GSB

Table des matières

I- Gérer le patrimoine Informatique

A- Mettre en place et vérifier les niveaux d'habilitation associés à un service

Nous avons utilisé le security bundle du framework symfony grâce à la commande :

`composer require symfony/security-bundle`

Celui-ci nous permet de générer une entité user basique composé d'un id, un email, de roles et d'un password, ainsi que les getter et setter qui y sont associés.

```php
// src/Entity/User.php
namespace App\Entity;

use App\Repository\UserRepository;
use Doctrine\ORM\Mapping as ORM;
use Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterface;
use Symfony\Component\Security\Core\User\UserInterface;

/**
 * @ORM\Entity(repositoryClass=UserRepository::class)
 */
class User implements UserInterface, PasswordAuthenticatedUserInterface
{
    /**
     * @ORM\Id
     * @ORM\GeneratedValue
     * @ORM\Column(type="integer")
     */
    private $id;

    /**
     * @ORM\Column(type="string", length=180, unique=true)
     */
    private $email;

    /**
     * @ORM\Column(type="json")
     */
    private $roles = [];

    /**
     * @var string The hashed password
     * @ORM\Column(type="string")
     */
    private $password;
```

Les mots de passes sont encodé par le Password encoder fournis avec le framework symfony.

On a ensuite pu définir dans le fichier security.yml la property qui va définir permettre d'identifier notre utilisateur, nous avons donc choisi l'adresse e-mail.

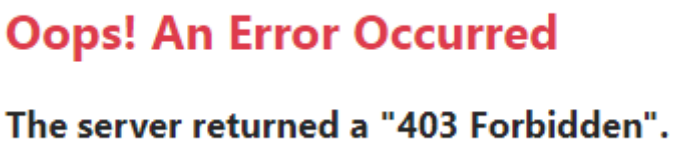
```yaml
providers:
  # used to reload user from session & other features (e.g. switch_user)
  app_user_provider:
    entity:
      class: App\Entity\User
      property: email
```

Toujours dans ce fichier nous pouvons définir les différents rôle attribuable à un utilisateur, ainsi que les accès que cela offre sur la plateforme.

```yaml
role_hierarchy:
  ROLE_VISITEUR: [ROLE_USER]
  ROLE_RH: [ROLE_USER]
  ROLE_RD: [ROLE_USER]
  ROLE_ADMIN: [ROLE_USER, ROLE_RH, ROLE_VISITEUR, ROLE_RD]
  ROLE_SUPER_ADMIN: [ROLE_USER, ROLE_RH, ROLE_ADMIN, ROLE_VISITEUR, ROLE_RD]
```

```yaml
access_control:
  - { path: ^/admin, roles: ROLE_ADMIN }
  - { path: ^/rh, roles: ROLE_RH }
  - { path: ^/visiteur, roles: ROLE_VISITEUR }
  - { path: ^/rapport/visiteur, roles: ROLE_VISITEUR }
  - { path: ^/rapport, roles: [ROLE_VISITEUR, ROLE_RD]  }
  - { path: ^/secured, roles: ROLE_USER }
  - { path: ^/authapi, roles: IS_AUTHENTICATED_FULLY }
```

Par exemple si une personne avec le rôle « ROLE_RH » essaye d'accéder à un page réservé au « ROLE_VISITEUR» une erreur lui sera renvoyé.

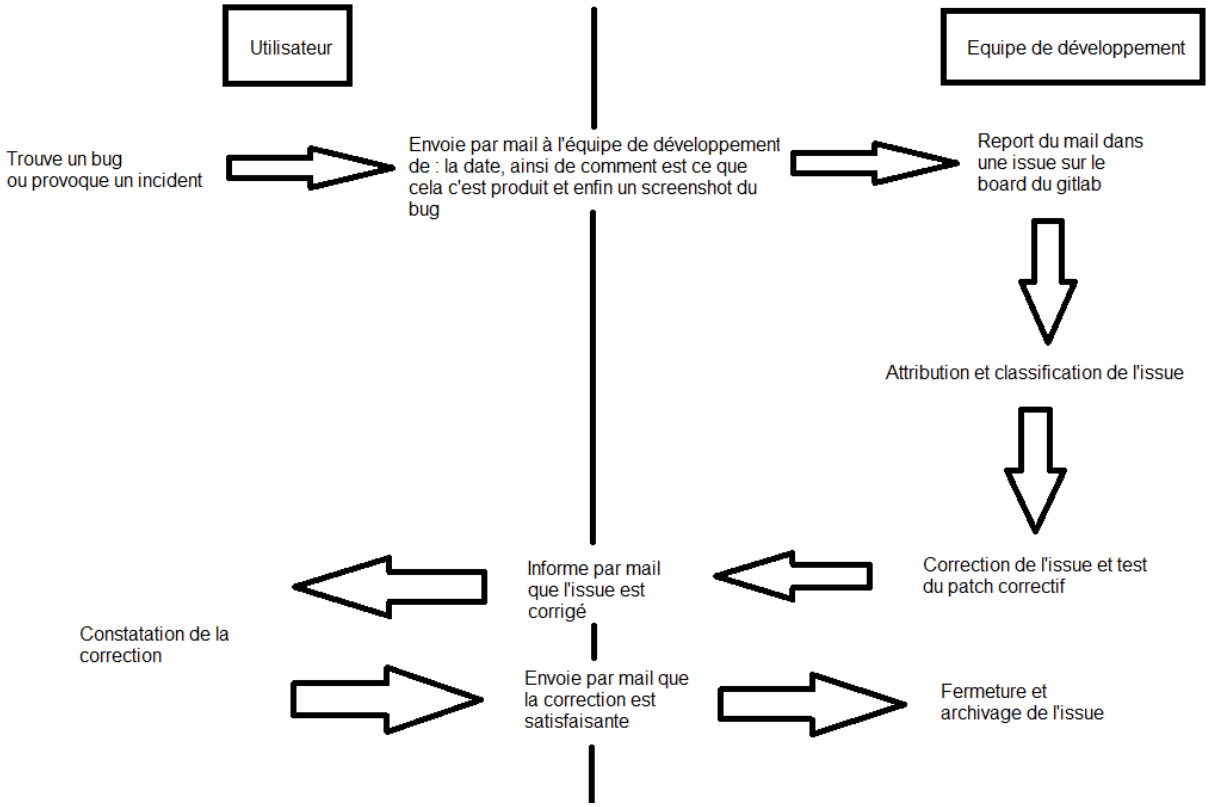


Cela nous permet de restreindre l'accès aux différentes fonctionnalité selon le niveau d'habilitation de l'utilisateur.

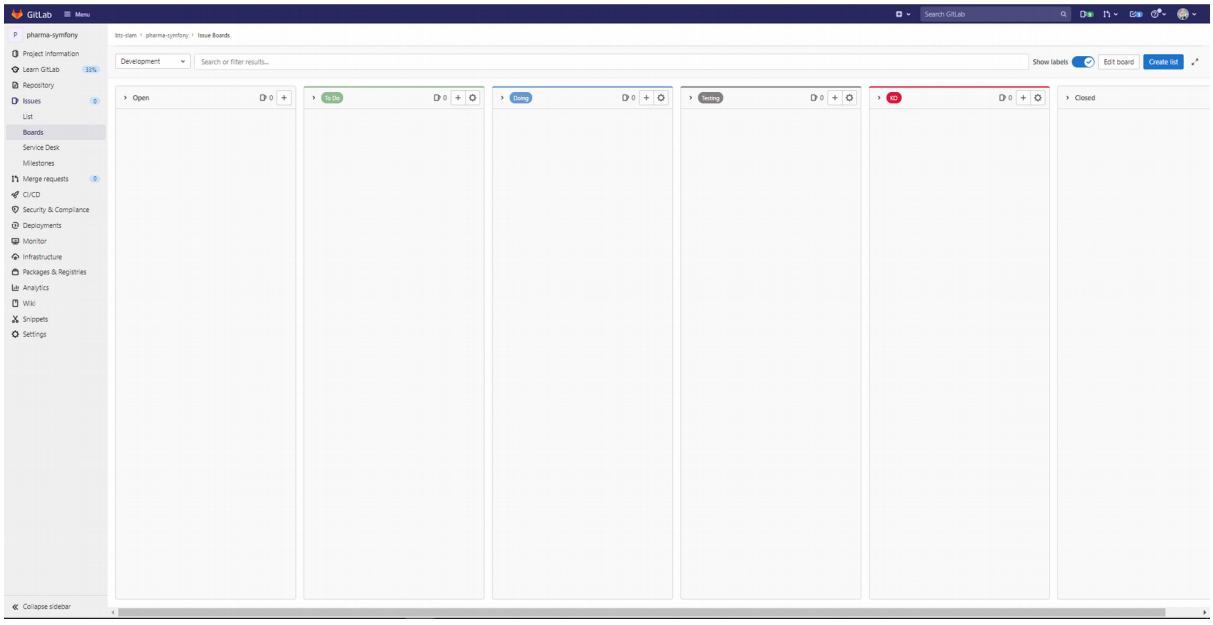
II- Répondre aux incidents et aux demandes d’assistance et d’évolution

A- Prise en charge d’incidents

Pour la gestion des remontés de bugs ou d’incidents nous avons utilisé la méthode suivante :



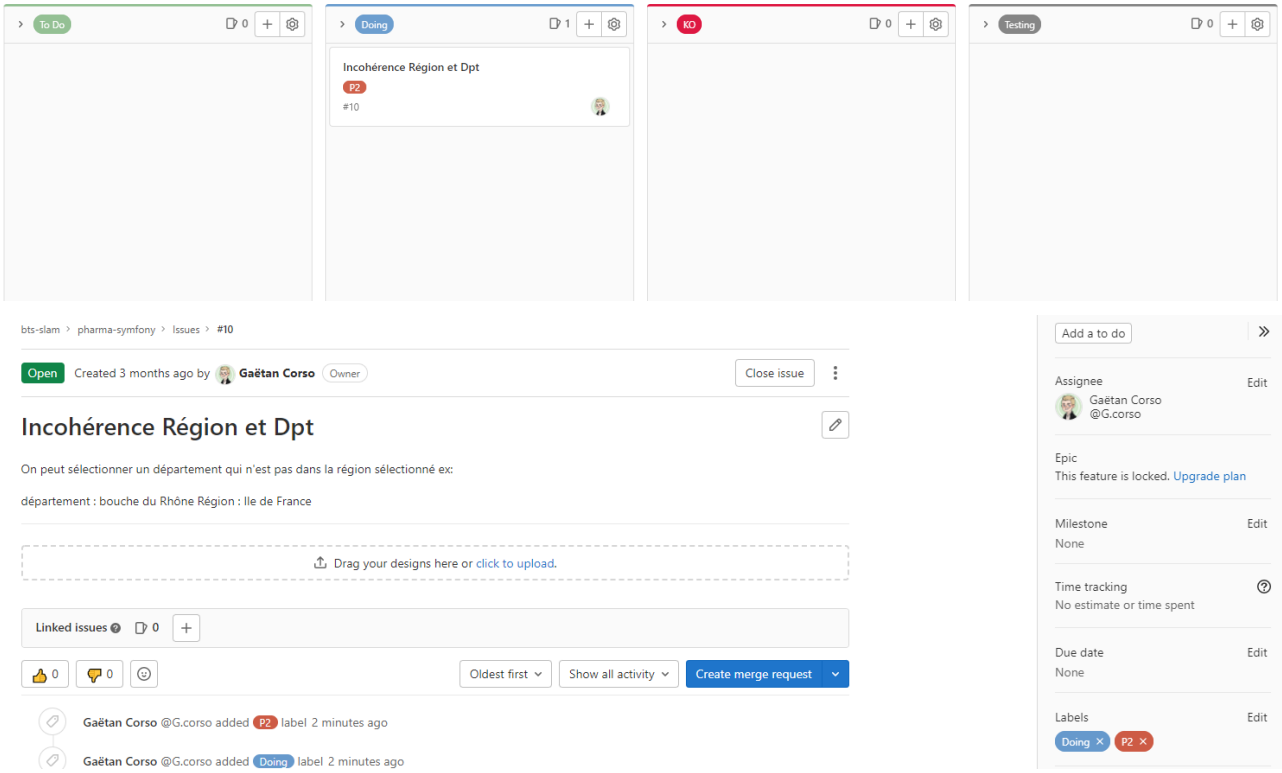
Voici comment se présente le board du gitlab.



Chaque colonne représente le stade d'avancement de l'issue. Des Labels permet de les classer par catégorie de priorité.

P1 : qui indique que l'issue est de TRÈS HAUTE priorité, elle devra donc être faite au plus tôt.
P2 : qui indique que l'issue est de Haute priorité, elle devra donc être faite après les P1.
P3 : qui indique que l'issue est de basse priorité, elle devra donc être faite après les P2.

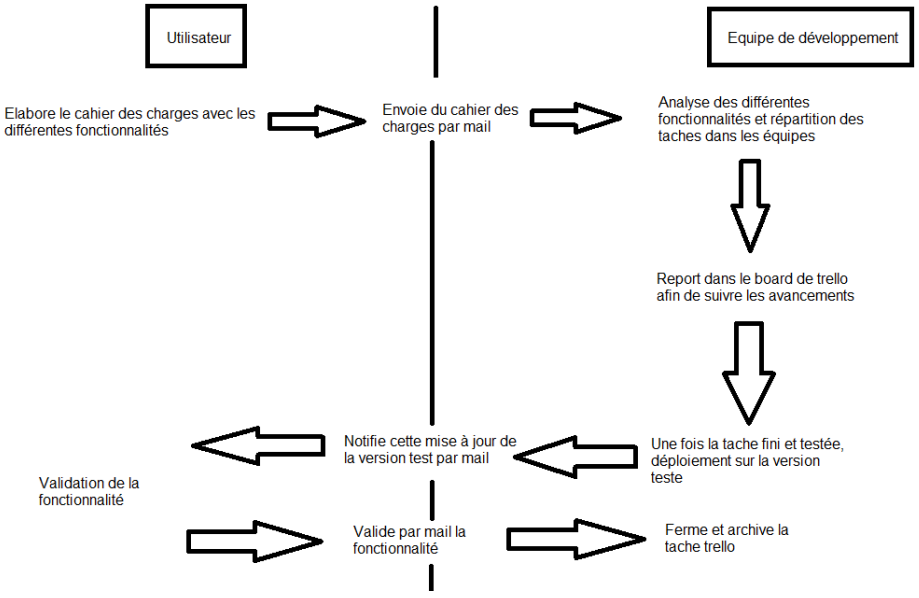
Voici un exemple d'issue pour le projet GSB.



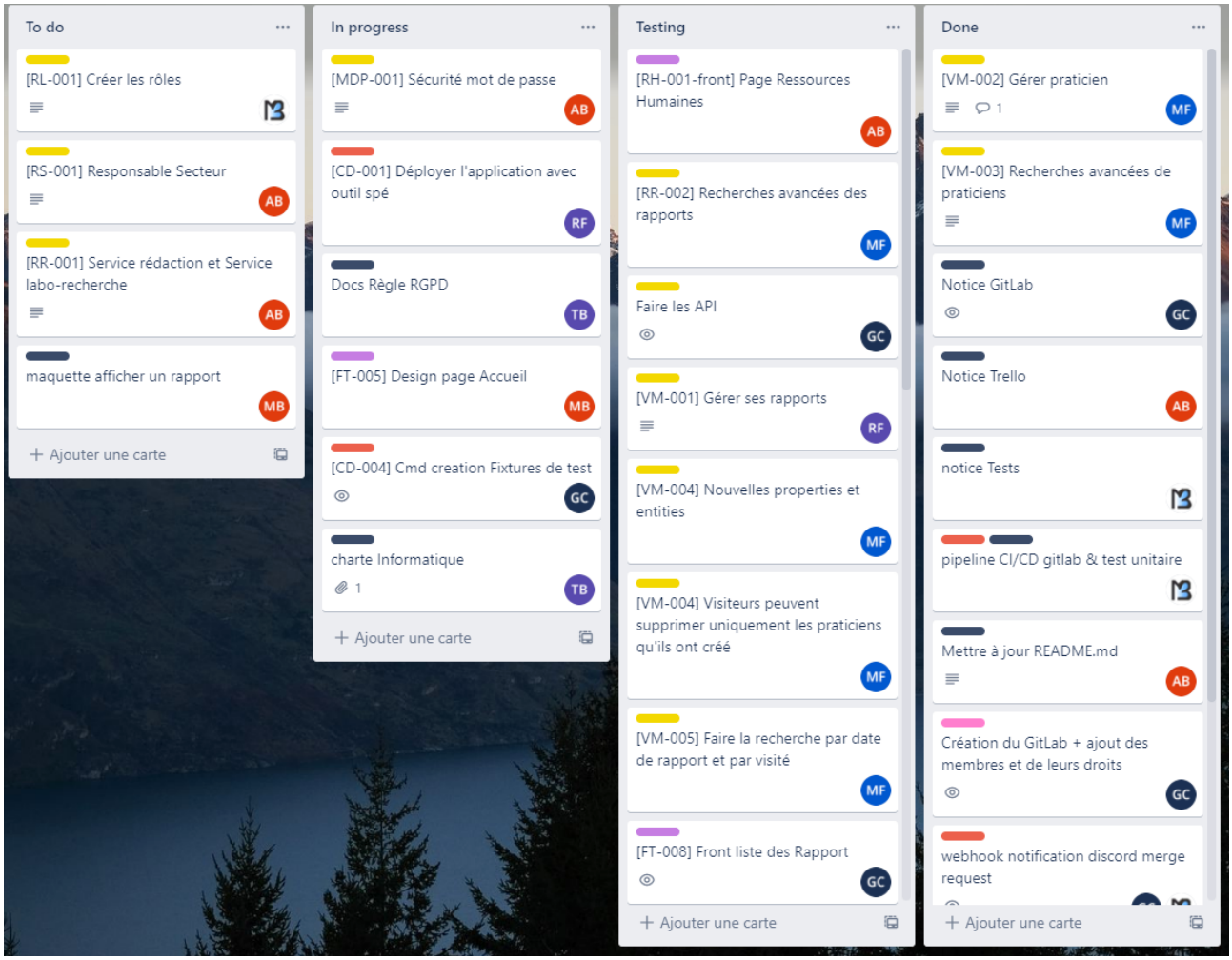
Nous pouvons voir dans une vue plus détaillé, ça description, la personne chargé de sa résolution ainsi que son historique d'avancement.

B- Prise en charge des demandes d'évolution de l'application

Pour prendre en charge les demandes d'évolution nous avons utilisé la méthode suivante :



Voici comment se présente le dashboard du trello :



Chaque colonne représente le stade d’avancement de la tache. Les codes entre crochet représente les acronymes des fonctionnalités concernés et les couleurs la partie du projet visée.

Jaune : Touche principalement au backEnd

Rouge : Touche principalement au Commande et déploiement

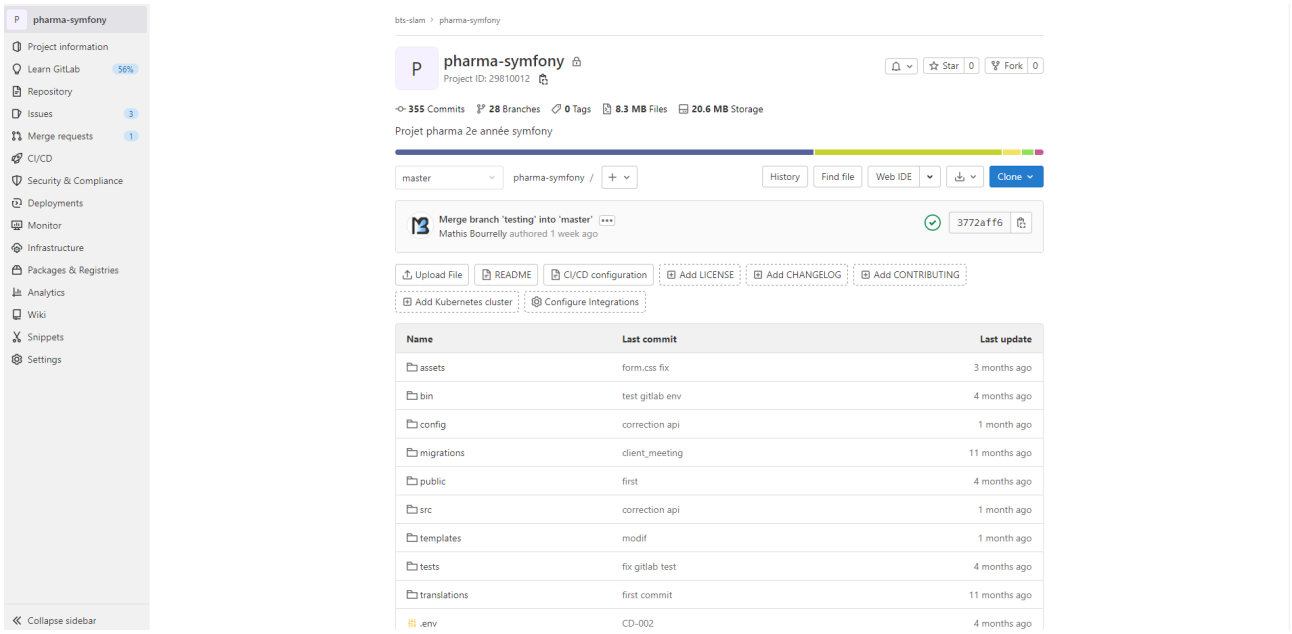
Rose : Touche principalement au frontEnd

Gris : Touche à de la documentation technique

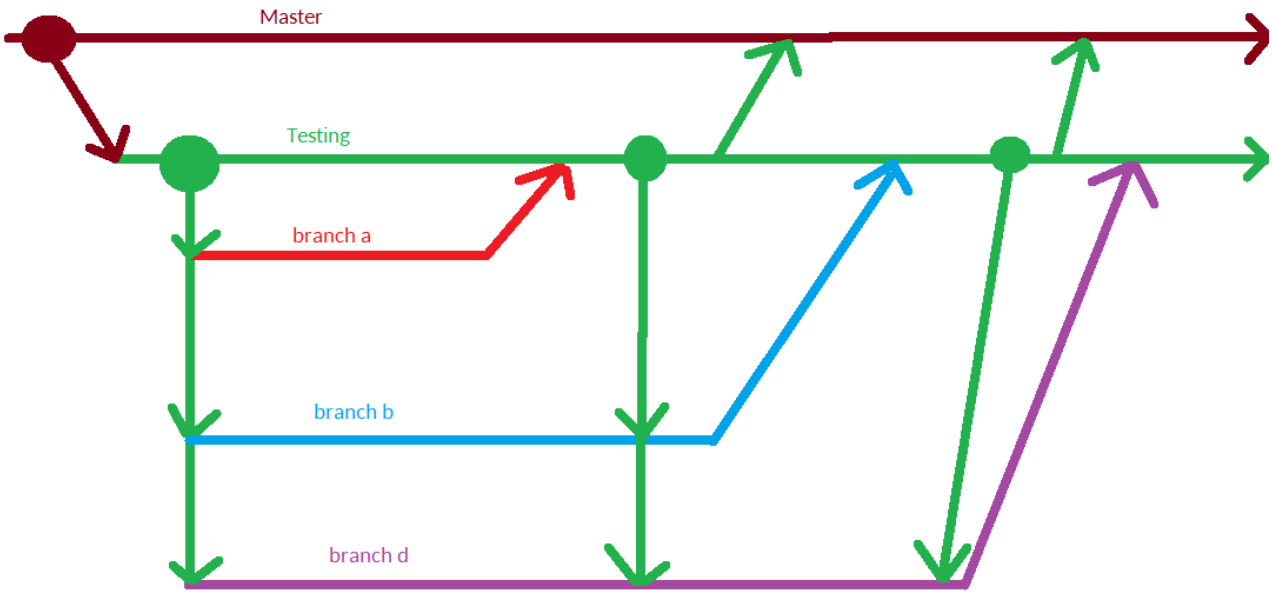
III- Travailler en mode Projet

A- Mise en place d’un répository

Nous avons mis en place un repository et un workflow pour faciliter notre travail en équipe. Nous avons utilisé gitLab.



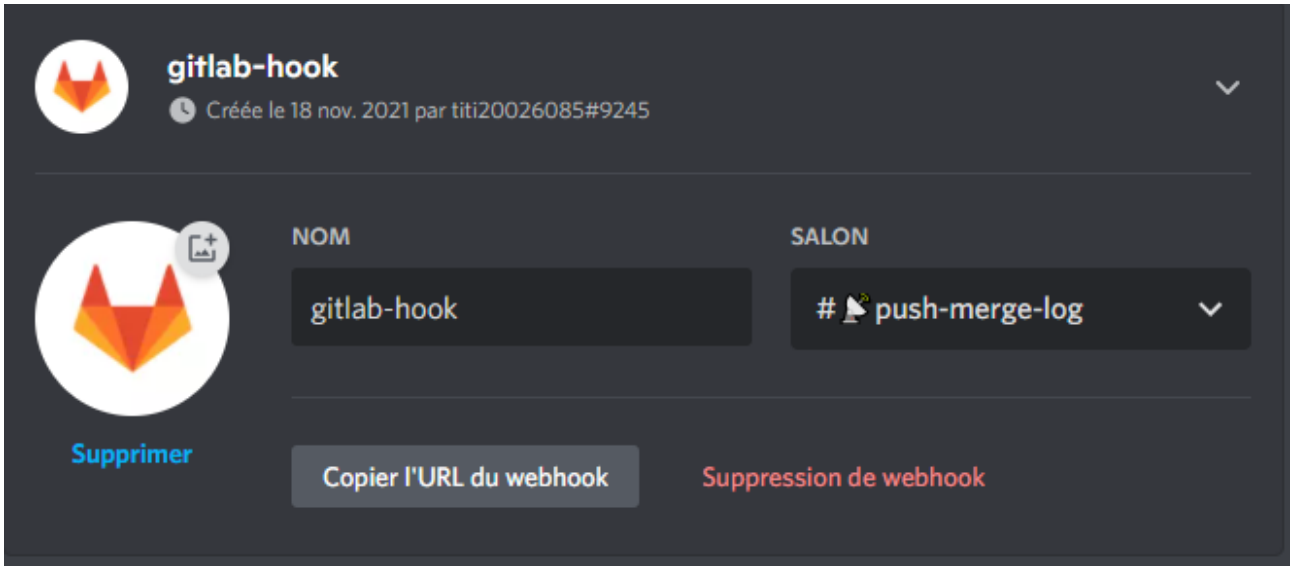
Le répository est configuré en privé et seul les personne travaillant sur le projet y ont accès.. Notre projet suivait le workflow suivant :



Les branches étaient nommées suivant cette règle « issue_*acronymeDeLaFonctionnalité* » ou « issue_*numéroDeL’issue* » notre branche de test s’appelle *testing*.

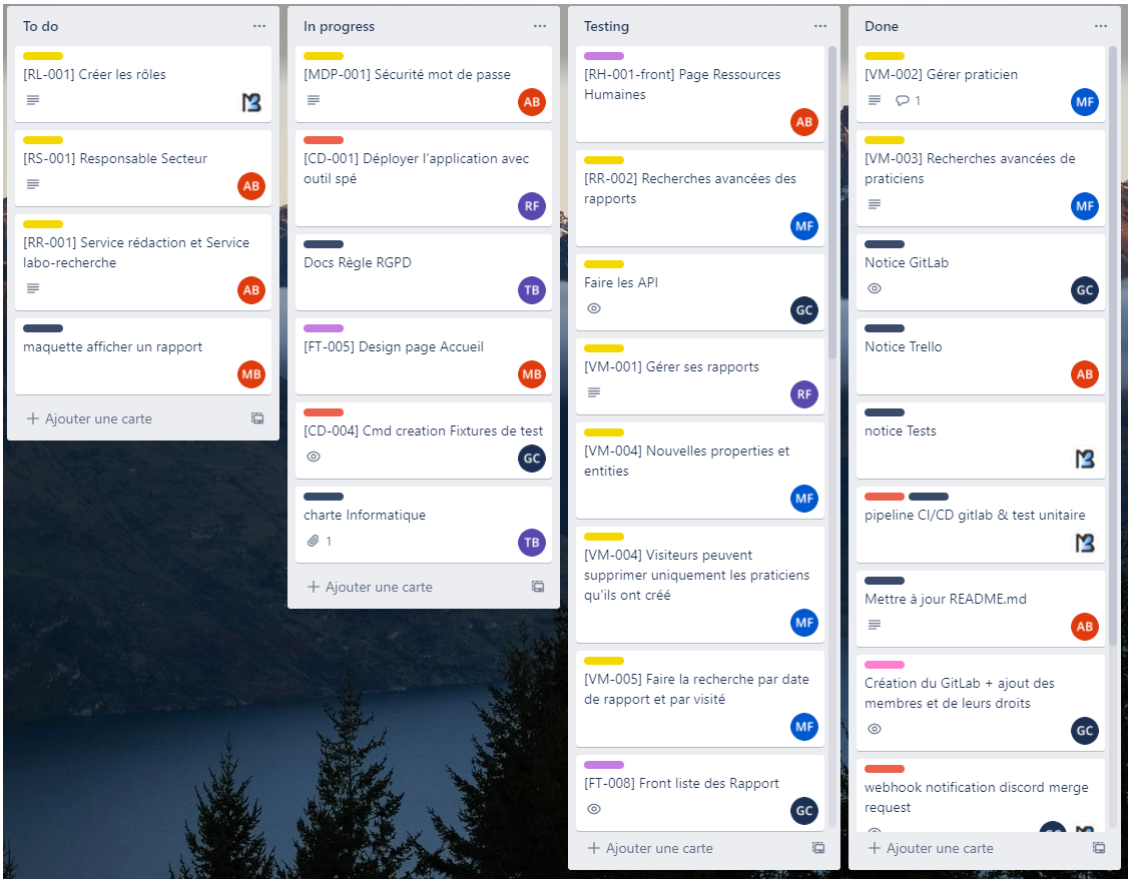
Un webHook a été mise en place pour signaler sur la plateforme de communication de l'équipe de developpement qu'une merge request venait d'être approuvé.

https://discord.com/api/webhooks/910806744576446504/dgieY2k6kWjsIo9aqj4sRpVurbpzycf5ZVyX0a10JHWx0C5zKeYPBfgwJQii-UWlkP7V



B- Répartition des taches avec Trello

Grâce à des réunions hebdomadaires nous pouvions attribuer les différentes taches sur le trello du projet.



Chaque colonne représente le stade d'avancement de la tache. Les codes entre crochet représente les acronymes des fonctionnalités concernés et les couleurs la partie du projet visée.

Jaune : Touche principalement au backEnd

Rouge : Touche principalement au Commande et déploiement

Rose : Touche principalement au frontEnd

Gris : Touche à de la documentation technique

IV- Mettre à disposition des utilisateurs un service informatique

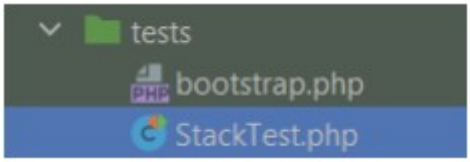
A- Tests unitaire, d'acceptation et Fonctionnel

1- Tests unitaires

Dans notre projet GSB, les test sont réalisé grâce au package phpUnit installable via composer .

```
symfony composer req phpunit -dev
```

Ensuite nous rédigeons les test dont nous avons besoins dans le répertoire test :



Pour lancer les tests deux possibilité, tous les lancer ou lancer une fonction d'un fichier en particulier.

```
./vendor/bin/phpunit
```

```
./vendor/bin/phpunit tests/StackTest.php
```

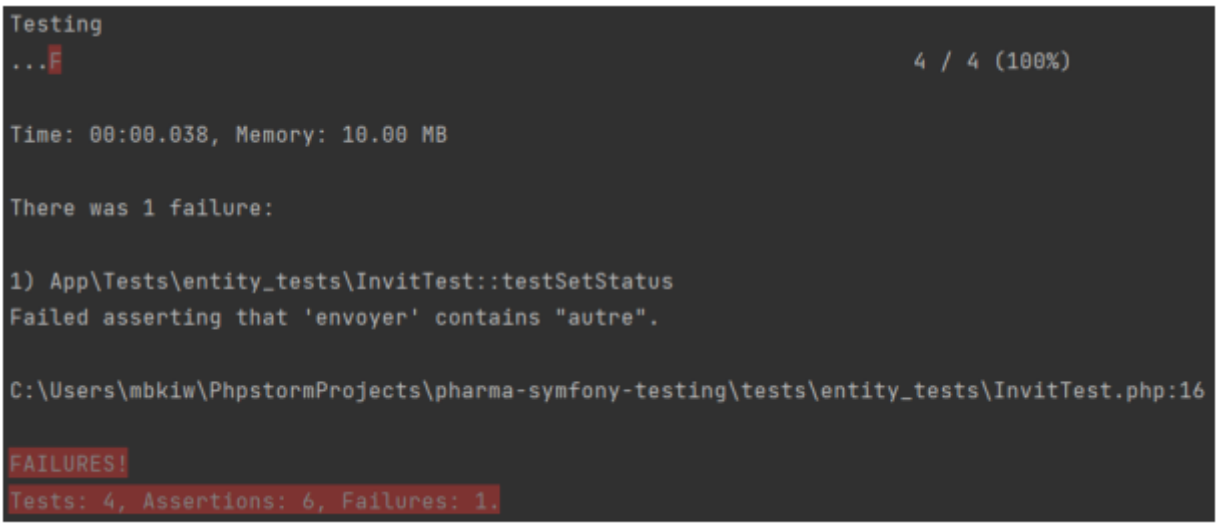
Le test s'exécute puis nous affiche le résultat, soit tout est correcte soit il y a une erreur.

```
PHPUnit 9.5.10 by Sebastian Bergmann and contributors.

Testing
....                                                                4 / 4 (100%)

Time: 00:00.229, Memory: 10.00 MB

OK (4 tests, 6 assertions)
```



2- Tests d’acceptation

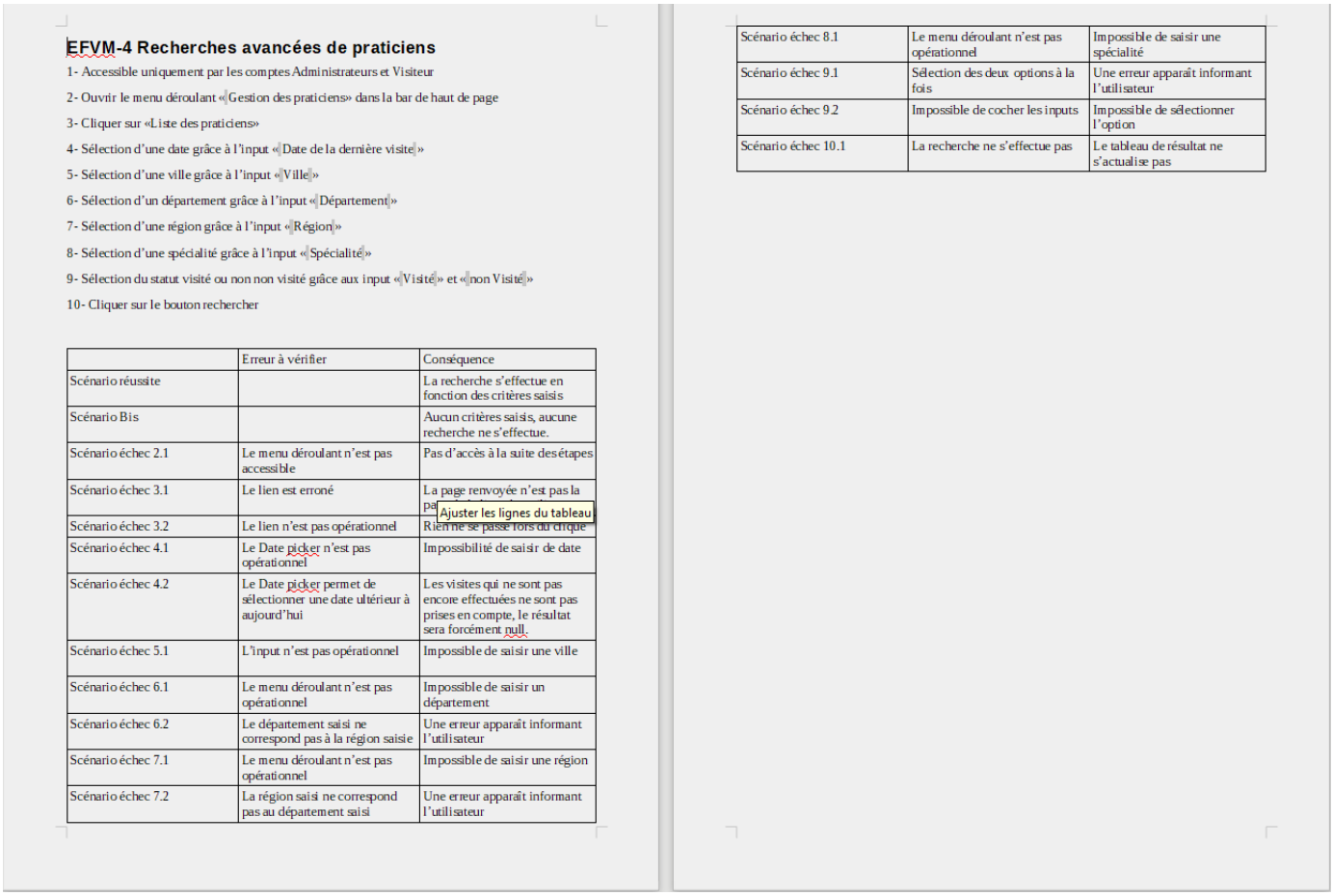
Avant chaque mise à jour de la branche de production l’équipe de développement se mobilise avec les responsables envoyés par le client et repasse chaque nouvelles fonctions. Si toutes les fonctionnalités remplissent bien leurs rôle alors elles seront mises sur la version de production.

3- Test Fonctionnel

Afin de vérifier qu'une fonctionnalité suit bien le fonctionnement que l'on attend, nous mettons en place des tests fonctionnel.

Un document regroupant tous les tests fonctionnels est tenu à jour par l'équipe de développement.

Voici la notice du test fonctionnel des recherches avancés des praticiens.



EFVM-4 Recherches avancées de praticiens

1- Accessible uniquement par les comptes Administrateurs et Visiteur

2- Ouvrir le menu déroulant «Gestion des praticiens» dans la bar de haut de page

3- Cliquer sur «Liste des praticiens»

4- Sélection d'une date grâce à l'input «Date de la dernière visite»

5- Sélection d'une ville grâce à l'input «Ville»

6- Sélection d'un département grâce à l'input «Département»

7- Sélection d'une région grâce à l'input «Région»

8- Sélection d'une spécialité grâce à l'input «Spécialité»

9- Sélection du statut visité ou non non visité grâce aux input «Visité» et «non Visité»

10- Cliquer sur le bouton rechercher

	Erreur à vérifier	Conséquence
Scénario réussite		La recherche s'effectue en fonction des critères saisis
Scénario Bis		Aucun critères saisis, aucune recherche ne s'effectue.
Scénario échec 2.1	Le menu déroulant n'est pas accessible	Pas d'accès à la suite des étapes
Scénario échec 3.1	Le lien est erroné	La page renvoyée n'est pas la pa
Scénario échec 3.2	Le lien n'est pas opérationnel	Rien ne se passe lors du clique
Scénario échec 4.1	Le Date picker n'est pas opérationnel	Impossibilité de saisir de date
Scénario échec 4.2	Le Date picker permet de sélectionner une date ultérieur à aujourd'hui	Les visites qui ne sont pas encore effectuées ne sont pas prises en compte, le résultat sera forcément null.
Scénario échec 5.1	L'input n'est pas opérationnel	Impossible de saisir une ville
Scénario échec 6.1	Le menu déroulant n'est pas opérationnel	Impossible de saisir un département
Scénario échec 6.2	Le département saisi ne correspond pas à la région saisie	Une erreur apparaît informant l'utilisateur
Scénario échec 7.1	Le menu déroulant n'est pas opérationnel	Impossible de saisir une région
Scénario échec 7.2	La région saisi ne correspond pas au département saisi	Une erreur apparaît informant l'utilisateur

Ajuster les lignes du tableau

Scénario échec 8.1	Le menu déroulant n'est pas opérationnel	Impossible de saisir une spécialité
Scénario échec 9.1	Sélection des deux options à la fois	Une erreur apparaît informant l'utilisateur
Scénario échec 9.2	Impossible de cocher les inputs	Impossible de sélectionner l'option
Scénario échec 10.1	La recherche ne s'effectue pas	Le tableau de résultat ne s'actualise pas

B- Déploiement de l'application

Le déploiement se fait automatiquement après chaque merge ou push de la branche de production « master » sur un server Windows 10. Pour cela nous avons configuré le projet tel que le .env déployé soit en mode production mais aussi qu'il utilise la base de donnée du serveur.

Le script de déploiement en .bat .

```
start C:\wamp64\wampmanager.exe
D:
cd gsb
php bin/console cache:clear
php bin/console doctrine:schema:update -f
start symfony server:start
yarn run build
exit
```

Nous avons aussi intégré le fichier .gitlab-ci.yaml pour faire de l'intégration continue :

```
image: jakzal/phpqa:php7.4
before_script:
  - composer install --no-scripts
  - composer update
.only-master:
  only:
    - master
stages:
   - Deploy
deploy:
    stage: Deploy
    extends: .only-master
    script:
      - apt-get update -qq && apt-get install -y -qq lftp
      - lftp -u gsbGitDeploy,GalaxySBdeploy13 ftp.pharma-gsb.fr -p
42421 -e "set ftp:ssl-allow no;debug;mirror --parallel=10 -Rev ./
/pharmagsb --ignore-time --exclude-glob .git* --exclude .git/ --
exclude vendor/ --exclude node_modules/ --exclude public/ --exclude
var/"  || true
      - lftp -u gsbGitDeploy,GalaxySBdeploy13 ftp.pharma-gsb.fr -p
42421 -e "quit"
      - echo "transfert terminé !"
```

Voici quelque stats sur nos déploiement fournis par gitLab