Chapitre 4
Les bases de données avec Java EE

1. Modélisation de la base de données avec UML 2

Comme dans tout projet de développement d'application, il est primordial de bien penser et de bien concevoir en amont une base de données adaptée au besoin.

En effet, une base de données bien modélisée permet à la fois d'optimiser les requêtes SQL et le code d'un point de vue de la performance et de la maintenabilité.

1.1 Cahier des charges

L'objectif est de modéliser une base de données objet à l'aide d'UML 2.

Tout d'abord, nous allons lister les principales fonctionnalités attendues par les utilisateurs. Il va en découler naturellement les différentes tables à créer, implémenter, les relations entre elles et les différentes autres contraintes.

L'application web Java Devis Pro BTP permet à un artisan de gérer son portefeuille client, d'éditer des devis et des factures correspondant aux travaux qu'il va réaliser ou qu'il a réalisés.

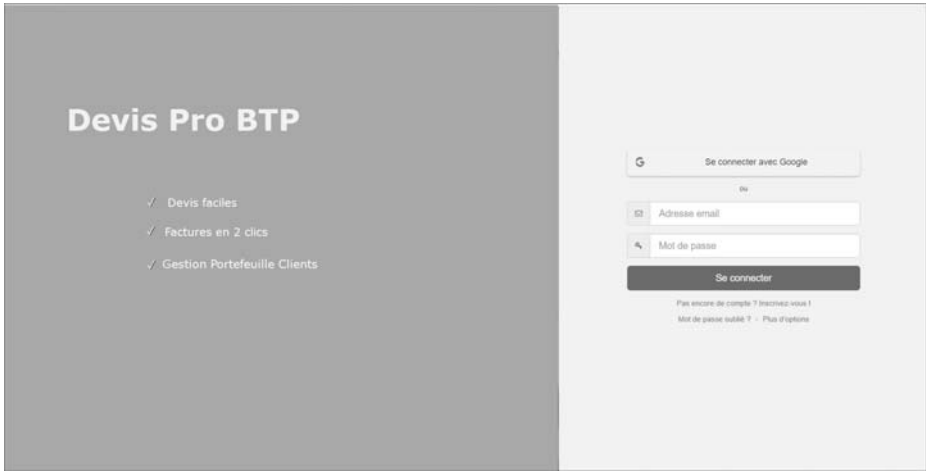
Voici ci-dessous les principales fonctionnalités de l'application :

- créer un devis,
- consulter un devis,
- modifier un devis,
- supprimer un devis,
- envoyer le devis à un client,
- ajouter un nouveau client,
- consulter une fiche client,
- rechercher un client à partir de son nom,
- mettre à jour les informations relatives à un client,
- supprimer une fiche client,
- créer une facture,
- consulter une facture,
- modifier une facture,
- supprimer une facture,
- rechercher une facture,
- gérer son profil artisan,
- se connecter,
- se déconnecter,
- changer son mot de passe,
- gérer le mot de passe et l'identifiant perdus.

1.2 Interface utilisateur

Afin de modéliser au mieux le besoin utilisateur, il est intéressant de s'appuyer dès à présent sur quelques prototypes ou maquettes des futures interfaces utilisateur.

Ci-dessous se trouve la maquette correspondant à l'interface qui permet à l'artisan de s'authentifier à l'application web de gestion pro des devis. Grâce à elle, il peut, par exemple créer un devis des travaux à réaliser chez un client.



Il est possible de s'authentifier de deux façons auprès de l'application.

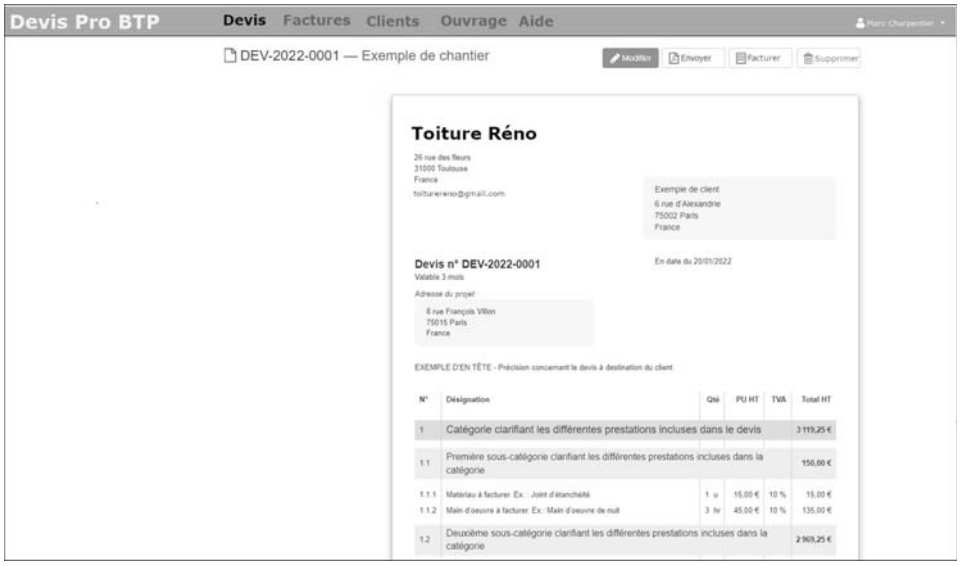
La première possibilité est de se connecter à partir de son adresse Gmail, si l'artisan en dispose une. La deuxième possibilité est de s'authentifier directement auprès de l'application à partir de son e-mail stocké dans la base de données ainsi que du mot de passe stocké de façon encryptée. Par exemple, on pourra créer une table nommée **Artisan** dans le but de stocker ces informations de connexion.

À la première connexion, l'artisan peut créer son compte en cliquant sur **Pas encore de compte ? Inscrivez-vous** s'il ne souhaite pas s'authentifier à partir d'une adresse Gmail.

S'il a oublié son mot de passe, en cliquant sur **Mot de passe oublié**, il peut le réinitialiser.

Une fois connecté, l'artisan a accès à l'interface principale de l'application Devis Pro BTP et aux fonctionnalités essentielles comme la création d'un nouveau devis, la création d'une nouvelle facture, l'ajout de nouveaux clients, la gestion et le suivi des paiements des factures, l'enrichissement de la liste des ouvrages liés à son activité.

Voici, ci-dessous, la maquette correspondant à l'interface principale de l'application web Devis Pro BTP :



Le menu principal comporte les items suivants : **Devis**, **Facture**, **Client**, **Ouvrage** et **Aide** pour accéder à chacune des interfaces web.

En haut en droite, l'artisan étant authentifié, son nom et prénom apparaissent, ainsi que l'icône permettant d'accéder à la déconnexion et à la gestion des informations liées à son profil.

1.3 Schéma UML2 : base de données devisprobtp

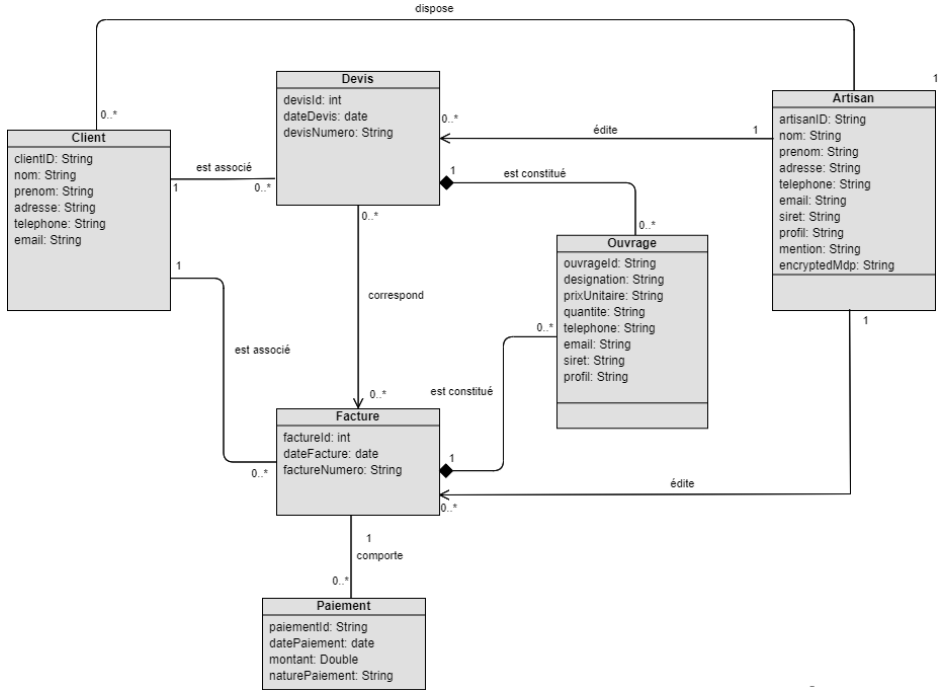
1.3.1 Schéma global de la base de données devis pro BTP

Voici, ci-dessous, la modélisation de la base de données avec UML 2 représentant les relations entre les différentes tables de l'application Devis Pro BTP :

– La table **Artisan** contient les données personnelles de l'artisan (nom, prénom, e-mail, adresse, mot de passe, Siret, profil…).

– La table **Devis** contient le `devisId` identifiant unique auto-incrémenté, `dateDevis` la date d'émission du devis et `devisNumero`.

– La table **Ouvrage** permet de retrouver les désignations des travaux liés à un devis ou à une facture.



1.3.2 Informations relatives à la table Artisan

D'un point de vue relations et cardinalités entre la table **Artisan** et les tables **Client**, **Facture** et **Devis**, nous pouvons noter que :

– Un artisan possède 0 ou plusieurs clients.

– Un artisan édite 0 ou plusieurs devis.

– Un artisan édite 0 ou plusieurs factures.

1.3.3 Informations relatives à la table Client

D'un point de vue relation et cardinalités entre la table **Client** et les autres tables **Artisan**, **Facture** et **Devis**, on peut noter que :

– 0 ou plusieurs clients correspondent à un artisan.

– 0 ou plusieurs devis sont édités par un artisan.

– 0 ou plusieurs factures sont éditées par un artisan.

1.3.4 Informations relatives à la table Devis

D'un point de vue relation et cardinalités entre la table **Devis** et les tables **Facture**, **Ouvrage**, **Artisan** et **Client**, nous pouvons noter que :

– 0 ou plusieurs devis sont édités par un artisan donné.

– 0 ou plusieurs devis correspondent à un client donné. Un client peut souhaiter obtenir plusieurs devis comparatifs, par exemple avec des matériaux de construction différents ou des choix techniques différents, qui auront un impact sur le coût final des travaux.

– 0 ou plusieurs devis correspondent à 0 ou plusieurs factures. Ainsi, l'artisan aura la possibilité de transformer facilement un devis accepté par le client en facture finale.

– Un devis donné est constitué de 0 ou plusieurs ouvrages ou réalisations effectués par l'artisan (plusieurs lignes dans le devis).

1.3.5 Informations relatives à la table Facture

D'un point de vue relation et cardinalités entre la table **Facture** et les tables **Paiement**, **Client**, **Artisan**, **Ouvrage** et **Devis**, nous pouvons noter que :

– 0 ou plusieurs factures correspondent à un artisan donné.

– Une facture donnée correspond à 0 ou plusieurs paiements. En effet, un client peut régler en plusieurs fois, verser un acompte au préalable...

– 0 ou plusieurs factures sont associées à un client donné.

– Une facture donnée est constituée de 0 ou plusieurs ouvrages (liste de travaux).

– 0 ou plusieurs factures correspondent à 0 ou plusieurs devis.

1.4 Script SQL base de données devisprobtp

Pour créer la base de données devisprobtp, avant de pouvoir exécuter le script SQL qui permettra de créer les différentes tables de l'application :

- Lancez une console DOS ou un terminal sous Linux suivant votre système d'exploitation.
- Connectez-vous avec l'utilisateur postgres sur linux `sudo -i -u postgres` sous Windows `psql -U postgres`.
- Créez un compte spécifique pour la base de données devisprobtp avec la commande suivante :
  `create user admin WITH PASSWORD ' devisprobtp ';`
- Créez la base de données devisprobtp avec la commande SQL suivante : `create database devisprobtp`.
- Donnez tous les droits à l'utilisateur admin qui vient d'être créé sur la base de données devisprobtp : `GRANT ALL PRIVILEGES ON DATABASE devisprobtp to admin;`.
- Saisissez `\q` pour quitter.
- Reconnectez-vous en tant qu'utilisateur postgres sur linux `sudo -i -u postgres` sous windows `psql -U postgres`.
- Sélectionnez la base de données devisprobtp nouvellement créée `\c devisprobtp` (pour voir la liste de toutes les bases de données faites `\l`).

Voici, ci-dessous, le script SQL qui permet de créer les différentes tables pour la base de données devisprobtp :

```
/* Table Artisan */
create table artisan(
     artisanID serial PRIMARY KEY not null,
     nom varchar(60),
     prenom varchar(60),
     adresse varchar(100),
     telephone varchar(10),
     email varchar(100),
     siret varchar(14),
     profil varchar(60),
     encryptedMdp varchar(20)
```

```
);
/* Table Client */
create table client(
     clientId serial primary key not null,
     nom varchar(60),
     prenom varchar(60),
     addresse varchar(100),
     email varchar(100),
     artisanId integer,
     CONSTRAINT artisan_client_fk FOREIGN KEY (artisanId)
REFERENCES artisan (artisanId));

/* Table Facture */
create table facture(
     factureId serial primary key not null,
     dateFacture date,
     factureNumero varchar(100),
     artisanId integer,
     clientId integer,
     CONSTRAINT artisan_facture_fk FOREIGN KEY (artisanId)
REFERENCES artisan (artisanId),
     CONSTRAINT client_facture_fk FOREIGN KEY (clientId)
REFERENCES client (clientId)
);

/* Table Devis*/
create table devis(
     devisId serial primary key not null,
     dateDevis date,
     devisNumero varchar(100),
     artisanId integer,
     clientId integer,
     CONSTRAINT artisan_devis_fk FOREIGN KEY (artisanId)
REFERENCES artisan (artisanId),
     CONSTRAINT client_devis_fk FOREIGN KEY (clientId) REFERENCES
client (clientId)
);

/* Table Paiement */
create table paiement(
     paiementId serial PRIMARY KEY not null,
     datePaiement varchar(50),
     montant double,
     naturePaiement varchar(50),
```