Bénédicte Motto

Document de conception détaillée :

API RESTFull URL shortener

Table des matières

L'API décrite dans ce document fournie trois services :
- la création de lien court, plus facile à mémoriser que le lien d'origine
- la redirection de l'utilisateur vers le lien d'origine
- l'administration de ces codes courts, c'est-à-dire la visualisation du nombre d'appel du code court demandé par l'administrateur.

Les paragraphes suivants décrivent en détail le fonctionnement de chacun des services ainsi que le fonctionnement générale de l'application.

1. Fonctionnalité

1.1 La création de lien court

Dans un premier temps l'utilisateur souhaite enregistrer des liens courts afin de pouvoir les utiliser dans le future.

L'utilisateur a à sa disposition deux techniques permettant de créer des liens courts :

Il peut simplement donner le lien qu'il souhaite enregistrer et voir ce que l'API lui propose comme code court (voit figure suivante)

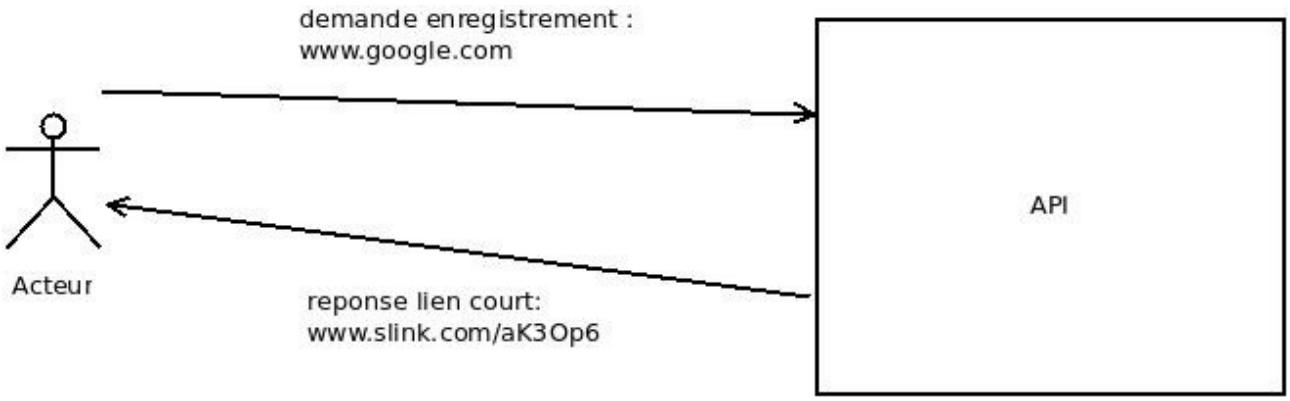


figure 1: Code court aléatoire associé au lien www.google.com

La deuxième possibilité consiste à fournir à l'API un paramètre (custom), ce paramètre permet d'utiliser cet élément dans le code court. En effet si l'élément n'est pas déjà utilisé comme code court pour un autre lien alors il sera directement utilisé, dans le cas contraire, un chiffre aléatoire lui sera ajouté afin de le rendre unique (voit figure)

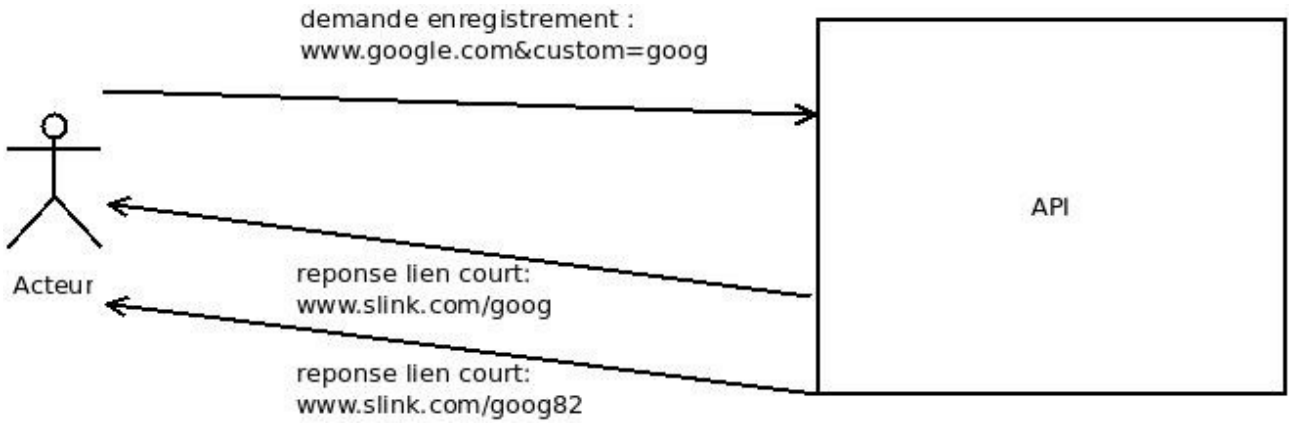


figure 2: Code court personnalisé à l'aide du paramètre custom

Pour les URL un peu plus longues contenant un ou plusieurs "/", l'utilisateur doit passer les

paramètres dans le body de la requête POST, ces arguments doivent ce trouver au format json, example:
- {"link":"www.youtube.com/watch?v=RoePjPQP7XE","custom":"millio"}

Par défaut, l'API prend les données transmises dans le json, si celui-ci est vide alors l'API prend ceux présents dans l'URL.

1.2 La redirection

L'utilisateur souhaite accéder à www.google.com, il lui envoie alors le lien court correspondant à ce lien et l'API le redirige.

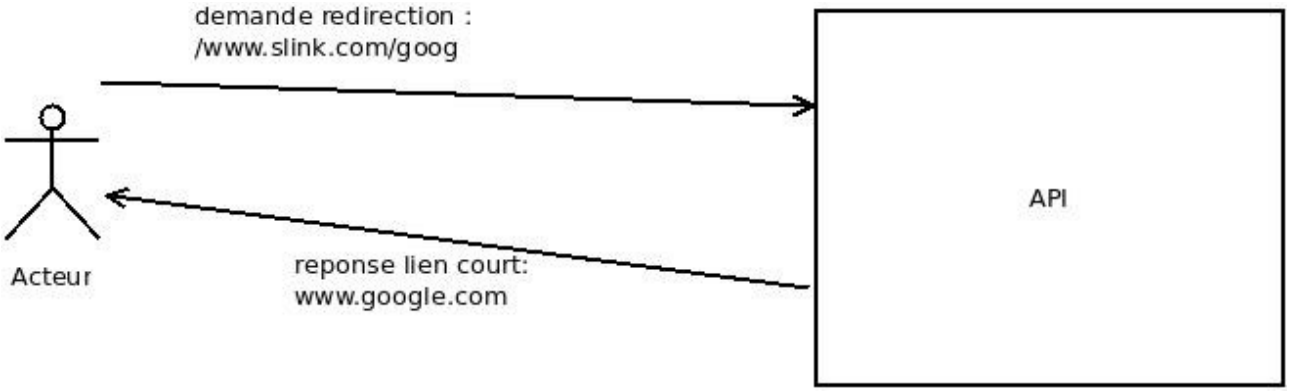


figure 3: Redirection vers le lien d'origine

1.3. L'administration

L'utilisateur souhaite savoir combien de fois un lien court a été utiliser. Il fournit alors le code court à l'API qui lui retransmet les informations au forma json.

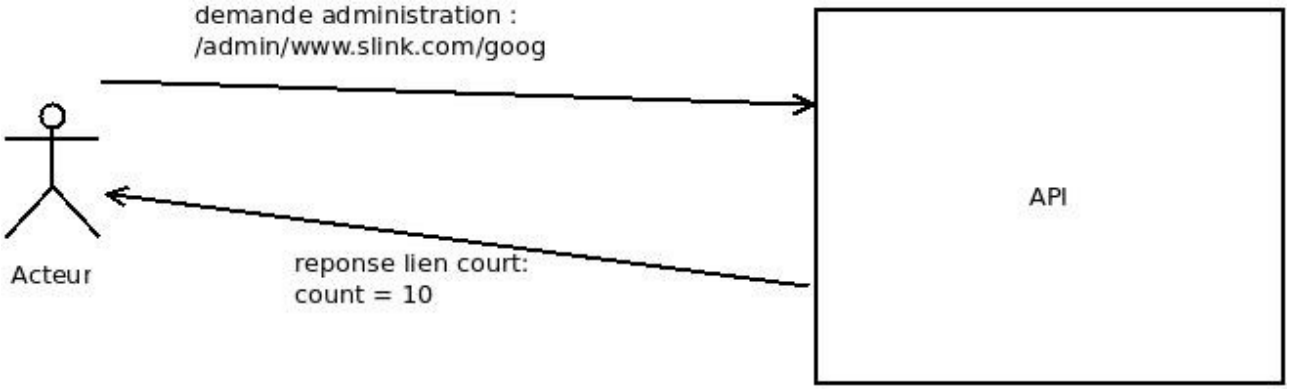


figure 4: Administration des codes courts

2. Description de l'API

Le fonctionnement de l'application est assez simple. Tout d'abord elle rentre dans une phase d'initialisation pendant laquelle la configuration en yaml est chargée puis les webservices sont associés à des chemins d'accès :

- ***/shortlink/{link}*** est associé au webservice shortlinkHandler en charge de la génération des liens courts
- ***/{link}/{token}*** est associé au webservice redirectionHandler en charge de la redirection et des statistiques liées à chaque code court.
- ***/admin/{link}/{token}*** est associé au webservice adminHandler en charge de la gestion des codes courts. Ce webservice permet de connaître les différentes informations liées à chaque code court : le lien d'origine, le nombre de fois qu'une redirection a été effectuée depuis ce code court ainsi que la date de création.

Ensuite, l'application attend que des événements de produisent, l'utilisateur tape l'une des trois URLs présentées si dessus, pour déclencher le service correspondant.

Ce mécanisme est décrit dans la figure si dessous.

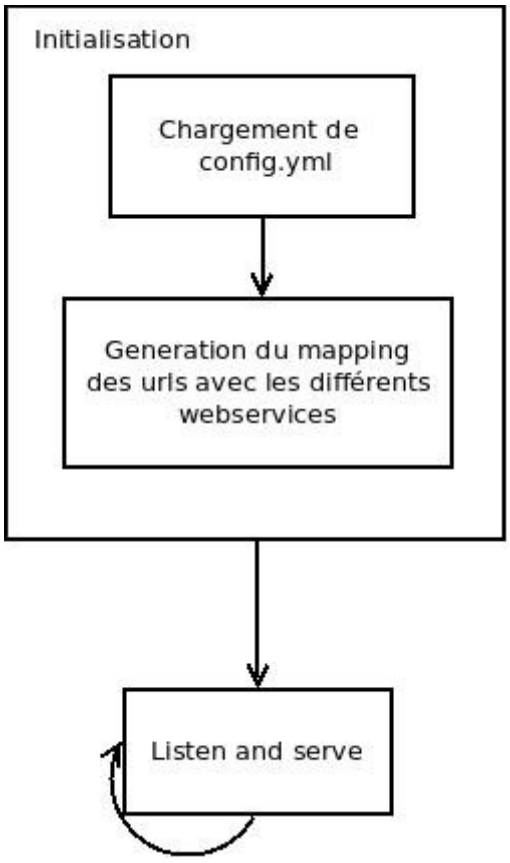


figure 5: Fonctionnement de l'application

La figure suivante décrit d'organigramme des différents packages utiliser dans cette application.

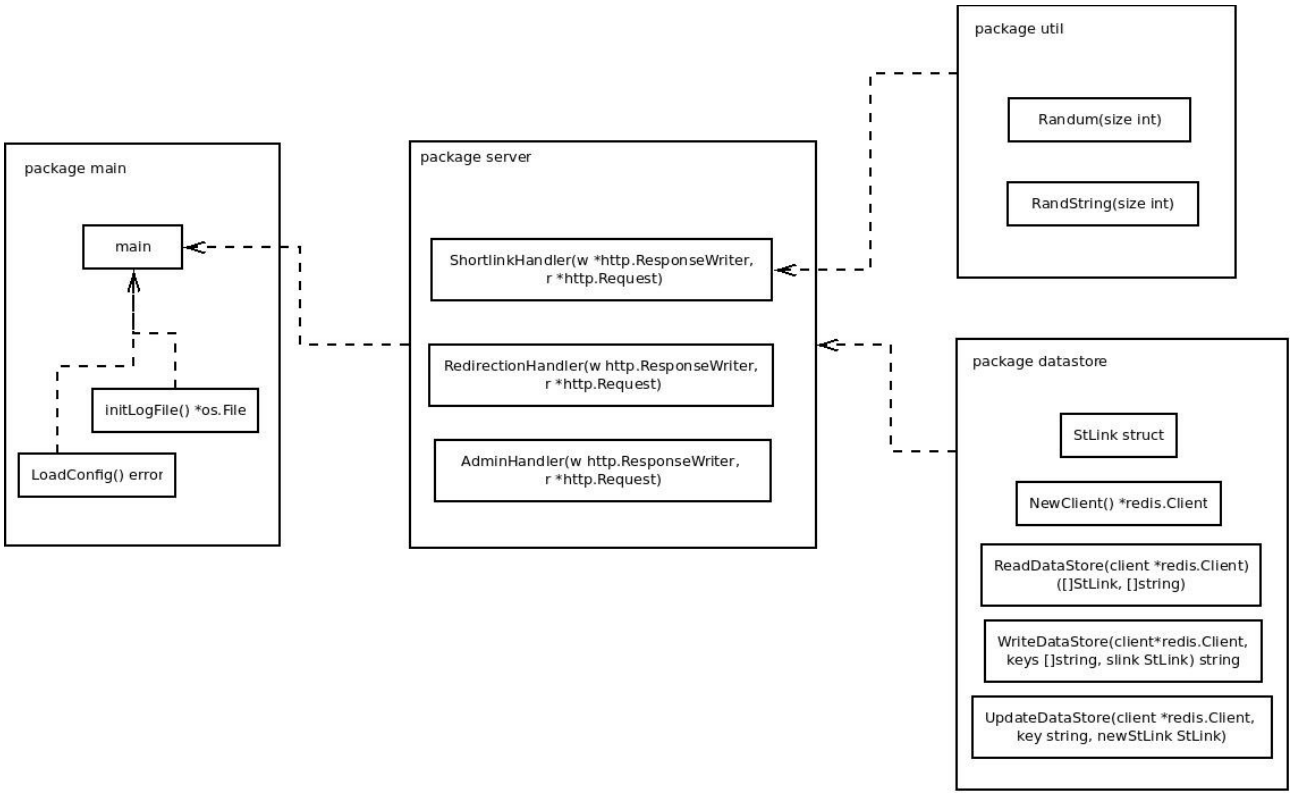


figure 6: Diagramme décrivant l'architecture de l'application

Le package principale, *main*, comprend uniquement l'initialisation ainsi que la gestion des différentes services.

Les trois services sont placés dans le package *server*. Chaque service a besoin d'accéder à la base de donnée en lecture et/ou en écriture. Toutes les fonctions liées à la gestion de cette base de donnée ce trouvent dans le package *datastore*.

Il existe un quatrième package, le package *util*, qui contient tout ce qui n'est pas lié directement au fonctionnement de chaque service ou de la base de donnée mais qui est nécessaire, ou, peut être utilisé par tout le monde. Dans ce cas, on y trouve les fonctions de génération aléatoire de chaîne de caractère contenant uniquement des chiffres (*RandNum*) ou contenant chiffres et lettres (*RandString*).

3. Les services

3.1 ShortLinkHandler

La figure suivante décrit le fonctionnement de la génération d'un code court unique associé à une URL choisi par l'utilisateur. Ce code court peut être personnalisé par le passage d'un paramètre.

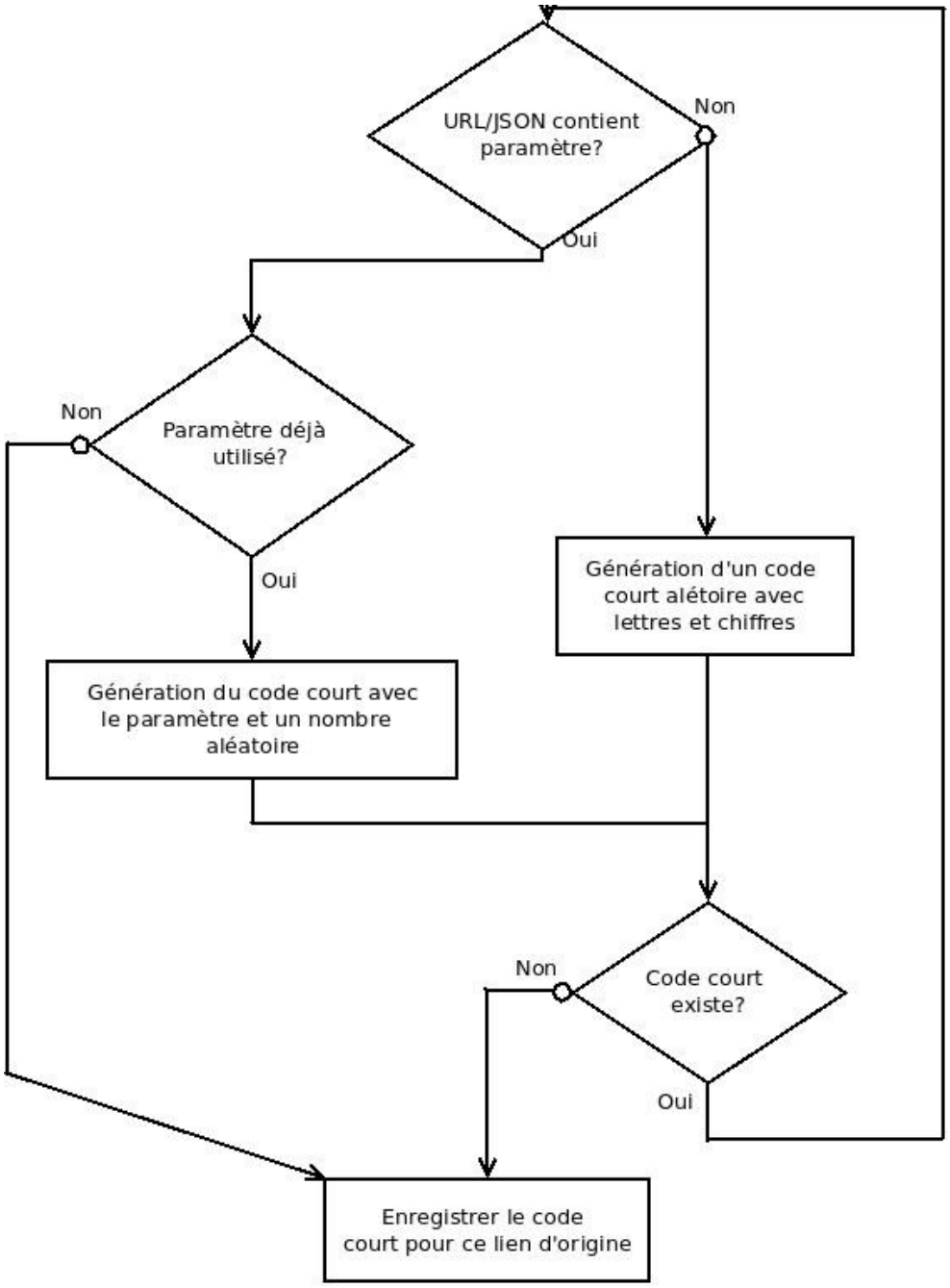


figure 7: Génération d'un code court

Tout d'abord, on teste la présence du paramètre custom dans l'URL, ou le json, fournit par l'utilisateur.

Si ce paramètre est présent, alors, on vérifie qu'il est pas déjà utilisé comme code court pour un autre lien. Dans le cas où le paramètre n'est pas déjà utilisé pour référencer une autre URL, alors il est utilisé directement comme code court. Sinon, on génère un nombre aléatoire qu'on rajoute à ce paramètre dans me but d'avoir un code court unique. Une fois encore, l'unicité de ce code court est testée. Cette opération est effectuée jusqu'à l'obtention d'un code court non utilisé.

Si l'utilisateur n'a pas indiqué de paramètre, l'application génère un code court aléatoire contenant des chiffres et des lettres d'une longueur définie dans le fichier config.yml. Ce code est également vérifié afin de ne pas avoir deux codes courts qui pointerait sur le même lien.

Une fois le code court choisi par l'application, celui-ci est enregistré dans une base de donnée de type NoSql, avec le lien d'origine ainsi que la date de création et de nombre de redirection vers le lien d'origine initialisé à zéro.

3.2 RedirectionHandler

La figure suivante décrit le fonctionnement de la redirection vers l'adresse originale lorsque l'utilisateur fournit au navigateur le code court.

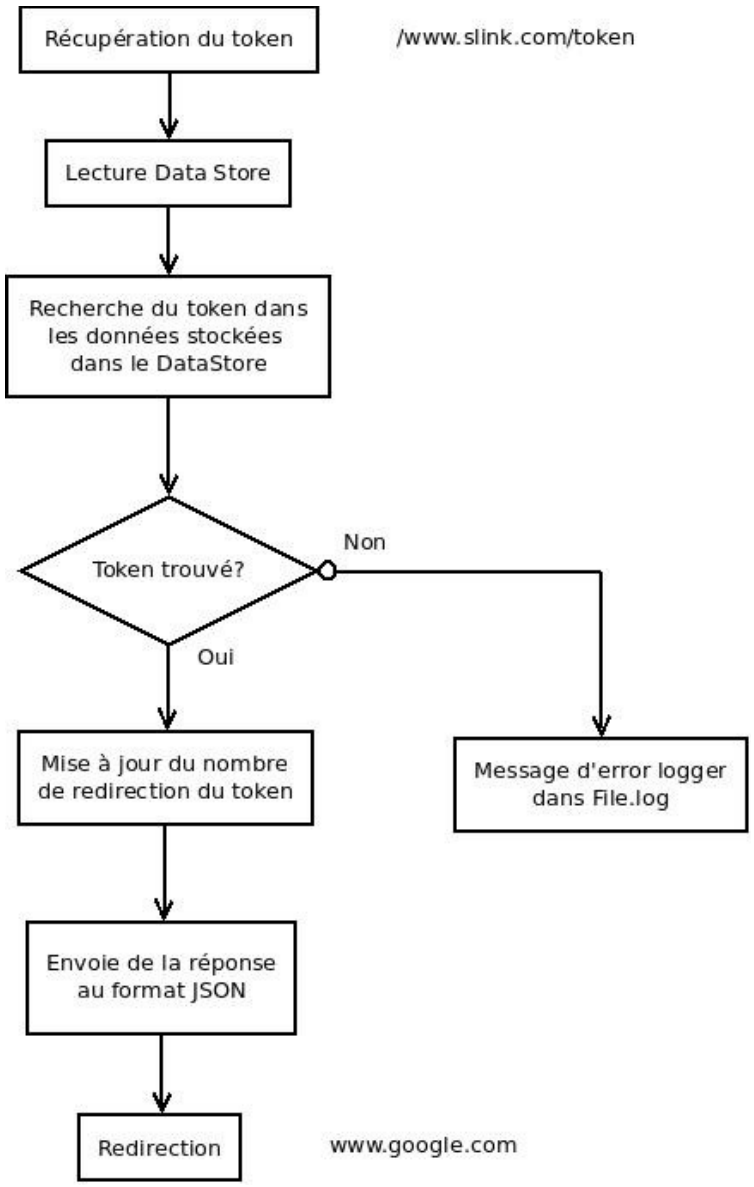


figure 8: Fonctionnement du service de redirection

Tout d'abord, on récupère le token présent dans l'URL, ainsi que toutes les données contenues dans la base de donnée. On cherche le token, donnée par l'utilisateur dans l'URL, dans la liste des données. Une fois ce token trouvé, on incrémente le compteur de redirection. Puis le service fournit les données mises à jour au format json dans le corps de la réponse, avant de rediriger l'utilisateur vers le lien d'origine qu'il souhaite rejoindre.

Si le token fournir par l'utilisateur n'est pas présent dans la base de donnée, l'application génère un log d'erreur dans le fichier File.log dédier à cet effet.

3.3 AdminHandler

Le fonctionnement de ce service est assez proche de celui gère la redirection. La figure suivante en décrit le mécanisme.

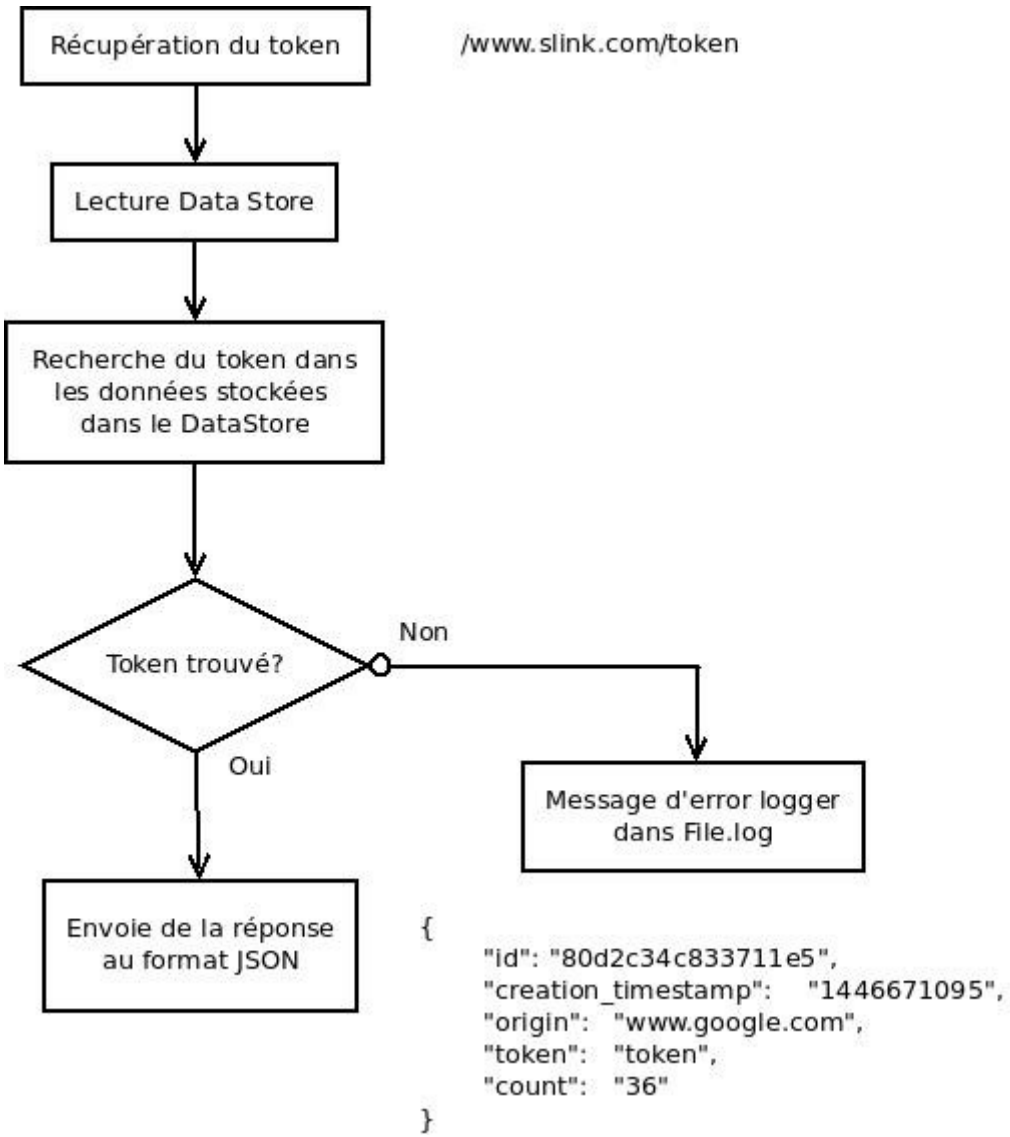


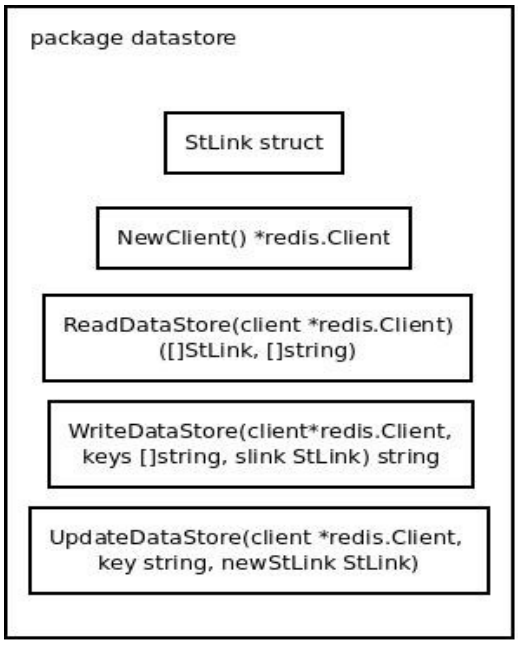
figure 9: Fonctionnement du service de monitoring

Tout d'abord, on récupère le token présent dans l'URL, ainsi que toutes les données contenues dans la base de donnée. On cherche le token, donnée par l'utilisateur dans l'URL, dans la liste des données. Une fois ce token trouvé, le service fournit les données mises à jour au format json dans le corps de la réponse.

Si le token fournir par l'utilisateur n'est pas présent dans la base de donnée, l'application génère un log d'erreur dans le fichier File.log dédier à cet effet.

4. Gestion de la base de donnée

La gestion des données, lien d'origine, token associé, compteur de redirection, est fait à l'aide d'une base de donnée de type ***redis*** (de l'anglais *REmote DIctionary Server*). Ce système gère les base de donnée à l'aide d'un système de clef valeur. Il fait partie de la mouvance NoSQL qui a pour but de fournir de meilleur performances. L'unité logique n'y est plus la table.



On définit une structure qui contient tous les éléments nécessaire à l'API :

- creation_timestamp: "1446671095",
- origin: "www.google.com",
- token: "paris123",
- count : "36"

4.1 Écriture de nouvelle donnée dans la base

On génère une clef unique de 20 caractères, contenant des chiffres et des lettres. Cette clef doit être unique, à chaque génération, on doit vérifier qu'elle n'est pas déjà présente dans la base. Si tel est le cas, une nouvelle clef est regénérée.

Cette fonction prend en argument un nouvel élément de type StLink, la structure contenant toutes les informations liées au nouveau code court à enregistrer dans la base. Cette structure est ensuite transformée au format json ce qui permet d'écrire les données au format clé valeur.

4.2 Lecture des données dans la base

Cette fonction permet de récupérer toutes les données écrites dans la base. Comme on l'a vu précédemment, ces données sont écrites au format json. De ce fait il suffit de reconstruire l'objet de base à partir de ces données.

Cette fonction fourni à l'appelant une liste de StLink et la liste des clefs.

4.3 Mise à jour des données dans la base

Cette fonction permet de mettre à jour les données lorsque le compteur de redirection est mis à jour. Cette fonction prend en entrée la clef de la valeur à mettre à jour ainsi que les nouvelles données au format StLink. Comme pour l'écriture de nouvelles données, celle-ci doivent être sérialisées au format json pour être sauvegardées dans la base.