

# UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS

## FACULTAD DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

### ESCUELA PROFESIONAL DE INGENIERÍA DE SOFTWARE



### Sistema de colas aplicado al llenado de balones de oxígeno

#### GRUPO 07

#### ANÁLISIS Y DISEÑO DE ALGORITMOS

ESPINOZA DOMINGUEZ, ROBERT

#### Integrantes:

- |  |            |
|--|------------|
| ■ Alberto Ramos, Harold Giusseppe      | (18200306) |
| ■ Alvarado Pardo, Vladimir Frank Felix | (18200070) |
| ■ Marin Evangelista, Jorge Luis        | (18200275) |
| ■ Quinteros Peralta, Rodrigo Ervin     | (18200316) |
| ■ Tirado Julca, Juan Jose              | (18200117) |

**Perú - 2021**

## Índice

<b>Objetivo del proyecto</b>	4
<b>Descripción del proyecto</b>	4
<b>Alcances y limitaciones del proyecto</b>	4
<b>Requerimientos funcionales del sistema</b>	5
Tabla general de requerimientos funcionales	5
Requisito Funcional 01	6
Requisito Funcional 02	6
Requisito Funcional 03	7
Requisito Funcional 04	8
Requisito Funcional 05	8
Requisito Funcional 06	9
Requisito Funcional 07	10
<b>Diagrama de clase</b>	11
<b>Diccionario de las clases</b>	12
Clase cliente	12
Clase balón de oxígeno	13
Clase pedido	13
Clase nodo	14
Clase lista de pedidos	15
Interfaz cola	15
Interfaz cola de prioridad	16
Clase cola de llenado que implementa la interfaz cola	16
Clase cola de espera que implementa la interfaz cola de prioridad	17
Clase archivo	17
<b>Algoritmos principales utilizados</b>	18
Lista enlazada	18
Agregar pedido a la lista enlazada	18
Modificar estado de un pedido	18
Modificar pedido identificado por id	18
Obtener pedido identificado por id	19
Cola	19
Encolar pedido	19
Desencolar pedido	19
Obtener pedido	19
Cola de prioridad	19
Ordenar pedidos por prioridad	19
Archivo	20
Guardar lista de pedidos en archivo	20
Recuperar lista de pedidos del archivo	20

<b>Guía de usuario</b>	21
Registrar pedido	21
Botón agregar balón de oxígeno	22
Botón eliminar balón de oxígeno	22
Botón generar pedido	22
Botón cancelar pedido	22
Cola de espera	22
Botón validar	23
Botón no validar	23
Cola de llenado	24
Botón despachar pedido	24
Rechazados	24
Botón modificar balón de oxígeno	25
Botón eliminar balón de oxígeno	25
Botón modificar pedido	26
Botón eliminar pedido	26
Historial	26
Pedido seleccionado en la tabla	27
Estado del pedido	27
<b>Referencias y bibliografía utilizada</b>	28
<b>Conclusiones</b>	28
<b>Recomendaciones</b>	28

## **1. Objetivo del proyecto**

En vista de la situación actual de pandemia en el que nos encontramos por el covid-19, se ha generado una gran necesidad de abastecimiento de oxígeno, esto ha traído como consecuencia una gran dificultad al intentar conseguirlo, tanto por el factor de desorden ya que no se abastece el número de demanda en comparación con el número de ofertantes como también el de demora, siendo este último un factor muy determinante en la vida de aquellos que se encuentra contagiados.

El objetivo de este proyecto es lograr que la solicitud de pedido de un balón de oxígeno sea dinámica y fluida, es decir que se subsane el factor de desorden y de demora que se padece actualmente, logrando de esta manera beneficiar a miles de familias afectadas por el covid-19.

## **2. Descripción del proyecto**

Este sistema permitirá hacer el registro del pedido, su búsqueda, modificación, eliminación y validación, lo que ayudará a tener un registro más ordenado para aquellos que ofrecen el servicio de llenado. Se utilizará el sistema de colas lo que permitirá agilizar el tiempo de despacho del pedido, lo que beneficia en gran medida a los clientes, atendiéndolos por el orden en el que hacen la solicitud y así evitar las malas disposiciones y entregas que se pueden malintencionar.

## **3. Alcances y limitaciones del proyecto**

- No todos los integrantes del grupo poseemos los mismos conocimientos en el lenguaje de programación Java, lo cual se debe de invertir un tiempo en la capacitación.
- Con este proyecto se implementará un demo o prototipo de un sistema de gestor de colas para la venta o llenado de oxígeno.
- Cabe destacar que con este demo o prototipo no pretendemos distribuir este sistema a los diferentes negocios dedicados a este rubro.
- Este sistema está destinado a ser usado por el dueño de negocio de venta o llenado de oxígeno y no va ser usado por quien está interesado en comprar el oxígeno.
- Buscamos ofrecer un soporte o una solución a las largas colas que se realizan fuera del negocio, con tal de que el interesado en comprar oxígeno no pierda su oportunidad o espacio en la fila.
- No todos los miembros del equipo contamos con conocimiento de programación en java .
- Debido al limitado tiempo con el que contamos se decidió centrarnos en el llenado de balones de oxígeno.

#### 4. Requerimientos funcionales del sistema

En la siguiente tabla se enumeran los requerimientos funcionales del sistema de llenado de balones de oxígeno.

##### 4.1. Tabla general de requerimientos funcionales

ID	Nombre	Descripción
RF_01	Registro de pedido	La aplicación debe permitir ingresar toda la información necesaria de cada pedido, almacenados en archivos.
RF_02	Búsqueda de pedido	La aplicación debe permitir buscar la información del pedido en la estructura de datos dado el código del pedido y mostrarla en pantalla.
RF_03	Modificación de pedido	La aplicación debe permitir modificar un pedido que aún se encuentre en la lista de espera dado el código del pedido y realizar las actualizaciones correspondientes.
RF_04	Eliminación de pedido	La aplicación debe permitir eliminar el pedido solo si este aún se encuentra en la lista de espera dado el código del pedido.
RF_05	Validación de pedido	La aplicación debe permitir validar la información del pedido cuando este se encuentre en la lista de espera dado el código del pedido.
RF_06	Inserción de pedido a la cola de llenado	La aplicación debe permitir ingresar toda la información necesaria para cada pedido en la cola de llenado.
RF_07	Despacho de pedido de la cola de llenado	La aplicación debe permitir ingresar toda la información relevante acerca de la salida de pedidos de la cola de llenado, almacenarlos en archivos y realizar las actualizaciones correspondientes.

A continuación se detallan los requerimientos funcionales previamente mencionados en la tabla anterior.

#### 4.1.1. Requisito Funcional 01

RF_01	
<b>Nombre:</b>	Registro de pedido
<b>Autor:</b>	Jorge Marin
<b>Fecha:</b>	27/09/2021
<b>Descripción:</b> La aplicación debe permitir ingresar toda la información necesaria de cada pedido, para posteriormente almacenarla en la base de datos.	
<b>Actores:</b> Cliente. Recepcionista.	
<b>Precondiciones:</b> El sistema debe contar con persistencia de datos a través archivos.	
<b>Flujo normal:</b> <ol style="list-style-type: none"> <li>1. El cliente inicia el proceso solicitando un pedido.</li> <li>2. El recepcionista recibe los datos del cliente así como los datos del balón de oxígeno suministrado y lo ingresa al sistema.</li> <li>3. El pedido es almacenado en el archivo.</li> <li>4. El pedido entra a la lista de espera por validación.</li> </ol>	
<b>Flujo alternativo:</b> Ninguno.	
<b>Postcondiciones:</b> El pedido y sus datos han sido registrados correctamente.	

#### 4.1.2. Requisito Funcional 02

RF_02	
<b>Nombre:</b>	Búsqueda de pedido
<b>Autor:</b>	Jorge Marin
<b>Fecha:</b>	27/09/2021
<b>Descripción:</b> La aplicación debe permitir buscar la información del pedido en la estructura de datos dado el código del pedido y posteriormente mostrarla en pantalla.	
<b>Actores:</b> Recepcionista. Supervisor.	

<b>Precondiciones:</b> El sistema debe contar con persistencia de datos. El pedido debe encontrarse almacenado en la estructura de datos.
<b>Flujo normal:</b> 1. El recepcionista o supervisor digita el código del pedido. 2. El sistema busca el pedido en la estructura de datos. 3. El sistema muestra en pantalla los datos del pedido.
<b>Flujo alternativo:</b> 2. El sistema busca el pedido en la estructura de datos, de no ser encontrado notifica que el pedido no se encuentra registrado. 3. El sistema solicita el ingreso de un código válido.
<b>Postcondiciones:</b> Ninguno.

#### 4.1.3. Requisito Funcional 03

RF_03	
<b>Nombre:</b>	Modificación de pedido
<b>Autor:</b>	Rodrigo Quinteros
<b>Fecha:</b>	27/09/2021
<b>Descripción:</b> La aplicación debe permitir modificar un pedido que aún se encuentre en la lista de espera dado el código del pedido y realizar las actualizaciones correspondientes.	
<b>Actores:</b> Recepcionista. Supervisor.	
<b>Precondiciones:</b> El sistema debe contar con persistencia de datos. El sistema debe contar con una lista de espera.	
<b>Flujo normal:</b> 1. El recepcionista o supervisor digita el código del producto. 2. El sistema busca el pedido en la estructura de datos. 3. El sistema comprueba si el pedido se encuentra en la estructura de datos. 4. El sistema comprueba si el pedido se encuentra en la lista de espera. 5. El recepcionista o supervisor procede a actualizar los datos del pedido. 6. El sistema comprueba la validez de los datos y los almacena.	
<b>Flujo alternativo:</b> 4. El sistema notifica que el pedido no ha sido encontrado en la estructura de datos. 5. El sistema solicita ingresar un código válido.	

**Postcondiciones:**

El pedido y sus datos han sido actualizados correctamente.

## 4.1.4. Requisito Funcional 04

RF_04	
<b>Nombre:</b>	Eliminación de pedido
<b>Autor:</b>	Rodrigo Quinteros
<b>Fecha:</b>	27/09/2021
<b>Descripción:</b> La aplicación debe permitir eliminar el pedido solo si este aún se encuentra en la lista de espera dado el código del pedido.	
<b>Actores:</b> Supervisor	
<b>Precondiciones:</b> El sistema debe contar con persistencia de datos. El sistema debe contar con una lista de espera.	
<b>Flujo normal:</b> <ol style="list-style-type: none"><li>1. El supervisor digita el código del producto.</li><li>2. El sistema comprueba el pedido en la estructura de datos.</li><li>3. El sistema solicita confirmación al supervisor.</li><li>4. El sistema elimina el pedido.</li></ol>	
<b>Flujo alternativo 01:</b> <ol style="list-style-type: none"><li>2. El sistema comprueba el pedido en la estructura de datos, de no ser así notifica que el pedido no se encuentra registrado permitiéndole que corrija el código ingresado.</li></ol> <b>Flujo alternativo 02:</b> <ol style="list-style-type: none"><li>3. El sistema solicita confirmación al supervisor, si este declina el proceso se cancela y el sistema solicita un nuevo código de pedido.</li></ol>	
<b>Postcondiciones:</b> El pedido y sus datos han sido eliminados correctamente.	

## 4.1.5. Requisito Funcional 05

RF_05	
<b>Nombre:</b>	Validación de pedido
<b>Autor:</b>	Jorge Marin



<b>Fecha:</b>	27/09/2021
<b>Descripción:</b> La aplicación debe permitir validar la información del pedido cuando este se encuentre en la lista de espera dado el código del pedido.	
<b>Actores:</b> Supervisor	
<b>Precondiciones:</b> El sistema debe contar con una lista de espera El pedido debe encontrarse registrado en la estructura de datos.	
<b>Flujo normal:</b> <ol style="list-style-type: none"> <li>1. El supervisor verificará los datos ingresados del primer pedido de la lista de espera.</li> <li>2. De esta todo conforme el supervisor ratifica los datos ingresados presionado el botón validar.</li> <li>3. El sistema actualiza el estado del pedido y lo almacena en archivo.</li> <li>4. El sistema envía el pedido a la cola de llenado.</li> </ol>	
<b>Flujo alternativo:</b> <ol style="list-style-type: none"> <li>2. De presentar irregularidades el supervisor cancela el pedido presionando el botón denegar y se le notificará al cliente el nuevo estatus del pedido para su posterior modificación o eliminación.</li> </ol>	
<b>Postcondiciones:</b> El pedido y sus datos se encuentran almacenados en un archivo.	

#### 4.1.6. Requisito Funcional 06

RF_06	
<b>Nombre:</b>	Inserción de pedido a la cola de llenado
<b>Autor:</b>	Jorge Marin
<b>Fecha:</b>	27/09/2021
<b>Descripción:</b> La aplicación debe permitir ingresar toda la información necesaria para cada pedido en la cola de llenado.	
<b>Actores:</b> Supervisor.	
<b>Precondiciones:</b> El sistema debe contar con una cola de llenado. El sistema debe contar con una lista de espera.	
<b>Flujo normal:</b> <ol style="list-style-type: none"> <li>1. El sistema recibe el pedido validado de la lista de espera</li> </ol>	

2. El sistema coloca el pedido recibido en la cola de llenado 3. El sistema notifica al supervisor.
<b>Flujo alternativo:</b> Ninguno
<b>Postcondiciones:</b> El pedido y sus datos se encuentran en la cola de llenado.

#### 4.1.7. Requisito Funcional 07

RF_ 07	
<b>Nombre:</b>	Despacho de pedido de la cola de llenado
<b>Autor:</b>	Rodrigo Quinteros
<b>Fecha:</b>	27/09/2021
<b>Descripción:</b> La aplicación debe permitir ingresar toda la información relevante acerca de la salida de pedidos de la cola de llenado, almacenarlos en archivos y realizar las actualizaciones correspondientes.	
<b>Actores:</b> Personal encargado.	
<b>Precondiciones:</b> El sistema debe contar con persistencia de datos. El sistema debe contar con una cola de llenado.	
<b>Flujo normal:</b> <ol style="list-style-type: none"> <li>1. El sistema selecciona el primer pedido de la cola de llenado y muestra los datos al personal encargado..</li> <li>2. El personal encargado procede con el llenado del balón.</li> <li>3. El personal encargado despacha el pedido, presionando el botón despachar.</li> <li>4. El sistema almacena los datos y realiza las actualizaciones correspondientes.</li> <li>5. El sistema notifica que el pedido fue despachado.</li> </ol>	
<b>Flujo alternativo:</b> Ninguno	
<b>Postcondiciones:</b> El proceso del pedido ha sido concluido correctamente.	

```

classDiagram
    class Cliente {
        -dni:string
        -nombre:string
        -apellido:string
        -estado:int
        Cliente():constructor
        +getDni():string
        +getNombre():string
        +getApellido():string
        +getEstado():int
        +setDni(string):void
        +setNombre(string):void
        +setApellido(string):void
        +setEstado(string):void
    }

    class BalonDeOxigeno {
        -codigo:string
        -capacidad:float
        BalonDeOxigeno():constructor
        +getCodigo():string
        +getCapacidad():float
        +setCodigo(string):void
        +setCapacidad(float):void
    }

    class Pedido {
        -id:int
        -fecha:date
        -estado:int
        -cliente:Cliente
        -balones[]:BalonDeOxigeno
        Pedido():constructor
        +getId():int
        +getFecha():date
        +getEstado():int
        +getCliente():Cliente
        +getBalones():BalonesDeOxigeno[]
        +setFecha(date):void
        +setEstado(int):void
        +setCliente(Cliente):void
        +setBalones(Balones[]):void
    }

    class Archivo {
        -PATH:string
        -miFile:file
        -miFileInputStream:FileInputStream
        -miFileOutputStream:FileOutputStream
        -miObjectInputStream:ObjectInputStream
        -miObjectOutputStream:ObjectOutputStream
        +GuardarDatosEnArchivo(LinkedList<Pedidos> list):void
        +RecuperarDatoDeArchivo():void
    }

    class ListaPedidos {
        -cabeza:Nodo
        -ultimo:Nodo
        -tamanio:int
        +agregarPedido(Pedido p):void
        +getListaEnlazada():LinkedList<Pedido> list
        +getTamanio():int
        +isEmpty():boolean
        +modificarEstado(int id, int estado):void
        +modificarPedidoPorId(int id, Pedido p):void
        +obtenerPedido(int id):Pedido p
    }

    class Cola {
        <<Interfaz>>
        +encolar(Pedido p):void
        +desencolar():void
        +obtener():Pedido
        +getCola():LinkedList<Pedido> cola
    }

    class ColaDeLlenado {
        -cabeza:Nodo
        -ultimo:Nodo
    }

    class ColaDeEspera_Rechazados {
        ColaDeEspera / ColaDeRechazados
    }

    class ColaPrioridad {
        <<Interfaz>>
        Cola prioridad
        +ordenarPrioridad():void
    }

    class Nodo {
        -dato:Pedido
        -nodoSiguiente:Nodo
    }

    Cliente "1" *-- "N" Pedido
    BalonDeOxigeno "1" *-- "N" Pedido
    Pedido "1" o-- "N" ListaPedidos
    Archivo "1" -- "N" ListaPedidos : Hace uso
    ListaPedidos "1" -- "N" Nodo
    ColaDeLlenado <|-- ColaDeEspera_Rechazados
    ColaDeLlenado <.. Cola : <--
    ColaDeLlenado <.. ColaPrioridad : <--
    ColaDeLlenado --> ListaPedidos : Inserta
    ColaDeEspera_Rechazados --> ListaPedidos : Inserta
    ColaDeLlenado --> Cola : Hace uso
    ColaPrioridad <.. ColaDeEspera_Rechazados : <--
    
```

## 7. Diccionario de las clases

### 7.1. Clase cliente

Nombre de la clase:	Cliente	
Objetivo:	El objetivo de esta clase es representar quién va a realizar el pedido de un producto que en este caso será un balón de oxígeno.	
Atributos		
Nombre	Tipo de dato	Descripción
dni	string	Este atributo se refiere al documento de identidad perteneciente a los clientes.
nombre	string	Hace referencia al nombre del cliente.
apellido	string	Hace referencia al apellido del cliente.
estado	int	Hace referencia al estado de salud del paciente que requiere el balón de oxígeno, pudiendo ser: leve (3), moderado (2), grave (1).
Métodos		
Nombre	Descripción	
getDni()	Servirá para enviar el dato de dni.	
getNombre()	Servirá para enviar el dato de nombre.	
getApellido()	Servirá para enviar el dato de apellido.	
getEstado()	Servirá para enviar el dato de estado del paciente.	
setDni()	Servirá para recibir el dato de dni.	
setNombre()	Servirá para recibir el dato de nombre.	
setApellido()	Servirá para recibir el dato de apellido.	
setEstado()	Servirá para recibir el dato de estado del paciente.	

### 7.2. Clase balón de oxígeno

Nombre de la clase:	BalonDeOxigeno	
Objetivo:	El objetivo de esta clase es funcionar como aquella clase que representará al objeto que va a solicitar el cliente.	
Atributos		
Nombre	Tipo de dato	Descripción
codigo	string	Este atributo se refiere al código que identificará y diferenciará un balón de otro.
capacidad	float	Hace referencia a si el balón se encuentra totalmente vacío o si aún contiene algo de oxígeno.
Métodos		
Nombre	Descripción	
getCodigo()	Servirá para enviar el código del balón de oxígeno.	
getCapacidad()	Servirá para enviar la capacidad que tiene el balón.	
setCodigo()	Servirá para recibir el código del balón de oxígeno.	
setCapacidad()	Servirá para recibir la capacidad que tiene el balón.	

### 7.3. Clase pedido

Nombre de la clase:	Pedido	
Objetivo:	El objetivo de esta clase es representar al pedido de llenado de oxígeno realizado por el cliente.	
Atributos		
Nombre	Tipo de dato	Descripción
id	int	Este atributo se refiere al código que identificara y diferenciara a un pedido de otro.
fecha	date	Hace referencia a la fecha en que se realizará el pedido.
estado	int	Hace referencia en qué cola se encuentra el pedido,

		cola de espera (1), cola de llenado (2), cola de rechazados (3), eliminado (4), despachado (5).
cliente	Cliente	Hace referencia al nombre del cliente que irá en el pedido.
<b>Métodos</b>		
<b>Nombre</b>	<b>Descripción</b>	
getId()	Servirá para enviar el dato de id del pedido.	
getFecha()	Servirá para enviar el dato de la fecha en que se realizó el pedido.	
getEstado()	Servirá para enviar el dato del estado del pedido.	
getCliente()	Servirá para enviar el dato del nombre del cliente que está en el pedido.	
getBalones()	Servirá para enviar el dato de los balones solicitados.	
getFecha()	Servirá para recibir el dato de la fecha en que se realizó el pedido.	
setEstado()	Servirá para recibir el dato del estado del pedido.	
setCliente	Servirá para recibir el dato del nombre del cliente que está en el pedido.	
setBalones	Servirá para recibir el dato de los balones solicitados.	

#### 7.4. Clase nodo

Nombre de la clase:	Nodo	
Objetivo:	El objetivo de esta clase es hacer referencia al Nodo que se generará.	
Atributos		
Nombre	Tipo de dato	Descripción
dato	Pedido	Se refiere al dato con respecto al pedido
nodoSiguiente	Nodo	Se refiere al dato con respecto al nodo siguiente.

### 7.5. Clase lista de pedidos

Nombre de la clase:		ListaPedidos
Objetivo:		El objetivo de esta clase es alojar los pedidos a manera de una lista.
Atributos		
Nombre	Tipo de dato	Descripción
cabecera	Nodo	Se refiere al dato que está adelante en la lista.
último	Nodo	Se refiere al dato que está último en la lista.
tamano	int	Se refiere a la cantidad total de elementos de la lista.
Métodos		
Nombre		Descripción
agregarPedido(pedido)		Se encargará de registrar el pedido.
obtenerPedido(id)		Permitirá obtener el pedido según el id respectivo.
modificarPedidoPorId(id, pedido)		Permitirá modificar un pedido según el id dado.
getTamano()		Permitirá obtener la cantidad de elementos de la lista
isEmpty()		Permitirá saber si la lista se encuentra vacía o no.
modificarEstado(id, estado)		Permitirá modificar el estado del pedido
getListaEnlazada()		Permitirá obtener todos los elementos de la lista en un objeto LinkedList para su almacenamiento y manejo en la interfaz gráfica.

### 7.6. Interfaz cola

Nombre de la interfaz:	Cola
Objetivo:	Esta clase funcionará a manera de interfaz y permitirá visualizar la cola.

Métodos	
Nombre	Descripción
encolar(Pedido)	Permitirá encolar los pedidos.
desencolar()	Permitirá desencolar los pedidos.
obtener(int)	Permitirá obtener los pedidos.
getCola()	Permitirá obtener todos los elementos de la cola en un objeto LinkedList para su almacenamiento y manejo en la interfaz gráfica.

#### 7.7. Interfaz cola de prioridad

Nombre de la interfaz:	Cola prioridad
<b>Objetivo:</b>	Esta clase funcionará a manera de una cola de prioridad de los pedidos que llegan.
Métodos	
Nombre	Descripción
ordenarPrioridad()	Permitirá ordenar en prioridad los pedidos en la cola.

#### 7.8. Clase cola de llenado que implementa la interfaz cola

Nombre de la clase:	ColaDeLlenado	
Objetivo:	El objetivo de esta clase es recepcionar los pedidos a manera de una cola.	
Atributos		
Nombre	Tipo de dato	Descripción
cabecera	Nodo	Se refiere al dato que está adelante en la cola.
último	Nodo	Se refiere al dato que está último en la cola.



7.9. Clase cola de espera que implementa la interfaz cola de prioridad

<b>Nombre de la clase:</b>	<b>ColaDeEspera</b>
<b>Objetivo:</b>	El objetivo de esta clase es poner los pedidos en una cola de espera.

7.10. Clase archivo

Nombre de la clase:	Archivo	
Objetivo:	El objetivo de esta clase es guardar los datos de la lista de pedidos.	
Atributos		
Nombre	Tipo de dato	Descripción
PATH	string	Define la ruta y nombre del archivo.
miFile	File	Se usará para obtener información sobre el archivo.
miFileInputStream	FileInputStream	Crea un fichero, salvo que exista y sea de sólo lectura.
miFileOutputStream	FileOutputStream	Escribe datos de una clase de una forma portable.
miObjectInputStream	ObjectInputStream	Permite leer objetos de una clase.
miObjectOutputStream	ObjectOutputStream	Realiza la serialización de los objetos de una clase.
Métodos		
Nombre		Descripción
GuardarDatosEnArchivo(listaDePedidos)		Se encargará de guardar los datos de los pedidos.
RecuperarDatosDeArchivo()		Se encargará de recuperar los datos de los pedidos.

## 8. Algoritmos principales utilizados

### 8.1. Lista enlazada

#### 8.1.1. Agregar pedido a la lista enlazada

##### **Algoritmo agregarPedido(mi Pedido) : Vacío**

```
nuevoNodo.pedido ← miPedido
Si (cabeza = nulo)
    cabeza ← nuevoNodo
Sino
    ultimo.siguiete ← nuevoNodo
Fsi
ultimo ← nuevoNodo
tamanio++

// Inserta en su respectiva cola según estado del pedido
Según (miPedido.estado)
    caso 1:
        miColaDeEspera.encolar(miPedido)
    caso 2:
        miColaDeLlenado.encolar(miPedido)
    caso 3:
        miColaDeRechazados.encolar(miPedido)
Fsegún
```

#### 8.1.2. Modificar estado de un pedido

##### **Algoritmo modificarEstado(idPedido, nuevoEstado) : Vacío**

```
// Busca pedido y modifica el estado
Si (cabeza <> nulo)
    nodo actual ← cabeza
    Mientras (actual.pedido.Id() <> idPedido)
        actual ← actual.siguiete
    Fmientras
    actual.pedido.Estado(nuevoEstado)

// Inserta en su respectiva cola según nuevo estado
según (nuevoEstado)
    caso 1:
        miColaDeEspera.encolar(actual.pedido)
    caso 2:
        miColaDeLlenado.encolar(actual.pedido)
    caso 3:
        miColaDeRechazados.encolar(actual.pedido)
Fsegún
Fsi
```

#### 8.1.3. Modificar pedido identificado por id

##### **Algoritmo modificarPedidoPorID(idPedido, pedidoModificado) : Vacío**

```
// Busca pedido en posición dada y lo modifica
Si (cabeza <> nulo)
    nodo actual ← cabeza
    Mientras (actual.pedido.Id() <> idPedido)
        actual ← actual.siguiete
    Fmientras
    actual.pedido ← pedidoModificado

// Una vez modificado lo encola en la lista de espera
miColaDeEspera.encolar(pedidoModificado)
Fsi
```

#### 8.1.4. Obtener pedido identificado por id

##### **Algoritmo obtenerPedido(idPedido) : Nulo, Pedido**

```
// Busca pedido en posición dada y lo retorna
pedidoBuscado ← nulo
Si (cabeza <> nulo)
    nodo actual ← cabeza
    Mientras (actual.pedido.Id() <> idPedido)
        actual ← actual.siguiente
    Fmientras
    pedidoBuscado ← actual.pedido
Fsi
Retorna pedidoBuscado
```

### 8.2. Cola

#### 8.2.1. Encolar pedido

##### **Algoritmo encolar(pedido) : Vacío**

```
nuevoNodo.pedido ← pedido
Si (cabeza = nulo)
    cabeza ← nuevoNodo
Sino
    ultimo.siguiente ← nuevoNodo
Fsi
ultimo ← nuevoNodo
tamanio++
```

#### 8.2.2. Desencolar pedido

##### **Algoritmo desencolar() : Vacío**

```
Si (cabeza <> null)
    nodo eliminar ← cabeza
    cabeza ← cabeza.siguiente
    eliminar.siguiente ← nulo
    Si (cabeza = nulo)
        ultimo ← nulo
    Fsi
Fsi
```

#### 8.2.3. Obtener pedido

##### **Algoritmo Pedido obtener() : Nulo, Pedido**

```
Si (cabeza = nulo)
    Retorna nulo
Sino
    Retorna cabeza.pedido
Fsi
```

### 8.3. Cola de prioridad

#### 8.3.1. Ordenar pedidos por prioridad

##### **Algoritmo ordenarPrioridad() : Vacío**

```
nodo actual ← cabeza
Si (actual <> nulo)
    pedido aux
    nodoSiguiente
    Mientras (actual.siguiente <> nulo)
        nodoSiguiente ← actual.siguiente
        Mientras (nodoSiguiente <> nulo)
```

```

        Si (actual.pedido.Cliente().EstadoPaciente() >
nodoSiguiente.pedido.Cliente().EstadoPaciente())
            aux ← actual.pedido
            actual.pedido ← nodoSiguiente.pedido
            nodoSiguiente.pedido ← aux
        Fsi
        nodoSiguiente ← nodoSiguiente.siguiente
    Fmientras
    actual ← actual.siguiente
Fmientras
Fsi

```

#### 8.4. Archivo

##### 8.4.1. Guardar lista de pedidos en archivo

###### **Algoritmo guardarDatosEnArchivo(miLista) : Vacío**

```

    Abre conexión con archivo
    Escribe objeto lista enlazada en archivo
    Cierra conexión con archivo

```

##### 8.4.2. Recuperar lista de pedidos del archivo

###### **Algoritmo recuperarDatosDeArchivo() : Nulo, LinkedList**

```

    LinkedList datosRecuperados = nulo
    Abre conexión con archivo
    datosRecuperados ← Lee objeto lista enlazada
    Cierra conexión con archivo
    Retorna datosRecuperados

```

## 9. Guía de usuario

Esta sección describe el conjunto de pasos para el correcto uso del programa y analiza las funcionalidades que nos ofrece.

### 9.1. Registrar pedido

En este panel, el usuario una vez solicitado los datos al cliente (DNI, nombre, apellido, estado del paciente, código del balón y capacidad del balón) podrá rellenar los campos respectivos y registrarlo en el sistema.

Queuing system project

Sistema de colas - LLenado balones de oxígeno

Registrar pedido Cola de espera Cola de llenado Rechazados Historial

D.N.I. del cliente : 73367034 Estado del paciente : Moderado

Nombre del cliente : Jorge Luis Apellido del cliente : Marin Evangelista

Código del balón Capacidad del balón (m3) :

Balones de oxígeno

Código	Capacidad
18200275	10.8

Agregar balón de oxígeno Eliminar balón de oxígeno

Generar pedido Cancelar pedido

Además es posible agregar más de un balón para un mismo cliente cada uno de estos será identificado por el código del mismo.

A Agregar balón de oxígeno B Eliminar balón de oxígeno

C Generar pedido D Cancelar pedido

A continuación se procede a detallar la funcionalidad de los botones visualizados en el panel.

A. Botón agregar balón de oxígeno

Con este botón es posible agregar uno o más de un balón de oxígeno por cliente.

Cada vez que agregue un nuevo balón los campos a ingresar deben estar correctamente escritos.

B. Botón eliminar balón de oxígeno

Ya sea por un error de tipeo o un balón incorrecto será posible eliminar el balón respectivo seleccionando y haciendo uso de este botón.

C. Botón generar pedido

Una vez ingresado correctamente todos los datos en sus respectivos campos, se podrá generar un nuevo pedido haciendo uso de este botón.

D. Botón cancelar pedido

Si el cliente desiste en continuar la orden por uno u otro motivo, el usuario es capaz de cancelar el pedido en su totalidad haciendo uso de este botón.

Una vez registrado un nuevo pedido, este entrará a la cola de espera esperando validación por el supervisor en turno.

9.2. Cola de espera

En este panel, los nuevos pedidos son ordenados de acuerdo al estado del paciente suministrado al momento de registro, el orden empieza por el estado “grave” seguido

por “moderado” y finalizando en “leve” en la cola de prioridad.

ID	Fecha	Estado del Paciente
7	Fri Sep 03 07:28:25 PET 2021	Grave
3	Fri Sep 03 07:28:26 PET 2021	Moderado
8	Fri Sep 03 07:28:04 PET 2021	Leve

D.N.I. del cliente :	Estado del paciente :
<input type="text" value="73002886"/>	<input type="text" value="Grave"/>
Nombre del cliente :	Apellido del cliente :
<input type="text" value="Harold Giusseppe"/>	<input type="text" value="Alberto Ramos"/>

Balones de oxígeno	
Código	Capacidad
<input type="text" value="18200306"/>	<input type="text" value="12.25"/>

El supervisor en turno solo podrá revisar el primer pedido de la cola, decidir si los parámetros ingresados son válidos y verídicos, si es así procede a validarlos y enviarlos a la cola de llenado, caso contrario el pedido es rechazado y enviado a la cola de rechazados para su modificación o eliminación.

A continuación se procede a explicar el uso de los botones visualizados en el panel.

**A**  **B**

**A. Botón validar**

Haciendo uso de este botón se valida el primer pedido de la cola de espera.

**B. Botón no validar**

Haciendo uso de este botón se rechaza el primer pedido de la cola de espera.

Si el pedido fue validado entonces este pasa a la cola de llenado para su posterior despacho, caso contrario el pedido entra en la cola de rechazados para su modificación y así subsanar errores en el registro o su eliminación definitiva.

### 9.3. Cola de llenado

En este panel, se insertan los pedidos validados en una cola de acuerdo al orden de llegada para su llenado y despacho.

ID	Fecha	Estado del Paciente
7	Fri Sep 03 07:28:25 PET 2021	Grave
3	Fri Sep 03 07:28:26 PET 2021	Moderado
8	Fri Sep 03 07:28:04 PET 2021	Leve

Detalles del primer pedido

D.N.I. del cliente : 73002886

Estado del paciente : Grave

Nombre del cliente : Harold Giusseppe

Apellido del cliente : Alberto Ramos

Balones de oxígeno

Código 18200306

Capacidad 12.25

Validar No Validar

El encargado en cuestión deberá llenar el balón de oxígeno y actualizar la cola haciendo uso del botón despachar para así proseguir con el siguiente pedido hasta finalizar.

A continuación se procede a explicar el uso del botón visualizado en el panel.

A

#### A. Botón despachar pedido

Haciendo uso de este botón, el encargado podrá actualizar la cola de llenado y proseguir con el siguiente pedido hasta finalizarla.

### 9.4. Rechazados

En este panel se insertarán todos los pedidos que no fueron validados en la cola de espera para su modificación o eliminación, estos pedidos son manejados a través de



una cola de prioridad según el estado del paciente.

Queuing system project

### Sistema de colas - LLenado balones de oxígeno

Registrar pedido Cola de espera Cola de llenado Rechazados Historial

Cola de pedidos rechazados

ID	Fecha	Estado Del Paciente
4	Fri Sep 03 07:28:24 PET 2021	Grave
5	Fri Sep 03 07:07:54 PET 2021	Moderado
2	Fri Sep 03 07:02:25 PET 2021	Leve

Detalles del primer pedido

D.N.I. del cliente : 74997292

Estado del paciente : Grave

Nombre del cliente : Rodrigo Ervin

Apellido del cliente : Quinteros Peralta

Código del balón :

Capacidad del balón (m3) :

Balones de oxígeno

Código	Capacidad
18200316	15.7

Modificar balón de oxígeno Eliminar balón de oxígeno

Modificar pedido Eliminar pedido

A continuación se procede a explicar el uso del botón visualizado en el panel.

A Modificar balón de oxígeno B Eliminar balón de oxígeno

C Modificar pedido D Eliminar pedido

A. Botón modificar balón de oxígeno

Haciendo uso de este botón el usuario podrá modificar los parámetros del balón de oxígeno del primer pedido de la cola de prioridad.

Para poder modificar dicho parámetro el usuario deberá seleccionar el balón a modificar en la tabla “balones de oxígeno”.

B. Botón eliminar balón de oxígeno

Haciendo uso de este botón el usuario podrá eliminar un balón previamente seleccionado en la tabla.

C. Botón modificar pedido

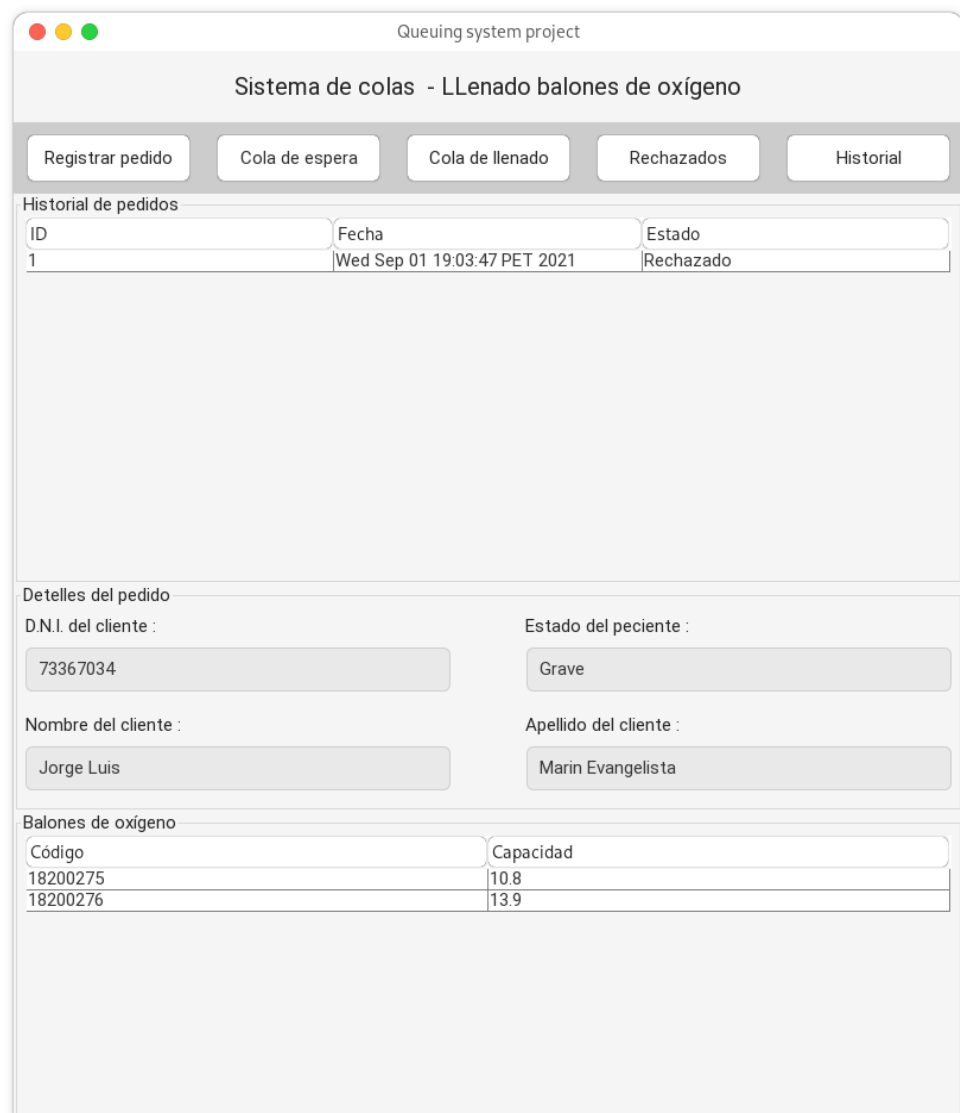
Haciendo uso de este botón el usuario podrá guardar los parámetros modificados del primer pedido de la cola y enviarlo a la cola de espera para su nueva validación.

D. Botón eliminar pedido

Haciendo uso de este botón el usuario podrá eliminar el primer pedido rechazado de la cola de prioridad y proseguir con el siguiente hasta finalizar esta misma.

## 9.5. Historial

En este panel, se almacenarán todos los pedidos ingresados al sistema, ya sean válidos, rechazados o eliminados, nunca se pierde el seguimiento de estos.



Queuing system project

Sistema de colas - LLenado balones de oxígeno

Registrar pedido Cola de espera Cola de llenado Rechazados Historial

Historial de pedidos

ID	Fecha	Estado
1	Wed Sep 01 19:03:47 PET 2021	Rechazado

Detalles del pedido

D.N.I. del cliente : 73367034 Estado del paciente : Grave

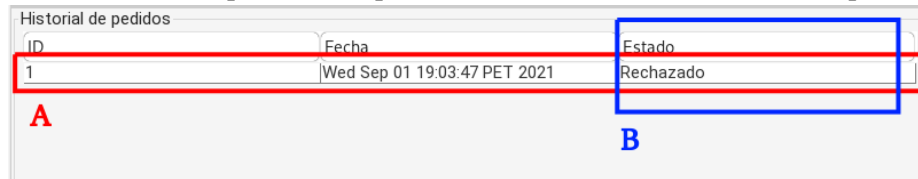
Nombre del cliente : Jorge Luis Apellido del cliente : Marin Evangelista

Balones de oxígeno

Código	Capacidad
18200275	10.8
18200276	13.9

Además el estado del pedido se actualiza según en qué cola se encuentre el pedido. Los pedidos son ordenados según el id que les genera el sistema, no habrá dos pedidos con el mismo id.

A continuación se procede a explicar el uso del botón visualizado en el panel.



The screenshot shows a web interface titled 'Historial de pedidos'. It contains a table with three columns: 'ID', 'Fecha', and 'Estado'. The first row of the table is highlighted with a red border, and the 'Estado' column is highlighted with a blue border. Below the table, there are two labels: a red 'A' pointing to the first row and a blue 'B' pointing to the 'Estado' column.

ID	Fecha	Estado
1	Wed Sep 01 19:03:47 PET 2021	Rechazado

A. Pedido seleccionado en la tabla

Cada vez que el usuario seleccione un pedido de la tabla, los datos de este, serán cargados en el formulario para mayor detalle.

Cuando el usuario cambie de panel, el formulario será limpiado.

B. Estado del pedido

Según el que cola del sistema se encuentre el pedido, la columna de estado se irá actualizando para darle seguimiento en todo momento.

## **10. Referencias y bibliografía utilizada**

Ruiz Castellanos, J. & González Muñoz, L. (2014). Proyecto de aplicación de teoría de colas, caso aplicado a departamentos de servicios estudiantiles, UNPHU [Tesis de Bachillerato, Universidad Nacional Pedro Henriquez Ureña]. Repositorio Institucional – Universidad Nacional Pedro Henriquez Ureña.

Arista Arevalo, J (2016) Aplicación de la teoría de colas al problema de atención al cliente para la optimización del número cajeros en ventanillas en la organización BCP [Tesina de Bachillerato, Universidad Nacional Mayor de San Marcos]. Repositorio Institucional – Universidad Nacional Mayor de San Marcos.

Arévalo Pabón, A. L. (2018). Aplicación de la Teoría de Colas en Tiempos de Espera para la Atención de Usuarios en el Laboratorio Clínico de la Empresa IPS Unipsalud 2000 Guaduas Ltda [Tesis de Bachillerato, Universidad Militar Nueva Granada]. Repositorio Institucional – Universidad Militar Nueva Granada.

## **11. Conclusiones**

Por todo lo visto anteriormente durante el desarrollo del informe, se llegó a una serie de conclusiones las cuales se menciona a continuación:

- De acuerdo a los resultados mostrados durante la puesta en escena del demo implementado según lo descrito en el informe, podemos concluir que, en primera instancia, cumple con el objetivo inicial del proyecto.
- Adicionalmente los resultados arrojados por la ejecución demuestran que son necesarios ciertos ajustes en algunas funcionalidades, antes de su aplicación en un entorno laboral real.
- Podemos concluir también que debido a lo mayoritariamente simple e intuitivo que resulta la interfaz del proyecto, el tiempo invertido en el aprendizaje sobre la operación del mismo será de un periodo corto.

## **12. Recomendaciones**

- Extender los estudios del caso del proyecto con el fin de una futura mejora en cuanto las funcionalidades del mismo.
- Tomar en consideración la implementación del demo del proyecto en una futura implementación completa, práctica y funcional del proyecto
- Extender los estudios teóricos acerca de colas presentados en el informe a fin de ampliar los conocimientos relacionados a este tema.