Parcial Metodologías de Desarrollo de Software I (8/6/2022)

El Ministerio de Salud de la Provincia de Buenos Aires quiere implementar un sistema para poder administrar la disponibilidad de camas en los hospitales de la provincia y poder entender donde se encuentra cada paciente.

En cada hospital de la provincia se designará a una responsable "local" del sistema que deberá cargar los pacientes que se internan en el hospital. Por cada paciente se deberá ingresar su DNI. El sistema entonces deberá interactuar con el SIHC (Sistema Integrado de Historias Clínicas) para obtener su historia clínica. La historia clínica está formada por los datos personales del paciente (nombre, apellido, dirección, y fecha de nacimiento) y por todos los registros de atenciones que haya recibido en los hospitales de la provincia. En caso de que el SIHC indique que el paciente no cuenta con historia clínica, el sistema deberá solicitar adicionalmente nombre, apellido, dirección, y fecha de nacimiento del paciente. Entonces, el sistema enviará estos datos al SIHC para que la cree. Para finalizar la carga de un paciente, la responsable local deberá ingresar día y hora de ingreso al hospital, la enfermedad que está cursando el paciente y el tipo de cama que ocupa (sala general, terapia intermedia o terapia intensiva).

El sistema deberá también permitirle a la responsable local cargar la capacidad del hospital. Esta se define como la cantidad de camas totales (no importa si están ocupadas). Por cada cama, la responsable deberá indicar su tipo (sala general, terapia intermedia o terapia intensiva). La responsable local deberá volver a cargar la cantidad de camas totales cada vez que se amplie su hospital o que agregue nuevas camas. Además, si al momento de cargar un paciente no se había registrado anteriormente la capacidad del hospital, el sistema le deberá solicitar que cargue este dato antes de continuar con la carga del paciente.

A nivel provincial habrá una responsable de IT. El sistema deberá permitir a la responsable de IT agregar hospitales al sistema. Por cada hospital se le solicitará nombre, zona sanitaria, ciudad y dirección. La responsable de IT también deberá crear los usuarios para las responsables locales de los hospitales. Al crear un responsable local se le solicitará nombre, apellido, nombre de usuario, contraseña, casilla de e-mail, y seleccionar el hospital del cual será responsable. Además, el sistema deberá enviar un email a la casilla de e-mail suministrada informando de sus datos de acceso.

Finalmente, el sistema deberá permitir al Ministerio de Salud acceder a un mapa de la provincia donde se muestre un mapa de calor (usando tonalidades rojo a verde) indicando la disponibilidad o no de camas en cada uno de los hospitales. Para analizar con mejor detalle esta ocupación, el mapa deberá permitir filtrar por tipo de cama y/o enfermedad. También, todos los días lunes a las 8 de la mañana deberá generarse un reporte de cantidad de camas ocupadas y disponibles en cada hospital discretizado por el tipo de cama. Este reporte deberá estar ordenado por ciudad (orden alfabético ascendente). El reporte se deberá guardar en el Google Drive del Ministerio y podrá ser accedido por un link que deberá encontrarse en el mapa de calor.

Ejercicio 1

a) Realizar el diagrama de casos de uso.

b) Realizar la especificación del/los caso/s de uso relacionado/s con la funcionalidad de cargar un paciente

Ejercicio 2

Responda a las siguientes preguntas (no es necesario desarrollar)

a) ¿Cuáles son las etapas del ciclo de vida en un proceso de desarrollo de software?
b) ¿Cuáles son los eventos de Scrum?
c) ¿En que evento de Scrum se realiza el diseño del sistema?
d) ¿En que evento de Scrum se realiza el testing del sistema?
e) ¿Qué técnica propone Scrum para especificar los PBIs?
f) Cuando quiero modelar la interacción entre objetos ¿qué diagrama UML debo utilizar?

Ejercicio 3

Se acaba de crear un nuevo archivo "Parcial.java". Escriba los comandos necesarios para i) agregar el archivo a un repo git, ii) persistir los cambios en el repositorio local, iii) enviar los cambios al repositorio remoto.

Ejercicio 4

Dado el siguiente código fuente parcial de la aplicación descrita en el Ejercicio 1 realice el diagrama de clases correspondiente. Solo tenga en cuenta las clases, métodos y atributos indicados en el código fuente. No es necesario incluir los nombres de parámetros. Incluya adornos cuando corresponda. No incluya relaciones que no puedan ser deducidas del código fuente *o de la narrativa del Ejercicio 1.*

```
public class RegistroAtencion {
  protected Date fechaAtencion;
  private String descripcion;

  public RegistroAtencion(Date
      fechaAtencion, String descripcion)
{
    this.fechaAtencion = fechaAtencion;
    this.descripcion = descripcion;
  }
  public Date getFechaAtencion() {
    return fechaAtencion;
  }
  public String getDescripcion() {
    return descripcion;
  }
}
```

```
public class AtencionAmbulatoria extends
                              RegistroAtencion
{
  private String dependencia;

  public AtencionAmbulatoria(Date
fechaAtencion, String descripcion, String
dependencia) {
    super(fechaAtencion, descripcion);
    this.dependencia = dependencia;
  }

  public String getDependencia() {
    return dependencia;
  }
}
```

```
public class Internacion extends
RegistroAtencion {
  private Date fechaAlta;
  private String responsableInternacion;


  public Internacion(Date fechaAtencion,
String descripcion, String
responsableInternacion) {
    super(fechaAtencion, descripcion);
    this.responsableInternacion =

responsableInternacion;
    this.fechaAlta=null;
  }
  public void setFechaAlta(Date fechaAlta)
  {
    this.fechaAlta = fechaAlta;
  }
  public long calcularDiasInternacion() {
    if(fechaAlta!=null)
      return fechaAlta.getTime()-
      fechaAtencion.getTime();
    return -1;
  }
}
```

```
public class DatosPaciente {
  private String nombre;
  private String apellido;
  private String direccion;
  private Date fechaDeNacimiento;

  public DatosPaciente(String nombre,
String apellido, String direccion, Date
fechaDeNacimiento) {
    this.nombre = nombre;
    this.apellido = apellido;
    this.direccion = direccion;
    this.fechaDeNacimiento =

fechaDeNacimiento;
  }
  public String getNombre(){return
nombre;}
  public String getApellido() {
    return apellido;
  }
  public String getDireccion() {
    return direccion;
  }
  public Date getFechaDeNacimiento() {
    return fechaDeNacimiento;
  }
}
```

```
public class HistoriaClinica {
  private DatosPaciente datosPaciente;
  private List<RegistroAtencion>

atenciones;

  public HistoriaClinica(DatosPaciente
datosPaciente, RegistroAtencion
primeraAtencion) {
    this.datosPaciente=datosPaciente;
    atenciones=new

ArrayList<RegistroAtencion>();
    atenciones.add(primeraAtencion);
  }

  public void addAtencion(RegistroAtencion
atencion) {
    atenciones.add(atencion);
  }
}
```

```
public class Cama {
  public static int GENERAL=1;
  public static int TERAPIA_INTERMEDIA=2;
  public static int TERAPIA_INTENSIVA=3;
  private int tipoCama;
  private boolean disponible;
  private DatosPaciente ocupanteActual;

  public Cama(int tipoCama) {
    this.tipoCama = tipoCama;
    disponible=true;
    ocupanteActual=null;
  }
  public boolean isDisponible() {
    return disponible;
  }
  public int getTipoCama() {
    return tipoCama;
  }
  public void setOcupanteActual(
            DatosPaciente ocupanteActual)
{
    this.ocupanteActual = ocupanteActual;
  }
  public DatosPaciente
getOcupanteActual(){
    return ocupanteActual;
  }
}
```

```
public class AdministradorDeCamas {
	private List<Cama> camas;

	public AdministradorDeCamas() {
		camas=new ArrayList<Cama>();
	}

	public void cargarNuevaCama(int tipoCama) {
		camas.add(new Cama(tipoCama));
	}

	public int camasDisponibles() {
		int acum=0;
		for (Cama cama : camas) {
			if(cama.isDisponible())
				acum++;
		}
		return acum;
	}

	public int capacidadCamasUTI() {
		int acum=0;
		for (Cama cama : camas) {
			if(cama.getTipoCama()==Cama.TERAPIA_INTENSIVA)
				acum++;
		}
		return acum;
	}
	public List<DatosPaciente> listarOcupantesCamas(){
		List<DatosPaciente> ret=new ArrayList<DatosPaciente>();
		for (Cama cama : camas) {
			if(!cama.isDisponible())
				ret.add(cama.getOcupanteActual());
		}
		return ret;
	}
}
```