PROYECTO 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 una plataforma que facilita la creación y gestión de arte en píxeles. Cuenta con un área de administración donde se pueden agregar artistas y solicitantes, ya sea individualmente o por carga masiva. Los usuarios acceden Pal sistema mediante un login seguro. Mediante una interfaz amigable, los solicitantes envían archivos XML que contienen detalles para generar una matriz dispersa y crear la imagen solicitada. La solicitud debe ser aprobada por un artista; una vez validada, el sistema genera la pieza, registra a los involucrados y guarda la imagen en la galería del solicitante. IPCArt-Studio simplifica la creación de arte digital, promoviendo la colaboración entre artistas y solicitantes de manera eficiente y segura.

Palabras clave

XML, arte, plataformas, pixeles, Python, galeria, Graphviz, sistemas.

Abstract

Art can manifest in various ways, and nowadays there are accessible means to present it. IPCArt-Studio is a platform that facilitates the creation and management of pixel art. It includes an administration area where artists and applicants can be added, either individually or through bulk uploads. Users access the system through a secure login. Through a user-friendly interface, applicants send XML files containing details to generate a sparse matrix and create the requested image. The request must be approved by an artist; once validated, the system generates the artwork, records those involved, and saves the image in the applicant's gallery. IPCArt-Studio simplifies the creation of digital art, promoting collaboration between artists and applicants in an efficient and secure manner.

Keywords

XML, art, platforms, pixels, Python, gallery, Graphviz, systems.

Introducción

Se desarrolló un programa llamado IPCArt-Studio, diseñado para facilitar la creación, gestión y administración de arte en píxeles. La plataforma permite cargar configuraciones a través de archivos XML, que contienen la información necesaria para registrar a artistas y solicitantes en el sistema. Los artistas son los encargados de crear las piezas de arte solicitadas, mientras que los solicitantes 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.

La aplicación utiliza matrices dispersas para optimizar el uso de memoria al momento de generar las imágenes solicitadas, las cuales se almacenan en una lista doblemente enlazada que organiza y permite a los solicitantes navegar a través de su galería de imágenes de manera fluida. Los artistas también tienen acceso a las solicitudes a través de una lista circular, donde gestionan y visualizan las tareas asignadas.

Para la implementación de la interfaz gráfica, se utilizó la librería Tkinter con el lenguaje de programación Python, mientras que las visualizaciones de las estructuras de datos, como listas y pilas, se lograron utilizando Graphviz. IPCArt-Studio no solo promueve la organización y optimización de recursos, sino también la colaboración creativa entre artistas y solicitantes, ofreciendo una solución robusta, eficiente y funcional para la creación de pixel art.

Desarrollo del tema

La aplicación consisten en una aplicación de escritorio que permite administrar usuarios, y dependiendo rol de estos, muestra diferentes característica y funcionalidades. Los usuarios de tipo solicitantes pueden hacer solicitudes para crear piezas de arte en pixeles, y los artistas puede aceptarlas asi como generar la imagen final que será guardada en la galería del solicitante.

Se ha optado por el uso del lenguaje de programación Python debido a que es multiparadigma y multiplataforma ya que su aplicación es sencilla de utilizar y se tuvo como complemento la ayuda de la librería de Tkinter para desarrollar toda la interfaz gráfica de la aplicación.

Para la elaboración de la solución se proveen archivos con extensión y estructura XML, los cuales deben seguir una estructura definida para la correcta importación y manejo de los datos dentro de la aplicación.

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

1. Listado de solicitantes

```xml
<?xml version="1.0" encoding="UTF-8"?>
<solicitantes>
  <solicitante id="IPC-001" pwd="1234">
      <NombreCompleto>Juan Pérez</NombreCompleto>
      <CorreoElectronico>juan.perez@example.com</CorreoElectronico>
      <NumeroTelefono>12345678</NumeroTelefono>
      <Direccion>Ciudad de Guatemala, Guatemala</Direccion>
  </solicitante>
  ...
</solicitantes>
```

Figura 1. XML solicitantes

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.

2. Listado de artistas

```xml
<Artistas>
    <Artista id="ART-001" pwd="1234">
        <NombreCompleto>María López</NombreCompleto>
        <CorreoElectronico>maria.lopez@example.com</CorreoElectronico>
        <NumeroTelefono>87654321</NumeroTelefono>
        <Especialidades>Pixel Art, Animación</Especialidades>
        <NotasAdicionales>Disponible para proyectos urgentes.</NotasAdicionales>
    </Artista>
    <!-- Más artistas aquí -->
</Artistas>
```

Figura 2. XML artistas

Fuente: elaboración propia, 2024

Donde:

<Artistas> = Etiqueta raiz que agrupa a todos los artistas.

<Artista id="ART-593" pwd="9142"> = Representa un artistas con atributos id y pwd.

<NombreCompleto>: El nombre del artista.

<CorreoElectronico>: El email del artista.

<NumeroTelefono>: El número de teléfono del artista.

<Especialidades>: Especialidades del artista.

<NotasAdicionales>: Notas adicionales sobre el artista.

Obtenido este formato, el usuario administrador puede cargar los usuarios necesarios por medio de lo que contiene cada uno de los archivos archivo, los datos se guarda en las siguientes estructuras de datos abstractas:

1. Lista Solicitantes: Aquí se guardan los solicitantes o clientes.

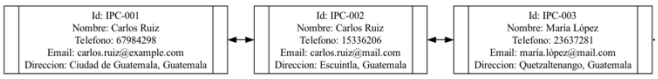

Figura 3. Representación gráfica Lista Doble Solicitantes

Fuente: elaboración propia, 2024

2. Lista Artistas: Aquí se guardan los artistas.

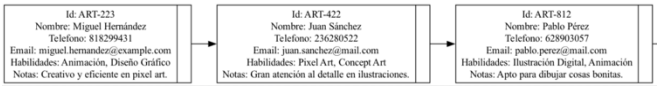

Figura 4. Representación gráfica Lista Simple Artistas

Fuente: elaboración propia, 2024

Cada una las listas mencionadas alamacenan la información necesaria, es decir, nombre de usuario y contraseña, para que tanto los solicitantes como artistas 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. Debido a que una solicitud para por diferentes etapas, dependiendo en cual etapa esta, así será como se almacenarán los datos.

A continuación, se muestran las diferentes estructuras de datos utilizadas en el área del solicitante.

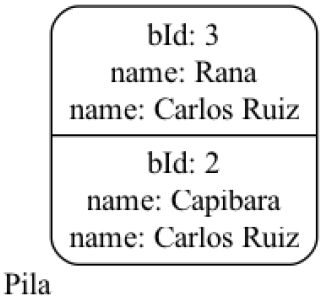

Figura 6. Pila que almacena las solicitudes

Fuente: elaboración propia, 2024

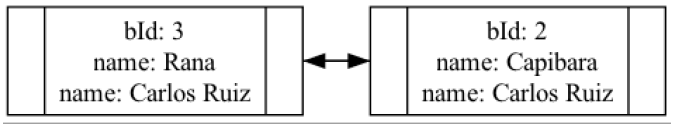

Figura 7.Lista doblemente enlazada para almacernar las solicitudes aceptadas.

Fuente: elaboración propia, 2024

En el caso del usuario artistas, este tiene el control sobre las solicitudes que son aceptadas, asi como de la generación de las piezas de arte en pixeles y recepción de las mismas hacia los solicitantes.

La forma en la que se maneja las solicitudes en el lado del artista es mediante una cola. Cuando un artista acepta una solicitud, esta se elimina de la cola, se genera la imagen en pixeles, y se guarda un registro tanto del lado del solicitante como del artista.

La cola tiene la siguiente representación.

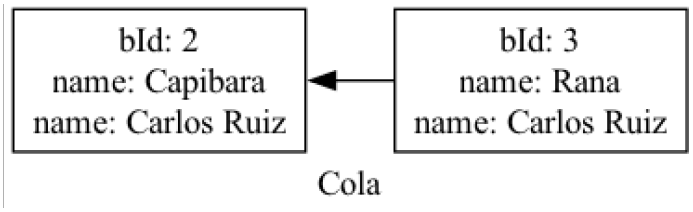

Figura 8.Cola para almacernar las solicitudes en proceso.

Fuente: elaboración propia, 2024

Como se mencionó, se mantiene referencia de las solicitudes que acepta cada artista, y para hacer esto se utiliza una lista circular simplemente enlazada, como se muestra a continuación:

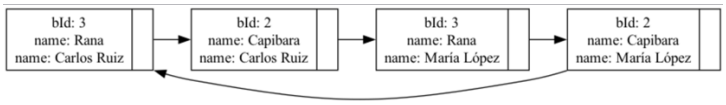

Figura 9.Lista circular para almacernar las solicitudes aceptadas por el artista.

Fuente: elaboración propia, 2024

Regresando al lado del solicitante, una vez sus solicitudes han sido aceptadas, tiene acceso a una lista doblemente en lazada donde pude ver detalle sobre su solicitud.

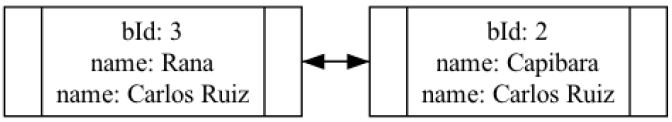

Figura 10. Lista doblemente enlazada para almacernar las solicitudes aceptadas

Fuente: elaboración propia, 2024

Además, para poder graficar la figura solicitada, se hace uso de una matriz dispersa. Para hacer esta matriz se necesitó de 2 listas de tipo cabecera junto con la creación de nodos celda para insertarlos en la matriz.

Con las estructuras ya creadas, se realizaron los siguientes algoritmos:

a. Procesar Archivo Solicitantes: 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.
b. Procesar Archivo Artistas: 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 artistas es correcta. Si lo es, importa cada uno de los datos y los agrega a la lista correspondiente.
c. 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.
d. Solicitar figura: este algoritmo funciona del lado del solicitante, y se encarga de eliminar la cima de la pila, y agregarla a la cola de solicitudes del sistema en general.
e. Aceptar solicitud: Este algoritmo funciona del lado del artista, y se encarga de trabajar con la cola de solicitudes del sistema completo. Cuando un artista acepta una solicitud, esta es eliminada de la cola, se agrega una copia a la lista del artista que la aceptó, se agrega a la lista del usuario y también a su galería para poder visualizarse posteriormente.
f. Galeria: Se encuentra en la vista de cada solicitante. Esta funcionalidad se encarga de mostrar de manera grafica todas las figuras generadas por los artistas, mostrando solo aquellas solicitadas por el solicitante registrado.
g. 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 abirta dinamicamente por el sistema para que el usuario pueda visualizarla.

Conclusiones

El desarrollo del proyecto permitió comprender y aplicar los TDA, como listas enlazadas, pilas y colas, para gestionar de manera eficiente los datos relacionados con artistas, solicitantes y solicitudes, demostrando su versatilidad en aplicaciones prácticas como la creación y administración de arte en píxeles.

El uso de una pila para acumular las solicitudes y una cola para procesarlas en orden de llegada optimizó el flujo de trabajo del sistema. Esto asegura un manejo justo y estructurado de las solicitudes, reforzando la importancia de estas estructuras en la programación orientada a objetos.

La integración de Python como lenguaje de desarrollo y la utilización de archivos XML como insumos demostraron ser una combinación eficaz para gestionar configuraciones y datos. Esto facilitó la generación de matrices dispersas para representar imágenes en píxeles, optimizando el uso de memoria y recursos computacionales.

La implementación de Graphviz para graficar las estructuras de datos permitió validar visualmente el correcto funcionamiento del sistema. Este enfoque, combinado con la programación orientada a objetos en Python, aseguró la creación de un proyecto robusto, funcional y alineado con las necesidades prácticas del arte digital y la colaboración entre usuarios.

Referencias bibliográficas

USAC, T.d.(7 de Marzo de 2022). Proyecto 2, UEDi. Obtenido de *https://uedi.ingenieria.usac.edu.gt/campus/mod/resource/view.php?id=876299*

Anexos

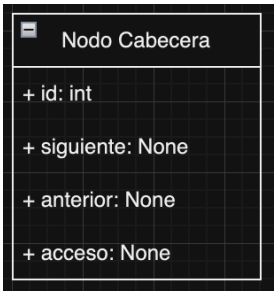

Figura 11..Modelo del tipo de dato NodoCabecera

Fuente: elaboración propia, 2024

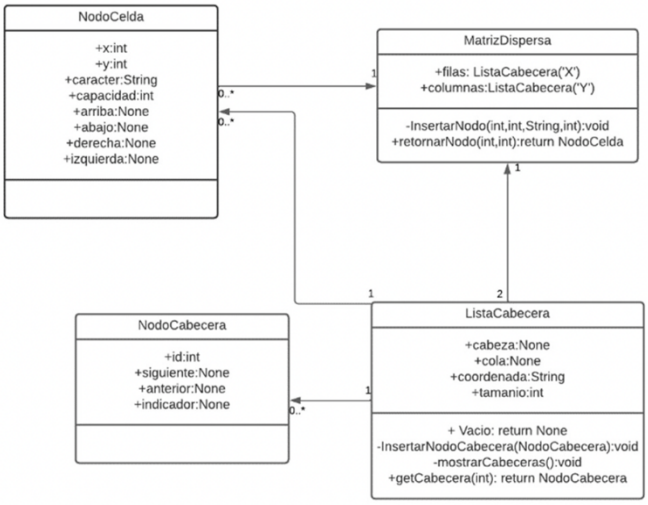

Figura 12..Modelo del tipo de dato MatrizDispersa

Fuente: https://uedi.ingenieria.usac.edu.gt/campus/mod/resource/view.php?id=872779

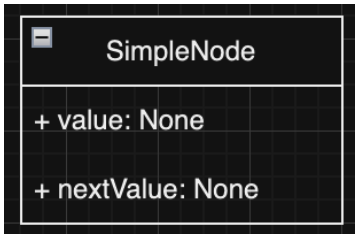

Figura 13..Modelo del tipo de dato SimpleNode

Fuente: elaboración propia, 2024

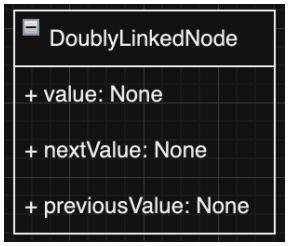

Figura 14..Modelo del tipo de dato DoublyLinkedNode

Fuente: elaboración propia, 2024

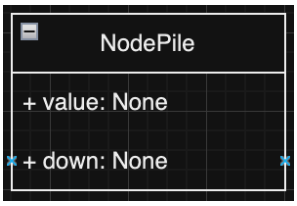

Figura 15..Modelo del tipo de dato NodePile

Fuente: elaboración propia, 2024

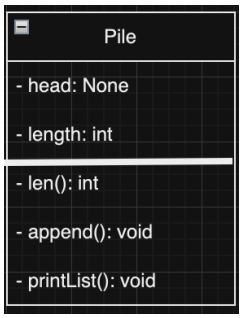

Figura 16..Modelo de la Pila

Fuente: elaboración propia, 2024

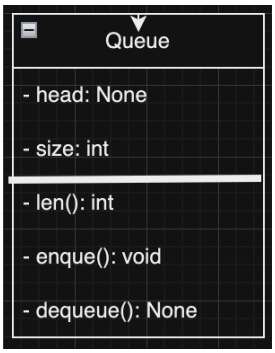

Figura 17..Modelo de la Queue

Fuente: elaboración propia, 2024

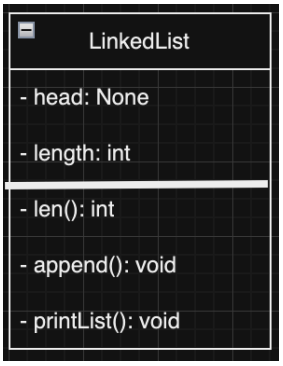

Figura 18..Modelo de la LinkedList

Fuente: elaboración propia, 2024

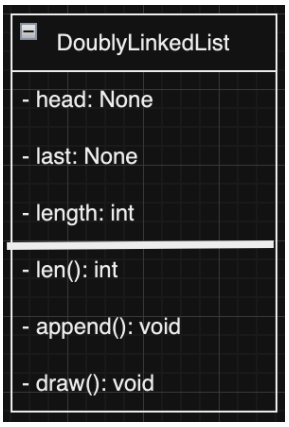

Figura 19..Modelo de la DoublyLinkedList

Fuente: elaboración propia, 2024

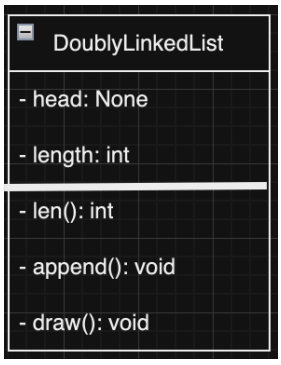

Figura 20..Modelo de la CircularLinkedList

Fuente: elaboración propia, 2024

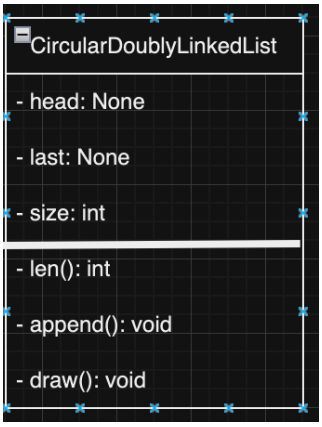

Figura 21..Modelo de la CircularDoublyLinkedList

Fuente: elaboración propia, 2024

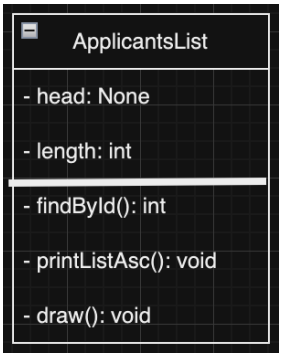

Figura 22..Modelo de la ApplicantsList

Fuente: elaboración propia, 2024

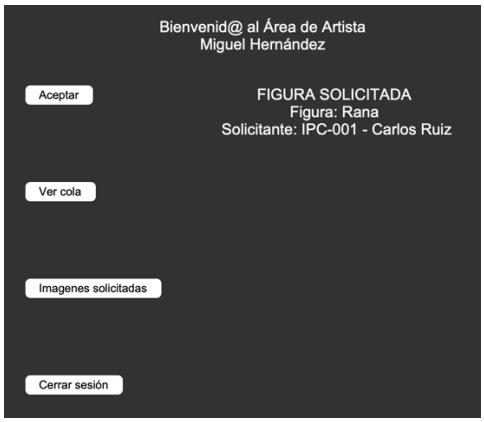

Figura 23. Interfaz de artista que muestra la cola

Fuente: elaboración propia, 2024

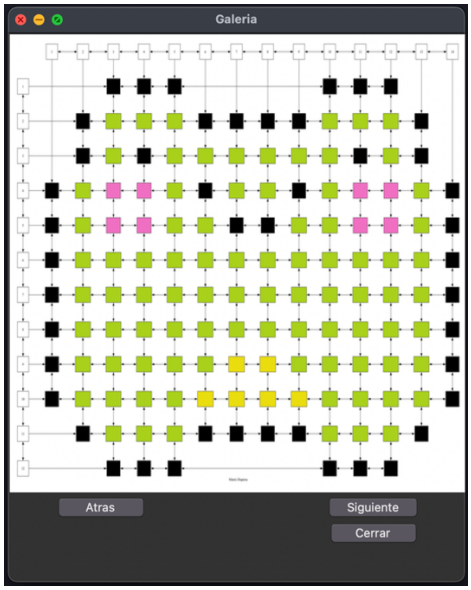

Figura 23. Galeria de imágenes en el area del solicitante que muestra una rana en pixeles.

Fuente: elaboración propia, 2024

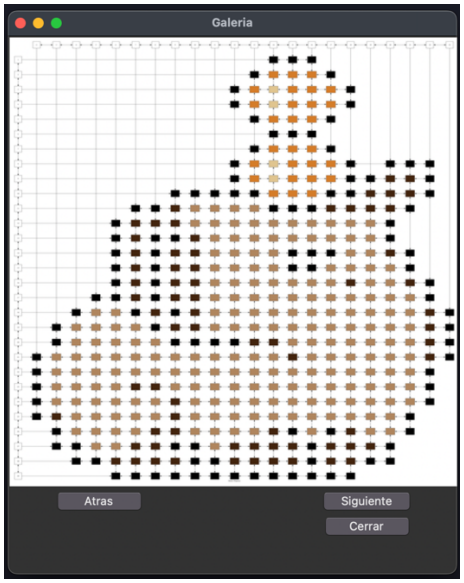

Figura 24. Galeria de imágenes en el area del solicitante que muestra una rana en pixeles.

Fuente: elaboración propia, 2024

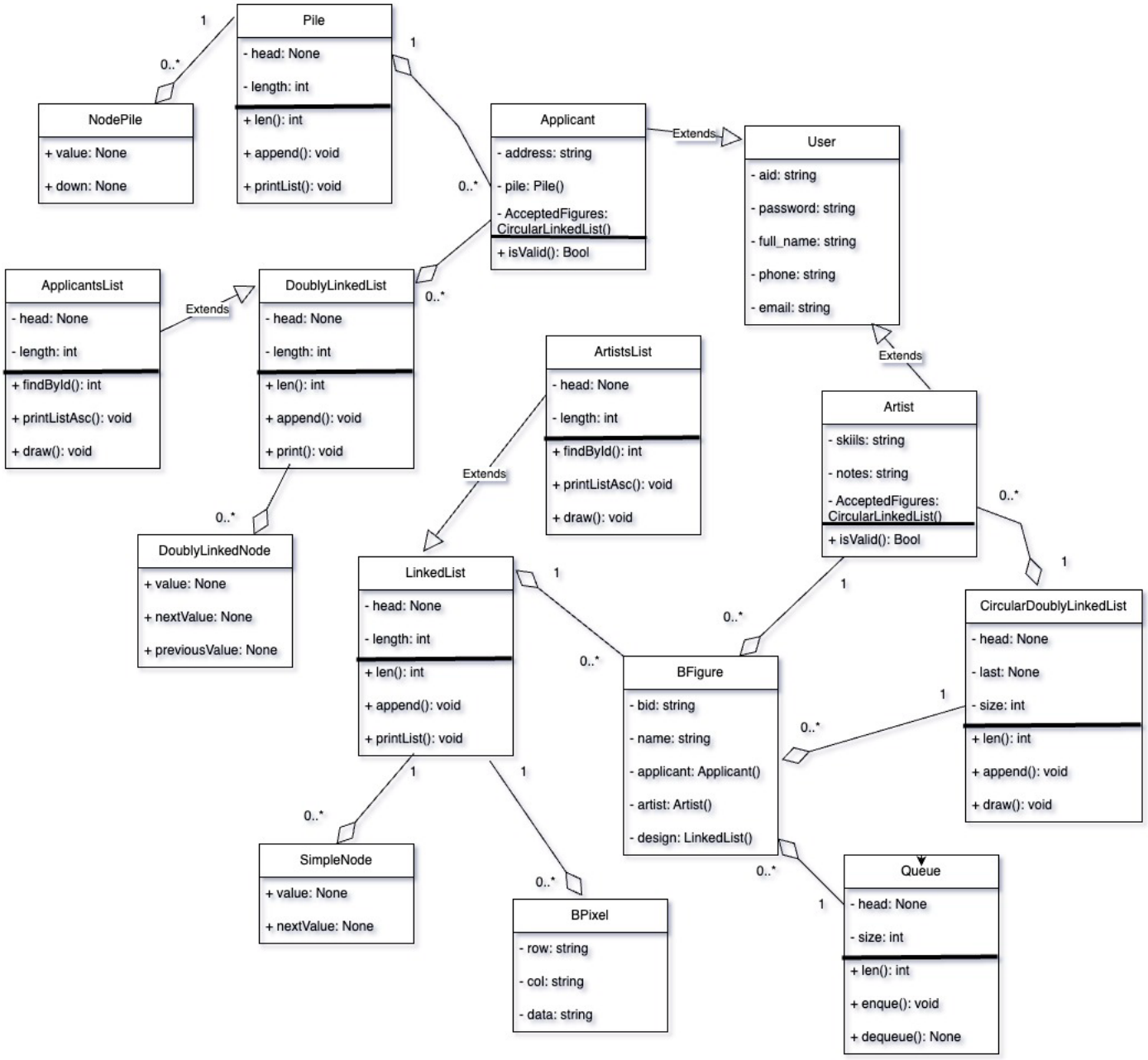

Figura 25. Diagrama de clases

Fuente: elaboración propia, 2024