Memoria de Proyecto

Ingeniería web

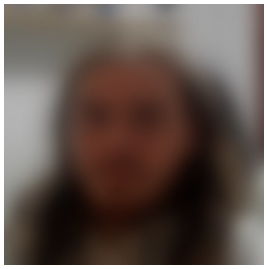
Grupo 13

Octubre 2021

Líder: Salcedo Navarro, Andoni (785649)

Pelayo Benedet, Tomás (779691)

Subías Rodríguez, Rubén (759406)



Velasco Calvo, Isaac (758986)

1. Introducción y Objetivo

Se ha creado una aplicación web, basada principalmente en SpringBoot[2] y Kotlin[1], buscando crear un servicio de acortación de URLs Web, añadiendo funcionalidades adicionales.

Los objetivos y funcionalidades principales que se querian añadir a esta aplicación son los siguientes:

- Los usuarios deben poder iniciar sesión y poder obtener información de aquellas URLs que han acortado en el pasado.
- La aplicación debe verificar que las URLs que se acorten sean alcanzables.
- Un sistema de redirección para que el usuario pueda monetizar (La monetización está simulada con anuncios falsos)

2. Organización

Las tareas han podido ser divididas entre los miembros del equipo, con cada uno de ellos pudiendo hacer lo siguiente:

- **Andoni Salcendo**: Organización de proyecto y tareas, ????
- **Tomás Pelayo**: ????
- **Rubén Subías**: ????
- **Isaac Velasco**: ????

3. Funcionalidades y Tecnologías

La mayoría de la interacción entre Backend y Frontend se realiza mediante una REST API creada con SpringBoot[2]

3.1. Gestión de URLs

La gestión de URLs trata sobre el guardado de nuevas URLs acortadas y su recuperación, al igual que la gestión de códigos QR.

Para esto, se utiliza un Controlador de URLs (`controllers/UrlController.kt`), el cual usa una REST API para permitir todas las acciones que debe. Para acceder a este controlador, se crea un punto de acceso con la URL `/api`.

La principal funcionalidad que permite este controlador es acortar URLs a partir del endpoint `/api/shorter`, el cual, al darle una URL completa, primero comprueba que no exista ya en la base de datos. Si existe, devuelve la URL ya acortada, si no, comprueba si esta página es alcanzable con la función asincrónica, genera la versión acortada, la guarda en la base de datos y la devuelve al usuario que realizo la petición. También genera el código QR si este ha sido pedido también.

También se puede recuperar aquellas URLs que han sido acortadas en el pasado por un usuario registrado en el endpoint `/api/user/urls` y el código QR de una URL acortada si existe con `/api/qr`.

3.2. Gestión de Usuarios

TODO: Esto

3.3. Gestión de Tiempo de Espera y Anuncios

En la pantalla de espera, donde el usuario debe esperar 10 segundos antes de recibir la URL des-acortada, se utiliza un WebSocket para, a la vez, forzar al usuario a esperar el tiempo de espera deseado y obtener la URL real de una URL acortada.

Para esto configuramos un WebSocket de Springboot en `UrlShorterApplication.kt` y en `controllers/WSController.kt`. En el primer archivo, se realiza la configuración básica del websocket con la clase `WSConfig`, como el endpoint donde se accede (`/wstimer`) e indicar a websocket que utilize cierta clase como controlador del WebSocket. Esta clase es el controlador `controllers/WSController.kt` la cual gestiona que debe de hacer el WebSocket cuando recibe una petición desde un cliente, en este caso, esperar 10 segundos y devolver la URL real que ha pedido el usuario.

También en la pantalla de espera, se muestran anuncios elejidos de forma aleatoria por el backend. Esto se realiza con una controlador REST API en `controllers/AdController.kt`, el cual responde a aquellas peticiones HTTP al endpoint `/ad/obtain` con dos URLs a imagenes de anuncios falsos elejidos de una lista estática en el backend.

3.4. Asincronía

3.5. Frontend

React[4] es una librería permite crear interfaces gráficas en páginas web.

Esta aplicación utiliza esta en su Frontend, donde la página se divide en diversas categorias/pantallas:

- Página de Inicio (`UrlPage.jsx`): Esta pantalla es la que un nuevo usuario ve cuando se introduce por primera vez a la página web. Aquí se puede pedir recortar URLs, generar QRs, Iniciar Sesión y Registrarse.
- Página de Inicio para Usuarios Registrados (`UrlUserPage.jsx`): Parecia a la pantalla anterior, pero con funcionalidades solo disponibles a usuarios registrados, como poder ver las URLs acortadas por este en el pasado y desconectarse de la aplicación.
- Página de Inicio de Sesión (`SingIn.jsx`): Usada para iniciar sesión en la aplicación web.
- Página de Registro (`SignUp.jsx`): Usada para registrarse en la aplicación web.
- Página de Espera (`WaitPage.jsx`): La página de espera que aparece cuando se intenta utilizar una URL acortada. Aquí, el usuario tendra que esperar 10 segundos para poder continuar a la URL deseada. Para poder esperar estos 10 segundos, se utiliza

el WebSocket descrito en el apartado 3.3. También se muestran una serie de anuncios falsos, escojidos aleatoriamente de una colección por el backend, y se muestran al usuario.

Para poder comunicarse con el backend, exceptuando el uso de WebSockets para el tiempo de espera, se utiliza la libreria Axios[5] para poder realizar peticiones HTTP.

4. Pruebas

Referencias

[1] Kotlin Programming Language - https://kotlinlang.org/

[2] SpringBoot - https://spring.io/projects/spring-boot

[3] SpringBoot's Websockets - https://spring.io/guides/gs/messaging-stomp-websocket/

[4] React - https://reactjs.org

[5] Axios Documentation - https://axios-http.com/docs/intro