GSB-RV-DR

LE CONTEXTE

Le laboratoire Galaxy Swiss Bourdin (GSB) est issu de la fusion entre le géant américain Galaxy (spécialisé dans le secteur des maladies virales dont le SIDA et les hépatites) et le conglomérat européen Swiss Bourdin (travaillant sur des médicaments plus conventionnels), lui-même déjà union de trois petits laboratoires. En 2009, les deux géants pharmaceutiques ont uni leurs forces pour créer un leader de ce secteur industriel. L'entité Galaxy Swiss Bourdin Europe a établi son siège administratif à Paris. Le siège social de la multinationale est situé à Philadelphie, Pennsylvanie, aux États-Unis.

Cette situation professionnelle a été produite durant les heures de PPE. Ce projet porte sur le contexte Galaxy Swiss Bourdin (GSB). L'application GSB sera développée pour un laboratoire nommé Galaxy Swiss Bourdin (GSB) issu de la fusion entre deux groupes pharmaceutiques. Le laboratoire américain Galaxy et le groupe européen Swiss Bourdin. Leur activité de présentation des médicaments aux professionnels de la médecine génère des frais qui doivent être pris en charge par le service comptable du groupe. Cette prise en charge fait partie de l'image de marque des grands groupes pharmaceutiques.

L'objectif

Ce projet doit permettre la mise en place d'une application graphique (dite « de bureau ») pour le suivi des rapports de visite, appelé GSB-RV-DR. Celle-ci doit permettre aux délégués régionaux de consulter les rapports de visites ainsi que les praticiens hésitants.

Les contraintes

Au niveau environnement :

- L'application graphique dédiée au suivi des rapports de visite sera développée avec la bibliothèque JavaFX.
- MySQL est le SGBDR retenu, néanmoins le code produit doit pouvoir s'adapter à d'autres SGBDR.

Au niveau architecture :

- L'architecture du code produit doit respecter les conventions d'usage : développement basé sur des bibliothèques de fonctions, architecture MVC.

Au niveau module :

- Module Délégué : permet la vision de l'activité de chaque visiteur rattaché à une région.

Au niveau documentation :

- La documentation devra présenter le descriptif des éléments classes et bibliothèques utilisées, la liste des frameworks ou bibliothèques externes utilisés.

Au niveau responsabilités :

- Le commanditaire fournira à la demande toute information sur le contexte nécessaire à la production de l'application.
- Le prestataire est à l'initiative de toute proposition technique complémentaire.
- Le prestataire fournira un système opérationnel, une documentation technique permettant un transfert de compétence et un mode opératoire propre à chaque module.

Les cas d'utilisations

Numéro	Cas d'utilisation
1	S'authentifier
2	Consulter le rapport de visite d'un visiteur
3	Consulter la liste des praticiens hésitants

Environnement

L'application est développée sur une machine virtuelle sous Linux.Langage de développement : JavaFX.

Serveur : MySQL.

IDE (Environnement de Développement Intégré) : NetBeans 8.2.

Ma démarche

J'ai commencé par créer le projet *GSB-RV-DR* sur NetBeans (la classe principale est automatiquement crée, nommée « GSBRVDR »). Puis, je personnalise la fenêtre principale suivant le résultat attendu : Création de la barre de menus, du menu et de ses items. Je crée des écouteurs d'événements sur ces items en n'oubliant pas la gestion des états suivant la session (si elle est ouverte ou fermée).

Je commence le 1er cas d'utilisation : J'importe le script SQL qui contient la base de données déjà préalablement créée (nommé *gsbrv*) puis je l'exécute sur ma machine virtuelle (où se trouve le projet). Ensuite, J'importe le pilote JDBC (*mysql-connector-java-x.y.z.jar*) dans la bibliothèque du projet. Je crée des packages où figureront différentes classes (classes : *Session, Visiteur dans le package entité*), aussi j'importe entre autres des classes via la plateforme GitHub et je modifie leur code selon mon besoin. Ce sont des classes qui permettent la connexion d'un utilisateur, bien sûr je fais des tests pour m'assurer que mon code fonctionne. Je crée une boîte de dialogue qui sera la couche présentation, c'est-à-dire l'authentification (*texte, champs à remplir, boutons*). Avant de pouvoir rendre la navigation possible, après avoir cliqué sur un des menus, j'implémente 3 vues qui sont de type Pane (*panneau Accueil, Rapports, Praticiens*) puis je les ajoute à la classe principale de l'application sous forme d'attribut. Elles ont à ce stade juste but d'afficher un simple texte. A cette étape, le délégué régional saisit son matricule et son mot de passe et après un contrôle, le système active l'interface de cet utilisateur.

Je commence le 2eme cas d'utilisation : J'implémente une nouvelle classe (*Praticien)* et d'autres classes qui me permettrons d'avoir un critère de tri (*de comparer le coefficient de confiance, le coefficient de notoriété et la date de la dernière visite pour ma table de praticiens*). Je fais des tests pour m'assurer que les méthodes de la classe (*Praticien)* fonctionnent bien. Par ailleurs, j'ajoute différents éléments dédiés à l'affichage des praticiens (texte, boutons, table) dans le panneau *Praticien* ainsi que différentes méthodes. Je fais en sorte que cette table soit une liste « observable » de praticiens, c'est-à-dire que toute modification dans la base de données aura pour répercussion la modification du contenu de la liste. Puis, j'implémente des écouteurs d'événements sur ces comparateurs pour pouvoir les liés à la table. Enfin, dans ma classe principale, j'effectue différentes démarches pour que le critère de tri « coefficient de confiance » soit en premier lieu. A

cette étape la fenêtre en question de l'application est affichée en fonction des actions du délégué régional et de l'état de la session.

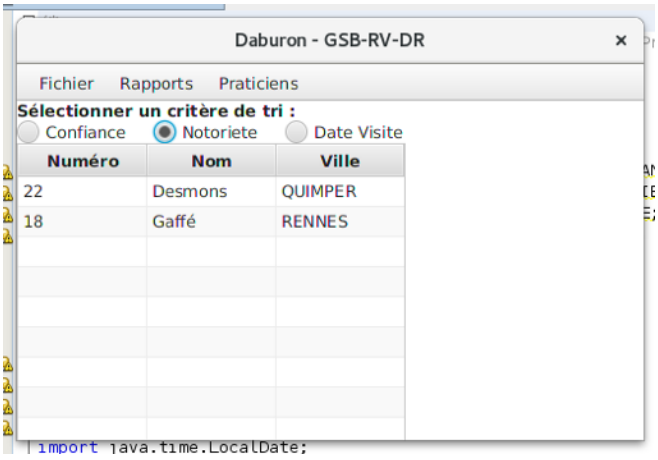
Je commence le 3eme cas d'utilisation : J'effectue d'abord quelques tests de requêtes SQL afin d'ajouter des méthodes dans différentes classes par la suite. J'implémente une classe (*RapportVisite*), et je fais des tests pour m'assurer que l'application fonctionne. J'ajoute différents éléments dédiés à l'affichage des praticiens (formulaires, bouton, table) dans le panneau *Rapport*. Puis j'implémente la liste des visiteurs, des mois et des années qui figureront dans un formulaire. Le but étant qu'une fois tout le formulaire rempli avec le bouton valider, la liste des praticiens selon les critères du formulaire s'affiche.
Pour finir, j'ajoute une méthode, lorsque l'utilisateur double clic sur une ligne du tableau, le rapport de visite en question est marqué comme lu, avec derrière une mise à jour de la base de données et ça affiche une boite de dialogue.

ANNEXE :

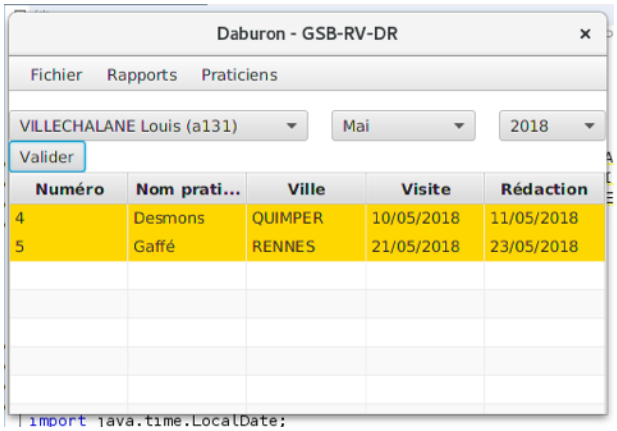
Accueil

Connexion

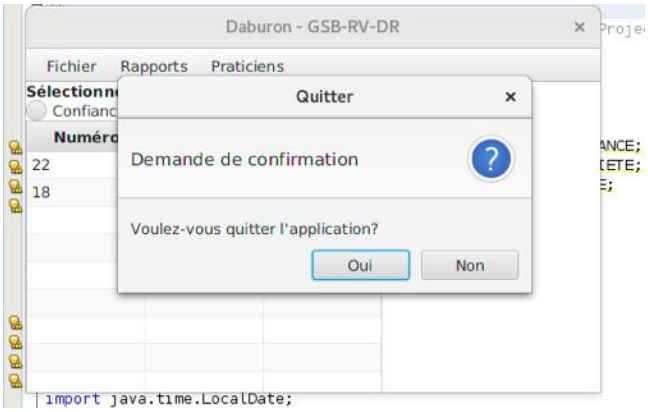
Consulter les praticiens hésitants



Consulter les rapports de visites



Quitter l'application



Code source sur mon github : https://github.com/ImaneBenach