Desarrollo del Perfil del estudiante dentro de No estás solo

Federico Melo Barrero[1]

5 de enero de 2025

Resumen

El presente proyecto de grado describe el desarrollo del Perfil del estudiante al interior de la plataforma No estás solo. El Perfil del estudiante consiste en un conjunto de visualizaciones interactivas que presentan información académica y contextual sobre los estudiantes de la Universidad de los Andes. Originalmente dirigido a profesores en su rol como consejeros, su uso se extendió a estudiantes, coordinadores y administrativos, quienes encontraron valiosa la información para procesos de consejería y toma de decisiones. El documento relata el contexto que suscitó la creación del Perfil y los objetivos asociados a la misma; además, detalla el desarrollo técnico del módulo, incluyendo las capas de datos, lógica y presentación. Se describen las tecnologías empleadas, las prácticas de ingeniería de software implementadas y los principios de claridad y relevancia para la exposición de la información que fueron tenidos en cuenta, para finalizar con los resultados obtenidos.

Abstract

This thesis project presents the development of the Student Profile within the *No estás solo* platform. The Student Profile is a set of interactive visualizations that display academic and contextual information about students at Universidad de los Andes. Initially designed for professors in their role as advisors, its use expanded to students, coordinators, and administrative staff, who found the information valuable for counseling processes and decision-making. The document outlines the context that motivated the creation of the Student Profile and the objectives set for its development. It also details the technical aspects of the module, including the data, logic, and presentation layers. Additionally, thorough descriptions are provided of the technologies used, the software engineering practices implemented, the principles of clarity and relevance applied to information presentation, and the results achieved.

Índice general

Agradecimientos

Hay una infinidad de personas a quienes debo un agradecimiento en este proyecto. En este espacio, me gustaría destacar a algunas de ellas.

Antes que a nadie, quiero agradecer a Angélica, William y Sebastián: mi familia. Su amor y apoyo incondicional moldean mi vida y forjan mi persona. A ellos les debo todo lo que soy.

Luego, debo agradecer a mis amigos más cercanos, quienes son un pilar fundamental en mi vida y son los cimientos de mi felicidad. Especialmente, quiero agradecer, por un lado, a David, Santiago y Andrés; y por otro, a Laura, Laura, Mariana y Sarah. Sin esos dos conjuntos de personas, mi vida sería mucho más triste.

Por mera causalidad, debo agradecer también a todos quienes han invertido su tiempo, recursos y esfuerzos en mi educación. Eso incluye a todos quienes fueron partícipes de mi formación tanto en el Colegio San Carlos como en la Universidad de los Andes. Entre otros, debo agradecimientos especiales a Jaime Rueda, quien me enseñó a estudiar; a Nicolás Rincón, quien me forzó a estudiar; a José Bocanegra, quien me enseñó sobre desarrollo web y construcción de APIs; a Ruby Casallas y a Nicolás Cardozo, quienes me enseñaron sobre calidad de software; y a todos mis profesores de la Universidad.

Por último, quiero agradecer a quienes estuvieron directamente involucrados en este proyecto. A Mariana y a Nicolás, con quienes tomamos el desarrollo de No Estas Solo desde hace más de un año. A Manuel y (nuevamente) a Santiago, quienes hicieron posible el pipeline de analítica. A Oscar, que estaba siempre pendiente de todo. Y, por supuesto, a Marcela, que me dio la oportunidad de trabajar en este proyecto y me acompañó en todo el proceso.

Big Brother is watching you.

—George Orwell, *Nineteen Eighty-Four*

Capítulo 1

Introducción

Resumen del capítulo

NES es una plataforma web de la Universidad de los Andes que centraliza sus recursos y servicios. El Perfil del estudiante complementa esta plataforma al proveer información académica y socioeconómica del estudiantado, antes faltante e imprescindible para mejorar las consejerías a estudiantes y las decisiones administrativas.

1.1. Contexto

La Universidad de los Andes es una institución educativa colombiana de gran renombre, reconocida por su excelencia académica y su compromiso con la formación integral de sus estudiantes. En aras de propulsar el éxito de sus estudiantes en todo ámbito, la Universidad realiza un esfuerzo constante por ofrecer servicios y recursos que fomenten el desarrollo académico, social, personal y profesional de sus estudiantes, así como el bienestar de la comunidad universitaria en general.

1.1.1. La plataforma *No estás solo*

En los últimos dos años, la Vicedecantura de Asuntos Estudiantiles de la Universidad de los Andes identificó dos dolencias significativas concernientes al estudiantado y al profesorado, respectivamente. La primera, consiste en el desconocimiento por parte de los estudiantes de la inmensa cantidad de recursos y servicios de apoyo que la Universidad pone a su disposición en pos de su éxito académico, social, personal y profesional. La segunda, radica en la dificultad y falta de recursos que tenían los profesores a la hora de ejercer su labor como consejeros en apoyo a los estudiantes.

Como herramienta para subsanar estas dolencias, la Vicedecantura ha propulsado el desarrollo y la extensión de una aplicación web: *No estás solx* (en adelante, NES). Esta plataforma fue concebida por la Vicedecanatura de Asuntos Estudiantiles de la Facultad de Ingeniería como una solución integral para garantizar que todos los miembros de la comunidad, especialmente los estudiantes, puedan acceder de manera centralizada y clara a todos los recursos y servicios disponibles.

Desde la perspectiva del estudiantado, NES centraliza el acceso a todas las herramientas que fomentan su éxito académico, personal y profesional. Esto incluye información sobre eventos académicos y culturales, procesos y solicitudes administrativas,

y servicios de apoyo como consejerías y tutorías. Sumado a eso, facilita el acceso de cada estudiante a su red de apoyo, compuesta por profesores consejeros, coordinadores académicos y otros profesionales de la Universidad. Toda esta información previamente se encontraba dispersa en las distintas plataformas de la Universidad, lo que dificultaba su acceso y su uso conjunto. Esto suponía un reto importante para los estudiantes, quienes debían navegar entre múltiples sistemas y portales para encontrar la información que necesitaban, además de constituir una barrera de entrada significativa para los estudiantes menos experimentados.

Para los profesores y otros usuarios administrativos, NES constituye una herramienta que facilita su labor como consejeros y mejora su capacidad de toma de decisiones. La plataforma incluye una base de preguntas frecuentes y guías para poder orientar a los estudiantes en dificultades académicas, personales o financieras. Finalmente, para los coordinadores académicos, NES contribuye a descongestionar las solicitudes frecuentes mediante herramientas de autogestión, permitiendo que los estudiantes resuelvan problemas simples por sí mismos y optimizando el uso del tiempo en las coordinaciones.

En conjunto, NES busca mejorar la experiencia universitaria al simplificar procesos y fortalecer los canales de apoyo.

1.1.2. La necesidad del Perfil del estudiante

NES representa un avance significativo en la centralización y claridad de los recursos y servicios ofrecidos por la Universidad. Sin embargo, previo a la implementación del Perfil del estudiante, los profesores y administrativos enfrentaban desafíos al intentar ofrecer consejerías personalizadas o tomar decisiones informadas. La información sobre los estudiantes estaba dispersa en múltiples sistemas y formatos, lo que dificultaba tener una visión integral de su contexto académico y socioeconómico.

El Perfil del estudiante surge como una respuesta directa a esta necesidad. Este módulo se integra dentro de NES para ofrecer una herramienta que:

- Centralice y organice la información relevante sobre los estudiantes.
- Presente esta información de manera intuitiva, clara y accesible, sin sacrificar la exhaustividad.
- Facilite a profesores y administrativos brindar consejerías personalizadas y tomar decisiones basadas en datos.

Con estas características, el Perfil del estudiante no solo mejora la experiencia de los usuarios de NES, sino que también fortalece su capacidad de impacto al abordar de manera directa las problemáticas identificadas por la Vicedecanatura de Asuntos Estudiantiles.

1.2. Objetivos

El objetivo general de este proyecto es construir el Perfil del estudiante.

Más específicamente, el objetivo radica en proporcionar a todo el estudiantado y al profesorado de la Universidad de los Andes una herramienta digital que les permita

consultar información relacionada con el desempeño académico actual y pasado, así como con el contexto socioeconómico, de los estudiantes de la Universidad.

1.2.1. Objetivos específicos

El objetivo general enunciado recién se desglosa en los siguientes objetivos específicos:

- Conseguir, mediante una herramienta de software, el acceso a la información académica de los estudiantes de la Universidad de los Andes de manera que pueda interactuar con una aplicación web, idealmente siguiendo los lineamientos de un API REST.
- Diseñar e implementar un módulo de la aplicación web NES que extraiga y exhiba la información académica de cualquier estudiante de la Universidad de los Andes. Realizar la distinción entre la forma en la que se presenta la información para estudiantes de pregrado y estudiantes de posgrado, en particular, distinguir entre los resultados de un mismo estudiante en su pregrado y en su posgrado, teniendo completa trazabilidad de su trayectoria académica cuando sea el caso.
- Conectar el módulo con la navegación y funcionalidades ya existentes en la aplicación web NES, de forma que cada estudiante pueda tener acceso a su información académica (mas no a la de otros estudiantes), cada profesor consejero pueda tener acceso a la información académica de sus estudiantes aconsejados y otros usuarios específicos, determinados por la Vicedecantura, puedan tener acceso a la información académica de los estudiantes que les conciernan.
- Garantizar la integridad y seguridad de la información académica de los estudiantes de la Universidad de los Andes, tanto informáticamente como en su presentación, de forma que se cumpla con la normativa de protección de datos personales y de información académica de la Universidad.

1.2.2. Resultados esperados

El presente proyecto se puede dar por culminado una vez se satisfaga el siguiente conjunto de resultados:

- Existencia de un módulo de la aplicación web NES que permita a cada estudiante de la Universidad de los Andes, tanto de pregrado como de posgrado, consultar información relacionada con su desempeño académico.
- Existencia de un módulo de la aplicación web NES que permita a cada profesor de la Universidad de los Andes consultar información relacionada con el desempeño académico de cada uno de sus estudiantes aconsejados, de forma individual.
- Existencia de un módulo de la aplicación web NES que permita a usuarios específicos, como decanos, directores de programas u otros directivos de la Universidad de los Andes, consultar información relacionada con el desempeño académico de los estudiantes de la Universidad que les conciernan.

1.3. Usuarios y requerimientos

Describe a los usuarios principales del sistema, sus roles y las necesidades específicas identificadas. Aquí también puedes incluir cómo se realizó la recolección y análisis de requerimientos, y un resumen de las historias de usuario que guiaron el diseño del sistema.

1.4. Arquitectura general del sistema

Introduce la arquitectura three-tier (datos, lógica y presentación). Proporciona una visión general de cómo estas capas están integradas y cómo contribuyen al funcionamiento del Perfil del estudiante.

Capítulo 2

Capa de datos

El presente capítulo se ocupa de la capa de datos de la arquitectura del Perfil del estudiante. En particular, describe en detalle el proceso de extracción, transformación y carga al que los datos son sometidos. Adicionalmente, provee una perspectiva, a un más alto nivel de abstracción, de cómo los datos fluyen desde su origen hasta su destino final en el Perfil del estudiante, que resulta útil como contexto y preámbulo de los capítulos subsiguientes.

2.1. Fuentes de datos

En aras de que el Perfil del estudiante satisfaga su expectativa como núcleo de información académica y socioeconómica de los estudiantes de la Universidad de los Andes, es necesario que unifique la información que se encuentra dispersa en múltiples sistemas y formatos. En particular, se espera que el Perfil del estudiante sea capaz de reproducir, como mínimo, la información disponible en los sistemas de información académica de la Universidad, como Banner y Advise, así como información en el sistema de gestión financiera y en el portal de oferta de cursos. La tabla 2.1 detalla las fuentes de datos cuya información debe ser reproducida por el Perfil del estudiante, incluyendo su nombre formal y una breve descripción de la información que contienen.

Para el acceso a la información contenida en estas fuentes de datos, la Universidad dispone de un Ecosistema de analítica institucional, cuyo propósito es precisamente facilitar el acceso a la información y la generación de conocimiento a partir de los datos. El Ecosistema tiene entre sus objetivos específicos “Entregar los datos con estándares de calidad y seguridad para suplir las necesidades de la Universidad”, “Hacer el uso de datos para garantizar una toma de decisiones informada” e “Impulsar una cultura de uso de datos en la Universidad” [2]. El uso del Ecosistema evita tener acceder directamente a las fuentes de datos, lo que simplifica el proceso de extracción de datos y terceriza la responsabilidad de la consistencia de los mismos, al menos en su origen, a la Universidad. En la subsección 2.2.1 se detalla cómo se realiza la extracción de los datos del Ecosistema.

Como complemento a lo anterior, teniendo en mente la enorme responsabilidad que recae sobre el Ecosistema respecto a la calidad de los datos, un aspecto crítico para que el Ecosistema pueda cumplir con sus objetivos y en efecto ser útil y confiable para las diversas dependencias de la Universidad, se realizó el esfuerzo de registrar en el apéndice

Tabla 2.1
Fuentes de datos utilizadas por el Perfil del estudiante

Nombre	Nombre Formal	Descripción
Banner	Ellucian Banner	Sistema de planificación de recursos empresariales (ERP) diseñado para instituciones de educación superior. Es capaz de gestionar procesos académicos y administrativos, incluyendo matrículas, registros académicos, finanzas y recursos humanos. [3] Almacena el registro académico completo de cada uno de los estudiantes de la Universidad en todos los niveles académicos, así como algunos de los antecedentes académicos del estudiante.
Advise	Ellucian CRM Advise	Sistema de gestión de relaciones con los estudiantes (CRM) que permite a las instituciones identificar y apoyar proactivamente a estudiantes en riesgo, facilitando intervenciones personalizadas para mejorar su éxito académico. [4]
Oferta de Cursos	Portal de Oferta de Cursos	Plataforma web que permite a los estudiantes consultar los cursos disponibles en cada periodo académico, incluyendo información detallada sobre horarios y profesores. [1]
Sistema de Gestión Financiera		Representación genérica de los sistemas contables y financieros de la universidad, en los que se maneja la información relacionada con becas, créditos educativos y pagos realizados por los estudiantes, al igual que información socioeconómica del estudiante relevante para la asignación de auxilios económicos.

B todos los problemas de calidad de los datos que se han encontrado en el Ecosistema y cómo se han abordado.

2.2. Proceso de extracción, transformación y carga

La recuperación y el procesamiento de los datos se realizan mediante un *pipeline* de analítica implementado en cuadernos de Jupyter en Python, en colaboración con los estudiantes Santiago Martínez Novoa y Manuel Felipe Porras Tascón [*]. El pipeline de analítica realiza las tres etapas clásicas de un proceso de ETL: extracción, transformación y carga de los datos. A cada una de estas etapas se le dedica una subsección en las siguientes líneas.

2.2.1. Extracción de los datos

El proceso de extracción, a alto nivel, consiste de tomar la información relevante del Ecosistema de analítica institucional.

[*]Mi más sincero agradecimiento a ambos: a Manuel, en aquel momento estudiante de la Maestría en Ingeniería de la Información, por su valiosa labor al crear y trabajar inicialmente en el cuaderno; y a Santiago, entonces estudiante de Ingeniería de Sistemas y Computación, por tomar la posta, mejorar y mantener este trabajo. Sin ellos, este proyecto no habría sido posible.

La información del Ecosistema se encuentra almacenada en Azure Blob Storage, que es un servicio de almacenamiento de objetos en la nube de Microsoft Azure. Está distribuida en diversas cuentas de almacenamiento, cada una de las cuales contiene archivos en formato PARQUET y XLSX. Para el Perfil del estudiante, se hace uso de diez de esos archivos, distribuidos en dos cuentas de almacenamiento. La figura 2.1 muestra la estructura de directorios y archivos en el Blob Storage que contienen la información relevante para el Perfil del estudiante.

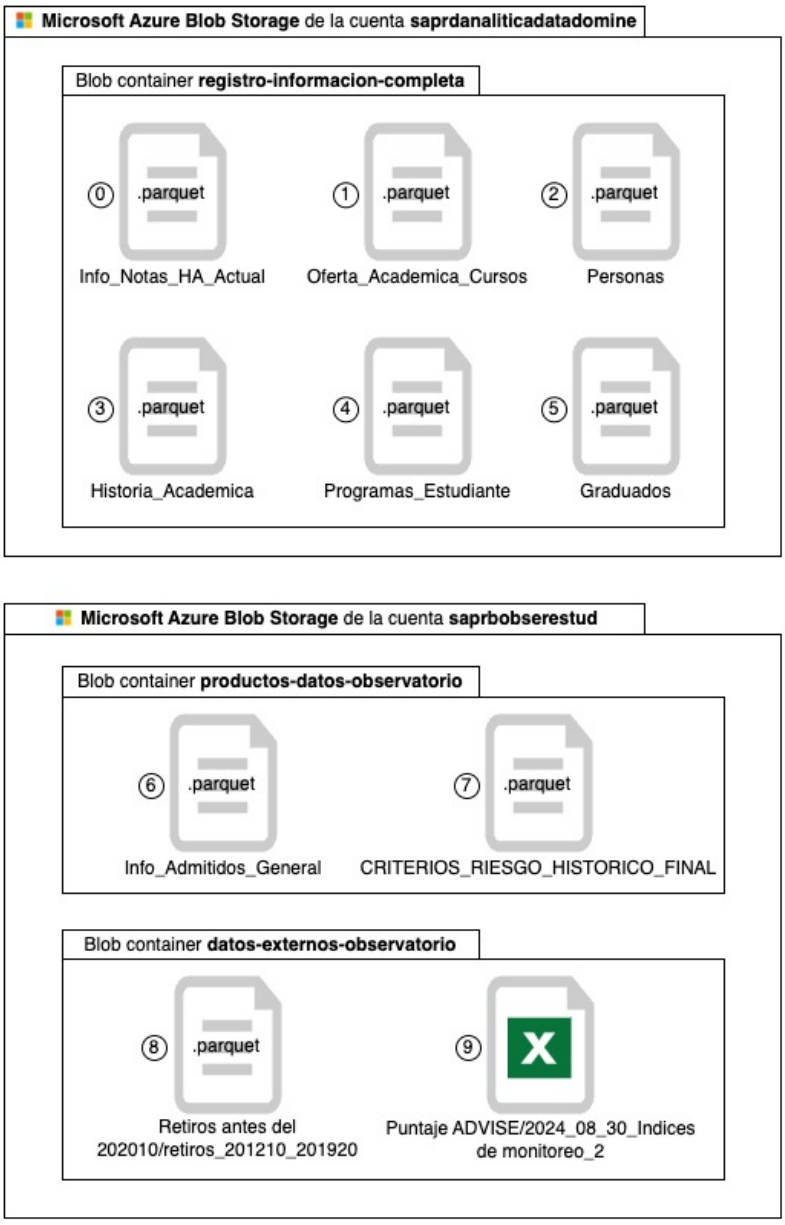

Figura 2.1: Archivos fuente y estructura de directorios en Azure Blob Storage

Cada uno de los archivos presentados en la figura 2.1 se cargan en el cuaderno de Jupyter y se extrae la información relevante para el Perfil del estudiante. La tabla 2.2 detalla el orden en el que se realiza cada extracción, describe la información que se extrae y especifica el archivo fuente del cual se obtiene, empleando la numeración de los archivos

definida en la figura 2.1.

Tabla 2.2
Extracción de datos

Orden	Información	Descripción	Archivos fuente
1	Histórico académico	Información de las notas obtenidas por los estudiantes en cada una de las asignaturas que han cursado.	0
2	Oferta académica	Información de las asignaturas que se ofertan en cada uno de los semestres académicos.	1
3	Estudiantes	Información básica de los estudiantes.	2, 3, 6
4	Programas académicos	Información de los programas académicos en los que se encuentran inscritos los estudiantes.	4
5	Información adicional de retiros	Información sobre los retiros de los estudiantes para periodos anteriores al 2019-20, que no se encuentra en el Histórico académico.	8
6	Graduados	Información de los estudiantes que se han graduado.	5
7	Criterios de riesgo	Información de los criterios de riesgo que se utilizan para identificar a los estudiantes en riesgo académico.	7
8	Advise	Información de los estudiantes tomada de la plataforma Advise.	9

2.2.2. Transformación de los datos

2.2.3. Carga de los datos

2.3. Flujo de los datos

Esta última sección de la capa de datos se ocupa de ofrecer una visión de alto nivel del flujo que los datos atraviesan desde su origen hasta su destino final en el perfil del estudiante. Es sensato presentar este flujo en este punto del documento, pues ya se han detallado las fuentes de datos y el proceso de ETL, por lo que se da un abrebocas del paso de los datos por el API REST, que es el siguiente componente de la arquitectura del Perfil del estudiante, y del consumo de los datos por parte del frontend. Por ende, esta sección contextualiza al lector y opera como preámbulo a los próximos capítulos.

La figura 2.2 muestra el flujo de los datos mencionado. Los datos se extraen del Ecosistema de analítica institucional, se transforman y se cargan en el API REST, que a su vez los almacena en una base de datos PostgreSQL. El API REST expone una interfaz de programación que permite a los clientes, como el frontend del Perfil del estudiante, consumir los datos de manera estructurada y segura.

Figura 2.2: Flujo de los datos, desde su origen en el Ecosistema de analítica institucional hasta su visualización en el frontend del Perfil del estudiante

Capítulo 3

Capa de lógica

Este capítulo se ocupa de describir la capa de lógica del sistema, que se encarga de procesar los datos extraídos y exponerlos como recursos accesibles mediante una API REST. Primeramente, inicia por justificar la construcción de un API REST como interfaz de la capa de lógica. Posteriormente, se detalla por completo el diseño del API, desde dos perspectivas: por un lado, se describe la estructura de los datos y cómo se representan como objetos, en particular como clases, agnósticas de la fuente de los datos y de la implementación de la API; por otro lado, se expone el diseño de la interfaz del API, detallando los recursos que expone, su representación y los endpoints que brinda para acceder a ellos. Finalmente, se dedica una sección a la calidad del código, detallando las herramientas y prácticas empleadas para garantizar la calidad del código.

3.1. ¿Por qué un API REST?

Antes de entrar en detalles sobre el diseño del API, es importante justificar la elección de un API REST como interfaz de la capa de lógica del sistema. En primer lugar, un API REST es una interfaz de programación de aplicaciones que sigue los principios del estilo arquitectónico REST. Este estilo se basa en la transferencia de representaciones de recursos, que son identificados por URIs, y la manipulación de estos recursos mediante métodos estándar de HTTP. En particular, un API REST es una interfaz que permite la interoperabilidad de aplicaciones heterogéneas, permitiendo a un programa acceder a las funciones y datos de otro. En el caso de este proyecto, un API REST es la interfaz ideal para exponer los datos procesados por la capa de lógica, ya que permite a cualquier aplicación, como el frontend del perfil del estudiante, acceder a estos datos de manera sencilla y eficiente.

En los últimos años, los API REST han ganado popularidad en el desarrollo de aplicaciones web, ya que son fáciles de entender, escalables y flexibles. Además, un API REST es independiente de la implementación subyacente, lo que significa que el frontend del perfil del estudiante no necesita conocer los detalles de cómo se procesan los datos o cómo se almacenan, sino que solo necesita conocer la interfaz del API. Por último, un API REST es fácil de probar y depurar, ya que se basa en estándares abiertos y bien conocidos, como HTTP y JSON.

Todo esto hace que un API REST sea la elección ideal para la capa de lógica de

este sistema, ya que permite exponer los datos procesados de manera sencilla y eficiente, independiente de la implementación subyacente, y fácil de probar y depurar.

3.2. Diseño del API

El diseño del API se divide en dos partes. La primera parte está en el marco de la programación orientada a objetos, y se encarga de describir cuál es la estructura de clases del API: qué clases existen, cómo se relacionan entre sí y qué atributos tienen. La segunda parte se enfoca en la interfaz del API, es decir, en cómo se exponen los datos procesados como recursos accesibles mediante una API REST. Eso incluye cuáles son los recursos que se exponen, cómo se representan y en qué formato se devuelven al cliente, y cuáles son los endpoints que brinda el API para acceder a estos recursos.

Es importante señalar que el diseño del API es completamente independiente de dos factores: por un lado, de la fuente de los datos, es decir, de cómo se extraen los datos de las fuentes originales; por otro lado, de la implementación del API, es decir, de qué lenguaje y cuál framework se utiliza para escribir el código del API. En este sentido, esta sección es independiente del capítulo anterior, que estudia la capa de datos, y de la sección siguiente, que explica la implementación del API.

3.2.1. Estructura de clases del API

Para modelar la información del Perfil del estudiante como objetos se optó por realizar un diagrama de clases conforme al estándar UML. Esto tiene la ventaja de que permite visualizar de manera clara y concisa la estructura de clases del API y de que es un artefacto bien conocido y ampliamente utilizado en el desarrollo de software.

Diagrama de clases

Se realizaron varias iteraciones en la construcción del diagrama de clases, teniendo en cuenta las necesidades del frontend del perfil del estudiante y las restricciones de la capa de datos. La figura 3.1 muestra el diagrama de clases final del API.

El diagrama de clases está escrito en inglés, siguiendo las convenciones de UML. Cada clase tiene un nombre, que es un sustantivo en singular, y una serie de atributos. Los atributos están escritos en el formato `nombre: tipo`, donde `nombre` es el nombre del atributo y `tipo` es el tipo de dato del atributo. Por conveniencia, los tipos de datos empleados son los mismos que se utilizan en el lenguaje de programación Python, que es el lenguaje en el que se implementa el API, pero bien podrían usarse los tipos de datos de cualquier otro lenguaje de programación orientado a objetos sin afectar semánticamente el diagrama de clases, por lo que mantiene su independencia de la implementación.

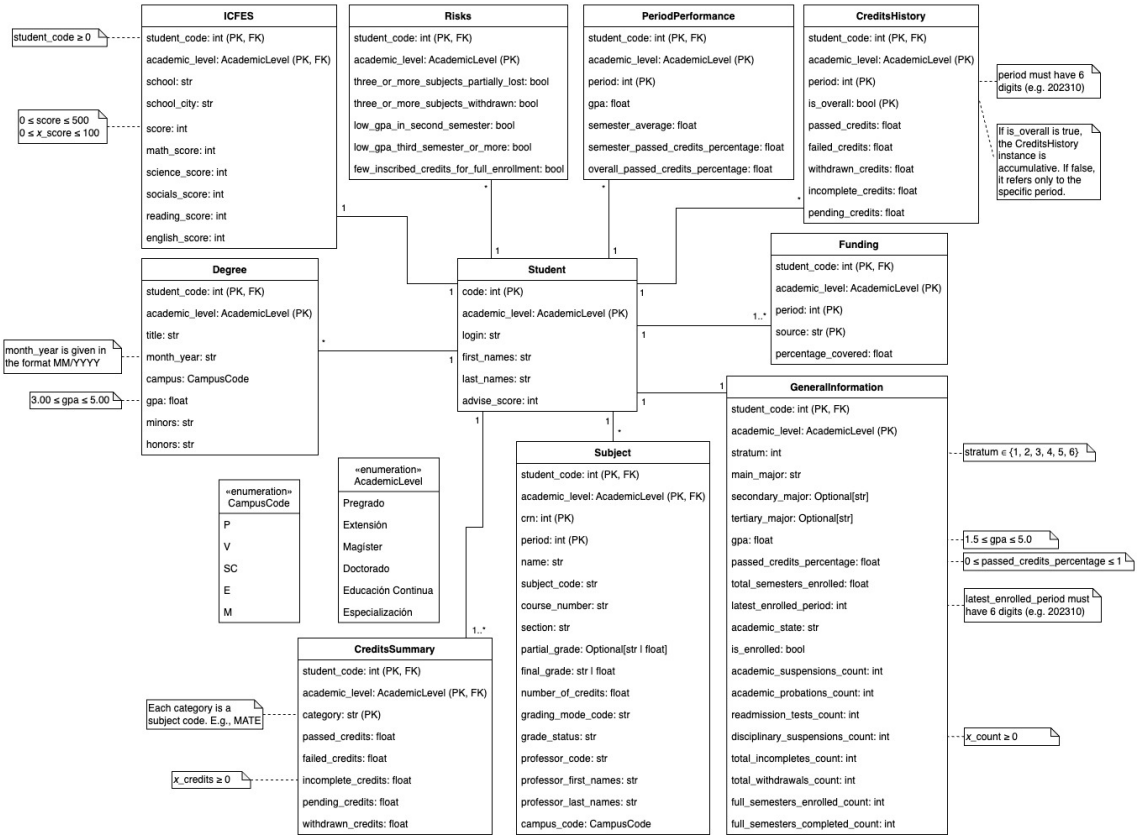

Figura 3.1: Diagrama de clases del API

Clases del API

La estructura de clases del API se basa en el principio de responsabilidad única, que establece que cada clase debe tener una sola responsabilidad y representar un solo concepto en el dominio del problema. Las clases propuestas en el diagrama se presentan alfabetizadas en la tabla 3.1, con su nombre original en inglés, la traducción al español y una breve descripción de su responsabilidad.

Tabla 3.1
Clases del API y sus responsabilidades

Nombre original	Nombre traducido	Responsabilidad
CreditsHistory	Historial de créditos	Rastrea el estado global de los créditos del estudiante, incluyendoaprobados, reprobados, retirados, pendientes e incompletos, tanto para periodos específicos, por ejemplo un semestre particular, como en general hasta la fecha.

Continúa en la página siguiente

Tabla 3.1: Clases del API y sus responsabilidades (Continuación)

Nombre original	Nombre traducido	Responsabilidad
CreditsSummary	Resumen de créditos	Proporciona un desglose categórico de los créditos aprobados, reprobados, retirados, pendientes e incompletos, para una categoría específica de materias. Algunos ejemplos de categoría pueden ser: las materias de la carrera principal, las materias de matemáticas, las materias de física, entre otras.
Degree	Título	Almacena información sobre los títulos académicos obtenidos por el estudiante, como el nombre del título, el campus, el promedio y menciones especiales, como títulos con honores.
Funding	Financiamiento	Detalla las fuentes de financiamiento del estudiante, que pueden incluir becas y créditos educativos, con el porcentaje cubierto por cada fuente.
GeneralInformation	Información general	Contiene datos generales del estudiante, como su carrera principal, estrato, GPA actual, entre otros indicadores relevantes.
ICFES	ICFES	Almacena los puntajes del estudiante en las áreas evaluadas por el examen ICFES, como matemáticas, lectura y ciencias.
PeriodPerformance	Rendimiento por periodo	Guarda información del desempeño académico del estudiante en un periodo específico, incluyendo el promedio del semestre y el porcentaje de créditos aprobados.
Risks	Riesgos	Identifica indicadores de riesgo académico, como bajo GPA en los primeros semestres o retiro de múltiples materias.

Continúa en la página siguiente

Tabla 3.1: Clases del API y sus responsabilidades (Continuación)

Nombre original	Nombre traducido	Responsabilidad
Student	Estudiante	Representa a un estudiante de la universidad cuando se encontraba en un nivel académico específico. Almacena la información esencial para la identificación del estudiante, como nombres, apellidos, código y usuario (`login`, que corresponde al nombre de usuario en el correo electrónico institucional).
Subject	Materia	Representa una materia específica cursada por el estudiante, incluyendo el nombre, el código, los créditos, el profesor, la nota obtenida y el estado de la materia.

Como complemento del diagrama de clases y de la tabla 3.1, vale la pena ahondar en algunas de las decisiones de diseño que se tomaron en la creación de las clases con base en el dominio del negocio:

- La clase `Student` es la clase principal del modelo. Resalta que no representa a un estudiante por completo, sino que representa a un estudiante en un nivel académico específico. Esto se debe a que resulta conveniente tratar de forma distinta al mismo estudiante en diferentes niveles académicos, ya que su contexto puede haber cambiado significativamente: el título al que aspira es distinto, las materias que cursa son distintas, su desempeño académico puede haber mejorado o empeorado, e incluso su situación socioeconómica puede haber cambiado. La misma persona en pregrado no puede ser tratada de la misma forma que en posgrado.

- La clase `CreditsSummary` es una clase auxiliar que se encarga de proporcionar un desglose categórico de los créditos aprobados, reprobados, retirados, pendientes e incompletos, para una categoría específica de materias. Naturalmente, no para todos los estudiantes se incluyen las mismas categorías. Es decir, para un estudiante de ingeniería, Categorías relevantes pueden ser las materias Específicas de su rama del ingeniería, las materias de matemática y las materias de física. Sin embargo, para un estudiante de arte, incluso si el estudiante ha optado por cursar materias de matemática y física, esas categorías pueden no ser relevantes para su desempeño académico. Esto es un matiz que es difícil representar en un diagrama de clases y que corresponde a especificidades del dominio del problema, por lo cual se aclara en este apartado.

- Puede que el lector se haya percatado que la clase `Risks` está nombrada con un sustantivo en plural, lo cual aparentemente transgrede las convenciones de construcción de diagramas de clase. Eso no es un error, sino una decisión consciente de diseño. Se debe a que la clase `Risks` no representa un riesgo académico en

particular, sino que cada instancia de la clase corresponde a un conjunto de riesgos académicos, entre ellos bajo GPA y retiro de múltiples materias. Por lo tanto, el nombre en plural es más adecuado para representar la naturaleza de la clase.

Relaciones entre las clases

Las relaciones entre las clases se representan mediante líneas que unen las clases. Las relaciones en este diagrama son muy simples, lo cual fue una elección deliberada en pos de la simplicidad y la claridad. Cada línea representa una asociación entre dos clases y en sus extremos cuenta con números que indican la multiplicidad de la asociación, es decir, cuántos objetos de una clase están asociados con cuántos objetos de la otra clase. Un asterisco indica que hay muchos objetos asociados, mientras que un número indica la cantidad exacta de objetos asociados. Por ejemplo, un estudiante está asociado con muchas materias, pero cada materia está asociada con un solo estudiante, por lo que la asociación entre la clase `Student` y la clase `Subject` es de uno a muchos, representada por la línea que une ambas clases con un 1 en el extremo de la clase `Student` y un asterisco en el extremo de la clase `Subject`. La tabla 3.2 detalla las relaciones entre las clases del diagrama de clases, con una breve descripción de la relación y la multiplicidad de la asociación. Se ordenan en orden de aparición, de arriba a abajo y de izquierda a derecha, en el diagrama de clases.

Tabla 3.2

Relaciones entre las clases del diagrama de clases

Clase 1	Clase 2	Descripción y Multiplicidad
`Student`	`GeneralInformation`	Un estudiante está asociado con una única instancia de información general. Multiplicidad: 1 a 1.
`Student`	`ICFES`	Un estudiante puede tener una única instancia de resultados del ICFES. Multiplicidad: 1 a 1.
`Student`	`Risks`	Un estudiante tiene una única instancia de indicadores de riesgos académicos. Multiplicidad: 1 a 1.
`Student`	`PeriodPerformance`	Un estudiante puede estar asociado con muchos periodos de desempeño académico. Multiplicidad: 1 a muchos.
`Student`	`CreditsHistory`	Un estudiante puede tener muchos historiales de créditos, cada uno relacionado con un periodo específico o de manera acumulativa. Multiplicidad: 1 a muchos.

Continúa en la página siguiente

Tabla 3.2: Relaciones entre las clases del diagrama de clases (Continuación)

Clase 1	Clase 2	Descripción y Multiplicidad
`Student`	`CreditsSummary`	Un estudiante tiene un resumen de créditos categorizado. Multiplicidad: 1 a 1.
`Student`	`Subject`	Un estudiante está asociado con muchas materias, pero cada materia pertenece a un único estudiante. Multiplicidad: 1 a muchos.
`Student`	`Degrees`	Un estudiante puede haber obtenido múltiples títulos. Multiplicidad: 1 a muchos.
`Student`	`Funding`	Un estudiante puede tener múltiples fuentes de financiamiento. Multiplicidad: 1 a muchos.
`Subject`	`CreditsSummary`	Cada materia está categorizada en un resumen de créditos. Multiplicidad: 1 a 1.

Reglas de negocio en el diagrama de clases

En el diagrama de clases se pueden evidenciar diversas anotaciones en algunos atributos. Varias de estas anotaciones corresponden a restricciones de integridad que se deben cumplir en el modelo de datos, que en últimas representan reglas del dominio del problema. A esto, usualmente se le denomina *reglas de negocio* y son restricciones uqe se deben cumplir en el modelo de datos para que reflejen la realidad de manera fiel.

La tabla 3.3 detalla las anotaciones en el diagrama de clases que tienen esa función, junto con su significado en términos del dominio del problema, es decir, en el contexto de la Universidad. Se ordenan en orden de aparición en el diagrama de clases, de arriba a abajo y de izquierda a derecha.

Tabla 3.3

Anotaciones del diagrama de clases y su significado

Anotación	Significado	Justificación
`student_code` $\geq$ `0`	El código del estudiante debe ser mayor o igual a 0	El código del estudiante debe ser un entero positivo que identifica únicamente a cada estudiante. No se permiten códigos negativos.

Continúa en la página siguiente

Tabla 3.3: Anotaciones del diagrama de clases y su significado (Continuación)

Anotación	Significado	Justificación
`0 ≤ score ≤ 500`	El puntaje del ICFES (formalmente, la prueba Saber 11) debe estar entre 0 y 500	El puntaje máximo en la prueba Saber 11 es 500, y el mínimo es 0.
`0 ≤ x_score ≤ 100`	Los puntajes de las áreas evaluadas por el ICFES deben estar entre 0 y 100	Los puntajes de las áreas evaluadas por el ICFES se normalizan a una escala de 0 a 100.
`month_year is given in the format MM/YYYY`	La fecha de graduación, que se representa como un mes y un año, debe tener el formato MM/YYYY	La fecha de graduación se codifica en el formato de mes y año para garantizar consistencia en los registros.
`3.00 ≤ gpa ≤ 5.0`	El GPA debe estar entre 3.0 y 5.0	En el sistema de la Universidad, el GPA mínimo para graduarse es 3.0, y el máximo posible es 5.0.
`Each category is a subject code. E.g., MATE`	Cada categoría de un Resumen de créditos es un código de materia. Por ejemplo, MATE representa las materias de matemáticas	Las categorías de un Resumen de créditos se representan con códigos de materia. Esta decisión de diseño se explica en detalle arriba, en la descripción de la clase `CreditsSummary`.
`x_credits ≥ 0`	Los créditos aprobados, reprobados, retirados, pendientes e incompletos deben ser mayores o iguales a 0	No tiene sentido registrar un número negativo de créditos en ningún contexto académico.
`period must have 6 digits (e.g. 202310)`	El periodo académico debe tener 6 dígitos	Los periodos académicos se codifican en el formato de año seguido por el indicador del tipo de periodo. Usualmente se usa un guión entre el año y el tipo de periodo, e.g., 2023-10, pero en este caso resulta más conveniente usar un número entero de 6 dígitos.

Continúa en la página siguiente

Tabla 3.3: Anotaciones del diagrama de clases y su significado (Continuación)

Anotación	Significado	Justificación
`stratum ∈ {1, 2, 3, 4, 5, 6}`	El estrato socioeconómico debe estar entre 1 y 6	Los estratos socioeconómicos en Colombia están normados en este rango por ley. Si el estudiante no es residente en Colombia, no se registra su estrato.
`1.5 ≤ gpa ≤ 5.0`	El GPA debe estar entre 1.5 y 5.0	En el sistema de la Universidad, el GPA mínimo posible es 1.5, y el máximo posible es 5.0.
`0 ≤ passed_credits_percentage ≤ 1`	El porcentaje de créditos aprobados debe estar entre 0 y 1	Todo porcentaje es un número entre 0 y 1, inclusive.
`latest_enrolled_period must have 6 digits (e.g. 202310)`	El periodo académico más reciente en el que el estudiante estuvo inscrito debe tener 6 dígitos	Esta explicación es análoga a la de la anotación `period must have 6 digits (e.g. 202310)`.

3.3. Implementación del API

3.4. Calidad de código

Teniendo en cuenta que el software producido es de alto valor para la universidad y por ende probablemente querrá ser mantenido, extendido y reutilizado en el futuro, su desarrollo fue sometido a rigurosos estándares de calidad de código. Esta sección detalla las herramientas y prácticas empleadas para garantizar la calidad del código.

Resumen del capítulo

Se realizaron múltiples esfuerzos para garantizar la calidad del código del proyecto:

- Se documentaron todas las funciones y métodos con docstrings siguiendo la convención de Pandas.
- Se utilizaron isort y black como formateadores, Flake8 y SonarLint como linters, y doctest y Pytest para las pruebas unitarias.
- Se configuró un hook de pre-commit de Git para garantizar el formato y ejecutar las pruebas antes de cada commit.
- Se estableció un pipeline en GitHub Actions que ejecuta las pruebas, genera un reporte de cobertura y lo envía a SonarQube.
- Se configuró SonarQube para realizar análisis estático sobre el código, usando el perfil de calidad estándar para Python 3.11.

Todas las funciones y métodos de la API están documentados mediante *docstrings*. Naturalmente, los docstrings siguen las convenciones definidas en el estándar PEP 257. Más aún, debido a que las disposiciones del estándar son notablemente flexibles, se optó por seguir la afamada convención de docstrings de NumPy. Más específicamente, se satisfacen todos los lineamientos de la guía de docstrings de Pandas.

Su sintaxis satisface el estilo reStructuredText (REST),

Se hace uso del linter Flake8,

Se emplea la librería `flake8-docstrings`, con la configuración `docstring-convention=` en el archivo de configuración de Flake8, para verificar que los docstrings sigan la convención de NumPy.

3.4.1. Representación de los datos como recursos de una API REST

Una vez los datos han sido procesados y almacenados en el Blob Storage, se exponen como recursos accesibles mediante una API REST. La API se encuentra escrita en Python, utilizando el framework FastAPI, y se despliega en un contenedor Docker que reside en una máquina virtual provista por la universidad.

La siguiente sección trata en detalle todos los aspectos técnicos relacionados con este API REST. Por lo pronto, basta con mostrar, a alto nivel, cuáles son los recursos que provee la API para el consumo por parte del frontend del perfil del estudiante. La tabla 3.4 detalla dichos recursos y la información que contienen. Nótese que el API expone muchos más recursos que los mostrados en la tabla, pero solo esos se utilizan en el perfil del estudiante.

Tabla 3.4
Recursos de la API REST

Ruta del recurso	Información
`/nes/student/`	Información básica del estudiante: nombres, apellidos, código, usuario y niveles académicos en los que ha estado inscrito.

Capítulo 4

Front

Ruta al perfil del estudiante

Como ruta para acceder al perfil del estudiante desde NES se seleccionó `/perfil-estudiante/<correo>`, donde `<correo>` corresponde al correo Uniandes, sin el dominio, del estudiante de quien se quiere ver el perfil. Por ejemplo, la ruta para acceder al perfil del autor de este documento es `/perfil-estudiante/f.melo`.

Cualquier usuario que esté autenticado en NES y cuente con los permisos adecuados podrá acceder y visualizar directamente el perfil del estudiante al ingresar a la ruta indicada.

Esto permite que entre profesores y directivos de la Universidad se compartan fácilmente perfiles de estudiantes. En caso de que el lector tenga ese rol dentro de la Universidad, puede probar la funcionalidad iniciando sesión en NES y luego pulsando en el siguiente enlace: `https://noestassolo.virtual.uniandes.edu.co/perfil-estudiante/f.melo`.

Se hace hincapié en que el usuario debe estar autenticado en NES y debe contar con los permisos necesarios para acceder al perfil de un estudiante. De lo contrario, no podrá visualizarlo. En particular, ningún estudiante podrá acceder al perfil de algún otro de esta manera (ni de ninguna otra, salvo que el otro estudiante o algún otro usuario autorizado deliberadamente se lo muestre).

Capítulo 5

Diseño Conceptual

5.1. Principios de Exposición de Información

Explica los principios fundamentales que guiaron el diseño del Perfil del estudiante. Incluye claridad, relevancia, simplicidad, y cómo se aseguraron de que la información sea comprensible y útil para los diferentes usuarios.

5.2. Visualizaciones y Tablas

Describe las decisiones tomadas para las visualizaciones y tablas interactivas, incluyendo qué información se priorizó y cómo se seleccionaron las herramientas o librerías para representarlas.

Apéndice A

Contribuciones al CAPP

Este apéndice describe las contribuciones realizadas al CAPP durante el desarrollo de este proyecto. En particular, se detallan los recursos expuestos por la API REST para consumo exclusivo del CAPP, a petición de Daniel Arango Cruz y Marilyn Stephany Joven Fonseca, quienes lideran el desarrollo de aquel proyecto.

A.1. Contexto

El CAPP es un sistema que permite a los estudiantes de pregrado de la Universidad de los Andes visualizar su progreso académico. Su propósito principal es determinar si un estudiante ha cumplido con los requisitos mínimos para graduarse de un programa académico. Para ello, el CAPP se conecta con diversas fuentes de datos, entre ellas, con el API REST diseñado y desarrollado en este proyecto.

A.2. Recursos expuestos

Los recursos provistos fueron diseñados específicamente para el CAPP ante solicitud de los desarrolladores. Se solicitó que la nomenclatura para los recursos y sus atributos fuera en español, en lugar de inglés, como se había manejado en el resto de la API REST.

A causa de eso y como forma de separación semántica, se decidió agrupar los *endpoints* bajo el prefijo `/capp/`. La tabla A.1 detalla los *endpoints* expuestos por la API REST para el CAPP.

Tabla A.1
Recursos expuestos por la API REST para el CAPP

Método HTTP	URL	Descripción	Cuerpo de la petición	Respuesta
`GET`	`/capp /materias /estudiante /{codigo estudiante}`	Obtiene todas las materias aprobadas por un estudiante en sus estudios de pregrado.	–	Lista de objetos `Materia` con las materias aprobadas por el estudiante.
`GET`	`/capp /materia /{codigo materia}`	Obtiene la información básica de una materia a partir de su código.	–	Objeto `Materia` correspondiente al código.
`PUT`	`/capp /materias/`	Obtiene la información básica de múltiples materias a partir de una lista de sus códigos.	Lista de códigos de materias	Lista de objetos `Materia` correspondientes a los códigos provistos.
`GET`	`/capp /estudiante /programa/`	Obtiene la información de los programas académicos cursados por el estudiante.	–	Lista de nombres de los programas académicos, ordenados desde el principal.

El recurso `Materia` mencionado en la tabla es una representación simplificada de una materia, que contiene la información mínima necesaria para el CAPP. La figura A.1 muestra un ejemplo de esta representación. No se hace uso de la representación `SubjectResponse` mencionada en la sección **??**, ya que esta contiene información adicional y se incurriría en sobreexposición de datos.

```
{
  "codigo_curso": "MATE",
  "numero_curso": "1104",
  "numero_creditos": 3,
  "nombre_curso": "TEORIA DE GRAFOS",
  "atributo_sec": "",
  "periodo": 202320
}
```

Figura A.1: Ejemplo de representación del recurso `Materia` para el CAPP

Apéndice B

Apuntes para el Ecosistema de analítica institucional

Este apéndice contiene apuntes y recomendaciones para el Ecosistema de analítica institucional de la Universidad de los Andes. Estas recomendaciones se basan en la experiencia adquirida durante el desarrollo de este proyecto y en la revisión de literatura sobre calidad de datos y buenas prácticas en la arquitectura de malla, que es la utilizada por el Ecosistema.

Glosario

API (*Application Programming Interface*) Mecanismo que permite la interoperabilidad de aplicaciones heterogéneas, permitiendo a un programa acceder a las funciones y datos de otro. 26

API REST API que sigue los principios arquitectónicos REST, permitiendo interacciones con los recursos usando métodos estándar HTTP. 17, 26, 27, 33

Azure Blob Storage Servicio de almacenamiento de objetos en la nube de Microsoft Azure. Más información en su página oficial. 13

black Librería o utilidad de Python que formatea automáticamente el código fuente usando un estilo consistente con cualquier otro código fuente formateado con black, de acuerdo con su página de web. 26

CAPP (Currículum, Asesoría y Planeación de Programas) Sistema de información de la Universidad de los Andes que determina si un estudiante satisface los requisitos para graduarse de un programa académico. 33, 34

docstring Cadena de texto especiales que se escriben al inicio de un módulo, clase, método o función en Python, que describe su funcionalidad y cómo usarlo. 26

doctest Módulo en Python que permite probar fragmentos de código incluidos en los docstrings, validando que la salida coincida con los ejemplos dados. 26

ETL (*Extract, Transform, Load*) Proceso utilizado en la integración de datos, que consiste en extraer datos de múltiples fuentes, realizar sobre ellos las transformaciones necesarias y cargarlos en un destino. 12, 14

Flake8 Herramienta de Python que combina varios linters para verificar el cumplimiento de las guías de estilo de Python. Más información en su documentación. 26

formateador Herramienta que modifica automáticamente el código fuente para que siga convenciones de estilo predefinidas. 26

Git Sistema de control de versiones distribuido. 26

GitHub Actions Servicio de integración continua de GitHub que permite automatizar flujos de trabajo como pruebas y despliegues en repositorios de código fuente. Más información en su documentación. 26

hook En Git, es un script que se ejecuta automáticamente en momentos específicos del flujo de trabajo, como antes de hacer un commit o un push. 26

HTTP (*Hypertext Transfer Protocol*) Protocolo de comunicación que permite la transferencia de información en la web, caracterizado por operar sin estado y basado en el modelo cliente-servidor. 17

isort Librería o utilidad de Python que alfabetiza las importaciones en el código fuente y las separa por tipo, de acuerdo con su página de web. 26

JSON (*JavaScript Object Notation*) Notación de objetos de JavaScript, un formato ligero de intercambio de datos que es fácil de leer y escribir para humanos y fácil de analizar y generar para máquinas. 17

Jupyter Software de código abierto que permite crear y compartir documentos interactivos que contienen código, visualizaciones y texto explicativo. Más información en su página oficial. 12

linter Herramienta que analiza el código fuente para encontrar errores, malas prácticas o incumplimientos de convenciones de estilo. 26

NES (No Estás Solo) Plataforma web de la Universidad de los Andes para el éxito estudiantil. `https://noestassolo.virtual.uniandes.edu.co/`. 29

NumPy "El paquete fundamental para la computación científica en Python", de acuerdo con su página oficial. Librería de Python que añade soporte para arreglos y matrices de gran tamaño, junto con una colección de funciones matemáticas para operarlos. 26

Pandas Librería de análisis y manipulación de datos en Python. Más información en su página oficial. 26

PEP (*Python Enhancement Proposal*) Documento que propone y describe una nueva característica o modificación en Python. En la PEP 1 se define qué es un PEP. 26

perfil de calidad Conjunto de reglas y configuraciones que definen los estándares de calidad del código. 26

pipeline Flujo de automatización en el que se definen una serie de pasos o tareas que deben ejecutarse de manera secuencial o en paralelo. 26

pruebas unitarias Pruebas automáticas que se enfocan en verificar el correcto funcionamiento de las unidades más pequeñas de un programa, como funciones o métodos individuales. 26

Pytest Framework de pruebas para Python. Más información en su página oficial. 26

Python Lenguaje de programación de propósito general y alto nivel, interpretado y multiparadigma, aunque principalmente orientado a objetos. 12, 26

REST (*Representational State Transfer*) Estilo arquitectónico para sistemas hipermedia distribuidos, como la *World Wide Web*. Se basa en la transferencia de representaciones de recursos, que son identificados por URIs, y la manipulación de estos recursos mediante métodos estándar de HTTP. Para una descripción detallada, leer el quinto capítulo de la tesis doctoral de Roy Fielding. 17

reST (reStructuredText) Sintaxis de marcado ligero utilizada para documentación, como docstrings en Python. Más información en la documentación oficial. 26

SonarLint Extensión de análisis estático de código que se integra en el editor o IDE para detectar problemas de calidad en el código y proponer correcciones. 26

SonarQube Plataforma que realiza análisis estático de código para detectar vulnerabilidades, errores y problemas de mantenimiento en proyectos de software. 26

UML (*Unified Modeling Language*) Lenguaje unificado de modelado, un lenguaje estándar para visualizar, especificar, construir y documentar los artefactos de un sistema de software. 18

URI (*Uniform Resource Identifier*) Cadena de caracteres con estructura fija que identifica un recurso en la web de forma única. 17

Bibliografía

[1] Universidad de los Andes. Oferta de cursos. Consultado el 20 de noviembre de 2024.

[2] Universidad de los Andes. Ecosistema de analítica institucional, 2024. Consultado el 20 de noviembre de 2024.

[3] Ellucian. Ellucian banner. Consultado el 20 de noviembre de 2024.

[4] Ellucian. Ellucian crm advise. Consultado el 20 de noviembre de 2024.