URL COUNTER Java Spring Developer Assignment

Software Architecture Document

Version 1.3

07/30/2021

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Description of Versions / Changes** | **Responsible Party** | **Date** |
| 1.0 | Initial version | Sriteja | 07/30/21 |
| 1.1 | Created the Project with Database | Sriteja | 07/30/21 |
| 1.2 | Updated the code with the Data Persistance using JpaRepository | Sriteja | 07/30/21 |
| 1.3 | Added the Schema PNG and  Architecture & Design Doc | Sriteja | 07/30/21 |

Table of Contents

[1. Introduction 1](#_Toc353470033)

[1.1. Purpose 1](#_Toc353470034)

[1.2. Scope 1](#_Toc353470035)

[1.3. Overview 1](#_Toc353470038)

[2. Architectural Representation 3](#_Toc353470039)

[3. Architectural Goals and Constraints 4](#_Toc353470040)

[4. Use-Case View 4](#_Toc353470041)

[4.1. Actors 4](#_Toc353470042)

[4.2. Use-Case Realizations 5](#_Toc353470043)

[4.2.1. Store URL 5](#_Toc353470044)

[4.2.2. Get UniqueKey 7](#_Toc353470045)

[4.2.3. Access Count of URL 9](#_Toc353470046)

[4.2.4. Return List of URLs with Pagination 10](#_Toc353470047)

[4.2.5. Last Access Date of URL 11](#_Toc353470048)

[5. Logical View 12](#_Toc353470049)

[5.1. Overview 12](#_Toc353470050)

[5.2. Interface Definitions 14](#_Toc353470051)

[6. Data View 23](#_Toc353470052)

[7. Deployment View 25](#_Toc353470053)

Software Architecture Document

# Introduction

This document provides a high level overview and explains the architecture of the URL Counter Application.

The document defines goals of the architecture, the use cases supported by the system, architectural styles and components that have been selected. The document provides a rationale for the architecture and design decisions made from the conceptual idea to its implementation.

## Purpose

The Software Architecture Document (SAD) provides a comprehensive architectural overview of the Application which was developed to Create and Read the data of URLs which are stored in the Database. The primary goals this design will accomplish are to Create, Read and analyse the data of URLs stored in the Database.

## Scope

The scope of this SAD is to explain the architecture of the URL Counter Application.

This document describes the various aspects of the Application design that are considered to be architecturally significant. These elements and behaviors are fundamental for guiding the construction of the URL Counter Application system and for understanding this project as a whole.

## Overview

In order to fully document all the aspects of the architecture, the Software Architecture Document contains the following subsections.

Section 2: describes the use of each view

Section 3: describes the architectural goals and constraints of the system

Section 4: describes the most important use-case realizations

Section 5: describes logical view of the system including interface and operation definitions.

Section 6: describes significant persistence elements.

Section 7: describes how the system will be deployed.

|  |  |
| --- | --- |
| **ID** | **Requirement** |
| U01 | Store the URL given in the API call in a Database/In-memory Data-Structures |
| U02 | Return the Unique short key for the requested URL. |
| U03 | Return the count of number of times the URL is accessed. |
| U04 | Return the List of data of URLs for the provided Page Number and Size. |
| U05 | Return the Last Accessed date for the requested URL. |

# Architectural Representation

This document details the architecture using the views defined. The views used to document the URL Counter Application system are:

**Use Case view**

**Audience**: Anyone who has access to GitRepository.

**Area**: describes the set of scenarios and/or use cases that represent some significant, central functionality of the system. Describes the use cases for the system, this view presents the needs of the user and is elaborated further at the design level to describe discrete flows and constraints in more detail. This domain vocabulary is independent of any processing model or representational syntax (i.e. XML).

**Related Artifacts** : Use-Case Model, Use-Case documents

**Logical view**

**Audience**: Designers.

**Area**: Functional Requirements: describes the design's object model. Also describes the most important use-case realizations and business requirements of the system.

**Related Artifacts**: Design model

**Data view**

**Audience**: Data specialists, Database administrators

**Area**: Persistence: describes the architecturally significant persistent elements in the data model as well as how data flows through the system.

**Related Artifacts**: Data model.

# Architectural Goals and Constraints

There are some key requirements and system constraints that have a significant bearing on the architecture. They are:

1. The system is meant as a proof of concept for a more complete project prediction system to be built in the future. Therefore one of the primary stakeholders in this document and the system as a whole are future architects and designers, not necessarily users as is normally the case. As a result, one goal of this document is to be useful to future architects and designers.
2. The system will be written using JAVA technologies but will use an open source RDBMS system (Oracle SQL) for data persistence.
3. One of the primary goals of the system architecture is to minimize the impact of these changes by minimizing the amount of code that would need to be modified to implement them. The architecture seeks to do this through the use of modularization and information hiding to isolate components that are likely to change from the rest of the system.

# Use-Case View

The purpose of the use-case view is to give additional context surrounding the usage of the system and the interactions between its components. For the purposes of this document, each component is considered a use-case actor. In section 4.1, the most common use-cases are outlined and illustrated using UML use-case diagrams and sequence diagrams to clarify the interactions between components.

## Use-Case Realizations

### Store URL

User sends a URL and it will be stored in the Database if it was not present in the DB.

### Get Unique Key

User requests Unique Short Key of the provide URL and it will be provided if URL present in DB.

### Access Count of URL

User requests the Access Count of the provided URL and it will be provided if URL present in DB.

### Return List of URLs with Pagination

User requests a List of URLs with the Pagination applied and a List of URLs will be sent back.

# Logical View

## Overview

The main goal of the logical view is to define the components that will make up the system and to define the interfaces through which they will communicate and interact with one another. The primary decision-making factor behind defining the system components is the need to isolate the components that are likely to change from the rest of the system. By clearly defining the interfaces of these components and hiding their internal implementations from the rest of the system, the impact of expected changes can be minimized. A summary of these changes and how the logical decomposition of the architecture addresses them is as follows:

1. Changes to the Store URL API
   1. The architecture addresses this by implementing the calls to the Store URL API in a Controller component. The rest of the application will communicate with Store URL only through the interface exposed by this component. Therefore any changes to the system to deal with changes in the Store URL API need only be made in the internal implementation of this component.
2. Changes to the Get API
   1. Similar to the above, this is addressed by implementing calls to the Get API in a Controller component. Changes required to deal with changes to the Get API need only be made in the internal implementation of this component and not to the rest of the system.
3. Changes to the Count API
   1. Similar to the above, this is addressed by implementing calls to the Count API in a Controller component. Changes required to deal with changes to the Count API need only be made in the internal implementation of this component and not to the rest of the system.
4. Changes to the List API
   1. Similar to the above, this is addressed by implementing calls to the List API in a Controller component. Changes required to deal with changes to the List API need only be made in the internal implementation of this component and not to the rest of the system.
5. Changes to the Access Date API
   1. Similar to the above, this is addressed by implementing calls to the Access Date API in a Controller component. Changes required to deal with changes to the Access Date API need only be made in the internal implementation of this component and not to the rest of the system.

**Table 5.1** Element Responsibilities

|  |  |
| --- | --- |
| **Element** | **Responsibilities** |
| Assembla Data Store | * Persist data to MySQL data store. * Provide query interface to the MySQL data store. |
| URL Controller | * Handle all communication with all the APIs. * Provide a native JAVA interface for other components to use to access all the APIa. |
| URL Service | * Provide an interface to get a bussiness Logic implemented for a given URL Counter project. |
| URL Repository Store | * Persist application specific data such as URL Objects to the Oracle SQL data store. * Provide a query interface to the application specific SQL data store |

## Interface Definitions

**URL Service:: URL Service Interface**

**public** **interface** UrlService {

**public** Urls storeUrl(String url) **throws** UrlException;

**public** Integer get(String url) **throws** UrlException;

**public** Integer count(String url) **throws** UrlException;

**public** List<Urls> list(**int** page, **int** size) **throws** UrlException;

**public** LocalDateTime accessDate(String url) **throws** UrlException;

}

Operation Definitions

**storeUrl(String url)**

*Description*: This operation Stores the provided URL to the DB if it was not already existing in the Db.

*Precondition:* None

*Postcondition:* All publicly accessible URLs are saved to the database.

**get(String url)**

*Description*: This operation gets URL short Key data for URL saved in the database and updates the last access date and usage Count to the database.

*Precondition:* A URL is available in the database.

**count (string url)**

*Description*: This operation gets the usage count of provided url.

*Precondition:* A URL is available in the database.

**list (int page, int size)**

*Description*: This operation gets a list of URLs present in the DB in the Pagination order.

*Precondition:* A List of URLs are available in the database

**accessdates(string url)**

*Description*: This operation gets the Last accessed date of the provided URL from the DB.

*Precondition:* A List of URLs are available in the database

# Data View

**Figure 8.1** Static Data Structure Diagram

This diagram illustrates the static data structure and relationships of the main entities that will be stored by the application in its database. Each element nominally represents a database table.

**Figure 8.2** Data Flow Diagram

This diagram illustrates how data will flow between external entities and the application. Ovals represent external entities, tall boxes represent points where data is processed, arrows show the direction of data flow, and short boxes represent persistent data stores.

