Presentación

Infraestructura de Digital Money

Digital Money es una aplicación de billetera virtual, diseñada para ofrecer una experiencia similar a plataformas como Mercado Libre o PayPal. Su objetivo es permitir a los usuarios realizar transacciones digitales de manera fácil y segura, gestionando su dinero de forma eficiente a través de una interfaz web. La infraestructura detrás de Digital Money está pensada para manejar la autenticación de usuarios, el almacenamiento seguro de tokens de sesión, y la interacción con servicios externos.

A continuación, presento el diseño de la infraestructura necesaria para garantizar que esta aplicación funcione de manera eficiente y segura, utilizando tecnologías modernas como Next.js, Redis, y Docker.

Componentes Principales de la Infraestructura

1. Aplicación Web con Next.js La aplicación web está construida con Next.js, que se encarga tanto del lado del cliente (interfaz de usuario) como del lado del servidor (API interna). Next.js maneja varias rutas API, incluyendo la ruta de autenticación, gestión de sesiones y manejo de información del usuario.
2. API en Java Nuestra aplicación interactúa con una API en Java, que es la encargada de autenticar usuarios y generar los tokens necesarios para las sesiones. Esta API es crucial para el inicio de sesión, ya que devuelve un token que luego almacenamos para mantener la sesión activa en Redis.
3. Redis para Gestión de Sesiones Para manejar las sesiones de usuario de manera eficiente, utilizamos Redis. Redis almacena los tokens de sesión con un tiempo de expiración definido, lo que permite que las sesiones se gestionen de forma rápida y segura. Redis también ofrece una recuperación rápida de estos tokens para validar las solicitudes de usuario.
4. Cookies para Persistencia en el Cliente Utilizamos cookies para mantener el estado de la sesión del lado del cliente. Cookies como digitalMoneyID y digitalMoneyEmail permiten que las sesiones persistan entre distintas solicitudes, garantizando que el usuario no tenga que autenticarse continuamente.

Diagrama de la Arquitectura

El diagrama a continuación ilustra cómo los distintos componentes interactúan entre sí.

1. El usuario envía una solicitud a través del navegador a la API interna de Next.js.
2. Next.js redirige esta solicitud a la API en Java para autenticar al usuario. La API devuelve un token de sesión.
3. Redis almacena este token de sesión para mantenerlo accesible durante un periodo de tiempo específico.
4. Next.js usa este token para validar la sesión y, mediante cookies, permite al usuario interactuar con la aplicación de forma segura.

Contenedores y Docker

La infraestructura está completamente contenida en Docker, lo que asegura que todos los servicios se ejecuten de manera aislada y eficiente.

* Next.js se ejecuta en un contenedor web.
* Redis está en otro contenedor dedicado, que asegura la persistencia de los tokens de sesión.
* Ambos contenedores están conectados a través de una red de Docker (digitalMoney\_network), permitiendo una comunicación fluida y segura entre ellos.

El uso de Docker garantiza que la aplicación sea fácilmente desplegable y escalable, tanto en entornos de desarrollo como en producción.

Docker Compose

El archivo docker-compose.yml define los servicios y redes que conectan nuestra infraestructura. Aquí se describen los aspectos más importantes de este archivo:

* Servicio Web (Next.js): Maneja las solicitudes del cliente y está expuesto en el puerto 3000.
* Redis: Utiliza el puerto 6379 y almacena tokens de sesión con seguridad, protegido por una contraseña (digitalMoneyPass).
* Red de Docker: Ambos servicios están conectados en una red interna (digitalMoney\_network), asegurando una comunicación privada entre los servicios sin exponer Redis directamente al exterior.

Manejo de Variables de Entorno

Las configuraciones sensibles, como URLs y claves API, están gestionadas mediante variables de entorno definidas en el archivo .env.local. Esto incluye:

* API\_JAVA: El endpoint de la API en Java.
* REDIS\_URL: La URL que se utiliza para conectarse a Redis.
* REDIS\_API\_TOKEN: Un token de acceso para interactuar con Redis.

Esto permite una configuración flexible que puede ser fácilmente adaptada en diferentes entornos (desarrollo, producción, etc.).

Conclusión

La infraestructura de Digital Money está diseñada para ser escalable, segura y eficiente. Al combinar la potencia de Next.js, la velocidad de Redis, y la robustez de la API en Java, aseguramos que el sistema maneje de manera efectiva las autenticaciones de usuario y la gestión de sesiones.

Este diseño modular, implementado a través de Docker, permite que la aplicación sea fácilmente desplegable y mantenible, garantizando la seguridad y rapidez necesarias para su operación.