Servicios Críticos:

Ingreso de Jugadores:

```
/// complegidad | O(1)      You, hace 1 segundo • Uncommitted changes
void encolar(Persona *persona)
{
    NodoTurno *nuevo = new NodoTurno(persona);
    if (finalCola)
    {
        finalCola->siguiente = nuevo;
    }
    finalCola = nuevo;
    if (!frente)
    {
        frente = nuevo;
    }
}
```

Ingreso de Fichas

```
/// Metodo para ingresar una ficha
/// complegidad algoritmica O(n)
void insertar(Ficha *ficha)
{

    NodoListFicha *nuevo = new NodoListFicha(ficha);   /// O(1)

    if (this->primero == NULL)    /// O(1)
    {
        this->primero = nuevo;
        this->ultimo = nuevo;
    }
    else   /// O(1)      You, hace 1 segundo • Uncommitted changes
    {
        this->ultimo->siguiente = nuevo;
        this->ultimo = nuevo;
    }
    this->size++;
};
```

Gestión de turnos con cola:

```
/// Complejidad O(1)
void moverFrenteFinal()
{
    if (esVacia())  /// O(1)
    {
        cout << "La cola está vacía" << endl;
        return;
    }

    if (frente == finalCola)  /// O(1)
    {
        cout << "Solo hay un elemento, no se mueve" << endl;
        return;
    }

    NodoTurno *temp = frente; // Guardamos el nodo a mover

    frente = frente->siguiente; // Avanzamos el frente
    temp->siguiente = nullptr;  // Quitamos la referencia al siguiente nodo

    finalCola->siguiente = temp; // Conectamos el último nodo con el nuevo nodo final
    finalCola = temp;            // Actualizamos el final de la cola
}
  You, hace 3 días • iniciando repo
```

Inserción de Fichas en una Lista Enlazada:

```
    /// Metodo para eliminar una ficha
    /// complegidad algoritmica O(n)      You, hace 7 segundos •
    void eliminar(Ficha *ficha)
    {
        NodoListFicha *actual = this->primero;
        NodoListFicha *anterior = NULL;

        while (actual != NULL)  /// O(n)
        {
            if (actual->ficha == ficha)  /// O(1)
            {
                if (anterior == NULL)   /// O(1)
                {
                    this->primero = actual->siguiente;
                }
                else                    /// O(1)
                {
                    anterior->siguiente = actual->siguiente;
                }
                delete actual;
                this->size--;
                return;
            }
            anterior = actual;
            actual = actual->siguiente;
        }
    };
```

Ordenamiento de Ficha alfabéticamente:

```
/// bubble Sort
/// complegidad algoritmica O(n²)
// es simple y rapido simpre y cuando las lista no son demasiado Grandes

void ordenarListAlfabe()
{
    if (this->primero == nullptr || this->primero->siguiente == nullptr)
    {
        return; // Lista vacía o con un solo elemento, ya está ordenada
    }       You, ayer • ordenando ficha
    bool cambiado;
    do /// O(n)
    {
        cambiado = false;
        NodoListFicha *actual = this->primero;
        NodoListFicha *siguiente = actual->siguiente;

        while (siguiente != nullptr) /// O(n)
        {
            if (actual->ficha->getLetra() > siguiente->ficha->getLetra())
            {
                // Intercambiar fichas
                Ficha *temp = actual->ficha;
                actual->ficha = siguiente->ficha;
                siguiente->ficha = temp;

                cambiado = true;
            }
            actual = siguiente;
            siguiente = siguiente->siguiente;
        }
    } while (cambiado);
};
```