**Architecture of the Nimbus Frame work**

**Nimbus is a framework which stands for Network Information management client-based user services**

This framework reduces application development time by

a)Providing the ability to build application through configuration.

b)Providing boilerplate code for cross cutting concerns.

--> In web you can find location of documentation

in <https://github.com/openanthem/nimbus-docs/tree/master/src/nimbus-core>

**Key Features of Framework**

Open source - no licensing cost

Customized Workflow

Configurable UI

Multi Tenancy

Cloud based solution

--> Following technologies are using in this framework. They are

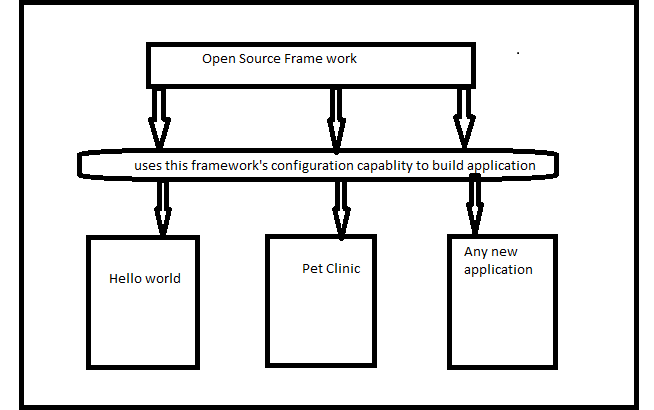
**FrontEnd** : Angular 4, RxJs(Reactive Extensions Library for JavaScript), SASS(Syntactically awesome style sheets)

**BackEnd** : Spring framework components, Activiti, Drools, Query DSL, RxJava(implementation of ReactiveX in Java)

#### **Famework Structure**

The framework can be used to churn out different applications based on the client needs.

nimbus framework enables an individual/team to build application quickly.



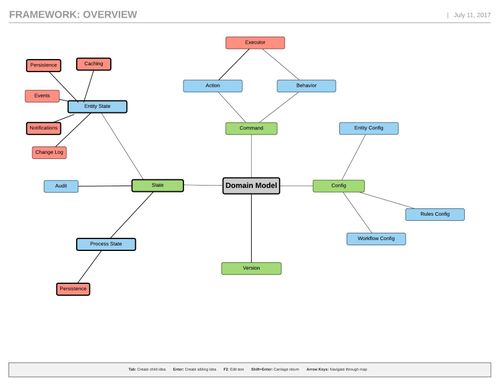
#### **Technical Architecture overview**

1. **Domain Model** ..> To build the application first need to understand business entities and definitions.

**2)Config** ..>Define the configurations to view.It can mapped to the domain

**3)Command** ..>It is the instruction that framework execute and return with an output.Example click the button excute business logic

**4)State** - The value of every entity and its corresponding attributes is referred to as state by the framework.

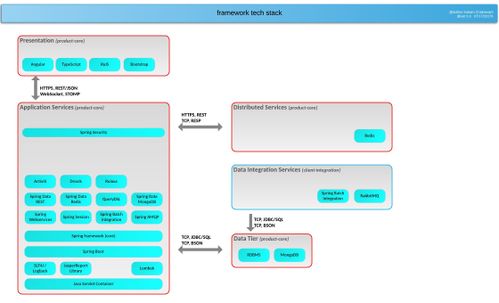


Above diagram showing how Domain module mapping to config, command ,state,version and also how each one having different states.

#### Need configurations Business Entity Configuration,View Configuration,Business Rule Configuration,Workflow Configuration

The framework processes information using url. The url is based on a query dsl structure.

Commands to be using for the Quey DSL are /p,\_new(create new instance model),\_get(fetch instance based on the ID),\_save(save model in DB),\_replac(replace state of model),\_update(Update state of the model),\_remove (remove the model),\_search(search model under criteria),\_process(Execute work flow)



**Configurations**-Framework has the ability to create Configuration an Audit Configuration for leaf parameters to generate audit history on state change of the annotated parameter.

Collection configuration is introduced in the framework to eliminate the creation and use of different functional handlers for adding, editing and deleting elements from/ to a collection.

**New Nimbus Project steps**

**Here can find the steps how to create Nimbus Project**

Create a new Maven project to create the Hello World app. Under src/man/java, create a class called "ViewHelloWorld.java" Lets configure the page of our UI component.

package com.anthem.platform.client.helloworld;

import javax.validation.constraints.NotNull;

import com.anthem.oss.nimbus.core.domain.definition.Domain;

 import com.anthem.oss.nimbus.core.domain.definition.Domain.ListenerType;

import com.anthem.oss.nimbus.core.domain.definition.Model;

import com.anthem.oss.nimbus.core.domain.definition.ViewConfig.Page;

import com.anthem.oss.nimbus.core.domain.definition.ViewConfig.Section;

import com.anthem.oss.nimbus.core.domain.definition.ViewConfig.TextBox;

 import com.anthem.oss.nimbus.core.domain.definition.ViewConfig.Tile;

import lombok.Getter;

 import lombok.Setter;

 @Domain(value = "helloworld", includeListeners={ListenerType.websocket})

@Getter @Setter

**public** **class** ViewHelloWorld {

 @Page(route="", defaultPage=true)

 private Page\_HelloWorld helloWorldPage; @Model @Getter @Setter

 public static class Page\_HelloWorld {

 @Tile(title="Hello World Sample", size=Tile.Size.Medium)

 private Card\_HelloWorld helloWorldCard;

}

After creating Configure Page create a new Application.java class to start up our spring boot project.

package com.anthem.platform.client.helloworld;

import org.springframework.boot.SpringApplication;

 import org.springframework.boot.autoconfigure.SpringBootApplication;

 @SpringBootApplication

 public class Application {

public static void main(String[] args) {

SpringApplication.run(Application.class, args);

System.out.println("\*\*\*\* Platform-helloworld \*\*\*\*\* Started !!");

} }

Define rules file which contains details on how the state of the form variables are set through web sockets. Create a file helloworld.drl and place under src/main/resources/.

package com.anthem.nimbus.platform.client.helloworld;

 import com.anthem.oss.nimbus.core.domain.model.state.internal.DefaultModelState;

import com.anthem.oss.nimbus.core.domain.model.state.EntityState.Param;

 import com.anthem.oss.nimbus.core.domain.model.state.internal.DefaultParamState;

 import java.time.LocalDate;

import java.time.Period; rule "Set State Change" no-loop true when $view : Param() then String type = (String) $view.findParamByPath("/helloWorldPage/helloWorldCard/helloWorldSec/type").getState(); System.out.println("type is"+type); $view.findParamByPath("/helloWorldPage/helloWorldCard/helloWorldSec/output").setState("Welcome to Nimbus Hello World! You typed: "+type); String output = (String) $view.findParamByPath("/helloWorldPage/helloWorldCard/helloWorldSec/output").getState(); System.out.println("output is "+ output);

end

As the last step, we need to let the framework know of the classes we configured. This can be done by creating a new application.yml file and including the below host name ,port number,domain and security

Run the web as spring boot app. The application we created is listening on port 8080. Now open your browser and enter [http://localhost:8080](http://localhost:8080/). You will now see the app that you configured.

**MangoDb**

Tech talk : Interface with MongoDB within the framework. Provide examples to show mapping of mongodb with Nimbus

**We are using mangodb in this framework. Following are the interfaces and classes where mangodb with this framework**

DBSearch.java( Interface)

MongoDBSearch.java ( Abstract class)

SearchCriteria.java (Abstract class): This contains static inner classes PageFilter,ProjectCriteria,QuerySearchCriteria,ExampleSearchCriteria,LookupSearchCriteria MongoSearchByQuery.java (Class which extends MongoDBSearch abstract class): It contains one inner class name as QueryBuilder and this class contains methods searchByAggregation(..),findAllPageable(..),buildProjectionPathBuilder(..)

MongoSearchByExample.java ( Class which also extends MongoDBSearch abstract class) : It contains methods like search(..) , pageRequestAndResponse(..),recurseAllFieldsAndBuildMatcher(..), buildQuery(..) DefaultMongoModelRepository.java: Contains all curd operation methods like \_new(..),\_update(..),\_replace(..),\_delete(..)

**We can find mangodb related information using following link**

https://github.com/openanthem/nimbus-core/tree/develop/nimbus-core/src/main/java/com/antheminc/oss/nimbus/domain

/model/state/repo/db