



Rapport de projet d’IAE



|  |  |
| --- | --- |
| Projet réalisé par  Jonas ELYSÉE et Paul FERRARON | Pour la date du  31/03/2017 |

**Table des matières**

[1. Rappel du sujet 1](#_Toc478768855)

[2. Réalisation du projet 2](#_Toc478768856)

[A. Outils utilisés 2](#_Toc478768857)

[B. Modélisation du problème 2](#_Toc478768858)

[C. EJBs 2](#_Toc478768859)

[D. Webservice REST 3](#_Toc478768860)

[E. Tests unitaires 3](#_Toc478768861)

[3. Configuration de l’application 4](#_Toc478768862)

## Rappel du sujet

Ce projet d’IAE (Intégration d’Applications pour l’Entreprise) devait être réalisé en binôme pour le 31/03/2017. Il consiste en la réalisation d’une application J2EE permettant la gestion d’un cabinet médical. En voici un schéma conceptuel :

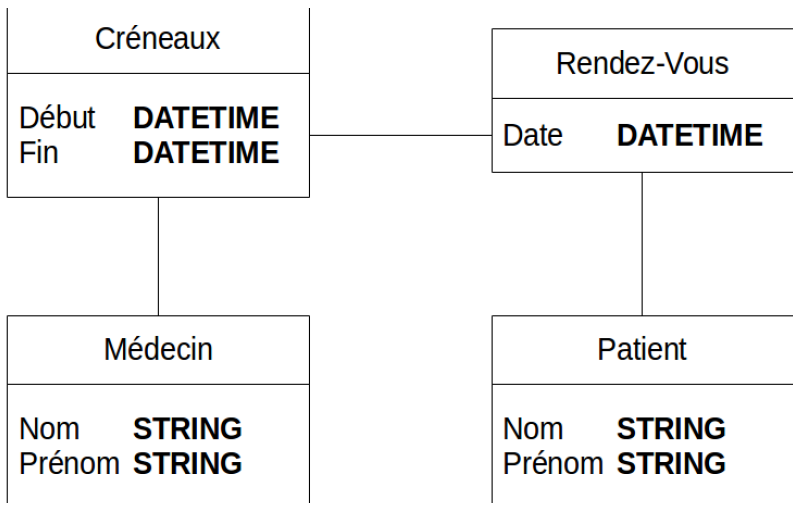


Figure 1 : Schéma conceptuel d'un cabinet médical

Pour le mener à bien, il fallait :

1. Proposer une modélisation sous la forme d’un ensemble d’entité JPA (Médecin, Créneau, RDV et Patient) ;
2. Créer des EJB s’appuyant sur l’EntityManager permettant de gérer le CRUD (Create, Read, Update & Delete) sur les entités Médecin et Patient ;
3. Créer des EJB permettant de lister les créneaux libres, prendre un rendez-vous, d’annuler et de modifier un rendez-vous ;
4. Exposer les différents EJB dans des Web Service REST (Comme dans les TPs précédents).

## Réalisation du projet

### Outils utilisés

Nous avons utilisé plusieurs outils pour mener à bien ce projet :

* NetBeans, le célèbre IDE (Integrated Development Environment) ;
* Payara, un server de type GlassFish sur lequel l’application se lance ;
* phpMyAdmin, le SGBD permettant d’interagir avec la BDD de l’application.
* Git & GitHub, respectivement le logiciel de gestion de version et le site web d’hébergement de fichiers sources.

### Modélisation du problème

Pour modéliser le problème, nous avons utilisé les 4 classes suggérées : Médecin, Créneau, RDV et Patient. De plus, nous nous sommes fixés quelques conventions :

* Chaque Médecin possède ses propres créneaux ;
* Chaque Créneau a la même durée (exemple 1h) ;
* Un Créneau ne peut accueillir qu’un RDV au maximum ;
* La date d’un RDV correspond à celle de début de son Créneau, d’où l’absence d’attribut Date dans la classe RDV ;
* Un RDV consiste en la rencontre d’un seul Médecin avec un seul Patient ;
* Un Patient fait référence à une personne et/ou une famille désirant avoir un RDV avec un Médecin ;

### EJBs

L’application comporte plusieurs EJBs (Enterprise Java Bean), 4 pour être exact, chacun destiné à des tâches bien précises :

* L’EJB gérant les CRUD de Médecin
* L’EJB gérant les CRUD de Patient
* L’EJB gérant les RDV (les prendre, les modifier, les annuler)
* L’EJB gérant les Créneaux

### Webservice REST

Comme demandé, nous avons exposé les EJBs dans un webservice REST.

Le webservice permet d’accéder aux différentes fonctions des EJBs en utilisant des commandes HTTP. Pour que ce service soit exploité à 100%, il serait nécessaire de faire un client complet (en angularJS par exemple) qui permettrait à un utilisateur de voir les différentes fonctionnalités.

### Tests unitaires

Le projet comporte des tests unitaires permettant de s’assurer du bon fonctionnement de chaque composant de l’application. Ils concernent donc les différentes entités, les différents managers et les différents services.

Ils passent pour la plupart, seuls quelques-uns sont en échec.

## Configuration de l’application

Pour pouvoir lancer l’application sur le serveur Payara, plusieurs étapes sont à suivre.

1. Importer le projet Maven dans NetBeans
2. Créer la BDD avec phpMyAdmin
3. Créer le serveur Payara dans NetBeans et le lancer
4. Exécuter la commande « asadmin multimode --file asadmin-commands.txt» dans un terminal ouvert dans le dossier bin du serveur Payara, en ayant modifié la 2ème ligne du fichier .txt avec les identifiant et mot-de-passe de la BDD
5. Exécuter l’application via NetBeans