Rapport de projet BD6

David Galichet Baptiste Fontaine

13 mai 2012

Introduction

Le but du projet était de créer un système de gestion et suivi de colis à partir du catalogue de produits disponibles jusqu'à la livraison des produits chez le client, en passant par l'élaboration des commandes, l'emballage des produits dans des colis, le placement de ces colis sur des palettes, et le transport du tout. Il fallait également importer un fichier de données fictives sur les produits du catalogue et les noms des clients et des employés, ainsi que générer 250 commandes factices de façon aléatoire.

Choix

Techniques

Le langage de la base de données (PostgreSQL) et le langage utilisé pour l'interface (Java) était imposés. Nous avons choisi d'utiliser **Python** pour l'import des données initiales dans la base de données, et ce en utilisant une interface avec PostgreSQL directement dans le programme[1].

Pour la génération des commandes aléatoires[2], nous avons choisi d'utiliser **Java**, et d'appeler des méthodes d'interface avec la base de données, déjà utilisées dans l'interface du projet[3], et ce pour éviter la duplication de code. Lesdites méthodes ont été testées[4] une par une en utilisant des **tests unitaires**[5].

L'interface graphique a été développée en Java avec **Swing**.

Nous avons également **versionné** le projet, en utilisant **Git**, et en hébergeant le projet sur un dépôt privé Github.

[1] Sous Ubuntu/Debian, il est nécessaire d'installer le paquet `python3-postgresql`.
[2] fichier `java/GenerateCommands.java`
[3] fichier `java/ConnexionBDD.java`
[4] fichier `java/ConnexionBDD_tests.java`
[5] réalisés avec le framework *JUnit* (paquet `junit4` sous Ubuntu/Debian).

Structure

La structure de la base de données est définie dans le fichier `sql/create_db.sql`. Afin d'automatiser certaines opérations, le fichier, qui est inclu dans la base de données lors de sa création, défini un certain nombre de *triggers* (à la fin de celui-ci). La structure des tables a été choisie de façon à éviter la duplication d'informations.

Ainsi, chaque personne (client, gérant, douanier, employé) a une entrée dans la table `personne`. Les douaniers ont également une entrée dans la table `douane`, qui spécifie leur pays; tandis que les clients ont une entrée dans la table `client` qui spécifie leurs coordonnées. Le champ `personne.login` est utilisé comme identifiant unique pour une personne. Les détails des produits sont dans une table `catalogue`, et les commandes sont dans une table `commande`. Les produits d'une commande sont dans la table `commande_produits`. Lorsqu'une commande est emballée, des colis (table `colis` pour les détails, et table `colis_produits` pour associer un colis aux produits qu'il contient) sont créés. Il sont ensuite mis sur une palette (tables `palette` et `palette_colis`), puis celle-ci est mise dans un container (tables `container` et `container_palettes`). Enfin, lorsqu'une commande est livrée, tous les colis associés sont supprimés, mais le nom des produits commandés (table `commande_produits`) et les détails de la commande (table `commande`) sont conservés pour référence ultérieure.

Fonctionnement

Afin de tester plus facilement, nous avons choisi de tester en local avec une base `bd6`[6]. Pour préparer la base, il suffit de se rendre dans le répertoire `java`, puis d'exécuter la commande suivante :

```
$ make reset # supprime la BDD si elle existe, et la re-crée
```

Ensuite, pour générer les commandes, il suffit d'exécuter la commande suivante :

```
$ make generate
```

Le `Makefile` fourni propose d'autres actions, comme `clean` (supprimer les fichiers `*~` et `*.class`), `tests` (exécute les tests) ou `reset` (supprime la base de données, la re-crée, et re-importe `data.csv`).

[6] l'utilisateur et le mot de passe sont le nom de l'utilisateur courant

Interface

Un début d'interface graphique a été écrit, mais le temps ne nous a pas permis de la terminer. Une interface textuelle est néanmoins fournie (`java/Interface_texte.java`). l'interface se lance avec trois arguments: le nom de la base, le nom de l'utilisateur, et son mot de passe. Elle demande ensuite à une personne de se connecter (membre de la table `personne`), puis déduit du login le type de la personne et affiche ensuite l'interface adaptée.

Améliorations possibles

À cause des projets dans les autres matières et du peu de temps accordé pour élaborer ce projet, nous n'avons pas pu le terminer entièrement; même s'il est fonctionnel, il manque quelques fonctionnalités que nous n'avons pas pu implémenter, comme l'interface pour la douane, ou les containers (le transporteur livre directement les palettes, sans les mettre dans des containers).

Organisation

La structure de la base de données et l'architecture générale du projet ont été choisies conjointement. David s'est principalement occupé des interfaces avec l'utilisateur (graphique et textuelle), tandis que Baptiste s'est occupé de l'interface du programme avec la base de données (schéma et *triggers* PostgreSQL). L'écriture des méthodes d'interaction avec la base de données s'est faite à deux.