

SISTEMA WEB DE RESERVACIÓN DE VEHÍCULOS DE TRASTEIO

PackyGo

PRESENTADO POR:

LUYER SEBASTIAN PEREZ VARGAS

ALISON GINETH OSPINA ARIZA

DINA MILANIA MALAVER MONROY

FICHA 3147247

PRESENTADO A:

INSTRUCTORA CAROLINA FORERO

SERVICIO NACIONAL DE APRENDIZAJE SENA

TECNOLOGO EN ANALISIS Y DESARROLLO DE SOFTWARE

Diagrama de Clases

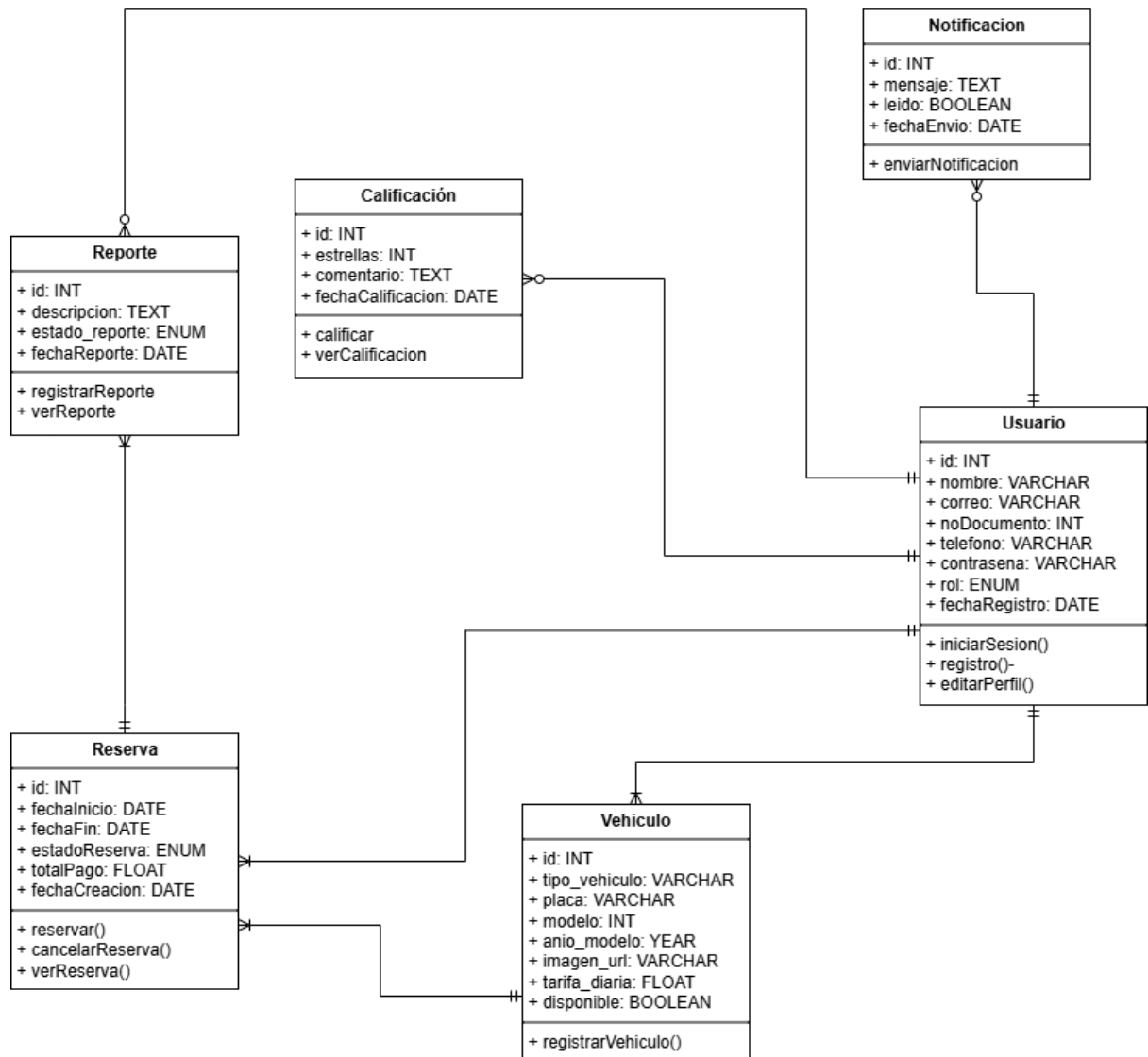


Diagrama relacional

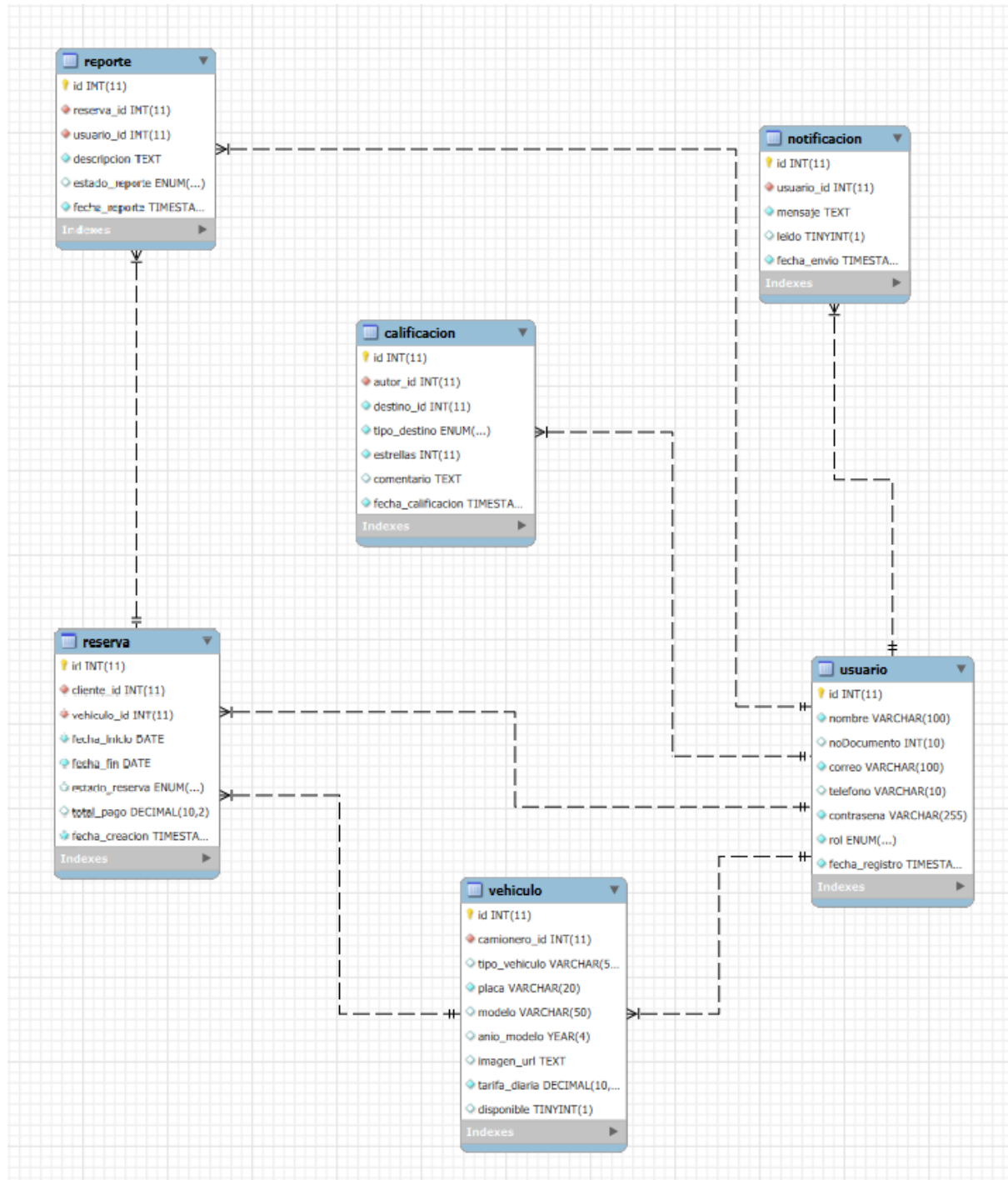
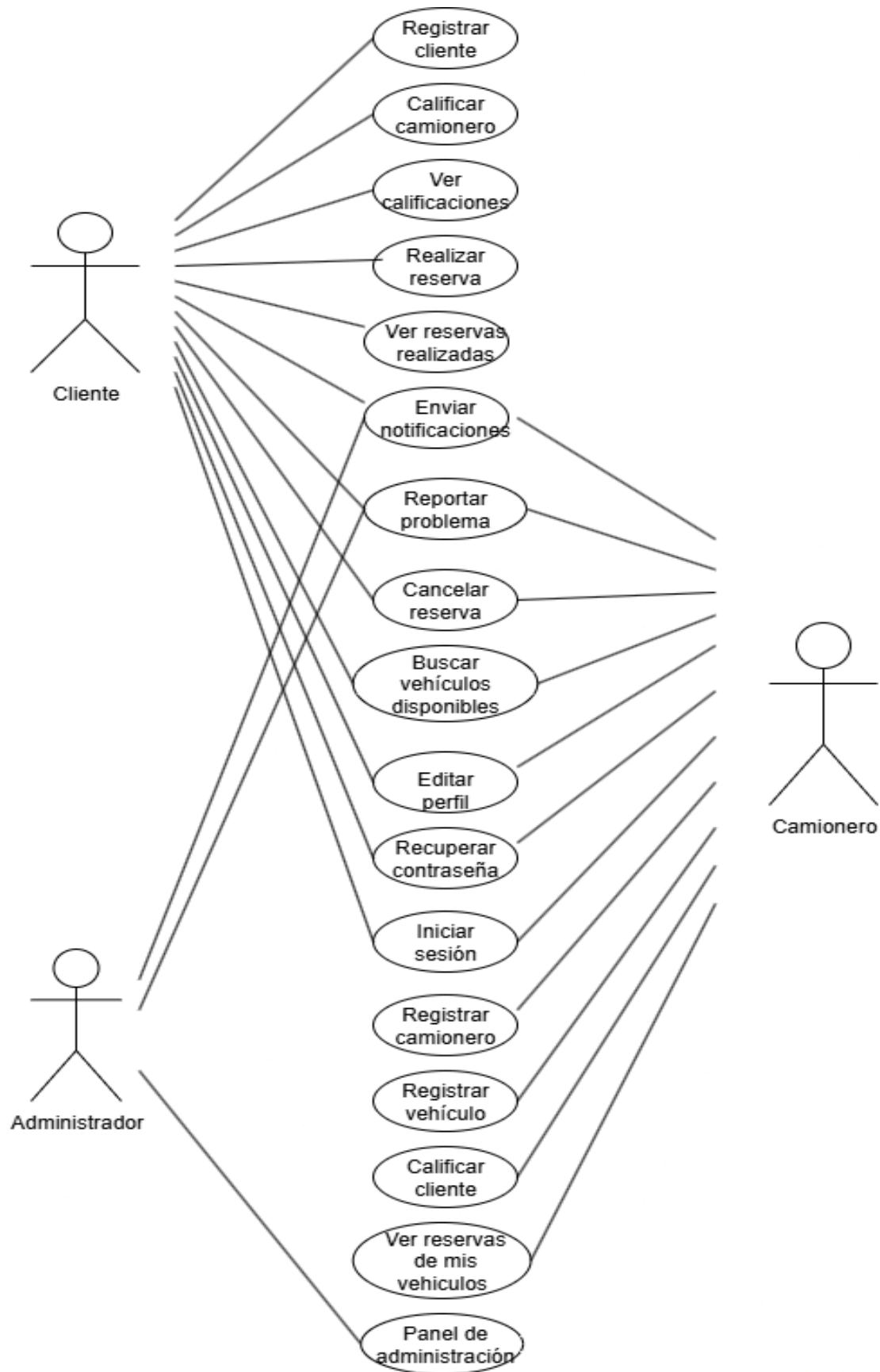


Diagrama Casos de Uso



Enlace de Mockups/Wireframes

<https://www.canva.com/design/DAGvyrOTe98/Ft9vyvdTayeTwD8QoBpJcg/edit>

Enlace al tablero ágil

<https://luyerperez.atlassian.net/jira/software/projects/SCRUM/boards/1/backlog?atlOrigin=eyJpIjoiMjI0NDI1YjRIZWJmNDIxZTk2ODFmYTlyYTc3NmYyM2MiLCJwIjoiajJ9>

Enlace a Backlog_Sprints.xlsx

https://docs.google.com/spreadsheets/d/1GXc5KA0jdvVJmIXBoVxRO4hRUlePHNQ/edit?usp=drive_link&oid=114581216777433290976&rtpof=true&sd=true

Enlace a plantillas de ceremonias.

<https://docs.google.com/spreadsheets/d/1K5KSEtG7--lg9oBWfggfZ5LqvHXu6Qlc/edit?usp=sharing&oid=114581216777433290976&rtpof=true&sd=true>

Enlace de bitácoras

<https://docs.google.com/spreadsheets/d/1fUGy0drYcoWx2YPQnNMiDsjDVbLGhXbR/edit?usp=sharing&oid=114581216777433290976&rtpof=true&sd=true>

Tabla de Comparación

Característica	MERN	LAPP
Tipo de base de datos	MongoDB (NoSQL, documentos JSON) → flexible, ideal para datos no estructurados.	PostgreSQL (SQL, relacional) → ideal para datos estructurados y relaciones complejas.
Escalabilidad	Muy escalable en entornos distribuidos gracias a MongoDB.	Escalable, pero mejor en vertical (aumentar capacidad de un solo servidor) que en horizontal.
Facilidad de aprendizaje	JavaScript en todo el stack → curva de aprendizaje más suave.	Requiere aprender Python + SQL (dos lenguajes distintos).
Desempeño en tiempo real	Excelente para webs en tiempo real (chat, tracking de vehículos) gracias a WebSockets en Node.js.	Puede manejar en tiempo real, pero requiere más configuración.
Manejo de datos complejos	Menos eficiente en datos muy relacionados (ej: reservas ↔ clientes ↔ vehículos).	Más eficiente para datos relacionales y reportes complejos.
Madurez y estabilidad	Stack moderno, muy popular pero cambia rápido.	Stack más maduro y estable en entornos corporativos.
Seguridad	Buenas librerías para JWT y control de acceso, pero requiere más cuidado en la validación de datos NoSQL.	Seguridad robusta de PostgreSQL y Apache, con manejo avanzado de permisos.
Flexibilidad	Muy flexible para prototipado rápido.	Más rígido pero con gran consistencia de datos.

Stack tecnológico del proyecto

Frontend

- Framework: [React.js](https://reactjs.org/)
 - Ventajas:
 - Alta velocidad y experiencia SPA (Single Page Application).
 - Gran comunidad y variedad de librerías.
 - Complementos recomendados:
 - Vite → compilación ultrarrápida.
 - Tailwind CSS → estilos responsive y modernos.
 - Axios → para consumir la API Flask.
 - React Router → rutas dinámicas.

Backend

- Lenguaje y Framework: Python + Flask
 - Ventajas:
 - Ligero y rápido de implementar.
 - Código limpio y fácil de mantener.
 - Compatible con API REST para conexión con React.
 - Complementos recomendados:
 - Flask-RESTful → para construir la API REST.
 - Flask-JWT-Extended → autenticación con tokens JWT.
 - Flask-CORS → permitir solicitudes desde el frontend React.
 - Flask-Migrate → para manejo de migraciones con SQLAlchemy.
 - SQLAlchemy → ORM para manejar MySQL de forma más sencilla.
 - Werkzeug → utilidades de seguridad y manejo de contraseñas.

Base de datos

- Motor: MySQL
 - Ventajas:
 - Ideal para datos relacionales (reservas, usuarios, vehículos, calificaciones).
 - Soporte robusto y rendimiento alto.
 - Diseño recomendado:
 - Tablas: Usuario, Vehículo, Reserva, Calificación, Reporte, Notificación.

Instalación de APPs

1. Git (Control de versiones)

<https://git-scm.com/downloads>

- **Windows:** Instalador `.exe`
- **macOS:** Paquete `.dmg` o Homebrew (`brew install git`)
- **Linux:** Uso de gestor de paquetes (`sudo apt install git` en Ubuntu)

2. Visual Studio Code (Editor de código)

<https://code.visualstudio.com/Download>

- Disponible para **Windows, macOS y Linux**.
- Se recomienda instalar extensiones como:
 - *Python* (para Flask y scripts)
 - *Prettier* (formateo de código)
 - *Docker* (si vas a contenerizar el proyecto)
- Librerías Flask y conexión MySQL:
 - `pip install flask flask-cors flask-mysql mysql-connector-python`
 - `pip install python-dotenv`
 - `pip install gunicorn # Para despliegue`

3. Python (Lenguaje backend con Flask)

<https://www.python.org/downloads/>

- Para Windows: Marca la casilla "Add Python to PATH" durante la instalación.
- Se recomienda la versión **3.10 o superior**.
- Después de instalar, verifica con:
`python --version`
- Luego instalar librerías necesarias:

```
pip install Flask
```

```
pip install Flask-RESTful Flask-SQLAlchemy
```

4. XAMPP (Servidor local para MySQL y Apache)

<https://www.apachefriends.org/download.html>

- Selecciona la versión que incluya **PHP 8+** y **MySQL**.
- En tu caso, usaremos solo **MySQL** desde XAMPP, ya que Flask será el backend.

5. Node.js + npm (para React)

<https://nodejs.org/en/download>

- Verificación:

```
node -v
```

```
npm -v
```

- Crear proyecto React:

```
npm create vite
```

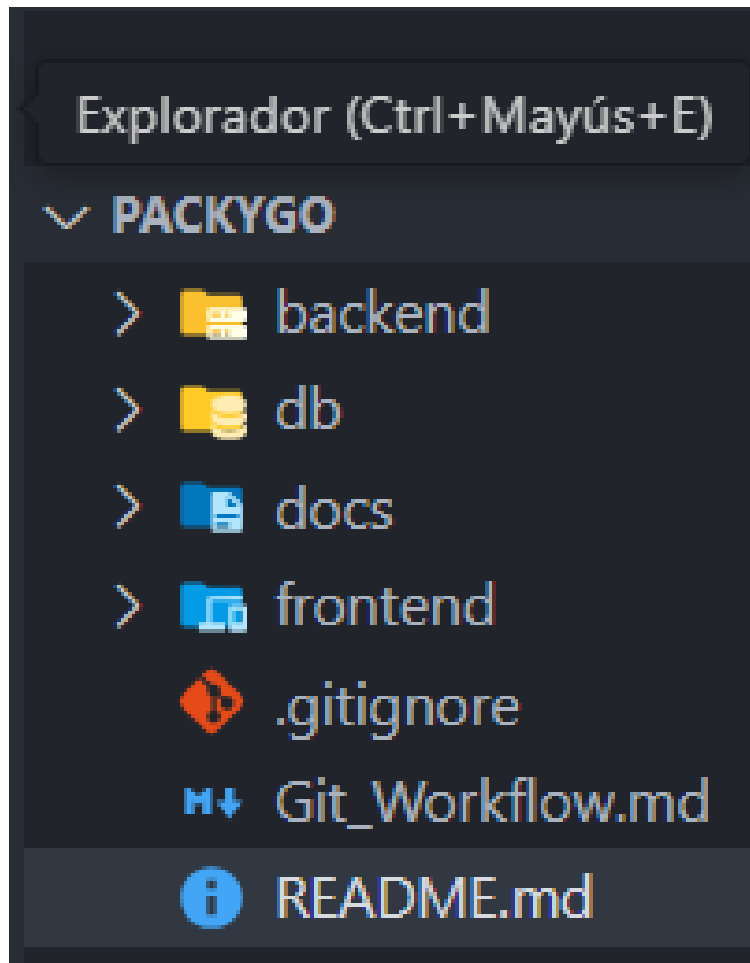
- Librerías recomendadas para React:

```
npm install axios react-router-dom bootstrap
```

Enlace de Repositorio

<https://github.com/LuyerPerez/PackGo.git>

Entorno Configurado



Enlace de Guia_EstandaresCodigo.md

https://github.com/LuyerPerez/PackGo/blob/main/Guia_EstandaresCodigo.md

Evidencia de código siguiendo la guía (capturas)

App.py

```
import mysql.connector
DB_HOST = "localhost"
DB_USER = "root"
DB_PASSWORD = "password"
DB_NAME = "packygo"

def conectar_base_datos():
    return mysql.connector.connect(
        host=DB_HOST,
        user=DB_USER,
        password=DB_PASSWORD,
        database=DB_NAME
    )

def obtener_clientes():
    conexion = conectar_base_datos()
    cursor = conexion.cursor()
    cursor.execute("SELECT nombre, correo FROM clientes")
    clientes = cursor.fetchall()
    conexion.close()
    return clientes

def main():
    clientes = obtener_clientes()
    for nombre, correo in clientes:
        print(f"Cliente: {nombre} - Correo: {correo}")

if __name__ == "__main__":
    main()
```