

Projet G.S.B Web

Table des matières

I - Contexte.....	2
II - Analyse des besoins.....	2
A- Énoncer le besoin.....	2
B- Besoin de sécurité.....	2
C- Besoin réglementaire.....	2
D- Objectif et Sprint.....	3
1- Sprint 1.....	3
a - Besoin 1.....	3
b - Besoin 2.....	4
c – Besoin 3.....	7
d – Bilan du sprint.....	8
2- Sprint 2.....	9
a - Besoin 1.....	9
b - Besoin 2.....	10
c – Bilan du sprint.....	10
E – Bilan de la production.....	11
III - Compétences validées.....	11

I - Contexte

Le laboratoire Galaxy Swiss Bourdin (GSB) souhaite désormais entamer une transformation digitale efficace. Les visiteurs récupèrent une information directe sur le terrain. Ceci concerne aussi bien le niveau de la confiance qu'inspire le laboratoire que la lisibilité des notices d'utilisation des médicaments ou encore les éventuels problèmes rencontrés lors de leur utilisation, etc. Ces informations ne sont actuellement pas systématiquement remontées au siège, ou elles le sont dans des délais jugés trop longs. Le service rédaction qui produit les notices souhaite avoir des remontées plus régulières et directes. Ceci permettra également au service labo-recherche d'identifier et d'engager des évaluations complémentaires. Au moment de l'embauche d'un visiteur médicale, les ressources humaines doivent être capable de créer son compte utilisateur en renseignant ses informations (login, mot de passe, nom, prénom, adresse, ...).

II - Analyse des besoins

A- Énoncer le besoin

Afin de d'optimiser l'exploitation des données terrains produites par les visiteurs médicaux qu'ils remontent de leurs visites auprès des différents prescripteurs (médecins, pharmaciens, infirmières et autres métiers de santé), nous proposons de gérer les rapports des visites via une application web pour que les service rédaction et labo-recherche puissent analyser les données. De plus cela permettra aux responsables secteur, de pouvoir générer des indicateurs de performance de suivi de visite.

B- Besoin de sécurité

L'environnement doit être accessible aux seuls acteurs de l'entreprise. Une authentification préalable sera nécessaire pour l'accès au contenu. Une politique d'accès aux ressources suivant le rôle utilisateur sera définie. Les visiteurs médicaux pourront créer, modifier et visualiser leurs rapports. Les données transitant sur les réseaux étant sensible, l'application devra assurer la sécurité transmission des informations.

C- Besoin réglementaire

Les laboratoires achètent des fichiers à des organismes spécialisés qui donnent, outre les diverses informations d'état civil et d'origine, les données sur l'influence du praticien sur son entourage professionnel (est-il prescripteur, membre d'une association, relais de l'ordre des médecins...). Dans leurs spécialités, les praticiens possèdent des diplômes (on ne conserve que le plus haut niveau) et ont un coefficient de prescription (sont-ils reconnus par leurs collègues comme référents sur la spécialité, sont-ils dans un cabinet pointu sur le sujet, etc). Les données récupérées sur les praticiens sont confidentielles et doivent respecter le règlement général sur la protection des données (RGPD).

D- Objectif et Sprint

1- Sprint 1

a - Besoin 1

Le premier besoin consiste à gérer les praticiens, c'est à dire créer un profil de praticien, pouvoir lire ce profil, modifier les informations du profil et supprimer le profil. Un profil de praticien contient les informations suivantes : un nom, prénom, une adresse mail, une adresse postale, son métier, son établissement et/ou si il est libéral. Pour des raisons statistiques on enregistrera la personne qui a créé le profil, la date de la dernière visite du praticien ainsi que le nombre de visite qu'il a reçu. Dans notre base de donnée cela se présente comme tel :

pharma_symfony praticien	pharma_symfony praticien_type
id : int(11)	id : int(11)
# type_id : int(11)	name : varchar(255)
firstname : varchar(255)	pharma_symfony departement
lastname : varchar(255)	id : int(11)
email : varchar(255)	name : varchar(255)
etablissement : varchar(255)	# region_id : int(11)
adresse : varchar(255)	pharma_symfony region
last_visit : datetime	id : int(11)
# nb_visit : int(11)	name : varchar(255)
ville : varchar(255)	pharma_symfony user
# is_liberal : tinyint(1)	id : int(11)
# creator_id : int(11)	email : varchar(180)
# region_id : int(11)	roles : json
# dpt_id : int(11)	password : varchar(255)
	name : varchar(255)
	firstname : varchar(255)
	phone : varchar(255)

Nous avons créé une entity praticien puis utilisé la commande : `php bin/console make:crud` et sélectionné l'entité praticien. La commande crée automatiquement un contrôleur, un repository ainsi que les vues pour faire les opérations Create Update Delete Edit. Nous les avons ensuite retouché pour correspondre à nos besoins.

Les régions, départements, métiers du praticien sont déjà entré dans la base de données et se sélectionne sous forme de liste dans le formulaire.

b - Besoin 2

Le second besoin consiste à ajouter un module de recherche pour trier la liste de praticien afin que l'on puisse rechercher les praticien selon les critères suivant : la ville, le département, la région, si il a été visité récemment ou non, la date de la dernière visite ainsi que le métier du praticien.

Nus avons donc utilisé une carte interactive fourni par le bundle leaflet, ainsi qu'une liste affichant les noms, prénoms, l'établissement ou son statut libéral ainsi que son métier.

Pour cela nous avons utilisé des requête api en javascript sur la page html pour la carte et des requête SQL en PHP pour la recherche par critères.

```
186 <script>
187     datePickerId.max = new Date().toISOString().split("T")[0];
188     window.addEventListener("load", function (event) {
189         var mymap = L.map('map').setView([46.925539, 2.409112], 6);
190         L.tileLayer('https://api.mapbox.com/styles/v1/{id}/tiles/{z}/{x}/{y}?access_token={accessToken}', {
191             attribution: 'Map data &copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors, Imagery © <a href="https://www.mapbox.com/attribution">Mapbox</a>',
192             maxZoom: 18,
193             id: 'mapbox/streets-v11',
194             tileSize: 512,
195             zoomOffset: -1,
196             accessToken: 'pk.eyJ1IjoieG9yY2VzIiwiaW50IjoiImNrdzZlMG9zYjJsZDAyb25vMmNidmdnb2gifQ.20frcp606JrWkd42dLzVeA'
197         }).addTo(mymap);
198         var addresses = document.getElementsByClassName('data-of-praticien');
199
200
201         for (var i = 0; i < addresses.length; i++) {
202             var formattedAddress = addresses[i].dataset.adresse.split(' ').join('+');
203             console.log(formattedAddress);
204             window.fetch('https://api-adresse.data.gouv.fr/search/?q=' + formattedAddress).then(function (response) {
205                 return response.json();
206             }).then(function (data) {
207                 var adr = document.getElementsByClassName('data-of-praticien');
208                 for (var j = 0; j < adr.length; j++) {
209                     if (adr[j].dataset.adresse === data.query) {
210                         var formattedName = adr[j].dataset.formatedname;
211                         var isLiberal = adr[j].dataset.liberal;
212                     }
213
214                 }
215                 var coord = data.features[0].geometry.coordinates;
216                 var pinLon = coord[0];
217                 var pinLat = coord[1];
218                 var pinAddress = data.features[0].properties.label;
219                 console.log(isLiberal);
220                 if (isLiberal == 1) {
221                     var greenIcon = new L.Icon({
222                         iconUrl: 'https://raw.githubusercontent.com/pointhi/leaflet-color-markers/master/img/marker-icon-2x-green.png',
223                         shadowUrl: 'https://cdnjs.cloudflare.com/ajax/libs/leaflet/0.7.7/images/marker-shadow.png',
224                         iconSize: [25, 41],
225                         iconAnchor: [12, 41],
226                         popupAnchor: [1, -34],
227                         shadowSize: [41, 41]
228                     });
229                     var marker = L.marker([pinLat, pinLon], {icon: greenIcon}).addTo(mymap);
230                 } else {
231                     var marker = L.marker([pinLat, pinLon]).addTo(mymap);
232                 }
233                 marker.bindPopup("<b>" + formattedName + "</b><br>" + pinAddress);
234             }
235         }
236     });
237 </script>
```

```

29  /**
30  * @Route("/", name="praticien_index", methods={"GET"})
31  */
32  public function index(PraticienRepository $praticienRepository, Request $request)
33  {
34      $praticiens = $praticienRepository->findAll();
35      $typeListe = $this->getDoctrine()->getRepository(App\Entity\PraticienType::class)->findAll();
36      $dptListe = $this->getDoctrine()->getRepository(Departement::class)->findAll();
37      $regionListe = $this->getDoctrine()->getRepository(Region::class)->findAll();
38
39      $ville = $request->get('ville', null);
40      $dpt = $request->get('dpt', null);
41      $region = $request->get('region', null);
42      $type = $request->get('type', null);
43      $lastVisit = $request->get('lastVisit', null);
44      $visited = $request->get('visited', false);
45      $notVisited = $request->get('notVisited', false);
46
47      if ($visited !== false) {
48          $visited = true;
49      }
50      if ($notVisited !== false) {
51          $notVisited = true;
52      }
53
54      $vIds = [];
55      $dIds = [];
56      $rIds = [];
57      $tIds = [];
58      $lvIds = [];
59      $nvIds = [];
60      $result = $praticiens;
61
62      /** @var QueryBuilder $qb */
63      $qb = $this->entityManager->createQueryBuilder();
64
65      if (!empty($ville)) {
66          $argByVille = $qb->select('v')
67              ->from(Praticien::class, 'v')
68              ->orWhere('v.ville LIKE :search')
69              ->setParameter('search', '%' . $ville . '%')
70              ->getQuery()->getResult();
71
72      } else {
73          $argByVille = $praticiens;
74      }
75
76      if ($argByVille !== null) {
77          foreach ($argByVille as $av) {
78              $vIds[] = $av->getVille();
79          }
80      }
81
82      if (!empty($dpt)) {
83          $dIds[] = $dpt;
84      }
85      if ($dIds) {
86          $argByDpt = $qb->select('dpt')
87              ->from(Departement::class, 'dpt')
88              ->getQuery()->getResult();
89          if ($argByDpt !== null) {
90              foreach ($argByDpt as $ad) {
91                  $dIds[] = $ad->getId();
92              }
93          }
94      }
95
96      if (!empty($region)) {
97          $rIds[] = $region;
98      }
99      if ($rIds) {
100          $argByRegion = $qb->select('r')
101              ->from(Region::class, 'r')
102              ->getQuery()->getResult();
103          if ($argByRegion !== null) {
104              foreach ($argByRegion as $ar) {
105                  $rIds[] = $ar->getId();
106              }
107          }
108      }
109
110      if (!empty($type)) {
111          $tIds[] = $type;
112      }
113      if ($tIds) {
114          $argByType = $qb->select('t')
115              ->from(App\Entity\PraticienType::class, 't')
116              ->getQuery()->getResult();
117          if ($argByType !== null) {
118              foreach ($argByType as $at) {
119                  $tIds[] = $at->getId();
120              }
121          }
122      }
123
124      if (!empty($lastVisit)) {
125          $argByLastVisit = $qb->select('lv')
126              ->from(Praticien::class, 'lv')
127              ->orWhere('lv.lastVisit LIKE :search')
128              ->setParameter('search', '%' . $lastVisit . '%')
129              ->getQuery()->getResult();
130
131      } else {
132          $argByLastVisit = $praticiens;
133      }
134      $result = $argByLastVisit;
135  }

```

```

136     if ($argByLastVisit != null) {
137         foreach ($argByLastVisit as $lv) {
138             $lvIds[] = $lv->getId();
139         }
140     }
141
142     if ($visited === true) {
143         $argByNbVisit = $qb->select("nv")
144             ->from(Praticien::class, 'nv')
145             ->orWhere($qb->expr()->isNotNull("nv.nbVisit"))
146             ->getQuery()->getResult();
147     } elseif ($notVisited === true) {
148         $argByNbVisit = $qb->select("nv")
149             ->from(Praticien::class, 'nv')
150             ->orWhere($qb->expr()->isNull("nv.nbVisit"))
151             ->getQuery()->getResult();
152     } else {
153         $argByNbVisit = $praticiens;
154     }
155
156     if ($argByNbVisit != null) {
157         foreach ($argByNbVisit as $nv) {
158             $nvIds[] = $nv->getNbVisit();
159         }
160     }
161
162     if (sizeof($vIds) == 0 or sizeof($dIds) == 0 or sizeof($rIds) == 0 or sizeof($tIds) == 0 or sizeof($lvIds) == 0 or sizeof($nvIds) == 0 or siz
163     $result = [];
164 } elseif ($visited === true) {
165     $result = $qb->select('p')
166         ->from(Praticien::class, 'p')
167         ->andWhere($qb->expr()->in('p.ville', $vIds))
168         ->andWhere($qb->expr()->in('p.dpt', $dIds))
169         ->andWhere($qb->expr()->in('p.region', $rIds))
170         ->andWhere($qb->expr()->in('p.type', $tIds))
171         ->andWhere($qb->expr()->in('p.id', $lvIds))
172         ->andWhere($qb->expr()->isNotNull("p.nbVisit"))
173         ->getQuery()->getResult();
174 } elseif ($notVisited === true) {
175     $result = $qb->select('p')
176         ->from(Praticien::class, 'p')
177         ->andWhere($qb->expr()->in('p.ville', $vIds))
178         ->andWhere($qb->expr()->in('p.dpt', $dIds))
179         ->andWhere($qb->expr()->in('p.region', $rIds))
180         ->andWhere($qb->expr()->in('p.type', $tIds))
181         ->andWhere($qb->expr()->in('p.id', $lvIds))
182         ->andWhere($qb->expr()->isNull("p.nbVisit"))
183         ->getQuery()->getResult();
184 } else {
185     $result = $qb->select('p')
186         ->from(Praticien::class, 'p')
187         ->andWhere($qb->expr()->in('p.ville', $vIds))
188         ->andWhere($qb->expr()->in('p.dpt', $dIds))
189         ->andWhere($qb->expr()->in('p.region', $rIds))
190         ->andWhere($qb->expr()->in('p.type', $tIds))
191         ->andWhere($qb->expr()->in('p.id', $lvIds))
192         ->getQuery()->getResult();
193 }
194
195 $praticiens = $result;
196
197 return $this->render('praticien/index.html.twig', [
198     'praticiens' => $praticiens,
199     'lastVisit' => $lastVisit,
200     'ville' => $ville,
201     'dpt' => $dpt,
202     'region' => $region,
203     'type' => $type,
204     'visited' => $visited,
205     'notVisited' => $notVisited,
206     'typeListe' => $typeListe,
207     'dptListe' => $dptListe,
208     'regionListe' => $regionListe,
209 ]);
210 }

```

Nous avons ensuite modifier le design de la liste des praticiens pour le faire correspondre à la charte visuelle du site.

Recherche avancée de praticiens

Date de la dernière visite
jj/mm/aaaa

Ville
Département
Région

Spécialité
Tous

☐ Visité
☐ Non Visité

Rechercher



Nom - Prénom	Établissement	Profession	Actions
Barrilo Enzo	Pharmacie	Psychologue	✎ ✖
Humerte Flora	Pharmacie	Psychologue	✎ ✖
Mari Julien	Aucun - Libéral	Infirmier(e)	✎ ✖

c – Besoin 3

Le troisième besoin consiste à gérer les comptes utilisateur des employés, c'est à dire créer un profil, pouvoir lire ce profil, modifier les informations du profil et supprimer le profil. Un profil d'employé contient les informations suivantes : un nom, prénom, une adresse mail, un mot de passe et un numéro de téléphone. Pour des raisons de sécurité nous attribuerons un rôle correspondant au département dans lequel est employé la personne afin qu'il n'est accès uniquement aux fonctionnalités qui lui sera utile pour son travail. Voici l'entité dans la base de donnée :

pharma_symfony user
id : int(11)
email : varchar(180)
roles : json
password : varchar(255)
name : varchar(255)
firstname : varchar(255)
phone : varchar(255)

Nous avons créé une entité user grâce au security bundle avec la commande : `php bin/console make:user` qui génère une entité basique d'utilisateur et la définit comme entité qui servira lors de l'authentification. La commande crée automatiquement un contrôleur un repository ainsi que les vues pour faire les opérations Create Update Delete Edit. Nous les avons ensuite retouchés pour correspondre à nos besoins.

Les rôles sont déclarés dans le fichier security.yaml et empêchent les utilisateurs d'accéder au contenu qui ne leur est pas utile.

```

1 security:
2   role_hierarchy:
3     ROLE_VISITEUR: [ROLE_USER]
4     ROLE_RH: [ROLE_USER]
5     ROLE_RD: [ROLE_USER]
6     ROLE_REDACTOR: [ROLE_USER]
7     ROLE_ADMIN: [ROLE_USER, ROLE_RH, ROLE_VISITEUR, ROLE_RD]
8     ROLE_SUPER_ADMIN: [ROLE_USER, ROLE_RH, ROLE_ADMIN, ROLE_VISITEUR, ROLE_RD]
9
10  encoders:
11    App\Entity\User:
12      algorithm: auto
13
14  # https://symfony.com/doc/current/security.html#where-do-users-come-from-user-providers
15  providers:
16    # used to reload user from session & other features (e.g. switch_user)
17    app_user_provider:
18      entity:
19        class: App\Entity\User
20        property: email
21  firewalls:
22    dev:
23      pattern: ^/(_(profiler|wdt)|css|images|js)/
24      security: false
25    main:
26      anonymous: true
27      lazy: true
28      provider: app_user_provider
29      guard:
30        authenticators:
31          - App\Security\AppCustomAuthenticator
32      logout:
33        path: app_logout
34        # where to redirect after logout
35        # target: app_any_route
36
37    # activate different ways to authenticate
38    # https://symfony.com/doc/current/security.html#firewalls-authentication
39
40    # https://symfony.com/doc/current/security/impersonating_user.html
41    # switch_user: true
42
43  # Easy way to control access for large sections of your site
44  # Note: Only the *first* access control that matches will be used
45  access_control:
46    - { path: ^/admin, roles: ROLE_ADMIN }
47    - { path: ^/rh, roles: ROLE_RH }
48    - { path: ^/visiteur, roles: ROLE_VISITEUR }
49    - { path: ^/rapport/visiteur, roles: ROLE_VISITEUR }
50    - { path: ^/rapport, roles: [ROLE_VISITEUR, ROLE_RD, ROLE_REDACTOR] }
51    - { path: ^/secured, roles: ROLE_USER }
52    - { path: ^/authapi, roles: IS_AUTHENTICATED_FULLY }

```

Création d'un nouvel utilisateur

Prénom

Poste

Utilisateur ▼

Nom

Numéro de téléphone

Email

Mot de passe (8 caractères)

Enregistrer

d – Bilan du sprint

Nous avons atteint les objectifs attendus, dans le temps qu'il nous a été accordé. Il a fallu plus de temps que prévu avant de commencer la phase de développement. La phase de débogage c'est quand elle s'est déroulée sans accros.

2- Sprint 2

a - Besoin 1

Le premier besoin consiste à gérer les rapports des visiteurs, c'est à dire créer un rapport, pouvoir lire ce rapport, modifier les informations du rapport et supprimer le rapport. Un rapport contient les informations suivantes : un titre, un créateur, un concerné, un corps ainsi qu'une date d'écriture. Voici l'entité dans la base de donnée :

pharma_symfony rapport	
id	int(11)
rapport_user_id	int(11)
titre	varchar(255)
date	datetime
text	longtext
praticien_id	int(11)

Nous avons créé une entity praticien puis utilisé la commande : `php bin/console make:crud` et sélectionné l'entité rapport. La commande créé automatiquement un controller un repository ainsi que les vues pour faire les opérations Create Update Delete Edit. Nous les avons ensuite retouché pour correspondre à nos besoins.

Création d'un rapport

Titre

Date de la visite

jj/mm/aaaa --:--

Corps du rapport

Praticien concerné

Praticien

Enzo Barrio

Confirmer

Annuler

Liste des rapports

Nouveau rapport

Date	Visiteur	Titre	Praticien	Profession	actions
08/05/2022	Jaures Roger	rapport 1	Mari Julien	Infirmier(e)	<div><div></div><div></div><div>Supprimer le Rapport</div></div>
10/04/2022	Jaures Roger	Rapport 1	vermont michel	Pneumologue	<div><div></div><div></div><div>Supprimer le Rapport</div></div>

b - Besoin 2

Le second besoin consiste à pouvoir lire de façon intuitive un rapport nous avons donc refait le design de la page de présentation d'un rapport :



c – Bilan du sprint

Nous avons atteint l'objectif attendu avec quelques jours d'avances. La phase de développement et de débogage se sont déroulé sans accros.

E – Bilan de la production

Les versions on était fournis à temps et tous les objectifs ont été rempli. Néanmoins il reste encore du débogage de problèmes non bloquant. Durant cette réalisation nous avons pu mettre en œuvre nos compétences en PHP, HTML, SCSS, JS, SQL. Nous avons aussi pu nous familiariser avec le framework Symfony. Cette réalisation a pu consolider la cohésion de cette équipe de développement naissante.

III - Compétences validées

J'ai pu validé au travers de cette réalisation 4 compétences.

- Gérer le patrimoine informatique
- Répondre aux incidents et aux demandes d'assistance et d'évolution
- Travailler en mode projet
- Mettre à disposition des utilisateurs un service informatique

Pour plus d'informations veuillez consulter ce [document](#)