**Arquitectura de la Aplicación**

**1. Frontend**

* **Componente**: **React con TypeScript**
  + **Función**: Interfaz de usuario que interactúa con el backend a través de solicitudes HTTP. Proporciona una experiencia interactiva y dinámica para los usuarios.
  + **Despliegue**: Alojado en **Amazon S3** y distribuido a través de **Amazon CloudFront** para asegurar una baja latencia y alta disponibilidad.

**2. Gateway API**

* **Componente**: **Amazon API Gateway**
  + **Función**: Punto de entrada para las solicitudes del frontend. Enruta las solicitudes a los microservicios adecuados, maneja autenticación, autorización, y proporciona funciones de caché y monitoreo.
  + **Ventajas**: Escalabilidad automática, soporte para CORS, y capacidad de gestionar diferentes versiones de APIs.

**3. Microservicios**

* **Componente**: **Microservicios** (implementados con FastAPI)
  + **Función**: Contienen la lógica de negocio, gestionan las reglas del dominio y se comunican con la base de datos. Cada microservicio maneja una parte específica de la funcionalidad de la aplicación.
  + **Despliegue**: Utiliza **uWSGI** con **Nginx** como proxy inverso y balanceador de carga, desplegado en instancias de **Amazon EC2** o **AWS ECS** para la ejecución de contenedores.

**4. Funciones Asincrónicas**

* **Componente**: **AWS Lambda**
  + **Función**: Maneja tareas que requieren procesamiento en segundo plano o asincrónico, como el procesamiento de eventos de colas y tareas programadas.
  + **Despliegue**: Ejecuta funciones en respuesta a eventos, sin necesidad de gestionar servidores.

**5. Colas de Mensajes**

* **Componente**: **Amazon SQS**
  + **Función**: Maneja la cola de mensajes para procesos asincrónicos y para desacoplar los componentes del sistema, permitiendo que los microservicios y las funciones Lambda se comuniquen de manera eficiente.
  + **Ventajas**: Permite el almacenamiento temporal de mensajes y asegura que los mensajes se procesen incluso si hay picos en el tráfico.

**6. Notificaciones**

* **Componente**: **Amazon SNS**
  + **Función**: Envia notificaciones y mensajes a múltiples suscriptores o sistemas en respuesta a eventos. Puede ser usado para notificaciones de usuarios o para otros componentes del sistema.
  + **Ventajas**: Soporta varios tipos de protocolos de entrega como SMS, correo electrónico, y otras aplicaciones.

**7. Gestión de Base de Datos**

* **Componente**: **Amazon RDS (PostgreSQL)**
  + **Función**: Almacena y gestiona los datos persistentes de la aplicación. Utiliza PostgreSQL como motor de base de datos relacional.
  + **Ventajas**: Proporciona escalabilidad, seguridad, y copias de seguridad automáticas.

**8. Balanceo de Carga y Proxy Inverso**

* **Componente**: **Nginx + uWSGI**
  + **Función**: **Nginx** actúa como un balanceador de carga y proxy inverso para los microservicios implementados con **uWSGI**, distribuyendo las solicitudes de manera eficiente y proporcionando características de proxy inverso.
  + **Ventajas**: Mejora el rendimiento y la disponibilidad de los microservicios.

**9. DNS y Distribución Global**

* **Componente**: **Amazon Route 53**
  + **Función**: Servicio de DNS que gestiona la resolución de nombres y proporciona alta disponibilidad y rendimiento de DNS.
  + **Ventajas**: Ofrece soporte para enrutamiento de tráfico global y administración de dominios.

**Diagrama de Arquitectura**

1. **Frontend** (React con TypeScript) → **API Gateway**
2. **API Gateway** → **Microservicios** (FastAPI)
3. **Microservicios** → **Base de Datos** (Amazon RDS)
4. **Microservicios** → **Colas de Mensajes** (Amazon SQS)
5. **Microservicios** → **Funciones Asincrónicas** (AWS Lambda) para procesar mensajes de SQS
6. **Funciones Asincrónicas** (AWS Lambda) → **Notificaciones** (Amazon SNS) si es necesario
7. **API Gateway** y **Microservicios** se despliegan detrás de **Nginx** y **uWSGI** para balanceo de carga y proxy inverso
8. **Frontend** y otros recursos están distribuidos globalmente usando **Amazon CloudFront** con DNS gestionado por **Amazon Route 53**

