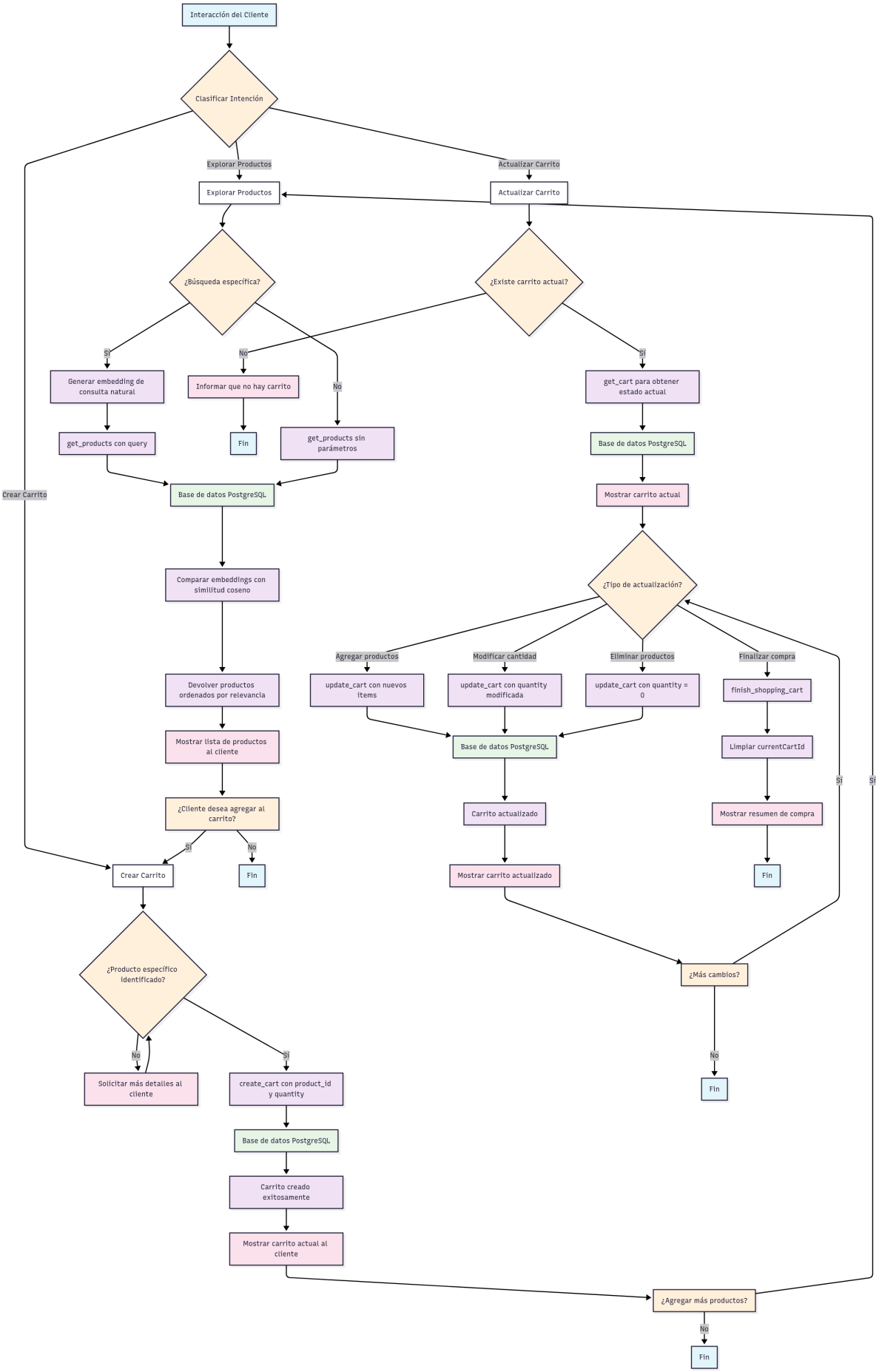
Desafío Técnico · Customer Success Engineer para Laburen.com

Fase Conceptual · Diseño del Agente de IA

Mapa de Flujo


Interacción del Cliente
Clasificar Intención
Explorar Productos
Actualizar Carrito
Explorar Productos
Actualizar Carrito
¿Búsqueda específica?
¿Existe carrito actual?
Sí
No
No
Sí
Generar embedding de consulta natural
Informar que no hay carrito
get_products sin parámetros
get_cart para obtener estado actual
get_products con query
Fin
Base de datos PostgreSQL
Crear Carrito
Base de datos PostgreSQL
Mostrar carrito actual
Comparar embeddings con similitud coseno
¿Tipo de actualización?
Agregar productos
Modificar cantidad
Eliminar productos
Finalizar compra
Devolver productos ordenados por relevancia
update_cart con nuevos ítems
update_cart con quantity modificada
update_cart con quantity = 0
finish_shopping_cart
Mostrar lista de productos al cliente
Base de datos PostgreSQL
Limpiar currentCartId
¿Cliente desea agregar al carrito?
Carrito actualizado
Mostrar resumen de compra
Sí
No
Crear Carrito
Fin
Mostrar carrito actualizado
Fin
Sí
Sí
¿Producto específico identificado?
¿Más cambios?
No
Sí
No
Solicitar más detalles al cliente
create_cart con product_id y quantity
Fin
Base de datos PostgreSQL
Carrito creado exitosamente
Mostrar carrito actual al cliente
¿Agregar más productos?
No
Fin

Endpoints

`GET` `/products` → Listado de productos con filtro opcional

`GET` `/products/:id` → Detalle de un producto (por id)

`POST` `/carts` → Crea un carrito de compras con sus correspondientes items

```
{
   "items": [
      {
         "product_id": "001",
         "qty": 2
      },
      {
         "product_id": "012",
         "qty": 1
      }
   ]
}
```

`PATCH` `/carts/:id` → Actualiza los items de un carrito (cantidades o elimina items)

```
{
   "items": [
      {
         "product_id": "001",
         "qty": 2
      },
      {
         "product_id": "012",
         "qty": 1
      }
   ]
}
```

Fase Práctica · API & Base de Datos

Stack tecnológico

Nombre	Descripción
NestJs	Framework de NodeJs - Backend (Nodejs v22.11)
PostgreSQL	Base de Datos
Langchain	Framework LLM
Gemini	LLM
Prisma	ORM
Docker	Base de datos dockerizada

🌐 **Aclaraciones**: tengo armado un compose que tambien levanta un pgAdmin, en el puerto 8080. Las credenciales de conexión estan en las variables de entorno.

Fundamentos

Decidí usar NestJS, debido a su rapidez para la construcción de endpoints y herramientas que el framework nos brinda a la hora de construir un backend.

Trabajo ya con prisma hace tiempo, es un ORM que me resulta muy cómodo.

Decidí Dockerizar mi base de datos PostgreSQL, para mayor portabilidad.

Variables de Entorno

```
# Prisma
DATABASE_URL=postgresql://default:Y8m!kP3rQ9zT1sL@localhost:5432/test-database

# Base de datos Postgres
POSTGRES_USER=default
POSTGRES_PASSWORD=Y8m!kP3rQ9zT1sL
POSTGRES_DB=test-database
POSTGRES_HOST=localhost
POSTGRES_PORT=5432

# Pg Admin Instance
PGADMIN_DEFAULT_EMAIL=default@gmail.com
PGADMIN_DEFAULT_PASSWORD=Y8m!kP3rQ9zT1sL

# Puerto de tu aplicación
PORT=3000

# Whatsapp - META
WHATSAPP_META_MY_PHONE_NUMBER=15551499059
WHATSAPP_META_PHONE_NUMBER_ID=784581108067391
WHATSAPP_META_BUSINESS_ACCOUNT_ID=1845284399534716
WHATSAPP_META_ACCESS_TOKEN=EAAI7ertty9IBPaXszaRqiMIECE6wNYsV0s8ombuMYfu1GLy0cZAAOaJvFcRupeuWZBpfXYSyv03y5DBafGBnG4IXfB1n7iZAErIcamhin9rEsxpXdnBoFIfT1vZASnYwBkUNnWbmDn4xpvPLGbOjzlhEPy44rO87eb2yhwo8vxd0P2pQtr8xprw5QUbnUZC6s5AZDZD
API_GRAPH_FACEBOOK_BASE_URL=https://graph.facebook.com/v22.0

# Gemini
GEMINI_API_KEY=AIzaSyCvHF2TWvCxVeOUQvX-RE-CIGgHcaIx5FI

# Api Base URL
API_BASE=http://localhost:3000
```

Pasos para levantar el proyecto

1. Clonar el repositorio de Github. `git clone` `https://github.com/PedrosaIgnacio/laburen`
2. Instalar dependencias `npm install`
3. Crear archivo `.env` con las variables de entorno mencionadas anteriormente
4. Crear contenedores de docker (requiere tener `Docker` instalado) `docker compose up --build -d`
5. Una vez levantado los contenedores, correr migraciones de prisma `npx prisma migrate deploy`
6. Levantar el proyecto `npm run start:dev` o buildear y levantar un entorno productivo `npm run build` y luego `npm run start:prod`

🌐 **Recordatorios:**

1. Es importante saber que el agente esta deployado en un numero de test de WhatsApp +1 (555) 149-9059
2. Por lo que lei, hay que dar autorización a un número, para que el agente pueda responder a ese numero en un entorno de test.
3. Recordar que estoy usando Gemini que tiene un límite de 50 request por dia.
4. El webhook de Meta esta actualmente usando mi ngrok que tengo levantado.