Sistema de Gestión de Reclutamiento de RRHH

Entrega 1: Sistema de Gestion de Reclutamiento en RRHH

Autor: Laura Manuela Pareja Arango

Curso: SQL Flex

Introducción

El presente proyecto tiene como finalidad diseñar y desarrollar una base de datos relacional orientada a la gestión integral de los procesos de reclutamiento y selección de personal en una organización. Este sistema permitirá almacenar, organizar y consultar eficientemente información clave sobre candidatos, vacantes, entrevistas y evaluaciones. La implementación de esta base de datos busca optimizar las operaciones del área de Recursos Humanos, reducir errores en la administración de postulaciones y facilitar la toma de decisiones en torno a la contratación de nuevos talentos.

Objetivo

El objetivo general del proyecto es implementar una base de datos robusta, escalable y normalizada que permita:

- Registrar datos personales de los candidatos.
- Gestionar las vacantes laborales disponibles.
- Rastrear el estado de cada postulación en tiempo real.
- Documentar entrevistas realizadas a los postulantes.
- Evaluar objetivamente el desempeño de los candidatos.

Todo esto con el fin de agilizar los procesos de selección, reducir tiempos de respuesta y aumentar la calidad de las contrataciones dentro de la empresa.

Modelo de Negocio

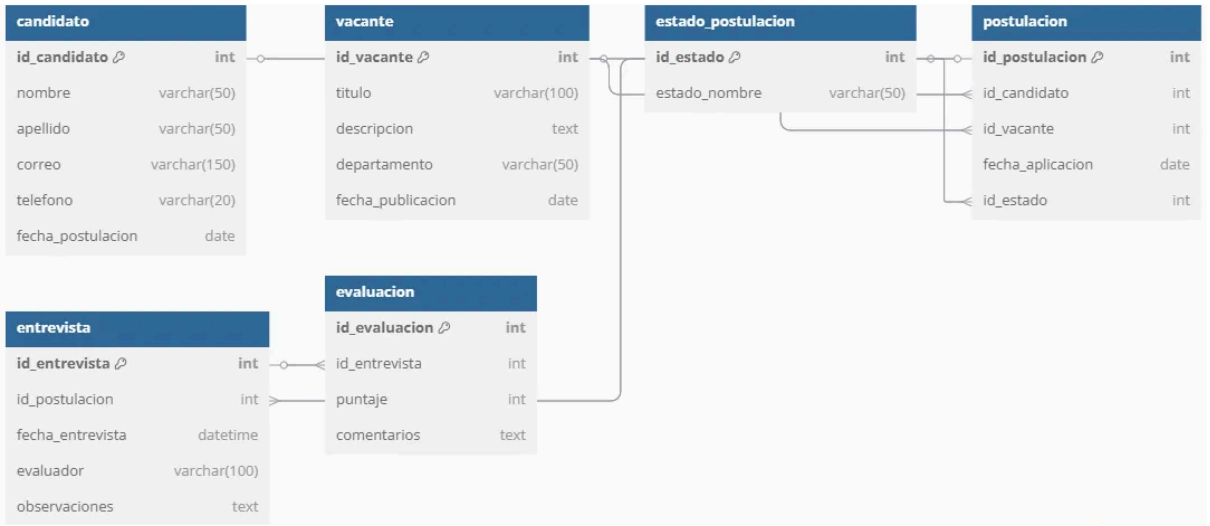
El sistema está diseñado para empresas medianas y grandes que manejan un volumen significativo de postulaciones. Estas organizaciones cuentan con un departamento de Recursos Humanos responsable de reclutar personal para diferentes áreas. Actualmente, muchas gestionan esta información de forma manual o descentralizada, lo que genera duplicidad de datos, pérdida de información y lentitud en los procesos. Con esta base de datos, se busca centralizar toda la gestión del reclutamiento, mejorar la trazabilidad de los candidatos y facilitar reportes para la toma de decisiones estratégicas.

Diagrama de Entidad-Relación (ER)

Este diagrama permite entender cómo están conectados los elementos principales del sistema, como los candidatos, vacantes, entrevistas y evaluaciones, mediante relaciones lógicas.

1. **Candidato**
    - Puede postularse a múltiples vacantes.
    - Está vinculado a varias entrevistas si participa en distintos procesos.
2. **Vacante**
    - Puede recibir múltiples postulaciones.
    - Se asocia a un conjunto de procesos de selección y entrevistas.

3. **Estado de Postulación**
    - Define las etapas del proceso (en revisión, entrevistado, rechazado, aprobado).
    - Cada postulación solo puede tener un estado activo a la vez.
4. **Postulación**
    - Vincula a un candidato con una vacante específica.
    - Se relaciona con entrevistas que pueden evaluarse posteriormente.
5. **Entrevista**
    - Pertenece a una postulación.
    - Puede ser evaluada individualmente.
6. **Evaluación**
    - Es el resultado de una entrevista.
    - Incluye puntaje y observaciones del evaluador.



Listado de Tablas y Descripción

1. Candidatos

Tabla que almacena los datos personales de los postulantes.

Campo	Descripción	Tipo de Dato	Clave
id_candidato	Identificador único del candidato	INT	Primaria
nombre	Nombre del candidato	VARCHAR(50)	
apellido	Apellido del candidato	VARCHAR(50)	
correo	Correo electrónico (único)	VARCHAR(150)	Única
telefono	Teléfono de contacto	VARCHAR(20)	
fecha_postulacion	Fecha de registro de postulación	DATE	

2. Vacantes

Contiene información sobre los puestos de trabajo disponibles.

Campo	Descripción	Tipo de Dato	Clave
id_vacante	Identificador único del puesto	INT	Primaria
titulo	Nombre del cargo ofertado	VARCHAR(100)	
descripcion	Descripción detallada del puesto	TEXT	
departamento	Área o departamento asociado	VARCHAR(50)	
fecha_publicacion	Fecha de publicación de la vacante	DATE	

3. Estado de Postulación

Define los diferentes estados que puede tener una postulación.

Campo	Descripción	Tipo de Dato	Clave
id_estado	Identificador del estado	INT	Primaria
estado_nombre	Nombre del estado (Ej. Aprobado)	VARCHAR(50)	

4. Postulación

Registra la relación entre candidatos y vacantes.

Campo	Descripción	Tipo de Dato	Clave
id_postulacion	Identificador único	INT	Primaria
id_candidato	Candidato que aplica	INT	Foránea
id_vacante	Vacante a la que aplica	INT	Foránea
fecha_aplicacion	Fecha de la postulación	DATE	

id_estado	Estado actual de la postulación	INT	Foránea

5. Entrevistas

Información sobre entrevistas realizadas a postulantes.

Campo	Descripción	Tipo de Dato	Clave
id_entrevista	Identificador de la entrevista	INT	Primaria
id_postulacion	Relación con la postulación	INT	Foránea
fecha_entrevista	Fecha y hora de la entrevista	DATETIME	
evaluador	Nombre del entrevistador	VARCHAR(100)	
observaciones	Comentarios del entrevistador	TEXT	

6. Evaluaciones

Contiene los resultados de las entrevistas.

Campo	Descripción	Tipo de Dato	Clave
id_evaluacion	Identificador único de la evaluación	INT	Primaria
id_entrevista	Relación con la entrevista	INT	Foránea
puntaje	Puntaje obtenido (0-100)	INT	
comentarios	Observaciones del evaluador	TEXT	

Script SQL en GitHub

El script de creación de la base de datos esta en del siguiente enlace:

https://github.com/LauraManu15/SistemadeGesti-ndeReclutamientodeRRHH_Pareja/blob/main/README.md

Entrega 2: Proyecto Sistema de Gestión de Reclutamiento de RRHH

1. Listado de Vistas

Vista 1: `vista_postulaciones_detalle`

- **Descripción:** Muestra información consolidada de cada postulación con datos del candidato, la vacante y el estado actual.
- **Objetivo:** Facilitar la consulta rápida y completa de las postulaciones sin necesidad de hacer múltiples joins en consultas repetitivas.
- **Tablas involucradas:** `postulacion`, `candidato`, `vacante`, `estado_postulacion`.

```
CREATE OR REPLACE VIEW vista_postulaciones_detalle AS
SELECT
    p.id_postulacion,
    c.nombre,
    c.apellido,
    c.correo,
    v.titulo AS puesto,
    v.departamento,
    e.estado_nombre,
    p.fecha_aplicacion
FROM postulacion p
JOIN candidato c ON p.id_candidato = c.id_candidato
JOIN vacante v ON p.id_vacante = v.id_vacante
JOIN estado_postulacion e ON p.id_estado = e.id_estado;
```

Vista 2: `vista_entrevistas_evaluacion`

- **Descripción:** Presenta la información de las entrevistas junto con la evaluación obtenida y comentarios del evaluador.
- **Objetivo:** Mostrar la evaluación del candidato tras la entrevista para seguimiento y toma de decisiones.

- **Tablas involucradas:** `entrevista`, `evaluacion`, `postulacion`, `candidato`.

```
CREATE OR REPLACE VIEW vista_entrevistas_evaluacion AS
SELECT
    ent.id_entrevista,
    c.nombre,
    c.apellido,
    ent.fecha_entrevista,
    ent.evaluador,
    ent.observaciones AS observaciones_entrevista,
    ev.puntaje,
    ev.comentarios AS comentarios_evaluacion
FROM entrevista ent
JOIN evaluacion ev ON ent.id_entrevista = ev.id_entrevista
JOIN postulacion p ON ent.id_postulacion = p.id_postulacion
JOIN candidato c ON p.id_candidato = c.id_candidato;
```

---

2. Listado de Funciones

- **Descripción:** Calcula el puntaje promedio de las evaluaciones para un candidato dado.
- **Objetivo:** Facilitar la obtención de un promedio general del desempeño de un candidato en sus entrevistas.
- **Tablas involucradas:** `evaluacion`, `entrevista`, `postulacion`.

```
DELIMITER$$

CREATE FUNCTION fn_calcular_promedio_puntaje(p_id_candidato INT)
RETURNS FLOAT
DETERMINISTIC
BEGIN
    DECLARE promedio FLOAT;
    SELECT AVG(ev.puntaje) INTO promedio
    FROM evaluacion ev
    JOIN entrevista ent ON ev.id_entrevista = ent.id_entrevista
    JOIN postulacion p ON ent.id_postulacion = p.id_postulacion
```

```
    WHERE p.id_candidato = p_id_candidato;
    RETURN IFNULL(promedio, 0);
END$$
```

Función 2: `fn_contar_postulaciones_estado`

- **Descripción:** Retorna el número de postulaciones que están en un estado específico (Ejemplo: 'En revisión').
- **Objetivo:** Permitir reportes rápidos del número de postulaciones según su estado.
- **Tablas involucradas:** `postulacion`, `estado_postulacion`.

```
CREATE FUNCTION fn_contar_postulaciones_estado(p_estado_nombre VA
RCHAR(50))
RETURNS INT
DETERMINISTIC
BEGIN
    DECLARE cantidad INT DEFAULT 0;

    SELECT COUNT(*) INTO cantidad
    FROM postulacion p
    JOIN estado_postulacion e ON p.id_estado = e.id_estado
    WHERE e.estado_nombre = p_estado_nombre;

    RETURN cantidad;
END $$

DELIMITER ;
```

Stored Procedure 1: `sp_actualizar_estado_postulacion`

- **Descripción:** Actualiza el estado de una postulación específica.

- **Objetivo:** Automatizar el cambio de estado de la postulación, manteniendo integridad y seguimiento.
- **Tablas involucradas:** `postulacion`.

```
DELIMITER $$

CREATE PROCEDURE sp_actualizar_estado_postulacion(
    IN p_id_postulacion INT,
    IN p_id_estado INT
)
BEGIN
    UPDATE postulacion
    SET id_estado = p_id_estado
    WHERE id_postulacion = p_id_postulacion;
END $$
```

Stored Procedure 2: `sp_insertar_entrevista_con_evaluacion`

- **Descripción:** Inserta una entrevista y su evaluación asociada en una sola operación.
- **Objetivo:** Facilitar el registro integral del proceso de entrevista y evaluación de un candidato.
- **Tablas involucradas:** `entrevista`, `evaluacion`.

```
CREATE PROCEDURE sp_insertar_entrevista_con_evaluacion(
    IN p_id_postulacion INT,
    IN p_fecha_entrevista DATETIME,
    IN p_evaluador VARCHAR(100),
    IN p_observaciones TEXT,
    IN p_puntaje INT,
    IN p_comentarios TEXT
)
BEGIN
```

```
    DECLARE v_id_entrevista INT;

    INSERT INTO entrevista (id_postulacion, fecha_entrevista, evaluador, obs
ervaciones)
    VALUES (p_id_postulacion, p_fecha_entrevista, p_evaluador, p_observaci
ones);

    SET v_id_entrevista = LAST_INSERT_ID();

    INSERT INTO evaluacion (id_entrevista, puntaje, comentarios)
    VALUES (v_id_entrevista, p_puntaje, p_comentarios);
END $$

DELIMITER ;
```

4. Listado de Triggers

Trigger 1: `trg_postulacion_insert`

- **Descripción:** Al insertar una nueva postulación, asegura que el estado inicial sea 'En revisión'.
- **Objetivo:** Garantizar que toda postulación nueva comience con el estado correcto.
- **Tabla involucrada:** `postulacion`.

```
DELIMITER $$

CREATE TRIGGER trg_postulacion_insert
BEFORE INSERT ON postulacion
FOR EACH ROW
BEGIN
    IF NEW.id_estado IS NULL THEN
        SET NEW.id_estado = 1; -- Asigna estado 'En revisión' por defecto
    END IF;
END $$
```

```
DELIMITER ;
```