



Manual Técnico

Proyecto: Control de empleados y administración de vacaciones

Por:

Miguel Antonio Flores Cerrato

Astrid Nicoll Gonzales Herrera

Olman Elías Martínez Muñoz

Kenny Daniel Arias Canales

Universidad Católica de Honduras

Campus Dios Espíritu Santo

Facultad de Ingeniería en Ciencias de la computación



Propósito del Documento

Este manual técnico tiene como objetivo documentar de manera detallada el **Sistema de Control de Empleados y Administración de Vacaciones**, proporcionando una referencia completa sobre su estructura, funcionamiento y mantenimiento.

En este documento se incluyen los siguientes aspectos fundamentales:

- **Base de Datos:** Diseño del modelo de datos, estructura de las tablas, relaciones, procedimientos almacenados y consultas SQL utilizadas.
- **Arquitectura del Sistema:** Descripción de la estructura en capas, incluyendo la capa de datos, la capa de negocio, la capa entidad y la capa de presentación.
- **Código Fuente:** Explicación detallada de los módulos principales del código y su funcionalidad dentro del sistema.

Este documento está dirigido a desarrolladores, administradores de bases de datos y personal técnico encargado del mantenimiento y evolución del sistema.



TALLER INDUSTRIAL REAUTO MANUAL TÉCNICO

Contenido

Arquitectura de la Base de Datos.....	6
Ubicación y acceso a la base de datos	7
Tablas.....	8
Relaciones	22
Procesos Almacenados	31
Triggers	44
Vistas	48
Código Fuente.....	54



TALLER INDUSTRIAL REAUTO MANUAL TÉCNICO

Tabla de Ilustraciones

Figura 1 Diagrama entidad relación de la base de datos	6
Figura 2 Tabla de datos del empleado	8
Figura 3 Tabla de datos del administrador.....	9
Figura 4 Tabla de firmas de los empleados	10
Figura 5 Tabla de solicitudes de vacaciones.....	11
Figura 6 Tabla de usuarios.....	12
Figura 7 Tabla de respuesta de solicitudes	13
Figura 8 Tabla que almacena datos específicos de los solicitantes	14
Figura 9 Tabla de rangos.....	16
Figura 10 Tabla de puestos	17
Figura 11 Tabla histórica de usuarios	18
Figura 12 Bitácora del sistema	20
Figura 13 Relación Empleado - Datos Solicitante.....	22
Figura 14 Relación Usuarios - Empleado.....	23
Figura 15 Relación Usuarios - Puestos.....	24
Figura 16 Relación Administrador - Empleado.....	25
Figura 17 Relación Puestos - Rangos	26
Figura 18 Relación Solicitud Vacaciones - Empleado.....	27
Figura 19 Relación Solicitud Vacaciones - Respuestas	28
Figura 20 Relación Firmas - Empleado	29
Figura 21 Relación Bitácora - Usuarios	30
Figura 22 Proceso de Solicitar Vacaciones.....	31
Figura 23 Proceso de Registrar Solicitante.....	32
Figura 24 Proceso de Contratación.....	33



TALLER INDUSTRIAL REAUTO MANUAL TÉCNICO

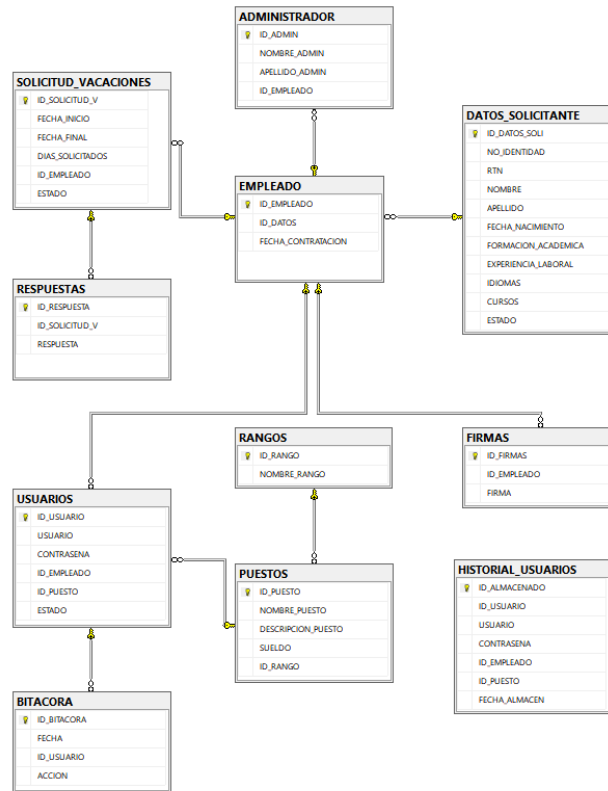
Figura 25 Proceso Calcular Días de Antigüedad	34
Figura 26 Proceso Contestar Solicitud	35
Figura 27 Proceso de Creación de Puesto	36
Figura 28 Proceso para Crear Usuarios	37
Figura 29 Proceso para Registrar Firmas de Empleados	38
Figura 30 Proceso de Cantidad de Empleados	39
Figura 31 Proceso de Empleados de Vacaciones	40
Figura 32 Proceso de Solicitudes Pendientes	41
Figura 33 Proceso de Solicitudes Por Estado	42
Figura 34 Proceso de Empleados de Vacaciones Por Mes	43
Figura 35 Trigger Actualizar Estado de Solicitudes de Vacaciones	44
Figura 36 Trigger de Registrar Administrador	45
Figura 37 Trigger de Historial de Usuarios Creados	46
Figura 38 Trigger de Solicitante Contratado	47
Figura 39 Vista para Reporte de Empleados	48
Figura 40 Vista de Inicio de Sesión	49
Figura 41 Vista de Empleados sin Usuario.....	50
Figura 42 Vista de Registrar Solicitante	51
Figura 43 Vista de Reporte de Vacaciones	52
Figura 44 Vista de Reporte de Puestos	53



TALLER INDUSTRIAL REAUTO MANUAL TÉCNICO

Arquitectura de la Base de Datos

Figura 1 Diagrama entidad relación de la base de datos



El diagrama entidad-relación de la base de datos ilustra la estructura y las relaciones entre las tablas, detallando los campos de cada una. En este esquema, se presentan las entidades principales junto con sus atributos, así como las conexiones entre ellas, según el modelo de base de datos diseñado. Este diagrama permite visualizar cómo se organizan y relacionan los datos dentro del sistema, asegurando la integridad referencial y facilitando la comprensión del flujo de información entre las distintas tablas.



TALLER INDUSTRIAL REAUTO MANUAL TÉCNICO

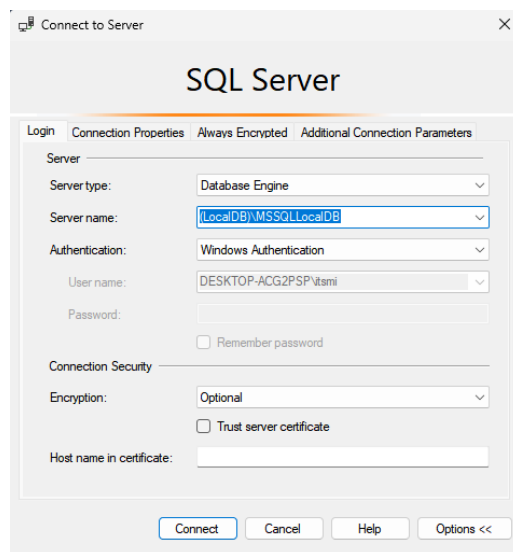
Ubicación y acceso a la base de datos

Para administrar la base de datos local utilizada por la aplicación, se debe establecer conexión con la instancia predeterminada de SQL Server LocalDB. La instancia debe especificarse en el campo **Nombre del servidor** con la siguiente sintaxis:

(LocalDB)\MSSQLLocalDB

Pasos para acceder desde SQL Server Management Studio (SSMS):

1. Abrir SQL Server Management Studio.
2. En la ventana **Conectar al servidor**, ubicar el campo **Nombre del servidor**.
3. Ingresar la instancia local:



4. Seleccionar el tipo de autenticación **Autenticación de Windows**.
5. Hacer clic en el botón **Conectar** para acceder al entorno de administración.



Tablas

Figura 2 Tabla de datos del empleado

EMPLEADO	
	ID_EMPLEADO
	ID_DATOS
	FECHA_CONTRATACION


La tabla **EMPLEADO** es una entidad central en la base de datos que almacena la información fundamental de los empleados dentro del sistema. Su estructura está compuesta por los siguientes campos:

- **ID_EMPLEADO** (*INT, PRIMARY KEY*): Actúa como identificador único de cada empleado.
- **ID_DATOS** (*INT*): Establece una relación con la tabla **DATOS_SOLICITANTE** para almacenar información adicional del empleado.
- **FECHA_CONTRATACION** (*DATE*): Registra la fecha en que el empleado fue contratado.

Esta tabla es clave para garantizar la coherencia e integridad de los datos, ya que permite que otras tablas, como **ADMINISTRADOR**, **FIRMAS**, **SOLICITUD_VACACIONES** y **USUARIOS**, recuperen información de los empleados de manera estructurada y eficiente. La relación con **DATOS_SOLICITANTE**, a través de **ID_DATOS**, facilita la gestión detallada de los empleados, permitiendo vincular información relevante en el contexto de contrataciones y solicitudes dentro del sistema.



Figura 3 Tabla de datos del administrador

ADMINISTRADOR	
	ID_ADMIN
	NOMBRE_ADMIN
	APELLIDO_ADMIN
	ID_EMPLEADO


La tabla **ADMINISTRADOR** es una entidad destinada a gestionar la información del personal con privilegios administrativos dentro del sistema. Su estructura se compone de los siguientes campos:

- **ID_ADMIN** (*INT, IDENTITY, PRIMARY KEY*): Actúa como identificador único del administrador.
- **NOMBRE_ADMIN** (*VARCHAR(30)*): Almacena el nombre del administrador.
- **APELLIDO_ADMIN** (*VARCHAR(30)*): Almacena el apellido del administrador.
- **ID_EMPLEADO** (*INT*): Funciona como clave foránea para establecer una relación directa con la tabla **EMPLEADO**.

Esta relación garantiza que cada administrador corresponda a un empleado registrado en el sistema, asegurando la integridad referencial de la base de datos. Al vincular los administradores con los empleados, se facilita una gestión eficiente de los privilegios administrativos, permitiendo un control estructurado y seguro de los accesos dentro del sistema.



Figura 4 Tabla de firmas de los empleados

FIRMAS	
	ID_FIRMAS
	ID_EMPLEADO
	FIRMA

La tabla **FIRMAS** se encarga de almacenar las firmas de los empleados en el sistema, garantizando su autenticidad y vinculación con los registros de personal. Su estructura se compone de los siguientes campos:

- **ID_FIRMAS** (*INT, PRIMARY KEY*): Actúa como identificador único de cada firma.
- **FIRMA** (*NVARCHAR(50)*): Almacena la firma en formato digital o escaneado.
- **ID_EMPLEADO** (*INT*): Funciona como clave foránea para establecer una relación directa con la tabla **EMPLEADO**.

Esta relación asegura que cada firma esté correctamente asociada con el empleado correspondiente, permitiendo su uso en validaciones, autorizaciones y otros procesos administrativos dentro del sistema. Además, el uso de una clave foránea garantiza la integridad referencial, evitando registros huérfanos y facilitando la trazabilidad de la información.



Figura 5 Tabla de solicitudes de vacaciones

SOLICITUD_VACACIONES	
	ID_SOLICITUD_V
	FECHA_INICIO
	FECHA_FINAL
	DIAS_SOLICITADOS
	ID_EMPLEADO
	ESTADO


La tabla **SOLICITUD_VACACIONES** tiene como objetivo gestionar las solicitudes de vacaciones de los empleados, garantizando un control eficiente y transparente del período de descanso del personal. Su estructura está compuesta por los siguientes campos:

- **ID_SOLICITUD_V (INT, IDENTITY, PRIMARY KEY):** Actúa como identificador único de cada solicitud.
- **FECHA_INICIO (DATE):** Registra la fecha de inicio del período de vacaciones solicitado.
- **FECHA_FINAL (DATE):** Registra la fecha final del período de vacaciones solicitado.
- **DIAS_SOLICITADOS (INT):** Indica la cantidad de días requeridos para las vacaciones.
- **ESTADO (VARCHAR(60)):** Almacena el estado de la solicitud (pendiente, aprobada o rechazada).
- **ID_EMPLEADO (INT):** Funciona como clave foránea para establecer una relación con la tabla EMPLEADO, asegurando que cada solicitud esté vinculada con un empleado registrado en el sistema.



TALLER INDUSTRIAL REAUTO MANUAL TÉCNICO

Figura 6 Tabla de usuarios

USUARIOS	
	ID_USUARIO
	USUARIO
	CONTRASENA
	ID_EMPLEADO
	ID_PUESTO
	ESTADO

La tabla **USUARIOS** es fundamental para la gestión de accesos dentro del sistema, ya que almacena la información de cada usuario registrado. Su estructura está compuesta por los siguientes campos:

- **ID_USUARIO** (*INT, PRIMARY KEY*): Identificador único de cada usuario.
- **USUARIO** (*VARCHAR(30), NOT NULL, UNIQUE*): Nombre de usuario, garantizando su unicidad dentro del sistema.
- **CONTRASENA** (*NVARCHAR(100), NOT NULL*): Clave de acceso almacenada de forma segura.
- **ID_PUESTO** (*INT*): Clave foránea que establece el puesto asignado al usuario dentro de la organización.
- **ESTADO** (*VARCHAR(15)*): Indica si el usuario está activo o inactivo.
- **ID_EMPLEADO** (*INT*): Clave foránea que vincula cada usuario con un empleado registrado en la tabla **EMPLEADO**.



Figura 7 Tabla de respuesta de solicitudes

RESPUESTAS	
	ID_RESPUESTA
	ID_SOLICITUD_V
	RESPUESTA

La tabla **RESPUESTAS** es esencial para la administración de las solicitudes de vacaciones dentro del sistema, permitiendo almacenar el estado de cada solicitud realizada por los empleados. Su estructura está compuesta por los siguientes campos:


- **ID_RESPUESTA** (*INT, IDENTITY, PRIMARY KEY*): Identificador único de cada respuesta.
- **ID_SOLICITUD_V** (*INT*): Clave foránea que establece una relación con la tabla **SOLICITUD_VACACIONES**, asegurando la correcta asociación entre solicitudes y respuestas.
- **RESPUESTA** (*VARCHAR(15)*): Indica si la solicitud fue "**Aprobada**" o "**Rechazada**".

Esta estructura facilita un seguimiento preciso de cada solicitud y su resolución, optimizando la gestión del proceso de vacaciones dentro del sistema y garantizando la integridad referencial entre las solicitudes y sus respectivas respuestas.



TALLER INDUSTRIAL REAUTO MANUAL TÉCNICO

Figura 8 Tabla que almacena datos específicos de los solicitantes

DATOS_SOLICITANTE	
	ID_DATOS_SOLI
	NO_IDENTIDAD
	RTN
	NOMBRE
	APELLIDO
	FECHA_NACIMIENTO
	FORMACION_ACADEMICA
	EXPERIENCIA_LABORAL
	IDIOMAS
	CURSOS
	ESTADO

La tabla **DATOS_SOLICITANTE** desempeña un papel fundamental en el proceso de reclutamiento y selección de personal dentro de la empresa, permitiendo almacenar información detallada sobre los postulantes a un puesto. Su estructura está compuesta por los siguientes campos:

- **ID_DATOS_SOLI** (*INT, PRIMARY KEY*): Identificador único de cada solicitante.
- **NO_IDENTIDAD** (*VARCHAR(13)*): Número de identidad del solicitante.
- **RTN** (*VARCHAR(14), NOT NULL*): Número de Registro Tributario Nacional, garantizando la identificación de cada postulante.
- **NOMBRE** (*VARCHAR(30)*): Nombre del solicitante.
- **APELLIDO** (*VARCHAR(30)*): Apellido del solicitante.
- **FECHA_NACIMIENTO** (*DATE*): Fecha de nacimiento del solicitante, utilizada para calcular la edad y evaluar la idoneidad para el puesto.




TALLER INDUSTRIAL REAUTO MANUAL TÉCNICO

- **FORMACION_ACADEMICA** (*VARCHAR(150)*): Nivel educativo alcanzado.
- **EXPERIENCIA_LABORAL** (*VARCHAR(150)*): Trayectoria profesional del postulante.
- **IDIOMAS** (*VARCHAR(100)*): Registro de idiomas que domina el solicitante.
- **CURSOS** (*VARCHAR(100)*): Cursos realizados que complementan su formación.
- **ESTADO** (*VARCHAR(15)*): Indica si el postulante está "**CONTRATADO**" o "**PENDIENTE**".

Esta estructura proporciona un sistema integral para gestionar el talento humano, asegurando un proceso eficiente de selección y contratación en la empresa



Figura 9 Tabla de rangos

RANGOS	
	ID_RANGO
	NOMBRE_RANGO


La tabla **RANGOS** es un componente esencial para la administración del sistema y la gestión de usuarios, permitiendo definir y estructurar los distintos niveles jerárquicos dentro de la plataforma. Su estructura está compuesta por los siguientes campos:

- **ID_RANGO** (*INT, PRIMARY KEY*): Identificador único del rango, utilizado como clave foránea en la tabla **PUESTOS** para establecer la relación entre los puestos y los niveles de autoridad dentro de la organización.
- **NOMBRE_RANGO** (*VARCHAR(50)*): Nombre del rango, facilitando la categorización de los usuarios según sus privilegios y responsabilidades dentro del sistema.

Esta estructura garantiza una gestión eficiente de los permisos y accesos, optimizando la administración y seguridad de la plataforma.



Figura 10 Tabla de puestos

PUESTOS	
	ID_PUESTO
	NOMBRE_PUESTO
	DESCRIPCION_PUESTO
	SUELDO
	ID_RANGO


La tabla **PUESTOS** es fundamental para la gestión organizativa dentro del sistema, ya que permite almacenar información clave sobre los diferentes cargos existentes en la empresa. Su estructura está compuesta por los siguientes campos:

- **ID_PUESTO** (*INT, PRIMARY KEY*): Identificador único del puesto, asegurando la integridad de los datos.
- **NOMBRE_PUESTO** (*VARCHAR(30)*): Define el nombre del puesto.
- **DESCRIPCION_PUESTO** (*VARCHAR(150)*): Proporciona una descripción detallada de las funciones y responsabilidades asociadas al cargo.
- **SUELDO** (*INT*): Almacena el salario correspondiente a cada puesto, facilitando la gestión de remuneraciones.
- **ID_RANGO** (*INT*): Clave foránea vinculada a la tabla **RANGOS**, permitiendo establecer una relación estructurada entre los distintos niveles jerárquicos de la empresa.

Esta estructura optimiza la administración de los usuarios y sus funciones dentro del sistema, garantizando una jerarquía organizativa bien definida.



Figura 11 Tabla histórica de usuarios

HISTORIAL_USUARIOS	
	ID_ALMACENADO
	ID_USUARIO
	USUARIO
	CONTRASENA
	ID_EMPLEADO
	ID_PUESTO
	FECHA_ALMACEN

La tabla **HISTORIAL_USUARIOS** está diseñada para almacenar un registro histórico de todos los usuarios que han existido en el sistema, proporcionando trazabilidad y transparencia a lo largo del tiempo. Su estructura incluye los siguientes campos:

- **ID_ALMACENADO** (*INT, IDENTITY, PRIMARY KEY*): Identificador único de cada entrada histórica, garantizando la unicidad de los registros.
- **ID_USUARIO** (*INT*): Almacena el identificador del usuario.
- **USUARIO** (*VARCHAR(30), UNIQUE*): Guarda el nombre de usuario asignado, el cual debe ser único en el sistema.
- **CONTRASENA** (*NVARCHAR(100)*): Almacena la contraseña del usuario de forma segura.
- **ID_EMPLEADO** (*INT*): Identificador del empleado correspondiente a ese usuario.
- **ID_PUESTO** (*INT*): Guarda el puesto asignado al usuario en el momento de la entrada histórica.




TALLER INDUSTRIAL REAUTO MANUAL TÉCNICO

- **FECHA_ALMACEN** (*DATETIME*): Registra la fecha y hora en la que se almacenó la información, permitiendo el seguimiento detallado de la existencia y cambios de los usuarios a lo largo del tiempo.

Esta estructura asegura que los datos históricos de los usuarios sean conservados, incluso si los usuarios son eliminados de otras tablas del sistema, lo cual es fundamental para auditorías, análisis o trazabilidad en el futuro.



Figura 12 Bitácora del sistema

BITACORA	
	ID_BITACORA
	FECHA
	ID_USUARIO
	ACCION

La tabla **BITACORA** está diseñada para almacenar un registro detallado de todas las acciones realizadas dentro del sistema, lo que permite una trazabilidad completa de las actividades de los usuarios. La estructura de la tabla incluye los siguientes campos:

- **ID_BITACORA** (*INT, PRIMARY KEY, IDENTITY*): Identificador único de cada acción registrada, garantizando la unicidad de los registros.
- **FECHA** (*DATETIME*): Almacena la fecha y hora exactas en las que se realizó la acción, proporcionando contexto temporal para cada actividad.
- **USUARIO** (*VARCHAR(30)*): Guarda el nombre del usuario que realizó la acción, permitiendo identificar quién llevó a cabo la actividad.
- **ACCION** (*VARCHAR(150)*): Describe detalladamente la acción realizada, ofreciendo información sobre el tipo de operación ejecutada.

Adicionalmente, como mejora en una implementación más completa, se podría agregar una clave foránea **ID_USUARIO** (*INT*), vinculada con la tabla **USUARIOS**, lo que permitiría una relación directa entre la acción registrada y el usuario que la ejecutó.



TALLER INDUSTRIAL REAUTO MANUAL TÉCNICO

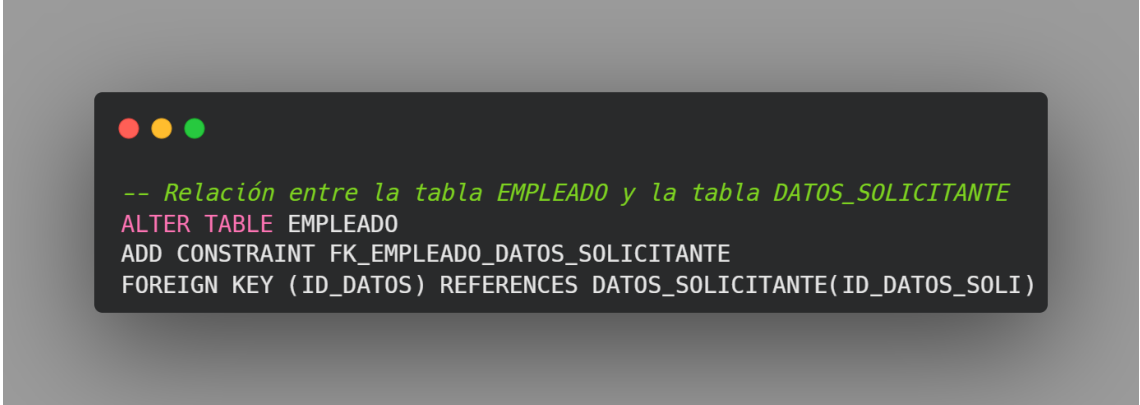
Esto facilitaría el seguimiento de las actividades de los usuarios dentro del sistema y garantizaría la integridad referencial en el registro de las acciones.

Este diseño asegura un historial de las actividades del sistema, lo cual es crucial para auditar, controlar y analizar las interacciones de los usuarios en la plataforma.



Relaciones

Figura 13 Relación Empleado - Datos Solicitante



```
-- Relación entre la tabla EMPLEADO y la tabla DATOS_SOLICITANTE
ALTER TABLE EMPLEADO
ADD CONSTRAINT FK_EMPLEADO_DATOS_SOLICITANTE
FOREIGN KEY (ID_DATOS) REFERENCES DATOS_SOLICITANTE(ID_DATOS_SOLI)
```

El comando SQL **ALTER TABLE EMPLEADO** agrega una restricción de clave foránea en la tabla **EMPLEADO**, creando la relación con la tabla **DATOS_SOLICITANTE** mediante el campo **ID_DATOS**. Esta restricción, denominada **FK_EMPLEADO_DATOS_SOLICITANTE**, asegura la integridad referencial entre ambas tablas, de modo que el valor de **ID_DATOS** en **EMPLEADO** debe coincidir con un valor válido de **ID_DATOS_SOLI** en **DATOS_SOLICITANTE**, garantizando que solo se asocien registros de empleados con datos solicitantes existentes, lo que facilita la gestión de las relaciones entre los datos de empleados y sus solicitudes dentro del sistema.



Figura 14 Relación Usuarios - Empleado

```
-- Relación entre la tabla USUARIOS Y EMPLEADO
ALTER TABLE USUARIOS
ADD CONSTRAINT FK_USUARIO_EMPLEADO
FOREIGN KEY (ID_EMPLEADO) REFERENCES EMPLEADO(ID_EMPLEADO)
```

El comando SQL **ALTER TABLE USUARIOS** agrega una restricción de clave foránea en la tabla **USUARIOS**, creando una relación con la tabla **EMPLEADO** mediante el campo **ID_EMPLEADO**. Esta restricción, denominada **FK_USUARIO_EMPLEADO**, asegura la integridad referencial entre ambas tablas, de manera que el valor de **ID_EMPLEADO** en **USUARIOS** debe coincidir con un valor válido de **ID_EMPLEADO** en **EMPLEADO**, garantizando que cada usuario esté asociado con un empleado registrado en el sistema.



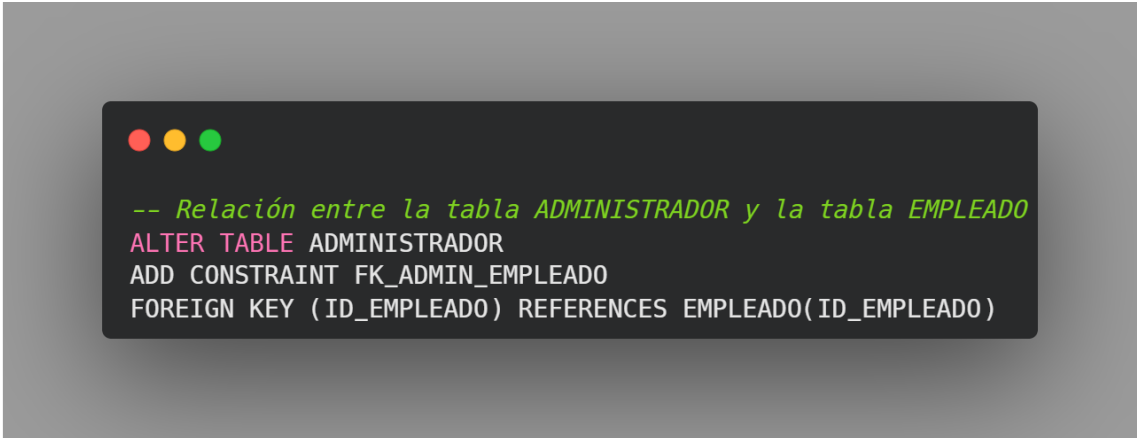
Figura 15 Relación Usuarios - Puestos

```
--Relacion entre la tabla USUARIOS Y PUESTOS  
ALTER TABLE USUARIOS  
ADD CONSTRAINT FK_USUARIOS_PUESTOS  
FOREIGN KEY (ID_PUESTO) REFERENCES PUESTOS(ID_PUESTO)
```

El comando SQL **ALTER TABLE USUARIOS** agrega una restricción de clave foránea en la tabla **USUARIOS**, estableciendo una relación con la tabla **PUESTOS** mediante el campo **ID_PUESTO**. Esta restricción, denominada **FK_USUARIOS_PUESTOS**, asegura la integridad referencial entre ambas tablas, de modo que el valor de **ID_PUESTO** en **USUARIOS** debe coincidir con un valor válido de **ID_PUESTO** en **PUESTOS**, garantizando que cada usuario esté asociado a un puesto existente dentro del sistema.



Figura 16 Relación Administrador - Empleado



```
-- Relación entre la tabla ADMINISTRADOR y la tabla EMPLEADO
ALTER TABLE ADMINISTRADOR
ADD CONSTRAINT FK_ADMIN_EMPLEADO
FOREIGN KEY (ID_EMPLEADO) REFERENCES EMPLEADO(ID_EMPLEADO)
```

El comando SQL **ALTER TABLE ADMINISTRADOR** agrega una restricción de clave foránea en la tabla **ADMINISTRADOR**, estableciendo una relación con la tabla **EMPLEADO** mediante el campo **ID_EMPLEADO**. Esta restricción, denominada **FK_ADMIN_EMPLEADO**, asegura la integridad referencial entre ambas tablas, de modo que el valor de **ID_EMPLEADO** en **ADMINISTRADOR** debe coincidir con un valor válido de **ID_EMPLEADO** en **EMPLEADO**, garantizando que cada administrador esté asociado con un empleado registrado en el sistema.



Figura 17 Relación Puestos - Rangos

```
-- Relación entre la tabla PUESTOS y la tabla RANGOS
ALTER TABLE PUESTOS
ADD CONSTRAINT FK_PUESTO_RANGO
FOREIGN KEY (ID_RANGO) REFERENCES RANGOS(ID_RANGO)
```

El comando SQL **ALTER TABLE PUESTOS** agrega una restricción de clave foránea en la tabla **PUESTOS**, estableciendo una relación con la tabla **RANGOS** mediante el campo **ID_RANGO**. Esta restricción, denominada **FK_PUESTO_RANGO**, asegura la integridad referencial entre ambas tablas, de modo que el valor de **ID_RANGO** en **PUESTOS** debe coincidir con un valor válido de **ID_RANGO** en **RANGOS**, garantizando que cada puesto esté asociado con un rango existente dentro del sistema.



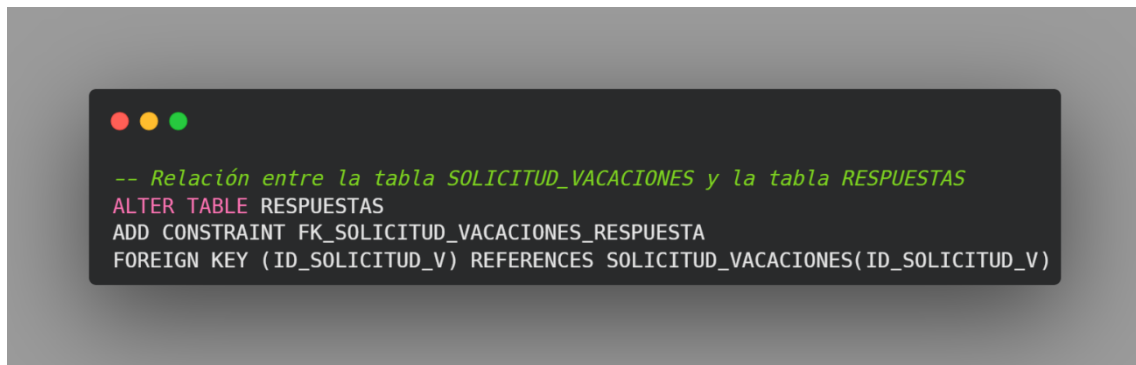
Figura 18 Relación Solicitud Vacaciones - Empleado

```
-- Relación entre la tabla SOLICITUD_VACACIONES y la tabla EMPLEADO
ALTER TABLE SOLICITUD_VACACIONES
ADD CONSTRAINT FK_SOLICITUD_VACACIONES_EMPLEADO
FOREIGN KEY (ID_EMPLEADO) REFERENCES EMPLEADO(ID_EMPLEADO)
```

El comando SQL **ALTER TABLE SOLICITUD_VACACIONES** agrega una restricción de clave foránea en la tabla **SOLICITUD_VACACIONES**, estableciendo una relación con la tabla **EMPLEADO** mediante el campo **ID_EMPLEADO**. Esta restricción, denominada **FK_SOLICITUD_VACACIONES_EMPLEADO**, asegura la integridad referencial entre ambas tablas, de modo que el valor de **ID_EMPLEADO** en **SOLICITUD_VACACIONES** debe coincidir con un valor válido de **ID_EMPLEADO** en **EMPLEADO**, garantizando que cada solicitud de vacaciones esté asociada a un empleado registrado en el sistema.



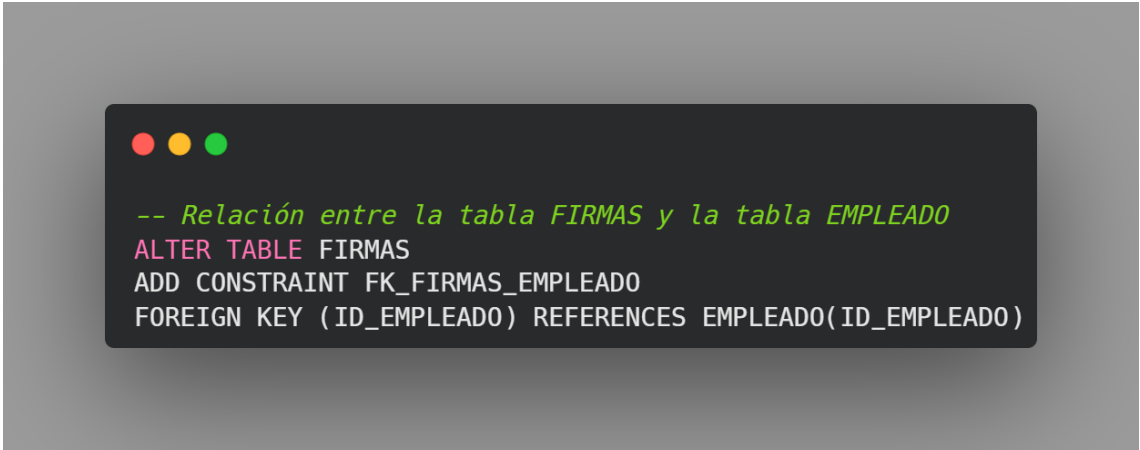
Figura 19 Relación Solicitud Vacaciones - Respuestas



El comando SQL **ALTER TABLE RESPUESTAS** agrega una restricción de clave foránea en la tabla **RESPUESTAS**, estableciendo una relación con la tabla **SOLICITUD_VACACIONES** mediante el campo **ID_SOLICITUD_V**. Esta restricción, denominada **FK_SOLICITUD_VACACIONES_RESPUESTA**, asegura la integridad referencial entre ambas tablas, de modo que el valor de **ID_SOLICITUD_V** en **RESPUESTAS** debe coincidir con un valor válido de **ID_SOLICITUD_V** en **SOLICITUD_VACACIONES**, garantizando que cada respuesta esté asociada a una solicitud de vacaciones existente dentro del sistema.



Figura 20 Relación Firmas - Empleado



```
-- Relación entre la tabla FIRMAS y la tabla EMPLEADO
ALTER TABLE FIRMAS
ADD CONSTRAINT FK_FIRMAS_EMPLEADO
FOREIGN KEY (ID_EMPLEADO) REFERENCES EMPLEADO(ID_EMPLEADO)
```

El comando SQL **ALTER TABLE FIRMAS** agrega una restricción de clave foránea en la tabla **FIRMAS**, estableciendo una relación con la tabla **EMPLEADO** mediante el campo **ID_EMPLEADO**. Esta restricción, denominada **FK_FIRMAS_EMPLEADO**, asegura la integridad referencial entre ambas tablas, de modo que el valor de **ID_EMPLEADO** en **FIRMAS** debe coincidir con un valor válido de **ID_EMPLEADO** en **EMPLEADO**, garantizando que cada firma esté asociada a un empleado registrado en el sistema.



Figura 21 Relación Bitácora - Usuarios

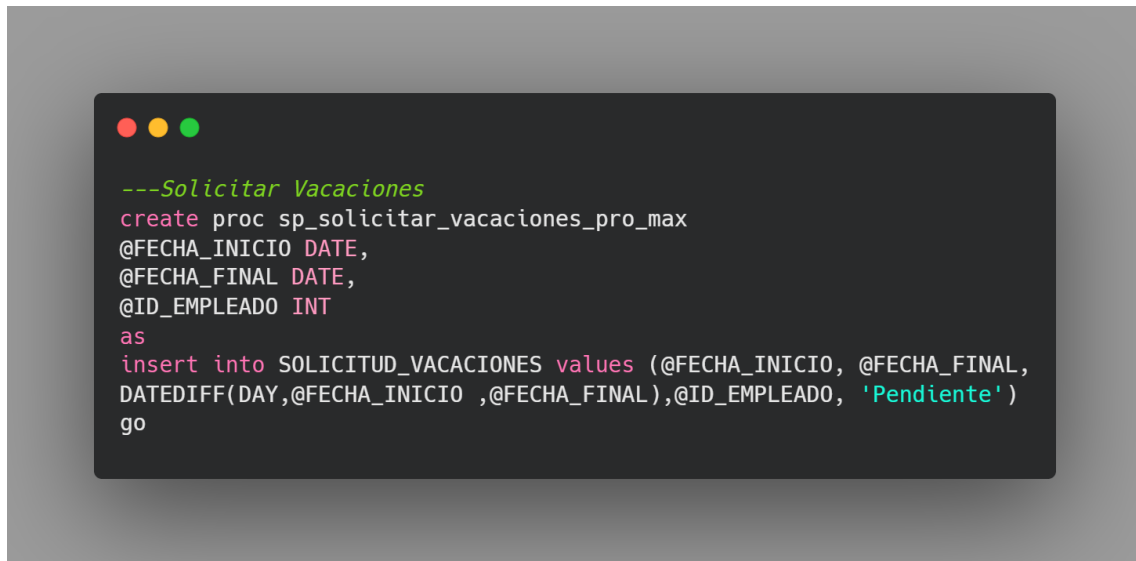
```
-- Relación entre la tabla BITACORA y la tabla USUARIOS  
ALTER TABLE BITACORA  
ADD CONSTRAINT FK_BITACORA_USUARIO  
FOREIGN KEY (ID_USUARIO) REFERENCES USUARIOS(ID_USUARIO)
```

El comando SQL **ALTER TABLE BITACORA** agrega una restricción de clave foránea en la tabla **BITACORA**, estableciendo una relación con la tabla **USUARIOS** mediante el campo **ID_USUARIO**. Esta restricción, denominada **FK_BITACORA_USUARIO**, asegura la integridad referencial entre ambas tablas, de modo que el valor de **ID_USUARIO** en **BITACORA** debe coincidir con un valor válido de **ID_USUARIO** en **USUARIOS**, garantizando que cada entrada en el registro de la bitácora esté asociada a un usuario registrado en el sistema.



Procesos Almacenados

Figura 22 Proceso de Solicitar Vacaciones



El procedimiento almacenado **sp_solicitar_vacaciones_pro_max** permite gestionar la solicitud de vacaciones de un empleado en el sistema, insertando un nuevo registro en la tabla **SOLICITUD_VACACIONES**. Este procedimiento recibe como parámetros la **fecha de inicio** (**@FECHA_INICIO**), la **fecha final** (**@FECHA_FINAL**) y el **identificador del empleado** (**@ID_EMPLEADO**). Utiliza la función **DATEDIFF** para calcular automáticamente el número de días solicitados y asigna el estado inicial de la solicitud como **'Pendiente'**, asegurando un control adecuado del proceso antes de su aprobación o rechazo por parte de los administradores.



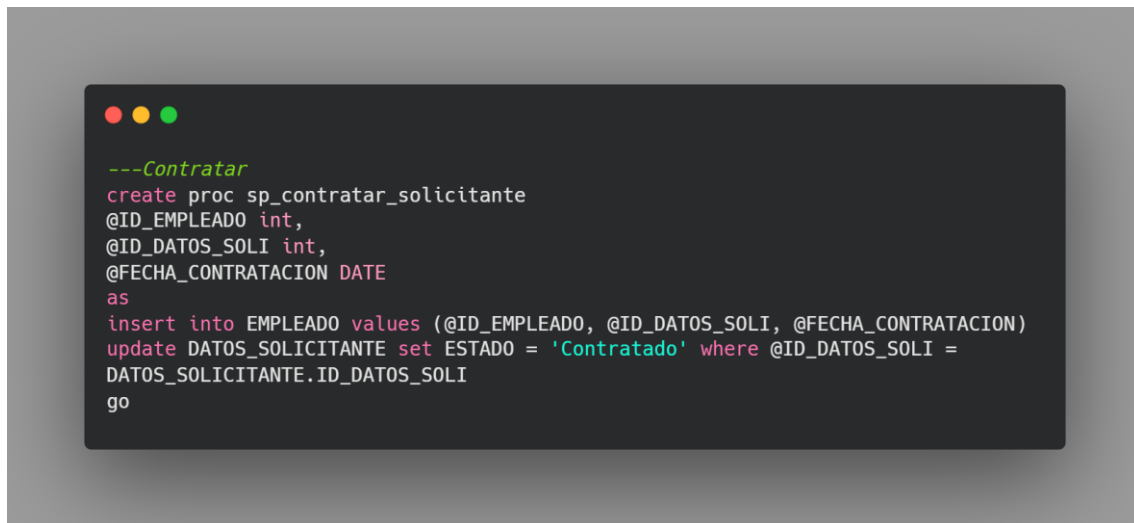
Figura 23 Proceso de Registrar Solicitante

```
---Registrar Solicitante
create proc sp_registrar_solicitante
@ID_DATOS_SOLI INT,
@NO_IDENTIDAD VARCHAR(13),
@RTN VARCHAR(14),
@NOMBRE VARCHAR(30),
@APELLIDO VARCHAR(30),
@FECHA_NACIMIENTO DATE,
@FORMACION_ACADEMICA VARCHAR(150),
@EXPERIENCIA_LABORAL VARCHAR(150),
@IDIOMAS VARCHAR(100),
@CURSOS VARCHAR(100)
as
insert into DATOS_SOLICITANTE values (@ID_DATOS_SOLI, @NO_IDENTIDAD, @RTN, @NOMBRE, @APELLIDO,
@FECHA_NACIMIENTO, @FORMACION_ACADEMICA, @EXPERIENCIA_LABORAL, @IDIOMAS, @CURSOS, 'En revision')
go
```

El procedimiento almacenado **sp_registrar_solicitante** permite registrar en la base de datos a una persona interesada en aplicar a un puesto dentro de la empresa, insertando un nuevo registro en la tabla **DATOS_SOLICITANTE**. Recibe como parámetros el **identificador del solicitante** (**@ID_DATOS_SOLI**), el **número de identidad** (**@NO_IDENTIDAD**), el **RTN** (**@RTN**), el **nombre** (**@NOMBRE**) y **apellido** (**@APELLIDO**), así como la **fecha de nacimiento** (**@FECHA_NACIMIENTO**). Además, almacena información adicional como la **formación académica** (**@FORMACION_ACADEMICA**), la **experiencia laboral** (**@EXPERIENCIA_LABORAL**), los **idiomas que domina** (**@IDIOMAS**) y los **cursos realizados** (**@CURSOS**). Al registrar un nuevo solicitante, el sistema asigna automáticamente el estado inicial como **'En revisión'**, asegurando que el proceso de evaluación y selección se lleve a cabo de manera estructurada.



Figura 24 Proceso de Contratación



El procedimiento almacenado **sp_contratar_solicitante** permite registrar en la base de datos la contratación de un solicitante, insertándolo en la tabla **EMPLEADO** y actualizando su estado en la tabla **DATOS_SOLICITANTE**. Recibe como parámetros el **identificador del empleado (@ID_EMPLEADO)**, el **identificador del solicitante (@ID_DATOS_SOLI)** y la **fecha de contratación (@FECHA_CONTRATACION)**. Primero, se inserta un nuevo registro en la tabla **EMPLEADO**, estableciendo la relación con los datos previos del solicitante. Posteriormente, se actualiza el campo **ESTADO** en la tabla **DATOS_SOLICITANTE**, cambiándolo a '**Contratado**', asegurando así la trazabilidad del proceso de selección y contratación dentro del sistema.



Figura 25 Proceso Calcular Días de Antigüedad

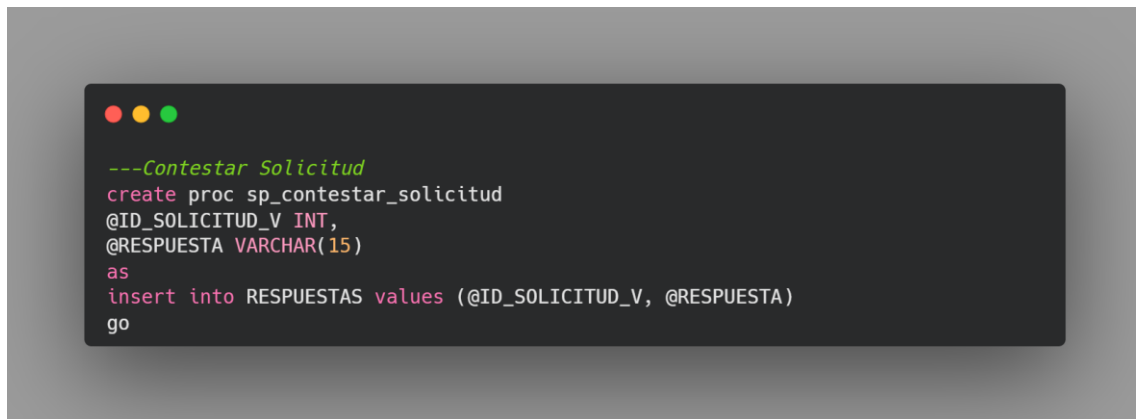
```
---Calcular Días de Antigüedad
CREATE PROC sp_Calcular_Dias
    @ID_EMPLEADO INT
AS
BEGIN
    SET NOCOUNT ON;

    SELECT DATEDIFF(DAY, EMPLEADO.FECHA_CONTRATACION, GETDATE()) AS
    DIAS_ANTIGÜEDAD
    FROM EMPLEADO
    WHERE EMPLEADO.ID_EMPLEADO = @ID_EMPLEADO;
END;
go
```

El procedimiento almacenado **sp_Calcular_Dias** permite determinar la cantidad de días de antigüedad de un empleado dentro de la empresa. Recibe como parámetro el **identificador del empleado (@ID_EMPLEADO)** y, mediante la función **DATEDIFF(DAY, EMPLEADO.FECHA_CONTRATACION, GETDATE())**, calcula la diferencia en días entre la fecha de contratación registrada en la tabla **EMPLEADO** y la fecha actual. Este procedimiento es útil para obtener información sobre el tiempo de servicio del empleado dentro del sistema, facilitando la gestión de beneficios, reconocimientos y derechos laborales asociados a la antigüedad.



Figura 26 Proceso Contestar Solicitud



El procedimiento almacenado **sp_contestar_solicitud** permite registrar una respuesta a una solicitud de vacaciones dentro del sistema. Recibe como parámetros el **identificador de la solicitud (@ID_SOLICITUD_V)** y la **respuesta (@RESPUESTA)**, que indica si la solicitud ha sido aprobada o rechazada. Posteriormente, inserta estos valores en la tabla **RESPUESTAS**, asegurando el registro de la decisión tomada sobre la solicitud de vacaciones del empleado. Este procedimiento facilita la gestión eficiente de las solicitudes dentro del sistema.



Figura 27 Proceso de Creación de Puesto

```
---Crear Puestos
create proc sp_crear_puesto
@ID_PUESTO INT,
@NOMBRE_PUESTO VARCHAR(30),
@DESCRIPCION_PUESTO VARCHAR(150),
@SUELDO INT,
@ID_RANGO INT
as
insert into PUESTOS values (@ID_PUESTO, @NOMBRE_PUESTO, @DESCRIPCION_PUESTO,
@SUELDO, @ID_RANGO)
go
```

El procedimiento almacenado **sp_crear_puesto** permite registrar nuevos puestos dentro del sistema. Recibe como parámetros el **identificador del puesto** (**@ID_PUESTO**), el **nombre del puesto** (**@NOMBRE_PUESTO**), la **descripción del puesto** (**@DESCRIPCION_PUESTO**), el **sueldo asociado** (**@SUELDO**) y el **identificador del rango** (**@ID_RANGO**), el cual establece la relación con la tabla **RANGOS**. Posteriormente, inserta estos valores en la tabla **PUESTOS**, permitiendo gestionar y clasificar los diferentes puestos dentro de la empresa de manera estructurada y eficiente.



Figura 28 Proceso para Crear Usuarios

```
---Crear Usuarios
create proc sp_crear_usuario
@ID_USUARIO INT,
@USUARIO VARCHAR(30),
@CONTRASENA NVARCHAR(100),
@ID_EMPLEADO INT,
@ID_PUESTO INT
as
insert into USUARIOS values (@ID_USUARIO, @USUARIO, @CONTRASENA, @ID_EMPLEADO,
@ID_PUESTO)
go
```

El procedimiento almacenado **sp_crear_usuario** permite registrar nuevos usuarios dentro del sistema. Recibe como parámetros el **identificador del usuario** (**@ID_USUARIO**), el **nombre de usuario** (**@USUARIO**), la **contraseña** (**@CONTRASENA**), el **identificador del empleado** (**@ID_EMPLEADO**) y el **identificador del puesto** (**@ID_PUESTO**). Estos valores se insertan en la tabla **USUARIOS**, asegurando la correcta asociación entre los empleados y sus credenciales de acceso, facilitando la administración y control del sistema.



Figura 29 Proceso para Registrar Firmas de Empleados



El procedimiento almacenado **sp_registrar_firma** tiene como objetivo insertar un nuevo registro en la tabla **FIRMAS**. Este procedimiento recibe tres parámetros de entrada: **@ID_FIRMAS**, que corresponde al identificador único de la firma, **@ID_EMPLEADO**, que es la clave foránea que vincula la firma al empleado correspondiente, y **@FIRMA**, que almacena la firma digital o escaneada del empleado en formato **NVARCHAR(50)**. Al ejecutarse, el procedimiento realiza una inserción en la tabla **FIRMAS**, asegurando que la firma esté correctamente asociada con el **ID_EMPLEADO**, estableciendo así la integridad referencial y permitiendo la gestión adecuada de las firmas dentro del sistema.



Figura 30 Proceso de Cantidad de Empleados

```
---Cantidad Empleados
CREATE PROCEDURE sp_Cantidad_Empleados
    @Mes INT = NULL,
    @Cantidad_Empleados INT OUTPUT
AS
BEGIN
    IF @Mes IS NULL OR @Mes = 0
    BEGIN
        SELECT @Cantidad_Empleados = COUNT(*) FROM EMPLEADO;
    END
    ELSE
    BEGIN
        SELECT @Cantidad_Empleados = COUNT(*)
        FROM EMPLEADO
        WHERE MONTH(FECHA_CONTRATACION) = @Mes;
    END
END
go
```

El procedimiento almacenado **sp_Cantidad_Empleados** se utiliza para obtener la cantidad de empleados registrados en el sistema. Este procedimiento recibe dos parámetros: **@Mes**, que es un parámetro opcional que representa el mes por el cual se desea filtrar la cantidad de empleados, y **@Cantidad_Empleados**, que es un parámetro de salida que devuelve el resultado de la cantidad de empleados. Si el parámetro **@Mes** es **NULL** o igual a 0, el procedimiento contará todos los empleados sin aplicar ningún filtro de mes. Si se proporciona un valor para **@Mes**, se filtrará la cantidad de empleados según el mes de su **FECHA_CONTRATACION**. El resultado de la consulta se almacena en el parámetro de salida **@Cantidad_Empleados**, permitiendo su posterior uso fuera del procedimiento.



Figura 31 Proceso de Empleados de Vacaciones

```
---Empleados que estan de vacaciones
CREATE PROCEDURE sp_Empleados_De_Vacaciones
    @Mes INT = NULL,
    @Cantidad_Vacaciones INT OUTPUT
AS
BEGIN
    IF @Mes IS NULL OR @Mes = 0
        SET @Mes = MONTH(GETDATE());

    SET @Cantidad_Vacaciones = (
        SELECT COUNT(DISTINCT EMPLEADO.ID_EMPLEADO)
        FROM SOLICITUD_VACACIONES
        INNER JOIN EMPLEADO ON SOLICITUD_VACACIONES.ID_EMPLEADO =
        EMPLEADO.ID_EMPLEADO
        WHERE MONTH(SOLICITUD_VACACIONES.FECHA_INICIO) = @Mes
        AND YEAR(SOLICITUD_VACACIONES.FECHA_INICIO) = YEAR(GETDATE())
        AND SOLICITUD_VACACIONES.ESTADO = 'Solicitud Aprobada'
    )
END
go
```

El procedimiento almacenado **sp_Empleados_De_Vacaciones** tiene como objetivo calcular la cantidad de empleados que están de vacaciones en un mes determinado. Este procedimiento recibe dos parámetros: **@Mes**, que es un parámetro opcional para especificar el mes en el que se desea realizar el conteo, y **@Cantidad_Vacaciones**, un parámetro de salida que devuelve el número de empleados de vacaciones. Si **@Mes** es **NULL** o **0**, el procedimiento utiliza el mes actual para realizar la consulta. Dentro de la lógica del procedimiento, se cuenta el número de empleados únicos (**DISTINCT EMPLEADO.ID_EMPLEADO**) que tienen solicitudes de vacaciones aprobadas (**SOLICITUD_VACACIONES.ESTADO = 'Solicitud Aprobada'**) en el mes y año especificados, basándose en la fecha de inicio de las vacaciones (**SOLICITUD_VACACIONES.FECHA_INICIO**). El resultado se almacena en **@Cantidad_Vacaciones** para su posterior uso.



Figura 32 Proceso de Solicitudes Pendientes

```
---Solicitudes Pendientes
CREATE PROCEDURE sp_Solicitudes_Pendientes
    @Mes INT = NULL,
    @Solicitudes_Pendientes INT OUTPUT
AS
BEGIN
    IF @Mes IS NULL OR @Mes = 0
        SET @Mes = MONTH(GETDATE());

    SET @Solicitudes_Pendientes = (
        SELECT COUNT(SOLICITUD_VACACIONES.ID_SOLICITUD_V)
        FROM SOLICITUD_VACACIONES
        WHERE MONTH(SOLICITUD_VACACIONES.FECHA_INICIO) = @Mes
              AND YEAR(SOLICITUD_VACACIONES.FECHA_INICIO) = YEAR(GETDATE())
              AND SOLICITUD_VACACIONES.ESTADO = 'Pendiente'
    )
END
go
```

El procedimiento almacenado **sp_Solicitudes_Pendientes** tiene como objetivo calcular la cantidad de solicitudes de vacaciones pendientes en un mes determinado. Recibe dos parámetros: **@Mes**, que es un parámetro opcional para especificar el mes en el que se desea realizar el conteo, y **@Solicitudes_Pendientes**, que es un parámetro de salida que devuelve el número de solicitudes pendientes. Si **@Mes** es **NULL** o **0**, el procedimiento utiliza el mes actual para la consulta. La lógica del procedimiento filtra las solicitudes de vacaciones en la tabla **SOLICITUD_VACACIONES**, contando aquellas cuyo estado es **Pendiente**, y cuya fecha de inicio (**FECHA_INICIO**) coincide con el mes y año especificados. El resultado se almacena en **@Solicitudes_Pendientes** para su uso posterior.



Figura 33 Proceso de Solicitudes Por Estado

```
---SOLICITUDES AGRUPADAS POR ESTADO
CREATE PROCEDURE sp_Solicitudes_Estado
    @Mes INT = NULL
AS
BEGIN
    IF @Mes IS NULL OR @Mes = 0
        SET @Mes = MONTH(GETDATE());

    SELECT
        SOLICITUD_VACACIONES.ESTADO,
        COUNT(SOLICITUD_VACACIONES.ID_SOLICITUD_V) AS SOLICITUDES
    FROM SOLICITUD_VACACIONES
    WHERE MONTH(SOLICITUD_VACACIONES.FECHA_INICIO) = @Mes
        AND YEAR(SOLICITUD_VACACIONES.FECHA_INICIO) = YEAR(GETDATE())
    GROUP BY SOLICITUD_VACACIONES.ESTADO;
END
go
```

El procedimiento almacenado **sp_Solicitudes_Estado** tiene como propósito agrupar y contar las solicitudes de vacaciones por su estado (**Pendiente, Aprobada, Rechazada**, etc.) en un mes específico. El procedimiento recibe un parámetro **@Mes**, que indica el mes para filtrar las solicitudes. Si **@Mes** es **NULL** o **0**, el procedimiento utiliza el mes actual. Luego, realiza una consulta a la tabla **SOLICITUD_VACACIONES** donde filtra las solicitudes según el mes y año de la **FECHA_INICIO**. La consulta agrupa los resultados por el campo **ESTADO** y devuelve el conteo de solicitudes por estado. El resultado final muestra cuántas solicitudes existen en cada estado dentro del mes especificado, proporcionando una visión general del estado de las solicitudes en ese período.



Figura 34 Proceso de Empleados de Vacaciones Por Mes

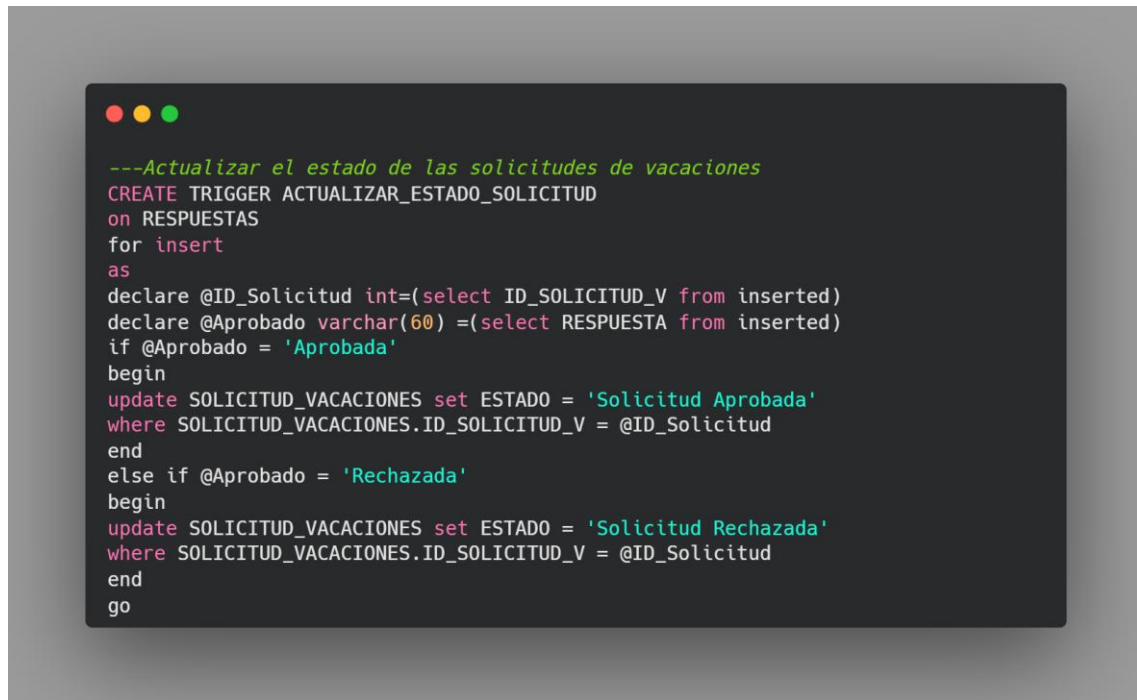
```
---Empleados de vacaciones por mes
create proc Empleados_Vacaciones_Por_Mes
as
SELECT
    DATENAME(MONTH, SOLICITUD_VACACIONES.FECHA_INICIO) AS MES,
    COUNT(DISTINCT EMPLEADO.ID_EMPLEADO) AS EMPLEADOS_DE_VACACIONES
FROM SOLICITUD_VACACIONES
INNER JOIN EMPLEADO ON SOLICITUD_VACACIONES.ID_EMPLEADO = EMPLEADO.ID_EMPLEADO
WHERE (YEAR(SOLICITUD_VACACIONES.FECHA_INICIO) = YEAR(GETDATE()))
    OR YEAR(SOLICITUD_VACACIONES.FECHA_FINAL) = YEAR(GETDATE()))
    AND SOLICITUD_VACACIONES.ESTADO = 'Solicitud Aprobada'
GROUP BY DATENAME(MONTH, SOLICITUD_VACACIONES.FECHA_INICIO),
    MONTH(SOLICITUD_VACACIONES.FECHA_INICIO)
ORDER BY MONTH(SOLICITUD_VACACIONES.FECHA_INICIO);
go
```

El procedimiento almacenado **Empleados_Vacaciones_Por_Mes** tiene como objetivo obtener el número de empleados que están de vacaciones, agrupados por mes, durante el año en curso. Utiliza una consulta que se une entre las tablas **SOLICITUD_VACACIONES** y **EMPLEADO** para identificar los empleados que tienen solicitudes de vacaciones aprobadas. La consulta filtra las solicitudes de acuerdo con el año actual utilizando las funciones **YEAR(GETDATE())** para comparar las fechas de inicio y final de las vacaciones. Luego, agrupa los resultados por el nombre del mes (**DATENAME(MONTH, SOLICITUD_VACACIONES.FECHA_INICIO)**) y el número del mes (**MONTH(SOLICITUD_VACACIONES.FECHA_INICIO)**), para obtener el conteo de empleados por mes. Finalmente, los resultados se ordenan según el mes para asegurar que se presenten de forma cronológica.



Triggers

Figura 35 Trigger Actualizar Estado de Solicitudes de Vacaciones



El **TRIGGER ACTUALIZAR_ESTADO_SOLICITUD** se ejecuta después de una inserción en la tabla **RESPUESTAS**. Su propósito es actualizar el estado de las solicitudes de vacaciones en la tabla **SOLICITUD_VACACIONES** según la respuesta registrada. Al activarse el **trigger**, extrae el **ID_SOLICITUD_V** y la **RESPUESTA** de la tabla **inserted**, que contiene los registros recién insertados en **RESPUESTAS**. Luego, verifica si la respuesta es "Aprobada" o "Rechazada" y, dependiendo del valor, actualiza el campo **ESTADO** de la solicitud correspondiente en **SOLICITUD_VACACIONES** a "Solicitud Aprobada" o "Solicitud Rechazada", respectivamente. Esto garantiza que el estado de la solicitud se mantenga actualizado según las respuestas de los usuarios.



Figura 36 Trigger de Registrar Administrador

```
---Registrar Admin
create trigger registrar_admin
ON USUARIOS
FOR INSERT
as
declare @ID_PUESTO INT=(SELECT ID_PUESTO FROM inserted)
declare @ID_RANGO INT=(SELECT RANGOS.ID_RANGO FROM RANGOS INNER JOIN PUESTOS
ON RANGOS.ID_RANGO = PUESTOS.ID_RANGO WHERE PUESTOS.ID_PUESTO = @ID_PUESTO)
IF @ID_RANGO = 1
BEGIN
declare @ID_EMPLEADO INT=(SELECT ID_EMPLEADO from inserted)
declare @NOMBRE_ADMIN VARCHAR(30)=(SELECT DATOS_SOLICITANTE.NOMBRE FROM
DATOS_SOLICITANTE INNER JOIN EMPLEADO
ON DATOS_SOLICITANTE.ID_DATOS_SOLI = EMPLEADO.ID_DATOS INNER JOIN USUARIOS
ON EMPLEADO.ID_EMPLEADO = USUARIOS.ID_EMPLEADO
WHERE EMPLEADO.ID_EMPLEADO = @ID_EMPLEADO AND USUARIOS.ID_PUESTO = 1 )
declare @APELLIDO_ADMIN VARCHAR(30)=(SELECT DATOS_SOLICITANTE.APELLIDO FROM
DATOS_SOLICITANTE INNER JOIN EMPLEADO
ON DATOS_SOLICITANTE.ID_DATOS_SOLI = EMPLEADO.ID_DATOS INNER JOIN USUARIOS
ON EMPLEADO.ID_EMPLEADO = USUARIOS.ID_EMPLEADO
WHERE EMPLEADO.ID_EMPLEADO = @ID_EMPLEADO AND USUARIOS.ID_PUESTO = 1 )
insert into ADMINISTRADOR values (@NOMBRE_ADMIN, @APELLIDO_ADMIN, @ID_EMPLEADO)
END
go
```

El **TRIGGER registrar_admin** se activa en la tabla **USUARIOS** cuando se realiza una inserción. Su propósito es verificar si el usuario insertado tiene un puesto con un rango específico (rango 1, correspondiente a administradores). Primero, obtiene el **ID_PUESTO** del usuario insertado y, a continuación, consulta la tabla **PUESTOS** y **RANGOS** para obtener el **ID_RANGO** asociado con el puesto. Si el **ID_RANGO** es 1 (es decir, si es un administrador), entonces obtiene los datos del empleado correspondientes (nombre, apellido) de la tabla **DATOS_SOLICITANTE**, **EMPLEADO** y **USUARIOS**. Finalmente, inserta un nuevo registro en la tabla **ADMINISTRADOR**, asociando el **ID_EMPLEADO**, el **NOMBRE_ADMIN** y el **APELLIDO_ADMIN** del empleado insertado. Esto asegura que solo los usuarios con rango de administrador sean registrados automáticamente en la tabla **ADMINISTRADOR**.



Figura 37 Trigger de Historial de Usuarios Creados

```
---Historial de Usuarios
create trigger Historial_Usuarios_Creados
on Usuarios
for insert
as
declare @ID_Usuario int=(select ID_USUARIO from inserted)
declare @USUARIO VARCHAR(30)=(select USUARIO from inserted)
declare @CONTRASENA NVARCHAR(100)=(select CONTRASENA from inserted)
declare @ID_EMPLEADO int=(select ID_EMPLEADO from inserted)
declare @ID_PUESTO int=(select ID_PUESTO from inserted)
insert into [HISTORIAL_USUARIOS] values (@ID_Usuario, @USUARIO, @CONTRASENA,
@ID_EMPLEADO, @ID_PUESTO, GETDATE())
go
```

El **TRIGGER Historial_Usuarios_Creados** se ejecuta en la tabla **Usuarios** cada vez que se realiza una inserción, y su objetivo es registrar el historial de creación de usuarios en la tabla **HISTORIAL_USUARIOS**. Al momento de insertar un nuevo usuario, el **TRIGGER** extrae los siguientes valores: **ID_USUARIO**, **USUARIO**, **CONTRASENA**, **ID_EMPLEADO**, y **ID_PUESTO** de la fila recién insertada. Posteriormente, inserta un nuevo registro en la tabla **HISTORIAL_USUARIOS**, almacenando estos valores junto con la fecha y hora actuales obtenidas mediante la función **GETDATE()**. Esto garantiza que toda creación de un usuario quede registrada en el historial, permitiendo un seguimiento de las acciones realizadas sobre la tabla **USUARIOS**.



Figura 38 Trigger de Solicitante Contratado

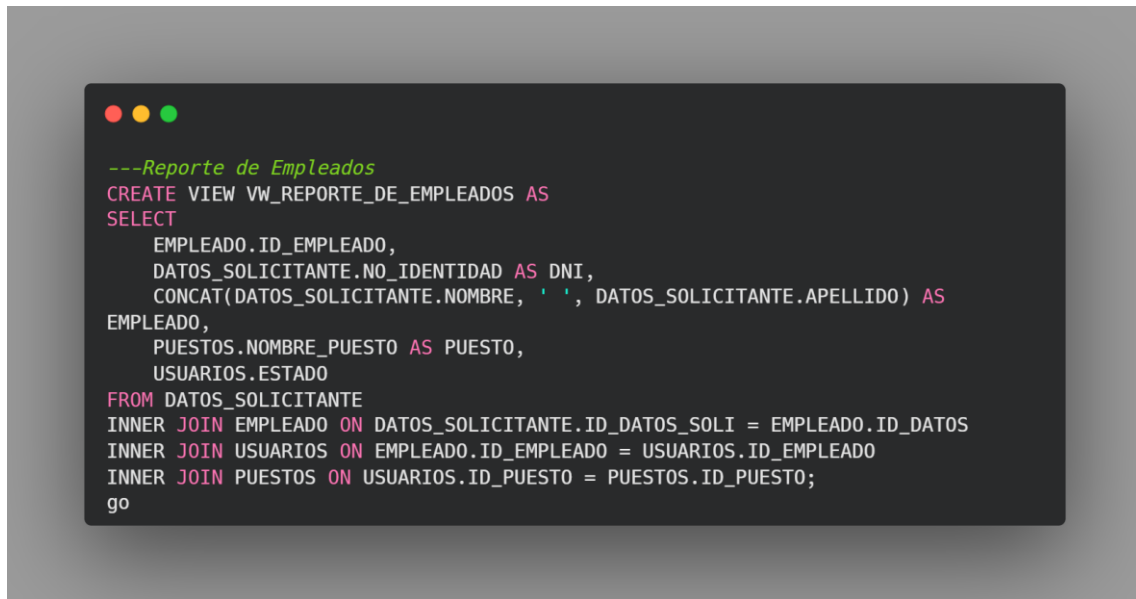


El **TRIGGER Solicitante_Contratado** se activa en la tabla **Empleado** cada vez que se inserta un nuevo registro. Su objetivo es actualizar el estado del solicitante en la tabla **DATOS_SOLICITANTE** a **Contratado** cuando se agrega un nuevo empleado. El **TRIGGER** obtiene el **ID_DATOS** del registro recién insertado en la tabla **Empleado**, y luego actualiza el campo **ESTADO** de la tabla **DATOS_SOLICITANTE** para reflejar que el solicitante ha sido contratado, basándose en la relación entre el **ID_DATOS_SOLI** y el **ID_DATOS**. Esto asegura que, al ser contratado un solicitante, su estado se actualice automáticamente a **Contratado** en el sistema.



Vistas

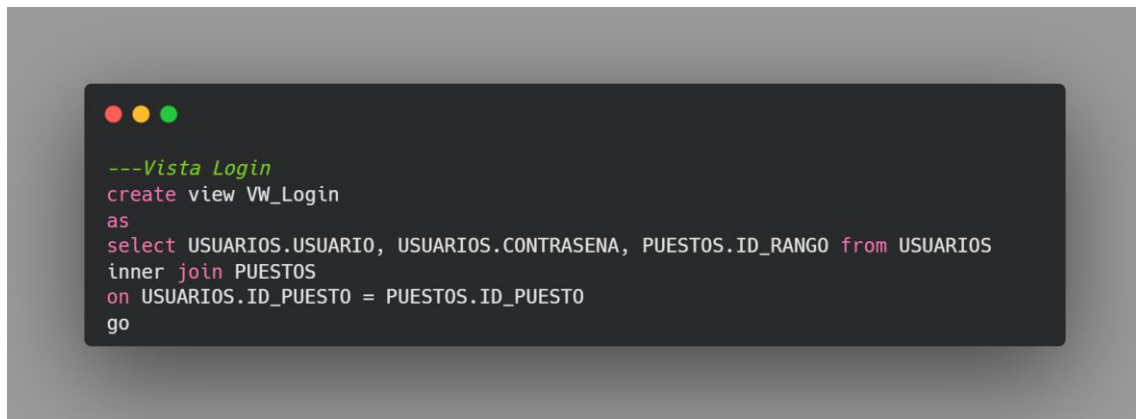
Figura 39 Vista para Reporte de Empleados



La vista **VW_REPORTES_DE_EMPLEADOS** combina información de las tablas **DATOS_SOLICITANTE**, **EMPLEADO**, **USUARIOS** y **PUESTOS** para generar un informe detallado de los empleados en el sistema. Este informe incluye el **ID_EMPLEADO**, el **DNI** del solicitante (obtenido de **DATOS_SOLICITANTE**), el **nombre completo** del empleado (concatenando nombre y apellido de la misma tabla), el **nombre del puesto** del empleado (proporcionado por **PUESTOS**) y el **estado** del usuario (activo o inactivo, proveniente de **USUARIOS**). Esta vista facilita la obtención de un reporte integral sobre los empleados, optimizando la consulta de datos esenciales sobre su identidad, puesto y estado en el sistema, y permite una visualización centralizada y eficiente sin la necesidad de realizar consultas complejas repetidas.



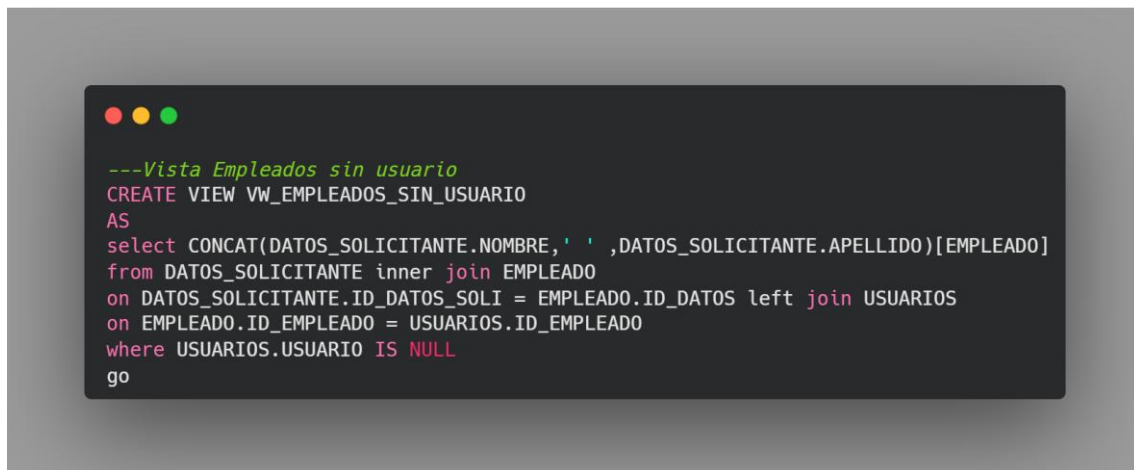
Figura 40 Vista de Inicio de Sesión



La vista **VW_Login** proporciona una consulta simplificada para la validación de acceso de los usuarios al sistema. Esta vista recupera los campos **USUARIO** y **CONTRASENA** de la tabla **USUARIOS**, junto con el **ID_RANGO** del puesto correspondiente, obteniendo esta información a través de una **unión interna** con la tabla **PUESTOS**. Esta estructura facilita la autenticación, permitiendo verificar el usuario y la contraseña, así como el nivel de acceso o rango asociado al puesto del usuario, lo que es fundamental para implementar controles de seguridad y autorización dentro del sistema.



Figura 41 Vista de Empleados sin Usuario



La vista **VW_Login** proporciona una consulta simplificada para la validación de acceso de los usuarios al sistema. Esta vista recupera los campos **USUARIO** y **CONTRASEÑA** de la tabla **USUARIOS**, junto con el **ID_RANGO** del puesto correspondiente, obteniendo esta información a través de una **unión interna** con la tabla **PUESTOS**. Esta estructura facilita la autenticación, permitiendo verificar el usuario y la contraseña, así como el nivel de acceso o rango asociado al puesto del usuario, lo que es fundamental para implementar controles de seguridad y autorización dentro del sistema.



Figura 42 Vista de Registrar Solicitante

```
---Vista de registrar solicitante
CREATE VIEW VW_REGISTRAR_SOLICITANTE
AS
SELECT DATOS_SOLICITANTE.NO_IDENTIDAD, DATOS_SOLICITANTE.RTN,
DATOS_SOLICITANTE.NOMBRE, DATOS_SOLICITANTE.APELLIDO,
DATOS_SOLICITANTE.FECHA_NACIMIENTO, DATOS_SOLICITANTE.FORMACION_ACADEMICA,
DATOS_SOLICITANTE.EXPERIENCIA_LABORAL,
DATOS_SOLICITANTE.IDIOMAS, DATOS_SOLICITANTE.CURSOS, DATOS_SOLICITANTE.ESTADO
from DATOS_SOLICITANTE
WHERE ESTADO = 'Pendiente'
go
```

La vista **VW_REGISTRAR_SOLICITANTE** permite consultar los detalles de los solicitantes cuyo estado es **Pendiente**. Esta vista extrae información relevante de la tabla **DATOS_SOLICITANTE**, incluyendo el **NO_IDENTIDAD**, **RTN**, **NOMBRE**, **APELLIDO**, **FECHA_NACIMIENTO**, **FORMACION_ACADEMICA**, **EXPERIENCIA_LABORAL**, **IDIOMAS**, **CURSOS** y el **ESTADO**. Al filtrar solo aquellos solicitantes con el estado **Pendiente**, facilita la gestión y revisión de solicitudes aún no procesadas, permitiendo a los administradores revisar los datos y tomar decisiones antes de proceder con el proceso de contratación.



Figura 43 Vista de Reporte de Vacaciones

```
---Vista del reporte de vacaciones
CREATE VIEW VW_REPORTER_VACACIONES
AS
SELECT SOLICITUD_VACACIONES.ID_SOLICITUD_V[CODIGO SOLICITUD],
DATOS_SOLICITANTE.NO_IDENTIDAD[DNI],
CONCAT(DATOS_SOLICITANTE.NOMBRE, ' ', DATOS_SOLICITANTE.APELLIDO)[EMPLEADO],
CONCAT(SOLICITUD_VACACIONES.DIAS_SOLICITADOS, ' ', '(' ,
SOLICITUD_VACACIONES.FECHA_INICIO,
' ----> ', SOLICITUD_VACACIONES.FECHA_FINAL, ')')[DIAS SOLICITADOS],
SOLICITUD_VACACIONES.ESTADO[ESTADO] from SOLICITUD_VACACIONES INNER JOIN EMPLEADO
ON SOLICITUD_VACACIONES.ID_EMPLEADO = EMPLEADO.ID_EMPLEADO INNER JOIN
DATOS_SOLICITANTE
ON EMPLEADO.ID_DATOS = DATOS_SOLICITANTE.ID_DATOS_SOLI
go
```

La vista **VW_REPORTER_VACACIONES** genera un reporte detallado de las solicitudes de vacaciones, proporcionando una visión clara de las solicitudes realizadas por los empleados. Esta vista extrae la información de las tablas **SOLICITUD_VACACIONES**, **EMPLEADO** y **DATOS_SOLICITANTE** para mostrar el **CODIGO SOLICITUD** (ID de la solicitud), el **DNI** del solicitante, el nombre completo del **EMPLEADO**, los **DIAS SOLICITADOS** junto con el rango de fechas, y el **ESTADO** de cada solicitud. La concatenación de los días solicitados con las fechas de inicio y fin de la solicitud proporciona una representación clara y comprensible del periodo solicitado. Este reporte es útil para realizar seguimientos y auditorías de las solicitudes de vacaciones dentro de la organización.



Figura 44 Vista de Reporte de Puestos

```
---Vista del reporte de puestos
CREATE VIEW Vista_Reporte_Puestos AS
SELECT
    Puestos.ID_PUESTO AS ID,
    Puestos.NOMBRE_PUESTO AS NOMBRE,
    Puestos.DESCRIPCION_PUESTO AS DESCRIPCION,
    Puestos.SUELDO AS SUELDO,
    COUNT(Usuarios.ID_USUARIO) AS [CANTIDAD EMPLEADOS]
FROM REAUTO.dbo.PUESTOS AS Puestos
LEFT JOIN REAUTO.dbo.USUARIOS AS Usuarios
    ON Puestos.ID_PUESTO = Usuarios.ID_PUESTO Where USUARIOS.ESTADO = 'Activo'
GROUP BY
    Puestos.ID_PUESTO,
    Puestos.NOMBRE_PUESTO,
    Puestos.DESCRIPCION_PUESTO,
    Puestos.SUELDO
go
```

La vista **Vista_Reporte_Puestos** proporciona un informe detallado sobre los puestos dentro de la organización, incluyendo el **ID**, **NOMBRE**, **DESCRIPCIÓN** y **SUELDO** de cada puesto, además de contar la cantidad de empleados asociados a cada uno. Utilizando una combinación de la tabla **PUESTOS** y **USUARIOS**, esta vista filtra a aquellos usuarios cuyo **ESTADO** es **Activo**, asegurando que solo se incluyan empleados actualmente activos en la organización. La información se agrupa por los atributos clave del puesto para ofrecer una visión clara de la distribución de los empleados por puesto dentro de la estructura organizacional. Esta vista es útil para el análisis de la cantidad de empleados en cada puesto y la evaluación de la distribución de roles dentro de la empresa.



TALLER INDUSTRIAL REAUTO MANUAL TÉCNICO

Código Fuente

<https://github.com/itsxMasato/REAUTO-APP>