VFI Architecture



VFI Architecture is built on micro services architecture where each service would run on a spring boot application (Tomcat server or any other java application server such as jetty) backed with a online memory database H2 for this demo. The database would get resurrected for every run of the server for this demonstration.

The services itself could be tested with any REST client such as Postman or any GUI can be hooked up to call the same services. We have a sample Angular Application for this demonstration.

# Assumptions

The following tool set would be needed to run the server and client.

For running the server

1. Java JDK 8
2. Maven installed with latest version
3. Need a terminal window to run the server or any java IDE such as eclipse

For running the client, one would need

1. Node installed
2. Need a terminal window to run the server or any java IDE such as eclipse
3. Angular-cli installed or plugin for angular cli installed in IDE

# API’s supported for the demo

1. Add the demo data from a data set ( http://localhost:8090/api/vfi/data)
2. Get a list of subscribers in the database. (http://localhost:8090/api/vfi)
3. Get a subscriber based on a key (<http://localhost:8090/api/vfi/0000002178-18-000067>)

# Continuous Integration, Continuous Deployment (CICD)

All code created would have unit tests created along with it. Jenkins will be used to create jobs that would trigger a build and test as soon as new code is checked in.

# Production Deployment

This service is one of multiple services that would be deployed in premise or in a cloud. All these services would be handled by an API Gateway that would take care of the load balancing and version control of the api management of the delivery.



Production deployments can be done using on-premise or cloud installations. Docker can be used to create, deploy and run this application in a container model that could be managed and used to scale vertically and horizontally using kubernetes which is beyond the scope of this demonstration.