

A dark blue vertical bar on the left side of the page. A blue arrow points to the right from the bar, containing the date.

7-2-2025

Twitch analytics

Entrega 1

Several thin, curved, light blue lines that originate from the bottom left and sweep upwards and to the right, creating a sense of movement or growth.

Autores:

ADRIAN JUAREZ, ISRAEL HUALDE, SAUL ALEN, UNAI MARTINEZ

ÍNDICE

1.	API de Twitch	3
2.	Servidor Público.....	3
3.	Realización de código.....	4
4.	Distribución de tareas.....	5
5.	Conclusión	6

1. API de Twitch

El primer desafío al que nos enfrentamos fue obtener la información de la API de Twitch. Tuvimos dificultades para comprender todos los requisitos necesarios, como el uso del Client ID y el Client Secret, así como el proceso para realizar una solicitud a Twitch y obtener el token de acceso, junto con su caducidad y el mecanismo de renovación.



2. Servidor Público

El segundo problema principal fue encontrar un servidor público adecuado. Inicialmente, intentamos localizar una opción gratuita, pero al no encontrar una solución viable, optamos por contratar un servidor en IONOS. Sin embargo, la tramitación de nuestra solicitud tomó más tiempo del esperado.



IONOS

3. Realización de código

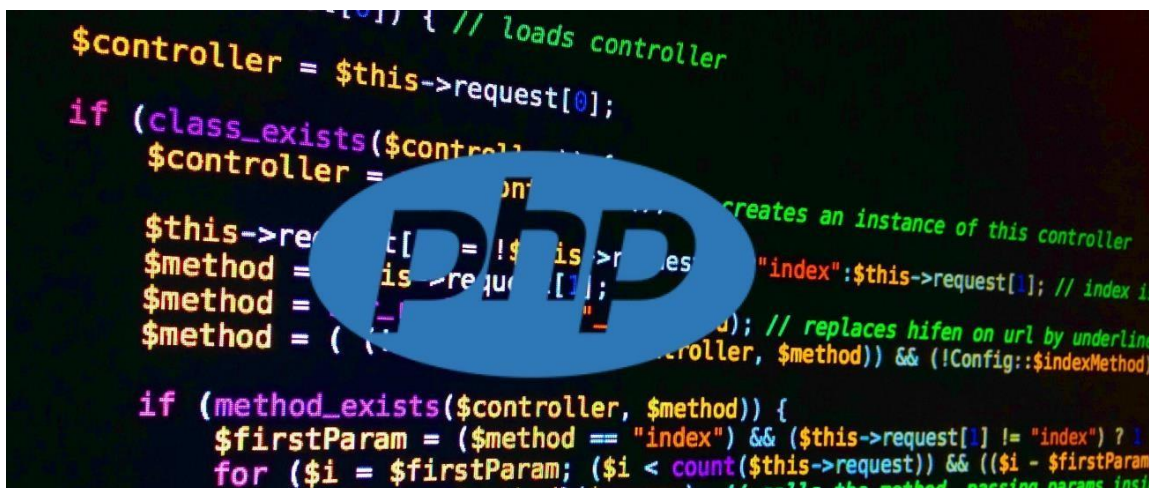
En cuanto al código, programamos usando la versión más reciente (a 07/02/2025) de PHP. Dado que trabajamos con tres casos distintos, decidimos no crear tres páginas separadas. En su lugar, implementamos un enfoque más modular en el archivo index.php, que, según el atributo recibido en la URL, dirige la petición a uno de los tres casos ubicados en la carpeta src. Además, para mantener el código más limpio y estructurado, agrupamos en un archivo aparte aquellas funciones compartidas entre los tres casos.

Hemos decidido implementar el código de esta manera para evitar redundancias y mejorar la mantenibilidad del proyecto. Al centralizar la lógica en un solo archivo y dividir los casos en módulos, facilitamos futuras modificaciones y añadidos sin necesidad de alterar múltiples archivos. Además, esta estructura modular permite una mejor organización del código y una ejecución más eficiente, ya que cada parte del sistema cumple un rol específico sin afectar el funcionamiento general.

Durante el desarrollo, surgió la duda sobre si debíamos incluir comentarios en el código. Inicialmente, queríamos mantenerlo lo más limpio posible, pero finalmente decidimos integrar comentarios principales para clarificar su contenido y facilitar su comprensión. Estos comentarios destacan los aspectos clave del código, ayudando tanto a los integrantes del equipo como a futuros desarrolladores que necesiten trabajar con el proyecto.

Otra duda que nos surgió fue a la hora de implementar funciones en los casos de uso que se repetían. En un principio pensamos en hacer simplemente los tres casos de uso, pero decidimos hacer un archivo con funciones comunes para englobar funcionalidades usadas por varios casos de uso.

Nos aparecieron problemas a la hora de implementar la aparición del error 401 para todos los casos de uso. Se nos ocurrió hacer que el usuario pueda (mediante la URL) escribir él mismo un token, pero en caso de no introducir ningún token se establece por defecto un token, de tal forma que si lo hace de manera incorrecta emergiera el error 401.



```
    } // loads controller
    $controller = $this->request[0];
    if (class_exists($controller)) {
        $controller = new $controller(); // creates an instance of this controller
        $this->request[1] = $this->request[1];
        $method = $this->request[2];
        $method = ($method == "index") ? "index" : $method; // replaces hifen on url by underline
        $method = ($method == "index") ? "index" : $method; // replaces hifen on url by underline
        $method = ($method == "index") ? "index" : $method; // replaces hifen on url by underline
        if (method_exists($controller, $method)) {
            $firstParam = ($method == "index") && ($this->request[1] != "index") ? 1 : 0;
            for ($i = $firstParam; $i < count($this->request); $i++) {
                // calls the method passing params inside
```

4. Distribución de tareas

En este proyecto nos hemos distribuido las tareas en 4 bloques, de tal forma que cada integrante de Easy Money se ha focalizado en su correspondiente función.

Uno de nosotros se encargó de la creación del código requerido para el correcto funcionamiento de los tres casos de uso requeridos. A su vez, también se encargó de la comprobación del código utilizando la herramienta XAMPP, comprobando así la ejecución del código y los diferentes códigos obtenidos (Código 200, 404, etc).

Por otra parte, otra tarea de la que se encargó uno de nosotros fue realizar el código correspondiente a index.php y las funciones comunes, siendo este un código que hace de nexo entre los diferentes casos de uso, facilitando así la ejecución de los mismos.

Como tercer rol, uno de nosotros principalmente se encargó de la obtención del dominio web y del servidor, en el cual se implementó el programa entregado. También ayudó con la creación del código del tercer caso de uso.

Por último, el cuarto integrante tuvo la función de realizar los documentos de texto a entregar (tanto Entrega1.pdf como el ReadMe). Otra tarea que tuvo fue comprobar los archivos subidos a GitHub y al servidor, verificando así la correcta inclusión del trabajo en el repositorio y en la página web. También fue el encargado de supervisar el código de los casos de uso para controlar el correcto funcionamiento del programa.

Pese a que todos tuvimos nuestras tareas concretas en el proyecto, habitualmente compartíamos nuestros avances y colaborábamos con el trabajo del compañero, creando así un proyecto más sólido y robusto.

5. Conclusión

A lo largo de este proyecto, logramos implementar un programa en PHP que obtiene datos a través de la API pública de Twitch, superando diversos desafíos técnicos y organizativos. Desde la configuración del acceso a la API y la gestión del servidor hasta la modularización del código y la distribución de tareas, cada paso fue clave para el éxito del desarrollo. Gracias a una estructura bien definida y una colaboración constante entre los integrantes de Easy Money, conseguimos un sistema funcional y bien estructurado. Este proceso no solo nos permitió adquirir experiencia en el uso de la API de Twitch y la programación en PHP, sino también fortalecer nuestras habilidades en trabajo en equipo y resolución de problemas.