System Design Information for URL Shortener Service

Project Name: tiny-url

Version: 0.0.1

Date: 1st April 2024

Introduction

This document describes the architecture and system design of a URL Shortener Service. The service generates short and unique aliases for input URLs, allowing users to share and access them easily.

System Overview

The URL Shortener Service is built using Node.js and TypeScript, Nest.js for the application framework. It utilizes Prisma with PostgreSQL for data persistence and GraphQL along with REST API for client-server communication.

Features

- Generate short links converts long URLs into short, unique aliases for easier sharing and memorability, using a secure hash function.
- Retrieve all shortened URLs and fetches a list of all shortened URLs, including their original links, visit counts, and creation dates.
- Seamless URL redirection automatically redirects users from a shortened URL to its original long URL, ensuring a smooth user experience.
- Detailed URL information provides comprehensive details for a specific shortened URL, including visit statistics and redirect type.
- Modified details of the short URLs allow updating the destination of existing short URLs, enabling corrections or changes to the original URL.
- Remove Short URLs Enables the removal of shortened URLs from the database, cleaning up unused or outdated links.

Technology Stack

- Node.js and TypeScript: For programming and strong typing, enhancing code quality and maintainability.
- Nest.js: A progressive Node.js framework for building efficient and scalable server-side applications. It structures the app in modules, making it organized and modular.
- Prisma: Acts as the ORM, simplifying database operations with PostgreSQL, ensuring smooth data access and manipulation.

- PostgreSQL: Selected for its robustness, reliability, and support for complex queries and transactions.
- GraphQL: Provides a flexible and efficient approach to data queries and manipulation over a single endpoint, improving performance for clients.
- REST API: Used alongside GraphQL for operations where RESTful services are more suitable, offering flexibility in client-server interaction.

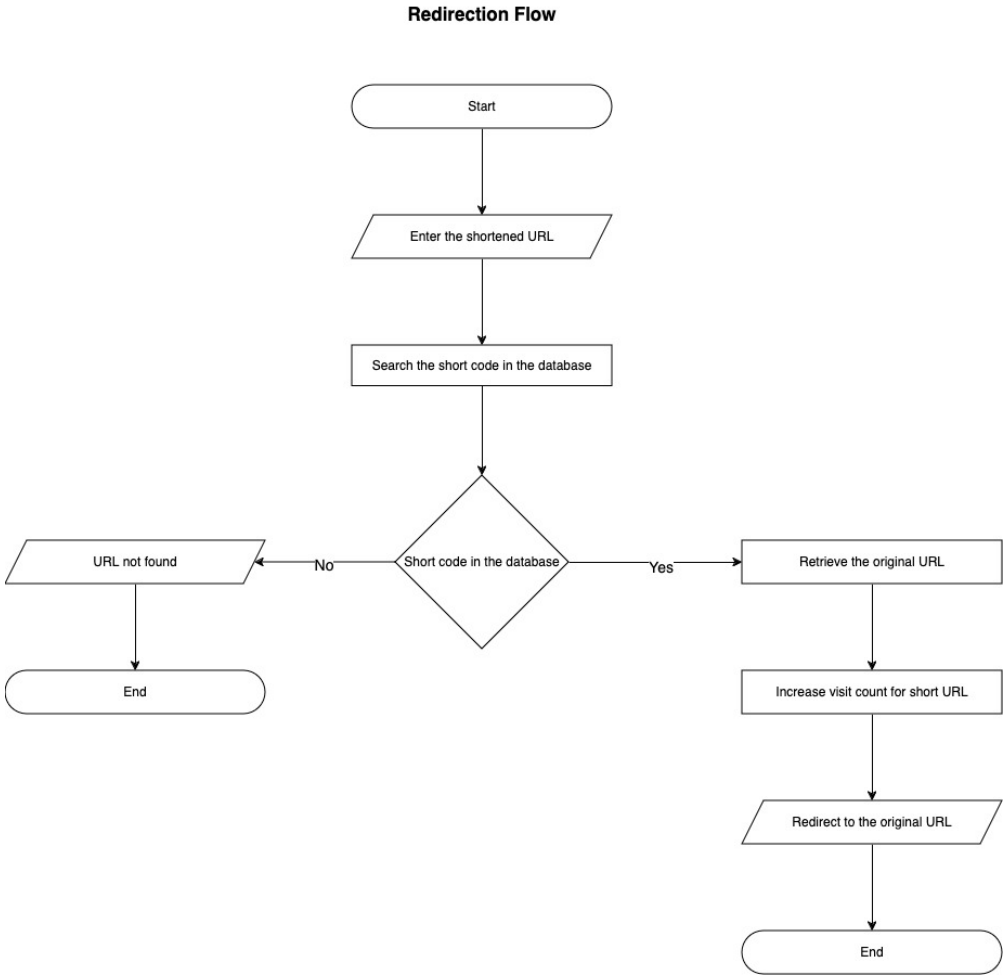
Database Design

Schema Diagram: The database schema primarily consists of a urlMapping table, which includes fields for id, longUrl, shortCode, visitCount, lastVisited, redirectType, createdAt, and updatedAt.

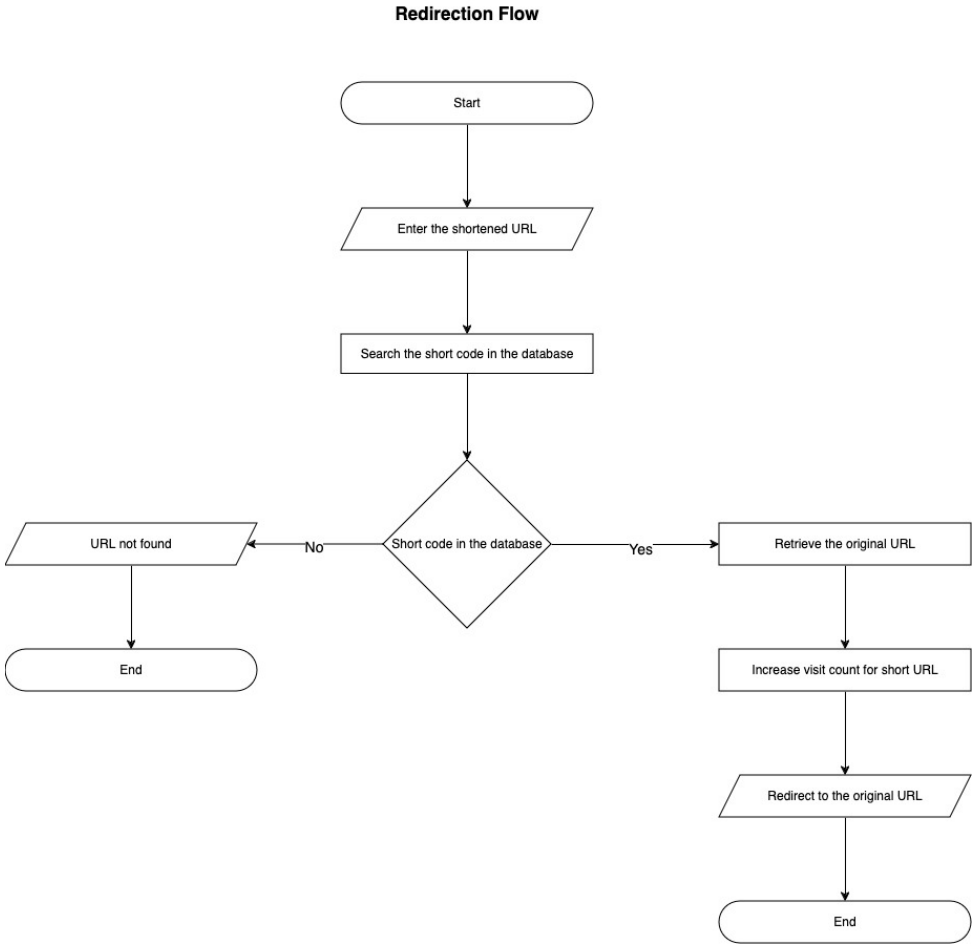
Table Descriptions: The table stores mappings between original URLs and their shortened counterparts, along with usage metrics and timestamps for creation and last update.

Process Flow

URL Shortening Flow: Users submit a URL via the GraphQL or REST API endpoint. The system generates a unique short code, stores the mapping in the database, and returns the shortened URL.



Redirection Flow: When a client requests a shortened URL, the system looks up the original URL from the database and redirects the client, incrementing the visit count.



API Design

GraphQL Schema: Includes operations for creating short URLs, fetching URL details, and retrieving all URLs.

REST API Endpoints: Provide endpoints for creating short URLs, redirecting to original URLs based on short codes, and administrative tasks such as fetching all URLs or specific details.

Note: If you are a developer, please flow code base README.md file to set up the application and to execute the Endpoints (API Documentations) to check the application.