Desarrollador Java intermedio

Trabajo Práctico Integrador

Sistema de Reporte de Incidentes

**Contexto general**

Una importante empresa de soporte operativo solicita el diseño y desarrollo de un sistema que le permita la generación y seguimiento de los incidentes que se presentan.

La empresa en cuestión se dedica a brindar soporte operativo sobre distintas aplicaciones (SAP, Tango, etc.) y sistemas operativos (Windows, MacOS, Linux Ubuntu).

------------

El área de RRHH se encarga de realizar las altas, bajas y modificaciones de los técnicos que se encargan de resolver los incidentes reportados.

Cada técnico tiene una o varias especialidades y solo se le pueden asignar incidentes que coincidan con las mismas.

Se crean la clase:

**Técnico, Especialidad**

Un técnico puede tener una o varias especialidades

Una especialidad puede ser atributo de uno o varios técnicos.

Se necesitan todos los métodos CRUD

El área comercial es responsable de incorporar nuevos clientes a la empresa. Administra las altas, bajas y modificaciones de los datos de cada uno de ellos.

Se crean las Clases:

**Cliente, Servicio, Contrato, Tipo de problema, Detalle de servicio,**

Un cliente puede tener cero o varios contratos.

Un Contrato tiene un solo cliente y uno varios servicios.

Un Contrato tiene fecha inicio, fecha de fin y estado (pendiente, activo, suspendido, finalizado)

Un Servicio puede tener uno o varios tipos de problemas, configurando el detalle del servicio.

Cada detalle de servicio (Servicio + Tipo de problema) tiene una duración estándar.

Se necesitan todos los métodos CRUD

Finalmente, la mesa de ayuda es responsable de atender las llamadas e ingresar al sistema los incidentes reportados.

Se crean la clase:

**Incidentes, problema**

- Un incidente puede tener uno o varios problemas

**Ciclo de vida de un incidente**

Cuando un cliente llama, la mesa de ayuda le solicita los datos para identificarlo (razón social, CUIT) y los ingresa en el sistema para que el mismo le muestre los servicios que el cliente tiene contratados.

El operador (de la mesa de ayuda) solicita que le informen por cuál de esos servicios desea reportar un incidente, junto con una descripción del problema y el tipo del problema.

**Cliente**

* Método para que devuelva cliente (busque cliente por CUIT)
* Método para que devuelva cliente (busque cliente por RAZON SOCIAL.)
* Método para que devuelva los contratos, y seleccione UNO
* public static Cliente buscarPorCuit(List<Cliente> clientes, String cuit)
* public static Cliente buscarPorRazonSocial(List<Cliente> clientes, String razonSocial)
* public List<Contrato> obtenerContratosDelCliente(Cliente cliente)
* public Contrato seleccionarContrato(int idContrato)

**Contrato**

* Método para que devuelva el SERVICIO del contrato
* public Servicio obtenerServicioDelContrato(Contrato contrato)

**Servicio**

* Método para que devuelva la lista de detalle de servicio y seleccione uno
* public List<Detalle\_servicio> obtenerDetallesServicio(Servicio servicio)
* public Detalle\_servicio seleccionarDetalleServicioPorId(int idDetalleServicio)

**Detalle del servicio**

* Método para la duración estándar del detalle de servicio selecciónado

**Incidentes**

* Método que DEVUELVE la fecha de creación, se activa automática cuando se crea el incidente, devuelve la fecha actual.

Se puso

this.fecha\_creacion = new Date(); en los contructores

* Método que DEVUELVE la fecha de inicio, se usa cuando se selecciona el TECNICO por el operador. DEVUELVE: Si el técnico seleccionado no tiene Incidentes asignados es igual a la fecha de creación, SINO igual a la "fecha\_fin\_estimada" del último incidente del técnico seleccionado, más un dia.
* public void determinarFechaInicio(int idTecnico, List<Incidente> incidentesAsignadosAlTecnico)
* private Incidente obtenerUltimoIncidenteDelTecnico(List<Incidente> incidentesAsignadosAlTecnico)
* Método que CALCULA la duracion\_estimada, se invoca cuando se termina de cargar todos los problemas del incidente. Sería la sumatoria total de todos los tiempos de cada problema.
* public Duration calcularDuracionEstimada()
* Método que CALCULA el estado, devuelve pendiente, hasta que comience la fecha de inicio, en proceso, cuando comienza la fecha de inicio, demorado si la fecha\_fin\_estimada es la fecha actual y no esta cerrado por el técnico. y "resuelto" cuando el técnico lo marca como "finalizado" el último de los problemas.
* public String calcularEstado()
* Método que DEVUELVE la fecha de cierre del Incidente si el mismo ha sido **resuelto**, es igual a la fecha del ultimo problema del incidente.
* public Date obtenerFechaCierre()
* Método que DEVUELVE la fecha de fin estimada fecha de inicio + duración.
* public Date obtenerFechaFinEstimada()

**Problema**

Se necesitan todos los métodos CRUD

Al ingresar el incidente, el sistema devuelve un listado de técnicos disponibles para resolver el problema. El operador selecciona uno de los técnicos disponibles y el sistema le informa el tiempo estimado de resolución.

**Detalle de servicio**

* Método que DEVUELVE la lista de especialidades que puede resolver el detalle de servicio selecciónado.
* public List<Especialidad> getEspecialidades()

**Especialidad**

* cual seria el Método que DEVUELVE la lista de técnicos que puede resolver la lista de especialidades seleccionada.
* public static List<Tecnico> obtenerTecnicosPorEspecialidades(List<Especialidad> especialidadesSeleccionadas, List<Tecnico> listaDeTecnicos)

Luego, informa al cliente que el incidente ha sido ingresado y

la fecha posible de resolución.

**Incidentes**

* Método de notificación al cliente: realiza el envió del mail al cliente con la fecha posible resolución, si el método es activado cuando se crea el incidente.
* public void notificarClienteFechaPosibleSolucion(Incidente incidente)
* private Date obtenerFechaPosibleResolucion(Incidente incidente) Al confirmarse el incidente, el sistema debe enviar una notificación al técnico informándole

que tiene un nuevo incidente para resolver.

**Incidentes**

* Método de notificación: según el medio de notificación seleccionado por el técnico asignado éste método realiza la notificación.
* public void notificarTecnico(Incidente incidente)

1

Desarrollador Java intermedio

Cuando el técnico atiende y resuelve el incidente, lo debe marcar como “resuelto”, indicando las consideraciones que crea necesarias. Cuando esto ocurra, el sistema debe enviar un email al cliente informándole que su incidente ya está solucionado.

**Incidentes**

* Método de notificación al cliente: realiza el envió del mail al cliente con la fecha REAL resolución y el estado de resuelto, si el método es activado cuando se cierra el incidente.
* public void notificarClienteEstadoResulto(Incidente incidente)

**Otros requerimientos**

● El sistema debe permitir al área de RRHH emitir diariamente reportes con los incidentes asignados a cada técnico y el estado de los mismos.

**Incidentes**

* Método ReporteRRHH-Estado\_Incidente\_Tecnico., devuelve todos los incidentes de una fecha particular (danto de ingreso fecha)
* public static List<Incidente> obtenerIncidentesPorFechaEnLista(Date fecha, List<Incidente> listaIncidentes)

● El sistema debe permitir que el operador agregue “un colchón” de horas estimadas para la resolución del problema, si el mismo es considerado “complejo”.

**Problema**

* Método CalculaColchonDuracionEstimada. devuele la suma del tiempo estimda de un tipo de problema que se este tratando y agrega el colchón que le operador asigna en caso de considerarlo completo.
* public Duration CalculaColchonDuracionEstimada(Duration colchon)

● El sistema debe permitir el alta de incidentes que contengan un conjunto de problemas de un mismo servicio. Dichos problemas deben estar relacionados.

● El sistema debe dar la posibilidad de informar:

○ Quién fue el técnico con más incidentes resueltos en los últimos N días

**Incidentes**

* Método que devuelve el técnico con más incidentes resueltos en los últimos N días
* public Tecnico obtenerTecnicoMasIncidentesResueltos(int dias) FALTA CODIGO

○ Quién fue el técnico con más incidentes resueltos de una determinada

especialidad en los últimos N días

**Incidentes**

* Método que devuelve el técnico con más incidentes resueltos de una determinada

especialidad en los últimos N días

* public Tecnico obtenerTecnicoMasIncidentesResueltosEspecialidad(int dias, int idEspecialidad) FALTA CODIGO

○ Quién fue el técnico que más rápido resolvió los incidentes

* Método que devuelve fue el técnico que más rápido resolvió los incidentes

**Consideraciones**

● Cada tipo de problema particular puede ser solucionado por una o varias

especialidades.

● Cada operador puede definir, optativamente, su tiempo estimado de resolución por defecto por tipo de problema; el cual tendrá que ser menor al tiempo máximo de resolución definido para el tipo de problema.

**Problema**

* Método CalculaNuevaDuracionDelProblema, si el problema no es complejo, el operador puede cambiar la duración siempre y cuando esta supere la duracion éstandar.

HECHO

● Cada técnico puede definir su medio preferido de notificación, los cuales pueden ser:

Email o WhatsApp. No están definidas las bibliotecas que se utilizarán para realizar

estas notificaciones.

**Técnico**

* Método para DefinirNotificacion, devuele la selección del técnico

**Metodología**

Se propone una metodología de trabajo **iterativa** e **incremental**. Para esto, el TP se divide en tres entregas, las cuales se realizarán a través del envío del link al repositorio de trabajo por medio del Campus Virtual en la fecha estipulada por el Docente del curso.

En la última entrega (coincidente con la tercera) se deberá exponer el TP frente al curso, mostrando la solución generada y justificando las decisiones tomadas.

2

Desarrollador Java intermedio

Entregas

**Entrega 1**

En esta primera iteración nos encargaremos de modelar, a nivel datos y objetos, una solución

al dominio presentado. Además, comenzaremos con el proceso de codificación de la solución.

En particular, en esta entrega se solicita:

1. Modelo de datos (DER físico) que brinde solución al dominio.

2. Código con modelado de clases. El código debe estar subido a un repositorio de

GitHub.

Es necesario que el proyecto Java sea creado como un proyecto “Maven” para poder añadir algunas dependencias. Se recomienda la utilización de la dependencia de **Lombok** para

facilitar la tarea repetitiva de generación de Setters y Getters de las clases.

**Entrega 2**

En esta segunda iteración nos encargaremos de mapear, mediante anotaciones JPA, nuestras

clases implementadas para poder persistir el modelo en una Base de Datos Relacional.

Además, comenzaremos con la generación de los repositorios/servicios para que brinden

solución a algunos requerimientos planteados.

En particular, en esta entrega se solicita:

1. Modelado de clases con Mapeo (anotaciones JPA) de entidades para que las mismas

sean persistidas mediante el ORM Hibernate.

2. Repositorios/Servicios que den solución a los requerimientos planteados:

a. Quién fue el técnico con más incidentes resueltos en los últimos N días

b. Quién fue el técnico con más incidentes resueltos de una determinada

especialidad en los últimos N días

c. Quién fue el técnico que más rápido resolvió los incidentes

3