**Sistema de Trackeo de Tickets en tiempo real del Comedor de UNMSM (GT)**

**Documentación de Arquitectura del software**

**Lima, abril del 2023**

**ÍNDICE**

**Contenido**

Tabla de contenido

**Introducción**3

**Requerimientos de la arquitectura3**

**Diagrama de la arquitectura**4

**Descripción de los componentes**4

Recursos internos4

Recursos externos4

**Patrones de diseño**5

**Consideraciones de rendimiento**5

**Consideraciones de seguridad**5

**Consideraciones de escalabilidad**5

**Conclusión**5

Introducción

El proyecto en cuestión busca darle solución a la constante problemática de múltiples alumnos frente a si estos van a ser capaces de alcanzar un ticket para reclamar su comida o no de una manera conveniente, sencilla y rápida de tal manera que este pueda ahorrar el tiempo que le demoraría en trasladarse hasta el comedor y verificar la disponibilidad de los tickets. Se usarán múltiples herramientas para el desarrollo de esta aplicación, notablemente

* React
* Fastapi
* Gunicorn
* Postgres

Se han elegido dichos frameworks/herramientas con el fin de agilizar y acelerar los tiempos de desarrollo debido a los escasos recursos que poseemos y la facilidad de generar distintos documentos y registros que ciertas de estas herramientas nos proporcionan (especialmente Fastapi)

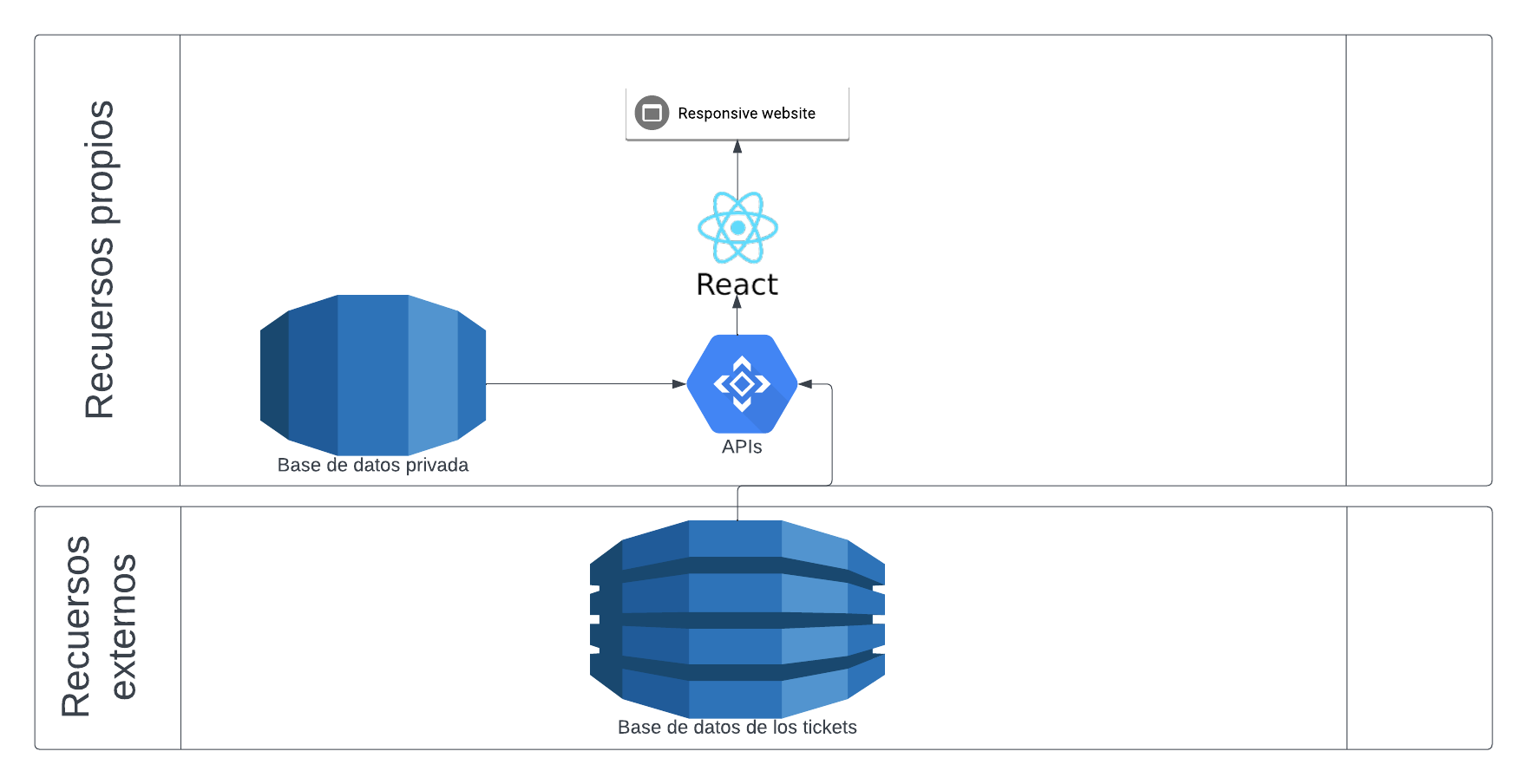
Requerimientos de la Arquitectura

En esta ocasión ya que vamos a usar información públicamente disponible la cual va a ser procesada, por ello los requerimientos clave serán:

* Seguridad: La aplicación deberá mantener un seguimiento adecuado y acorde a las peticiones que pueda soportar el servidor que mantenga el conteo de los tickets, por ello será necesario tomar medidas de aseguramiento para realizar requests.
* Rendimiento: La aplicación deberá ser muy optimizada para lograr el mayor alcance posible y todos los estudiantes tengan la oportunidad de utilizarla independientemente de la gama de sus dispositivos.

Diagrama de la Arquitectura

El diagrama de la arquitectura usada es el siguiente:



La manera que se está estructurando esto es en dos niveles, recursos externos e internos. Ya que la aplicación es pequeña en escala de desarrollo, todo se puede conglomerar en un único espacio. Sin embargo, al usar elementos externos es necesario señalar ello.

Descripción de los Componentes

Este desarrollo tiene 2 componentes:

Recursos internos

Es todo aquello que se desarrollará durante el tiempo establecido, base de datos, front y backend incluido, pues como todo cumple una función específica, no hace falta desenlazar una cosa de la otra.

Recursos externos

Son todos aquellos recursos que se van a utilizar para poder mostrar la información que debemos tener en la aplicación. Según el análisis realizado, se espera utilizar únicamente la información que utiliza la misma universidad para llevar el conteo de los tickets.

Patrones de Diseño

Se han utilizado los siguientes patrones de diseño para implementar la arquitectura de software:

* MVC: Se ha utilizado el patrón Modelo-Vista-Controlador para separar la lógica de negocio, la presentación y el control de la aplicación.
* Singleton: Se ha utilizado el patrón Singleton para garantizar que solo haya una instancia de conexión a la base de datos.

Consideraciones de Rendimiento

Para garantizar un rendimiento óptimo, se han implementado las siguientes medidas:

* Caché de datos: Los datos frecuentemente accedidos se almacenan en caché para reducir el número de consultas a la base de datos.
* Optimización de consultas: Se han optimizado las consultas a la base de datos para mejorar el rendimiento.

Consideraciones de Seguridad

En cuanto a consideraciones de seguridad se tiene:

* Uso de tokens para los requests.
* Limitar los requests a unos cuantos por minuto.
* Aseguramiento de encabezados a la base de datos.

Consideraciones de Escalabilidad

No se están tomando en cuenta consideraciones de escalabilidad pues la herramienta únicamente va a cumplir una función, solo se está considerando mantenimiento de la misma.

Conclusiones

Es de suma importancia el aseguramiento de los querys recibidos tanto a la base de datos privada como al api pues aunque no contengan información sensible, es muy importante para el rendimiento de la aplicación que estos funcionen de manera óptima. Otro aspecto es la visibilidad y entendimiento de la disposición de los datos para los usuarios pues se trata de tener una información representativa útil y atractiva para todos los usuarios. Limitar la cantidad de flujo de datos que se pueda tener, pues puede conllevar a un uso desmedido y defenestrar tanto el servidor como al usuario.