

DOCUMENTATION PROJET ARCHITECTURE LOGICIELLE

PHASE 2



Introduction

Le but de cette phase du projet est de récupérer le code déjà développé et le transformer en mode client serveur.

Procédure d'installation de Java 8 :

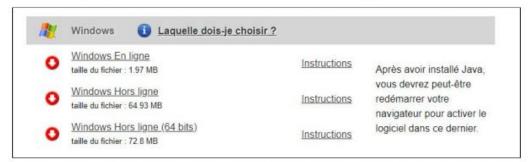
- Installation sur Windows : Télécharger la version de java sur la page d'accueil de java: https://www.java.com/fr/ puis suivre les étapes ci-dessous :





Etape 1

Etape 2



Etape 3





Etape 4

Installation Linux : Dans un terminal, vérifiez votre version de Java :

```
java --version
```

Si votre version Java diffère de 1.8.xxx, vous pouvez exécuter les commandes suivantes :

```
//installation du paquet
sudo apt update
sudo apt install openjdk-8-jdk -y
//changement de version
sudo update-alternatives --config java
```

Installation du projet :

Une fois l'environnement mis en place, il suffit de télécharger les fichiers zip du projet, de les extraire à l'emplacement désiré puis de démarrer un nouveau projet Eclipse : - importer le serveur comme un projet maven:

File->import->existing maven projects

Importer aussi le client principal comme un projet java :

File -> import>from existing projects

Ou utiliser le .jar

Pour la compilation du projet, il faut lancer le serveur:

Run as -> maven Build .

Et par la suite lancer le client principal comme un projet java:

Run as -> application java.

Ou utiliser le .jar

Une interface client s'ouvre et permet d'interagir avec le serveur.

Désinstallation du projet :

Pour désinstaller le projet il suffit de supprimer les projets du client et du serveur du works d'Eclipse (suppression dans le workspace d'Eclipse).

- Nouvelles exigences non fonctionnelles

Req-arc-08 : Le système est composé de deux parties : un client présentant l'IHM et un serveur gérant les données (priorité 1)

Req-arc-09 : Le client et le serveur peuvent être déployés sur deux machines distinctes (priorité 1)

Req-arc-10 : La communication entre client et serveur passe par une API REST (priorité 1)

Req-arc-11 : Le port d'écoute du serveur est paramétrable via un fichier dans un format standard (XML, JSON, YAML ou INI) (priorité 2)

Req-arc-12 : Le serveur utilisé par le client est paramétrable via un fichier dans un format standard (XML, JSON, YAML ou INI) (priorité 2)

Reg-arc-13: Le déploiement du serveur passe par une unique commande (priorité 3).

Reg-arc-14: Le démarrage du serveur passe par une unique commande (priorité 3).

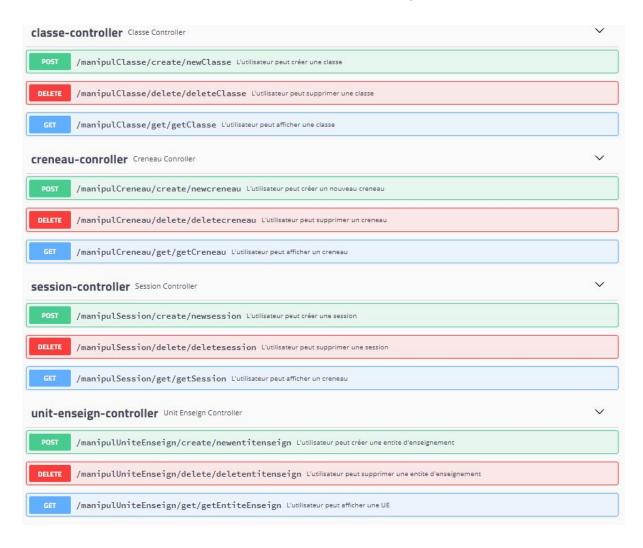
Req-arc-15 : L'arrêt du serveur passe par une unique commande (priorité 3). Req-arc-16 :

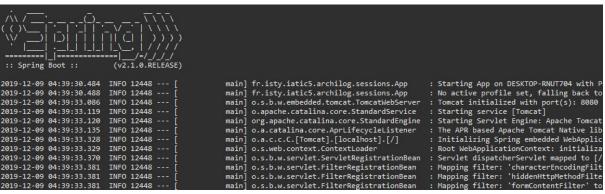
L'API REST utilisée est exprimée en OpenAPI (priorité 4).

Création de la partie serveur

Le framework swagger est utilisé pour respecter les exigences de l'API REST et de l'OpenAPI.

Le framework Spring Boot permet de démarrer rapidement le développement des services nécessaire en fournissant les dépendances et en auto-configurant celles-ci.





Création de la partie client

Malgré les possibles problèmes de redondance on a repris entièrement le projet existant et avons ajouté les requêtes HTTP et les différentes méthodes de REST directement là où été codé les requêtes SQL.

Les requêtes HTTP ce font par manipulation d'objet JSON

Pour les classes :

```
POST
{
  "id": "9f4af752-4e70-49fc-9fc8-d1924ffc3073",
  "promotion": 1,
  "filiere": "INFO",
}
       DELETE
  "UUID": "9f4af752-4e70-49fc-9fc8-d1924ffc3073",
}
       GET
  "id": "9f4af752-4e70-49fc-9fc8-d1924ffc3073",
  "promotion": 1,
  "filiere": "INFO",
 },
  "id": "f15ffe2a-b325-4a3d-81f6-b406e486ca58",
  "promotion": 1,
  "filiere": "INFO",
]
```

Pour les créneaux :

```
- POST
{
  "id": "f65e6835-312c-492a-9f74-71cdbfc454e6",
  "debut": "10:20:00",
  "fin": "10:21:00",
  "jour": "2016-10-08",
  "classe": null,
  "uniteEnseignement": null
}
   - DELETE
  "UUID": "f65e6835-312c-492a-9f74-71cdbfc454e6",
}
   - GET
[
  "id": "f65e6835-312c-492a-9f74-71cdbfc454e6",
  "debut": "10:20:00",
  "fin": "10:21:00",
  "jour": "2016-10-08",
  "classe": null,
  "uniteEnseignement": null
 },
  "id": "d07c5133-7c9c-4d4d-9010-37bc91e48eda",
  "debut": "10:20:00",
  "fin": "10:21:00",
  "jour": "2016-10-08",
  "classe": null,
  "uniteEnseignement": null
```

Pour les unités d'enseignement :

```
- POST
{
  "id": "1d4dd42e-ceb2-42b4-a4da-aaf9b39def66",
  "code": "1243",
  "intitule": "test",
  "cours": 1.5,
  "td": 1.5,
  "tp": 1.5,
  "valeur": 1.5,
  "classe": null,
  "creneau": null
}
   - DELETE
  "UUID": "1d4dd42e-ceb2-42b4-a4da-aaf9b39def66",
}
   - GET
[
  "id": "1d4dd42e-ceb2-42b4-a4da-aaf9b39def66",
  "code": "1243",
  "intitule": "test",
  "cours": 1.5,
  "td": 1.5,
  "tp": 1.5,
  "valeur": 1.5,
  "classe": null,
  "creneau": null
 },
  "id": "cf8ae061-08e5-4159-9e6c-684b9ac6d9fe",
  "code": "1243",
  "intitule": "test",
  "cours": 1.5,
```

```
"td": 1.5,
    "tp": 1.5,
    "valeur": 1.5,
    "classe": null,
    "creneau": null
}
```

Création du tunnel entre deux machines

Ngrok, c'est quoi?

Ngrok est un simple exécutable permettant la création de tunnel entre machines. Il est alors possible d'accéder à son environnement local depuis l'extérieur.

Téléchargement et utilisation d'Ngrok

Ngrok est téléchargeable depuis l'adresse URL suivante:

https://ngrok.com/download

Lancer simplement l'exécutable.

Pour se connecter à un port particulier du localhost, taper la commande suivante dans la console Ngrok:

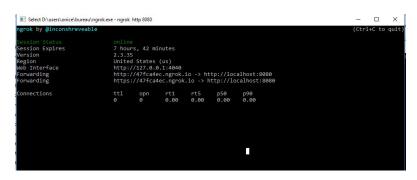
ngrok http <port>

<port> prend un numéro au choix.

Il est possible d'avoir un compte Ngrok.

Pour s'authentifier depuis la console Ngrok, taper la commande suivante:

ngrok authtoken <token>



Les URLs des requêtes HTTP ainsi que leur méthodes s'afficheront dans la console sous la ligne "Connections".