REST APIs for DB services using Django

A thesis submitted to the department of

Computer Science & Engineering

of

International Institute of Information Technology, Bhubaneswar

in partial fulfilment of the requirements

for the degree of

Bachelor of Technology

by

Bhushan Prakash Khanale

(B516025)

under the supervision of

Harshad Mulmuley & Prof. Tapan Kumar Sahoo

Department of Computer Science and Technology

International Institute of Information Technology Bhubaneswar

Bhubaneswar Odisha - 751003, India

International Institute of Information Technology Bhubaneswar

Bhubaneswar Odisha -751 003, India. www.iiit-bh.ac.in

May 07, 2019

Undertaking

I declare that the work presented in this thesis titled REST APIs for DB services using Django, submitted to the Department of Computer Science and Engineering, International Institute of Information Technology, Bhubaneswar, for the award of the Bachelors of Technology degree in the Computer Science and Engineering, is my original work. I have not plagiarised or submitted the same work for the award of any other degree. In case this undertaking is found incorrect, I accept that my degree may be unconditionally withdrawn.

Bhushan Prakash Khanale

B516025

International Institute of Information Technology Bhubaneswar

Bhubaneswar Odisha -751 003, India. www.iiit-bh.ac.in

May 07, 2019

Certificate

This is to certify that the work in the thesis entitled *REST APIs for DB services using Django* by *Bhushan Prakash Khanale* is a record of an original research work carried out by her under my supervision and guidance in partial fulfillment of the requirements for the award of the degree of *Bachelor of Technology* in *Computer Science & Engineering*. Neither this thesis nor any part of it has been submitted for any degree or academic award elsewhere.

Harshad Mulmuley
Senior Software Engineer
Turtlemint Pvt. Ltd.

Tapan Kumar Sahoo
Associate Professor, Computer Science
IIIT, Bhubaneswar

Acknowledgement

The elation and gratification of this seminar will be incomplete without mentioning all the people who helped me to make it possible, whose gratitude and encouragement were invaluable to me. I would like to thank God, almighty, our supreme guide, for bestowing is blessings upon me in my entire endeavor. I express my sincere gratitude to Harshad Mulmuley & Prof. Tapan Kumar Sahoo for his guidance and support and students of my class for their support and suggestions.

Bhushan Khanale

B516025, CE

Abstract

Most of the databases now are shared between different tenants giving it a more complex architecture. Hence any updates being made to the database have to be properly authenticated and verified that the changes are for that specific tenant only. This project report introduces the process of creating a REST API service to manage database changes with integrated authentication using Django. REST is acronym for **RE**presentational **S**tate **T**ransfer. It is architectural style for distributed hypermedia systems. Django is an open-source high-level Python Web framework that encourages rapid development and clean, pragmatic design. By the features of Django and Django REST Framework these updates to the database are are much simpler and protected.
***Keywords*:** rest, django, database

Contents

Chapter 1

Introduction

In any service dealing with the database it becomes extremenly important to have a constant database structure in place before moving on towards the buisiness logic. In Django we define the service in terms of `app`, `models`, `views` and `services`. These four parts represent the core logic service. `Views` take care of the exchange of the request and response objects from APIs. Usually when a API is called, a request object is sent to the server containing information about the request being made. The server then has to return the appropriate Response object which then the browser parses and outputs for the user. This exchange between request and response is a part of Views.

1.1 Background

Turtlemint has a separate database which records most of the things related to insurance policy issuance. This data is very volatile and is expected to change every month. Due to this, it becomes harder to change the database everytime there is a change in the information. To handle this issue, the purpose of the project is to create a new service which would wrap the information change in terms of database calls and let the user seemlessly update the information.

1.2 Significance

The new service will be able to handle all information changes related to the database. Moreover the service would have an integrated authentication and authorization which allows multiple users to use this service at a time. Previously, someone from the development team had to intervene with the data team to manually create database queries and update accordingly. This process was not only time consuming but also was inefficient. The new service would solve this issue and would allow the data team itself to update the database.

1.3 Method used

The service is built using Django and Django Rest Framework (DRF) which are two Python packages built for faster development of database-driven web applications. Django is also open-source and allows users to modify the report, modify any bugs if they found any. This helps for long term support applications. Django has three major parts: `models`, `views` and `templates`. Models are used to create database schema, views contain the business logic and templates are used for user interface.

1.4 Limitations

Django being open-source does help is most issues. Although, since Django was built to reduce the developement time significantly it might still not have all features of a system with independent database architecture. Django also introduces the concept of migrations which are a set of database schema changes maintained as a set of files. These migrations can be difficult to manage if an applications is prone to lot of database changes.

1.5 Project Structure

Django has already defined its project structure. Every Django project has some applications. Every applications represents set of logic related to one purpose or business objective. Every project can have any number of applications inside it. There is a common `settings.py` file which is used for managing settings for all applications.

The basic structure of the project can be represented as below:

```
project
├── settings.py
├── app1
│   ├── models
│   ├── views
│   └── templates
└── app2
    ├── models
    ├── views
    └── templates
```