Projet Publicom_Admin 2

BLEE Leny - RAMOS Ambre - DELMAS Romain - GUILLAUMA Anthony

Sommaire

Présentation générale du projet

Contexte

PubliCom est, depuis plus de 30 ans, le leader dans la fabrication de supports d'affichage pour les municipalités. Au service des collectivités, ils ont développé un savoir-faire unique dans la gestion des projets des villes de France et d'Europe. Tourné vers l'innovation et le service, le groupe est le premier fabricant à proposer des écrans couleur à LED issus de ses propres usines en France, maîtrisant ainsi la conception, la fabrication, l'installation et l'après-vente de ses panneaux d'information lumineux pour les collectivités.

Besoin initial

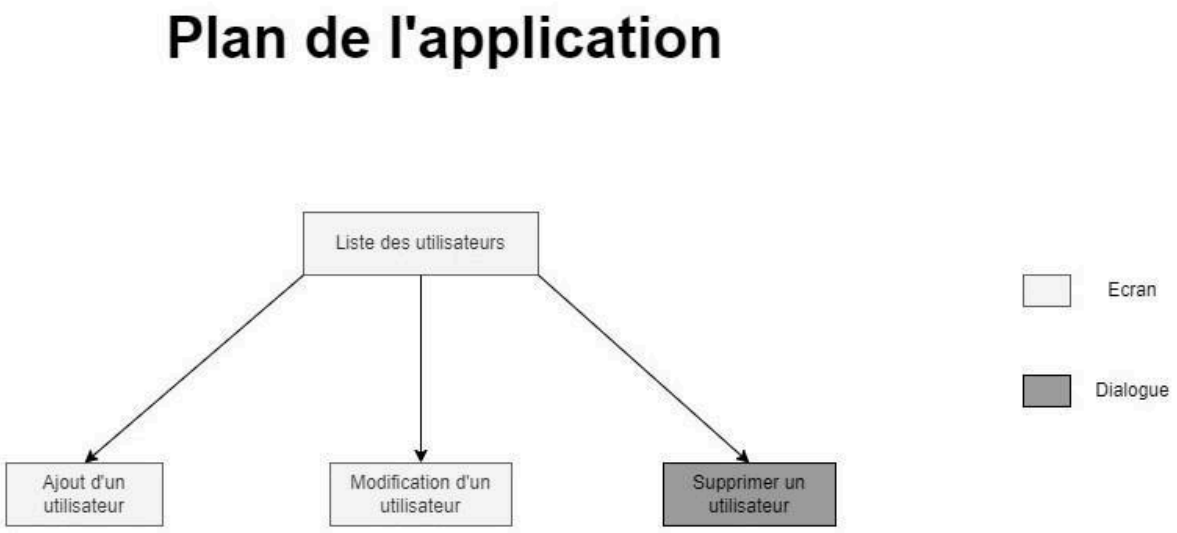
Nous avons été chargés de créer une application de gestion des utilisateurs de Publicom, en prenant compte des caractéristiques obligatoires suivantes :
1 : Application développée en langage **Java** avec une interface graphique basée sur les bibliothèques **Swing**.
2 : Une base de données **SQL** (MySQL, MariaDB)
3 : Application réalisant les opérations **CRUD** (Create, Read, Update, Delete)
4 : Mise en œuvre du motif d'architecture logicielle **MVC** (Modèle-Vue-Contrôleur)
5 : Mise en œuvre de **patrons de conception** (Design Patterns)

Solutions proposées

Pour répondre aux besoins de ce projet, nous avons choisi d'utiliser les outils suivants :
1 : draw.io (réalisation de la maquette et du plan de l'application)
2 : Netbeans (code et Swing)
3 : serveur LAMP sur Proxmox (gestion des données en pré-production)
4 : Wampserver (gestion des données en développement)
5 : WinDesign (réalisation du MCD)
6 : GitHub (gestion des versions du projet)
L'avancement du projet est disponible sur GitHub.

Maquette et plan de l’application

Plan de l’application



Maquette

Liste des utilisateurs

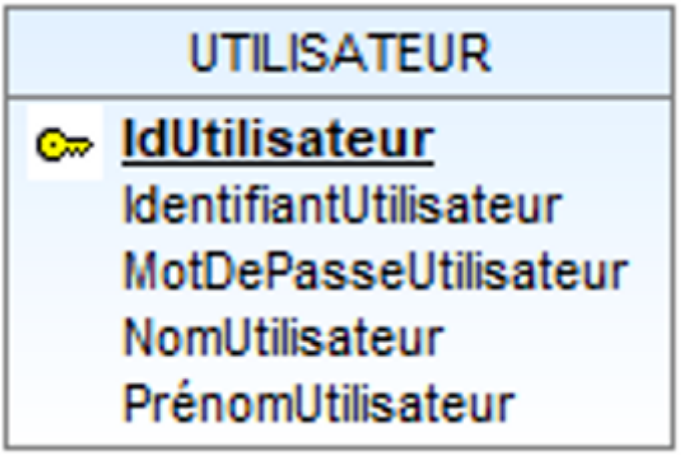
Supprimer un utilisateur (pop-up sur la fenêtre Liste des utilisateurs)

Ajouter un nouvel utilisateur

Modifier un utilisateur

MCD

Visualisation dans WinDesign



Script SQL (pour importation dans base de données MySQL)

```sql
DROP DATABASE IF EXISTS mlr_publicom;

CREATE DATABASE IF NOT EXISTS mlr_publicom;
USE mlr_publicom;
# ------------------------------------------------------------
#       TABLE : UTILISATEUR
# ------------------------------------------------------------

CREATE TABLE IF NOT EXISTS UTILISATEUR
 (
   IDUTILISATEUR INTEGER NOT NULL AUTO_INCREMENT,
   IDENTIFIANTUTILISATEUR VARCHAR(32) NULL  ,
   MOTDEPASSEUTILISATEUR VARCHAR(255) NULL  ,
   NOMUTILISATEUR VARCHAR(32) NULL  ,
   PRENOMUTILISATEUR VARCHAR(32) NULL
   , PRIMARY KEY (IDUTILISATEUR)
 )
 comment = "";
```

Descriptif technique de l'application

Publicom_Admin est une application client lourd en Java, dont l'objectif est de gérer les utilisateurs de la plateforme Publicom. Aucune connexion à cette application n'est nécessaire, puisqu'il s'agit d'un outil indépendant et seulement accessible par les personnes qui le possèdent.

Pour la réalisation de ce projet, nous avons choisi d'utiliser les solutions suivantes :
1 : draw.io (réalisation de la maquette et du plan de l'application)
2 : Netbeans (code et Swing)
3 : serveur LAMP sur Proxmox (gestion des données en pré-production)
4 : Wampserver (gestion des données en développement)
5 : WinDesign (réalisation du MCD)
6 : GitHub (gestion des versions du projet)
L'avancement du projet est disponible sur GitHub.

Le code de l'application est organisé dans les packages suivants :
1 : Le package View contient les classes de l'interface graphique (Swing)
2 : Le package Model contient les classes « métier » (définies à partir du MCD)
3 : Le package Controller contient les classes contrôleurs (un pour chaque vue)
4 : Le package DAO (Data Access Object) contient les classes d'accès aux données
5 : Le package Utilities pour les objets utilitaires (nécessaires pour exploiter les classes et API externes)

Un des requis concernant la réalisation de cette application est également la production de code SOLID, en travaillant sur les deux premiers principes :
1 : Single responsibility (Une classe ne doit faire qu'une seule chose et elle doit bien la faire. Elle ne doit avoir qu'une seule raison de changer.)
2 : Open/closed (Une classe doit être ouverte à l'extension, mais fermée à la modification.)

Guide d’utilisation

Quand l’utilisateur lance l’application, il arrive par défaut sur la page “liste des utilisateurs”. Sur cette page, sont affichés l’identifiant, le nom et le prénom de tous les utilisateurs existants dans un tableau.

Ici, l’utilisateur peut donc visualiser la liste des utilisateurs existants, et ajouter un nouvel utilisateur dans la base de données à partir du bouton “ajouter un nouvel utilisateur”, mais il peut également faire une sélection : s’il sélectionne un utilisateur, il peut le modifier en cliquant sur le bouton “modifier l’utilisateur”, mais s’il en sélectionne plusieurs, il ne pourra que les supprimer à partir du bouton “supprimer l’utilisateur”.

Si l’utilisateur choisit d’ajouter un nouvel utilisateur, il est redirigé vers la page “ajouter un nouvel utilisateur” où il saisit le nom, prénom et mot de passe de l’utilisateur qu’il veut créer. Son identifiant, qui sera plus tard affiché sur la page “liste des utilisateurs”, est créé automatiquement par concaténation (par exemple, si l’on crée un utilisateur qui s’appelle Kevin Joando, son identifiant sera kevin.joando).

Si l’utilisateur choisit de modifier un utilisateur, il est redirigé vers la page “modifier l’utilisateur”. Sur cette page, l’utilisateur peut modifier le prénom, le nom et le mot de passe d’un utilisateur déjà existant.

Si l’utilisateur choisit de supprimer un utilisateur, une fenêtre d’alerte s’ouvre par-dessus la page “liste des utilisateurs” et demande de confirmer ce choix, car la suppression d’un utilisateur dans la base de données est définitive.

Descriptif des tâches

Tâches réalisées par Leny

Réalisation de la page “modifier un utilisateur” avec Swing + code
Requête de récupération des utilisateurs

Tâches réalisées par Anthony

Réalisation de la maquette de l’application avec draw.io
Dépôt du projet sur GitHub
Réalisation de la page “liste des utilisateurs” avec Swing + code
Contrôleurs

Tâches réalisées par Romain

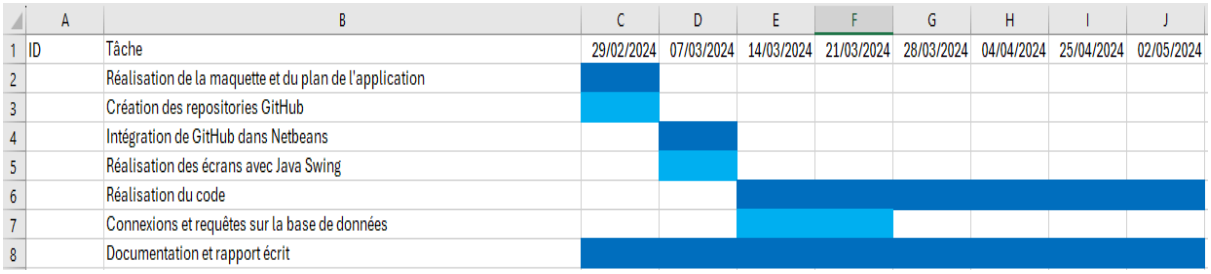
Réalisation de la page “créer un utilisateur” avec Swing + code

Tâches réalisées par Ambre

Réalisation de la documentation (rapport écrit)

Planning des activités réalisées

Diagramme de Gantt



	A	B	C	D	E	F	G	H	I	J
1	ID	Tâche	29/02/2024	07/03/2024	14/03/2024	21/03/2024	28/03/2024	04/04/2024	25/04/2024	02/05/2024
2		Réalisation de la maquette et du plan de l'application								
3		Création des repositories GitHub								
4		Intégration de GitHub dans Netbeans								
5		Réalisation des écrans avec Java Swing								
6		Réalisation du code								
7		Connexions et requêtes sur la base de données								
8		Documentation et rapport écrit								

Détail des activités

29/02/2024 : début du projet, réalisation de la maquette et du plan de l'application avec draw.io, reprise de l'ancien MCD PubliCom afin de l'adapter au projet actuel, création des repository GitHub

07/03/2024 : intégration de GitHub dans Netbeans, début de réalisation des écrans de l'interface graphique de l'application avec Java Swing

14/03/2024 : début de réalisation du code pour chaque vue, connexion à la base de données

21/03/2024 : création des requêtes, affichage des requêtes
28/03/2024 : finalisation des projets, tests en environnement réel.

Déroulement du projet

Solutions choisies

Pour répondre aux besoins de ce projet, nous avons choisi d'utiliser les outils suivants :
1 : draw.io (réalisation de la maquette et du plan de l'application)
2 : Netbeans (code et Swing)
3 : serveur LAMP sur Proxmox (gestion des données en pré-production)
4 : Wampserver (gestion des données en développement)
5 : WinDesign (réalisation du MCD)
6 : GitHub (gestion des versions du projet)
L'avancement du projet est disponible sur GitHub.

Difficultés rencontrées

Lors du début du projet, nous avons été contraints de remplacer la solution Balsamiq initialement demandée dans le cahier des charges par draw.io (licences expirées). Nous avons également dû changer de version de JDK, pour passer de la version 18 à la version 21 à cause de problèmes d'incompatibilité.