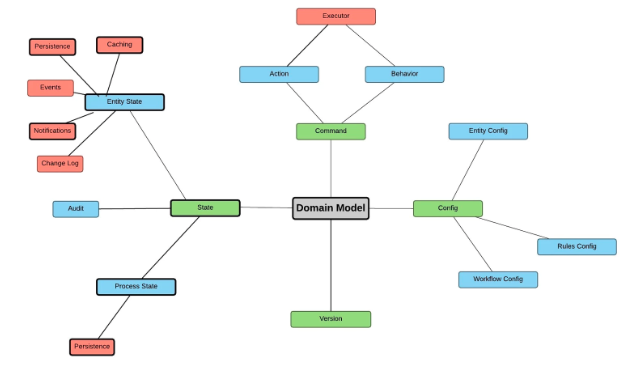
**Domain Model** - For any application we first have to define the business entity/entities. This would be the first step in the process of building the product.

**Config** - Once we have the domain model, we can define the configuration for the view, workflow and the rules. The view definition configs, the mapping to the domain model, the workflow(if any) and the corresponding view and core domain rules can be written.

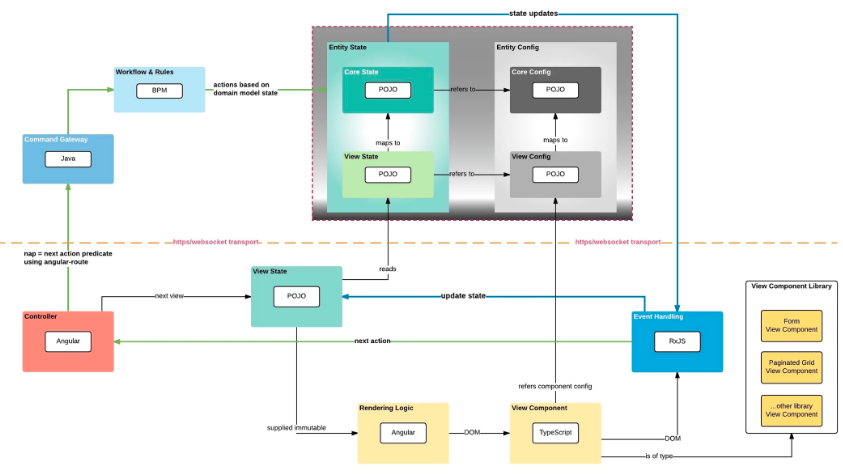
**Command** - The command is the instruction that the framework understands to execute and come back with an output. It is similar to writing the traditional method calls for button click to do some business logic but just that we have standardized the process of writing such to the domain specific language that the framework understands.

**State** - The value of every entity and its corresponding attributes is referred to as state by the framework. There could various events for example generated based on the state and certain other things associated with the state. To get the history of changes that happened on an entity attribute, we would need audit to be enabled.



Configuration

* Business Entity configuration
* View configuration
* Business Rule configuration
* Workflow Configuration



Command

* Query DSL: Query DSL provides a typesafe querying layer on top of JPA, JDO, JDBC and other backends.

To get started with Querydsl for MongoDB using a Maven 2 based build environment, follow the following steps.

Add the following dependencies to your Maven project and make sure that the Maven 2 repo of Mysema Source is accessible from your POM :

<dependency>

<groupId>com.mysema.querydsl</groupId>

<artifactId>querydsl-mongodb</artifactId>

<version>${querydsl.version}</version>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-log4j12</artifactId>

<version>1.6.1</version>

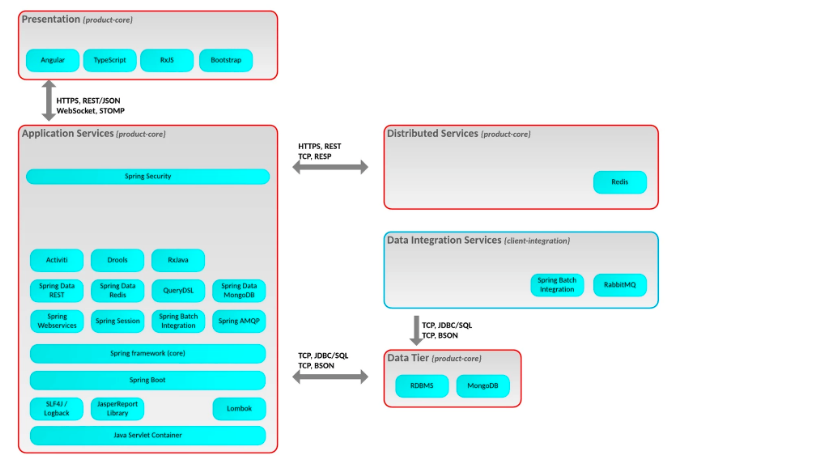
</dependency>

Framework Tech Stack

The framework is primarily having 2 main components -

**FrontEnd** - Built using Angular 4, RxJs, SASS.

**BackEnd** - Built using Spring framework components, Activiti, Drools, Query DSL, RxJava



**Nimbus Platform reduces application development time by**

* Providing the ability to build application through configuration.
* Providing boilerplate code for cross cutting concerns.