## Requerimientos funcionales

1. Permitir cargar la información de los pasajeros correspondiente a un vuelo mediante un archivo de texto plano.
2. Buscar la información completa de un pasajero de manera eficiente una vez que llega a la sala de abordaje.
3. Registrar la llegada a la sala del pasajero.
4. El sistema debe permitir premiar la puntualidad de los pasajeros ingresando al avión en el orden de llegada.
5. El sistema debe permitir establecer un orden de salida del avión teniendo en cuenta la configuración de los asientos, donde los que se encuentran en las primeras filas son los primeros en salir y para cada fila el orden está establecido por cercanía al pasillo u orden de llegada como última instancia.
6. Priorizar el ingreso de la primera clase en base a otros datos relevantes, como millas acumuladas, atención especial requerida, tercera edad, etc.
7. Mostrar entrada

## Definición del problema

Se identifican las siguientes necesidades y síntomas:

* Los procesos de ingreso y salida del avión de la aerolínea actualmente son caóticos y no premian la puntualidad de los pasajeros.
* El proceso de ingreso y salida del avión no prioriza de manera adecuada a los pasajeros de primera clase.
* No hay un sistema que permita la carga de información de pasajeros y su búsqueda eficiente en la sala de abordaje.
* No hay un sistema que registre la llegada de los pasajeros a la sala de abordaje.
* No hay un sistema que permita mostrar en qué orden deben ingresar los pasajeros al avión teniendo en cuenta el orden de llegada y las secciones del avión.
* No hay un sistema que permita priorizar el ingreso de la primera clase en base a otros datos relevantes.
* No hay un sistema que permita establecer el orden de salida de los pasajeros del avión en base a la configuración de las filas del avión.
* La aerolínea necesita mejorar la experiencia del usuario y su relación con los clientes

Definición del problema

La aerolínea requiere de un sistema que permita mejorar el proceso de ingreso y salida del avión.

## Recopilación de información necesaria

Vista de soluciones de problemas similares:

**-SabreSonic** es un sistema de gestión de pasajeros en la cual se basa en gestionar todo el proceso de reserva de vuelos, emisión de boletos, check-in y la gestión de la información del pasajero, dividiendo este en varios módulos que son:

* SaRreSonic Res: encarga de la reserva de vuelos y la emisión de boletos.
* SabreSonic Inventory: gestiona la capacidad de los vuelos y los asientos disponibles.
* SabreSonic Check-In: Gestiona el embarque de los pasajeros.
* SabreSonic Customer Sales and Service: Se encarga de la gestión de la información del pasajero.

Estos modulos funcionan como un sistema que trabajan en conjunto para obetener el objetivo deseado que es el control de todo lo relaciona en la gestion de los pasajeros.

-**Atlea** es un sistema que proporciona servicios para la gestión de pasajeros, que tiene diferentes módulos que, trabajando conjuntamente, permiten el funcionamiento del sistema. Estos modulos son:

Altéa Reservation: Maneja la gestión de los vuelos y boletos.

Atlea inventory: Gestiona la capacidad de los vuelos y los asientos.

Altéa Departure Control: Se encarga en el proceso del check-in y embarque.

Definiciones del problema:

* **Sistema de registro de pasajeros:** Es un conjunto de tecnologías y herramientas encargadas de recopilar información detallada de los pasajeros de una aerolínea o vuelo en específico.
* **Primera Clase**: Es una categoría más preferencial y exclusiva que se le ofrecen a los pasajeros. Esta categoría permite a los pasajeros tener mayores comodidades como mejores asientos o permitir la entrada al avión primero, a comparación de otras categorías.
* **Sistema de ingreso de pasajeros:** Es un conjunto de procesos que permite a los pasajeros tener acceso a los diferentes servicios que ofrece la aerolínea.
* **Sistema de salida de pasajeros**: es un conjunto de procedimientos que permite a las personas de la aerolínea y a los pasajeros que estos últimos tengas una salida del avión de manera ordenada al haber llegado al destino.
* **Embarque:** El embarque es el ingreso de pasajeros a un vehículo
* **Pasajeros preferenciales:** Son pasajeros que tienes ciertos privilegios y derechos extra que un pasajero común, ya que estos privilegios permites seguridad y proteger la dignidad de los pasajeros preferenciales. Estos pasajeros pueden ser personas de tercera edad, mujeres embarazadas o personas con enfermedades.

## BÚSQUEDA DE SOLUCIONES CREATIVAS

Nuestro objetivo es el mejor ordenamiento y eficiencia de para organización de los pasajeroos, para esto debemos observar los diferentes metodos que tenemos para organizar elementos y obtener la mejor eficiencia, estos pueden ser Stack, queues y colas de prioridad utilizando heap –Max, ahora mostraremos las diferentes soluciones que podemos usar:

1. Utilizar colas de prioridad para el ingreso y salida de los pasajeros, debido a que este permite ordenar elementos de una manera que tengan en cuenta la prioridad de estos mismos. Es decir, la cola de prioridad asegura que los elementos se procesan en orden de prioridad, de mayor a menor.
2. Utilzar Queue para el ingreso y salida de los pasajeros. Queue es una estructura de datos que se utiliza para almacenar y organizar una colección de elementos en la que el primer elemento en entrar es el primero en salir.
3. Utilizar Colas de prioridad y Queue para el ingreso y salida de los pasajeros. Dando prioridades a los pasajeros con las colas de prioridad y permitiendo con queue la mayor eficiencia y ordenamiento de los pasajeros.

## TRANSICIÓN DE LA FORMULACIÓN DE IDEAS A LOS DISEÑOS PRELIMINARES

Se descartará la segunda solución, debido a que solo usar queue supondría varios problemas al intentar dar la prioridad a los pasajeros que lo necesiten, debido a que este método prioriza la llegada, mas no otros factores.

Nos quedaríamos con la primera y tercera solución. La tercera solución, nos ayudaría mediante queue tener en cuenta el orden de llegada de los pasajeros de la clase económica, y de la primera clase nos ayudaría las colas de prioridad, teniendo en cuanta las millas acumuladas. Y para la salida, se podría hacer un arryalist , al ser una secuencia de filas, no es necesario hacer una estructura de datos , como queue o priority queue, siempre se va a ingresar de una manera determinada las filas , un ejemplo serai 1 ,2 3 ,4 5. Pero dentro de estas filas , va a ver un priority queue , que organizara de manera ordenada teniendo encuentra la distancia del pasillo y el tiempo de llegada

Para la primera solución, el uso de solo colas de prioridad permitiría hacer algo parecido a la tercera solución, solo que se debería hacer una cola de prioridad que tenga en cuenta la llegada de los pasajeros de los que son de clase económica.

## EVALUACIÓN Y SELECCIÓN DE LA MEJOR SOLUCIÓN

Los criterios para dar la mejor solucione son los siguientes:

Criterio 1: cargue incial de la base de datos.

[2] Aparece todos los pasajeros

[1] No aparece todos los pasajeros  
  
Criterio 2: registrar hora de llegada del pasajero.

[3] solución con eficiencia.

[2] solución ineficiente

[1] da solución

Criterio 3: Buscar la información completa del pasajero.

[3] solución con eficiencia.

[2] solución ineficiente

[1] da solución

Criterio 4: Orden de ingreso al avión.

[3] solución con eficiencia.

[2] solución ineficiente

[1] da solución

Criterio 5: Salida de ingreso al avión.

[3] solución con eficiencia.

[2] solución ineficiente

[1] da solución

Criterio 6: Muestra el orden de salida y de entrada a la tripulacion.

[2] muestra el orden ideal para el ingreso de pasajeros.

[1] muestra la lista de forma ineficiente.

Criterio 6: Facilidad de implementación y mantenimiento.

[3] La solución es fácil de implementar y mantener.

[2] La solución es medianamente fácil de implementar y mantener.

[1] La solución es difícil de implementar y mantener.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Alternativas/Criterios | C1 | C2 | C3 | C4 | C5 | C6 | C7 | Total |
| usar Colas de prioridad | 2 | 3 | 3 | 3 | 3 | 2 | 3 | 19 |
| usar queue y colas de prioridad | 2 | 3 | 2 | 2 | 3 | 1 | 2 | 15 |

Con esto nos damos cuenta de que la mejor solución es solo usar colas de prioridad para el ordenamiento de los pasajeros en los diferentes escenarios planteados.

## Preparación de Informes y Especificaciones

Especificaciones del problema

Problema: ingreso y salida de los pasajeros de un avión

Entradas: -un txt con la información de cada de pasajeros, que tendrá su nombre, apellido, si pertenece a primera clase, asiento, millas acomuladas,id.

* El numero de pasajeros que se van a ingresar al avión
* un txt con el orden de llegada de cada pasajero , con su hora correspondiente y su id.
* La configuración del avión , con el numero de asientos, numero de avión , numero de filas y columnas

Salida:- lista de ingreso de los pasajeros

* Lista de salida de los pasajeros

Consideraciones:

Los casos posibles que pueden ocurrir durante el ingreso del avión son:

-Si la lista de ingreso de los pasajeros esta vacia, se deberá ingresar otra vez una lista con por lo menos un pasajeros

-Se tomara en cuenta que ningún pasajero de la lista entregada se repetirá.

-Se tomara en cuenta que ningún pasajero de la lista entregada tendrá el mismo número de asiento y que entrera entre los números de asientos del avión

Los casos de que pueden ocurrir en durante la salida del avión son:

-Si la lista de hora se salida esta vacia , se debería ingresar otra vez la lista con por lo menos un pasajero

Algoritmo de organización de entrada del avión

Diagrama

Descripción generada automáticamente

Algoritmo de entrada del avión y entrada de txt

Diagrama

Descripción generada automáticamente

Ingreso de llegada de pasajeros

Diagrama

Descripción generada automáticamente

Ingreso de los pasajeros a la salida

Diagrama

Descripción generada automáticamente

## Implementación y diseño

Implementación hecha en Java

Lista de tareas a completar:

1. Ingreso del txt de pasajeros y añadirlo al hashmap
2. Ingreso del txt con las llegada de los pasajeros
3. Ingreso de los pasajeros de primera clase al priorityqueue
4. Ingreso de los pasajeros de clase económica al queue
5. Ingreso de los pasajeros sus respectivos priorityqueue de salida
6. Mostrar orden de salida
7. Mostrar orden de entrada

|  |  |
| --- | --- |
| Nombre: | LoadPassenger |
| Descripcion: | Toma los datos del txt y crea los pasajeros, para que luego , ingresarlos al hashmap |
| Entrada : | Txt:String , la dirección de los pasajeros  Count: int , cantidad de pasajeros |
| Retorno: | Void, al final del método ya se tendrá todos los pasajeros ingresados al hashmap |

public void LoadPassenger(String txt, int count) {

        File file = new File(txt);

        hashTable = new HashTable<>(count);

        try {

            FileInputStream fis = new FileInputStream(file);

            BufferedReader reader = new BufferedReader(new InputStreamReader(fis));

            String line;

            while ((line = reader.readLine()) != null) {

                String[] parts = line.split(" ");

                Passenger p = new Passenger(parts[0], parts[1], Boolean.parseBoolean(parts[2]),

                        Integer.parseInt(parts[3]),

                        Integer.parseInt(parts[4]), Integer.parseInt(parts[5]));

                int fila = (Integer.parseInt(parts[3]) - 1) / plane.getColumn() + 1;

                int columna = (Integer.parseInt(parts[3]) - 1) % plane.getColumn() + 1;

                p.setSeat(new Seat(columna, fila));

                int p1Dist = Math.min(p.getSeat().getColumn(), plane.getColumn() - p.getSeat().getColumn() + 1);

                p.setDis(p1Dist);

                hashTable.insert(parts[5], p);

            }

            fis.close();

        } catch (FileNotFoundException e) {

            e.printStackTrace();

        } catch (IOException e) {

            e.printStackTrace();

        }

    }

|  |  |
| --- | --- |
| Nombre: | ArriveP |
| Descripcion: | Toma los datos del txt y busca el pasajero en hashmap, para luego validar si son primera clase o no, e ingresarlo a su base de datos correspondiente , además que en este método se crea el la base de datos de la salida y se añade a los pasajeros a esta para organizarla |
| Entrada : | Txt:String , la dirección de los pasajeros |
| Retorno: | Void |

public void ArriveP(String txt) {

OutArray();

File file = new File(txt);

try {

FileInputStream fis = new FileInputStream(file);

BufferedReader reader = new BufferedReader(new InputStreamReader(fis));

String line;

while ((line = reader.readLine()) != null) {

String[] parts = line.split(" ");

Passenger p = hashTable.get(parts[0]);

if (p != null) {

p.toString();

LocalTime horaIngreso = LocalTime.parse(parts[1]);

p.setTime(horaIngreso);

if (p.getTime().isBefore(plane.getTime())) {

System.out.println(p.getName() + " se gano un descuento del 10% para su proximo vuelo");

}

if (p.isFirst\_class()) {

arrival\_queue\_firts\_class.insert(p);

out.get(p.row()-1).insert(p);

} else {

arrival\_queue\_economic\_class.enqueue(p);

out.get(p.row()-1).insert(p);

}

} else {

System.out.println("no se encontro");

}

}

fis.close();

} catch (FileNotFoundException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

}

|  |  |
| --- | --- |
| Nombre: | insert |
| Descripcion: | Añade los pasajeros al priority queue |
| Entrada : | P Passenger :pasajero a ingresar |
| Retorno: | Void |

public void insert(T item) {

heap.add(item);

int i = heap.size() - 1;

int parent = (i - 1) / 2;

while (i > 0 && heap.get(i).compareTo(heap.get(parent)) > 0) {

swap(i, parent);

i = parent;

parent = (i - 1) / 2;

}

}

|  |  |
| --- | --- |
| Nombre: | enqueue |
| Descripcion: | Añade los pasajeros al queue |
| Entrada : | P Passenger :pasajero a ingresar |
| Retorno: | Void |

public void enqueue(T data) {

queue.add(data);

}

|  |  |
| --- | --- |
| Nombre: | OutArray |
| Descripcion: | Crea los objetos filas , donde se ingresaran los pasajeros de salida |
| Entrada : | void |
| Retorno: | Void |

public void OutArray(){

for(int i=1;i<=plane.getRow();i++){

Row s= new Row(i);

out.add(s);

}

}

|  |  |
| --- | --- |
| Nombre: | insert |
| Descripcion: | Añade los pasajeros al priority queue de salida |
| Entrada : | P Passenger :pasajero a ingresar |
| Retorno: | Void |

public void insert(T item) {

heap.add(item);

int i = heap.size() - 1;

int parent = (i - 1) / 2;

while (i > 0 && heap.get(i).compare(heap.get(parent)) < 0) {

swap(i, parent);

i = parent;

parent = (i - 1) / 2;

}

}

|  |  |
| --- | --- |
| Nombre: | ShowOrderIngress |
| Descripcion: | Muestra el orden de ingreso |
| Entrada : | void |
| Retorno: | Void |

public void showOrderIngress() {

System.out.println("primera clase");

arrival\_queue\_firts\_class.print();

System.out.println("clase economica");

arrival\_queue\_economic\_class.print();

}

|  |  |
| --- | --- |
| Nombre: | print |
| Descripcion: | Imprime un queue |
| Entrada : | void |
| Retorno: | Void |

public void print() {

for (T data : queue) {

System.out.println(data);

}

System.out.println();

}

|  |  |
| --- | --- |
| Nombre: | print |
| Descripcion: | Imprime un priorityqueue |
| Entrada : | void |
| Retorno: | Void |

public void print() {

for (T item : heap) {

System.out.println(item + " ");

}

System.out.println();

}