PROYECTO FINAL

Integrantes: Anny Santamaria, Johandys Crespo

Instituto Tecnológico de Informática - UTU

Base de Datos

Docente: Andres Alvarez

Julio 2025

ÍNDICE

INTRODUCCIÓN

En la actualidad, la gestión segura de contraseñas es un componente esencial en los sistemas informáticos debido al aumento de amenazas cibernéticas. Este proyecto tiene como objetivo desarrollar un sistema de gestión de contraseñas que permita registrar, verificar y generar contraseñas seguras, en cumplimiento con políticas de seguridad predefinidas. Utilizando una base de datos relacional (MySQL) como backend y Python como lenguaje de programación, se garantiza la integridad, trazabilidad y validación de las credenciales de acceso de los usuarios. Esta solución está orientada a organizaciones o usuarios individuales que buscan fortalecer sus medidas de autenticación.

REALIDAD PLANTEADA

Hoy en día, muchas personas utilizan contraseñas débiles o repetidas para múltiples servicios, lo que pone en riesgo la seguridad de sus cuentas. A nivel organizacional, no contar con un sistema que valide la fortaleza de las contraseñas y su adherencia a políticas puede derivar en accesos no autorizados o vulneraciones. Además, el uso de métodos tradicionales como el almacenamiento de contraseñas en texto plano representa un alto riesgo de exposición.

En este contexto, se detectó la necesidad de contar con un sistema automatizado que:

- Registre usuarios y contraseñas.
- Verifique la fuerza y cumplimiento de políticas de seguridad.
- Genere contraseñas seguras de forma personalizada,
- Almacene todo con medidas criptográficas (hash con SHA-256).

REQUERIMIENTOS

Requerimientos funcionales:

1. Registro de Usuarios
    - Permitir el registro de usuarios con nombre y correo electrónico.

2. Gestión de Contraseñas
    - El sistema debe permitir que un usuario cree y asocie múltiples contraseñas.
    - Almacenar contraseñas de forma segura mediante hashing (SHA-256 + salt).
    - Las contraseñas deben tener atributos como longitud, fecha de creación y valor hash.
3. Políticas de contraseña

    Verificar contraseñas ingresadas según una política activa de seguridad:

    - Longitud mínima y máxima,

    - Inclusión de mayúsculas, números y caracteres especiales.
4. Validación de seguridad
    - El sistema debe verificar que las contraseñas nuevas cumplan con la política asignada al usuario.
    - Calcular el nivel de fuerza de una contraseña.

    - En caso de que no cumpla, debe rechazarse con un mensaje indicando los errores.

5. Generar contraseñas seguras personalizadas.

6. Asociar cada contraseña registrada al usuario correspondiente.

7. Mantener registros históricos en las tablas **usuario**, **contraseña**, **politica_contraseña**, **ingresa** y **nivel_seguridad**.

Requerimientos no funcionales:

1. Utilizar MySQL como sistema gestor de base de datos.
2. El sistema debe ser ejecutable en entorno de consola Python.
3. La aplicación debe ofrecer retroalimentación clara al usuario sobre el estado de su contraseña.
4. Garantizar integridad referencial en las relaciones entre tablas mediante claves foráneas.
5. Cumplir con las restricciones estructurales y no estructurales definidas en el modelo entidad-relación.

CÓDIGO FUENTE DEL PROGRAMA

```
import hashlib
import string
import random
from datetime import date
import mysql.connector

# Configura tu conexión MySQL
conexion = mysql.connector.connect(
    host="localhost",
    user="root",
    password="Johacrespo1.",        # ← cambia esto
    database="JYA"
)
cursor = conexion.cursor()

# Función para hashear la contraseña
def hashear_contraseña(contraseña):
    salt = contraseña[::-1]
    return hashlib.sha256((contraseña + salt).encode()).hexdigest()

# Verifica fuerza de contraseña
def nivel_fuerza(contraseña):
    criterios = [
        len(contraseña) >= 8,
        any(c.isupper() for c in contraseña),
        any(c.islower() for c in contraseña),
        any(c.isdigit() for c in contraseña),
        any(c in string.punctuation for c in contraseña)
    ]
    puntos = sum(criterios)

    if puntos == 5:
        return "Muy fuerte"
    elif puntos == 4:
        return "Fuerte"
    elif puntos == 3:
        return "Moderada"
```

```
    else:
        return "Débil"

# Verifica si cumple política
def verificar_politica(contraseña):
    return (
        len(contraseña) >= 8 and len(contraseña) <= 16 and
        any(c.isupper() for c in contraseña) and
        any(c.isdigit() for c in contraseña) and
        any(c in string.punctuation for c in contraseña)
    )

# Inserta usuario en BD
def insertar_usuario(nombre, correo):
    fecha = date.today()
    cursor.execute("INSERT INTO usuario (nombre, correo, fecha_registro) VALUES (%s, %s,
%s)", (nombre, correo, fecha))
    conexion.commit()
    return cursor.lastrowid

# Inserta contraseña y relación en BD
def registrar_contraseña(id_usuario, contraseña):
    hash_pwd = hashear_contraseña(contraseña)
    fecha = date.today()
    cursor.execute("INSERT INTO contraseña (id_usuario, longitud, hash, fecha_creacion)
VALUES (%s, %s, %s, %s)",
                   (id_usuario, len(contraseña), hash_pwd, fecha))

    conexion.commit()
    id_contraseña = cursor.lastrowid

    cursor.execute("INSERT INTO ingresa (id_usuario, id_contraseña, fecha) VALUES (%s, %s,
%s)",
                       (id_usuario, id_contraseña, fecha))
    conexion.commit()

# Generador personalizado de contraseñas
def generar_contraseña_personalizada():
    print("\n=== Generador de Contraseña Segura Personalizada ===")
    try:
```

```
        longitud = int(input("Longitud deseada: "))
        mayus = input("¿Incluir mayúsculas? (s/n): ").lower() == 's'
        nums = input("¿Incluir números? (s/n): ").lower() == 's'
        simb = input("¿Incluir símbolos? (s/n): ").lower() == 's'

        caracteres = string.ascii_lowercase
        if mayus: caracteres += string.ascii_uppercase
        if nums: caracteres += string.digits
        if simb: caracteres += string.punctuation

        if longitud < 8 or not caracteres:
            print("Error: mínimo 8 caracteres y al menos un tipo seleccionado.")
            return

        contraseña = ''.join(random.choice(caracteres) for _ in range(longitud))
        print(f"Contraseña generada: {contraseña}")
        print(f"Nivel de fuerza: {nivel_fuerza(contraseña)}")
    except ValueError:
        print("Entrada inválida.")

# Menú principal
def mostrar_menu():
    print("\n=== MENÚ ===")
    print("1. Verificar y registrar contraseña")
    print("2. Generar contraseña segura personalizada")
    print("3. Salir")

# Función principal
def main():
    print("== Sistema de Gestión de Contraseñas ==")
    nombre = input("Nombre: ")
    correo = input("Correo: ")
    id_usuario = insertar_usuario(nombre, correo)
    print("Usuario registrado en la base de datos.")

    while True:
        mostrar_menu()
        opcion = input("Selecciona una opción: ")

        if opcion == "1":
```

```
            contraseña = input("Introduce tu contraseña: ")
            fuerza = nivel_fuerza(contraseña)
            print(f"Nivel de fuerza: {fuerza}")

            if verificar_politica(contraseña):
                registrar_contraseña(id_usuario, contraseña)
                print("Contraseña registrada en la base de datos.")
            else:
                print("Contraseña no cumple con la política.")

        elif opcion == "2":
            generar_contraseña_personalizada()

        elif opcion == "3":
            print("Saliendo del sistema...")
            break
        else:
            print("Opción inválida.")

    cursor.close()
    conexion.close()
if __name__ == "__main__":
    main()
```

MANUAL DEL PROGRAMA

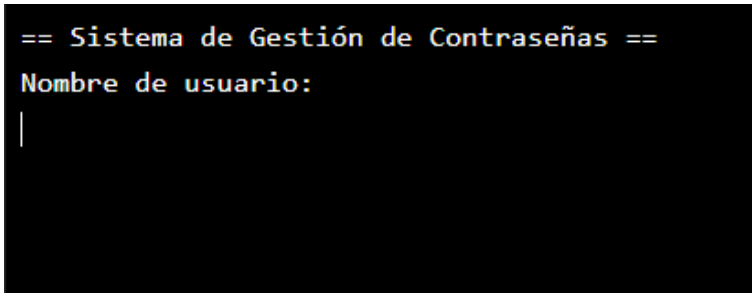
OBJETIVO

Este programa tiene como objetivo principal verificar, generar y registrar contraseñas de usuarios de acuerdo con una política de seguridad definida.

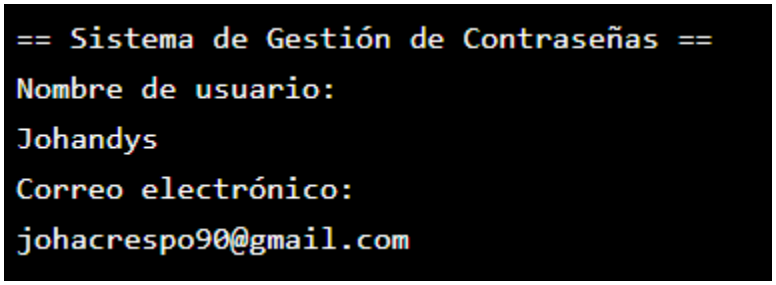
- Permite al usuario ingresar sus datos y una contraseña, la cual es evaluada según requisitos como longitud mínima, uso de mayúsculas, números y caracteres especiales.
- Además tienen la opción de que el programa les genere una contraseña segura personalizada.
- Si la contraseña cumple con las condiciones, se registra de forma segura utilizando un algoritmo de hash (SHA-256 con salt invertido).

REGISTRO DE USUARIO

Al iniciar, el programa le solicitará al usuario que ingrese su nombre.



Luego procede a pedirle un correo electrónico al usuario



MENÚ PRINCIPAL

se muestran 3 opciones

```
=== MENÚ PRINCIPAL ===
1. Verificar y registrar una contraseña
2. Generar contraseña segura personalizada
3. Salir
Selecciona una opción:
```

OPCIÓN 1: VERIFICAR Y REGISTRAR CONTRASEÑA:

Consiste en que el usuario ingrese una contraseña de su preferencia y el sistema verifica si la contraseña es segura, cumple con todas las políticas de seguridad y nivel de seguridad.

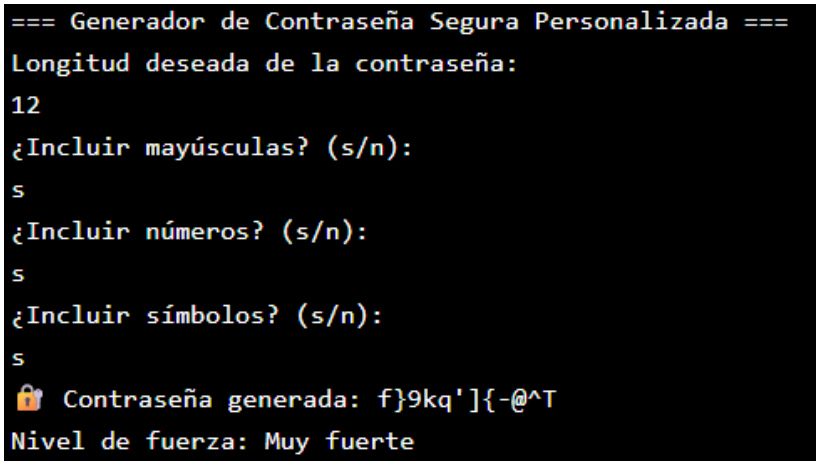
```
Selecciona una opción:
1
Introduce tu contraseña:
ANaperez10.
 Nivel de fuerza: Muy fuerte
verificación de política: Aceptada
✅ Contraseña registrada exitosamente.
```

Si no cumple con las políticas de seguridad y la contraseña es débil, no te registrara la contraseña

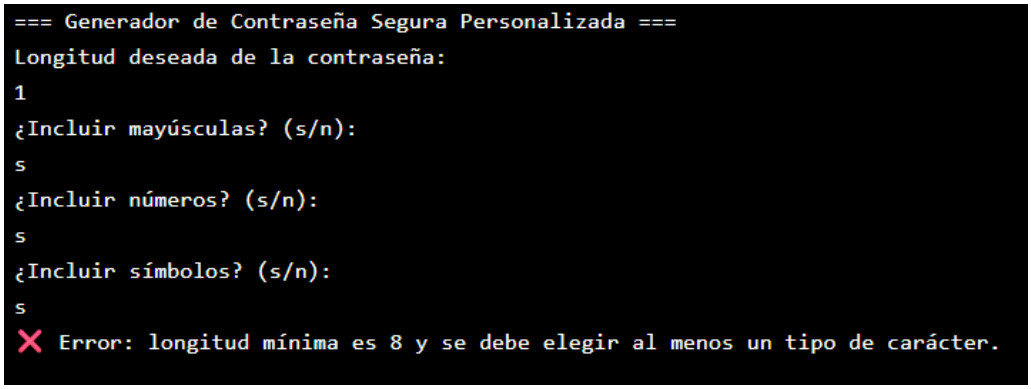
```
Selecciona una opción:
1
Introduce tu contraseña:
joha
 Nivel de fuerza: Débil
verificación de política: Rechazada: faltan mayúsculas
❌ Contraseña no registrada. No cumple con la política.
```

OPCIÓN 2: GENERAR CONTRASEÑA

El programa te genera una contraseña segura personalizada, donde el usuario puede ingresar una longitud (de acuerdo a la política de contraseña) y los tipos de caracteres.



Si el usuario ingresa al programa una longitud que está fuera de rango el programa no le genera la contraseña



OPCIÓN 3: SALIR

Finaliza la ejecución del programa

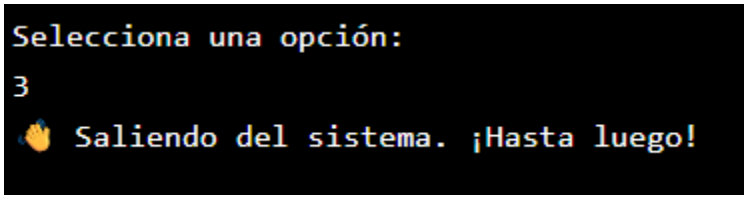
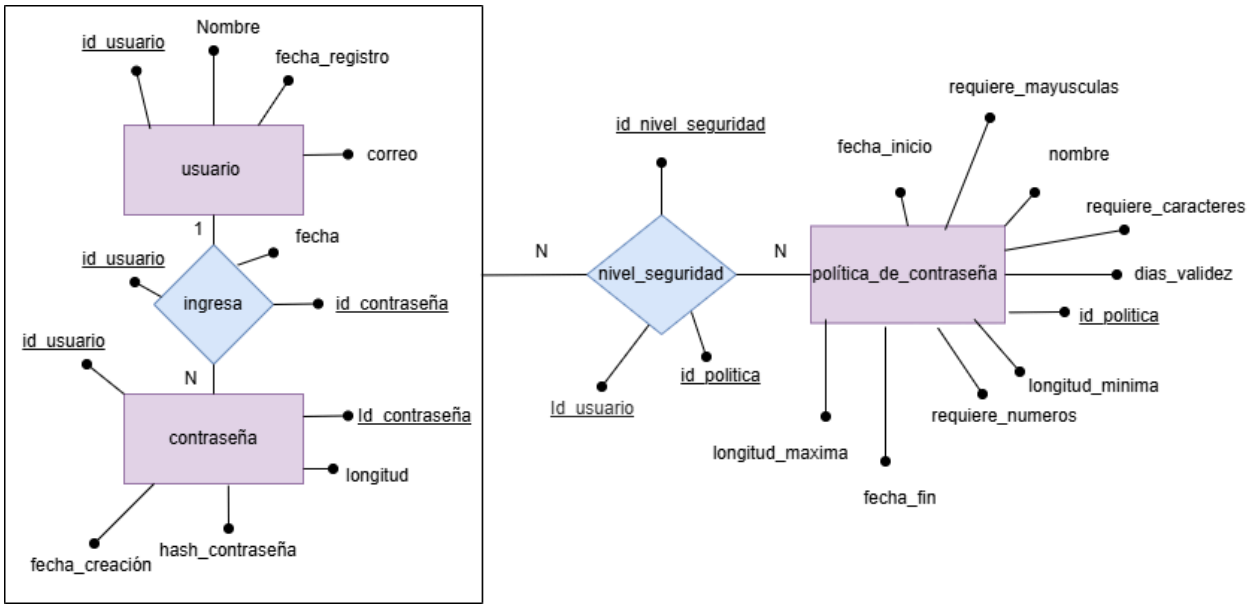


DIAGRAMA ENTIDAD RELACIÓN



RESTRICCIONES NO ESTRUCTURALES

- Política_de_contraseña: longitud_minima ≤ longitud_maxima.
- Política_de_contraseña: fecha_inicio < fecha_fin.
- Política_de_contraseña: requiere_mayusculas ∈ {0,1}
- Politica de contraseña: requiere_caracteres ∈ {0,1}
- Política_de_contraseña: requiere_numeros ∈ {0,1}
- Política_de_contraseña: dias_validez > 0.
- nivel_seguridad: un usuario no puede tener dos políticas activas simultáneamente.
- nivel_seguridad: cada política debe estar vigente entre fecha_inicio y fecha_fin cuando se aplique a un usuario.

ESQUEMA RELACIONAL EN 3FN

usuario (id_usuario (PK), nombre, correo, fecha_registro)

contraseña (id_contraseña (PK), id_usuario(FK), hash_contraseña, longitud, fecha_creacion)

politica_contraseña (id_politica (PK), nombre, longitud_maxima, longitud_minima requiere_mayusculas, fecha_inicio_ fecha_fin, requiere_numeros, dias_validez)|

ingresa (id_usuario (FK), id_contraseña (FK), fecha_ingreso)

nivel_seguridad (id_usuario (FK), id_politica(FK), id_nivel_seguridad (PK))

SENTENCIAS SQL

DDL

```
CREATE DATABASE JYA;
USE JYA;

-- Tabla: usuario
CREATE TABLE usuario (
    id_usuario INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    correo VARCHAR(100) NOT NULL,
    fecha_registro DATE NOT NULL
);

-- Tabla: contraseña
CREATE TABLE contraseña (
    id_contraseña INT AUTO_INCREMENT PRIMARY KEY,
    id_usuario INT NOT NULL,
    longitud INT NOT NULL,
    hash VARCHAR(256) NOT NULL,
    fecha_creacion DATE NOT NULL,
    FOREIGN KEY (id_usuario) REFERENCES usuario(id_usuario)
);


-- Relación: ingresa (usuario usa una contraseña)
CREATE TABLE ingresa (
     id_usuario INT,
    id_contraseña INT,
    fecha DATE NOT NULL,
    FOREIGN KEY (id_usuario) REFERENCES usuario(id_usuario),
    FOREIGN KEY (id_contraseña) REFERENCES contraseña(id_contraseña),
    PRIMARY KEY (id_usuario, id_contraseña, fecha)
);
```

-- Tabla: política de contraseña

```
   id_politica INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    longitud_minima INT NOT NULL,
    longitud_maxima INT NOT NULL,
    requiere_mayusculas BOOLEAN NOT NULL,
    requiere_numeros BOOLEAN NOT NULL,
    requiere_caracteres BOOLEAN NOT NULL,
    dias_validez INT NOT NULL,
    fecha_inicio DATE NOT NULL,
    fecha_fin DATE NOT NULL

);

-- Relación: nivel_seguridad (N:N entre usuario y política)
CREATE TABLE nivel_seguridad (
    id_nivel_seguridad INT AUTO_INCREMENT PRIMARY KEY,
    id_usuario INT,
    id_politica INT,
    FOREIGN KEY (id_usuario) REFERENCES usuario(id_usuario),
    FOREIGN KEY (id_politica) REFERENCES politica_contraseña(id_politica)
);
```

DML

-- Tabla: usuario (5 registros)

```
INSERT INTO usuario (nombre, correo, fecha_registro) VALUES
('Anny Santamaria', 'anny.santamaria@email.com', CURRENT_DATE),
('Johandys Crespo', 'johandys.crespo@email.com', CURRENT_DATE),
('Javier Medina', 'javier.medina@email.com', CURRENT_DATE),
('Laura Méndez', 'laura.mendez@email.com', CURRENT_DATE),
('Luis Torres', 'luis.torres@email.com', CURRENT_DATE);
```

-- Tabla: politica_contraseña

```
INSERT INTO politica_contraseña (
    id_politica, nombre, longitud_minima, longitud_maxima,
    requiere_mayusculas, requiere_numeros, requiere_caracteres,
    dias_validez, fecha_inicio, fecha_fin
) VALUES
(1, 'Política Estándar', 8, 16, TRUE, TRUE, TRUE, 90, CURRENT_DATE, DATE '2099-12-31'),
(2, 'Política Básica', 6, 12, FALSE, TRUE, FALSE, 60, CURRENT_DATE, DATE '2099-12-31'),
(3, 'Política Fuerte', 10, 20, TRUE, TRUE, TRUE, 120, CURRENT_DATE, DATE
'2099-12-31'),
(4, 'Política Moderada', 8, 16, TRUE, FALSE, TRUE, 45, CURRENT_DATE, DATE
'2099-12-31'),
(5, 'Política Laxa', 6, 20, FALSE, FALSE, FALSE, 30, CURRENT_DATE, DATE '2099-12-31');
```

-- Tabla: nivel_seguridad (5 registros)

```
INSERT INTO nivel_seguridad (id_usuario, id_politica) VALUES
(1, 1),
(2, 3),
(3, 2),
(4, 4),
(5, 5);
```

-- Tabla: contraseña (5 registros)

```
-- (Hash de ejemplo usando SHA-256, puedes cambiarlos por reales si lo deseas)
INSERT INTO contraseña (id_usuario, longitud, hash, fecha_creacion) VALUES
(1, 12, 'f8a8ab7ccde0cfa1570e03287cbf2b8792c7fe6eb5a0b0eac47c4c9cb8aa75ad',
CURRENT_DATE),
(2, 10, 'd4e8bc213cd546ad9d57413769e4b13f13e7769382f62f0a2faff2aaf6b9b94f',
CURRENT_DATE),
(3, 14, 'a7b9ef21cc19eada3ff6b7d2a3fc997991c3f9893b6d10595d59a2ddc6cbe3d4',
CURRENT_DATE),
(4, 11, 'd1a40d47bc32e72615aaf6ddc71dce9c9aa6c6e2c6e37d1386a7ae899d7de35c',
CURRENT_DATE),
```

```
(5, 13, 'abc97d4dbf95fc72b1930e8eab10aa377f5fae587af80a7603f4c1bcaa9c6cde',
CURRENT_DATE);

-- Tabla: ingresa (5 registros)

INSERT INTO ingresa (id_usuario, id_contraseña, fecha) VALUES
(1, 1, CURRENT_DATE),
(2, 2, CURRENT_DATE),
(3, 3, CURRENT_DATE),
(4, 4, CURRENT_DATE),
(5, 5, CURRENT_DATE);
```

Manipular la BD

Consultar a la BD

● Una consulta a su elección sobre datos de una tabla

```
SELECT nombre, correo, fecha_registro
FROM usuario
ORDER BY fecha_registro DESC;
```

● Una consulta a su elección que implemente algún tipo de join

```
SELECT
   u.nombre AS usuario,
   u.correo,
   pc.nombre AS politica,
   pc.longitud_minima,
   pc.requiere_mayusculas,
   pc.requiere_numeros,
   pc.requiere_caracteres
FROM usuario u
JOIN nivel_seguridad ns ON u.id_usuario = ns.id_usuario
JOIN politica_contraseña pc ON ns.id_nivel_seguridad = pc.id_politica;
```

Actualizar a la BD

● Una de cada una de las sentencias CRUD a su elección

CREATE

Con este comando se inserta un nuevo usuario.

```
INSERT INTO usuario (nombre, correo, fecha_registro)
VALUES ('Javier Medina', 'javier.medina@email.com', CURDATE());
```

READ

Aquí consultamos las contraseñas que se encuentran registradas.

```
SELECT * FROM contraseña;
```

UPDATE

El siguiente comando lo usamos para cambiar el correo de un usuario específico

```
UPDATE usuarios
SET correo = 'lucia.nueva@correo.com'
WHERE id_usuario = 6;
```

DELETE

Con este comando se puede eliminar la contraseña, en este caso la ID 5.

```
DELETE FROM politicas
WHERE id_politica = 5;
```

● Modificación que actúe sobre todos los valores de una columna.

Con este comando podemos modificar la longitud máxima de todas las políticas. Esto afecta todos los registros de la tabla políticas.

```
UPDATE politicas
SET longitud_max = 20;
```

Tipos de informes:

● Tabular

Muestra todas las politicas en formato tabla.

```
SELECT * FROM politicas;
```

● Resumen o agregado

Hace un promedio de longitud de contraseñas

```
SELECT AVG(longitud) AS promedio_longitud
FROM contraseñas;
```

● Detalle

Muestras la información detallada de los usurario y sus contraseñas

```
SELECT u.id_usuario, u.nombre, u.correo, c.id_contraseña, c.longitud, c.fecha_creacion
FROM usuarios u
JOIN ingresos i ON u.id_usuario = i.id_usuario
JOIN contraseñas c ON i.id_contraseña = c.id_contraseña;
```

● Comparativo

Compara el número de contraseñas por usuario

```
SELECT u.nombre, COUNT(i.id_contraseña) AS cantidad_contraseñas
FROM usuarios u
LEFT JOIN ingresos i ON u.id_usuario = i.id_usuario
GROUP BY u.id_usuario;
```