Sistema de Venta de Flores

INDICE

REQUERIMIENTOS 3

Requerimientos Funcionales 3

Requerimientos no Funcionales 3

CASOS DE USO 4

Actores 4

Casos de uso 4

PROCESO DE LOGUEO 5

Caso de Uso 5

Especificación 5

Prototipo 6

Diagrama de Secuencia 6

Diagrama E-R 6

El Servicio 7

# REQUERIMIENTOS

## Requerimientos Funcionales

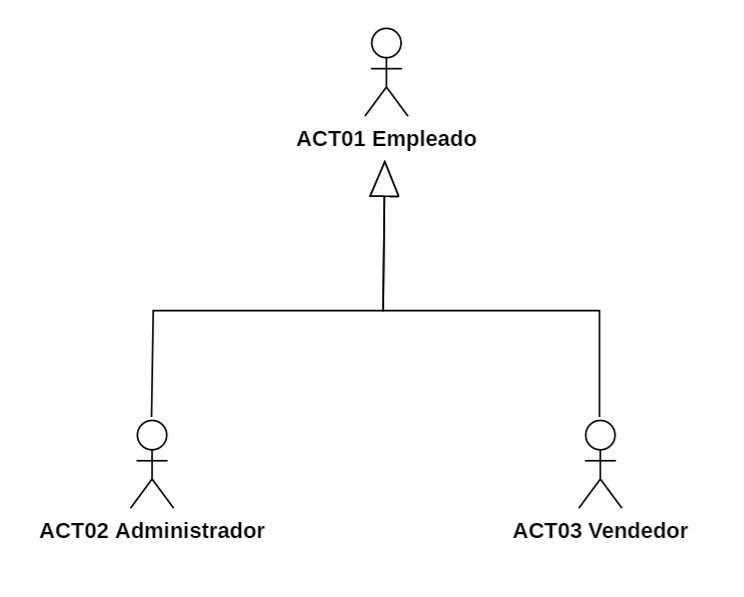
1. El acceso al sistema será solo de personal autorizado mediante usuario y clave. Cada usuario debe tener su rol o perfil.
2. Mantenimiento de clientes.
3. Mantenimiento de productos.
4. Gestión de ventas.
5. Reportes y consultas de ventas por diferentes indicadores.

## Requerimientos no Funcionales

1. Los permisos de acceso al sistema podrán ser cambiados solamente por el administrador de acceso a datos.
2. Si se identifican ataques de seguridad o brecha del sistema, el mismo no continuará operando hasta ser desbloqueado por un administrador de seguridad.
3. Desarrollado en Lenguaje Java con base de datos SQL Server.

# CASOS DE USO

## Actores



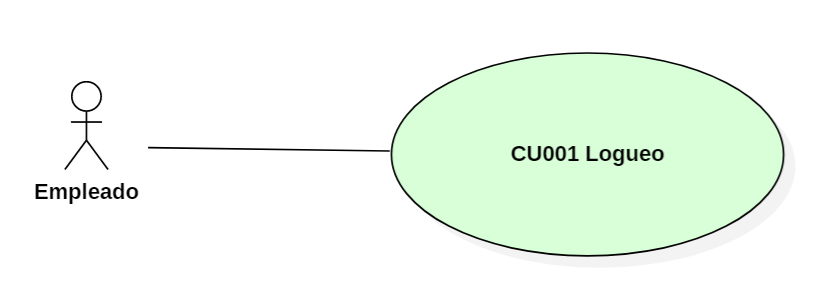
|  |  |  |
| --- | --- | --- |
| **CODIGO** | **NOMBRE** | **DESCRIPCION** |
| ACT01 | Empleado | Representa a todos los empleados que serán usuarios del sistema. |
| ACT02 | Administrador | Representa a la persona que realiza tareas administrativas en el sistema. |
| ACT03 | Vendedor | Representa a todos los vendedores. |

## Casos de uso

|  |  |  |
| --- | --- | --- |
| **CODIGO** | **NOMBRE** | **DESCRIPCION** |
| CU001 | Logueo | Representa el proceso de inicio de sesión de los usuarios de la aplicación. |
| CU002 |  |  |
| CU003 |  |  |

# PROCESO DE LOGUEO

## Caso de Uso



## Especificación

|  |  |
| --- | --- |
| **Identificador** | **CU0001** |
| **Descripción** | Representa el proceso de inicio de sesión de los usuarios de la aplicación. Es necesario que exista el usuario en la base de datos y se encuentre habilitado. |
| **Actor** | ACT01 |
| **Flujo Básico** | Para poder iniciar este caso de uso, el empleado debe tener su usuario y clave a la mano.  El proceso a ejecutar es el siguiente:   1. El empleado ejecuta la aplicación. 2. La aplicación muestra la ventana de inicio de sesión o logueo. 3. El empleado ingresa su nombre de usuario. 4. El empleado ingresa su clave. 5. El usuario ejecuta el formulario. 6. Si los datos no son correctos, muestra el mensaje **"Datos incorrectos"** y regresa al punto 4. 7. Si los datos si son correctos, la aplicación muestra el formulario principal. |
| **Flujos Alternativos** | No aplica |
| **Escenarios Claves** | Si el empleado no cuenta con usuario y clave, debe informar y solicitar sus credenciales al área respectiva. |
| **Pre Condiciones** | El empleado debe tener su usuario y clave, y además deben estar activos. |
| **Post Condiciones** | Después de iniciar sesión correctamente, el empleado puede ejecutar las tareas que le corresponden según su perfil. |
| **Puntos de extensión** | No aplica. |
| **Requerimientos especiales** | La aplicación debe estar funcionando correctamente. |
| **Información adicional** | La aplicación solo puede ejecutarse desde la red interna de la empresa. |
| **Requerimientos Relacionados** | No aplica. |

## Prototipo



## Diagrama de Secuencia



## Diagrama E-R



## El Servicio

### Diseño



En la clase **LogonService** se implementa el servicio validar, este servicio recibe como datos de entrada el **usuario** y la **clave**, si existen en la base de datos retorna un objeto de tipo **UsuarioDto** con los datos del usuario.

En caso de error, este se maneja a través de las propiedades **code** y **message** de la clase **LogonSevice**.

### Implementación

public UsuarioDto validar(String usuario, String clave) {

UsuarioDto usuDto = null;

String query = "select u.IDEMPLEADO, u.USUARIO, '\*\*\*\*\*' CLAVE, "

+ "u.IDROL, r.NOMBRE, u.ACTIVO "

+ "from dbo.USUARIO u "

+ "join dbo.EMPLEADO e on u.IDEMPLEADO = e.IDEMPLEADO "

+ "join dbo.ROL r on u.IDROL = r.IDROL "

+ "where u.USUARIO = ? and u.CLAVE = ? "

+ "and u.ACTIVO = 1 and e.ACTIVO = 1";

PreparedStatement pstm = null;

ResultSet rs = null;

Connection cn = null;

this.setCode(1);

this.setMessage("Proceso ok!!!");

try {

cn = AccesoDB.getConnection();

pstm = cn.prepareStatement(query);

pstm.setString(1, usuario);

pstm.setString(2, clave);

rs = pstm.executeQuery();

if(!rs.next()){

rs.close();

pstm.close();

throw new SQLException("Datos incorrectos.");

}

usuDto = new UsuarioDto();

usuDto.setIdempleado(rs.getInt("IDEMPLEADO"));

usuDto.setUsuario(rs.getString("USUARIO"));

usuDto.setClave(rs.getString("CLAVE"));

usuDto.setIdrol(rs.getInt("IDROL"));

usuDto.setRol(rs.getString("NOMBRE"));

usuDto.setActivo(rs.getInt("ACTIVO"));

rs.close();

pstm.close();

} catch (SQLException e) {

this.setCode(-1);

this.setMessage(e.getMessage());

} catch (Exception e) {

this.setCode(-1);

this.setMessage("Error en el proceso, intenteo de nuevo.");

} finally {

try {

cn.close();

} catch (Exception e) {

}

}

return usuDto;

}

# CRUD DE CLIENTES

## Análisis



## Diseño

