Práctica 2: Sistemas Distribuidos - Gentrificacion en la República Bananera 🍌

GitHub Repository

🏷 Tabla de Contenidos

🌟 Introducción

En esta práctica se implementa un sistema concurrente en **Go** para modelar la **gestión de tráfico aéreo** en un **aeropuerto**. El programa simula el **aterrizaje**, la **asignación de puertas de desembarque** y el **desembarque de pasajeros**, categorizando los aviones según su **capacidad** y **prioridad**. Para garantizar un control eficiente, se emplean mecanismos concurrentes como **goroutines** y **canales**, los cuales permiten una interacción fluida entre los diversos componentes del sistema.

El enfoque práctico de esta implementación busca fortalecer las habilidades en el manejo de **concurrencia en Go**, destacando el uso eficiente de recursos y la correcta sincronización en escenarios complejos. Además, se incluyen **pruebas automatizadas** para analizar el rendimiento bajo distintas configuraciones.

🎯 Objetivo del Programa

El programa tiene como propósito:

- **Modelar** el tráfico aéreo de un aeropuerto simulando **aterrizajes**, **asignación de recursos** y **desembarques**.
- **Clasificar** los aviones en tres categorías según el número de pasajeros y priorizarlos:
  - **Categoría A**: Más de 100 pasajeros (**prioridad alta**).
  - **Categoría B**: Entre 50 y 100 pasajeros (**prioridad media**).
  - **Categoría C**: Menos de 50 pasajeros (**prioridad baja**).
- Garantizar el **uso eficiente** de pistas y puertas mediante **concurrencia**.
- Evaluar el sistema con **pruebas automatizadas** para diferentes combinaciones de tráfico aéreo.

🛠 Descripción Técnica

El sistema consta de los siguientes componentes principales:

Torre de Control:

- **Coordina** los aterrizajes y asigna las pistas a los aviones.
- **Gestiona** un límite máximo de aviones en espera.

Pistas de Aterrizaje:

- **Controla** los aterrizajes simultáneos según la cantidad disponible.
- **Simula** tiempos variables de uso.

Puertas de Desembarque:

- **Asigna** aviones para desembarque después de aterrizar.
- **Simula** tiempos variables de desembarque de pasajeros.

Aviones:

- Cada avión tiene un **identificador**, **número de pasajeros** y una **categoría** asignada en base a su capacidad.

Concurrencia:

- **Goroutines**: Manejan los procesos concurrentes de aterrizaje y desembarque.
- **Canales**: Facilitan la comunicación y sincronización entre componentes.
- **Sincronización**: El uso de canales garantiza que cada avión complete su ciclo antes de finalizar la simulación.

Algoritmo General:

- Se crean **N aviones** con atributos aleatorios de pasajeros.
- Los aviones se **clasifican y priorizan** según su categoría.
- La **torre de control** gestiona los aterrizajes asignando pistas disponibles.
- Tras aterrizar, los aviones se mueven a las **puertas de desembarque**, que son gestionadas de manera concurrente.
- El programa finaliza cuando **todos los aviones completan su ciclo**.

📊 Diagramas de Flujo

Diagrama de Secuencia

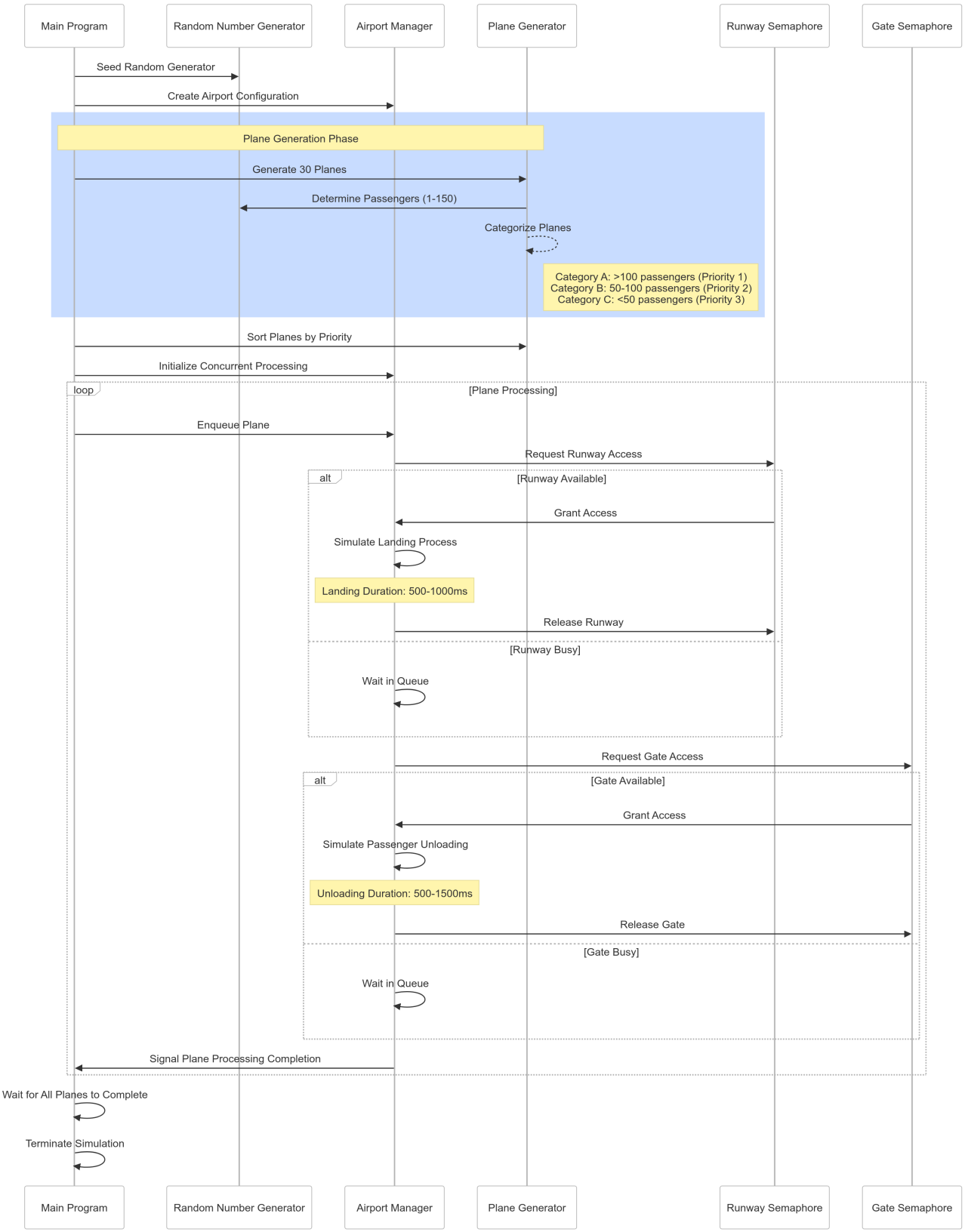
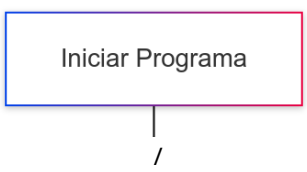
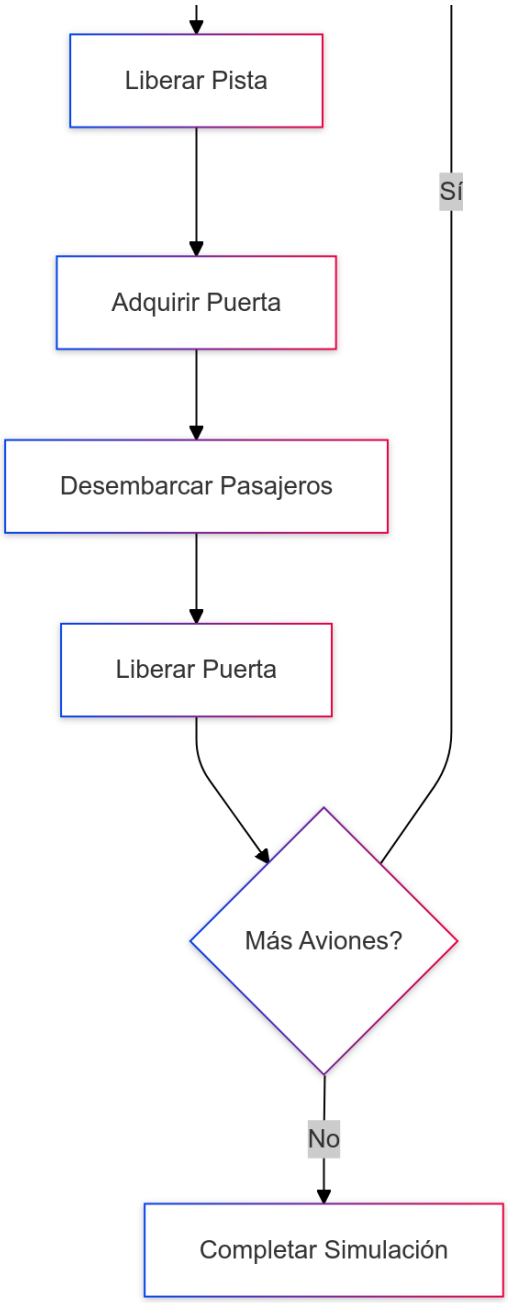


Diagrama de flujo




Generar 30 Aviones
Categorizar Aviones
Pasajeros > 100
Pasajeros 50-100
Pasajeros < 50
Categoria A, Prioridad 1
Categoria B, Prioridad 2
Categoria C, Prioridad 3
Ordenar Aviones por Prioridad
Iniciar Procesamiento de Aeropuerto
Cola de Aviones
Avión en Cola
Adquirir Pista
Simular Aterrizaje

/



> Nota: Los diagramas han sido creados utilizando Mermaid.

🧪 Resultados de las Pruebas

Distribución Equitativa:

- **Categorías**: 10 aviones de cada tipo (A, B, C).
- **Resultados**: El sistema procesó los 30 aviones en un tiempo promedio de 5 segundos.

Más Aviones de Alta Prioridad:

- **Categorías**: 20 A, 5 B, 5 C.
- **Resultados**: Los aviones de categoría A completaron el proceso significativamente antes que las demás categorías.

Más Aviones de Baja Prioridad:

- **Categorías**: 5 A, 5 B, 20 C.
- **Resultados**: Los aviones de baja prioridad tuvieron tiempos de espera más altos debido a la asignación prioritaria.

Métricas Clave

Tipo de Configuración	Tiempo Total	Aviones Procesados Correctamente
Distribución Equitativa (10 A, 10 B, 10 C)	8.27s	30
Más Aviones de Alta Prioridad (20 A, 5 B, 5 C)	6.77s	30
Más Aviones de Baja Prioridad (5 A, 5 B, 20 C)	8.43s	30

Análisis Detallado

- **Eficiencia del Sistema**: En todos los casos, el sistema fue capaz de procesar correctamente todos los aviones, demostrando su robustez y capacidad para manejar diferentes configuraciones de tráfico aéreo.
- **Tiempos de Espera**: Los tiempos de espera variaron significativamente según la configuración. La configuración con más aviones de alta prioridad fue la más eficiente, seguida por la distribución equitativa y finalmente la configuración con más aviones de baja prioridad.
- **Uso de Recursos**: La asignación de recursos (pistas y puertas) fue más eficiente en la configuración con más aviones de alta prioridad, lo que sugiere que priorizar aviones con más pasajeros puede mejorar el rendimiento general del sistema.
- **Impacto de la Prioridad**: La prioridad asignada a los aviones tuvo un impacto directo en los tiempos de espera y el uso de recursos. Los aviones de alta prioridad fueron procesados más rápidamente, mientras que los de baja prioridad experimentaron mayores tiempos de espera.

Explicación de los Resultados

- **Más Aviones de Alta Prioridad**: Esta configuración resultó ser la más eficiente porque los aviones de alta prioridad (Categoría A) tienen más pasajeros y, por lo tanto, se les da preferencia en el uso de recursos como pistas y puertas. Esto reduce los tiempos de espera para estos aviones y permite un procesamiento más rápido y eficiente. Además, al procesar primero los aviones con más pasajeros, se maximiza el uso de los recursos disponibles, ya que estos aviones ocupan las pistas y puertas por menos tiempo en comparación con una configuración con más aviones de baja prioridad.

- **Distribución Equitativa**: La distribución equitativa de aviones (10 de cada tipo) proporciona un balance entre las diferentes categorías. Aunque no es tan eficiente como la configuración con más aviones de alta prioridad, esta configuración permite un uso balanceado de los recursos y evita que una categoría específica monopolice las pistas y puertas. Esto resulta en tiempos de espera moderados para todas las categorías.

- **Más Aviones de Baja Prioridad**: Esta configuración resultó ser la menos eficiente debido a que los aviones de baja prioridad (Categoría C) tienen menos pasajeros y, por lo tanto, se les da menor prioridad en el uso de recursos. Esto significa que los aviones de baja prioridad deben esperar más tiempo para acceder a las pistas y puertas, lo que aumenta los tiempos de espera y reduce la eficiencia general del sistema. Además, como estos aviones ocupan los recursos por más tiempo, se crea un cuello de botella que afecta negativamente el rendimiento del sistema.

En resumen, priorizar aviones con más pasajeros (alta prioridad) mejora el rendimiento general del sistema al reducir los tiempos de espera y maximizar el uso de los recursos disponibles. Por otro lado, una mayor cantidad de aviones de baja prioridad puede crear cuellos de botella y aumentar los tiempos de espera, lo que reduce la eficiencia del sistema.

📜 Conclusiones

Manejo Eficiente de Concurrencia:

- El uso de **goroutines** y **canales** permitió una simulación fluida y eficiente, demostrando la capacidad de **Go** para manejar tareas concurrentes de manera efectiva.

Priorización Funcional:

- Los aviones de **alta prioridad** completaron sus procesos antes, cumpliendo con las reglas definidas y asegurando un uso óptimo de los recursos disponibles.

Escalabilidad:

- El programa demostró ser **escalable** al manejar diversas configuraciones de tráfico aéreo, adaptándose a diferentes escenarios sin pérdida significativa de rendimiento.

Áreas de Mejora:

- Incorporar **métricas más detalladas**, como tiempos de espera por categoría, podría proporcionar una visión más completa del rendimiento del sistema y ayudar a identificar posibles cuellos de botella.

🚀 Ejemplos de Uso

Ejemplo Básico

```
adrian@adrian-System-Product-Name:~/Escritorio/SistemasDistribuidos/P3_GO$
go run main.go
Plane 16 (Category A) is landing...
Plane 20 (Category C) is landing...
Plane 28 (Category A) is landing...
Plane 20 (Category C) is unloading passengers...
Plane 6 (Category A) is landing...
Plane 16 (Category A) is unloading passengers...
Plane 27 (Category A) is landing...
Plane 28 (Category A) is unloading passengers...
Plane 21 (Category A) is landing...
Plane 27 (Category A) is unloading passengers...
Plane 23 (Category A) is landing...
Plane 20 disembarked.
Plane 28 disembarked.
Plane 21 (Category A) is unloading passengers...
Plane 10 (Category A) is landing...
Plane 6 (Category A) is unloading passengers...
```

```
Plane 12 (Category A) is landing...
Plane 16 disembarked.
Plane 23 (Category A) is unloading passengers...
Plane 11 (Category A) is landing...
Plane 10 (Category A) is unloading passengers...
Plane 13 (Category A) is landing...
Plane 21 disembarked.
Plane 27 disembarked.
Plane 12 (Category A) is unloading passengers...
Plane 30 (Category A) is landing...
Plane 23 disembarked.
Plane 4 (Category A) is landing...
Plane 11 (Category A) is unloading passengers...
Plane 15 (Category B) is landing...
Plane 13 (Category A) is unloading passengers...
Plane 6 disembarked.
Plane 30 (Category A) is unloading passengers...
Plane 26 (Category B) is landing...
Plane 10 disembarked.
Plane 4 (Category A) is unloading passengers...
Plane 14 (Category B) is landing...
Plane 11 disembarked.
Plane 15 (Category B) is unloading passengers...
Plane 24 (Category C) is landing...
Plane 9 (Category B) is landing...
Plane 13 disembarked.
Plane 26 (Category B) is unloading passengers...
Plane 30 disembarked.
Plane 12 disembarked.
Plane 14 (Category B) is unloading passengers...
Plane 8 (Category C) is landing...
Plane 24 (Category C) is unloading passengers...
Plane 22 (Category C) is landing...
Plane 4 disembarked.
Plane 26 disembarked.
Plane 3 (Category C) is landing...
Plane 9 (Category B) is unloading passengers...
Plane 22 (Category C) is unloading passengers...
Plane 7 (Category C) is landing...
Plane 19 (Category C) is landing...
Plane 15 disembarked.
Plane 8 (Category C) is unloading passengers...
Plane 24 disembarked.
Plane 3 (Category C) is unloading passengers...
Plane 18 (Category C) is landing...
Plane 14 disembarked.
Plane 9 disembarked.
Plane 7 (Category C) is unloading passengers...
Plane 29 (Category C) is landing...
Plane 19 (Category C) is unloading passengers...
Plane 5 (Category C) is landing...
Plane 1 (Category B) is landing...
Plane 22 disembarked.
Plane 18 (Category C) is unloading passengers...
```

```
Plane 3 disembarked.
Plane 29 (Category C) is unloading passengers...
Plane 2 (Category B) is landing...
Plane 8 disembarked.
Plane 7 disembarked.
Plane 5 (Category C) is unloading passengers...
Plane 17 (Category B) is landing...
Plane 2 (Category B) is unloading passengers...
Plane 25 (Category B) is landing...
Plane 18 disembarked.
Plane 1 (Category B) is unloading passengers...
Plane 19 disembarked.
Plane 29 disembarked.
Plane 25 (Category B) is unloading passengers...
Plane 17 (Category B) is unloading passengers...
Plane 5 disembarked.
Plane 1 disembarked.
Plane 25 disembarked.
Plane 2 disembarked.
Plane 17 disembarked.
Simulation completed.
```

📂 Código Fuente

El código completo del programa y las pruebas están disponibles en los archivos `main.go` y `main_test.go` dentro del directorio `src`. Puedes encontrar el código fuente y las pruebas automatizadas en el repositorio de GitHub.

Archivos Principales:

- **`main.go`**: Contiene la implementación principal del programa, incluyendo la creación de aviones, la configuración del aeropuerto y la lógica de concurrencia.
- **`main_test.go`**: Incluye las pruebas automatizadas para evaluar el rendimiento del sistema bajo diferentes configuraciones de tráfico aéreo.

Enlace al Repositorio:

Puedes acceder al código fuente completo y a las pruebas en el siguiente enlace: GitHub Repository.

Para ejecutar el programa y las pruebas, sigue estos pasos:

1. Clona el repositorio:

```
git clone https://github.com/aMonteSl/P3_GO.git
cd P3_GO/src
```
