Yvert Faustine
BTS SIO 2B

DOCUMENTATION TECHNIQUE
Galaxy Swiss Bourdin

2021 - 2022

SOMMAIRE

I. Cadre du projet
    1) Contexte de l'entreprise
    2) Enjeux et objectifs
    3) Modélisation

II. Le projet
    1) L'architecture
    2) Processus de génération (build)

CADRE DU PROJET

1) Contexte de l'entreprise

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.

L'activité à gérer

L'activité commerciale d'un laboratoire pharmaceutique est principalement réalisée par les visiteurs médicaux. En effet, un médicament remboursé par la sécurité sociale n'est jamais vendu directement au consommateur mais prescrit au patient par son médecin.
Il est important, pour l'industrie pharmaceutique, de promouvoir ses produits directement auprès des praticiens.

Les visiteurs médicaux

L'activité des visiteurs médicaux consiste à visiter régulièrement les médecins généralistes, spécialistes, les services hospitaliers ainsi que les infirmiers et pharmaciens pour les tenir au courant de l'intérêt de leurs produits et des nouveautés du laboratoire.
Chaque visiteur dispose d'un portefeuille de praticiens, de sorte que le même médecin ne reçoit jamais deux visites différentes du même laboratoire.
Pour affiner la définition des objectifs et l'attribution des budgets, il sera nécessaire d'informatiser les comptes rendus de visite.

L'activité des visiteurs

L'activité est composée principalement de visites : réalisées auprès d'un praticien (médecin dans son cabinet, à l'hôpital, pharmacien, chef de clinique…), on souhaite en connaître la date, le motif (6 motifs sont fixés au préalable), et savoir, pour chaque visite, les médicaments présentés et le nombre d'échantillons offerts. Le bilan fourni par le visiteur devra aussi être enregistré.

Les produits

Les produits distribués par le laboratoire sont des médicaments : ils sont identifiés par un numéro de produit (dépôt légal) qui correspond à un nom commercial (ce nom étant utilisé par les visiteurs et les médecins).
Comme tout médicament, un produit a des effets thérapeutiques et des contre-indications.
On connait sa composition (liste des composants et quantité) et les interactions qu'il peut avoir avec d'autres médicaments (éléments nécessaires à la présentation aux médecins).
Un produit relève d'une famille (antihistaminique, antidépresseur, antibiotique, …).

Les médecins

Les médecins sont le cœur de cible des laboratoires. Aussi sont-ils l'objet d'une attention toute particulière.
Pour tenir à jour leurs informations, les laboratoires achètent des fichiers à des organismes spécialisés qui donnent, les diverses informations d'état civil (nom, prénom, adresse, numéro de téléphone, …) et la spécialité complémentaire.

2) Enjeux et objectifs

Gérer les rapports de visite

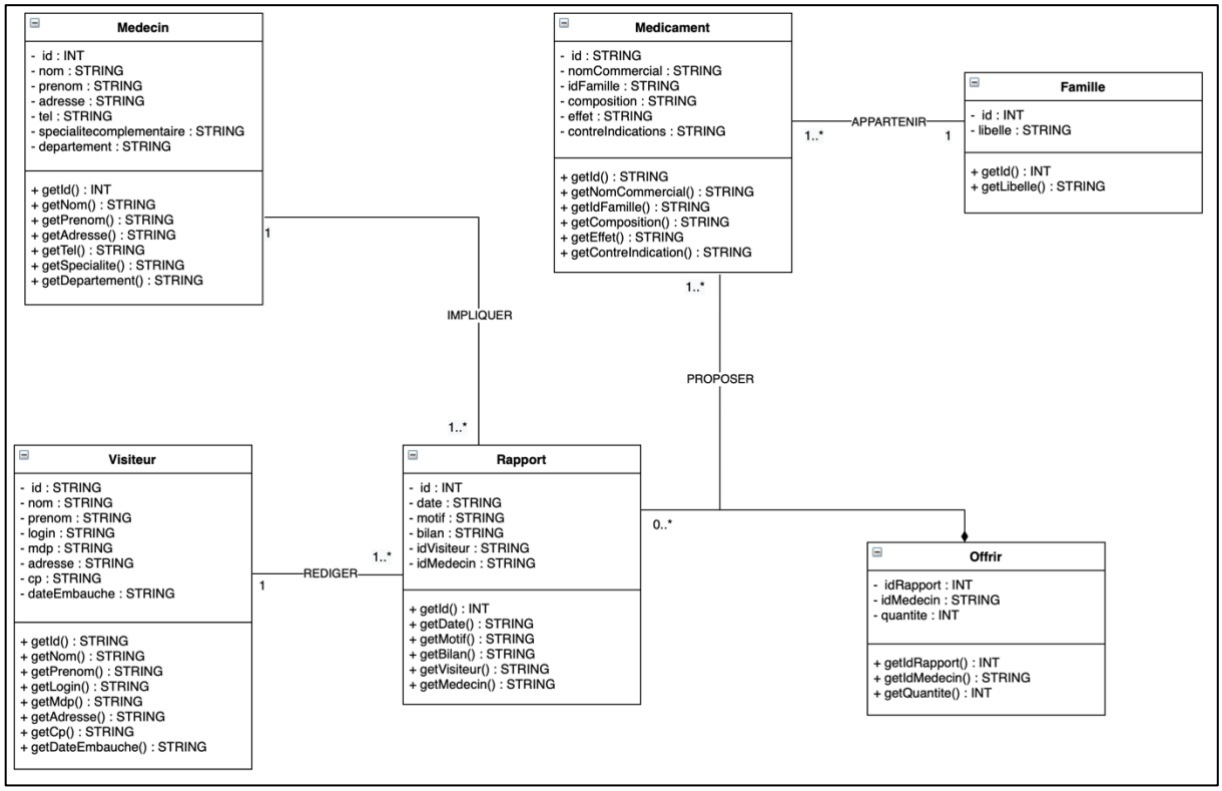
- Le visiteur demande à créer un nouveau rapport de visite
- Le système retourne un formulaire avec la liste des médecins et des champs de saisie
- Le visiteur sélectionne un médecin à partir de son début de nom, sélectionne la date et remplit les différents champs, sélectionne les médicaments et les quantités offertes et valide
- Si des champs ne sont pas remplis, le système en informe le visiteur
- Le système enregistre le rapport.
- Le visiteur demande à modifier un rapport
- Le système retourne un formulaire avec une date à sélectionner
- Le visiteur sélectionne la date
- Le système retourne les rapports que le visiteur a effectués à cette date
- Le visiteur sélectionne un rapport de visite
- Le système retourne les informations déjà saisies concernant le motif et le bilan
- Le visiteur modifie les informations
- Le système enregistre les modifications

Gérer les médecins

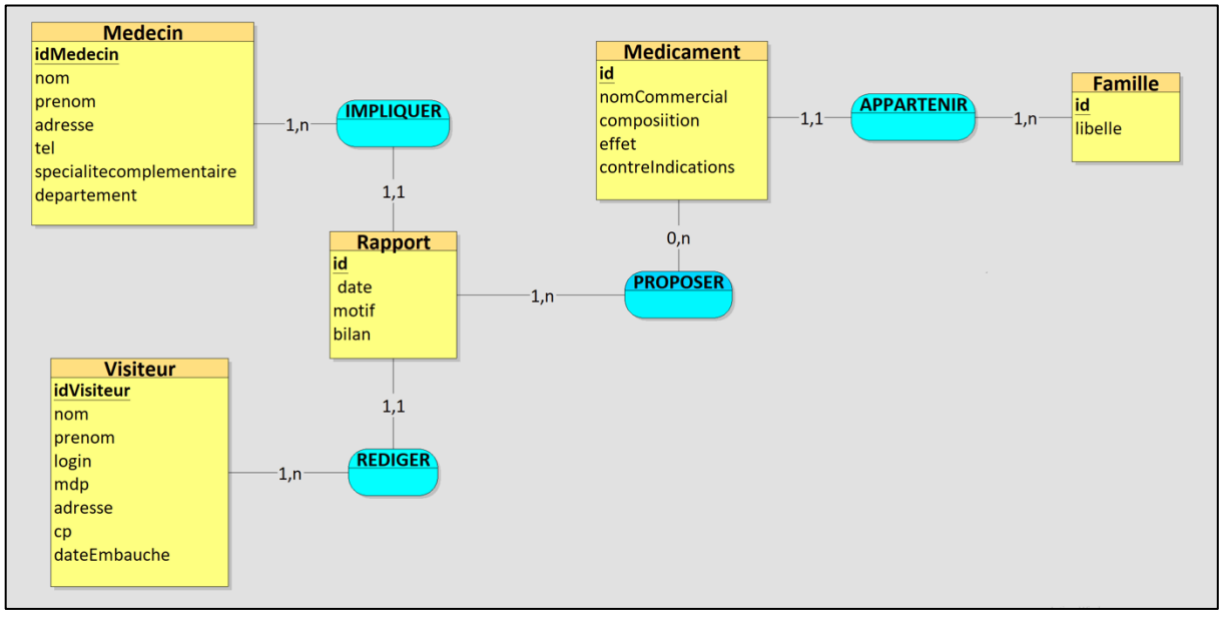
- Le visiteur demande à voir les informations concernant un médecin
- Le système retourne un formulaire avec un champ de recherche du médecin
- Le visiteur sélectionne un médecin à partir de son début de nom et valide
- Le système retourne les informations personnelles concernant ce médecin
- Le visiteur clique sur le numéro de téléphone du médecin
- Le système compose le numéro
- Le visiteur demande à voir tous les anciens rapports de visite concernant ce médecin
- Le système retourne tous ses rapports
- Le visiteur demande à modifier certains champs concernant des informations du médecin
- Le système enregistre ces modifications.

- Modélisation

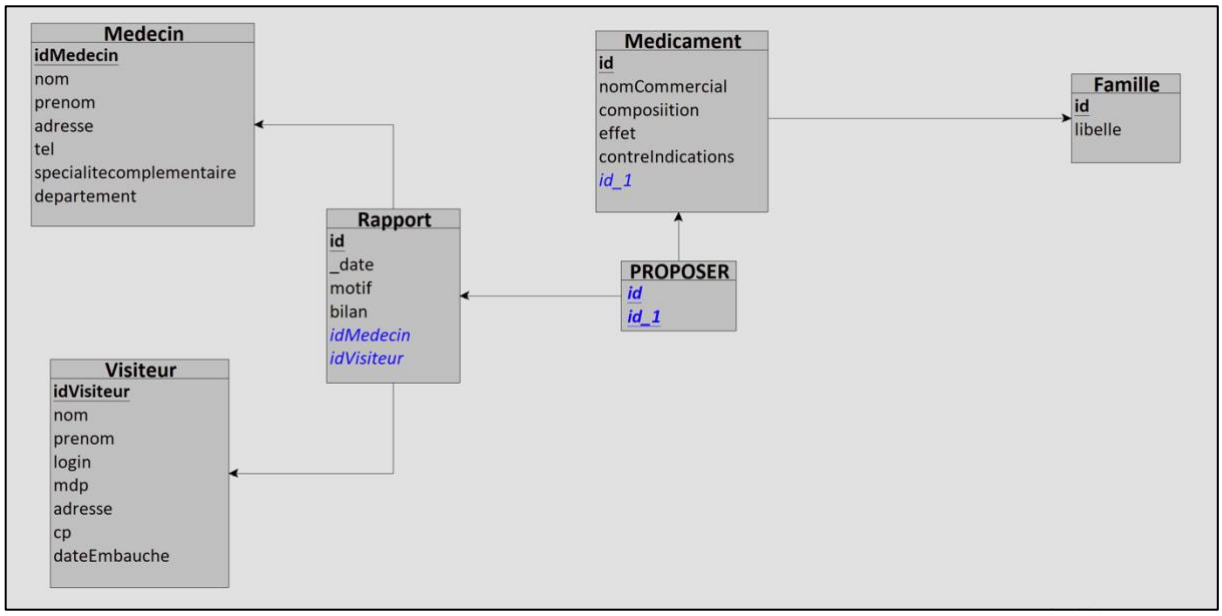
Unified Modeling Language (UML)



Modèle Conceptuel de Données (MCD)



Modèle Logique des Données (MLD)



Modèle Logique des Données textuel (MLD)

Medecin = (idMedecin *INT*, nom *VARCHAR(50)*, prenom *VARCHAR(50)*, adresse *VARCHAR(50)*, tel *VARCHAR(10)*, specialitecomplementaire *VARCHAR(100)*, departement *VARCHAR(50)*);

Visiteur = (idVisiteur *VARCHAR(50)*, nom *VARCHAR(50)*, prenom *VARCHAR(50)*, login *VARCHAR(50)*, mdp *VARCHAR(50)*, adresse *VARCHAR(50)*, cp *VARCHAR(50)*, dateEmbauche *VARCHAR(50)*);

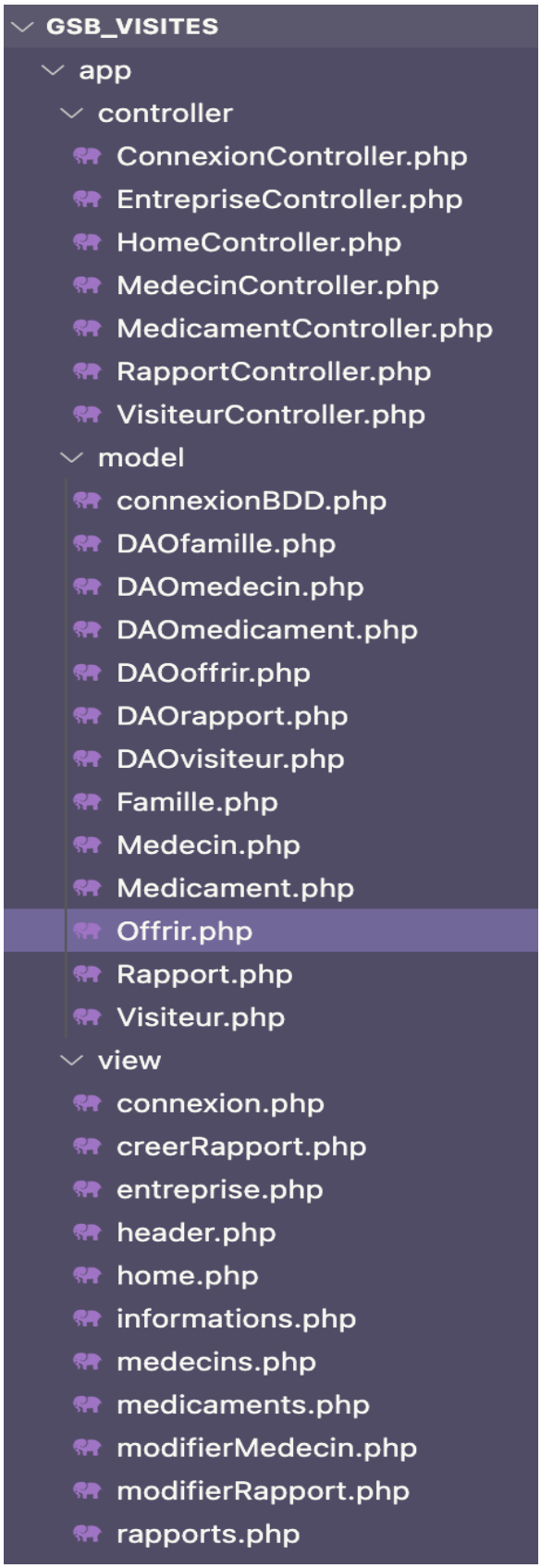
Rapport = (id *INT*, _date *DATE*, motif *VARCHAR(50)*, bilan *VARCHAR(50), #idMedecin, #idVisiteur*);

Famille = (id *INT*, libelle *VARCHAR(50)*);

Medicament = (id *VARCHAR(50)*, nomCommercial *VARCHAR(50)*, composiition *VARCHAR(50)*, effet *VARCHAR(50)*, contreIndications *VARCHAR(50), #id_1*);

PROPOSER = (*#id, #id_1*);

LE PROJET



1) L'architecture

Le motif d'architecture est en MVC (Modèle, Vue, Contrôleur).

Le pattern MVC permet de bien organiser son code source. Il va vous aider à savoir quels fichiers créer, mais surtout à définir leur rôle. Le but de MVC est justement de séparer la logique du code en trois parties que l'on retrouve dans des fichiers distincts.

- **Modèle** : cette partie gère les données du site. Son rôle est d'aller récupérer les informations « brutes » dans la base de données, de les organiser et de les assembler pour qu'elles puissent ensuite être traitées par le contrôleur. On y trouve donc entre autres les requêtes SQL.
- **Vue** : cette partie se concentre sur l'affichage. Elle ne fait presque aucun calcul et se contente de récupérer des variables pour savoir ce qu'elle doit afficher. On y trouve essentiellement du code HTML mais aussi quelques boucles et conditions PHP très simples, pour afficher par exemple une liste de messages.
- **Contrôleur** : cette partie gère la logique du code qui prend des décisions. C'est en quelque sorte l'intermédiaire entre le modèle et la vue : le contrôleur va demander au modèle les données, les analyser, prendre des décisions et renvoyer le texte à afficher à la vue. Le contrôleur contient exclusivement du PHP. C'est notamment lui qui détermine si le visiteur a le droit de voir la page ou non (gestion des droits d'accès).

- **L'index** : fichier répertoriant les actions (?action=)

2) Processus de génération (build)

- **Les rapports (exemple pour l'ajout d'un rapport)**

➔ Connexion à la base de données

1ère étape : la connexion à la base de données est très importante, afin que le modèle puisse récupérer toutes les informations de celle-ci.

```php
<?php
    namespace App\model;
    use PDO;

    // Class
    class connexionBDD {

        // Variables
        private static $conn_DB = null;
        private static $host_DB = '127.0.0.1';
        private static $name_DB = 'gsb_gestion_visites';
        private static $user_DB = 'root';
        private static $pass_DB = '';

        // Fonctions
        protected static function getPDO() {
            if(is_null(self::$conn_DB)) {
                $connectPDO = new PDO('mysql:host=' . self::$host_DB . ';dbname=' . self::$name_DB . ';charset=utf8',
self::$user_DB, self::$pass_DB);
                $connectPDO->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

                self::$conn_DB = $connectPDO;
            }

            return self::$conn_DB;
        }

        protected static function query($statement) {
            $stat = self::getPDO()->query($statement);

            return $stat;
        }

        protected static function prepare($statement, $attributes) {
            $stat = self::getPDO()->prepare($statement);
            $stat->execute($attributes);
```

```
        return $stat->fetchAll();
    }

    protected static function prepareFetch($statement, $attributes) {
        $stat = self::getPDO()->prepare($statement);
        $stat->execute($attributes);

        return $stat->fetch();
    }

    protected static function request($statement, $attributes) {
        $stat = self::getPDO()->prepare($statement);

        $stat->execute($attributes);
    }
  }
?>
```

➔ Class Rapport

2ème étape : création de la classe ainsi que les déclarations de variables afin de créer leurs fonctions attribuées.

```
<?php

namespace App\model;

Class Rapport {
private $id;
private $date;
private $motif;
private $bilan;
private $idVisiteur;
private $idMedecin;


public function __construct($unId, $unDate, $unMotif, $unBilan, $unIdVisiteur, $unIdMedecin){
$this->id=$unId;
$this->date=$unDate;
$this->motif=$unMotif;
$this->bilan=$unBilan;
$this->idVisiteur=$unIdVisiteur;
```

```
$this->idMedecin=$unIdMedecin;


}
public function getId(){
return $this->id;
}
public function getDate(){
return $this->date;
}
public function getMotif(){
return $this->motif;
}
public function getBilan(){
return $this->bilan;
}
public function getVisiteur(){
return $this->idVisiteur;
}
public function getMedecin(){
return $this->idMedecin;
}


}
?>
```

➔ Rapport DAO et Offrir DAO

3ème étape : création de la fonction creerUnRapport avec comme paramètre « date, motif, idVisiteur, idMedecin » ainsi que la création de la requête, ici « INSERT INTO … VALUES » en utilisant les valeurs misent en paramètre.
Création de la fonction creerOffrir.

```
// Ajout d'un rapport
public static function creerUnRapport($date, $motif, $bilan, $idVisiteur, $idMedecin) {
$request = self::request('INSERT INTO Rapport(date, motif, bilan, idVisiteur, idMedecin) VALUES (:date, :motif, :bilan,
:idVisiteur, :idMedecin);', array(':date' => $date, ':motif' => $motif, ':bilan' => $bilan, ':idVisiteur' => $idVisiteur, ':idMedecin' =>
$idMedecin));
}
```

```
// Offrir des médicaments
        public static function creerOffrir($idRapport, $medicament, $quantite) {
            $request = self::request('INSERT INTO Offrir VALUES (:idRapport, :medicament, :quantite);', array(':idRapport' =>
$idRapport, ':medicament' => $medicament, ':quantite' => $quantite)); }
```

➔ Rapport Contrôleur

4ème étape : utilisation de la requête SQL de rapport et offrir grâce à la méthode « verifierRapport ». On utilise aussi les DAO rapport et offrir.

```
private static function verifierRapport() {

if(isset($_POST['bilan_RapportCreate'])) {
$date = $_POST['date_RapportCreate'];
$motif = $_POST['motif_RapportCreate'];
$bilan = $_POST['bilan_RapportCreate'];
$idVisiteur = $_SESSION['visiteurID'];
$idMedecin = $_POST['medecin_RapportCreate'];
$medicament = $_POST['medicament_RapportCreate'];
$quantite = $_POST['quantite_RapportCreate'];

DAOrapport::creerUnRapport($date, $motif, $bilan, $idVisiteur, $idMedecin);
DAOoffrir::creerOffrir(DAOrapport::getDernierRapport(), $medicament, $quantite);

header('Location: ./?action=rapports');
}
}
```

➔ Rapport (vue)

5ème étape : utilisation dans la vue grâce au contrôleur.
(Exemple pour l'affichage des motifs dans la création du rapport)

```
<?php foreach(RapportController::recupererLesMotifs() as $key => $val) { ?>
                <option value=<?= $val; ?>><?= $val; ?></option>
            <?php  } ?>
```