0 |
| homework\_snapshot | json | 1 | 0 |
| created\_at | timestamp | 0 | 0 |
| updated\_at | timestamp | 1 | 0 |
| deleted\_at | timestamp | 1 | 0 |

Tabla 21: Asignaciones – Entidades

## Enrolamientos

|  |  |  |  |
| --- | --- | --- | --- |
| **Atributo** | **Tipo** | **Nulo** | **PK** |
| id | int | 0 | 1 |
| execution\_id | int | 0 | 0 |
| user\_id | int | 0 | 0 |
| score | float | 0 | 0 |
| active | boolean | 0 | 0 |
| created\_at | timestamp | 1 | 0 |
| updated\_at | timestamp | 0 | 0 |

Tabla 22: Enrolamientos - Entidades

# Llaves primarias

|  |  |
| --- | --- |
| **Tabla** | **Nombre** |
| users | id |
| role\_user | id |
| roles | id |
| ability\_role | id |
| abilities | id |
| tags | id |
| program\_tag | id |
| developers | id |
| trainers | id |
| programs | id |
| modules | id |
| topics | id |
| questions | id |
| evaluation\_criterias | id |
| gradings | id |
| executions | id |
| assignments | id |
| enrollments | id |

Tabla 23: Llaves primarias - Diccionario de datos

# Cardinalidad

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tabla primaria** | **Llave primaria** | **Tabla foránea** | **Llave foránea** | **Unión** |
| users | id | role\_user | user\_id | 1:M |
| roles | id | role\_user | role\_id | 1:M |
| roles | id | ability\_role | role\_id | 1:M |
| abilities | id | ability\_role | ability\_id | 1:M |
| users | id | developers | user\_id | 1:M |
| programs | id | developers | program\_id | 1:M |
| tags | id | program\_tag | tag\_id | 1:M |
| programs | id | program\_tag | program\_id | 1:M |
| programs | id | modules | program\_id | 1:M |
| modules | id | topics | module\_id | 1:M |
| modules | id | evaluation\_criterias | module\_id | 1:M |
| topics | id | questions | topic\_id | 1:M |
| evaluation\_criterias | id | gradings | gradable\_id | 1:M |
| questions | id | gradings | gradable\_id | 1:M |
| programs | id | executions | program\_id | 1:M |
| executions | id | trainers | execution\_id | 1:M |
| users | id | trainers | user\_id | 1:M |
| executions | id | assignments | execution\_id | 1:M |
| modules | id | assignments | module\_id | 1:M |
| assignments | id | enrollments | assignment\_id | 1:M |
| assignments | id | gradings | assignment\_id | 1:M |

Tabla 24: Cardinalidad - Diccionario de datos

ENDPOINTS

Primeramente, entendiendo que son los endpoints (véase anexo 1: Glosario) y que para este desarrollo se ha trabajado con datos de prueba (también llamados datos *fake*) y no con el contenido real de los programas que se implementaran en un futuro dentro de la empresa interesada, se definieron los métodos que usarían acorde a su acción y las URLs que usarían. Todos los endpoints desarrollados para esta API siguen los patrones *api resource* para una estandarización de la acción y el recurso al que se intenta acceder, por lo tanto, el tipo de acción solicitado en última instancia viene definido por el método HTTP que se emplee para realizar la petición. A continuación, la lista detallada de los endpoints programados para este proyecto:

# Endpoints disponibles para el usuario de tipo manager

## Obtener lista de programas: */api/v1/programs/{id?}*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| GET | Token | program:id (int) | - | Lista de programas (JSON) |

Tabla 25: Obtener lista de programas (manager) - Endpoints

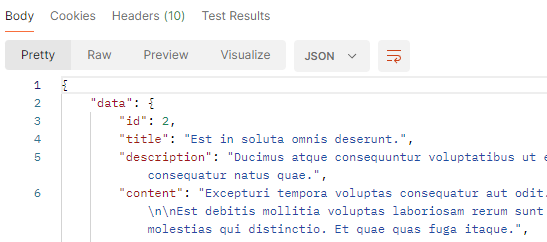
Acción: Obtener toda la información del recurso *programs* o una lista del recurso *programs* dependiendo si se envía un parámetro de búsqueda por medio de la URL por id o no.

Ilustración 6: Respuesta del endpoint /api/v1/programs/{id?} para manager

## Crear un programa: */api/v1/programs*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| POST | Token | - | title (string), description (string), content (string), tags (string) | Programa (JSON) |

Tabla 26: Crear un programa (manager) - Endpoints

Acción: Crear un nuevo registro correspondiente al recurso *programs* que su vez es la persistencia de la tabla *programs* de la base de datos con los parámetros definidos en el cuerpo de la petición.

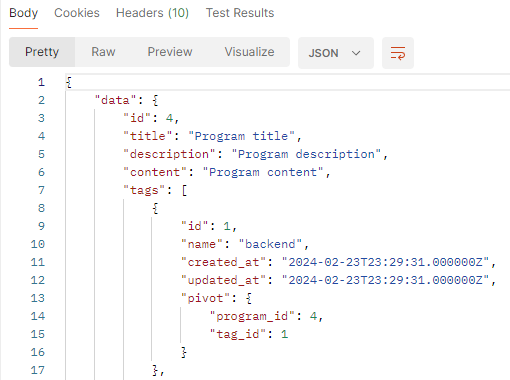


Ilustración 7: Respuesta del endpoint /api/v1/programs para manager

## Crear una ejecución: */api/v1/executions*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| POST | Token | - | program\_id (int), start\_date (date), end\_date (date) | Ejecución (JSON) |

Tabla 27: Crear una ejecución (manager) - Endpoints

Acción: Crear un nuevo registro correspondiente al recurso *executions* que su vez es la persistencia de la tabla *executions* de la base de datos con los parámetros definidos en el cuerpo de la petición. Para crear un registro de este recurso debe existir el programa al que se le está haciendo referencia por el id en el cuerpo de la petición.

Ilustración 8: Respuesta del endpoint /api/v1/executions para manager

## Enrolar un trainee a una ejecución: */api/v1/executions/{id}/enroll-trainee/{id}*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| GET | Token | execution:id (int), user:id (int) | - | Enrolamiento (JSON) |

Tabla 28: Enrolar un trainee a una ejecución (manager) - Endpoints

Acción: Este endpoint tiene como función enrolar a un usuario de tipo trainee a una ejecución que es la puesta en marcha de un programa, para que el resultado sea exitoso el id de la ejecución a la que buscamos enrolar debe ser un id existente dentro de los registros del recurso executions en la base de datos puesto que el prime parámetro por URL de id hace referencia al índice de ese recurso.

Ilustración 9: Respuesta del endpoint /api/v1/executions/{id}/enroll-trainee/{id} para manager

## Obtener lista de ejecuciones: */api/v1/executions/{id?}*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| GET | Token | execution:id (número) | - | Ejecución (JSON) |

Tabla 29: Obtener lista de ejecuciones (manager) - Endpoints

Acción: Obtener toda la información del recurso *executions* o una lista del recurso *executions* dependiendo si se envía un parámetro de búsqueda por medio de la URL por id o no. En este recurso también se incluye en la respuesta la fecha inicio y fecha fin del programa al que se refiere el registro de le ejecución.

Ilustración 10: Respuesta del endpoint /api/v1/executions/{id?} para manager

## Asignar un módulo a una ejecución: */api/v1/executions/assign-trainee-module*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| POST | Token | - | execution\_id (int), module\_id (int), user\_id (int) | Asignación (JSON) |

Tabla 30: Asignar un módulo a una ejecución (manager) - Endpoints

Acción: Asigna un elemento del recurso *modules* de un recurso *programs* para que se habilitado dentro del recurso *executions* del mismo junto con el usuario al que se le habilita el contenido del módulo, los 3 registros a los que referencian los índices del cuerpo de la petición previamente deben existir o ser creados para así poder realizar la acción solicitada correctamente.



Ilustración 11: Respuesta del endpoint /api/v1/executions/assign-trainee-module

para manager

## Crear un empleado: */api/v1/users/create-employee-account*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| POST | Token | - | name (string), email (string), password (string), roles (array [manager?, developer?, trainer?]) | Usuario (JSON) |

Tabla 31: Crea un usuario de tipo no trainee (manager) - Endpoints

Acción: Crea un usuario de tipo manager, developer o trainer, aunque es posible que un usuario contenga los permisos de mas de uno de los roles presentes en el programa, pero si requiere de al menos un parámetro para el rol. Esto se logra porque el parámetro de roles es un arreglo el que se espera en la petición y este puede contener uno o mas elementos.



Ilustración 12: Respuesta del endpoint /api/v1/users/create-employee-account para manager

## Crear un trainee: */api/v1/users/create-trainee-acount*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| POST | Token | - | name (string), email (string), password (string) | Usuario (JSON) |

Tabla 32: Crea un usuario de tipo trainee (manager) - Endpoints

Acción: Crea un usuario que será siempre de tipo trainee, esto es para diferenciar los dos endpoints de creación de usuarios. Uno exclusivamente para los empleados que serán puestos a prueba y otro reservado para los empleados independientemente del tipo de empleado a excepción del que se someterá a un programa de evaluación de conocimientos, es decir un trainee.

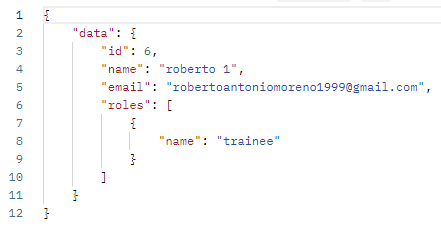


Ilustración 13: Respuesta del endpoint /api/v1/users/create-trainee-account para manager

## Asignar un developer a un programa: */api/v1/programs/{id}/assign-developer/{id}*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| GET | Token | program:id (int),  user:id (int) | - | Mensaje de developer en el programa (JSON) |

Tabla 33: Asignar un developer a un programa (manager) - Endpoints

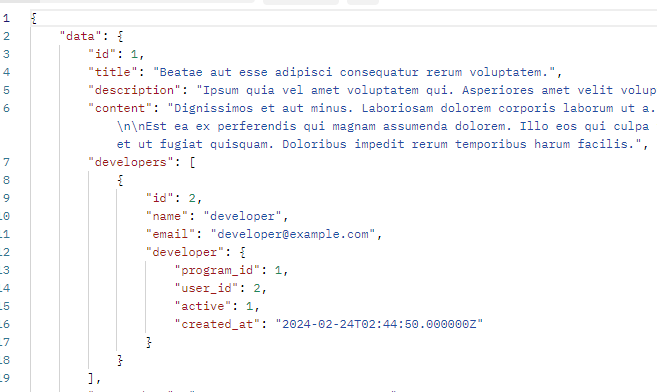
Acción: Asignar un usuario de tipo developer para que sea el encargado de asignar el contenido de un programa, es decir sus modulos, sus temas, sus preguntas y sus criterios de evaluación, según el id seleccionado del programa por parámetro a través de la URL. Tanto el registro de usuario que hace referencia el id del trainer como el id de la ejecución deben estar presentes en los registros de estos recursos para poder efectuarse la acción.

Ilustración 14: Respuesta del endpoint /api/v1/programs/{id}/assign-developer/{id} para manager

## Asignar un trainer a una ejecución: */api/v1/executions/{id}/assign-trainer/{id}*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| GET | Token | execution:id (int), user:id (int) | - | Mensaje de trainer en la ejecución (JSON) |

Tabla 34: Asignar un trainer a una ejecución (manager) - Endpoint

Acción: Asignar un tipo de usuario trainer para que se el mentor de los trainees en la ejecución seleccionada. La ejecución debe ser previamente creada o haber registro de una existente para ser aceptada por el endpoint como parámetro y efectuar la acción de asignación.



Ilustración 15: Respuesta del endpoint /api/v1/executions/{id}/assign-trainer/{id} para manager

## Obtener lista de usuarios: */api/v1/users*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| GET | Token | - | - | Lista de Usuarios (JSON) |

Tabla 35: Obtener una lista de usuarios (manager) - Endpoints

Acción: Obtiene la lista de usuarios registrados en la API.



Ilustración 16: Respuesta del endpoint /api/v1/users para manager

## Obtener detalle de usuarios: */api/v1/users/{id}*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| GET | Token | user:id (int) | - | Detalle de usuario (JSON) |

Tabla 36: Obtener detalle de un usuario para manager - Endpoints

Acción: Obtiene el detalle del usuario seleccionado.



Ilustración 17: Respuesta del endpoint /api/v1/users/{id} para manager

## Editar usuario: /api/v1/users/{id}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| PATCH | Token | users:id (int) | name (string) | Usuario (JSON) |

Tabla 37: Editar nombre de usuario para manager - Endpoints

Acción: Edita el nombre del usuario seleccionado por el id a través del parámetro de la URL, en el caso de que se requiera cambiar cosas mas fundamentales como el tipo de usuario no se hace directamente desde este endpoint, sino que la idea es crear otro usuario de tipo empleado y así poder mantener el registro de todos los trainees.

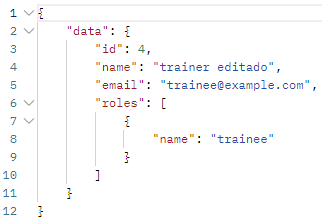


Ilustración 18: Respuesta del endpoint /api/v1/users/{id} para manager

## Eliminar usuario: /api/v1/users/{id}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| DELETE | Token | user:id (int) | - | Mensaje de éxito (JSON) |

Tabla 38: Eliminar usuario para manager - Endpoints

Acción: Eliminar al usuario seleccionado. En realidad, nunca se borran los usuarios solo se agrega un discriminante como eliminado para mantener siempre el registro en la base de datos, pero se muestran solo los que no están marcados como eliminados.

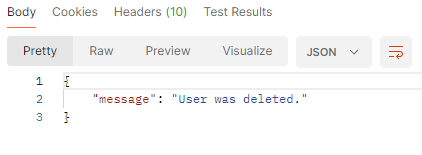


Ilustración 19: Respuesta del endpoint /api/v1/users/{id} para manager

## Obtener lista de programas: /api/v1/programs/{id?}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| GET | Token | id (número) | - | Usuario (JSON) |

Acción: Obtener toda la información de un programa o una lista de programas dependiendo si se envía un parámetro de búsqueda por id o no.



## Obtener lista de programas: /api/v1/programs/{id?}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| GET | Token | id (número) | - | Usuario (JSON) |

Acción: Obtener toda la información de un programa o una lista de programas dependiendo si se envía un parámetro de búsqueda por id o no.



## Obtener lista de programas: /api/v1/programs/{id?}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| GET | Token | id (número) | - | Usuario (JSON) |

Acción: Obtener toda la información de un programa o una lista de programas dependiendo si se envía un parámetro de búsqueda por id o no.



## Obtener lista de programas: /api/v1/programs/{id?}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| GET | Token | id (número) | - | Usuario (JSON) |

Acción: Obtener toda la información de un programa o una lista de programas dependiendo si se envía un parámetro de búsqueda por id o no.



## Obtener lista de programas: /api/v1/programs/{id?}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| GET | Token | id (número) | - | Usuario (JSON) |

Acción: Obtener toda la información de un programa o una lista de programas dependiendo si se envía un parámetro de búsqueda por id o no.



## Obtener lista de programas: /api/v1/programs/{id?}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| GET | Token | id (número) | - | Usuario (JSON) |

Acción: Obtener toda la información de un programa o una lista de programas dependiendo si se envía un parámetro de búsqueda por id o no.



## Obtener lista de programas: /api/v1/programs/{id?}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| GET | Token | id (número) | - | Usuario (JSON) |

Acción: Obtener toda la información de un programa o una lista de programas dependiendo si se envía un parámetro de búsqueda por id o no.



## Obtener lista de programas: /api/v1/programs/{id?}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| GET | Token | id (número) | - | Usuario (JSON) |

Acción: Obtener toda la información de un programa o una lista de programas dependiendo si se envía un parámetro de búsqueda por id o no.



## Obtener lista de programas: /api/v1/programs/{id?}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| GET | Token | id (número) | - | Usuario (JSON) |

Acción: Obtener toda la información de un programa o una lista de programas dependiendo si se envía un parámetro de búsqueda por id o no.



## Obtener lista de programas: /api/v1/programs/{id?}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| GET | Token | id (número) | - | Usuario (JSON) |

Acción: Obtener toda la información de un programa o una lista de programas dependiendo si se envía un parámetro de búsqueda por id o no.



## Obtener lista de programas: /api/v1/programs/{id?}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| GET | Token | id (número) | - | Usuario (JSON) |

Acción: Obtener toda la información de un programa o una lista de programas dependiendo si se envía un parámetro de búsqueda por id o no.



## Obtener lista de programas: /api/v1/programs/{id?}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| GET | Token | id (número) | - | Usuario (JSON) |

Acción: Obtener toda la información de un programa o una lista de programas dependiendo si se envía un parámetro de búsqueda por id o no.



## Obtener lista de programas: /api/v1/programs/{id?}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| GET | Token | id (número) | - | Usuario (JSON) |

Acción: Obtener toda la información de un programa o una lista de programas dependiendo si se envía un parámetro de búsqueda por id o no.



## Obtener lista de programas: /api/v1/programs/{id?}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| GET | Token | id (número) | - | Usuario (JSON) |

Acción: Obtener toda la información de un programa o una lista de programas dependiendo si se envía un parámetro de búsqueda por id o no.



## Obtener lista de programas: /api/v1/programs/{id?}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| GET | Token | id (número) | - | Usuario (JSON) |

Acción: Obtener toda la información de un programa o una lista de programas dependiendo si se envía un parámetro de búsqueda por id o no.



## Obtener lista de programas: /api/v1/programs/{id?}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| GET | Token | id (número) | - | Usuario (JSON) |

Acción: Obtener toda la información de un programa o una lista de programas dependiendo si se envía un parámetro de búsqueda por id o no.



## Obtener lista de programas: /api/v1/programs/{id?}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| GET | Token | id (número) | - | Usuario (JSON) |

Acción: Obtener toda la información de un programa o una lista de programas dependiendo si se envía un parámetro de búsqueda por id o no.



## Obtener lista de programas: /api/v1/programs/{id?}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| GET | Token | id (número) | - | Usuario (JSON) |

Acción: Obtener toda la información de un programa o una lista de programas dependiendo si se envía un parámetro de búsqueda por id o no.



## Obtener lista de programas: /api/v1/programs/{id?}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| GET | Token | id (número) | - | Usuario (JSON) |

Acción: Obtener toda la información de un programa o una lista de programas dependiendo si se envía un parámetro de búsqueda por id o no.



## Obtener lista de programas: /api/v1/programs/{id?}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| GET | Token | id (número) | - | Usuario (JSON) |

Acción: Obtener toda la información de un programa o una lista de programas dependiendo si se envía un parámetro de búsqueda por id o no.



## Obtener lista de programas: /api/v1/programs/{id?}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| GET | Token | id (número) | - | Usuario (JSON) |

Acción: Obtener toda la información de un programa o una lista de programas dependiendo si se envía un parámetro de búsqueda por id o no.



## Obtener lista de programas: /api/v1/programs/{id?}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| GET | Token | id (número) | - | Usuario (JSON) |

Acción: Obtener toda la información de un programa o una lista de programas dependiendo si se envía un parámetro de búsqueda por id o no.



## Obtener lista de programas: /api/v1/programs/{id?}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| GET | Token | id (número) | - | Usuario (JSON) |

Acción: Obtener toda la información de un programa o una lista de programas dependiendo si se envía un parámetro de búsqueda por id o no.



ENDPOINTS

Sdadas

Endpoints

Diccionadio de datos

Tabla de tablas

Tablas de atributos por tabla

referencias

# Glosario

API

Endpoints:

Endpoints y Métodos HTTP en APIs

Los endpoints son las URL que permiten a los clientes (como aplicaciones web o móviles) acceder a los recursos de una API. Cada endpoint tiene una ruta específica y un método HTTP asociado que define la acción que se desea realizar.

Métodos HTTP:

GET: Se utiliza para recuperar datos de un recurso.

POST: Se utiliza para crear un nuevo recurso.

PUT: Se utiliza para actualizar un recurso existente.

PATCH: Se utiliza para actualizar un atributo de un recurso existente.

DELETE: Se utiliza para eliminar un recurso.

Ejemplo GET:

Endpoint: /api/users/{id}

Método: GET

Donde “users” corresponde al recurso solicitado, es decir que lo que se le está pidiendo al servidor es obtener datos del registro de usuarios y el parámetro “{id}” se refiere al id del usuario que se quiere acceder dicha información. A esto se le conoce como método *show*, porque se quiere mostrar el detalle del recurso usuario. Cuando no se envía el parámetro de id se entiende que quiere la lista completa de usuarios y a esta lista genérica de un recurso se le conoce como método *index*.

Descripción: Obtiene información sobre un usuario específico.

Ejemplo: POST

Endpoint: /api/users

Método: POST

En la URL se apunta al recurso con el método POST es cuando significa que se quiere crear un nuevo registro de dicho recurso, en este ejemplo se está creando un nuevo usuario.

Descripción: Crea un nuevo usuario.

Endpoint: /api/users/{id}

Método: PUT

Descripción: Actualiza la información de un usuario existente.

Endpoint: /api/users/{id}

Método: DELETE

Descripción: Elimina un usuario.

Body:

Request header:

Authorization:

Api rest

Cliente http

Requerimientos funcionales

url

persistencia

depuración de codigo

Requerimientos no funcionales

script

Historias de usuario

RESTful

Plataforma

peticion

Header.

Programación

Migrations

Models

Resources

Mysql

Licencia

Visual studio

PHP

Lenguaje de programación

framework

Star uml

Login

git

Github

json

Postman

Git

Github

Peticion

Gitlab

Heidysql

Sql

OAuth

Navegador web

Mvc

Editor de texto

Cliente http

http

ofimatica

backend

frontend

código http

repositorio de archivos

git

outsorcing

token

datos fake

Bibliografía

Postman

Swagger.io

*https://restfulapi.net/*