Aplicaciones LLM

Full Stack Level 1 App: To Do App

Objetivo

- Comprender el ciclo completo de desarrollo de una aplicación web full stack para poder aplicarlo en aplicaciones web LLM como las analizadas de LlamaIndex.

Full Stack Web Application

- Backend: FastAPI (Python) con Postgres en Render.com.
- Frontend: Next.js (React, Javascript) en Vercel.
- CRUD Application.
  - Create: crear nuevo item.
  - Read: mostrar todos los ítems o uno determinado.
  - Update: actualizar item.
  - Delete: eliminar item.

Recomendaciones previas

- Lo más importante es que entiendas los conceptos básicos. Las versiones del código pueden cambiar, pero los conceptos se mantienen.

- No necesitas aprender todo sobre Python, FastAPI, Postgress, Next.js, Vercel y Render.com. Aprende las técnicas básicas y luego sigue aprendiendo conforme lo necesites para cada proyecto que realices. Mantén la vista en la “big picture”, no pierdas la atención en pequeños detalles ahora.

- Recuerda: ningún ingeniero de software domina todo lo necesario para desarrollar un proyecto de antemano. Tiene conocimientos básicos y progresa a partir de ahí según las necesidades de cada proyecto.

- Recuerda dónde puedes encontrar respuestas en caso de duda o error: chatGPT, google, stackOverflow, etc.

Lógica esencial del backend

- Queremos crear una aplicación CRUD que responda a 5 instrucciones:
  - Crear un ítem en una base de datos.
  - Mostrar todos los ítems de la base de datos.
  - Mostrar un ítem de la base de datos localizado por su ID.
  - Actualizar un ítem de la base de datos localizado por su ID.
  - Eliminar un ítem de la base de datos localizado por su ID.

- Cada vez que se ordena uno de esas instrucciones:
  - Se ejecuta una función de python (CRUD helper) que lleva a cabo la acción CRUD seleccionada.
  - Se muestra el resultado en una URL de la web (CRUD route).

Parte 1: Backend con FastAPI y Postgres

1. Setup inicial.
2. Base de datos.
3. CRUD helpers.
4. Routes y endpoints.
5. Probar el backend en un servidor local.

1.1 Setup inicial

1. Crear entorno virtual.
2. Instalar los paquetes necesarios.

1.2 Base de datos

1. Crear la base de datos.
2. Introducir credenciales en archivo .env
3. Validar los tipos de datos con Pydantic.
4. Crear una tabla en la base de datos con SQLAlchemy y Alembic.

1.3 CRUD Helpers

1. Crear el **schema** para determinar los criterios de validación de los datos que introduce el usuario.

2. Crear el **modelo de datos** para determinar el formato de los datos que se van a introducir en la tabla de la base de datos.

3. Utilizando el schema y el modelo de datos, crear los **CRUD helpers** que gestionan las operaciones de entrada y salida de datos en la base de datos:
    a. Create: crear nuevo item.
    b. Read: mostrar todos los ítems o uno determinado.
    c. Update: actualizar item.
    d. Delete: eliminar item.

1.4 Routes (URLs) y endpoints (función ejecutada)

1. FastAPI CRUD route: URL asociada con cada acción del CRUD.
2. Endpoint: función Python que se ejecuta cuando se llama a cada route.
    a. Cada CRUD endpoint hace uso de un CRUD helper.

1.5 Probar el backend en un servidor local

1. Crear un servidor local con unicorn
2. Abrir una segunda ventana del terminal para probar el backend

Parte 2: Frontend con Next.js

1. Crear un starter template de Next.js
2. Introducir la URL de la API del backend en el archivo .env
3. Crear la página principal.
4. Crear los componentes incluidos en la página principal.
5. Detalles del componente clave.
6. Crear los estilos CSS de la aplicación.

2.1 Crear un starter template de Next.js

- Versión básica.

2.2 URL de la API del backend en .env

- Clave para conectar front y backend.

2.3 Crear la página principal de la aplicación

- Composition: insertar un componente dentro de otro.
- JSX: React parece HTML.

2.4 Crear los componentes de React/Next.js

- Componente Layout para aplicar a todas las páginas con composition.
- Componente ToDoList para mostar la funcionalidad to-do.
- Componente ToDo para mostrar cada to-do task.

2.5 Detalles del componente clave

1. Inicialización de las variables.
2. Cargar datos iniciales de la base de datos.
3. Funciones

Función addTodo(): activa la C del CRUD

- Activa el endpoint “/todos” de la API.
- Envía la nueva tarea to-do al servidor de backend.

Función fetchTodos(): activa la R del CRUD

- Activa el endpoint “/todos” de la API.
- Carga todas las tareas to-do desde el backend.
- Convierte esos datos en formato JSON.

Función updateTodo(): activa la U del CRUD

- Activa el endpoint “/todos/{id}” de la API.
- Envía los cambios en la tarea to-do al servidor de backend.
- Función handleToDoChange: actualiza la tarea to-do editada en la lista de tareas que guarda el frontend.

Función handleDeleteToDo(): activa la D del CRUD

- Activa el endpoint “/todos/{id}” de la API.
- Envía la tarea to-do eliminada al servidor de backend.
- Elimina la tarea to-do de la lista de tareas que guarda el frontend.

2.6 Crear los estilos CSS

- Se utilizan módulos CSS

Parte 3: Full Stack

1. Ejecutar el frontend y el backend simultáneamente en local
2. Desplegar el backend en Render.com
3. Desplegar el frontend en Vercel

3.1 Ejecutar frontend y backend simultáneamente en local

1. Abre una ventana de terminal en el directorio del backend:
   - uvicorn main:app --reload
2. Abre otra ventana de terminal en el directorio del frontend:
   - npm run dev
3. Abre tu navegador en http://localhost:3000
4. Prueba la app en local:
   - Introduce nuevas tareas
   - Lista las tareas pendientes y completadas
   - Edita una tarea de pendiente a completada
   - Borra una tarea

3.2 Desplegar el backend en Render.com

1. Sube el backend a github.
2. Crea una cuenta gratuita en Render.com
3. Crea una base de datos postgres en Render.com y guarda las credenciales.
4. Crea un servicio web en Render.com importando el backend de github.
5. Introduce las variables de entorno (credenciales de la base de datos) en Render.com
6. Crea la tabla todos en la base de datos remota desde tu terminal.
7. Chequea el log de Render.com para confirmar que el backend se carga sin problemas ni mensajes de error.
8. Recuerda dónde buscar respuestas en caso de dudas o problemas.

3.3 Desplegar el frontend en Vercel

1. Sube el frontend a github.
2. Crea una cuenta gratuita en Vercel.
3. Crea un nuevo proyecto en Vercel importando el frontend de github.
4. Introduce las variables de entorno en Vercel (URL del backend en Render.com).
5. Comprueba que la configuración de CORS es correcta para que no bloquee la comunicación entre frontend y backend.
6. Recuerda dónde buscar respuestas en caso de dudas o problemas.