# Respuestas de la prueba técnica

Juan Fernando Otoya

Link Repositorio: https://github.com/Guaberx/Prueba_Tecnica_igniweb

## Analisis de requerimientos

**Requisitos Funcionales (RF)**

- RF1: Seleccionar un conjunto personalizado de criptomonedas.
- RF2: Mostrar precios, cambios porcentuales y volumen en tiempo real.
- RF3: Persistir datos históricos en base de datos.
- RF4: Consultar historial por rango de tiempo.
- RF5: Mostrar gráficos con evolución de precios.
- RF6: Actualización automática de datos (sin recargar la página).

**Requisitos No Funcionales (RNF)**

- RNF1: Aplicación de **una sola página (SPA)**.
- RNF2: UI responsiva para diferentes dispositivos.
- RNF3: Uso eficiente de la API (evitar sobrecarga).
- RNF4: Código versionado con Git (ramas claras).
- RNF5: Escalabilidad en caso de aumentar criptos y usuarios.

# Diseño de esquemas de la base de datos

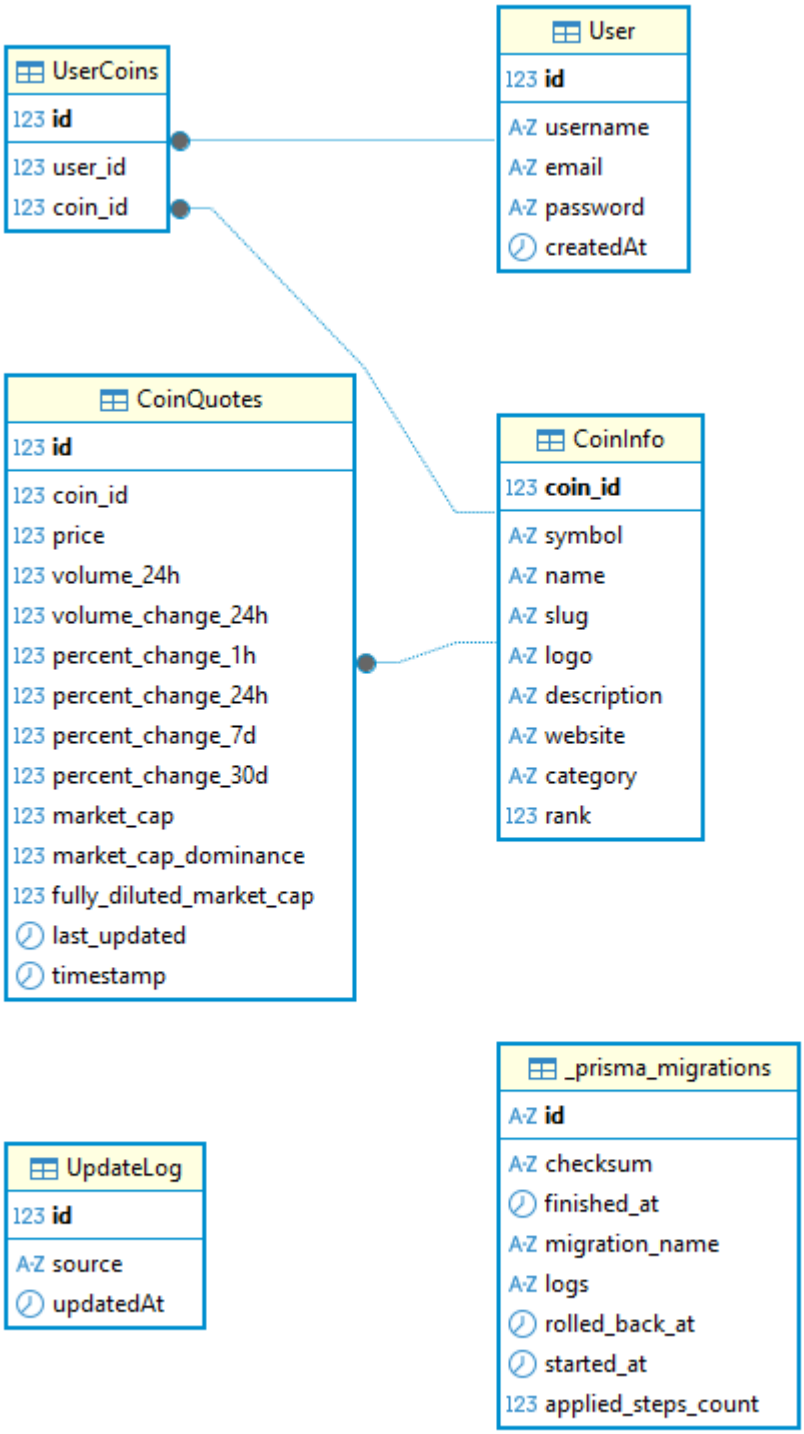
Basado en los endpoints expuestos por CoinMarketCap

- https://coinmarketcap.com/api/documentation/v1/#operation/getV3CryptocurrencyQuotesHistorical
- https://coinmarketcap.com/api/documentation/v1/#operation/getV2CryptocurrencyQuotesLatest
- https://coinmarketcap.com/api/documentation/v1/#operation/getV1CryptocurrencyListingsLatest
- https://coinmarketcap.com/api/documentation/v1/#operation/getV1CryptocurrencyListingsHistorical
- https://coinmarketcap.com/api/documentation/v1/#operation/getV2CryptocurrencyInfo

Decido crear una base de datos Para almacenar datos recurrentes como la lista de cripto monedas y su metadata. Esto queda almacenado en CoinInfo. Luego, cada una de sus variaciones de precio “quotes” esta en CoinQuotes.

Se agrego también una tabla de usuarios para almacenar la selección de monedas preferidas del usuario.

En cuanto a la tabla de UpdateLog, esta se utiliza para llevar constancia de las ultimas actualizaciones en la base de datos del listado de monedas y sus metadatos porque estas son operaciones costosas. Funciona como un “cache”, que es actualizado diariamente en el backend, utilizando la librería crono-node


UserCoins
123 id
123 user_id
123 coin_id
User
123 id
A-Z username
A-Z email
A-Z password
createdAt
CoinQuotes
123 id
123 coin_id
123 price
123 volume_24h
123 volume_change_24h
123 percent_change_1h
123 percent_change_24h
123 percent_change_7d
123 percent_change_30d
123 market_cap
123 market_cap_dominance
123 fully_diluted_market_cap
last_updated
timestamp
CoinInfo
123 coin_id
A-Z symbol
A-Z name
A-Z slug
A-Z logo
A-Z description
A-Z website
A-Z category
123 rank
UpdateLog
123 id
A-Z source
updatedAt
_prisma_migrations
A-Z id
A-Z checksum
finished_at
A-Z migration_name
A-Z logs
rolled_back_at
started_at
123 applied_steps_count

# Diseño de los endpoints

Se ha creado un backend utilizando Express en NodeJS

Se adjunta una colección de Postman para mostrar los endpoints creados

GET /coins?query=<symbol,name,slug>

Retorna las monedas que se pueden identificar por symbol, name o slug

Ejemplo: GET /coins?query=BTC,ETH,Ethereum

GET /coins/rank?start=0&limit=100

Retorna las monedas con mejor ranking paginado desde start y los siguientes “limit” elementos

GET /coins/historical?identifiers=3912,2594

Retorna informacion sobre los precios y variación de precios en un periodo de tiempo de las monedas dadas en “identifiers”. Tiene que ser el id de las cripto monedas que CoinMarketCap le asigna a cada una

GET /coins/latest?ids=3912,2594

Retorna informacion sobre el ultimo precio de una lista de criptomonedas

POST /users/signup

Crea un usuario

body:{

username,

password,

Email }

GET /watchlist/:userID

Retorna las monedas que el usuario quiere tener en cuenta

En esta caso /watchlist/:userID tiene un CRUD completo lo que le permite al usuario agregar, modificar y borrar sus monedas seleccionadas