

APPLICATION COMPTES-RENDUS



DOCUMENTATION GSB

DESCRIPTION D'UNE RÉALISATION PROFESSIONNELLE		N° réalisation : 2
Nom, prénom : DOMAS-VASSEROT Jade		N° candidat : 02146788571
Épreuve ponctuelle <input checked="" type="checkbox"/>	Contrôle en cours de formation <input type="checkbox"/>	Date : 24 / 04 / 2022
Contexte de la réalisation professionnelle L'organisation Galaxy Swiss Bourdin est un laboratoire médical qui entreprend des recherches à propos des soins médicaux. Les salariés sont amenés à se déplacer. A chaque visite, ils doivent faire un compte-rendu de la visite en renseignant le praticien, les informations et les produits.		
Intitulé de la réalisation professionnelle Application pour la gestion des comptes rendus		
Période de réalisation : 2^{ème} année Lieu : Lyon Modalité : <input checked="" type="checkbox"/> Seul(e) <input type="checkbox"/> En équipe		
Compétences travaillées <input checked="" type="checkbox"/> Concevoir et développer une solution applicative <input checked="" type="checkbox"/> Assurer la maintenance corrective ou évolutive d'une solution applicative <input checked="" type="checkbox"/> Gérer les données		
Conditions de réalisation¹ (ressources fournies, résultats attendus) Un cahier des charges concernant le projet. Un aperçu d'une pré-version attendue sur Access. Résultats attendus : Développement d'une application web sécurisé et authentifié pour permettre la gestion des comptes-rendus Création d'un espace visiteur où l'on peut voir ses comptes-rendus ainsi qu'en saisir (une page profil également). Un espace où l'on peut visualiser ou chercher les différentes informations concernant un visiteur, un praticien ou un médicament.		
Description des ressources documentaires, matérielles et logicielles utilisées² On a pu utiliser la documentation et les ressources données (Cahier des charges et prévisualisation de l'appli sur access). <u>Logiciels :</u> PhpMyAdmin (MySQL), WampServer, Visual Studio Code, Navigateur Chrome, Apache, Lucid.app <u>Langages :</u> Framework : Symfony (Php, Twig) et Vue JS (HTML, CSS, JavaScript)		
Modalités d'accès aux productions³ et à leur documentation⁴ <u>Documentation :</u> https://jadedomasvasserot.fr/index.php/ppe-2/ et dans le dossier doc du projet sur GitHub https://github.com/JadeDomasVasserot/GSB_BTS_CR/tree/main/doc <u>Accès aux productions :</u> Versionning sur GitHub : https://github.com/JadeDomasVasserot/GSB_BTS_CR		

¹ En référence aux *conditions de réalisation et ressources nécessaires* du bloc « Conception et développement d'applications » prévues dans le référentiel de certification du BTS SIO.

² Les réalisations professionnelles sont élaborées dans un environnement technologique conforme à l'annexe II.E du référentiel du BTS SIO.

³ Conformément au référentiel du BTS SIO « Dans tous les cas, les candidats doivent se munir des outils et ressources techniques nécessaires au déroulement de l'épreuve. Ils sont seuls responsables de la disponibilité et de la mise en œuvre de ces outils et ressources. La circulaire nationale d'organisation précise les conditions matérielles de déroulement des interrogations et les pénalités à appliquer aux candidats qui ne se seraient pas munis des éléments nécessaires au déroulement de l'épreuve. ». Les éléments peuvent être un identifiant, un mot de passe, une adresse réticulaire (URL) d'un espace de stockage et de la présentation de l'organisation du stockage.

⁴ Lien vers la documentation complète, précisant et décrivant, si cela n'a été fait au verso de la fiche, la réalisation professionnelle, par exemple service fourni par la réalisation, interfaces utilisateurs, description des classes ou de la base de données.

SOMMAIRE

- I. LE CONTEXTE GSB DE L'APPLI DE GESTION COMPTES-RENDUS***
- II. CONFIGURATION DE L'ENVIRONNEMENT SYMFONY + VUE JS***
- III. MISE EN PLACE DE LA BASE DE DONNEES***
- IV. DEVELOPPEMENT DE LA CONNEXION***
- V. DEVELOPPEMENT DE L'INTERFACE MEDICAMENTS***
- VI. DEVELOPPEMENT DE L'INTERFACE PRATICIENS***
- VII. DEVELOPPEMENT DE L'INTERFACE VISITEURS***
- VIII. DEVELOPPEMENT DE L'INTERFACE DU VISITEUR CONNECTE PROFIL ET VOIR SES COMPTES-RENDUS***
- IX. DEVELOPPEMENT DE LA CREATION DE COMPTE-RENDU***

CONTEXTE GSB – APPLI **COMPTE-RENDU**

L'organisation Galaxy Swiss Bourdin (GSB) est un laboratoire médical. Elle résulte de la fusion en Galaxy (spécialiste dans les maladies virales comme le SIDA) et Swiss Bourdin (plutôt sur les médicaments). Ils ont une équipe d'employés amenés à participer à des réunions et visiter d'autres laboratoires. Après chaque visite, il faut élaborer un compte rendu de la visite

renseignant quel praticien on est allé voir, s'il était remplacé, la date de la visite et les informations complémentaires.

Pour cela, il nous a été demandé de développer une application web dans le but de centraliser les comptes-rendus.

CONFIG DE SYMFONY/VUE JS

Dans un premier temps, il nous fallut créer un projet Symfony grâce à :

`Symfony new`

Puis, j'ai téléchargé les packages Vue et Encore (package pour permettre au fichier twig de lire le Vue JS) grâce à npm et yarn.

J'ai activé dans le package Encore : Vue Loader

```
// Enable Vue loader
.enableVueLoader()
```

J'ai ensuite configuré mes fichiers twig.

Un fichier est présent dans le répertoire /doc du GitHub, j'ai suivi cet exemple.

```
{% block stylesheets %}
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/c
{% endblock %}
<div class="text-center">
    
</div>
<div id="app" class="mt-5"> {% block body %}{% endblock %}
</div>
{% block javascripts %}
    {{ encore_entry_script_tags('app') }}
{% endblock %}

{# Fichier twig pour la page d'accueil qui permet de se connecter
appelle le composant Home.vue #}
{% extends 'base.html.twig' %}
{% block body %}

<div class="text-center">
    <home></home>
</div>

{% endblock %}
```

2 commandes pour lancer le symfony et le vue :

`symfony serve:start`

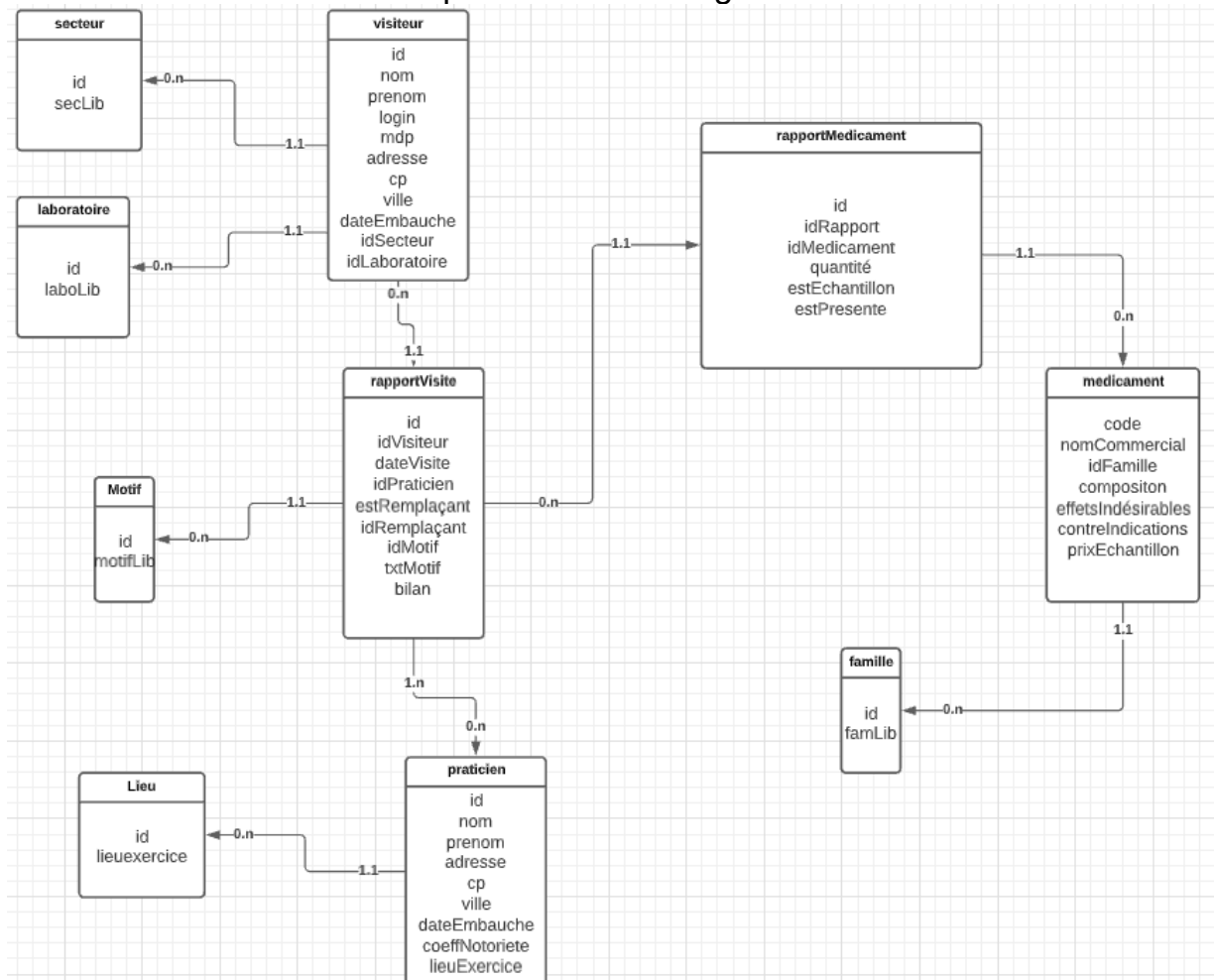
`yarn encore dev-server --hot`

J'ai décidé d'utiliser un outil que je connaissais qui permet de créer des API basiques. Il s'agit d'API Platform. Je l'ai donc installé via

`Composer require api`

MISE EN PLACE DE LA BDD

J'ai tout d'abord fait la conception de ma BDD grâce à Lucid.



J'ai ensuite créé mes entités avec :

Php bin/console make :entity

J'ai renseigné pour chaque les différents champs et type.

```
namespace App\Entity;

use Doctrine\ORM\Mapping as ORM;
use ApiPlatform\Core\Annotation\ApiResource;
/**
 * Famille
 *
 * @ORM\Table(name="famille")
 * @ORM\Entity
 * @ApiResource()
 */
class Famille
{
    /**
     * @var int
     *
     * @ORM\Column(name="idFamille", type="integer", nullable=false)
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="IDENTITY")
     */
    private $idFamille;

    /**
     * @var string
     *
     * @ORM\Column(name="famLib", type="string", length=25, nullable=false)
     */
    private $famLib;
    public function getIdFamille(): ?int
    {
        return $this->idFamille;
    }

    public function getFamLib(): ?string
    {
        return $this->famLib;
    }

    public function setFamLib(string $famLib): self
    {
        $this->famLib = $famLib;

        return $this;
    }
}
```

Grâce à **php bin/console make:entity --regenerate App**

J'ai généré les getters/setters
Il faut bien annoter
@ApiResource pour que
Api platform puisse lire l'entité

Puis j'ai fait un **php bin/console doctrine:database:create**

• **php bin/console doctrine:schema:create**

Ce qui permet de créer la BDD

J'ai ensuite inséré des
données.

DEVELOPPEMENT CONNEXION

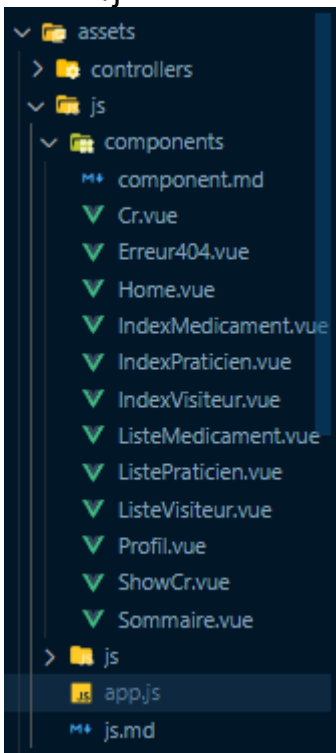
J'ai créé le composant HomeController grâce à la commande `php bin/console make:controller HomeController`

Dans le dossier templates j'ai créé [home.html.twig](#) dans le dossier /Home.

Il s'étend de base.html.twig où l'on ajoute notre config pour le vue JS. Il appelle le composant `<home></home>`

```
/**
 * @Route("/", name="app_home")
 * affiche la page home de connexion
 */
public function index()
{
    return $this->render('Home/home.html.twig');
}
```

Dans mon répertoire /assets qui comprend mes fichiers VueJS dans le répertoire assets/js.



Dans le fichier app.js dans ce dernier j'ajoute mon composant VueJS Home.vue à app.js.

```
import Home from '../components/Home.vue'
new Vue({
  el: '#app',
  components: {
    'home': Home,
```

J'ai créé un répertoire js et components dans le répertoire assets/js où je crée un fichier Home.vue et Home.js.

Home.js correspond au javascript du composant VueJS

```
<script src="../../js/Home.js">
</script>
```

Tandis que Home.vue comprendra le code front de la page. C'est un formulaire avec 2 inputs text permettant à l'utilisateur de se connecter via l'action /connexion. J'ai créé des fonctions JS permettant à l'utilisateur d'envoyer le formulaire avec entrée ainsi que de gérer l'état disabled ou non du bouton se connecter.



CONNEXION

Nom d'utilisateur*

Mot de passe*

Connexion

Du côté back, j'ai créé la route /connexion qui permet de récupérer les variables POST. J'effectue une recherche dans la BDD par le nom d'utilisateur entré. Puis, on vérifie que la password entré correspond bien à celui en rapport avec celui de l'user. Si c'est le cas on débute une session grâce à SessionInterface de Symfony et je set les variables dans la session. Je return la vue profilVisiteur.html.twig dont on parlera plus tard si ça marche sinon sur erreurConnexion.html.twig si le mot de passe ne correspond pas. S'il y a une erreur avec le serveur, on affiche la page erreur404.html.twig (chaque twig est relié à un composant Vue JS).

J'ai créé en parallèle ma route /deconnexion qui déconnecte l'utilisateur et clear la session. Elle amène sur la page twig deconnexion.html.twig et remet le formulaire pour se connecter.

Vous avez bien été déconnecté

CONNEXION

Nom d'utilisateur*

Mot de passe*

Connexion

J'ai aussi créé le composant sommaire qui est ajouté à chaque vue.

DEVELOPPEMENT MEDICAMENTS

Pour l'interface médicament, il y a la partie « Liste des médicaments » et l'autre partie qui est « Choisir un médicament ».

J'ai créé le Controller : `MediacamentController` qui définit les différentes routes pour nos pages twigs.

```
/**
 * @Route("/", name="app_home_medicament")
 * affiche la page home où l'on choisit un médicament
 */
public function index()
{
    return $this->render('Medicament/index.html.twig');
}
/**
 * @Route("/liste", name="app_liste_medicament")
 * affiche la liste des médicaments
 */
public function liste()
{
    return $this->render('Medicament/liste.html.twig');
}
```

Le fonctionnement sera le même pour les interfaces praticiens et visiteurs, je n'expliquerais donc que pour l'exemple de médicament.

1) Pour la vue « Liste des médicaments ».

J'ai créé un composant `ListeMedicament.vue` que j'ai ajouté dans mon `app.js`.
Mon fichier twig appelle ce composant ainsi que le sommaire en haut.

```
{# Fichier twig pour la page liste des médicaments, appelle le composant ListeMedicament.vue #}
{% extends 'base.html.twig' %}
{% block body %}

<div class="text-center">
    <sommaire></sommaire>
    <listemedicament></listemedicament>
</div>

{% endblock %}
```

J'ai utilisé un tableau avec du css bootstrap pour faire la Vue.


Je fais un appel axios (j'ai installé le package axios avec npm avant) pour récupérer l'ensemble des données de l'item grâce à une API créée par API Platform.


```
import axios from 'axios';
import moment from 'moment';
import Vue from 'vue';
Vue.prototype.moment = moment;
moment.updateLocale('fr');
export default {
  name: 'listemedicament',
  data() {
    return {
      medicaments: axios.get('http://127.0.0.1:8000/api/medicaments').then(rep => this.medicaments = rep.data)
    }
  },
};
```

Avec le Vue JS j'affiche les informations grâce à un v-for

```
<template>  
  <div>  
    <table class="table table-light table-hover text-center">  
      <thead>  
        <tr>  
          <th scope="col">Nom commercial</th>  
          <th scope="col">Composition</th>  
          <th scope="col">Effets indésirables</th>  
          <th scope="col">Contre-indications</th>  
          <th scope="col">Prix de l'échantillon</th>  
          <th scope="col">Famille</th>  
        </tr>  
      </thead>  
      <tbody>  
        <tr>  
          v-for="medicament in médicaments['hydra:member']"  
          :key="medicament.idmedicament"  
          :value="medicament.idmedicament"  
        >  
          <td>{{medicament.nomcommercial}}</td>  
          <td>{{medicament.composition}}</td>  
          <td>{{medicament.effetsindesirables}}</td>  
          <td>{{medicament.contreindications}}</td>  
          <td>{{medicament.prixechantillon}}</td>  
          <td>{{medicament.idfamille.familib}}</td>  
        </tr>  
      </tbody>  
    </table>  
  </div>  
</template>
```

Pour praticien et visiteur j'utilise le package Moment afin de formater la date.



[Créer un compte-rendu](#)
[Voir ses comptes-rendus](#)
[Profil](#)
[Choisir un visiteur](#)
[Liste des visiteurs](#)
[Choisir un praticien](#)
[Liste des praticiens](#)
[Choisir un médicament](#)
[Liste des médicaments](#)
[Déconnexion](#)

Nom commercial	Composition	Effets indésirables	Contre-indications	Prix de l'échantillon	Famille
Doliprane	Paracétamol	Vertiges	1x par toutes les 3h	2	Gellule
Onctose	Crèmeux	Allergies	Pas à mélanger avec d'autres crèmes	5	Crème

2) Pour la vue « Choisir un médicament ».

J'ai créé un composant `IndexMedicament.vue` que j'ai ajouté dans mon `app.js`

Mon fichier twig appelle ce composant ainsi que le sommaire en haut.


Choisissez un médicament

Médicament :

▼

Doliprane
Onctose

Nom Commercial
Composition
Effets indésirables
Contre-indications
Prix de l'échantillon
Famille



Voici le rendu du template lorsque l'on n'a pas choisi de médicament.

Il y a donc un select pour choisir l'ensemble des médicaments dans la BDD grâce à Axios on va faire un appel API (avec API Plateform) afin de récupérer la liste des médicaments qu'on affiche par nom grâce à un v-for sur les options du select.

Ensuite, je récupère l'id passé en value du select ce qui me permettra de faire un autre appel Axios d'une Api de Api Platform qui me permettra d'obtenir les informations de 1 éléments grâce à son id.

Explication des méthodes :

Je fais un appel Axios pour récupérer la liste de l'item en question.

J'affiche l'ensemble grâce à un v-for avec un keys et une value pour chaque item qui sera l'id.

La personne clique sur le son choix qu'il souhaite dans le select avec v-model une variable afin de récupérer la valeur choisie.

Ce qui va lancer la méthode choixPraticien() qui fait l'appel Axios afin de récupérer les informations de l'item choisi.

Puis on affiche les information en vérifiant que l'item a bien été choisi avec la méthode isPraticienChoisi()

Avec le Vue JS j'affiche les informations.

Choisissez un médicament

Médicament :

▼

Onctose


Nom Commercial	Onctose
Composition	Crémeux
Effets indésirables	Allergies
Contre-indications	Pas à mélanger avec d'autres crèmes
Prix de l'échantillon	5 €
Famille	Crème



2 Onctose

DEVELOPPEMENT PRATICIENS

1) Liste des praticiens

 Créer un compte-rendu Voir ses comptes-rendus Profil Choisir un visiteur Liste des visiteurs Choisir un praticien Liste des praticiens Choisir un médicament Liste des médicaments Déconnexion							
Nom	Prénom	Adresse	CodePostal	Ville	Date d'embauche	Coefficient de notoriété	Lieu de l'exercice
Stephens	Jesse	8976 A Av.	8766	Tabuk	23/08/2021	8	Clinique
Russo	Carl	957 Ornare Street	13758	Oaxaca	29/06/2022	9	Hopital
Palmer	Shea	Ap #801-7689 A Av.	35452	Belfast	31/07/2021	1	Hopital
Luna	Abigail	P.O. Box 412, 9816 Nullam St.	65453	Ligney	20/04/2023	1	Privé
Myers	Miranda	P.O. Box 479, 6212 In St.	4640	Pangkalpinang	31/03/2023	6	Clinique

2) Choisir un praticien

PRATICIEN :
▼
Stephens - Jesse
Russo - Carl
Palmer - Shea
Luna - Abigail
Myers - Miranda



Choisissez un praticien

Nom de famille

Prénom

Adresse

Code Postal

Ville


Date d'embauche

Coefficient de notoriété

Lieu d'Exercice

Avant

PRATICIEN :
Stephens - Jesse ▼


6 Jesse Stephens

Choisissez un praticien

Nom de familleStephens

PrénomJesse

Adresse8976 A Av.

Code Postal8766

VilleTabuk

Date d'embauche23/08/2021

Coefficient de notoriété8

Lieu d'ExerciceClinique

Après

DEVELOPPEMENT VISITEURS

1) Liste des visiteurs



[Créer un compte-rendu](#)

[Voir ses comptes-rendus](#)

[Profil](#)

[Choisir un visiteur](#)

[Liste des visiteurs](#)

[Choisir un praticien](#)

[Liste des praticiens](#)

[Choisir un médicament](#)

[Liste des médicaments](#)

[Déconnexion](#)

Id du visiteur	Nom	Prénom	Adresse	CodePostal	Ville	Date d'embauche	Laboratoire	Secteur
bfr	Freeman	Bradley	P.O. Box 760, 3172 Orci. Rd.	90668	Pioneer	24/10/2021	Swiss	Ouest
jdj	DOMAS	Jade	A30 rue rebatel	69008	Lyon	23/07/2021	Galaxy	Nord
ola	Lane	Owen	Ap #890-9987 Sit St.	83803	Port Elizabeth	23/07/2022	Galaxy	Est
sri	Richard	Sopoline	947-8276 Arcu Avenue	6758	Asan	02/05/2021	Spectra	Nord
tgu	Guerra	Teegan	1864 Tellus. Avenue	6134	Jennersdorf	05/06/2021	Swiss	Ouest
tst	Strickland	Tamekah	6426 Ipsum. Avenue	81666	Muzaffarpur	01/11/2022	Galaxy	Ouest

2) Choisir un visiteur

Visiteur :

Freeman - Bradley


DOMAS - Jade

Lane - Owen

Richard - Sopoline

Guerra - Teegan

Strickland - Tamekah



Choisissez un visiteur

Nom de famille

Prénom

Adresse

Code Postal

Ville

Date d'embauche


Secteur

Laboratoire

Avant

Visiteur :

Freeman - Bradley



bfr Bradley Freeman

Choisissez un visiteur

Nom de famille

Prénom

Adresse

Code Postal

Ville

Date d'embauche

Secteur

Laboratoire

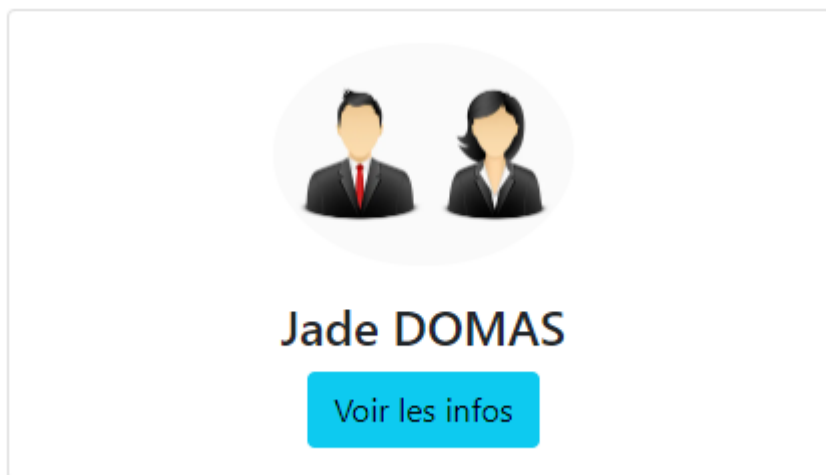
Après

DEVELOPPEMENT INTERFACE

UTILISATEUR CONNEXE

1) Espace Profil

J'ai créé un composant Profil.vue que j'ai ajouté dans mon app.js
Mon fichier twig appelle ce composant ainsi que le sommaire en haut.



On arrive sur cette page lorsqu'on clique sur « Profil » dans le sommaire.
Cela dirige vers la route /visiteur/

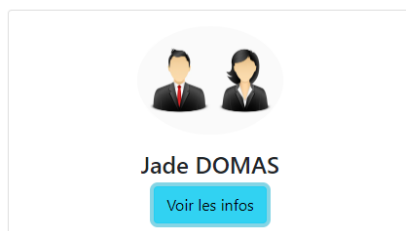
Dans notre vue, on appelle la requête axios /visiteur/session de l'API que j'ai créée, elle retourne un JSON avec les informations du visiteur dans la session. Puis, on affiche son nom et prénom.

On peut voir les informations si on clique sur « voir les infos ».

J'ai utilisé la même mise en page que pour « choisir le médicament ».

Lorsque l'utilisateur clique sur le bouton, cela lance une requête axios vers l'API.

Puis, j'affiche les informations si le bouton a été cliqué.



Nom de famille	DOMAS
Prénom	Jade
Login	jdomas
Adresse	A30 rue rebatel
Code Postal	69008
Ville	Lyon
Date d'embauche	23/07/2021
Laboratoire	Galaxy
Secteur	Nord

2) Espace «Voir ses comptes-rendus »

J'ai créé un composant ShowCr.vue que j'ai ajouté dans mon app.js
Mon fichier twig appelle ce composant ainsi que le sommaire en haut.

Je fais un appel API grâce à Axios de l'API /rapportvisite/show/list que j'ai créé. Je recherche les rapports de visites du visiteur enregistré dans la session.

Je return un tableau (array) au format JSON comprenant (id, dateVIsite, estRemplaçant, bilan, idmotif, motifText, le nom et prénom du praticien et celui du remplaçant s'il y en a un).

Id Rapport	Date de la visite	Praticien	Est remplaçant ?	Praticien Remplaçant	Bilan	Motif	Produits
18	30/03/2022	Shea Palmer	true	Carl Russo	Sollicit	Actualisation	Voir infos
21	10/06/2002	Carl Russo	true	Miranda Myers	Le praticien remplaçant semble OK	Autre	Voir infos
23	20/04/2022	Carl Russo	true	Abigail Luna	Remplacé par Luna Abigail	Autre	Voir infos
24	08/04/2022	Shea Palmer	true	Carl Russo	Bilan visite remplacée par russo Carl	Autre	Voir infos
25	06/04/2022	Abigail Luna	true	Miranda Myers	Visite 06/04/2022	Actualisation	Voir infos
26	12/04/2022	Carl Russo	false		Période	Périodicité	Voir infos
27	10/03/2022	Carl Russo	true	Jesse Stephens	Relance	Relance	Voir infos
28	13/04/2022	Jesse Stephens	false		OK pour mettre en place un module RGPD	Autre	Voir infos
29	28/03/2022	Miranda Myers	false		Actu	Actualisation	Voir infos
30	21/04/2022	Shea Palmer	true	Carl Russo	Visite	Autre	Voir infos
31	21/04/2022	Shea Palmer	true	Carl Russo	Visite	Autre	Voir infos

Nom du médicament	Est un échantillon ?	Est présenté ?	Quantité	Id du rapport sélectionné
Doliprane	false	true		29
Onctose	false	true		29
Doliprane	true	false	1	29

On peut cliquer sur le bouton « voir Infos » pour voir les détails Produits du rapport sélectionné.

On fait un appel API vers la route /rapportvisite/voirProduits/{id}

Dans notre back, cette route retourne un JSON. On cherche tout d'abord les RapportMedicament qui ont l'idRapport passé en paramètre.

Et pour chaque rapportMedicament, on a créé un tableau avec id, quantité, estEchantillon, estPresente, nomMedicament, idRapport.

On retourne un tableau comprenant tous les tableaux.

```
/**
 * @Route("/voirProduits/{id}", name="app_rapport_show_product", methods={"GET"})
 * page pour les infos des produits du rapport
 */
public function showPageProduct(int $id)
{
    $rapportsTabMedi = array();
    $rapportsMedi = $this->getDoctrine()
        ->getRepository(Rapportmedicament::class)
        ->findBy([
            'idRapport' => $id,
        ]);
    foreach($rapportsMedi as $rappMedi){
        $id = $rappMedi->getIdrapport()->getIdRapportvisite();
        $quantite = $rappMedi->getQuantite();
        $estEchantillon = $rappMedi->getEstechantillon();
        $estPresente = $rappMedi->getEstpresente();
        $nomMedica = $rappMedi->getIdmedicament()->getNomcommercial();
        $rappMediTab = array(
            'id' => $id,
            'quantite' => $quantite,
            'estEchantillon' => $estEchantillon,
            'estPresente' => $estPresente,
            'nomMedica' => $nomMedica,
            'idRapport' => $id,
        );
        array_push($rapportsTabMedi, $rappMediTab);
    }
    $response = new Response();
    $response->setContent(json_encode([$rapportsTabMedi]));
    $response->headers->set('Content-Type', 'application/json');

    return $response;
}
```

CREATION DE COMPTE-RENDU

J'ai créé un composant Cr.vue que j'ai ajouté dans mon app.js
Mon fichier twig appelle ce composant ainsi que le sommaire en haut.

J'ai créé du côté front-end le formulaire qui a comme action la route
/rapportvisite/new

Grâce à axios j'ai fait appel à des API de API Platform afin de récupérer la
liste des médicaments, des motifs et des praticiens

```

medicaments : axios.get('http://127.0.0.1:8000/api/medicaments').then(rep => this.medicaments = rep.data),
motifs : axios.get('http://127.0.0.1:8000/api/motifs').then(rep => this.motifs = rep.data),
praticiens: axios.get('http://127.0.0.1:8000/api/praticiens').then(rep => this.praticiens = rep.data),

```

Je peux ensuite les afficher dans des <select> et <option> grâce à un v-for.
Je récupère la valeur grâce à v-model sur le select en passant une variable

```

<select
  name="PRA_REPLACANT"
  :disabled="isSelectRemplaçant=='"
  class="form-select"
  v-model="selectRemplaçant"
>
  <option
    v-for="praticien in praticiens['hydra:member']"
    :key="praticien.idpraticien"
    :value="praticien.idpraticien"
  >{{ praticien.nom }} - {{ praticien.prenom }}</option>

```

J'ai géré la gestion de l'affichage et du comportement de mes inputs.
- Voir si remplaçant a été coché afin de disabled ou non le select du remplaçant.

REPLACANT :

☐

REPLACANT :

☒

Stephens - Jesse
Russo - Carl
Palmer - Shea
Luna - Abigail
Myers - Miranda

- Voir si un praticien a été choisi afin d'afficher son coefficient de notoriété en dessous

PRATICIEN :

COEFFICIENT :

PRATICIEN :

Russo - Carl

COEFFICIENT :

9

- Voir si le motifs choisi est == à "Autre" pour disabled ou non l'input text

MOTIF :

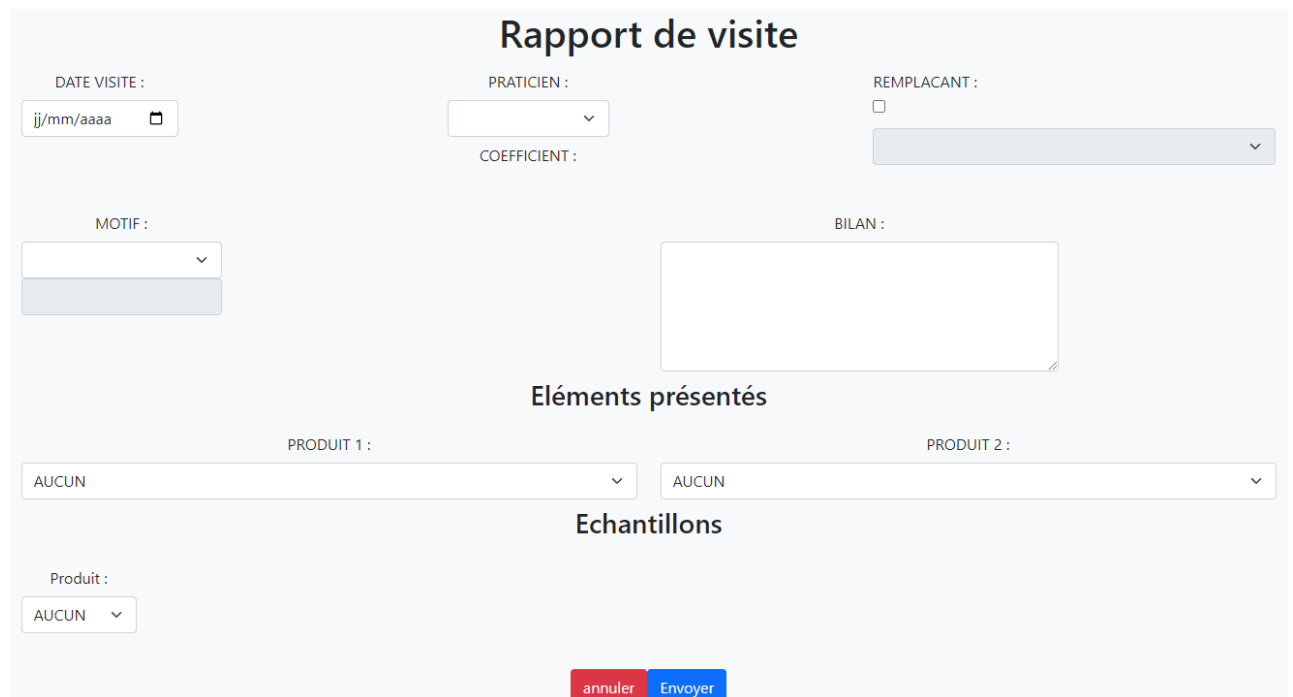
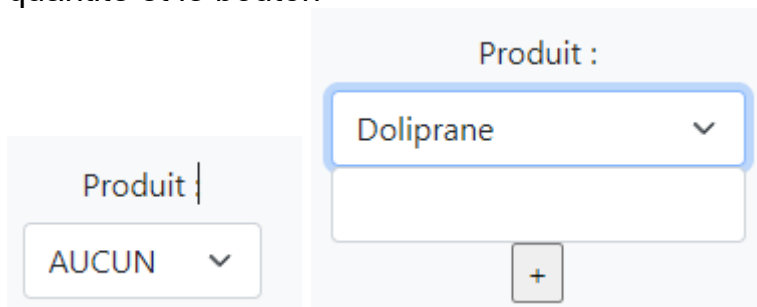
Sollicitation praticien

MOTIF :

Autre

FSGDSF

- Voir si le produit échantillon = AUCUN pour ne pas afficher (grâce à v-if) la quantité et le bouton +



Du côté Back-end, la route/rapportvisite/new

```
/**
 * @Route("/new", name="app_rapport_new", methods={"POST"})
 * créer un Rapport de visite et ajout en bdd
 * Ajout dans RapportVisite et dans RapportMedicament
 */
public function newRapport(SessionInterface $session, EntityManagerInterface $entityManager)
{
    // création d'un nouveau rapport de visite
    $rapport = new Rapportvisite();
    $entityManager = $this->getDoctrine()->getManager();
    //on cherche l'user selon l'id de la session
    $user = $this->getDoctrine()
        ->getRepository(Visiteur::class)
        ->findOneBy([
            'id' => $session->get('id'),
        ]);
    // on ajoute l'user correspondant à la création du rapport
    $rapport->setIdvisiteur($user);
}
```

On récupère l'id du visiteur connecté grâce à SessionInterface et on le set dans un nouveau rapport de Visite qu'on a créé.

Avec les valeurs POST récupérées on les sets dans le rapportVisite.

```

$dateVisite = $_POST["RAP_DATEVISITE"];
$date = new \DateTime($dateVisite);
$rapport->setDatevisite($date);
$bilan = $_POST["RAP_BILAN"];
$rapport->setBilan($bilan);

```

Exemple pour dateVisite (on l'on a transformé en dateTime) et bilan.

Un rapport de visite contient des clefs étrangères.

Pour l'id Motif par exemple, je recherche le motif correspondant à la valeur POST (idMotif sélectionné)

```

$motif = $_POST["RAP_MOTIF"];
// On ajoute les informations grâce aux setters et en récupérant les variables POST
// On cherche le motif sélectionné par rapport à la variable POST
$motifSearch = $this->getDoctrine()
->getRepository(Motif::class)
->findOneBy([
    'idmotif' => $motif,
]);
// on ajoute le motif qui correspond au rapport
$rapport->setIdmotif($motifSearch);

```

On fait la même pour le praticien et s'il y a un remplaçant.

```

// Si le rapport à un remplaçant on cherche le praticien qui correspond à la variable POST
if (isset($_POST["PRA_REEMPLACANT"])) {
    $remplacant = $_POST["PRA_REEMPLACANT"];
    $praticienRempla = $this->getDoctrine()
->getRepository(Praticien::class)
->findOneBy([
        'idpraticien' => $remplacant,
    ]);
    // on set qu'il est remplaçant et on set le remplaçant
    $rapport->setEstremplacant(true);
    $rapport->setIdremplacant($praticienRempla);
}
else{
    // sinon on dit qu'il n'y a pas de remplaçant
    $rapport->setEstremplacant(false);
}
// on ajoute en cherchant le praticien selon la variable POST obtenue
$idPraticien = $_POST["PRA_NUM"];
$praticien = $this->getDoctrine()
->getRepository(Praticien::class)
->findOneBy([
    'idpraticien' => $idPraticien,
]);

```

On utilise EntityManagerInterface pour persiste puis flush le rapport de visite.

Ensuite, à chaque rapport de visite, il faut ajouter les produits référencés dans la table rapportMedicament.

On regarde si l'utilisateur à entré un produit (c'est-à-dire qu'on détecte la variable \$_POST) et si la valeur n'est pas égale à NULL (ce qui signifie que l'utilisateur ait entré AUCUN).

On doit créer un nouveau RapportMedicament

Exemple pour Produit 1 :

```

// Si Produit1 est détecté et pas == null alors je créer un nouveau Rapportmedicament
if (isset($_POST["PROD1"])&& $_POST["PROD1"] !== "NULL") {
    $rapportMedicament1 = new Rapportmedicament();
    $entityManager = $this->getDoctrine()->getManager();
    $produit1 = $_POST["PROD1"];
    $rapportMedicament1->setEstechantillon(false);
    // Je mets la variable estEchantillon à faux et je recherche le médicament selon la variable Post
    $medicament = $this->getDoctrine()
        ->getRepository(Medicament::class)
        ->findOneBy([
            'idmedicament' => $produit1,
        ]);
    $rapportMedicament1->setIdmedicament($medicament);
    $rapportMedicament1->setEstpresente(true);
    // Je dis que c'est un médicament qui a été présenté et j'ajoute l'id du rapport qui a été créé juste au dessus
    $rapportMedicament1->setIdrapport($rapport);
    // on ajoute l'entité rapport créée à la BDD
    $entityManager->persist($rapportMedicament1);
    $entityManager->flush();
}

```

On utilise EntityManagerInterface pour persiste puis flush.

CONCLUSION

J'ai évité de mettre trop de captures d'écran du code. Vous pouvez trouver le code sur GitHub et il est commenté.

Le fichier README.md récapitule l'ensemble des commandes utilisées.

Merci de votre lecture.