L3 DANT
Génie Logiciel LU3IN116

Documentation personnelle

et

choix de conception

IBuySU.com

Sommaire

Note : un mode d'emploi est disponible dans le fichier Readme à la racine du projet sur Github pour le lancement de l'application.
Le fichier IBUYSU.COM FINAL RENDU.ZIP contient les diagrammes UML réalisés sous Modélio.
La javadoc se trouve dans le dossier ./Javadoc.

Tâches effectués par Zeid Fazazi :

- Agrégation des diagrammes UML du groupe au sein d'un même projet Modelio, révision et actualisation des diagrammes
- Mise en place de la base de donnée SQL et des requêtes avec la bibliothèque JDBC sur les différentes tables
- Création des diagrammes de séquence de conception et d'analyse des Uses Cases "Evaluer" et "Enchérir"
- Répartition des tâches au sein du groupe de projet et organisation d'appels sur Discord
- Correction du code de plusieurs uses cases pour les rendre opérationnels

Notes sur les choix de conception et le projet dans son ensemble :

J'ai mis un accent sur la base de données en ligne afin de rendre l'application utilisable sur plusieurs machines en simultanés et maintenir une persistance sur les données des utilisateurs et des produits. Du fait des liens entre les classes, il a fallu mettre des contraintes sur des clés primaires et des clés étrangères pour garantir l'unicité et les relations entre les Produits, les Inscrits, les Évaluations… Cela s'est traduit aussi par des requêtes précises, faisant intervenir plusieurs tables à la fois.

Concernant la fonction "Évaluer" j'ai détaillée dans les diagrammes, j'ai préféré assigner les attributs d'auteur et de destinataire dans la classe Évaluation et utiliser le type Inscritpour englober à la fois les vendeurs et les acheteurs. Cela permet de mieux répartir les responsabilités, de ne pas léguer la gestion des évaluations sur les classes User et profiter de l'héritage User -Inscrit - Acheteur/Vendeur. Néanmoins, dans la base de données, cela crée pas mal de redondances sur les tables de la BDD, on a dû compenser avec pas mal de requêtes intriquées qui, dans une optique de scalabilité, n'est pas optimale du tout.

Plus généralement, le schéma de la BDD a souvent du mal à suivre la structure de notre projet Java, ce qui a nécessité pas mal de travail sur les tables et les requêtes.

Nous n'avons pas jugé nécessaire de les implémenter dans le code final du fait de la complexité du projet et de certains problèmes de communication dans la vie du projet. Les retards se sont cumulés et des choix ont dû être fait pour prioriser les fonctionnalités essentiels et fournir un produit viable, d'où l'absence de la fonctionnalité "Enchérir".

Aussi, étant la personne chargée de collecter les diagrammes du groupe et les recopier, j'ai pris contact avec l'ensemble du groupe, c'était l'occasion d'endosser le rôle de chef de projet et apprendre à s'adapter aux disponibilités et aux compétences de chacun, fixer les deadlines ou les reporter, compenser les manquements…

Tâches effectuées par Maëlys Delouis:

- Travail sur le diagramme de cas d'utilisation.
- Diagramme de classe de conception : ajout des paramètres des méthodes et du code en commentaire.
- Réalisation des diagrammes de séquence d'analyse et de conception sur les cas d'utilisation: Rechercher, Naviguer par Catégorie, Naviguer par Mot clef, Ouvrir un compte Acheteur, Ouvrir un compte Vendeur.
- Génération du code, implémentation de base de la classe IHM (système de menus et de formulaires), création de la base de données, implémentation de base de la classe API (mise en place de la connexion).
- Implémentation dans le système des fonctionnalités : Rechercher, Naviguer par Catégorie, Naviguer par Mot clef, Ouvrir un compte Acheteur, Ouvrir un compte Vendeur, Connexion, ainsi que se déconnecter et quitter le système.
- Mise en place des requêtes et des mises à jour de la base de données sur : Ouvrir un compte Acheteur, Ouvrir un compte Vendeur, Connexion, Création de Vente directe, Création de Mot Clef et de Catégorie.
- Rédaction du mode d'emploi.

Note sur les choix de conception :

Nous avons choisi de nous concentrer principalement sur les fonctionnalités du menu de base ainsi que la vente de produits.

L'IHM est assez peu développée dans le sens où elle aurait pu assurer une gestion des formats, mais il était plus simple de passer uniquement un tableau de String à une classe lors de la création d'une des instances de cette dernière : en effet, cela facilite amplement l'insertion dans la base de données tout en permettant de passer directement les paramètres de l'utilisateur dans les constructeurs.

Pour la classe système, nous avons employé un pattern singleton. Puisque la classe système IBuySu est le point d'entrée du package system, elle garanti

Nous avons également matérialisé une classe Utilisateur : cette dernière peu paraître accessoire, mais permet en réalité d'assurer la flexibilité de l'affichage. En effet, le système récupère à chaque fois que l'utilisateur souhaite utiliser une nouvelle fonctionnalité un menu. Au lieu de mettre un menu avec des conditions dans la classe système, nous avons trouvé

plus approprié de le mettre dans la classe Utilisateur, ainsi que ses sous-classes. Puisque le système garde dans son attribut user une trace du type d'utilisateur en cours, même lorsque ce dernier n'est pas connecté, nous pouvons appeler une unique méthode qui utilisera l'héritage pour récupérer le menu adéquat. Il en est de même pour toutes les fonctionnalités partagées par tous les utilisateurs. Enfin, ce choix rend la connexion et la déconnexion élémentaires.

Problème avec la classe API, le type de retour incorrect sur certaines fonctions car il aurait fallu mettre dans le diagramme de classe toutes les classes importées de java.sql, surchargeant donc un diagramme déjà dense et rendant la lecture difficile. C'est notamment le cas de tous les paramètres ou retours qui sont des instances des classes de cette bibliothèque (par exemple le retour de la fonction connexion est un objet de type Connection, et non un string, mais Modélio ne permet pas d'entrer un type absent des diagrammes).

A plusieurs points du diagramme de classe, nous avons décidé d'établir une double navigabilité : en effet, certaines de nos classes doivent permettre un accès dans un sens et dans l'autre. A titre d'exemple, la classe Catégorie référence de nombreux Produit, et c'est par cette dernière que s'effectue la recherche. Cependant, un produit doit également permettre de récupérer sa catégorie, c'est pourquoi la navigabilité entre ces deux classes est double. Il en est de même pour les MotClef, ainsi que les Contrats.

Nous aurions pu afin d'aller jusqu'au bout de la démarche d'achat et mettre en place un design pattern notify afin que les Acheteurs et Vendeurs soient notifiés directement dans leur menu lors de l'avancé d'une vente ou d'un contrat, de sorte à faciliter l'expérience utilisateur.

Tâches effectuées par Sofia Khali :

Nous avons chacun envoyé notre version du diagramme de cas d'utilisation pour ensuite décider d'une version finale prenant en compte le travail de chacun.

Je me suis ensuite occupé des diagrammes de séquence d'analyse puis de conception : seConnecter et seDeconnecter.

Pour seConnecter il a fallu réfléchir à la manière dont l'IHM communique avec IBuySu pour récupérer des informations.

L'utilisateur envoie une demande de connexion ce qui lance getListInscrit afin de récupérer la liste des personnes inscrites et leurs données.

Il envoie ses logs (mot de passe et ID). On vérifie ensuite que l'ID existe bien dans la base de données, si non on renvoit une erreur, si oui on vérifie si le mot de passe match bien

avec l'ID. On envoie un message d'erreur ou de succès à l'IHM qui affiche la page ou le message d'erreur.

Je me suis également occupée des diapos.

Tâches effectuées par Etienne Païta:

- Aide à la conception du diagramme de cas d'utilisation.
- Réalisation des diagrammes de séquence d'analyse et de conception sur les cas d'utilisation: Refuser la vente, Accepter la vente.
- Implémentation dans le système des fonctionnalités d'achat d'un objet, gestion des ventes d'un vendeur (accepter et refuser une vente), et modification de certaines fonctions en lien avec mes cas d'utilisation comme la fonctionnalité Rechercher.
- Participation à la javadoc

<u>Détail des cas d'utilisations de la gestion de vente (accepter/refuser vente):</u>

Un acheteur pourra gérer ces ventes, via l'IHM. Le système récupérera les ventes du vendeur et les affichera. Ce dernier pourra accéder au détail de ses ventes en les sélectionnant. Si une offre d'achat est disponible sur un des produits du vendeur, il pourra choisir d'accepter la vente ou de la refuser. S'il accepte la vente, le produit est vendu et le contrat conclu. S'il la refuse, le produit est laissé en vente et le contrat mis à null.

Tâches effectuées par Rashid ALIEV:

Pour les cas d'utilisation, nous avons décidé de faire la version de chacun et par la suite sortir le cas d'utilisation commune. Par conséquent, j'ai travaillé sur ma version de diagramme de cas d'utilisation. Ma contribution majeure a été la proposition de l'implémentation de l'utilisateur. A la fin, j'ai participé à la discussion et à la réalisation de la version finale de cas d'utilisation.

J’ai également participé à la réalisation de diagramme de classe.

J’ai aussi réalisé des diagrammes de séquence d’analyse et de conception sur les cas d’utilisation: vente directe et vente aux enchères.

Implémentation dans le système des fonctionnalités créer un vente directe, vente aux enchères. Modification des constructeurs, création et modifications certaines fonctions en lien avec les cas d’utilisation, comme la fonctionnalité: addOrCreatMotClef(), getCategorie().

Participation à la création de java doc.