

Projet Personnel Encadré (PPE) n°3

Rapport d'activité

Contexte : GSB

Intitulé du projet	Laboratoire GSB
Nom, prénom	DELAPORTE, Thomas
Année session examen	2019

Activités exigées pour l'examen E4

- ☐ A1.1.1 - Analyse du cahier des charges d'un service à produire
- ☐ A1.2.2 - Rédaction des spécifications techniques de la solution retenue (adaptation d'une solution existante ou réalisation d'une nouvelle solution)
- ☐ A5.2.1 - Exploitation des référentiels, normes et standards adoptés par le prestataire informatique

Important : vous devez présenter des tests vraisemblables et représentatifs
pour chaque traitement.

Sommaire

1.	Contexte général et technique.....	3
A.	Description de l'entreprise	3
a.	Activité.....	3
b.	Organisation	3
B.	Le système informatique	3
a.	Infrastructure des serveurs	3
C.	Objectifs du projet.....	4
2.	Analyse des données	5
A.	Règles de gestion.....	5
B.	MCD (jMerise)	6
C.	MLD.....	7
D.	Modèle physique.....	8
E.	Validation de la base et tests.....	9
F.	Particularité de champs	9
G.	Script de la base de données	9
H.	Description des traitements	9
a.	Vue.....	9
b.	Déclencheurs standards	10
c.	Héritages.....	10
d.	Déclencheurs sur héritage	11
e.	Procédures stockées	13
3.	Framework Laravel	15
A.	Enchaînement des pages	15
B.	Connexion via délégué	16
C.	CRUD	17
4.	Service-web, Web Service	19
A.	Organisation des fichiers	19
B.	Service	19
C.	Coté client.....	20
5.	Conclusion	21
A.	Éléments restant à développer	21
B.	Difficultés rencontrées	21
C.	Notions apprises et comprises.....	21

1. Contexte général et technique

A. Description de l'entreprise

a. Activité

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 Etats-Unis. La France a été choisie comme témoin pour l'amélioration du suivi de l'activité de visite.

b. Organisation

Une conséquence de cette fusion, est la recherche d'une optimisation de l'activité du groupe ainsi constitué en réalisant des économies d'échelle dans la production et la distribution des médicaments (en passant par une nécessaire restructuration et vague de licenciements), tout en prenant le meilleur des deux laboratoires sur les produits concurrents.

L'entreprise compte 480 visiteurs médicaux en France métropolitaine (Corse comprise), et 60 dans les départements et territoires d'outre-mer. Les territoires sont répartis en 7 régions géographiques (Paris-Centre, Sud, Nord, Ouest, Est, DTOM Caraïbes-Amériques, DTOM Asie-Afrique).

B. Le système informatique

a. Infrastructure des serveurs

Environnement :

- **SGBD** : SQL Server (serveur SRVLYCJ)
- **Environnement de développement** : Laragon (local)
 - Apache, PHP 7 (PDO3, protocole Webservice : SOAP)
 - Framework : Laravel

Outils de développement

Logiciels utilisés :

- **Gestion des versions** : Gitlab
- **Client et serveur FTP** : Filezilla
- **Gestion base de données** : SQL Server management studio
- **Modélisation** : jMerise
- **IDE** : Visual studio Code

Normes et conventions utilisées

Convention de nommage, base de données :

- **Convention principale** : *Camel*
- **Nom de table** : pluriel
- **Clé primaire** : *id<Nom de table avec majuscule> (sauf exception)*

Convention de codage, nommage appliqué par *Laravel*.

- Architecture du projet : Model, View, Controller (MVC)

C. Objectifs du projet

Mise en place d'une application WEB à disposition du personnel du laboratoire permettant de gérer les activités des visiteurs médicaux. (*Visites et Activités Complémentaires*)

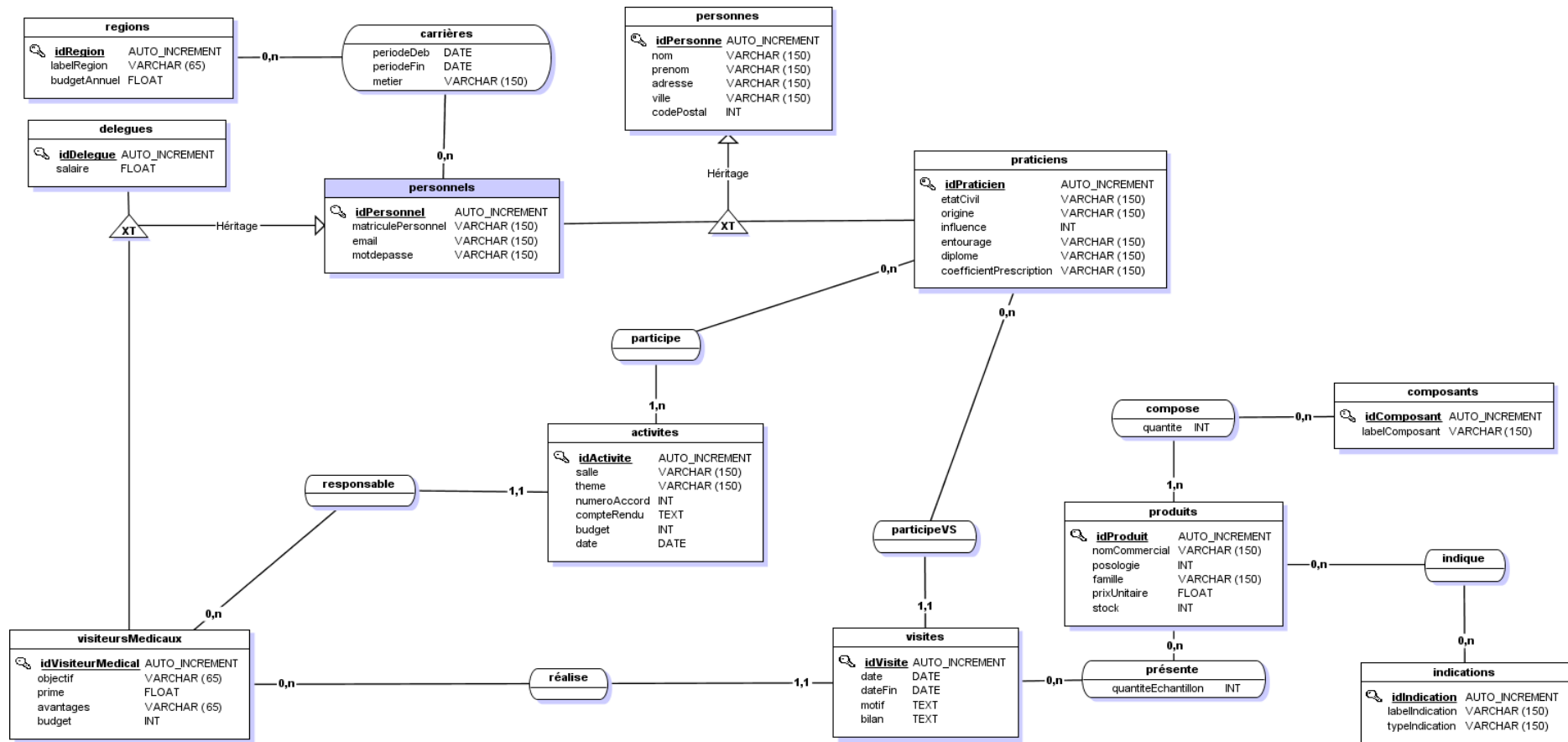
L'application doit proposer une authentification sécurisée.

2. Analyse des données

A. Règles de gestion

- Membre du personnel
 - Possède une identité, matricule, date d'embauche.
 - Matricule sous format : 'DUJE14' (Dupont Jean, identifiant : 14)
 - Connaître les différentes régions par lesquelles il est passé au cours de sa carrière (une région peut être en double pour le même personnel).
 - Distingue le visiteur médical et les autres membres du personnel.
 - Les visiteurs médicaux possèdent un objectif, une prime, des avantages et un budget.
- Produits
 - Identifiés par un numéro de produit.
 - Possède un nom commercial.
 - Effets thérapeutiques (indications), contre-indications.
 - Compositions du médicament (liste des composants et quantité).
 - La posologie (quantité périodique ~ *par type d'individu : adulte, jeune adulte, enfant, jeune enfant ou nourrisson*).
 - Catégorie de famille (antihistaminique, antidépresseur, antibiotique, ...)
 - Prix d'un échantillon
 - Quantité en stock
- Praticiens
 - Les diverses informations d'état civil et d'origine.
 - Influence du praticien (par coefficient, notoriété).
 - Entourage professionnel (est-il prescripteur, membre d'une association, relais de l'ordre des médecins...).
 - Diplôme (on ne conserve que le plus haut niveau).
 - Coefficient de prescription (sont-ils reconnus par leurs collègues comme référents sur la spécialité, sont-ils dans un cabinet pointu sur le sujet, ...)
- Régions
 - Dispose d'un budget global annuel
- Missions des visiteurs :
 - Visites :
 - Réalisé auprès d'un praticien
 - Connaître la date, le motif (rédigé librement par le visiteur)
 - Les médicaments présentés et le nombre d'échantillons offerts de chaque médicament
 - Bilan fourni par le visiteur (le médecin a paru convaincu ou pas, une autre visite a été planifiée...).
 - Activités complémentaires :
 - Participation des praticiens.
 - Numéro d'ordre pour accord (ce numéro servant pour l'engagement des frais).
 - Salle utilisée.
 - Compte rendu sous forme de texte-libre.
 - Visiteur responsable de l'AC.
 - Thème de l'activité.
 - Budget accordé pour l'activité

B. MCD (jMerise)



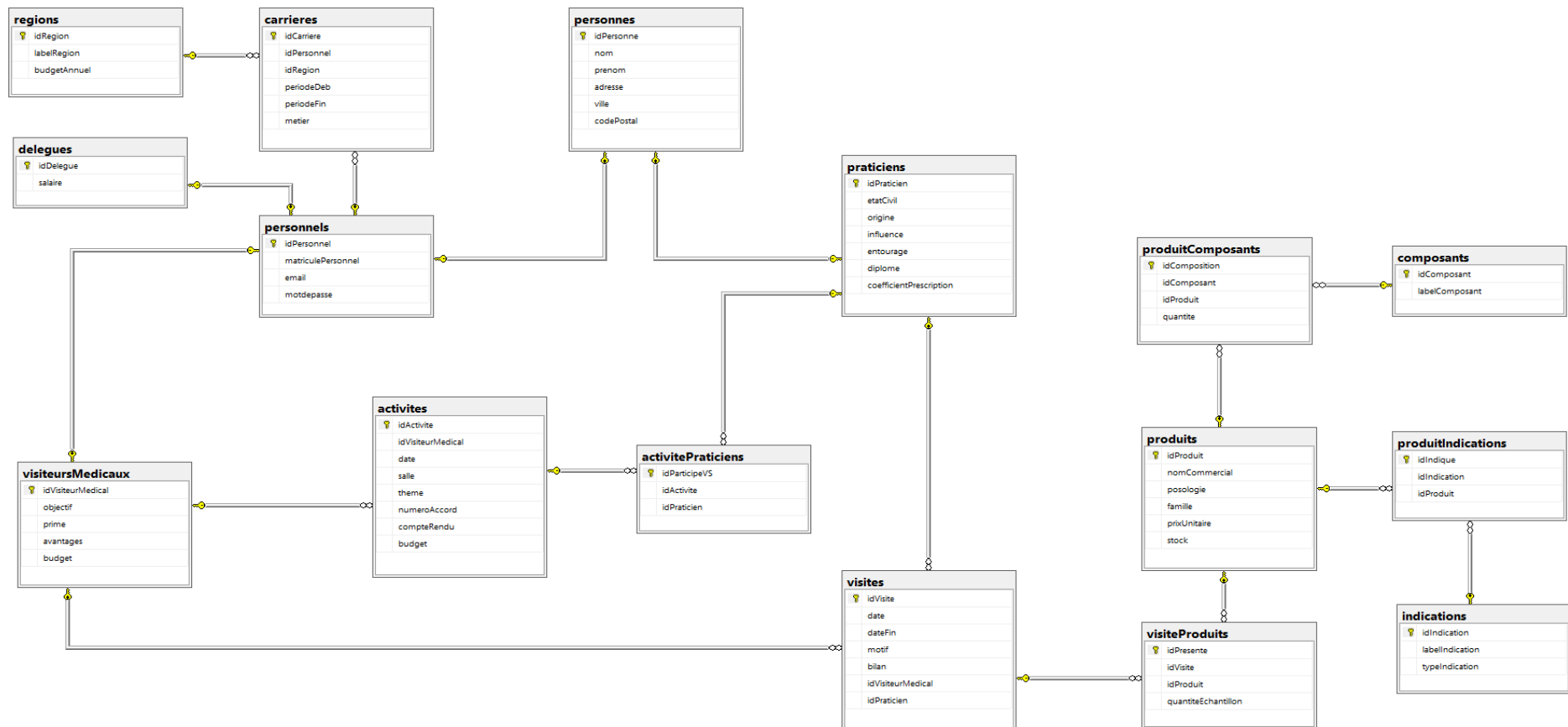
C. MLD

personnes (idPersonne, nom, prenom, adresse, ville, codePostal)
praticiens (idPraticien, etatCivil, origine, influence, entourage, diplôme, coefficientPrescription)
personnels (idPersonnel, matriculePersonnel, email, motdepasse, dateEmbauche)
visiteurMedicaux (idVisiteurMedical, objectif, prime, avantages, budget)
delegates(idDelegue, salaire)
regions (idRegion, labelRegion, budgetAnnuel)
carrières (idCarriere, #idRegion, #idPersonnel, periodeDeb, periodeFin, metier)
activites (idActivite, #idVisiteurMedical, date, salle, theme, numeroAccord, compteRendu, budget)
activitePraticiens(idParticipeVS, #idActivite, #idPraticien)
produits (idProduit, nomCommercial, posologie, famille, prixUnitaire, stock)
composants (idComposant, labelComposant)
produitComposants (idComposition, #idComposant, #idProduit, quantite)
indications (idIndication, labelIndication, typeIndication)
produitIndications (idIndique, #idIndication, #idProduit)
visites (idVisite, date, dateFin, motif, bilan, #idVisiteurMedical, #idPraticien)
visiteProduits (idPresente, #idVisite, #idProduit, quantiteEchantillon)

histoVisites(idHistoVisite, idVisite, idVisiteurMedical, idPraticien, date, motif, bilan, dateFin)

D. Modèle physique

Diagramme des relations SQL Server



E. Validation de la base et tests

Points à traiter :

- Vérification des types de données
- Respect des conventions de nommage
- Contraintes sur champs [PK, index, NULL interdit, ...]
 - Personnels
 - Matricule **champ UNIQUE**
 - Email **champ UNIQUE**
- Relations et leur contrainte [FK, contrainte de référence, effacement en cascade]
 - Suppression en cascade de « Personnes » vers « Personnels »
 - Suppression en cascade de « Personnels » vers « VisiteurMedicaux »
 - Suppression en cascade de « Personnels » vers « Delegates »
- Jeux d'essai représentatifs et vraisemblables

F. Particularité de champs

- **dateEmbauche** dans la table « *Personnels* » prend comme valeur par défaut : « getdate() »
- **prime** dans la table « *VisiteursMedicaux* » prend comme valeur par défaut : 0
- **budget** dans la table « *VisiteursMedicaux* » prend comme valeur par défaut : 0

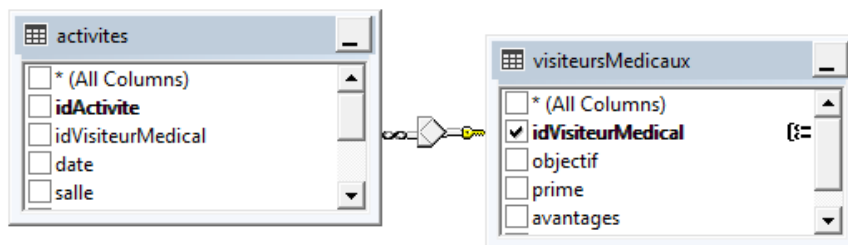
G. Script de la base de données

Script de base de données : « PPE3/PPE3_creation.sql »

H. Description des traitements

a. Vue

Récupérer le nombre d'activités complémentaires fait pour chaque visiteur Médical. (*dbo.activiteParVisiteur*)



1. **SELECT** COUNT(dbo.visiteursMedicaux.idVisiteurMedical) **AS** [Nombre de AC], dbo.visiteursMedicaux.idVisiteurMedical
2. **FROM** dbo.activites **LEFT OUTER JOIN** dbo.visiteursMedicaux **ON** dbo.visiteursMedicaux.idVisiteurMedical = dbo.activites.idVisiteurMedical
3. **GROUP BY** dbo.visiteursMedicaux.idVisiteurMedical

b. Déclencheurs standards

Historique des visites. (*dbo.visites -> histoVisite*)

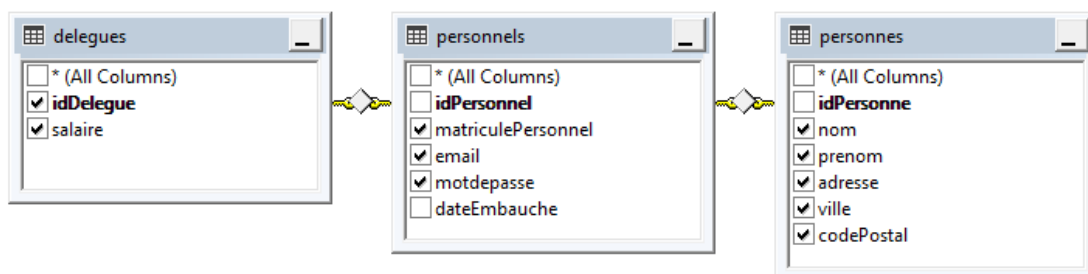
```
1. CREATE TRIGGER histoVisite ON visites
2. AFTER DELETE
3. AS
4. BEGIN
5.
6. INSERT INTO histoVisites(idVisite, date, dateFin, motif, bilan, idVisiteurMedical, idPraticien)
7.     SELECT * FROM deleted;
8. END
```

Création du matricule personnel. (*dbo.personnels -> matriculePersonnel*)

```
1. CREATE TRIGGER matriculePersonnel ON personnels
2. AFTER INSERT, UPDATE
3. AS
4. BEGIN
5.
6.     DECLARE @matricule VARCHAR(30),
7.             @idPersonne INT,
8.             @nom VARCHAR(150),
9.             @prenom VARCHAR(150);
10.
11.     SELECT @idPersonne = personnes.idPersonne, @nom = nom, @prenom = prenom FROM inserted
12.     LEFT JOIN personnes ON personnes.idPersonne = inserted.idPersonnel;
13.
14.     SET @matricule = UPPER(SUBSTRING(@nom, 1, 2) + SUBSTRING(@prenom, 1, 2)) + CAST(
15.         @idPersonne AS VARCHAR);
16.
17.     UPDATE personnels SET matriculePersonnel = @matricule
18.     WHERE idPersonnel = @idPersonne
19. END
```

c. Héritages

Vue héritage sur les visiteurs Médicaux. (*dbo.personnelVisiteur*)



```
1. SELECT visiteursMedicaux.budget, visiteursMedicaux.avantages, visiteursMedicaux.prim
2.     e, visiteursMedicaux.objectif, personnels.email, personnes.nom, personnes.prenom,
3.     personnes.adresse, personnes.ville, personnes.codePostal, personnels.motdepasse, perso
4.     nnes.idPersonne, personnels.matriculePersonnel, visiteursMedicaux.idVisiteurMedical
5. FROM personnels
6. INNER JOIN visiteursMedicaux ON personnels.matriculePersonnel = visiteursMedicaux.ma
7.     triculePersonnel
```

Rapport d'activité GSB

4. **INNER JOIN** personnes **ON** personnels.idPersonnel = personnes.idPersonne

Vue héritage sur Délégués

Vue héritage sur Praticiens

d. Déclencheurs sur héritage

Création d'un visiteur médical. (*dbo.personnelVisiteurs -> creationVisiteur*)

```
1. CREATE TRIGGER creationVisiteur ON personnelVisiteurs
2. INSTEAD OF INSERT
3. AS
4. BEGIN
5.
6.     DECLARE @nom VARCHAR(150),
7.             @prenom VARCHAR(150),
8.             @adresse VARCHAR(150),
9.             @ville VARCHAR(150),
10.            @codePostal INT,
11.            @email VARCHAR(150),
12.            @motdepasse VARCHAR(200),
13.            @objectif VARCHAR(65),
14.            @prime FLOAT,
15.            @avantages VARCHAR(65),
16.            @budget int;
17.
18.     SELECT @nom = nom,
19.            @prenom = prenom,
20.            @adresse = adresse,
21.            @ville = ville,
22.            @codePostal = codePostal,
23.            @email = email,
24.            @motdepasse = motdepasse,
25.            @objectif = objectif,
26.            @prime = prime,
27.            @avantages = avantages,
28.            @budget = budget
29.     FROM INSERTED
30.
31.
32.     INSERT INTO personnes(nom, prenom, adresse, ville, codePostal)
33.     VALUES(@nom, @prenom, @adresse, @ville, @codePostal);
34.
35.     INSERT INTO personnels(idPersonnel, email, motdepasse)
36.     VALUES(SCOPE_IDENTITY(), @email, @motdepasse);
37.
38.     INSERT INTO visiteursMedicaux(idVisiteurMedical, objectif, prime, avantages, bud
39.     get)
40.     VALUES(@@IDENTITY, @objectif, @prime, @avantages, @budget);
41. END
```

```
1. INSERT INTO personnelVisiteurs(nom, prenom, adresse, ville, codePostal, email,
2. motdepasse)
3. VALUES('Delaporte', 'Thomas', '17 Avenue jean de la fontaine', 'Milhaud', 30540, 'thomas.delaporte30@gmail.com',
4. '$2y$10$WKVLvpVFKlNTrcPehwKepOTJrAGiDp1jVgPWpdJCSA6lgcsxCLHMK')
```

Rapport d'activité GSB

Suppression des visiteurs médicaux. (*dbo.personnelVisiteurs -> suppressionVisiteur*)

```
1. CREATE TRIGGER suppressionVisiteur ON personnelVisiteur
2. INSTEAD OF DELETE
3. AS
4. BEGIN
5.
6.     DECLARE @idVisiteurMedical INT;
7.     SELECT @idVisiteurMedical = idVisiteurMedical FROM deleted;
8.
9.     DELETE FROM personnes WHERE idPersonne = @idVisiteurMedical;
10. END
```

Modification des données des visiteurs médicaux. (*dbo.personnelVisiteurs -> modificationVisiteur*)

```
1. CREATE TRIGGER modificationVisiteur ON personnelsVisiteur
2. INSTEAD OF UPDATE
3. AS
4. BEGIN
5.
6.     IF UPDATE(idVisiteurMedical)
7.     BEGIN
8.         RAISERROR('Impossible de modifier les clés primaires', 16, 1); ROLLBACK
9.     END
10.    ELSE
11.    BEGIN
12.
13.        DECLARE
14.            @idVisiteurMedical INT,
15.            @nom VARCHAR(100),
16.            @prenom VARCHAR(100),
17.            @adresse VARCHAR(150),
18.            @ville VARCHAR(150),
19.            @codePostal INT,
20.
21.            @email VARCHAR(150),
22.            @password VARCHAR(200),
23.
24.            @budget INT,
25.            @avantages VARCHAR(150),
26.            @prime INT,
27.            @objectif VARCHAR(150);
28.
29.        SELECT
30.            @nom = nom, @prenom = prenom, @adresse = adresse, @ville = ville, @codeP
31.            ostal = codePostal,
32.            @email = email, @password = motdepasse,
33.            @idVisiteurMedical = idVisiteurMedical, @budget = budget, @avantages = a
34.            vantages, @prime = prime, @objectif = objectif
35.        FROM inserted;
36.
37.        UPDATE personnes
38.        SET nom = @nom, prenom = @prenom, adresse = @adresse, ville = @ville, co
39.        dePostal = @codePostal
40.        WHERE idPersonne = @idVisiteurMedical;
41.
42.        UPDATE personnels
43.        SET email = @email, motdepasse = @password
44.        WHERE idPersonnel = @idVisiteurMedical;
45.
46.        UPDATE visiteursMedicaux
47.        SET budget = @budget, avantages = @avantages, prime = @prime, objectif =
48.        @objectif
49.        WHERE idVisiteurMedical = @idVisiteurMedical;
```

```
46.     END
47. END
```

Héritage praticien

- Création praticien.
- Modification praticien.
- Suppression praticien.

Héritage délégué

- Création déléguée.
- Modification déléguée.
- Suppression déléguée.

e. Procédures stockées

Budget total utilisé par un visiteur médical lors des activités complémentaires.

Argument : idVisiteurMedical

```
1. CREATE PROCEDURE budgetUtilise @idVisiteurMedical int, @budgetUtilise int OUTPUT
2. AS
3.     IF NOT EXISTS(SELECT idVisiteurMedical FROM visiteursMedicaux WHERE idVisiteurMedical = @idVisiteurMedical)
4.     BEGIN
5.         RAISERROR('Aucun visiteur médical avec cet identifiant', 10, 1); RETURN;
6.     END
7.
8.     SELECT @budgetUtilise = COALESCE(SUM(budget), 0) FROM activites
9.     WHERE idVisiteurMedical = @idVisiteurMedical
10.
11. GO

1. DECLARE @budgetUtilise INT;
2. EXEC budgetUtilise 1, @budgetUtilise OUTPUT
3.
4. PRINT(@budgetUtilise);
```

Seuils produits en stock

Arguments : seuil, idProduit (Optionnel)

```
1. CREATE PROCEDURE produitsSeuil @seuil int, @idProduit int = NULL
2. AS
3.     IF @idProduit IS NULL
4.         SELECT * FROM produits WHERE stock < @seuil;
5.     ELSE
6.         IF NOT EXISTS(SELECT idProduit FROM produits WHERE idProduit = @idProduit)
7.             RAISERROR('Aucun produit avec cet identifiant', 10, 1);
8.         ELSE SELECT * FROM produits WHERE stock < @seuil AND idProduit = @idProduit;
9. GO
10.
11. EXEC produitsSeuil 4;
12. EXEC produitsSeuil 4,5;
```

Visites par date

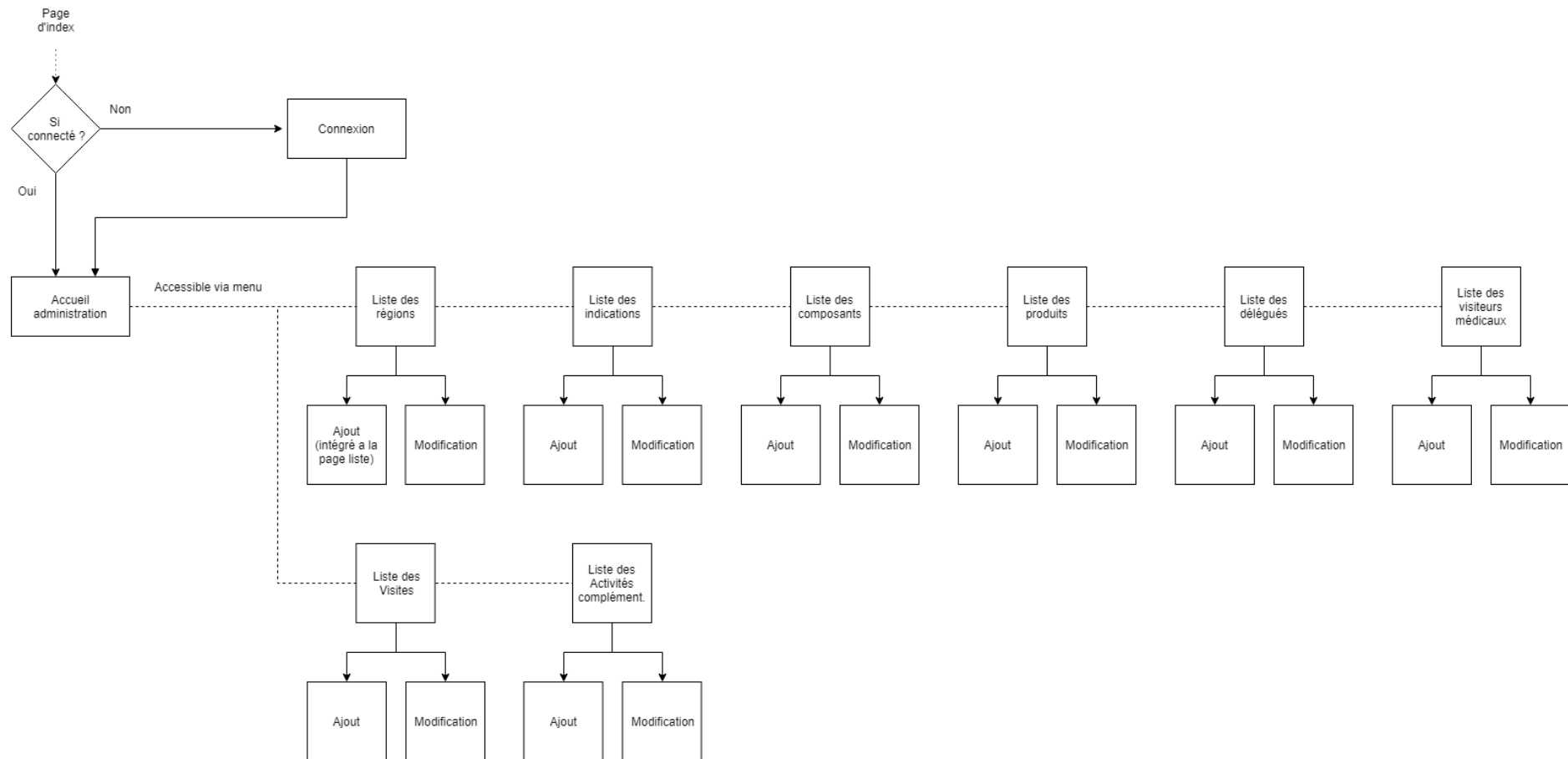
Arguments : Année, Mois (Optionnel), Jour (Optionnel)

```
1. CREATE PROCEDURE visitesDate @year int, @month int = NULL, @day int = NULL
2. AS
3.
4.     IF @day IS NOT NULL
5.
6.         SELECT * FROM visites
7.         WHERE DAY(date) = @day AND MONTH(date) = @month AND YEAR(date) = @year
8.     ELSE IF @month IS NOT NULL
9.
10.        SELECT * FROM Visites
11.        WHERE MONTH(date) = @month AND YEAR(date) = @year
12.    ELSE
13.
14.        SELECT * FROM Visites
15.        WHERE YEAR(date) = @year
16. GO

1. EXEC visitesDate 2018;
2. EXEC visitesDate 2018, 6;
3. EXEC visitesDate 2018, 6, 10;
```

3. Framework Laravel

A. Enchainement des pages



B. Connexion via délégué

La connexion pour atteindre l'administration est limitée aux délégués. Tout ce fait à travers des champs « email » et « motdepasse » qui se trouvent dans la table « Personnels ».

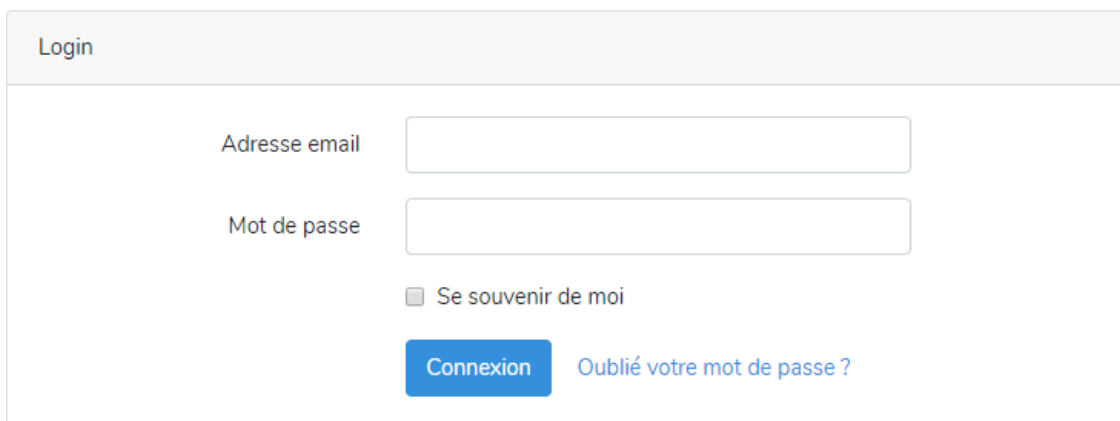
Le système est basé sur l'implémentation de base fournis par Laravel : « **Auth** »

L'inscription a été désactivé.

Model Delegate

```
1. class Delegate extends Authenticatable {
2.
3.     use Notifiable;
4.
5.     protected $table = 'personnelDelegates';
6.     protected $primaryKey = 'idDelegate';
7.
8.     public $timestamps = false;
9.
10.    protected $hidden = [
11.        'motdepasse'
12.    ];
13.
14.    protected $guarded = [];
15.
16.    public function getAuthPassword() {
17.        return $this->motdepasse;
18.    }
19. }
```

Page de connexion



Login

Adresse email

Mot de passe

☐ Se souvenir de moi

[Oublié votre mot de passe ?](#)

Le mot de passe est basé sur la fonction de hachage de la librairie « bcrypt ».

Compte de test :

- Identifiant : delegate@cci.fr
- Mot de passe : cci

C. CRUD

L'ensemble de l'administration est composé de 9 CRUD principaux :

- Visites
- Activités complémentaires
 - Invitation des praticiens
- Régions
- Indications
- Composants
- Produits
 - Ajout / Suppression des composants pour un produit.
 - Ajout / Suppression des indications pour un produit.
- Délégués
- Praticiens
- Visiteurs médicaux

Chaque CRUD fonctionne sur le même principe :

Fichiers :

- Un **model** : situe dans le fichier « /app »
- Un **controller** : situe dans le fichier « /app/Http/Controllers »
- **Routing** : « /routes/web.php »

Exemple de routing pour le CRUD Visites :

```
1. Route::prefix('visite')->group(function(){
2.
3.     Route::get('', 'VisiteController@list')->name('visite.list');
4.     Route::get('item/{id}', 'VisiteController@item')->name('visite.item');
5.
6.     Route::get('create', 'VisiteController@create')->name('visite.create');
7.     Route::post('store', 'VisiteController@store')->name('visite.store');
8.
9.     Route::get('edit/{id}/', 'VisiteController@edit')->name('visite.edit');
10.    Route::patch('update/{id}', 'VisiteController@update')->name('visite.update');
11.
12.    Route::delete('destroy/{id}', 'VisiteController@destroy')->
    >name('visite.destroy');
13. });
```

Rapport d'activité GSB

Liste des visites				
ID	Date	Visiteur	Praticien	Action
1	2019-05-07 00:00:00.000	Metivier	Gregoire	<button>Modifier</button> <button>Supprimer</button>

Création d'une visite

Date:

Visiteur médical:

Praticien:

Ajouter

Modification visite : 1

Date:

Visiteur médical:

Praticien:

Mettre à jour

Visite : 1

Date:

Date fin:

Visiteur:

Praticien:

Motif:

Bilan:

Produit présentés

Nom	Quantité
Spasfon	5

4. Service-web, Web Service

A. Organisation des fichiers

Le Webservice est structuré comme ceci :

- **src**
 - **core** : Fichiers utilitaires (connexion base de données)
 - **models** : Fichiers représentant les tables de la base de données
 - **operations** : Toutes les opérations ou méthodes gérées par le Webservice séparées sous forme de fichier.
 - config.ini
 - index.php

Le projet utilise l'auto-loader (psr-4) proposé dans composer.

B. Service



services.wsdl

Liste des méthodes :

- Information d'un délégué en donnant son identifiant
 - *Méthode* : getDelegateById
 - *Entrée* : \$idDelegate : entier
 - *Sortie* : Objet Delegate
- Information d'une visite en donnant son identifiant
 - *Méthode* : getVisiteById
 - *Entrée* : \$idVisite : entier
 - *Sortie* : Objet Visite
- Liste des visites par rapport au visiteur médical
 - *Méthode* : getVisitesByVisiteur
 - *Entrée* : \$idVisiteur: entier
 - *Sortie* : Liste d'objet Visite
- Récupérer un mot de passe haché avec la librairie BCrypt
 - *Méthode* : getHashPassword
 - *Entrée* : \$password: chaîne de caractère
 - *Sortie* : Chaîne de caractère haché

C. Coté client

Page d'accueil administration

Sur l'application « Laravel », le Webservice est utilisé dans la page d'accueil :

Information utilisateur connecté	WebService
idDelegate : 1	
Nom : Thomas	
Prénom : Delaporte	
Adresse : 17, Avenue jean de la fontaine	
Ville : Milhaud	
Code postal : 30540	
Email : thomas.delaporte30@gmail.com	
Salaire : 1250€	

Fonction « index » dans controller « HomeController »

```
1. public function index(){
2.
3.     $informations = null;
4.
5.     try{
6.
7.         $address = config('app.webservice');
8.         $client = new \SoapClient($address, [
9.             'trace' => 1,
10.            'exceptions' => 1
11.        ]);
12.
13.        $informations = $client->__soapCall('getDelegateById', ['idDelegate' => Auth::id()]);
14.    } catch(Throwable $ignored){}
15.
16.    return view('home', ['informations' => $informations]);
17. }
```

5. Conclusion

A. Éléments restant à développer

Détailler la posologie des produits pour chaque type d'individu : *adulte, jeune adulte, enfant, jeune enfant ou nourrisson*.

Administration disponible pour d'autre rôle avec des fonctionnalités différentes :

- Responsable de région : assigner un numéro d'ordre, un budget pour les activités.
- Visiteurs médicaux : pouvoir gérer ses visites et ses activités complémentaires.

B. Difficultés rencontrées

Connexion a la base de données (problème d'installation de driver)

C. Notions apprises et comprises

Création de déclencheur.

Création de vue (vue sur héritage).

Utilisation et compréhension de Laravel :

- Création de Model
- Création de Controller
- Utilisation / Modification des routes
- Utilisation des middlewares

Création d'un Webservice :

- Création / Modification du fichier .wsdl (*Web Services Description Language*)
- Utilisation des class **SoapClient** / **SoapServer**