Design Documentation

1 Functionalities

1. Create a unique short URL with a path length of 7 characters for any given URL.
2. Redirect to the original URL using a given registered short URL.
3. Retrieve the registered URL information by sending either a generated short URL or an original URL.
4. User registration and authentication for session management.
5. User account is required to create a new short URL, and each user can generate a limited number of short URLs. The limit can be increased by request.
6. Custom short URL paths that are longer than seven characters.
7. Private URLs that only the owning user can use through an unexpired JWT token.
8. Generated short URLs have an expiration time that can be extended upon request.
9. Users can fetch information about their URLs.
10. URLs can be activated and disactivated by the owning user.
11. (Todo) Add Spark/Hadoop for data analysis such as number of clicks

2 Non-functionalities

1. Scalability

   All microservices are horizontally scalable except for the oken service, which does not need to be scaled. Automatically generated short URL paths use 64 characters and a length of 7, which allows about 4.4 trillion unique short URLs. MySQL database can be sharded as number of URLs increase.

2. Low latency

   Redis reduces the latency of fetching recently used data from the MySQL database. OAuth 2.0 with JWT token is implemented for future use, which can authenticate users without querying a database, thereby lessening the latency in redirecting with a private URL.

3. Low cost

   Data for expired URLs or tokens are regularly cleaned up in all databases by setting recurring events or expiration times.

4. Security

   API Gateway is used to prevent external access to internal communication. A JSON Web Token is used for session management which has a default expiration time of 30 days.

5. Strong consistency

   User data is modified using the row lock functionality of MySQL with a timeout. Token Service also uses a lock on the database table to ensure that it does not generate the same token even if several instances are running. Once a URL is registered, it is not deleted unless expired.

6. Fault tolerance

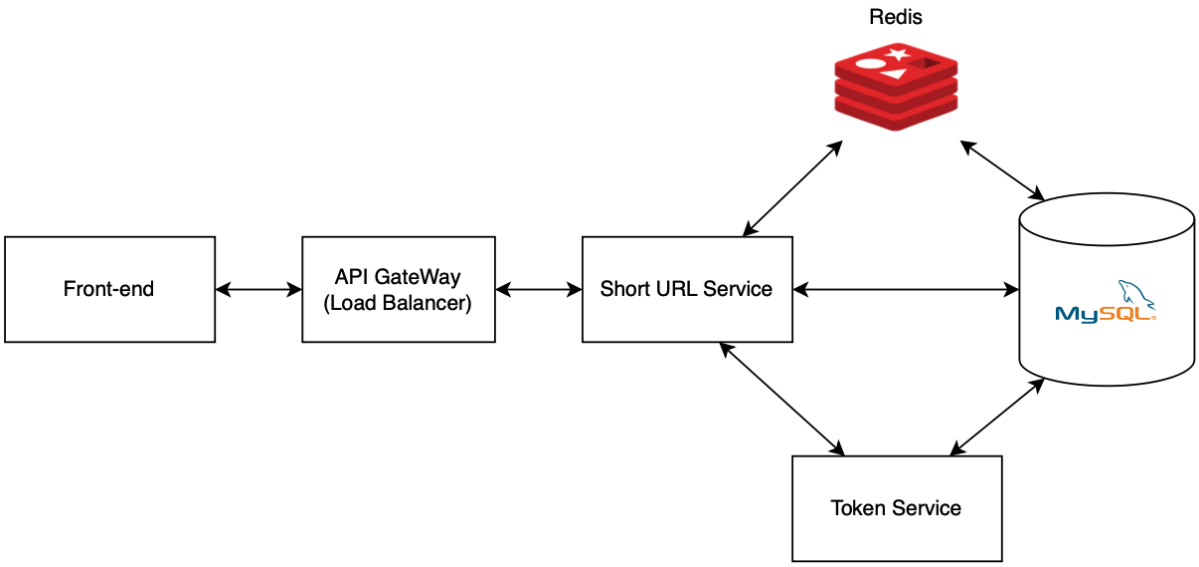
   By scaling the microservices in a Cloud using Kubernetes, the program will be fault-tolerant, highly available, and have low latency.

7. High availability

   Fault tolerance and scaling will allow high availability.

8. Durability MySQL has durability by maintaining a transaction log file and can further enhance durability by setting master-slave replication.

3 System Design



3.1 Token Service

The token service gives a token that allows about 4.4 million pre-defined short paths to Short URL servers. A seed is generated by adding a fixed big number to the previous seed and

dividing it by 1 million. A generated seed is sent to a User service by request. A URL service generates a number by adding 1 million to the previous number. This number is converted into base 64 and each place value is mapped to an alphanumeric character, a dash, or an underscore.

The token service uses a MySQL database to keep the records of generated seeds. While running more than one instance will not generate a duplicate URL, only one instance of the token service should be running to reduce unnecessary database queries.

3.2 User service and Short URL service

User service and Short URL service can horizontally scale up to handle more requests. API calls to these services are shown in section 4.

3.3 MySQL Database

3.3.1 Table ‘app_user‘

column name	data type	description
id	integer not null	Primary key
email	varchar(100) not null	A unique email address
password	varchar(255) not null	Password
firstname	varchar(50)	First name
lastname	varchar(50)	Last name
available_short_url	integer	Remaining number of URLs the user can generate
app_user_created_at	datetime(6)	The time that the user was registered
app_user_role	varchar(255)	Role: ADMIN, USER

3.3.2 Table ‘url‘

column name	data type	description
url_id	integer not null	Primary key
short_url_path	varchar(30) not null	Short URL path (Primary Key)
long_url	varchar(2047) not null	Original URL
url_created_at	datetime(6) not null	Created date and time
url_expire_date	date not null	Expire date
is_private	bit	True if private (Not updatable)
is_active	bit	True if active
user_id	integer not null	Foreign key for the id of ‘app_user‘

3.3.3 Table ‘token‘

column name	data type	description
seed	integer not null	Seed of a token
created_at	datetime(6) not null	Created date and time
expire_date	date not null	Expire date

4 User API

4.1 User services

- **Register a new user**

```
POST /api/v1/auth/register
```

Payloads: firstname, lastname, email, password
Return: a JWT token

- **Login**

```
POST /api/v1/auth/authenticate
```

Payloads: email, password
Return: a JWT token

- **Get profile**

```
GET /api/v1/user/profile
```

No payloads
Return: firstname, lastname, email, availableShortUrl, createdAt, numUrl

- **Increase the number of short URLs for current user**

```
POST /api/v1/user/refill
```

Payloads: number
Return: availableBefore, availableAfter

4.2 URL management

- **Search for URLs registered by a user**

```
POST /api/v1/user/urls
```

Payloads: isActive
Return: list of (index, shortUrl, longUrl, description, isPrivate, isActive, expiresAt)

- **Generate a new short URL**

```
POST /api/v1/user/urls/generate
```

Payloads: shortUrlPath, longUrl, isPrivate, description
Return: isGenerated, availableShortUrl, shortUrl, isPrivate, isActive, expireDate

- **Enable or disable a shortUrl**

```
PUT /api/v1/user/urls/set-is-active
```

Payloads: shortUrl, isPrivate, isActive
No returns

- **Modify short URL path**

```
POST /api/v1/user/urls/modify
```

Payloads: longUrl, isPrivate, newShortUrlPath, newDescription
Return: prevShortUrl, newShortUrl, expireDate

- **Extend URL Expiration time**

```
POST /api/v1/user/urls/extend
```

Payloads: number, longUrl, isPrivate
Returns: isExtended, prevExpireDate, newExpireDate, remainingNumber, shortUrl

- **Delete URL**

```
POST /api/v1/user/urls/delete
```

Payloads: shortUrlToDelete, isActiveForGetUrls
Return: list of (index, shortUrl, longUrl, description, isPrivate, isActive, expiresAt)

4.3 URL Services

URLs can be registered both 1) publicly or 2) privately by users. If URL is registered privately, it can be only accessed by the corresponding user.

- **Get a short URL of a given long URL**

```
POST /api/v1/user/urls/short
```

PayLoads: longUrl
Return: shortUrl

- **Get a long URL of a given short URL**

```
POST /api/v1/user/urls/long
```

PayLoads: shortUrl
Return: longUrl

- **Redirect to the long URL website**

```
GET /{short path}
```

Payloads: longUrl
Return: The redirected page will be opened.

(Redirection may fail if the long URL doesn't have prefixes such as "http://")

5 Local Run

- Run docker instances

  ```
  $ docker compose up -d
  ```

  MySQL Database (port 3306), Redis Database (port 6379), Zipkin (port 9411)

- Run back-end services

  ```
  $ mvn clean install
  $ java -jar eureka-server/target/eureka-server-1.0-SNAPSHOT.jar
  $ java -jar apigw/target/apigw-1.0-SNAPSHOT.jar
  $ java -jar token/target/token-1.0-SNAPSHOT.jar
  $ java -jar appuser/target/appuser-1.0-SNAPSHOT.jar
  ```

  Eureka server (port 8761), API Gateway (port 80), User service (port 8081), Token (port 8083)

- Run front-end in dev mod

  ```
  $ sudo npm install --prefix frontend
  $ sudo npm start --prefix frontend
  ```

  Front-end (port 3000)

  The default hostname for URL generation is http://127.0.0.1, which is the local host. The hostname can be configured by modifying 'hostname' in appuser/src/resources/application.yml.