UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

CENTRO UNIVERSITARIO DE OCCIDENTE

DIVISIÓN DE CIENCIAS DE LA INGENIERÍA

Manual Técnico

BoxBuilder Tc

Nombre: Erikson Denilson Orozco Monterroso

Carnet: 202131770

Estructura de Datos

Practica:

Totito Chino

Quetzaltenango 27 de ago. de 25

1. Introducción al Proyecto

BoxBuilderTC es una aplicación de escritorio desarrollada en C++ utilizando el framework Qt 6. Es un juego estratégico basado en el clásico *Dots and Boxes*, donde el objetivo es completar cuadros en un tablero. Cada jugador dispone de poderes especiales que se obtienen del tablero los cuales alteran la mecánica del juego, aumentando la complejidad estratégica.

El proyecto está organizado bajo un patrón de diseño MVC (Modelo-Vista-Controlador), con la siguiente estructura:

- Modelos: Contienen la lógica de los jugadores, tablero, poderes y nodos del juego, esto de manera nativa, sin uso de librerías como vector, todo fue manejado con punteros.
- Controladores: Gestionan las operaciones de los modelos, incluyendo la manipulación de vectores <Arreglos dinámicos> y la aplicación de poderes.
- Vistas: Implementadas mediante archivos .ui de Qt Designer y la clase MainWindow, responsables de la interacción visual con el usuario.

Todo el proyecto se implementa usando C++ nativo, sin dependencias externas para la manipulación de estructuras de datos.

2. Tecnologías Utilizadas

- Lenguaje de Programación: C++ (estándar moderno, memoria manual con new y delete)
- Framework de Interfaz Gráfica: Qt 6.x
- IDE: Qt Creator (incluye Qt Designer)
- Sistema de Compilación: CMake

Qt se encargó de estas instalaciones: https://www.qt.io/download-dev

Qt proporciona todas las herramientas para la construcción de interfaces gráficas ricas y multiplataforma, mientras que CMake simplifica la compilación y el manejo de dependencias.

3. Entorno de Desarrollo

3.1. Descargas e Instalación

Para desarrollar o compilar BoxBuilderTC necesitas:

3.1.1. Qt Framework

- Descargar desde: Qt Official Downloads: https://www.qt.io/download-dev
- Durante la instalación:
    - Selecciona la versión Qt 6.x correspondiente al proyecto. (Version 6.9 en adelante es la recomendación para BoxBuilderTC)
    - Incluye el compilador MinGW (Windows) o GCC/Clang según tu sistema.
    - Instala Qt Creator IDE.

3.1.2. Compilador de C++

- Windows: MinGW (incluido con Qt) o Visual Studio Build Tools (No es tan recomendable usara Visual Studio IDE).
- macOS: Clang (incluido con Xcode Command Line Tools).
- Linux: GCC (puede instalarse con sudo apt-get install build-essential).

3.2. Requisitos del Sistema (Recomendados)

- Sistema Operativo: Windows 10/11, macOS 12+, Ubuntu 20.04+, Fedora 41+
- Procesador: CPU de 64 bits
- Memoria RAM: 8 GB o más
- Espacio en Disco: 10 GB mínimo

4. Estructura del Proyecto

La organización de BoxBuilderTC facilita la modularidad y escalabilidad:

BoxBuilderTC/

├─ CMakeLists.txt # Configuración de compilación

├─ main.cpp # Entrada principal

├─ mainwindow.* # Interfaz principal (UI + lógica)

├─ viewplay.* # Vistas del tablero y controles

├─ models/ # Modelos de jugadores, nodos y tablero

├─ methods/ # Funciones y estructuras: listas, pilas, colas

├─ powers/ # Controladores y definición de poderes

├─ utils/ # Componentes gráficos y utilidades

5. Diagrama de Tipos de Datos Abstractos (TAD)

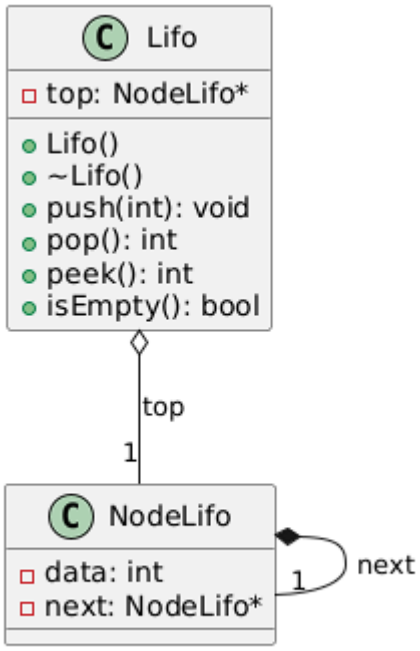
El juego utiliza estructuras de datos nativas implementadas manualmente:

5.1. Lista Enlazada

- Representa el tablero y la disposición de nodos.
- Cada nodo apunta al siguiente, permitiendo recorridos eficientes.

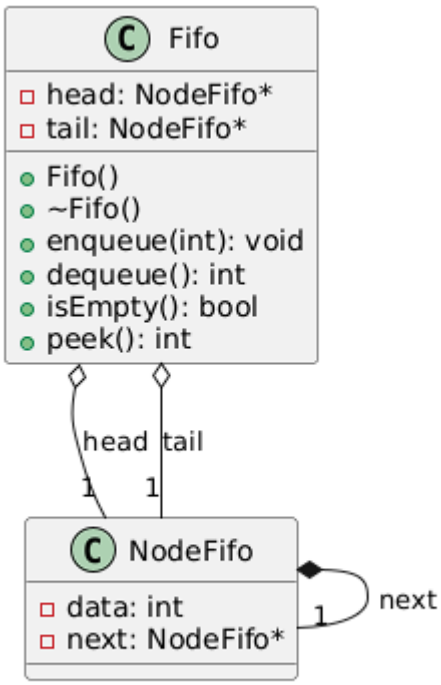
5.2. Pilas

- Gestionan los poderes de cada jugador.
- El jugador obtiene y consume poderes siguiendo el orden LIFO (último en entrar, primero en salir).

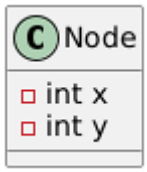


5.3. Colas

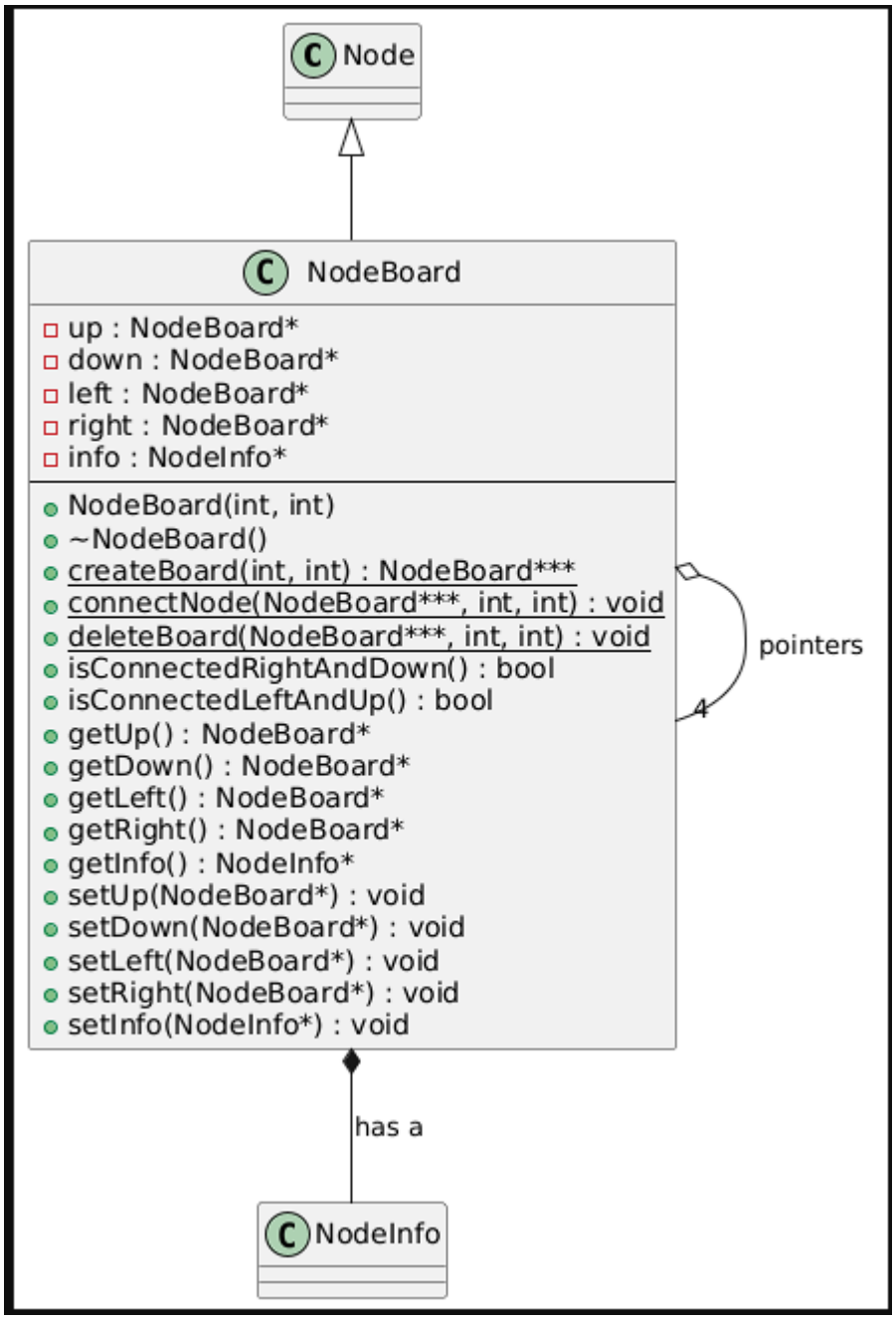
- Controlan el orden de turno de los jugadores.
- Se utiliza una cola tipo FIFO (primero en entrar, primero en salir) para asegurar el flujo del juego.



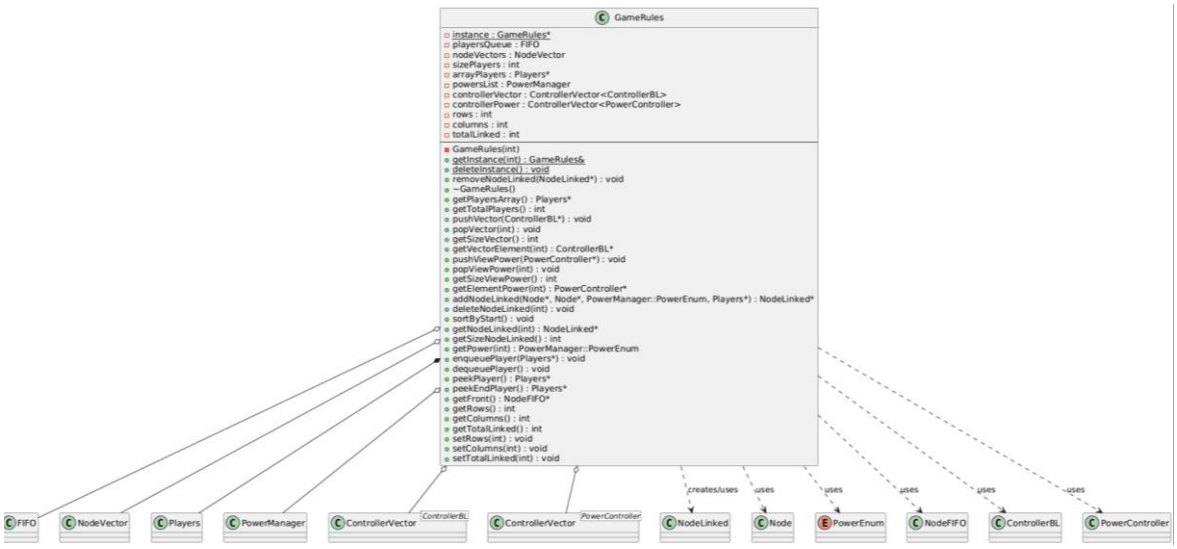
Nodo (Node)



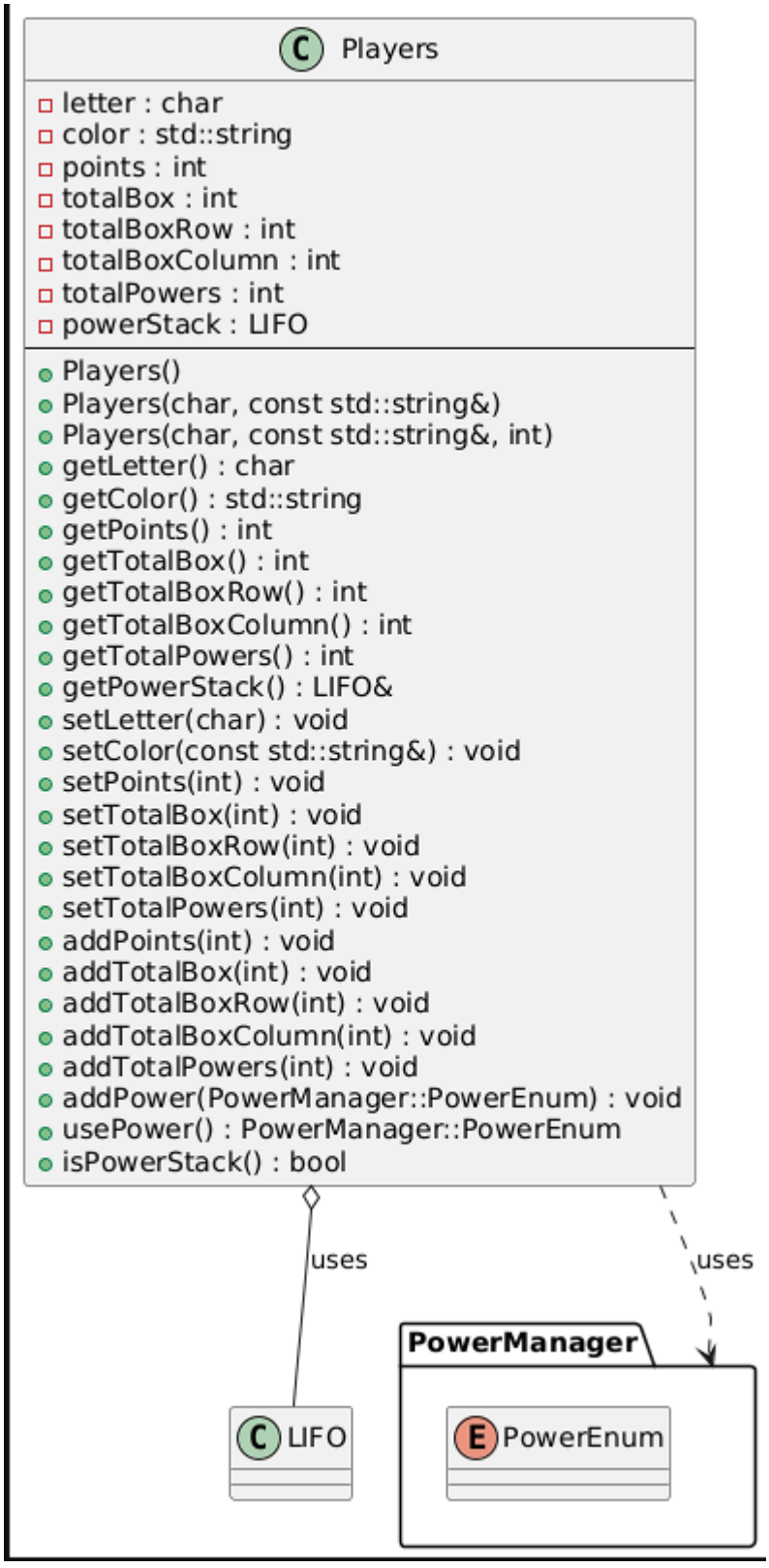
Tablero (NodeBoard)



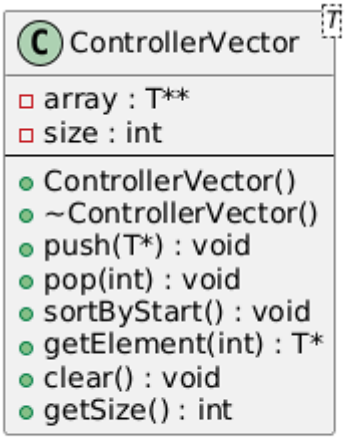
Reglas de Juego (Game Rules)



Jugadores (Players)

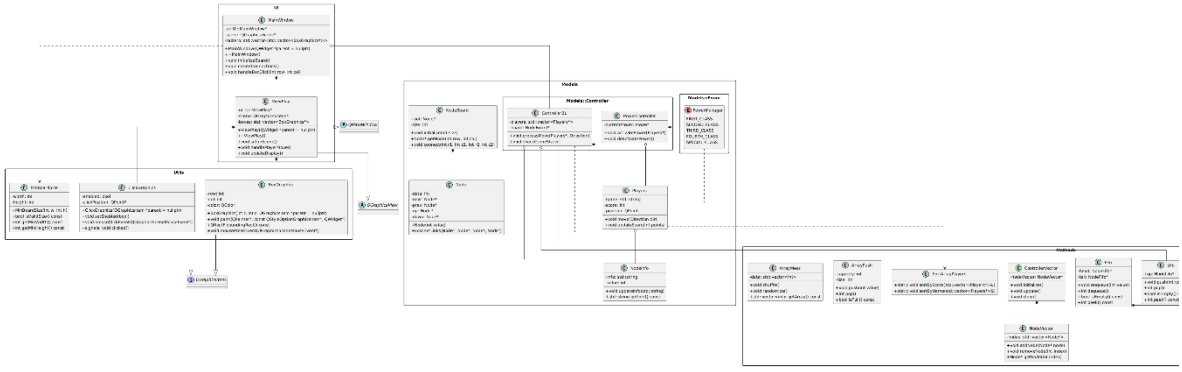

Players
- letter : char
- color : std::string
- points : int
- totalBox : int
- totalBoxRow : int
- totalBoxColumn : int
- totalPowers : int
- powerStack : LIFO
+ Players()
+ Players(char, const std::string&)
+ Players(char, const std::string&, int)
+ getLetter() : char
+ getColor() : std::string
+ getPoints() : int
+ getTotalBox() : int
+ getTotalBoxRow() : int
+ getTotalBoxColumn() : int
+ getTotalPowers() : int
+ getPowerStack() : LIFO&
+ setLetter(char) : void
+ setColor(const std::string&) : void
+ setPoints(int) : void
+ setTotalBox(int) : void
+ setTotalBoxRow(int) : void
+ setTotalBoxColumn(int) : void
+ setTotalPowers(int) : void
+ addPoints(int) : void
+ addTotalBox(int) : void
+ addTotalBoxRow(int) : void
+ addTotalBoxColumn(int) : void
+ addTotalPowers(int) : void
+ addPower(PowerManager::PowerEnum) : void
+ usePower() : PowerManager::PowerEnum
+ isPowerStack() : bool
uses
uses
PowerManager
LIFO
PowerEnum

Vector

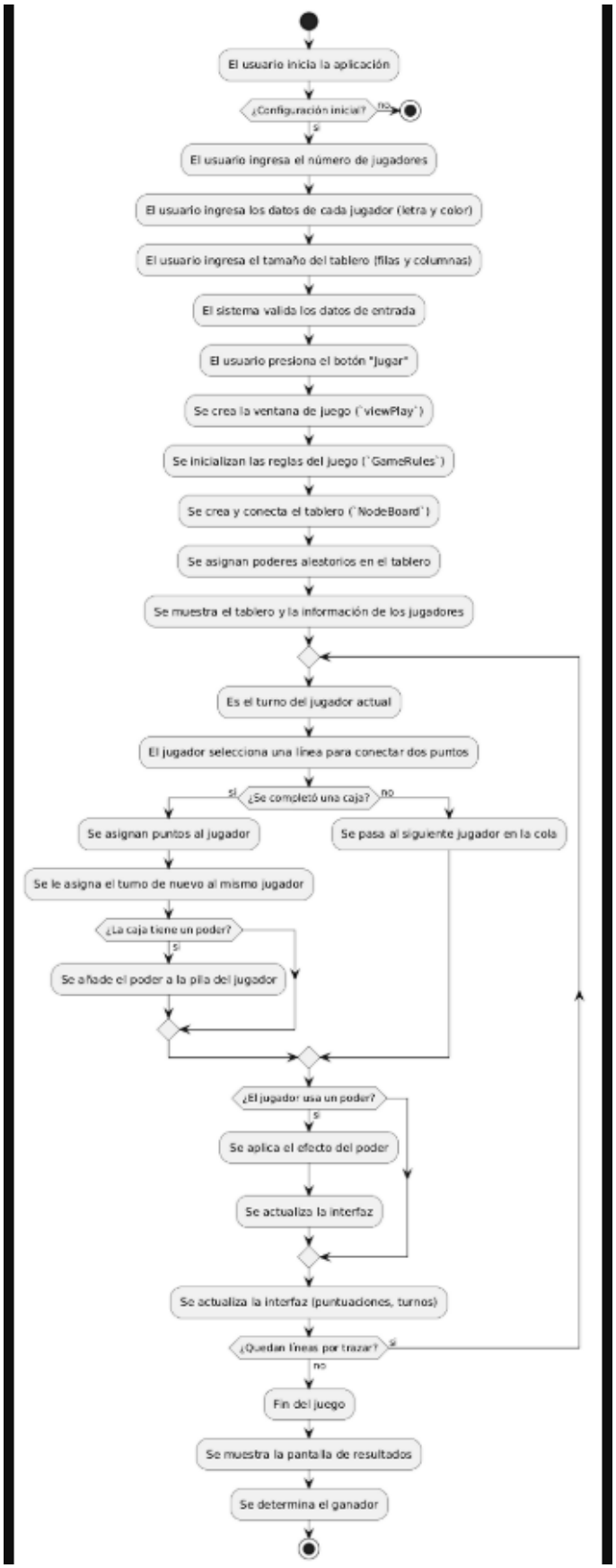


Diagramas

Clase



Flujo


El usuario inicia la aplicación
¿Configuración inicial?
no
si
El usuario ingresa el número de jugadores
El usuario ingresa los datos de cada jugador (letra y color)
El usuario ingresa el tamaño del tablero (filas y columnas)
El sistema valida los datos de entrada
El usuario presiona el botón "Jugar"
Se crea la ventana de juego (`viewPlay`)
Se inicializan las reglas del juego (`GameRules`)
Se crea y conecta el tablero (`NodeBoard`)
Se asignan poderes alestorios en el tablero
Se muestra el tablero y la información de los jugadores
Es el turno del jugador actual
El jugador selecciona una línea para conectar dos puntos
¿Se completó una caja?
si
no
Se asignan puntos al jugador
Se pasa al siguiente jugador en la cola
Se le asigna el turno de nuevo al mismo jugador
¿La caja tiene un poder?
si
Se añade el poder a la pila del jugador
¿El jugador usa un poder?
si
Se aplica el efecto del poder
Se actualiza la interfaz
Se actualiza la interfaz (puntuaciones, turnos)
¿Quedan líneas por trazar?
si
no
Fin del juego
Se muestra la pantalla de resultados
Se determina el ganador

11. Recomendaciones Técnicas

- Revisar siempre la liberación de memoria (delete) en vectores dinámicos.
- Utilizar QTableWidget para mostrar poderes con colores y estilos.
- Evitar librerías externas, el proyecto se basa en C++ nativo y Qt.
- Realizar pruebas unitarias manuales para validar estructuras de datos y turnos.

12. Contacto

Desarrollado por: Erikson Orozco (EDK367)