API para administration de colegios y curso de estudiantes

En el proyecto se implementa una API RESTful para la administración de colegios y cursos de estudiantes. Se van a poder de alta los modelos de

- **Alumnos**: Representa a los estudiantes que están inscritos en los colegios.
- **Docentes**: Representa a los profesores que imparten clases en los colegios.
- **Coordinador**: Representa a los a una persona que puede realizar taras de administración.
- **Especialidad**: Representa las especialidades o áreas de conocimiento que pueden tener los docentes.
- **Materias**: Representa las asignaturas que se enseñan en los colegios.
- **comisiones**: Representa las comisiones o grupos de estudiantes que cursan las materias.
- **Examenes**: Representa los exámenes que se realizan a los estudiantes.

Características Principales

- **Autenticación y Autorización**: Implementa autenticación básica para proteger los endpoints de la API.
- **Operaciones CRUD**: Permite crear, leer, actualizar y eliminar registros de todos los modelos.
- **Serialización**: Utiliza serializadores para convertir los modelos de Django en formatos JSON y viceversa.
- **Paginación**: Implementa paginación para manejar grandes volúmenes de datos de manera eficiente.
- **Validación de Datos**: Incluye validaciones para asegurar la integridad de los datos ingresados.
- **Documentación de la API**: Utiliza herramientas como Swagger o ReDoc para generar documentación interactiva de la API.
- **Relaciones entre Modelos**: Maneja relaciones entre los modelos, como la relación entre alumnos y comisiones, o docentes y materias.
- **Manejo de Errores**: Implementa un manejo adecuado de errores para proporcionar respuestas claras en caso de fallos.
- **Fragment Caching**: Utiliza técnicas de fragment caching para optimizar el rendimiento en la visualización de exámenes.
- **Middleware de Validación**: Incluye un middleware que valida si el usuario es válido y aplica técnicas de caché para bloquear IPs sospechosas.
- **Landing Page**: Proporciona una página de inicio para visualizar los exámenes creados, mejorando la experiencia del usuario.

Tecnologías Utilizadas

- **Django**: Framework web para el desarrollo de aplicaciones en Python.

- **Django REST Framework**: Extensión de Django para construir APIs RESTful.
- **SQLite**: Base de datos ligera para el almacenamiento de datos.
- **PDM**: Gestor de paquetes para manejar las dependencias del proyecto.
- **Swagger/ReDoc**: Herramientas para la documentación de la API.

Descripción del proyecto

El proyecto implementa una API RESTful que permite la gestión de colegios y cursos de estudiantes, la misma incluye los modelos antes mencionados, con la implementación de permisos para que no todos los usuarios puedan hacer todas las operaciones de CRUD. Cada modelo cuenta con sus respectivos serializadores para la conversión de datos entre formatos JSON y los modelos de Django. También incluye una landing page para ver los exámenes creados, la misma implementa las técnicas de fragmente caching para el listado de exámenes. En los modelos están distribuidos los diferentes requerimientos del trabajo como servios, implementación de técnicas de uso del ORM ect. El modelos Persona incluye un middleware, que valida si el usuario es valido e implementa la autenticación básica, y técnicas de chace para bloquear la ip. El proyecto también incluye una serie de pruebas unitarias para asegurar el correcto funcionamiento de la API, y se ha configurado GitHub Actions para la automatización del despliegue en un cluster de Docker Swarm administrado por Portainer, esto configuración no lo la había hecho antes y tuve que aprender a configurarla para que se ejecuten los test y se genere el despliegue automático, para lo cual use una Acction del Marketplace de GitHub que me permitía hacer el despliegue automático en el cluster usando la API de Portainer.

Retos

Los retos con los lo que me encontré al desarrollar este proyecto fueron:

- El meideware de validación de usuarios, ya que el middleware de Django no tiene en cuenta la autenticación básica, y no me reconoció al usuario y tube que implementar la lógica de validación de usuarios, una vez que confirme esta situación.
- Los test, con los que no tengo mucha experiencia, y tuve que aprender a implementarlos correctamente para asegurar el correcto funcionamiento de la API.
- La automatización mediante GitHub Actions, ya que no tenía experiencia previa con esta herramienta y tuve que aprender a configurarla para que se ejecuten los test y se genere genere el despliegue automático en el cluster de Docker Swarm.

Conclusión

Fue un lindo desafió, por la aplicación y consolidación de los conocimientos adquiridos en el bootmcamp, y por la implementación de nuevas tecnologías como GitHub Actions para la automatización del despliegue, lo que me permitió aprender a configurar un flujo de trabajo de CI/CD (Integración Continua/Despliegue Continuo) que mejora la eficiencia del desarrollo y despliegue de aplicaciones.