Universidad de Guadalajara

Centro Universitario de Ciencias Exactas e Ingenierías

Licenciatura en ingeniería informática

Reporte- TaskAI

Programación para Internet

Integrantes:

Miranda Mercado, Valeria – 219416801

Navarro Rodríguez, Javier - 212552416

Vargas Zavala, Jeremy, Gustavo – 219369323

Índice

TaskAI - Reporte

TaskAI es una aplicación web de gestión de tareas en la que integramos una inteligencia artificial a través de la API de Chat GPT. Esta herramienta está diseñada para ayudar a los usuarios a organizar sus tareas de manera eficiente, estableciendo prioridades y recibiendo sugerencias personalizadas que optimizan su productividad, de esta forma, volvemos más sencilla la manera de llevar a cabo las tareas pendientes.

Lo novedoso o destacable de este proyecto es que, como tal, TaskIA nos da la oportunidad a nosotros como usuarios de simplificar tareas que podríamos no haber considerado para ello. Lo hace de forma estratégica al dividir o desglosar tareas complejas en subtareas más sencillas cuando se lo pedimos a la IA y también al generar una tarea principal, lo que nos brinda una experiencia más agradable al momento de visualizar qué haremos y cómo lo haremos.

Objetivos

- **Organizar Tareas:** Permitir a los usuarios crear, editar y eliminar tareas.
- **Establecimiento de Prioridades:** Ofrecer la opción de clasificar tareas según su urgencia e importancia.
- **Recomendaciones Inteligentes:** Utilizar la IA para sugerir tareas y generar subtareas basadas en la tarea del usuario.
- **Compatibilidad Multidispositivo:** Asegurar que la aplicación sea accesible y funcional en dispositivos móviles y de escritorio.

Características Principales

- **Interfaz Intuitiva:** Diseño de usuario simple y amigable que facilita la navegación.
- **Integración de IA**: Utiliza la API de ChatGPT para proporcionar recomendaciones de tareas y generar subtareas.
- **Funcionalidad Responsiva:** La aplicación se adapta a diferentes tamaños de pantalla, permitiendo su uso en celulares, tabletas y ordenadores.
- **Registro de Progreso:** Los usuarios pueden seguir su progreso y ajustar sus tareas según sea necesario.
- **Tags**: La aplicación incluye un sistema de etiquetado que permite a los usuarios clasificar y organizar sus tareas según diferentes categorías. Esto facilita la búsqueda

y filtrado de tareas, permitiendo una gestión más eficiente del tiempo y una visualización clara de las prioridades.

Tecnologías Utilizadas

- **Frontend:** React con TypeScript para la construcción de la interfaz de usuario.
- **Herramientas de Desarrollo:** Vite como empaquetador y Tailwind CSS para los estilos.
- **Enrutamiento:** React Router para la navegación entre diferentes páginas de la aplicación.
- **Funciones Serverless:** Implementación de funciones serverless en Vercel para la gestión de la lógica del servidor.
- **API de IA:** Integración con la API de OpenAI para proporcionar sugerencias y recomendaciones inteligentes.
- **Axios**: Manejo de solicitudes HTTP. Utilizado para realizar solicitudes a la API de OpenAI y a las funciones serverless, facilitando la comunicación entre el frontend y el backend de manera eficiente y sencilla.
- **Firebase**: Plataforma utilizada para la gestión de la autenticación, almacenamiento y otras funcionalidades backend.
- **Firestore**: Base de datos NoSQL de Firebase que permite almacenar y sincronizar datos en tiempo real, facilitando el desarrollo de aplicaciones interactivas y en tiempo real.

Estructura del Proyecto

La aplicación está organizada en diferentes módulos que manejan la interfaz de usuario, la lógica de la aplicación y la integración con la API de ChatGPT.

```
AITask/
├── api/                  # Código relacionado con la API del backend
├── docs/                 # Documentación del proyecto (presentaciones, guías, etc.)
├── node_modules/         # Dependencias del proyecto instaladas por npm
├── public/               # Archivos estáticos (index.html, imágenes, etc.)
├── src/                  # Código fuente de la aplicación frontend
│   ├── components/       # Componentes de React reutilizables
│   ├── pages/            # Páginas de la aplicación
```

```
│   ├── hooks/              #  Hooks personalizados de React
│   ├── context/            #  Context API para el manejo de estado
│   ├── services/           # Servicios para interactuar con la API
│   └── styles/             # Archivos de estilos (CSS, Tailwind, etc.)
├── .env                    # Variables de entorno para la configuración
├── .gitignore              # Archivos y carpetas a ignorar en Git
├── eslint.config.js        # Configuración de ESLint para la calidad del código
├── firestore.rules         # Reglas de seguridad para Firestore
├── index.html              # Archivo principal HTML de la aplicación
├── package-lock.json       # Archivo que mantiene las versiones de las dependencias
├── package.json            # Dependencias y scripts del proyecto
├── postcss.config.js       # Configuración de PostCSS
├── README.md               # Documentación principal del proyecto
├── tailwind.config.js      # Configuración de Tailwind CSS
├── tsconfig.app.json       # Configuración de TypeScript para la aplicación
├── tsconfig.json           # Configuración general de TypeScript
├── tsconfig.node.json      # Configuración de TypeScript para Node.js
└── vercel.json             # Configuración para despliegue en Vercel
```

Desafíos y Soluciones

- **Integración de la API de OpenAI con Vercel y Vite**

  Originalmente, se planeó integrar la API de OpenAI con el framework Express para manejar los llamados en el backend y evitar exponer la API key al navegador. Sin embargo, esta integración no fue posible en Vercel, ya que no permite manejar un backend de forma convencional. Ante esta situación, se migró todo el sistema a funciones serverless, lo que provocó que los llamados a la API de OpenAI solo funcionaran en la versión desplegada en Vercel. Para solucionar esto, se creó un servidor middleware con un plugin personalizado de Vite que simula los endpoints de la API e intercepta las peticiones a /api de forma local. La API key quedó protegida porque al final no quedó en el código, sino que está guardada en las variables de entorno de Vercel.

- **Adaptación del Proyecto a Vercel**

  Durante el desarrollo, no se sabía que Vercel no podría manejar el framework de Express. Como resultado, se tuvieron que realizar numerosos cambios y pruebas de regresión para confirmar que tanto las funciones serverless como el plugin de Vite funcionaran en ambos entornos: de manera local y en su versión desplegada.

- **Adaptación a la base de datos que se implementó con firebase y firestore.**

  Al final, para poder terminar el proyecto y que estuviera en su totalidad se implementaron estas herramientas para poder tener una base de datos en el proyecto. En un principio no sabíamos si funcionaria pero con algunos arreglos y experiencias de la tarea realizada con firebase la implementación fue aplicable con éxito.

Posibles Futuras Mejoras

- Mejorar la precisión de las sugerencias de IA.
- Agregar un sistema de notificaciones personalizadas para recordar tareas pendientes.
- Mejorar la personalización de la interfaz de usuario.
- Opciones avanzadas de filtro y búsqueda de tareas.

Guía de Despliegue

1. **Clonar proyecto**
   a. git clone https://github.com/HarumiAme/AITask.git
   b. cd AITask
2. **Instalar dependencias**
   a. npm install
3. **Agregar tu clave de API de openAI (en la raíz del proyecto)**
   a. echo "OPENAI_API_KEY=[Tu clave de API]" > .env
4. **Ejecutar**
   a. npm run dev

Conclusión

TaskAI se presenta como una solución eficaz para la gestión de tareas, combinando la organización personal con la inteligencia artificial. Con su interfaz amigable y funcionalidades avanzadas, la aplicación busca optimizar la productividad de los usuarios, haciéndola una herramienta invaluable en la gestión diaria de tareas.