

PLATAFORMA DE CONTROL DE PEDIDOS MULTIPLATAFORMA

INTRODUCCIÓN Y CONTEXTO

1.1 Descripción del Proyecto

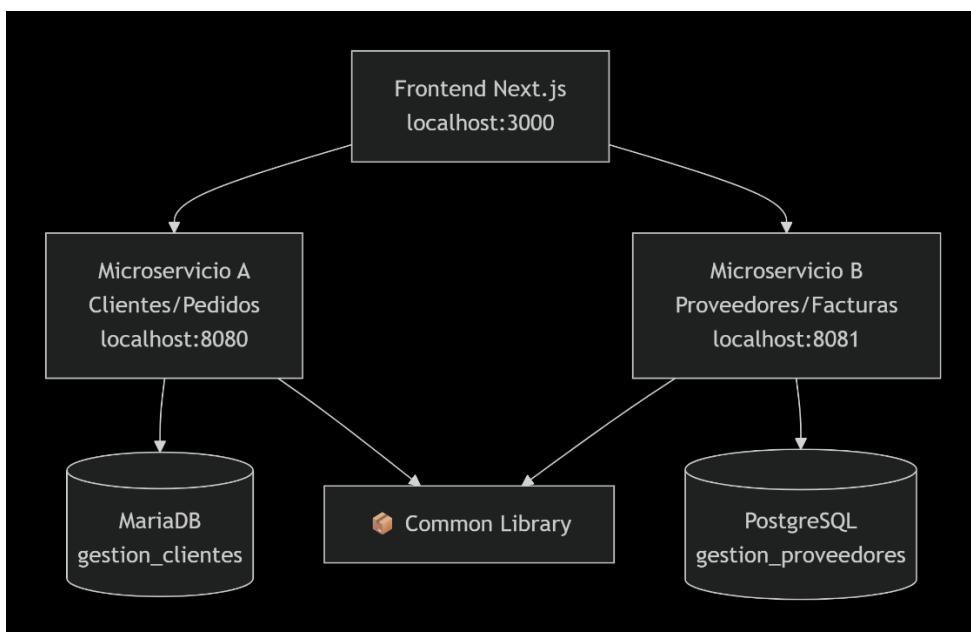
La empresa **MultiPedidos S.A.** requiere una solución de arquitectura distribuida para gestionar pedidos de distintos clientes y proveedores en varias regiones, operando con múltiples bases de datos por políticas internas y requerimientos técnicos.

1.2 Objetivos del Sistema

- Gestionar clientes y pedidos en MariaDB
- Gestionar proveedores y facturación en PostgreSQL
- Compartir lógica de negocio común mediante librería Maven
- Proporcionar dashboard administrativo unificado en Next.js

2. ARQUITECTURA DEL SISTEMA

2.1 Diagrama de Arquitectura General



2.2 Especificaciones Técnicas por Componente

Componente	Tecnología	Base de Datos	Puerto	Responsabilidad
Frontend Dashboard	Next.js 14	-	3000	Interfaz administrativa unificada
Microservicio A	Spring Boot 3.x	MariaDB	8080	Gestión de Clientes y Pedidos
Microservicio B	Spring Boot 3.x	PostgreSQL	8081	Gestión de Proveedores y Facturas
Common Library	Maven 3.9+	-	-	Lógica de negocio compartida

3. INSTALACIÓN Y CONFIGURACIÓN

3.1 Prerrequisitos del Sistema

[INSERTAR SCREENSHOT AQUÍ: prerequisitos-terminal.png]

Software Requerido:

- Java JDK 17 o superior
- Apache Maven 3.9 o superior
- Node.js 18 o superior
- npm 9 o superior
- MariaDB 10.11 o superior
- PostgreSQL 15 o superior

Comandos de Verificación:

```
java -version
```

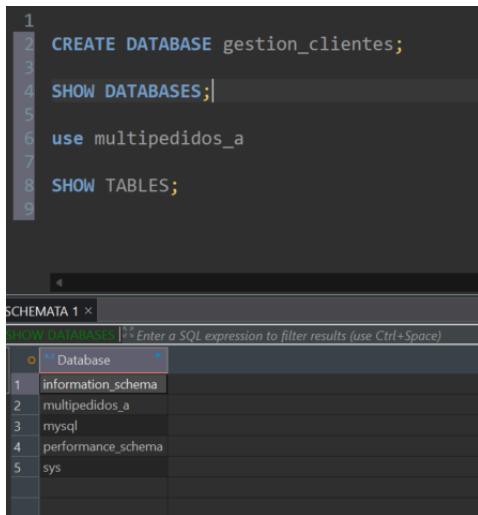
```
mvn -version
```

```
node --version
```

```
npm --version
```

3.2 Configuración de Bases de Datos

3.2.1 MariaDB (Microservicio A)

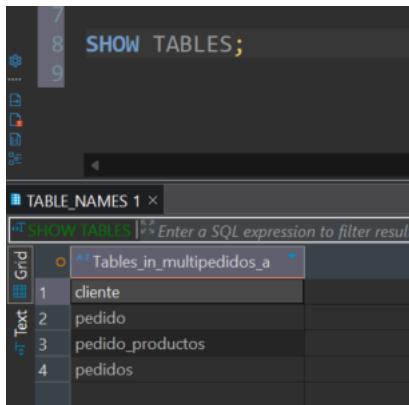


The screenshot shows the MySQL Workbench interface. In the top-left pane, there is a code editor with the following SQL script:

```
1 CREATE DATABASE gestion_clientes;
2
3 SHOW DATABASES;
4
5 use multipedidos_a
6
7 SHOW TABLES;
```

In the bottom-right pane, there is a results grid titled "SCHEMATA 1". It lists the following databases:

Database
information_schema
multipedidos_a
mysql
performance_schema
sys



The screenshot shows the MySQL Workbench interface. In the top-left pane, there is a code editor with the following SQL command:

```
7
8 SHOW TABLES;
9
```

In the bottom-right pane, there is a results grid titled "TABLE_NAMES 1". It lists the tables in the "multipedidos_a" database:

Tables_in_multipedidos_a
cliente
pedido
pedido_productos
pedidos

3.2.2 PostgreSQL (Microservicio B)

3.3 Instalación de Microservicios

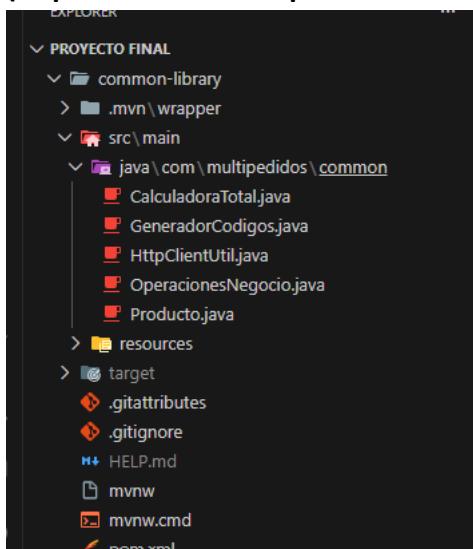
3.3.1 Common Library (Componente C)

```
cd common-library
```

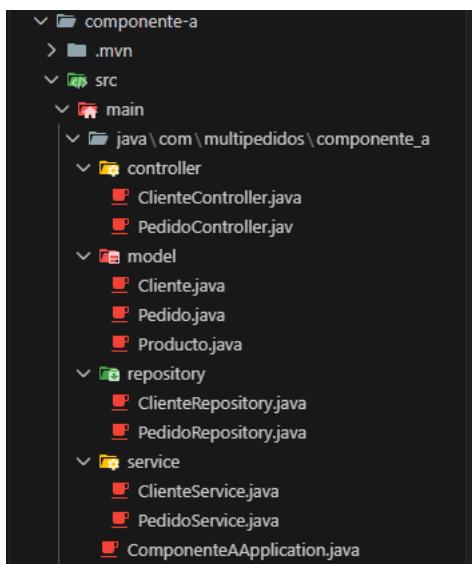
```
# Compilar e instalar
```

```
mvn clean install
```

(se puede crear la carpeta con mkdir en la carpeta raíz)



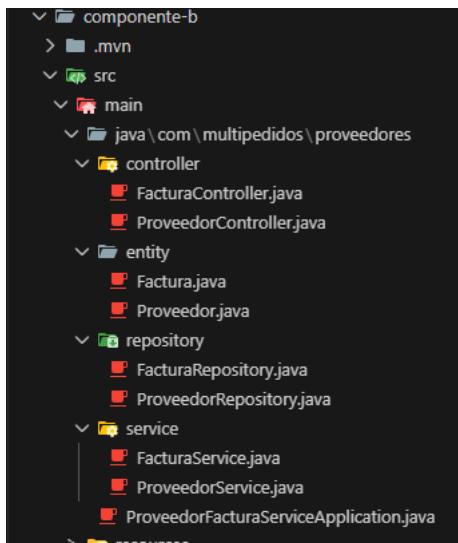
3.3.2 Microservicio A (Clientes y Pedidos)



```
# Ejecutar la aplicación
```

```
mvn spring-boot:run
```

3.3.3 Microservicio B (Proveedores y Facturas)



3.4 Instalación del Frontend

Instalar dependencias

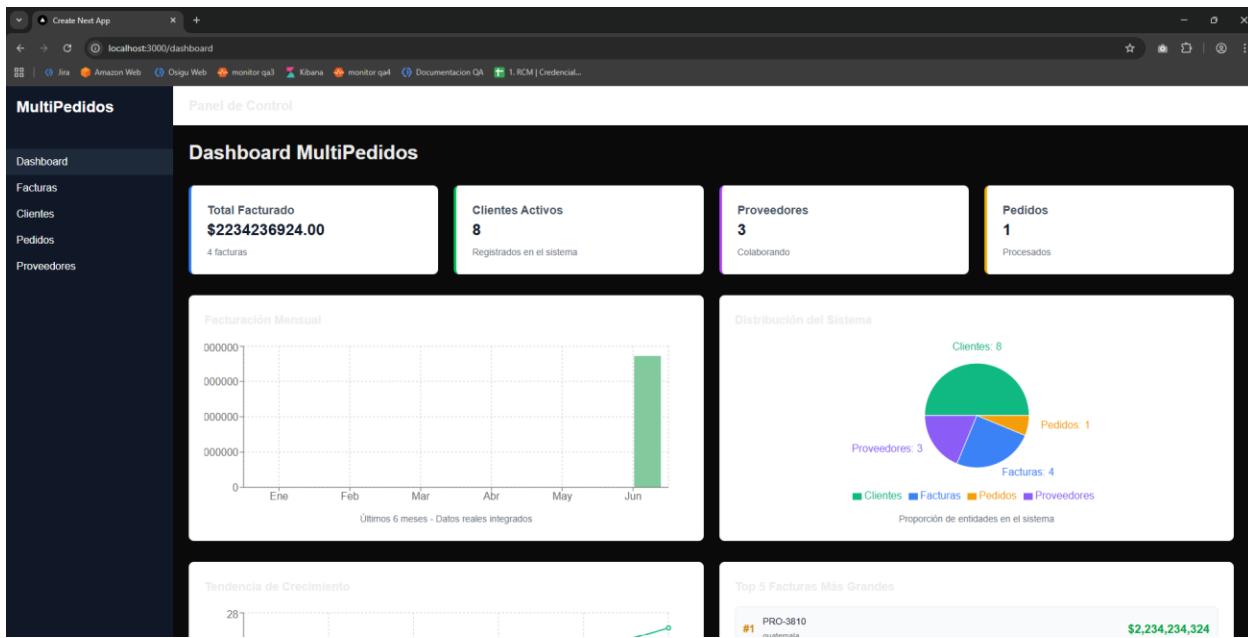
npm install

Ejecutar en modo desarrollo

npm run dev

4. MANUAL DE USUARIO

4.1 Dashboard Principal



Descripción: Vista general del dashboard de Next.js mostrando métricas y accesos rápidos

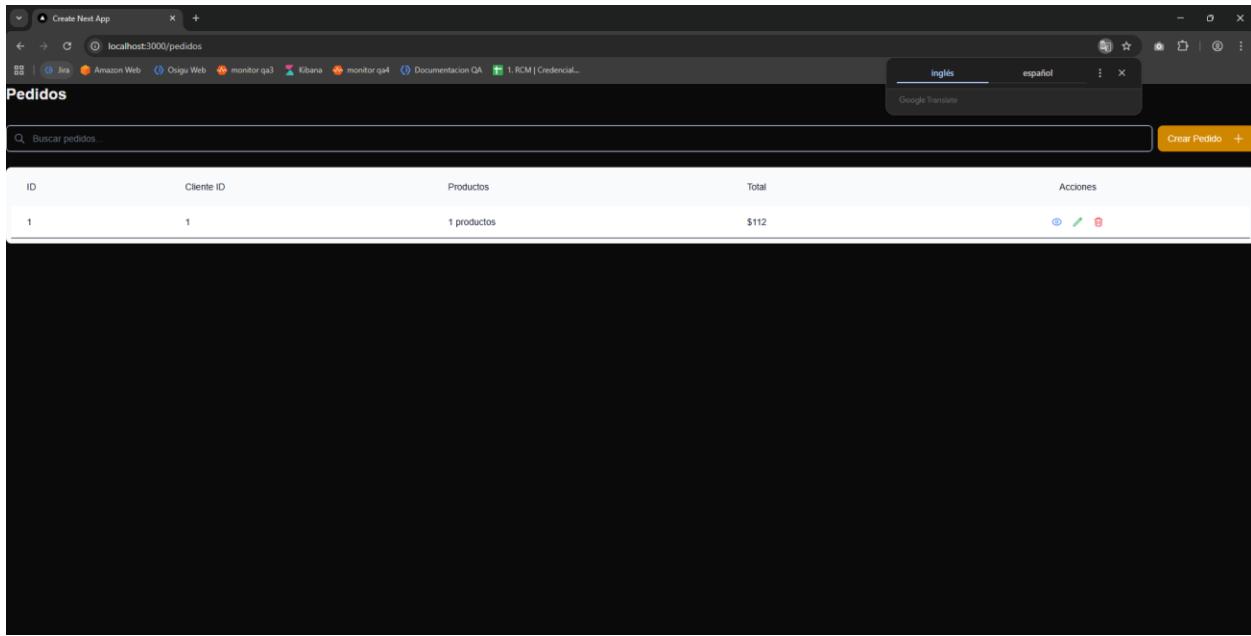
4.2 Gestión de Clientes

The screenshot shows a web browser window titled "Create Next App" with the URL "localhost:3000/clientes". The page has a dark header with the title "Clientes". Below the header is a search bar with the placeholder "Buscar clientes..." and a green button labeled "Crear Cliente". The main content area is a table with the following data:

ID	Nombre	Email	Acciones
1	Juan Perez	juan.perez@example.com	
2	Jesus Palma	jesus@ejemplo.com	
3	jose pando	jose.pando@example.com	
4	jose pando	jose.pando@example.com	
5	Jesus Palma	jesus@ejemplo.com	
6	jose guate	partido@gf.com	
7	jesus palma	jesupalma9@gmail.com	
8	hola mundo	hola@mundogmail.com	

Descripción: Interfaz para crear, editar y listar clientes

4.3 Gestión de Pedidos

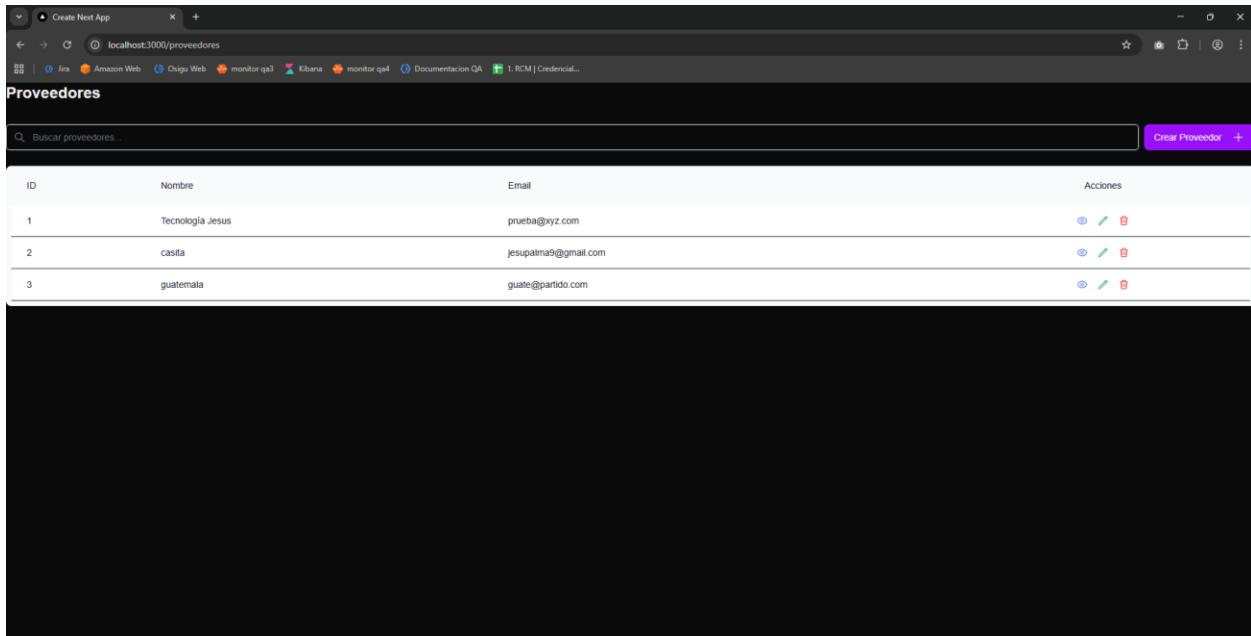


The screenshot shows a web browser window titled 'Create Next App' with the URL 'localhost:3000/pedidos'. The page has a dark header with tabs for 'Pedidos', 'Buscar pedidos...', 'Crear Pedido', and a language switcher between 'inglés' and 'español'. Below the header is a search bar and a table with one row of data.

ID	Cliente ID	Productos	Total	Acciones
1	1	1 productos	\$112	

Descripción: Interfaz para crear y visualizar pedidos con cálculos automáticos

4.4 Gestión de Proveedores

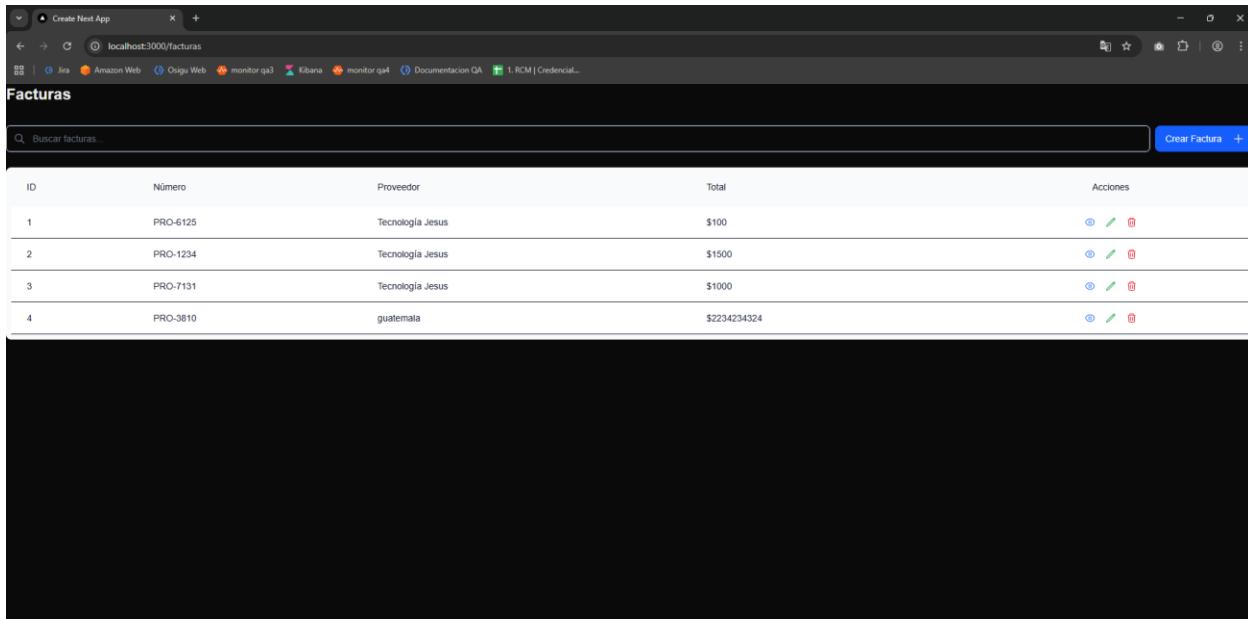


The screenshot shows a web browser window titled 'Create Next App' with the URL 'localhost:3000/proveedores'. The page has a dark header with tabs for 'Proveedores', 'Buscar proveedores...', 'Crear Proveedor', and a language switcher between 'inglés' and 'español'. Below the header is a search bar and a table with three rows of data.

ID	Nombre	Email	Acciones
1	Tecnología Jesus	prueba@xyz.com	
2	casita	jesupalma9@gmail.com	
3	guatemala	guate@partido.com	

Descripción: Interfaz para administrar proveedores

4.5 Gestión de Facturas



A screenshot of a web browser window titled "Create Next App". The address bar shows "localhost:3000/facturas". The page title is "Facturas". A search bar at the top left contains "Buscar facturas...". On the right, there are buttons for "Crear Factura" and a plus sign. The main content is a table with the following data:

ID	Número	Proveedor	Total	Acciones
1	PRO-6125	Tecnologia Jesus	\$100	 
2	PRO-1234	Tecnologia Jesus	\$1500	 
3	PRO-7131	Tecnologia Jesus	\$1000	 
4	PRO-3810	guatemala	\$2234234324	 

Descripción: Interfaz para generar y consultar facturas

5. API REFERENCE

5.1 Microservicio A - Clientes y Pedidos

URL Base: <http://localhost:8080>

The screenshot shows the Swagger UI interface for an OpenAPI definition. The top navigation bar includes the Swagger logo, the URL `/v3/api-docs`, and a green "Explore" button. Below the header, the title "OpenAPI definition" is displayed with a green "VALID" badge. A dropdown menu labeled "Swagger" is open, showing the option "http://localhost:8080 - Generated server url".

The main content area is titled "cliente-controller". It contains three sections for different HTTP methods:

- GET /clientes**: Shows no parameters and a single response for status code 200 (OK). The response schema is defined as follows:

```
{  "id": "string",  "name": "string",  "surname": "string"}  
Content-Type: application/json
```
- POST /clientes**: Shows no parameters and a single response for status code 200 (OK). The response schema is defined as follows:

```
{  "id": "string",  "name": "string",  "surname": "string"}  
Content-Type: application/json
```
- GET /clientes/{id}**: Shows a parameter "id" (required) of type "integer" and a single response for status code 200 (OK). The response schema is defined as follows:

```
{  "id": "string",  "name": "string",  "surname": "string"}  
Content-Type: application/json
```

At the bottom of the interface, there is a "Schemas" section containing a link to "Cliente".

Endpoints de Clientes:

Método	Endpoint	Descripción	Body Request
POST	/clientes	Crear nuevo cliente	{ "nombre": "string", "correo": "string" }
GET	/clientes	Listar todos los clientes	-
GET	/clientes/{id}	Obtener cliente por ID	-

Endpoints de Pedidos:

Método	Endpoint	Descripción	Body Request
POST	/pedidos	Crear nuevo pedido	{ "clientId": "number", "productos": [...] }
GET	/pedidos	Listar todos los pedidos	-
GET	/pedidos/{id}	Obtener pedido por ID	-

5.2 Microservicio B - Proveedores y Facturas

URL Base: <http://localhost:8081>

The screenshot shows the Swagger UI interface for Microservicio B. It displays two main sections: **provider-controller** and **factura-controller**.

provider-controller:

- Endpoints:** /api/provider/{id}, /api/provider/{id}/facturas, /api/provider/{id}/facturas/{id}, /api/provider/{id}/facturas/{id}/detalles.
- Responses:** 200 OK (application/json).
- Code Snippets:** Java, C#, Python, JavaScript, Go, PHP, Ruby, C/C++.

factura-controller:

- Endpoints:** /api/factura/{id}, /api/factura/{id}/detalles.
- Responses:** 200 OK (application/json).
- Code Snippets:** Java, C#, Python, JavaScript, Go, PHP, Ruby, C/C++.

The interface includes tabs for **Definitions**, **Parameters**, **Responses**, and **Responses**. A sidebar on the left lists **Models** and **Tags**.

Endpoints de Proveedores:

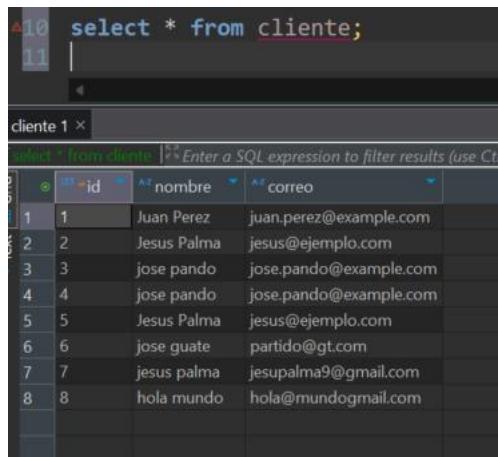
Método	Endpoint	Descripción	Body Request
POST	/proveedores	Registrar proveedor	{ "nombre": "string", "correo": "string" }
GET	/proveedores	Listar proveedores	-

Endpoints de Facturas:

Método	Endpoint	Descripción	Body Request
POST	/facturas	Registrar factura	{ "proveedorId": "number", "pedidos": [...] }
GET	/facturas	Listar facturas	-
GET	/facturas/{id}	Obtener factura por ID	-

6. ESTRUCTURA DE BASES DE DATOS

6.1 MariaDB - Esquema de Clientes y Pedidos



The screenshot shows a MySQL Workbench interface. At the top, there is a code editor window with two lines of SQL: "select * from cliente;" and an empty line below it. Below this is a results grid titled "cliente 1 ×". The grid has columns labeled "id", "nombre", and "correo". The data rows are:

	id	nombre	correo
1	1	Juan Perez	juan.perez@example.com
2	2	Jesus Palma	jesus@ejemplo.com
3	3	jose pando	jose.pando@example.com
4	4	jose pando	jose.pando@example.com
5	5	Jesus Palma	jesus@ejemplo.com
6	6	jose guate	partido@gt.com
7	7	jesus palma	jesupalma9@gmail.com
8	8	hola mundo	hola@mundogmail.com

CONCLUSIÓN

Esta plataforma representa una solución robusta y escalable para la gestión de pedidos multi-plataforma, implementando mejores prácticas de arquitectura de microservicios y desarrollo full-stack.