A Project Report

On

API Finder

Submitted in partial fulfilment of the requirement for the award of project credits

Submitted by

Varun Aggarwal [40225335]

Deepanshu Sehgal [40225333]

Anmol Arora [40172251]

Gina Cody School of Engineering and Computer Science

Masters of Applied Computer Science

SOEN - 6441

INTRODUCTION

Objective

The main objective of this project is to develop and design a system that can read API/Json data and produce results to some application. The system should be able to store the API data into the database and must be able to perform basic parameterized queries.

Problem Statement

The project should be created without the use of any frameworks and the aim should be at creating a small and manageable database. It must follow the coding standards with respect to the chosen programming language, applicable design patterns must be used, refactoring strategies and use of some testing tool must be applied and the system developed should be able to perform basic parameterized queries in the end.

Proposed System

The proposed system is to connect to an API available at https://api.publicapis.org/entries. In this project, we aimed to make an API finder using Java Programming language by inserting the data from a public API, (which consists of a list of free public APIs in raw JSON format) into our local database and then performing various operations on it like retrieving the data, adding new data, updating the existing data, deleting the data that is not required and searching the data through keyword. We observed that a lot of APIs are locked behind a paywall, which can make API testing a bit difficult to do - luckily, free APIs do exist. With the help of a free API, you can do testing and create flexible, powerful apps in record time. With a quick Google search, you'll discover tens of thousands of free and open APIs - all well-made and easily accessible to

developers all over the world. But with so many APIs to choose from, looking for one that fits your needs can be incredibly time-consuming. So, we combed the web with a simple goal in mind - to make a comprehensive list of the best free APIs.

REQUIREMENTS

System Requirements

- Local Workstations
- Communication Links
- Relational Database
- Application developed in a Browser or GUI

Functional Requirements

- The system will display all the APIs and information about them.
- The user shall be able to add a new API.
- The user shall be able to update any existing API.
- The user shall be able to delete any API that is not required.
- The user can search any API based on keywords.
- The user can be able to reset API data to load any new data into the local database from the live API data.

Software Requirements

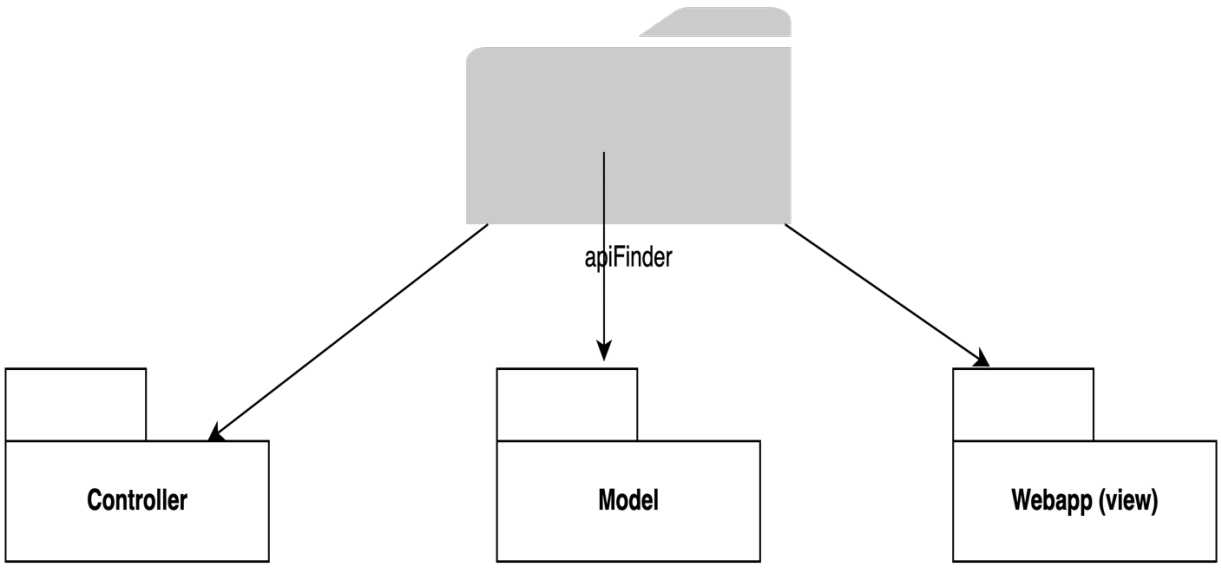
- Coding Language: Java
- JDK
- Working with API
- MySQL Workbench
- Eclipse IDE
- Testing Tool: JUnit
- Compatible web browser (IE/Chrome/Firefox)
- UML
- Github Repository

Design Pattern

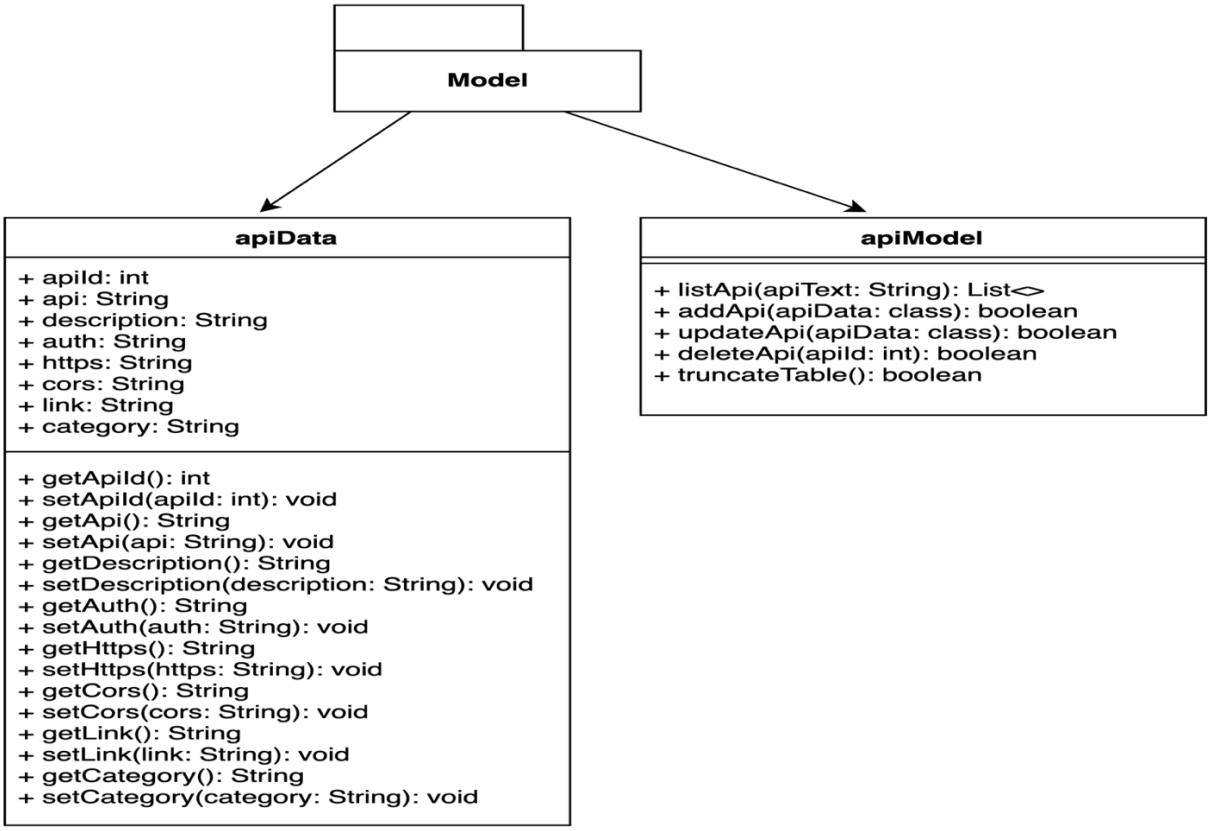
We have used MVC framework in this Java application development. MVC framework is used to separate the data access layer, business logic code and the graphical user interface that must be defined and designed to let the user interact with the application. This application has three parts:

1. **Model** - This part of the framework is to store the data of the application, such as databases, text data, files and/or other web resources.
2. **View** - This is the graphical user interface of the application. That would contain different buttons, text boxes and other controls to let the user interact with the application to complete his projects depending on the sort of the software he is using.
3. **Controller** - The actual back-end code constitutes the controller of the framework. A controller controls the data coming from the users or going to the user from a model.

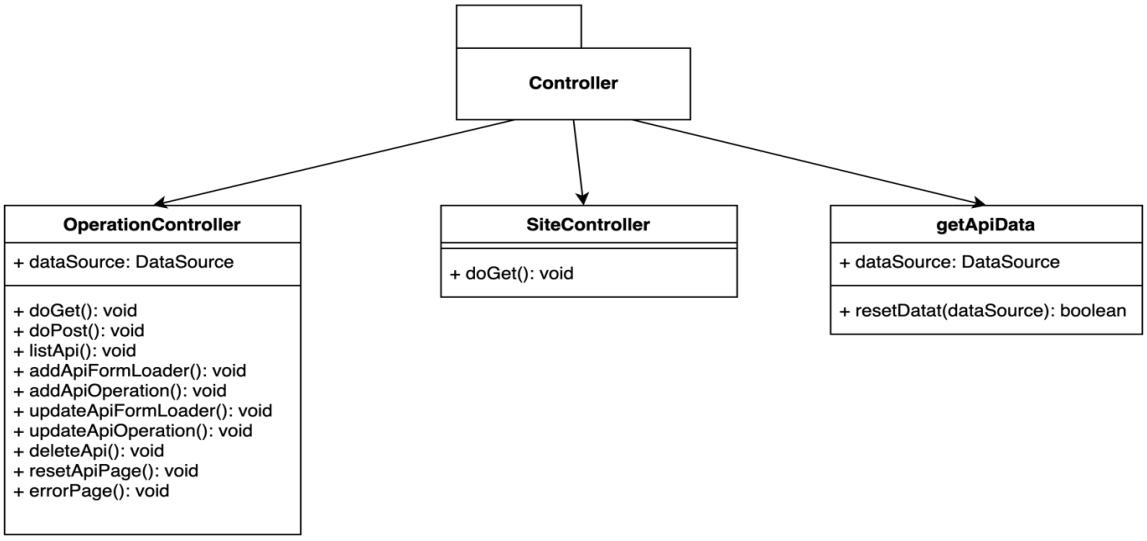
PACKAGE STRUCTURE



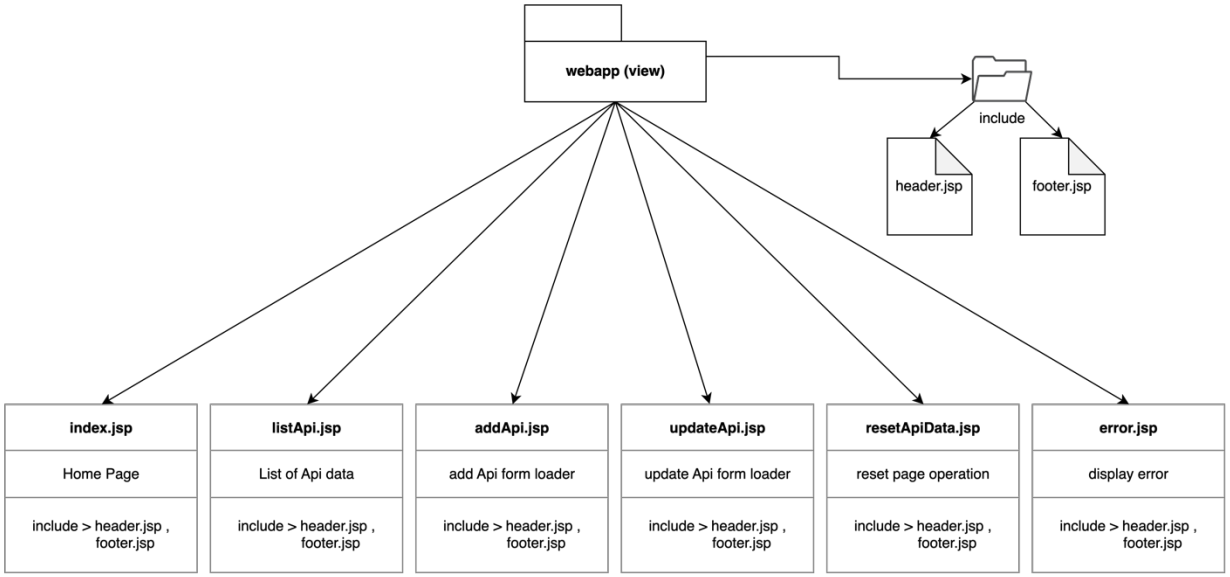
MODEL OVERVIEW



CONTROLLER OVERVIEW



VIEW OVERVIEW

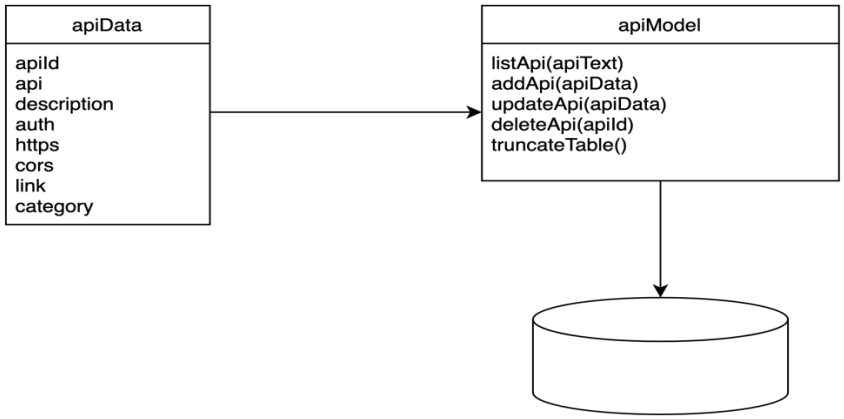


index.jsp	listApi.jsp	addApi.jsp	updateApi.jsp	resetApiData.jsp	error.jsp
Home Page	List of Api data	add Api form loader	update Api form loader	reset page operation	display error
include > header.jsp , footer.jsp	include > header.jsp , footer.jsp	include > header.jsp , footer.jsp	include > header.jsp , footer.jsp	include > header.jsp , footer.jsp	include > header.jsp , footer.jsp

Data Source Architecture

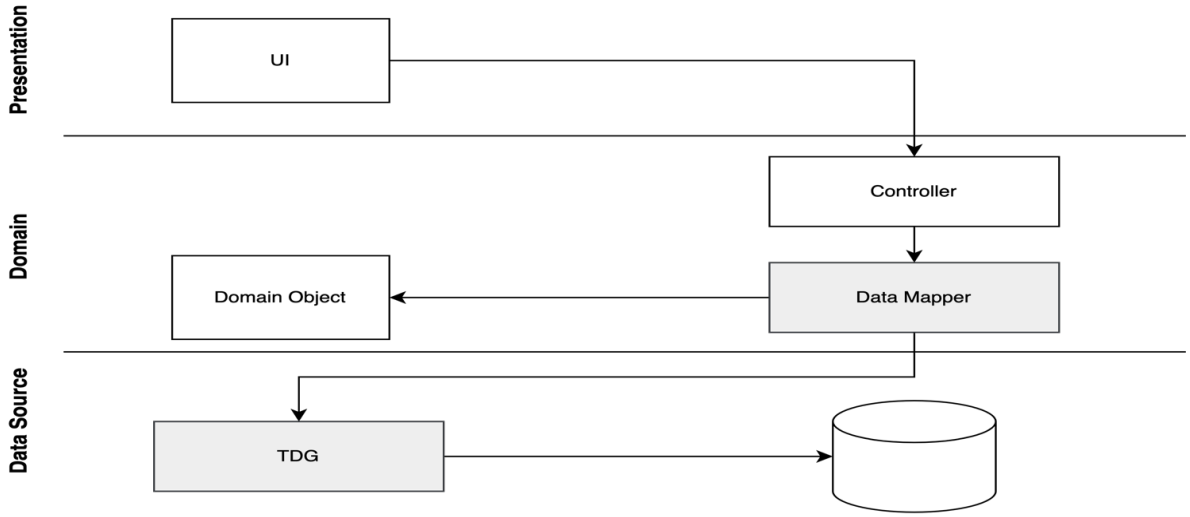
Table Data Gateway

A Table Data Gateway holds all the SQL for accessing a single table or view selects, inserts, updates, and deletes. Other code calls its methods for all interaction with the database.



apiId	api	description	auth	https	cors	link	category
1	AdoptAPet	Resources to get...	apiKey	true	yes	www.adoptapet.com	Animals
2	Axolotl	collection od Axolotl...	null	true	no	https://theaxolotl.net	Animals

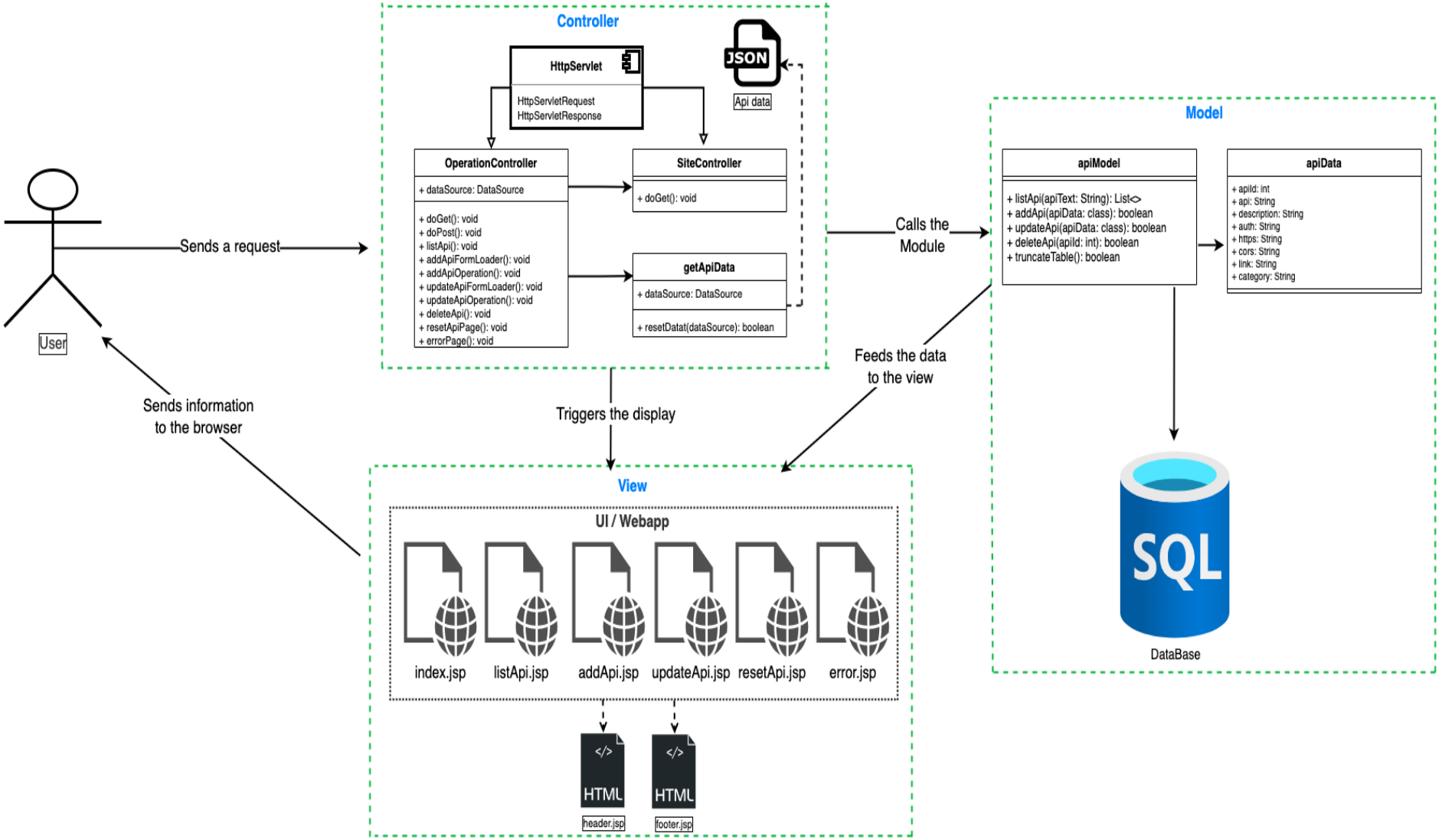
3-Layered architecture combining Data mapper and TDG



Overview

A Table Data Gateway (TDG) also known as Data Access Object is a Gateway to a database table which acts as an intermediary between domain objects and the database. An TDG has an interface for accessing (all the rows in) a single table: select, insert, update and delete. Domain objects send messages to the TDG for all interaction with the database. A Data Mapper moves data between objects and a database while keeping them independent of (and isolated from) each other and the Mapper itself.
In the above architecture, the controller calls the method in the Data Mapper which sends message to TDG for interaction with the database. In this project, apiData is the Domain Object and apiModel is the mapper having SQL parametrized queries such as listApi, addApi, updateApi, deleteApi, etc., The apiModel sends the message to the Table Data Gateway for all interaction with the database.

Object - Relational Architecture



GitHub repository links

- https://github.com/VarunAggarwal1998/API_Finders.git
- Google drive link for demonstration video: https://drive.google.com/drive/folders/1qE3Uo8iX_SG9Sf05pvZfy7vO-PknEODk?usp=sharing