PROYECTO 2 - IPC2 “IPCArt-Studio”

202202410 – Marcos Daniel Bonifasi de León

Resumen

El arte puede manifestarse de diversas maneras, y hoy en día existen medios accesibles para presentarlo. IPCArt-Studio es ahora una plataforma web que facilita la creación y gestión de arte en píxeles. Esta migración permite a los usuarios acceder desde cualquier dispositivo con conexión a Internet, garantizando mayor flexibilidad y accesibilidad.

La plataforma cuenta con un área de administración donde se pueden agregar solicitantes, ya sea de forma individual o mediante carga masiva. Los usuarios acceden al sistema mediante un login seguro y personalizado. A través de una interfaz web amigable, los solicitantes pueden enviar archivos XML que contienen detalles para generar una matriz dispersa y crear la imagen solicitada.

Una vez validada, el sistema genera la pieza, registra a los involucrados y guarda la imagen en la galería del solicitante. Esta migración al entorno web ha simplificado la creación de arte digital, promoviendo la colaboración con los solicitantes de manera eficiente, segura y accesible desde cualquier lugar.

Palabras clave

XML, arte, plataformas, pixeles, Python, galeria, Graphviz, Web, Django, Flask, HTTP, Jinja.

Abstract

Art can manifest itself in a variety of ways, and today there are accessible means to present it. IPCArt Studio is now a web platform that facilitates the creation and management of pixel art. This migration allows users to access it from any device with an Internet connection, ensuring greater flexibility and accessibility.

The platform has an administration area where applicants can be added, either individually or via bulk upload. Users access the system through a secure and personalized login. Through a user-friendly web interface, applicants can submit XML files containing details to generate a sparse matrix and create the requested image.

Once validated, the system generates the part, registers those involved and saves the image in the requester's gallery. This migration to the web environment has simplified the creation of digital art, promoting collaboration with requesters in a way that is efficient, secure and accessible from anywhere.

Keywords

XML, art, platforms, pixels, Python, gallery, Graphviz, Web, Django, Flask, HTTP, Jinja.

Introducción

Se desarrolló una aplicación web llamada IPCArt-Studio, diseñada para facilitar la creación, gestión y administración de arte en píxeles. La plataforma permite a los usuarios cargar configuraciones a través de archivos XML, que contienen la información necesaria para registrar a usuarios normales en el sistema.

El sistema ya no cuenta con artistas; en su lugar, los usuarios normales envían solicitudes para la generación de imágenes en píxeles, las cuales se gestionan mediante una pila y una cola, asegurando un flujo de trabajo ordenado y eficiente. Un usuario administrador es el encargado de supervisar y validar las solicitudes enviadas, garantizando el correcto funcionamiento de la plataforma.

La aplicación utiliza matrices dispersas para optimizar el uso de memoria al momento de generar las imágenes solicitadas. Estas imágenes se almacenan en una lista doblemente enlazada que organiza y permite a los usuarios navegar de manera fluida a través de su galería personal.

La migración al entorno web proporciona una interfaz amigable y accesible desde cualquier dispositivo con conexión a Internet. Además, las visualizaciones de las estructuras de datos, como listas y pilas, se implementan mediante Graphviz para ofrecer una experiencia visual clara y funcional.

Desarrollo del tema

La aplicación ahora consiste en una aplicación web que permite administrar usuarios y, dependiendo del tipo de usuario, muestra diferentes características y funcionalidades. Los usuarios normales pueden realizar subidas de achivos y crear piezas de arte en píxeles, mientras que el usuario administrador se encarga de gestionar y validar los usuarios, asi como ver estadisticas sobre los mismos.

Se ha optado por utilizar el lenguaje de programación Python junto con Django para el desarrollo del frontend y Flask para el backend, aprovechando sus capacidades para crear aplicaciones web robustas y escalables. Esto proporciona una interfaz moderna y accesible desde cualquier navegador, optimizando la experiencia del usuario.

Para la elaboración de la solución, los usuarios pueden cargar archivos con extensión y estructura XML, los cuales deben seguir un formato definido para garantizar la correcta importación y manejo de los datos dentro de la plataforma.

En el área de administración, el usuario a encargado puede cargar archivos con las siguientes extructura:

1. Listado de solicitantes/usuarios

```xml
<?xml version='1.0' encoding='utf-8'?>
<solicitantes>
    <solicitante id="IPC-001" pwd="3267">
        <NombreCompleto>Carlos Ruiz</NombreCompleto>
        <CorreoElectronico>carlos.ruiz@example.com</CorreoElectronico>
        <NumeroTelefono>67984298</NumeroTelefono>
        <Direccion>Ciudad de Guatemala, Guatemala</Direccion>
        <perfil>https://cdn-icons-png.flaticon.com/512/3135/3135768.png</perfil>
    </solicitante>
    <!--y muchos más solicitantes-->
</solicitantes>
```

Figura 1. XML solicitantes/usuarios

Fuente: elaboración propia

Donde:

<solicitantes> = Etiqueta raíz que agrupa a todos los soliciantes.

<solicitante id="IPC-001" pwd="3267"> = Representa un solicitante con atributos id y pwd.

<NombreCompleto>: El nombre del solicitante.

<CorreoElectronico>: El email del solicitante.

<NumeroTelefono>: El número de teléfono del solicitante.

<Direccion>: La dirección del soliciante.

<perfil>: Link hacia el perfil del usuario.

Obtenido este formato, el usuario administrador puede cargar los usuarios necesarios por medio de lo que contiene cada uno de los archivos.

Listas donde se almacena la información necesaria, es decir, nombre de usuario y contraseña, para que los solicitantes puedan acceder al sistema a través del Login.

Para el área de solicitante, el usuario respectivo puede cargar un tipo de archivo XML, el cual contiene toda la estructura necesaria para, luego, generar la pieza de arte en pixeles. Este archivo también se utiliza para generar la matriz dispersa.

En la siguiente imagen se muestra un ejemplo de la estructura que debe llevar este archivo.

```xml
<figura>
  <nombre id="1">Figura1</nombre>
  <diseño>
    <pixel fila="0" col="2">#000000</pixel>
    <pixel fila="0" col="3">#000000</pixel>
    <pixel fila="0" col="4">#000000</pixel>
    <pixel fila="1" col="1">#FFFFFF</pixel>
    <!-- Más pixeles aquí -->
  </diseño>
</figura>
```

Figura 5. XML para generar la figura de PixelArt

Fuente: elaboración propia, 2024

Donde:

<figura> = Representa la etiqueta raíz que agrupa a todos los pixeles que serán graficados.

<nombre id=”id”> = Contiene el nombre de la figura que se graficará. Contiene el atributo “id”, el cual representa el identificador único de la figura.

<diseño> = Agrupa todos los pixeles que conforma la figura.

<pixel> = Representa la coordenada en la cual irá el color especificado. Contiene los atributos “fila” y “col”, los cuales especifican la fila y columna, respectivamente, donde se localizará el color especificado. Es importante mencionar que el color deberá estar escrito en notación hexadecimal.

Obtenido este formato, el usuario solicitante puede cargar las figuras que guste.

En el Backend se programaron los siguientes endpoints:

Endpoint	**Verbo**
/sign_in.json	POST
/users/load.json	POST
/users.json	GET
/users.xml	GET
/users/stats	GET
/users/<string:uid>/gallery.json	GET
/user/<string:uid>/images.json	POST
/user/<string:uid>/images/edit.json	POST

Se realizaron los siguientes algoritmos:

a. Sign in: esta algoritmo valida la información que el usuario provee en el fronted y la compara con la información real en el backend. Si los datos son correctos, el login es exitoso.
b. Procesar Archivo Solicitantes/Usuarios: Este algoritmo se encarga inicialmente de leer los archivos XML por medio de la librería Element Tree y verifica si cada una de las tags pertenecientes a la estructura definida para solicitantes es correcta. Si lo es, importa cada uno de los datos y los agrega a la lista correspondiente.
c. Ver usuarios XML: este algortimo hace una consulta en las listas y muestra en formato XML todos los usuarios en el sistema y las imágenes que estos hayan creado.
d. Ver usuario: este algoritmo muestra una galeria de cards con todos los usuarios en el sistema, asi como mostrando su información específica.
e. Ver estadísticas: este algoritmo hace la consulta respectiva en el sistema y retorna los datos necesarios para dos graticas que mas tarde se mostrarán en la vista del administrador. Estas consultas retornan el oop 3 de usuarios con más imágenes cargadas y la cantidad de imágenes editadas por cada usuario en orden descendente.
f. Procesar Archivo Figuras: Este algoritmo se encarga inicialmente de leer los archivos XML por medio de la librería Element Tree y verifica si cada una de las tags pertenecientes a la estructura definida para las figuras es correcta. Si lo es, importa cada uno de los datos y los agrega a la lista correspondiente. Estas figuras son caargadas en la pila perteneciente el cada solicientante, respectivamente.
g. Galeria: Se encuentra en la vista de cada solicitante. Esta funcionalidad se encarga de mostrar de manera grafica todas las figuras generadas por los usuaios, mostrando solo aquellas solicitadas por el solicitante registrado.
h. Graficar: este algoritmo esta implementado en todas las listas mencionadas anteiormente, y su funcion es generar un grafo mostrando graficamente como estan distribuidos los datos en cada una de las estructuras de datos. Cuando ya esta generada la imagen, esta se muestra dinamicamente por el sistema para que el usuario pueda visualizarla.
i. Editor de imágenes: en este algoritmo se puede modificar la escala a las imágenes, a escala de grises o sepia.
j. Ayuda: en esta sección solo se muestra la información del desayador y el manual de usuario.
k. Cerrar sesión: este algoritmo borra toda la información de sesión del usuario en las cookies y retorna al cliente a la vista del login.

Conclusiones

La migración de IPCArt-Studio a una aplicación web ha permitido que la plataforma se adapte a las necesidades actuales de accesibilidad y escalabilidad. El uso de una arquitectura cliente-servidor garantiza que los usuarios puedan interactuar con el sistema desde cualquier lugar con conexión a Internet, mejorando significativamente la experiencia del usuario y su alcance.

La elección de Flask para el backend y Django para el frontend asegura una separación clara entre las capas de la aplicación, lo que facilita el desarrollo, mantenimiento y escalabilidad del proyecto. Estas herramientas, junto con el uso de estructuras de datos avanzadas y expresiones regulares, garantizan un procesamiento de datos eficiente y un sistema robusto.

La aplicación emplea matrices dispersas para manejar de manera eficiente las imágenes de arte en píxeles, maximizando el uso de recursos de memoria. Asimismo, la persistencia mediante archivos XML asegura una solución flexible y portable que no depende de bases de datos relacionales tradicionales.

La incorporación de herramientas como Plotly para la generación de estadísticas visuales dinámicas proporciona a los usuarios y al administrador una visión clara del desempeño del sistema. Estas visualizaciones permiten analizar datos clave de manera interactiva, facilitando la toma de decisiones informadas.

La implementación de funcionalidades como la carga masiva de usuarios, la validación de datos mediante expresiones regulares y la generación de imágenes personalizadas demuestra un enfoque orientado a satisfacer las necesidades de los usuarios. Además, el proyecto cumple con los estándares académicos establecidos, promoviendo el aprendizaje práctico y el desarrollo profesional en programación orientada a objetos.

Referencias bibliográficas

USAC, T.d.(29 de diciembre de 2024). Proyecto 2, UEDi.
Obtenido de *https://uedi.ingenieria.usac.edu.gt/campus/mod/resource/view.php?id=877675*

Anexos

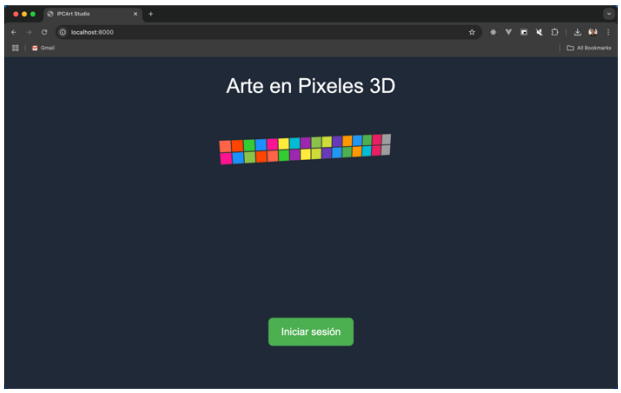

Figura 11. Inicio de la aplicación

Fuente: elaboración propia, 2024

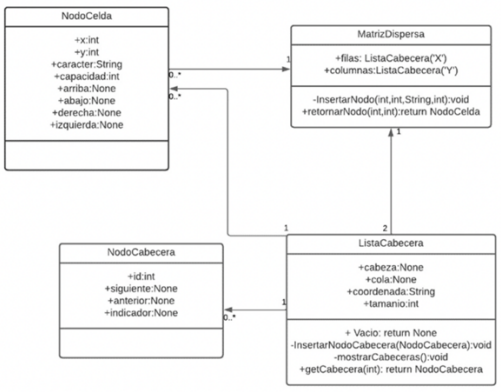

Figura 12. Modelo del tipo de dato MatrizDispersa

Fuente: Elaboración propia, 2024

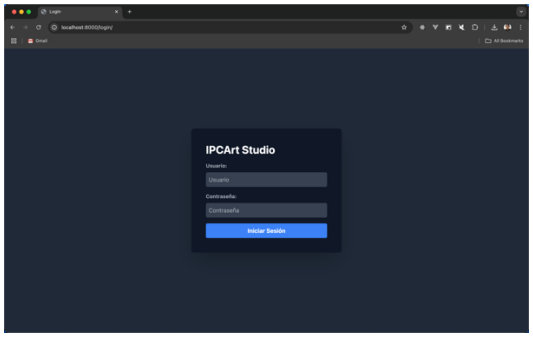

Figura 13. Login

Fuente: elaboración propia, 2024

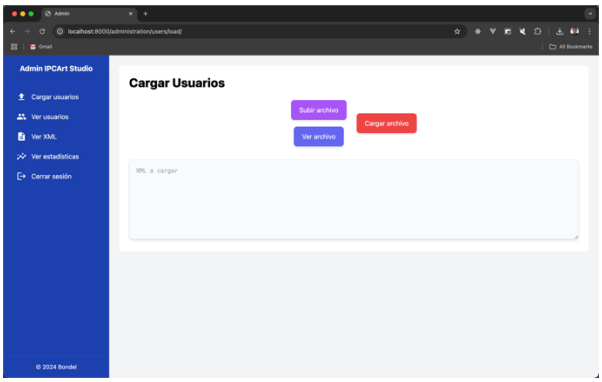

Figura 14. Admin: Cargar usuarios Administración

Fuente: elaboración propia, 2024

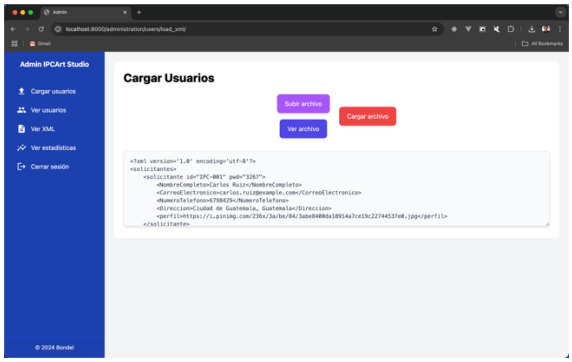

Figura 15. Admin: Usuarios XML

Fuente: elaboración propia, 2024

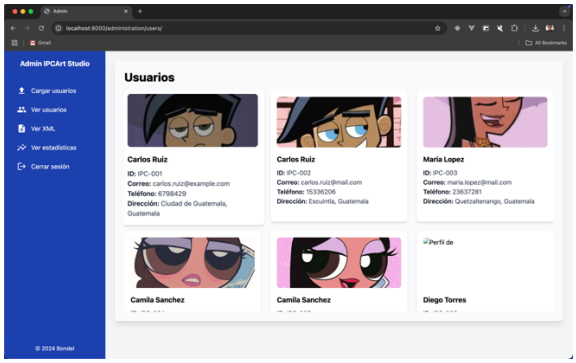

Figura 16. Admin: Galeria usuarios

Fuente: elaboración propia, 2024

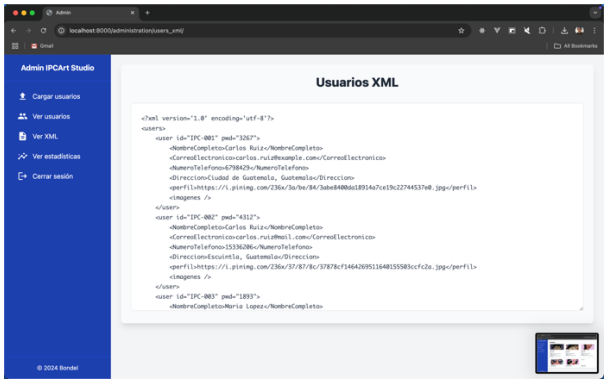

Figura 17. Admin: Usuarios XML

Fuente: elaboración propia, 2024

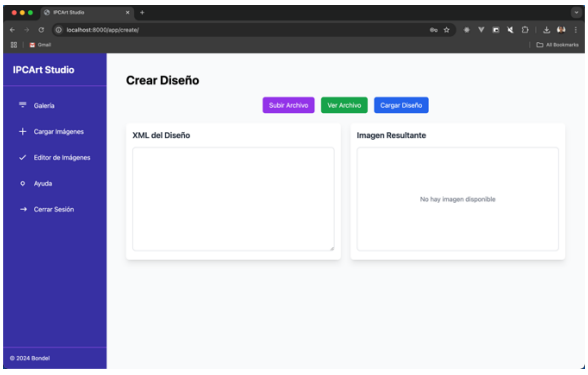

Figura 18. Usuario: Crear diseño

Fuente: elaboración propia, 2024

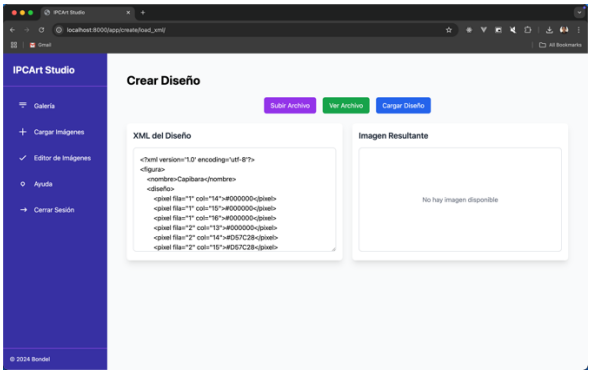

Figura 19. Usuario: Crear diseño XML

Fuente: elaboración propia, 2024

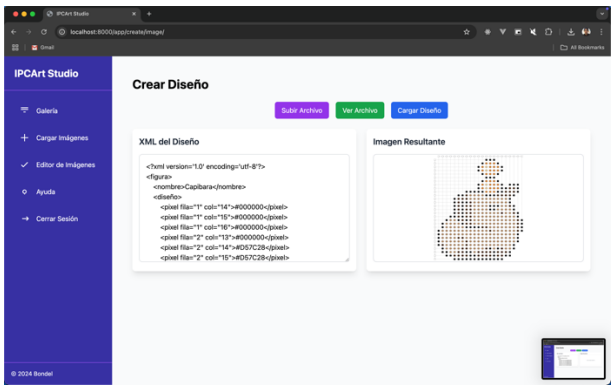

Figura 20. Usuario: Diseño creado

Fuente: elaboración propia, 2024

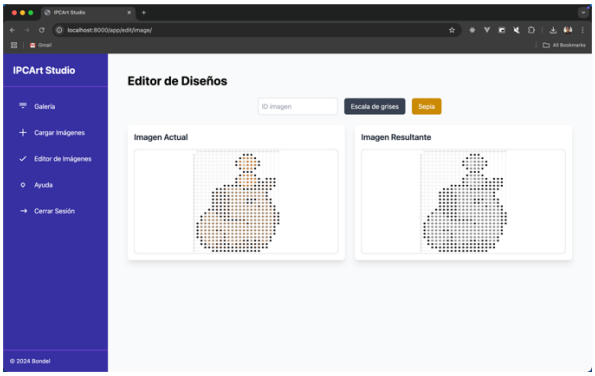

Figura 21. Usuario: Diseño Escala de Grises

Fuente: elaboración propia, 2024

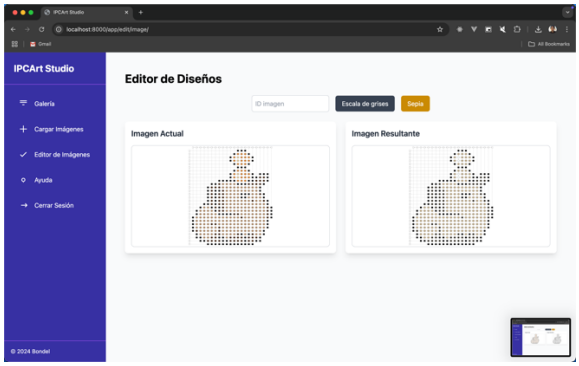

Figura 22. Usuario: Diseño Sepia

Fuente: elaboración propia, 2024

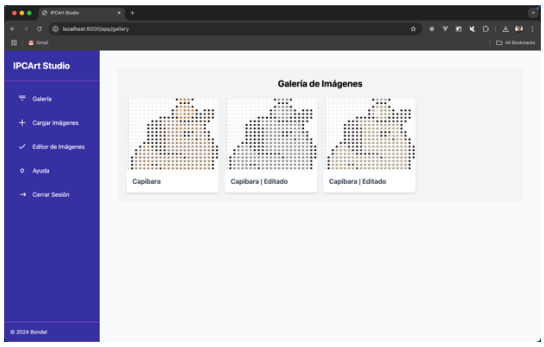

Figura 23. Usuario: Galeria

Fuente: elaboración propia, 2024

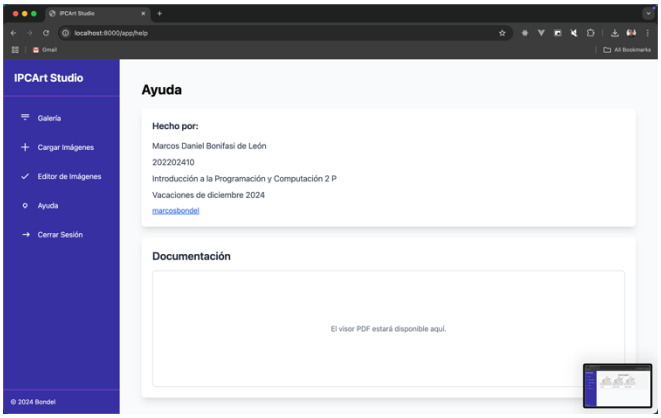

Figura 23. Usuario: Vista de ayuda

Fuente: elaboración propia, 2024

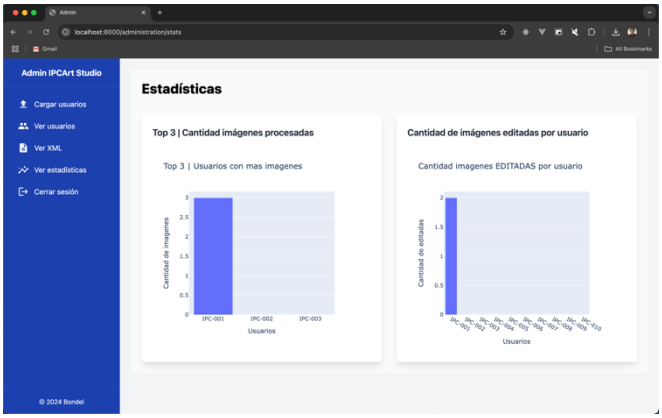

Figura 24. Admin: Estadisticas

Fuente: elaboración propia, 2024

Figura 25. Disebar administración

Fuente: elaboración propia, 2024

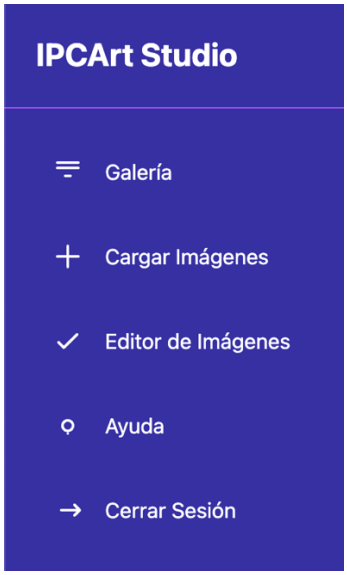

Figura 26. Disebar usuario

Fuente: elaboración propia, 2024

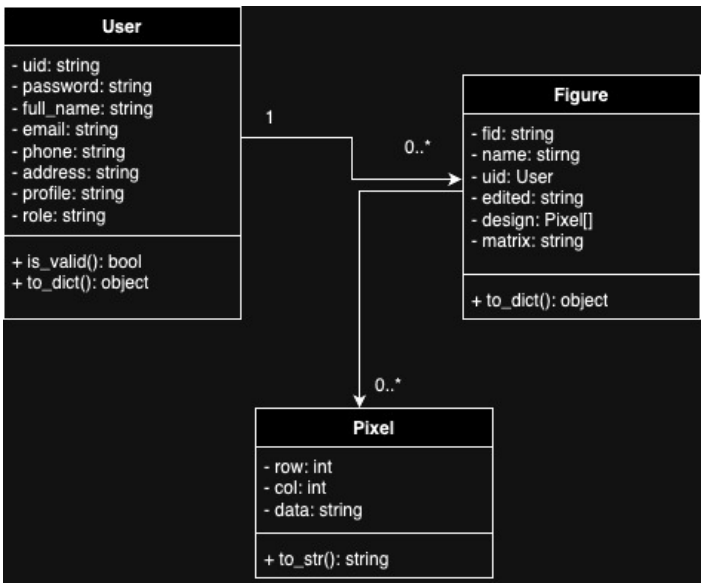

Figura 27. Diagrama de clases UML

Fuente: elaboración propia, 2024