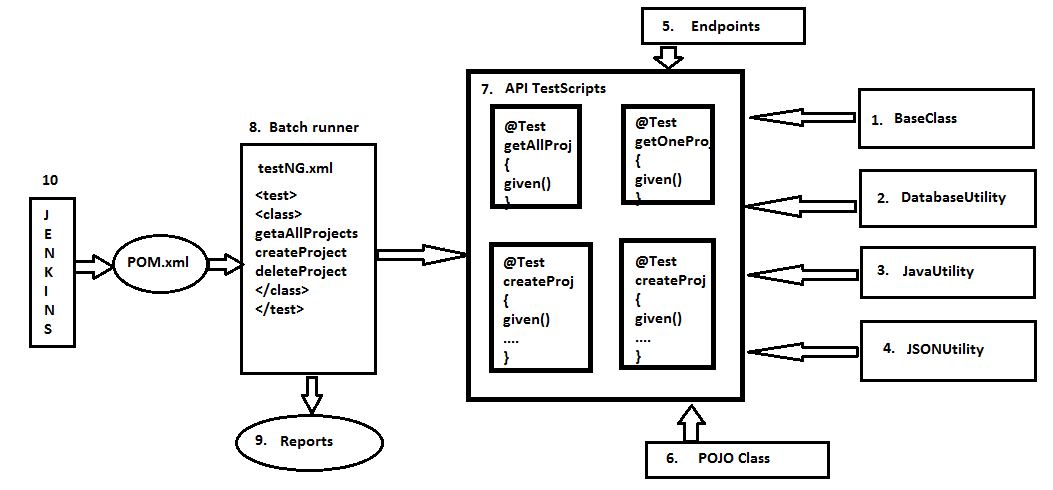
**API Rest Assured Framework Architecture**



Framework has the following basic components:

1. Base Class
2. Database Utility
3. Java Utility
4. JSON Utility
5. Endpoints
6. POJO Class
7. API Test Scripts
8. Run Me/driver/XML file
9. Reports
10. Jenkins

**Base class**: API base class contains common TestNG configuration annotations which is required for all the test script. As per the automation rule every test script should extend base class to use those annotations. In API we make use of following annotations:

* **@Before Suite**: this annotation is used to configure the base URI of the API and also the data base connection is established.
* **@After Suite**: data base connection is closed in this configuration

**Database Utility**: This is developed using JDBC which is utilised to connect to any database from the automation test script. In real time few scenarios require to connect to database for validation to execute precondition. It contains few generic methods to connect to database, close database, read data from data base.

**Java Utility**: It contains java specific methods which can be used for all the test scripts like get-Random-number, get-Date etc

**JSON Utility**: This class contains the JSON path, which is required for complex response validation, fetch individual data from the response and also for chaining two requests.

**End Points**: this is an interface in the framework where all the necessary endpoints of different APIs are stored and fetched as required.

**POJO class**: POJO class is a java design pattern which helps to handle complex requests, using POJO class serialization and de-serialization becomes easy

**API test scripts**: Itcontains collection of TestNG test scripts which is automated using @Test. During test script development make sure generic libraries, POJO class is being used. Typically BDD type of test script development is followed to achieve code optimisation and to avoid unnecessary storage of the responses.

**Run me/Driver/XML File:** It is a TestNG component used to execute all the test scripts in batch, parallel, group. A separate testNG.xml file is created for every build and the execution is triggered.

**Reports:** Whenever TestNG execution is completed it automatically generates HTML report which helps us to know the status of request. In Real time reporting component is very much important because it provides the quality of application and same report is being submitted to customer. In real time report helps the engineer to know the root cause of the issue whenever test Script is getting failed.

**Jenkins**: It’s a tool which will trigger the execution through the POM.xml file invoked in a maven project, in this pom.xml file using sure-fire plug-in, we can specify which particular testNG.xml has to be executed, and that will be triggered in a Jenkins job.