



UNIVERSIDAD
NACIONAL
DE COLOMBIA

Universidad Nacional de Colombia - Sede Bogotá
Facultad de Ingeniería
Curso: Ingeniería de Software I, Gr. 1
Grupo de Proyecto 8

Proyecto Final

Punto 2: Levantamiento de requerimientos

a) Narrativa del levantamiento inicial de requerimientos

En los primeros encuentros entre los integrantes del grupo, en horas de clase, se realizaron algunas discusiones y lluvias de ideas sobre posibles proyectos, dados los requerimientos y alcance del curso, además del tiempo disponible; dichas reuniones fueron complementadas con comunicación asincrónica.

Eventualmente, se determinó que el proyecto tomase un enfoque académico, en torno a educación superior, esto considerando que, como estudiantes activos de la Universidad, sería más fácil identificar posibles problemas a solucionar, necesidades que satisfacer o incluso, herramientas que podrían facilitar ciertos procesos en el entorno universitario, todo esto basado en experiencias generales del grupo de desarrolladores.

Finalmente, surgió la idea de centralizar el seguimiento y entrega virtual/asincrónica de proyectos, asignaciones o tareas en una sola plataforma, esto con el fin de abordar problemáticas relacionadas con la falta de eficiencia, coordinación y consistencia entre profesores y estudiantes a la hora de asignar, entregar (parcial o totalmente) y retroalimentar dichos proyectos académicos.

Se sabe que, en contexto de entregas virtuales o asíncronas, se suele utilizar más de una herramienta existente para las tareas ya mencionadas. Por ejemplo, correo electrónico para asignar las entregas, Moodle o Classroom para obtener material y realizar las entregas, y, posiblemente, nuevamente el correo electrónico para las entregas intermedias o retroalimentación directa que surja.

Este modelo de gestión virtual puede conllevar ciertas dificultades organizativas requiriendo cambiar de aplicaciones constantemente, potencialmente dificultando la monitorización de proyectos entregados en cada paso del desarrollo del proyecto y la comunicación retroalimentativa, conllevando posibles equivocaciones, retrasos y falta de transparencia.

Se busca proporcionar un sistema en el que el público objetivo, profesores y estudiantes, encuentre funcionalidades prácticas e intuitivas para la creación de grupos o cursos concretos, asignación de proyectos, entrega de avances de proyecto o proyectos ya finalizados y dar/entregar su retroalimentación correspondiente de forma oportuna, gestión de varios proyectos, subida de documentos relacionados con los proyectos, registro de actividades para hacer un seguimiento del progreso individual o grupal, entre otras.

La expectativa general es que este sistema simplifique la interacción entre docentes y estudiantes, minimizando la pérdida de información y mejorando la experiencia académica, todo sin perder funciones que otorgan calidad de vida a la creciente necesidad de

digitalizar los procesos académicos, como por ejemplo, las notificaciones de nueva tarea que otorgan plataformas como Classroom.

Como equipo de estudiantes de ingeniería y desarrollo de software, esperamos que este proyecto nos permita poner en práctica nuestros conocimientos en programación, bases de datos, y gestión de proyectos en un contexto realista, a medida que los adquirimos, aprendiendo nuevas herramientas, lenguajes y frameworks relevantes para el desarrollo backend y frontend, como FastAPI o React, sin perder de vista el mejoramiento de nuestras habilidades colaborativas de trabajo en equipo y coordinación, ya sea en tiempo real o asíncrona.

También deseamos contribuir una posible solución adaptable a contextos educativos reales como un aporte potencialmente significativo y tangible a la mejora de estos procesos académicos.

b) Identificación inicial de requerimientos

En este paso se logra, también, identificar algunos posibles procesos asociados a los requerimientos.

- **Gestión de Usuarios:** Los Usuarios deben poder registrarse e iniciar sesión en el sistema, con roles diferenciados para estudiantes y docentes. El Administrador gestiona los roles.
- **Creación y Asignación de Proyectos y Tareas:** Los Docentes pueden crear proyectos académicos (véase, tareas/asignaciones), definiendo un título, descripción, fecha de entrega, y asignando Estudiantes al proyecto.
- **Gestión de Proyectos y Tareas:** Los Docentes pueden actualizar el estado del proyecto.
- **Seguimiento del Estado del Proyecto:** Cada proyecto tiene un estado (por ejemplo, "En progreso", "Finalizado") que se actualiza según el progreso de las tareas. Los Docentes pueden actualizar el estado del proyecto. Los Estudiantes pueden consultar el estado del proyecto.
- **Actualización de Avances y Observaciones:** Los estudiantes pueden registrar avances en cada tarea y recibir comentarios del docente para asegurar el cumplimiento de objetivos.
- **Adjuntar Documentos y Archivos:** Se permite que los Estudiantes suban archivos o enlaces relevantes para sus proyectos asignados. Los Docentes pueden subir archivos y enlaces a cada proyecto (por ejemplo, guía o materiales relacionados a este).
- **Accesibilidad y Usabilidad:** La aplicación debe funcionar en una plataforma web, accesible tanto en computadoras de escritorio como en dispositivos móviles. Debe ser una plataforma intuitiva.
- **Seguridad:** La plataforma almacenará datos personales de las personas involucradas (estudiantes, docentes y otro personal académico), además de credenciales de inicio de sesión acorde a cada rol asignado. Estos datos deben ser protegidos adecuadamente mediante técnicas de cifrado de datos.
- **Notificaciones de Actividades:** El sistema envía notificaciones sobre actualizaciones importantes, como nuevos comentarios, tareas asignadas o cambios en las fechas límite.

- Historial de Actividades: La aplicación mantiene un registro de todas las actividades relacionadas con cada proyecto, así, los Estudiantes y Docentes pueden consultar el progreso de estas.
- Visualización: Proporcionar a la plataforma un diseño intuitivo y legible, con tipografías claras y simples, íconos y botones llamativos, y colores profesionales.

Punto 3: Análisis de requerimientos

a) Lista consolidada de requerimientos

Organizando, desglosando y consolidando los requerimientos y procesos identificados con el desarrollo y síntesis de la idea del proyecto. Se visualizan primero los Requerimientos Funcionales (RF), luego los No-Funcionales (RNF).

Requerimientos Funcionales (RF):

***Prioridad asociada antes de hacer la clasificación MoSCoW**

ID	Usuario (Opcional): Funcionalidad	Proceso Asociado	Prioridad*
RF-001	Usuario (cualquiera): El Usuario se registra en la Aplicación con su información– el registro no otorga rol por defecto.	Gestión de usuarios	Alta
RF-002	Usuario (cualquiera): Iniciar sesión en la Aplicación con sus datos registrados.	Gestión de usuarios	Alta
RF-003	Administrador: Asignar roles de Estudiante o Profesor al Usuario	Gestión de usuarios	Alta
RF-004	Profesor: Crear grupos (clases) y asignar Estudiantes a dichos grupos para el seguimiento o entrega de sus proyectos.	Gestión de usuarios	Alta
RF-005	Profesor: Crear proyectos (véase: tareas o asignaciones) para un grupo (clase). Contiene información como título, descripción, fecha de entrega y estado.	Gestión de asignaciones	Alta
RF-006	Profesor: Asignar proyectos a Estudiante(s) en específico del curso en cuestión.	Gestión de asignaciones	Media
RF-007	Profesor: Actualizar el estado de un proyecto acorde al avance de las tareas (entregas intermedias registradas).	Gestión de asignaciones	Baja
RF-008	Estudiante: Consultar el estado de un proyecto asignado y su información correspondiente.	Gestión de asignaciones	Alta
RF-009	Estudiante: Registrar un avance (ej. entrega intermedia) en cada proyecto asignado, según sea requerido.	Gestión de asignaciones	Media
RF-010	Estudiante: Realizar una entrega final para un proyecto asignado.	Gestión de asignaciones	Alta
RF-011	Profesor: Otorgar retroalimentación a las entregas de proyecto realizadas por estudiantes asignados; se entrega a ellos.	Gestión de asignaciones	Media
RF-012	Estudiante: Subir archivos o enlaces relevantes para las entregas de proyectos asignados.	Archivos adjuntos	Alta

RF-013	Profesor: Subir archivos o enlaces para cada proyecto (ej. guías o materiales).	Archivos adjuntos	Alta
RF-014	Usuario (cualquiera): Subir una foto o ícono de perfil asociada al usuario en cuestión.	Archivos adjuntos	Baja
RF-015	Usuario (cualquiera): Consultar progresos anteriores de actividades asignadas.	Registro de actividades	Baja
RF-016	Estudiante: Consultar los proyectos asignados y su identificador.	Gestión de Asignaciones	Alta

Requerimientos No-Funcionales (RNF):

***Prioridad asociada antes de hacer la clasificación MoSCoW**

ID	Usuario (Opcional): Funcionalidad	Proceso Asociado	Prioridad*
RNF-001	La Aplicación debe ser compatible con dispositivos móviles (smartphones, tablets) y de escritorio (PC).	Accesibilidad y usabilidad	Alta
RNF-002	La Aplicación debe ser intuitiva y fácil de usar.	Accesibilidad y usabilidad	Alta
RNF-003	Asegurar la protección de datos personales y/o sensibles mediante cifrado.	Seguridad	Alta
RNF-004	Utilizar credenciales seguras para evitar brechas de seguridad o accesos de rol no autorizados.	Seguridad	Alta
RNF-005	La Aplicación en menos de 5 segundos en conexiones en condiciones estándar.	Rendimiento	Alta
RNF-006	La Aplicación debe soportar usuarios múltiples sin detrimento del rendimiento.	Rendimiento	Media
RNF-007	La Aplicación envía notificaciones a los Usuarios sobre actualizaciones importantes (ej. nueva tarea, comentarios, cambios en fechas, entrega realizada).	Notificaciones	Media
RNF-008	La Plataforma mantiene un registro de actividades de cada proyecto.	Registro de actividades	Baja
RNF-009	El diseño del sitio debe permitir la integración a futuro de nuevas funcionalidades sin requerir una reestructuración completa.	Escalabilidad	Media
RNF-010	Diseño visual limpio, con una tipografía clara.	Diseño	Media
RNF-011	Paleta de colores profesional. Íconos y botones llamativos.	Diseño	Media

b) Procesos Asociados identificados

- Para Requerimientos Funcionales (RF):
 - Gestión de usuarios
 - Gestión de asignaciones
 - Archivos adjuntos
 - Gestión de cursos (opcional)
- Para Requerimientos No-Funcionales (RNF):
 - Accesibilidad y usabilidad
 - Seguridad
 - Rendimiento
 - Notificaciones
 - Registro de actividades
 - Escalabilidad
 - Diseño

c) Clasificación MoSCoW

La priorización realizada, tanto para Requerimientos Funcionales (RF) como para Requerimientos No-Funcionales (RNF), puede resumirse de la siguiente manera:

Must Have (M)	Should Have (S)	Could Have (C)	Won't Have (W)
RF-001	RF-004	RF-006	RF-014
RF-002	RF-011	RF-007	<i>RNF-011</i>
RF-003	RF-012	RF-009	
RF-005	RF-013	RF-015	
RF-008	<i>RNF-006</i>	<i>RNF-005</i>	
RF-010		<i>RNF-007</i>	
RF-016		<i>RNF-008</i>	
<i>RNF-001</i>		<i>RNF-010</i>	
<i>RNF-002</i>			
<i>RNF-003</i>			
<i>RNF-004</i>			
<i>RNF-009</i>			

Se han tenido en cuenta ciertos factores para la clasificación de las funcionalidades y su estimación de esfuerzo.

Recursos disponibles: Un equipo de 4 estudiantes universitarios que harán las veces de desarrolladores, arquitectos (de software), diseñadores de la ux y testers. Se cuenta con ciertos recursos cuyas limitaciones varían:

- Tiempo para el diseño, desarrollo y testing: varía según cada miembro del equipo. Se tienen en cuenta factores como los estudios universitarios y trabajos fuera de la universidad.
- Repositorio: Para este proyecto se utilizará un repositorio de GitHub. Al contar con una cuenta Pro, se dispone de herramientas de despliegue, en caso de ser necesarias.
- Almacenamiento en la nube: para la sincronización y/o construcción colaborativa de documentos varios, se utilizan servicios como Google Docs o Microsoft 365, disponibles mediante el correo universitario. El límite de almacenamiento varía entre 15 y 100 GB, según cada servicio. La sincronización del desarrollo del proyecto depende del repositorio de GitHub.
- Herramientas externas: El software de desarrollo preferido también puede variar según el desarrollador y posteriormente pueden surgir más opciones a considerar; entre ellos pueden incluirse Visual Studio Code, PyCharm Community Edition*, etc. Entre los frameworks considerados, se encuentran: React y FastAPI. Otras herramientas de apoyo pueden incluir GitHub Copilot Pro y OpenAI ChatGPT.

*(*al no usarse en un entorno de desarrollo profesional, no debería implicar violación a los ToS).*

Complejidad técnica: dadas las características y necesidades del proyecto, además de los recursos disponibles con los que cuenta el equipo de desarrollo, se han identificado tres tipos de tareas, según su complejidad de implementación, independientemente de su prioridad:

- Tareas Básicas (2 a 5 puntos)
- Tareas Medias/moderadas (5 a 8 puntos)
- Tareas Complejas (8 a 13 puntos)

Impacto en la experiencia del usuario y Relación con los objetivos del proyecto:

En primera instancia, el objetivo se centra en construir una aplicación capaz de trabajar en sus funcionalidades primordiales, priorizadas como Must (y Should) en este módulo inicial. Las mejoras en experiencia de usuario son prioridad en funcionalidades posteriores, proyectadas como Should, Could y Won't.

La clasificación detallada de las funcionalidades, cada una con su valor de esfuerzo y justificación es la siguiente:

Must Have (M) Esenciales para funcionamiento básico.		
Funcionalidad	Esfuerzo (Fibonacci)	Motivo

Must Have (M) Esenciales para funcionamiento básico.		
RF-001 - Registro de usuarios	3	Requiere validación de datos como correo electrónico (formato único y correcto), además de almacenar contraseñas de forma segura.
RF-002 - Inicio de sesión	5	Involucra autenticación segura, y validación de usuario. Requiere respuesta rápida del sistema.
RF-003 - Asignación de roles	5	Requiere gestión y control de acceso según los roles asignados, probablemente mediante una interfaz destinada a ello.
RF-005 - Creación de proyectos	5	Involucra almacenamiento en la base de datos, manejo de formularios y validación de datos.
RF-008 - Consulta de estado de proyectos	3	Necesita consultas a la base de datos. Visualización en frontend.
RF-010 - Entrega final de proyectos	5	Involucra permitir carga de archivos y/o enlaces externos, validación del estado del proyecto y control de fechas previstas.
RF-016 - Consulta de qué proyectos están asignados	3	Necesita consultas a la base de datos. Visualización en frontend
RNF-001 - Compatibilidad multiplataforma web	8	Requiere pruebas en varias plataformas, además de un diseño responsivo.
RNF-002 - Usabilidad	5	Necesita un diseño bien estructurado de navegación, UI con un diseño bien realizado.
RNF-003 - Protección de datos	8	Puede involucrar la implementación de medidas de seguridad tales como cifrado de datos sensibles.
RNF-004 - Seguridad de credenciales	13	Puede involucrar hashing de contraseñas, además de otras políticas de contraseñas seguras, además de medidas de protección de roles autorizados tales como autenticación segura, basada en roles o protección contra escalada de privilegios.
RNF-009 - Diseño modular	5	Necesita una buena estructuración de la arquitectura del proyecto que permita escalabilidad

Should Have (S) Importantes, pero no críticas en el módulo 1		
Funcionalidad	Esfuerzo (Fibonacci)	Motivo
RF-004 - Creación de grupos (clases)	5	Necesita consultas de bases de datos, además de implementar un sistema de asociación entre profesores y estudiantes (como IDs de grupos en la base de datos).
RF-011 - Retroalimentación de entregas de proyectos	5	Debe crearse una interfaz para comentarios y almacenamiento en la base de datos. Deben manejarse permisos para la retroalimentación (que se entregue a el/los estudiante(s) de la entrega en particular).
RF-012 - Subida de archivos por estudiantes	8	Implica almacenamiento en el servidor. Necesita restricciones de tamaño y formato de archivos.
RF-013 - Subida de archivos por profesores	8	Similar a la subida de archivos por estudiantes. Necesita cambios en ciertos permisos.
RNF-006 - Capacidad para múltiples usuarios sin detrimento de rendimiento	8	Necesita optimizar consultas en la base de datos, además de un manejo eficiente de las sesiones.

Could Have (C) Deseables, pero pueden implementarse luego		
Funcionalidad	Esfuerzo (Fibonacci)	Motivo
RF-006 - Asignación de proyectos a estudiantes en específico	5	Requiere implementar un mecanismo adicional de asignación y control de permisos en proyectos (ejemplo, IDs de proyecto).
RF-007 - Actualización del estado de avance del proyecto por parte del profesor	5	Puede postergarse, debido a que el primer módulo sólo maneja entregas finales. Requiere manejo de estados “intermedios” de un proyecto (ejemplo, un campo de estado reportado) en la base de datos. Necesita una interfaz para el registro y la visualización.
RF-009 - Registro de avances en proyectos (como entregas intermedias)	5	Similar a la funcionalidad de actualización de estado por parte del profesor, requiere manejo de estados “intermedios” de un

Could Have (C) Deseables, pero pueden implementarse luego		
por parte de estudiantes		proyecto en la base de datos (ejemplo, historial de entregas intermedias). Necesita una interfaz para el registro y la visualización.
RF-015 - Consulta de progresos previos	2	Similar al registro de actividades. Requiere mayor organización en base de datos y consultas especializadas por proyectos y usuarios.
RNF-005 - Respuesta rápida del sistema en conexiones estándar	5	Gran parte de la optimización es una consecuencia de la implementación inicial de la arquitectura y la base de datos. Puede optimizarse más una vez se desarrolle la funcionalidad principal.
RNF-007 - Notificaciones	8	Requiere manejo de eventos y notificaciones, principalmente por correo electrónico. Se considera manejarlas en tiempo real también.
RNF-008 - Registro de actividades	5	Puede añadirse posteriormente sin afectar la funcionalidad principal de la aplicación. Requiere consultas en base de datos para cada proyecto e interfaz de visualización.
RNF-010 - Diseño visual limpio	3	Puede mejorarse en iteraciones posteriores sin afectar la funcionalidad del sistema. Requiere indagar sobre e implementar buenas prácticas de diseño web.

Won't Have (W) Fuera del alcance actual del proyecto		
Funcionalidad	Esfuerzo (Fibonacci)	Motivo
RF-014 - Personalizar ícono de perfil	3	No impacta la funcionalidad del sistema. Implica almacenamiento en el servidor, además de un control de permisos (cada usuario sólo maneja su foto; dicha foto es pública).
RNF-011 - Paleta de colores profesional, elementos llamativos	2	Al igual que el diseño visual limpio, requiere implementar buenas prácticas de diseño y puede implementarse posteriormente.

Punto 4: Análisis gestión de software

Este análisis lo componen tres elementos fundamentales: Tiempo, Costo y Alcance. Después de indagar, se ha concluido lo siguiente.

a) Tiempo

Basándonos en el tiempo disponible para la parte final del semestre– las fases del proyecto realizadas hasta la semana 11, además del contexto grupal– 4 estudiantes en etapa de aprendizaje de ingeniería de software, se han estimado los siguientes tiempos para las funcionalidades clasificadas como Must y Should; se han agrupado en módulos, los cuales, aunque similares a los procesos asociados, no necesariamente coinciden uno-a-uno:

Módulo	Funcionalidades	Tiempo estimado	Actividades principales
Autenticación y gestión de usuarios	<ul style="list-style-type: none"> RF-001 RF-002 RF-003 	6-7 días	<ul style="list-style-type: none"> Implementación de registro e inicio de sesión de usuarios Gestión de roles
Gestión de asignaciones: Asignación de proyectos	<ul style="list-style-type: none"> RF-005 RF-008 RF-016 	4-5 días	<ul style="list-style-type: none"> Creación y asignación de proyectos por parte del profesor Consulta de detalles
Gestión de asignaciones: Entrega de proyectos	<ul style="list-style-type: none"> RF-010 RF-012 RF-013 	6-7 días	<ul style="list-style-type: none"> Gestión de entregas finales de proyectos Implementación y gestión de archivos para entregas y materiales
Creación de clases, registro de retroalimentación	<ul style="list-style-type: none"> RF-004 RF-011 	5-6 días	<ul style="list-style-type: none"> Crear sistema de asociación de clase-profesor Implementación de interfaz de comentarios
Usabilidad, optimización y seguridad	<ul style="list-style-type: none"> RNF-001 RNF-002 RNF-003 RNF-004 RNF-006 	8-10 días	<ul style="list-style-type: none"> Implementación de cifrado de contraseñas y validaciones de seguridad Diseño de aplicación responsiva y fácil de usar Optimización para múltiples usuarios en simultánea
Documentación, pruebas y ajustes finales	<ul style="list-style-type: none"> RNF-009 	6-7 días	<ul style="list-style-type: none"> Realizar Testing e integración Corrección de errores, optimización Redacción de

			documentación faltante
--	--	--	------------------------

b) Costo

La estimación de costo de desarrollo de software se ha realizado a modo semanal, teniendo en cuenta ciertos factores.

Nota: para precios en Dólares Estadounidenses (USD), su estimación equivalente en Pesos Colombianos (COP) se ha realizado utilizando el tipo de cambio con corte a 30 de enero de 2025, siendo este \$1(USD) = \$4176(COP).

i) Sueldos de desarrolladores:

Rol/Experiencia/Especialidad	Salario promedio (COP) mensual	Equivalente en USD
Desarrollador Junior	\$2'200.000	\$530
Desarrollador semi-Senior	\$3'610.854	\$870
Desarrollador Senior/Arquitecto	\$5'400.000	\$1.301
Full-Stack	\$5'400.000	\$1.301
Back-End	\$5'400.000	\$1.301
Front-End/UX	\$4'300.000	\$1.036
Web	\$4'300.000	\$1.036
Tester	\$2'600.000	\$626

Fuentes:

- <https://platzi.com/tutoriales/3208-programacion-basica/24348-descubre-cuanto-gana-un-programador-en-mexico-colombia-argentina-y-ee-uu-en-2023/>
- <https://evalart.com/es/blog/cuanto-ganan-los-desarrolladores-en-america-latina/>
- https://www.glassdoor.com.ar/Sueldos/bogota-colombia-qa-tester-sueldo-SRCH_IL.0%2C15_IM1064_KO16%2C25.htm
- <https://co.indeed.com/career/tester/salaries>

ii) Servicios en la nube:

Se han utilizado estimados que se aproximan a las necesidades y alcance de este proyecto. Se incluyen servicios de API, Almacenamiento e Integración Continua para esta estimación.

Servicio	Precio en USD mensual	Equivalente en COP
Microsoft Azure	\$21	\$87.159
Google Cloud	\$22.67	\$94.090
AWS	\$21.94	\$91.060

Fuentes:

- <https://azure.microsoft.com/es-es/pricing/calculator/>
- <https://cloud.google.com/products/calculator>

- <https://calculator.aws/>

iii) Herramientas externas

Estas herramientas incluyen APIs, Frameworks y Software de Desarrollo. En el caso de las APIs, muchas ofrecen niveles gratuitos con limitaciones en el uso, a su vez que los planes de pago varían según el proveedor y cantidad de solicitudes. En cuanto a las librerías, la mayoría son de código abierto y gratuitos (FOSS), aunque algunas pueden requerir licencias comerciales según su uso.

En cuanto al software de desarrollo, para el alcance de este proyecto, se tienen tres opciones principales viables:

IDE	Tipo de Licencia	Costo USD/COP	Notas
Microsoft Visual Studio Code	Privada y comercial	Gratuita	Permite uso privado y comercial
Microsoft Visual Studio IDE	Profesional (Empresa)	\$45 USD/\$186.768 COP por mes	Licencia por usuario + plan básico de Azure DevOps
JetBrains PyCharm Professional	Profesional (Organizaciones)	\$24.90/\$103.345 COP por mes	Licencia por usuario

Fuentes:

- <https://code.visualstudio.com/docs/supporting/faq>
- <https://visualstudio.microsoft.com/es/vs/pricing/?tab=business>
- <https://www.jetbrains.com/es-es/pycharm/buy/?section=commercial&billing=monthly>

Con estos factores en consideración, a continuación se estiman los potenciales costos mensuales y semanales del desarrollo para este proyecto:

Estimación de costos de personal de desarrollo			
Roles de Desarrollador	Costo Mensual c/u (USD/COP)	Costo Semanal c/u (USD/COP)	Notas
Desarrollador Junior	\$450 / \$1'872.174	\$113 / \$470.123	Equipo de 4 desarrolladores para este proyecto.
	Costo Mensual total (USD/COP)	Costo Semanal total (USD/COP)	
	\$1.800 / \$7'488.695	\$450 / \$1'872.174	
Estimación de costo de servicios en la nube y software de desarrollo			
Servicios y programas	Costo Mensual (USD/COP)	Costo Semanal (USD/COP)	Notas
AWS	\$50 / \$208.025	\$13 / \$54.087	Haciendo una estimación

Estimación de costos de personal de desarrollo			
			de precio más conservadora.
Microsoft Visual Studio Code	Gratuito (\$0 / \$0)	Gratuito (\$0 / \$0)	Software a utilizar en el proyecto.
Total de estimaciones de Costo			
	Costo Mensual (USD/COP)	Costo Semanal (USD/COP)	Notas
TOTAL	\$1.850 / \$7'696.715	\$463 / \$1'926.259	Los centavos (decimales) de cada moneda han sido redondeados.

c) Alcance

De manera general, el objetivo de este proyecto es proporcionar una plataforma funcional para la gestión, asignación, seguimiento, entrega y retroalimentación de proyectos en un entorno académico.

Conceptualmente, se busca que la plataforma entregada permita que usuarios, previamente registrados por su propia cuenta, en sus roles de estudiantes y profesores, gestionen y den seguimiento a asignaciones y trabajos en un entorno académico.

Dentro de la aplicación, los usuarios podrán iniciar sesión en la plataforma y una vez logueados, utilizarla según los roles que tengan asignados. Dichos roles dependen del administrador de la plataforma.

Los profesores podrán tener la capacidad de crear y gestionar grupos equivalentes a sus clases o cursos, y en él, crear y gestionar asignaciones dirigidas a dicha clase. Gestionar clases implica agrupar un número de estudiantes registrados en ella y crear asignaciones y otras actualizaciones dentro de ella. Gestionar asignaciones implica crear y eliminar tareas, además de asignarlas con su información y material relevante; también se deberá poder asignar tareas a subgrupos de estudiantes específicos y otorgar retroalimentación para cada entrega de asignaciones (o proyectos o tareas).

Los estudiantes deberán tener la capacidad de revisar los proyectos asignados, visualizando su estado y demás información relevante. Además, podrán realizar entregas intermedias o finales, acorde a la situación, y recibir la retroalimentación provista por el profesor en cada entrega.

Los límites establecidos en este módulo del proyecto, dividen las funcionalidades en dos grupos: las que están incluidas en el MVP y las cuales quedan fuera del alcance del proyecto.

Las funcionalidades incluidas en el MVP incluyen las funcionalidades clasificadas como Must (M) y Should (S); aseguran que el sistema sea funcional para los usuarios destinados (estudiantes y profesores) en su versión inicial.

Funcionalidades incluidas en el MVP			
Módulo	ID	Funcionalidad	Proceso Asociado
Autenticación y gestión de usuarios	RF-001	Registro de usuarios	Gestión de usuarios
	RF-002	Inicio de sesión	Gestión de usuarios
	RF-003	Asignación de roles	Gestión de usuarios
Gestión de asignaciones: Asignación de proyectos	RF-005	Creación de proyectos	Gestión de asignaciones
	RF-008	Consulta de estado de proyectos	Gestión de asignaciones
	RF-016	Consulta de qué proyectos están asignados	Gestión de asignaciones
Gestión de asignaciones: Entrega de proyectos	RF-010	Entrega final de proyectos	Gestión de asignaciones
	RF-012	Subida de archivos por estudiantes	Archivos adjuntos
	RF-013	Subida de archivos por profesores	Archivos adjuntos
Creación de clases, registro de retroalimentación	RF-004	Creación de grupos (clases)	Gestión de usuarios
	RF-011	Retroalimentación de entregas de proyectos	Gestión de asignaciones
Usabilidad, optimización y seguridad	RNF-001	Compatibilidad multiplataforma web	Accesibilidad y usabilidad
	RNF-002	Usabilidad de la aplicación	Accesibilidad y usabilidad
	RNF-003	Protección de datos	Seguridad
	RNF-004	Seguridad de credenciales	Seguridad

Funcionalidades incluidas en el MVP			
	RNF-006	Capacidad para múltiples usuarios sin detrimento de rendimiento	Rendimiento
	RNF-009	Diseño modular	Escalabilidad

Las características que quedan fuera del alcance del MVP son las clasificadas como Could (C) y Won't (W). No son esenciales para el funcionamiento inicial del sistema, aunque pueden considerarse para implementar en futuras iteraciones.

Funcionalidades fuera del alcance del MVP			
Módulo	ID	Funcionalidad	Proceso Asociado
Gestión avanzada de proyectos	RF-006	Asignación de proyectos a estudiantes en específico	Gestión de asignaciones
	RF-007	Actualización del estado de avance del proyecto por parte del profesor	Gestión de asignaciones
	RF-009	Registro de avances en proyectos (como entregas intermedias) por parte de estudiantes	Gestión de asignaciones
Mejoras en la experiencia del usuario	RNF-005	Respuesta rápida del sistema en conexiones estándar	Rendimiento
	RNF-007	Notificaciones relevantes de eventos	Notificaciones
	RNF-008	Registro de actividades	Registro de actividades
	RF-015	Consulta de progresos previos	Registro de actividades
Diseño y personalización	RF-014	Personalizar ícono de perfil	Archivos adjuntos
	RNF-010	Diseño visual limpio	Diseño
	RNF-011	Paleta de colores profesional, elementos llamativos	Diseño

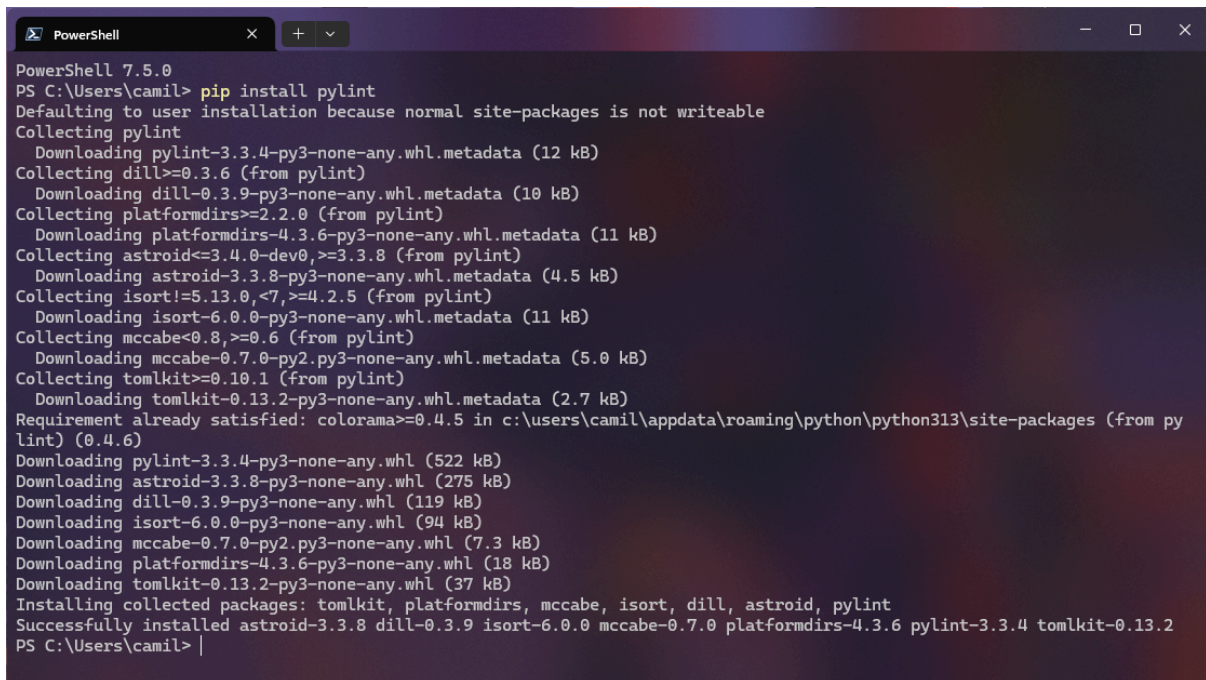
Punto 7: Clean Code

a) Backend: Implementación de PyLint y Black:

El código backend del proyecto está implementado principalmente con Python y FastAPI. Se ha determinado utilizar dos herramientas para el análisis estático de código y la automatización de formato de código: PyLint y Black, respectivamente.

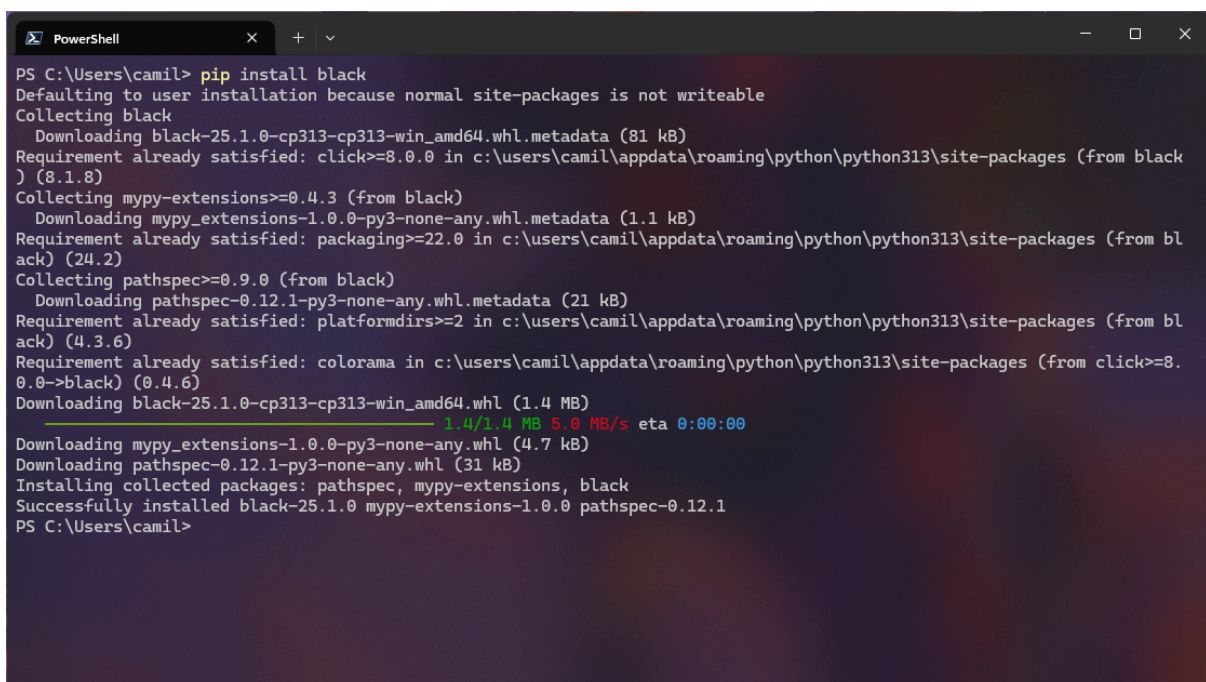
Para empezar la implementación de las herramientas mencionadas, se procede a instalar sus correspondientes paquetes, utilizando el gestor de paquetes pip:

- *Instalación de PyLint:*



```
PowerShell 7.5.0
PS C:\Users\camil> pip install pylint
Defaulting to user installation because normal site-packages is not writeable
Collecting pylint
  Downloading pylint-3.3.4-py3-none-any.whl.metadata (12 kB)
Collecting dill>=0.3.6 (from pylint)
  Downloading dill-0.3.9-py3-none-any.whl.metadata (10 kB)
Collecting platformdirs>=2.2.0 (from pylint)
  Downloading platformdirs-4.3.6-py3-none-any.whl.metadata (11 kB)
Collecting astroid<=3.4.0-dev0,>=3.3.8 (from pylint)
  Downloading astroid-3.3.8-py3-none-any.whl.metadata (4.5 kB)
Collecting isort!=5.13.0,<7,>=4.2.5 (from pylint)
  Downloading isort-6.0.0-py3-none-any.whl.metadata (11 kB)
Collecting mccabe<0.8,>=0.6 (from pylint)
  Downloading mccabe-0.7.0-py2.py3-none-any.whl.metadata (5.0 kB)
Collecting tomlkit>=0.10.1 (from pylint)
  Downloading tomlkit-0.13.2-py3-none-any.whl.metadata (2.7 kB)
Requirement already satisfied: colorama>=0.4.5 in c:\users\camil\appdata\roaming\python\python313\site-packages (from pylint) (0.4.6)
Downloading pylint-3.3.4-py3-none-any.whl (522 kB)
Downloading astroid-3.3.8-py3-none-any.whl (275 kB)
Downloading dill-0.3.9-py3-none-any.whl (119 kB)
Downloading isort-6.0.0-py3-none-any.whl (94 kB)
Downloading mccabe-0.7.0-py2.py3-none-any.whl (7.3 kB)
Downloading platformdirs-4.3.6-py3-none-any.whl (18 kB)
Downloading tomlkit-0.13.2-py3-none-any.whl (37 kB)
Installing collected packages: tomlkit, platformdirs, mccabe, isort, dill, astroid, pylint
Successfully installed astroid-3.3.8 dill-0.3.9 isort-6.0.0 mccabe-0.7.0 platformdirs-4.3.6 pylint-3.3.4 tomlkit-0.13.2
PS C:\Users\camil> |
```

- *Instalación de Black:*

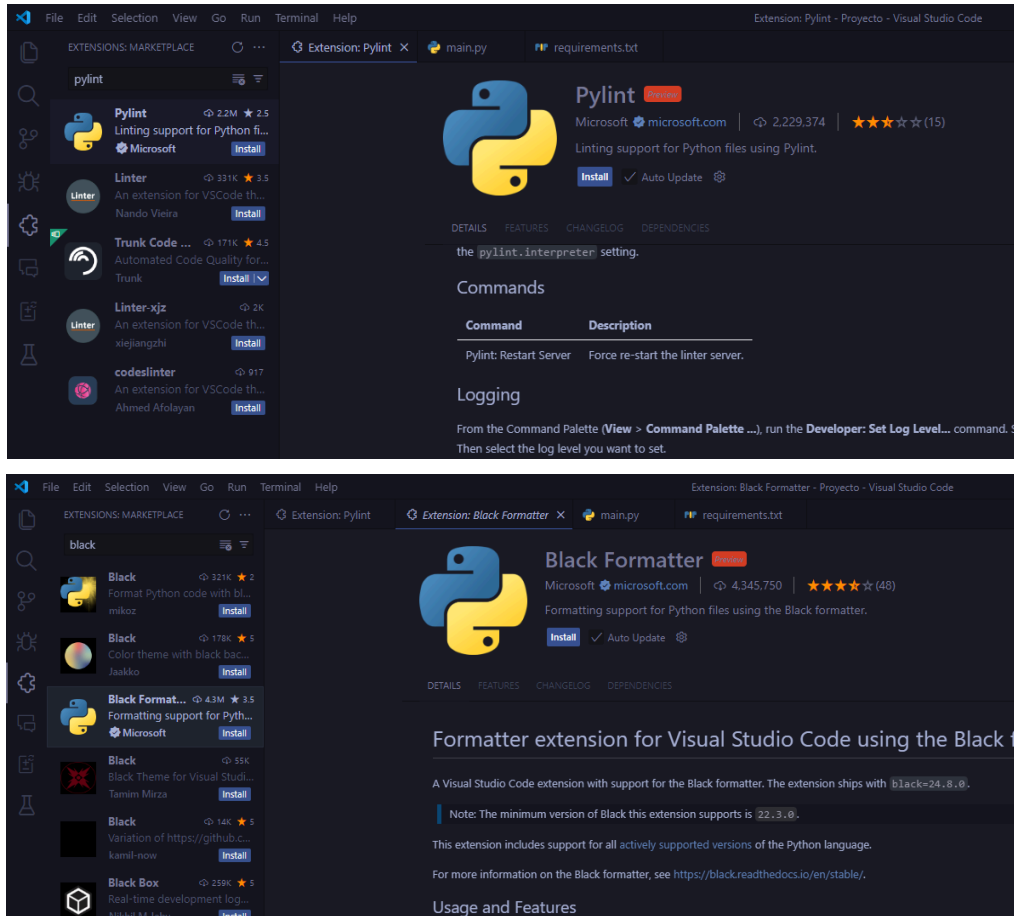


```
PowerShell 7.5.0
PS C:\Users\camil> pip install black
Defaulting to user installation because normal site-packages is not writeable
Collecting black
  Downloading black-25.1.0-cp313-cp313-win_amd64.whl.metadata (81 kB)
Requirement already satisfied: click>=8.0.0 in c:\users\camil\appdata\roaming\python\python313\site-packages (from black) (8.1.8)
Collecting mypy_extensions>=0.4.3 (from black)
  Downloading mypy_extensions-1.0.0-py3-none-any.whl.metadata (1.1 kB)
Requirement already satisfied: packaging>=22.0 in c:\users\camil\appdata\roaming\python\python313\site-packages (from black) (24.2)
Collecting pathspec>=0.9.0 (from black)
  Downloading pathspec-0.12.1-py3-none-any.whl.metadata (21 kB)
Requirement already satisfied: platformdirs>=2 in c:\users\camil\appdata\roaming\python\python313\site-packages (from black) (4.3.6)
Requirement already satisfied: colorama in c:\users\camil\appdata\roaming\python\python313\site-packages (from click>=8.0.0->black) (0.4.6)
Downloading black-25.1.0-cp313-cp313-win_amd64.whl (1.4 MB)
   1.4/1.4 MB 5.0 MB/s eta 0:00:00
Downloading mypy_extensions-1.0.0-py3-none-any.whl (4.7 kB)
Downloading pathspec-0.12.1-py3-none-any.whl (31 kB)
Installing collected packages: pathspec, mypy_extensions, black
Successfully installed black-25.1.0 mypy_extensions-1.0.0 pathspec-0.12.1
PS C:\Users\camil>
```

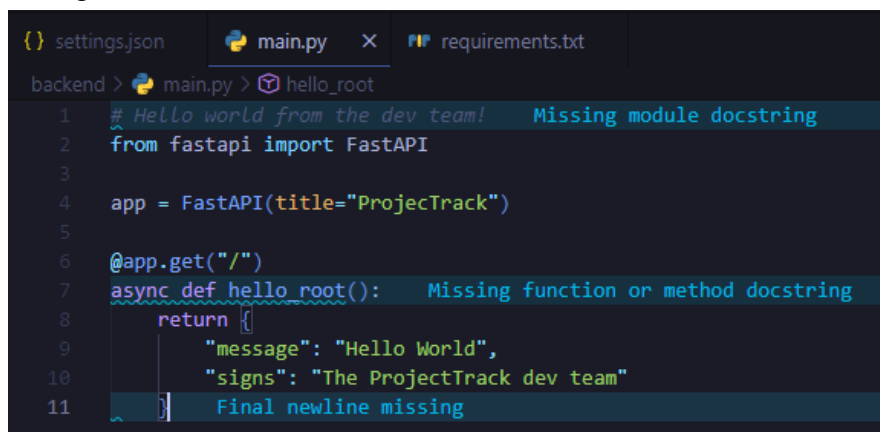
Así, los paquetes ya se encuentran instalados en el sistema. (Sin embargo, fueron instalados en el entorno global, práctica que no suele ser recomendable al trabajar con Python).

El editor de texto/software de desarrollo a utilizar en este proyecto es Microsoft Visual Studio Code. Este software, para mayor conveniencia, cuenta con extensiones que integran funcionalidades tanto de PyLint como de Black dentro del programa, más allá de invocar dichos paquetes dentro de una terminal. Para instalar las funcionalidades, se entra a la sección de marketplace de VSCode y se instalan las siguientes extensiones:

- Instalación de Extensiones en VSCode.



La extensión PyLint VSCode una vez instalada, ofrece al instante sugerencias sobre buenas prácticas de código, acorde a los estándares PEP8.



La activación del formato automático de Black Formatter en sí depende de configurar la extensión en el editor. Esto se realiza al configurar el archivo **settings.json** de la sesión. A su vez, se han revisado algunas configuraciones relevantes de ese archivo para la práctica de Clean Code.

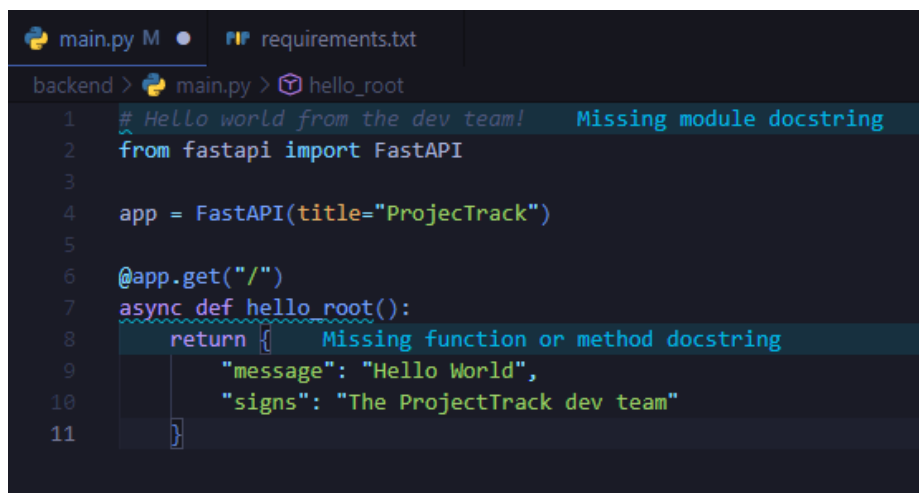
```

1 {
2   "editor.formatOnSave": true,           // Automatically formats the code when saving a file
3   "editor.tabSize": 4,                  // Defines the number of spaces equivalent to a tab press
4
5   "files.trimTrailingWhitespace": true, // Removes unnecessary spaces at the end of lines
6   "files.trimFinalNewlines": true,      // Removes extra newlines at the end of files
7   "files.insertFinalNewline": true,     // Ensures there is one final newline at the end of files
8
9   "python.analysis.extraPaths": [       // Adds more Python paths for module resolution
10    "./backend"
11  ],
12  "[python]": {                          // Overrides editor settings specifically for Python files, such as the default formatter
13    "editor.defaultFormatter": "ms-python.black-formatter",
14    "editor.formatOnSave": true
15  }
16 }

```

Una vez, esté configurado el editor, se puede comprobar el formato automático al guardar:

- **Antes de guardar:**

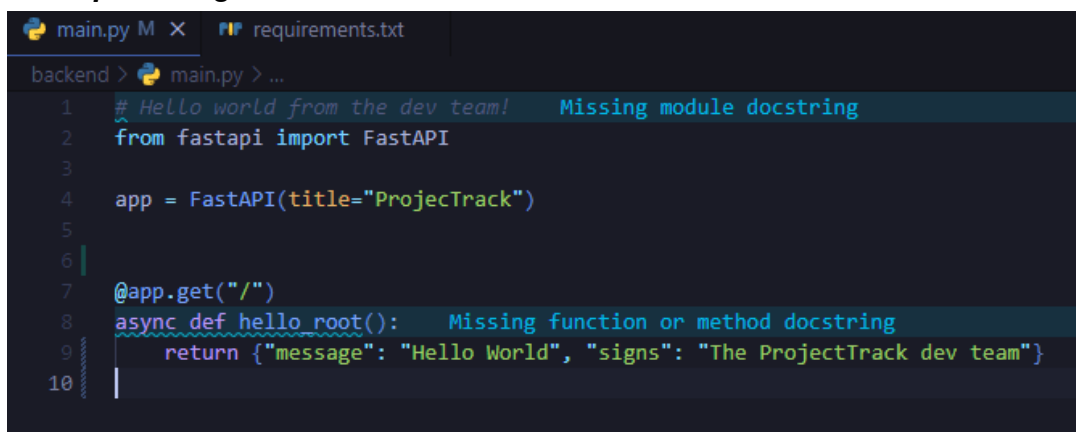


```

main.py M requirements.txt
backend > main.py > hello_root
1 # Hello world from the dev team! Missing module docstring
2 from fastapi import FastAPI
3
4 app = FastAPI(title="ProjecTrack")
5
6 @app.get("/")
7 async def hello_root():
8     return { Missing function or method docstring
9         "message": "Hello World",
10        "signs": "The ProjectTrack dev team"
11    }

```

- **Después de guardar:**



```

main.py M X requirements.txt
backend > main.py > ...
1 # Hello world from the dev team! Missing module docstring
2 from fastapi import FastAPI
3
4 app = FastAPI(title="ProjecTrack")
5
6
7 @app.get("/")
8 async def hello_root(): Missing function or method docstring
9     return {"message": "Hello World", "signs": "The ProjectTrack dev team"}
10

```

Al guardar, el formato del documento cambia de forma automática. Para mejorar aún más las prácticas implementadas el código que se está trabajando, se procede a realizar acciones acorde a las sugerencias de Pylint, basados en los estándares de Clean Code. Black Formatter es capaz de corregir automáticamente algunas, pero no todas, las sugerencias que Pylint propone.

En este caso, hacían falta los docstrings (strings de documentación) del módulo y de la función **hello_root()** en sí. Una vez añadidos los docstring, el formato final y más adecuado de este holamundo de ejemplo se ve así:



```

1  """ Hello world from the dev team! """
2
3  from fastapi import FastAPI
4
5  app = FastAPI(title="ProjecTrack")
6
7
8  @app.get("/")
9  async def hello_root():
10     """Hello world line!"""
11     return {"message": "Hello World", "signs": "The ProjectTrack dev team"}
12

```

Dependiendo de las necesidades de formato y seguimiento de estándares, se pueden desactivar advertencias específicas de PyLint, también configurando el archivo **settings.json**:



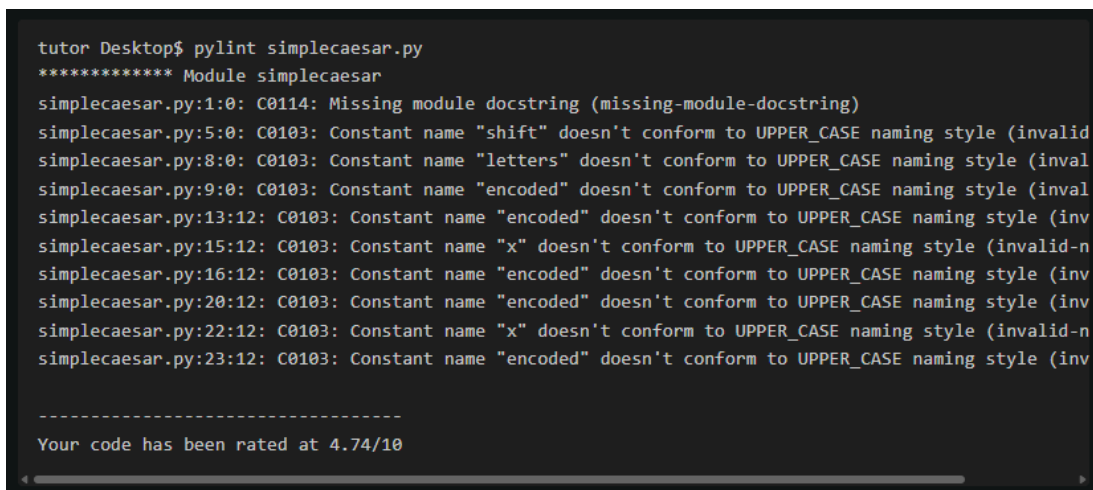
```

1  {
2      "[python]": {
3          "editor.defaultFormatter": "ms-python.black-formatter",
4          "editor.formatOnSave": true
5      },
6      "pylint.args": [
7          "--disable=E0307" // Ignore list
8      ]
9  }

```

b) Backend: informes de Clean Code utilizando PyLint:

All ejecutar un informe con PyLint en un archivo o módulo, desde la línea de comandos, el output de dicho análisis presenta información similar a las sugerencias que la extensión de PyLint presenta dentro del editor de código, además de la calificación que el módulo da al programa basado en sus convenciones.



```

tutor Desktop$ pylint simplecaesar.py
***** Module simplecaesar
simplecaesar.py:1:0: C0114: Missing module docstring (missing-module-docstring)
simplecaesar.py:5:0: C0103: Constant name "shift" doesn't conform to UPPER_CASE naming style (invalid
simplecaesar.py:8:0: C0103: Constant name "letters" doesn't conform to UPPER_CASE naming style (inval
simplecaesar.py:9:0: C0103: Constant name "encoded" doesn't conform to UPPER_CASE naming style (inval
simplecaesar.py:13:12: C0103: Constant name "encoded" doesn't conform to UPPER_CASE naming style (inv
simplecaesar.py:15:12: C0103: Constant name "x" doesn't conform to UPPER_CASE naming style (invalid-n
simplecaesar.py:16:12: C0103: Constant name "encoded" doesn't conform to UPPER_CASE naming style (inv
simplecaesar.py:20:12: C0103: Constant name "encoded" doesn't conform to UPPER_CASE naming style (inv
simplecaesar.py:22:12: C0103: Constant name "x" doesn't conform to UPPER_CASE naming style (invalid-n
simplecaesar.py:23:12: C0103: Constant name "encoded" doesn't conform to UPPER_CASE naming style (inv

-----
Your code has been rated at 4.74/10

```

Es posible exportar este informe a un archivo de texto, utilizando el siguiente comando (en este caso, se realiza la ejecución de PyLint en toda la carpeta de backend):

```

(.venv) PS D:\Datos\Documentos\Google Drive\Universidad Nacional\9- 2024-2\Ingeniería de Software I\Proyecto\Repositorio\Proyecto> pylint backend > pylint_report.txt
  
```

La salida no se efectuará en la terminal, sino que en el archivo indicado. Al ser ejecutado en un archivo ya mejorado en cuanto a convenciones se refiere, el archivo de texto generado tiene el siguiente contenido:

```

1 |
2 | -----
3 | Your code has been rated at 10.00/10 (previous run: 0.00/10, +10.00)
4 |
5 |
  
```

Ejecutando PyLint en un archivo con problemas de convención, arroja el siguiente resultado.

```

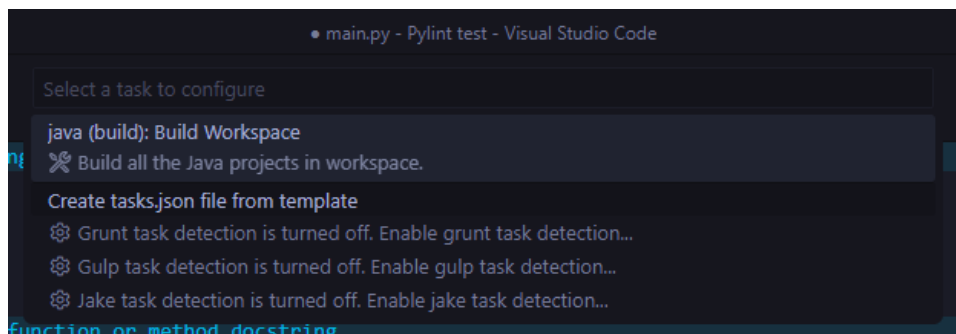
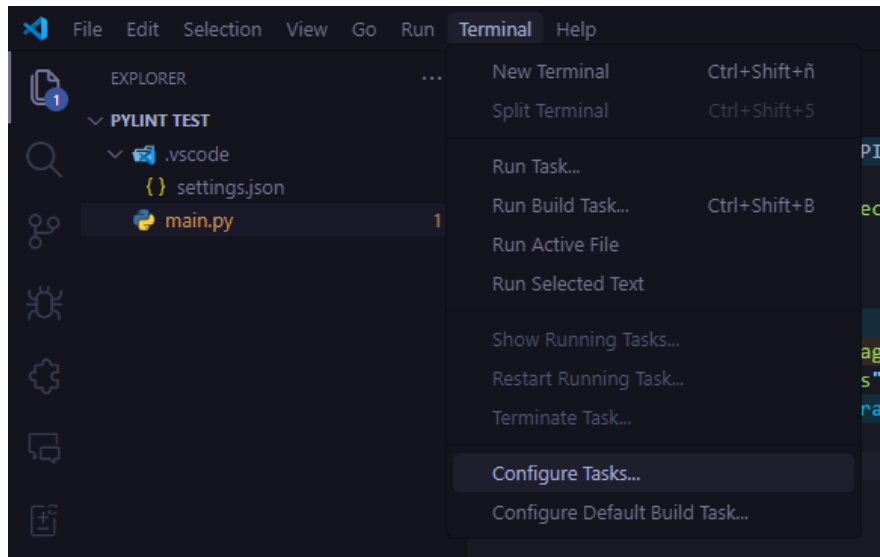
1 | ***** Module main
2 | backend\main.py:1:0: C0114: Missing module docstring (missing-module-docstring)
3 | backend\main.py:7:0: C0116: Missing function or method docstring (missing-function-docstring)
4 |
5 | -----
6 | Your code has been rated at 5.00/10
7 |
8 |
  
```

c) Backend: automatizando informes de Clean Code con PyLint:

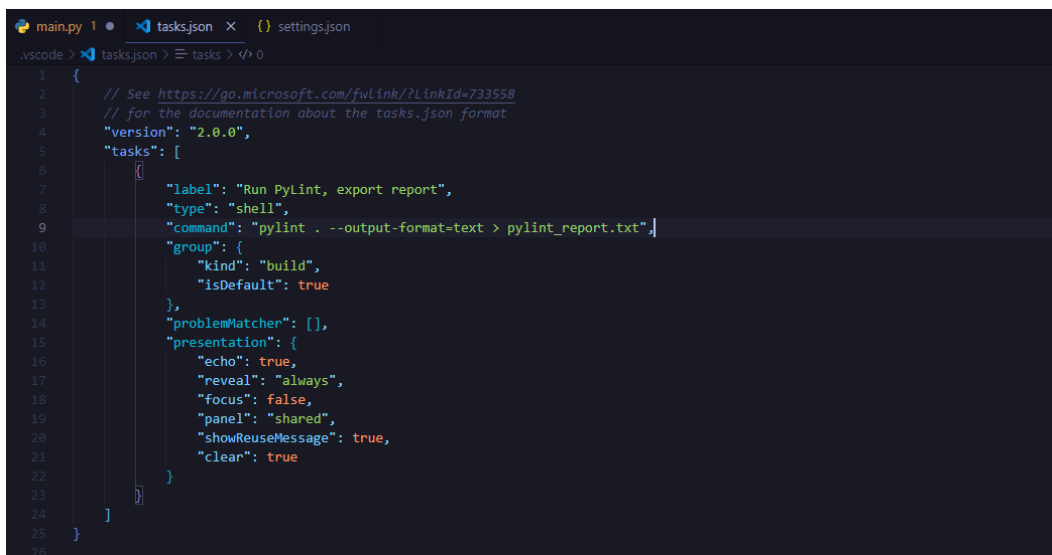
Es posible automatizar esta generación de informes de código limpio a la hora de ejecutar o guardar algún, algunos, o todos los archivos y módulos que se requieran, según las necesidades.

Para ello, en Visual Studio Code, se requiere utilizar la funcionalidad de Tasks, la cual permite, dentro del programa, la ejecución de herramientas y scripts que, generalmente, se ejecutan manualmente desde la terminal.

Para ello, en el menú de VSCode se selecciona la opción de **Terminal -> Configurar tareas... -> Crear archivo tasks.json desde plantilla -> Otros**. Con esto, se abre o crea el archivo **.vscode/tasks.json**, necesario para la automatización.



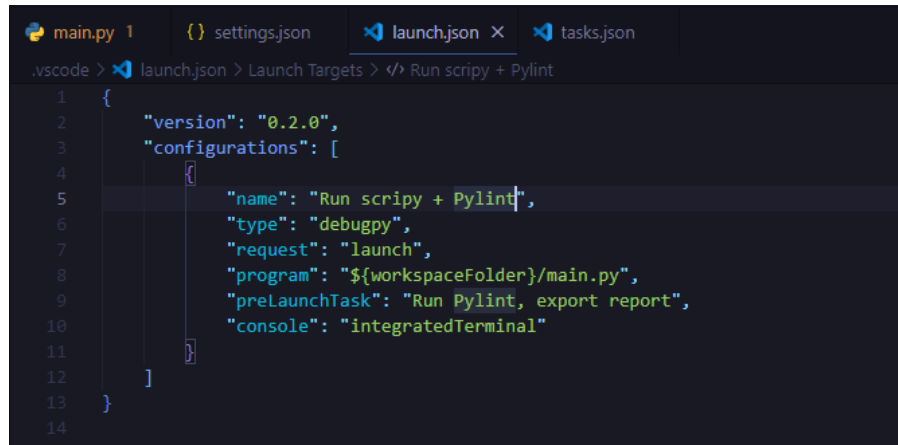
Se abrirá el archivo `.vscode/tasks.json`; si no existe, será creado dentro de dicha carpeta de configuración de usuario. En el archivo `tasks`, se añade la siguiente tarea:



Lo que hará esta tarea es que, cada que se ejecuta algún script de Python, se ejecutará PyLint y analizará dicho script; posteriormente, guardará el reporte realizado en un archivo llamado `pylint_report.txt`. Así, el reporte se guardará de manera automática cada vez que se ejecute el script deseado.

Por defecto, las tareas de Visual Studio Code se ejecutan manualmente, mediante la paleta de comandos (`Ctrl + Shift + P`), pero es posible configurarlas para que se ejecuten automáticamente al iniciar el programa.

Para configurar la ejecución automática de la tarea, en la carpeta de configuración del proyecto de VSCode, `.vscode`, hay que abrir (o crear) el archivo `launch.json`. En él, hay que agregar la siguiente configuración:

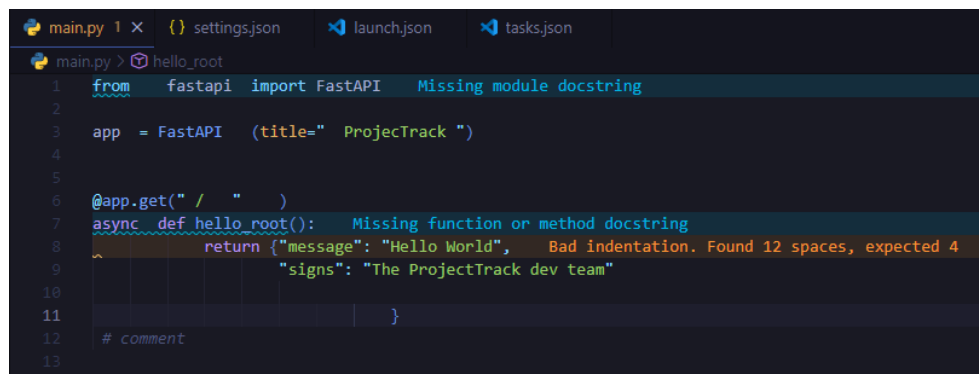


```

1 {
2   "version": "0.2.0",
3   "configurations": [
4     {
5       "name": "Run scripy + Pylint",
6       "type": "debugpy",
7       "request": "launch",
8       "program": "${workspaceFolder}/main.py",
9       "preLaunchTask": "Run Pylint, export report",
10      "console": "integratedTerminal"
11    }
12  ]
13 }
14

```

Así, cada vez que se corra el script con el comando `Ctrl + Shift + B` de Run Build Task, el script será ejecutado, y un reporte de Clean Code en formato txt será exportado a la carpeta del script ejecutado. A continuación, un ejemplo demostrativo, utilizando el siguiente código de Holamundo en FastAPI, intencionalmente desorganizado:



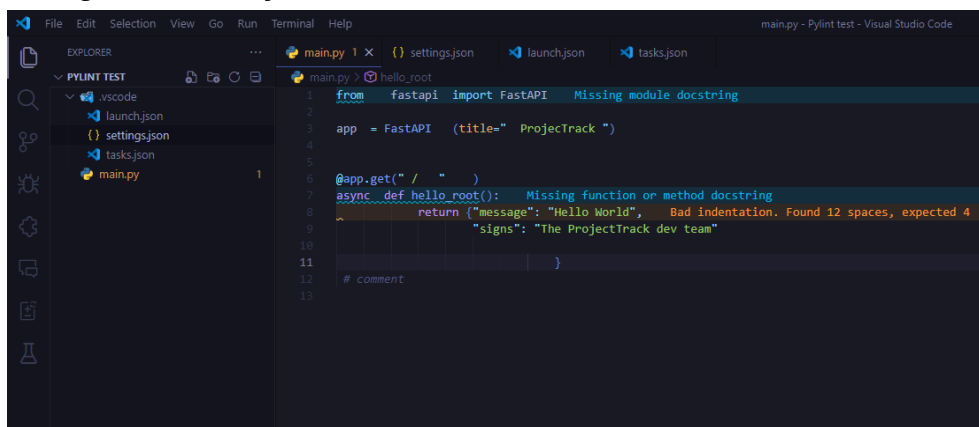
```

1 from fastapi import FastAPI Missing module docstring
2
3 app = FastAPI (title=" ProjectTrack ")
4
5
6 @app.get("/ ")
7 async def hello_root(): Missing function or method docstring
8     return {"message": "Hello World", Bad indentation. Found 12 spaces, expected 4
9         "signs": "The ProjectTrack dev team"
10
11
12 # comment
13

```

Es posible observar que, la extensión propia del linter ya señala las acciones a tomar para corregir el código, idénticas a lo que generaría ejecutar PyLint en la línea de comandos. Ejecutando un Build Task con `Ctrl + Shift + B`, un archivo txt con el reporte generado aparece en la carpeta del archivo ejecutado:

- Código antes de ejecutar:

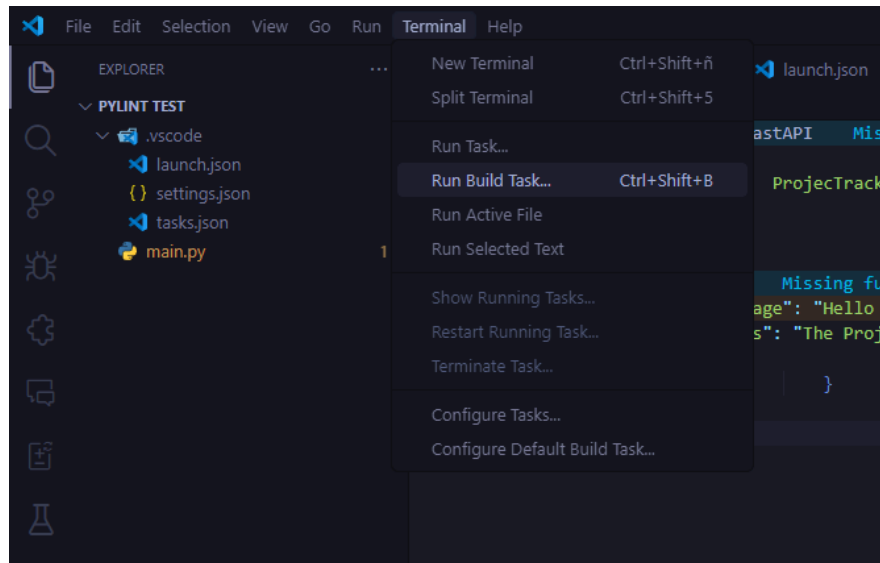


```

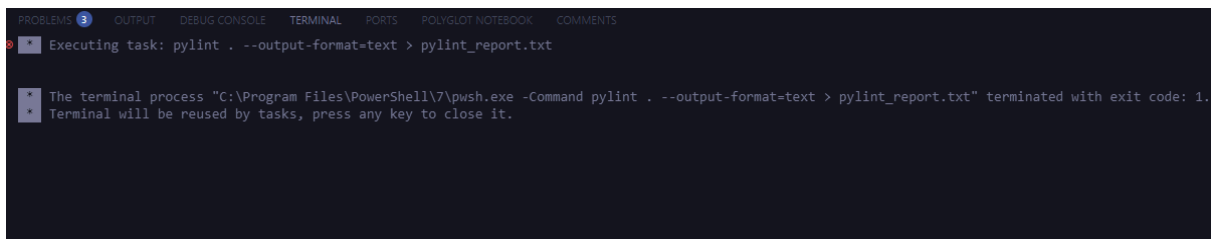
1 from fastapi import FastAPI Missing module docstring
2
3 app = FastAPI (title=" ProjectTrack ")
4
5
6 @app.get("/ ")
7 async def hello_root(): Missing function or method docstring
8     return {"message": "Hello World", Bad indentation. Found 12 spaces, expected 4
9         "signs": "The ProjectTrack dev team"
10
11
12 # comment
13

```

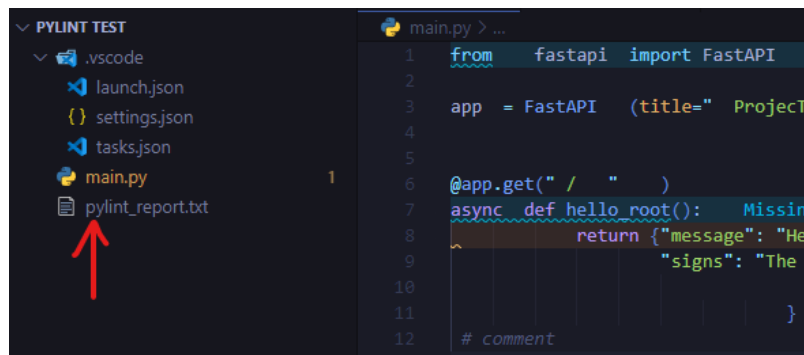
- Corriendo el Run Build Task:



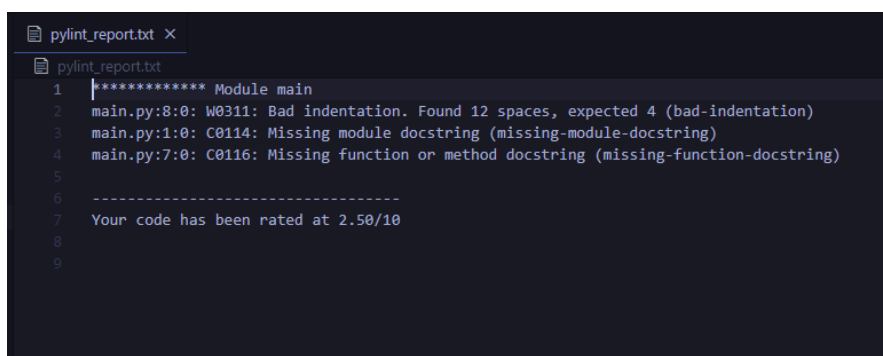
- Se ejecuta el Task en el terminal:



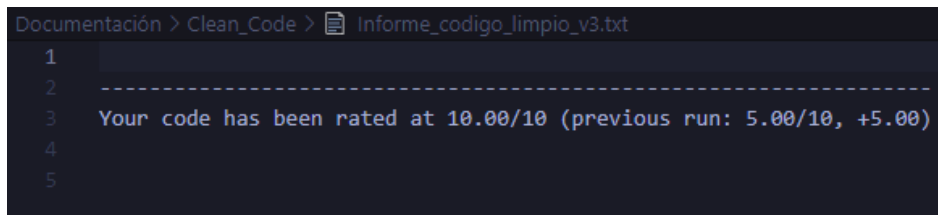
- Se crea el reporte de Clean Code:



Revisando el archivo de texto, este corresponde completamente al de una ejecución de PyLint en la línea de comandos, incluyendo el puntaje que le otorga al código, basado en sus convenciones:



Finalmente, siguiendo las sugerencias del Pylint y, habilitando la opción de formato automático de Black, el código queda limpio como en el ejemplo del inciso b, y el nuevo reporte de clean code resulta siguiente manera, con un score actualizado:



```
Documentación > Clean_Code > Informe_codigo_limpio_v3.txt
1
2 -----
3 Your code has been rated at 10.00/10 (previous run: 5.00/10, +5.00)
4
5
```

Si se quisiera exportar el informe en otro formato, como JSON o HTML, basta con cambiar el campo de “command” al crear el Task de Visual Studio Code, como por ejemplo:

- Exportar JSON: `pylint . --output-format=json > pylint_report.json`
- Exportar HTML: `pylint . --output-format=html > pylint_report.html`

d) Narración sobre el código entre integrantes:

Camilo Herrera: A pesar de ser una etapa temprana de codificación, me gusta cómo se van viendo las cosas en temas de código. Los primeros commits pueden ser retadores de leer, al ver cambios en código que uno mismo pone, y utilizarse tecnologías nuevas para mí (como pueden ser React, Vite, FastAPI y MongoDB, en caso de usarse), sin embargo, en este caso han sido llevaderos; hay claridad en qué hace cada sección, es completamente legible y para mí, fácilmente expandible.

Más allá de cualquier tema de formato, el cual herramientas como las convenciones y los linters ayudan a resolver, veo con buenos ojos cómo está encaminado el código para el proyecto

Punto 8: Diseño y Arquitectura

a) Arquitectura utilizada en el proyecto:

Para este proyecto, se ha optado por utilizar una arquitectura de tipo Cliente-Servidor. Esta arquitectura permite separar el proyecto en frontend y backend, facilitando futuras expansiones de características, o cambios en las tecnologías utilizadas dentro del proyecto. Además, la API puede ser consumida por otros tipos de clientes que se puedan implementar a futuro, por ejemplo, una app para móviles.

Dado el alcance del proyecto, además del equipo disponible, se ha optado por utilizar un monolito modular bien estructurado para la arquitectura del proyecto. La arquitectura monolítica requiere menos complejidad en el desarrollo inicial, al no requerir comunicación entre múltiples microservicios; esto facilita el trabajo a equipos pequeños y con experiencia limitada como el nuestro, ya que permite que los desarrolladores trabajen en una sola base de código, más allá de la implementación del control de versiones con Git.

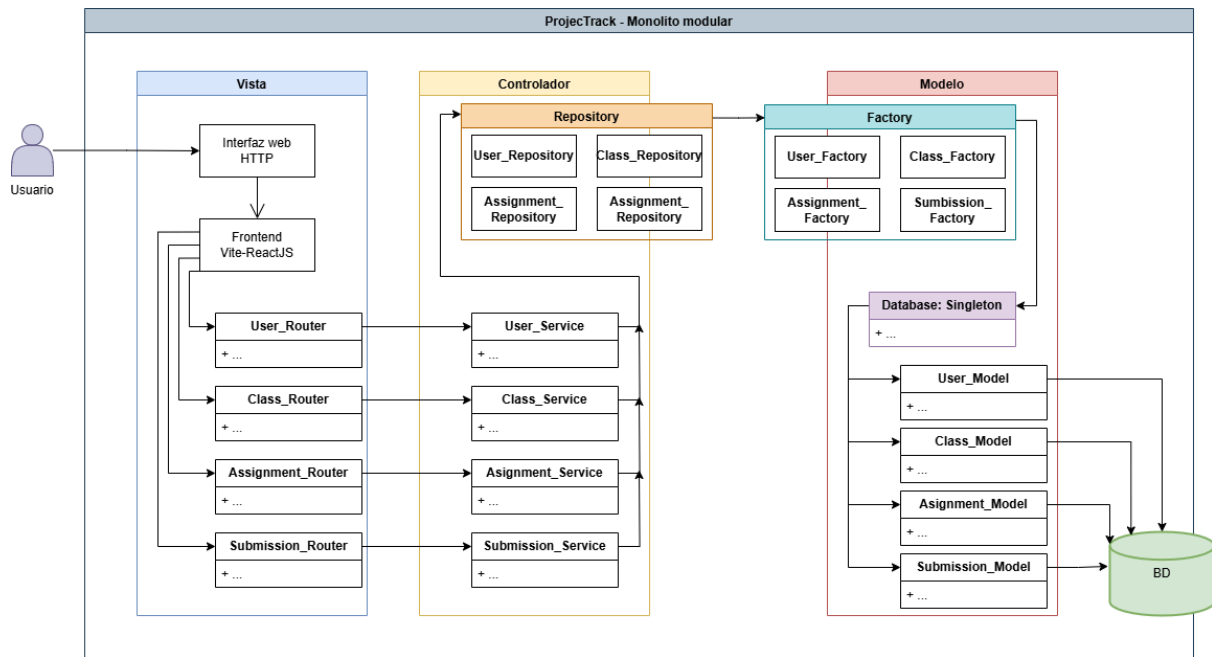
El Backend, encargado de procesar la lógica de negocio y gestionar la API RESTFUL, está planteado para implementarse en FastAPI, utilizando un patrón de diseño de Modelo-Vista-Controlador (MVC). A grandes rasgos, se utilizará MVC teniendo en cuenta su clara separación de responsabilidades dentro del backend del proyecto, así:

- Los Modelos (M) definen la estructura de los datos, a través de la base de datos SQL (PostgreSQL o SQLAlchemy).
- Las Vistas (V) equivalen a las rutas o direcciones de la API; estas controlan el acceso a los datos.
- Los Controladores (C) se tratan de servicios, los cuales procesan la lógica de negocio.

En la sección de Patrones de Diseño, se explicará más a detalle la decisión de implementar MVC en la arquitectura del proyecto.

En cuanto al Frontend, se plantea el uso de React con Vite, capaces de proporcionar una interfaz de usuario que facilita la experiencia de usuario, a su vez que cuenta con componentes reutilizables. Además, se integra bien con la API REST del backend, y, de ser necesario permite gestiones globales con Redux (contenedor de estado de aplicaciones) o Context API (compartiendo datos mediante una implementación producer-consumer).

Se ha planteado el siguiente diagrama para la arquitectura del sistema en el proyecto, también estará disponible en el repositorio del proyecto:



b) Diseño de la Base de Datos

Sobre el diseño del modelo Entidad-Relación para la base de datos del proyecto, se han tomado una serie de decisiones en pro de garantizar un sistema eficiente, escalable, seguro y funcional.

i) Normalización

Las formas normales en los esquemas de bases de datos previenen problemas de inconsistencia y evitan la redundancia de datos, optimizando su diseño, tomando en cuenta algunas pautas: Datos complejos se vuelven tablas más simples, facilidad de comprensión y mantenimiento, ahorro de esfuerzos posteriores al desarrollar y mantener el sistema.

Se buscó emplear la Tercera Forma Normal (3NF) para el diseño de la base de datos, ya que esta forma normal implementa ciertas pautas.

Desde la 1NF:

- Mantiene la atomicidad de los datos (valores únicos e indivisibles)
- Toda tabla mantiene una clave primaria
- No tiene valores nulos o tuplas repetidas

Desde la 2NF:

- Todo campo secundario depende en su totalidad de la clave primaria y no de una parte de ella

A partir de la 3NF:

- Cada campo que no sea la clave primaria depende solamente de la clave primaria o secundaria y no otros campos.

Así, se evita la redundancia de datos, mejorando la consistencia, además de separar las entidades en tablas específicas (usuarios, asignaciones, etc.).

ii) Claves Primarias y Foráneas

El uso de claves primarias y foráneas tiene como objetivo mantener la integridad referencial para evitar registros huérfanos. Puede facilitar consultas eficientes mediante JOINS.

Por ejemplo, la tabla *students_assignments*, que vincula proyectos asignados en un curso con estudiantes en específico, realiza esta acción utilizando claves foráneas *project_id* y *student_id*.

Si se elimina el proyecto, las asignaciones asociadas también se eliminan, manteniendo una base de datos limpia.

iii) Índices

Se utilizan generalmente para la optimización de consultas, mejorando el rendimiento de búsquedas y filtrado de datos. Así, se reduce el tiempo de respuesta en consultas con alta concurrencia.

Un ejemplo concreto planteado para la base de datos del proyecto es indexar entregas de proyectos por su fecha de entrega, utilizando el índice *submission_date_idx*.

iv) Tablas intermedias y cardinalidad

Se plantea una relación de muchos a muchos a la hora de asignar estudiantes específicos a proyectos dentro de un curso. Esto porque un estudiante puede tener varios proyectos, además que un proyecto puede ser asignado a varios estudiantes. Por ello, se plantea el uso de la tabla intermedia *students_assignments*.

c) ¿SQL o No-SQL?

En primera instancia, se había planteado utilizar MongoDB, una base de datos no relacional, para la implementación de este proyecto, pero dadas las limitaciones técnicas y de experiencia, se optó por una base de datos relacional. Dada la reducida escala del proyecto, se considera que MongoDB es una opción factible.

Sin embargo, teniendo en cuenta el alcance y las necesidades del proyecto, se ha optado por diseñar una base de datos relacional (SQL).

Las bases de datos de este tipo ofrecen unas relaciones bien definidas, necesarias para el alcance del proyecto, como las múltiples relaciones entre estudiantes, profesores, cursos y entregas.

Además, las bases de datos SQL aseguran integridad referencial a través de las claves foráneas.

El rendimiento también se tiene en cuenta, al usar índices y JOINS para optimizar consultas, y dejando la posibilidad de implementar bases de datos como PostgreSQL y MySQL, capaces de escalar sin afectar el rendimiento.

Punto 9: Patrones de diseño

a) Patrones de diseño dentro del proyecto:

i) *Modelo-Vista-Controlador*

Se ha planteado seguir una arquitectura Cliente-Servidor para este proyecto. Debido a esto, para el backend del proyecto se ha decidido utilizar el patrón de diseño de Modelo-Vista-Controlador con el framework FastAPI.

Un potencial problema que resuelve es que la lógica de negocio y la gestión de los datos estén mezcladas con la API; el código del backend se organiza en capas separadas, mejorando la modularidad y escalabilidad del backend.

Su uso se justifica al facilitar la separación entre los Modelos (BD), Controladores (API REST) y Vistas (JSON response); esto permite que se trabaje en módulos independientes sin afectar otras partes del código en conjunto.

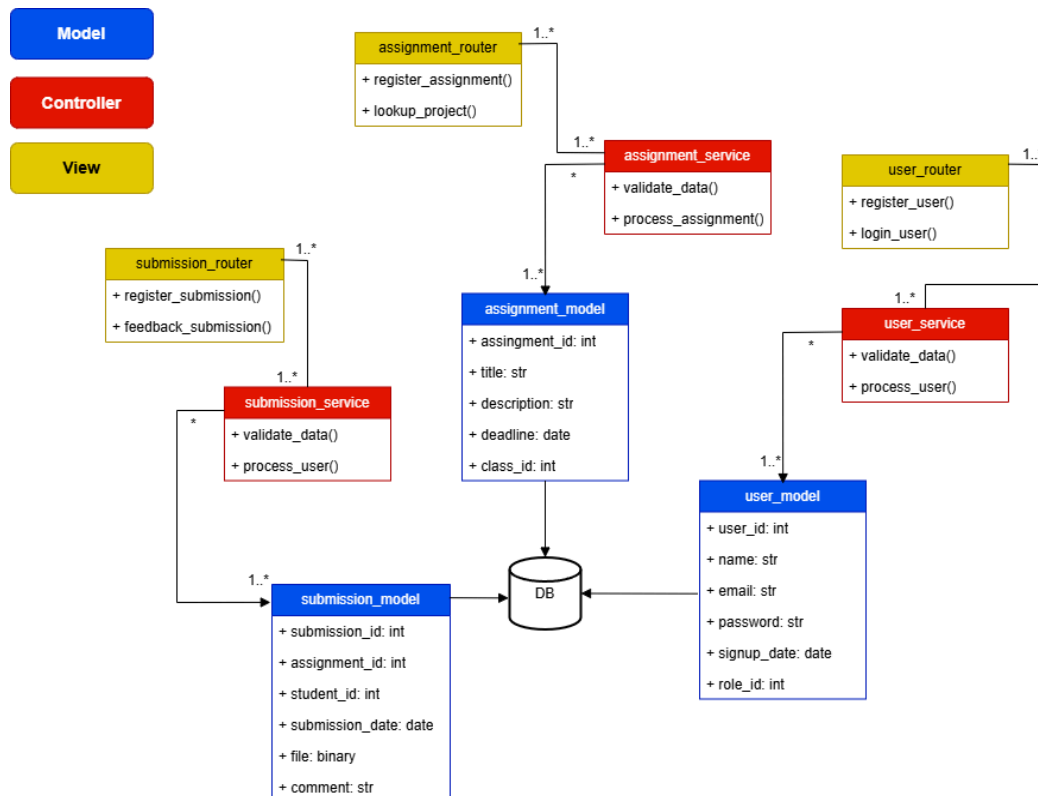
Su implementación diferencia el backend en “módulos”, de la siguiente manera:

- *models/*: Modelos; definen la estructura de los datos en la BD.
- *routes/*: Vistas; definen los endpoints de la API REST que usa el front end/clientes.
- *services/*: Servicios; procesan la lógica de negocio, separada de los controladores.

Otros módulos factibles que se pueden implementar; relacionados con otros patrones:

- *schemas/*: Esquemas; validan datos de entrada y salida con Pydantic.
- *repository/*: Repositorio; separa las consultas de la BD con la lógica de negocio.
- *auth/*: Manejo de seguridad.
- *utils/*: Modularidad; funciones auxiliares reutilizables para varias partes del sistema.

- Diagrama UML:



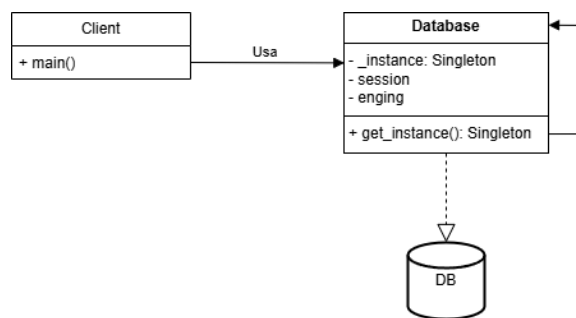
ii) Singleton

Este modelo es idóneo para gestionar la conexión con la base de datos. Evita crear conexiones múltiples y redundantes a la BD en cada solicitud realizada, gestionando una instancia única de conexión, mejorando así el rendimiento y asegurando la integridad de los datos en cada transacción.

FastAPI, al manejarse de manera asíncrona, maneja peticiones simultáneas. Debido a esto, se justifica implementar Singleton para optimizar la gestión de conexiones con la base de datos, crucial para el funcionamiento correcto del framework.

Se implementa en un archivo *database.py* para que la conexión con la BD sea única en toda la aplicación.

- Diagrama UML:



iii) Repository

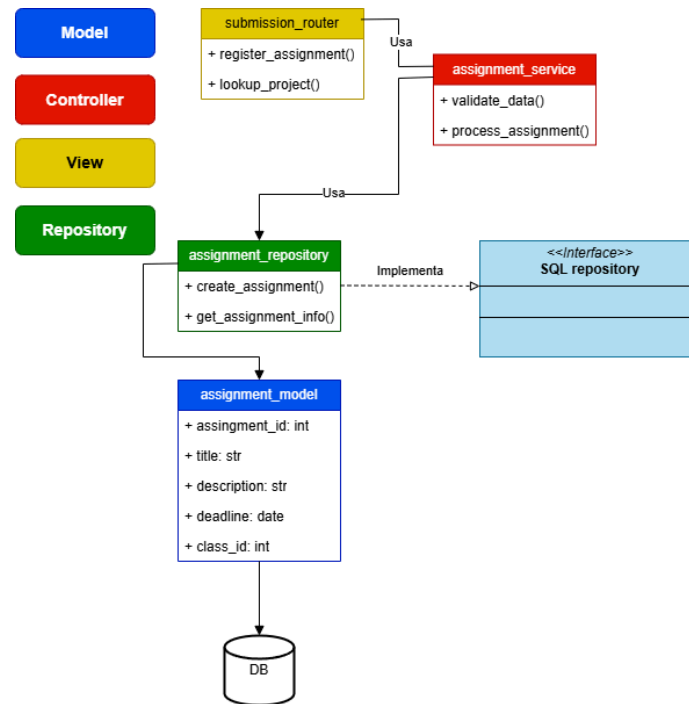
Maneja el acceso a la base de datos, evitando que la lógica de negocio acceda directamente a esta, manteniendo su necesaria separación de funciones; las consultas SQL son separadas a una capa independiente.

Este patrón de diseño permite la reutilización de código, evitando duplicaciones en las consultas, además, facilita modificaciones en la estructura de la BD, manteniendo intacta la lógica de negocio.

Se implementa a través del módulo *repository/*, para manejar las consultas con los diferentes actores.

- Diagrama UML:

[sig. pág.]



iv) Factory Method

Es necesario crear múltiples instancias de objetos de *proyecto*, *usuario* (independiente del rol), *curso* y *entrega* sin repetir código; este patrón de diseño permite crear objetos complejos de manera estructurada, sin repetir lógica, además, permite que los objetos sean modificables a futuro, de forma sencilla.

Para implementarlo, se crean clases *xFactory* en el módulo *services* (lógica de negocio), las cuales generan instancias de la clase *x* a partir de parámetros establecidos. Para usarse, se hace a través del CRUD del Repositorio, para crear instancias sin repetir la lógica. En FastAPI, se integra en los endpoints para gestionar las solicitudes de creación.

- Diagrama UML:

