Assignment 7

Roll no: 43144

Name: Ruchika Pande

Title: Microservices framework based distributed applications. .

Problem Statement:

To develop microservices framework based distributed applications.

Objectives:

- To study about microservices framework.

Theory:

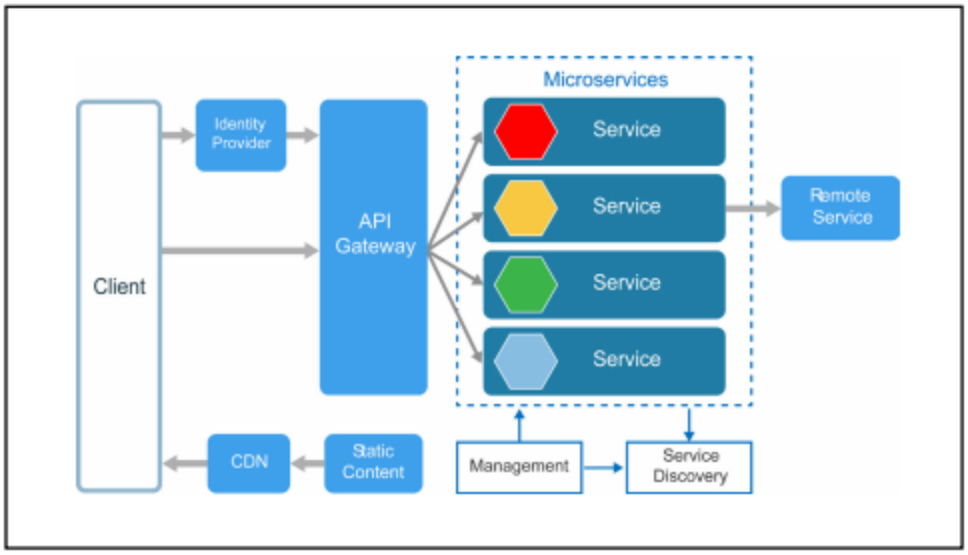
Microservices:

Microservices - also known as the microservice architecture - is an architectural style that structures an application as a collection of services that are

- Highly maintainable and testable
- Loosely coupled
- Independently deployable
- Organized around business capabilities
- Owned by a small team

The microservice architecture enables the rapid, frequent and reliable delivery of large, complex applications. It also enables an organization to evolve its technology stack.

Microservice architecture:



Features of microservices:

- Characteristics of its software: It may be broken down into several independent parts. To accomplish that, each of the services may be deployed and modified without affecting other functional aspects of the application.
- Characteristics of its organisation: The way in which they are organised implies a contrast with the monolithic environment, since they account for aspects such as the capabilities, needs and preferences of the business or client where they are to be implemented. As for the architecture, multi-purpose modules are used, achieving the creation of a common module for everyone, offering a specific service. Undoubtedly, the greatest advantage is time savings and convenience in maintenance tasks, making sure the rest of the team can complete their workday while maintaining a module.
- Characteristics of its architecture: Each module is independent, since each one of them has its own database; in other words, not all services connect with the same database. Therefore, overloading and application downtime is prevented.
- Characteristics of their warnings and actions: Since several services are communicated, we need to have warning and action systems in case there is any failure on these services. In other words, we would get a warning, an email would be sent to support, etc. This system is beneficial, since it favours a good operation between the remaining functional modules.

Advantages:

- Minimal work team

- Scalability
- Modular functionality, independent modules.
- Developer freedom to develop and deploy services independently
- Use of containers, allowing for a quick deployment and development of the application

Disadvantages:

- High memory use
- Time required to fragment different microservices
- Complexity of managing a large number of services
- Developer need to solve problems such as network latency or load balancing
- Complex testing over the distributed deployment

Web frameworks:

'Web application frameworks' or 'web frameworks' as *"a software framework that is designed to support the development of web applications including web services, web resources and web APIs".* In simple words, web frameworks are a piece of software that offers a way to create and run web applications. Thus, you don't need to code on your own and look for probable miscalculations and faults.

In earlier days of web app development, web frameworks were introduced as a means to end hand-coding of applications where just the developer of a particular app could change it. That was long ago, now we have web-specific languages and the trouble with changing an app's structure is resolved because of the arrival of a general performance. Now, depending upon your task you may choose one web framework that fulfills all your requirements or converges multiple frameworks.

Types of Web frameworks:

As web standards began to advance, the app logic shifted towards the client- ensuring smarter communication between the user and the web application. With logic on client-side, they (client) can react swiftly to user input. This makes web apps more responsive, easily navigate-able on any device. Thus, we have two functions of frameworks – a) the one to work on the server side, that helps to set up app logic on the server (backend) or b) to work on the client-side (front end).

The front-end frameworks mostly manage the external part of the website, i.e. what the user sees when they open the application. The back-end manages the internal part. Let's take a deeper look

Server-side Frameworks:

The rules and architecture of the server-side frameworks permit you to create simple pages, landings and forms of different kinds. To create a web application with a well-developed interface you need a wider range of functionality. Server-side frameworks handle HTTP requests, database control and management, URL mapping, etc. These frameworks can improve security and form the output data- simplifying the development process. Some of the top server-side frameworks are –

- NET (C#)
- Django (Python)
- Ruby on Rails (Ruby)
- Express (JavaScript/Node.JS)
- Symfony (PHP)

Client-side Frameworks:

Client-side frameworks don't take care of the business logic like the server-side ones. They function inside the browser. Therefore, you can enhance and implement new user interfaces. A number of animated features can be created with frontend and single page applications. Every client-side framework varies in functionality and use. Here are some client-side frameworks for comparison's sake; all of whom use JavaScript as their programming language-

- Angular
- Ember.JS
- Vue.JS
- React.JS

Web application framework- Architecture

Most of the web frameworks depend on the MVC (Model-View-Controller) architecture. The reason why this pattern is preferred lies in its rational design that separates the app logic from the interface and forms the three essential parts that are represented in the architecture's name – MVC (Model-View-Controller).

Model

The Model comprises all the data, business logic layers, its guidelines and functions. The Model, upon getting user input data from the Controller, tells the way an updated interface should be displayed directly to the View.

View

The View is for the graphical representation of the data like graphs or charts etc. It is the apps' front-end. The View gets the user input and communicates the same to the Controller for examination and then updates and reconstructs itself according to the Model's instructions, or the Controller's in case the modification requirement is minimum.

Controller

The Controller translates the input data into the scope of commands of the previous ones. It is the midway between the Model and the View. It gets the user input from the View; after processing it, the Controller notifies the Model (or View) of the changes required.

The talking point about many discussions regarding the Controller is that it isn't always essential (giving more importance to the separation of logic from the interface). Assigning input processing, however, to either Model or View messes up the MVC's traditional mantra of 'Separated Presentation'-where tasks are separated on type-basis. For the project to remain transparent, adaptable and maintainable, each component in the MVC must be responsible for a sole line of tasks.

Common Web Framework Functionalities:

Frameworks give functionality in their code or through extensions to complete everyday operations needed to run web applications. These operations involve:

1. URL routing
2. Input form managing and validation
3. HTML, XML, JSON, and other product setups with a templating engine
4. Database connection configuration and resolute data manipulation through an object-relational mapper (ORM)
5. Web security against Cross-site request forgery (CSRF), SQL Injection, Cross-site Scripting (XSS) and other frequent malicious attacks
6. Session repository and retrieval

Implementing the solution:

1. Using Virtual Environments: Install virtualenv for development environments. virtualenv is a virtual Python environment builder. It helps a user to create multiple Python environments side-by-side. Thereby, it can avoid compatibility issues between the different versions of the libraries.

The following command installs virtualenv: Sudo apt-get install virtualenv

2. Flask Module: Importing flask modules in the project is mandatory. An object of Flask class is our WSGI application. Flask constructor takes the name of the current module (__name__) as argument. The route() function of the Flask class is a decorator, which tells the application which URL should call the associated function.

Route decorator: The route() decorator in Flask is used to bind URL to a function. For example − @app.route('/hello') def hello_world(): return 'hello world'

Here, the URL '/hello' rule is bound to the hello_world() function. As a result, if a user visits http://localhost:5000/hello URL, the output of the hello_world() function will be rendered in the browser.

3. Writing the subroutine for the four microservices: There are four microservices viz., user, Showtimes, Bookings and Movies for which microservices are to be implemented.

Conclusion:

In this assignment, we learned about microservices. We learned that with microservices, modules within software can be independently deployable. In a microservices architecture, each service runs a unique process and usually manages its own database. This not only provides development teams with a more decentralized approach to building software, it also allows each service to be deployed, rebuilt, redeployed and managed independently.