Pratibodh

A Journal for Engineering

A free and Open Access Journal

Homepage: https://pratibodh.org

RATNA & BM RANVEER Foundation

Development of Food Delivery Service Using Django

Akashat Jain[1], Sakshi Jain[2], Sarthak Jain[3], Manju Vyas[4]
Department of AI&DS
JECRC, Jaipur

Abstract

The Django framework works with the PostgreSQL database to create a simple online notification flow. User authentication: Use Django's built-in authentication system to use user registration, access and manage personal information to secure access to the platform. Order Management: Create an intuitive interface for users to add items to their cart, view options, place orders and access order history. Push notifications: Leverage Django WebSocket to provide instant updates on order status, including confirmations, meal plans, and estimated delivery times. Deployment: Deploy the entire project in docker and use Kubernetes to provision the capabilities of the application to multiple users. PostgreSQL database: Use PostgreSQL as a database management system to store and retrieve data efficiently, benefiting from its reliability and scalability.

Article Status

Keywords: Python, Django, Docker, Kubernetes, SQL, Postgres

Available online :

1. Introduction

In recent years, web applications continue to meet interactivity and user experience requirements. This change was achieved in part by the integration of real-time applications, which allows users to interact with applications and receive updates without needing to refresh the page. Among the technologies that facilitate this change, the WebSocket protocol, which enables the responsive, engaging and information-rich web, stands out as the main means of communication between the client and the server.
Django Rest Framework (DRF) is a widely used framework; It is a conceptual, comprehensive API model for building RESTful APIs with Django. Essentially, DRF integrates with Django's core functionality (templates, views, and URLs) to easily and seamlessly build RESTful APIs. DRF includes elements such as views or serializers. Model in Django is python Django's SQL used to create tables with their own properties. In a model-view-controller (MVC) architecture, look at how content is presented to users for consumption and analysis. In the Django framework, a view is a Python function or class that receives a web request and returns a web response. This response can be a simple HTTP response, an HTML template response, or an HTTP redirect response that redirects the user to another page. Views contain the logic needed to respond by returning data to the user in each form. As a best practice, the logic for rendering views is stored in the Django application's views.py file. Now let's go back to the database application we used in this project, PostgreSQL is one of the most widely used database applications general Since it is open source software, the source code is available under the PostgreSQL license (Free Open Source License). Anyone with knowledge is free to use, modify, and distribute PostgreSQL in any form. Since it is very stable, very little effort is required to maintain this DBMS. Next, we installed WebSocket, an advanced technology designed for communication between two websites and the customer's browser. It is an alternative to HTTP communication on the web, allowing you to send messages to the server and receive a response without waiting for a response from the server. Provides continuous data, full-duplex communication with low latency.

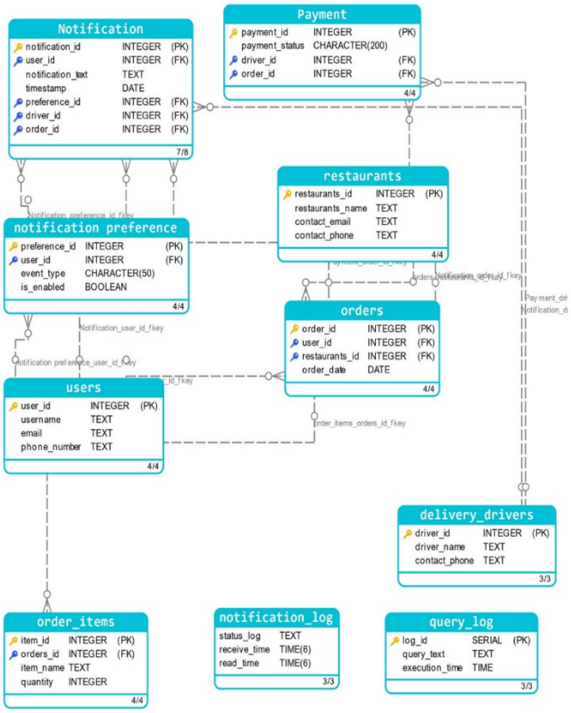
2. Research Methodology

Many websites and research documents have been created to remind us of the concept and importance of learning, here are a few: Django Rest Framework, documentation for Django, Postgres repository, WebSocket, Docker, and Kubernetes.
StackOverFlow: This is an error handling website for customers.
Medium: This website's standard blog works.

3. Proposed Approach

Database configuration: Combining Postgres and Django by taking a Django model with relevant fields (such as User, Order, Restaurant, Notification) and connecting it to PostgreSQL using psycopg2. The diagram of the table is below.

This creates a powerful SQL database to store customer information, order details and delivery status. This information includes tables for customers, orders, delivery personnel, and notification preferences. From Figure 1, we can understand that these tables are related to each other and one table depends on another table.



Notification		
notification_id	INTEGER	(PK)
user_id	INTEGER	(FK)
notification_text	TEXT	
timestamp	DATE	
preference_id	INTEGER	(FK)
driver_id	INTEGER	(FK)
order_id	INTEGER	(FK)
		7/8

Payment		
payment_id	INTEGER	(PK)
payment_status	CHARACTER(200)	
driver_id	INTEGER	(FK)
order_id	INTEGER	(FK)
		4/4

restaurants		
restaurants_id	INTEGER	(PK)
restaurants_name	TEXT	
contact_email	TEXT	
contact_phone	TEXT	
		4/4

notification preference		
preference_id	INTEGER	(PK)
user_id	INTEGER	(FK)
event_type	CHARACTER(50)	
is_enabled	BOOLEAN	
		4/4

orders		
order_id	INTEGER	(PK)
user_id	INTEGER	(FK)
restaurants_id	INTEGER	(FK)
order_date	DATE	
		4/4

users		
user_id	INTEGER	(PK)
username	TEXT	
email	TEXT	
phone_number	TEXT	
		4/4

delivery_drivers		
driver_id	INTEGER	(PK)
driver_name	TEXT	
contact_phone	TEXT	
		3/3

order_items		
item_id	INTEGER	(PK)
orders_id	INTEGER	(FK)
item_name	TEXT	
quantity	INTEGER	
		4/4

notification_log	
status_log	TEXT
receive_time	TIME(6)
read_time	TIME(6)
	3/3

query_log		
log_id	SERIAL	(PK)
query_text	TEXT	
execution_time	TIME	
		3/3

Notification_preference_id_fkey
Notification_user_id_fkey
Notification_preference_user_id_fkey
id_fkey
orders_restaurants_id_fkey
Notification_orders_id_fkey
Payment_dri
Notification_d
order_items_orders_id_fkey

Fig-1.Database schema

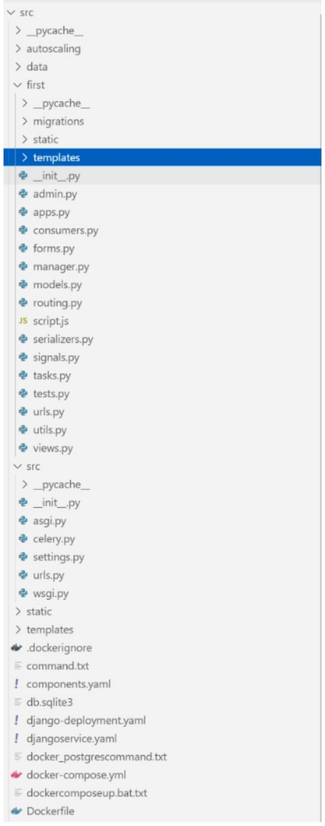
4.Backend with Django:

Django is a Python-based web framework that allows you to quickly create web applications. It is also known as battery-based because it provides functionality for everything from Django, including the Django admin interface, the default database SQLlite3, and more. When you create a website, you always need a similar system: a way to manage user authentication (registration, login, logout), website admin panel, information, ways to send information, etc. Django gives you a ready-to-use product, making rapid development easy. In Figure 2 we can see what the design of the Django backend looks like.

Models: Django models are Django's built-in functions for creating tables, fields, and various constraints. Django Models is a SQL library used with Django.

Views: Views are part of the user interface - they are often used as Html/CSS template views as classes or functions from which we can refer to the application's URLs. In the function or class, if the request is POST or GET, the relevant data will be saved in the file with the help of http response and we can change the html from referencing the URL in the application.

Templates: Django templates are written in .html files using HTML, CSS.



```
v src
  > __pycache__
  > autoscaling
  > data
  v first
    > __pycache__
    > migrations
    > static
    > templates
    __init__.py
    admin.py
    apps.py
    consumers.py
    forms.py
    manager.py
    models.py
    routing.py
    script.js
    serializers.py
    signals.py
    tasks.py
    tests.py
    urls.py
    utils.py
    views.py
  v src
    > __pycache__
    __init__.py
    asgi.py
    celery.py
    settings.py
    urls.py
    wsgi.py
  > static
  > templates
  .dockerignore
  command.txt
  components.yaml
  db.sqlite3
  django-deployment.yaml
  djangoservice.yaml
  docker_postgrescommand.txt
  docker-compose.yml
  dockercomposeup.bat.txt
  Dockerfile
```

Fig 2. Django File Structure

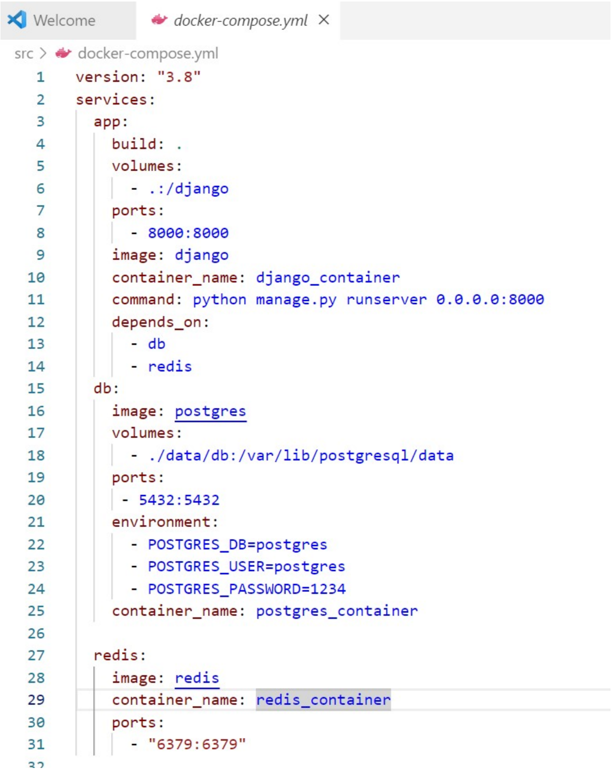
5.Docker:

With docker and docker domain we can run multiple containers simultaneously.

Docker Image: It is a file consisting of many layers and used to run the code in the docker container.

Docker Container: Time to work on the images. It allows developers to package applications with all the necessary components such as libraries and other addresses, can be seen in Figure 3 below.

Docker file: It is a script that contains the necessary commands that, when executed, helps compile the Docker image. Docker images are created from Docker files.



```
version: "3.8"
services:
  app:
    build: .
    volumes:
      - .:/django
    ports:
      - 8000:8000
    image: django
    container_name: django_container
    command: python manage.py runserver 0.0.0.0:8000
    depends_on:
      - db
      - redis
  db:
    image: postgres
    volumes:
      - ./data/db:/var/lib/postgresql/data
    ports:
     - 5432:5432
    environment:
      - POSTGRES_DB=postgres
      - POSTGRES_USER=postgres
      - POSTGRES_PASSWORD=1234
    container_name: postgres_container

  redis:
    image: redis
    container_name: redis_container
    ports:
      - "6379:6379"
```

Fig 3.Docker Compose

6.Kubernetes:

Kubernetes is a flexible, extensible, and open-source environment for managing containers and services, simplifying deployment and automation.
As the number of people increases, Kubernetes will automatically add pods via auto-scaling to prevent the server from failing. It shows the growth of pods as shown in Figure 4.
Kubernetes Features:

- Automatic rollout and rollback
- Storage orchestration

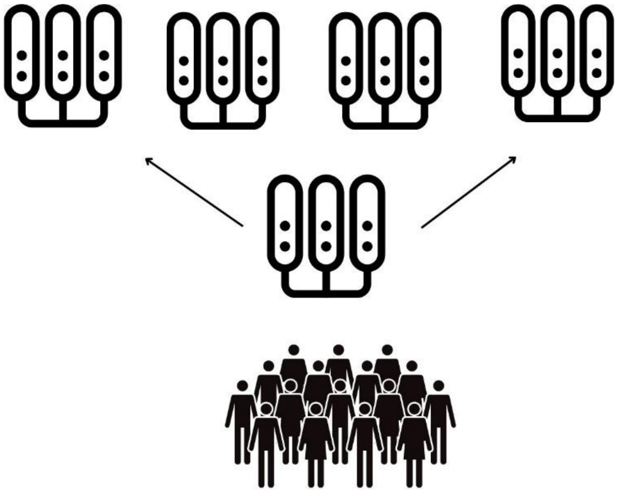


Fig 4.Kubernetes scaling up

As the number of users decreases, the server reduces its Pods based on the current number of users on the server.
In Figure 5, we can see that Pods reduce the throttling rate of people.

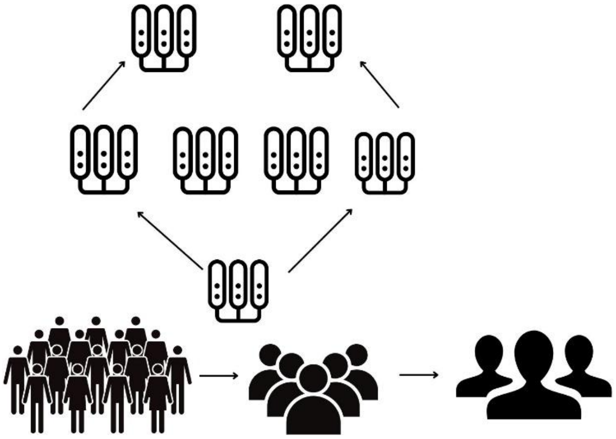


Fig 5. Kubernetes Scaling Down

Steps to use Docker and Kubernetes:

- Create a Docker profile.
- Create images from Docker archives.
- Use docker-compose to combine multiple applications into a single application and configure commands for each application so that all dependent applications can work together
- Upload the image to hub .docker.com. Use docker to login and push images to account
- Create deployment files and yaml service files and deploy them using kubectl commands.
- Use the kubectl define command to view containers running on Kubernetes cluster workers.

7.Conclusion:

In conclusion, using notification flow in our Django project brings a new level of efficiency and customer interaction.
We take user experience to a new level by leveraging the power of real-time innovation and communication. From order placement to delivery, our systems ensure transparency, accuracy and personalized service. Make every deployment a smooth and enjoyable journey using Docker and Kubernetes.

8.Future Enhancement:

Future improvements may include improving the functionality of the food reporting portal, introducing new technologies such as blockchain to protect user data and any changes, and allowing users to pay through the app. To make the portal more user-friendly, cooking video facilities are also provided and translation facilities are also provided.

References:

1. About Company. (n.d.). Retrieved from http://www. tutorialspoint.com/.
2. The Physics Classroom. (n.d.). Retrieved from https://www.physicsclassroom.com/.
3. Kelly, L., & Breault, K. (2006). Developing Educational Websites: Investigating Internet Use by Students and Teachers. In Proceedings of Thinking, Evaluating, Rethinking, ICOM-CECA Conference, Rome.
4. AglaSem Admission. (n.d.). Retrieved from https://admission.aglasem.com/
5. Sachan, N. (2019, February 20). Welcome to BHU Student Club, BHU Student Club. Retrieved from http://bhustudentclub.in/.
6. Jalote, P. (2003). An Integrated Approach towards Software Engineering. Narosa Publishing House.
7. Musciano, C., & Kennedy, B. (1996). HTML, The Definitive Guide. O'Reilly & Associates.
8. Powell, T. A. (2010). HTML & CSS: The Complete Reference. The McGraw-Hill Companies.
9. Flanagan, D. (2006). JavaScript: The Definitive Guide. O'Reilly Media, Inc.
10. Shenoy, A., & Sossou, U. (2014). Learning Bootstrap. Packt Publishing Ltd.
11. Kuhlman, D. (2011). A Python Book: Beginning Python, Advanced Python, and Python Exercises. Platypus Global Media.
12. Holovaty, A., & Kaplan-Moss, J. (2008). The Definitive Guide to Django: Web Development done right. Apress.
13. Lokhande, P. S., Aslam, F., Hawa, N., Munir, J., & Gulamgaus, M. (2015). Efficient way of Web Development using Python and Flask. International Journal of Advanced Research in Computer Science, 54-57.
14. Thakur, M. S. (2017). Review on Structural Software Testing Coverage Approaches. International Journal of Advance Research, Ideas and Innovations in Technology, 281-286.
15. MKdos, Encode -OS LTD 2011-presenthttps://www.django-rest-framework.org/
16. Read the docs -sphinx using theme https://channels.readthedocs.io/en/latest/
17. SpringGuide -Github : https://github.com/spring-guides/gs-messaging-stomp-websocket
18. Akash Shrivastava (2022)-Published in ScaleReal Medium - https://medium.com/scalereal/push-notifications-through-django-db528c303b91