# CODE DOCUMENTATION

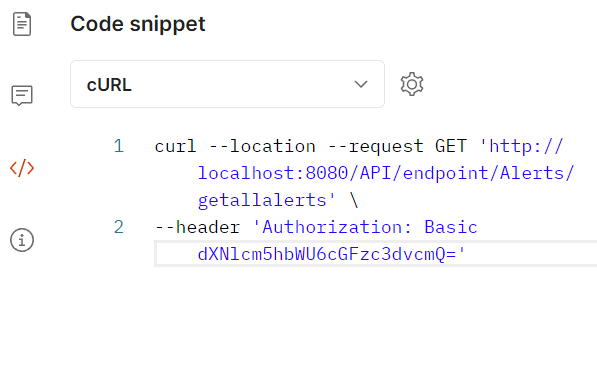
**I GET All Alerts**

End Point <http://localhost:8080/API/endpoint/Alerts/getallalerts>

Method GET

Produces JSON

Session Management N/A – Cookies

Curl

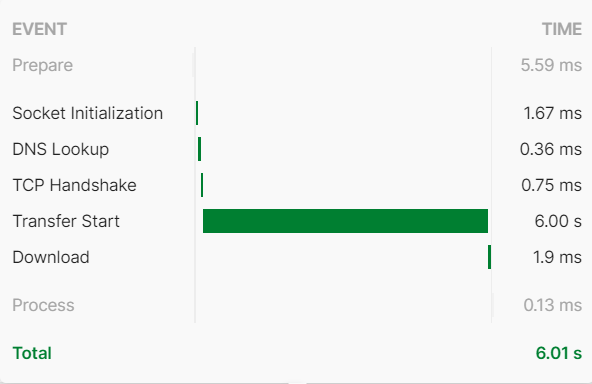
 Image 1- CURL

Image 2 (Transmission Rate)

**Transaction output**

Dec 21, 2022 4:20:01 PM org.glassfish.jersey.filter.LoggingFilter log

INFO: 2 \* Server has received a request on thread http-nio-8080-exec-7

2 > GET http://localhost:8080/API/endpoint/Alerts/getallalerts

2 > accept: \*/\*

2 > accept-encoding: gzip, deflate, br

2 > authorization: authorization: **Basic dXNlcm5hbWU6cGFzc3dvcmQ=**

2 > connection: keep-alive

2 > host: localhost:8080

2 > postman-token: 560f8ce4-b3d9-416c-9af2-f501c570adec

2 > user-agent: PostmanRuntime/7.30.0

username

password

security filter called

GET ALETS FUNCTION INVOKED

Hibernate:

/\*

from

Alert \*/ select

a1\_0.alertId,

a1\_0.AlertMessage,

a1\_0.AlertTitle,

a1\_0.date

from

AlertModel a1\_0

FINAL executed

Dec 21, 2022 4:20:01 PM org.glassfish.jersey.filter.LoggingFilter log

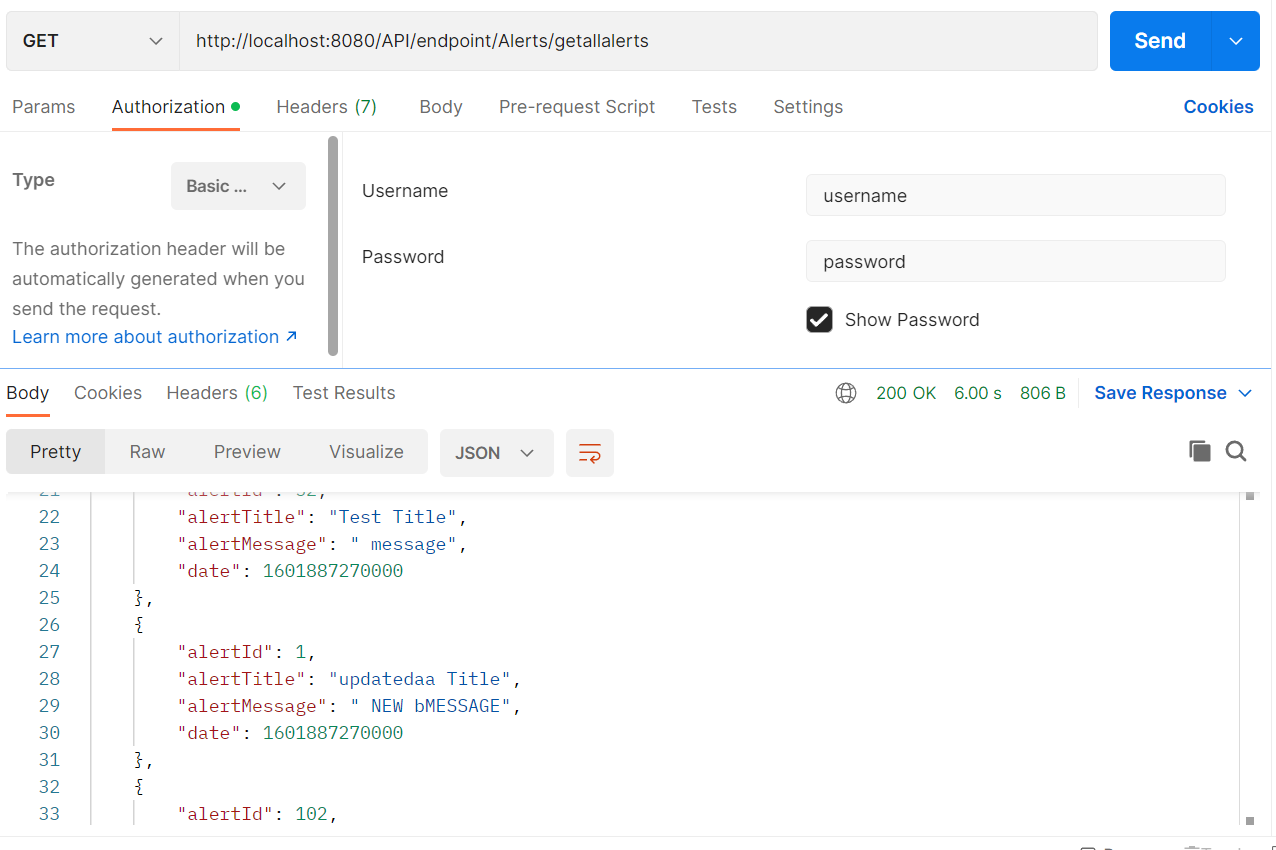
INFO: 2 \* Server responded with a response on thread http-nio-8080-exec-7

2 < 200

2 < Content-Type: application/json

Encryption with BASIC-AUTH (authentication and Authorization)



 Response output

**II – GET SINGLE ALERT**

End Point http://localhost:8080/API/endpoint/Alerts/getalert/1

Method GET

Produces JSON

Session Management N/A – Cookies

Transmission output

INFO: 1 \* Server has received a request on thread http-nio-8080-exec-3

1 > GET http://localhost:8080/API/endpoint/Alerts/getalert/1

1 > accept: \*/\*

1 > accept-encoding: gzip, deflate, br

1 > authorization: **Basic dXNlcm5hbWU6cGFzc3dvcmQ=**

1 > connection: keep-alive

1 > host: localhost:8080

1 > postman-token: 7cfd2063-155f-4b08-9b51-6837d5eccaec

1 > user-agent: PostmanRuntime/7.30.0

username

password

security filter called

Dec 21, 2022 4:17:03 PM org.hibernate.Version logVersion

INFO: HHH000412: Hibernate ORM core version 6.0.2.Final

Dec 21, 2022 4:17:04 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl configure

WARN: HHH10001002: Using built-in connection pool (not intended for production use)

Dec 21, 2022 4:17:04 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator

INFO: HHH10001005: Loaded JDBC driver class: org.postgresql.Driver

Dec 21, 2022 4:17:04 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator

INFO: HHH10001012: Connecting with JDBC URL [jdbc:postgresql://localhost:5432/WeConnect?createDatabaseIfNotExist=true&&autoReconnect=true]

Dec 21, 2022 4:17:04 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator

INFO: HHH10001001: Connection properties: {password=\*\*\*\*, user=postgres}

Dec 21, 2022 4:17:04 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator

INFO: HHH10001003: Autocommit mode: false

Dec 21, 2022 4:17:04 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl$PooledConnections <init>

INFO: HHH10001115: Connection pool size: 20 (min=1)

Dec 21, 2022 4:17:04 PM org.hibernate.engine.jdbc.dialect.internal.DialectFactoryImpl logSelectedDialect

INFO: HHH000400: Using dialect: org.hibernate.dialect.PostgreSQLDialect

Dec 21, 2022 4:17:07 PM org.hibernate.resource.transaction.backend.jdbc.internal.DdlTransactionIsolatorNonJtaImpl getIsolatedConnection

INFO: HHH10001501: Connection obtained from JdbcConnectionAccess [org.hibernate.engine.jdbc.env.internal.JdbcEnvironmentInitiator$ConnectionProviderJdbcConnectionAccess@41d2c01d] for (non-JTA) DDL execution was not in auto-commit mode; the Connection 'local transaction' will be committed and the Connection will be set into auto-commit mode.

Hibernate:

/\*

FROM

api.main.AlertSystem.Alert object

WHERE

object.alertId = : id \*/ select

a1\_0.alertId,

a1\_0.AlertMessage,

a1\_0.AlertTitle,

a1\_0.date

from

AlertModel a1\_0

where

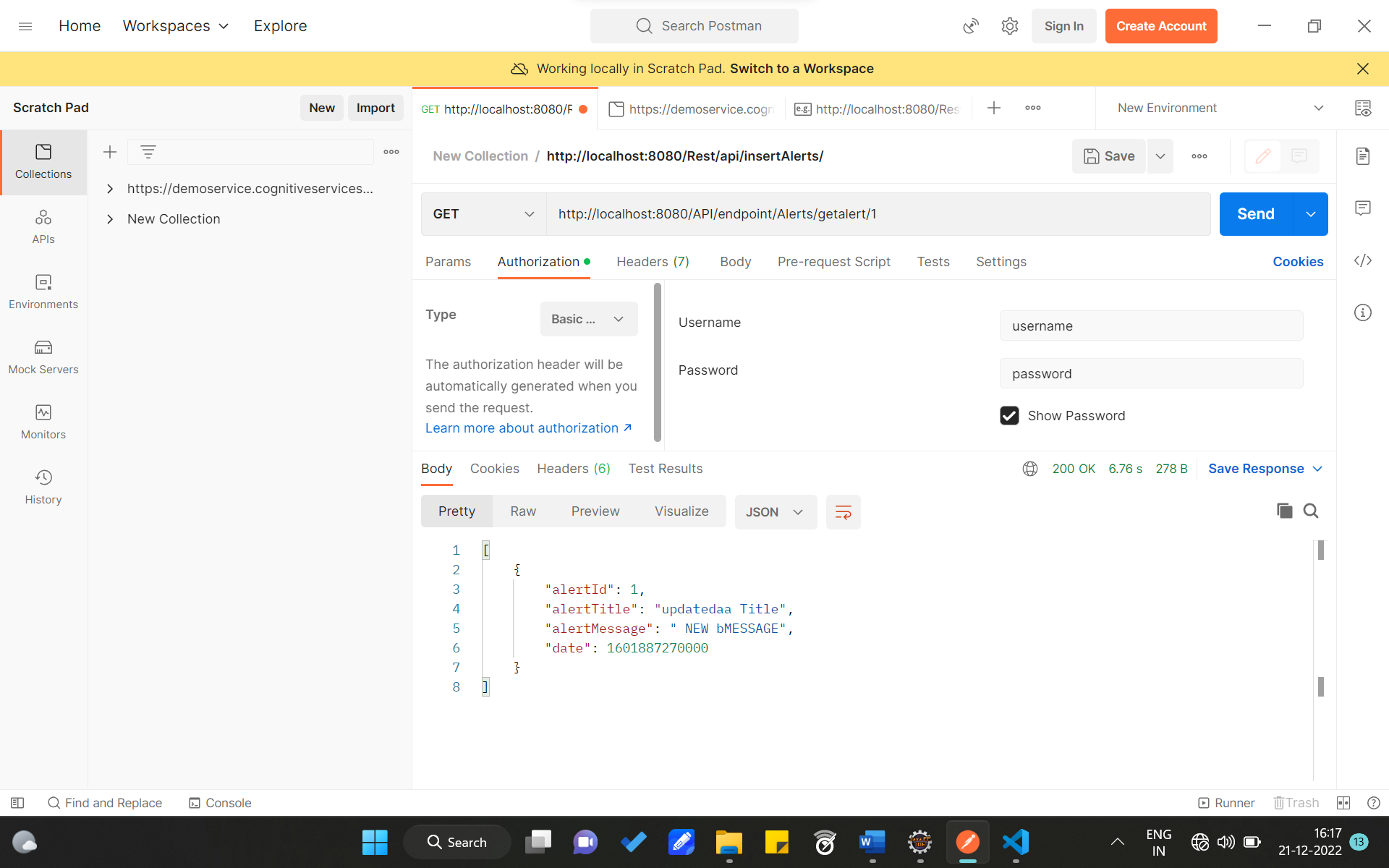
a1\_0.alertId=?

Dec 21, 2022 4:17:09 PM org.glassfish.jersey.filter.LoggingFilter log

INFO: 1 \* Server responded with a response on thread http-nio-8080-exec-3

1 < 200

1 < Content-Type: application/json

Output

**III Delete alert**

End Point http://localhost:8080/API/endpoint/Alerts/deletealerts/1

Method Delete

Produces 204 NO\_CONTENT

Session Management N/A – Cookies

Transmission output

Dec 21, 2022 4:27:23 PM org.glassfish.jersey.filter.LoggingFilter log

INFO: 1 \* Server has received a request on thread http-nio-8080-exec-1

1 > DELETE http://localhost:8080/API/endpoint/Alerts/deletealerts/1

1 > accept: \*/\*

1 > accept-encoding: gzip, deflate, br

1 > authorization: **Basic dXNlcm5hbWU6cGFzc3dvcmQ=**

1 > connection: keep-alive

1 > host: localhost:8080

1 > postman-token: 44379134-4428-4bbd-abe8-0a12550b4aad

1 > user-agent: PostmanRuntime/7.30.0

username

password

NFO: HHH10001501: Connection obtained from JdbcConnectionAccess [org.hibernate.engine.jdbc.env.internal.JdbcEnvironmentInitiator$ConnectionProviderJdbcConnectionAccess@53cb6147] for (non-JTA) DDL execution was not in auto-commit mode; the Connection 'local transaction' will be committed and the Connection will be set into auto-commit mode.

Hibernate:

/\* DELETE

FROM

api.main.AlertSystem.Alert object

WHERE

object.alertId= : id \*/ delete

from

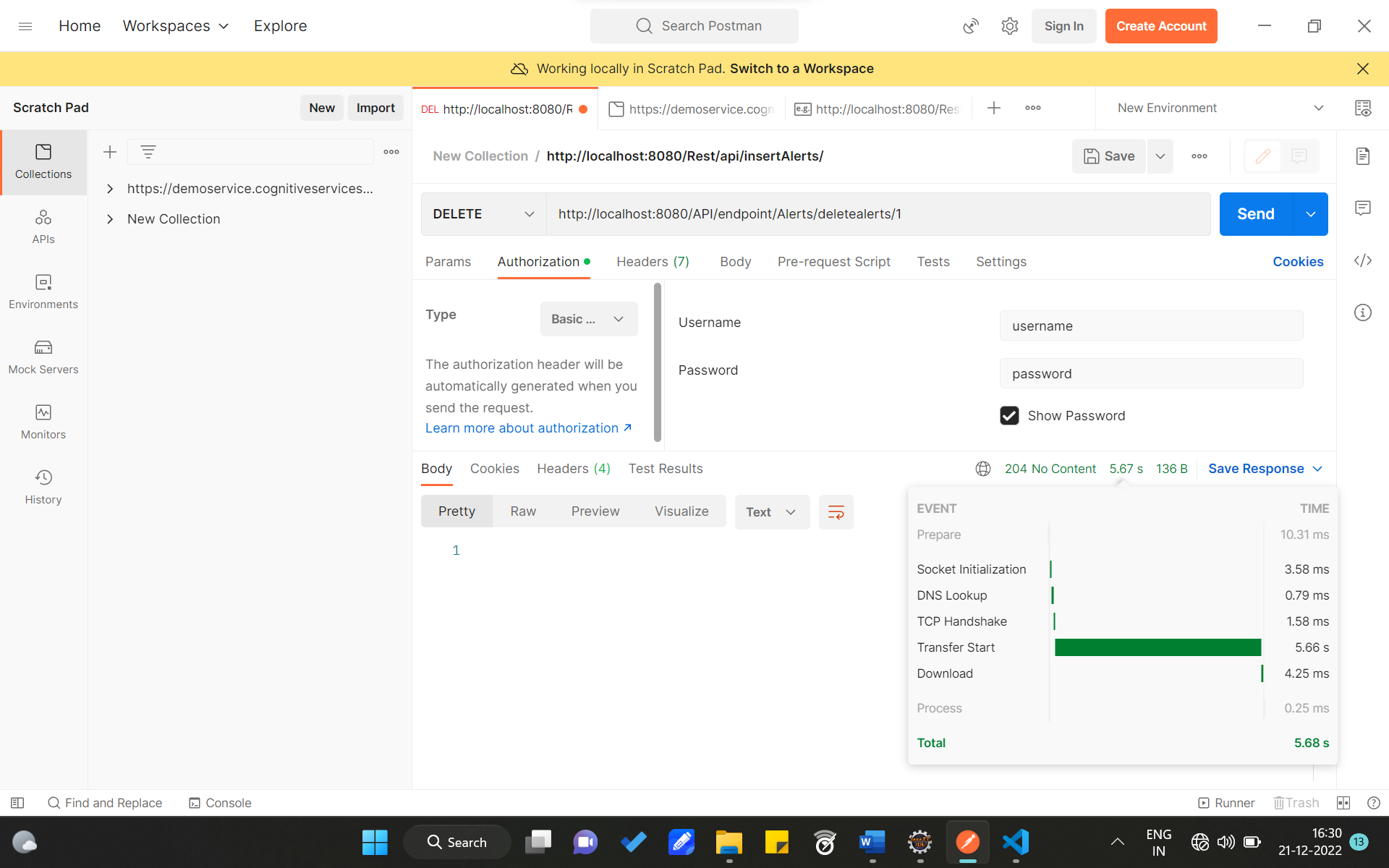
AlertModel

where

alertId=?

Dec 21, 2022 4:27:29 PM org.glassfish.jersey.filter.LoggingFilter log

INFO: 1 \* Server responded with a response on thread http-nio-8080-exec-1

1 < 204

**IV Update alerts**

**End point** <http://localhost:8080/API/endpoint/Alerts/updatealerts/2>

Method PUT

Produces 200 OK "NO\_CONTENT"

Consumes JSON

Session Management N/A – Cookies

Transmission output

Dec 21, 2022 5:37:14 PM org.glassfish.jersey.filter.LoggingFilter log

INFO: 1 \* Server responded with a response on thread http-nio-8080-exec-10

1 < 200

1 < Content-Type: application/json

Dec 21, 2022 5:37:54 PM org.glassfish.jersey.filter.LoggingFilter log

INFO: 2 \* Server has received a request on thread http-nio-8080-exec-12

2 > GET http://localhost:8080/API/endpoint/Alerts/updatealerts/2

2 > accept: \*/\*

2 > accept-encoding: gzip, deflate, br

2 > authorization: **Basic dXNlcm5hbWU6cGFzc3dvcmQ=**

2 > connection: keep-alive

2 > content-length: 133

2 > content-type: application/json

2 > host: localhost:8080

2 > postman-token: afcf1347-1864-4c7d-9bc8-f6145d7a9099

2 > user-agent: PostmanRuntime/7.30.0

Dec 21, 2022 5:37:54 PM org.glassfish.jersey.filter.LoggingFilter log

INFO: 2 \* Server responded with a response on thread http-nio-8080-exec-12

2 < 405

2 < Allow: OPTIONS,PUT

Dec 21, 2022 5:37:58 PM org.glassfish.jersey.filter.LoggingFilter log

INFO: 3 \* Server has received a request on thread http-nio-8080-exec-13

3 > PUT http://localhost:8080/API/endpoint/Alerts/updatealerts/2

3 > accept: \*/\*

3 > accept-encoding: gzip, deflate, br

3 > authorization: Basic dXNlcm5hbWU6cGFzc3dvcmQ=

3 > connection: keep-alive

3 > content-length: 133

3 > content-type: application/json

3 > host: localhost:8080

3 > postman-token: 220e9d04-b2b4-433d-9ffe-2101bbaf2129

3 > user-agent: PostmanRuntime/7.30.0

username

password

security filter called

Hibernate:

/\*

FROM

api.main.AlertSystem.Alert object

WHERE

object.alertId = : id \*/ select

a1\_0.alertId,

a1\_0.AlertMessage,

a1\_0.AlertTitle,

a1\_0.date

from

AlertModel a1\_0

where

a1\_0.alertId=?

alert msg and tile

Hibernate:

/\* UPDATE

api.main.AlertSystem.Alert object

SET

object.alertMessage= : msg ,

object.alertTitle= : title

where

object.alertId= : id \*/ update AlertModel

set

AlertMessage=?,

AlertTitle=?

where

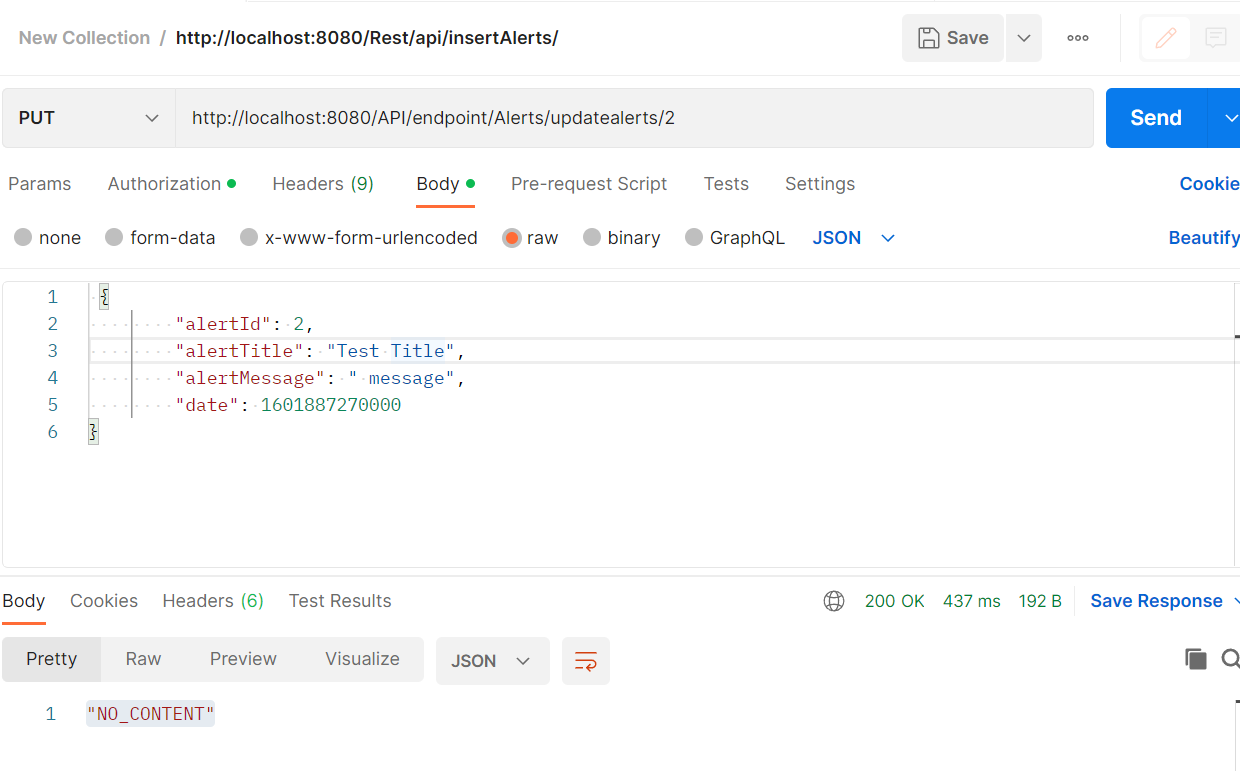
alertId=?

Dec 21, 2022 5:37:59 PM org.glassfish.jersey.filter.LoggingFilter log

INFO: 3 \* Server responded with a response on thread http-nio-8080-exec-13

3 < 200

3 < Content-Type: application/json



**V.INSERT ALERTS**

**End point** http://localhost:8080/API/endpoint/Alerts/insertAlerts

Method POST

Produces 201 OK "created”

Consumes JSON

Session Management N/A – Cookies

**Transmission output**

Dec 21, 2022 5:46:12 PM org.glassfish.jersey.filter.LoggingFilter log

INFO: 1 \* Server has received a request on thread http-nio-8080-exec-15

1 > POST http://localhost:8080/API/endpoint/Alerts/insertAlerts

1 > accept: \*/\*

1 > accept-encoding: gzip, deflate, br

1 > authorization: Basic dXNlcm5hbWU6cGFzc3dvcmQ=

1 > connection: keep-alive

1 > content-length: 148

1 > content-type: application/json

1 > host: localhost:8080

1 > postman-token: 20f5d266-065a-4534-a13e-3e0462453c20

1 > user-agent: PostmanRuntime/7.30.0

username

password

security filter called

Dec 21, 2022 5:46:13 PM org.hibernate.Version logVersion

INFO: HHH000412: Hibernate ORM core version 6.0.2.Final

Dec 21, 2022 5:46:14 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl configure

WARN: HHH10001002: Using built-in connection pool (not intended for production use)

Dec 21, 2022 5:46:14 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator

INFO: HHH10001005: Loaded JDBC driver class: org.postgresql.Driver

Dec 21, 2022 5:46:14 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator

INFO: HHH10001012: Connecting with JDBC URL [jdbc:postgresql://localhost:5432/WeConnect?createDatabaseIfNotExist=true&&autoReconnect=true]

Dec 21, 2022 5:46:14 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator

INFO: HHH10001001: Connection properties: {password=\*\*\*\*, user=postgres}

Dec 21, 2022 5:46:14 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator

INFO: HHH10001003: Autocommit mode: false

Dec 21, 2022 5:46:14 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl$PooledConnections <init>

INFO: HHH10001115: Connection pool size: 20 (min=1)

Dec 21, 2022 5:46:14 PM org.hibernate.engine.jdbc.dialect.internal.DialectFactoryImpl logSelectedDialect

INFO: HHH000400: Using dialect: org.hibernate.dialect.PostgreSQLDialect

Dec 21, 2022 5:46:16 PM org.hibernate.resource.transaction.backend.jdbc.internal.DdlTransactionIsolatorNonJtaImpl getIsolatedConnection

INFO: HHH10001501: Connection obtained from JdbcConnectionAccess [org.hibernate.engine.jdbc.env.internal.JdbcEnvironmentInitiator$ConnectionProviderJdbcConnectionAccess@58fa939] for (non-JTA) DDL execution was not in auto-commit mode; the Connection 'local transaction' will be committed and the Connection will be set into auto-commit mode.

Hibernate:

select

nextval('AlertModel\_SEQ')

Hibernate:

/\* insert api.main.AlertSystem.Alert

\*/ insert

into

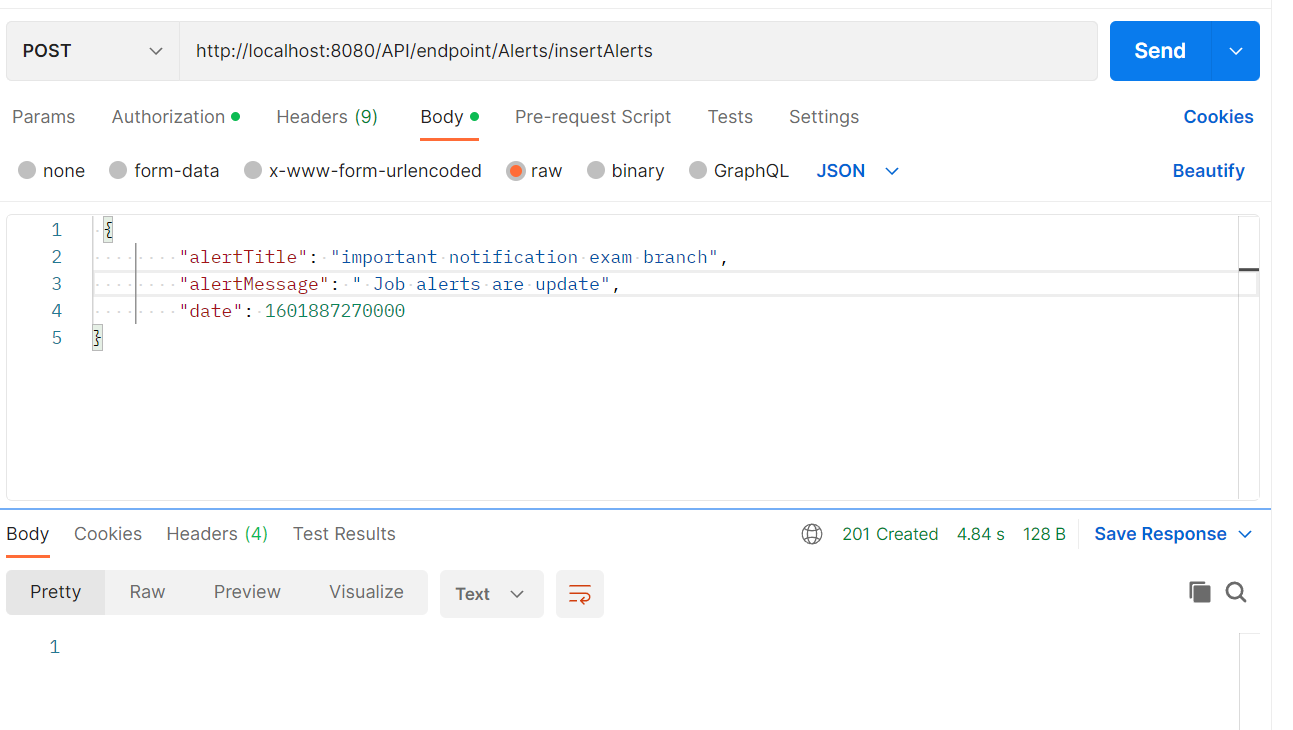
AlertModel (AlertMessage, AlertTitle, date, alertId)

values

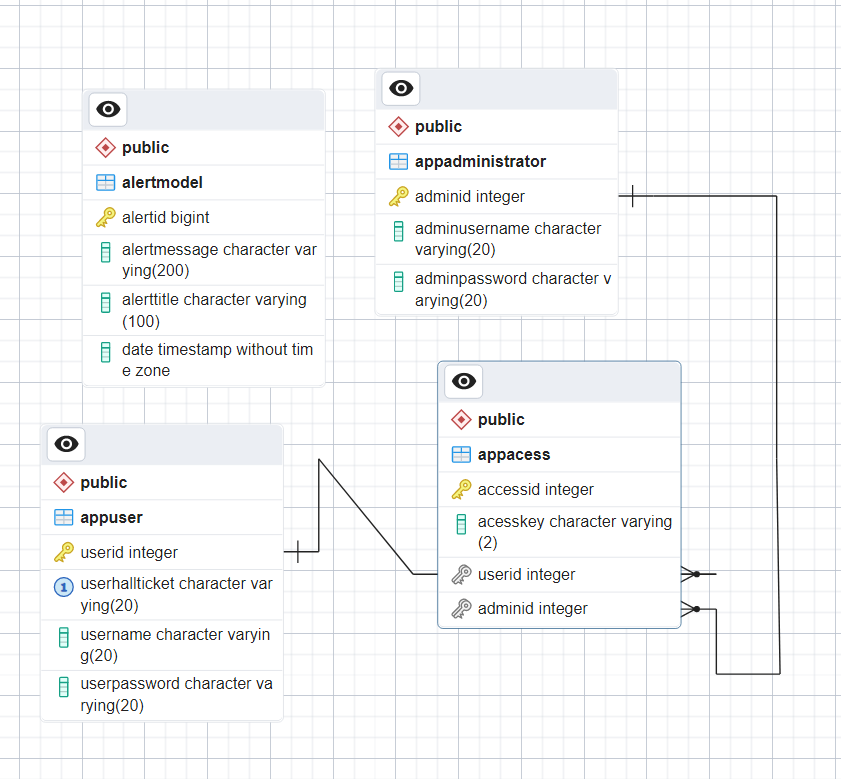
(?, ?, ?, ?)

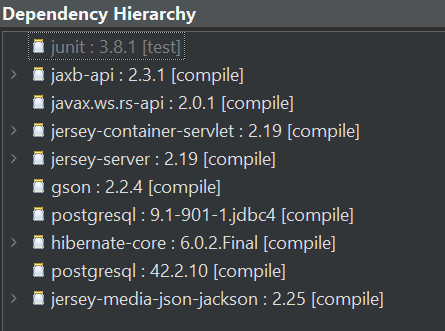
Dec 21, 2022 5:46:17 PM org.glassfish.jersey.filter.LoggingFilter log

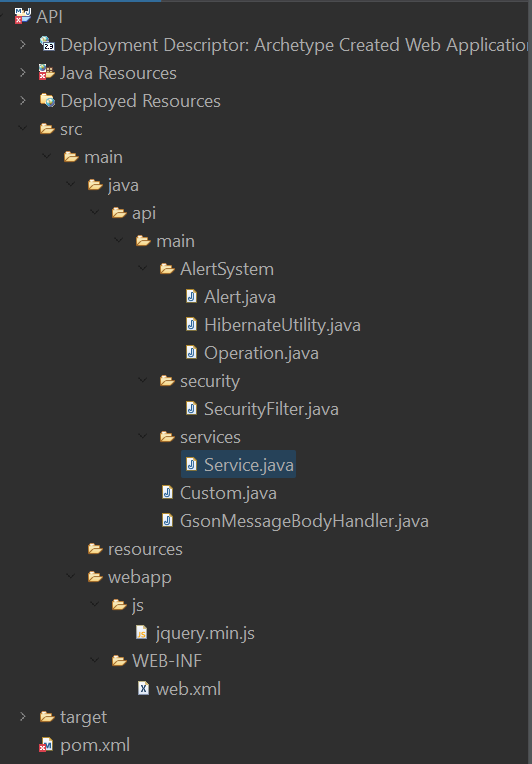
INFO: 1 \* Server responded with a response on thread http-nio-8080-exec-15

1 < 201

**DATABASE SCHEMA**



 **Dependency**

 **Project Structure**

**ORM (Object Relational Model)**

*@XmlRootElement*()

*@Entity*

*@Table*(name="AlertModel")

public class Alert implements Serializable{

private static final long ***serialVersionUID*** = 1L;

*@Id*

*@GeneratedValue*(strategy = *GenerationType*.***AUTO***)

long alertId;

*@Column*(name="AlertTitle",length=100,nullable=false)

String alertTitle;

*@Column* (name="AlertMessage",length=200,nullable=false)

String alertMessage;

*@Column*(name="date", columnDefinition="TIMESTAMP DEFAULT CURRENT\_TIMESTAMP",nullable=false)

private Date date;

public long getAlertId() {

return alertId;

}

public void setAlertId(long alertId) {

this.alertId = alertId;

}

public String getAlertTitle() {

return alertTitle;

}

public void setAlertTitle(String alertTitle) {

this.alertTitle = alertTitle;

}

public String getAlertMessage() {

return alertMessage;

}

public void setAlertMessage(String alertMessage) {

this.alertMessage = alertMessage;

}

public Date getDate() {

return date;

}

public void setDate(Date date) {

this.date = date;

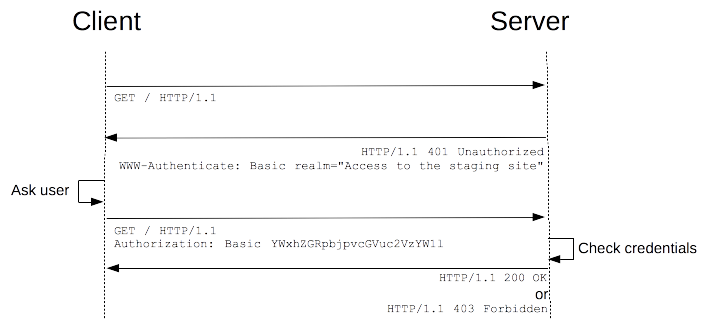
}

public static long getSerialversionuid() {

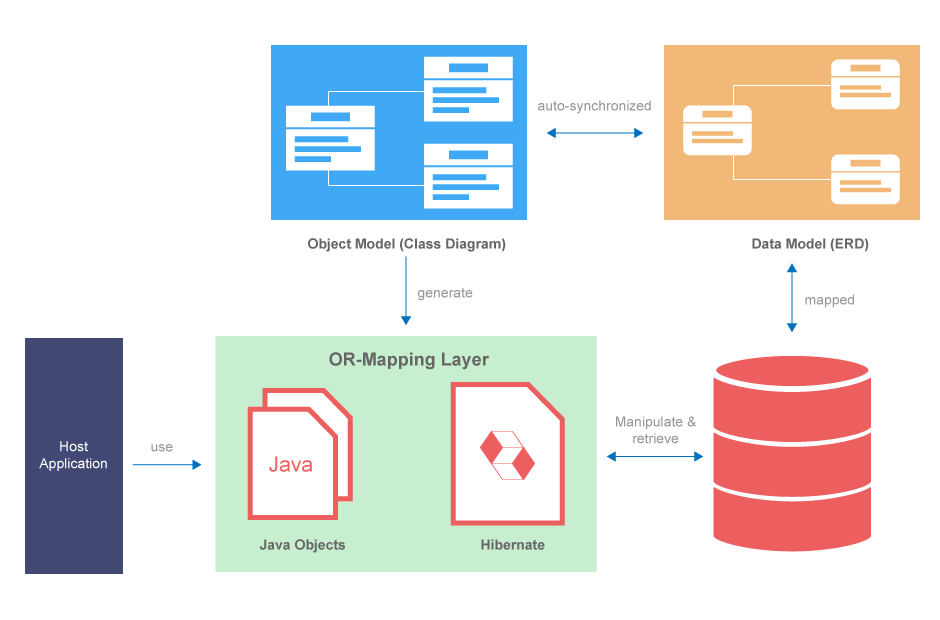
return ***serialVersionUID***;

}

}

 **UML**

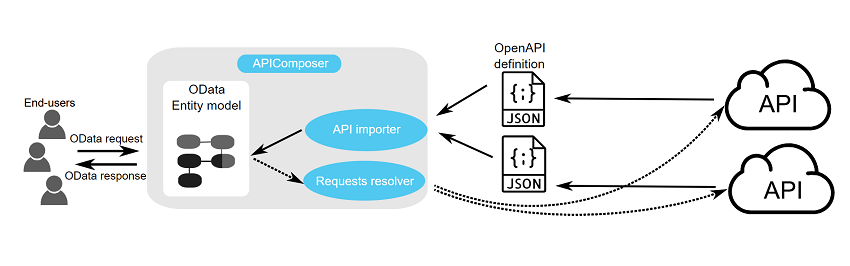
**Basic authorization**



**Hibernate**



**Model View controller**



**Representational state behaviour API**

**Project source Code**

Root\POM.XML

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4\_0\_0.xsd">

  <modelVersion>4.0.0</modelVersion>

  <groupId>WeConnect</groupId>

  <artifactId>API</artifactId>

  <packaging>war</packaging>

  <version>0.0.1-SNAPSHOT</version>

  <name>API Maven Webapp</name>

  <url>http://maven.apache.org</url>

  <repositories>

        <repository>

            <id>maven2-repository.java.net</id>

            <name>Java.net Repository for Maven</name>

            <url>http://download.java.net/maven/2/</url>

            <layout>default</layout>

        </repository>

    </repositories>

    <properties>

        <jersey2.version>2.19</jersey2.version>

        <jaxrs.version>2.0.1</jaxrs.version>

    </properties>

  <dependencies>

    <dependency>

      <groupId>junit</groupId>

      <artifactId>junit</artifactId>

      <version>3.8.1</version>

      <scope>test</scope>

    </dependency>

    <dependency>

    <groupId>javax.xml.bind</groupId>

    <artifactId>jaxb-api</artifactId>

    <version>2.3.1</version>

</dependency>

    <!-- JAX-RS -->

        <dependency>

            <groupId>javax.ws.rs</groupId>

            <artifactId>javax.ws.rs-api</artifactId>

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

        </dependency>

        <dependency>

            <groupId>org.glassfish.jersey.containers</groupId>

            <artifactId>jersey-container-servlet</artifactId>

            <version>${jersey2.version}</version>

        </dependency>

        <dependency>

            <groupId>org.glassfish.jersey.core</groupId>

            <artifactId>jersey-server</artifactId>

            <version>${jersey2.version}</version>

        </dependency>

        <dependency>

            <groupId>com.google.code.gson</groupId>

            <artifactId>gson</artifactId>

            <version>2.2.4</version>

        </dependency>

        <dependency>

            <groupId>postgresql</groupId>

            <artifactId>postgresql</artifactId>

            <version>9.1-901-1.jdbc4</version>

        </dependency>

        <dependency>

            <groupId>org.hibernate.orm</groupId>

            <artifactId>hibernate-core</artifactId>

            <version>6.0.2.Final</version>

        </dependency>

        <dependency>

            <groupId>org.postgresql</groupId>

            <artifactId>postgresql</artifactId>

            <version>42.2.10</version>

        </dependency>

        <dependency>

            <groupId>org.glassfish.jersey.media</groupId>

            <artifactId>jersey-media-json-jackson</artifactId>

            <version>2.25</version>

        </dependency>

  </dependencies>

  <build>

    <finalName>API</finalName>

    <plugins>

            <plugin>

                <artifactId>maven-compiler-plugin</artifactId>

                <configuration>

                    <source>1.7</source>

                    <target>1.7</target>

                </configuration>

            </plugin>

        </plugins>

  </build>

</project>

Root\.classpath

<?xml version="1.0" encoding="UTF-8"?>

<classpath>

    <classpathentry kind="src" output="target/classes" path="src/main/java">

        <attributes>

            <attribute name="optional" value="true"/>

            <attribute name="maven.pomderived" value="true"/>

        </attributes>

    </classpathentry>

    <classpathentry excluding="\*\*" kind="src" output="target/classes" path="src/main/resources">

        <attributes>

            <attribute name="maven.pomderived" value="true"/>

        </attributes>

    </classpathentry>

    <classpathentry kind="src" output="target/test-classes" path="src/test/java">

        <attributes>

            <attribute name="optional" value="true"/>

            <attribute name="maven.pomderived" value="true"/>

            <attribute name="test" value="true"/>

        </attributes>

    </classpathentry>

    <classpathentry kind="con" path="org.eclipse.jdt.launching.JRE\_CONTAINER/org.eclipse.jdt.internal.debug.ui.launcher.StandardVMType/JavaSE-1.7">

        <attributes>

            <attribute name="maven.pomderived" value="true"/>

        </attributes>

    </classpathentry>

    <classpathentry kind="con" path="org.eclipse.m2e.MAVEN2\_CLASSPATH\_CONTAINER">

        <attributes>

            <attribute name="maven.pomderived" value="true"/>

        </attributes>

    </classpathentry>

    <classpathentry kind="src" path="target/generated-sources/annotations">

        <attributes>

            <attribute name="optional" value="true"/>

            <attribute name="maven.pomderived" value="true"/>

            <attribute name="ignore\_optional\_problems" value="true"/>

            <attribute name="m2e-apt" value="true"/>

        </attributes>

    </classpathentry>

    <classpathentry kind="src" output="target/test-classes" path="target/generated-test-sources/test-annotations">

        <attributes>

            <attribute name="optional" value="true"/>

            <attribute name="maven.pomderived" value="true"/>

            <attribute name="ignore\_optional\_problems" value="true"/>

            <attribute name="m2e-apt" value="true"/>

            <attribute name="test" value="true"/>

        </attributes>

    </classpathentry>

    <classpathentry kind="output" path="target/classes"/>

</classpath>

Root\src\main\webapp\WEB-INF\web.xml

<!DOCTYPE web-app PUBLIC

 "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"

 "http://java.sun.com/dtd/web-app\_2\_3.dtd" >

<web-app>

  <display-name>Archetype Created Web Application</display-name>

  <servlet>

    <servlet-name>Jersey REST Service</servlet-name>

    <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-class>

    <init-param>

      <param-name>javax.ws.rs.Application</param-name>

      <param-value>api.main.Custom</param-value>

    </init-param>

    <load-on-startup>1</load-on-startup>

  </servlet>

  <servlet-mapping>

    <servlet-name>Jersey REST Service</servlet-name>

    <url-pattern>/\*</url-pattern>

  </servlet-mapping>

  <security-constraint>

    <display-name>Secure REST Area</display-name>

    <web-resource-collection>

      <web-resource-name>Secure REST</web-resource-name>

      <url-pattern>/\*</url-pattern>

      <http-method>PUT</http-method>

      <http-method>GET</http-method>

      <http-method>POST</http-method>

      <http-method>DELETE</http-method>

    </web-resource-collection>

  </security-constraint>

  <login-config>

    <auth-method>BASIC</auth-method>

    <realm-name>default</realm-name>

  </login-config>

</web-app>

Root\ src\main\java\api\main\AlertSystem\Alert.java

package api.main.AlertSystem;

import java.io.Serializable;

import java.util.Date;

import javax.xml.bind.annotation.XmlRootElement;

import jakarta.persistence.Column;

import jakarta.persistence.Entity;

import jakarta.persistence.GeneratedValue;

import jakarta.persistence.GenerationType;

import jakarta.persistence.Id;

import jakarta.persistence.Table;

@XmlRootElement()

@Entity

@Table(name="AlertModel")

public class Alert  implements Serializable{

    private static final long serialVersionUID = 1L;

    @Id

    @GeneratedValue(strategy = GenerationType.AUTO)

    long alertId;

    @Column(name="AlertTitle",length=100,nullable=false)

    String alertTitle;

    @Column (name="AlertMessage",length=200,nullable=false)

    String alertMessage;

    @Column(name="date", columnDefinition="TIMESTAMP DEFAULT CURRENT\_TIMESTAMP",nullable=false)

    private Date date;

    public  long getAlertId() {

        return alertId;

    }

    public void setAlertId(long alertId) {

        this.alertId = alertId;

    }

    public String getAlertTitle() {

        return alertTitle;

    }

    public void setAlertTitle(String alertTitle) {

        this.alertTitle = alertTitle;

    }

    public String getAlertMessage() {

        return alertMessage;

    }

    public void setAlertMessage(String alertMessage) {

        this.alertMessage = alertMessage;

    }

    public Date getDate() {

        return date;

    }

    public void setDate(Date date) {

        this.date = date;

    }

    public static long getSerialversionuid() {

        return serialVersionUID;

    }

}

Root\ src\main\java\api\main\AlertSystem\HibernateUtility.java

package api.main.AlertSystem;

import org.hibernate.boot.registry.StandardServiceRegistryBuilder;

import org.hibernate.SessionFactory;

import java.util.Properties;

import org.hibernate.service.ServiceRegistry;

import org.hibernate.cfg.Configuration;

import org.hibernate.cfg.Environment;

public class HibernateUtility {

    public static SessionFactory sessionfactory;

    public static SessionFactory getsessionfactory() {

        if(sessionfactory == null) {

            try {

                Configuration configuration= new Configuration();

                Properties settings = new Properties();

                settings.put(Environment.DRIVER, "org.postgresql.Driver");//?autoReconnect=true

                settings.put(Environment.URL, "jdbc:postgresql://localhost:5432/WeConnect?createDatabaseIfNotExist=true&&autoReconnect=true");

                settings.put(Environment.USER, "postgres");

                settings.put(Environment.PASS, "root");

                settings.put(Environment.DIALECT, "org.hibernate.dialect.PostgreSQLDialect");

                settings.put(Environment.SHOW\_SQL, "true");

                settings.put(Environment.FORMAT\_SQL,true);

                settings.put(Environment.USE\_SQL\_COMMENTS,true);

                settings.put(Environment.CURRENT\_SESSION\_CONTEXT\_CLASS, "thread");

                settings.put(Environment.HBM2DDL\_AUTO, "validate");

                //settings.put(Environment.HBM2DDL\_AUTO, "create");

                configuration.setProperties(settings);

                configuration.addAnnotatedClass(Alert.class);

                ServiceRegistry serviceregistry = new StandardServiceRegistryBuilder().applySettings(configuration.getProperties()).build();

                sessionfactory = configuration.buildSessionFactory(serviceregistry);

            }catch(Exception e) {

                e.printStackTrace();

            }

        }

        return sessionfactory;

    }

    }

Root\ src\main\java\api\main\AlertSystem\Operation.java

package api.main.AlertSystem;

import java.util.List;

import org.hibernate.query.Query;

import org.hibernate.Session;

import org.hibernate.Transaction;

public class Operation {

    public List<Alert> getAlerts(){

        Transaction transaction = null;

        Session session = null;

        List<Alert> data = null;

        try {

            System.out.println("GET ALETS FUNCTION INVOKED");

            session = HibernateUtility.getsessionfactory().openSession();

            transaction = session.beginTransaction();

            Query<Alert> query = session.createQuery("from Alert",Alert.class);

            data = query.getResultList();

            System.out.println("FINAL executed");

            }catch(Exception ex) {

                System.out.println(ex.toString());

                System.out.println("ERROR");

                if(transaction != null) {

                    transaction.rollback();

                }

        }finally {

            if(session!=null)session.close();

        }

        return data;

    }

    public List<Alert> getAlert(long id){

        Transaction transaction = null;

        Session session = null;

        List<Alert> data = null;

        try{

            session  = HibernateUtility.getsessionfactory().openSession();

            transaction = session.beginTransaction();

            Query<Alert> query = session.createQuery("FROM api.main.AlertSystem.Alert object WHERE object.alertId = : id",Alert.class);

            query.setParameter("id",id);

            data =  query.list();

            session.flush();

        }catch(Exception ex) {

            if(transaction!= null) transaction.rollback();

            ex.printStackTrace();

        }finally{

             if(session!=null)

                 session.close();

          }

        return data;

    }

    public boolean saveAlert(Alert model){

        Transaction transaction = null;

        Session session = null;

        try{

            session = HibernateUtility.getsessionfactory().openSession();

            transaction = session.beginTransaction();

            session.persist(model);

            transaction.commit();

            return true;

        }catch(Exception ex){

            if(transaction!= null)transaction.rollback();

                System.out.println(ex.toString());

        return false;

    }

     finally

     {

         if(session!=null) session.close();

     }

}

    public Boolean updateAlert(Alert alert){

        Session session = null;

        try {

            session = HibernateUtility.getsessionfactory().openSession();

            @SuppressWarnings ("deprecation" )

            Query<?> query = session.createQuery("UPDATE api.main.AlertSystem.Alert object SET object.alertMessage= : msg ,object.alertTitle= : title where object.alertId= : id");

            query.setParameter("msg", alert.getAlertMessage());

            query.setParameter("title",alert.getAlertTitle());

            query.setParameter("id",alert.getAlertId());

            session.beginTransaction();

            int res = query.executeUpdate();

            session.flush();

            if(res>0)return true;

        }catch(Exception ex) {

            System.out.println(ex.toString());

            return false;

        }finally {

             if(session!=null)

                 session.close();

        }

        return false;

    }

    public Boolean DeleteAlert(long id) {

        //Transaction transaction = null;

        Session session = null;

        try {

            session  = HibernateUtility.getsessionfactory().openSession();

            //transaction = session.beginTransaction();

            @SuppressWarnings("deprecation")

            Query<?> query = session.createQuery("DELETE FROM api.main.AlertSystem.Alert object WHERE object.alertId= : id");

            query.setParameter("id",id);

            session.beginTransaction();

            int result =query.executeUpdate();

            //session.getTransaction().commit();

            //transaction.commit();

            session.flush();

            if(result>0) return true;

        }catch(Exception ex){

            ex.printStackTrace();

            return false;

        }finally {

             if(session!=null)

                 session.close();

        }

        return false;

        }

}

Root\ src\main\java\api\main\security\SecurityFilter.java

package api.main.security;

//import java.io.IOException;

import java.lang.reflect.Method;

import java.util.Arrays;

import java.util.HashSet;

import java.util.List;

import java.util.Set;

import java.util.StringTokenizer;

import javax.annotation.security.DenyAll;

import javax.annotation.security.PermitAll;

import javax.annotation.security.RolesAllowed;

//import javax.ws.rs.WebApplicationException;

import javax.ws.rs.container.ContainerRequestContext;

import javax.ws.rs.container.ContainerRequestFilter;

import javax.ws.rs.container.ResourceInfo;

import javax.ws.rs.core.Context;

import javax.ws.rs.core.MultivaluedMap;

import javax.ws.rs.core.Response;

import javax.ws.rs.ext.Provider;

//import org.glassfish.jersey.filter.LoggingFilter;

import org.glassfish.jersey.internal.util.Base64;

//import org.glassfish.jersey.server.ResourceConfig;

@Provider

public class SecurityFilter implements ContainerRequestFilter{

    @Context

    private ResourceInfo resourceInfo;

       private static final String AUTHORIZATION\_PROPERTY = "Authorization";

        private static final String AUTHENTICATION\_SCHEME = "Basic";

     private static final Response ACCESS\_DENIED = Response.status(Response.Status.UNAUTHORIZED)

                .entity("Invalid").build();

private static final Response ACCESS\_FORBIDDEN = Response.status(Response.Status.FORBIDDEN)

                .entity("Access blocked for all users !!").build();

    @Override

    public void filter(ContainerRequestContext requestContext) {

        try {

        Method method = resourceInfo.getResourceMethod();

         //Access allowed for all

        if( ! method.isAnnotationPresent(PermitAll.class))

        {

            //Access denied for all

            if(method.isAnnotationPresent(DenyAll.class))

            {

                requestContext.abortWith(ACCESS\_FORBIDDEN);

                return;

            }

            //Get request headers

            final MultivaluedMap<String, String> headers = requestContext.getHeaders();

            //Fetch authorization header

            final List<String> authorization = headers.get(AUTHORIZATION\_PROPERTY);

            //If no authorization information present; block access

            if(authorization == null || authorization.isEmpty())

            {

                requestContext.abortWith(ACCESS\_DENIED);

                return;

            }

            //Get encoded user name and password

            final String encodedUserPassword = authorization.get(0).replaceFirst(AUTHENTICATION\_SCHEME + " ", "");

            //Decode user name and password

            String usernameAndPassword = new String(Base64.decode(encodedUserPassword.getBytes()));;

            //Split user name and password tokens

            final StringTokenizer tokenizer = new StringTokenizer(usernameAndPassword, ":");

            final String username = tokenizer.nextToken();

            final String password = tokenizer.nextToken();

            //Verifying User name and password

            System.out.println(username);

            System.out.println(password);

            System.out.println("security filter called");

            //Verify user access

            if(method.isAnnotationPresent(RolesAllowed.class))

            {

                RolesAllowed rolesAnnotation = method.getAnnotation(RolesAllowed.class);

                Set<String> rolesSet = new HashSet<String>(Arrays.asList(rolesAnnotation.value()));

                //Is user valid?

                if( ! isUserAllowed(username, password, rolesSet))

                {

                    requestContext.abortWith(ACCESS\_DENIED);

                    return;

                }

            }

        }//end of if block

        }//end of try block

        catch(Exception ex) {

            //throws IOException, WebApplicationException

            System.out.println(ex);

            //requestContext.setRequestUri(

            //return;

        }

    }

    private boolean isUserAllowed(String username, String password, Set<String> rolesSet) {

        // TODO Auto-generated method stub

        boolean isAllowed = false;

        //Step 1. Fetch password from database and match with password in argument

        //If both match then get the defined role for user from database and continue; else return isAllowed [false]

        //Access the database and do this part yourself

        //String userRole = userMgr.getUserRole(user name);

        if(username.equals("username") && password.equals("password"))

        {

            String userRole = "ADMIN";

            //Step 2. Verify user role

            if(rolesSet.contains(userRole))

            {

                isAllowed = true;

            }

        }

        return isAllowed;

    }

}

**Path**  Root\src\main\java\api\main\services\Service.java

package api.main.services;

import java.util.List;

import javax.annotation.security.RolesAllowed;

import javax.ws.rs.Consumes;

import javax.ws.rs.DELETE;

import javax.ws.rs.GET;

import javax.ws.rs.POST;

import javax.ws.rs.PUT;

import javax.ws.rs.Path;

import javax.ws.rs.PathParam;

import javax.ws.rs.Produces;

import javax.ws.rs.core.Application;

import javax.ws.rs.core.MediaType;

import javax.ws.rs.core.Response;

import api.main.AlertSystem.\*;

@Path("/endpoint")

public class Service extends Application{

    @RolesAllowed("ADMIN")

    @Path("test")

    @GET

    @Produces(MediaType.TEXT\_PLAIN)

    public String testing() {

        return "hello";

    }

    @RolesAllowed("ADMIN")

    @GET

    @Path("/Alerts/getallalerts")

    @Produces(MediaType.APPLICATION\_JSON)

    public Response getAllAlerts(){

           List < Alert > data=new Operation().getAlerts();

            if (!data.isEmpty()) {

                return Response.ok(data).build();

            } else {

                return Response.status(Response.Status.NOT\_FOUND).build();

            }

    }//end of get all alerts

    @GET

    @Path("/Alerts/getalert/{id}")

    //@Produces(MediaType.APPLICATION\_XML)

    @Produces(MediaType.APPLICATION\_JSON)

    public List<Alert> getAlert(@PathParam("id") long id){

    //  return Response.ok(new Repository().getAlert(id)).build();

    List<Alert> data=   new Operation().getAlert(id);

    if(!data.isEmpty())

        return data;

            return null;

    }

    @DELETE

    @Path("/Alerts/deletealerts/{id}")

    @Produces(MediaType.APPLICATION\_JSON)

    public Response DeleteAlert(@PathParam("id") long id) {

        if(new Operation().DeleteAlert(id))

                    return Response.ok().status(Response.Status.NO\_CONTENT).build();

                    return Response.notModified().build();

        }

    @PUT

    @Path("/Alerts/updatealerts/{id}")

    @Produces(MediaType.APPLICATION\_JSON)

    @Consumes(MediaType.APPLICATION\_JSON)

    public Response UpdateAlerts(@PathParam("id") long id,Alert alerts){

        boolean res = false;

        if(id >= 0){

            List<Alert> model = new Operation().getAlert(id);

        if(!model.isEmpty())

            if(alerts.getAlertMessage()!=null && alerts.getAlertTitle()==null) {

                System.out.println("alert msg");

                model.get(0).setAlertMessage(alerts.getAlertMessage());

                res = (new Operation().updateAlert(model.get(0)))?true:false;

            }

            else if(alerts.getAlertTitle() != null &&alerts.getAlertMessage()==null ) {

                System.out.println("alert title");

                model.get(0).setAlertTitle(alerts.getAlertTitle());

                res = (new Operation().updateAlert(model.get(0)))?true:false;

            }else if(alerts.getAlertMessage()!= null && alerts.getAlertTitle()!=null) {

                System.out.println("alert msg and tile");

                model.get(0).setAlertMessage(alerts.getAlertMessage());

                model.get(0).setAlertTitle(alerts.getAlertTitle());

                res = (new Operation().updateAlert(model.get(0)))?true:false;

            }

        }

        if(res)

            return Response.status(200).entity(Response.Status.NO\_CONTENT).build();

            return Response.notModified().build();

    }

    @POST

    @Produces(MediaType.APPLICATION\_JSON)

    @Path("/Alerts/insertAlerts")

    //@Produces(MediaType.TEXT\_PLAIN)

    @Consumes(MediaType.APPLICATION\_JSON)

    public Response insertAlert(Alert model){

        //boolean res =new Repository().updateAlert(model);

         if (new Operation().saveAlert(model)) {

                return Response.ok().status(Response.Status.CREATED).build();

            } else {

                return Response.notModified().build();

            }

    }

}

Path : Root\src\main\java\api\main\Custom.java

package api.main;

import org.glassfish.jersey.filter.LoggingFilter;

import org.glassfish.jersey.server.ResourceConfig;

import api.main.security.\*;

public class Custom extends ResourceConfig

{

    public Custom()

    {

        packages("api.main");

        register(LoggingFilter.class);

        register(GsonMessageBodyHandler.class);

        register(SecurityFilter.class);

    }

}

GIT Repository <https://github.com/itsrinuhere/WeConnect.git>

For any Queries contact @itsrinuhere1@gmail.com