Rapport de modélisation

Génie Logiciel

Projet de modélisation et d’implémentation

Simulateur pour un distributeur de billet de train





Table des matières

**Introduction4**

**Choix de conception4**

**Diagramme de cas d’utilisations1**

Description du diagramme2

Diagramme3

Description des cas d’utilisation3

**Diagramme global d’interaction4**

Description du diagramme5

Diagramme6

**Diagramme de classes4**

Description du diagramme5

Diagramme6

**Diagrammes de séquences4**

*Paiement*5

Description du diagramme6

Diagramme4

*Impression*5

Description du diagramme6

Diagramme4

*Acheter abonnement*5

Description du diagramme5

Diagramme5

*Renouveler abonnement*5

Description du diagramme6

Diagramme4

*Acheter pass*5

Description du diagramme6

Diagramme4

*Acheter billet*5

Description du diagramme5

Diagramme4

*Activer/désactiver composant optionnel*5

Description du diagramme6

Diagramme4

*Créer/gérer une panne*5

Description du diagramme5

Diagramme5

*Vérifier horaire trains*5

Description du diagramme6

Diagramme4

**Diagramme d’état4**

Description du diagramme5

Diagramme6

***Introduction :***

Voici le rapport de modélisation d’un simulateur pour un distributeur de billets de train, projet de Math-info de 2e année. Il décrira nos choix de conception, chaque diagramme un à un mais aussi une maquette de l’interface graphique.

***Choix de conception :***

* Le diagramme de cas d’utilisation représente non pas un simulateur de distributeur, mais bien un distributeur. Nous avons pris cette décision car il est ainsi plus évident de comprendre les réelles interactions entre les utilisateurs et le système. Etant donné que ce projet est en réalité un simulateur, le technicien et le client sont bien sûr la même personne (dans la suite, j’appellerai cette même personne l’utilisateur A MODIFIER DANS LES DIAG DE SEQ ?). En effet, celle-ci pourra à la fois interagir avec le distributeur, mais pourra aussi gérer elle-même les pannes (par exemple, recharger en encre et en papier le distributeur quand il n’y en aura plus). A la suite de ce diagramme, vous pourrez comprendre exactement les cas d’utilisation à l’aide des descriptions qui sont données. Le système bancaire, lui, sera en réalité (dans la suite du projet) la combinaison entre la base de données et le gestionnaire de base de données (dans le diagramme de classe, vous pouvez le voir apparaitre à travers la classe GestionBaseDeDonnees).
* Le diagramme de classe représente les classes principales et leurs associations. Une partie de leur comportement sera explicité dans la suite de ce document à travers les diagrammes de séquences. Nous avons décidé de représenter chaque type de titre de transport à l’aide d’une classe différente (car chaque titre est un objet). La classe abstraite TitreDeTransport qui généralise Billet, Pass et Abonnement sert à éviter la redondance des attributs communs (le montant à payer, les dates de validité et d’expiration, …). Ces titres de transports sont rassemblés avec la classe Recu (la preuve de paiement d’un titre) dans le package Imprimable, car ceux sont tous les objets que l’on peut imprimer.

Un autre package InterfaceGraphique rassemble ce qui sera représenté (voir maquette de l’interface graphique) visuellement lors de l’implémentation. On peut y trouver Reception (l’endroit où on peut récupérer son argent, ainsi que tout ce qui aura été imprimé), LecteurCarte (le lecteur de carte de crédit), Ecran (ce qui affichera les messages, ainsi que les différents menus, l’utilisateur va principalement être en interaction avec celui-ci), et finalement FenetreConfiguration et FenetreSimulation (la fenêtre de configuration va permettre à l’utilisateur de choisir le type de distributeur qu’il souhaite, et la fenêtre de simulation va être le simulateur du distributeur. Dans le dernier package Système, nous avons rassemblé toutes les classes qui touchent de près ou de loin à la logique du système. Il est composé de Controleur (comprend la logique principale du système), GestionBaseDeDonnees (qui permettra d’apporter des modifications à la base de données) et HoraireTrains (qui se chargera d’aller chercher les informations sur des horaires de train). Nous pouvons aussi y trouver PaiementLiquide (qui se chargera principalement de vérifier que l’utilisateur donne le bon montant lors d’un paiement liquide), Imprimante (qui se chargera, entre autres, de vérifier qu’il reste suffisamment d’encre et de papier pour imprimer un reçu ou un titre de transport) et Carte (cela représente les cartes de crédit, les données des cartes seront stockées dans la base de données). Finalement, la classe abstraite ComposantPanne est une généralisation des composants (du moins ceux qui sont représentés sous forme de classe) qui peuvent tomber en panne.

* Le diagramme Sortie de veille représente tout simplement le fait que lorsque l’utilisateur va appuyer sur le bouton « Démarrer », Ecran va afficher le menu principal (voir maquette de l’interface graphique).
* Le diagramme Acheter abonnement représente le cas où, dans le menu principal, l’utilisateur appuie sur le bouton « Acheter un abonnement ». S’en suivra un appel d’une méthode qui permettra à Controleur de connaitre le choix de l’utilisateur, afin qu’il crée une instance de Abonnement. Ecran affichera un nouveau menu où l’utilisateur pourra taper toutes les informations sur l’abonnement qu’il désire. *La suite est généralisée dans les diagrammes Paiement et Impression.* Lorsque toute la procédure est finie, ce nouvel abonnement sera ajouté à la base de données.
* Le diagramme Acheter billet représente le cas où, dans le menu principal, l’utilisateur appuie sur le bouton « Acheter un billet ». S’en suivra un appel d’une méthode qui permettra à Controleur de connaitre le choix de l’utilisateur, afin qu’il crée une instance de Billet. Ecran affichera un nouveau menu où l’utilisateur pourra taper toutes les informations sur le billet qu’il désire. *La suite est généralisée dans les diagrammes Paiement et Impression.*
* Le diagramme Acheter pass représente le cas où, dans le menu principal, l’utilisateur appuie sur le bouton « Acheter un pass». S’en suivra un appel d’une méthode qui permettra à Controleur de connaitre le choix de l’utilisateur, afin qu’il crée une instance de Pass. Ecran affichera un nouveau menu où l’utilisateur pourra choisir le type de pass qu’il souhaite (10 trajets, 10 trajets entre 2 gares prédéfinies ou pass illimité). Une méthode donnera l’information à Pass, qui demandera à Ecran d’afficher un nouveau menu, qui permettra à l’utilisateur de taper toutes les informations sur le pass qu’il désire. *La suite est généralisée dans les diagrammes Paiement et Impression.*
* Le diagramme Renouveler abonnement représente le cas où, dans le menu principal, l’utilisateur appuie sur le bouton « Renouveler un abonnement ». S’en suivra un appel d’une méthode qui permettra à Controleur de connaitre le choix de l’utilisateur, afin qu’il crée une instance de Abonnement. Ecran affichera un nouveau menu où l’utilisateur choisira s’il veut taper ou scanner le code de l’abonnement. S’il choisit de scanner le code, GestionBaseDeDonnees ira chercher tous les codes des abonnements existants dans la base de données, et Ecran les affichera. L’utilisateur pourra donc choisir un code parmi ceux affichés. S’il a choisi de taper lui-même son code, Ecran affichera une fenêtre où l’utilisateur pourra taper son code. Qu’il ait tapé ou choisi son code, la suite est similaire. GestionBaseDeDonnees va aller chercher les informations de l’abonnement en fonction du code, et Ecran va afficher un nouveau menu qui affichera l’abonnement en question et où l’utilisateur pourra taper les derniers détails du renouvellement. *La suite est généralisée dans les diagrammes Paiement et Impression.* Lorsque toute la procédure est finie, la nouvelle date d’expiration de l’abonnement sera mise à jour dans la base de données.
* Le diagramme Paiement représente non seulement le paiement d’un titre de transport, mais aussi la fin de la création de ce titre (cette décision a été prise afin d’évider une quelconque redondance). Il est important de savoir que le paiement ne se fait qu’à la condition que nombresImpressions soit supérieure ou égale à nombreTitre. Ce choix a été pris pour éviter que l’utilisateur paye sans pouvoir recevoir ses billets. Lorsque l’utilisateur aura tapé toutes les informations concernant son titre de transport, il va confirmer. A ce moment-là, la méthode preparation() va aller attribuer à chaque variable de titre de transport une des données tapées par l’utilisateur auparavant. GestionBaseDeDonnees va, lui, calculer le prix de ce titre de transport à l’aide de la base de données (qui stockera pareillement les réductions, les types de titres de transport, …), et ensuite attribuer cette valeur en tant que montantAPayer du titre. Ecran va alors afficher un nouveau menu où l’utilisateur pourra choisir son type de paiement. S’il choisit par carte, Controleur va créer une instance de Carte, et Ecran va demander à l’utilisateur d’insérer sa carte. Lorsque l’utilisateur appuie sur « Insérer carte », GestionBaseDeDonnees, va faire une recherche dans la base de données de toutes les cartes stockées, et Ecran va les afficher. Lorsque l’utilisateur va choisir une carte, LecteurCarte va passer son attribut carteInseree à vrai, et les informations sur la carte (code PIN, …) vont être ajouté à l’instance de Carte qui a été créée auparavant. Ecran va alors demander à l’utilisateur son code PIN, et tant que ça ne sera pas le même que celui de l’instance de Carte, il devra recommencer. S’il tape le code correctement, ça modifiera le solde sur sa carte et dans la base de données. S’il décide de payer en liquide, une instance de PaiementLiquide sera créée, qui aura comme attribut le prix du titre de transport. L’utilisateur va alors pour insérer des pièces ou des billets comme bon lui semble. Quand le montant reçu excède le montant à payer, l’argent donnée en trop sera rendue à travers Reception. Nous portons votre attention sur notre choix de « créer une instance de la classe abstraite TitreDeTransport ». Nous sommes bien conscients que cela n’a pas de sens dans la programmation même, mais cette décision a été prise afin de pouvoir généraliser le comportement de Paiement pour un Abonnement, un Billet, ou un Pass.

* Le diagramme Impression représente quant à lui le fait d’imprimer le titre de transport que l’utilisateur vient de payer. Ce dernier peut, s’il le désire, imprimer un reçu. A chaque impression, l’attribut nombresImpressions est décrémenté. Il est important de rappeler qu’il n’y a pas de soucis pour imprimer les titres (cfr paiement). Cependant, pour le reçu, il se peut que nombreImpression soit égal à 0, à ce moment là un message d’erreur est affiché et on revient à la page d’accueil.
* Le diagramme RechargerCaisse représente le technicien qui recharge la caisse. Ici les pièces et les billets sont représentés par 2 tableaux où chaque élément du tableau représente une valeur. Ils sont triés par ordre croissant, donc le premier élément de pièce représente le nombre de pièces de 1 cent. Donc RechargerCaisse consiste à actualiser le nombre de pièces et de billets que la machine possède. ViderCaisse représente quant à lui le fait de vider la caisse, on n’a donc plus de billets ; d’où le fait que tous les éléments du tableau passent à 0.

- Le diagramme RechargerNombresImpressions représente le technicien qui remet de l’encre et/ou du papier. Ici l’attribut nombresImpressions correspond à la capacité du distributeur d’imprimer des tickets ou des reçus car un distributeur a une capacité limitée en encre et en papier. Ici, nombreImpressions est cette limite. ViderNombresImpressions, quant à lui, consiste à passer cet argument à 0. Il est principalement là pour permettre à l’utilisateur de créer la panne.

- CréerUnePanne passe un attribut d’un composant appelé enMarche à faux. Cet attribut est la représentation du fonctionnement ou du dysfonctionnement du composant en question. Nous ne représentons pas ici les mises en panne des composants qui ne sont pas représentés sous forme de classe.

- GérerUnePanne passe l’attribut enMarche à vrai

* L’interaction overview diagramme représente le fonctionne typique du distributeur par un utilisateur. Il active la machine, choisi une fonctionnalité proposée par le distributeur, et une fois cette utilisation de l’appareil finie, le distributeur revient à la page d’accueil. Le diagramme comporte une répétition de paiement et d’impression car un achat comprend un paiement et une impression. Le but de cette répétition était d’insister sur l’importance que dans une utilisation normale, on a toujours un paiement et une impression.
* Ce diagramme représente le choix de l’utilisateur de vérifier l’horaire d’un train. Ecran va afficher une fenêtre où l’utilisateur pourra taper les informations sur le trajet qu’il désire. En fonction des données entrées, HoraireTrains va calculer les trajets, et Ecran va les afficher.
* Les diagrammes Activer/Désactiver ComposantOptionnel représente le choix de l’utilisateur de modifier FenêtreSimulation quand il désire. S’il coche/décoche une case dans le menu Composants optionnels, ça modifie l’attribut equipe dans FenetreSimulation, en passant à true/false à l’indice correspondant au composant en question.
* IL EST CERTAINEMENT IMPORTANT DE PRECISER QUE CES DIAGRAMMES REPRESENTENT LE CAS OU LE DISTRIBUTEUR EST DOTE DE TOUT ! SI CE NEST PAS LE CAS, CERTAINS MENUS N APPARAITRONT PAS, ET L UTILISATEUR N AURA DONC PAS LE CHOIX (à mettre dans le paragraphe « Choix de conception », pour ne pas le répéter à chaque fois)
* Dans les différents diagrammes de séquence, les liens qui lient le Client à Ecran ou à FenetreConfiguration sont le fait que le client appuie sur un bouton (i.e. à chaque fois, une méthode est lancée pour afficher ce qu’il faut).
* Dans le diagramme de séquence « AcheterPass », nous avons un fragment combiné de type « Alt » qui va représenter le choix de pass que va faire le Client.
* De la même manière, dans le diagramme de séquence « Paiement », le fragment combiné « Alt » qui va lui aussi représenter le choix de paiement du Client.
* Il en va de même pour l’impression, il peut choisir ou non d’imprimer un reçu.
* A chaque fois que le Client va lancer une panne, c’est le fait d’appuyer sur un bouton qui lui aussi va lancer une méthode. Il en va de même pour le choix des composants optionnels.