Segunda entrega

–

Recogida de datos desde la API

Grupo: grupo_70

Autores: Bogdan Razvant, Diego Suárez

Fecha: 09/03/2025

Descripción

Para la recogida de la información sobre criptoactivos hemos utilizado la API de coingecko. Hemos utilizado esta API por distintos motivos:

- Es una web muy popular para revisar información sobre criptoactivos.
- API muy intuitiva.
- API basada en la arquitectura REST.
- Autenticación sencilla.
- El número de peticiones disponibles es muy elevado.
- Gran variedad de endpoints.

Información otorgada por la API

Para la visualización de los endpoints y de la aclaración de la API se puede consultar la página oficial.

Entre muchos de los endpoints que tiene la API hemos seleccionado los que nos parecen los más interesantes para mostrar al usuario en los listados de la página:

Información sobre cada criptomoneda

Para obtener la información más básica como el nombre de las criptomonedas, el símbolo o “ticker”, el precio y las gráficas vamos a utilizar las siguientes URLS:

- https://api.coingecko.com/api/v3/coins/markets?vs_currency={currency}&ids={coinId1},{coinId2},{coinId3}

- https://api.coingecko.com/api/v3/coins/{coinId}/market_chart?vs_currency={currency}&days={numDays}

Para la "currency" vamos a utilizar EUR como base.

La primera URL nos va a ayudar a localizar el precio, volumen, cápita de mercado y otros parámetros de cada una de las criptomonedas que van definidas por el id.

La segunda sin embargo solo tiene en cuenta un solo id y nos devuelve información sobre el volumen, el cápita y el precio a lo largo del tiempo (como una gráfica).

El problema que vamos a tener es que no conocemos la id de cada criptomoneda sin utilizar el endpoint que nos provee con ello coins/list. Pero este endpoint lo que nos da es TODOS los coins, lo que haremos es escoger unos cuantos y guardarlos, nosotros tenemos en total 15.

Información sobre páginas de intercambio de criptomonedas

A las páginas que se utilizan para el intercambio de criptomonedas las vamos a llama "exchanges" (en inglés). Para ello vamos a usar su propia URL:

- https://api.coingecko.com/api/v3/exchanges

De aquí seleccionaremos los diez primeros que nos salgan, ya que suele devolver muchos y los primeros siempre suelen ser los más populares.

Información sobre criptoactivos populares

Querremos conocer que monedas y que NFTs están ahora en auge para poder investigarlas. Para ello vamos a usar la siguiente URL:

- https://api.coingecko.com/api/v3/search/trending

Este endpoint nos va a otorgar información sobre las 15 monedas más populares del momento y de los 7 NFTs con más carrerilla del momento.

Aclaraciones

Hemos decidido no hacer una tabla NFTs porque ya sería mucha información y mucho más trabajo teniendo ya en cuenta las criptomonedas en sí. También porque el trabajo se centra en sí en las criptomonedas.

Tampoco se han puesto los formatos de respuesta porque... La verdad son muy extensos y bastante complejos (además de largos) en si, por lo tanto los hemos excluido.

Estructura de la base de datos

Se van a definir ahora las tablas empleadas para esta entrega. Se van a definir tablas para guardar las criptomonedas, sus gráficas, precios y más información:

Tabla COINS

Script de creación

```
CREATE TABLE `FINAL_COINS` (
  `ID_COIN` int NOT NULL,
  `id` varchar(128) COLLATE utf8_spanish2_ci NOT NULL,
  `symbol` varchar(128) CHARACTER SET utf8 COLLATE utf8_spanish2_ci NOT NULL,
  `name` varchar(128) CHARACTER SET utf8 COLLATE utf8_spanish2_ci NOT NULL,
  `image` varchar(256) COLLATE utf8_spanish2_ci DEFAULT NULL,
  `current_price` int DEFAULT NULL,
  `market_cap` bigint DEFAULT NULL,
  `price_change_percentage_24h` double DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3 COLLATE=utf8_spanish2_ci;
```

Estructura de la tabla

Nombre	Tipo	Descripción
ID_COIN	int	Identificador único de la moneda
id	varchar(128)	ID de la moneda en la API externa

symbol	varchar(128)	Símbolo de la moneda (ej. BTC, ETH)
name	varchar(128)	Nombre completo de la moneda
image	varchar(256)	URL de la imagen de la moneda
current_price	int	Precio actual de la moneda
market_cap	int	Capitalización de mercado
price_change_percentage_24h	double	Cambio porcentual del precio en 24h

Tabla COINS_CHART

Script de creación

```
CREATE TABLE `FINAL_COINS_CHART` (
  `ID_COINS_CHART` int NOT NULL,
  `ID_COIN` int NOT NULL,
  `unix_time` bigint NOT NULL,
  `price` double NOT NULL,
  `market_cap` double NOT NULL,
  `total_volume` double NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3 COLLATE=utf8_spanish2_ci;
```

Estructura de la tabla

Nombre	Tipo	Descripción
ID_COINS_CHART	int	Identificador único de la entrada en el gráfico
ID_COIN	int	Identificador de la moneda (relación con FINAL_COINS)
unix_time	int	Marca de tiempo en formato UNIX
price	double	Precio de la moneda en ese momento
market_cap	double	Capitalización de mercado en ese momento
total_volume	double	Volumen total operado en ese momento

Tabla EXCHANGES

Script de creación

```
CREATE TABLE `FINAL_EXCHANGES` (
  `ID_EXCHANGE` int NOT NULL,
  `id` varchar(32) COLLATE utf8_spanish2_ci NOT NULL,
  `name` varchar(32) COLLATE utf8_spanish2_ci NOT NULL,
  `year_established` int DEFAULT NULL,
  `country` varchar(32) CHARACTER SET utf8 COLLATE utf8_spanish2_ci DEFAULT NULL,
  `image` varchar(256) COLLATE utf8_spanish2_ci DEFAULT NULL,
  `trust_score` int DEFAULT '0',
  `trade_volume_24h_btc` double DEFAULT '0'
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3 COLLATE=utf8_spanish2_ci;
```

Estructura de la tabla

Nombre	Tipo	Descripción
ID_COINS_CHART	int	Identificador único del exchange
id	varchar(32)	ID del exchange en la API externa
name	varchar(32)	Nombre del exchange
year_established	int	Año de fundación del exchange
country	varchar(32)	País de origen del exchange
image	varchar(256)	URL de la imagen del exchange
trust_score	int	Puntuación de confianza del exchange
trade_volume_24h_btc	double	Volumen de comercio en 24h en BTC

Tabla TRENDING_COINS

Script de creación

```
CREATE TABLE `FINAL_TRENDING_COINS` (
  `ID_TRENDING_COINS` int NOT NULL,
  `id` varchar(64) COLLATE utf8_spanish2_ci NOT NULL,
  `name` varchar(64) COLLATE utf8_spanish2_ci NOT NULL,
  `thumbnail` varchar(256) CHARACTER SET utf8 COLLATE utf8_spanish2_ci DEFAULT
NULL,
  `price` double DEFAULT NULL,
  `price_change_percentage_24h` double DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3 COLLATE=utf8_spanish2_ci;
```

Estructura de la tabla

Nombre	Tipo	Descripción
ID_TRENDING_COINS	int	Identificador único de la moneda en tendencia
id	varchar(64)	ID de la moneda en la API externa
name	varchar(64)	Nombre de la moneda
thumbnail	varchar(256)	URL de la imagen en miniatura
price	double	Precio actual de la moneda
price_change_percentage_24h	double	Cambio porcentual en 24h

Tabla TRENDING_NFTS

Script de creación

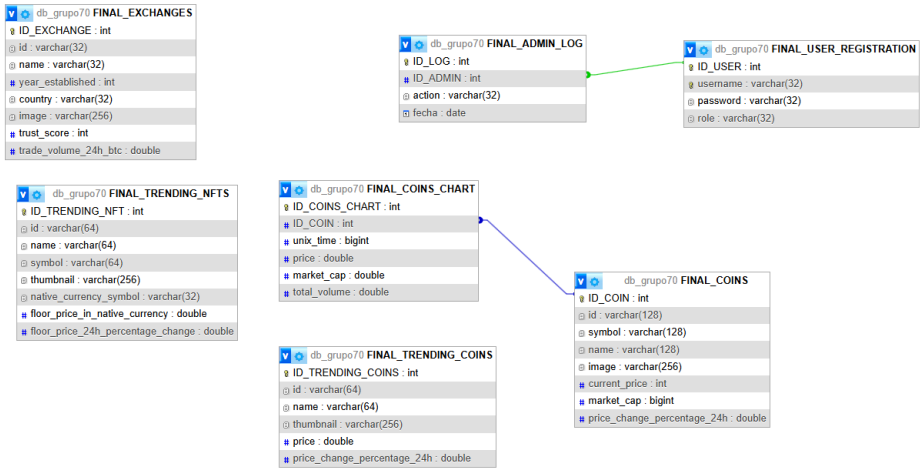
```
CREATE TABLE `FINAL_TRENDING_NFTS` (
  `ID_TRENDING_NFT` int NOT NULL,
  `id` varchar(64) COLLATE utf8_spanish2_ci NOT NULL,
  `name` varchar(64) COLLATE utf8_spanish2_ci NOT NULL,
  `symbol` varchar(64) COLLATE utf8_spanish2_ci NOT NULL,
  `thumbnail` varchar(256) CHARACTER SET utf8 COLLATE utf8_spanish2_ci DEFAULT
NULL,
  `native_currency_symbol` varchar(32) CHARACTER SET utf8 COLLATE
utf8_spanish2_ci DEFAULT NULL,
  `floor_price_in_native_currency` double DEFAULT NULL,
  `floor_price_24h_percentage_change` double DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3 COLLATE=utf8_spanish2_ci;
```

Estructura de la tabla

Nombre	Tipo	Descripción
ID_TRENDING_NFT	int	Identificador único del NFT en tendencia
id	varchar(64)	ID del NFT en la API externa
name	varchar(64)	Nombre del NFT
symbol	varchar(64)	Símbolo del NFT
thumbnail	varchar(256)	URL de la imagen en miniatura
native_currency_symbol	varchar(32)	Símbolo de la moneda nativa del NFT
floor_price_in_native_currency	double	Precio mínimo del NFT en la moneda nativa
floor_price_24h_percentage_change	double	Cambio porcentual en el precio mínimo en 24h

Estructura de la base de datos

Como se puede observar la base de datos no está tan interconectada como se puede desear por varios factores que no tuvimos en cuenta. La complejidad de la API y la falta de previsión al hacer los trending coins y NFTs complicaron el intercambio de información entre todas las tablas.



Aclaraciones

Se puede ver que la tabla TRENDING_ COINS no tiene una clave foránea a COINS. Esto se debe a que no siempre va a existir el coin en la base de datos, porque no suele ser el que más capitalización de mercado tiene, por ejemplo. Nosotros hemos escogido los 15 coins en función a la capitalización de mercado.

También hemos decidido no hacer dos tablas separadas para el precio de la criptomoneda y la información de la misma, ya que no nos proporcionaría demasiado valor. Aunque se acerque más al modelo relacional, teniendo solo una tabla facilita la recogida y la inserción de los datos porque es casi igual a la estructura que te devuelva la API.

Validación

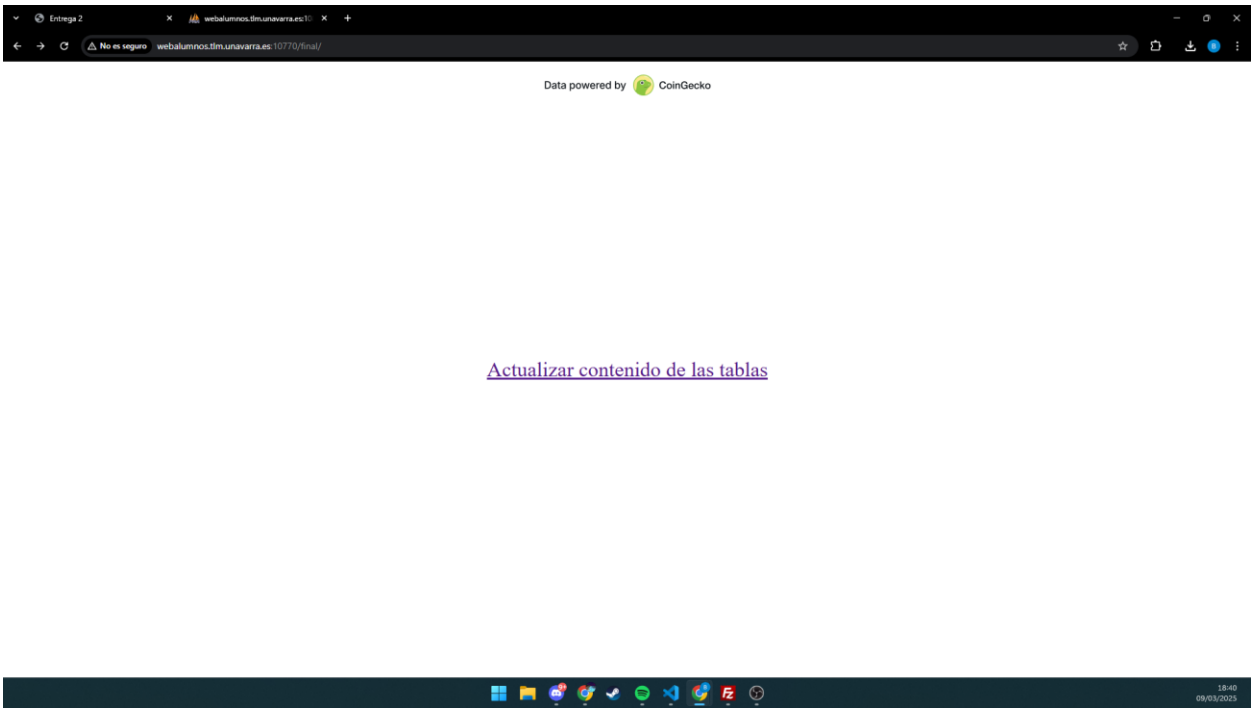
Para poder validar nuestro trabajo adecuadamente lo vamos a dividir en varias partes; cómo se va a validar, que código se emplea para validar y las pruebas.

Cómo se va a validar

En la carpeta de producción del servidor (carpeta final) se ha preparado una página web muy sencilla con un texto en el centro que, al clicarlo llevará al script de validación.

Hay que tener **MUCHO** cuidado que no clicarlo muchas veces porque la API tiene un límite de 30 llamadas por minuto, por lo tanto, podría dar error si se ejecuta más de una vez en ese periodo de tiempo. También, después de clicar va a tardar un rato en reaccionar

porque está haciendo toda la funcionalidad, no es algo dinámico, habrá que esperar unos 30 segundos.



Código de verificación

Este código de verificación nos va a servir más tarde también para la funcionalidad del administrador de la web que permite actualizar los datos de la base de datos. Ahora voy a comentar de forma funcional el código de verificación.

Recogida de datos de la API

Mediante los endpoints de los que nos provee la API de Coingecko vamos a recoger toda la información.

Primero se obtiene el contenido de la llamada al endpoint y después se construye el array que nos va a servir más adelante para insertar de manera más sencilla en la base de datos toda la información.

```php
echo "Contactando con la API...<br>";

// Conseguir respuestas de la API y construir arrays
$coinsResponse = CoinRequest::getCoinResponseContent();
$coinsArrayResponse = Coin::constructCoinArray(responseArray: $coinsResponse);

$coinsChartsArrayResponse = [];
foreach (MAIN_COINS_ID as $coinId) {
    $intCoinId = MAIN_COINS_ID_INT[$coinId];
    $coinChartArrayResponse = CoinChartRequest::getCoinChartResponseContent( id: $coinId);
    $coinChartInstanceArrayResponse = CoinChartInstance::constructCoinChartArray(responseArray: $coinChartArrayResponse, ID_COIN: $intCoinId);
    $coinsChartsArrayResponse[] = $coinChartInstanceArrayResponse;
}

$exchangeResponse = ExchangeRequest::getExchangeResponseContent();
$exchangeArrayResponse = Exchange::constructExchangeArray(responseArray: $exchangeResponse);

$trendingResponse = TrendingRequest::getTrendingResponseContent();
$trendingCoinsArrayResponse = TrendingCoin::constructTrendingCoinArray(responseArray: $trendingResponse);
$trendingNftArrayResponse = TrendingNft::constructTrendingNftArray(responseArray: $trendingResponse);
```

Eliminación de información de las tablas

Para que no haya inconsistencias en los datos antes de insertar todo se va a limpiar todo el contenido que tiene dentro.

```php
echo "Eliminando contenido de las tablas...<br>";

// Eliminar el contenido que ya había en la base de datos
$ejecucionCorrecta = true;

try {
    $dbDeletor = new DBDeletor();

    $responseCode = $dbDeletor->deleteAllIntormationFromTables();
    if (!$responseCode) {
        echo "Ha habido algun problema eliminando el contenido de las tablas<br>";
        $ejecucionCorrecta = false;
    }
} catch (Exception $e) { echo "Error: " . $e->getMessage(); }
```

Inserción de los datos

Ya casi para cerrar vamos a insertar todos los datos que hemos ido obteniendo de las llamadas.

```php
// Insertar todo el contenido en las bases de datos dados los arrays
try {
    $dbInsertor = new DBInsertor();

    $fullInformationArray = buildFullInformationArray(coinsArrayResponse: $coinsArrayResponse, coinsChartsArrayRespons…$coinsChartsArrayResponse,
    exchangeArrayResponse: $exchangeArrayResponse, trendingCoinsArrayRe…$trendingCoinsArrayResponse,
    trendingNftArrayResponse: $trendingNftArrayResponse);

    $responseCode = $dbInsertor->insertAllInformation(tableElements: $fullInformationArray);
    if (!$responseCode) {
        echo "Ha habido algun problema insertando el contenido de las tablas<br>";
        $ejecucionCorrecta = false;
    }
} catch (Exception $e) { echo "Error: " . $e->getMessage(); }
```

Para no tenerle que pasar muchos parámetros a la función “insertAllInformation” construimos un array con los distintos campos que queremos insertar mediante una función propia.

```
    1 reference
78  function buildFullInformationArray( $coinsArrayResponse, $coinsChartsArrayResponse,
79                                      $exchangeArrayResponse, $trendingCoinsArrayResponse,
80                                      $trendingNftArrayResponse ) : array
81  {
82      return array (
83          "coins" => $coinsArrayResponse,
84          "coinsCharts" => $coinsChartsArrayResponse,
85          "exchanges" => $exchangeArrayResponse,
86          "trendingCoins" => $trendingCoinsArrayResponse,
87          "trendingNfts" => $trendingNftArrayResponse,
88      );
89  }
```

Comprobación final

Ya después de haber pasado por todo el código anterior, si todo ha salido correctamente se va a mostrar un mensaje informativo sobre la correcta actualización de las tablas y el momento en el cual se ha dado esa misma.

```
70  if ($ejecucionCorrecta) {
71      echo "Información de las tablas actualizada correctamente<br>";
72      echo "Fecha de actualización: " . date(format: 'Y-m-d H:i:s') . "<br>";
73  }
74  else {
75      echo "No se ha podido actualizar la información correctamente <br>";
76  }
```

Video demo sobre ejecución

Que mejor que una demostración breve y práctica sobre cómo funciona la verificación, el video está subido en la misma tarea.

Problemáticas y conclusiones

Aunque habíamos planteado muy bien la estructura de esta primera entrega y el trabajo en general nos han fallado varias cosas.

Primero de todo, después de estar desarrollando por bastante tiempo todo el código, la base de datos, las consultas... No probamos **NADA** en el servidor hasta el final lo que hizo que muchos problemas se acumulasen al final (al intentar probar todo). Esto hizo muy

pesado y complejo el desarrollo al intentar llevarlo al servidor de producción llegando a dar sustos como el que tuvimos.

Cloudfare (el servicio intermediario de coingecko) bloqueó las llamadas a la API de por ver que estaba haciendo llamadas con un formato de URL extraño e incorrecto. Esto casi hace que casi todo el trabajo se fuese al garete y que tuviésemos que mirar otra API.

Por problemas de compatibilidad, no hemos podido trabajar el mismo tiempo los dos integrantes, teniendo que poner más de su parte otra persona. Esto deberíamos haberlo previsto también trabajando con más antelación y con bastante más interés de antemano, que para la siguiente iteración lo tendremos en cuenta.

Otro problema que nos surgió, aunque no fuese grave es que el servidor está usando una versión entera más baja de PHP de la que se usa actualmente (la 7.2) lo que hizo que tuviésemos que cambiar el código al probarlo en el servidor. Nosotros estábamos trabajando con la versión de PHP 8.2.

Por último, para concluir con la documentación hay que hablar sobre la API.

No me esperaba menos de una API pública de tales características de una web muy grande. Pero, aunque estuviese muy bien explicada era bastante complejo encontrar lo que buscabas, y el formato de respuesta de las peticiones es bastante extenso y con mucha información. Nos tomó mucho tiempo solo averiguar qué era lo que necesitábamos de todo el contenido y después convertirlo todo en código.