IPC2 proyecto 2

202200231 – Pablo Jose Ortiz Linares

Resumen

Este proyecto consistió en el diseño, modelado, documentación e implementación de una solución para una página web la cual, sería una simulación para llevar un control de notas de diferentes cursos. Siendo capaz de recibir archivos XML y cargar la informacion. Todo esto desarrollado con el modelo vista controlador el cual consiste en una API que recibe las peticiones y un Cliente web que manda las peticiones. El objetivo fue aprender conceptos de desarrollo web basado en Python haciendo uso de django y Flask para la creación del ForntEnd y el Backend respectivamente.

La pagina consta de tres tipos de usuarios los cuales tienen diferentes roles. El administrador puede cargar a los estudiantes y a los tutores y asignar a cada uno o varios cursos. El tutor puede cargar sus horarios y subir las calificaciones y finalmente el estudiante puede revisar sus notas.

Palabras clave

Máximo cinco palabras que servirán para identificar el estudio realizado.

Abstract

This project consisted of the design, modeling, documentation, and implementation of a solution for a website that simulates a system to manage grades for different courses. It is capable of receiving XML files and loading the information. All of this was developed using the Model-View-Controller (MVC) pattern, which includes an API that handles requests and a web client that sends them. The main goal was to learn web development concepts using Python by implementing the frontend and backend with Django and Flask, respectively.

The platform includes three types of users, each with different roles. The administrator can upload students and tutors and assign one or more courses to each. The tutor can set their schedules and upload grades, and finally, the student can view their grades.

Keywords

Traducción al idioma inglés de las palabras clave.

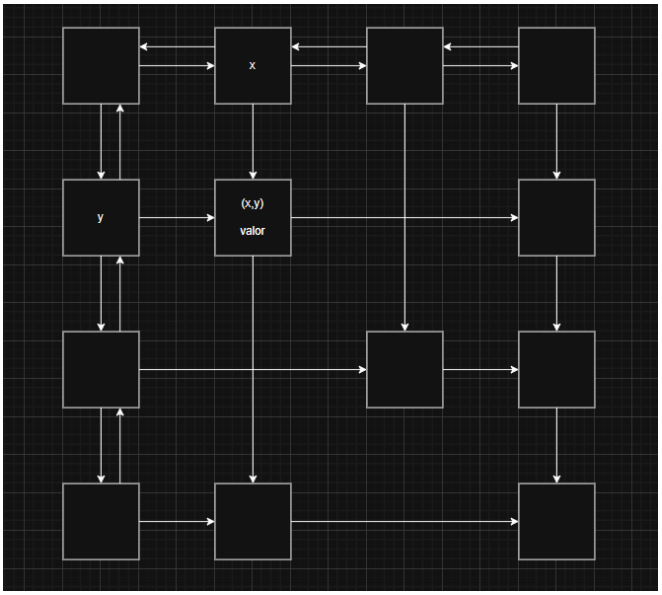
Introducción

El presente proyecto, *IPC2-AcadNet*, fue desarrollado como parte del curso Introducción a la Programación y Computación 2, con el objetivo de construir una plataforma web educativa orientada a la gestión y seguimiento del aprendizaje académico. La solución implementada integra una arquitectura cliente-servidor, compuesta por un frontend en Django y una API backend desarrollada en Flask. Se implementaron funcionalidades clave como la carga y transformación de archivos XML, el uso de matrices dispersas para representar el progreso académico, y la generación de reportes estadísticos mediante visualizaciones gráficas. El proyecto fue abordado bajo el paradigma de programación orientada a objetos, promoviendo buenas prácticas de desarrollo y escalabilidad.

Debe contener un máximo de 150 palabras.

Desarrollo del tema

***Matrices dispersas*:** Una matriz dispersa es una estructura de datos que se caracteriza por contener principalmente ceros o valores nulos, es decir, muy pocos elementos tienen un valor significativo. A diferencia de una matriz densa, donde la mayoría de las posiciones están ocupadas por números distintos de cero, una matriz dispersa ahorra espacio al enfocarse solo en los valores relevantes.
El principal beneficio de usar matrices dispersas es el ahorro de memoria y tiempo de procesamiento. En lugar de almacenar todos los elementos, una matriz dispersa se puede representar de manera más eficiente guardando solo la posición y el valor de los elementos no nulos. Por ejemplo, una matriz grande de 1000x1000 con solo 100 valores útiles puede representarse como una lista de tuplas o como un diccionario, sin necesidad de almacenar todos los millones de ceros.
Este tipo de estructura es muy útil en programación, especialmente en campos donde se manejan grandes volúmenes de datos. Por ejemplo, en machine learning, al convertir textos en vectores de palabras, se generan matrices con miles de columnas por cada palabra posible, pero en cada documento solo se usan unas pocas. En este caso particular se creo una matriz dispersa con dos listas doblemente enlazadas para los headers y nodos para el contenido como se muestra a continuación.



Frontend: Es la parte de una aplicación o sitio web que interactúa directamente con el usuario. Es todo lo que el usuario puede ver y manipular en su pantalla: los botones, menús, formularios, colores,

animaciones, textos, y diseños en general. Se le conoce también como la interfaz de usuario UI. Cuando accedes a una página web, como una tienda en línea o una red social, todo lo que ves en el navegador es resultado del trabajo del frontend.
El desarrollo frontend se encarga de construir esta interfaz utilizando principalmente tres tecnologías esenciales: HTML, CSS y JavaScript. HTML (HyperText Markup Language) define la estructura básica del contenido, CSS (Cascading Style Sheets) se encarga del estilo visual (colores, fuentes, márgenes, etc.), y JavaScript añade interactividad, permitiendo que los elementos respondan a acciones del usuario, como clics o desplazamientos. Sin embargo, para la realización de esta practica se usó Django y Python para su realización.
Además de estas tecnologías básicas, hoy en día se usan frameworks y bibliotecas modernas como React, Vue.js, Angular o Svelte, que facilitan el desarrollo de interfaces más dinámicas y complejas. Estas herramientas permiten construir aplicaciones web más rápidas, organizadas y fáciles de mantener, dividiendo el código en componentes reutilizables.

Backend: es la parte de una aplicación que no es visible para el usuario, pero que se encarga de toda la lógica interna, el almacenamiento de datos y la comunicación con el frontend. Es como el "motor" de una aplicación: no lo ves, pero hace que todo funcione correctamente. Cuando un usuario realiza una acción en la interfaz —como iniciar sesión, publicar un comentario o hacer una compra— el backend se encarga de procesar esa solicitud. Una de las funciones clave del backend es crear y gestionar APIs (interfaces de programación), que permiten que el frontend se conecte con los datos del sistema. Por ejemplo, si en una tienda online un usuario quiere ver sus pedidos, el frontend hace una solicitud a la API del backend, que busca esa información en la base de datos y la devuelve para ser mostrada en pantalla.
El backend también se encarga de tareas importantes como la autenticación de usuarios, el manejo de permisos, la validación de datos, el procesamiento de archivos, la encriptación de información sensible (como contraseñas), y la integración con servicios externos como pasarelas de pago o servicios de correo electrónico.

Conclusiones

Esta sección debe orientarse a evidenciar claramente las principales ideas generadas, propuestas que deriven del análisis realizado y si existen, expresar las conclusiones o aportes que autor quiera destacar.

Enfatizando, lo importante es destacar las principales posturas fundamentadas del autor, que desea transmitir a los lectores.

Adicionalmente, pueden incluirse preguntas abiertas a la reflexión y debate, temas concatenados con el tema expuesto o recomendaciones para profundizar en la temática expuesta.

A continuación, procedo a explicar los métodos de las clases.

Time:

- to_dict(): Este método devuelve como un diccionario los atributos de la clase.

Course:

- setNotes(): Configura el valor del atributo note del objeto recibiendo un int.
- setTutor(): Configura el valor del atributo tutor recibiendo un objeto Tutor.

Tutor:

- getCourse(): Devuelve un objeto Course de la lista courses dado un int.
- to_dict(): Este método devuelve como un diccionario los atributos de la clase.

Student:

- getCourse(): Devuelve un objeto Course de la lista courses dado un int.

- to_dict(): Este método devuelve como un diccionario los atributos de la clase.

Activity:

- to_dict(): Este método devuelve como un diccionario los atributos de la clase.

Matrix:

- checkHead(): Este método revisa si el nodo cabeza ya existe.
- add(): Agrega un elemento nuevo a la matriz
- display(): este método imprime en consola los elementos de la matriz.
- getColum(): Devuelve todos los elementos de la matriz que están en la misma columna.
- getRow(): Devuelve todos los elementos de la matriz que están en la misma fila.
- setAct(): Configura el valor de la variable act del objeto Matriz.

Admin:

- getCourses(): Devuelve un objeto Course de la lista courses dado un int.
- getTutor(): Devuelve un objeto Tutor de la lista tutors dado un int
- getStudent(): Devuelve un objeto Student de la lista students dado un int
- getMatrix(): devuelve un objeto Matrix de la lista notes dado un int

HeadList:

- addHeadNode(): agrega un nuevo nodo cabeza a la headlist
- getHead(): devuelve el nodo cabeza dado un int

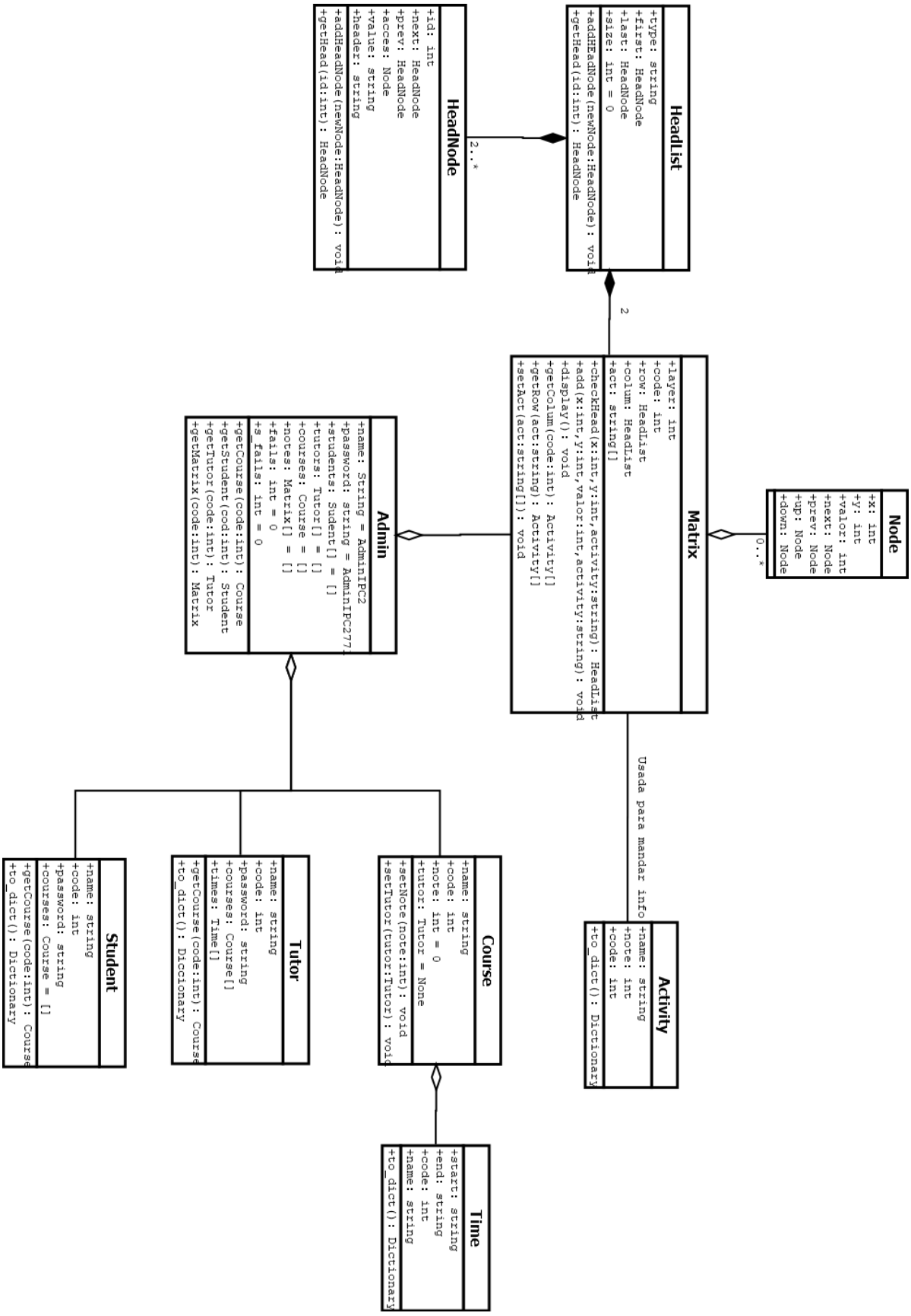
HeadList:

- addHeadNode(): agrega un nuevo nodo cabeza a la headlist
- getHead(): devuelve el nodo cabeza dado un int

Desarrollo del tema

- Se logró implementar de forma efectiva una arquitectura cliente-servidor utilizando Django y Flask, permitiendo una interacción fluida entre el frontend y el backend.
- La aplicación de POO permitió encapsular adecuadamente la lógica del sistema, especialmente en la construcción y manipulación de la matriz dispersa.
- Gracias al uso de Plotly, se desarrollaron reportes visuales que permiten analizar el desempeño académico por actividad y por estudiante.

Anexos


HeadList
+type: string
+first: HeadNode
+last: HeadNode
+size: int = 0
+addHEadNode(newNode:HeadNode): void
+getHead(id:int): HeadNode
HeadNode
+id: int
+next: HeadNode
+prev: HeadNode
+acces: Node
+value: string
+header: string
+addHeadNode(newNode:HeadNode): void
+getHead(id:int): HeadNode
2..*
2
Matrix
+layer: int
+code: int
+row: HeadList
+colum: HeadList
+act: string[]
+cheKHead(x:int,y:int,activity:string): HeadList
+add(x:int,y:int,valor:int,activity:string): void
+display(): void
+getColum(code:int): Activity[]
+getRow(act:string): Activity[]
+setAct(act:string[]): void
Node
+x: int
+y: int
+valor: int
+next: Node
+prev: Node
+up: Node
+down: Node
0..*
Admin
+name: String = AdminIPC2
+password: string = AdminIPC277
+students: Sudent[] = []
+tutors: Tutor[] = []
+courses: Course = []
+notes: Matrix[] = []
+fails: int = 0
+s_fails: int = 0
+getCourse(code:int): Course
+getStudent(cod:int): Student
+getTutor(code:int): Tutor
+getMatrix(code:int): Matrix
Usada para mandar info
Activity
+name: string
+note: int
+code: int
+to_dict(): Dictionary
Course
+name: string
+code: int
+note: int = 0
+tutor: Tutor = None
+setNote(note:int): void
+setTutor(tutor:Tutor): void
Tutor
+name: string
+code: int
+password: string
+courses: Course[]
+times: Time[]
+getCourse(code:int): Course
+to_dict(): Dicciomary
Student
+name: string
+code: int
+password: string
+courses: Course = []
+getCourse(code:int): Course
+to_dict(): Dictionary
Time
+start: string
+end: string
+code: int
+name: string
+to_dict(): Dictionary

Referencias bibliográficas

- Raj, A. (2023, 30 enero). *Matriz dispersa en Python*. Delft Stack. https://www.delftstack.com/es/howto/python/sparse-matrix-in-python/
- Remigio Huarcaya Almeyda. (2020, 25 marzo). *matrices con python* [Vídeo]. YouTube. https://www.youtube.com/watch?v=OW0FT3BUdxA

C. J. Date, (1991). *An introduction to Database Systems.* Addison-Wesley Publishing Company, Inc.