Propuesta Técnica – Full Stack Developer

# 1. Modelo Entidad-Relación (MER)

* Usuarios: id\_usuario (PK), nombre\_usuario, email, password (encriptada), rol\_id (FK → Roles).
* Roles: id\_rol (PK), nombre\_rol (Admin, Usuario, etc.).
* Clientes: id\_cliente (PK), tipo\_documento, documento, nombre, apellido, dirección, teléfono, fecha\_nacimiento, url\_imagen.
* Proveedores: id\_proveedor (PK), nit, nombre, ciudad, correo, teléfono.
* Productos: id\_producto (PK), nombre, categoría, marca, valor\_unitario, stock, descripción, proveedor\_id (FK → Proveedores).
* Compras: id\_compra (PK), cliente\_id (FK → Clientes), fecha\_compra.
* Detalle\_Compra: id\_detalle (PK), compra\_id (FK → Compras), producto\_id (FK → Productos), cantidad, valor\_unitario.

# 2. Backend (Python con Flask o Django)

* Autenticación: /auth/login con JWT.
* Usuarios y Roles: /users, /users/{id}.
* Clientes CRUD: /clients, /clients/{id}.
* Productos CRUD: /products, /products/{id}.
* Histórico de compras: /purchases/history (cliente, producto, proveedor, valor, fecha).

# 3. Frontend (React con TypeScript + Redux Toolkit)

* Login: autenticación y persistencia de token.
* Dashboard: muestra nombre y rol del usuario.
* Clientes: listado y CRUD.
* Productos: listado y CRUD.
* Historial de compras: tabla con cliente, producto, proveedor, fecha, total.
* Alertas y notificaciones con react-toastify.

# 4. Respaldo y Despliegue

* Versionamiento: Git con ramas develop y master en Azure DevOps.
* CI/CD: pipelines en Azure DevOps para test y despliegue.
* Infraestructura: Azure Web App (frontend), Azure Functions (backend), Azure Database for PostgreSQL.

# 5. Entregables

* Repositorio con carpetas frontend/ y backend/.
* Script SQL para la base de datos con datos de prueba.
* MER en imagen (draw.io, dbdiagram.io).
* Matriz de usuarios y roles con credenciales de prueba.