le cnam
Bourgogne – Franche-Comté

Revue NFA019

ANNÉE 2020-2021

Étudiants :

RIBES Jérémy / MUTH Kévin

Revue :

20 / 06 / 2021

Formation Cnam :
Licence Sciences Technologies Santé (L2), mention informatique générale – Cnam

Projet : Élaboration d'une application de réservations de salles pour des réunions en JAVA

Acteurs du projet

Les élèves :

Nom et Prénom : RIBES Jérémy / MUTH Kévin

Courriel : jeremy.ribes.auditeur@lecnam.net / mailto:kevin.muth.auditeur@lecnam.net

Le projet :

Nom du projet / groupe : Booking

Le professeur responsable :

MARTINEZ Hervé

Table des matières

Description générale de l'application

1.1 Perspective de l'application

Cette application s'inscrit dans le domaine de l'entreprise dans le cadre de l'optimisation de réservations de salles de réunion en interne. Si une personne étrangère souhaite réserver, il sera nécessaire de passer par une secrétaire. Celle-ci se trouve en dehors de toute interface, de toute base de données et de tout logiciel déjà existant et sera développé avec des composantes simples et efficaces.

1.2 Vue d'ensemble des fonctionnalités du produit

Cette application distingue plusieurs fonctionnalités :

- On souhaite mettre en place un logiciel multi-utilisateurs avec une interface graphique disposant de toutes les fonctionnalités nécessaires à la création, la maintenance et l'utilisation d'une organisation, de ses salles et de ses utilisateurs.
- Le logiciel devra afficher la semaine de travail en cours avec les plages disponibles et réservées pour 1 salle, devra permettre de passer d'une salle à une autre et offrira la possibilité de naviguer dans les semaines.
- On devra disposer d'accès différenciés par type d'utilisateur, sécurisés par mot de passe.
- On devra disposer d'outils de recherche de disponibilité avancés en fonction de critères comme une date, une durée, une taille de salle …
- On souhaite pouvoir obtenir des données statistiques sur l'utilisation des différentes salles et l'activité des utilisateurs.
- On souhaite pouvoir affecter un délai de rétractation depuis un compte administrateur

1.3 Caractéristiques des utilisateurs

Cette application distingue 3 types d'utilisateurs :

- **Des utilisateurs simples** pouvant être invités à une réunion, autorisés à consulter le planning de réservation mais non habilités à réserver. Ceux-ci peuvent aussi voir qui a réservé, avec quel matériel et le nombre de personne(s).
- **Des utilisateurs standard** habilités à réserver une salle, à modifier ou supprimer leurs propres réservations et à inviter d'autres utilisateurs.
- **Des administrateurs** :
    - Ils disposent des mêmes droits que l'utilisateur standards mais peuvent aussi agir sur les réservations d'un autre utilisateur (modification, suppression),
    - Ils peuvent confirmer les réservations et envoyer un mail de confirmation,
    - Ils peuvent créer, modifier et administrer les salles.

Utilisateur est caractérisé par son nom, prénom, mail et mot de passe.

Une simple maîtrise de Windows est suffisante pour un utilisateur simple et standard, nous estimerons 2 à 3h de formation. Cependant pour futur administrateur il lui faudra une demi-journée de formation.

La liste des tâches à réaliser

- Réalisation de BDD
- Création interface ADMIN, STANDARD, SIMPLE
- Création formulaire d'inscription
- Création de page d'authentification
- Visualisation d'un agenda
- Fonction de création de compte, de création de réunion, d'équipement, de salle
- Possibilité d'envoyer des invitations lors de la création d'une réunion
- Fonction de validation de l'administrateur

L'architecture

L'ensemble de ce projet est basé sur le modèle MVC et suit la structure suivante :

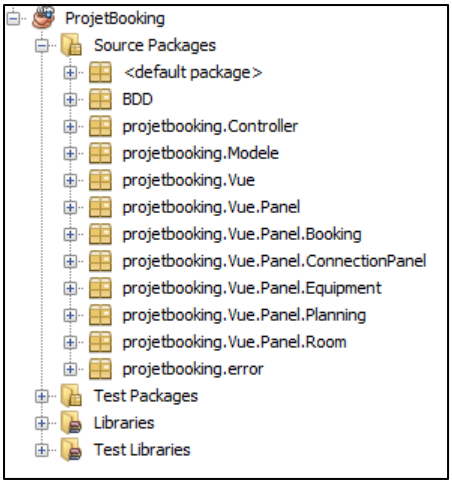

Figure 1. Structure de l'application

Pour pouvoir utiliser ce modèle, il est nécessaire de découper le projet en trois parties qui seront reliées, d'où les différents packages.

- Dans BDD se trouve la connexion à la base de données et les requêtes.
- Dans Controller se trouve le Controleur et un fichier empêchant des erreurs.
- Dans Vue se trouve le Main permettant de lancer l'application et la fenêtre nommée Vue ().
- Dans Modele se trouve les différentes classes objets des éléments de la base de données.

- Les packages regroupant des noms « panel » servent à changer de page.
- Et celui se nommant « error » permet de générer un message d'erreur.

Les choix technologiques

Pour mener ce projet à bien, nous utilisons comme vu précédemment le langage Java. Parmi les technologies utilisées se trouvent :

- La plupart des éléments principaux constituant Swing, pour les interfaces graphiques ;
- JDBC connect () pour se connecter à la base ;
- Une revue des classes objets ;
- Le principe du MVC

Outils et logiciels utilisés

Pour l'élaboration de ce projet, nous avons utilisé les outils suivants :

- L'IDE NetBeans, le principal logiciel utilisé pour développer le logiciel de Booking ;
- PhpMyAdmin, pour créer et héberger localement notre base de données ;
- WampServer pour pouvoir accéder au PhpMyAdmin ;
- GitHub pour mettre en commun les scripts ;

Figure 2. Logiciels utilisés

Document d'utilisation

Lors du lancement de l'application, l'utilisateur aura plusieurs choix :

- Créer un compte, si celui-ci n'en possède pas
- Se connecter, si celui-ci possède un compte
- Quitter l'application

Dans le cas ou l'utilisateur souhaite créer un compte, celui-ci doit remplir un formulaire comprenant les informations suivantes : NOM, PRENOM, MAIL et MOT DE PASSE. Ces champs sont obligatoires. En revanche, le champ du numéro de Téléphone est quant à lui non obligatoire. Après validation des informations, vous accèderez à **l'interface des utilisateurs simple**. De cette interface, vous serez capable de seulement consulter le planning de réservation de réunion et voir les informations liées à cette réunion : l'organisateur, l'identifiant de la salle, sa taille, les équipements ainsi que la date et l'heure de la réunion. Pour pouvoir avoir accès à des droits plus élevé, vous devez contacter un administrateur afin de vous changer les droits.

Dans le cas où vous posséder déjà un compte, vous devez saisir vos identifiants (mail et mot de passe). Passer cette étape, en fonction du niveau de rôle vous aurez certain droit :

- **Niveau 1 : vous êtes utilisateur simple** , vous n'aurez le droit qu'à la lecture des réservations de réunions comprenant ses composantes (organisateur, équipement, heure, numéro de salle etc…).
- **Niveau 2 : vous êtes un utilisateur standard**, en cliquant sur le bouton ‘’réservation’’ vous aurez accès à la consultation des réunions réservé mais également vous aurez la possibilité de créer une réunion, de modifier et supprimer une réunion dont vous êtes vous-même organisateur.
- **Niveau 3 : Vous êtes administrateur**, dès la connexion plusieurs boutons seront disponibles : « réservation », « équipement », « salle ». Chacun de ces boutons vous emmène dans une interface dédier à ceux clic, vous aurez ensuite la possibilité de : ‘’Ajouter’’, ‘’Modifier’’ et ‘’Supprimer’’ une réservation, un équipement ou une salle.

Descriptions des classes

Les classes du projet sélectionnées sont :

- **Vue.java**
  - Raison : Fenêtre principale
- **Controleur.java**
  - Raison : Permet d'effectuer des actions grâce aux écouteurs
- **Requetes.java**
  - Raison : Contient l'ensemble des requêtes principales du projet

Vue.java

Cette classe a pour rôle de générer une fenêtre en utilisant une méthode de changement de panneaux. C'est-à-dire que chaque nouvelle page reste dans la même fenêtre et son contenu est adapté.

Au début du fichier, des champs privés sont déclarés pour utiliser les panneaux et les variables.

Parmi l'ensemble de ses méthodes se trouvent :

- **public Vue() throws SQLException{} :** Permet de construire la fenêtre en utilisant certains champs privés à l'intérieur et prend en compte les exceptions SQL puisque des requêtes sont faites dans certains panneaux ;

- **public void centerFrame() {} :** Permet de recentrer la fenêtre ;

- **public void setPane(JPanel pane){} :** Permet de changer de panneau en passant en paramètre un panneau ;

- **public void backPanel(){} :** Permet de définir le panneau précédent et donc de revenir en arrière ;

- **public void activatePanel(String namePanel){} :** Permet de récupérer un nom de panneau en paramètre et d'activer la méthode setPane().

- **public void ajouterEcouteurBouton(String nomBouton, ActionListener listener) {} :** Permet d'écouter si le bouton a été cliqué dans un panneau en utilisant une String pour identifier le nom du bouton et un ActionListener permettant d'écouter ;

- **public ArrayList getText(String nomPanel) {} :** Permet de récupérer le texte de certaines balises grâce au paramètre String nomPanel et de le renvoyer sous forme d'ArrayList ;

- **public String getMail() {} :** Permet de récupérer le mail de l'utilisateur.

Controleur.java

La classe Controleur a pour rôle d'effectuer des actions grâce aux écouteurs.

Elle contient trois champs privés de type Statement, de type Vue et de type Modele.

Au sein de cette classe contient les méthodes suivantes :

- **public Controleur(Vue uneVue, Modele unModele) {} :** C'est le constructeur de la Classe Controleur, elle nécessite deux paramètres de type Vue et de type Modele. Dans cette méthode ce retrouve plusieurs utilisation de la Vue, exemple :
  - *uneVue.ajouterEcouteurBouton("Quitter", this);*
  - *uneVue.ajouterEcouteurBouton("Se créer un compte", this);*
  - *uneVue.ajouterEcouteurBouton("Se connecter", this);*
    Permet d'écouter si le bouton est cliqué ou pas, il utilise un string pour identifier le bouton.

- **public Vue getVue() {} :** Permet de récupérer la Vue

- **public Modele getModele() {}** : Permet de récupérer le modèle

- **public void actionPerformed(ActionEvent e) {}** : Une méthode la casse abstraite KeyAdapter, permettant d'effectuer des actions lorsqu'un évènement est déclenché.

Requetes.java

Cette classe a pour rôle de contenir la plupart des requêtes du projet.

Les méthodes sont les suivantes :

- **public static Statement connexion() {}** : permet la connexion à la base de donné
- **public static boolean createAccount(Statement statement, String name, String firstName, String mail, String password, String telephone) {}** : méthode permettant la creation d'un compte utilisateur, en passant par le formulaire d'inscription
- **public static int connectAccount(Statement statement, String mail, String password) {}** : Permet à un utilisateur de ce connecté à l'application
- **public static boolean addReservation(Statement statement, String mail, String date, String heure, String numSalle) {}** : Méthode permettant la création d'une réunion
- **public static void editReservation(Statement statement, Object[][] data) {}** : Méthode permettant la modification d'une réunion
- **public static void removeReservation(Statement statement, Object[][] data) {} :** Méthode permettant de supprimer une réunion
- **public static void validateReservation(Statement statement, Object[][] data) {} :** Méthode permettant à l'administrateur de validé la demande de réunion
- **public static void promoteUser(Statement statement, Object[][] data) {}** : Méthode permettant à l'administrateur de promouvoir les droits d'un utilisateur
- **public static boolean addEquipment(Statement statement, String nom) {}** : Méthode permettant d'ajouter un équipement
- **public static void editEquipment(Statement statement, Object[][] data) {}** : Méthode permettant la modification d'un équipement
- **public static void removeEquipment(Statement statement, Object[][] data) {} :** Méthode permettant permettant de supprimer un équipement
- **public static boolean addRoom(Statement statement, String nom, String taille, String equipement) {} :** Méthode permettant de créer une salle
- **public static void editRoom(Statement statement, Object[][] data) {} :** Méthode permettant de modifier la salle
- **public static void removeRoom(Statement statement, Object[][] data) {}** Méthode permettant de supprimer une salle

Les problèmes rencontrés

L'affichage du planning au passage en mode MVC :

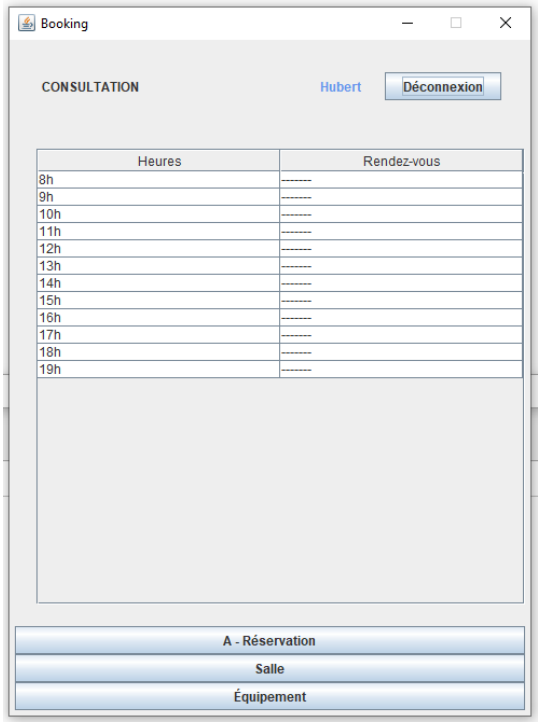

Figure 3. PlanningPanel

Au démarrage de notre projet, nous avions utilisé une simple fenêtre sans implémenter le modèle MVC. Nous avions donc un planning comme ci-dessous qui réagissait au clic sur les boutons :

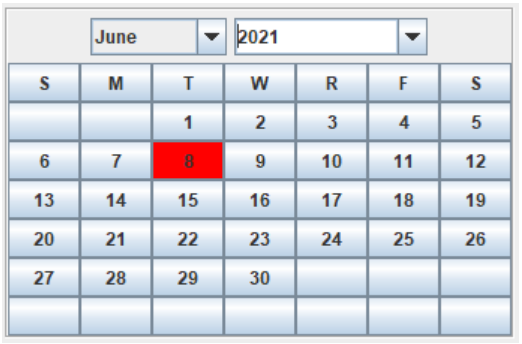

Figure 4. Illustration du calendrier

Mais au moment du passage en MVC, le planning ne s'est plus affiché et nous ne sommes pas arrivé dans les temps à régler le problème.

L'affichage d'une checkBox dans une JTable empêchant de faire les requêtes de modification, suppression et administrateur :

Lors du démarrage des fonctions de modifications, nous voulions mettre en place des checkBox qui nous permettraient de bien sélectionner la ligne et donc de bien récupérer les données.

En essayant de prendre exemple sur de la documentation et des recherches, nous avons inséré dans les JTable des booléens initialisés à false. Sur d'autres codes étrangers à notre projet tout fonctionnait mais pour notre part ce n'était pas le cas et nous ne comprenons pas pourquoi.

Par conséquent, les fonctions de modifications, suppressions et administrateurs sont présentes en requêtes et affichent les données mais ne peuvent pas fonctionner seulement à cause des checkBox et de la récupération des informations.

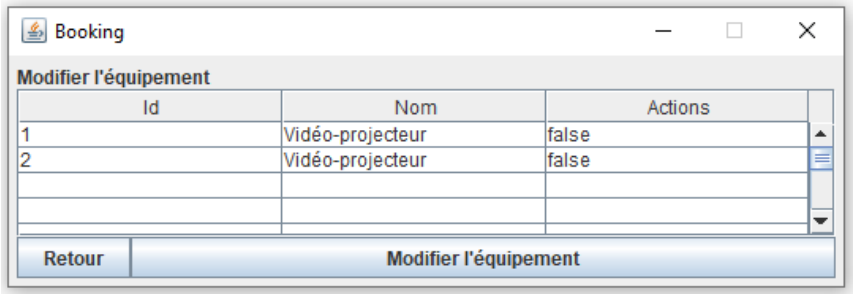

Figure 5. Illustration du problème de checkBox

Le résultat face aux attendus

Finalement, notre application est viable mais incomplète par rapport aux attendus du cahier des charges.

Les +	Les -
Fonction de connexion suivant les rôles	Affichage des données dans le planning
Fonction de création de compte suivant les rôles	Fonctions d'administrations
Fonction ajouter pour tous les panneaux	Fonctions de modifications
	Fonctions de suppression

L'ensemble du projet aurait pu être réalisé à la condition de comprendre les deux problèmes évoqués précédemment.
Swing n'est pas réellement maitrisé, nous retardant beaucoup, mais l'ensemble des requêtes sont écrites et la compréhension des liens entre les fichiers est complète. Le calendrier n'est certes pas affiché mais son code peut être retrouvé dans « PlanningPanel ». De plus, le modèle MVC est en place et toutes les classes possèdent sur chacune de ses méthodes une JavaDoc et des tests unitaires.

Table des illustrations