**Blogging Application**

**Building real time APIs for blogging application using Spring boot, Spring security, JWT, Spring Data JPA (Hibernate) and MYSQL**

**Technologies we use**

1. Spring Boot
2. Hibernate
3. MySQL
4. Postman
5. JWT
6. Swagger
7. AWS

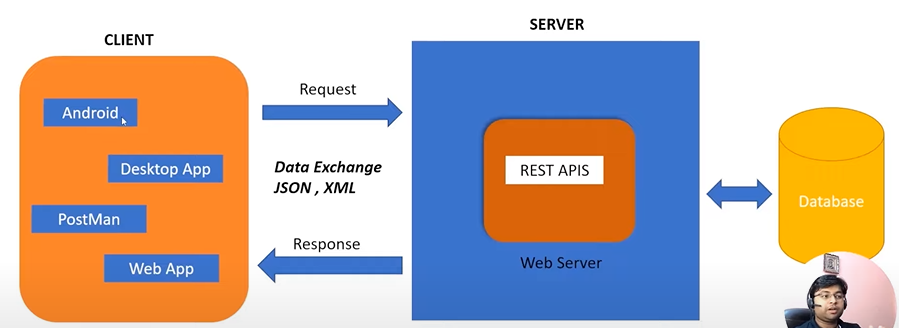
**What we will learn**

1. Creating Rest Endpoints
2. Complex DB Structure (JPA Entities)
3. Role Based Authentication
4. Handling Exceptions
5. Using DTO for data transfer
6. Swagger
7. How to add profiles for different environments
8. How to deploy sprint boot in productions.

**Pre-requisites**

1. Core java (oops, package, lambda, stream API etc)
2. Spring Framework (Spring Core (DI) , JPA and MVC)
3. Spring Boot Basics
4. Basics of MySQL DB
5. STS IDE

**Client Server Architecture (REST Architecture)**



**What is REST?**

Representational State Transfer

(Format xml, json) (Data) (Transfer between client/Server)

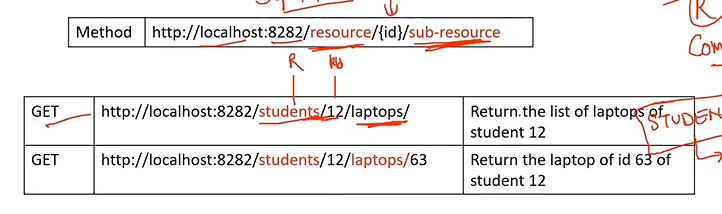
1. Software architectural style
2. URLs that we call

**Rest Guidelines**

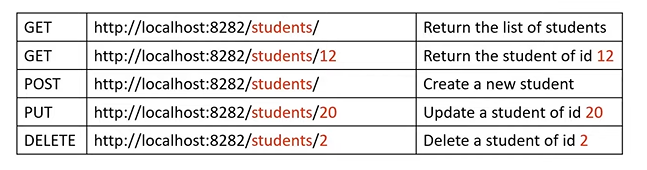
1. Follows Client server architecture (independent entity’s)
2. Stateless (No data to be stored on server (Client will manage data))
3. Cacheable
4. Layered Architecture
5. Uniform Interface
6. Code on Demand (run time code execute)

**Rest Concepts**

1. Resource
   1. Anything that we want to expose to outside world, through out application.
2. Sub-Resource
   1. Student -> Books
   2. If Student get deleted then book will also be deleted
   3. Sub resource can not exist without Resource



1. URI (uniform resource identifier)
   1. Used to identify the resource.



1. Http methods
   1. Defines some standard request methods which defines which actions to perform
      1. GET
      2. GET
      3. POST
      4. PUT
      5. DELETE
2. Http Response code
   1. 200 Ok
   2. 201 Resource created successfully
   3. 401 Unauthorized (Authentication required for the resource)
   4. 404 Resource Not Found
   5. 500 Internal Server error

**Requirements**

1. Build Simple Blogging application
2. Client wants blogging application where he/she can write blogs and articles.
3. User can comment on the blogs/article
4. User should create, update, delete and list posts.
5. User should add, update delete comments on posts
6. Categories the posts according to categories
7. User should be able to login to our application.
8. Post includes one picture too.

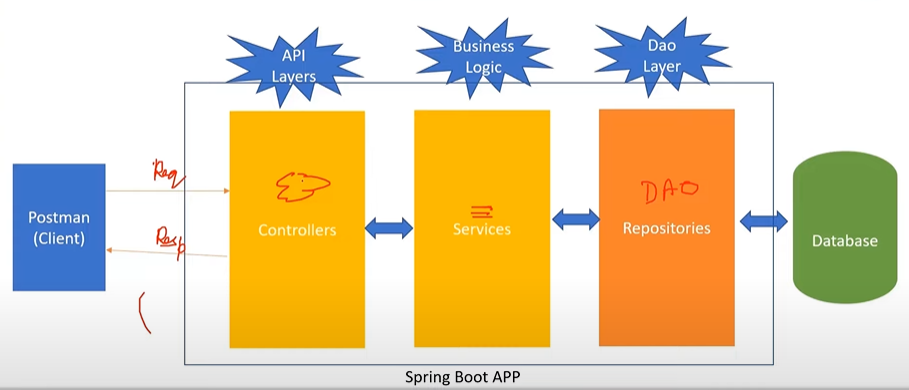
**Technical terms related to project**

1. Proper login and register API
2. Posts API include Pagination and Sorting
3. Proper user input validation handling
4. Proper exception handling
5. Role based authentication role-based security with APIs
6. JWT based authentication
7. Document all REST APIs so that consumer can easily understand.
8. Document the backend application on cloud platform.

**Resource for Blogging application**

1. User
2. Category
3. Post
4. Comments

**Best Architecture while using Spring boot for backend**



**Beans validation**

1. Java Bean is validated with JSR 380 known as Bean Validation 2.0.
2. JSR 380 is specification for the JAVA API for bean validation. Properties of bean meet the specific criteria.
3. For validation different annotations is used like @NotNull, @Min, @Size etc.
4. Hibernate Validator is an implementation for validation API.

**Important Annotations for the validations**

1. @NotNull
2. @Size
3. @Min
4. @Max
5. @Email
6. @NotEmpty etc

**How to use Bean Validation**

1. Add spring.boot.starter.validation dependency
2. Bean enable
3. Enable in controller

**JWT (Json Web Token)**

1. Used to secure REST APIs
2. Best wat to communicate security between client and server securely.
3. JWT follows a stateless authentication mechanism.

**Steps to Implement JWT**

1. Add Dependency (op.jsonwebtoken)
2. Create JWT authenticationEntryPoint class implement AuthenticationEntryPoint
3. Create JWTTokenHelper class
4. JWTAuthenicationFilter extends OncePerRequestFilter
   1. Get jwt token from request
   2. Validate token
   3. Get user from token
   4. Load user associated with token
   5. Set spring security
5. Create jwtAuthResponse
6. Configure jwt in spring security config
7. Create login api to return
8. Test the application

**Documenting API via Swagger Tool**

1. Add dependency
2. Add urls for swagger with permit all

**Deployment on Amazon**

1. Elastic BeanStack
2. Amazon RDS
3. Amazon Route S3
4. Cloud Object Storage (S3)