République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Constantine 2 – AbdelHamid Mehri

Faculté des Nouvelles Technologies de l'Information et de la Communication
Département des Technologies des Logiciels et Systèmes d'Information

---

Rapport

AL

Option : Génie Logiciel M2

Thème :

Conception et réalisation d'une application Web basée sur les micro-services

Dirigé par :
Mdm. Belguidom Meriem

Réalisé par :
Djouablia Mohamed Islam

Bouzebra Fouad Chaker

- L'année universitaire 2020/2021 –

1. Introduction :

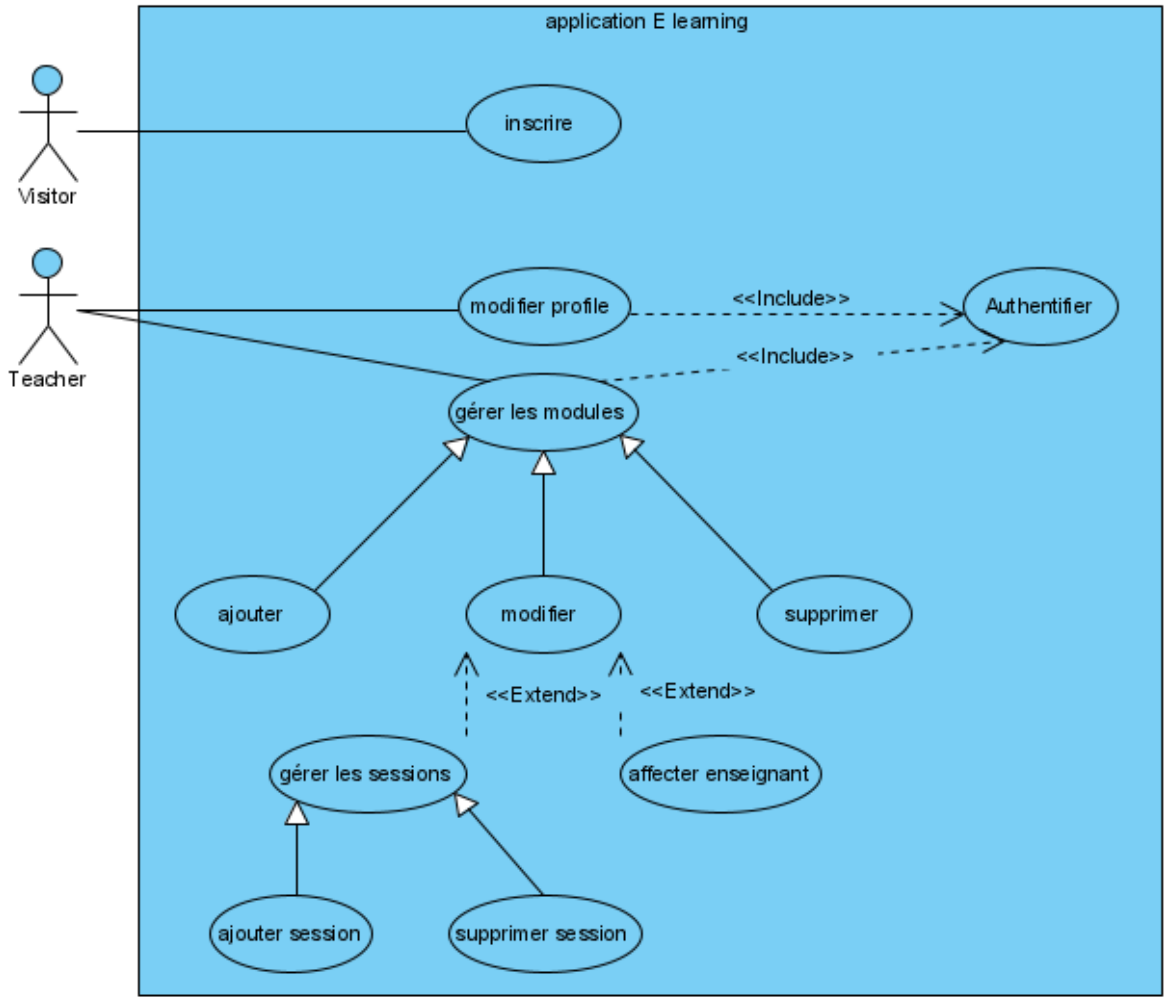
Les microservices sont un style d'architecture utilisé par de nombreuses organisations pour le développement de logiciels. Par le passé, l'industrie informatique utilisait des **solutions monolithiques** ou basées sur l'architecture orientée services (ou SOA pour Service-Oriented Architecture) comme standard.

Il existe un certain nombre de définition des microservices. La plus commune et généralisée reste celle de Martin Fowler qui dit :

« Le style architectural des microservices est une approche permettant de développer une application unique sous la forme d'une suite logicielle intégrant plusieurs services. Ces services sont construits autour des capacités de l'entreprise et peuvent être déployés de façon indépendante. »

Ce TP montre comment concevoir et programmer une application web représentant quelquesfonctionnalité d'une plateforme E-learning basée sur les micro-services.L'objectif de ce travail est de montrer comment créer plusieurs services indépendamment déployables qui communiquent entre eux, en utilisant les facilités offertes par Spring Cloud et Spring Boot. **Spring Cloud** fournit des outils pour les développeurs pour construire rapidement et facilement des patrons communs de systèmes répartis (tel que des services de configuration, de découverte ou de routage intelligent). **Spring Boot** permet de son côté de construire des applications Spring rapidement aussi rapidement que possible, en minimisant au maximum le temps de configuration, d'habitude pénible, des applications Spring.

2. Diagramme De Cas d'utilisation :



2.1. Description et représentation des scénarios :

La description d'un cas d'utilisation se fait par des scénarios qui définissent la suite logique des actions qui constituent ce cas.

La description d’un scénario peut se faire de manière simple, par un texte compréhensible par les personnes du domaine de l’application. Elle peut aussi être détaillée pour préciser les contraintes de l’acteur et celles du système, les différentes étapes ou actions avec leurs enchaînements et leurs concurrences.

Authentification :

Nom De Cas	**S’authentifier**
Type	**Primaire**
Acteur Principale	**Enseignant**
Objectif	**Permet à un membre ou à l'administrateur d'accéder à son propre espace.**
Pré-Condition	**L’utilisateur doit être créé dans la base de données et connaître ses identifiants.**
Post-Condition	**Permet à l'acteur d'accéder un son propre espace.**
Scénario nominal	**1. Le système affiche le formulaire d’identification.** **2. L’enseignant remplit le formulaire avec l’ensemble des informations nécessaires à son identification.** **3. Le système vérifie les informations saisies par l’utilisateur. Puis Le système affiche l'espace correspondant à l'acteur.**
Scénario Alternative	**1. L’enseignant n’as pas saisie les bons identifiants.** **2. Le système renvoie un message d’erreur et signale à l’utilisateur de recommencer.**

s’inscrire :

Nom De Cas	**S’inscrire**
Type	**Primaire**
Acteur Principale	**Visiteur**
Objectif	**Permet à un visiteur d'être un enseignant.**
Pré-Condition	**La Consultation de site Web**
Post-Condition	**La Creation d’une Compte Sur Le Site Web**
Scénario nominal	**1. le visiteur choisit de s'inscrire** **2. Le système affiche le formulaire correspondant** **3. le visiteur remplie le formulaire** **4. Le système vérifie les données saisies** **5. Le système affiche l'espace de l'enseignant**
Scénario Alternative	**1. Le Visiteur n’as pas saisie les bons identifiants.** **2. Le système renvoie un message d’erreur et signale à l’utilisateur de recommencer.**

Consultée Mes Modules :

Nom De Cas	**Consultée Mes Modules**
Type	**Primaire**
Acteur Principale	**Enseignant**
Objectif	**Permet La Consultation des Modules**
Pré-Condition	**Authentification**
Post-Condition	**L'affichage de la page My Courses**
Scénario nominal	**1. L'enseignant choisie le Button MyCourses.** **2. Le Systeme récupére les cours de l'enseignant.** **3. Le Systeme affiche les cours sur la page**

3.les BESOINS FONCTIONNELS ET DES BESOINS NON FONCTIONNELS :

3.1 les Besoins Fonctionnels :

Il s'agit des fonctionnalités du système. Ce sont les besoins spécifiant un comportement d'entrée / sortie du Système.

Le système doit permettre :

1. De s'identifier ;
2. Modifier le Profile ;
3. De consulter les Modules ;
4. De Gérer les Modules ;
5. De Gérer les sessions ;
6. S'inscrire.
7. Affecter Les Enseignants.

3.2 les Besoins Non Fonctionnels:

Il s'agit des besoins qui caractérisent le système. Ce sont des besoins en matière de performance, de type de matériel ou le type de conception.

Ces besoins peuvent concerner les contraintes d'implémentation (langage de programmation, type SGBD, de système d'Exploitation...).

1. La Fiabilité.
2. La Facilité D’utilisation (Utilisabilité).
3. Le Rendement et l’efficacité.
4. La Maintenabilité.
5. La Portabilité.

4- Conception De L’Architecture :

Selon le sujet propose on va choisir les microservices principales suivantes:

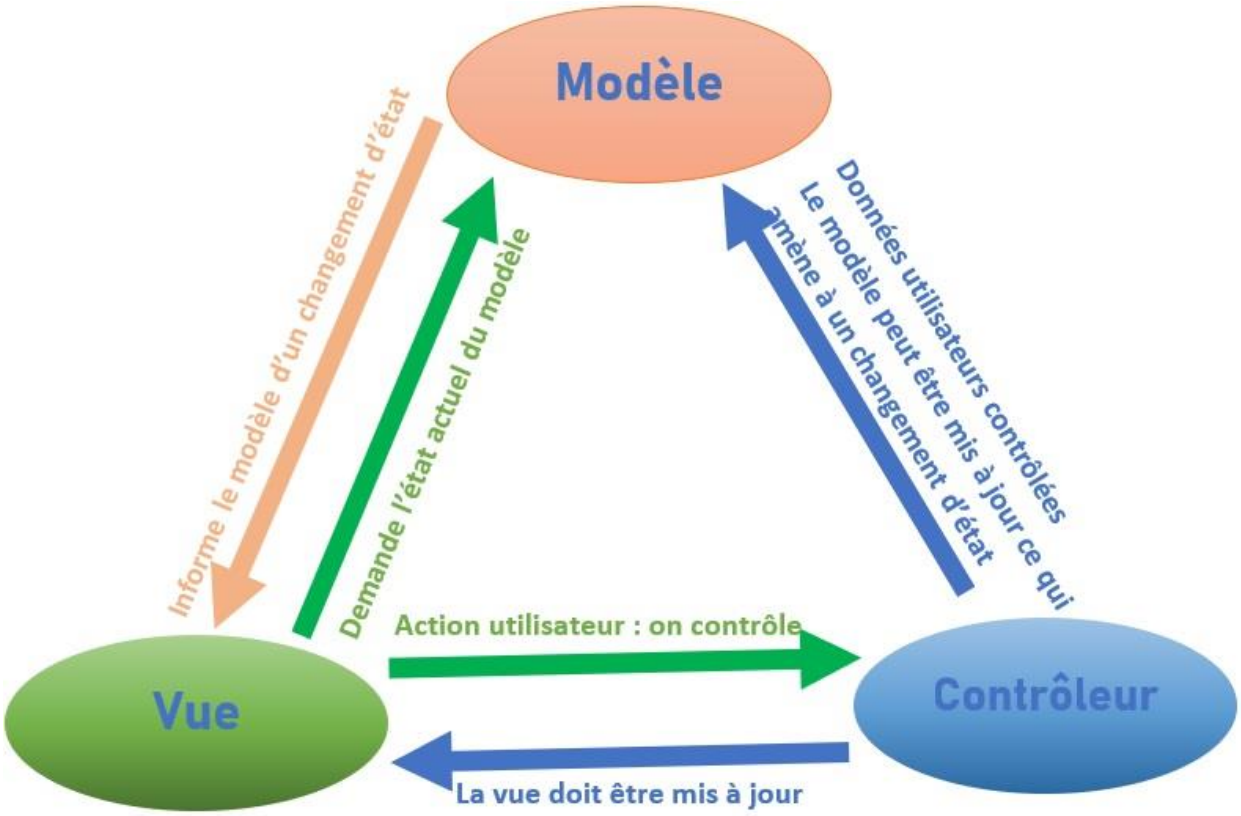
- **Microservice-cours** : qui offre une API REST pour lister des modules.
- **Microservice-enseignant** :qui offre une API REST pour lister des enseignants.
- **Config Service** : Service de configuration, dont le rôle est de centraliser les fichiers de configuration des différents microservices dans un endroit unique.
- **Proxy Service** : Passerelle se chargeant du routage d'une requête vers l'une des instances d'un service, de manière à gérer automatiquement la distribution de charge (ZUUL).
- **Discovery Service**: Service permettant l'enregistrement des instances de services en vue d'être découvertes par d'autres services (Eureka).
- **Microservice-Front-end** : permet d’afficher les pages web, et represent un client pour les autre API REST.

5- les patrons de Conception :

D’après notre conception on a les Patrons suivantes:

Le Patron MVC :

Le patron de conception **MVC** signifie Modèle Vue Contrôleur. Il permet de diviser son programme en trois parties indépendantes :
Le modèle : la logique de l'application.
Le contrôleur : interprète et formate les informations du modèle pour les passer à vue.
La vue : l'interface de l'application.
Le modèle et la vue ne peuvent JAMAIS communiquer.



Le Patron Proxy :

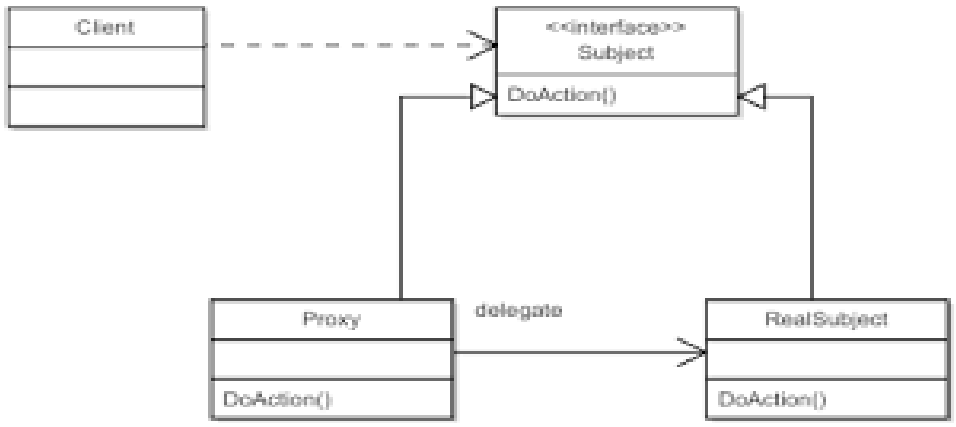
En programmation, un **proxy** (ou **délégation**) est un **patron de conception**.

Un proxy est une classe se substituant à une autre classe. Par convention et simplicité, le proxy implémente la même interface que la classe à laquelle il se substitue. L'utilisation de ce proxy ajoute une indirection à l'utilisation de la classe à substituer.

Un proxy est un cas particulier du patron de conception État. Un proxy implémente une et une seule interface (donc une seule classe). Un état peut implémenter un nombre quelconque d'interfaces.

Un proxy est utilisé principalement pour contrôler l'accès aux méthodes de la classe substituée. Un état est utilisé pour changer dynamiquement d'interface.

Outre l'utilisation principale du proxy (contrôle des accès), ce dernier est également utilisé pour simplifier l'utilisation d'un objet « complexe » à la base. Par exemple, si l'objet doit être manipulé à distance (via un réseau) ou si l'objet est consommateur de temps.



Le patron Injection De Dépandance :

L'injection de dépendance consiste à éviter une dépendance directe entre deux classes, et définissant dynamiquement la dépendance plutôt que statiquement.

Une classe A dépend d'une autre classe B quand la classe A possède un attribut de type B, ou possède une méthode utilisant la classe B (type de paramètre, valeur de retour, variable locale, appel de méthode de la classe B).

Pour mettre en œuvre l'injection de dépendance :

- Créer une interface I déclarant les méthodes de la classe B utilisées par la classe A ;
- Déclarer la classe B comme implémentation de cette interface I ;
- Remplacer toute référence à la classe B par des références à l'interface I ;
- Si la classe A instancie des instances de B à son initialisation, alors remplacer l'instanciation par un passage d'une instance de l'interface I au(x) constructeur(s) de A ;
- Si besoin, ajouter une méthode pour spécifier l'instance de l'interface I à utiliser.

Lorsque l'on développe un programme en java, afin d'éviter de tout avoir dans un seul fichier, le développeur écrit plusieurs classes (une classe par fichier). Cela permet non seulement d'assurer une meilleure lisibilité mais aussi une meilleure maintenance.

Seulement lorsque l'on passe en plusieurs classes, on souhaite pouvoir enlever une classe et la remplacer par une autre classe. Cela n'est malheuresement pas possible en java de base, il faut passer par de l'injection de dépendance. Cela permet entre autre "d'échanger" une classe par une autre et d'avoir plusieurs configurations d'application.

Il existe 4 types d'injections de dépendances :

- Injection par constructeur
- Injection par interface
- Injection par mutateur
- Injection par champs

Le concept fondamental du Spring est l'injection des dépendances (dependency injection).

chaque bean créé devient injectable via un fichier XML ou une annotation. Dans notre application on injecte les beans directement dans le XML et à travers l'annotation **@Autowired.**

6-les Technologies Utlisée :

7.1-java :

Java est un langage de programmation inspiré du langage C++, avec un modèle de programmation orienté objet.

Java permet de créer des applications complètes. Il peut également servir à créer un petit module d'application, dit applet, à intégrer dans une page Web.

7.2-Eclips :

Eclipse IDE est un environnement de développement intégré libre (le terme Eclipse désigne également le projet correspondant, lancé par IBM) extensible, universel et polyvalent, permettant potentiellement de créer des projets de développement mettant en œuvre n'importe quel langage de programmation. Eclipse IDE est principalement écrit en Java (à l'aide de la bibliothèque graphique SWT, d'IBM), et ce langage, grâce à des bibliothèques spécifiques, est également utilisé pour écrire des extensions.

La spécificité d'Eclipse IDE vient du fait de son architecture totalement développée autour de la notion de plug-in (en conformité avec la norme OSGi) : toutes les fonctionnalités de cet atelier logiciel sont développées en tant que plug-in.

Plusieurs logiciels commerciaux sont basés sur ce logiciel libre, comme par exemple IBM Lotus Notes 8, IBM Symphony ou Websphere Studio Application Developer.

7.3-Maven:

Maven est un outil de construction de projets (build) open source développé par la fondation Apache, initialement pour les besoins du projet Jakarta Turbine. Il permet de faciliter et d'automatiser certaines tâches de la gestion d'un projet Java.

Il permet notamment :

- d'automatiser certaines tâches : compilation, tests unitaires et déploiement des applications qui composent le projet.
- de gérer des dépendances vis-à-vis des bibliothèques nécessaires au projet.
- de générer des documentations concernant le projet.

8-Les Technologies Microservices Utilisée :

Spring est une plateforme construit pour le développement d'applications web en langage Java. Il a été introduit en 2004. En 2006, des sous-projets (sub-project) sont apparus. Chaque sous-projet se concentre sur un domaine différent. Jusqu'à présent, vous pouvez voir les sous-projets listés comme l'illustration suivante.

8.1-Spring Boot :

Spring Boot est un framework qui permet de démarrer rapidement le développement d'applications ou services en fournissant les dépendances nécessaires et en auto-configurant celles-ci.

8.2-Spring Cloud :

Spring Cloud est un Framework permettant de créer des applications cloud robustes. Spring Cloud fournit une solution aux modèles couramment rencontrés lors du développement d'un système distribué.

Pour fonctionner de concert dans un environnement distribué, ces microservices vont s'appuyer sur un ensemble d'outils proposés par Spring Cloud : une gestion centralisée da la configuration, la découverte automatisée des autres microservices, la répartition de charge et le routage d'API.

Intégrer ces différentes fonctionnalités Spring Cloud dans une application Spring Boot commence par la déclaration de starters Spring Cloud au niveau du pom.xml. Pour utiliser ces starters sans se soucier de leur version, un prérequis consiste à importer le BOM spring-cloud-dependencies (ex : le pom parent de Spring Petclinic Microservices) .

Il y a plusieurs technologies disponibles Sur Spring Cloud Pour Faire Face a certaines Probléme, Spring Cloud vous fournit certaines solutions technologiques comme ci-dessous :

8.2.1-Eureka :

Quand l'application répond à une montée en charge et que vous avez plusieurs instances de chaque Microservice, il est vital de pouvoir garder un registre de toutes les instances disponibles afin de distribuer la charge entre celles-ci.

Eureka de Netflix remplit précisément cette fonction. Une fois en place, les instances des Microservices viennent s'enregistrer dans le registre d'Eureka. Pour appeler un Microservice, il suffira de piocher dans cette

liste d'instances qu'Eureka expose via une API REST.

Eureka offre un client capable de réaliser des opérations de récupération des listes d'instances.

8.2.2-Zuul :

Zuul est un reverse proxy intégré à la stack Netflix OSS. Il s'agit donc du même type de brique que peuvent l'être Spring Cloud Gateway ou Nginx. Mais Zuul a un gros avantage sur ses « concurrents » : il permet la gestion de filtres (pouvant être enregistrés en base). En somme, ces filtres peuvent être utilisés pour réaliser des traitements sur les requêtes entrantes avant de les transmettre aux services sous-jacents. Le fait de pouvoir les stocker dans une base de données permet de modifier ces filtres sans devoir relancer le serveur (seule Cassandra semble supportée pour le moment).

8.2.3- Confi-Starter :

Spring Cloud Config fournit une prise en charge côté serveur et côté client pour la configuration externalisée dans un système distribué. Avec le serveur de configuration, vous disposez d'un emplacement central pour gérer les propriétés externes des applications dans tous les environnements. Les concepts à la fois sur le client et le serveur mappent de manière identique aux abstractions **Spring Environment** et **PropertySource,** de sorte qu'ils s'adaptent très bien aux applications Spring mais peuvent être utilisés avec n'importe quelle application fonctionnant dans n'importe quelle langue. Au fur et à mesure qu'une application se déplace dans le pipeline de déploiement du

développement au test et à la production, vous pouvez gérer la configuration entre ces environnements et être certain que les applications ont tout ce dont elles ont besoin pour s'exécuter lors de leur migration. L'implémentation par défaut du **backend** de stockage du serveur utilise **git**, il prend donc facilement en charge les versions étiquetées des environnements de configuration tout en étant accessible à un large éventail d'outils de gestion du contenu. Il est facile d'ajouter des implémentations alternatives et de les brancher avec la configuration Spring.

8.3-Spring Data :

Spring Data est un projet Spring qui a pour objectif de simplifier l'interaction avec différents systèmes de stockage de données : qu'il s'agisse d'une base de données relationnelle, d'une base de données NoSQL, d'un système Big Data ou encore d'une API Web.

8.4-Base De Donné H2:

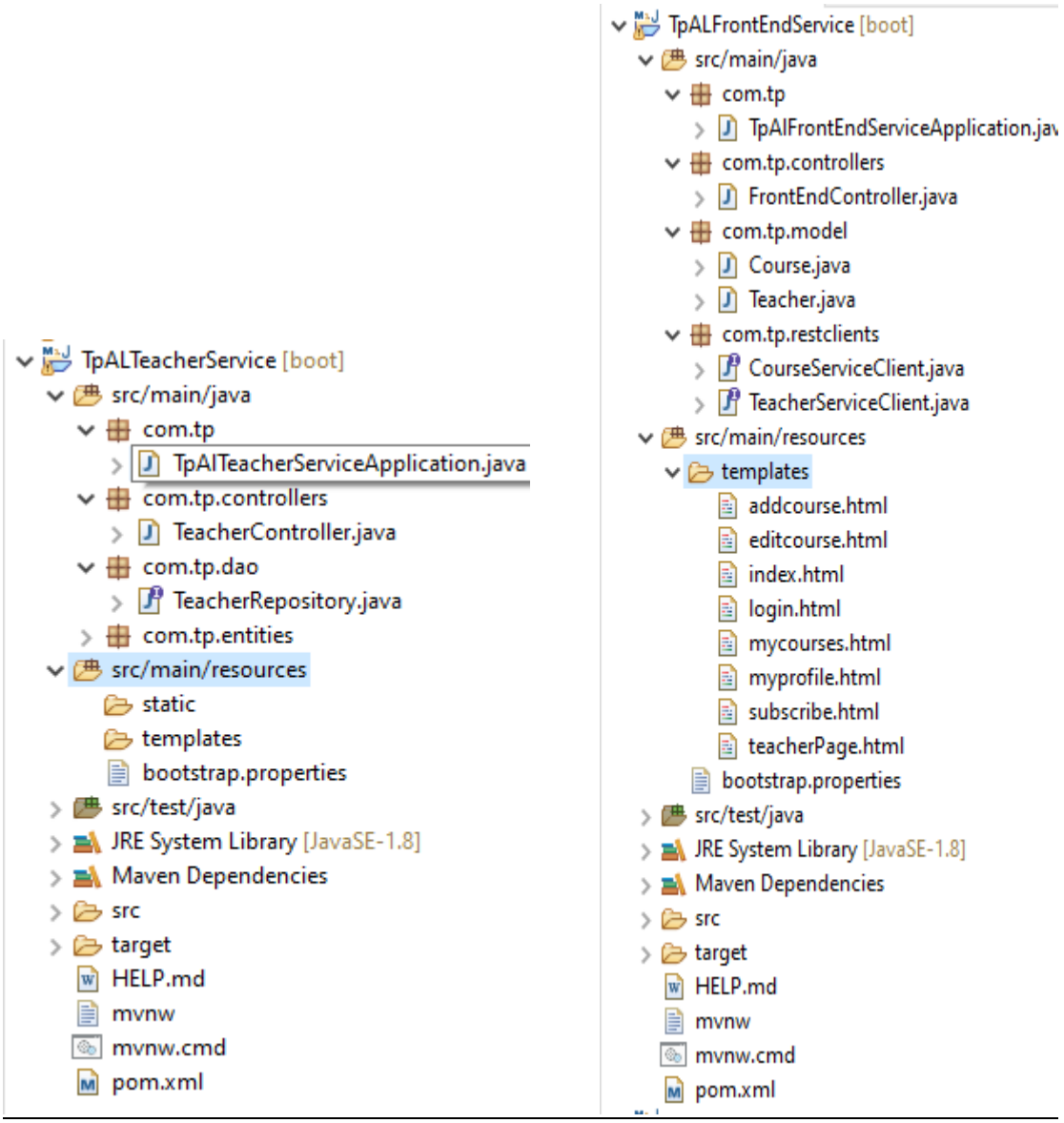
H2 est une base de données relationnelle (Relational database), source ouverte, compacte, écrite par la langage Java. La capacité de l'installeur H2 est très restreinte, environ 8MB.

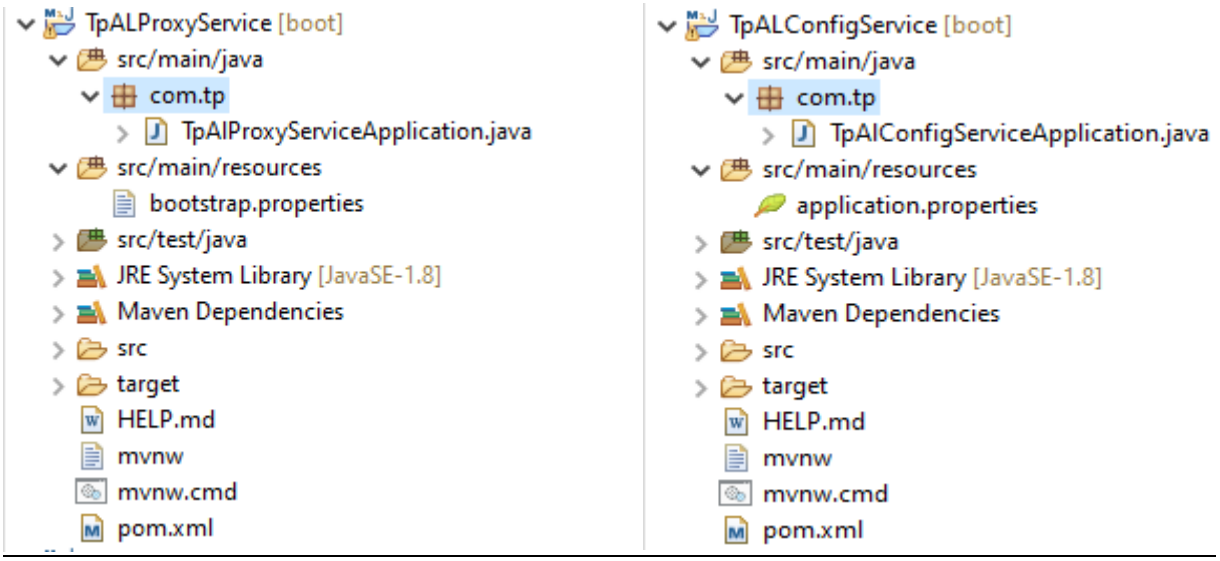
Une des caractéristiques intéressantes de H2 est que vous pouvez créer

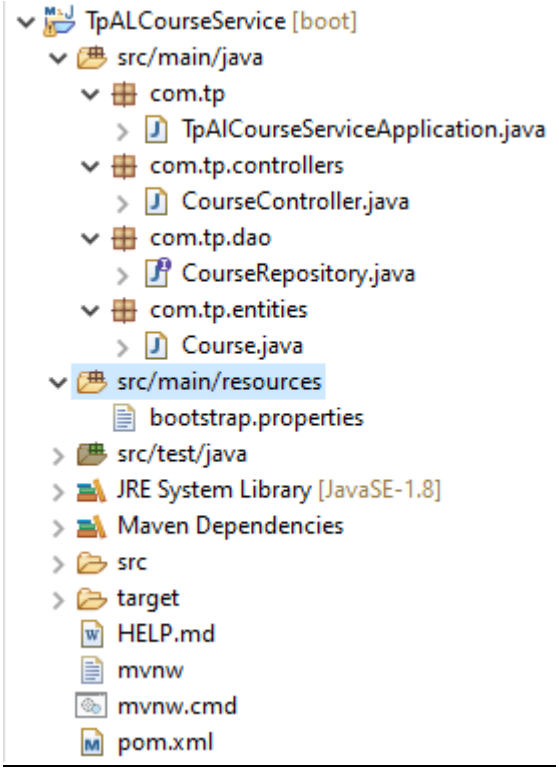
une base de données en mémoire (In Memory Database) au lieu d'être stocké sur un disque dur de l'ordinateur. Cela rend la requête rapide et la manipulation des données très rapide. Cependant, si vous sélectionnez la fonction "In Memory Database", les données n'existent que lorsque l'application fonctionne, lorsque l'application est arrêtée (shutdown), les données sont également supprimées de la mémoire.

9-Partie Implémentation(Code) :

9-1-Structre des Fichiers de l'application :




TpALProxyService [boot]
src/main/java
com.tp
TpAlProxyServiceApplication.java
src/main/resources
bootstrap.properties
src/test/java
JRE System Library [JavaSE-1.8]
Maven Dependencies
src
target
HELP.md
mvnw
mvnw.cmd
pom.xml
TpALConfigService [boot]
src/main/java
com.tp
TpAlConfigServiceApplication.java
src/main/resources
application.properties
src/test/java
JRE System Library [JavaSE-1.8]
Maven Dependencies
src
target
HELP.md
mvnw
mvnw.cmd
pom.xml


TpALCourseService [boot]
src/main/java
com.tp
TpAlCourseServiceApplication.java
com.tp.controllers
CourseController.java
com.tp.dao
CourseRepository.java
com.tp.entities
Course.java
src/main/resources
bootstrap.properties
src/test/java
JRE System Library [JavaSE-1.8]
Maven Dependencies
src
target
HELP.md
mvnw
mvnw.cmd
pom.xml

10-Bibliographie :

- https://www.jmdoudoux.fr/java/dej/chap-spring_core.htm
- https://fr.wikipedia.org
- https://openclassrooms.com/en/courses/4668216-optimisez-votre-architecture-microservices/5176545-externalisez-la-configuration-de-vos-microservices
- https://howtodoinjava.com/spring-cloud/spring-cloud-config-server-git/
- https://rgouzerh.files.wordpress.com