Primer Obligatorio Algoritmos y estructura de datos

Francisco Arenas

332800

Pedro Wattimo

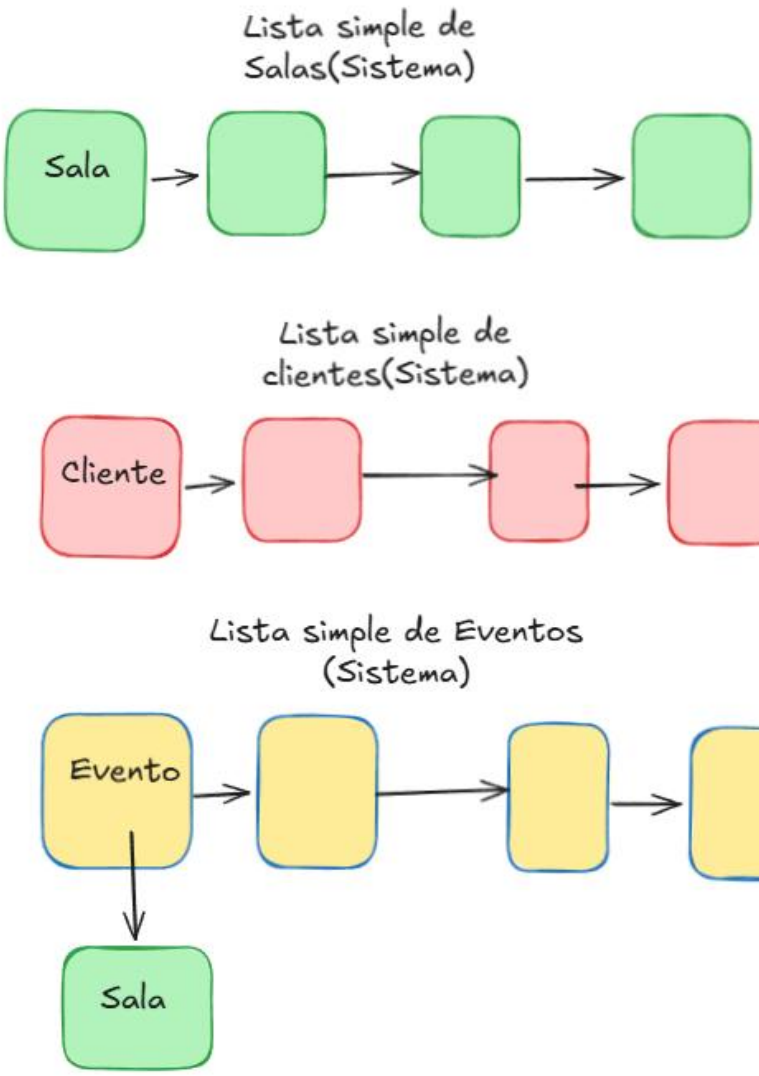
338849

Docente: Rafael Cohen

Fecha de entrega: 12/05/2025

# Indice

## Representación Gráfica



# Entidades creadas:

-Cliente(String Nombre, String Cedula)

-Evento(String codigo, String Descripcion, int aforoNecesario,LocalDate fecha,Sala sala)

-Sala(String Sala, int Capacidadd, List<LocalDate> fechasOcupadas)

# Requerimientos

## 1.1 crearSistemaDeGestion()

Para inicializar el sistema, se requiere una estructura base que permita almacenar y administrar dinámicamente conjuntos de datos como salas, eventos y clientes. Las listas simples de nodos fueron elegidas por su flexibilidad para crecer dinámicamente y por la facilidad para implementar desde cero TADs personalizados, sin depender de estructuras del lenguaje.

## 1.2 registrarSala(nombre, capacidad)

Las salas se almacenan en una lista simple de nodos, ya que:

- El orden de inserción es relevante para listarlas en orden inverso. Al utilizar un método que agregue al inicio, al listarlo ya estará en orden inverso.
- Permite verificar duplicados (nombre ya existente) recorriendo la lista.

## 1.3 eliminarSala(nombre)

La eliminación de una sala se simplifica utilizando una lista simple al:

- Recorrer los nodos secuencialmente.
- Eliminar el nodo con coincidencia de nombre, manteniendo la integridad del resto de la estructura.
  Además, la implementación resulta más clara y didáctica en el contexto del curso, reforzando el uso de punteros y referencias.

## 1.4 registrarEvento(codigo, descripcion, aforoNecesario, fecha)

Los eventos también se almacenan en una lista simple, ya que:

- Se requiere buscar salas disponibles para la fecha (se recorren ambas listas secuencialmente).

- Permite validar códigos duplicados y asociar fácilmente eventos con salas ya registradas.
- La flexibilidad de las listas simples se ajusta a la naturaleza dinámica de los eventos y facilita su manipulación posterior (como eliminarlos o listar por orden alfabético).

## 1.5 registrarCliente(cedula, nombre)

El almacenamiento de clientes se resuelve con listas simples, ya que:

- La búsqueda por cédula es directa recorriendo la lista.
- Permite validar formato y duplicación antes de agregar un nuevo nodo.

2.1 Listar Salas

La implementación de la función listarSalas utilizando una lista simple de nodos es adecuada por varias razones:

- Orden inverso al registro: Al agregar las salas al inicio de la lista, se preserva el orden de inserción y, al listarlas, se obtienen en orden inverso sin necesidad de reordenar.
- Facilidad para recorrer la lista: Utilizando una lista simple, el recorrido para listar las salas es directo y eficiente, mostrando el nombre y la capacidad de cada sala de forma clara.
- Simplicidad y eficiencia: La solución es simple y eficiente, ya que no requiere estructuras adicionales ni algoritmos complejos para mantener el orden inverso.

## 2.2 Listar Eventos

La implementación de la función listarEventos es adecuada por varias razones:

- Orden alfabético: Ordenar los eventos por el código alfabéticamente permite que la lista sea fácilmente legible y de fácil acceso para el usuario.
- Eficiencia: El uso de un algoritmo de ordenamiento eficiente, como el ordenamiento alfabético por el código del evento, hace que la lista se genere rápidamente.

## 2.3 Listar Clientes

La implementación de la función listarClientes es adecuada por varias razones:

- Orden por cédula de identidad: Ordenar los clientes por cédula permite que la lista esté organizada de forma lógica y fácil de consultar.

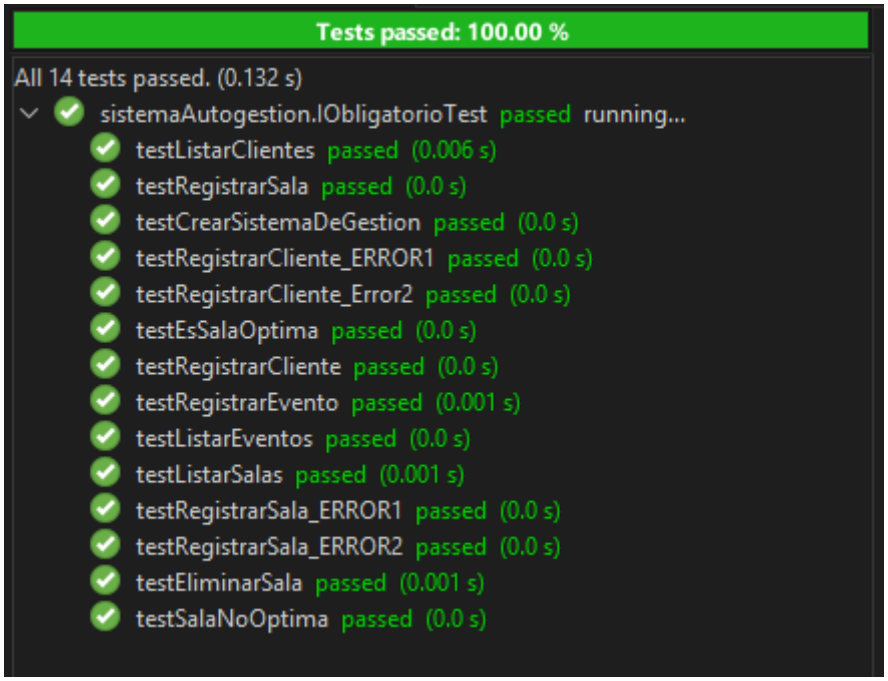
- Simplicidad: Utilizar un algoritmo de ordenamiento sencillo, como el de ordenación por números, garantiza que la lista se mantenga ordenada sin complicaciones.
- Eficiencia: La función es eficiente y no requiere operaciones complejas para listar los clientes, lo que la hace adecuada incluso para listas grandes de clientes.
- Escalabilidad: Este enfoque permite fácilmente agregar nuevos clientes a la lista y ordenar la información según diferentes criterios si fuera necesario en el futuro.

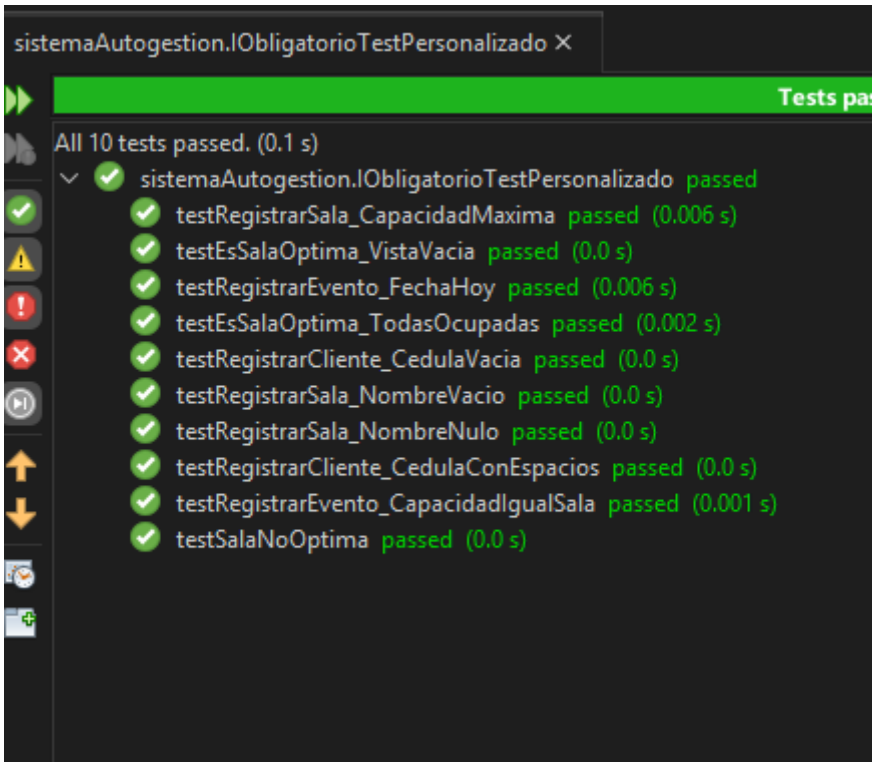
### 2.4 esSalaOptima(vistaSala[][])

Creemos que la implementación de la función es adecuada por varias razones:

- Cumple con los requisitos: Recorre cada columna para verificar si los asientos ocupados consecutivos superan a los libres, como se especifica en el enunciado.
- Eficiencia: El recorrido de las columnas de forma independiente simplifica la lógica y mejora la eficiencia.
- Escalabilidad: El enfoque es escalable, permitiendo su aplicación a salas de diferentes tamaños sin cambios importantes en el código.

## Tabla Final de Resultados






# Documentación de pruebas.

## testCrearSistemaDeGestion

- **Propósito:** Verifica que el sistema se pueda crear correctamente.
- **Resultado esperado:** Retorno.Resultado.OK.

## testRegistrarSala

- **Propósito:** Registrar dos salas válidas.
- **Resultado esperado:** Ambas con Retorno.Resultado.OK.

## testRegistrarSala_ERROR1

- **Propósito:** Detectar error por nombre duplicado de sala.
- **Condición:** Se intenta registrar "Sala A" dos veces.
- **Resultado esperado:** Último registro con ERROR_1.

## testRegistrarSala_ERROR2

- **Propósito:** Validar errores por capacidad inválida.
- **Condición:** Capacidad 0 y negativa.

- **Resultado esperado:** ERROR_2 en ambos casos.

### ◇ testEliminarSala

- **Propósito:** Probar eliminación exitosa de una sala.
- **Resultado esperado:** Retorno.Resultado.OK.

### ◇ testRegistrarEvento

- **Propósito:** Registrar un evento válido.
- **Condición:** Sala con suficiente capacidad ya registrada.
- **Resultado esperado:** Retorno.Resultado.OK.

### ◇ testRegistrarCliente

- **Propósito:** Registrar dos clientes válidos.
- **Resultado esperado:** Ambos con Retorno.Resultado.OK.

### ◇ testRegistrarCliente_ERROR1

- **Propósito:** Validar formato incorrecto de cédula.
- **Condición:** Cédula con menos de 8 dígitos y caracteres no numéricos.
- **Resultado esperado:** Ambos con ERROR_1.

### ◇ testRegistrarCliente_Error2

- **Propósito:** Detectar duplicación de cédula.
- **Resultado esperado:** Segundo intento con ERROR_2.

### ◇ testListarSalas

- **Propósito:** Listar salas ordenadas por capacidad descendente.
- **Resultado esperado:** "Sala C-100#Sala B-70#Sala A-50".

### ◇ testListarEventos

- **Propósito:** Verificar listado de eventos con formato correcto.
- **Resultado esperado:** Eventos ordenados y bien formateados.

## ◇ testListarClientes

- **Propósito:** Listar clientes en orden alfabético por nombre.
- **Resultado esperado:** "23331111-Martina Rodríguez#35679992-Ramiro Perez#45678992-Micaela Ferrez".

## ◇ testEsSalaOptima

- **Propósito:** Evaluar una vista de sala con asientos para determinar si es óptima.
- **Resultado esperado:** Se espera una evaluación que devuelve un Retorno con resultado determinado (el final del método no se ve completamente en el archivo).

## ◇ testRegistrarSala_NombreVacio

- **Propósito:** Validar que no se permita registrar una sala con nombre vacío.
- **Entrada:** ("", 50)
- **Resultado esperado:** ERROR_1.

## ◇ testRegistrarSala_NombreNulo

- **Propósito:** Verificar que un nombre nulo no sea aceptado.
- **Entrada:** (null, 50)
- **Resultado esperado:** ERROR_1.

## ◇ testRegistrarSala_CapacidadMaxima

- **Propósito:** Probar el sistema con capacidad máxima posible de sala.
- **Entrada:** ("Sala Grande", Integer.MAX_VALUE)
- **Resultado esperado:** OK.

## ◇ testRegistrarEvento_FechaHoy

- **Propósito:** Validar que se pueda registrar un evento con fecha igual a la actual.
- **Entrada:** ("E2", "Evento Hoy", 50, LocalDate.now())
- **Resultado esperado:** OK.

### ◇ testRegistrarEvento_CapacidadIgualSala

- **Propósito:** Verificar que se acepte un evento con la misma capacidad que la sala.
- **Entrada:** ("E3", "Justo", 80, fecha futura)
- **Resultado esperado:** OK.

### ◇ testRegistrarCliente_NombreVacio

- **Propósito:** Validar que no se pueda registrar cliente con nombre vacío.
- **Entrada:** ("87654321", "")
- **Resultado esperado:** ERROR_1.

### ◇ testRegistrarCliente_CedulaConEspacios

- **Propósito:** Evaluar si el sistema acepta cédulas con espacios.
- **Entrada:** (" 87654321 ", "Con espacios")
- **Resultado esperado:** ERROR_1.

### ◇ testEsSalaOptima_TodasOcupadas

- **Propósito:** Evaluar si una sala totalmente ocupada se considera óptima.
- **Entrada:** Matriz 5x5 llena de "X".
- **Resultado esperado:** "No es óptimo".

### ◇ testEsSalaOptima_VistaVacia

- **Propósito:** Probar comportamiento con una matriz vacía.
- **Entrada:** {}
- **Resultado esperado:** ERROR_1.