rapport de projet

Architecture logiciel



Projet TP : Conception et réalisation d’une application web pour E learning des M2GL  
basée sur les micro-services

Contents

[Cahier des Charges 2](#_Toc67694806)

[1. Description du projet: 2](#_Toc67694807)

[2. Besoins Fonctionnels: 2](#_Toc67694808)

[3. Besoins non Fonctionnels : 2](#_Toc67694809)

[Diagrammes de Cas d’utilisation 2](#_Toc67694810)

[Acteur : Enseignant 2](#_Toc67694811)

[Acteur : étudiant 3](#_Toc67694812)

[Acteur : Admin 4](#_Toc67694813)

[Fiches Descriptives 4](#_Toc67694814)

[Implémentation 5](#_Toc67694815)

[1. Architecture de l’application 5](#_Toc67694816)

[2. Hiérarchie du projet 5](#_Toc67694817)

[3. Outils utilisés 6](#_Toc67694818)

[4. Captures d’écran 6](#_Toc67694819)

# Cahier des Charges

## Description du projet:

Concevoir et développer une application web d’une plateforme E-learning en basant sur l’architecture du micro-services

## Besoins Fonctionnels :

Inscription et Authentification des enseignants.

- Ajouter/supprimer/ modifier un module.

- Ajouter/supprimer/modifier un cours/TP/TP d’un module.

## Besoins non Fonctionnels :

- Chaque module possède les caractéristiques suivantes : Nom, crédits, coefficient, volume horaire en cours TD et TP, les renseignements sur les enseignants qui lui sont affectés (Nom prénom et email) .

- L’enseignant ne peux manipuler ses modules, cours, etc. qu’après authentification.

# Diagrammes de Cas d’utilisation

## Acteur : Enseignant

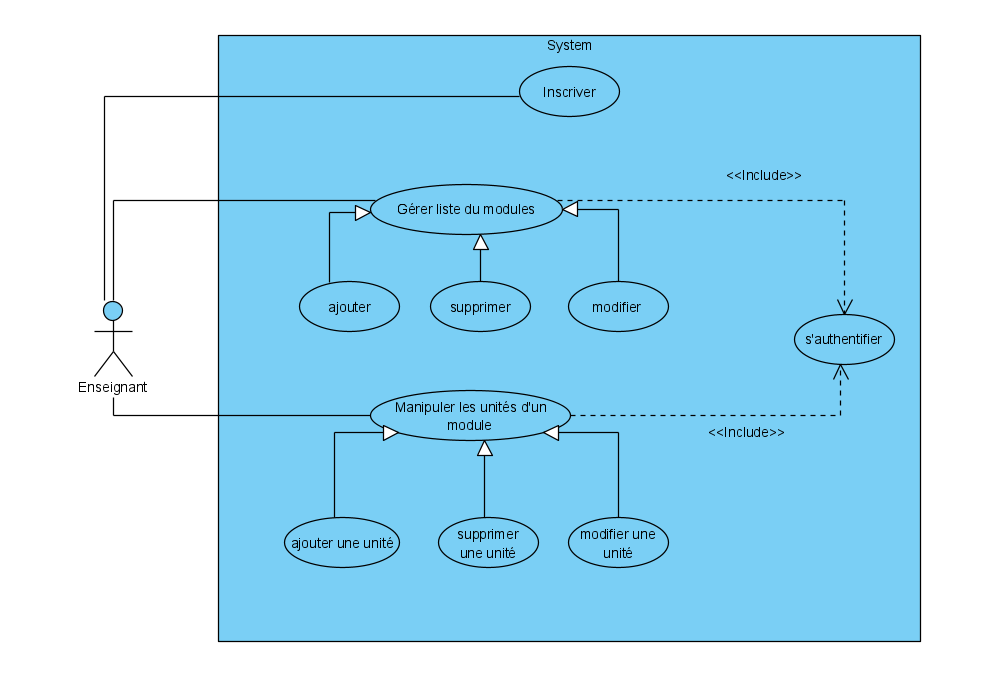


Figure 1 diagramme de cas d'utilisation Enseignant

## Acteur : étudiant

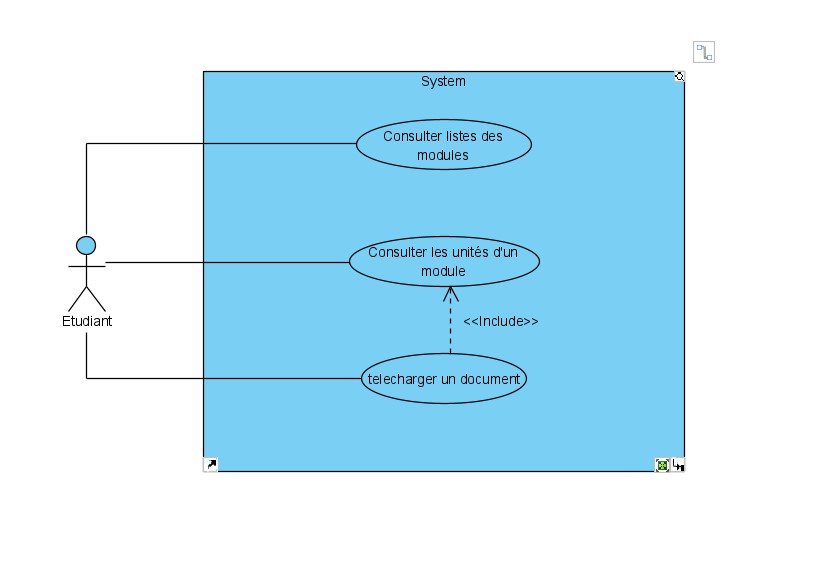


Figure 2 Diagramme de cas d'utilisation étudiant

## Acteur : Admin

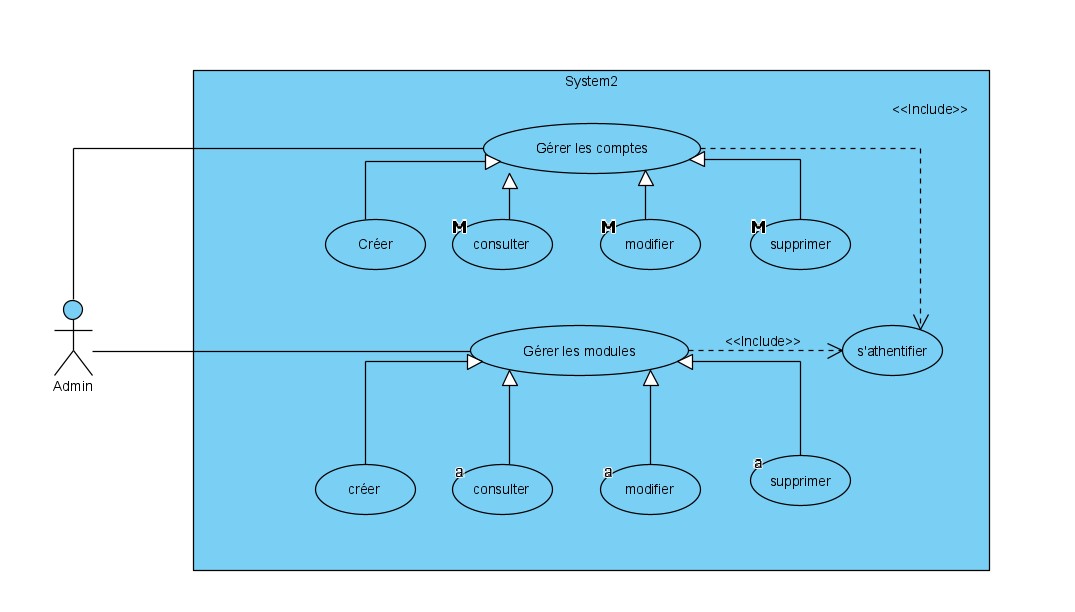


Figure 3 Diagramme cas d'utilisation Admin

# Fiches Descriptives

**Cas d’utilisation 1 :**

* Action : ajouter une unité à un module
* Type : primaire
* Acteur principal : Enseignant
* Précondition : s’authentifier
* Post condition : document ajouté
* Scénario nominale :
  + 1- L’enseignant clique ajouter un document
  + 2- L’enseignant fait la sélection d’un document au format « .pdf » et valide.
  + 3- L’enseignant remplit un formulaire contenant les détails sur le document à ajouter.
  + 4- L’enseignant valide l’opération et le document fut monté dans la plateforme.
* Scénario alternatif : fichier endommagé (retour à 2)

**Cas d’utilisation 2 :**

* Action : s’authentifier
* Type : primaire
* Acteur principal : Enseignant
* Précondition : /
* Post condition : authentification avec succes
* Scénario nominale :
  + 1- L’enseignant clique a rubrique login
  + 2- L’enseignant remplit le formulaire.
  + 3- authentification avec succes, redirecter vers acceuil
* Scénario alternatif : informations entrées sons faut (retour à 2)

# Implémentation

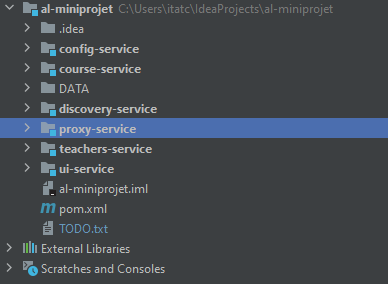
## Architecture de l’application

L’application est développée sur l’architecture des microservices qui structure une application comme un ensemble de services faiblement couplés. Les microservices indépendants communiquent les uns avec les autres en utilisant des API indépendantes du langage de programmation

Les micro services réalisés:

* Teacher/course micro services : RESTAPI for CRUD
* Micro service UI (user interface) : le service consommateur du micro service teacher/course
* Micro service de configuration : le rôle de ce service et de gérer et centraliser tous les configurations des autres services
* Micro service Discovery : le rôle de ce service et d’enregistrer les instances des autres services, ça sert comme un annuaire contenants les informations de l’ensemble des services en utilisants le service spring Eureka Discovery
* Micro service proxy : ce micro service fonctionne comme un passerelle (Gateway) pour filtrer et router les requetés HTTP vers les services concernés du manière automatique en utilisant le service Netflix Zuul

## Hiérarchie du projet



## Outils utilisés

* Spring boot : framework pour construire et définir l'infrastructure d'une application Java
* Spring security : framework Java / Java EE qui fournit l'authentification, l'autorisation et d'autres fonctionnalités de sécurité
* Eureka: Un service open source développé en Java par Netflix, dont le rôle est de gérer l’enregistrement et la localisation de services
* Netflix Zuul: Un service open source développé en Java par Netflix, Zuul va se positionner comme un passerelle de service, le point d’entrée unique des différentes services d’une application
* Base de données H2 : un système de gestion de base de données relationnelle (file based configuration).
* Netflix Ribbon : un outil pour faire l'équilibrage de charge.

## Captures d’écran

