

## Compte Rendu : Projet Web Services

### Introduction

Dans le cadre de notre projet Web Services, nous avons réalisé deux serveurs l'un implémentant la méthode SOAP, l'autre la méthode REST.

Le projet a été réalisé en collaboration par M. Ansari Antoine, M. Tancev Simon, M. Vovard Hugo et M. Douailly Yann. Il est disponible à l'adresse web :

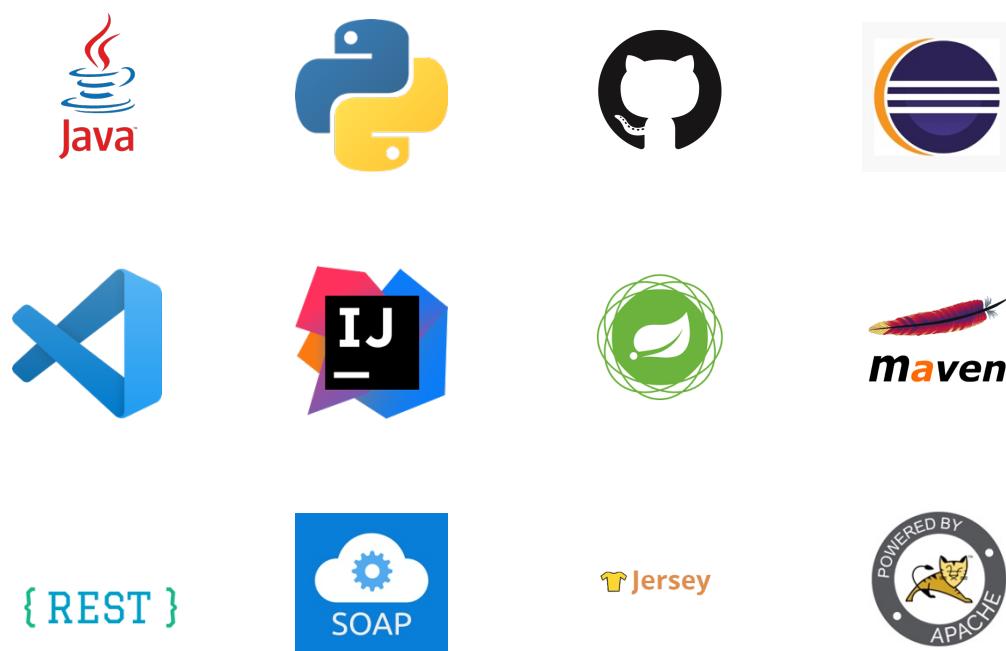
<https://github.com/Mohsen51/ProjetWebServices>

lien de téléchargement : <https://github.com/Mohsen51/ProjetWebServices/archive/master.zip>

Le serveur SOAP répond à la demande du cahier des charges d'implémenter une communication avec une API externe, nous avons choisi d'interroger l'API gratuite COVID19 API (<https://covid19api.com/>) qui fournit des données sur l'évolution de la pandémie à travers le monde. Le serveur SOAP fonctionne sur le port 7777 et implémente donc dans son fonctionnement interne un client REST Java qui récupère les données. Nous avons choisi de se limiter aux données mondiales, pour se concentrer sur le reste du projet. Les données sont ensuite parsées et stockées dans une classe à part.

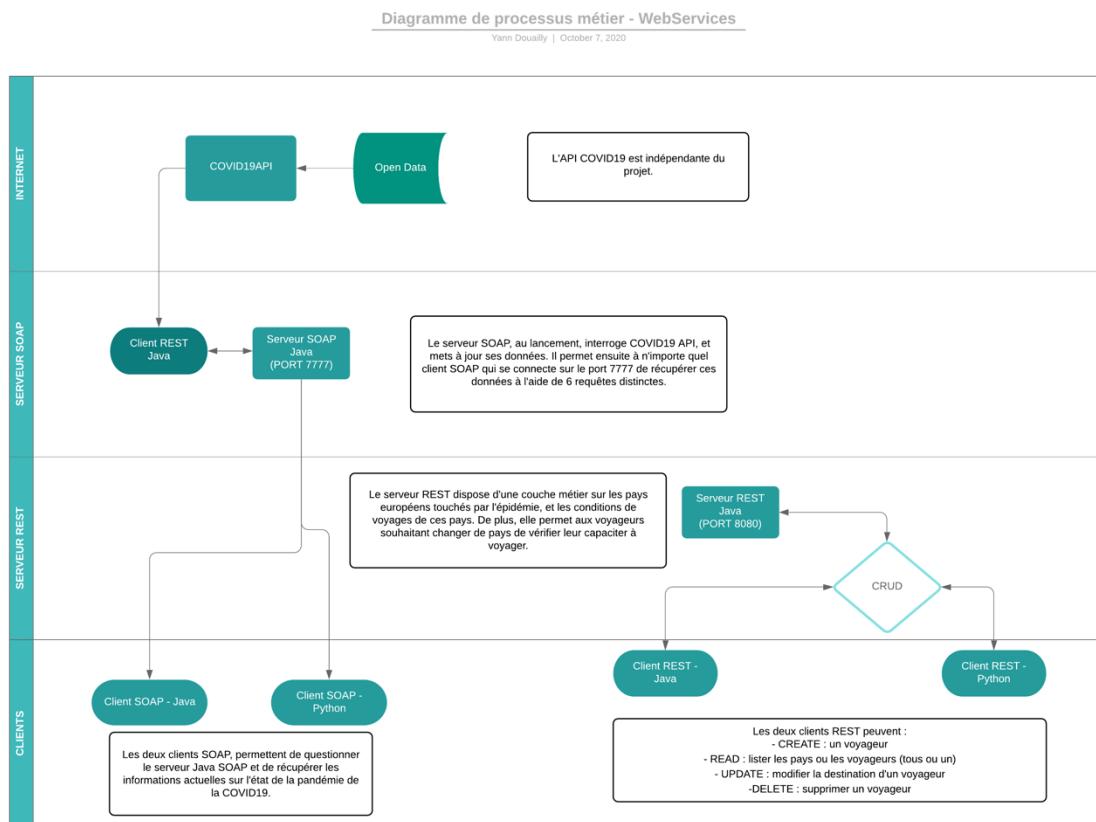
Le serveur REST est écrit en Java. Il est conçu avec la technologie Spring Framework, qui permet de gérer automatiquement les dépendances Maven. Il implémente un serveur Tomcat sur le port 8080. En pleine crise due au coronavirus, il permet de stocker des informations concernant les politiques d'entrées dans les différents pays européens (test PCR, isolement, etc.) ainsi que les noms, prénoms, et pays de destination des personnes qui souhaiteraient voyager. Pour répondre au cahier des charges du projet, le serveur implémente les opérations CRUD, sur les voyageurs. Pour les pays, seul un READ est permis, puisque ces politiques n'ont pas à être modifier par le client.

Chaque serveur est accompagné de deux clients, l'un en Python et l'autre en Java.

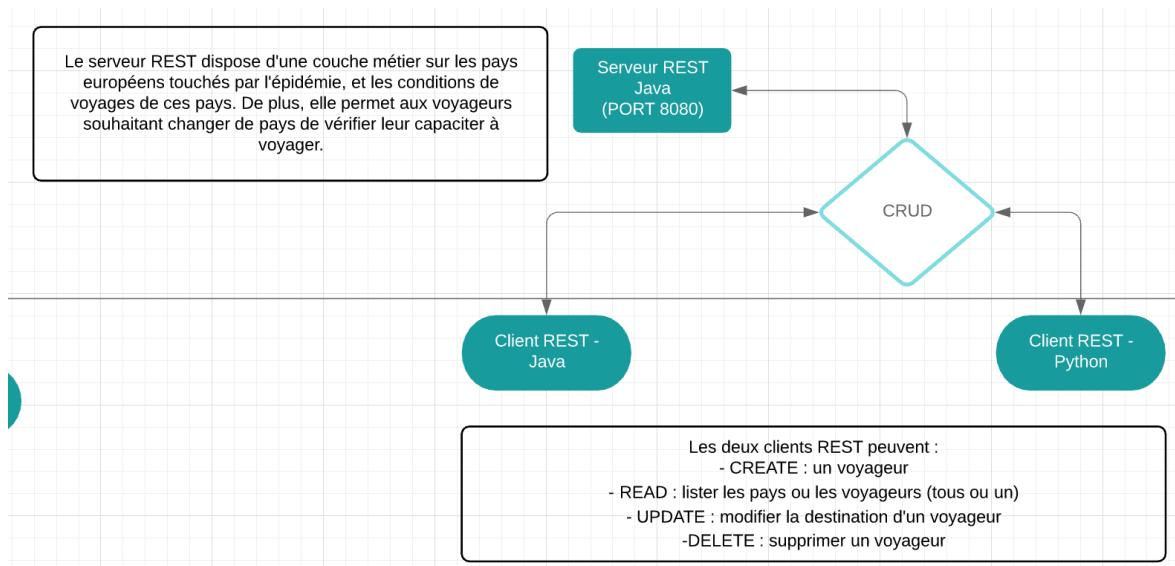


## Partie 1 : Modélisation

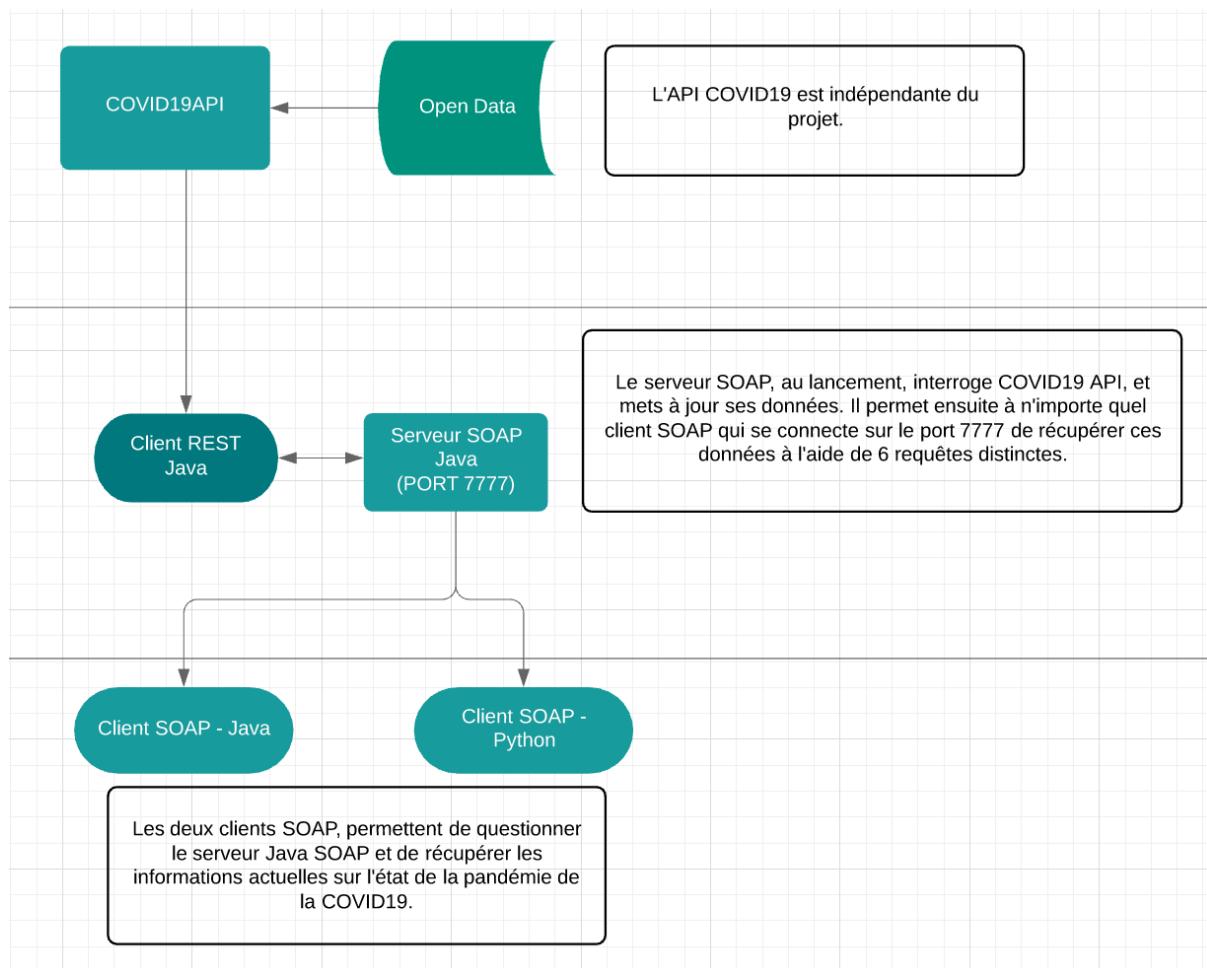
### a) Modélisation du projet



### a) Modélisation de la partie REST



a) Modélisation de la partie SOAP



## Partie 2 : Fonctionnalités

### a) Serveur SOAP

Le serveur SOAP est organisé en 4 classes Java :

- Server.java : la classe principale qui initie l'objet API, et les services SOAP sur le port 7777
- API.java : la classe client REST qui interroge l'API COVID19
- Data.java : la classe qui stocke les données recueillis par API.java, les parse et les affiche
- Services.java : la classe qui définit les services SOAP

The image shows four Java code editors side-by-side:

- Server.java:** Contains the main method that creates an API object, publishes it to a local host port, and prints "SERVEUR : +URL".
- API.java:** Contains static methods for requesting the API, printing requests, updating data, and a private static Data object.
- Data.java:** Contains static variables for pandemic statistics and methods for printing, updating, and printing all values.
- Services.java:** A WebService class with various methods returning integers representing total deaths, total recovered, total confirmed, new deaths, new recovered, and new confirmed cases.

Le serveur SOAP fournit donc 6 requêtes à chaque client qui s'y connecte, afin de recueillir les informations récentes sur la propagation de la COVID19, à l'échelle mondiale.

### b) Client SOAP – Java

Le client SOAP Java est composé d'une multitude de classes. Cependant, celle qui est intéressante est la classe Client.java. Elle permet d'interroger le serveur jusqu'à ce qu'on ait les informations souhaitées et qu'on quitte.

The screenshot shows an IDE interface with two main parts. The top part is a code editor window titled "Client.java" containing the following Java code:

```
1 package client;
2
3 import services.COVID19openData;
4
5 public class Client {
6
7     public static void main(String[] args) {
8         Services stub = new COVID19openData().getServicesPort();
9         Scanner sc = new Scanner (System.in);
10        int i = -1;
11
12        while (i != 0) {
13            System.out.println("Bonjour,\nEntrez 0 pour quitter.\n"
14                + "Entrez 1 pour obtenir le nombre de nouveau cas sur 24h.\n"
15                + "Entrez 2 pour obtenir le nombre de mort sur 24h.\n"
16                + "Entrez 3 pour obtenir le nombre de personne soignée sur 24h\n"
17                + "Entrez 4 pour obtenir le nombre total de cas.\n"
18                + "Entrez 5 pour obtenir le nombre total de décès.\n"
19                + "Entrez 6 pour obtenir le nombre total de personne soignée.\n");
20            i = sc.nextInt();
21            switch (i) {
22                case 1 :
23                    System.out.println("Nouveaux cas sur 24h : "+ stub.newConfirmed() +"\n");
24                    break;
25                case 2 :
26                    System.out.println("Morts sur 24h : "+ stub.newDeaths() +"\n");
27                    break;
28                case 3 :
29                    System.out.println("Soignées sur 24h : "+ stub.newRecovered() +"\n");
30                    break;
31                case 4 :
32                    System.out.println("Cas total : "+ stub.totalConfirmed() +"\n");
33                    break;
34                case 5 :
35                    System.out.println("Morts total : "+ stub.totalDeaths() +"\n");
36                    break;
37                case 6:
38                    System.out.println("Soignées total : "+ stub.totalRecovered() +"\n");
39                    break;
40            }
41        }
42    }
43
44    System.out.println("Merci et à bientôt!\n");
45}
```

The bottom part is a file browser showing the project structure:

- Client\_SOAP\_java [ProjetWebServices master]
  - JRE System Library [AdoptOpenJDK 8 [1.8.0\_265]
  - src
    - client
      - Client.java
    - services
      - COVID19OpenData.java
      - NewConfirmed.java
      - NewConfirmedResponse.java
      - NewDeaths.java
      - NewDeathsResponse.java
      - NewRecovered.java
      - NewRecoveredResponse.java
      - ObjectFactory.java
      - package-info.java
      - Services.java
      - TotalConfirmed.java
      - TotalConfirmedResponse.java
      - TotalDeaths.java
      - TotalDeathsResponse.java
      - TotalRecovered.java
      - TotalRecoveredResponse.java
  - README.md

L'ensemble des classes a été généré avec SOAPUI, et la bibliothèque WS-Import. La classe Client.java a été créé pour formaliser l'interface.

### c) Client SOAP – Python

---

Le client python n'utilise pas la même structure que celui écrit en Java, il est composé d'un seul fichier, comportant 7 fonctions, une pour l'interface du menu, et une pour chaque requête proposée par le serveur SOAP.

```
clientSOAPpython.py
1 import requests
2 import lxml.etree
3 import xml.etree.ElementTree as ET
4 import colorama
5 import time
6
7 colorama.init()
8
9 url = "http://localhost:7777/?wsdl"
10 headers = {'content-type': 'text/xml'}
11
12 > class bcolors: ...
13
14 > def NewConfirmed(): ...
15
16 > def NewDeath(): ...
17 > </soapenv:Envelope>"'" ...
18
19
20
21
22
23 > def NewRecovered(): ...
24
25 > def TotalConfirmed(): ...
26 > </soapenv:Envelope>"'" ...
27
28
29
30
31 > def TotalDeaths(): ...
32 > </soapenv:Envelope>"'" ...
33
34
35
36 > def TotalRecovered(): ...
37 > </soapenv:Envelope>"'" ...
38
39
40
41
42 > def menu(): ...
43
44
45
46
47
48
49
50
51
52
53
54
55
56 > def NewRecovered(): ...
57 > </soapenv:Envelope>"'" ...
58
59
60
61
62 > def TotalConfirmed(): ...
63 > </soapenv:Envelope>"'" ...
64
65
66
67 > def TotalDeaths(): ...
68 > </soapenv:Envelope>"'" ...
69
70
71
72
73 > def TotalRecovered(): ...
74 > </soapenv:Envelope>"'" ...
75
76
77
78
79
80 > def menu(): ...
81
82
83
84
85
86
87
88
89
89 > def menu(): ...
90 > </soapenv:Envelope>"'" ...
91
92
93
94
95
96 > def menu(): ...
97 > </soapenv:Envelope>"'" ...
98
99
100
101
102
103
104
105 > def menu(): ...
106 > </soapenv:Envelope>"'" ...
107
108
109
110
111 > def menu(): ...
112 > </soapenv:Envelope>"'" ...
113
114
115
116
117 > def menu(): ...
118 > </soapenv:Envelope>"'" ...
119
120
121
122
123
124 > def menu(): ...
125 > </soapenv:Envelope>"'" ...
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
159 > def menu(): ...
160 > </soapenv:Envelope>"'" ...
161
162
163
164
165
166
167
168
169
169 > def menu(): ...
170 > </soapenv:Envelope>"'" ...
171
172
173
174
175
176
177 menu()
```

### d) Serveur REST – Java

---

Le serveur REST est organisé en 3 packages et 6 classes :

- Package **test** : TestApplication.java la classe principale qui permet de lancer le serveur AppConfig.java qui détermine le path utilisé pour accéder à l'api (/api)
- Package **resources** : Resource.java qui détermine le path utilisé pour chaque opérations CRUD (PUT, GET, POST, DELETE), les types de retours sur chaque opération (JSON), et les méthodes Java qui sont appelées par ces opérations.
- Package **metiers** : methodsMetiers.java qui contient les méthodes Java permettant de faire les opérations sur les données stockées  
Pays.java qui définit la représentation des données sur un pays en Java  
Voyageur.java qui définit la représentation des données sur un voyageur en Java

Les dépendances du projet Java sont résolues par l'outil Maven. Le Framework Spring est utilisé pour implémenter le serveur Tomcat ainsi que l'outil Jersey. L'ensemble des dépendances est visible dans l'IDE IntelliJ.

```
package com.test.test;

import org.glassfish.jersey.server.ResourceConfig;
import org.springframework.stereotype.Component;

import javax.ws.rs.ApplicationPath;

@Component
@ApplicationPath("/api")
public class AppConfig extends ResourceConfig
{
    public AppConfig() { packages("com.test.resources"); }
}
```

AppConfig.java

```
package com.test.test;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class TestApplication {

    public static void main(String[] args) { SpringApplication.run(TestApplication.class, args); }

}
```

TestApplication.java

```
package com.test.resources;

import com.test.metiers.*;
import com.test.metiers.entites.Pays;
import com.test.metiers.entites.Voyageur;

import javax.inject.Singleton;
import javax.ws.rs.*;
import javax.ws.rs.core.MediaType;
import java.util.List;

//Singleton est utilisé pour n'appeler qu'une seule fois la méthode de création des pays seulement
@Singleton
@Path("/")
public class Resource {

    private methodsMetiers metier;

    public Resource() {...}

    @GET
    @Path("pays")
    //Produce permet à d'indiquer dans quel format de données va être retourner l'info (ici JSON)
    @Produces({MediaType.APPLICATION_JSON})
    public List<Pays> getAllPays() { return metier.getAllPays(); }
}
```

Resource.java – screenshot 1

```

    @GET
    @Path("voyageurs")
    @Produces(MediaType.APPLICATION_JSON)
    public List<Voyageur> getAllVoyageurs() { return metier.getAllVoyageurs(); }

    @GET
    @Path("pays/{idPays}")
    @Produces(MediaType.APPLICATION_JSON)
    public Pays getPays(@PathParam("idPays") String nomPays) { return metier.getPays(nomPays); }

    @GET
    @Path("voyageurs/{idUser}")
    @Produces(MediaType.APPLICATION_JSON)
    public Voyageur getPays(@PathParam("idUser") Integer idVoyageur) { return metier.getVoyageur(i...}

    @POST
    @Path("voyageur/{idPays}")
    //Consumes permet d'indiquer dans quel format de données va être reçu l'info (ici JSON)
    @Consumes(MediaType.APPLICATION_JSON)
    @Produces(MediaType.APPLICATION_JSON)
    public Voyageur createVoyageur(Voyageur v, @PathParam("idPays") String nomPays) {...}
  
```

Resource.java – screenshot 2

```

    //Supprime un utilisateur
    @DELETE
    @Path("voyageurs/{idVoyageur}")
    @Produces(MediaType.APPLICATION_JSON)
    public int deleteVoyageur(@PathParam("idVoyageur") Integer idVoyageur) { return metier.deleteVoyageur(idVoyageur); }

    //Update le pays de destination d'un voyageur
    @PUT
    @Path("voyageurs/{idUser}")
    @Consumes(MediaType.APPLICATION_JSON)
    @Produces(MediaType.APPLICATION_JSON)
    public Voyageur changeVoyageur(Pays p, @PathParam("idUser") Integer idVoyageur) {...}
  
```

Resource.java – screenshot 3

```

package com.test.metiers;

import com.test.metiers.entites.Pays;
import com.test.metiers.entites.Voyageur;

import java.util.*;

public class methodsMetiers {

    //Creation des collections de données map stockant tous les pays et voyageurs
    private Map<String, Pays> pays = new HashMap<>();
    private Map<Integer, Voyageur> voyageurs = new HashMap<>();

    //Obtenir les infos de tous les voyageurs (GET)
    public List<Pays> getAllPays() { return new ArrayList<Pays>(pays.values()); }

    //Obtenir les infos de tous les pays (GET)
    public List<Voyageur> getAllVoyageurs() { return new ArrayList<Voyageur>(voyageurs.values()); }

    //Obtenir les infos d'un pays particulier (GET)
    public Pays getPayByName(String nomPays) {...}
  
```

methodsMetiers.java – screenshot 1

```
//Obtenir les infos d'un voyageur particulier (GET)
public Voyageur getVoyageur(Integer idVoyageur) {...}

//Méthode d'ajout de voyageur (On va l'utiliser pour le POST)
public Voyageur addVoyageur(Voyageur v, String nomPays){...}

//Ajout d'un pays, utilisé exclusivement à l'initialisation du serveur
public Pays addPays(Pays p){...}

//Méthode qui utilise la méthode de suppression de Map (utilisée pour DELETE)
public int deleteVoyageur(Integer idVoyageur){...}

//Méthode de mise à jour (utilisée pour PUT)
public Voyageur updateVoyageur(Voyageur v, String nomPays, Integer idVoyageur){...}

//Création des objets pays et leurs règles
public void creationPays() {...}
}
```

methodsMetiers.java – screenshot 2

```
public class Pays implements Serializable {
    private String nomPays;
    private String accesPays;

    public String getNomPays() { return nomPays; }

    public String getAccesPays() { return accesPays; }

    public void setNomPays(String nomPays) { this.nomPays = nomPays; }

    public void setAccesPays(String accesPays) { this.accesPays = accesPays; }

    @Override
    public String toString() {
        return "Pays{" +
            "nomPays='" + nomPays + '\'' +
            ", accesPays='" + accesPays + '\'' +
            '}';
    }

    public Pays(String nomPays, String accesPays) {
        this.nomPays = nomPays;
        this.accesPays = accesPays;
    }
}
```

Pays.java

```
package com.test.metiers.entites;

import java.io.Serializable;

public class Voyageur implements Serializable {
    private int idVoyageur;
    private String nomVoyageur;
    private String prenomVoyageur;
    private Pays pays;

    public Voyageur() {}

    public Voyageur(String nomVoyageur, String prenomVoyageur) {}

    public Voyageur(int idVoyageur, String nomVoyageur, String prenomVoyageur) {}

    public Voyageur(int idVoyageur, String nomVoyageur, String prenomVoyageur, Pays pays) {}

    public String getNomVoyageur() { return nomVoyageur; }

    public void setNomVoyageur(String nomVoyageur) { this.nomVoyageur = nomVoyageur; }

    public int getIdVoyageur() { return idVoyageur; }
```

Voyageur.java – screenshot 1

```
public void setIdVoyageur(int idVoyageur) { this.idVoyageur = idVoyageur; }

public String getPrenomVoyageur() { return prenomVoyageur; }

public void setPrenomVoyageur(String prenomVoyageur) { this.prenomVoyageur = prenomVoyageur; }

public Pays getPays() { return pays; }

public void setPays(Pays pays) { this.pays = pays; }
```

Voyageur.java – screenshot 2

#### e) Client REST – Java

Le serveur REST est organisé en 4 classes Java :

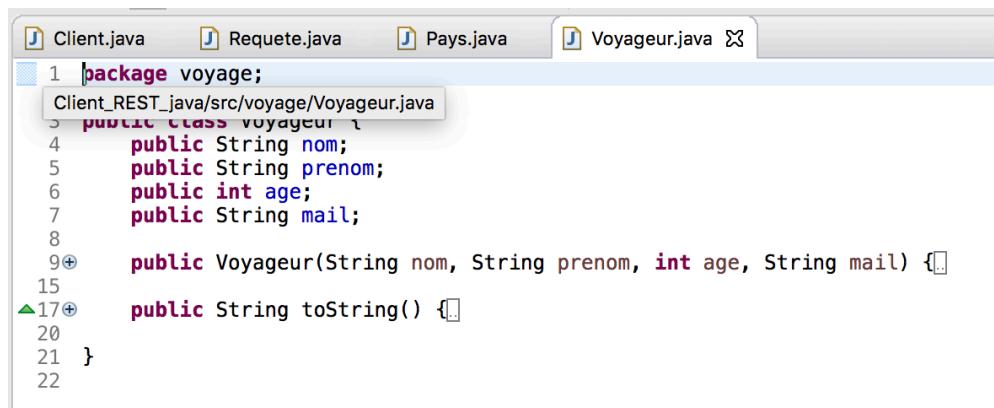
- Client.java : la classe principale qui dispense l'interface du client, elle utilise Requete.java
- Requete.java : la classe client REST qui interroge l'API du serveur et stocke les informations
- Pays.java : la classe qui modélise les données de chaque pays, elle est utilisée par Requete.java
- Voyageur.java : la classe qui modélise les données de chaque voyageur, elle est utilisée par Requete.java

The image shows three Java code editors side-by-side, each with tabs for Client.java, Requete.java, Pays.java, and Voyageur.java. The first editor (Client.java) contains methods for inscription, desinscription, informations, modification, and main. The second editor (Requete.java) contains methods for get\_pays, get\_all\_pays, get\_voyageur, get\_all\_voyageur, get\_format, print\_requete, add\_voyageur, post\_requete, update, and delete\_voyageur. The third editor (Pays.java) contains methods for constructor, set\_regle, and toString.

```
Client.java
package voyage;
import java.util.*;
public class Client {
    public static void inscription (Scanner sc, Requete rq) {}
    public static void desinscription (Scanner sc, Requete rq) {}
    public static void informations (Scanner sc, Requete rq) {}
    public static void modification (Scanner sc, Requete rq) {}
    public static void main(String[] args) {}
}
```

```
Requete.java
package voyage;
import com.sun.jersey.api.client.Client;
public class Requete {
    private Client client;
    private String domaine;
    public Requete () {}
    public void get_pays(String s) {}
    public void get_all_pays() {}
    public void get_voyageur (int id) {}
    public void get_all_voyageur() {}
    private String get_format (String s) {}
    public void print_requete (String s) {}
    public void add_voyageur (String nom, String prenom, String pays) {}
    private void post_requete(String path, String json) {}
    public void update (int id, String pays) {}
    public void delete_voyageur (int id) {}
}
```

```
Pays.java
import java.util.*;
public class Pays {
    public String name;
    protected int regle;
    protected ArrayList<Voyageur> list_voyageur;
    public Pays(String s, int regle, ArrayList<Voyageur> list_voyageur) {}
    public Pays(String name, int regle) {}
    public Pays(String name) {}
    public int set_regle (int a) {}
    public String toString() {}
}
```

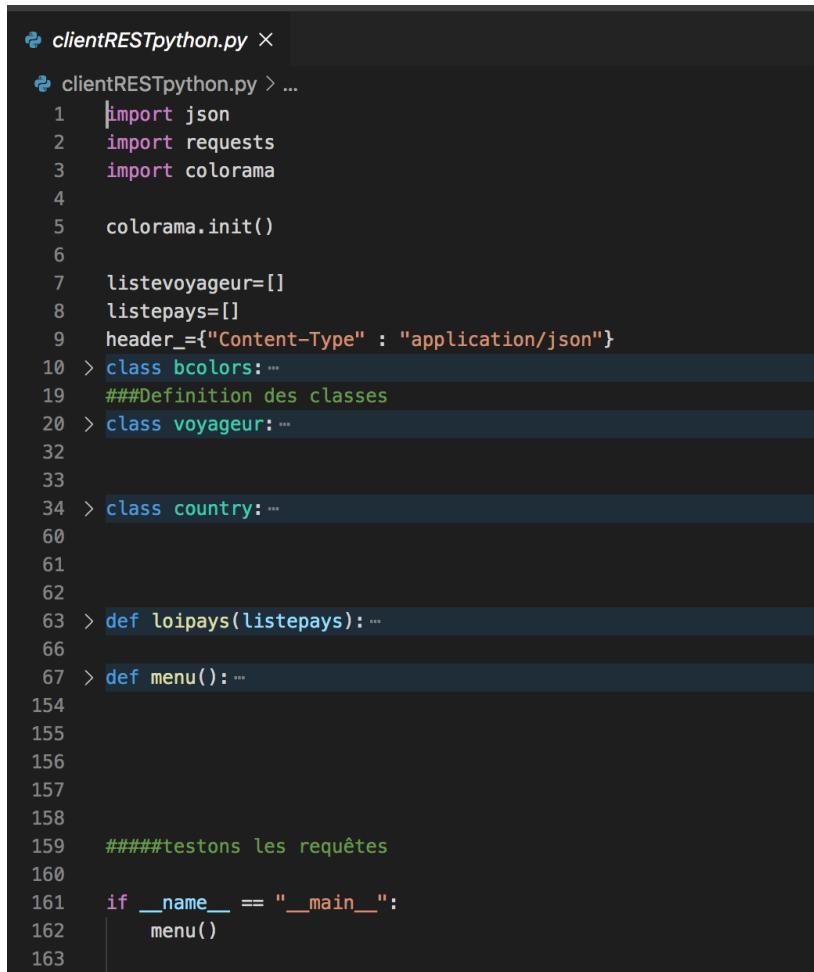


A screenshot of a Java code editor showing the `Voyageur.java` file. The code defines a class `Voyageur` with attributes `nom`, `prenom`, `age`, and `mail`. It includes a constructor and a `toString` method. The code editor interface shows tabs for other files like `Client.java`, `Requete.java`, `Pays.java`, and `Voyageur.java`.

```
1 package voyage;
2 Client_REST_java/src/voyage/Voyageur.java
3 public class Voyageur {
4     public String nom;
5     public String prenom;
6     public int age;
7     public String mail;
8
9     public Voyageur(String nom, String prenom, int age, String mail) {
10
11
12     }
13
14     public String toString() {
15
16
17     }
18
19 }
20
21
22
```

#### f) Client REST – Python

Le client REST Python modélise les données avec les classes « voyageur » et « country ». L’interface est dispensée par la fonction « menu() » il permet de lancer toutes les requêtes fournies par le serveur REST.



A screenshot of a Python code editor showing the `clientRESTpython.py` file. The code imports `json`, `requests`, and `colorama`. It initializes `colorama`, defines lists for `voyageur` and `pays`, and sets a header for JSON requests. It contains sections for defining classes `voyageur` and `country`, a function `loipays` to handle `pays` requests, and a `menu` function for the application logic. The code ends with a check for the main module execution.

```
clientRESTpython.py ×
clientRESTpython.py > ...
1 import json
2 import requests
3 import colorama
4
5 colorama.init()
6
7 listevoyageur=[]
8 listepays=[]
9 header_={"Content-Type" : "application/json"}
10 > class bcolors:...
11     """Definition des classes
12     > class voyageur:...
13
14     > class country:...
15
16
17     > def loipays(listepays):...
18
19     > def menu():...
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159     #####testons les requêtes
160
161     if __name__ == "__main__":
162         menu()
163
```

### Partie 3 : Exécution

#### a) Serveur REST – Java 12+

Un README est fourni avec le serveur REST.

Il est lancé grâce à l'invite de commandes, intégré à l'IDE ou directement dans l'OS.

The screenshot shows the IntelliJ IDEA interface with the "RestServer" project open. The terminal window displays the following log output:

```
[ 20-10-07 10:05:38.832 INFO 7401 --- [ication.main()] com.test.test.TestApplication : Starting TestApplication on MacBook-YAN.local with P
ID 7401 (/Users/yan/Documents/GitHub/ProjetWebServices/RestServer/target/classes) started by yan in /Users/yan/Documents/GitHub/ProjetWebServices/RestSe
rver)
2020-10-07 10:05:38.836 INFO 7401 --- [ication.main()] com.test.test.TestApplication : No active profile set, falling back to default profi
les: default
2020-10-07 10:05:39.837 INFO 7401 --- [ication.main()] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2020-10-07 10:05:39.852 INFO 7401 --- [ication.main()] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2020-10-07 10:05:39.852 INFO 7401 --- [ication.main()] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.38]
2020-10-07 10:05:39.957 INFO 7401 --- [ication.main()] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2020-10-07 10:05:39.957 INFO 7401 --- [ication.main()] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed
```

#### b) Client REST - Python

IDE conseillé : Visual Studio Code (latest)

OS Compatible : Windows / macOS / Linux

Lancement : “Run without debugging”

The screenshot shows the Visual Studio Code interface with the file "clientRESTPython.py" open. The terminal window displays the following interaction with the application:

```
clientRESTPython.py — Clients Python
clientRESTPython.py
  ↴ class country:
    ↴
    ↴ Que voulez-vous faire ?
    ↴ 1. Voir les lois covid des différents pays
    ↴ 2. Voir la liste de tous les voyageurs
    ↴ 3. Voir la législation des pays
    ↴ 4. Voir les informations d'un Voyageur
    ↴ 5. Rajouter un voyageur
    ↴ 6.Modifier la destination d'un voyageur
    ↴ 7. Supprimez un voyageur
    ↴
    ↴ Votre choix : 1
    ↴ Italie : Ouvert
    ↴ Allemagne : test PCR à l'entrée
    ↴ Andorre : Ouvert
    ↴ Monaco : Ouvert
    ↴ Suisse : test PCR à l'entrée
    ↴ RoyaumeUni : Isolement de 14jours à l'entrée
    ↴ France : Pas de restriction entre les régions françaises
    ↴ Espagne : Ouvert
    ↴ Belgique : Ouvert
    ↴ Appuyez sur nimporte quelle touche pour continuer...
    ↴
    ↴ Bienvenues sur l'interface covid-WS
    ↴
    ↴ Que voulez-vous faire ?
    ↴ 1. Voir les lois covid des différents pays
    ↴ 2. Voir la liste de tous les voyageurs
    ↴ 3. Voir la législation des pays
    ↴ 4. Voir les informations d'un Voyageur
    ↴ 5. Rajouter un voyageur
    ↴ 6.Modifier la destination d'un voyageur
    ↴ 7. Supprimez un voyageur
    ↴
    ↴ Votre choix : 2
    ↴ 0
    ↴ 1 : Nom : Dupont - Prénom : Jean - Pays : Allemagne
    ↴ Appuyez sur nimporte quelle touche pour continuer...
```

```

clientRESTpython.py — Clients Python
clientRESTpython.py > clientRESTpython.py > Class country:
5. Rajouter un voyageur
6. Modifier la destination d'un voyageur
7. Supprimez un voyageur

Votre choix : 5
Quel pays ? France
Prénom Yann
Nom Douailly
Tout est bon! Le voyageur a été ajouté!
Appuyez sur n'importe quelle touche pour continuer...

Bienvenues sur l'interface covid-WS

Que voulez-vous faire ?

1. Voir les lois covid des différents pays
2. Voir la liste de tous les voyageurs
3. Voir la législation des pays
4. Voir les informations d'un Voyageur
5. Rajouter un voyageur
6.Modifier la destination d'un voyageur
7. Supprimez un voyageur

Votre choix : 2
0
1 : Nom : Dupont - Prénom : Jean - Pays : Allemagne
2 : Nom : Douailly - Prénom : Yann - Pays : France
Appuyez sur n'importe quelle touche pour continuer...

Bienvenues sur l'interface covid-WS

Que voulez-vous faire ?

1. Voir les lois covid des différents pays
2. Voir la liste de tous les voyageurs
3. Voir la législation des pays
4. Voir les informations d'un Voyageur
5. Rajouter un voyageur
6.Modifier la destination d'un voyageur
7. Supprimez un voyageur

Votre choix : 7
Quel est le voyageur que vous voulez retirer? Donnez son id 1
Tout est bon! Le voyageur a été supprimé!

```

### c) Client REST – Java 8+

IDE conseillé : Eclipse IDE (2020.06 ou inférieure pour compatibilité Java 8)  
OS Compatible : Windows / macOS / Linux

```

eclipse-workspace - Eclipse IDE
File Edit Navigate Search Project Run Window Help
Client Java Application] /Library/Java/JavaVirtualMachines/adoptopenjdk-8.jdk/Contents/Home/bin/java (Oct 7, 2020, 10:09:29 AM)
Que voulez-vous faire :
ins : inscription
des : desinscription
inf : information
mod : modification des regles d'un pays

inf
INFORMATIONS : Entrez 1 pour les informations sur les voyageurs.
Entrez 2 pour les informations sur les pays.
2
INFORMATIONS : Souhaitez vous consulter un pays en particulier ? (y/n)
n
[{"nomPays":"Italie","accesPays":"Ouvert"}, {"nomPays":"Allemagne","accesPays":"Test PCR à l'entrée"}, {"nomPays":"Andorre","accesPays":"Ouvert"}]

Que voulez-vous faire :
ins : inscription
des : desinscription
inf : information
mod : modification des regles d'un pays

ins
INSCRIPTION :
Veuillez entrez votre nom :
Douailly
INSCRIPTION : Veuillez entrez votre prenom :
Yann
INSCRIPTION : Veuillez entrez le nom du pays en minuscule:
italie
Output from Server ....
{"idVoyageur":13,"nomVoyageur":"Douailly","prenomVoyageur":"Yann","pays":{"nomPays":"Italie","accesPays":"Ouvert"}}

Que voulez-vous faire :
ins : inscription
des : desinscription
inf : information
mod : modification des regles d'un pays

inf
INFORMATIONS : Entrez 1 pour les informations sur les voyageurs.
Entrez 2 pour les informations sur les pays.

```

```

eclipse-workspace - Eclipse IDE

Client [Java Application] /Library/Java/JavaVirtualMachines/adoptopenjdk-8.jdk/Contents/Home/bin/java (Oct 7, 2020, 10:09:29 AM)
inf
INFORMATIONS : Entrez 1 pour les informations sur les voyageurs.
Entrez 2 pour les informations sur les pays.
1
INFORMATIONS : Souhaitez vous consulter un voyageur en particulier ? (y/n)
n
[{"idVoyageur":2,"nomVoyageur":"Douailly","prenomVoyageur":"Yann","pays":{"nomPays":"France","accesPays":"Pas de restriction entrée"}, "modifications":[]}
Que voulez-vous faire :
ins : inscription
des : désinscription
inf : information
mod : modification des règles d'un pays

mod
MODIFICATION : Veuillez entrez l'ID du voyageur :
2
MODIFICATION : Veuillez entrez le nom du pays en minuscule :
allemande
Output from Server ....
{"idVoyageur":12,"nomVoyageur":"Douailly","prenomVoyageur":"Yann","pays":{"nomPays":"Allemagne","accesPays":"Test PCR à l'entrée"}, "modifications":[]}
Que voulez-vous faire :
ins : inscription
des : désinscription
inf : information
mod : modification des règles d'un pays

des
DESINSCRIPTION : Veuillez entrez l'ID du voyageur à désinscrire :
2
Voyageur n°2 inscrit.

Que voulez-vous faire :
ins : inscription
des : désinscription
inf : information
mod : modification des règles d'un pays

```

#### d) Serveur SOAP – Java 8+

IDE conseillé : Eclipse IDE (2020.06 ou inférieure pour compatibilité Java 8)

OS Compatible : Windows / macOS / Linux

Lancement : « Run project »

```

eclipse-workspace - Eclipse IDE

Client [Java Application] /Library/Java/JavaVirtualMachines/adoptopenjdk-8.jdk/Contents/Home/bin/java (Oct 7, 2020, 10:21:53 AM)
api.Data.java
----- Global COVID19 situation -----
NewConfirmed : 0
TotConfirmed : 0
NewDeaths : 0
TotDeaths : 0
NewRecovered : 0
TotRecovered : 0
----- End of report -----
----- Global COVID19 situation -----
NewConfirmed : 325379
TotConfirmed : 35469779
NewDeaths : 7003
TotDeaths : 1043874
NewRecovered : 238703
TotRecovered : 24739432
----- End of report -----
SERVEUR : http://localhost:7777/?wsdl

```

### e) Client SOAP – Java 8+

IDE conseillé : Eclipse IDE (2020.06 ou inférieure pour compatibilité Java 8)  
OS Compatible : Windows / macOS / Linux

```
<terminated> Client (1) [Java Application] /Library/Java/JavaVirtualMachines/adoptopenjdk-8.jdk/Contents/Home/bin/java (Oct 7, 2020, 10:24:50 AM - 10:25:10 AM)
2 Morts sur 24h : 7003
Bonjour,
Entrez 0 pour quitter.
Entrez 1 pour obtenir le nombre de nouveau cas sur 24h.
Entrez 2 pour obtenir le nombre de mort sur 24h.
Entrez 3 pour obtenir le nombre de personne soigné sur 24h
Entrez 4 pour obtenir le nombre total de cas.
Entrez 5 pour obtenir le nombre total de décès.
Entrez 6 pour obtenir le nombre total de personne soigné.

3 Soignés sur 24h : 238703
Bonjour,
Entrez 0 pour quitter.
Entrez 1 pour obtenir le nombre de nouveau cas sur 24h.
Entrez 2 pour obtenir le nombre de mort sur 24h.
Entrez 3 pour obtenir le nombre de personne soigné sur 24h
Entrez 4 pour obtenir le nombre total de cas.
Entrez 5 pour obtenir le nombre total de décès.
Entrez 6 pour obtenir le nombre total de personne soigné.

4 Cas total : 35469779
Bonjour,
Entrez 0 pour quitter.
Entrez 1 pour obtenir le nombre de nouveau cas sur 24h.
Entrez 2 pour obtenir le nombre de mort sur 24h.
Entrez 3 pour obtenir le nombre de personne soigné sur 24h
Entrez 4 pour obtenir le nombre total de cas.
Entrez 5 pour obtenir le nombre total de décès.
Entrez 6 pour obtenir le nombre total de personne soigné.

0 Merci et à bientôt!
```

### f) Client SOAP - Python

IDE conseillé : Visual Studio Code (latest)  
OS Compatible : Windows / macOS / Linux  
Lancement : “Run without debugging”

```
clientSOAPpython.py — Clients Python
clientSOAPpython.py > ...
clientSOAPpython.py > ...
1 import requests
good its in
Le nombre de cas confirmé est de :
325379
personnes
Oupsie ça fait beaucoup
Bienvenue sur Covid-WS, que voulez-vous savoir ?
1. Quels sont le nombre de cas confirmé?
2. Combien y a-t-il de nouveaux morts?
3. Combien y a-t-il de nouveaux soignés?
4. Quel est le nombre total de cas confirmés?
5. Quel est le nombre total de personnes qui sont soignés?
6. Quel est le nombre de total de décès?
Choisissez...
2
good its in
Le nombre de nouveau morts est de :
7003
personnes
Bienvenue sur Covid-WS, que voulez-vous savoir ?
1. Quels est le nombre de cas confirmé?
2. Combien y a-t-il de nouveaux morts?
3. Combien y a-t-il de nouveaux soignés?
4. Quel est le nombre total de cas confirmés?
5. Quel est le nombre total de personnes qui sont soignés?
6. Quel est le nombre de total de décès?
Choisissez...
3
good its in
Le nombre de nouveau soignés est de :
238703
personnes
Bienvenue sur Covid-WS, que voulez-vous savoir ?
1. Quels est le nombre de cas confirmé?
2. Combien y a-t-il de nouveaux morts?
3. Combien y a-t-il de nouveaux soignés?
4. Quel est le nombre total de cas confirmés?
5. Quel est le nombre total de personnes qui sont soignés?
6. Quel est le nombre de total de décès?
Choisissez...
```

#### Partie 4 : Conclusion

Nous avons réussi dans ce projet à mettre en œuvre un serveur REST d'application CRUD, ainsi que deux clients dans des langages différents.

Notre deuxième serveur, utilisant la technologie SOAP, dispose-lui aussi de deux clients fonctionnels dans des langages distincts.

Nous avons implémenté une application couche métier qui permet de recueillir des informations sur l'état de la crise sanitaire et des statistiques sur celle-ci.

Avec plus de temps, nous aurions probablement intégré une couche DOA avec une base de données minimale mais fonctionnelle pour stocker les informations des deux serveurs.

Cependant nous sommes déjà satisfaits de l'avancée du projet et de l'ensemble de ses fonctionnalités.