**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**

**2018-MZ-601**

**PARA OPTAR AL TÍTULO DE:**

**INGENIERO EN DESARROLLO DE SOFTWARE**

**SANTA ANA, FEBRERO DE 2024**

**UNIVERSIDAD CATÓLICA DE EL SALVADOR**

**FACULTAD DE INGENIERÍA Y ARQUITECTURA**

**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, agradezco a Dios por ser mi guía y faro durante el transcurso de mi vida; por iluminarme y darme la ciencia en mis estudios y siempre estar a mi lado.

A mi madre, Ana Carolina de Moreno, por siempre estar junto a mí en todo momento de mi vida y especialmente durante esta etapa de aprendizaje 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 hasta su partida; por todas las enseñanzas que pudo sembrar en mi memoria y por estar presente en mis pensamientos y corazón todos los días de mi vida.

A mi familia, que me ha apoyado de una u otra forma especialmente a mis hermanos Toby y Luisito, por sus incontables noches en desvelo junto a mí, por darme animo en todas las etapas de mi vida, especialmente durante mi formación académica; el apoyo incondicional y su paciencia para orientarme; y también a Ligia, quien siempre ha estado presente y darme su ayuda cuando lo he necesitado, su paciencia y su fe en mi esfuerzo.

A mis, compañeros quienes han acompañado mi viaje estos 6 años especialmente a mi amigo Balmore por la orientación que me brindo en el transcurso de mi carrera y aportar 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.

**ÍNDICE DE** **CONTENIDO**

[INTRODUCCIÓN 1](#_Toc160548318)

[ANTECEDENTES DE LA INSTITUCIÓN 2](#_Toc160548319)

[Ubicación geográfica 2](#_Toc160548320)

[Misión 2](#_Toc160548321)

[Visión 3](#_Toc160548322)

[Valores 3](#_Toc160548323)

[DESCRIPCIÓN DEL PROBLEMA 4](#_Toc160548324)

[JUSTIFICACIÓN 6](#_Toc160548325)

[ALCANCE 8](#_Toc160548326)

[OBJETIVOS 9](#_Toc160548327)

[Objetivo General 9](#_Toc160548328)

[Objetivos Específicos 9](#_Toc160548329)

[DESCRIPCIÓN NARRATIVA DE LOS OBJETIVOS 10](#_Toc160548330)

[ACTORES INTERESADOS DEL PROYECTO 13](#_Toc160548331)

[PLAN DE TRABAJO GENERAL 15](#_Toc160548332)

[Fase 1: 15](#_Toc160548333)

[Fase 2: 15](#_Toc160548334)

[Fase 3: 16](#_Toc160548335)

[Fase 4: 16](#_Toc160548336)

[PLAN DE TRABAJO INDIVIDUAL 17](#_Toc160548337)

[Cronograma por presupuesto 17](#_Toc160548338)

[Herramientas para el desarrollo del sistema 19](#_Toc160548339)

[Tabla de herramientas y costos 21](#_Toc160548340)

[Recursos 22](#_Toc160548341)

[Recursos de hardware. 22](#_Toc160548342)

[Recursos de software para el desarrollo del sistema 22](#_Toc160548343)

[DESARROLLO 23](#_Toc160548344)

[Requerimientos funcionales: 23](#_Toc160548345)

[Requisitos no funcionales 24](#_Toc160548346)

[Descripción de la API 24](#_Toc160548347)

[Manager: 26](#_Toc160548348)

[Developer: 26](#_Toc160548349)

[Trainer: 27](#_Toc160548350)

[Trainee: 27](#_Toc160548351)

[Generales: 27](#_Toc160548352)

[Tecnologías utilizadas: 28](#_Toc160548353)

[Ingles 29](#_Toc160548354)

[Laravel 29](#_Toc160548355)

[MySQL 31](#_Toc160548356)

[Control de versiones 31](#_Toc160548357)

[Diseño del sistema 32](#_Toc160548358)

[Diagrama Entidad Relación 33](#_Toc160548359)

[Diccionario de datos 34](#_Toc160548360)

[Tablas 34](#_Toc160548361)

[Entidades 35](#_Toc160548362)

[Usuario 35](#_Toc160548363)

[Rol-Usuario 35](#_Toc160548364)

[Roles 35](#_Toc160548365)

[Habilidad-Rol 35](#_Toc160548366)

[Etiquetas 35](#_Toc160548367)

[Programa-Etiqueta 36](#_Toc160548368)

[Programas 36](#_Toc160548369)

[Módulos 36](#_Toc160548370)

[Temas 37](#_Toc160548371)

[Preguntas 37](#_Toc160548372)

[Criterios de Evaluación 37](#_Toc160548373)

[Notas 38](#_Toc160548374)

[Desarrolladores 38](#_Toc160548375)

[Entrenadores 38](#_Toc160548376)

[Ejecuciones 39](#_Toc160548377)

[Asignaciones 39](#_Toc160548378)

[Enrolamientos 40](#_Toc160548379)

[Llaves primarias 40](#_Toc160548380)

[Cardinalidad 41](#_Toc160548381)

[RESULTADOS 42](#_Toc160548382)

[Endpoints disponibles para el usuario de tipo manager 42](#_Toc160548383)

[Obtener lista de programas: */api/v1/programs/{id?}* 42](#_Toc160548384)

[Crear un programa: */api/v1/programs* 43](#_Toc160548385)

[Crear una ejecución: */api/v1/executions* 44](#_Toc160548386)

[Enrolar un trainee a una ejecución: */api/v1/executions/{id}/enroll-trainee/{id}* 45](#_Toc160548387)

[Obtener lista de ejecuciones: */api/v1/executions/{id?}* 45](#_Toc160548388)

[Asignar un módulo a una ejecución: */api/v1/executions/assign-trainee-module* 46](#_Toc160548389)

[Crear un empleado: */api/v1/users/create-employee-account* 47](#_Toc160548390)

[Crear un trainee: */api/v1/users/create-trainee-acount* 48](#_Toc160548391)

[Asignar un developer a un programa: */api/v1/programs/{id}/assign-developer/{id}* 49](#_Toc160548392)

[Asignar un trainer a una ejecución: */api/v1/executions/{id}/assign-trainer/{id}* 49](#_Toc160548393)

[Obtener lista de usuarios: */api/v1/users* 50](#_Toc160548394)

[Obtener detalle de usuarios: */api/v1/users/{id}* 51](#_Toc160548395)

[Editar usuario: */api/v1/users/{id}* 51](#_Toc160548396)

[Eliminar usuario: */api/v1/users/{id}* 52](#_Toc160548397)

[Endpoints disponibles para el tipo de usuario developer 52](#_Toc160548398)

[Obtener lista de programas a los que pertenece: */api/v1/programs* 52](#_Toc160548399)

[Obtener detalle de un programa al que pertenece: */api/v1/programs/{id}* 53](#_Toc160548400)

[Crear un módulo: */api/v1/modules* 53](#_Toc160548401)

[Edita un módulo: */api/v1/modules/{id}* 54](#_Toc160548402)

[Elimina un módulo: */api/v1/modules/{id}* 54](#_Toc160548403)

[Crea un tema: */api/v1/topics* 55](#_Toc160548404)

[Edita un tema: */api/v1/topics/{id}* 55](#_Toc160548405)

[Elimina un tema: */api/v1/topics/{id}* 56](#_Toc160548406)

[Crear una pregunta: */api/v1/questions* 56](#_Toc160548407)

[Editar una pregunta: */api/v1/questions/{id}* 57](#_Toc160548408)

[Eliminar una pregunta: */api/v1/questions/{id}* 57](#_Toc160548409)

[Crea un criterio de evaluación: */api/v1/evaluations* 57](#_Toc160548410)

[Editar un criterio de evaluación: */api/v1/evaluations/{id}* 58](#_Toc160548411)

[Eliminar un criterio de evaluación: */api/v1/evaluations/{id}* 58](#_Toc160548412)

[Endpoints disponibles para el tipo de usuario trainer 59](#_Toc160548413)

[Obtener lista de ejecuciones a las que pertenece: */api/v1/executions* 59](#_Toc160548414)

[Obtener detalle de ejecución al que pertenece: */api/v1/executions/{id}* 59](#_Toc160548415)

[Inicia la entrevista: */api/v1/assignments/{id}/interview-start* 60](#_Toc160548416)

[Guardar respuestas de entrevista: */api/v1/assignments/save-question* 60](#_Toc160548417)

[Finaliza la entrevista: */api/v1/assignments/{id}/interview-finish* 61](#_Toc160548418)

[Inicia evaluación de código: */api/v1/assignments/{id}/homework-start* 62](#_Toc160548419)

[Guardar evaluación de código: */api/v1/assignments/save-evaluation-criteria* 62](#_Toc160548420)

[Finaliza la evaluación de código: */api/v1/assignments/{id}/homework-finish* 64](#_Toc160548421)

[Obtener reporte de trainee: */api/v1/report/{id}* 64](#_Toc160548422)

[Endpoints disponibles para el tipo de usuario trainer 65](#_Toc160548423)

[Obtener listado de ejecuciones a las que pertenece: */api/v1/executions* 65](#_Toc160548424)

[Obtener detalle de ejecución a la que pertenece: */api/v1/executions/{id}* 66](#_Toc160548425)

[Carga el código a evaluar: */api/v1/executions* 66](#_Toc160548426)

[Endpoints disponibles para todos los usuarios 67](#_Toc160548427)

[Autentificación de usuario por plataforma interna: */api/v1/login* 67](#_Toc160548428)

[Autentificación de usuario por Google OAuth 2.0: */api/v1/social/google/login* 67](#_Toc160548429)

[ELEMENTOS DE SOPORTE 69](#_Toc160548430)

[1. Investigación y planificación: 69](#_Toc160548431)

[2. Desarrollo: 69](#_Toc160548432)

[3. Pruebas y entrega: 70](#_Toc160548433)

[4. Impacto: 70](#_Toc160548434)

[Tecnologías utilizadas: 70](#_Toc160548435)

[Herramientas utilizadas: 71](#_Toc160548436)

[PRUEBAS UNITARIAS 71](#_Toc160548437)

[EXPOSICIÓN DE RESULTADOS FINALES AL GERENTE 79](#_Toc160548438)

[BIBLIOGRAFIA 81](#_Toc160548439)

[ELEMENTOS DE SOPORTE 82](#_Toc160548440)

[Anexo 1: Glosario 82](#_Toc160548441)

[Anexo 2: Manual de usuario 90](#_Toc160548442)

[Anexo 3: Carta de solicitud de creación por parte de la empresa 91](#_Toc160548443)

[CARTA DE RECEPCCIÓN DEL PROYECTO 92](#_Toc160548444)

**ÍNDICE DE IMÁGENES**

[Ilustración 1: Cronograma de actividades 15](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159800778)

[Ilustración 2: Cronograma de por presupuesto agosto a septiembre 17](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159800779)

[Ilustración 3: Cronograma de por presupuesto diciembre a enero 18](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159800780)

[Ilustración 4: Cronograma de por presupuesto octubre a noviembre 18](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159800781)

[Ilustración 5: Diagrama ERD 33](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159800782)

[Ilustración 6: Respuesta del endpoint /api/v1/programs/{id?} para manager 43](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159800783)

[Ilustración 7: Respuesta del endpoint /api/v1/programs para manager 44](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159800784)

[Ilustración 8: Respuesta del endpoint /api/v1/executions para manager 44](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159800785)

[Ilustración 9: Respuesta del endpoint /api/v1/executions/{id}/enroll-trainee/{id} para manager 45](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159800786)

[Ilustración 10: Respuesta del endpoint /api/v1/executions/{id?} para manager 46](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159800787)

[Ilustración 11: Respuesta del endpoint /api/v1/executions/assign-trainee-module 47](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159800788)

[Ilustración 12: Respuesta del endpoint /api/v1/users/create-employee-account para manager 48](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159800789)

[Ilustración 13: Respuesta del endpoint /api/v1/users/create-trainee-account para manager 48](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159800790)

[Ilustración 14: Respuesta del endpoint /api/v1/programs/{id}/assign-developer/{id} para manager 49](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159800791)

[Ilustración 15: Respuesta del endpoint /api/v1/executions/{id}/assign-trainer/{id} para manager 50](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159800792)

[Ilustración 16: Respuesta del endpoint /api/v1/users para manager 50](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159800793)

[Ilustración 17: Respuesta del endpoint /api/v1/users/{id} para manager 51](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159800794)

[Ilustración 18: Respuesta del endpoint /api/v1/users/{id} (PATCH) 51](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159800795)

[Ilustración 19: Respuesta del endpoint /api/v1/users/{id} (DELETE) 52](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159800796)

[Ilustración 20: Respuesta del endpoint /api/v1/programs para developer 52](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159800797)

[Ilustración 21: Respuesta del endpoint /api/v1/programs/{id} para developer 53](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159800798)

[Ilustración 22: Respuesta del endpoint /api/v1/modules (POST) 53](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159800799)

[Ilustración 23: Respuesta del endpoint /api/v1/modules/{id} (PATCH) 54](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159800800)

[Ilustración 24: Respuesta del endpoint /api/v1/modules/{id} (DELETE) 54](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159800801)

[Ilustración 25: Respuesta del endpoint /api/v1/topics (POST) 55](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159800802)

[Ilustración 26: Respuesta del endpoint /api/v1/topics/{id} (PATCH) 55](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159800803)

[Ilustración 27: Respuesta del endpoint /api/v1/topics/{id} (DELETE) 56](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159800804)

[Ilustración 28: Respuesta del endpoint /api/v1/questions (POST) 56](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159800805)

[Ilustración 29: Respuesta del endpoint /api/v1/questions/{id} (PATCH) 57](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159800806)

[Ilustración 30: Respuesta del endpoint /api/v1/questions/{id} (DELETE) 57](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159800807)

[Ilustración 31: Respuesta del endpoint /api/v1/evaluations (POST) 57](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159800808)

[Ilustración 32: Respuesta del endpoint /api/v1/evaluations/{id} (PATCH) 58](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159800809)

[Ilustración 33: Respuesta del endpoint /api/v1/evaluations/{id} (DELETE) 58](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159800810)

[Ilustración 34: Respuesta del endpoint /api/v1/executions para trainer 59](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159800811)

[Ilustración 35: Respuesta del endpoint /api/v1/executions/{id} para trainer 59](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159800812)

[Ilustración 36: Respuesta del denpoint /api/v1/assignments/{id}/interview-start para trainer 60](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159800813)

[Ilustración 37: Respuesta del endpoint /api/v1/assignments/save-question para trainer 61](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159800814)

[Ilustración 38: Respuesta del endpoint /api/v1/assignments/{id}/interview-finish para trainer 62](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159800815)

[Ilustración 39: Respuesta del endpoint /api/v1/assignments/{id}/homework-start para trainer 62](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159800816)

[Ilustración 40: Respuesta del endpoint /api/v1/assignments/save-evaluation-criteria para trainer 63](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159800817)

[Ilustración 41: Respuesta del endpoint /api/v1/assignments/{id}/homework-finish para trainer 64](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159800818)

[Ilustración 42: Crea el reporte dentro de la carpeta publica del proyecto 64](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159800819)

[Ilustración 43: Respuesta del endpoint /api/v1/report/{id} para trainer 64](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159800820)

[Ilustración 44: Reporte csv del promedio final de trainee 65](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159800821)

[Ilustración 45: Respuesta del endpoint /api/v1/executions para trainee 65](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159800822)

[Ilustración 46: Respuesta del endpoint /api/v1/executions/{id} para trainee 66](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159800823)

[Ilustración 47: Respuesta del endpoint /api/v1/executions para trainee 66](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159800824)

[Ilustración 48: Respuesta del endpoint /api/v1/login para todos los usuarios 67](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159800825)

[Ilustración 49: Si las credenciales son correctas el usuario es autentificado 68](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159800826)

[Ilustración 50: Respuesta del endpoint /api/v1/social/google/login donde procedemos a autentificarnos en Google 68](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159800827)

[Ilustración 51: Los 158 tests finalizados de forma satisfactoria 78](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159800828)

[Ilustración 52: Reunión sobre avances finales y seguimiento de resultados 78](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159800829)

[Ilustración 53: El facilitador (izquierda) y el desarrollador (derecha) del proyecto en la reunión de exposición de resultados. 79](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159800830)

[Ilustración 54: Demostración de las configuraciones de google cloud platform para el OAuth 79](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159800831)

[Ilustración 55: Explicación de los cuatro tipos de tokens 79](file:///C:\UNICAES\Trabajo%20de%20Graduacion\informe-final\Informe%20Final%20-%20API%20Evaluacion%20de%20Empleados%20-%20Sistemas%20Pliret.docx#_Toc159800832)

**ÍNDICE DE TABLAS**

[Tabla 1: Definición y análisis de herramientas con sus costos 20](#_Toc159800833)

[Tabla 2: Recursos humanos - Recursos 21](#_Toc159800834)

[Tabla 3: Recursos de hardware - Recursos 21](#_Toc159800835)

[Tabla 4: Recursos de software - Recursos 22](#_Toc159800836)

[Tabla 5: Tablas 34](#_Toc159800837)

[Tabla 6: Usuarios – Entidades 35](#_Toc159800838)

[Tabla 7: Rol-Usuario – Entidades 35](#_Toc159800839)

[Tabla 8: Roles – Entidades 35](#_Toc159800840)

[Tabla 9:Habilidad-Rol -Entidades 35](#_Toc159800841)

[Tabla 10: Etiquetas – Entidades 36](#_Toc159800842)

[Tabla 11: Programa-Etiqueta – Entidades 36](#_Toc159800843)

[Tabla 12: Programas – Entidades 36](#_Toc159800844)

[Tabla 13: Módulos – Entidades 37](#_Toc159800845)

[Tabla 14: Temas – Entidades 37](#_Toc159800846)

[Tabla 15: Preguntas – Entidades 37](#_Toc159800847)

[Tabla 16: Criterios de Evaluación – Entidades 38](#_Toc159800848)

[Tabla 17: Notas – Entidades 38](#_Toc159800849)

[Tabla 18: Desarrolladores – Entidades 38](#_Toc159800850)

[Tabla 19: Entrenadores – Entidades 39](#_Toc159800851)

[Tabla 20: Ejecuciones – Entidades 39](#_Toc159800852)

[Tabla 21: Asignaciones – Entidades 40](#_Toc159800853)

[Tabla 22: Enrolamientos - Entidades 40](#_Toc159800854)

[Tabla 23: Llaves primarias - Diccionario de datos 41](#_Toc159800855)

[Tabla 24: Cardinalidad - Diccionario de datos 42](#_Toc159800856)

[Tabla 25: Obtener lista de programas (manager) - Endpoints 43](#_Toc159800857)

[Tabla 26: Crear un programa (manager) - Endpoints 43](#_Toc159800858)

[Tabla 27: Crear una ejecución (manager) - Endpoints 44](#_Toc159800859)

[Tabla 28: Enrolar un trainee a una ejecución (manager) - Endpoints 45](#_Toc159800860)

[Tabla 29: Obtener lista de ejecuciones (manager) - Endpoints 45](#_Toc159800861)

[Tabla 30: Asignar un módulo a una ejecución (manager) - Endpoints 46](#_Toc159800862)

[Tabla 31: Crea un usuario de tipo no trainee (manager) - Endpoints 47](#_Toc159800863)

[Tabla 32: Crea un usuario de tipo trainee (manager) - Endpoints 48](#_Toc159800864)

[Tabla 33: Asignar un developer a un programa (manager) - Endpoints 49](#_Toc159800865)

[Tabla 34: Asignar un trainer a una ejecución (manager) - Endpoint 49](#_Toc159800866)

[Tabla 35: Obtener una lista de usuarios (manager) - Endpoints 50](#_Toc159800867)

[Tabla 36: Obtener detalle de un usuario (manager) - Endpoints 51](#_Toc159800868)

[Tabla 37: Editar nombre de usuario (manager) - Endpoints 51](#_Toc159800869)

[Tabla 38: Eliminar usuario (manager)- Endpoints 52](#_Toc159800870)

[Tabla 39: Lista de programas a los que pertenece el developer - Endpoints 52](#_Toc159800871)

[Tabla 40: Obtiene el detalle de un programa al que es encargado el developer - Endpoints 53](#_Toc159800872)

[Tabla 41: Crea un módulo - Endpoints 53](#_Toc159800873)

[Tabla 42: Edita un módulo - Endpoints 54](#_Toc159800874)

[Tabla 43: Elimina un módulo - Endpoints 54](#_Toc159800875)

[Tabla 44: Crea un tema - Endpoints 55](#_Toc159800876)

[Tabla 45: Edita un tema - Endpoints 55](#_Toc159800877)

[Tabla 46: Elimina un tema - Endpoints 56](#_Toc159800878)

[Tabla 47: Crea una pregunta - Endpoints 56](#_Toc159800879)

[Tabla 48: Edita una pregunta - Endpoints 56](#_Toc159800880)

[Tabla 49: Elimina una pregunta - Endpoints 57](#_Toc159800881)

[Tabla 50: Crea un criterio de evaluación - Endpoints 57](#_Toc159800882)

[Tabla 51: Edita un criterio de evaluación - Endpoints 58](#_Toc159800883)

[Tabla 52: Elimina un criterio de evaluación - Endpoints 58](#_Toc159800884)

[Tabla 53: Obtener lista de ejecuciones (trainer) - Endpoints 58](#_Toc159800885)

[Tabla 54: Obtener detalle de ejecución (trainer) - Endpoints 59](#_Toc159800886)

[Tabla 55: Iniciar entrevista (trainer) - Endpoints 60](#_Toc159800887)

[Tabla 56: Guarda la información obtenida de la entrevista (trainer) - Endpoints 60](#_Toc159800888)

[Tabla 57: Finaliza y promedia la entrevista (trainer) - Endpoints 61](#_Toc159800889)

[Tabla 58: Iniciar evaluación de código (trainer) - Endpoints 62](#_Toc159800890)

[Tabla 59: Guarda la información de la evaluación de codigo (trainer) - Endpoints 62](#_Toc159800891)

[Tabla 60: Finaliza y promedia la evaluación de código (trainer) - Endpoints 63](#_Toc159800892)

[Tabla 61: Crea el reporte de la ejecución (trainer) - Endpoints 64](#_Toc159800893)

[Tabla 62: Lista las ejecuciones para el trainee - Endpoints 65](#_Toc159800894)

[Tabla 63: Lista las ejecuciones a las que pertenece el trainee - Endpoints 66](#_Toc159800895)

[Tabla 64: Carga código a evaluar - Endpoints 66](#_Toc159800896)

[Tabla 65: Autentificación interna de usuario - Endpoints 67](#_Toc159800897)

[Tabla 66: Autentificacion por google - Endpoints 67](#_Toc159800898)

[Tabla 67: Unit Testing de funcionalidades de la API 77](#_Toc159800899)

# 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 2012 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 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 de programa de contenidos que posteriormente al desarrollo del sistema serán definidos por la empresa.

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.

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.

# ACTORES INTERESADOS DEL 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.

# PLAN DE TRABAJO GENERAL

La planificación de la API de evaluación de conocimiento de empleados se realizó siguiendo un enfoque metódico que comprendió cuatro etapas principales las cuales se plasman en el siguiente cronograma.

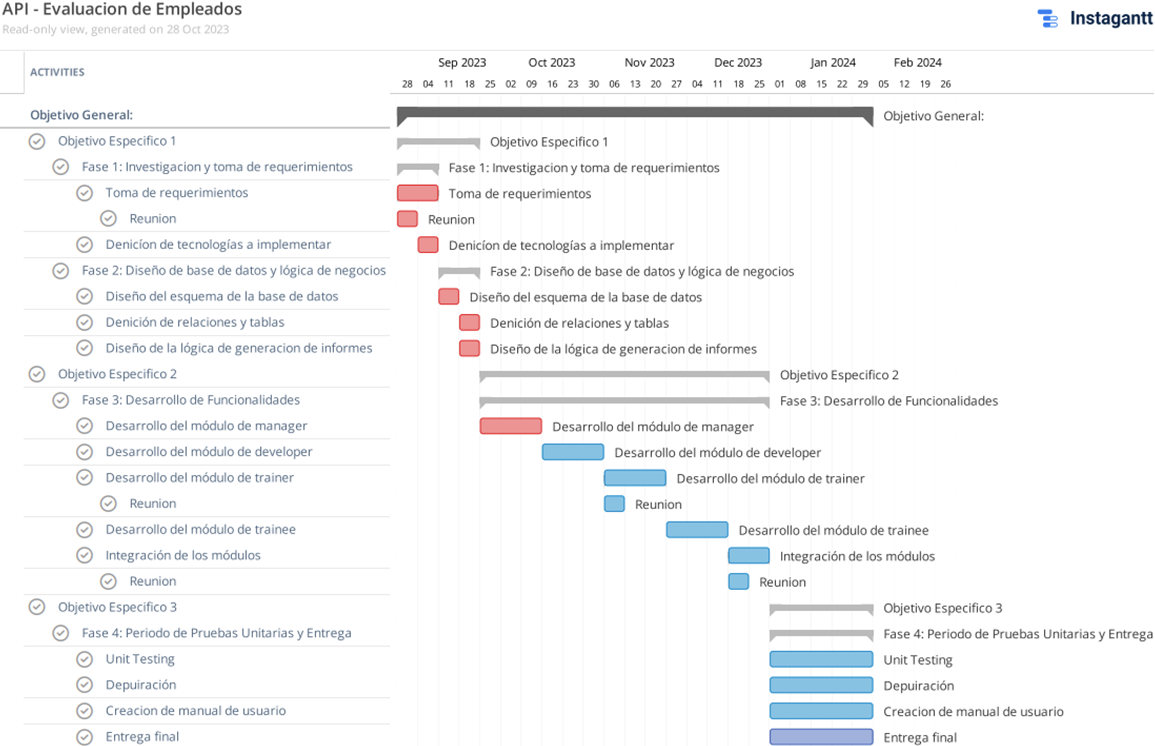
****

Ilustración 1: Cronograma de actividades

# Fase 1:

La fase de investigación y toma de requerimientos se desarrolló entre el 28 de agosto y el 8 de septiembre de 2023, con una duración total de 12 días. Si bien este período puede parecer breve, fue cuidadosamente planificado y ejecutado para asegurar la obtención de información precisa y relevante para el éxito del proyecto.

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

# Fase 2:

El diseño de la base de datos y la lógica de negocios son dos pilares fundamentales en el desarrollo de cualquier software. La fase dedicada a estas tareas se llevó a cabo entre el 11 y el 22 de septiembre de 2023, con una duración de 12 días. Durante este período, se definieron las estructuras que almacenarán la información del sistema y las reglas que gobernarán su comportamiento.

* 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:

El desarrollo de funcionalidades es la etapa donde el proyecto cobra vida. En esta fase, se implementó la lógica de negocios y se construyeron los *endpoints* de usuario que permiten a los usuarios interactuar con el sistema. La fase de desarrollo de funcionalidades se llevó a cabo entre el 25 de septiembre de 2023 y el 29 de diciembre de 2023, con una duración de 96 días.

* 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:

En esta fase, se asegura la calidad del sistema mediante la ejecución de pruebas exhaustivas y se entrega el producto final al cliente. Este periodo se llevó a cabo entre el 1 y el 31 de enero de 2024, con una duración de 31 días.

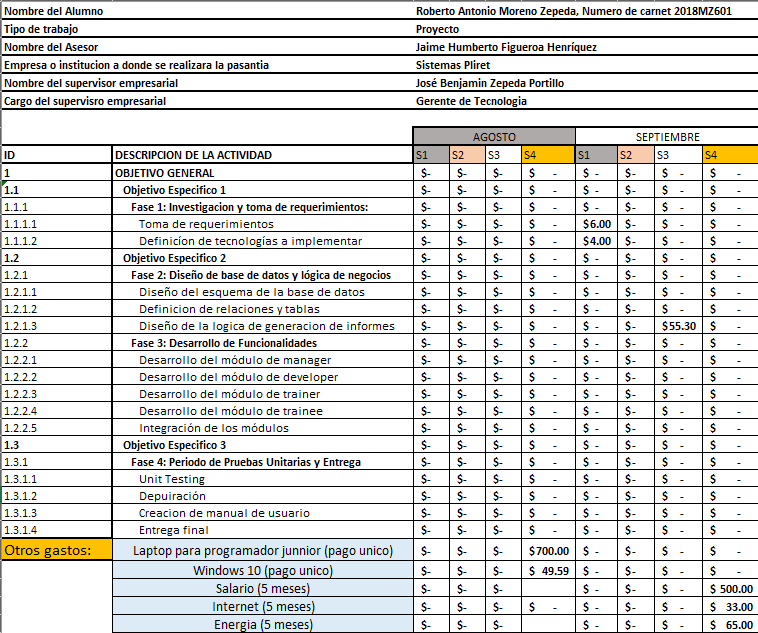
* Depuración (1 de enero al 31 de enero).
* Creación de manual de usuario (1 de enero al 31 de enero).
* Entrega final (1 de enero al 31 de enero).

# PLAN DE TRABAJO INDIVIDUAL

Para el desarrollo del proyecto se elaboró una lista detallada de las herramientas necesarias, incluyendo su costo individual y la distribución de estos costos a lo largo de los cinco meses que duró el desarrollo de la API. Esta lista abarcaba tanto software como hardware, licencias, suscripciones a servicios y cualquier otro recurso indispensable para la ejecución del proyecto.

El cronograma por presupuesto dividió las cuatro etapas de desarrollo (definición y análisis, diseño y desarrollo, implementación y lanzamiento, y monitoreo y mejora) en los cinco meses planeados para el proyecto. Se asignó un presupuesto específico a cada etapa, tomando en cuenta la complejidad de las tareas a realizar, los recursos necesarios y la duración estimada de cada fase.

# Cronograma por presupuesto



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

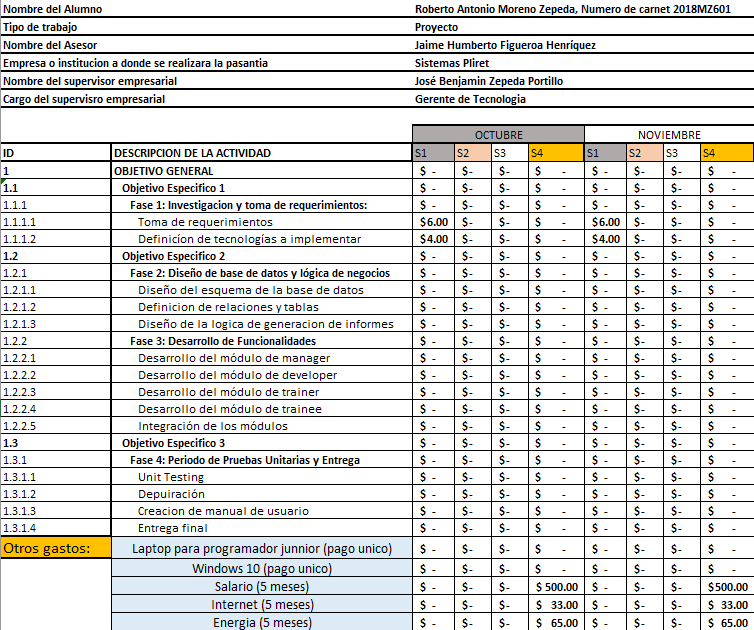


Ilustración 3: Cronograma de por presupuesto octubre a noviembre

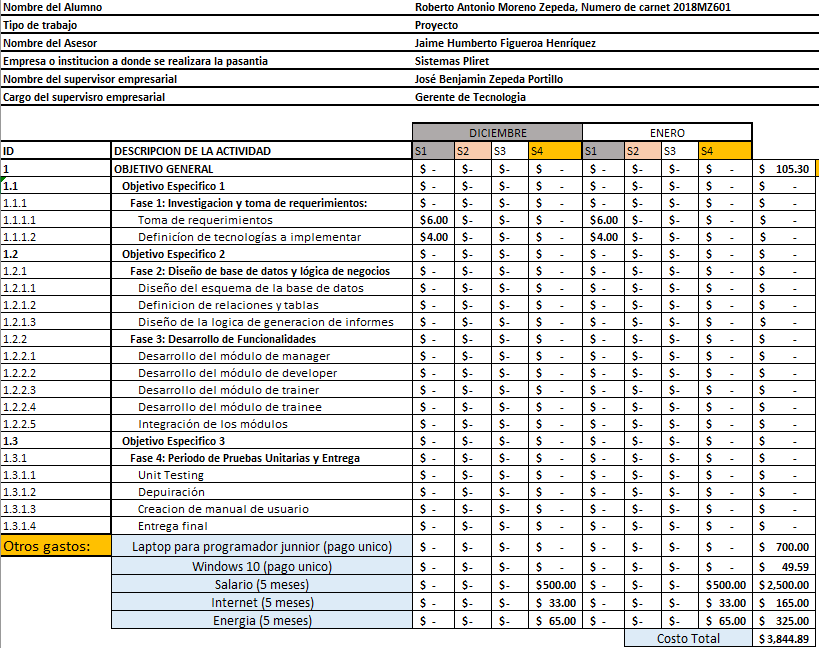


Ilustración 4: Cronograma de por presupuesto diciembre a enero

# 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 que abarca los 5 meses de desarrollo y desglosa las actividades diferenciadas por las 4 fases del desarrollo que se planearon para la realización de la *API*, el cual fue aprobado formalmente por las partes involucradas.

# Tabla de herramientas y costos

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Concepto** | **Meses a usar** | **Costo por mes** | **Observación** | **Patrocinador** | **Costo Total** |
| PHP, Laravel, MySQL. | 5 | $0.00 | - | - | $0.00 |
| Computador tipo laptop | - | $700.00 | Costo estimado | - | $700.00 |
| Instagantt | 5 | $6.00 | Comercial | Estudiante | $30.00 |
| Licencia Windows 10 Home | - | $49.59 | Comercial | - | $49.59 |
| Salario Programador Junior | 5 | $500.00 | Suponiéndose que hubiese pago para el programador | - | $2,500 |
| Internet | 5 | $33.00 | Contratado por el estudiante | Estudiante | $165.00 |
| Energía eléctrica | 5 | $65.00 | Contratado por el estudiante | Estudiante | $325.00 |
| Visual Studio | 5 | $4.00 | Open Source | Estudiante | $20.00 |
| 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

# DESARROLLO

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.

# Descripción 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.

# Lenguaje natural de la codificación

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* está 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.

El diagrama entidad relación o ERD es una herramienta gráfica que permite representar la estructura de una base de datos de forma conceptual. Se compone de dos elementos principales:

**Entidades**: Representan objetos, personas o conceptos del mundo real que se quieren almacenar en la base de datos. Se representan como rectángulos con su nombre en el interior y hacen referencia a las tablas que almacenaran los registros de los diferentes recursos en el proyecto.

**Relaciones**: Representan las conexiones entre las entidades. Se representan como líneas que unen las entidades y tienen un nombre que describe la relación. Pueden ser de 1 a 1 (1:1), 1 a muchos (1:M) o de muchos a muchos (M:M).

# Diagrama Entidad Relación

La base de datos creada para el proyecto se codifico a partir del siguiente diagrama entidad relación. El diagrama corresponde a la siguiente imagen que consta de 18 tablas que representan a las entidades que se interrelacionan para llevar a cabo las diferentes funcionalidades de la API. Cabe aclarar que para este proyecto todas las tablas creadas poseen una cardinalidad de uno a muchos (1:M) para asegurar una futura escalabilidad en los tipos de permisos que puede llegar a tener un usuario como en las múltiples ejecuciones de programas a los que pueden pertenecer tanto los desarrolladores como los usuarios trainers y trainees que serán puesto a la evaluación de conocimiento.

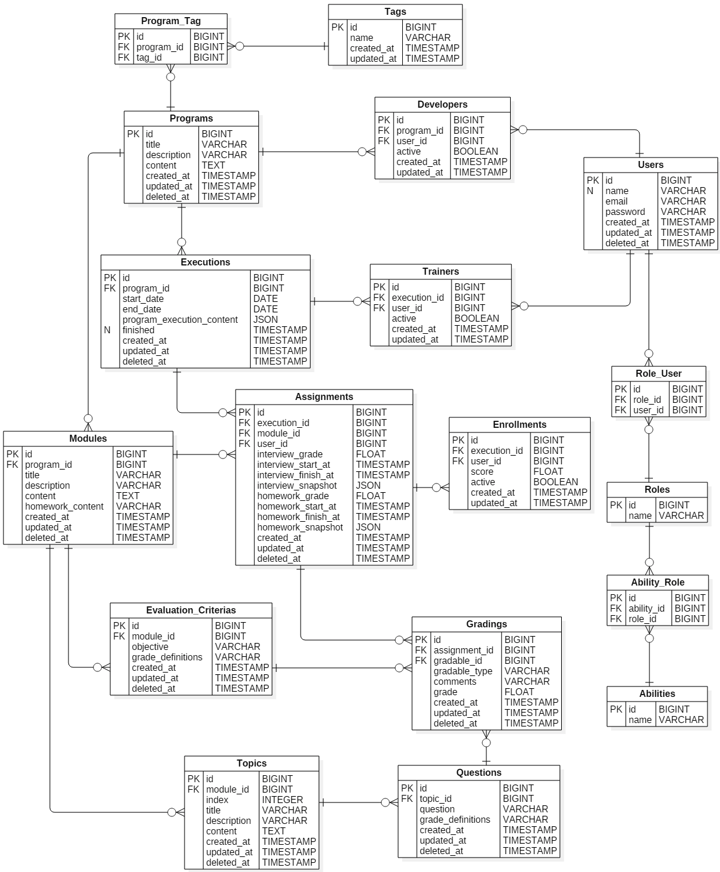


Ilustración 3: Diagrama ERD

# Diccionario de datos

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

También muestran la posible nulidad de estos, es decir que al momento de su creación pueden no tener un valor definido, y si la columna representa a la llave primaria de la tabla.

# Tablas de la base de datos

|  |  |
| --- | --- |
| **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 de la base de datos

## Tabla 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: Tabla de Usuarios – Entidades

## Tabla Rol-Usuario

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

Tabla 7: Tabla de Rol-Usuario – Entidades

## Tabla Roles

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

Tabla 8: Tabla de Roles – Entidades

## Tabla Habilidad-Rol

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

Tabla 9:Tabla de Habilidad-Rol -Entidades

## Tabla 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: Tabla de Etiquetas – Entidades

## Tabla 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: Tabla de Programa-Etiqueta – Entidades

## Tabla 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: Tabla de Programas – Entidades

## Tabla 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: Tabla de Módulos – Entidades

## Tabla 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: Tabla de Temas – Entidades

## Tabla 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: Tabla de Preguntas – Entidades

## Tabla 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: Tabla de Criterios de Evaluación – Entidades

## Tabla 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:Tabla de Notas – Entidades

## Tabla 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: Tabla de Desarrolladores – Entidades

## Tabla 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: Tabla de Entrenadores – Entidades

## Tabla 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: Tabla de Ejecuciones – Entidades

## Tabla 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:Tabla de Asignaciones – Entidades

## Tabla 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: Tabla de 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

# RESULTADOS

Los resultados obtenidos con el desarrollo de este proyecto consisten principalmente en la creación de los diferentes endpoints (véase anexo 1: Glosario) que servirán como puntos de control de las acciones que realizara el usuario de acuerdo con un determinado evento (listar, ver, crear, modificar, eliminar, captura de contenido, creación de reportes, autorización y autentificación) según el recurso seleccionado a través de la URL al que corresponde el endpoint y su respectivo método HTTP.

Las URLs listadas en la API corresponden a los puntos de acceso, es decir los endpoints, que llevan a cabo las funcionalidades planificadas para el proyecto. Se han dividido en 5 tipos:

* Los endpoints del tipo de usuario manager.
* Los endpoints del tipo de usuario developer.
* Los endpoints del tipo de usuario trainer.
* Los endpoints del tipo de usuario trainee.
* Los endpoints del tipo generales.

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.

Cada endpoint tiene asociado los parámetros de autorización que son requeridos dependiendo del usuario, para este proyecto se ha utilizado el tipo de *token*, Bearer Token y los parámetros de un *form request* que corresponden al cuerpo junto con los parámetros que se mandan por URL.

Para los propósitos definidos en este proyecto, se manejaron únicamente parámetros de tipo id, que corresponden a tipos de datos *integer*; estos dependen del tipo de recurso que le precede para representar al índice que hace referencia el endpoint en su parámetro.

Al ser una *API* se da por sentado que en cada petición se hace un llamado exclusivo de un tipo de dato JSON por la petición del header:

*Accept: application/json*

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 (manager) | 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.

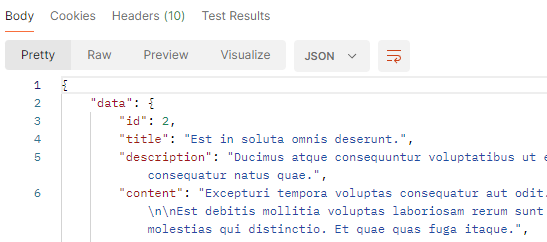


Ilustración 4: Respuesta del endpoint /api/v1/programs/{id?} para manager

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| POST | Token (manager) | - | 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.

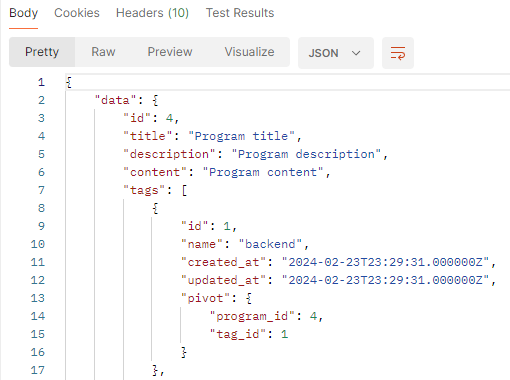


Ilustración 5: Respuesta del endpoint /api/v1/programs para manager

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| POST | Token (manager) | - | 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.

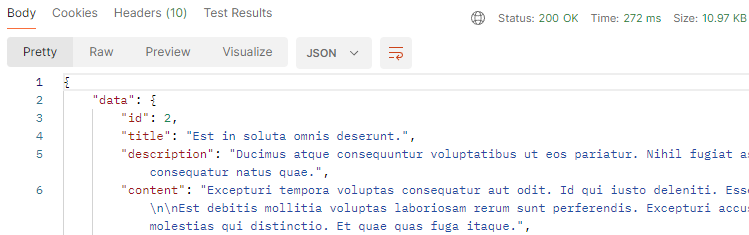


Ilustración 6: Respuesta del endpoint /api/v1/executions para manager

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| GET | Token (manager) | 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 7: 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 (manager) | 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.

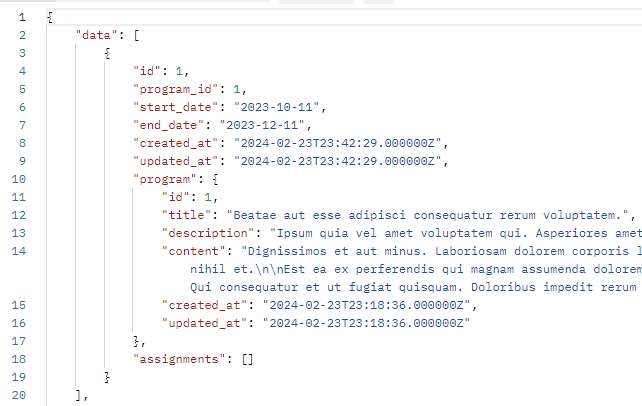


Ilustración 8: Respuesta del endpoint /api/v1/executions/{id?} para manager

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| POST | Token (manager) | - | 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 9: 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 (manager) | - | 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 más 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 más elementos.



Ilustración 10: 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 (manager) | - | 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.

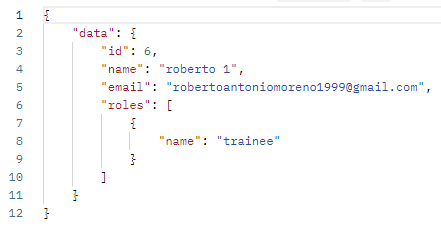


Ilustración 11: Respuesta del endpoint /api/v1/users/create-trainee-account para manager

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| GET | Token (manager) | 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 módulos, 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 12: 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 (manager) | 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 13: 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 (manager) | - | - | Lista de Usuarios (JSON) |

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

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



Ilustración 14: Respuesta del endpoint /api/v1/users para manager

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

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

Tabla 36: Obtener detalle de un usuario (manager) - Endpoints

Acción: Obtiene el detalle del usuario seleccionado.



Ilustración 15: Respuesta del endpoint /api/v1/users/{id} para manager

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

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

Tabla 37: Editar nombre de usuario (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 más 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.

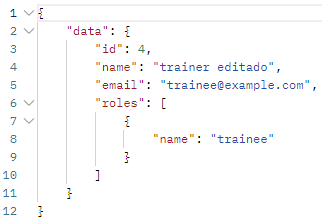


Ilustración 16: Respuesta del endpoint /api/v1/users/{id} (PATCH)

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

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

Tabla 38: Eliminar usuario (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.

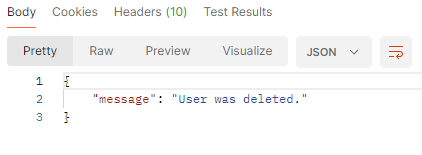


Ilustración 17: Respuesta del endpoint /api/v1/users/{id} (DELETE)

# Endpoints disponibles para el tipo de usuario developer

## Obtener lista de programas a los que pertenece: */api/v1/programs*

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

Tabla 39: Lista de programas a los que pertenece el developer - Endpoints

Acción: Lista todos los programas a los que ha sido asignado el developer que solicita la petición al endpoint.



Ilustración 18: Respuesta del endpoint /api/v1/programs para developer

## Obtener detalle de un programa al que pertenece: */api/v1/programs/{id}*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| GET | Token (developer) | program:id (int) | - | Programa (JSON) |

Tabla 40: Obtiene el detalle de un programa al que es encargado el developer - Endpoints

Acción: Obtiene el detalle de un programa junto con sus dependencias seleccionando el id al que pertenece el programa solicitado según el developer que hace la petición al servidor.

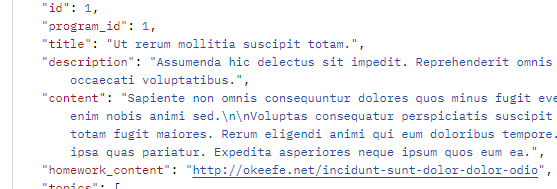


Ilustración 19: Respuesta del endpoint /api/v1/programs/{id} para developer

## Crear un módulo: */api/v1/modules*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| POST | Token (developer) | - | program\_id (int), title (string), description (string), content (string), homework\_content (string) | Módulo (JSON) |

Tabla 41: Crea un módulo - Endpoints

Acción: Crea un módulo con los parámetros del cuerpo de la petición.

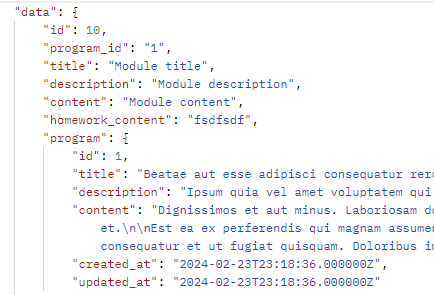


Ilustración 20: Respuesta del endpoint /api/v1/modules (POST)

## Edita un módulo: */api/v1/modules/{id}*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| PATCH | Token (developer) | module:id (int) | title (string), description (string), content (string), program\_id (int) | Módulo (JSON) |

Tabla 42: Edita un módulo - Endpoints

Acción: Edita un módulo con los parámetros del cuerpo de la petición según el id por URL referente al índice del recurso.



Ilustración 21: Respuesta del endpoint /api/v1/modules/{id} (PATCH)

## Elimina un módulo: */api/v1/modules/{id}*

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

Tabla 43: Elimina un módulo - Endpoints

Acción: Elimina un módulo por el parámetro id de la URL.

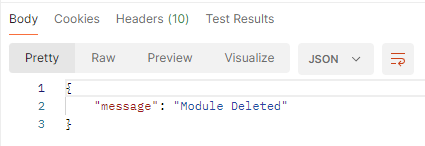


Ilustración 22: Respuesta del endpoint /api/v1/modules/{id} (DELETE)

## Crea un tema: */api/v1/topics*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| POST | Token (developer) | - | module\_id (int), title (string), description (string), content (string), index (int) | Tema (JSON) |

Tabla 44: Crea un tema - Endpoints

Acción: Crea un tema con los parámetros del cuerpo de la petición.

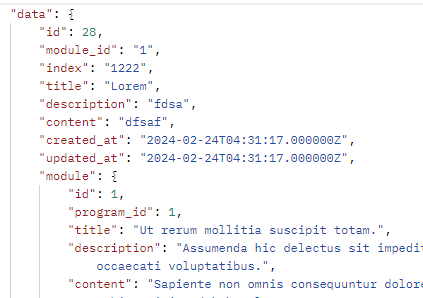


Ilustración 23: Respuesta del endpoint /api/v1/topics (POST)

## Edita un tema: */api/v1/topics/{id}*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| PATCH | Token (developer) | topic:id (int) | title (string), description (string), content (string), module\_id (int), index (int) | Tema (JSON) |

Tabla 45: Edita un tema - Endpoints

Acción: Edita un tema con los parámetros del cuerpo de la petición según el id por URL referente al índice del recurso además de los parámetros adicionales por *form request* que se requieren editar. Es necesario recalcar que aparte de ser un usuario de tipo developer el usuario debe ser previamente asignado al programa en cuestión para tener la autorización de editarlo.

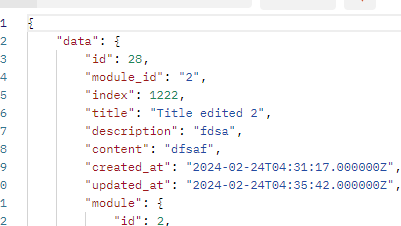


Ilustración 24: Respuesta del endpoint /api/v1/topics/{id} (PATCH)

## Elimina un tema: */api/v1/topics/{id}*

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

Tabla 46: Elimina un tema - Endpoints

Acción: Elimina un tema por el parámetro id de la URL.

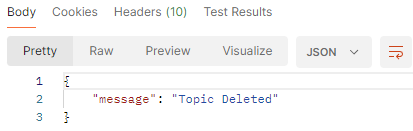


Ilustración 25: Respuesta del endpoint /api/v1/topics/{id} (DELETE)

## Crear una pregunta: */api/v1/questions*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| POST | Token (developer) | - | topic\_id (int), question (string), grade\_definitions (string) | Pregunta (JSON) |

Tabla 47: Crea una pregunta - Endpoints

Acción: Crea una pregunta con los parámetros del cuerpo de la petición.

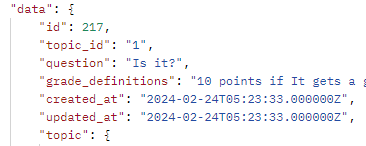


Ilustración 26: Respuesta del endpoint /api/v1/questions (POST)

## Editar una pregunta: */api/v1/questions/{id}*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| PATCH | Token (developer) | question:id (int) | Topic\_id (int), question (string), grade\_definitions (string) | Pregunta (JSON) |

Tabla 48: Edita una pregunta - Endpoints

Acción: Edita una pregunta con los parámetros del cuerpo de la petición según el id por URL referente al índice del recurso.

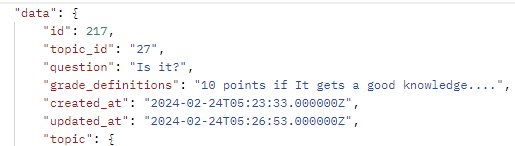


Ilustración 27: Respuesta del endpoint /api/v1/questions/{id} (PATCH)

## Eliminar una pregunta: */api/v1/questions/{id}*

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

Tabla 49: Elimina una pregunta - Endpoints

Acción: Elimina una pregunta por el parámetro id de la URL.

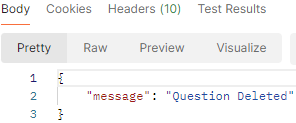


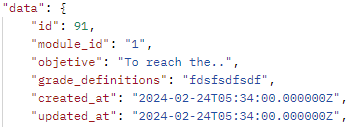
Ilustración 28: Respuesta del endpoint /api/v1/questions/{id} (DELETE)

## Crea un criterio de evaluación: */api/v1/evaluations*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| POST | Token (developer) | - | module\_id (int), objetive (string), grade\_definitions (string) | Criterio de evaluación (JSON) |

Tabla 50: Crea un criterio de evaluación - Endpoints

Acción: Crea un criterio de evaluación con los parámetros del cuerpo de la petición.



## Editar un criterio de evaluación: */api/v1/evaluations/{id}*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| PATCH | Token (developers) | evaluation\_criterias:id (int) | module\_id (int), objetive (string), grade\_definitions (string) | Criterio de evaluación (JSON) |

Tabla 51: Edita un criterio de evaluación - Endpoints

Acción: Edita un criterio de evaluación con los parámetros del cuerpo de la petición según el id por URL referente al índice del recurso.



Ilustración 29: Respuesta del endpoint /api/v1/evaluations/{id} (PATCH)

## Eliminar un criterio de evaluación: */api/v1/evaluations/{id}*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| DELETE | Token (developer) | evaluation\_criterias:id (int) | - | Mensaje de éxito (JSON) |

Tabla 52: Elimina un criterio de evaluación - Endpoints

Acción: Elimina un criterio de evaluación por el parámetro id de la URL.

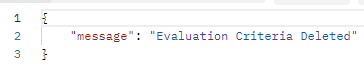


Ilustración 30: Respuesta del endpoint /api/v1/evaluations/{id} (DELETE)

# Endpoints disponibles para el tipo de usuario trainer

## Obtener lista de ejecuciones a las que pertenece: */api/v1/executions*

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

Tabla 53: Obtener lista de ejecuciones (trainer) - Endpoints

Acción: Lista todas las ejecuciones a las que pertenece el trainer.



Ilustración 31: Respuesta del endpoint /api/v1/executions para trainer

## Obtener detalle de ejecución al que pertenece: */api/v1/executions/{id}*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| GET | Token (trainer) | executions:id (int) | - | Ejecución (JSON) |

Tabla 54: Obtener detalle de ejecución (trainer) - Endpoints

Acción: Obtiene el detalle de la ejecución a la que pertenece el usuario por el parámetro id a través de la URL.



Ilustración 32: Respuesta del endpoint /api/v1/executions/{id} para trainer

## Inicia la entrevista: */api/v1/assignments/{id}/interview-start*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| POST | Token (trainer) | assignment:id (int) | - | Assignment (JSON) |

Tabla 55: Iniciar entrevista (trainer) - Endpoints

Acción: Inicia la entrevista a su trainee, paralelamente hace la captura del contenido de la entrevista a la tabla de assignments para su posterior análisis.



Ilustración 33: Respuesta del denpoint /api/v1/assignments/{id}/interview-start para trainer

## Guardar respuestas de entrevista: */api/v1/assignments/save-question*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| PUT | Token (trainer) | - | assignment\_id (int), gradables (array[gradables[\*][comments] (string), gradables[\*][grade] (float), gradables[\*][gradable\_id] (int)]) | Detalle de nota de la evaluación (JSON) |

Tabla 56: Guarda la información obtenida de la entrevista (trainer) - Endpoints

Acción: Registra las respuestas y comentarios adicionales por cada pregunta a la que el trainer le halla preguntado durante la entrevista a su trainee.

El cuerpo recibe las preguntas concatenadas dentro de un array bidimensional tanto para hacer referencia a las preguntas como a sus comentarios y este campo lo recibe la *API* como un solo input de tipo array múltiple. Esto se hizo de esta manera para guardar toda la información de la entrevista de una sola vez y no pregunta por pregunta.

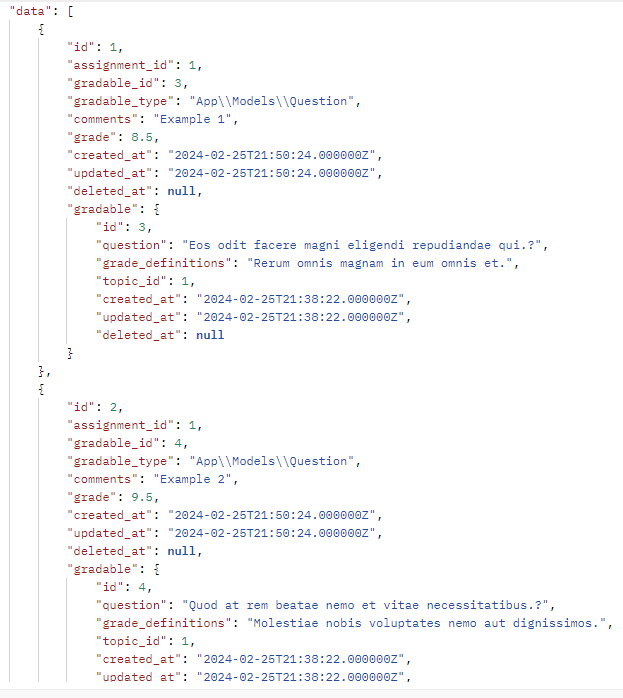


Ilustración 34: Respuesta del endpoint /api/v1/assignments/save-question para trainer

## Finaliza la entrevista: */api/v1/assignments/{id}/interview-finish*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| POST | Token (trainer) | assignment:id (int) | - | Hora inicio, hora fin y promedio de la entrevista (JSON) |

Tabla 57: Finaliza y promedia la entrevista (trainer) - Endpoints

Acción: Finaliza la entrevista y promedia la nota de la respuesta de todas las preguntas según los criterios de evaluación del módulo en ejecución definidos por el developer.



Ilustración 35: Respuesta del endpoint /api/v1/assignments/{id}/interview-finish para trainer

## Inicia evaluación de código: */api/v1/assignments/{id}/homework-start*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| POST | Token (trainer) | assignment:id (int) | - | Assignment (JSON) |

Tabla 58: Iniciar evaluación de código (trainer) - Endpoints

Acción: Inicia la evaluación de código a su trainee, paralelamente hace la captura del contenido de los criterios de evaluación a la tabla de assignments para su posterior análisis.



Ilustración 36: Respuesta del endpoint /api/v1/assignments/{id}/homework-start para trainer

## Guardar evaluación de código: */api/v1/assignments/save-evaluation-criteria*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| PUT | Token (trainer) | - | assignment\_id (int), gradables (array[gradables[\*][comments] (string), gradables[\*][grade] (float), gradables[\*][gradable\_id] (int)]) | Detalle de nota de la evaluación (JSON) |

Tabla 59: Guarda la información de la evaluación de codigo (trainer) - Endpoints

Acción: Registra la asignación de nota y comentarios adicionales por cada criterio que el trainer haya encontrado en el código de su trainee, el cuerpo recibe las notas concatenadas dentro de un array bidimensional tanto para hacer referencia a los criterios como a sus comentarios y este campo lo recibe la API como un solo input de tipo array múltiple. Esto se hizo de esta manera para guardar toda la información de la evaluación de código de una sola vez y no criterio por criterio.



Ilustración 37: Respuesta del endpoint /api/v1/assignments/save-evaluation-criteria para trainer

## Finaliza la evaluación de código: */api/v1/assignments/{id}/homework-finish*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| POST | Token | assignment:id (int) | - | Hora inicio, hora fin y promedio de la evaluación de código (JSON) |

Tabla 60: Finaliza y promedia la evaluación de código (trainer) - Endpoints

Acción: Finaliza la evaluación de código y promedia la nota de todos los criterios de evaluación definidos por el developer.



Ilustración 38: Respuesta del endpoint /api/v1/assignments/{id}/homework-finish para trainer

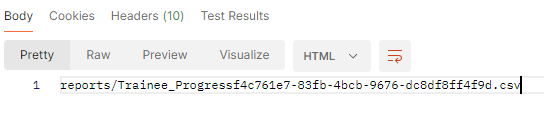
## Obtener reporte de trainee: */api/v1/report/{id}*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| GET | Token (trainer) | user:id (int) | - | Información del enrolamiento del trainee en la ejecución (.csv) |

Tabla 61: Crea el reporte de la ejecución (trainer) - Endpoints

Acción: Crea un reporte de tipo archivo de hojas de cálculo (.csv) del usuario trainee seleccionado por el trainer. La respuesta directa del endpoint será mostrar el nombre del archivo creado.

Ilustración 39: Crea el reporte dentro de la carpeta publica del proyecto



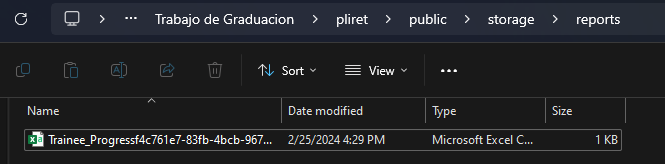


Ilustración 40: Respuesta del endpoint /api/v1/report/{id} para trainer

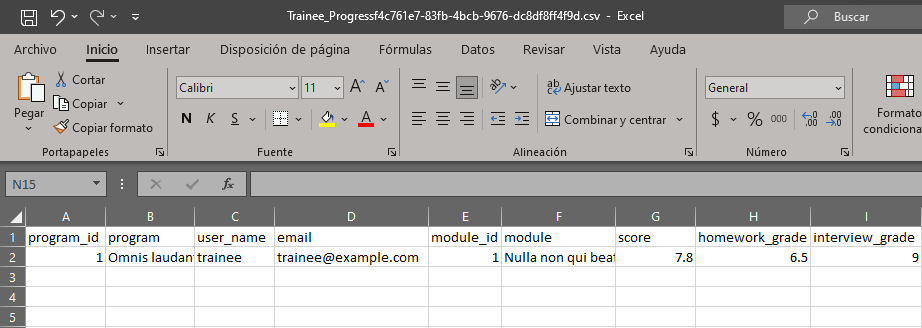


Ilustración 41: Reporte csv del promedio final de trainee

# Endpoints disponibles para el tipo de usuario trainer

## Obtener listado de ejecuciones a las que pertenece: */api/v1/executions*

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

Tabla 62: Lista las ejecuciones para el trainee - Endpoints

Acción: Lista las ejecuciones a las que se ha enrolado el usuario trainee por medio del manager según los programas de evaluación de contenidos al que se iba a someter que hace la petición a la API.

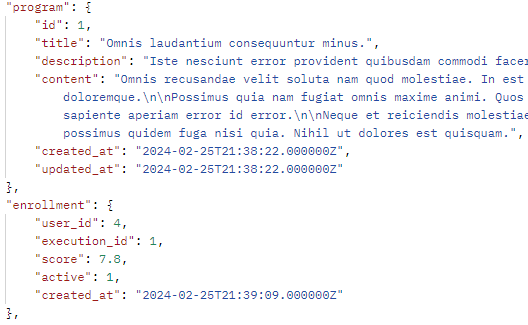


Ilustración 42: Respuesta del endpoint /api/v1/executions para trainee

## Obtener detalle de ejecución a la que pertenece: */api/v1/executions/{id}*

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

Tabla 63: Lista las ejecuciones a las que pertenece el trainee - Endpoints

Acción: Obtiene toda la información de la ejecución seleccionada por medio del parámetro id por URL a la que el trainee pertenece.



Ilustración 43: Respuesta del endpoint /api/v1/executions/{id} para trainee

## Carga el código a evaluar: */api/v1/executions*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Método** | **Autorización** | **Parámetros** | **Cuerpo** | **Respuesta** |
| GET | Token (trainer) | - | homework-solution (file/string) | Assignment (JSON) |

Tabla 64: Carga código a evaluar - Endpoints

Acción: Carga el código a evaluar por el trainer, esto mediante la subida de un archivo o por el texto del código mismo.



Ilustración 44: Respuesta del endpoint /api/v1/executions para trainee

# Endpoints disponibles para todos los usuarios

## Autentificación de usuario por plataforma interna: */api/v1/login*

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

Tabla 65: Autentificación interna de usuario - Endpoints

Acción: Envía las credenciales introducidas por el cliente HTTP para autentificar y obtener las credenciales de acceso (token) según su tipo de usuario.



Ilustración 45: Respuesta del endpoint /api/v1/login para todos los usuarios

## Autentificación de usuario por Google OAuth 2.0: */api/v1/social/google/login*

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

Tabla 66: Autentificacion por google - Endpoints

Acción: Accede a los servicios de autentificación de Google como un servicio externo que pide la autorización del usuario que busca autentificarse para autorizar a la API a obtener la información básica del usuario, que en este caso únicamente corresponde al correo electrónico.

Seguidamente toma el valor del correo electrónico extraído y los compara con los registros de la base de datos si existe un usuario registrado con el correo electrónico seleccionado.

Si es el caso la *API* da por autorizado al usuario y crea un nuevo token de acceso para luego ser concatenado dentro de la información del usuario a mostrar en el cual contiene la información del usuario encriptada junto con los permisos que este tiene, es decir el tipo de usuario, lo que autentifica a la vez que autoriza el usuario para hacer uso de los endpoints de la *API* a través del servicio de Google login.

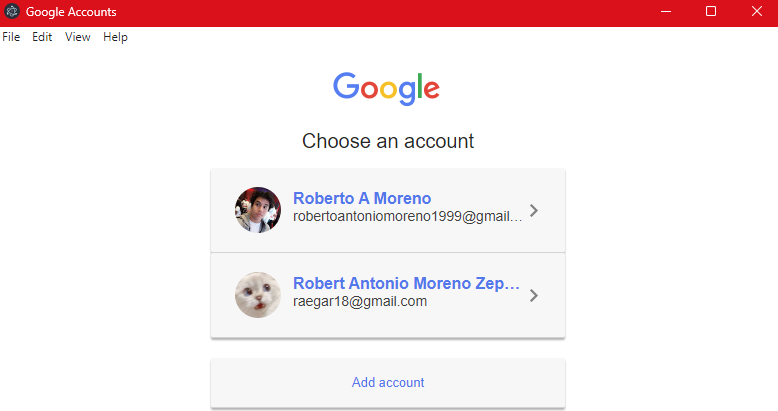


Ilustración 46: Si las credenciales son correctas el usuario es autentificado



Ilustración 47: Respuesta del endpoint /api/v1/social/google/login donde procedemos a autentificarnos en Google

# ELEMENTOS DE SOPORTE

# Investigación y planificación:

* 1. Entrevista con el gerente de tecnología:

Se realizó una exhaustiva entrevista con el gerente de tecnología para comprender las necesidades y objetivos de la *API*. Se discutieron las funcionalidades deseadas, la arquitectura general y las tecnologías a utilizar.

* 1. Análisis de requisitos:

Se realizó un análisis detallado de los requisitos técnicos y funcionales de la API. Se definieron los tipos de usuarios, las operaciones que se podían realizar, los datos que se iban a almacenar y los formatos de entrada y salida.

* 1. Diseño de la arquitectura:

Se diseñó la arquitectura de la *API*, definiendo los diferentes módulos, las relaciones entre ellos y los protocolos de comunicación. Se seleccionaron las tecnologías más adecuadas para cada componente.

* 1. Plan de trabajo:

Se elaboró un plan de trabajo con un cronograma detallado y tareas específicas para cada fase del desarrollo.

# Desarrollo:

* 1. Desarrollo de la *API*:

Se desarrollo la *API* utilizando el framework Laravel en PHP. Se siguió un enfoque de desarrollo ágil, con entregas incrementales y pruebas constantes.

* 1. Pruebas unitarias:

Se implementaron pruebas unitarias para garantizar la calidad del código. Se verificó el correcto funcionamiento de cada módulo de la API y se corrigieron los errores encontrados.

* 1. Control de versiones:

Se utilizó Git y GitHub para la gestión del versionado del código. Se realizó un seguimiento de los cambios realizados y se crearon ramas para las diferentes versiones de la API.

* 1. Base de datos:

Se utilizó el gestor de base de datos MySQL para almacenar la información. Se diseñó el esquema de la base de datos y se optimizó el rendimiento para las consultas.

# Pruebas y entrega:

* 1. Pruebas exhaustivas:

Se realizaron pruebas exhaustivas de la *API* para asegurar su correcto funcionamiento. Se probaron diferentes escenarios de uso y se verificó la seguridad de la *API*.

* 1. Manual de usuario:

Se elaboró un manual de usuario con instrucciones detalladas para la utilización de la API. Se incluyeron ejemplos de código y casos de uso para facilitar la comprensión.

* 1. Entrega del Proyecto:

Se entregó el proyecto al cliente mediante un enlace al repositorio de GitHub y el manual de usuario.

# Impacto:

* 1. Creación, gestión y evaluación de pruebas de conocimiento:

La API permite la creación, gestión y evaluación de pruebas de conocimiento de forma automatizada en forma de programas de contenido. Los managers pueden crear pruebas con diferentes tipos de temas, preguntas y criterios de evaluación. Los empleados pueden completar las pruebas y obtener sus resultados al finalizar el programa.

* 1. Generación de informes:

Se pueden generar informes con los resultados para cada empleado y para la organización en general. Estos informes permiten a los administradores evaluar el rendimiento de los empleados y tomar decisiones sobre la formación y el desarrollo profesional.

* 1. Mejora de la eficiencia y la eficacia:

La API ha contribuido a mejorar y reorganizar todo el planteamiento de seguimiento de procesos de evaluación para la empresa Pliret. Porque además de uso interno el resultado de los programas de evaluación irá anexados al historial de los empleados que manejan para tomar una decisión basada en datos sobre el *outsourcing* al que se dedican.

# Tecnologías utilizadas:

* Lenguaje de programación: PHP 8.1
* Framework: Laravel 9
* Pruebas unitarias: PHPUnit 9.6
* Control de versiones: Git, GitHub
* Base de datos: MySQL 8

# Herramientas utilizadas:

* Entorno de desarrollo: Visual Studio Code, Postman
* Administradores de bases de datos: MySQL, Laragon

# Pruebas unitarias

Las pruebas unitarias son un tipo de prueba de software que evalúa el correcto comportamiento de unidades individuales de código, como funciones, métodos o clases. En este caso se ha enfocado a funciones para, valga la redundancia, poner a prueba las funcionalidades de la API.

Para este proyecto se han creado 158 pruebas las cuales son las siguientes.

|  |  |
| --- | --- |
| Número | Función a poner a prueba |
| 1 | Asignar un módulo a un usuario trainee |
| 2 | Iniciar evaluación |
| 3 | Finalizar evaluación |
| 4 | Iniciar revisión de código |
| 5 | Finalizar evaluación de código |
| 6 | Cargar código a evaluar |
| 7 | Crear un empleado con el rol “manager” |
| 8 | Crear un empleado con el rol “developer” |
| 9 | Crear un empleado con el rol “trainer” |
| 10 | Empleado creado tiene la autorización correcta para endpoints de invitado |
| 11 | Empleado creado tiene la autorización correcta para endpoints de manager |
| 12 | Empleado creado tiene la autorización correcta para endpoints de developer |
| 13 | Empleado creado tiene la autorización correcta para endpoints de trainer |
| 14 | Empleado creado tiene la autorización correcta para endpoints de trainee |
| 15 | Crea un usuario trainee |
| 16 | Trainee creado tiene la autorización correcta para endpoints de invitado |
| 17 | Trainee creado tiene la autorización correcta para endpoints de manager |
| 18 | Trainee creado tiene la autorización correcta para endpoints de developer |
| 19 | Trainee creado tiene la autorización correcta para endpoints de trainer |
| 20 | Trainee creado tiene la autorización correcta para endpoints de trainee |
| 21 | Listar criterios de evaluación |
| 22 | Mostrar un criterio de evaluación |
| 23 | Crear un criterio de evaluación |
| 24 | Editar un criterio de evaluación |
| 25 | Eliminar un criterio de evaluación |
| 26 | Asignar trainer a ejecución |
| 27 | Enrolar trainee a ejecución |
| 28 | Listar ejecuciones |
| 29 | Trainer puede ver solo las ejecuciones en las que participa |
| 30 | Trainee puede ver solo las ejecuciones en las que participa |
| 31 | Mostrar detalle de ejecución |
| 32 | No muestra al trainer ejecuciones externas |
| 33 | No muestra al trainee ejecuciones externas |
| 34 | No muestra al trainee ejecuciones finalizadas |
| 35 | Muestra al trainee la puntuación de su ejecución |
| 36 | Trainer puede ver ejecución y notas de su trainee |
| 37 | Crear ejecución |
| 38 | Editar ejecución |
| 39 | Eliminar ejecución |
| 40 | CRUD de ejecución tiene la autorización correcta para la habilidad “guest\_index” |
| 41 | CRUD de ejecución tiene la autorización correcta para la habilidad “manager\_index” |
| 42 | CRUD de ejecución tiene la autorización correcta para la habilidad “developer\_index” |
| 43 | CRUD de ejecución tiene la autorización correcta para la habilidad “trainer\_index” |
| 44 | CRUD de ejecución tiene la autorización correcta para la habilidad “trainee\_index” |
| 45 | CRUD de ejecución tiene la autorización correcta para la habilidad “guest\_show” |
| 46 | CRUD de ejecución tiene la autorización correcta para la habilidad “manager\_show” |
| 47 | CRUD de ejecución tiene la autorización correcta para la habilidad “developer\_show” |
| 48 | CRUD de ejecución tiene la autorización correcta para la habilidad “trainer\_show” |
| 49 | CRUD de ejecución tiene la autorización correcta para la habilidad “trainee\_show” |
| 50 | CRUD de ejecución tiene la autorización correcta para la habilidad “guest\_store” |
| 51 | CRUD de ejecución tiene la autorización correcta para la habilidad “manager\_store” |
| 52 | CRUD de ejecución tiene la autorización correcta para la habilidad “developer\_store” |
| 53 | CRUD de ejecución tiene la autorización correcta para la habilidad “trainer\_store” |
| 54 | CRUD de ejecución tiene la autorización correcta para la habilidad “trainee\_store” |
| 55 | CRUD de ejecución tiene la autorización correcta para la habilidad “guest\_update” |
| 56 | CRUD de ejecución tiene la autorización correcta para la habilidad “manager\_update” |
| 57 | CRUD de ejecución tiene la autorización correcta para la habilidad “developer\_update” |
| 58 | CRUD de ejecución tiene la autorización correcta para la habilidad “trainer\_update” |
| 59 | CRUD de ejecución tiene la autorización correcta para la habilidad “trainee\_update” |
| 60 | CRUD de ejecución tiene la autorización correcta para la habilidad “guest\_destroy” |
| 61 | CRUD de ejecución tiene la autorización correcta para la habilidad “manager\_update” |
| 62 | CRUD de ejecución tiene la autorización correcta para la habilidad “developer\_update” |
| 63 | CRUD de ejecución tiene la autorización correcta para la habilidad “trainer\_update” |
| 64 | CRUD de ejecución tiene la autorización correcta para la habilidad “trainee\_update” |
| 65 | Finalizar ejecución |
| 66 | Dispara evento por finalizar ejecución |
| 67 | Solo manager puede terminar una ejecución con habilidad “guest” |
| 68 | Solo manager puede terminar una ejecución con habilidad “manager” |
| 69 | Solo manager puede terminar una ejecución con habilidad “developer” |
| 70 | Solo manager puede terminar una ejecución con habilidad “trainer” |
| 71 | Solo manager puede terminar una ejecución con habilidad “trainee” |
| 72 | Listar módulos |
| 73 | Mostrar detalle de módulo |
| 74 | Crear módulo |
| 75 | Editar módulo |
| 76 | Eliminar módulo |
| 77 | Asignar developer a programa |
| 78 | Listar programas |
| 79 | Developer puede ver solo programas en los que participa |
| 80 | Mostrar detalle de programa |
| 81 | No muestra a developer programa donde no participa |
| 82 | Developer puede ver a los developers de sus programas |
| 83 | Crear programa |
| 84 | Editar programa |
| 85 | Eliminar programa |
| 86 | CRUD de programa tiene la autorización correcta para la habilidad “guest\_index” |
| 87 | CRUD de programa tiene la autorización correcta para la habilidad “manager\_index” |
| 88 | CRUD de programa tiene la autorización correcta para la habilidad “developer\_index” |
| 89 | CRUD de programa tiene la autorización correcta para la habilidad “trainer\_index” |
| 90 | CRUD de programa tiene la autorización correcta para la habilidad “trainee\_index” |
| 91 | CRUD de programa tiene la autorización correcta para la habilidad “guest\_show” |
| 92 | CRUD de programa tiene la autorización correcta para la habilidad “manager\_show” |
| 93 | CRUD de programa tiene la autorización correcta para la habilidad “developer\_show” |
| 94 | CRUD de programa tiene la autorización correcta para la habilidad “trainer\_show” |
| 95 | CRUD de programa tiene la autorización correcta para la habilidad “trainee\_show” |
| 96 | CRUD de programa tiene la autorización correcta para la habilidad “guest\_store” |
| 97 | CRUD de programa tiene la autorización correcta para la habilidad “manager\_store” |
| 98 | CRUD de programa tiene la autorización correcta para la habilidad “developer\_store” |
| 99 | CRUD de programa tiene la autorización correcta para la habilidad “trainer\_store” |
| 100 | CRUD de programa tiene la autorización correcta para la habilidad “trainee\_store” |
| 101 | CRUD de programa tiene la autorización correcta para la habilidad “guest\_update” |
| 102 | CRUD de programa tiene la autorización correcta para la habilidad “manager\_update” |
| 103 | CRUD de programa tiene la autorización correcta para la habilidad “developer\_update” |
| 104 | CRUD de programa tiene la autorización correcta para la habilidad “trainer\_update” |
| 105 | CRUD de programa tiene la autorización correcta para la habilidad “trainee\_update” |
| 106 | CRUD de programa tiene la autorización correcta para la habilidad “guest\_destroy” |
| 107 | CRUD de programa tiene la autorización correcta para la habilidad “manager\_destroy” |
| 108 | CRUD de programa tiene la autorización correcta para la habilidad “developer\_destroy” |
| 109 | CRUD de programa tiene la autorización correcta para la habilidad “trainer\_destroy” |
| 110 | CRUD de programa tiene la autorización correcta para la habilidad “trainee\_destroy” |
| 111 | Listar preguntas |
| 112 | Mostrar detalle de pregunta |
| 113 | Crear pregunta |
| 114 | Editar pregunta |
| 115 | Eliminar pregunta |
| 116 | Crear nuevo criterio por pregunta |
| 117 | Crear nota por pregunta |
| 118 | Se dispara el evento de captura de código |
| 119 | Capturar contenido de código |
| 120 | Se dispara el evento de captura para preguntas |
| 121 | Tomar captura de preguntas del programa |
| 122 | Se dispara el evento de captura para contenido |
| 123 | Tomar captura de contenido del programa |
| 124 | Listar temas |
| 125 | Mostrar detalle de tema |
| 126 | Crear tema |
| 127 | Editar tema |
| 128 | Eliminar tema |
| 129 | Se dispara el evento por finalizar evaluación de código para asignments |
| 130 | Actualiza promedio de evaluación de código |
| 131 | Se dispara el evento por finalizar entrevista para asignments |
| 132 | Actualizar promedio de entrevista |
| 133 | Se dispara el evento por finalizar entrevista para enrolments |
| 134 | Actualiza promedio final |
| 135 | Listar usuario |
| 136 | Mostrar usuario |
| 137 | Editar usuario |
| 138 | Eliminar usuario |
| 139 | CRUD de usuario tiene la autorización correcta para la habilidad “guest\_index” |
| 140 | CRUD de usuario tiene la autorización correcta para la habilidad “manager\_index” |
| 141 | CRUD de usuario tiene la autorización correcta para la habilidad “developer\_index” |
| 142 | CRUD de usuario tiene la autorización correcta para la habilidad “trainer\_index” |
| 143 | CRUD de usuario tiene la autorización correcta para la habilidad “trainee\_index” |
| 144 | CRUD de usuario tiene la autorización correcta para la habilidad “guest\_show” |
| 145 | CRUD de usuario tiene la autorización correcta para la habilidad “manager\_show” |
| 146 | CRUD de usuario tiene la autorización correcta para la habilidad “developer\_show” |
| 147 | CRUD de usuario tiene la autorización correcta para la habilidad “trainer\_show” |
| 148 | CRUD de usuario tiene la autorización correcta para la habilidad “trainee\_show” |
| 149 | CRUD de usuario tiene la autorización correcta para la habilidad “guest\_update” |
| 150 | CRUD de usuario tiene la autorización correcta para la habilidad “manager\_update” |
| 151 | CRUD de usuario tiene la autorización correcta para la habilidad “developer\_update” |
| 152 | CRUD de usuario tiene la autorización correcta para la habilidad “trainer\_update” |
| 153 | CRUD de usuario tiene la autorización correcta para la habilidad “trainee\_update” |
| 154 | CRUD de usuario tiene la autorización correcta para la habilidad “guest\_destroy” |
| 155 | CRUD de usuario tiene la autorización correcta para la habilidad “manager\_destroy” |
| 156 | CRUD de usuario tiene la autorización correcta para la habilidad “developer\_destroy” |
| 157 | CRUD de usuario tiene la autorización correcta para la habilidad “trainer\_destroy” |
| 158 | CRUD de usuario tiene la autorización correcta para la habilidad “developer\_destroy” |

Tabla 67: Unit Testing de funcionalidades de la API

A continuación se muestra una parte del código de las pruebas unitarias haciendo énfasis en los tests mas importantes que son clave del funcionamiento de la API:

Para ejecutar los test se ejecuta el comando *php artisan test* en la consola del servidor donde esta alojada la *API*.

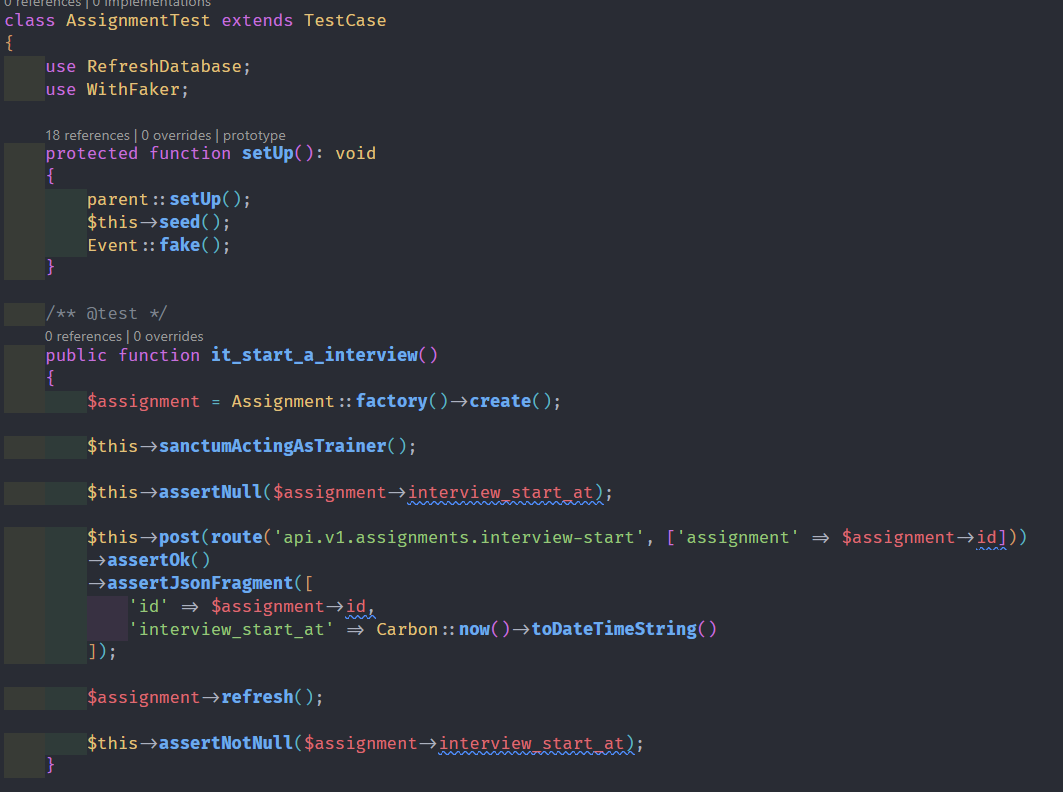


Ilustración 48: Test de iniciar entrevista



Ilustración 49: Test de terminar entrevista



Ilustración 50: Test de iniciar evaluación de código



Ilustración 51: Test de finalizar evaluación de código

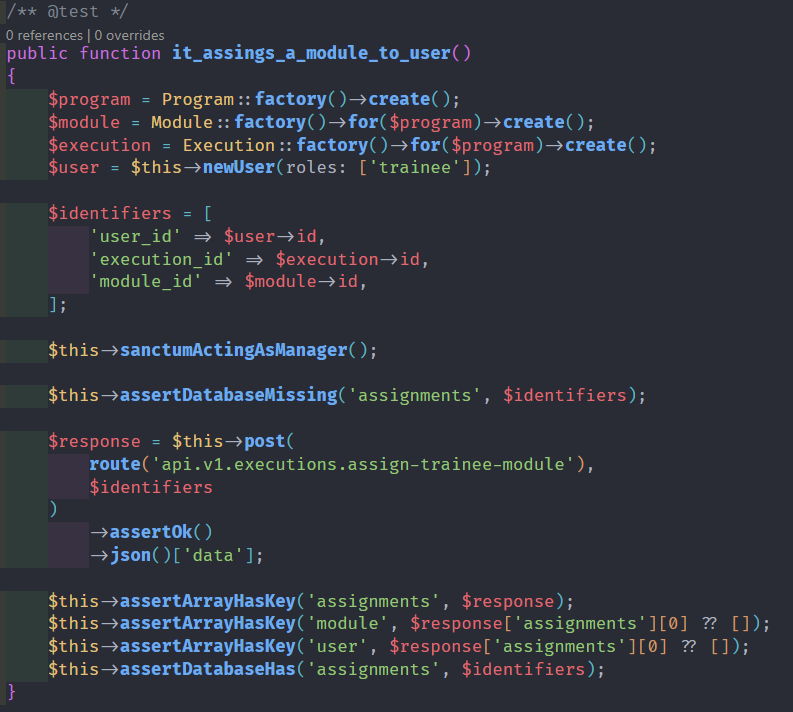


Ilustración 52: Test de asignar un módulo a un usuario



Ilustración 53: Test de verificación de habilidad de listar criterios de evaluación para developer

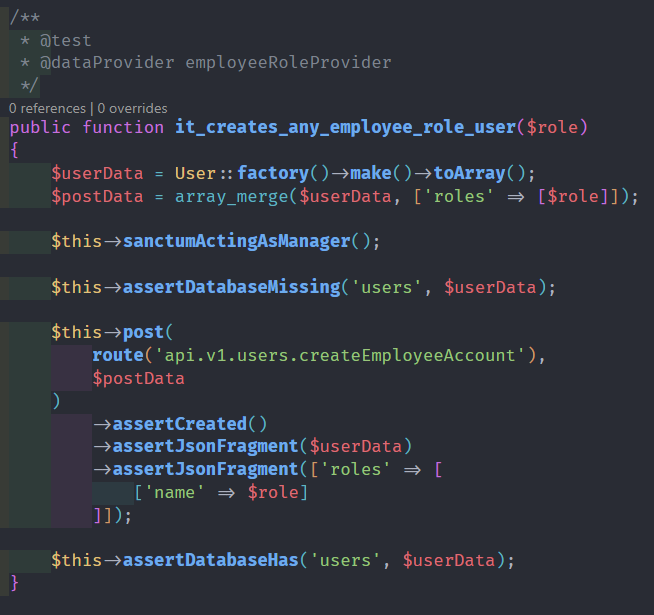


Ilustración 54: Test de creación de empleado

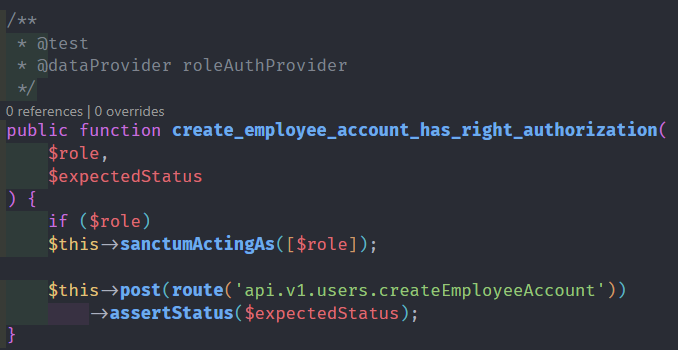


Ilustración 55: Test para verificar la autorización del empleado creado



Ilustración 56: Test de creación de trainee

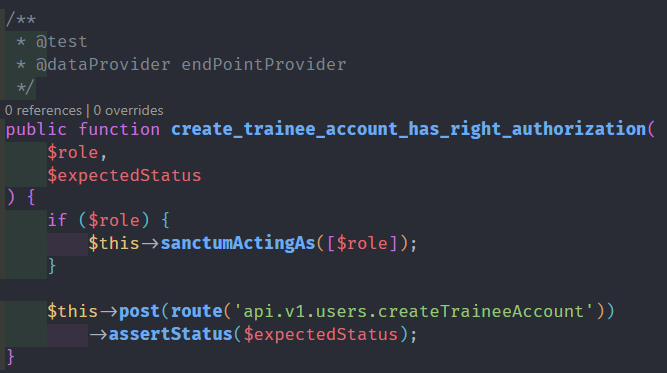


Ilustración 57: Test para verificar la autorización del trainee creado



Ilustración 58: Test para verificar la autorización de enrolar trainees a una ejecución por parte de un manager

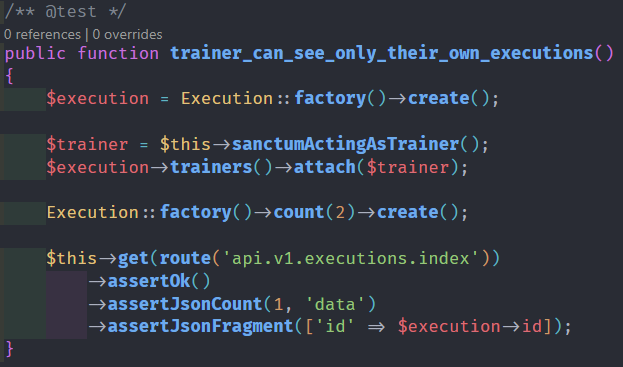


Ilustración 59: Test para verificar que un trainer puede ver solo las ejecuciones a las que ha sido asignado

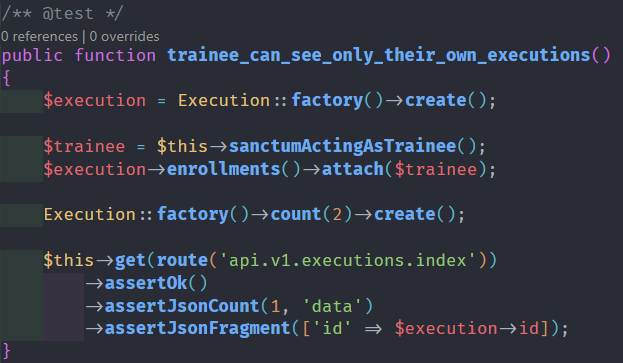


Ilustración 60: Test para verificar que un trainee puede ver solo las ejecuciones a las que ha sido enrolado

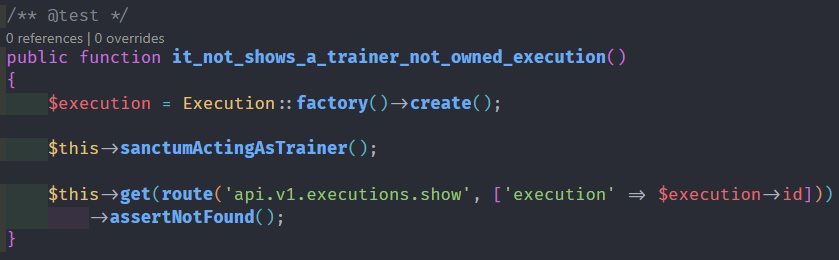


Ilustración 61: Test para verificar que un trainer no puede ver ejecuciones a las que no ha sido asignado

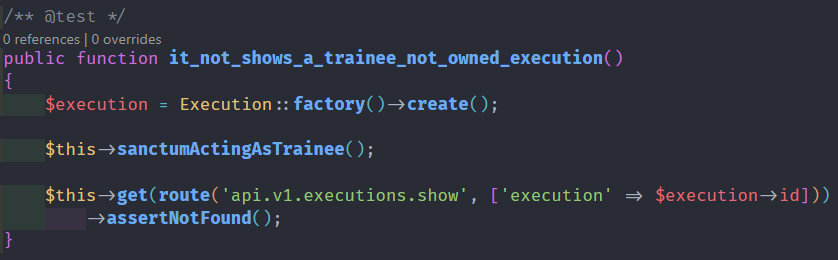


Ilustración 62: Test para verificar que un trainee no puede ver ejecuciones a las que no ha sido enrolado

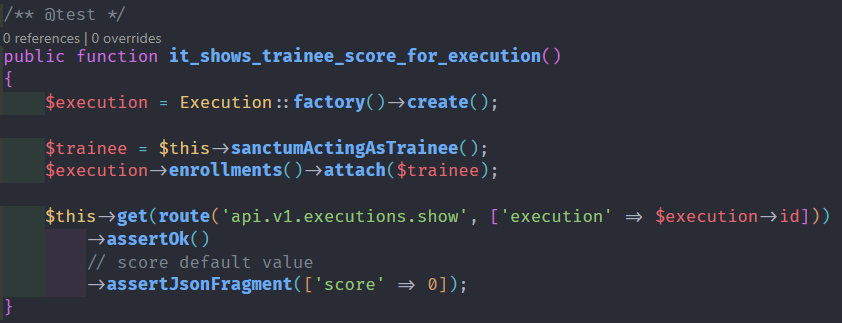


Ilustración 63: Test para verificar que un trainee puede ver su promedio en la ejecución



Ilustración 64: Test para verificar que un trainer puede ver a los trainees de sus módulos y sus notas

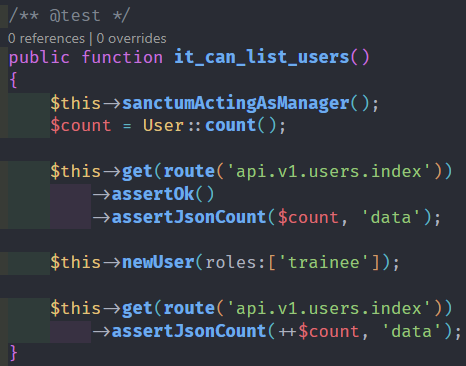


Ilustración 65: Test para verificar que se pueden listar usuarios

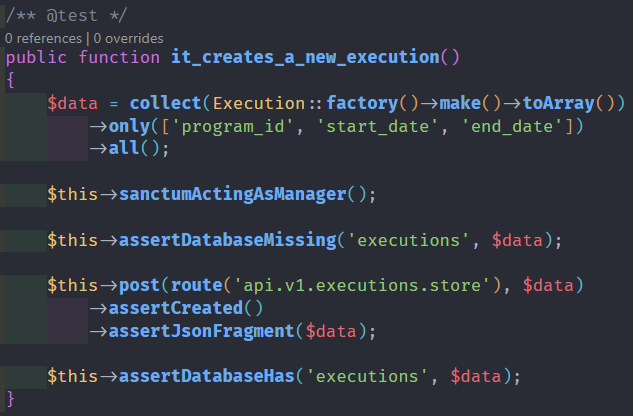


Ilustración 66: Test para verificar que se pueden crear ejecuciones basadas en programas



Ilustración 67: Test para verificar que se pueden finalizar las ejecuciones



Ilustración 68: Test para verificar que se pueden asignar developers a programas

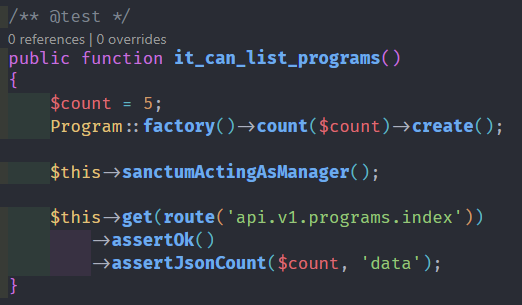


Ilustración 69: Test para verificar que un manager puede listar los programas

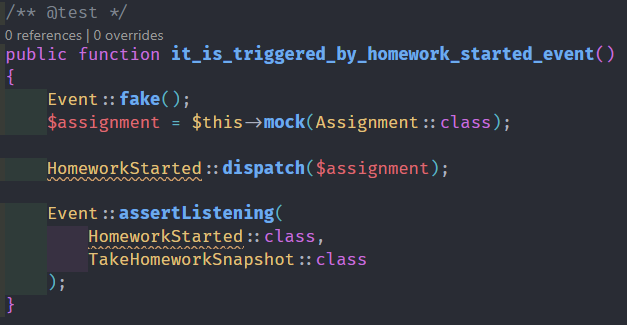


Ilustración 70: Test para verificar que el evento activado por la evaluación de código iniciada sea debidamente activado

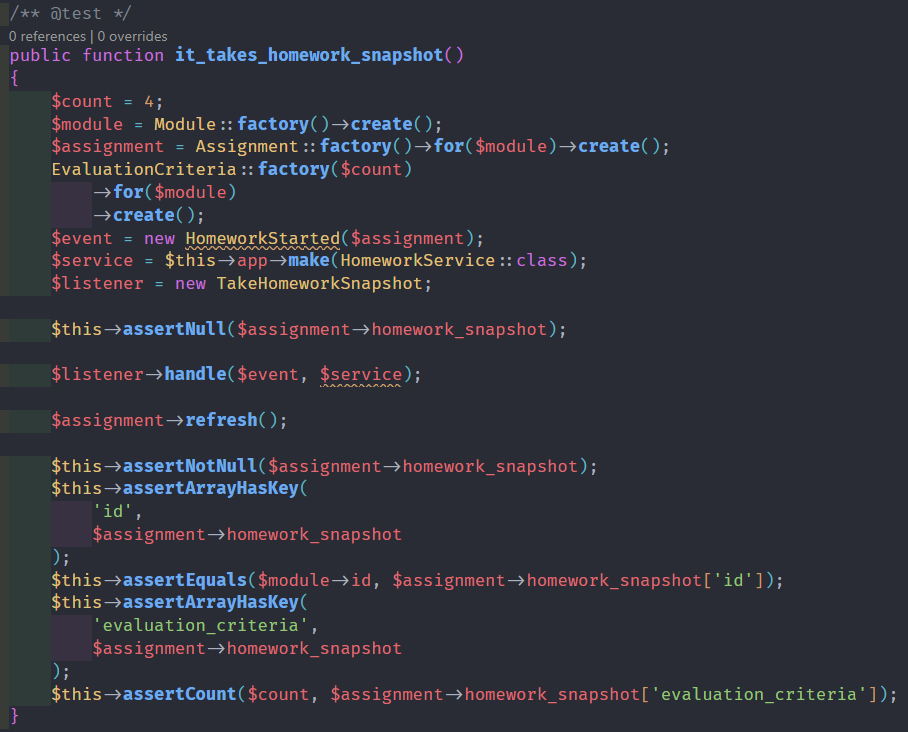


Ilustración 71: Evento para verificar que el evento de tomar captura del contenido de los criterios de evaluación sea guardado en el registro de assignments correctamente



Ilustración 72: Test para verificar que el evento activado por entrevista iniciada sea debidamente activado



Ilustración 73: Test para verificar que la puntuación del enrolamiento del trainee se promedia



Ilustración 74: Evento para verificar que el evento de tomar captura del contenido de las preguntas sea guardado en el registro de assignments correctamente



Ilustración 75: Test para verificar que se puede ver el detalle del usuario seleccionado

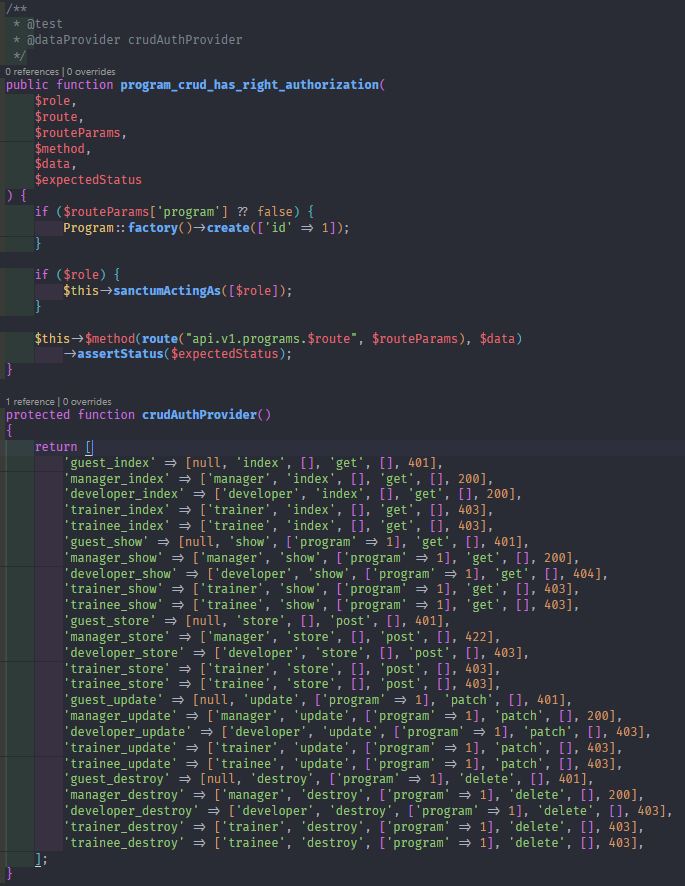


Ilustración 76: Test para verificar que el recurso "programs" se tenga la autorización correcta para los CRUDs en los diferentes endpoints que lo componen

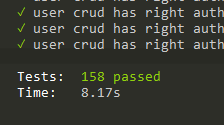


Ilustración 77: Los 158 tests finalizados de forma satisfactoria

# EXPOSICIÓN DE RESULTADOS FINALES AL GERENTE

La presentación de las ultimas correcciones y resultados finales con el gerente de tecnología coincidió con una reunión de seguimiento por parte del asesor de este proyecto el ingeniero Jaime Humberto Figueroa Henríquez. Se hizo énfasis en los tokens de permiso para el acceso de los endpoints y del *OAuth* con Google login que eran los apartados que faltaba por confirmar su validación el gerente de tecnología Benjamín Zepeda de Sistemas Pliret.

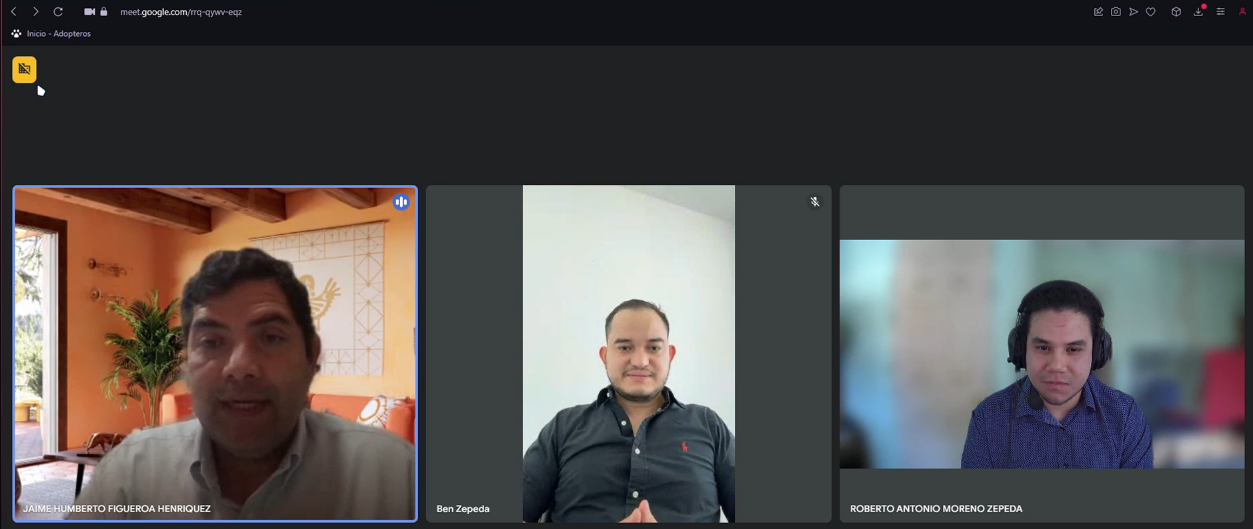


Ilustración 78: Reunión sobre avances finales y seguimiento de resultados

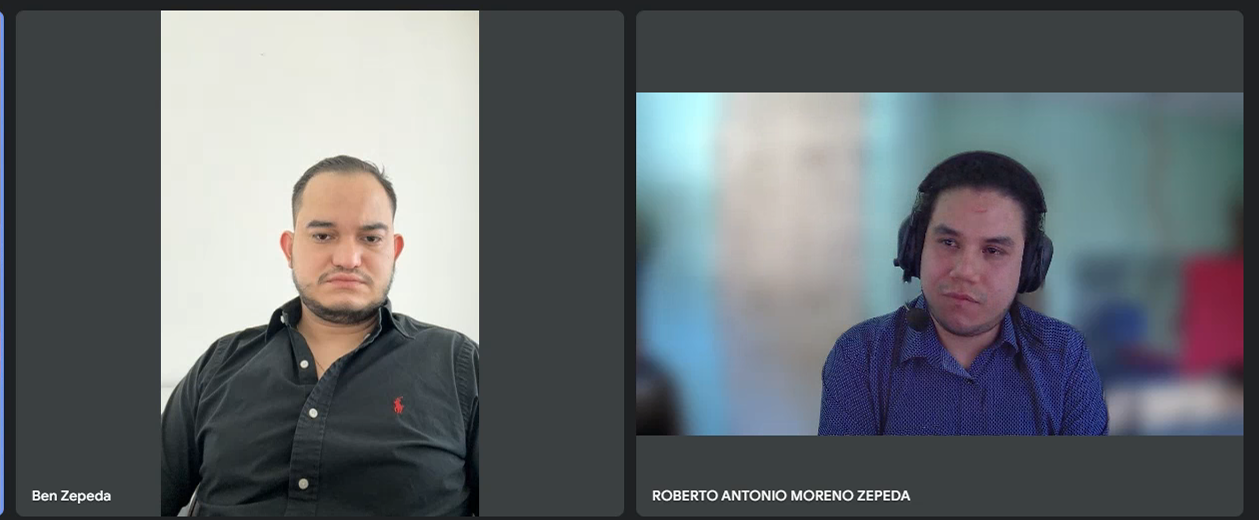


Ilustración 79: El facilitador (izquierda) y el desarrollador (derecha) del proyecto en la reunión de exposición de resultados.

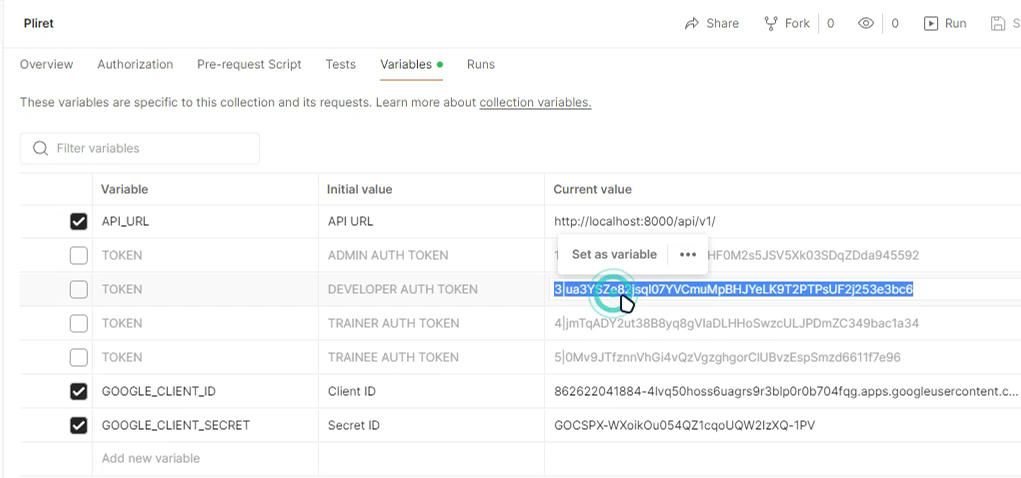


Ilustración 80: Explicación de los cuatro tipos de tokens

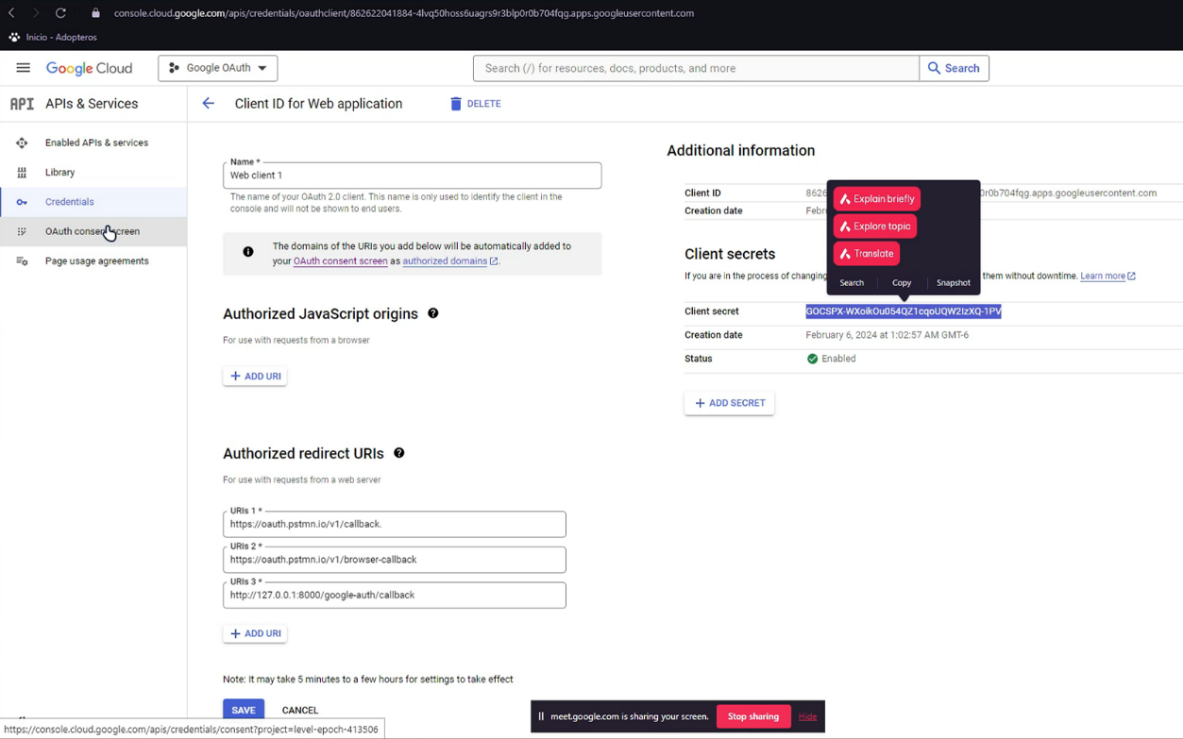


Ilustración 81: Demostración de las configuraciones de google cloud platform para el OAuth

# ELEMENTOS DE SOPORTE

# Anexo 1: Glosario

**API**: Interfaz de Programación de Aplicaciones (API), es un conjunto de reglas y definiciones que permiten que dos aplicaciones se comuniquen entre sí. Es como un menú de un restaurante, donde el cliente (la aplicación que solicita datos) pide al servidor (la aplicación que proporciona datos) información específica siguiendo las reglas del menú (la API). Las *APIs* permiten a los desarrolladores acceder a datos y funcionalidades de otras aplicaciones, integrar diferentes sistemas y automatizar tareas. Son herramientas esenciales para el desarrollo de software moderno, presentes en redes sociales, comercio electrónico, mapas, servicios meteorológicos y muchas otras áreas.

**Endpoints**: En el contexto de una *API*, es un punto de acceso específico dentro de la API que define una operación concreta.

En términos técnicos.

* Un Endpoint se define mediante una *URL* específica.
* Puede ser utilizado para obtener, enviar, eliminar o actualizar datos.
* Los Endpoints permiten modularizar la *API* y controlar el acceso a sus recursos.

**Endpoints y métodos HTTP en APIs**:

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.

Método GET:

Endpoint: /api/users/{id}

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.

Método POST:

Endpoint: /api/users

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.

Método PUT o PATCH:

Endpoint: /api/users/{id}

Se utiliza el método PUT si se quiere actualizar toda la información de un recurso o el método PATCH si se desea actualizar parcialmente el recurso

Descripción: Actualiza la información del recurso de usuario existente.

Método: DELETE

Endpoint: /api/users/{id}

Se utiliza para borrar un recurso usuario por medio del id.

Descripción: Elimina un usuario.

**Autorización**: Es el proceso de determinar si un usuario o aplicación tiene permiso para acceder a un recurso o realizar una acción.

**CRUD**: Las operaciones CRUD (Create, Read, Update, Delete) son las acciones básicas que se pueden realizar sobre datos en una aplicación. Corresponden respectivamente a crear, leer, actualizar y borrar.

**API REST**: Es un conjunto de reglas que define cómo dos aplicaciones se comunican entre sí a través de Internet. Se basa en el protocolo HTTP y utiliza verbos como GET, POST, PUT y DELETE para realizar operaciones sobre recursos. Las *APIs* *REST* son fáciles de usar y entender, lo que las hace muy populares para el desarrollo web.

**HTTP**: Es un protocolo de red que se utiliza para transferir datos entre un cliente y un servidor.

**Cliente HTTP**: Es un software que se ejecuta en un dispositivo (como un ordenador, teléfono o tablet) y se encarga de enviar solicitudes a un servidor web utilizando el protocolo HTTP. Estas solicitudes pueden ser para obtener información (GET), enviar información (POST), actualizar información (PUT) o eliminar información (DELETE). Un ejemplo es un navegador web.

**Requerimientos funcionales**: Definen las funcionalidades que debe tener un sistema o software para cumplir con las necesidades de los usuarios.

**Requerimientos no funcionales**: son características o restricciones que se aplican a un sistema o software, pero que no se relacionan directamente con su comportamiento funcional. Estos requisitos definen las cualidades del sistema, como su rendimiento, seguridad, usabilidad y mantenibilidad.

**URL**:Uniform Resource Locator o *URL* es una dirección única que se utiliza para identificar un recurso en Internet.

**Persistencia**: En el contexto de un proyecto de programación, la persistencia se refiere a la capacidad de almacenar datos en una base de datos de forma permanente, de modo que puedan ser recuperados y utilizados posteriormente, incluso después de que la aplicación se haya cerrado.

Cuando se persisten las tablas de una base de datos en las clases de un proyecto, se crea una representación en código de las estructuras y relaciones de la base de datos (en este proyecto son las clases *models* de Laravel). Esto permite que la aplicación interactúe con la base de datos de forma más eficiente y segura, ya que no necesita acceder directamente a los archivos de la base de datos.

**Depuración de código**: Es el proceso de identificar y corregir errores en el código fuente de un programa. Es una parte fundamental del desarrollo de software y ayuda a garantizar que el software sea funcional, confiable y eficiente.

**Script**: Es un conjunto de instrucciones que se ejecuta en el motor de la base de datos para realizar una tarea específica. Los scripts se pueden usar para automatizar tareas repetitivas, como la creación de tablas, la inserción de datos o la generación de informes.

**Historias de usuario**: son una herramienta ágil para describir las funcionalidades de un software desde la perspectiva del usuario final.

**Petición**: es una solicitud que un cliente envía a un servidor para obtener o modificar datos. La petición se envía a través de una *API*

**Header**: Un header, o encabezado, es una línea de información que se envía en una petición HTTP o en una respuesta HTTP. Los *headers* se utilizan para proporcionar información adicional sobre la petición o la respuesta, como el tipo de contenido, la longitud del contenido, el idioma o las cookies.

**Programación**: Es el proceso de crear instrucciones que una computadora puede seguir para realizar una tarea. Estas instrucciones se escriben en un lenguaje de programación, que es un conjunto de reglas y símbolos que la computadora puede entender.

**Migrations**: Las migraciones en Laravel son un sistema para controlar los cambios en la estructura de la base de datos de los proyectos. En lugar de modificar la base de datos directamente, se crean archivos de migración que definen los cambios que se deben realizar.

**Licencia**: Es un conjunto de términos legales que definen cómo se puede usar el software. La licencia determina quién puede usar el software, cómo se puede distribuir y si se puede modificar o no además de condicionar el uso libre o de pago.

**Visual Studio Code**: Es un editor de código fuente ligero pero potente desarrollado por Microsoft. Es gratuito, de código abierto y multiplataforma, disponible para Windows, macOS y Linux.

**PHP**: Por sus siglas en inglés *Hypertext Preprocessor* es un lenguaje de programación de código abierto, interpretado y de propósito general, especialmente adecuado para el desarrollo web.

**Framework**: Es un conjunto de herramientas y componentes de software que se utilizan para desarrollar aplicaciones. Los *frameworks* proporcionan una estructura básica sobre la cual se puede construir la aplicación, lo que facilita y agiliza el proceso de desarrollo.

**StarUML**: Es una herramienta de modelado UML de código abierto y multiplataforma. Se utiliza para crear diagramas UML, que son una forma gráfica de representar el diseño de un sistema de software.

**Login**: El login, o inicio de sesión, es el proceso de identificación de un usuario en un sistema informático. Este proceso generalmente implica que el usuario proporcione un nombre de usuario y una contraseña.

**Git**: Es un sistema de control de versiones distribuido. Permite a los usuarios realizar un seguimiento de los cambios realizados en archivos y directorios. Git es una herramienta muy popular entre los desarrolladores de software, pero también se puede utilizar para cualquier tipo de proyecto que implique la gestión de archivos.

**GitHub**: Es una plataforma de alojamiento de código fuente para control de versiones usando Git.

**GitLab**: Es una plataforma web de código abierto que ofrece herramientas para el desarrollo de software, gestión de proyectos y DevOps. Se basa en el sistema de control de versiones Git

**JSON**: Es un formato de intercambio de datos ligero y fácil de leer. Se basa en la sintaxis de objetos literales de JavaScript, pero es independiente del lenguaje, lo que significa que puede ser utilizado por cualquier lenguaje de programación.

**Postman**: Es una plataforma de software para el desarrollo y pruebas de *API*. Permite a los usuarios crear, enviar y recibir solicitudes HTTP, así como inspeccionar y analizar las respuestas.

**HeidiSQL**: Es una aplicación gratuita y de código abierto para la administración de bases de datos MySQL, MariaDB y PostgreSQL.

**SQL**: Es un lenguaje de consulta estructurado diseñado para administrar y recuperar información de sistemas de gestión de bases de datos relacionales.

**OAuth**: Es un protocolo abierto que permite a los usuarios autorizar a aplicaciones de terceros a acceder a sus datos sin compartir sus contraseñas.

**Navegador web**: También conocido como explorador web, es un programa informático que permite acceder a la *World Wide Web* (WWW). Su función principal es interpretar el código HTML y CSS de las páginas web para mostrarlas en la pantalla del usuario.

**Outsourcing**: Conocido como subcontratación o externalización, es una estrategia empresarial en la que una empresa delega a un proveedor externo la realización de ciertas tareas o actividades que no forman parte de su actividad principal.

**MVC**: Es un patrón de diseño de software que se utiliza para crear interfaces de usuario. Se basa en la separación de la lógica de la aplicación en tres capas:

* Modelo: La capa de modelo se encarga de almacenar y gestionar los datos de la aplicación.
* Vista: La capa de vista se encarga de presentar la información al usuario. En este proyecto corresponde a los *endpoints*.
* Controlador: La capa de controlador se encarga de gestionar la interacción entre el usuario y la aplicación.

**Ofimática**: La ofimática es el conjunto de herramientas, aplicaciones y métodos informáticos que se utilizan para realizar tareas de oficina. Su objetivo es mejorar la eficiencia y la productividad en el trabajo administrativo.

**Backend**: También conocido como lado del servidor, es la parte de una aplicación web o móvil que no es visible para el usuario. Se encarga de procesar las solicitudes del usuario, acceder a la base de datos y generar la respuesta que se envía al usuario.

**Frontend**: También conocido como lado del cliente, es la parte de una aplicación web o móvil que es visible para el usuario. Es la interfaz con la que el usuario interactúa directamente.

**Repositorio de archivos**: Es un sistema de almacenamiento centralizado que permite organizar, compartir y gestionar archivos digitales.

**Token**: Es una cadena de caracteres que se utiliza para autorizar a un usuario o aplicación a acceder a los recursos de una *API*. Es como una llave que permite al usuario abrir la puerta y acceder a la información o funcionalidades que ofrece la API.

Los tokens de acceso son necesarios para garantizar la seguridad de las *APIs*. Permiten controlar quién tiene acceso a los recursos de la *API* y qué acciones pueden realizar.

Las principales características de un token de acceso son:

* Es único: Cada token de acceso es único e identificativo.
* Tiene una vida útil limitada: Los tokens de acceso caducan después de un tiempo determinado, lo que obliga al usuario a autenticarse de nuevo.
* Puede ser revocado: Los tokens de acceso pueden ser revocados por el servidor de la API en caso de que sea necesario, como, por ejemplo, si se sospecha de un uso indebido.

Los tokens de acceso se pueden obtener de diferentes maneras, como:

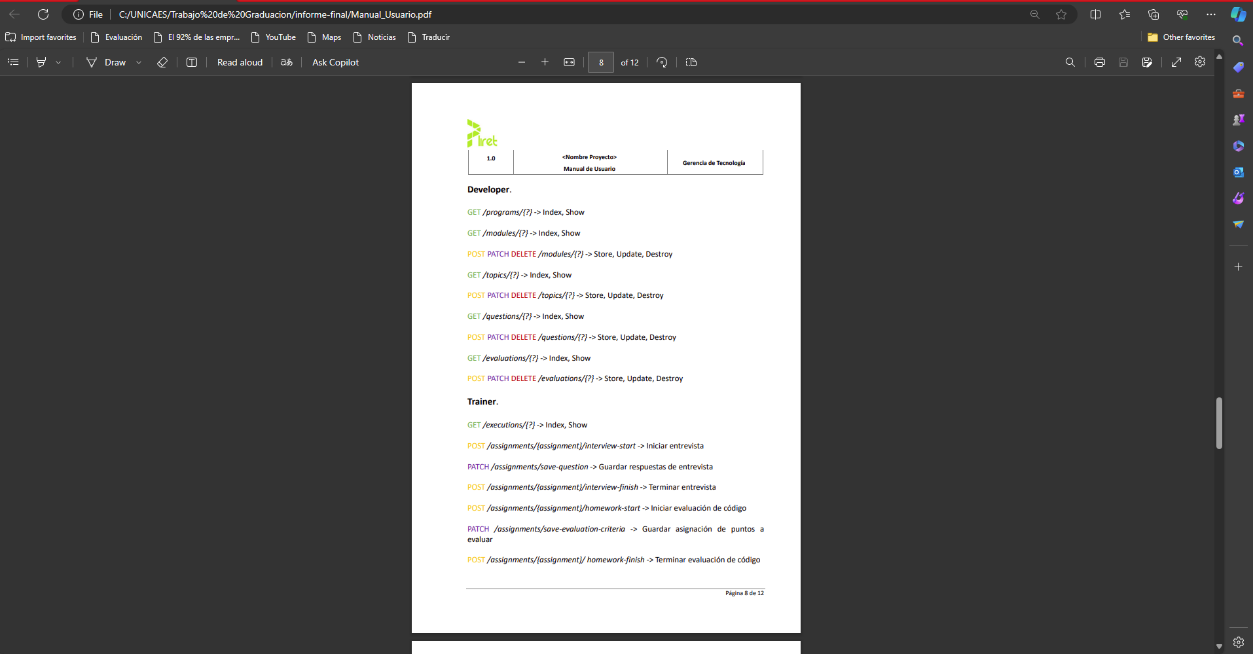
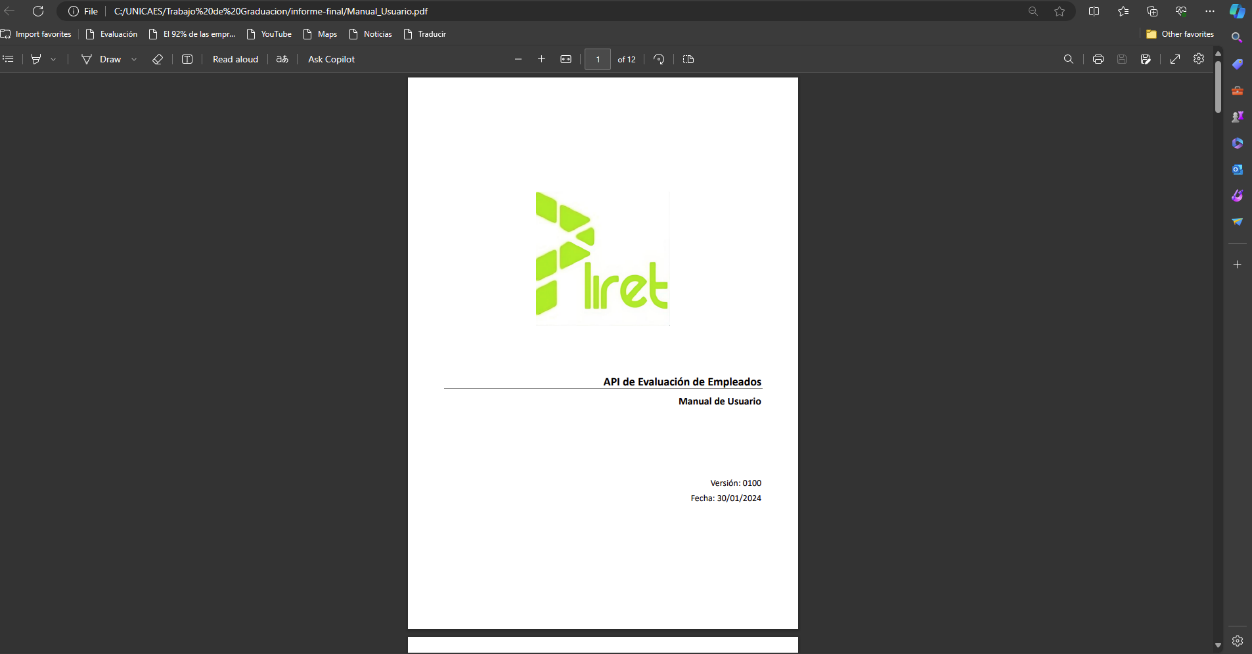
* Autenticación básica: El usuario proporciona su nombre de usuario y contraseña al servidor de la API, que genera un token de acceso si la autenticación es correcta.
* OAuth: Es un protocolo de autorización estándar que permite a los usuarios autorizar a aplicaciones de terceros a acceder a sus recursos sin necesidad de compartir sus credenciales.

Los tokens de acceso se suelen enviar en la cabecera de la solicitud HTTP al servidor de la *API*. El servidor de la *API* valida el token de acceso y, si es válido, permite al usuario acceder a los recursos de la *API*.

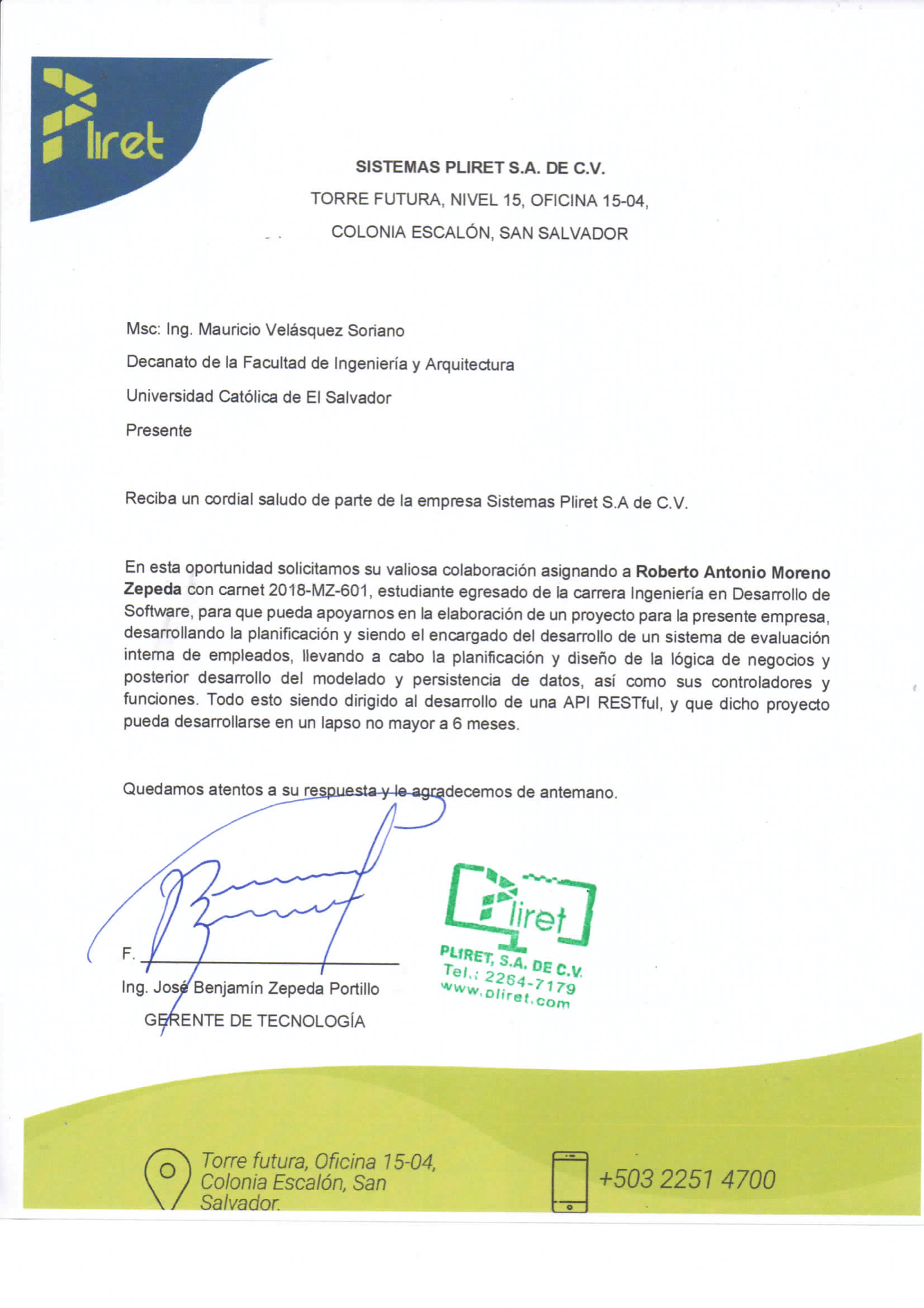
**Datos fake**: Son datos ficticios que se utilizan para probar la funcionalidad de la aplicación sin necesidad de utilizar datos reales. Estos datos se pueden generar de forma manual o automática utilizando la herramienta de *faker* de Laravel.

# Anexo 2: Manual de usuario

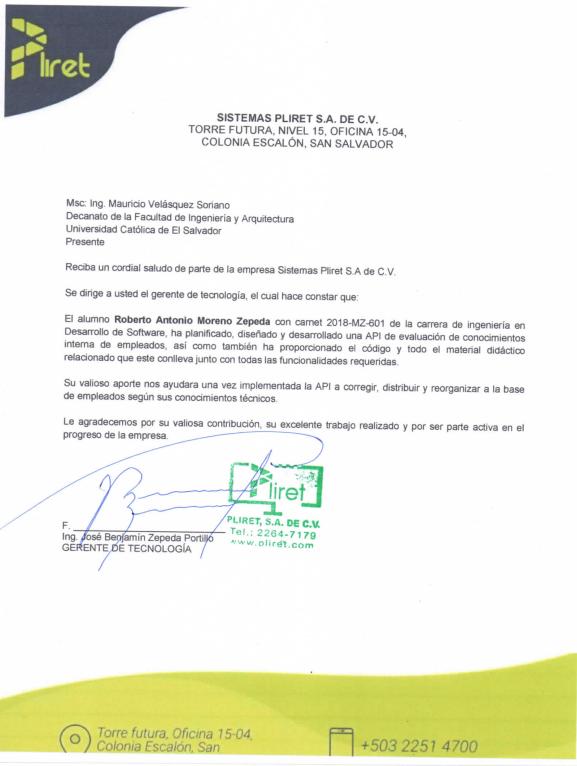
El manual de usuario se encuentra alojado en el repositorio del proyecto en la plataforma de GitHub y se compartió con el gerente de tecnología para que el lo haga llegar a los empleados de la empresa.



# Anexo 3: Carta de solicitud de creación por parte de la empresa



# CARTA DE RECEPCCIÓN DEL PROYECTO



# BIBLIOGRAFIA

*¿Como acceder a Google OAuth?* (13 de 03 de 2023). Obtenido de https://blog.postman.com/how-to-access-google-apis-using-oauth-in-postman/.

*¿Como construir una API con PHP?* (27 de 05 de 2021). Obtenido de https://code.tutsplus.com/how-to-build-a-simple-rest-api-in-php--cms-37000t.

*¿Qué es postman?* (19 de 10 de 2023). Obtenido de https://apidog.com/blog/what-is-postman/?utm\_source=google\_dsa&utm\_medium=g&utm\_campaign=20556541359&utm\_content=154844519700&utm\_term=&gad\_source=1&gclid=Cj0KCQiA5-uuBhDzARIsAAa21T-FGCbzTuYHqm7MYfNnx5BqzLXfOeDCcfX7\_5y0GqwfssIRkj280oAaAg50EALw\_wcB.

*¿Qué es REST?* (12 de 12 de 2023). Obtenido de https://restfulapi.net/.

*¿Qué es una documentación para una API?* (2022). Obtenido de https://www.postman.com/api-platform/api-documentation/.

*API Testing.* (19 de 10 de 2023). Obtenido de https://apidog.com/blog/what-is-postman/?utm\_source=google\_dsa&utm\_medium=g&utm\_campaign=20556541359&utm\_content=154844519700&utm\_term=&gad\_source=1&gclid=Cj0KCQiA5-uuBhDzARIsAAa21T-FGCbzTuYHqm7MYfNnx5BqzLXfOeDCcfX7\_5y0GqwfssIRkj280oAaAg50EALw\_wcB.

*Crear una API REST usando Laravel.* (2022). Obtenido de https://blog.treblle.com/how-to-create-rest-api-using-laravel/.

*Desarrollo de una API.* (s.f.). Obtenido de https://swagger.io.

*PHP Unit con Laravel.* (15 de 06 de 2022). Obtenido de https://semaphoreci.com/community/tutorials/getting-started-with-phpunit-in-laravel.

*Uso de Microsoft Graph API.* (26 de 07 de 2023). Obtenido de https://learn.microsoft.com/en-us/graph/use-the-api.