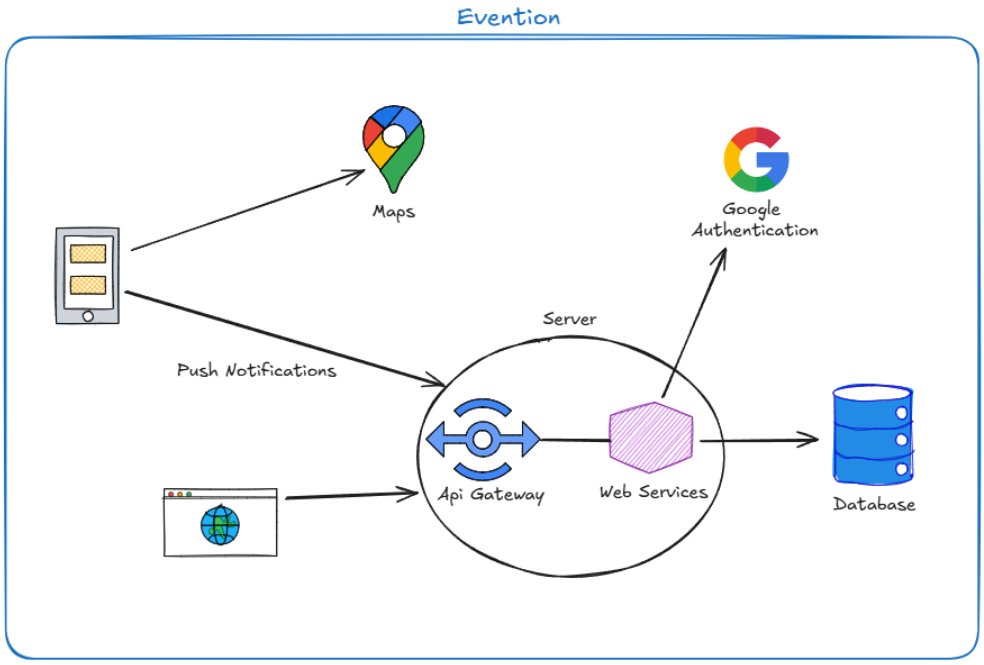
Requisitos Avançados

Arquitetura Funcional

O diagrama apresentado fornece uma visão resumida da arquitetura do sistema Evention, destacando os principais fluxos de comunicação e integração entre os componentes. A interação com os utilizadores é feita através de aplicações móveis e web, que comunicam com o sistema central por intermédio de um API Gateway.

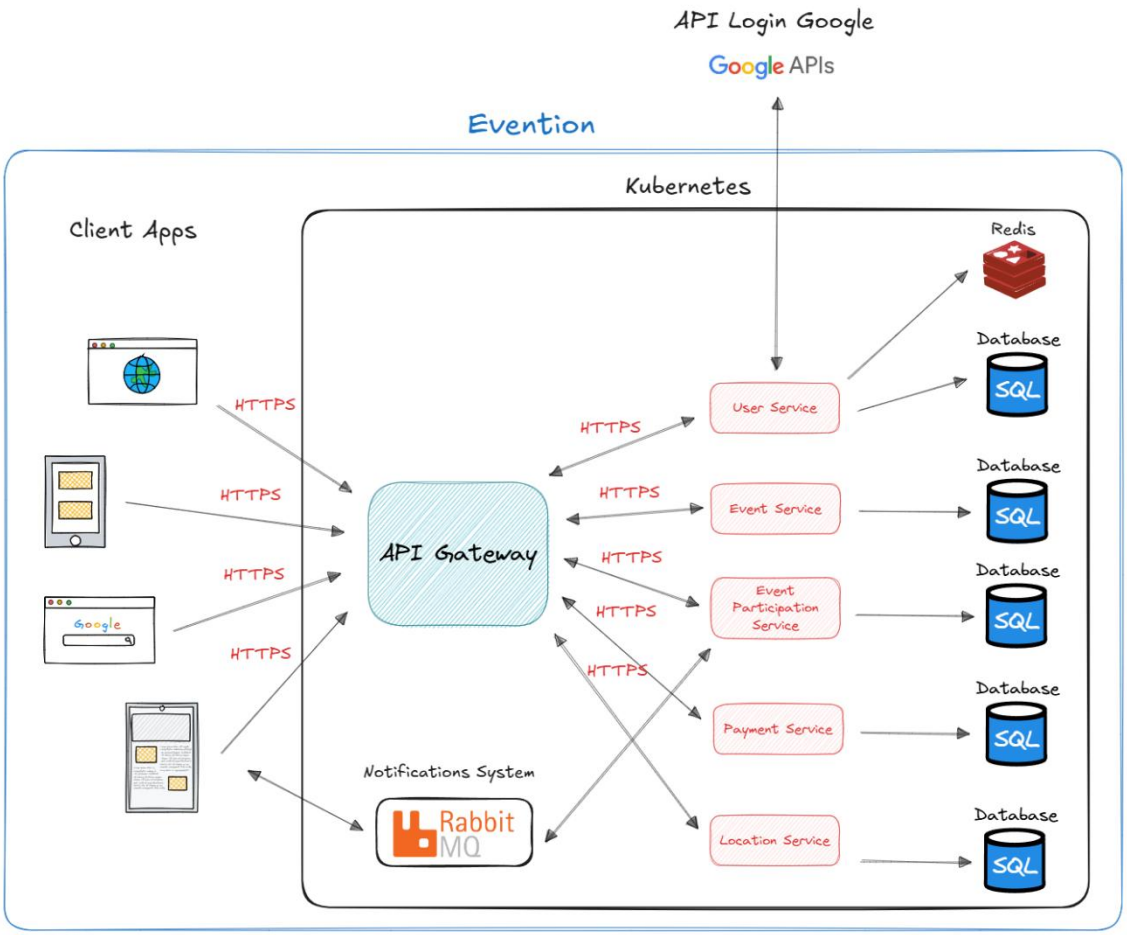
Para autenticação, o sistema utiliza integração com o Google Authentication, permitindo que os utilizadores façam login de forma rápida e segura com as suas contas Google. Além disso, existe integração com os serviços de mapas, que permitem funcionalidades como a visualização dos locais dos eventos.

O sistema implementa também notificações push para manter os utilizadores informados, em tempo real, sobre alterações relevantes nos eventos, como lembretes, mudanças de horário ou novas publicações.



Arquitetura Detalhada Evention

Na imagem seguinte, é possível visualizar o modelo arquitetural final detalhado e atualizado, com o padrão API Gateway, a comunicação utilizando o protocolo HTTPS, o sistema de notificações utilizando o RabbitMQ e a ligação à API externa da Google que permite o registo e login com conta Google.



Tecnologias Utilizadas

BackEnd:

- Node.js (Javascript)
- JWT
- Redis
- Nginx
- PostgreSQL
- RabbitMQ

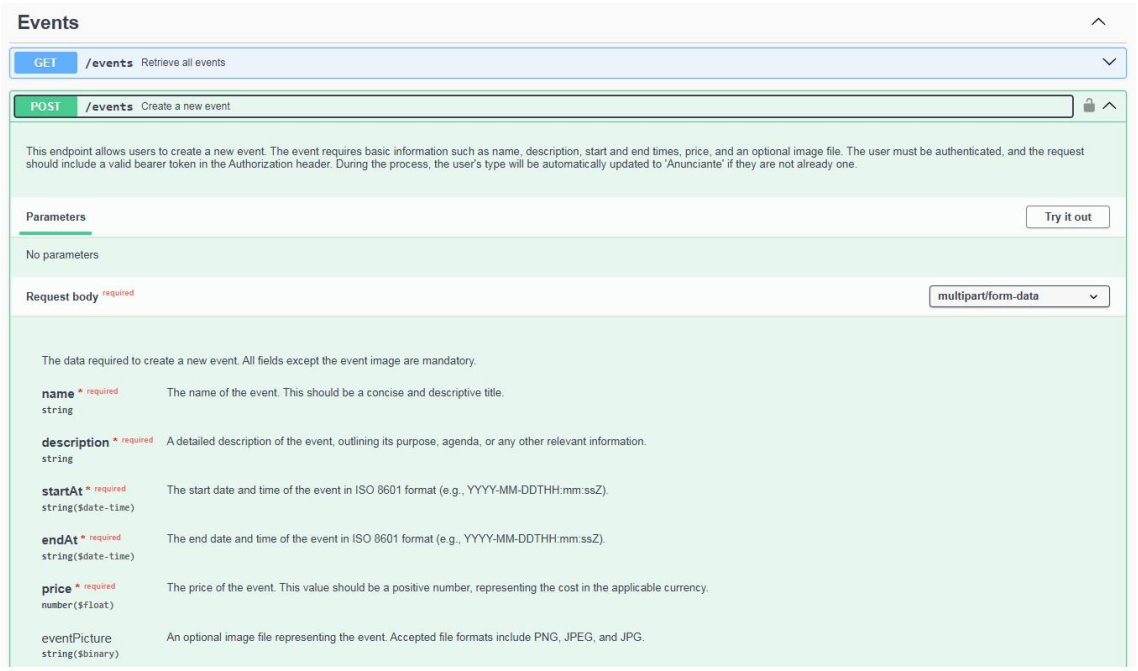
FrontEnd:

- TailWind

- APEXCHARTS.JS
- React
- Html/CSS
- ReactQuery

Documentação

Documentação do Código dos Microsserviços em Node.js utilizando Swagger.



Requisitos Servidor

Autenticação JWT;

População de base de dados utilizando um arquivo Seed.js;

Upload e exposição de Imagens;

Utilização de APIs externas (Google Auth);

Documentação da API utilizando a especificação OpenAPI (Swagger);

Requisitos Cliente

Aplicação React criada utilizando Vite;

WireFrames da aplicação Web;

Interface responsiva;

Utilização de uma biblioteca de estilos CSS (Tailwind);

Utilização de biblioteca para gráficos (APEXCHARTS.JS);

Utilização e configuração de Server-Side Rendering (SSR) e/ou componentes server-side através de um job no serviço Event para finalizar eventos quando estes ultrapassam a data final;

Cache de dados utilizando react-query;

Utilização de Mocks;

Requisitos Distribuição

Distribuição de BackEnd e FrontEnd através de dois containers Docker diferentes;

Container Docker específico para a aplicação Web (React);

Github Actions para dar deploy das imagens dos serviços de cada container Docker;

LoadBalancer configurado corretamente para cada serviço;

Configuração de certificados SSL;

Scripts para Mac e Windows para fazer deploy do ambiente Docker no Kubernetes (Repositório KubernetesRUN).

Melhorias Efetuadas

Container Docker específico para a aplicação Web (React).