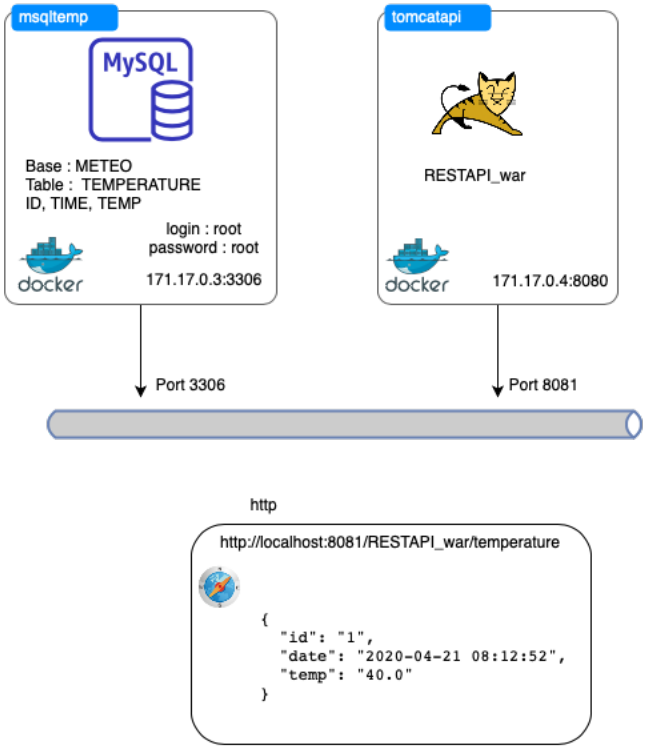
Codage d'une API REST Java



Route	Méthode	Requête
RESTAPI_war/temperature	GET	curl http://localhost:8081/RESTAPI_war/temperature
RESTAPI_war/temperature/{id}	GET	curl http://localhost:8081/RESTAPI_war/temperature/{id}
RESTAPI_war/temperature	POST	curl -d "TEMP=33" -X POST http://localhost:8081/RESTAPI_war/temperature/
RESTAPI_war/temperature/{id}	PUT	curl -X PUT -H "Content-Type : application/x-www-form-urlencoded" -d {"temp":"40} http://localhost:8081/RESTAPI_war/temperature/{id}
RESTAPI_war/temperature/{id}	DELETE	curl -X DELETE http://localhost:8081/RESTAPI_war/temperature/{id}

Docker-compose.yml :

```yaml
version: "3"
services:
 db:
   image: mysql:latest
   environment:
     MYSQL_ROOT_PASSWORD: root
     MYSQL_DATABASE: METEO
     MYSQL_USER: admin
     MYSQL_PASSWORD: root
   ports:
     - "3306:3306"
   volumes:
    - /Users/samuelbouhenic/dumps/Dump20200424:/docker-entrypoint-initdb.d
   networks:
```

```
            testing_net:
                ipv4_address: 171.17.0.3

 web:
   image: tomcat:9.0
   # Environment variables do not appear to be getting loaded the first time Tomcat
starts!
   #environment:
     #JDBC_URL:
jdbc:mysql://db:3306/example_db?connectTimeout=0&amp;socketTimeout=0&amp;autoReconn
ect=true
     #JDBC_USER: example_db_user
     #JDBC_PASS: example_db_pass
   ports:
    - "8081:8080"
   volumes:
    -
/Users/samuelbouhenic/IdeaProjects/RESTAPI/out/artifacts/RESTAPI_war2/RESTAPI_war.w
ar:/usr/local/tomcat/webapps/RESTAPI_war.war
  #links:
    #- db
   networks:
            testing_net:
                ipv4_address: 171.17.0.4
networks:
    testing_net:
        ipam:
            driver: default
            config:
                - subnet: 171.17.0.0/16
```

Structure de l'API :

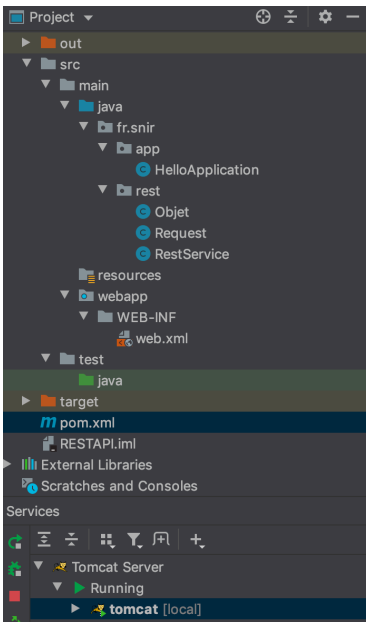
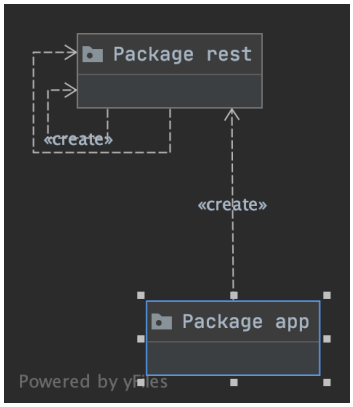
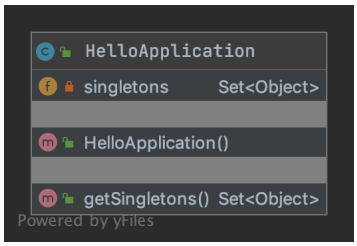
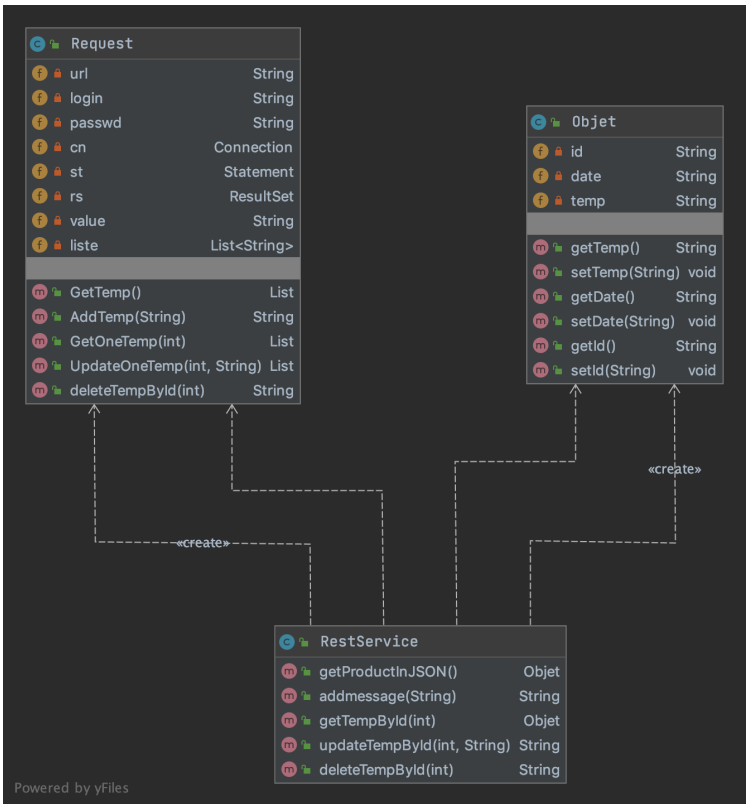


Diagramme de classe :







Fichier pom.xml :

```
<?xml version="1.0" encoding="UTF-8"?>
<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/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>fr.snir</groupId>
    <artifactId>RESTAPI</artifactId>
    <version>1.0-SNAPSHOT</version>
    <repositories>
        <repository>
            <id>JBoss repository</id>
            <url>https://repository.jboss.org/nexus/content/groups/public-jboss/</url>
        </repository>
    </repositories>
    <dependencies>
        <!--resteasy-->
        <dependency>
            <groupId>org.jboss.resteasy</groupId>
            <artifactId>resteasy-jaxrs</artifactId>
            <version>3.1.4.Final</version>
        </dependency>
        <!--rest-easy client-->
        <dependency>
            <groupId>org.jboss.resteasy</groupId>
            <artifactId>resteasy-client</artifactId>
            <version>3.1.4.Final</version>
        </dependency>
        <!--unit test-->
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.8.2</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.jboss.resteasy</groupId>
            <artifactId>resteasy-jackson-provider</artifactId>
            <version>3.11.2.Final</version>
        </dependency>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>8.0.19</version>
        </dependency>
    </dependencies>
    <packaging>war</packaging>
</project>
```

Fichier web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app id="WebApp_ID" version="2.4"
         xmlns="http://java.sun.com/xml/ns/j2ee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
    http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
    <display-name>Restful Web Application</display-name>

    <context-param>
        <param-name>resteasy.servlet.mapping.prefix</param-name>
        <!--Prefix for the endpoints-->
        <param-value>/</param-value>
    </context-param>

    <servlet>
        <servlet-name>resteasy-servlet</servlet-name>
        <servlet-class>
            org.jboss.resteasy.plugins.server.servlet.HttpServletDispatcher
        </servlet-class>
        <init-param>
```

```
            <param-name>javax.ws.rs.Application</param-name>
            <param-value>fr.snir.app.HelloApplication</param-value>
        </init-param>
    </servlet>

    <servlet-mapping>
        <servlet-name>resteasy-servlet</servlet-name>
        <!--Prefix for endpoint-->
        <url-pattern>/*</url-pattern>
    </servlet-mapping>

</web-app>
```

Classe HelloApplication :

```
package fr.snir.app;

import fr.snir.rest.RestService;
import javax.ws.rs.core.Application;
import java.util.HashSet;
import java.util.Set;
public class HelloApplication extends Application {
    private Set<Object> singletons = new HashSet<Object>();
    public HelloApplication() {
        // Register our temp service
        singletons.add(new RestService());
    }
    @Override
    public Set<Object> getSingletons() {
        return singletons;
    }
}
```

Classe RestService :

```
package fr.snir.rest;

import org.codehaus.jackson.JsonNode;
import org.codehaus.jackson.map.ObjectMapper;

import javax.ws.rs.*;
import java.io.IOException;

//Création de l'API REST en utilisant la dépendance RESTEASY

@Path("/")
public class RestService {
    @GET // Requête GET
    @Path("/temperature")
    @Produces("application/json")
    public Objet getProductInJSON() {
        Objet objetJava =new Objet(); //création d'un objet java
        Request requete = new Request(); //création d'une requête sql
        int nb = requete.GetTemp().size(); //recherche du nombre total d'enregistrement
        objetJava.setDate(String.valueOf(requete.GetTemp().get(nb-3))); //Ecriture du timestamp dans l'objet créé
        objetJava.setTemp(String.valueOf(requete.GetTemp().get(nb-2)));//Ecriture de la température dans l'objet créé
        objetJava.setId(String.valueOf(requete.GetTemp().get(nb-1)));//Ecriture de l'id dans l'objet créé
        return objetJava; //retour de l'objet java en objet json
    }

    @POST // Requête POST
    @Path("/temperature")
    @Produces("application/x-www-form-urlencoded")
    @Consumes("application/x-www-form-urlencoded")
    public String addmessage(@FormParam("TEMP") String temp){
        Request requete = new Request();//création d'une requête sql
        requete.AddTemp(temp);//Lancement de la méthode d'ajout
        return "temp="+temp+" ajoutée"; //création du message de retour
    }

    @GET // Requête GET sur un enregistrement
    @Path("/temperature/{id}")
```

```
    @Produces("application/json")
    public Objet getTempById(@PathParam("id") int id) {

        Objet objetJava =new Objet();
        Request requete = new Request();
        int nb = requete.GetOneTemp(id).size();
        objetJava.setDate(String.valueOf(requete.GetOneTemp(id).get(nb-3)));
        objetJava.setTemp(String.valueOf(requete.GetOneTemp(id).get(nb-2)));
        objetJava.setId(String.valueOf(requete.GetOneTemp(id).get(nb-1)));
        return objetJava;
    }

    @PUT // Requête put pour une mise à jour
    @Path("/temperature/{id}")
    @Produces("application/json")
    @Consumes("application/x-www-form-urlencoded")
    public String updateTempById(@PathParam("id") int id,String tempjson) throws IOException {

        ObjectMapper objectMapper = new ObjectMapper(); //création d'un objet objectMapper pour désérialiser un objet JSON
        JsonNode jsonNode = objectMapper.readTree(tempjson); //Lecture de l'objet JSON reçudans le corps de la requête PUT
        String temp = jsonNode.get("temp").asText();//sélection de la valeur de la température
        Request requete = new Request();//création d'une requête sql
        requete.UpdateOneTemp(id,temp);//Mise à jour de l'enregistrement demandé avec la valeur de la température.
        return "mise à jour effectuée de l'enregistrement " + id;
    }

    @DELETE // Requête Delete pour effacer un enregistrement
    @Path("/temperature/{id}")
    @Produces("application/json")
    public String deleteTempById(@PathParam("id") int id) {

        Objet objetJava =new Objet();
        Request requete = new Request();
        requete.deleteTempById(id);
        return "effacement enregistrement "+id+" effectué";
    }
```

Classe Request :

```
package fr.snir.rest;

import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class Request {
    String url =
"jdbc:mysql://171.17.0.3:3306/METEO?useUnicode=true&useJDBCCompliantTimezoneShift=true&useLegacyDatetimeCode=fa
lse&serverTimezone=UTC";
    String login ="root"; //identifiant bdd mysql
    String passwd = "root";//identifiant bdd mysql
    Connection cn =null;
    Statement st = null;
    ResultSet rs = null;
    String value = null;
    //String temp;
    //String time;
    //String id;
    List<String> liste =  new ArrayList<String>() ; //création d'une liste pour stocker id,temp et time


    public List GetTemp() { //Méthode pour lire la base de donnée
        this.url = url;
        this.login = login;
        this.passwd = passwd;
        this.cn = cn;
        this.st = st;
        this.rs = rs;

        try {
            try {
                Class.forName("com.mysql.cj.jdbc.Driver");//chargement du pilote mysql
            } catch (ClassNotFoundException e) {
                e.printStackTrace();
```

```
        }
        cn = DriverManager.getConnection(url,login,passwd);//ouverture d'une connexion à la base
        st = cn.createStatement(); //création d'une requête
        String sql = "SELECT * FROM TEMPERATURE";

        rs = st.executeQuery(sql); //exécution de la requête et récupération du résultat

        while (rs.next()){
            liste.add(rs.getString("TIME")) ; // récupération de chaque ligne et stockage dans une liste
            liste.add(rs.getString("TEMP")) ;
            liste.add(rs.getString("ID")) ;
        }

    } catch (SQLException throwables) {
        throwables.printStackTrace();
    }
    finally {
        try {
            cn.close(); //fermeture de la connexion
        } catch (SQLException throwables) {
            throwables.printStackTrace();
        }
        try {
            st.close();//fermeture de la requête
        } catch (SQLException throwables) {
            throwables.printStackTrace();
        }
    }

    return liste;
}


public String  AddTemp(String value) { //méthode pour ajouter un enregistrement
    this.url = url;
    this.login = login;
    this.passwd = passwd;
    this.value = value;
    this.cn = cn;
    this.st = st;
    this.rs = rs;

    try {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }
        cn = DriverManager.getConnection(url,login,passwd);
        st = cn.createStatement();
        String sql = "INSERT INTO TEMPERATURE(TEMP) VALUES('"+value+"')";

        st.executeUpdate(sql); //exécution de la mise à jour

    } catch (SQLException throwables) {
        throwables.printStackTrace();
    }
    finally {
        try {
            cn.close();
        } catch (SQLException throwables) {
            throwables.printStackTrace();
        }
        try {
            st.close();
        } catch (SQLException throwables) {
            throwables.printStackTrace();
        }
    }
    return "température ajoutée";
}

public List GetOneTemp(int ID) { //méthode pour sélectionner un enregistrement
    this.url = url;
    this.login = login;
    this.passwd = passwd;
```

```
        this.cn = cn;
        this.st = st;
        this.rs = rs;

        try {
            try {
                Class.forName("com.mysql.cj.jdbc.Driver");
            } catch (ClassNotFoundException e) {
                e.printStackTrace();
            }
            cn = DriverManager.getConnection(url,login,passwd);
            st = cn.createStatement();
            String sql = "SELECT * FROM TEMPERATURE WHERE ID="+ID+" LIMIT 1";

            rs = st.executeQuery(sql);


            while (rs.next()){
                //System.out.println(rs.getString("TIME")+" "+rs.getString("TEMP"));
                //System.out.println(rs.getString("TEMP"));
                liste.add(rs.getString("TIME")) ;
                liste.add(rs.getString("TEMP")) ;
                liste.add(rs.getString("ID")) ;
            }

        } catch (SQLException throwables) {
            throwables.printStackTrace();
        }
        finally {
            try {
                cn.close();
            } catch (SQLException throwables) {
                throwables.printStackTrace();
            }
            try {
                st.close();
            } catch (SQLException throwables) {
                throwables.printStackTrace();
            }

            return liste;
        }

    }

    public List UpdateOneTemp(int ID,String value) { //méthode pour mettre à jour un enregistrement
        this.url = url;
        this.login = login;
        this.passwd = passwd;
        this.cn = cn;
        this.st = st;
        this.rs = rs;

        try {
            try {
                Class.forName("com.mysql.cj.jdbc.Driver");
            } catch (ClassNotFoundException e) {
                e.printStackTrace();
            }
            cn = DriverManager.getConnection(url,login,passwd);
            st = cn.createStatement();
            String sql = "UPDATE TEMPERATURE SET TEMP='"+value+"' WHERE id="+ID;

            st.executeUpdate(sql);


        } catch (SQLException throwables) {
            throwables.printStackTrace();
        }
        finally {
            try {
                cn.close();
            } catch (SQLException throwables) {
                throwables.printStackTrace();
            }
            try {
```

```
            st.close();
        } catch (SQLException throwables) {
            throwables.printStackTrace();
        }

        return liste;
    }

}

public String deleteTempById(int ID) { //méthode pour effacer un enregistrement
    this.url = url;
    this.login = login;
    this.passwd = passwd;
    this.cn = cn;
    this.st = st;
    this.rs = rs;

    try {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }
        cn = DriverManager.getConnection(url,login,passwd);
        st = cn.createStatement();
        String sql = "DELETE FROM TEMPERATURE WHERE id="+ID;

        st.executeUpdate(sql);


    } catch (SQLException throwables) {
        throwables.printStackTrace();
    }
    finally {
        try {
            cn.close();
        } catch (SQLException throwables) {
            throwables.printStackTrace();
        }
        try {
            st.close();
        } catch (SQLException throwables) {
            throwables.printStackTrace();
        }

        return "Enregistrement effacé";
    }

}
```

Création de la classe Objet :

```
package fr.snir.rest;

public class Objet { //création d'un objet java
    private String id;
    private String date;
    private String temp;

    public String getTemp() {
        return temp;
    }

    public void setTemp(String temp) {
        this.temp = temp;
    }

    public String getDate() {
        return date;
    }

    public void setDate(String date) {
        this.date = date;
```

```
    }

    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }
}
```