Rapport de laboratoire

**Département de génie logiciel et des technologies de l’information**

|  |  |
| --- | --- |
| **No de laboratoire** | 3 |
| **Étudiant(s)** | François Gagné  Patrick Lavallée  Julien Pelletier-Morin  Simon Turcotte |
| **Code(s) permanent(s)** | GAGF20067801  LAVP12048408  PELJ03098909  TURS17118905 |
| **Cours** | GTI525 |
| **Session** | Hiver 2013 |
| **Groupe** | 01 |
| **Professeur(e)** | Éric Boivin |
| **Chargé(e) de laboratoire** | Mohamed Outellou |
| **Date** | 08/04/2013 |

Table des matières

[1 Introduction 3](#_Toc352017432)

[2 Conception 4](#_Toc352017433)

[2.1 Diagramme d’entité-relation 4](#_Toc352017434)

[2.2 Diagramme de classes 4](#_Toc352017435)

[3 Discussion 5](#_Toc352017436)

[4 Conclusion 6](#_Toc352017444)

# Introduction

Pour cette troisième itération du projet, l'équipe doit incorporer un système de gestion de la persistance des données. Après avoir précédemment fait un site web statique dans la première itération et l'avoir rendue dynamique dans la seconde, cette troisième itération permettra de finaliser le projet le rendant plus fidèle à une solution déployée dans un contexte professionnel.

Ce rapport présentera les techniques utilisées pour permettre de sauvegarder une transaction et d’afficher les données « *stubbées »* dans les itérations précédentes et migrées aux SGBD PostgreSQL. Pour y parvenir, le framework Hibernate sera utilisé pour faciliter la gestion de la couche de persistance.

De plus, un diagramme d’entité-relation est mis de l’avant pour démontrer le schéma de base de données utilisé. Ensuite, une discussion est fournie et explique les motivations de l’équipe à utiliser les patrons de conception Web implémenté ainsi que les difficultés rencontrées de même que les bons coups de l’équipe. Finalement, le document sera conclu.

# Diagramme d’entité-relation

Voici le schéma de base de données élaboré permettant d’illustrer la structure des tables de la base de données, leurs colonnes ainsi que leurs contraintes. La notation *Crow’s feet* est utilisée pour démontrer la cardinalité entre les entités. De plus, les relations *Many-to-Many* sont sauvegardées dans une table relationnelle avec une clé primaire composite assurant l’unicité de la relation. Les tables *Artistes\_Spectacle* et *Artistes\_Tags* sont un exemple de cette encapsulation et respectent la 3e forme de normalisation des données.

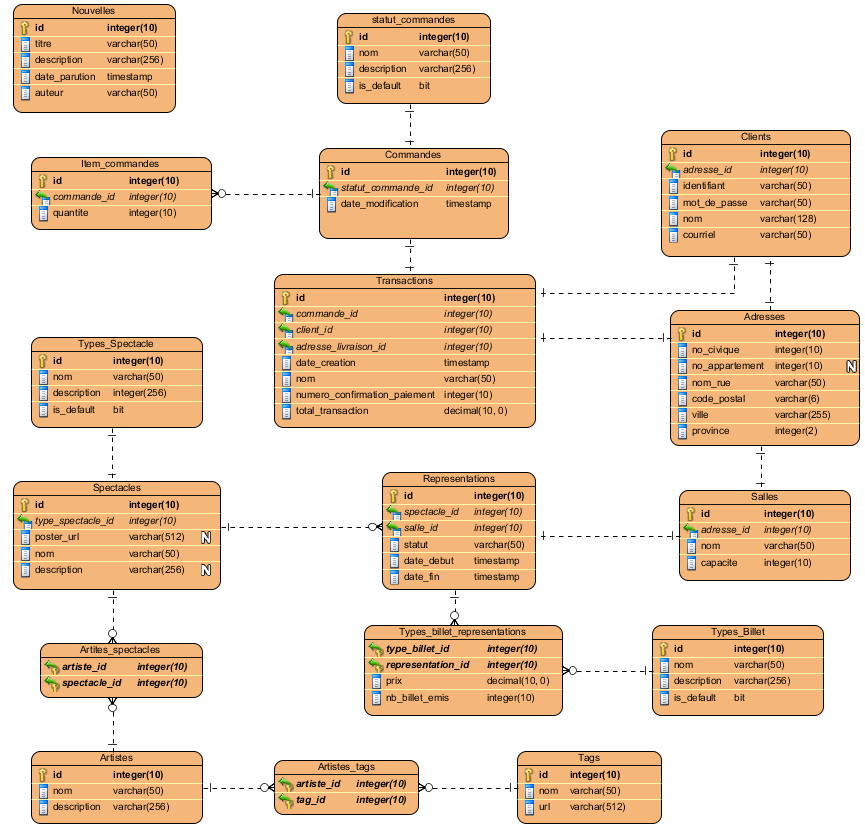


Figure 1- Schéma de la base de données

# Discussion

## Quels sont les patrons de persistance qui ont été utilisés dans votre application? Si vous avez utilisé un framework, quels patrons ont-ils permis d’implémenter?

L’utilisation du *framework* Hibernate permet l’utilisation de beaucoup de patrons de conceptions liés à la persistance des données. Certains sont utilisés sans efforts, c’est-à-dire que Hibernate les implémentes de base.

* DAO

Le patron Data Access Object rends disponible une interface abstraite permettant de rendre transparents les accès à la base de donnée. Cela permettra de rendre l'application évolutive en la séparant du type de base données utilisée pour la persistance.

* Query object

Le patron Query Object est implémenté en créant en java des objets représentant des requêtes SQL. De cette manière il est possible d'être certains de la syntaxe des requêtes. Ce patrons permet aussi de rendre l'application plus évolutive, en effet la création d'une classe par langage de base de donnée permet une meilleure séparation.

* Identity field

Le patron identity field est utilisé puisque l'application se sert du "ID" des objets de la base de données pour faire les appels. L'ensemble des objets ayant un "ID" unique, la correspondance des données est assurée.

* Lazy load

L'utilisation de ce patron permet de décider si l'on souhaite charger toute les données de l'application en mémoire au démarrage ou bien si l'on préfère que ces éléments soit chargé au fur et a mesure.

* Pool de connexion

Dans le fichier de configuration de l’application une limite de 5 connexions concurrentes à la base de données. Cela permet de mieux gérer la charge que la base de donnée doit pouvoir soutenir.

## Quel niveau d’effort a été nécessaire afin de passer d’un modèle basé sur les « stubs » au modèle avec persistance?

Afin de passer du modèle de persistance de l'itération 2 à celui de cette itération, il fallut tou t d'abords se trouver un serveur sur lequel hébergé notre base de donnée. La disponibilité d'un serveur d'un groupe étudiant duquel un des membres de notre équipe fait partie à facilité la décision quant à l'hôte de notre base de données. Par la suite, le choix du type de base de données s'est arrêté sur PostgreSQL pour sa facilité d'utilisation et son respect des standards SQL.

Ce qui demanda le plus d'effort en temps et en concentration fut la paramétrisation du fichier de configuration de notre application et l'ajout des annotations adéquates aux éléments du modèle.

## Quelles sont les principales failles de sécurité de votre application?

La faille de sécurité la plus présente dans notre application en est une qui se retrouve sur beaucoup de site web, c’est-à-dire la vulnérabilité aux attaques de dénis de service. En effet ces attaques sont difficiles à contrer puisque dans le cas d’une attaque faites par des ordinateurs zombies, les adresses IP des assaillants sont différentes. Il donc est difficile d’établir des règles pour bloquer ce type d’attaque.

Le mot de passe pour se connecter à la base de données est écrit en clair dans le fichier de configuration mais ce fichier n’est pas accessible des utilisateurs et c’est une pratique standard dans ce type d’application. Cela n’engendre donc pas de risques de sécurité.

## Quels moyens ont été mis en place afin d’éviter les failles de sécurité?

Il est possible d’ajouter une contrainte sur le nombre de connexions concurrentes en définissant un paramètre additionnel dans le fichier de configuration d’*Hibernate* telle que :

<prop key=*"hibernate.connection.pool\_size"*>5</prop>

De plus, en utilisant le *Data Access Layer* d’*Hibernate*, aucune logique SQL n’est définis au niveau des contrôleurs. En procédant de cette manière, le site n’est pas sujet à être victime d’injection de SQL malicieux qui viendrait corrompre l’intégrité des données.

## Avez-vous utilisé une technologie où un framework additionnel? Si oui, expliquez quel est le rôle de cette technologie dans votre application.

## Pour faire le lien entre les entités du modèles et leurs tables respectives dans la base de donnée, l'utilisation de Hibernate comme ORM à rendue cela possible. Hibernate permet d'utiliser des annotations dans les classes java qui sont par la suite traduite en éléments de base de données.

Voici des exemples d'annotations utilisés dans les classes du modèle:

@Entity

@Table()

@SequenceGenerator()

@Id @GeneratedValue()

@Column()

@one to many

@JoinTable

@LazyCollection

.Cascade.All();

Par la suite la création de la classe HibernateQueriesUtil.java rends disponibles des fonctions permettant de manipuler les entités de la base de données.

## Quelles modifications feriez-vous au service Web de paiement par carte de crédit?

## Pour le contexte académique de ce projet, le service fournit est amplement suffisant pour comprendre le fonctionnement des services web. Cependant une amélioration possible serait de rajouter un appel automatique à un autre service web qui permettrait de calculer le montant des taxes. Dans un concept plus réaliste, les transactions seraient originaires de plusieurs pays et un tel service permettrait de calculé les taxes sans erreurs.

## 

## Quelle a été, selon vous, l’élément le plus complexe à réaliser durant la session?

La partie la plus difficile de ce laboratoire était définitivement de passer d'un site statique à un site dynamique. En effet la première partie n'étant que la réalisation des pages web statique et la dernière étant grandement facilité par l'utilisation de la plateforme Hibernate, la dynamisation des pages et l'implémentation de la totalité des règles d’affaires fut un grand défi. En utilisant une solution axée sur l’utilisation de diverse plateforme de développement tel que *Spring* et *Hibernate*, le réel défi est de bien se documenter sur le fonctionnement des outils afin de bien les configurer.

# Conclusion

Le but de cette troisième itération consistait en l'ajout d'une couche de persistance à la solution développée lors des deux premiers laboratoires. L'utilisation du *framework Hibernate* à permit de faire le lien entre les classes du modèle et les tables de la base de donnée PostgreSQL.

L'utilisation des annotations est ce qui rend notre application et l'intégration de Hibernate aussi facile. De plus, les connaissances des membres de l'équipe dans les domaines à l’étude, c'est-à-dire le web et les bases de données, ont permit un développement avisé et structuré.

Suite à la finalisation de ce laboratoire, le résultat obtenu est une application web permettant de consulter du contenu obtenu dynamiquement à partir d’une base de données PostgreSQL hébergée sur un serveur disponible en permanence. Les requêtes HTTP sont traitées puis traduites en requêtes SQL à l'aide du *framework Hibernate*.

Finalement, cette itération conclue un projet de session qui a demandé un constant effort mais qui fut grandement formateur pour toute l'équipe. Dû au fait que certains des membres de l'équipe ne détenaient aucune notion de développement Web, ce projet permit de passer aux travers d'un grand nombre de notions élémentaires à notre formation.

# Références

<http://en.wikipedia.org/wiki/Entity%E2%80%93relationship_model#Crow.27s_Foot_Notation>