Documento de Diseño de Sistema

NOMBRE DEL PROYECTO
API Fútbol

INTEGRANTES

Código	Apellidos y Nombre	% de Participación
202010442	Arróspide González, Eduardo	100%
202010556	Cortez Gorbalan, Ian Augusto	100%
201910375	Dominguez Aspilcueta, Pedro Francisco	100%
202020159	Ugarte Quispe, Grover Eduardo	100%

CONTENIDO

1. Resumen

1.1. Resumen del producto o servicio

El servicio es una API que permite a los usuarios acceder a los resultados de partidos de fútbol. Asimismo, la información se mantiene actualizada para incluir los partidos más recientes.

1.2. Objetivos

- Proporcionar a los usuarios información de manera rápida sobre partidos de fútbol sin la necesidad de acceder a diferentes páginas web.
- Proporcionar la información de los partidos de fútbol de manera detallada, precisa y en un formato fácil de entender.

1.3. Glosario de términos

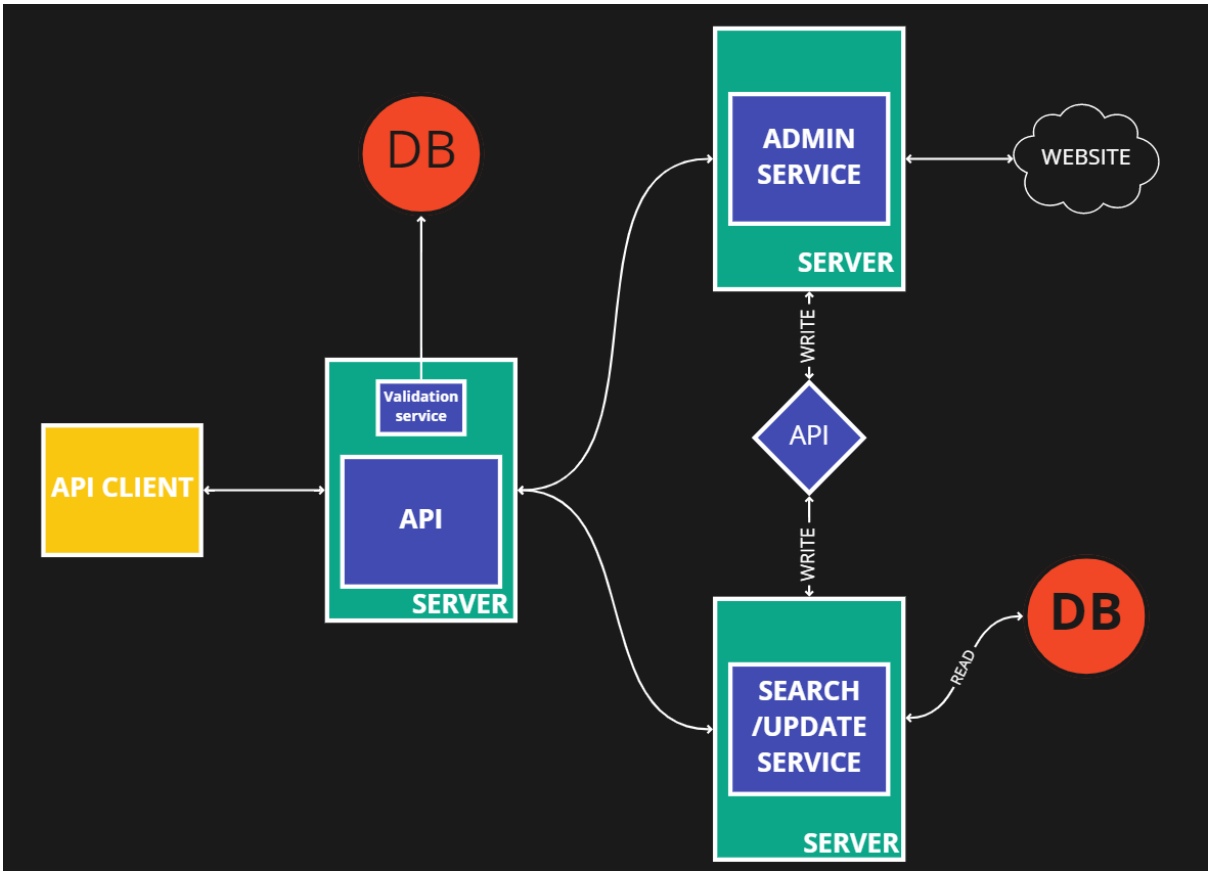
- Registro - Información de un partido de fútbol
- API - Application Programing Interface
- Token - Llave única con la que un usuario / administrador puede acceder a la API/ portal de actualización de datos respectivamente.
- Stripe - Herramienta utilizada para realizar transacciones monetarias empleada principalmente en negocios virtuales
- Frontend - Refiere a los elementos y características que interactúan de forma directa con el usuario.
- Backend - Refiere al servidor de una aplicación y a todo lo que se comunica entre el *storage* y el *frontend*.
- Storage - Mecanismo que permite a un ordenador retener datos.

2. Diagrama de Arquitectura

2.1. Glosario

- ADMIN SERVICE: servicio utilizado por un administrador para realizar el *scraping* de nuevo contenido y actualizar la base de datos.
- SEARCH/UPDATE SERVICE: servicio utilizado por un usuario para realizar búsquedas de resultados de partidos de fútbol.
- API CLIENT: simulador de entradas al servicio de API sin requerir de comandos GET o POST
- API SERVICE: servicio general que se encarga de redirigir al usuario del API CLIENT en caso sea un consumidor o administrador.
- VALIDATION SERVICE: servicio encargado de verificar a usuarios.

2.2. Diagrama



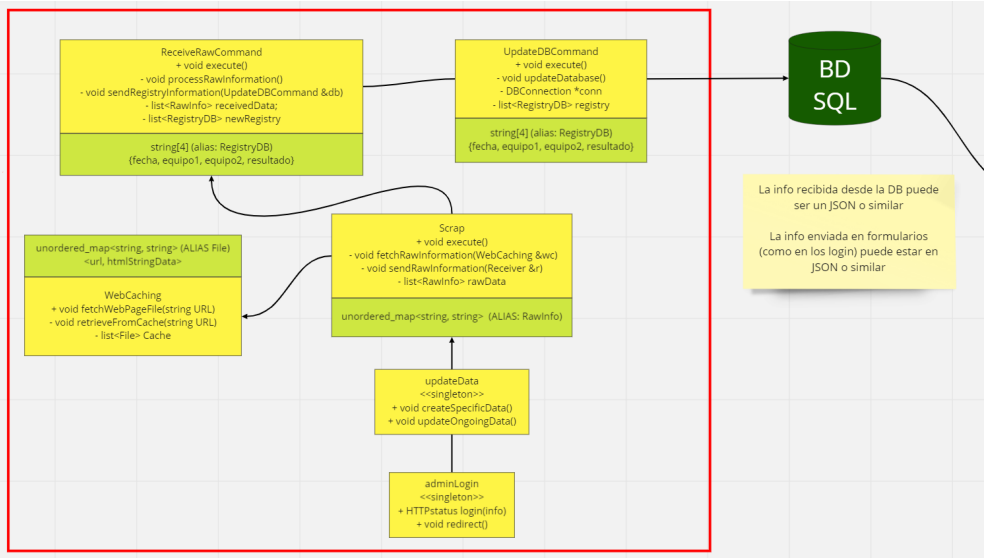
3. Diagrama de Clases

3.1. Glosario

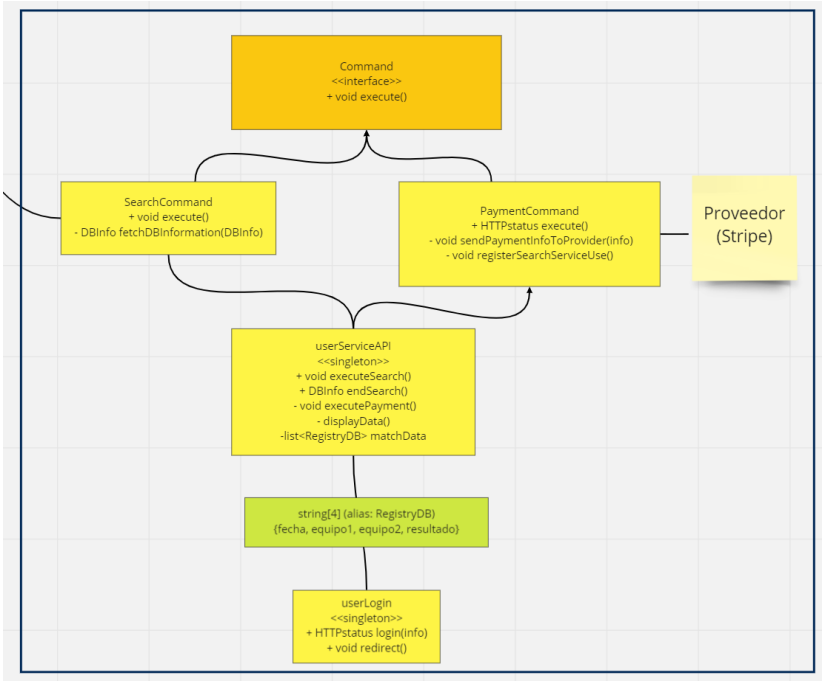
- ReceiveRawCommand: Clase comando que obtiene los datos otorgados por la página web sin algún procesamiento
- UpdateDBCommand: Clase comando para actualizar la base de datos
- WebCaching: Clase que en caso la página no se encuentre disponible, sea capaz de proporcionar información bruta
- Scrap: Clase que realiza el procedimiento de *scrapeo*
- UpdateData: Clase empleada para realizar actualizaciones a la base de datos
- AdminLogin: Clase empleada para realizar la comprobación de credenciales de un administrador
- UserLogin: Clase empleada para realizar la comprobación de credenciales de un usuario
- userServiceAPI: Clase empleada para manejar los comandos del usuario en la API
- SearchCommand: Clase comando utilizado para realizar búsquedas
- PaymentCommand: Clase comando utilizado para facturar las consultas realizadas por un usuario

3.2. Diagrama

Parte 1:



Parte 2:



4. Diagrama de Secuencia

4.1. Glosario

- Execute: ejecutar un comando
- Ok: mensaje que indica una solicitud hecha correctamente
- Fail: mensaje que indica un fallo al procesar una solicitud
- Authenticated: inició de sesión exitoso
- Unauthorized: acceso no autorizado
- Redirect: redirigir al usuario o administrador al servicio o comando correspondiente.

4.2. Diagrama

Diagrama de secuencia de administrador:

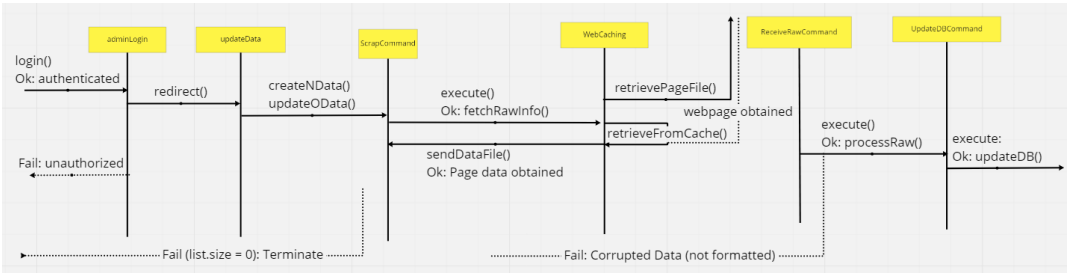
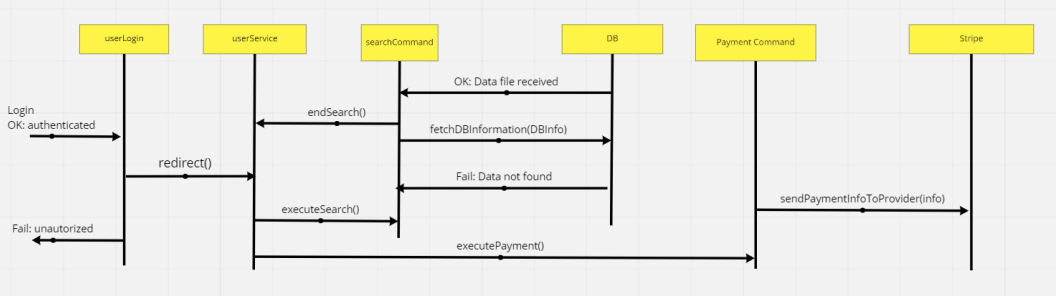


Diagrama de secuencia de usuario:



5. Tech Stack

5.1. Resumen

- No se desarrollará un *frontend*.
- Para el *backend* se utilizará Python y el Flask/Django framework.
- Para el *storage* se utilizará PostgreSQL y MongoDB.

5.2. Justificación

- No es necesario desarrollar un frontend ya que solo se necesita elaborar el API para realizar operaciones.
- Para el backend, se utilizará Python y Flask/Django porque se tiene experiencia previa con el lenguaje y los frameworks para desarrollar servicios web.
- Se creará el storage usando PostgreSQL porque permite realizar consultas simples o complejas en caso se necesite. Asimismo, también porque se tiene experiencia con PostgreSQL.
- Finalmente, se utilizará MongoDB para validar credenciales debido a su velocidad para realizar consultas, permitiendo verificar usuarios de manera rápida.