Contenido

[Alcance del Proyecto 1](#_Toc182399168)

[Endpoints del Proyecto 1](#_Toc182399169)

[UserController 1](#_Toc182399170)

[UserCompanyController 3](#_Toc182399171)

[TestController 4](#_Toc182399172)

[TestQuestionController 5](#_Toc182399173)

[TestAssignmentsController 5](#_Toc182399174)

[QuestionController 6](#_Toc182399175)

[AnswerController 7](#_Toc182399176)

[Composición Básica del Proyecto 7](#_Toc182399177)

[Resumen 8](#_Toc182399178)

# 

# Alcance del Proyecto

El proyecto es una aplicación Backend que gestiona usuarios, empresas, tests, preguntas y respuestas. La aplicación permite realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) sobre estas entidades y proporciona endpoints para interactuar con ellas. Los usuarios pueden ser administradores, estudiantes o profesores, y las empresas pueden registrarse y gestionar sus datos. Los tests pueden ser asignados a usuarios y las respuestas a las preguntas de los tests pueden ser almacenadas y recuperadas.

## Endpoints del Proyecto

http://localhost:8080

UserController

#### Listar todos los usuarios

- Método: GET

- URL: “/api/users”

#### Buscar usuario por ID

- Método: GET

- URL: “/api/users/{id}”

#### Buscar administrador por correo

- Método: GET

- URL: “/api/users/admin/{email}”

#### Crear administrador

- Método: POST

- URL: “/api/users/admin”

#### Actualizar administrador por correo

- Método: PUT

- URL: “/api/users/admin/update/{email}”

#### Eliminar administrador por ID

- Método: DELETE

- URL: “/api/users/admin/deleteById/{id}”

#### Buscar estudiante por correo

- Método: GET

- URL: “/api/users/student/{email}”

#### Crear estudiante

- Método: POST

- URL: “/api/users/student”

#### Actualizar estudiante por correo

- Método: PUT

- URL: “/api/users/student/update/{email}”

#### Actualizar rol de estudiante por correo

- Método: PUT

- URL: “/api/users/student/role/{email}”

#### Actualizar estado de estudiante por correo

- Método: PUT

- URL: “/api/users/student/active/{email}”

#### Eliminar estudiante por ID

- Método: DELETE

- URL: “/api/users/student/deleteById/{id}”

#### Buscar profesor por correo

- Método: GET

- URL: “/api/users/teacher/{email}”

#### Crear profesor

- Método: POST

- URL: “/api/users/teacher”

#### Actualizar profesor por correo

- Método: PUT

- URL: “/api/users/teacher/update/{email}”

#### Actualizar rol de profesor por correo

- Método: PUT

- URL: “/api/users/teacher/role/{email}”

#### Actualizar estado de profesor por correo

- Método: PUT

- URL: “/api/users/teacher/active/{email}”

#### Eliminar profesor por ID

- Método: DELETE

- URL: “/api/users/teacher/deleteById/{id}”

### UserCompanyController

#### Listar todas las empresas

- Método: GET

- URL: “/usersCompanies”

#### Buscar empresa por número de identificación

- Método: GET

- URL: “/usersCompanies/{identificationNumber}”

#### Crear empresa

- Método: POST

- URL: “/usersCompanies”

#### Actualizar empresa por número de identificación

- Método: PUT

- URL: “/usersCompanies/edit/{identificationNumber}”

#### Actualizar estado de empresa por número de identificación

- Método: PUT

- URL: “/usersCompanies/status/{identificationNumber}”

#### Eliminar empresa por número de identificación

- Método: DELETE

- URL: “/usersCompanies/{identificationNumber}”

#### Buscar empresa por correo

- Método: GET

- URL: “/usersCompanies/correo/{email}”

#### Eliminar empresa por correo

- Método: DELETE

- URL: “/usersCompanies/correo/{email}”

#### Actualizar empresa por correo

- Método: PUT

- URL: “/usersCompanies/edit/correo/{email}”

#### Actualizar estado de empresa por correo

- Método: PUT

- URL: “/usersCompanies/status/correo/{email}”

### TestController

#### Listar todos los tests

- Método: GET

- URL: “/test”

#### Buscar tests por ID de empresa

- Método: GET

- URL: “/test/{companyId}”

#### Crear test

- Método: POST

- URL: “/test”

#### Eliminar test por ID

- Método: DELETE

- URL: “/test/deleteById/{id}”

#### Actualizar test

- Método: PUT

- URL: “/test”

#### Buscar test por ID

- Método: GET

- URL: “/test/{id}”

### TestQuestionController

#### Listar todas las preguntas asociadas a un test por su ID

- Método: GET

- URL: “/test-questions/test/{testId}”

### TestAssignmentsController

#### Listar todas las asignaciones de tests

- Método: GET

- URL: “/testAssignments”

#### Buscar asignación de test por ID

- Método: GET

- URL: “/testAssignments/{id}”

#### Crear asignación de test

- Método: POST

- URL: “/testAssignments”

#### Eliminar asignación de test por ID

- Método: DELETE

- URL: “/testAssignments/deleteById/{id}”

#### Actualizar asignación de test por ID

- Método: PUT

- URL: “/testAssignments/{id}”

### QuestionController

#### Listar todas las preguntas

- Método: GET

- URL: “/api/questions”

#### Buscar pregunta por ID

- Método: GET

- URL: “/api/questions/{id}”

#### Crear pregunta

- Método: POST

- URL: “/api/questions/create”

#### Actualizar texto de pregunta por ID

- Método: PUT

- URL: “/api/questions/updatetext/{id}”

#### Actualizar estado de pregunta por ID

- Método: PUT

- URL: “/api/questions/updatestatus/{id}”

#### Eliminar pregunta por ID

- Método: DELETE

- URL: “/api/questions/delete/{id}”

#### Eliminar pregunta por ID con mensaje personalizado

- Método: DELETE

- URL: “/api/questions/deletebyid/{id}”

### AnswerController

#### Listar todas las respuestas asociadas a un test y una pregunta por sus IDs

- Método: GET

- URL: “/answers/test/{testId}/question/{questionId}”

#### Crear respuesta

- Método: POST

- URL: “/answers”

#### Eliminar respuesta por ID

- Método: DELETE

- URL: “/answers/{id}”

## Composición Básica del Proyecto

El proyecto está compuesto por varias capas que siguen el patrón de diseño MVC (Modelo-Vista-Controlador):

1. Modelos (“model”): Representan las entidades del dominio, como “User”, “UserCompany”, “Test”, “Question”, “Answer”, etc. Estas clases están anotadas con “@Entity” y mapeadas a tablas de la base de datos.
2. Repositorios (“repository”): Interfaces que extienden “JpaRepository” y proporcionan métodos para realizar operaciones CRUD en las entidades. Ejemplos incluyen “UserRepository”, “UserCompanyRepository”, “TestRepository”, “QuestionRepository”, “AnswerRepository”, etc.
3. Servicios (“service”): Clases que contienen la lógica de negocio y se comunican con los repositorios para realizar operaciones en las entidades. Ejemplos incluyen “UserService”, “UserCompanyService”, “TestService”, “QuestionService”, “AnswerService”, etc.
4. Controladores (“controller”): Clases que manejan las solicitudes HTTP y se comunican con los servicios para realizar operaciones en las entidades. Proporcionan endpoints para interactuar con la aplicación. Ejemplos incluyen “UserController”, “UserCompanyController”, “TestController”, “TestQuestionController”, “TestAssignmentsController”, “QuestionController”, “AnswerController”, etc.
5. Excepciones (“exception”): Clases personalizadas para manejar excepciones específicas del dominio. Ejemplo: “ResourceNotFoundException”.

## Colección Request

La Colección se encuentra almacenada en un espacio de trabajo en la aplicación de Postman, allí encontraremos una carpeta por cada endpoints posible a realizar. Se puede acceder a través del siguiente enlace:

<https://estramypymebackend.postman.co/workspace/EstraMyPyme_Backend~828e4b65-e033-47ad-90fa-104e2d4f6b78/collection/38604144-8ce70132-6cab-4bff-9816-704a8e84d3eb?action=share&creator=38604144>

## Tecnologías Utilizadas

De acuerdo a las necesidades encontradas en las entrevistas con el cliente, se consideran que las tecnologías pertinentes a utilizar son las siguientes:

* Java
* Spring Boot
* JWT
* MySQL
* Swagger
* Postman
* Maven
* Azure DevOps

## Instalación

Es importante que se contemple los siguientes prerrequisitos de instalación:

* Java 11+
* Maven
* MySQL

## Resumen

El proyecto es una aplicación Backend que gestiona usuarios, empresas, tests, preguntas y respuestas. Proporciona una serie de endpoints para realizar operaciones CRUD en estas entidades y sigue el patrón de diseño MVC para organizar el código. Los controladores manejan las solicitudes HTTP, los servicios contienen la lógica de negocio y los repositorios interactúan con la base de datos.