Cahier Des Charges Fonctionnel et Spécifications : Projet annuaire FABOP

Table des matières

I) Introduction

Conception supervisée par : Laure KAHLEM, Gérard ROZSAVOLGYI, Jacques CHABIN, Céline ADOBET, Clément JOUBERT.

Rédaction : Thibaud FAURIE, Alex LEROYER, Othmane EDDAMANI, Guillaume VISSE.

Client : .La Fabrique Opéra Val de Loire.

1) Étude d'opportunité

La Fabrique Opéra Val de Loire (FABOP) est une association loi 1901 à but non lucratif dont l'objectif est de démocratiser l'art lyrique et de permettre à de nouveaux publics de découvrir le dynamisme et le talent de la jeunesse d'un territoire par une approche professionnelle.

L'association fait régulièrement et de manière annuelle appel à des écoliers, étudiants, apprentis et artisans pour organiser une représentation artistique d'un Opéra inscrit dans le patrimoine culturel mondial. Au fil des années, le nombre de bénévoles et de participant n'a fait qu'augmenter pour finalement atteindre un seuil drastique.

Actuellement, l'association gère ces effectifs par le biais d'un système rudimentaire basé sur des modèles de feuilles de calcul. Au vu de l'obsolescence et du caractère inadapté à la manipulation en masse de données de cette solution, monsieur Clément Joubert a décidé de faire appel au département informatique de l'IUT d'Orléans dans le but de permettre à des étudiants en licence professionnelle « conception, développement, et test de logiciel, spécialité web et mobile » de produire une application de gestion d'annuaire personnalisée dans le cadre de leur projet tutoré.

La finalité de ce projet est un besoin urgent en gain de praticité quotidienne pour les administrateurs de cette base de données et soulève un réel soucis d'ergonomie. Elle devra aussi avoir comme fonction de permettre l'import/export depuis et vers l'ancien système de gestion adopté par FABOP.

2) Étude de faisabilité

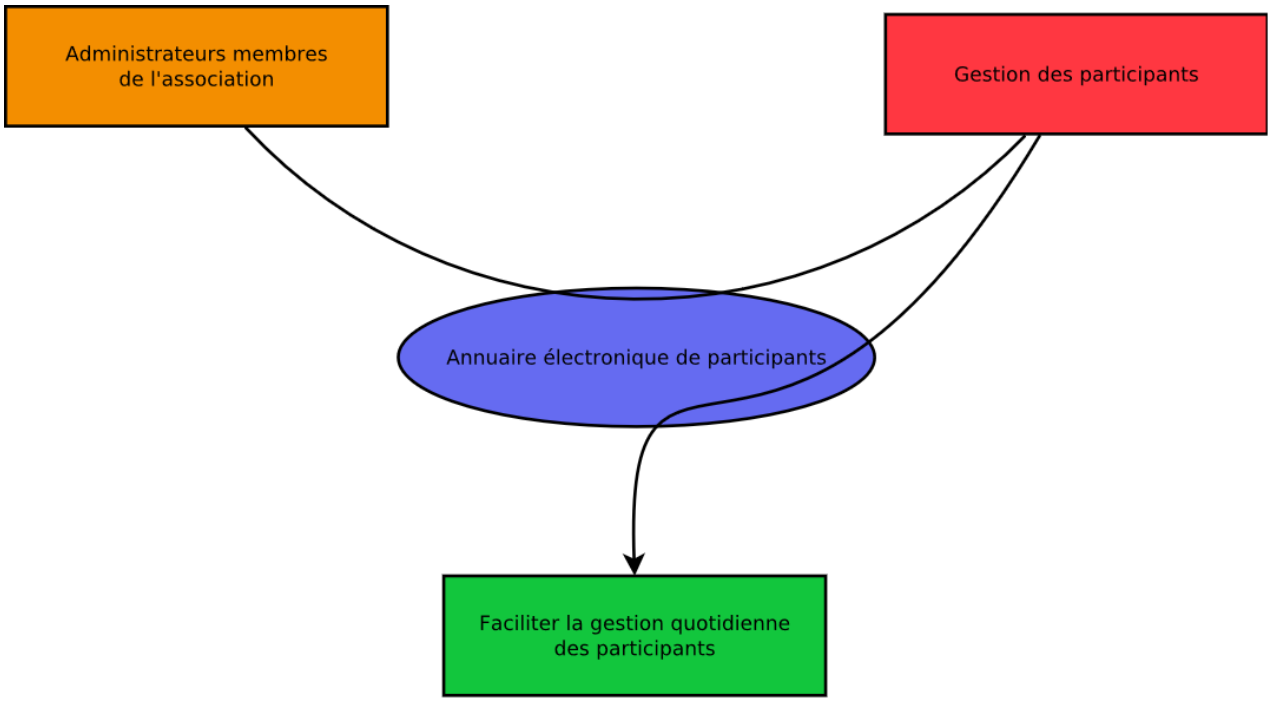
Le projet se voit alloué un budget très modéré et il est préférable de réduire au mieux les OPEX et les CAPEX afin de les faire tendre vers une valeur nulle.

La résultante de ce projet doit être exploitable en production à partir de février 2019.

La planification de projet, à la date du 12 novembre 2018 (voir annexe 1), prévoit une durée comprenant conception et réalisation avoisinant les 100 heures. Les détails cette conception sont explicités à la suite de ce document.

II) Analyse fonctionnelle

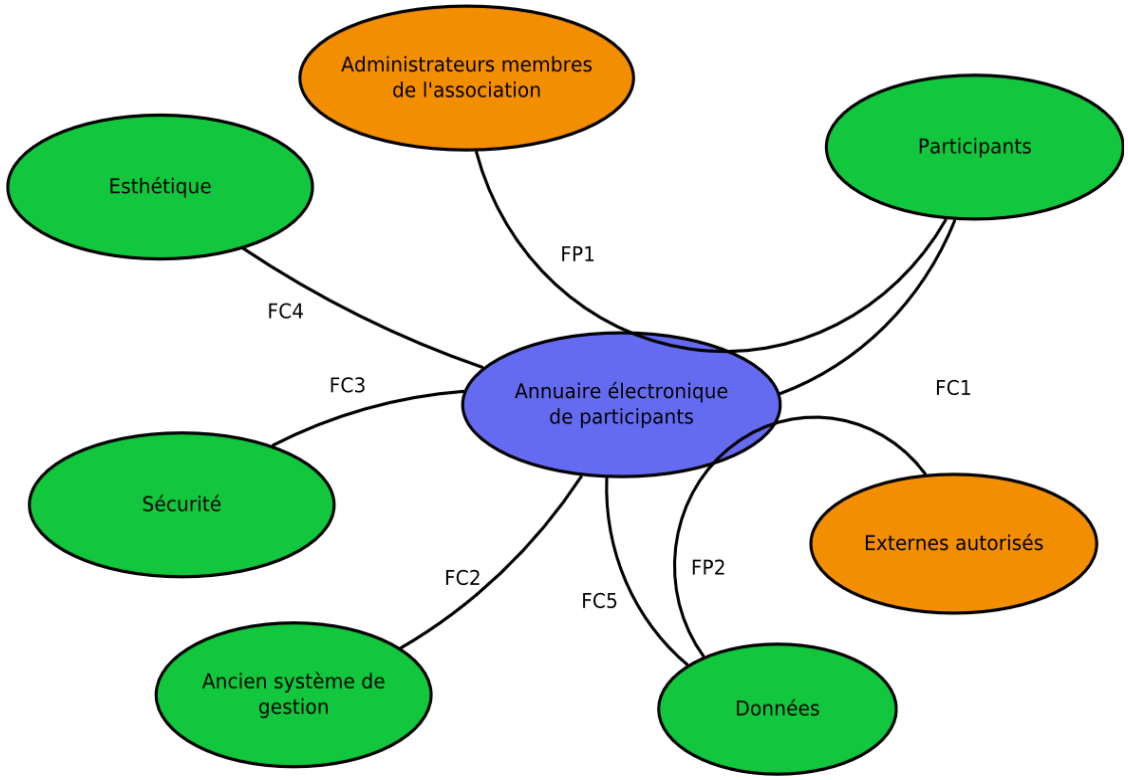
1) Expression du besoin



Fonction principale du produit :

Le produit final est un outil ayant pour but de faciliter la gestion des participants par les administrateurs de l'annuaire.

2) Diagramme des interacteurs



Fonctions :

Nom	Intitulé	Critère	Niveau	Flexibilité	Classe de Flexibilité
FP1	Gérer les participants	Gestion	Gestion manuelle absolue	nulle	F0
FP2	Permettre la saisie des données par les personnes autorisées	Accessibilité	Authentifiée et préalablement autorisée	nulle	F0
FC1	Permettre la personnalisation des champs au cas par cas	Personnalisation	Liberté totale dans le choix des infos à renseigner	nulle	F0
FC2	Permettre l'import/export depuis et vers l'ancien système	Type de fichier	Fichier formaté et type limité ceux utilisé par l'ancien système	+ ou – 1 type de fichiers	F3
FC3	Être sécurisé	Lisibilité des données	- Illisible à un utilisateur non autorisé - Authentification des utilisateurs	nulle	F0

FC4	Doit être agréable à l'utilisation, ergonomique et intuitif	Maturité digitale de l'utilisateur requise	Néophyte	nulle	F0
FC5	PRA/PCA : doit comporter un système simple d'utilisation de backup des données et des configurations	Maturité digitale de l'utilisateur requise	Néophyte	nulle	F1

a) FP1 :

L'utilisateur doit être capable de gérer entièrement la base de données manuellement. Le système doit comporter la possibilité de créer, lister, mettre à jour et supprimer des données.

b) FP2 :

L'utilisateur doit être capable d'autoriser ponctuellement des accès aux responsables de l'inscription de certains participants.

c) FC1 :

L'utilisateur doit être capable d'ajouter/supprimer ponctuellement un champs pour une entité donnée.

d) FC2 :

Pour des raisons de « rétrocompatibilité » et de praticité, l'outil comprendra un module permettant de réadapter le format des données pour le nouveau système et inversement.

e) FC3 :

L'accès à l'outil doit être sécurisé et se faire sur authentification uniquement. Les données ne doivent pas être lisibles/modifiables par un utilisateur non authentifiée.

f) FC4 :

L'outil doit être accessible à n'importe quelle personne même si celle-ci a une très faible connaissance et expérience de l'informatique et de la bureautique.

g) FC5 :

L'outil doit comporter un Plan de Continuité/Reprise d'Activité concernant les données et l'application qui les gère. L'outil devra donc comporter un utilitaire intégré de backup des données et des configurations très simple d'utilisation et très guidé.

III) Solutions techniques

1) FP1/FC1/FC4 : Gestion des données

a) Modèle logique de données :

Le SGBD doit mettre en relation une entité avec sa fiche d'informations.

Le modèle document se prête bien à la fiche puisqu'il permet une liberté presque totale dans le choix des champs de données de chaque entité.

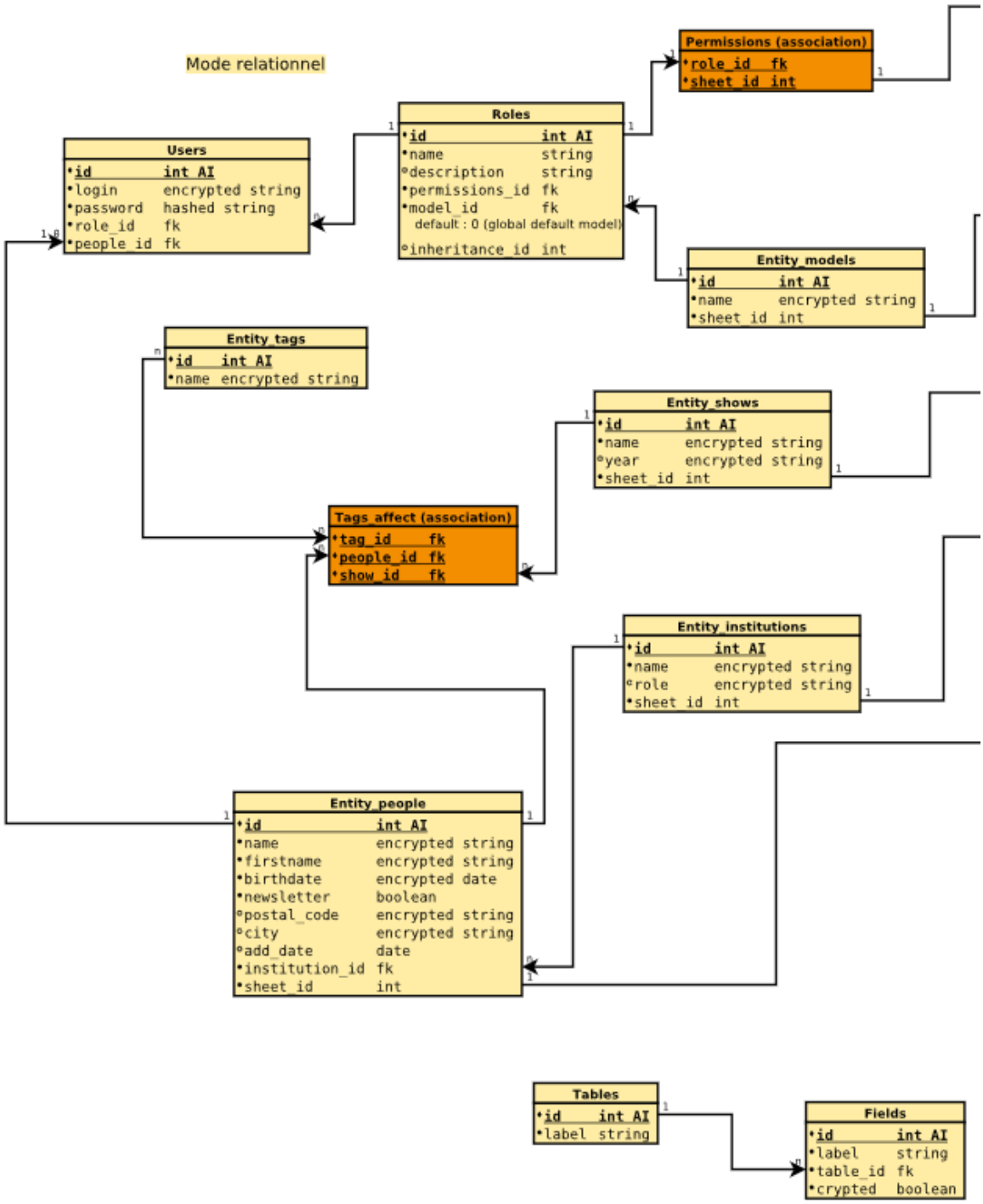
Un utilisateur admin pourra définir un modèle par défaut afin de le proposer comme document d'origine et définir des champs obligatoires pour le formulaire d'ajout d'entité.

Un utilisateur externe pourra utiliser un éditeur de modèle pour créer son propre modèle et ainsi personnaliser ses besoins en terme de renseignement d'informations.

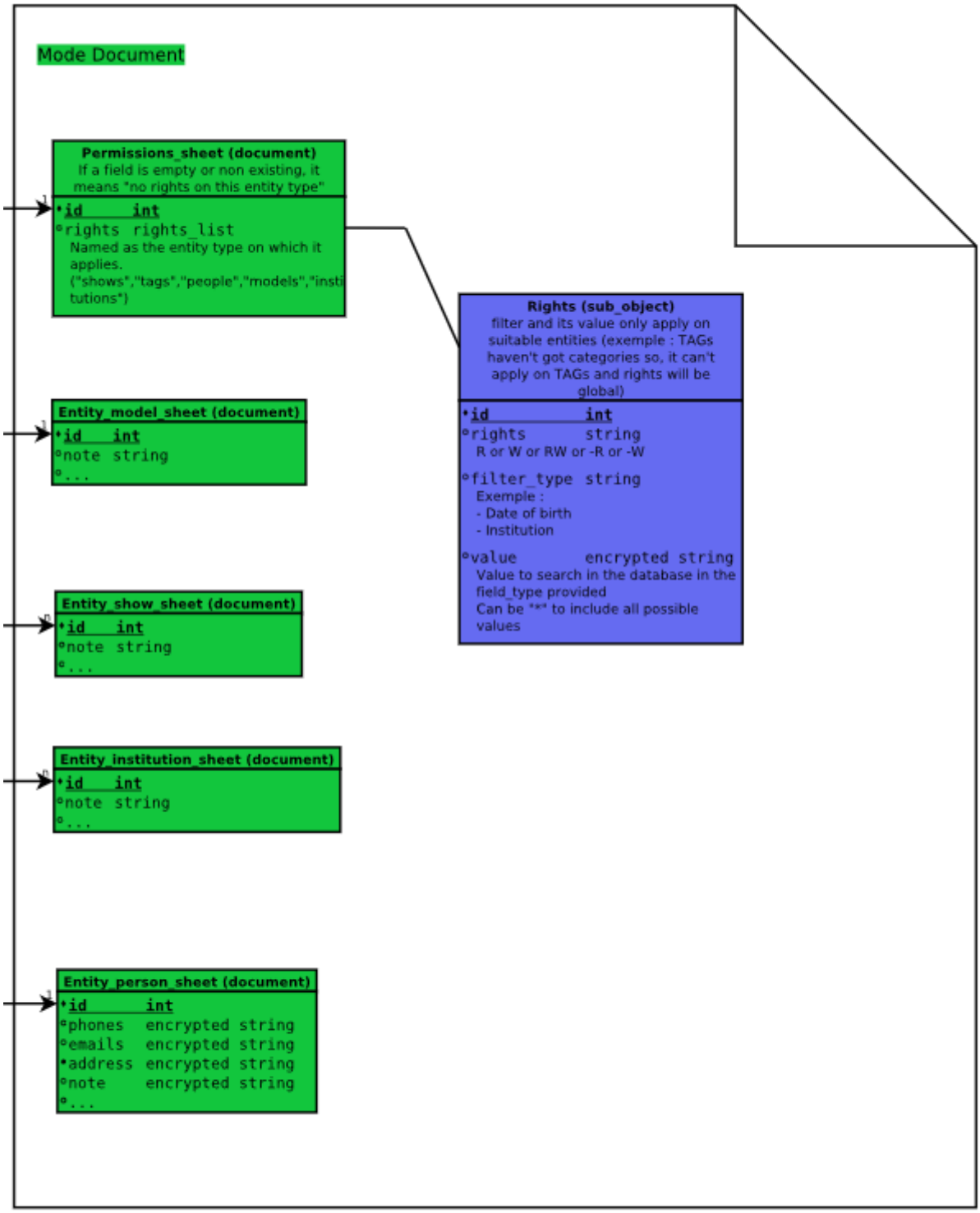
Un utilisateur admin pourra modifier n'importe quel modèle ou entité dans la base de données.

Un utilisateur externe ne pourra modifier que les entités auquel il est rattaché, sur lequel il a les droits (exemple : un professeur d'école ne pourra que modifier ou ajouter des élèves dont il est le responsable)

Le système de modèles de formulaire pourra être un élément central dans la distribution des formulaires de renseignement vers les utilisateurs externes désignés comme responsables d'inscription.



Modèle de données (partie en modèle relationnel)



Modèle de données (partie en modèle document)

Le modèle est séparé en deux modules : relationnel et document.

Le module relationnel est destiné à effectuer un traitement pratique des données ainsi que des filtrages de manière fixe est définie à l'avance alors que la partie créée sur le modèle document est destinée à rendre l'application modulaire et le type des données personnalisable.

Ainsi, le modèle relationnel permettrait d'optimiser les performances sur le traitement des données tandis que le modèle document permettra de définir une fiche informative à chaque entité permettant ainsi l'implémentation d'un schéma souple et ainsi l'entrée et la création de nouveaux types d'informations sans le bridage qu'implique un schéma fixe.

b) Représentation et gestion des données :

Lors du chargement du site, on sera automatiquement amené sur une page d'authentification pour s'identifier en tant qu'utilisateur enregistré.

Une fois connecté, le site sera composé d'une page principale qui intégrera la liste des participants au projet OPERA Val de Loire. Pour un simple lecteur, il sera possible d'avoir accès à la liste des participants qui le concerne. Il pourra donc faire un tri pour affiner les personnes dont il veut les informations, et exporter les données de celles-ci. La liste des participants sera donc accompagné d'une barre de recherche, ainsi que des filtres leur permettant d'être plus précis dans la recherche.

Un contributeur, quant à lui, aura le même accès que les lecteurs, avec en plus les droits de modifications sur les participants qui lui sont attribués. En ce qui concerne la modification, le contributeur aura accès à une page regroupant toutes les informations entrées pour le participant, ayant pour choix de les enregistrer, ou de les exporter pour la personne choisie. Le contributeur a également le droit de supprimer un participant de la liste, si celui-ci y a accès.

Enfin, l'administrateur pourra en plus, avoir accès à une page différente, dans laquelle il pourra gérer chaque utilisateur, leur rôle au sein de l'association, ainsi que les modèles que ces derniers pourront utiliser pour ajouter des données à la liste de participants. Un menu lui sera également accessible permettant d'avoir accès aux différentes institutions qui sont impliqués dans le projet (qui envoie des participants dans l'association), ainsi que les différents tags qui seront utilisés pour définir les tâches du participant au sein de l'association.

2) FP2/FC3 : Gestion des accès

L'accès à la saisie nécessitera une authentification préalable.

Le formulaire d'ajout de participants devra contenir des champs obligatoires, et des champs modulables pour permettre l'ajout des données nécessaires aux représentants.

La base de données devra contenir un référencement des droits ou/et des rôles qui sera attribué aux utilisateurs :

- On gérera les droits et/ou les rôles sur une page de saisie de données administrateur
- La possibilité de modifier des informations sera régie par les rôles et/ou les droits
- Des types de rôles seront déjà prédéfinis : Accès lecteur (Possibilité de tri et d'export des données), Administrateur (Accès total), Contributeur (Modification des fiches attribués et accès aux modèles d'export qui lui sont assignés)

Les permission seront en premier lieu une gestion de la lecture et de l'écriture de la BD.

Un participant sera automatiquement rattaché à un ou plusieurs responsables par le biais de son

institution.

Il faudra définir des “templates” de formulaires pour correspondre à tous les participants.

Les contributeurs pourront modifier les “templates” qui leur auront été assigné pour correspondre aux informations à renseigner.

L’accès aux données sera géré par les rôles définis par l’administrateur.

Les actions réalisées sur les participants (ajout, modification, etc ...) seront enregistrées dans un registre pour avoir un suivi.

formatage des entrées du registre : “Date;Heure;Élément;Type_action;Commentaire;ID_utilisateur”

3) FC2 : Système import / export

L’utilisateur administrateur doit pouvoir choisir entre l’ancien système et le nouveau. L’ancien système correspondrait à une manière alternative de sortir les données de la base.

Nécessité d’adapter un processus de conversion du formatage des données entre le nouveau et l’ancien format.

Un bouton d’export sera présent sur sur la fiche personnel et sur le listing pour récupérer les données sélectionnées.

Conversion entre l’ancien et le nouveau système d’import/export qui permet de garder le même format de fichier que celui déjà en place pour ne pas déstabiliser les utilisateurs.

L’utilisateur pourra exporter, soit la liste entière des participants (avec ou sans filtres selon son rôle et son choix), ou alors une fiche particulière si besoin d’informations sur un seul participant. L’exportation pourra se faire en fonctions des tags.

Un fichier sera mis à disposition d’un fichier modèle pour l’importation afin de donner un exemple des informations à renseigner sous les deux formats (ancien et nouveau).

L’importation/exportation dépendra des droits assignés au préalable aux utilisateurs

4) FC3/FC5 : Sécurisation de l’application et des données

Chiffrer les données nécessitera de passer par un système intégré dans le SGBD ou par une API externe ou le développement d’une API propre.

L’application comportera une authentification des utilisateurs à simple facteur ainsi qu’une vérification interne de l’authenticité de l’acteur (personne ou logiciel) lors de la modification des données par le biais d’un token d’authentification.

Les process de gestion des données seront uniquement utilisés en local.

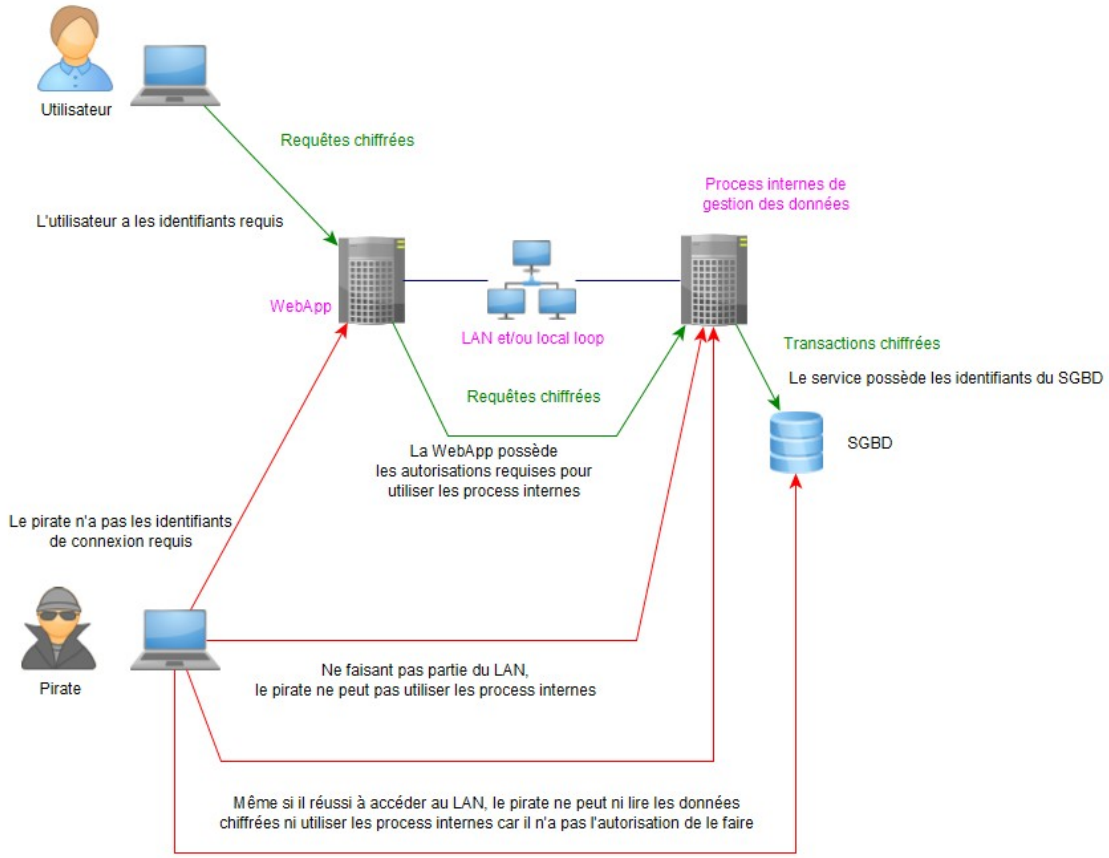


Schéma « sécurité » du projet FABOP

L'application comportera un système de sauvegarde des données utilisé au cas où les données soient corrompues malgré les précautions précédentes. (Donnera lieu au PRA/PCA).

IV) Choix technologiques

1) Front-end :

a) Gestion des paquets :

Pour la gestion des paquets, on utilisera yarn pour sa capacité de verrouillage des versions dans le but de faciliter la gestion des paquets entre toutes les machines de développement.

b) Gestion des assets :

Pour la gestion des assets (css/js), on utilisera gulp via node.js de manière à rendre la gestion plus efficace en automatisant via des tâches de routine.

On créera un dossier “assets” à la racine du projet de manière à centraliser les ressources du gulpfile.js. Ainsi, la gestion est plus efficace et permet une meilleure gestion dans le versionnage Git.

Les résultats des compilations iront trouver leur place dans les dossiers de chaque app/bundle.

Les types d’assets seront des images, des feuilles de style, des scripts JS ainsi que des feuilles scss.

Les tâches gulp seront réparties selon le type d’assets.

2) Back-end :

Après avoir réalisé l’analyse du projet qui nous a été légué, nous avons décidé de réfléchir sur les outils que nous allons utiliser pour continuer le projet.

En ce qui concerne le framework nous sommes dans l’obligation de continuer le travail en utilisant le framework Symfony, la version utilisé dans le projet est la 3.4, nous avons décidé de passer en Symfony 4 car celle-ci nous offre plus de fonctionnalités et surtout que celle-ci est déjà stable comme on peut le voir dans le diagramme ci-dessous :

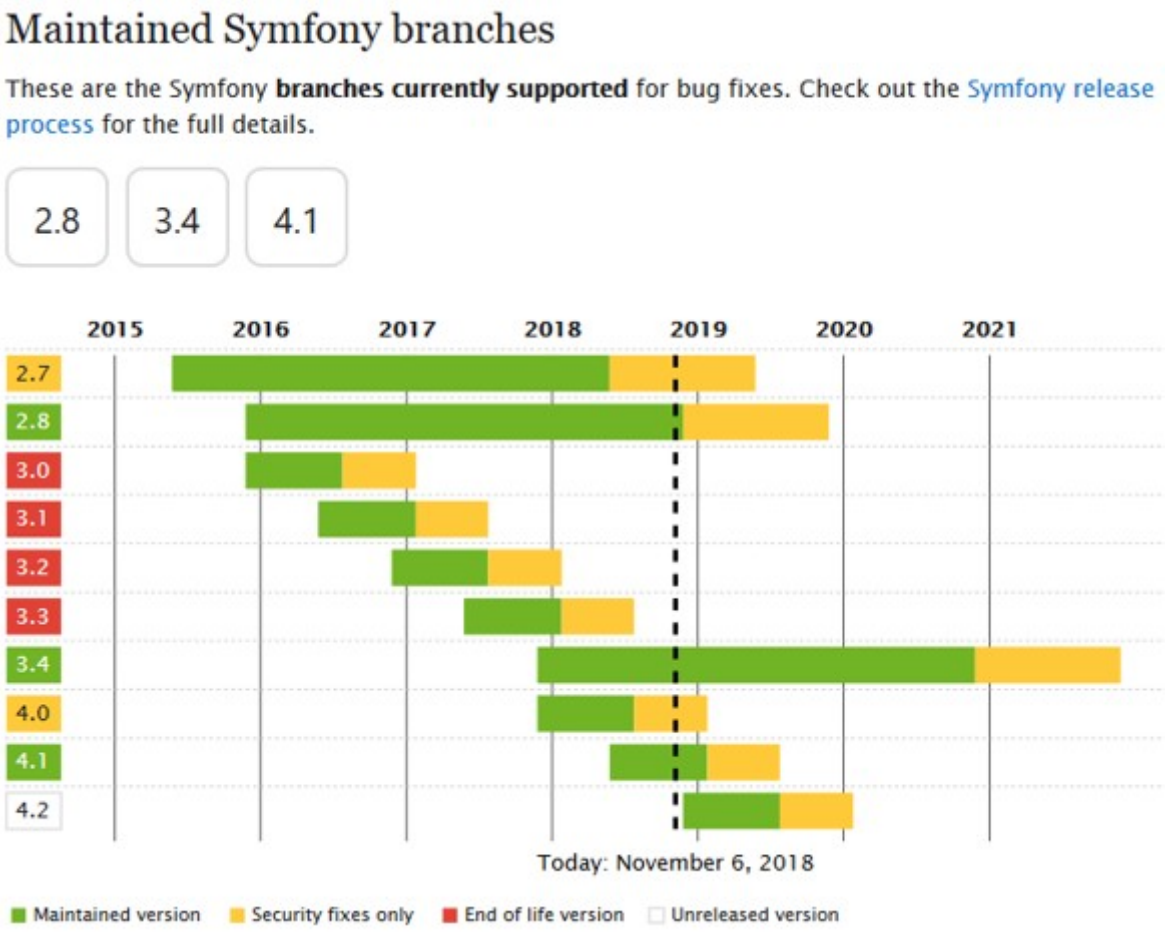
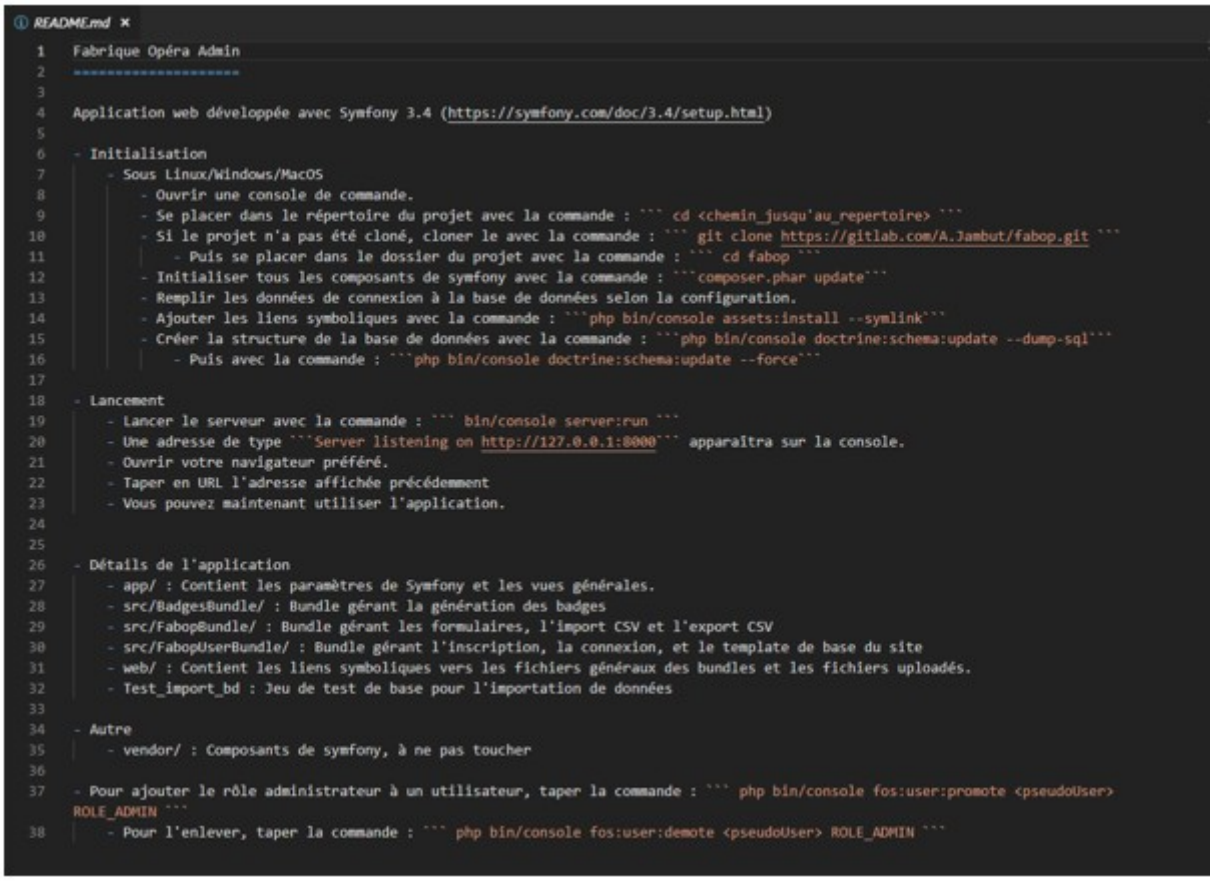


Diagramme des branches Symfony (source : symfony.com)

En ce qui concerne la version de php on remarque que, dans le ReadMe qui doit référencer la version de php qui est utilisé, cette dernière n'est pas donné comme on peut le voir ci-dessous:



README.md

```
Fabrique Opéra Admin
====================

Application web développée avec Symfony 3.4 (https://symfony.com/doc/3.4/setup.html)

- Initialisation
    - Sous Linux/Windows/MacOS
        - Ouvrir une console de commande.
        - Se placer dans le répertoire du projet avec la commande : ``` cd <chemin_jusqu'au_repertoire> ```
        - Si le projet n'a pas été cloné, cloner le avec la commande : ``` git clone https://gitlab.com/A.Jambut/fabop.git ```
            - Puis se placer dans le dossier du projet avec la commande : ``` cd fabop ```
        - Initialiser tous les composants de symfony avec la commande : ```composer.phar update```
        - Remplir les données de connexion à la base de données selon la configuration.
        - Ajouter les liens symboliques avec la commande : ```php bin/console assets:install --symlink```
        - Créer la structure de la base de données avec la commande : ```php bin/console doctrine:schema:update --dump-sql```
            - Puis avec la commande : ```php bin/console doctrine:schema:update --force```

- Lancement
    - Lancer le serveur avec la commande : ``` bin/console server:run ```
    - Une adresse de type ```Server listening on http://127.0.0.1:8000``` apparaîtra sur la console.
    - Ouvrir votre navigateur préféré.
    - Taper en URL l'adresse affichée précédemment
    - Vous pouvez maintenant utiliser l'application.


- Détails de l'application
    - app/ : Contient les paramètres de Symfony et les vues générales.
    - src/BadgesBundle/ : Bundle gérant la génération des badges
    - src/FabopBundle/ : Bundle gérant les formulaires, l'import CSV et l'export CSV
    - src/FabopUserBundle/ : Bundle gérant l'inscription, la connexion, et le template de base du site
    - web/ : Contient les liens symboliques vers les fichiers généraux des bundles et les fichiers uploadés.
    - Test_import_bd : Jeu de test de base pour l'importation de données

- Autre
    - vendor/ : Composants de symfony, à ne pas toucher

- Pour ajouter le rôle administrateur à un utilisateur, taper la commande : ``` php bin/console fos:user:promote <pseudoUser>
ROLE_ADMIN ```
    - Pour l'enlever, taper la commande : ``` php bin/console fos:user:demote <pseudoUser> ROLE_ADMIN ```
```

Extrait du README (projet année 2017-2018)

On suppose que si la version de PHP utilisée n'est pas renseignée, cela veut dire que le projet est compatible avec toutes les versions stables déjà sorties ainsi que sécurisés, soit toutes les versions maintenues.

En ce qui concerne notre projet nous avons choisis d'opter pour la version de PHP 7.2, qui est la dernière version stable sortie et aussi la plus complète.

Pour Symfony, nous avons un certain nombre de composants que nous voudrions intégrer tel que :

- Asset :

Gère la génération des URL et du versionnage des assets web tel que les pages CSS, les fichiers JS ainsi que les images

- Debug :

Des outils pour faciliter le deboguage du PHP

- Form :

Mets à disposition des outils pour faciliter la création, la gestion et la réutilisation

- PHPUnit Bridge :

Pour la gestion des tests unitaires

- twig-bundle :

Pour la gestion des templates

- orm-pack :

Pour la gestion des Entités et de la base de données

- web-server-bundle :

Met à disposition les commandes permettant de faire marcher des applications en utilisant le PHP Built in

- annotations :

Pour la gestion des routes

3) Sécurisation de l'infrastructure :

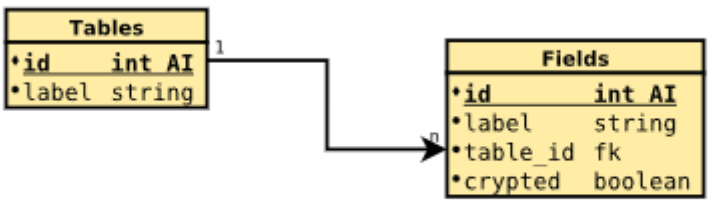
Afin de sécuriser l'infrastructure, nous avons décidé de choisir entre 2 bibliothèques, nous n'allions pas utiliser *crypt()* pour chiffrer des données personnelles car cette fonction hash juste le texte, nous avons besoin de le déchiffrer derrière et non de le comparer. Pour choisir entre les deux bibliothèques, nous les avons comparer. Nous avons fais notre choix entre Halite et Libsodium. Cette dernière étant la plus sécurisé des librairies de cryptographie actuellement et Halite étant une extension de Libsodium, elle embarque plus de choses.

Par exemple Halite permet de chiffrer des fichiers entiers, des cookies et d'autres encore.

De plus, Halite est plus intuitive, possède une interface plus simple et est plus rapide que Libsodium. Face à tous ces avantages, notre choix c'est porté sur Halite qui semble plus facile d'utilisation sans pour autant être moins sécurisé que Libsodium.

Les données que nous aurons à chiffrer sont des plus sensibles car ce sont des données personnelles, elles seront chiffrer en entrée et déchiffrer en sortie de la base de données. Les utilisateurs qui voudront s'authentifier passeront par le *SecurityBundle* de Symfony, ils auront ensuite accès aux données personnelles déchiffrées dès l'authentification réussie.

Pour identifier ces données, nous allons devoir les marquer en créant deux table comme-ci dessous. On y trouvera tous les champs de la base de données organisés selon un système permettant d'identifier leur table et leur politique de chiffrement.



4) Import/Export:

Afin de faciliter l'import et l'export de données, nous allons définir des modèles de tableau pour l'import et l'export. Par exemple, un modèle de fiche personnelle, de liste d'un orchestre, de liste d'élève… L'utilisateur pourra ajouter et modifier des modèles, afin de donner plus de détails ou de grouper les participants sous une autre catégorie.

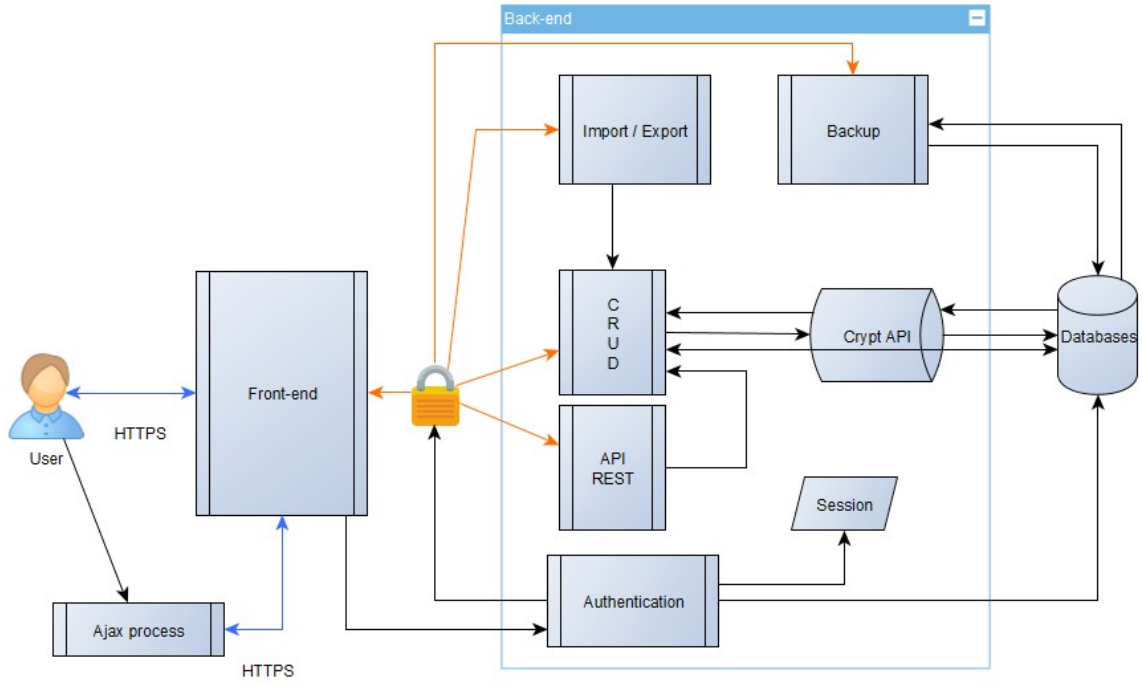
En cherchant des solutions technologiques, nous avons pus trouver deux librairies qui peuvent effectuer l'export : mk-j PHP_XLSXWriter et Spout. En les comparant nous pouvons voir que en terme de performance, ces deux librairies sont au même niveau. Par contre Spout est avantagé par le fait qu'elle puisse exporter en fichier csv, et ods. Pour effectuer l'export nous utiliserons Spout.

Comme notre projet est un projet qui date de l'année dernière, il y a déjà des fonctionnalités implémentées, notamment pour l'import depuis un fichier csv vers notre base de donnée. Il nous faudra juste détaillé cette fonctionnalité.

Quand l'utilisateur choisira d'exporter une liste ou une fiche personnelle, le fichier créé portera l'extension .xlsx

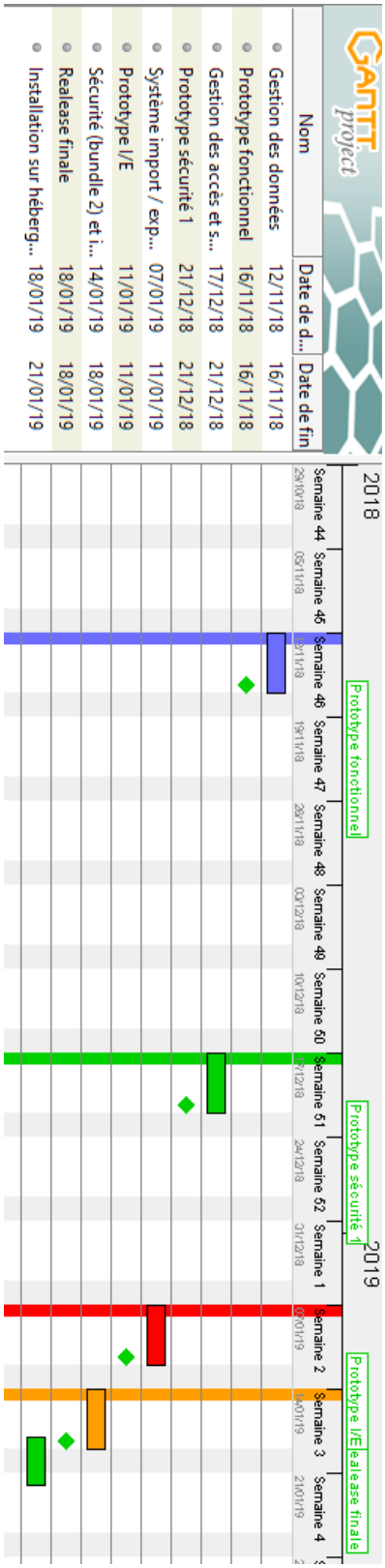
Conversion excel to JSON : https://www.npmjs.com/package/convert-excel-to-json

V) Architecture de l'application :



VI) Annexes

1) Annexe 1 : Gantt prévisionnel SCRUM



2) Annexe 2 : PRA

a) Plan de gestion de crise :

Responsables :

Administrateurs du système d'information.

Types de crises possibles :

- Panne matérielle sur la machine physique
- Panne matérielle sur le réseau
- Panne logicielle de la machine virtuelle (cessation de fonctionnement)
- Panne logicielle sur la machine virtuelle (cessation de fonctionnement d'un service)

Identification de la cause :

1. Vérification du bon fonctionnement du matériel physique de la machine d'accès jusqu'au serveur physique
2. Vérification du bon fonctionnement du système de virtualisation et de la machine virtuelle
3. Vérification du bon fonctionnement du service web et du SGBD

b) Plan de reprise informatique

Manipulations en fonction de la cause :

1. -Tenter une remise en route du matériel
   -Tenter un remplacement du matériel défectueux

2. -Tenter une remise en route de la machine virtuelle
   -Tenter une remise en route du système de virtualisation
   -Tenter un redémarrage du matériel physique
   -Tenter une restauration de la machine virtuelle à partir d'une sauvegarde

3. -Tenter un redémarrage des services
   -Tenter une restauration de la BDD
   -Tenter une restauration des fichiers de configuration
   -Tenter une restauration de la machine virtuelle à partir d'une sauvegarde

Mise en place d'une redondance BDD Maître-Maître ?

Mise en place d'un script de sauvegarde automatique A CRYPTER (contiendra sûrement le mdp + id de l'utilisateur admin de la BDD)

Le redémarrage des services et la restauration des fichiers de config seront effectués manuellement à l'aide de protocoles.

c) Plan de sauvegarde :

Manipulation récurrentes :

1. Sauvegarde quotidienne incrémentale de la BDD et des fichiers de configuration :
   - Les fichiers et la BDD seront gardés jusqu'à la validation et sauvegarde hebdomadaire de la BDD.
2. Sauvegarde hebdomadaire incrémentale de la BDD :
   - Sauvegarde informatique chaque vendredi soir