# Base de donnée

# Postgresql

Les recherches menées pour obtenir une instance de base de donnée accessible de façons centralisée on pris beaucoup de temps. Nous avons fait des recherches sur des site comme ovh qui fourni un VPS (virtual private serveur) accessible depuis internet. Le problème c’est que les offres de ces sites ne sont pas suffisamment détaillée et les caractéristique technique insuffisantes. Chez ovh par exemple nous n’avons pas eu de détails exacte sur ce qu’est un core

Nous avons pris une année de test chez amazon pour un serveur cloud afin d’éberger une base de données centralisé, le serveur se trouve en Amérique et de ce fait les temps de latences sont considérables. Nous avons ouvert un ticker chez infomaniaque pour obtenir également une machine sur le cloud, il nous font une offre gratuite uniquement pour un mois. Nous avons donc décliné. Comme nous ne développons pas de services pour le logiciel il est impératif qu’un serveur dans l’intranet de l’école ou en suisse soit accessible car le trafic entre les clients et le serveur de base de donnée est conséquent et pas optimisé par l’intermédiaire d’un service. Comme nous accédons pas au données à travers un service web mais directement avec un Object Relational Maping intégrée dans les clients nous ne pouvons pas optimiser le trafic de données et limitée la quantité d’information qui transite entre le serveur et le client.

Nous avons réalisé un script postgresql à partir du model UML générer avec MySQL, nous avons choisi de simplifier le script au maximum pour ne pas avoir des soucis d’adaptation entre les deux systèmes de base de données. Etant donnée que nôtre schéma de base de données ne comporte pas des relation 1,1 à 1,1 ou 1,1 à 1,N ou 1,N à 1,N. Nous n’avons pas nom plus besoins d’activer ou désactiver des contraintes temporairement d’où le fait que nous ne les renommons pas comme nous n’allons pas travailler dessus.

# SQLite

Nous avons installer un browser sqlite pour générer une base de donnée d’après un script SQL et le tester avec Hibernate et JPA

Le browser SQLite que nous avons utilisé peut se télécharger à l’adresse suivante : <http://sqlitebrowser.org/>

Suit à l’utilisation de SQLite avec hibernate nous avons rencontrer plusieurs problème pendant la mise en place du projet template. Ces problèmes provenait essentiellement des librairies qui ne supportent pas la base de donnée. Suite à la consultation de la page web <https://stackoverflow.com/questions/17587753/does-hibernate-fully-support-sqlite> Nous avons choisi de prendre Derby pour notre projet, Derby étant développée directement en Java

# Projet template

La création d’un projet Template avec maeven nous à pris du temps, nous avons du importer toutes les dépendances afin qu’Hibernate puisse fonctionner correctement avec la version 9 de java. Les premier projet template n’était pas fait avec maven, cela afin de comprendre comment configurer les mapping et hibernate. Comme nous avons deux sources de données différentes PostgreSQL et Derby nous avons passé du temps à comprendre comment les intégrer au projet.

Avec netbeans l’intégration de la spécification JPA et hibernate se fait beaucoup plus aisément que sur intellj, netBeans met directement tous les fichiers de mapping, de configurations aux bonnes place avec toutes les dépendances nécessaire à la bonne utilisation d’hibernate.