Technical Challenge: URL Shortener

Author: Victor Martinez <vcrmartinez@gmail.com>
Created at: Aug 22, 2023
Github Tracking: https://github.com/victormartinez/urlshortener

Summary

This document briefly outlines the architectural decisions that build up the technical challenge of developing a URL Shortener service that supports 1M RPM peaks of traffic.

Architecture

The designed architecture for the solution is illustrated by Figure 1, which evidences two high-level layers: API and Database.

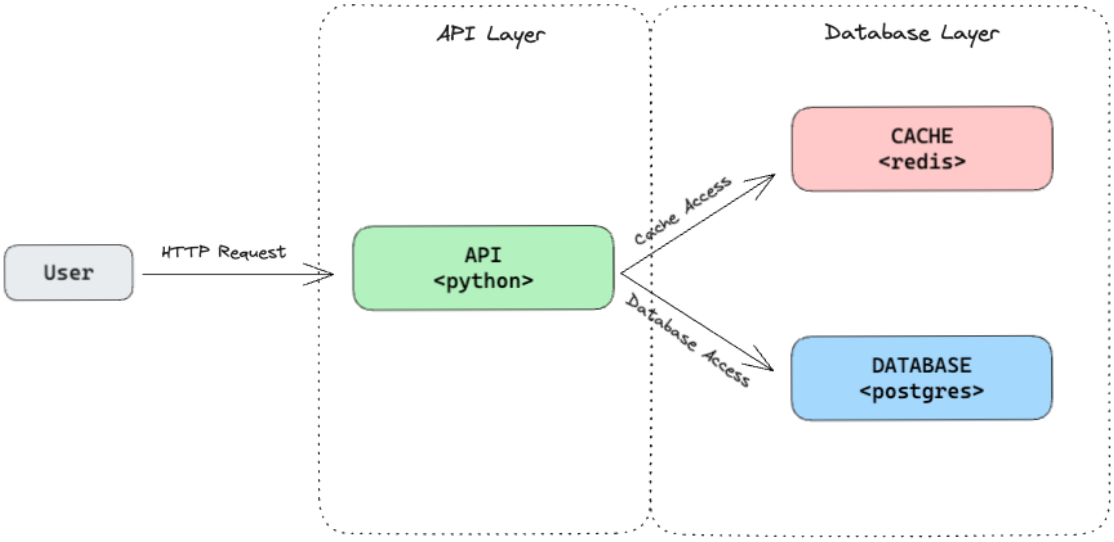

Figure 1: Overview of architectural components.

The user makes a request to API layer built in Python, which can access Cache and/or Database depending on the logic pointed as follows:

- Creating a shortened URL populates both cache and database;
- Updating a shortened URL flushes the record from cache;
- Reading requests makes the API layer follow the steps below:
    - At first, it accesses the cache layer in order to find the destination URL;
    - If cache misses, then API accesses the database.
    - Database connection is only established if cache misses; since this application is read-intensive and cache hit is expected, the application tries to bypass database connection overhead.