ESTRUCTURA DE DATOS Y ALGORITMOS 1

Obligatorio 2020/s2

Foto estudiante A
Matías Poletti
251602

Foto estudiante B
Fernando Ugarte
175433

Grupo N3B
Nombre del docente: Rafael Cohen
Fecha: 26/11/2020

Índice

Contenido

Interfaz Sistema: Pre y post condiciones

```
//Pre-condiciones: cantCiudades número entero positivo.
//Post-condiciones: Retorna una Lista de ciudades de largo =
cantCiudades.
```

- `public Retorno crearSistemaIngreso(int cantCiudades);`

```
//Pre-condiciones: no recibe datos
//Post-condiciones: devuelve una lista de ciudades vacías
```

- `public Retorno destruirSistemaIngreso();`

```
//Pre-condiciones: Recibe un String que no puede estar vacío (nulo) y
debe de ser único en la listaCiudades.
//Post-condiciones: Retorna un resultado.OK o resultado.ERROR en caso
de ser un string vacío o repetido.
```

- `public Retorno registrarCiudad(String Ciudad);`

```
//Pre-condiciones: Recibe strings Ciudad y Nombre, deben ser distintos
de null y únicos. Recibe también un int Estrellas >= 0 y <= 5 y un int
Ranking que debe ser == 0
//Post-condiciones: Retorna un resultado.OK o resultado.ERROR y un
Resultado.ValorString
```

- `public Retorno RegistrarHospital(String Ciudad, String Nombre, int Estrellas, int Capacidad, int ranking);`

```
//Pre-condiciones: Recibe strings Ciudad, Hospital y Servicio. Deben
ser distintos de null.
//Post-condiciones: Retorna un resultado.OK o resultado.ERROR en caso
de ser un string vacío o de no existir Ciudad u Hospital.
```

- `public Retorno ingresarServicio(String Ciudad, String Hospital, String Servicio);`

```
//Pre-condiciones: Recibe strings Ciudad, Hospital y Servicio. Todos
deben ser distintos de null. Ciudad debe existir en la listaCiudades y
elhospital debe existir en la lista hospitaled de la ciudad.
//Post-condición: Retorna resultado OK si elimina el servicio de la
lista. Error Si no se encuentra o si es null.
```

- `public Retorno borrarServicio(String Ciudad, String Hospital, String Servicio);`

```
//Pre-condiciones: Recibe strings Ciudad, Hospital y Comentario. Todos
deben ser distintos de null. Ciudad debe existir en la listaCiudades y
elhospital debe existir en la lista hospitaled de la ciudad. Recibe un
int Ranking >= 0 y <= 5.
//Post-condiciones: Ingresa un nuevo Comentario y ranking a las listas
del hospital y calcula el promedio y lo asigna a la propiedad ranking
del hospital. De lo contrario devuele Error
```

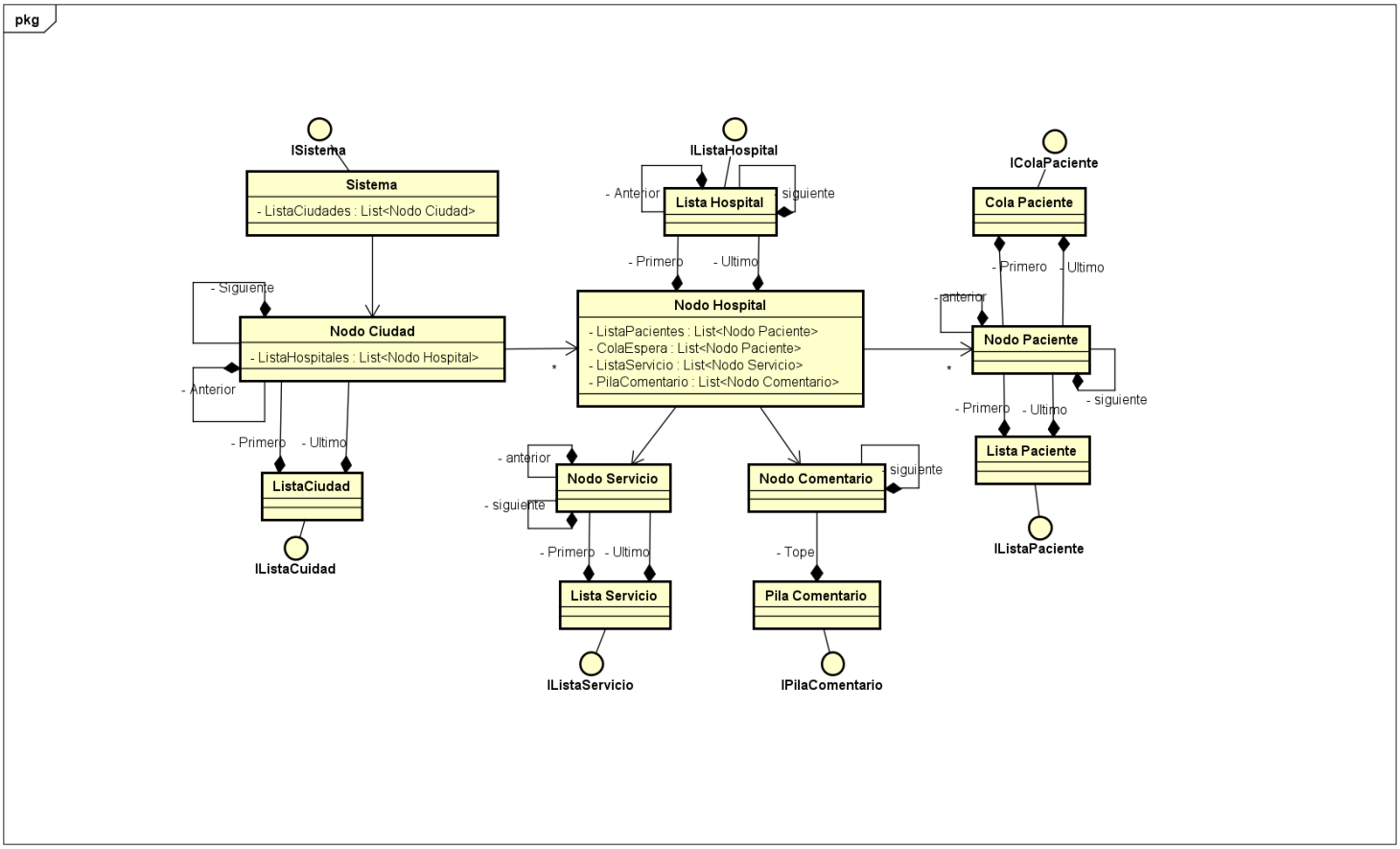
- `public Retorno ingresarComentario(String Ciudad, String Hospital, String Comentario, int Ranking);`

```
//Pre-condiciones: Recibe strings Ciudad, Hospital. Ambos deben ser
distintos de null. Ciudad debe existir en la listaCiudades y el
hospital debe existir en la lista hospitaled de la ciudad. Recibe un
int Paciente que debe ser único
//Post-condiciones: Ingresa un nuevo Paciente a las lista pacientes
del hospital pero en case que este llena se ingresará a la cola de
espera.
```

- public Retorno realizarIngreso(int Paciente, String Ciudad, String Hospital);
- public Retorno cancelarIngreso(int Paciente, String Ciudad, String Hospital);
- public Retorno trasladarPaciente(int Paciente, String CiudadOrigen,String HospitalOrigen,String CiudadDestino, String HospitalDestino);
- public Retorno listarServicios(String Ciudad, String Hospital);
- public Retorno listarHospitalesCiudad(String Ciudad);
- public Retorno listarHospitalesRanking();
- public Retorno listarComentarios(String Ciudad, String Hospital);
- public Retorno listarEspera(String Ciudad, String Hospital);
- public Retorno CargarDistancias(int[][] Ciudades);
- public Retorno BuscarCamino(int [][] M, String origen, String destino);

Solución escogida

Diagrama de la estructura de datos



Justificación (según corresponda)

CIUDAD	
Estructura	Lista Doble / Nodo con puntero Anterior y Siguiente

Implementación en Java	public interface IListaciudad {
	public boolean esVacia();
	public void agregarInicio(String dato);
	public void agregarFinal(String dato);
	public void agregarOrd(String n);
	public void borrarInicio();
	public void borrarFin();
	public void borrarElemento(String n);
	public boolean buscarelemento(String dato);
	public Nodociudad obtenerElemento(String n);
	public void vaciar();
	public void mostrar();
	public int cantElementos();
	}

Justificación	Permiten tanto recorrerlas en ambos sentidos, así como ingresar datos en ambas puntas.

HOSPITAL	
Estructura	Lista Doble / Nodo con puntero Anterior y Siguiente

Implementación en Java	public interface IListahospital {
	public boolean esVacia();
	public void agregarInicio(Nodohospital nuevoHospital);
	public void agregarFinal(String nombreHospital, int capacidad, int estrellas, int ranking);
	public void borrarInicio();
	public void borrarFin();
	public void borrarElemento(String nombreHospital);
	public boolean buscarelemento(String nombreHospital);
	public Nodohospital obtenerElemento(String nombreHospital);
	public void vaciar();
	public String mostrar();
	public int cantElementos();
	}

Justificación	Permiten tanto recorrerlas en ambos sentidos, así como ingresar datos en ambas puntas.

SERVICIO	
Estructura	Lista Doble / Nodo con puntero Anterior y Siguiente

Implementación en Java	public interface IListaDoble {
	public boolean esVacia();
	public void agregarInicio(String nombreServicio);
	public void borrarInicio();
	public void agregarFinal(String nombreServicio);
	public void borrarFin();
	public void vaciar();
	public String mostrar();
	public Nodoservicio buscarelemento(String nombreServicio);

	public void borrarelemento(String nombreServicio); }
Justificación	Permiten tanto recorrerlas en ambos sentidos, así como ingresar datos en ambas puntas.

COMENTARIO	
Estructura	Nodo con puntero único (siguiente) / Lista de tipo Pila

Implementación en Java	public interface IPilaComentarios {
	public void push(String comentario, int puntaje);
	public void pop();
	public boolean isEmpty();
	public int tamanioPila();
	public int topePila();
	}

Justificación	Elegimos utilizar una Lista del tipo Pila de manera de mostrar siempre los más actuales.

PACIENTE/Ingreso Hospital	
Estructura	Lista Doble / Nodo con puntero Anterior y Siguiente

Implementación en Java	public interface IListapaciente {
	public boolean esVacia();
	public void agregarInicio(Object dato);
	public void agregarFinal(Object dato);
	public void borrarInicio();
	public void borrarFin();
	public void borrarElemento(Object n);
	public boolean buscarelemento(Object dato);
	public Nodopaciente obtenerElemento(Object n);
	public void vaciar();
	public void mostrar();
	public int cantElementos();
	}

Justificación	Permiten tanto recorrerlas en ambos sentidos, así como ingresar datos en ambas puntas.

PACIENTE/Cola de Espera	
Estructura	Lista de tipo Cola / Nodo con puntero Anterior y Siguiente

Implementación en Java	public interface IColaFilaEspera {
	public void agregarCola (Nodopaciente paciente);
	public void sacarPrimero();
	public boolean esVacia();
	public Object alFrente();
	public int cantElementosCola();
	public void borrarElemento(Nodopaciente nroPaciente);
	public boolean buscarelemento(Nodopaciente nroPaciente);
	public Object obtenerElemento(Nodopaciente nroPaciente);
	}

Justificación	Preferimos utilizar una Cola ya que ingresan y salen en un único sentido.

Modificaciones Etapa Dos

nueva clase: NodoSalaHospital

nueva lista: ListaSalaHospital

nueva interface: IListaSalaHospital

```
@Override
 public NodoSala obtenerSala(int nroSala);

@Override
 public boolean salaAislada(int nroSala);

@Override
 public boolean tieneRespirador(int nroSala);
```