# DOCUMENTATION DU PROJET MDE

Table des matières

[DOCUMENTATION DU PROJET MDE 1](#_Toc130522156)

[INTRODUCTION 3](#_Toc130522157)

[I. ANALYSE DU DOMAINE DE MODELISATION 3](#_Toc130522158)

[1. Entités principales identifiées 3](#_Toc130522159)

[2. Relations principales identifiées 4](#_Toc130522160)

[3. Exemple de modèle (FIL) 4](#_Toc130522161)

[II. CONCEPTION DU LANGAGE DE MODELISATION 4](#_Toc130522162)

[1. Ebauche de notre langage pour décrire le site 5](#_Toc130522163)

[2. Glossaire des concepts clés 6](#_Toc130522164)

[3. METAMODELE 7](#_Toc130522165)

[4. Contraintes OCL 7](#_Toc130522166)

[III. VALIDATION DU LANGAGE DE MODELISATION 10](#_Toc130522167)

## INTRODUCTION

Ce document présente le travail qui a été effectué dans le cadre de la réalisation d’un projet basé sur basé sur l’ingénierie dirigées par les modèles. Nous y présentons notamment l’analyse et la modélisation faite pour mener ce projet dans EMF.

## ANALYSE DU DOMAINE DE MODELISATION

Dans le contexte décrit de ce projet, le domaine de modélisation concerne la construction d'un langage pour modéliser des sites pédagogiques pour les universités qui présentent des formations et leurs contenus. Le site en question contient des informations sur les formations, les UEs, les ressources pédagogiques, les prérequis, les perspectives de débouchés, les responsables, les semestres, les niveaux, les spécialités, les semainiers (calendriers), et autres informations connexes. Le site est organisé en pages et chaque responsable prend en charge la description de ses pages en se basant sur des Template distribués pour avoir une certaine homogénéité. La réalisation du langage de modélisation est basée sur une approche d'ingénierie dirigée par les modèles en utilisant des diagrammes de classes UML pour représenter les entités et les relations entre elles. Le langage peut également être implémenté à l'aide d'API REST pour permettre l'enrichissement et la consultation des informations du portail sur différents plans.

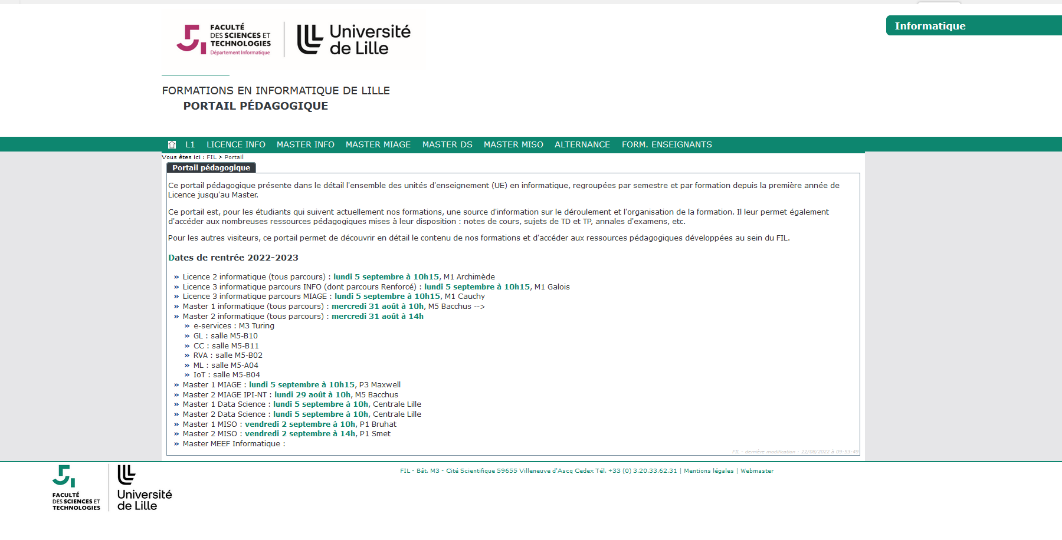
### Entités principales identifiées

* Formation : une formation proposée dans le portail pédagogique avec un nom, une description, une liste de niveaux, une liste de spécialités, une liste de semestres, etc.
* Niveau : il représente les différents niveaux (Licence, Master, Doctorat) associés ou non à des spécialités avec un nom, une liste de semestres, etc.
* Spécialité : une spécialisation possible pour un niveau donné (par exemple, Informatique, Biologie, Géologie).
* Semestre : un semestre pour un niveau donné.
* Unité d'Enseignement (UE) : une unité d'enseignement pour un semestre donné.
* Ressource pédagogique : un document ou un élément associé à chaque UE (slides, code, etc.).
* Responsable : une personne responsable d'une formation, d'un niveau, d'une spécialité, d'un semestre, ou d'une UE.
* Prérequis : représentant les prérequis nécessaires pour chaque UE.
* Débouchés : représentant les différents débouchés professionnels pour chaque formation.
* Semainier : représente un calendrier associé à un semestre avec les dates des différentes séances d'enseignement, les dates des examens, etc.

### Relations principales identifiées

* Un département offre plusieurs formations.
* Une formation a un ou plusieurs niveaux.
* Un niveau peut avoir une ou plusieurs spécialités.
* Un niveau a plusieurs semestres.
* Un semestre a plusieurs UE.
* Une UE a plusieurs ressources pédagogiques.
* Un responsable peut être associé à plusieurs entités (formation, niveau, spécialité, semestre, UE).
* Un Contenu de semainier est associé à une UE et à un Semestre

### Exemple de modèle (FIL)

****

## CONCEPTION DU LANGAGE DE MODELISATION

Pour concevoir le langage de modélisation, nous allons faire un langage personnalisé puis le formaliser en définissant un métamodèle après quoi nous ferons une formalisation des contraintes de modélisation à l'aide d'OCL.

### Ebauche de notre langage pour décrire le site

**Département** [Informatique] {

**Formations** {

[LicenceFonda] {

**Semestres** {

[Semestre1] {

**UEs** {

[Algorithmique] {

**Code** UE1

**Enseignants** {

Dr. Melat Paulin

}

**Ressources** {

cours1.pdf

cours2.pdf

}

**Programmation** [S1-UE1] {

Lundi 10h\_12h S08

Jeudi 14h\_16h S06

}

}

[Software Engineering] {

**Code** UE2

**Enseignants** {

Dr. Kimbi

Dr. Abessolo

}

**Ressources** {

slide1.pptx

syllabus.pdf

}

**Programmation** [S1-UE2] {

lundi\_10h\_12h S08

jeudi\_14h\_16h S06

}

}

}

}

[Semestre2] {

…

}

}

[LicenceICT] {

…

}

}

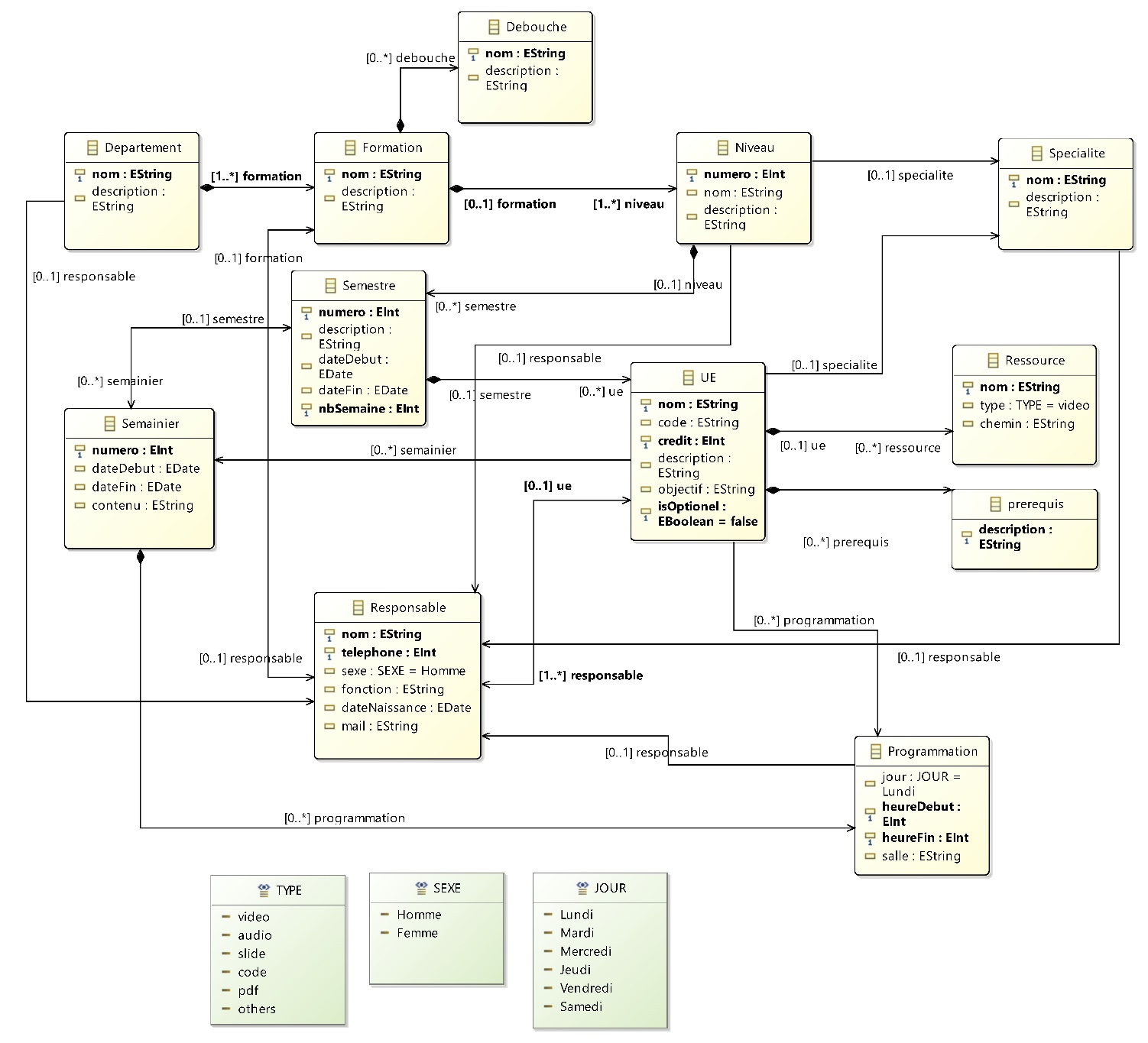
}

### Glossaire des concepts clés du langage de description du site

Après une analyse préliminaire du domaine, nous avons donc dresser le glossaire des concepts ci-dessous

|  |  |  |
| --- | --- | --- |
| Concepts | Propriétés | Documentation |
| Département | * Nom : String * Description : String | * Une Département dans le site. * Identifié par son nom et possède une description |
| Formation | * Nom : String * Description : String | * Une formation dans un département. * Identifié par son nom et possède une description |
| Débouché | * Nom : String * Description : String | * Un débouché d’une formation * Identifié par son nom et possède une description |
| Niveau | * Numéro : Entier * Nom : String * Description : String | * Un niveau dans une formation * Identifié par son numéro et possède un nom et une description |
| Spécialité | * Nom : String * Description : String | * Spécialité dans un niveau * Identifié par son nom et possède une description |
| Semestre | * Numéro : Entier * Description : String * dateDebut : Date * dateFin : Date | * Un semestre dans les niveaux des formations * Identifié par son numéro et possède une description, une date de début et une date de fin |
| UE | * Nom : String * Code : String * Crédit : Int * Description : String * Objectif : String * isOptionel : Boolean | * Une unité d’enseignement dans un semestre * Identifié par son code * Possède un nom, un crédit, une description, son objectif et un Boolean qui indique si elle est optionnelle ou obligatoire |
| Prérequis | * Description : String | * Prérequis d’une UE |
| Ressource | * Nom : String * Type : TYPE * Chemin : String | * Une ressource d’une UE * Identifié par son nom, possède un type (sa nature) et un chemin(url) pour le récupérer |
| Semainier | * Numéro : Entier * dateDebut : Date * dateFin : Date * Contenu : String | * Un semainier dans le semestre d’une formation * Identifié par son numéro * Possède une date de début, une date de fin et un contenu * Le contenu décrit le semainier |
| Programmation | * Jour : JOUR * heureDebut : Int * heureFin : Int * Salle : String | * La programmation d’une unité d’enseignement dans un semainier * Représentée par le jour, l’heure de début, l’heure de fin et la salle ou est prévu le cours * Chaque programmation d’une Ue est associé à un responsable |
| Responsable | * Téléphone : Entier * Nom : String * Mail : String * Sexe : SEXE * Fonction : String * dateNaissance : Date | * Une personne responsable d’un département(chef), d’une formation, d’un niveau, d’une spécialité ou d’une UE * Identifié par son numéro de téléphone ou par son mail * Possède un nom, un sexe, une date de naissance et une fonction |
| TYPE |  | Enumération des types d’une ressource : vidéo, slide, audio, code, pdf, autres. |
| SEXE |  | Enumération « *Homme » et « Femme »* |
| JOUR |  | Enumération des jours de Lundi à Samedi. |

### Métamodèle représentant le langage pour décrire les sites



### Contraintes OCL sur le métamodèle pour décrire les sites

* **Département**

**invariant** Named: **self**.*nom*->*notEmpty*();

**invariant** uniqueDepName: **self**.*oclType*().*allInstances*()->*forAll*(id1, id2 |*id1* <> *id2* **implies** *id1*.*nom* <> *id2*.*nom*);

**invariant** atLeastOneFormation: **self**.*formation*->*size*() > 0;

**invariant** UniqueFormationList: **self**.*formation*->*forAll*(f1, f2 |*f1* <> *f2* **implies** *f1*.*nom* <> *f2*.*nom*);

**invariant** auMoins1Responsable: **self**.*responsable*->*notEmpty*();

* **Formation**

**invariant** Named: **self**.*nom*->*notEmpty*();

**invariant** nomUnique: **self**.*oclType*().*allInstances*()->*isUnique*(*nom*);

**invariant** auMoins1Niveau: **self**.*niveau*->*size*() >= 1;

**invariant** toujoursDecris: **self**.*description*->*notEmpty*();

**invariant** niveauUnique: **self**.*niveau*->*isUnique*(n : *Niveau* | *n*.*formation*);

**invariant** niveauDistinct: **self**.*niveau*->*forAll*(n1, n2 |*n1* <> *n2* **implies** *n1*.*numero* <> *n2*.*numero* **and** *n1*.*nom* <> *n2*.*nom*);

* **Niveau**

**invariant** Named: **self**.*numero*->*notEmpty*() **and** **self**.*nom*->*notEmpty*();

**invariant** auMoins1Responsable: **self**.*responsable*->*notEmpty*();

**invariant** numeroSemestreUnique: **self**.*semestre*->*forAll*(s1, s2 | *s1* <> *s2* **implies** *s1*.*numero* <> *s2*.*numero*);

**invariant** associeFormation: **self**.*formation*->*notEmpty*();

**invariant** multiSpecialite: **self**.*specialite*->*size*() >= 0;

**invariant** associeSemestre: **self**.*semestre*->*notEmpty*();

**invariant** NotSpecialityWthDiffLevelWthSameName: **self**->*select*(n | *n*.*formation* = **self**.*formation*)->*select*(n | *n* <> **self**)->*forAll*(n | *n*.*specialite*->*select*(s | *s*.*nom* = **self**.*nom*)->*size*() = 0);

**invariant** semestreUnique: **self**.*semestre*->*isUnique*(s : *Semestre* | *s*.*niveau*);

* **UE**

**invariant** Named: **self**.*nom*->*notEmpty*() **and** **self**.*description*->*notEmpty*() **and** **self**.*isOptionel*->*notEmpty*();

**invariant** auMoins1Responsable: **self**.*responsable*->*size*() >= 1;

**invariant** plusieursRessources: **self**.*ressource*->*size*() >= 0;

**invariant** EnseignantsValides: **self**.*programmation*->*forAll*(enseignant\_prog | **self**.*responsable*->*includes*(*enseignant\_prog*));

**invariant** unSeulSemestre: **self**.*semestre*->*size*() = 1;

* **Specialite**

**invariant** Named: **self**.*nom*->*notEmpty*();

**invariant** nomUnique: **self**.*oclType*().*allInstances*()->*isUnique*(*nom*);

* **Responsable**

**invariant** UeDansFormation : **self**.*ue*.*semestre*.*niveau*.*formation* = **self**.*formation*;

* **Ressource**

**invariant** Named: **self**.*nom*->*notEmpty*() **and** **self**.*type*->*notEmpty*() **and** **self**.*chemin*->*notEmpty*();

**invariant** nomUnique: **self**.*oclType*().*allInstances*()->*isUnique*(*nom*);

**invariant** uneSeuleUe: **self**.*ue*->*size*() = 1;

* **Semestre**

**invariant** Named: **self**.*numero*->*notEmpty*() **and** **self**.*description*->*notEmpty*();

**invariant** numeroUniqueParNiveau: **self**.*niveau*.*semestre*->*forAll*(s | *s* <> **self** **implies** *s*.*numero* <> **self**.*numero*);

**invariant** AssocieNiveau: **self**.*niveau* <> **null** **and** **self**.*niveau*->*size*() = 1;

**invariant** nbSemaine3\_12: **self**.*nbSemaine* >= 3 **and** **self**.*nbSemaine* <= 12;

**invariant** NombreSeancesValide: **self**.*semainier*->*size*() <= **self**.*nbSemaine*;

* **Semainier**

**invariant** Named: **self**.*numero*->*notEmpty*();

**invariant** ContenuDefini: **self**.*contenu* <> **null**;

**invariant** ContenuNonVide: **not** **self**.*contenu*->*isEmpty*();

**invariant** DatesSeancesValides: **self**.*oclType*().*allInstances*()->*forAll*(s | *s*.*dateDebut* >= **self**.*semestre*.*dateDebut*);

**invariant** pourUnSeulSemestre: **self**.*semestre*->*size*() = 1;

**invariant** SemainierValideDate: **self**.*dateDebut* < **self**.*dateFin*;

* **Programmation**

**invariant** ContenuDefini: **self**.*jour* <> **null** **and** **self**.*heureDebut* <> **null** **and** **self**.*heureFin* <> **null** **and** **self**.*salle* <> **null**;

**invariant** HoraireValid: **self**.*heureDebut* < **self**.*heureFin*;

* **Prerequis**

**invariant** Named: **self**.*description*->*notEmpty*();

* **Debouche**

**invariant** Named: **self**.*nom*->*notEmpty*();

## CONCEPTION DU METAMODELE DES REST API PHP

Pour concevoir notre métamodèle cible pour générer les REST APIs en PHP, nous avons premièrement identifié quelques concepts clés puis nous l’avons formaliser à l’aide de UML dans EMF. Ecore le langage de modélisation, nous allons faire un langage personnalisé puis le formaliser en définissant un métamodèle après quoi nous ferons une formalisation des contraintes de modélisation à l'aide d'OCL.

### Glossaire des concepts clés PHP REST APIs

Après une analyse préliminaire du domaine, nous avons donc dresser le glossaire des concepts ci-dessous

|  |  |  |
| --- | --- | --- |
| Concepts | Propriétés | Documentation |
| Classe API | * BaseURL * Nom * Endpoint   Les méthodes :   * Get(), * Post(), * Put(), * Delete() | Classe API REST qui expose des ressources à travers des endpoints. |
| Classe Ressource | * Identifiant * Nom | Classe ressource qui est exposée par l'API REST |
| Classe EndPoint | * Url * HttpMethod * ResponseFormat * QueryParameters * ResponseBody   Les méthodes :   * addQueryParameter(), * addRequestBody() * addResponseBody() | Point d'entrée de l'API REST, qui permet d'accéder à une ou plusieurs ressources. |
| Classe Paramètre | * ParameterName * DataType * Description | Paramètre d'entrée d'un endpoint. |
| Classe Contrôleur | * EndpointsMethods() | Logique de contrôle qui gère les requêtes et les réponses de l'API REST. Cette classe contient des méthodes pour chaque endpoint de l'API, ainsi que des méthodes pour gérer les erreurs et les exceptions. |
| Classe Service | Méthodes pour accéder à la base de données et récupérer les données demandées par les Endpoints | Logique métier qui traite les requêtes de l'API REST. Cette classe contient des. |
| Classe Base de données | Tables pour chaque type de données du portail et les relations entre elles : *formations, Ues, Semestres, Niveau, Responsables, …etc* | Base de données MySQL qui stocke les informations du portail pédagogique. |
| Classe HttpMethod |  | Enumeration: GET, POST, PUT, DELETE |
| Classe DataType |  | Enumeration: STRING, INTEGER, BOOLEAN, FLOAT, DOUBLE, TEXT |

### Métamodèle cible des REST APIs

### Contraintes OCL du métamodèle des REST APIs

## VALIDATION DU LANGAGE DE MODELISATION