Micro projet Spring

Vous devez développer un *URL Shortener* sous forme d'API REST.

Cette API servira à créer et supprimer des liens raccourcis.

On pourra accéder aux liens raccourcis via une URL courte, qui redirigera vers l'URL réelle.

Par exemple, si je fournis cette URL :

```
https://www.google.com/
```

L'API me répondra avec un identifiant court (par exemple `aX8Pm2wt`) que je peux utiliser dans une URL courte.

```
http://localhost:8080/aX8Pm2wt
```

En y accédant, je suis redirigé automatiquement vers l'URL réelle, autrement dit Google.

Services à implémenter

Voici les services à implémenter et leur fonctionnement :

Créer un lien raccourci

Requête :

```
HTTP/1.1 POST /links
Content-Type: application/json

"https://url-a-raccourcir.com/"
```

Réponse :

```
201 Created HTTP/1.1
Content-Type: application/json
X-Removal-Token: RRsdWMJGABe0OqMrzV0VVJKzqRZE8tpOtIrhZsmC

{
    "id": "2bffc207-6fd5-4e1d-afc3-b09b4d380416",
    "short-id": "aX8Pm2wt",
    "real-url": "https://url-a-raccourcir.com/"
}
```

En cas de lien invalide :

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

"invalid url"
```

Les règles concernant les liens valides/invalides sont les suivantes :

- Le serveur n'accepte que les URLs HTTP/HTTPS.
- Le serveur n'accepte pas les URL qui pointent vers notre propre shortener.

Supprimer un lien raccourci

Une requête de suppression d'un lien existant doit contenir le token fourni dans l'en-tête `X-Removal-Token` lors de la création du lien.

Requête :

```
DELETE /links/2bffc207-6fd5-4e1d-afc3-b09b4d380416 HTTP/1.1
X-Removal-Token: RRsdWMJGABe0OqMrzV0VVJKzqRZE8tpOtIrhZsmC
```

En cas de suppression effective :

```
HTTP/1.1 204 No Content
```

En cas de lien inexistant :

```
HTTP/1.1 404 Not Found
```

En cas de token invalide :

```
HTTP/1.1 403 Forbidden
```

Redirection lors du clic sur un lien raccourci

Pour profiter de nos liens raccourcis, on utilisera des requêtes de ce format:

```
GET /aX8Pm2wt HTTP/1.1
```

En réponse le serveur devra rediriger vers l'URL complète:

```
HTTP/1.1 302 Found
Location: https://url-a-raccourcir.com/
```

Nettoyer les liens

Tous les liens qui n'ont pas été accédés depuis plus de 30 jours doivent être supprimés de manière automatique.

Gestion des erreurs

Sauf pour les liens invalides où un corps de réponse d'erreur sera présent, toutes les erreurs doivent faire l'objet d'un corps de réponse vide.

Toutes les erreurs doivent faire l'objet d'une journalisation au format suivant :

```
<méthode> <chemin HTTP> from <addresse IP source>, <type de l'erreur>:
<message d'erreur> (<fichier source> => <ligne de code>)
```

Les erreurs non explicitement gérées dans le code doivent faire l'objet de code `500`.

Contraintes techniques

- Java 17 obligatoire avec Maven
- On utilise uniquement Spring et les librairies Java standards.
- Une attention devra être portée sur la structure du code. Il faudra essayer de décomposer l'application en plusieurs parties :
    - La partie Web/REST qui concerne la gestion des requêtes et des réponses HTTP.
    - La partie métier qui contient la logique de l'application.
    - La partie base de données qui contiendra notre couche de persistence.
- Pas d'utilisation de base de données relationnelle. Il faudra stocker les liens dans un fichier JSON. Le chemin du fichier devra être spécifié via une configuration externe au code de l'application. Au démarrage de l'application, le fichier devra être lu pour charger les donneés en mémoire.

Documentation technique

- Externaliser la configuration dans Spring: https://docs.spring.io/spring-boot/docs/current/reference/html/features.html#features.external-config
- Spring Web: https://docs.spring.io/spring-framework/reference/web/webmvc.html
- Les objets URL et URI en Java
    - URI https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/net/URI.html
    - URL https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/net/URL.html