# Proyecto Final

## Objetivos

- Realizar un diseño lógico de datos a partir de requerimientos de datos.
- Implementar un diseño lógico utilizando SQL como lenguaje de definición de datos.
- Documentar una implementación.
- Insertar, actualizar y borrar registros en una base de datos.
- Recuperar información solicitada de una base de datos
- Crear funciones para obtener información solicitada sobre una base de datos.
- Crear procedimientos almacenados para realizar cambios sobre una base de datos.
- Crear triggers para responder ante eventos sobre una base de datos.
- Ejecutar operaciones en la base de datos desde el lenguaje Java usando JDBC
- Crear ejecuciones programadas para realizar tareas automatizadas sobre una base de datos.

## Enunciado

Una empresa de telefonía móvil ha decidido implementar un sistema para la gestión de solicitudes, daños y reclamos. Se deben manejar los siguientes procesos

## Registro

Un cliente puede registrar una **solicitud** de **creación** o **modificación** del producto, que puede ser de servicio de Voz, servicio de Datos o servicio Integrado (Voz y Datos). Para solicitar el producto, el cliente debe estar registrado en la base de datos previamente. Para registrar la solicitud de creación, el sistema requerirá el tipo de producto que desea (Voz, Datos, Integrado) y la cédula del cliente. También se permitirá ingresar una observación para suministrar información adicional. El cliente también puede solicitar la modificación de un producto (cambiarlo del servicio actual por Voz y Datos, por ejemplo). Adicionalmente un cliente también puede solicitar la **cancelación** (desconexión) de un producto, para lo cual debe suministrar una causa de cancelación en la observación.

Un cliente puede reportar la solicitud indicando que tiene un **daño**, si necesita denunciar alguna anomalía relacionada con el servicio de la empresa (por ejemplo, baja velocidad, mala calidad del servicio, etc.). El sistema solicitará la cédula del cliente, el producto sobre el cual desea registrar el daño, el tipo de anomalía (que se podrá seleccionar de una lista) y la descripción del problema.

Si un cliente está inconforme con la facturación de sus servicios, puede registrar una solicitud de tipo **reclamo** indicando que ha habido un error en el cobro. El sistema solicitará la cédula del cliente, el producto sobre el cual desea registrar el reclamo y una descripción del problema.

## Asignación

Una vez registrada una solicitud (solicitud, daño o reclamo), el sistema deberá asignarla automáticamente (con un **trigger**) a un funcionario para que la evalúe y tome una decisión al respecto. Para esto, el sistema debe tener un algoritmo que balancea la carga de los funcionarios. Por ejemplo, si un funcionario tiene más de tres (3) solicitudes acumuladas, el sistema no debería asignarle más solicitudes hasta que no libere carga. La cantidad máxima de solicitudes que puede tener un funcionario es un parámetro configurable en el sistema (ver Consideraciones adicionales). En este caso, la solicitud queda en estado "pendiente". En el momento de asignar, se debe guardar el funcionario asignado, la fecha de asignación y cambiar el estado a "asignada". Si ocurre un error durante la asignación de una solicitud, ésta quedará en estado pendiente y podrá ser asignada después de manera individual o masiva.
La asignación individual funciona de manera similar a la anterior, con la diferencia de que se puede forzar para que la solicitud quede asignada a un funcionario específico. Debe indicarse el código de la solicitud y el código del funcionario a asignar. El funcionario debe existir. La solicitud debe existir y estar en estado pendiente. De no ser así se mostrará un mensaje de error aclarando la situación y la operación no será realizada.
Para la asignación masiva, se deberá definir un **proceso programado** que evalúe las solicitudes (incluyendo daños y reclamos) que tengan más de cierto tiempo en estado Pendiente para tratar de asignarlas. Este tiempo será un valor parametrizado en la base de datos (ver consideraciones adicionales). Si ocurre un error durante la atención automática de una solicitud, ésta se dejará en estado pendiente y se procesará la siguiente solicitud.

**Atención**

Cada funcionario atiende de forma individual las solicitudes, daños o reclamos que tenga asignadas. Para esto, el proceso de atención siempre solicitará la cédula del funcionario y el código de la solicitud, validando que ésta esté asignada al funcionario. Si el funcionario no tiene asignada la solicitud, se mostrará un error informando la situación. Si se trata de una **solicitud** de nuevo producto, al atenderla se creará un registro en la relación de clientesXproductos, indicando la fecha de inicio del servicio (1er día del siguiente mes). Luego la solicitud debe quedar atendida, con fecha de atención (sysdate) y comentarios del funcionario. Si ocurre algún error durante el proceso, se deben deshacer los cambios de la transacción y la solicitud continuará asignada al funcionario.
Si se trata de un **retiro** de producto, el sistema deberá registrar la fecha de retiro (sysdate) en la relación de clientesXproductos y dejar la solicitud en estado atendida, guardando los comentarios del funcionario. Si ocurre algún error durante el proceso, se deben deshacer los cambios de la transacción.

Para los daños y reclamos, el funcionario deberá revisar el caso y conforme a su juicio y experiencia deberá aceptar (pasarla a estado atendido) o rechazar (estado anulado) la solicitud. En cualquier caso, se debe registrar un comentario. Si es rechazada, en el comentario se indicará la causa de rechazo.

Se deberá definir un **proceso programado** que evalúe las solicitudes (solamente daños y reclamos) que tengan más de cierto tiempo en estado Asignado para atenderlas automáticamente a favor del cliente. Este tiempo será un valor parametrizado en la base de datos (ver consideraciones adicionales). El comentario de atención será "Atendida

automáticamente por el sistema". Si ocurre un error durante la atención automática de una solicitud, ésta se dejará en estado Asignado y se procesará la siguiente solicitud.

## Consideraciones adicionales

***Parámetros del sistema.***

Usted debe crear una tabla de parámetros para configurar constantes o parámetros que puedan ser cambiados en tiempo de ejecución sin necesidad de modificar el código de la aplicación. Dicha tabla debe, como mínimo, tener un código del parámetro, un nombre y un valor.

***Agrupamiento de funciones y procedimientos en Niveles.***

Todas las funciones y procedimientos requeridos para la implementación del sistema serán agrupados en paquetes. Habrá tres niveles de paquetes:
Nivel 1: Paquetes para operaciones básicas sobre tablas. Deberá haber un paquete por cada tabla, con al menos cuatro funciones/procedimientos: insertar, borrar, modificar y consultar por id. Los nombres y modos de operación de estas funciones deben ser estándares (pInsertar, pBorrar, pModificar, fConsultar) para que sean utilizadas fácilmente por desarrolladores que no conozcan la implementación. Estas funciones solamente serán utilizadas por funciones o procedimientos de Nivel2.
**Ejemplo**: para la tabla clientes, se creará un paquete denominado pkClientes, el cual contará al menos con las funciones/procedimientos pInsertar, pBorrar, pModificar, fConsultar.

Nivel 2: Paquetes con lógica del negocio. Estos paquetes tendrán funciones y procedimientos para resolver las necesidades de cada funcionalidad indicada (registro, asignación, atención), utilizando subprogramas del Nivel 1. Debe existir un paquete por cada funcionalidad, así: pkRegistroNivel2, pkAsignacionNivel2, pkAtencionNivel2.

Nivel 3: A este nivel pertenecen los procesos programados y los elementos de la interfaz (pantallas o scripts), los cuales solamente podrán ejecutar funciones/procedimientos contenidos en paquetes de Nivel 2. Fuera de

***Manejo de Errores***

El sistema debe contar con un manejo robusto de los errores, que permita mostrar al usuario en todo momento mensajes de error en términos que pueda comprender. Por ejemplo: si un registro ya existe en una tabla, el mensaje debería ser: "El registro [id] ya existe en la tabla [nombre_tabla]" y no "ORA-00001: unique constraint (string.string) violated". Si la excepción ocurre en un paquete de Nivel 1 o Nivel 2, se debe elevar al nivel inmediatamente superior. Si la excepción ocurre en un proceso de Nivel 3, que es ejecutado directamente por un usuario, el error se debe imprimir por pantalla.

***Interfaz de Usuario***

- Se debe proporcionar un mecanismo (GUI o scripts) para el mantenimiento (insertar, actualizar, eliminar) de la siguiente información básica.
    - Clientes (nombre, cedula, fecha nacimiento, dirección y teléfono)
    - Funcionarios de la empresa (nombre, cédula, fecha nacimiento, dirección y teléfono)
    - Tipos de Producto (código, descripción): Inicialmente la empresa sólo provee **Telefonía** e **Internet**, pero en un futuro puede necesitar agregar nuevos tipos o modificar los existentes.
    - Tipos de solicitud (código, descripción): Pueden ser **solicitud**, **daño** o **reclamo**.
    - Cualquier otra entidad básica que se considere que requiere mantenimiento.
- Se debe proporcionar un mecanismo (Interfaz en Java) para permitir al usuario realizar el **registro** y la **atención** de solicitudes. La asignación es automática, por tanto no requiere de interfaz.
- Se debe proporcionar un mecanismo (Interfaz en Java) para consultar información de las diferentes entidades para verificar en cualquier momento la eficacia de las operaciones realizadas.
    - Consulta de Solicitudes asignadas X Funcionario
    - Consulta de Solicitudes X Estado
    - Consulta de Solicitudes X Tipo
    - Consulta de Productos X Cliente

**Notas**:

- El modelo relacional debe ser construido y acordado entre todos.

- Los objetos deben respetar el siguiente estándar para la nomenclatura (si aparecen otros tipos de objetos, el estándar será acordado con todo el grupo)

  funciones: fNombreFuncion
  procedimientos: pNombreProcedimiento
  variables: vNombreVariable
  parámetro entrada: ivNombreVariable
  parámetro salida: ovNombreVariable
  cursores: cuNombreCursor
  paquetes: pkNombrePaquete
  constantes: cNOMBRE_CONSTANTE

- Los scripts deben tener el mismo nombre del objeto principal contenido en ellos. Por ejemplo, si un paquete de llama pkCliente, debe estar contenido en un script denominado pkCliente.sql. Si contiene varios scripts, debe tener un nombre significativo de acuerdo a su contenido.

- No se aceptan funciones o procedimientos fuera de un paquete, a menos que exista una justificación válida.

**Rúbrica de Grupo:**

Cada uno de los siguientes aspectos será calificado así:
5: Cumple totalmente sin observaciones
4: Cumple totalmente y tiene observaciones menores
3: Cumple parcialmente y/o tiene observaciones importantes
2: Se implementó, pero no cumple la mayoría de los aspectos solicitados
1: Se implementó, pero no cumple lo solicitado
0: No se implementó o no se entregó

| Aspecto Evaluado | % |
|---|---|
| **General** | |
| Documentación del código (encabezados de procedimientos y funciones, explicando qué hace el método, entradas y resultado) | 5% |
| Claridad del código (sangría, legibilidad) | 5% |
| Manejo de errores acorde con cada nivel de paquete. | 10% |
| Funcionamiento correcto de las interfaces para crear/modificar información en el sistema (operación) | 5% |
| Funcionamiento correcto de las interfaces para extraer información del sistema (visualización) | 5% |
| El ciclo de vida de una solicitud funciona conforme a los requerimientos funcionales | 10% |
| **Diseño Lógico** | |
| El diseño lógico es completo y coherente con las especificaciones del problema. | 10% |
| **SQL** | |
| Los scripts DDL para implementar diseño lógico se ejecutan correctamente. Se pueden ejecutar para instalar la aplicación en cualquier computador. | 5% |
| Los scripts DML para crear/modificar datos se ejecutan correctamente | 5% |
| **PL/SQL** | |
| Se utilizan paquetes para agrupar funciones y procedimientos con un objetivo común. No hay funciones o procedimientos fuera de paquetes | 5% |
| Las funciones tienen un nombre significativo y son coherentes con su objetivo | 10% |
| Los procedimientos tienen un nombre significativo y son coherentes con su objetivo | 10% |
| Los procedimientos y funciones cumplen con el estándar en el nombramiento de variables, constantes y parámetros de entrada/salida. | 5% |
| Los jobs tienen un nombre significativo, cumplen con el estándar y son coherentes con su objetivo. | 5% |

| | |
|---|---|
| Los triggers tienen un nombre significativo, cumplen con el estándar y son coherentes con su objetivo. | 5% |
| **TOTAL** | **100%** |

**Rúbrica Individual:**

Cada uno de los siguientes aspectos será calificado así:
5: Conoce el aspecto evaluado
3: Muestra algún desconocimiento del aspecto
0: No sustentó o muestra total desconocimiento del aspecto

| **Aspecto Evaluado** | **%** |
|---|---|
| Explicación del modelo E/R | 15% |
| Explicación de los objetos que implementan el E/R | 15% |
| Descripción general de los objetos que componen el sistema (estructura y función) | 15% |
| Explicación del proceso de Registro/Asignación/Atención (secuencia y objetos involucrados) | 25% |
| Explicación del funcionamiento de los Jobs | 15% |
| Explicación del funcionamiento de los triggers. | 15% |
| **TOTAL INDIVIDUAL** | **100%** |

Fecha y hora de entrega: viernes de la semana 16 hasta las 23:55

Fecha y hora de presentación: lunes de la semana 17 en el horario de clase

Calificación:

| | |
|---|---|
| Análisis y código (nota grupal) | 40% |
| Sustentación (nota individual) | 60% |
| **Nota proyecto final** | 100% |