

# Rapport de stage

*Rapport de stage effectué du 6 Janvier au 28 février*

## Sujet de stage

Contribution au développement d'une application web pour le Secours Populaire de Rennes, visant à améliorer le système Backend de certains modules

## Thème abordé :

- Développement web

Tuteur de stage : BAUCHE Gwenael

Directeur de l'établissement : Jean-François ARRIVÉ

Entreprise d'accueil : Secours Populaire français de Rennes, 7 rue des Landelles, 35000 Rennes

# Sommaire

Sommaire .....	2
I Remerciement.....	4
II Introduction .....	5
Contexte général .....	5
Contexte du stage.....	6
Objectifs et structure du rapport .....	7
III Présentation du Secours Populaire Français et de son antenne à Rennes.....	8
A - Historique et missions du Secours Populaire Français.....	8
Création et évolution de l'association au niveau national.....	8
Valeurs et principes fondamentaux.....	9
Domaines d'intervention .....	9
Structure organisationnelle générale .....	10
B. Le Secours Populaire Français à Rennes .....	11
Historique de l'implantation locale .....	11
Structure et organisation de l'antenne rennaise .....	11
Public cible et typologie des bénéficiaires .....	11
Partenariats locaux .....	12
Moyens de financement et ressources .....	12
Présentation des différents lieux d'accueil à Rennes.....	12
Déroulement du stage et missions réalisées.....	13

A. Objectifs du stage, et présentation des actions réalisés. ....	14
B. Description détaillée des missions : .....	16
Technologies.....	26
Technologies utilisées .....	26
Technologies apprises .....	27
Ressenti du stage .....	27
Aspects positifs .....	27
Aspects négatifs .....	28
Apprentissages du monde professionnel grâce au Secours Populaire Français .....	28
Engagement et investissement des salariés .....	29
Apprentissages techniques et adaptation au contexte.....	29
Ouverture sur la suite et projet professionnel .....	30
Missions externes au stage .....	31
Conclusion .....	32
<b>Annexes .....</b>	<b>33</b>

## I Remerciement

Je tiens à exprimer ma sincère gratitude au **Secours Populaire Français de Rennes** pour m'avoir accueilli au sein de leur structure et m'avoir permis de réaliser ce stage dans des conditions optimales. Cette expérience m'a offert l'opportunité de mettre en pratique mes connaissances en développement informatique tout en contribuant à un projet à vocation solidaire.

Je remercie tout particulièrement **M. Gwenael Bauché**, mon tuteur de stage, pour son accompagnement, sa disponibilité et ses conseils précieux tout au long de cette période. Son expertise et son soutien ont grandement contribué à mon apprentissage et à la réussite de mes missions.

Je souhaite également remercier **M. Jean-François Arrivé**, directeur de l'établissement, pour m'avoir offert l'opportunité d'évoluer au sein de cette association et d'y découvrir un environnement de travail enrichissant.

Enfin, je remercie toutes les personnes avec qui j'ai pu échanger et collaborer durant ce stage. Cette expérience a été une étape formatrice dans mon parcours, m'apportant de nouvelles compétences et renforçant ma motivation à poursuivre dans le domaine du développement informatique.

## II Introduction

### Contexte général

Dans un contexte où la solidarité joue un rôle essentiel pour pallier les inégalités sociales et venir en aide aux populations en difficulté, les associations occupent une place centrale au sein de la société. Elles permettent de mobiliser des ressources humaines et matérielles afin d'accompagner les personnes les plus vulnérables. Face à l'augmentation des besoins en assistance sociale, les structures associatives doivent sans cesse s'adapter et optimiser leur organisation pour offrir un soutien efficace et durable.

Parmi ces associations, le Secours Populaire Français (SPF) se distingue par son engagement envers la lutte contre la pauvreté et l'exclusion. Créeé en 1945, cette organisation à but non lucratif a pour mission de venir en aide aux personnes en situation de précarité, sans distinction d'origine ou de statut. Grâce à un réseau de bénévoles et de partenaires, le SPF intervient sur de nombreux fronts : aide alimentaire et vestimentaire, accès aux soins, soutien scolaire, accompagnement des personnes isolées, ainsi que l'organisation d'actions de solidarité en France et à l'international.

Afin de remplir efficacement ses missions, le SPF repose sur une gestion rigoureuse de ses bénévoles et de ses ressources. L'informatique joue ainsi un rôle clé dans l'optimisation des processus internes, notamment à travers des outils de gestion adaptés aux besoins de l'association. Mon stage s'inscrit dans cette dynamique, avec pour objectif d'améliorer le logiciel de gestion des bénévoles et des missions, afin de faciliter le travail quotidien des acteurs engagés au sein de la structure.

## Contexte du stage

Le **Secours Populaire Français de Rennes** est une antenne locale du SPF qui œuvre au quotidien pour venir en aide aux personnes en situation de précarité dans le département d'Ille-et-Vilaine. En complément des actions menées au niveau national, cette antenne adapte ses initiatives aux besoins spécifiques de la région rennaise. Elle organise notamment des distributions alimentaires, des collectes de dons, ainsi que des événements de solidarité à destination des familles en difficulté. Pour assurer une gestion efficace de ses ressources humaines et matérielles, le SPF de Rennes s'appuie sur un réseau de bénévoles et de partenaires locaux.

C'est dans ce cadre que j'ai réalisé mon **stage du 6 janvier au 28 février**, sous la tutelle de **M. Gwenael Bauché**, assistant informatique du SPF d'Ille-et-Vilaine. Ce stage s'inscrit dans mon **parcours de formation en informatique à l'EPSI de Rennes**, où je prépare un BTS avec option SLAM. Mon projet principal a consisté à **contribuer au développement d'une application web** destinée à optimiser la gestion des bénévoles et des missions au sein de l'association. Ce travail m'a permis d'appliquer mes compétences en **développement web** tout en répondant à un besoin concret de l'organisation.

## Objectifs et structure du rapport

Ce rapport de stage a pour objectif de rendre compte de l'expérience professionnelle que j'ai vécue au sein du Secours Populaire Français de Rennes. Il vise à analyser les missions qui m'ont été confiées, à mettre en lumière les compétences techniques et relationnelles que j'ai développées, et à évaluer les résultats obtenus. Ce document me permet également de prendre du recul sur mon apprentissage et de réfléchir à l'impact de cette expérience sur mon parcours professionnel.

Afin d'offrir une lecture claire et structurée, ce rapport est organisé en plusieurs parties :

- Tout d'abord, une **présentation du Secours Populaire Français**, à travers son historique, ses missions et ses valeurs fondamentales, ainsi qu'un focus sur l'antenne de Rennes et son fonctionnement.
- Ensuite, le **déroulement du stage et les missions réalisées** seront détaillés, en mettant en avant les tâches effectuées, les outils utilisés et les résultats obtenus. Cette partie comprendra également une présentation des activités annexes auxquelles j'ai participé en dehors du développement informatique.
- **L'analyse et le bilan du stage** permettront d'évaluer les compétences acquises, de confronter les objectifs initiaux aux résultats obtenus et d'apporter une réflexion personnelle sur cette expérience.

- Enfin, une **conclusion** viendra synthétiser les principaux enseignements tirés du stage et ouvrir sur les perspectives d'avenir.

## III Présentation du Secours Populaire Français et de son antenne à Rennes

### A - Historique et missions du Secours Populaire Français

#### Création et évolution de l'association au niveau national

Le **Secours Populaire Français (SPF)** est une association de solidarité créée en **1945**, dans l'après-guerre, avec pour objectif d'apporter une aide immédiate aux populations touchées par les conflits et la misère. À l'origine, l'association s'est construite sur les valeurs de fraternité et d'entraide, visant à soutenir les plus démunis en leur fournissant une assistance matérielle et morale.

Au fil des décennies, le SPF a su **adapter ses actions aux évolutions sociales et économiques**. Il est passé d'une organisation centrée sur l'aide d'urgence à un acteur majeur de la solidarité en France et à l'international. Aujourd'hui, le SPF intervient dans des domaines variés pour répondre aux besoins croissants des personnes en difficulté, tout en développant des actions de prévention et d'insertion sociale.

Le SPF est reconnu d'**utilité publique depuis 1985** et a su s'imposer comme un acteur incontournable de l'**économie sociale et solidaire**. Présent sur l'ensemble du territoire français à travers de nombreuses antennes locales, il repose sur un réseau dense de bénévoles, qui constituent la force vive de l'association.

## Valeurs et principes fondamentaux

Le **Secours Populaire Français** fonde son action sur des valeurs fortes :

- **La solidarité** : Aider les personnes en situation de précarité, sans discrimination d'origine, de statut ou de religion.
- **L'engagement bénévole** : L'association fonctionne grâce à l'implication de milliers de bénévoles qui œuvrent au quotidien pour apporter leur soutien aux plus fragiles.
- **L'indépendance** : Le SPF est une organisation indépendante, à la fois politiquement et financièrement, ce qui garantit la liberté de ses actions et de ses prises de position.
- **La dignité et l'autonomie** : L'association ne se limite pas à l'assistance, mais cherche aussi à favoriser l'insertion et l'émancipation des bénéficiaires en leur donnant les moyens de surmonter leurs difficultés.
- **L'éducation et la sensibilisation** : Lutter contre la pauvreté passe aussi par l'éducation populaire et la mobilisation des citoyens sur les enjeux sociaux.

## Domaines d'intervention

Le SPF agit dans de nombreux domaines pour répondre aux besoins essentiels des populations en difficulté :

- **Aide alimentaire et vestimentaire** : Le SPF organise des distributions de denrées alimentaires et de vêtements pour garantir des conditions de vie dignes aux personnes en précarité.
- **Accès aux droits et à la santé** : L'association accompagne les bénéficiaires dans leurs démarches administratives, propose des consultations médicales et facilite l'accès aux soins pour les personnes exclues des circuits traditionnels.
- **Culture et loisirs** : Le SPF favorise l'accès à la culture, aux loisirs et aux vacances pour les enfants et les familles défavorisées, à travers des initiatives comme les **Journées des Oubliés des Vacances**.
- **Soutien aux enfants et à l'éducation** : Des aides sont apportées pour le soutien scolaire, l'accès aux fournitures scolaires et la lutte contre le décrochage scolaire.
- **Insertion sociale et professionnelle** : L'association met en place des dispositifs d'accompagnement pour favoriser le retour à l'emploi et l'insertion des publics les plus vulnérables.
- **Solidarité internationale** : Le SPF intervient également à l'étranger lors de crises humanitaires, en soutenant des populations touchées par des catastrophes naturelles, des conflits ou des crises économiques.

#### Structure organisationnelle générale

Le Secours Populaire Français repose sur un modèle décentralisé, organisé en plusieurs niveaux :

- **Le niveau national** : Il définit les grandes orientations stratégiques de l'association et coordonne les actions d'envergure nationale et internationale.
- **Les fédérations départementales et les comités locaux** : Ils assurent la mise en œuvre des actions sur le terrain, en fonction des besoins spécifiques des territoires. Chaque antenne locale fonctionne de manière autonome, tout en respectant la charte et les principes du SPF.
- **Les bénévoles** : Ils constituent le cœur de l'association et prennent en charge la gestion des activités, l'accompagnement des bénéficiaires et l'organisation des événements solidaires.

- **Les partenaires et donateurs :** Le SPF collabore avec d'autres associations, des entreprises, des collectivités locales et des citoyens engagés pour financer et déployer ses actions.

L'ensemble de cette organisation permet au Secours Populaire Français d'agir efficacement sur le terrain et d'apporter des solutions adaptées aux réalités locales, tout en restant fidèle à sa mission initiale : lutter contre toutes les formes d'exclusion et de précarité.

## B. Le Secours Populaire Français à Rennes

### Historique de l'implantation locale

Le Secours Populaire Français (SPF) est présent à Rennes depuis plusieurs décennies, s'engageant activement dans la lutte contre la pauvreté et l'exclusion sociale. Au fil des années, l'antenne rennaise a étendu son réseau et ses activités pour répondre aux besoins spécifiques de la population locale, en s'appuyant sur les valeurs de solidarité et d'entraide qui caractérisent l'association depuis sa création en 1945.

### Structure et organisation de l'antenne rennaise

L'antenne de Rennes est structurée autour de plusieurs équipes de bénévoles dédiées à différentes missions. Les bénévoles, véritables piliers de l'association, sont impliqués dans diverses activités telles que l'accueil, l'accompagnement des personnes en difficulté, l'organisation d'événements solidaires et la collecte de ressources. Les locaux principaux sont situés au 14 rue des Veyettes, offrant un espace pour les activités administratives et opérationnelles. D'autres antennes, comme celles de Villejean et Maurepas, disposent également de leurs propres locaux pour accueillir les bénéficiaires et mener à bien les actions de solidarité.

### Public cible et typologie des bénéficiaires

Le SPF de Rennes s'adresse à un public diversifié, incluant des familles en situation de précarité, des personnes sans domicile fixe, des étudiants en difficulté financière, des personnes âgées isolées, ainsi que des migrants et réfugiés. L'association veille à

apporter une aide adaptée aux besoins spécifiques de chacun, sans discrimination, en proposant des services tels que l'aide alimentaire, l'accès aux soins, le soutien scolaire et l'accompagnement vers l'insertion professionnelle.

### Partenariats locaux

Pour renforcer son action, le SPF de Rennes collabore avec de nombreuses associations locales, institutions publiques et entreprises privées. Ces partenariats permettent de mutualiser les ressources, d'élargir le champ d'action et d'offrir une aide plus complète aux bénéficiaires. Parmi les collaborations notables, on compte des projets communs avec des centres sociaux, des établissements scolaires, des services municipaux et des acteurs du secteur sanitaire et social.

### Moyens de financement et ressources

Les ressources financières du SPF de Rennes proviennent principalement des dons de particuliers, des subventions publiques et des partenariats avec des entreprises. Des événements solidaires, tels que des collectes, des ventes solidaires ou des manifestations culturelles, sont régulièrement organisés pour récolter des fonds supplémentaires. La gestion rigoureuse des ressources et l'engagement bénévole permettent à l'association de maximiser l'impact de chaque contribution au profit des personnes aidées.

### Présentation des différents lieux d'accueil à Rennes

- **Antenne des Veyettes** : Située au 14 rue des Veyettes, cette antenne est le siège principal du SPF en Ille-et-Vilaine. Elle centralise les activités administratives et accueille les bénéficiaires pour diverses aides, notamment alimentaires et vestimentaires.
- **Antenne de Villejean** : Localisée au 3 rue d'Armagnac, cette antenne propose des services d'accueil, d'écoute et d'accompagnement pour les habitants du quartier. Les horaires d'ouverture sont le mercredi de 14h00 à 17h00 et le jeudi de 10h00 à 12h00.
- **Antenne de Maurepas** : Située au 32 rue de Trégain, cette antenne offre une gamme complète de services, incluant l'aide alimentaire, l'accès aux droits, le soutien scolaire et l'organisation d'activités culturelles. Elle est ouverte du lundi au jeudi, de 14h00 à 17h00.

- **Solidaribus** : Le Solidaribus est un dispositif mobile du SPF qui parcourt les rues de Rennes pour aller à la rencontre des personnes en situation de précarité, notamment celles sans domicile fixe. Il distribue des repas chauds, des vêtements et propose une écoute attentive, permettant de créer un lien social et d'orienter les bénéficiaires vers les structures adaptées à leurs besoins.

## Déroulement du stage et missions réalisées

Avant d'évoquer ce sujet, voici quelques notions importantes à savoir pour mieux comprendre l'environnement de travail :

Le Secours Populaire Français (SPF) utilise plusieurs logiciels et modules pour optimiser ses activités et services. Voici une présentation détaillée des principaux outils :

- **Hercule** : Ce module, développé localement, est conçu pour gérer les bénévoles lors de missions ponctuelles. Il propose des interfaces adaptées pour les bénévoles et les administrateurs, accessibles sur smartphones et ordinateurs (ReactJS).
- **Ulysse** : Également développé localement, ce module facilite la gestion des inscriptions aux événements liés aux vacances, permettant une organisation efficace des sorties et séjours (google sheet).
- **Dionysos** : Ce module, autre développement local, est dédié à la gestion des inscriptions pour les événements de loisirs, sportifs et culturels, contribuant ainsi à promouvoir l'accès aux activités culturelles et récréatives (google sheet).
- **Atrium - Pop'accueil** : Outil national, Atrium - Pop'accueil est une base de données centralisée qui conserve l'historique de toutes les personnes aidées et

des événements auxquels elles ont participé, assurant un suivi précis et cohérent des bénéficiaires à travers le territoire.

Ces outils sont essentiels pour le bon fonctionnement du SPF, améliorant la coordination des bénévoles, la gestion des événements et le suivi des bénéficiaires.

## A. Objectifs du stage, et présentation des actions réalisés.

Au cours de mon stage, j'ai été chargé de plusieurs missions techniques visant à améliorer et développer les fonctionnalités du module "Hercule" du Secours Populaire Français. Voici les principales tâches que j'ai réalisées, présentées dans leur ordre chronologique :

1. **Mise en place de l'environnement pour un nouveau développeur** : J'ai configuré le serveur Hercule pour pouvoir développer proprement sur mon ordinateur, en local.
2. **Refonte du système de création de PDF depuis une page HTML** : J'ai restructuré le système existant pour générer des fichiers PDF à partir de pages HTML, qui était défaillant auparavant.
3. **Création d'un nouvel environnement "Externe" dans Hercule** : J'ai développé un environnement spécifique au sein d'Hercule destiné à l'appel de l'API depuis des logiciels externes.
4. **Mise en place du système d'authentification via un token** : J'ai implémenté un mécanisme d'authentification basé sur des Tokens, permettant une sécurisation renforcée des échanges de données et une gestion simplifiée des sessions utilisateur.

5. **Récupération et affichage des données depuis la base de données Atrium via l'API d'Hercule** : J'ai intégré des fonctionnalités permettant de récupérer des informations stockées dans la base de données Atrium et de les afficher dans Hercule, en utilisant des appels API pour assurer une communication fluide entre les systèmes.
  6. **Modification des scripts Google pour la synchronisation avec Ulysse** : J'ai adapté les scripts existants afin de synchroniser les données entre Hercule et le logiciel Ulysse, en appelant l'API d'Hercule pour assurer une cohérence et une mise à jour en temps réel des informations.
  7. **Création de nouvelles tables "Dossier" et "Personne" dans la base de données d'Hercule** : J'ai étendu le schéma de la base de données en ajoutant des tables dédiées à la gestion des dossiers et des personnes, structurant ainsi davantage les données pour une meilleure organisation et accessibilité.
- 
8. **Mise en place d'un système de cache lors de la synchronisation avec Ulysse** : J'ai développé une fonction de mise à jour du cache qui, lors de la synchronisation, vérifie la présence d'un dossier dans la base de données d'Hercule. Si le dossier existe, il est mis à jour ; sinon, les informations sont récupérées depuis Atrium, insérées dans Hercule, puis retournées pour utilisation.
  9. **Création des tables "Événement" et "Participation" dans la base de données d'Hercule** : J'ai ajouté de nouvelles structures de données pour gérer les événements et les participations, facilitant ainsi le suivi des activités des personnes aidées.
  10. **Développement d'une page d'affichage des événements pour les administrateurs** : J'ai conçu une interface dédiée aux administrateurs, leur permettant de visualiser, gérer et organiser les différents événements directement depuis le module Hercule.
  11. **Implémentation d'une fonction de calcul des points de participation** : J'ai mis en place une fonctionnalité calculant le total des points accumulés par une personne en fonction de ses participations aux événements, offrant ainsi une vue d'ensemble de son engagement au sein de l'association.

## 12. Développement d'un système de retour de données pour les points dans l'environnement externe :

J'ai créé une fonctionnalité permettant aux utilisateurs externes d'obtenir des informations détaillées sur une personne, incluant son dossier, son nom et le total de ses points de participation, renforçant ainsi la transparence et la communication entre les parties prenantes.

## B. Description détaillée des missions :

### 1 : Mise en place de l'environnement.

#### Référence images : Annexe page ...

Avant de pouvoir développer sainement dans le module Hercule, j'ai dû configurer les fichiers nécessaires au lancement de l'application, comme le fichier .env par exemple :

Changement du mot de passe de la base de données

Ajout du mode « ER » pour Enzo Reine, car chaque développeur a son propre mode pour lancer l'application.

Commande pour lancer Hercule :

« npm run build:devER » pour la partie web

« npm run start:devER » pour la partie API, qui va lancer le serveur sur l'URL mise dans le .env

Dans mon cas, il y avait un bug avec la base de données, c'était automatiquement redirigé vers MySQL lorsque je faisais une requête, hors la base de données Hercule est en MariaDB, j'ai donc rajouté une ligne de commande qui va effectuer la requête sur le

premier port qu'elle trouve correspondant (normalement 3307) et si il ne fonctionne pas, alors le faire sur 3306 (qui est celui de MariaDB)

Après avoir fais proprement ces modifications, j'ai enfin pus commencer à développer dans l'application sans soucis, ou plus ou moins...

## 2 : Refonte du système de création de PDF depuis une page HTML

Directement après avoir configuré mon environnement, je n'arrivais pas à lancer le serveur à cause de problèmes lié à la création de PDF depuis une page HTML.

```
859 | /**
860 | * @license
861 |
862 | at failureErrorWithLog (C:\Users\enzor\SPF\ORGAMI2\webAdminTs\node_modules\vite\node_modules\esbuild\lib\main.js:1476:15)
863 | at C:\Users\enzor\SPF\ORGAMI2\webAdminTs\node_modules\vite\node_modules\esbuild\lib\main.js:755:50
864 | at responseCallbacks.computed (C:\Users\enzor\SPF\ORGAMI2\webAdminTs\node_modules\vite\node_modules\esbuild\lib\main.js:622:9)
865 | at handleIncomingPacket (C:\Users\enzor\SPF\ORGAMI2\webAdminTs\node_modules\vite\node_modules\esbuild\lib\main.js:677:12)
866 | at Socket.readFromStdout (C:\Users\enzor\SPF\ORGAMI2\webAdminTs\node_modules\vite\node_modules\esbuild\lib\main.js:600:7)
867 | at Socket.emit (node:events:519:28)
868 | at addChunk (node:internal/streams/readable:559:12)
869 | at readableAddChunkPushByteMode (node:internal/streams/readable:510:3)
870 | at Readable.push (node:internal/streams/readable:390:5)
871 | at Pipe.onStreamRead (node:internal/stream_base_commons:191:23)
872 |
873 PS C:\Users\enzor\SPF\ORGAMI2\webAdminTs> npm run build:devER
```

(On ne voit pas que c'est à cause du système de PDF sur le screen ci-dessus, mais c'est bien le cas, je n'ai plus les preuves de ce problème malheureusement)

Il m'était impossible de lancer le serveur proprement à cause de ce problème, j'ai été donc confié comme mission de refaire le système de PDF, de la même façon mais avec d'autres bibliothèques.

J'ai commencé par supprimer la bibliothèque en question, et tous les imports la nécessitant pour éviter des erreurs d'import.

Après avoir tout supprimé, j'avais enfin accès au site en local, et j'ai pu commencer à recréer tout le système de création de PDF.

*Voir l'annexe pour les images*

### **3 : Création d'un nouvel environnement "Externe" dans Hercule**

Auparavant, il existait 3 majeures parties dans Hercule pour l'affichage web :

- WebAdmin : pour les administrateurs
- WebClient : pour ceux qui n'ont pas les permissions administrateur
- WebPublic : informations publique libre d'accès

J'ai été confié comme mission, après la réparation du système de création de PDF, de créer un autre environnement comme ceux cité au-dessus, mais avec une utilité autre que celles déjà présente.

En effet, ce nouvel environnement servira à être appelé d'un site/logiciel externe, pour pouvoir interagir et communiquer plus rapidement qu'auparavant.

J'ai donc commencé à recréer tout le système fonctionnel déjà présent dans webPublic, que j'ai modifié pour pouvoir répondre aux besoins de mon tuteur de stage.

Après avoir tout configuré, le nouvel environnement était bien fonctionnel, il n'avait aucune utilité pour le moment car aucune fonction n'était faite pour le moment.

La base de l'URL de webExterne est : /externe

*Voir l'annexe pour les images*

#### **4 : Mise en place du système d'authentification via un token**

Par mesure de sécurité, un système d'authentification est mis en place sur le serveur pour se protéger des menaces externes. J'ai donc dû apprendre le fonctionnement de ce système pour pouvoir le recréer sur ce nouvel environnement externe.

J'ai dû comprendre le fonctionnement de ce système d'authentification sur les autres environnements pour pouvoir le refaire proprement, et faire quelques modifications pour pouvoir l'adapter aux besoins de mon tuteur.

Après avoir fait proprement ce qui m'étais demandé, un utilisateur lambda ne pouvait pas atterrir sur l'URL avec la base /externe. Alors la page d'authentification apparait lorsque le token n'est pas présent sur sa session.

Lorsque l'utilisateur est connecté, on peut voir le token dans la partie « network » lorsqu'on inspecte l'élément.

*Voir l'annexe pour les images*

#### **5 : Récupération et affichage des données depuis la base de données Atrium via l'API d'Hercule**

Maintenant que le site est un minimum sécurisé, on peut commencer à aller chercher des données.

Mon objectif à présent était de récupérer les données d'un dossier (34160) dans la base de données Atrium.

Il y avait déjà des scripts Google App Script dans l'application Ulysse, je devais donc reporter tout ce code dans Hercule, le transformer, et l'adapter à la situation.

J'ai donc récupéré toute la fonction présente dans Ulysse pour récupérer la data dans l'Atrium, grâce à un token.

J'ai recréé toutes les fonctions nécessaires dans la partie API d'Hercule, car c'est ici qu'elles seront utilisées à l'avenir, que ce soit par Hercule lui-même, ou d'un logiciel externe (Ulysse).

Après avoir refait proprement toutes les fonctions permettant la récupération de la data, j'ai commencé à créer une nouvelle page web pour voir si les fonctions étaient correcte, et me permettais de récupérer la data dans Atrium correctement.

J'ai donc créer la route /externe/ulysses, qui, lorsqu'on arrive dessus, affichera automatiquement le dossier que l'on veut (mit en dure dans le code) et toute sa data. En sachant que dans le dossier il y a aussi toute la famille, et les informations des personnes de la famille. (Exemple : si un dossier contient 4 personnes, cela signifie que c'est une famille de 4 personnes, et qu'il y a la data de ces personnes dans le dossier).

*Voir l'annexe pour les images*

## **6 : Modification des scripts Google pour la synchronisation avec Ulysse**

Changement d'application, je passe maintenant sur Ulysse et découvre l'univers des Google Sheets et du langage Google App Script (très similaire à JavaScript).

Ma tache était maintenant de faire fonctionner le système d'API d'Hercule et d'appeler les fonctions depuis une application externe. J'ai donc modifié le code des scripts Google dans un environnement test, pour pouvoir tenter de récupérer la data de la base de données Hercule en appelant l'URL d'une des routes de l'API d'Hercule. Je devais récupérer le nom d'une mission et son libellé, juste pour tester si c'était possible et que cela fonctionnait bien.

Je devais donc en parallèle refaire un système de connexion avec un identifiant et mot de passe, mais aussi prendre le token présent dans l'url s'il y en a un, et l'utiliser comme authentification. S'il n'y en avait pas, alors on détectait le mail connecté dans la session du serveur Ulysse, et on tentait de se connecter lors de la requête avec les

nouveaux identifiants. J'ai au passage rajouté et modifié quelques fonctions d'Hercule pour pouvoir prendre en compte les mails lors de la connexion. J'ai aussi dû rajouter le champ « mail google » dans la base de données Hercule pour les utilisateurs.

Après que l'authentification est fonctionnée, en appuyant sur les boutons de l'interface Google Sheets, une suite d'évènements instinctive se suivent, pour enfin afficher la data que l'on voulait afficher.

*Voir l'annexe pour les images*

## **7 : Création de nouvelles tables "Dossier" et "Personne" dans la base de données d'Hercule**

Maintenant que l'on sait qu'il est possible de récupérer la data de la base de données d'Hercule grâce à l'API, on peut maintenant continué l'objectif en créant les tables Dossier et Personne, qui seront utile à l'insertion des données d'atrium.

L'objectif principale est de faire passé Hercule pour un « cache ». Car en effet, la base de données Atrium est relativement longue, ce qui rend la gestion des données assez longue dans le logiciel Hercule.

Une simple requête SQL m'a permis de créer ces tables. J'ai quand même longuement réfléchi et étudier les autres tables pour pouvoir créer ces dernières, car il y avait beaucoup de paramètre à prendre en compte, qui dépendait des besoins de mon tuteur de stage.

*Voir l'annexe pour les images*

## **8 : Mise en place d'un système de cache lors de la synchronisation avec Ulysse**

Comme dis précédemment, l'objectif est de faire un système de cache grâce à l'API d'Hercule. Pour cela, j'ai fait tout un système pour que tout se fasse proprement et logiquement il m'a aussi été demandé de faire en sorte que les données soient mises à jour en même temps que la synchronisation se fait.

Voici ce que la boucle de la fonction principale fait :

---

La synchronisation se fait d'Atrium vers le cache.

Les données reçues d'Atrium doivent mettre à jour le cache. Le flux est unidirectionnel : Atrium → Cache

Effectuer la recherche uniquement sur les personnes ayant un flag false

1 / Recherche par DATENAISANCE : Si une date de naissance correspond, mettre à jour automatiquement la personne concernée et définir son flag à true

2 / Recherche par DATENAISANCE : Si deux dates de naissance correspondent mais sans correspondance de NOMPRENOM, définir le flag à false

3 / Recherche par DATENAISANCE puis NOMPRENOM : Si deux DATENAISANCE correspondent et un NOMPRENOM correspond à une personne, mettre à jour cette personne et définir son flag à true.

L'autre personne est mise à jour selon le cas 1

4 / Recherche par NOMPRENOM : Si le NOMPRENOM correspond mais pas la date de naissance, mettre à jour la personne et définir son flag à true

5 / Si ni la DATENAISANCE ni le NOMPRENOM ne correspondent : insérer dans la base de données (POST) et définir le flag à true

---

Si après la boucle, le nombre de personnes dans le cache est supérieur au nombre de personnes dans le dossier Atrium concerné, supprimer toutes les personnes ayant un flag false

La boucle peut se faire plusieurs fois, car il y a des cas exceptionnels, comme par exemple des jumeaux, ou des erreurs dans la base de données. Le système anticipe au maximum les problèmes.

Pour tester si le système de mise à jour du cache fonctionne, j'ai du créer une autre page web pour pouvoir afficher et simuler la requête depuis Ulysse.

J'avais d'abord fait un système où lorsqu'on rentrait un numéro de dossier (exemple 34160), l'application allait regarder si le dossier était présent dans Hercule, si oui, alors il faisait ce qu'il devait faire (vu au-dessus), sinon, il allait le chercher dans Atrium pour l'insérer dans la base de donnée Hercule.

Par la suite, mon tuteur de stage a fait des modifications, pour pouvoir faire en sorte que l'on puisse choisir entre faire la fonction de recherche automatique (soit Hercule soit Atrium), mais également une recherche forcée pour les données Atrium, ou l'on va directement faire la recherche sur la base de données Atrium.

*Voir l'annexe pour les images*

## **9 : Création des tables "Événement" et "Participation" dans la base de données d'Hercule**

Pour ma tâche suivante, j'ai créé les tables **Événement** et **Participation** pour pouvoir anticiper les tâches qui me seront donnée par la suite. J'ai réutilisé le même schéma.

La table événement servira à stocker les événements faits par le Secours Populaire, ainsi que la valeur en point de cette événement (utile pour leur système de fonctionnement), et la table Participation sera la participation des Personnes sur ces événements. Les personnes n'ont pas le droit d'avoir une infinité d'événements car ils sont limités selon les points qu'ils ont.

*Voir l'annexe pour les images*

## **10 : Développement d'une page d'affichage des événements pour les administrateurs**

Après avoir créer les tables, j'ai commencé à créer les fonctions permettant d'interagir avec la base de données. Je pouvais donc ajouter, supprimer, modifier ou lire des évènements (CRUD). J'ai dû ajouter une nouvelle route pour accéder aux événements, ainsi que reproduire le système d'ajout des événements.

Lorsqu'on arrive sur la page des événements, on a la possibilité de les trier soit par type d'évènement, soit par année, mais aussi d'effectuer une recherche grâce à la barre de recherche. On peut ensuite, en appuyant sur les icônes à droite, soit voir l'événement en question, soit modifié cet événement. Lorsqu'on clique sur voir l'événement, on a à gauche un formulaire qu'on ne peut pas modifier, avec les informations de l'événement, et à droite, un tableau avec cet événement, qui sera par la suite remplacé par la liste des participations présente pour cet événement.

Lorsqu'on clique sur modifier l'événement dans la partie gauche, il est maintenant possible de modifier le formulaire, et après que l'on est changé ce qu'on voulait, on peut valider en appuyant sur le bouton vert, ou annuler le choix. On a aussi la possibilité de supprimer l'événement en appuyant sur le bouton rouge.

Si on veut ajouter un événement, il suffit d'appuyer sur le bouton du milieu dans la page des événements, on arrivera sur une URL finissant par « -1 », car en effet, le système est fait de cette sorte : si l'url contient un nombre en paramètre, alors cela veut dire qu'il y a un événement, et alors il faut afficher cet événement. Or, s'il est équivalent à -1, alors cela signifie qu'il faut créer un événement, alors on est redirigé vers le même formulaire que pour l'update, mais vide, et qui aura comme fonction d'ajouter un événement dans la base de donnée Hercule.

J'ai dû rajouter « événement » dans la barre du menu, pour pouvoir y accéder naturellement et instinctivement.

Pour pouvoir refaire toute cette page, je me suis adapté aux autres pages déjà présente et j'ai dû recopier le style de ces dernières.

*Voir l'annexe pour les images*

## **11 : Implémentation d'une fonction de calcul des points de participation**

Par la suite, j'ai ajouté une fonction permettant de faire le total des points, qui sera utile par la suite. Tout cette base sur une requête SQL qui va prendre le nom, le dossier des personnes en questions, regardé ou est ce qu'il y a une participation avec l'id (CODE) de la personne, et faire le total des points (valeur).

*Voir l'annexe pour les images*

## **12 : Développement d'un système de retour de données pour les points dans l'environnement externe**

Après avoir correctement crée la fonction pour calculer le total des points, j'ai pu l'implémenter pour l'utiliser dans la partie WebExterne d'Hercule. En effet, j'ai dû

recréer une autre page et simuler une requête venant d'Ulysse, pour pouvoir à la fin afficher des données qu'Ulysse pourrait interpréter (comme pour la mise à jour du cache avec les dossiers et les personnes). Le retour est affiché comme ceci :

- Numéro du dossier
- Nom et prénom de la personne
- Total des points de la personne

On peut envoyer plusieurs personnes en même temps, et afficher les points de chaque personne. Les données sont envoyées dans un tableau (comme pour la mise à jour du cache) pour pouvoir traitées plusieurs personnes en même temps.

*Voir l'annexe pour les images*

## Technologies

### Technologies utilisées

- **HTML/CSS** : Langages de base pour la structuration et la mise en forme des pages web.
- **JavaScript/TypeScript** : Langages de programmation pour le développement de fonctionnalités dynamiques et typées sur le web.
- **ReactJS** : Framework JavaScript pour la création d'interfaces utilisateur interactives et modulaires.
- **API REST** : Interface permettant l'échange de données au format JSON entre le client et le serveur.

- **SQL** : Langage de requête pour la gestion et l'interrogation des bases de données relationnelles.
- **GIT** : Système de contrôle de version pour le suivi et la gestion des modifications du code source.

## Technologies apprises

- **ReactJS** : Approfondissement des connaissances et des pratiques liées à ce framework.
- **TypeScript** : Apprentissage de ce sur-ensemble de JavaScript offrant un typage statique pour une meilleure robustesse du code.

## Ressenti du stage

Mon stage au sein du Secours Populaire Français de Rennes a été une expérience particulièrement enrichissante, tant sur le plan technique que sur le plan humain.

## Aspects positifs

L'un des principaux atouts de ce stage a été son caractère très instructif. J'ai eu l'occasion d'approfondir mes connaissances en développement web, en manipulant des technologies modernes comme React.js et TypeScript, tout en mettant en application des concepts avancés de gestion de bases de données et d'API REST. Chaque tâche confiée était concrète et directement intégrée à un projet en cours, ce qui m'a permis de comprendre l'impact réel de mon travail au sein de l'organisation.

L'environnement de travail était également un point fort. L'ambiance générale au sein de l'équipe était agréable, marquée par une bienveillance et une ouverture d'esprit.



appréciables. J'ai pu échanger facilement avec les membres du personnel et les bénévoles, qui ont toujours été disponibles pour répondre à mes questions et m'accompagner dans mes missions. Cette atmosphère conviviale et collaborative a grandement facilité mon intégration et m'a permis de travailler avec motivation et implication.

Enfin, ce stage m'a permis d'être confronté à une dimension plus humaine du travail en entreprise. Travailler au sein d'une association de solidarité m'a sensibilisé aux réalités du terrain et à l'importance de l'engagement social. Au-delà de mes missions techniques, j'ai pris conscience de l'impact concret que peuvent avoir les outils numériques sur l'amélioration de l'organisation et du fonctionnement des associations d'aide aux personnes en difficulté.

## Aspects négatifs

Le principal inconvénient rencontré durant ce stage concernait les conditions matérielles de travail, notamment la température des locaux. En raison d'un manque de chauffage, il faisait souvent très froid, au point que je devais coder avec mon manteau toute la journée. Bien que cela n'ait pas affecté la qualité de mon travail, cela représentait un certain inconfort au quotidien.

Malgré ce léger désagrément, mon expérience au sein du Secours Populaire Français de Rennes a été globalement très positive. Ce stage m'a non seulement permis de développer des compétences techniques solides, mais il m'a également offert une ouverture sur le monde associatif et ses enjeux, renforçant ainsi ma motivation à poursuivre dans le domaine du développement web, avec une sensibilité accrue aux problématiques sociales.

## Apprentissages du monde professionnel grâce au Secours Populaire Français

Mon stage au sein du Secours Populaire Français m'a permis de mieux comprendre le fonctionnement du monde professionnel, notamment en termes d'engagement et de méthodologie de travail.

## Engagement et investissement des salariés

L'un des aspects qui m'a particulièrement marqué est l'implication des salariés dans leur travail. Contrairement à certaines idées reçues sur le monde professionnel, j'ai constaté que les employés du Secours Populaire ne se contentent pas d'exécuter leurs tâches dans un cadre strictement défini par leurs horaires. Beaucoup d'entre eux consacrent du temps supplémentaire à leurs missions, allant jusqu'à poursuivre certaines tâches chez eux, y compris le week-end. Cette implication dépasse la simple obligation professionnelle : elle est portée par une adhésion profonde aux valeurs de l'association et à sa mission sociale.

Cette culture de l'engagement m'a fait prendre conscience de l'importance de travailler dans un environnement où les valeurs et les objectifs de l'organisation sont partagés par ses membres. Voir des professionnels aussi investis m'a inspiré et m'a conforté dans l'idée que la motivation et la passion jouent un rôle clé dans l'épanouissement au travail.

## Apprentissages techniques et adaptation au contexte

D'un point de vue technique, ce stage m'a permis d'améliorer ma capacité à lire et comprendre du code écrit par d'autres développeurs. J'ai gagné en autonomie dans l'analyse d'une base de code existante et dans la compréhension de l'architecture d'un logiciel ou d'une application. Cette compétence est essentielle dans le domaine du développement, où il est fréquent d'intégrer des projets en cours et de collaborer avec d'autres programmeurs.

J'ai également appris à m'adapter à un environnement technique différent de celui étudié en formation. Par exemple, certaines conventions de nommage dans la base de données étaient spécifiques au projet : les identifiants étaient désignés par "CODE" au lieu de l'habituel "Id" pour toutes les tables, et les noms des éléments étaient indiqués sous "DESIGNATION" plutôt que "nom". De plus, toutes les tables et les champs étaient écrits en majuscules, ce qui tranchait avec les pratiques rencontrées jusque-là. Cette expérience m'a appris à ne pas me limiter à un cadre strictement académique et à m'adapter aux standards propres à chaque organisation ou projet.

En somme, ce stage m'a offert un véritable aperçu du monde professionnel, à la fois en termes d'engagement humain et d'exigences techniques. J'ai pris conscience de l'importance de la rigueur, de l'adaptabilité et de l'investissement personnel dans la réussite d'un projet, des éléments qui me seront précieux dans la suite de mon parcours professionnel.

## Ouverture sur la suite et projet professionnel

Ce stage au sein du Secours Populaire Français a été une expérience déterminante pour mon orientation professionnelle. Il m'a permis d'approfondir mes connaissances dans des domaines techniques clés, tout en confirmant mes préférences et mes aspirations pour l'avenir.

L'un des aspects les plus enrichissants de ce stage a été la découverte et la manipulation avancée des API. Par rapport à mes expériences précédentes, j'ai eu l'opportunité d'interagir de manière bien plus approfondie avec ces interfaces, en travaillant sur des requêtes complexes et des échanges de données entre différents systèmes. Cette immersion dans l'univers des API REST m'a permis de mieux comprendre leur rôle fondamental dans le développement web moderne et l'importance de leur bonne conception pour garantir des applications performantes et évolutives.

Un autre élément clé de mon apprentissage a été la polyvalence entre le back-end et le front-end. Contrairement à mes précédents projets, où ces deux aspects étaient souvent dissociés, j'ai ici eu l'opportunité d'intervenir sur les deux parties

simultanément. Cette approche m'a offert une vision globale du cycle de développement d'une application et m'a aidé à affiner mes choix professionnels. J'ai particulièrement apprécié cette complémentarité entre la logique métier côté serveur et l'expérience utilisateur côté client, ce qui m'oriente désormais vers des perspectives de carrière en développement full-stack.

En somme, ce stage a été un véritable tremplin pour mon avenir. Il m'a permis de consolider mes compétences, de mieux cerner mes préférences techniques et de m'orienter vers des choix professionnels en adéquation avec mes intérêts et mes ambitions.

## Missions externes au stage

Durant mon stage au sein du Secours Populaire Français, j'ai eu l'opportunité de participer activement à diverses missions bénévoles, enrichissant ainsi mon expérience et ma compréhension du milieu associatif.

- **Distributions alimentaires** : J'ai contribué à la préparation et à la distribution de denrées alimentaires aux personnes en situation de précarité. Cette mission m'a permis de comprendre les enjeux logistiques liés à l'aide alimentaire et d'apprécier l'importance d'une organisation rigoureuse pour répondre efficacement aux besoins des bénéficiaires.
- **Tri des dons, notamment des jouets** : J'ai participé au tri et à la sélection des jouets offerts par les donateurs. Les articles en bon état étaient destinés à la boutique solidaire du SPF, où ils sont proposés à des prix symboliques, permettant ainsi aux familles défavorisées d'accéder à des biens de qualité à moindre coût. Cette tâche m'a sensibilisé à l'importance du recyclage et de la valorisation des dons pour soutenir les actions de solidarité.

- **Journée avec le Solidaribus :** J'ai consacré une journée au sein du Solidaribus, l'antenne mobile du SPF conçue pour aller à la rencontre des personnes isolées ou ayant des difficultés de mobilité. Ce véhicule itinérant permet de proposer des services variés tels que l'aide alimentaire, l'accès aux droits, et des activités culturelles, directement au cœur des zones rurales ou des quartiers éloignés des centres d'aide traditionnels. Cette expérience m'a montré l'importance de l'adaptabilité et de la proximité dans les actions solidaires pour toucher un public plus large et souvent marginalisé.

Ces missions externes m'ont offert une vision concrète des différentes facettes de l'engagement bénévole au sein du SPF et ont renforcé mon désir de contribuer activement à des actions solidaires.

## Conclusion

Ce stage au sein du Secours Populaire Français a été une expérience extrêmement enrichissante, tant sur le plan technique que humain. En intégrant une structure engagée dans la solidarité, j'ai pu mettre mes compétences en développement web au service d'un projet ayant un impact concret sur la gestion des bénévoles et des bénéficiaires.

D'un point de vue technique, j'ai eu l'opportunité de travailler sur des missions variées, allant de l'amélioration des fonctionnalités existantes à l'optimisation des performances et de la gestion des données. La refonte du système de génération de fichiers PDF, la mise en place d'un système de cache, l'amélioration de l'authentification par token, ainsi que l'interaction avec des bases de données et API ont été des défis stimulants qui ont renforcé mes compétences en développement full-stack. Cette immersion m'a également permis d'approfondir mes connaissances en ReactJS, TypeScript et en manipulation de bases de données SQL, tout en améliorant ma capacité à lire et comprendre une architecture logicielle complexe.



Au-delà des aspects techniques, cette expérience m'a offert un regard plus large sur le monde professionnel. J'ai pu constater l'investissement et l'engagement des salariés et bénévoles du SPF, qui ne comptent pas leurs heures et s'impliquent pleinement dans leur mission. Cette immersion dans un environnement où l'humain est au cœur des préoccupations m'a permis d'évoluer dans un cadre bienveillant et motivant, où la collaboration et la solidarité sont essentielles.

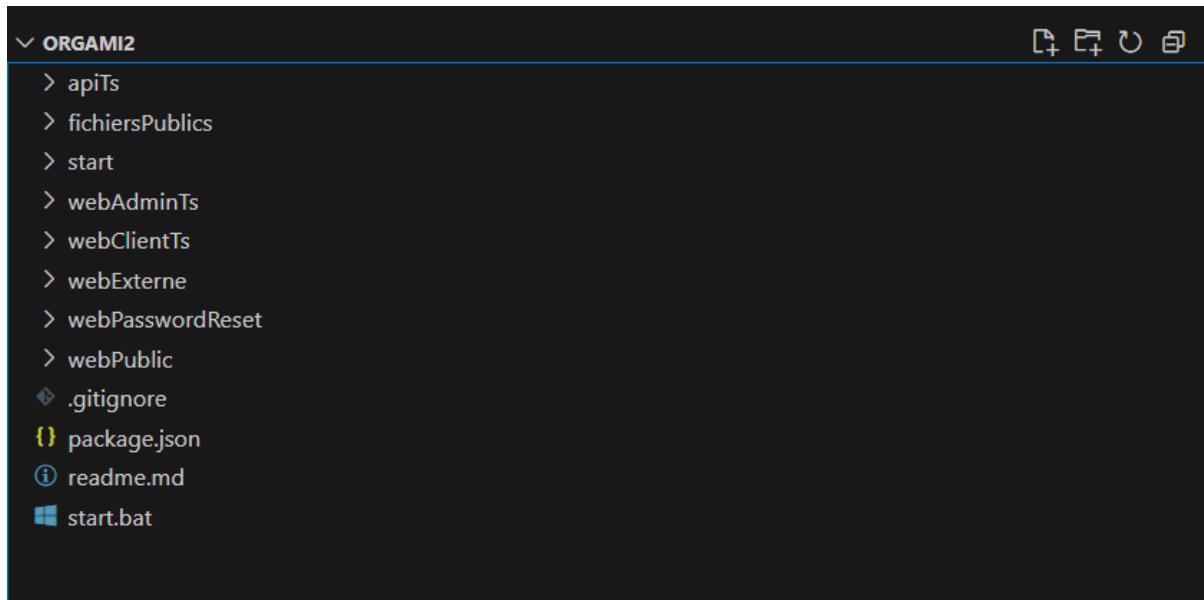
Ce stage a également eu un impact significatif sur mon orientation professionnelle. En travaillant à la fois sur le back-end et le front-end, j'ai pu affiner mes préférences et confirmer mon intérêt pour le développement full-stack. L'aspect technique lié aux API, à la gestion des bases de données et à l'optimisation des performances applicatives m'a particulièrement plu et me motive à poursuivre dans cette voie.

En conclusion, ce stage a été une expérience formatrice et déterminante. Il m'a permis d'acquérir de nouvelles compétences, de relever des défis techniques concrets et de mieux comprendre le fonctionnement d'une structure associative. Il a également renforcé ma motivation à poursuivre mon parcours dans le développement web, en m'orientant vers des projets à fort impact social et humain.

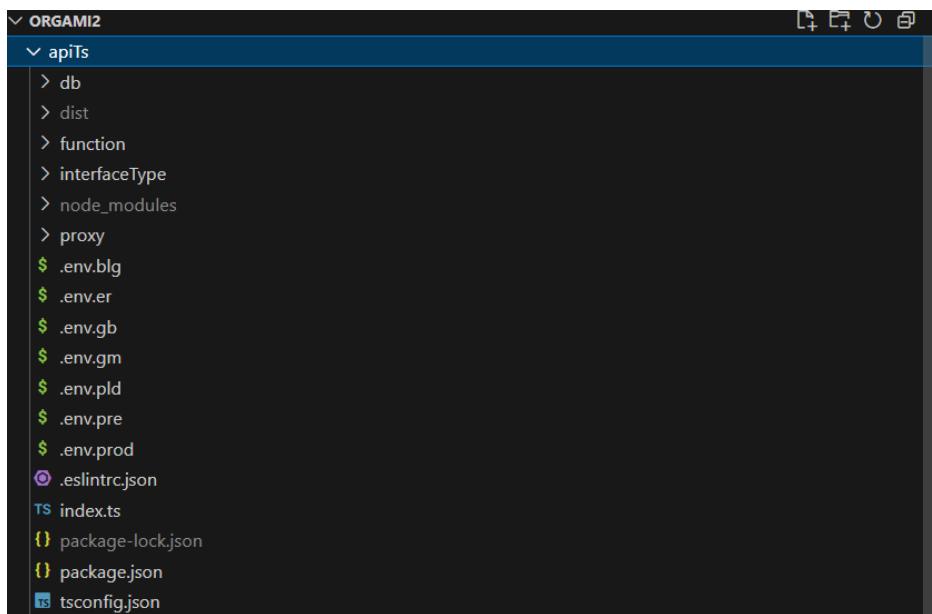
## Annexes

Note : la data manipulée dans les screens suivants sera caché pour des raisons de confidentialité.

Structure globale d'Hercule :



ApiTs :



WebAdmin :

```
✓ webAdminTs
  > dist
  > node_modules
  > public
  > src
  ⚙ .env
  Ⓜ .eslintrc.cjs
  ⛅ index.html
  ⚡ package-lock.json
  ⚡ package.json
  ⓘ README.md
  ⚡ tsconfig.json
  ⚡ tsconfig.node.json
  ⚡ vite.config.ts
```

WebExterne, WebPublic et WebClients sont similaire à webAdmin.

### 1 : Mise en place de l'environnement :

```
7
8 # mdp SQL
9+PASSWORD_SQL=kirito50250
10 HOST=127.0.0.1
```

```
7
8 # mdp SQL
9+PASSWORD_SQL=
10 HOST=127.0.0.1
```

```
VITE_API_URL_GM=http://localhost:8100/
VITE_API_URL_ER=http://localhost:8100/
```

```
"type": "module",
"scripts": {
  "devBlg": "vite --host --port 80 --open --mode blg",
  "devER": "vite --host --port 8100 --open --mode er",
```

```
"dependencies": {
    "dotenv": "^16.4.7",
    "esbuild": "^0.24.2",
    "html2canvas": "^1.4.1",
    "jspdf": "^2.5.2",
    "latest": "^0.2.0",
    "prop-types": "^15.8.1",
    "react": "^19.0.0",
    "react-dom": "^19.0.0",
    "react-router-dom": "^7.1.1"
},
"devDependencies": {
    "@types/dotenv": "^8.2.3",
    "@types/react": "^19.0.3",
    "@types/react-dom": "^19.0.2",
    "@typescript-eslint/eslint-plugin": "^8.19.0",
    "@typescript-eslint/parser": "^8.19.0",
    "@vitejs/plugin-react-swc": "^3.7.2",
    "eslint": "^9.17.0",
    "eslint-plugin-react-hooks": "^5.1.0",
    "eslint-plugin-react-refresh": "^0.4.16",
    "typescript": "^5.7.2",
    "vite": "^6.0.7"
}
```

```
-PORT_DB=3307
```

```
user: process.env.USER_SQL,
password: process.env.PASSWORD_SQL,
database: process.env.DATABASE,
port: process.env.PORT_DB ? parseInt(process.env.PORT_DB, 10) : 3306,
})
```

## 2 : Refonte du système de création de PDF depuis une page HTML

```
"esbuild": "^0.24.2",
"html2canvas": "^1.4.1",
"jspdf": "^2.5.2",
"latest": "^0.2.0",
"prop-types": "^15.8.1",
"react": "^19.0.0",
```

```
31 "html2canvas": "^1.4.1",
32 "latest": "^0.2.0",
33+ "pdf-lib": "^1.17.1",
34 "prop-types": "^15.8.1",
35 "react": "^19.0.0",
```

```
{ useState, useEffect } from 'react';
{ getFetch } from "../../functions/request";
{ formatDateCourte } from "../../functions/bao
{ isPlanningRapportArray, PlanningRapport } from
jsPDF from 'jspdf';
html2canvas from 'html2canvas';
{ getHoraireFormat } from "../../functions/bao
```

```
1+import React, { useState, useEffect } from 'reac
2 import { getFetch } from "../../functions/re
3 import { formatDateCourte } from "../../functi
4+import { PlanningRapport } from "../../functi
5+import { PDFDocument, rgb, StandardFonts } fr
6 import html2canvas from 'html2canvas';
7 import { getHoraireFormat } from "../../functi
```

```
// Fonction d'exportation du tableau en PDF
const exportPDF = () => {
  const input = document.getElementById('planning-tableau');

  html2canvas(input).then((canvas) => {
    const imgData = canvas.toDataURL('image/png');

    const pdf = new jsPDF('p', 'mm', 'a4');

    const imgWidth = 210; // Largeur en mm pour A4

    const pageHeight = 295; // Hauteur en mm pour A4

    const imgHeight = (canvas.height * imgWidth) / canvas.width;
    let heightLeft = imgHeight;
    let position = 0;

    pdf.addImage(imgData, 'PNG', 0, position, imgWidth, imgHeight);
    heightLeft -= pageHeight;

    while (heightLeft >= 0) {
      position = heightLeft - imgHeight;
      pdf.addPage();
      pdf.addImage(imgData, 'PNG', 0, position, imgWidth, imgHeight);
      heightLeft -= pageHeight;
    }
  });
}
```

```
31 // Fonction d'exportation du tableau en PDF
32+ const exportPDF = async () => {
33  const input = document.getElementById('planning-tableau');

34  // Capture le tableau en image
35+ const canvas = await html2canvas(input);
36  const imgData = canvas.toDataURL('image/png');

37+ // Créer un document PDF avec pdf-lib
38+ const pdfDoc = await PDFDocument.create();
39+ const page = pdfDoc.addPage([595.28, 841.89]); // A4
40+ const { height } = page.getSize();

41+ // Charger la police par défaut
42+ const font = await pdfDoc.embedFont(StandardFonts.Helvetica);

43+ // Dimensions de l'image capturée
44+ const imgWidth = 595.28; // Largeur du PDF (en point)
45+ const imgHeight = (canvas.height * imgWidth) / canvas.width;
46+ let yPosition = height; // Position initiale (en hauteur)

47+ // Ajouter l'image au PDF
48+ page.drawImage(await pdfDoc.embedPng(imgData), {
49+   x: 0,
50+   y: yPosition - imgHeight,
51+   width: imgWidth,
52+   height: imgHeight,
53+ });
54+ // Sauvegarde le PDF
55+ const pdfBytes = await pdfDoc.save();
56+ const blob = new Blob([pdfBytes], { type: 'application/pdf' });

57+ // Télécharger le PDF
58+ const link = document.createElement('a');
59+ link.href = URL.createObjectURL(blob);
60+ link.download = 'rapport.pdf';
61+ link.click();
```

 **HERCULE**  
Hello Gwenaël !

[Menu >](#)

[◀ Retour](#) [Export en PDF](#)

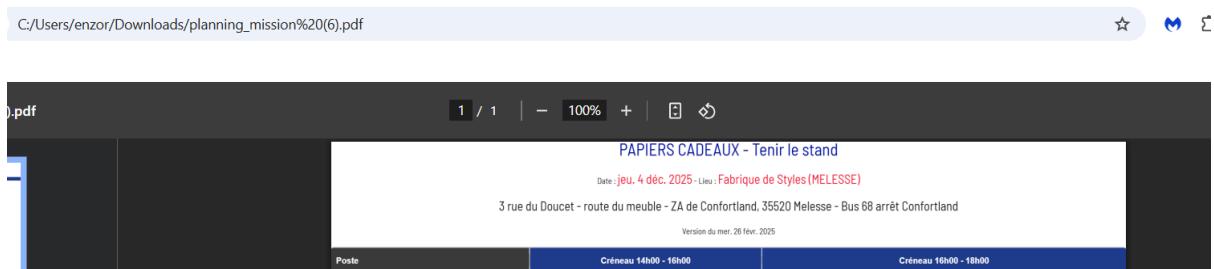
**PAPIERS CADEAUX - Tenir le stand**

Date : **jeu. 4 déc. 2025** - Lieu : **Fabrique de Styles (MELESSE)**

3 rue du Doucet - route du meuble - ZA de Confortland, 35520 Melesse - Bus 68 arrêt Confortland

Version du mer. 26 fevr. 2025

Poste	Créneau 14h00 - 16h00	Créneau 16h00 - 18h00
-------	-----------------------	-----------------------



### 3 : Création d'un nouvel environnement "Externe" dans Hercule

```
main.tsx webExterne\src A
1+ import React from 'react';
2+ import ReactDOM from 'react-dom/client';
3+ import App from './App';
4+ import { BrowserRouter } from "react-router-dom";
5+
6+ ReactDOM.createRoot(document.getElementById('root')!).render(
7+   <React.StrictMode>
8+     <BrowserRouter basename="/externe">
9+       <App />
10+      </BrowserRouter>
11+    </React.StrictMode>
12+
13+
```

```
❖ TestPage.tsx webExterne\src\pages A
1
2
3+import FirstLayer from "../components/pages/TestPage/First_layers/First_layers"
4+
5+function TestPage() {
6+    return (
7+        <div className='PageTestPage'>
8+            <FirstLayer />
9+        </div>
10+
11+export default TestPage;
12+
```

```
body {
  background: #e6f78893;
}
```

Partie API :

```
// api externe
app.use(validatePath(/^(\api\externe\||\externe\api\externe\|)/, (req, res, next) => {
  return apiExterne()(req, res, next);
}))
```

```
// distribution apiExterne
app.use(validatePath(/^\api\externe\|/, (req, res, next) => apiExterne()(req, res, next)));
```

```
+import { Request, Response, NextFunction } from 'express';
+import { pool } from "../../";
+import { returnErreurs, Erreurs, returnSuccesses } from "../../function/return";
+import { Poste } from "../../interfaceType/interfaces/tPostes";
+
+export async function getTest(req: Request, res: Response, next: NextFunction) {
+
+    // crée une requête SQL qui récupère les postes grâce au code
+    const query = `SELECT * FROM TPOSTES`;
+
+    // Effectuer la recherche avec une seule requête
+    const [results] = await pool.execute(query, []);
+
+    // vérifie si la réponse est un Array
+    if (!Array.isArray(results)) return returnErreurs(res, next, Erreurs.notArray);
+
+    // récupère le / les planning(s) du poste
+    const postes: Poste[] | undefined = results as Poste[];
+
+    return returnSuccesses(res, next, postes);
+
+}
```

```
    } else if (req.url.startsWith('/public')) {
        res.type('html');
        res.sendFile(path.join(__dirname, `${accessWebfile}webPublic/dist/index.html`));
    } else if (req.url.startsWith('/externe')) {
        res.type('html');
        res.sendFile(path.join(__dirname, `${accessWebfile}webExterne/dist/index.html`));
    } else {
        // envoie le fichier html
    }
}
} catch {
    try {
        readFileSync(path.join(__dirname, `${accessWebfile}webExterne/dist`, req.url));
        static_(path.join(__dirname, `${accessWebfile}webExterne/dist`))(req, res, next);
    } catch {
        console.log(req.url)
    }
}
```

```
import express from "express";
import { getTest } from "../../child/public/test";

export const apiExterne = () => {
  const apiExterneRouter = express.Router();

  // Benevole & Bénévoles
  apiExterneRouter.get("/api/externe/test", (req, res, next) => getTest(req, res, ne

  return apiExterneRouter;
}
```

#### 4 : Mise en place du système d'authentification via un token

Chemin des fonctions :

(request et baoConnexion)

```
▽ webExterne
  > dist
  > node_modules
  > public
  ▽ src
    > assets
    ▽ compoments
      > assets
      ▽ functions
        > DataType
        > interface
        ⚡ baoConnexion.tsx
        ⚡ request.tsx
    ▽ pages
```



```
export function postAuthInitFetch(body: unknown): InitConfig | InitError {
  const myHeaders = new Headers({
    "Access-Control-Allow-Origin": "*",
    "connection": 'keep-alive',
    'Content-Type': 'application/json',
  });

  const myInit = {
    method: 'POST',
    headers: myHeaders,
    mode: 'cors',
    cache: 'default',
    body: JSON.stringify(body, customStringify),
  };

  return myInit;
}
```

```
export function postFetchAuth(url:string, body:unknown) {
  console.log(`postFetchAuth(${url})`)

  const init = postAuthInitFetch(body) as RequestInit | InitError;
  return execFetch(url, init)
}
```

```

5  interface AuthResponse {
6    token: string;
7    // Autres propriétés de la réponse si nécessaire
8  }
9
10 const FirstLayer = () => {
11   const [dossier, setDossier] = useState<any | null>(null);
12   const [loading, setLoading] = useState<boolean>(true);
13   const [error, setError] = useState<string | null>(null);
14   const [tokenApi, setTokenApi] = useState<string | boolean>(false);
15   const [numDossier, setNumDossier] = useState<string>('');
16
17   useEffect(() => {
18     const requestBody = {
19       identifiant: [REDACTED],
20       password: [REDACTED]
21     };
22
23     postFetchAuth(`api/externe/auth/login`, requestBody).then(async (response) => {
24       if (typeof response === 'object' && response !== null && 'token' in response)
25         localStorage.setItem('tokenApi', (response as AuthResponse).token);
26         setTokenApi((response as AuthResponse).token);
27     } else {
28       throw new Error('La réponse du serveur n\'est pas conforme à AuthResponse')
29     }
30   }).catch((err) => {
31     console.log(`postFetchAuth error: ${err.message}`);
32     setError(err.message);
33     setLoading(false);
34   });
}

```

Si non connecté :

Impression élégante □

{"message":"Utilisateur requit"}

Si connecté, on affiche la page.

Token :

Name	X	Headers	Payload	Preview	Response	Initiator	Timing	Cookies
dossier	1							
index-D-Zva6g3.js	-							
index-DQ7G6vc4.css	-							
injection-topics.js	-							
preprod-yVogd99.css	-							
Dossier-BWuoPXiw.js	-							
request-B4yqZAoY.js	-							
BarlowSemiCondensed-Semi...	-							
orgamipicto-512.png	-							
manifest.json	-							
content-style.css	-							
login	2							
orgamipicto-144.png	-							
wide-screenshot.png	-							

## 5 : Récupération et affichage des données depuis la base de données Atrium via l'API d'Hercule

```

199
200 export async function getInfosDossierAtrium(req: Request, res: Response, next: NextFunction) {
201
202     let numDossier: number | undefined;
203     if (req.params.numDossier) {
204         const parsedNumDossier = parseInt(req.params.numDossier as string);
205         if (!isNaN(parsedNumDossier)) {
206             numDossier = parsedNumDossier;
207         }
208     }
209     if (!numDossier || numDossier <= 0) return returnErreur(res, next, Erreur.noParam);
210
211     try {
212         const dossier = await _getInfosDossierAtrium(numDossier);
213         return returnSucces(res, next, dossier);
214
215     } catch (erreur) {
216         return returnErreur(res, next, Erreur.exception, (erreur as Error).message);
217     }
218
219
220
221 }
```

```
6 const cookieAtrium = 'AUTHSPF-PROD=2YA0O2NAovyCQ8cwI4XLsg==; cookieSNG=AAEA0Cfxs_awDc91B-VZX_zmF-16T_p0DLNnHwCEuHIN34kBE3KAjnLlhS5Jh3TiZq1F9
7 const mois = ["Janvier", "Février", "Mars", "Avril", "Mai", "Juin", "Juillet", "Aout", "Septembre", "Octobre", "Novembre", "Décembre"]
8 const date_nouveau_rdv = 180; //date suggérée pour poser un nouveau rdv
9
10
11 function fetchUrl(url: string, params: RequestInit): Promise<string> {
12     return new Promise((resolve, reject) => {
13         fetch(url, params)
14
15             .then(response => {
16                 console.log(response.status);
17                 console.log(url);
18                 if (response.ok) {
19                     return response.text();
20                 } else {
21                     reject(new Error(`Erreur de requête: ${response.status}`));
22                 }
23             })
24             .then(data => {
25                 if (data !== undefined) {
26                     resolve(data);
27                 } else {
28                     reject(new Error('Data is undefined'));
29                 }
30             })
31             .catch(error => {
32                 reject(error);
33             });
34     });
35 }
```



```
async function _getInfosDossierAtrium(numero: number) {  
    var page = await EXT_recupererPageAtrium(numero);  
  
    var civilité = extraction(page, 'cphMain_familyPersonDetails_lblTitle">', '</span>');  
    var nomFamille = extraction(page, 'ctl00_cphMain_familyPersonDetails_lblLastName">', '<');  
    var prenom = extraction(page, 'ctl00_cphMain_familyPersonDetails_lblFirstName">', '<');  
    var nom = nomFamille + prenom;  
    var numeroTel = extraction(page, ' <span id="ctl00_cphMain_lblMobileNumber">', '</span>');  
    var adresse = reconstruireAdresse(extraction(page, 'ctl00_cphMain_lblAdresse">', '</span>'));  
    var adresseSansCplt = reconstruireAdressesSansComplement(extraction(page, 'ctl00_cphMain_lblAdresse">', '</span>'));  
    var dateDernierEntretien = extraction(page, 'Entretien réalisé le ', '</span>');  
    if (dateDernierEntretien.length > 50) {  
        dateDernierEntretien = "";  
    }  
    var dateProchainEntretien = extraction(page, 'id="ctl00_cphMain_lblNextMeetingDate">', '</span>');  
    var rcdpe = CalulerDateProchainEntretien(dateDernierEntretien.trim(), dateProchainEntretien.trim());  
    var dossierARevoir = rcdpe.aRevoir;  
    var dateProchaineEntretienSuggere = rcdpe.dateARevoir;  
    var nombrePersonnes = extraction(page, '<span id="ctl00_cphMain_lblNombrePersonnes">', '</span>')  
    var famille = [];  
    var blocFamille = extraction(page, '<table cellspacing="0" rules="all" border="0" id="ctl00_cphMain_familyPersonDetails_gvTabPersons" st  
    var indiceDernierePersonne = 0;  
  
    var idxMembreActif = 0;  
    while (idxMembreActif < parseInt(nombrePersonnes)) {  
        var idxMembreActif = 0;  
        while (idxMembreActif < parseInt(nombrePersonnes)) {  
  
            var blocMembre = extractionAvecIndiceDebut(blocFamille, '$lbtnFullName&#39;,&#39;&#39;)" style="white-space:pre-line;">', '/a>', indiceDernierePersonne);  
            var blocClass = extractionAvecIndiceDebut(blocFamille, '" class="custom_tab_header_"', '" href="javascript:_doPostBack', indiceDernierePersonne);  
            if (blocClass.texte != "attache" && blocClass.texte != "decede") {  
                console.log("membre "+blocClass.texte);  
  
                var membre = blocMembre.texte;  
                var premierChamp = extractionAvecIndiceDebut(membre, ">", "(", indiceDernierePersonne - 1);  
                var nomMembre = premierChamp.texte;  
                var deuxiemeChamp = extractionAvecIndiceDebut(membre, "(", ")", premierChamp.indiceFin - 1);  
                var lienMembre = deuxiemeChamp.texte;  
                var dernierChamp = extractionAvecIndiceDebut(membre, ")", "<", deuxiemeChamp.indiceFin - 1);  
                var membreDateNaissance = dernierChamp.texte;  
  
                var membreFamille = {  
                    "nom": nomMembre.trim(),  
                    "lien": lienMembre.trim(),  
                    "dateNaissance": membreDateNaissance.trim()  
                }  
  
                idxMembreActif += 1;  
                famille.push(membreFamille);  
            }  
            indiceDernierePersonne = blocMembre.indiceFin;  
        }  
    }  
}
```



Code de l'affichage web :

```
src / components / pages / Ulysse / First_Layer / First_Layer.jsx / ...  
import { useEffect, useState } from "react";  
import { getFetch, postFetchAuth } from "../../functions/request";  
import { Dossier } from "../../functions/interface/Ulysse/Ulysse";  
import './First_Layers.css';  
  
interface AuthResponse {  
    token: string;  
}  
  
const FirstLayer = () => {  
    const [data, setData] = useState<Dossier | null>(null);  
    const [loading, setLoading] = useState<boolean>(true);  
    const [error, setError] = useState<string | null>(null);  
    const [tokenApi, setTokenApi] = useState<string | boolean>(false);  
  
    useEffect(() => {  
  
        const requestBody = {  
            identifiant: "REINE",  
            password: "REINE5"  
        };  
  
        postFetchAuth(`api/externe/auth/login`, requestBody).then(async (response) => {  
            if (typeof response === 'object' && response !== null && 'token' in response) {  
                localStorage.setItem('tokenApi', (response as AuthResponse).token);  
                setTokenApi((response as AuthResponse).token);  
            } else {  
                throw new Error('La réponse du serveur n\'est pas conforme à AuthResponse');  
            }  
        })  
    }, []);  
};
```



```
useEffect(() => {
  getFetch("api/externe/ulysse/infosDossier/34160")
    .then((response) => {
      setData(response as Dossier);
      setLoading(false);
    })
    .catch((err) => {
      setError(err.message);
      setLoading(false);
    });
}, [tokenApi]);

if (loading) {
  return <div>Loading...</div>;
}

if (error) {
  return <div>Error: {error}</div>;
}

if (!data) {
  return <div>No data available</div>;
}

return (
  <div className="HomeFirstLayer">
    <h1>testtttt</h1>
```

```
const FirstLayer = () => {
  <div>
    |   <strong>numDossier :</strong> {data.numDossier}
  </div>
  <div>
    |   <strong>civilite :</strong> {data.civilite}
  </div>
  <div>
    |   <strong>nom :</strong> {data.nom}
  </div>
  <div>
    |   <strong>numeroTel :</strong> {data.numeroTel}
  </div>
  <div>
    |   <strong>adresse :</strong> {data.adresse}
  </div>
  <div>
    |   <strong>adresseSansCplt :</strong> {data.adresseSansCplt}
  </div>
  <div>
    |   <strong>dateDernierEntretien :</strong> {data.dateDernierEntretien}
  </div>
  <div>
    |   <strong>dateProchainEntretien :</strong> {data.dateProchainEntretien}
  </div>
  <div>
    |   <strong>dateProchaineEntretienSuggere :</strong> {data.dateProchaineEntretienSuggere}
  </div>
  <div>
    |   <strong>dossierARevoir :</strong> {data.dossierARevoir}
  </div>
  <div>
    |   <strong>nombrePersonnes :</strong> {data.nombrePersonnes}
  </div>
  <div>
    |   <strong>famille :</strong>
```

```
        </div>
      <div>
        <strong>famille :</strong>
        <ul>
          {data.famille.map((item, index) => (
            <li key={index}>
              <div>
                <strong>nom:</strong> {item.nom}
              </div>
              <div>
                <strong>lien:</strong> {item.lien}
              </div>
              <div>
                <strong>dateNaissance:</strong> {item.dateNaissance}
              </div>
            </li>
          ))}
        </ul>
      </div>
    </div>
  );
}

export default FirstLayer;
```

URL de l'API :

```
apiExterneRouter.get("/api/externe/ulysseinfosDossier/:numDossier", (req, res, next) => getInfosDossierAtrium(req, res, next, currentSe
```

← → ⌂ localhost:8100/externe/ulysse

```
testttt
Token: eyJhbGciOiJIUzI1NlslnR5cCl6IkpXVCJ9.eyJpZGVudGlmaWFudCl6ImdIYXVjaGUlCJhZGIpbl6dHJIZSwiaWF0IjoxNz0wOTM0MD0yLCJleHAiOjE3NDMzNTMyNDJ9.7tEJ_25W9l8LzIEC_FEsXD5XJRy9tv-
DrVCD0UV9e0
numDossier : 34160
civilite : M
nom : A
numeroTel : 062 33 05
adresse : [REDACTED]
adresseSansCplt : [REDACTED]
dateDernierEntretien :
dateProchainEntretien :
dateProchaineEntretienSuggere :
dossierARevoir :
nombrePersonnes : 4
famille :
nom: [REDACTED]
lien: Pere
dateNaissance: 01/01/1970
nom: [REDACTED]
lien: [REDACTED]
dateNaissance: [REDACTED]
nom: ROBIA
lien: [REDACTED]
dateNaissance: 20/03/1990
nom: ADILBEY
lien: [REDACTED]
dateNaissance: [REDACTED]
```

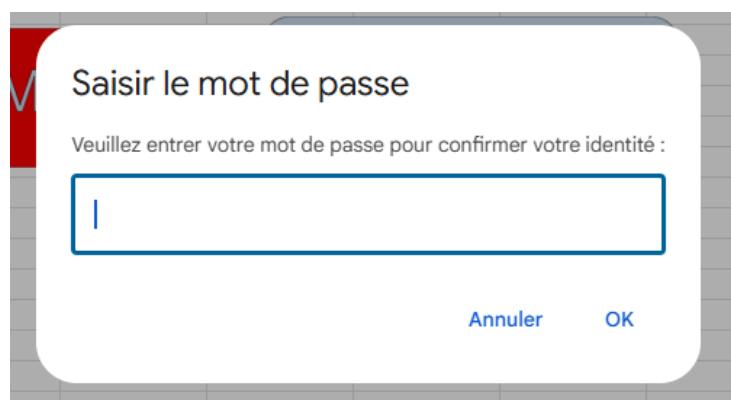
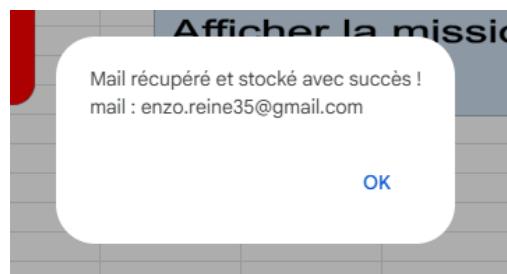
## 6 : Modification des scripts Google pour la synchronisation avec Ulysse

## Environnement Google Sheet de test :

Screenshot of a Google Sheets document titled "TEST - HERCULE". The sheet contains several cells with text and buttons:

- Cells A4 and A6 contain the text "Identifiant" and "Mot de passe" respectively.
- Cell A10 contains the text "Récupérer et afficher les infos".
- Cell A11 contains the text "Get Mail".
- Cell A12 contains the text "Afficher la mission".
- Cell A21 contains the text "get Cgoogle".
- Cell A22 contains the text "input password".
- Cell A23 contains the text "Afficher le token".

Get Mail





**Afficher la mission**



Scripts :

Code.gs

Exemples.gs

fetchDataToken.gs

test.gs

boutonToken.gs

**gmail.gs**

```

function getConnectedUserEmail() {
    // Utiliser le service Session pour obtenir l'email de l'utilisateur
    var email = Session.getActiveUser().getEmail();

    // Afficher l'email dans les logs
    Logger.log("Adresse Gmail de l'utilisateur connecté : " + email);

    // bouton mail affichage
    SpreadsheetApp.getUi().alert("Mail récupéré et stocké avec succès !" + "\nmail : " + email);
    // Retourner l'email
    return email;
}

```

```

✓ function getUserPassword() {
    // Créer une boîte de dialogue pour demander un mot de passe à l'utilisateur
    var ui = SpreadsheetApp.getUi();

    // Demander à l'utilisateur d'entrer un mot de passe via un prompt
    ✓ var response = ui.prompt(
        "Saisir le mot de passe",
        "Veuillez entrer votre mot de passe pour confirmer votre identité :",
        ui.ButtonSet.OK_CANCEL
    );

    // Vérifier la réponse de l'utilisateur
    ✓ if (response.getSelectedButton() == ui.Button.OK) {
        // Si l'utilisateur a cliqué sur "OK", récupérer le mot de passe
        var password = response.getResponseText();
        Logger.log("Mot de passe saisi : " + password);
        return password
        // Exemple : Afficher une alerte pour confirmer que le mot de passe a été reçu
        ui.alert("Mot de passe reçu avec succès !" + "\nmot de passe : " + password);

        // Appeler une autre fonction ou poursuivre le traitement avec le mot de passe
    } else {
        // Si l'utilisateur a cliqué sur "Annuler"
        ui.alert("Action annulée. Aucun mot de passe n'a été fourni.");
    }
}

```

```

// Fonction pour obtenir le token et le stocker dans le stockage de session
function getAndStoreToken() {
    // URL de l'API pour obtenir le token
    var url = "https://preprod.benevolat35.spfdev.org/api/admin/auth/login";

    mail = getConnectedUserEmail();
    password = getUserPassword();

    // Identifiants de l'utilisateur
    var payload = {
        identifiant: mail,
        password: password
    };

    const myHeaders = {
        "Content-Type": "application/json",
        "Access-Control-Allow-Origin": "*",
        "connection": 'keep-alive',
        "Accept": "application/json"
    };

    const myInit = {
        method: 'POST',
        headers: myHeaders,
        payload: JSON.stringify(payload),
        mode: 'cors',
        cache: 'default',
    };
}

// Effectuer la requête pour obtenir le token
try {
    var response = UrlFetchApp.fetch(url, myInit);
    var etatReponse = response.getResponseCode();

    // Vérification du code de la réponse
    if (etatReponse !== 200) {
        throw new Error("Erreur lors de la récupération du token. Code : " + etatReponse);
    }

    // Analyse du JSON de la réponse
    var jsonResponse = JSON.parse(response.getContentText());

    // Stockage du token dans le service "Properties" (équivalent du localStorage)
    var token = jsonResponse.token;
    if (token) {
        PropertiesService.getUserProperties().setProperty('session_token', token);
        Logger.log("Token stocké avec succès : " + token);
        return token;
    } else {
        throw new Error("Aucun token dans la réponse.");
    }
} catch (e) {
    Logger.log("Erreur : " + e.message);
    throw new Error("Impossible de récupérer ou de stocker le token.");
}
}

```

```
// Fonction pour récupérer le token depuis le stockage
function getStoredToken() {
  var token = PropertiesService.getUserProperties().getProperty('session_token');
  if (!token) {
    throw new Error("Aucun token trouvé dans le stockage.");
  }
  Logger.log("Token récupéré : " + token);
  return token;
}

// Exemple d'utilisation du token dans une autre requête
function getInit() {
  // Récupérer le token
  var token = getStoredToken();

  const myHeaders = {
    "Authorization": token, // En-tête avec le token JWT
    "Access-Control-Allow-Origin": "*",
    "connection": 'keep-alive',
    "Accept": "application/json" // On spécifie qu'on attend une réponse en JSON
  };

  const myInit = {
    method: 'GET',
    headers: myHeaders,
    mode: 'cors',
    cache: 'default',
  };

  return myInit;
}
```

```

function getMission() {
    // URL de la mission
    var url = "https://preprod.benevolat35.spfdev.org/api/admin/missions/missions/13";
    var init = getInit();
    // Effectuer la requête
    var response = UrlFetchApp.fetch(url, init);

    var etatReponse = response.getResponseCode(); // Récupère le code de réponse HTTP (200, 404, etc.)
    Logger.log("Code de réponse : " + etatReponse);
    // Vérification du code de la réponse
    if (etatReponse != 200) {
        throw new Error("Erreur lors de la récupération des données. Code : " + response.getResponseCode());
    }
    // Essayer de récupérer la réponse sous forme de JSON
    var body = response.getContentText();
    Logger.log("Contenu texte : " + body);

    // Extraire la réponse JSON
    var jsonResponse = JSON.parse(response.getContentText());

    // Afficher les données dans les logs
    Logger.log(jsonResponse);

    // Exemple d'accès aux différentes propriétés du JSON
    Logger.log("Libellé de l'opération : " + jsonResponse[0].LIBELLEOPERATION);
    Logger.log("Date de l'opération : " + jsonResponse[0].DATE);
    Logger.log("Lieu : " + jsonResponse[0].LIEU);
    Logger.log("Designation : " + jsonResponse[0].DESIGNATION);

    // Retourner l'objet JSON si nécessaire
    return jsonResponse[0];
}

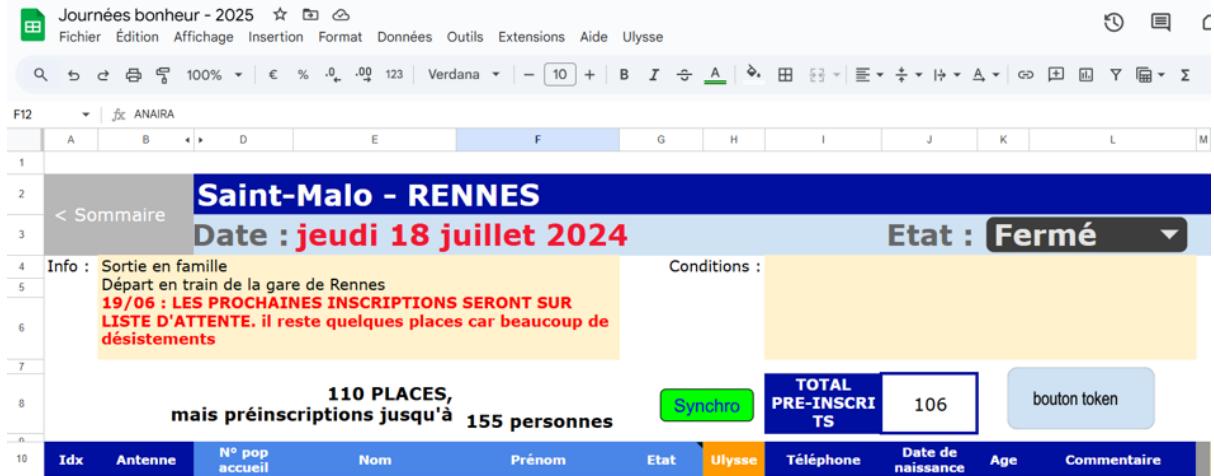
// Lier cette fonction au bouton pour demander le token
function boutonGetAndStoreToken() {
    try {
        var token = getAndStoreToken();
        SpreadsheetApp.getUi().alert("Token récupéré et stocké avec succès !" + "\ntoken : " + token);
    } catch (e) {
        SpreadsheetApp.getUi().alert("Erreur : " + e.message);
    }
}

// Lier cette fonction au bouton pour demander le token
function boutonGetStoredToken() {
    try {
        var token = getStoredToken();
        SpreadsheetApp.getUi().alert("Token récupéré et stocké avec succès !" + "\ntoken : " + token);
    } catch (e) {
        SpreadsheetApp.getUi().alert("Erreur : " + e.message);
    }
}

function boutonGetMission() {
    try {
        var missions = getMission();
        SpreadsheetApp.getUi().alert("data récupéré avec succès !" + "\n mission = " + missions.LIBELLEOPERATION);
    } catch (e) {
        SpreadsheetApp.getUi().alert("Erreur : " + e.message);
    }
}

```

Google Sheet actuelle utilisé par les bénévoles (il y a de la data en dessous) :



Idx	Antenne	N° pop accueil	Nom	Prénom	Etat	Ulysse	Téléphone	Date de naissance	Age	Commentaire
10										

Tests avec le site en ligne. :

```
vs111.secours-populaire benev X + ▾ Microsoft Windows [version 10.0.26100.2605]
(c) Microsoft Corporation. Tous droits réservés.

C:\Users\enzor>ssh -p 3522 benevolat35@145.239.186.107
The authenticity of host '[145.239.186.107]:3522 ([145.239.186.107]:3522)' can't be established.
ED25519 key fingerprint is SHA256:sgtroAb8jLfIK51ulOTjaIcwhtJJD4ocG0rzsofUT4.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added '[145.239.186.107]:3522' (ED25519) to the list of known hosts.
benevolat35@145.239.186.107's password:
Debian GNU/Linux 11

SERVICE          VERSION      [STATUS]  [PORTS]
apache2          2.4.62-1~deb11u2  [OK]      [443, 80, 8080]
cs-php7.4        7.4.33-cs3    [OK]
cs-php8.0        8.0.30-cs1    [OK]
cs-php8.1        8.1.31-cs1    [OK]
cs-php8.2        8.2.27-cs1    [OK]
cs-php8.3-fpm   8.3.15-cs1    [OK]
fail2ban          0.11.2-2     [OK]
influxdb          1.11.8-2     [OK]
mysecureshell    2.0-2+b2     [OK]
netfilter-persistent 1.0.15     [OK]
openssh-server   1:8.4p1-5+deb11u3  [OK]      [3522]
postfix          3.5.25-0+deb11u1  [OK]
```

## 7 : Création de nouvelles tables "Dossier" et "Personne" dans la base de données d'Hercule

```
CREATE TABLE TDOSSIERS (
    CODE INT AUTO_INCREMENT PRIMARY KEY,
    NUMDOSSIER INT NULL,
    CIVILITE VARCHAR(10),
    NOM VARCHAR(255),
    NUMEROTEL VARCHAR(20),
    ADRESSE VARCHAR(255),
    ADRESSESANSCLPT VARCHAR(255),
    DATEDERNIERENTRETIEN DATE NULL,
    DATEPROCHAINENTRETIEN DATE NULL,
    HORODATAGE TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
);

CREATE TABLE TPERSONNE(
    CODE INT AUTO_INCREMENT PRIMARY KEY,
    DOSSIER INT,
    NOM VARCHAR(255),
    LIENPARENTE VARCHAR(50)
    DATENAISANCE DATE NULL,
    HORODATAGE TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    FOREIGN KEY (DOSSIER) REFERENCES TDOSSIERS(CODE)
);
```

|

## 8 : Mise en place d'un système de cache lors de la synchronisation avec Ulysse





34160  Force Atrium

DOSSIER 34160  
Numéro Dossier : 34160  
Civilité : M  
Nom : ATEVATA  
Numéro Tel : 04 67 33 05  
Adresse : 2 RUE  
Adresse Sans Complément :  
Date Dernier Entretien :  
Date Prochain Entretien : 04/07/2022  
Date Prochaine Entretien Suggérée : 04/07/2022  
Dossier à Revoir : Oui  
Dossier actif : Non  
Nombre de Personnes : 4  
Personnes  
Nom :  
Lien Parenté :  
dateNaissance: 01/01/1970  
Nom :  
Lien Parenté :  
dateNaissance: 02/01/1975  
Nom : ATEVATA LV  
Lien Parenté :  
dateNaissance: 25/01/1999  
Nom : ATEVATA Y  
Lien Parenté :  
dateNaissance: 25/01/2004



Code de l'affichage web :

```
5  interface AuthResponse {
6      token: string;
7      // Autres propriétés de la réponse si nécessaire
8  }
9
10 const FirstLayer = () => {
11     const [dossier, setDossier] = useState<any | null>(null);
12     const [loading, setLoading] = useState<boolean>(true);
13     const [error, setError] = useState<string | null>(null);
14     const [tokenApi, setTokenApi] = useState<string | boolean>(false);
15     const [numDossier, setNumDossier] = useState<string>('');
16
17     useEffect(() => {
18         const requestBody = {
19             identifiant: [REDACTED],
20             password: [REDACTED]
21         };
22
23         postFetchAuth(`api/externe/auth/login`, requestBody).then(async (response) => {
24             if (typeof response === 'object' && response !== null && 'token' in response)
25                 localStorage.setItem('tokenApi', (response as AuthResponse).token);
26                 setTokenApi((response as AuthResponse).token);
27             } else {
28                 throw new Error('La réponse du serveur n\'est pas conforme à AuthResponse')
29             }
30         }).catch((err) => {
31             console.log(`postFetchAuth error: ${err.message}`);
32             setError(err.message);
33             setLoading(false);
34         });
35     });
36 }
```

Appele de l'URL de l'api

```
const handleSearch = async (e: React.FormEvent, forceAtrium = false) => {
    console.log(`handleSearch(dossier = ${numDossier})`);
    e.preventDefault();
    setLoading(true);
    setError(null);

    try {
        const dossierResponse = await getFetch(`api/externe/ulysses/infosDossier${forceAtrium?'/forceatrium':''}/#${numDossier}`);
        console.log(`dossierResponse: ${JSON.stringify(dossierResponse)}`);
        setDossier(dossierResponse);
    } catch (err) {
        if (err instanceof Error) {
            console.log(`getFetch error: ${err}`);
            setError(err.message);
        } else {
            console.log(`getFetch error: ${err}`);
            setError("Une erreur inconnue s'est produite");
        }
    } finally {
        setLoading(false);
    }
};
```

```

if (!tokenApi) {
  return <div>Authentification en cours...</div>;
}

return (
  <div className="Ulysse">
    <div className="search-container">
      <form>
        <input type="text" value={numDossier} onChange={(e) => setNumDossier(e.target.value)} placeholder="Numéro de dossier" />
        <button onClick={(e) => handleSearch(e)}>Rechercher</button>
        <button onClick={(e) => handleSearch(e, true)}>Force Atrium</button>
      </form>
    </div>
    {loading && <div>Chargement...</div>}
    {error && <div className="error">Erreur: {error}</div>}
  </div>
)

```

```

{dossier && (
  <>
    <h1><strong>DOSSIER </strong> {dossier.numDossier}</h1>
    <div>
      <strong>Numéro Dossier :</strong> {dossier.numDossier}
    </div>
    <div>
      <strong>Civilité :</strong> {dossier.civilite}
    </div>
  </div>
  <div>
    <strong>Civilité :</strong> {dossier.civilite}
  </div>
  <div>
    <strong>Nom :</strong> {dossier.nom}
  </div>
  <div>
    <strong>Numéro Tel :</strong> {dossier.numeroTel}
  </div>
  <div>
    <strong>Adresse :</strong> {dossier.adresse}
  </div>
  <div>
    <strong>Adresse Sans Complément :</strong> {dossier.adresseSansCplt}
  </div>
  <div>
    <strong>Date Dernier Entretien :</strong> {dossier.dateDernierEntretien}
  </div>
  <div>
    <strong>Date Prochain Entretien :</strong> {dossier.dateProchainEntretien}
  </div>
  <div>
    <strong>Date Prochaine Entretien Suggérée :</strong> {dossier.dateProchaineEntretienSuggere}
  </div>
  <div>
    <strong>Dossier à Revoir :</strong> {dossier.dossierARevoir ? "Oui" : "Non"}<br />
  </div>
  <div>
    <strong>Dossier actif :</strong> {dossier.actif ? "Oui" : "Non"}
  </div>
  <div>
    <strong>Nombre de Personnes :</strong> {dossier.nombrePersonnes}
  </div>
)

```

```

        <div>
            <strong>Nombre de Personnes :</strong> {dossier.nombrePersonnes}
        </div>
        <h2>Personnes</h2>
        <ul>
            {dossier.famille.map((personne: any, index: number) => (
                <li key={index}>
                    <div>
                        <strong>Nom :</strong> {personne.nom}
                    </div>
                    <div>
                        <strong>Lien Parenté :</strong> {personne.lien}
                    </div>
                    <div>
                        <strong>dateNaissance:</strong> {personne.dateNaissance}
                    </div>
                </li>
            ))}
        </ul>
    </div>
);

export default FirstLayer;

```

Fonction de l'API (tout le système de Mise à jour) :

Route de l'API :

```
apiExterneRouter.post("/api/externe/participation/majParticipation", (req, res, next) => majParticipation(req, res, next, currentSession))
```

Imports :

```

import { Request, Response, NextFunction } from 'express';
import { returnErreur, Erreur, returnSuccess } from '../../../../../function/returnMessage';
import { _getInfosDossierAtrium, _calulerDateProchainEntretien, DossierAtriumComplet } from './Atrium';
import { _getDossierCompletByNumero, _postDossier } from '../dossiers/code';
import { _patchPersonne, _deletePersonne, _postPersonne } from '../personnes/code';
import { DossierComplet } from '../../../../../interfaceType/interfaces/tDossiers';
import { pool } from '../../../../../index';
import { FieldPacket, ResultSetHeader } from 'mysql2/promise';
import { isTelephone, Telephone } from '../../../../../interfaceType/types/ElementBasic';
import { formatDate, parseDate } from "../../devFunction/formatage";
import { Personnes } from '../../../../../interfaceType/interfaces/tPersonnes';
import { formatDateISO } from "../../../../../function/formats";
import { returnSuccessInterne, returnErrorInterne, Retour } from "../../../../../function/returnMessage";
import { CurrentSession } from '../../../../../interfaceType/interfaces/currentSession';
import { _postSuivi } from "../suivi/code";

//const TIMEOUTCACHE = 1000; // 1 seconde pour débogage
const TIMEOUTCACHE = 1000 * 60 * 60 * 24 * 30 * 3; // 90 jours

```

```

export async function getInfosDossierAtriumForce(req: Request, res: Response, next: NextFunction, currentSession: CurrentSession) {
  let numDossier: number | undefined;

  if (req.params.numDossier) {
    const parsedNumDossier = parseInt(req.params.numDossier as string);
    if (!isNaN(parsedNumDossier)) {
      numDossier = parsedNumDossier;
    }
  }
  if (!numDossier || numDossier <= 0) {
    console.error("Erreur : numDossier invalide ou manquant");
    return returnErreurs(res, next, Erreur.noParam);
  }

  const result = await _getInfosDossierUlysse(numDossier, true);
  if (result.SUCCES) {
    return returnSucces(res, next, result.BODY);
  } else {
    _postSuivi(currentSession, null, null, `getInfosDossierAtriumForce(${numDossier}) : ${result.MESSAGE}`);
    return returnErreurs(res, next, result.CODEERREUR, result.MESSAGE);
  }
}

```

```

async function CreerDossier(dossierUlysse : DossierAtriumComplet) {
  try {

    if (!dossierUlysse) {
      throw new Error("Le dossier est vide");
    }

    const dossierComplet = convertDossierJSONToCache(dossierUlysse)
    const newDossier = await _postDossier(dossierComplet);
    if (!newDossier) {
      throw new Error("Échec de la création du dossier");
    }

    dossierComplet.TPERSONNES.forEach(async personne => {
      personne.DOSSIER = newDossier.CODE;
      await _postPersonne(personne);
    })

    return true;
  } catch (error) {
    console.error(error);
    return false;
  }
}

```



```
function convertDossierCacheToJson(dossierCache: DossierComplet) : DossierAtriumComplet {
    try {

        const rcdpe = CalulerDateProchainEntretien(dossierCache.DATEDERNIERENTRETIEN.toLocaleDateString(), dossierCache.DATEPROCHAINENTRETIEN);
        const dossierARevoir = rcdpe.aRevoir;
        const dateProchaineEntretienSuggere = rcdpe.dateARevoir;

        const retour : DossierAtriumComplet = {
            numDossier: dossierCache.NUMDOSSIER,
            civilite: dossierCache.CIVILITE,
            nom: dossierCache.NOM,
            numeroTel: dossierCache.TELEPHONE ? dossierCache.TELEPHONE.toString() : "",
            adresse: dossierCache.ADRESSE,
            adresseSansCplt: dossierCache.ADRESSESANSCPLT,
            dateDernierEntretien: formatDate(dossierCache.DATEDERNIERENTRETIEN),
            dateProchainEntretien: formatDate(dossierCache.DATEPROCHAINENTRETIEN),
            dateProchaineEntretienSuggere: dateProchaineEntretienSuggere,
            dossierARevoir: dossierARevoir,
            nombrePersonnes: dossierCache.TPERSONNES.length.toString(),
            famille: dossierCache.TPERSONNES.map((personnes) => {
                return {
                    nom: personnes.NOMPRENOM,
                    lien: personnes.LIENPARENTE,
                    dateNaissance: formatDate(personnes.DATENAISSANCE)
                }
            }),
            horodatage: formatDate(dossierCache.HORODATAGE)
        }
        return retour;
    } catch (error) {
        console.error(error);
        throw new Error("Erreur lors de la récupération des informations du dossier");
    }
}
```

```
function convertDossierJSONToCache(dossierAtrium: DossierAtriumComplet): DossierComplet {
    console.log("Conversion du dossier JSON vers le cache :");
    console.log(dossierAtrium);

    return {
        CODE: 0,
        NUMDOSSIER: dossierAtrium.numDossier,
        CIVILITE: dossierAtrium.civilite,
        NOM: dossierAtrium.nom,
        TELEPHONE: isTelephone(dossierAtrium.numeroTel) ? dossierAtrium.numeroTel : '0000000000' as Telephone,
        ADRESSE: dossierAtrium.adresse,
        ADRESSESANSCPLT: dossierAtrium.adresseSansCplt,
        DATEDERNIERENTRETIEN: parseDate(dossierAtrium.dateDernierEntretien),
        DATEPROCHAINENTRETIEN: parseDate(dossierAtrium.dateProchainEntretien),
        HORODATAGE: new Date(dossierAtrium.horodatage),
        TPERSONNES: dossierAtrium.famille.map(personne => ({
            CODE: 0,
            DOSSIER: 0,
            NOMPRENOM: personne.nom,
            LIENPARENTE: personne.lien,
            DATENAISSANCE: parseDate(personne.dateNaissance),
            HORODATAGE: new Date(dossierAtrium.horodatage),
            FLAG: false
        }))
    };
}
```

```

export async function MettreAJourCache(dossierCache: DossierComplet, dossierAtrium: DossierComplet): Promise<{ success: boolean; message: string }> {
    console.log("----- Mise à jour du cache en cours -----");
    // Initialisation des flags à false
    dossierCache.TPERSONNES.forEach(personne => {
        personne.FLAG = false;
    });
    dossierAtrium.TPERSONNES.forEach(personne => {
        personne.FLAG = false;
    });
    console.log("Dossier cache : ", dossierCache);
    console.log("Dossier Atrium : ", dossierAtrium);

    const updateDossierQuery = `

        UPDATE TDOSSIERS SET
            NUMDOSSIER = ?,
            CIVILITE = ?,
            NOM = ?,
            TELEPHONE = ?,
            ADRESSE = ?,
            ADRESSESANSPLT = ?,
            DATEDERNIERENTRETIEN = ?,
            DATEPROCHAINENTRETIEN = ?
        WHERE CODE = ?
    `;
}
try {

```

```

try {

    // Afficher les paramètres qui vont être envoyés à la requête d'update
    const dossierParams = [
        dossierAtrium.NUMDOSSIER,
        dossierAtrium.CIVILITE,
        dossierAtrium.NOM,
        dossierAtrium.TELEPHONE,
        dossierAtrium.ADRESSE,
        dossierAtrium.ADRESSESANSPLT,
        dossierAtrium.DATEDERNIERENTRETIEN,
        dossierAtrium.DATEPROCHATENTRETIEN,
        dossierCache.CODE
    ];

    const [dossierResults]: [ResultSetHeader, FieldPacket[]] = await pool.execute(updateDossierQuery, dossierParams);

    // Vérifier que le dossier a bien été mis à jour
    if (dossierResults.affectedRows === 0) {
        console.error("Aucune ligne affectée lors de la mise à jour du dossier. Vérifiez que le CODE est correct et que les données diffèrent.");
        return { success: false, message: "Échec de la mise à jour du dossier, aucune ligne affectée." };
    }

    // Initialisation des compteurs de synchronisation
    let personnesCacheMatch: Personnes[] | undefined;
    let cptMaj = 0;
    let cptMajMemoire = -1;


```

```

    // Synchronisation des personnes
    // Parcours les personnes trouvées dans atrium
    // en plusieurs passes jusqu'à ce qu'elles soient toutes traitées ou que la dernière passe n'a rien pu traiter
    while (cptMaj < dossierCache.TPERSONNES.length && cptMaj !== cptMajMemoire) {
        cptMajMemoire = cptMaj;

        for (const personneAtrium of dossierAtrium.TPERSONNES) {
            if (personneAtrium.FLAG) continue;

            // Filtrer sur les personnes non mises à jour avec une DATENAISANCE identique (format yyyy-mm-dd)
            personnesCacheMatch = dossierCache.TPERSONNES.filter(p =>
                p.FLAG &&
                formatDateISO(p.DATENAISANCE) === formatDateISO(personneAtrium.DATENAISANCE)
            );

            console.log(`= = = > traitement 1ère personne : ${personneAtrium.NOMPrenom} - ${personneAtrium.DATENAISANCE}`);

            if (personnesCacheMatch.length === 1) {
                console.log("Cas 1: Un seul match trouvé sur la date de naissance => Mise à jour de la personne");

                personnesCacheMatch[0].HORODATAGE = new Date();
                personnesCacheMatch[0].LIENPARENTE = personneAtrium.LIENPARENTE;
                personnesCacheMatch[0].NOMPrenom = personneAtrium.NOMPrenom;

                await _patchPersonne(personnesCacheMatch[0]);

                personneAtrium.FLAG = true;
                personnesCacheMatch[0].FLAG = true;
                cptMaj++;
            }
        }

        else { // Cas 2 & 3 : Plusieurs matches sur la date de naissance

```

```

        } else { // Cas 2 & 3 : Plusieurs matches sur la date de naissance

            if (personnesCacheMatch.length > 1) {
                console.warn("Cas 2 & 3 : Plusieurs personnes trouvées avec la même date de naissance => Filtre avec le nom");
                // Rechercher un match exact sur le NOMPRENOM parmi ceux trouvés avec la date de naissance
                personnesCacheMatch = dossierCache.TPERSONNES.filter(p =>
                    !p.FLAG &&
                    formatDateISO(p.DATENAISSANCE) === formatDateISO(personneAtrium.DATENAISSANCE) &&
                    p.NOMPRENOM === personneAtrium.NOMPRENOM
                );
            }

            if (personnesCacheMatch.length === 1) {
                console.log("cas 2: Un seul match trouvé sur la date de naissance et le NOMPRENOM => Mise à jour de la personne");
                await _patchPersonne(personnesCacheMatch[0]);

                personneAtrium.FLAG = true;
                personnesCacheMatch[0].FLAG = true;
                cptMaj++;

            } else {
                if (personnesCacheMatch.length > 1) {
                    console.warn("Echec Cas 2: Plusieurs personnes trouvées avec la même date de naissance et le même NOMPRENOM", personnesCacheMatch[0]);
                    return { success: false, message: "Échec de la mise à jour du dossier : Plusieurs personnes trouvées avec la même date de naissance et le même NOMPRENOM" };
                } else {
                    console.error("Cas 3: Aucun match sur le NOMPRENOM pour la date de naissance => passe le tour");
                }
            }
        } else {
            // Cas 4 / 5 : Aucun match sur la date => Recherche par NOMPRENOM uniquement
            console.warn("Cas 4 & 5 : Aucun match trouvé sur la date de naissance => Filtre sur le NOMPRENOM");
            personnesCacheMatch = dossierCache.TPERSONNES.filter(p =>
                !p.FLAG &&
                p.NOMPRENOM === personneAtrium.NOMPRENOM
            );
        }
    } else {
        // Cas 4 / 5 : Aucun match sur la date => Recherche par NOMPRENOM uniquement
        console.warn("Cas 4 & 5 : Aucun match trouvé sur la date de naissance => Filtre sur le NOMPRENOM");
        personnesCacheMatch = dossierCache.TPERSONNES.filter(p =>
            !p.FLAG &&
            p.NOMPRENOM === personneAtrium.NOMPRENOM
        );
    }

    if (personnesCacheMatch.length === 1) {
        console.log("Cas 4: Un seul match trouvé sur le NOMPRENOM => Mise à jour de la personne");

        personnesCacheMatch[0].HORODATAGE = new Date();
        personnesCacheMatch[0].DATENAISSANCE = personneAtrium.DATENAISSANCE;
        personnesCacheMatch[0].LIENPARENTE = personneAtrium.LIENPARENTE;

        await _patchPersonne(personnesCacheMatch[0]);

        personneAtrium.FLAG = true;
        personnesCacheMatch[0].FLAG = true;
        cptMaj++;

    } else {
        if (personnesCacheMatch.length > 1) {
            console.log("Echec Cas 4: Plusieurs personnes trouvées avec le même NOMPRENOM");
            return { success: false, message: "Échec de la mise à jour du dossier : Plusieurs personnes trouvées avec le même NOMPRENOM." };
        } else {
            // non trouvé ni par la date ni par le nom
            console.log("Cas 5: Insertion d'une nouvelle personne");

            const personne : Personnes = {

```

```

        }
    }
}

```

```

    } else {
      if (personnesCacheMatch.length > 1) {
        console.log("Échec Cas 4: Plusieurs personnes trouvées avec le même NOMPRENOM");
        return { success: false, message: "Échec de la mise à jour du dossier : Plusieurs personnes trouvées avec le même NOMPRENOM." };
      } else {
        // non trouvé ni par la date ni par le nom
        console.log("Cas 5: Insertion d'une nouvelle personne");

        const personne : Personnes = {
          CODE: -1,
          DOSSIER: dossierCache.CODE,
          NOMPRENOM: personneAtrium.NOMPRENOM,
          LIENPARENTE: personneAtrium.LIENPARENTE,
          DATENAISSANCE: personneAtrium.DATENAISSANCE,
          HORODATAGE: new Date(),
          FLAG: true
        }

        await _postPersonne(personne);

        personneAtrium.FLAG = true;
        cptMaj++;
      }
    }
  }
}

```

Outil Capture d'écran  
Capture d'écran copiée dans le Enregistrement automatique des captures d'écran.

```

export async function _MettreAJourCache(dossierCache: DossierComplet, dossierAtrium: DossierComplet): Promise<{ success: boolean; message: string }> {
  ...
}

const personnesNonTrouvees = dossierAtrium.TPERSONNES.filter(p => !p.FLAG);
if (personnesNonTrouvees.length > 0) {
  console.log(`Personnes non traitées : ${personnesNonTrouvees}`);
  return { success: false, message: `Échec de la mise à jour du dossier : ${personnesNonTrouvees.length} personne(s) dans le dossier n'ont pas pu être traitée` };
}

console.log("----- Mise à jour des personnes terminée -----");
console.log(dossierCache);

console.log("Suppression des non mises à jour du cache");

// Suppression des personnes du cache dont le flag est resté false si le nombre est supérieur à celui d'Atrium
const personnesNonMisesAJour = dossierCache.TPERSONNES.filter(p => p.FLAG === false);
if (personnesNonMisesAJour.length > 0) {
  console.log(`Suppression des personnes non mises à jour (flag false) du cache : ${personnesNonMisesAJour}`);
  personnesNonMisesAJour.forEach(async personne => {
    await _deletePersonne(personne.CODE);
  });
}

console.log("----- Mise à jour du cache terminée avec succès -----");
return { success: true, message: "Mise à jour réussie" };

} catch (error) {
  console.error(`Erreur lors de la synchronisation, rollback effectué : ${error}`);
  return { success: false, message: (error as Error).message };
}
}

```

## 9 : Création des tables "Événement" et "Participation" dans la base de données d'Hercule

```
CREATE TABLE TTYPESEVENEMENTS (
    CODE INT AUTO_INCREMENT PRIMARY KEY,
    DESIGNATION VARCHAR(50) NOT NULL UNIQUE,
);

INSERT INTO TTYPESEVENEMENTS (DESIGNATION)
VALUES ('APV'),
       ('BSV'),
       ('JB');

CREATE TABLE TEVENEMENTS [
    CODE INT AUTO_INCREMENT PRIMARY KEY,
    DESIGNATION VARCHAR(50) NOT NULL,
    TYPEEVENEMENT INT NOT NULL,
    ANNEE YEAR NOT NULL,
    VALEUR INT NOT NULL,
    COMMENTAIRES TEXT DEFAULT NULL,
    CONSTRAINT fk_event_TYPEOPERATION FOREIGN KEY (TYPEEVENEMENT) REFERENCES TTYPESEVENEMENTS(CODE)
];

CREATE TABLE TPARTICIPATIONS (
    CODE INT AUTO_INCREMENT PRIMARY KEY,
    PERSONNE INT NOT NULL,
    EVENEMENT INT NOT NULL,
    ETAT VARCHAR(2) NOT NULL,
    VALEUR INT NOT NULL,
    CONSTRAINT fk_participation_evenement FOREIGN KEY (EVENEMENT) REFERENCES TEVENEMENTS(CODE),
    CONSTRAINT fk_participation_personne FOREIGN KEY (PERSONNE) REFERENCES TPERSONNES(CODE)
);
```



## Interfaces :

The screenshot displays a code editor interface with two panes. The left pane shows a file structure for the 'ORGAMIZ2' project, specifically focusing on the 'apiTS' folder which contains various TypeScript interface definitions. The right pane shows the actual code for two of these interfaces: 'tEvenements.ts' and 'tParticipations.ts'.

**tEvenements.ts:**

```
apiTS > interfaceType > interfaces > tEvenements.ts > Evenement
1  export interface Evenement {
2    CODE: number;
3    DESIGNATION: string;
4    TYPEEVENEMENT: number;
5    ANNEE: number;
6    VALEUR: number;
7    COMMENTAIRES: string | null;
8  }
```

**tParticipations.ts:**

```
apiTS > interfaceType > interfaces > tParticipations.ts > Participation
1  // Interface representant une participation dans la base de données
2  export interface Participation {
3    CODE: number;
4    PERSONNE: number;
5    EVENEMENT: number;
6    ETAT: string;
7    VALEUR: number;
8  }
9
10 // Interface pour le cumul des points par personne
11 export interface PersonneCumulePoint {
12   NUMDOSSIER: number;
13   NOMPRENOM: string;
14   TOTALVALEUR: number;
15   COMMENTAIRE?: string | null;
16 }
```

## **10 : Développement d'une page d'affichage des événements pour les administrateurs**

## **Routes :**

```
// Evenements
apiAdminRouter.get("/api/admin/evenements/evenements", (req, res, next) => getEvenements(req, res, next));
apiAdminRouter.get("/api/admin/evenements/evenement/:code", (req, res, next) => getEvenement(req, res, next));
apiAdminRouter.get("/api/admin/evenements/evenements/:ordreAffichage/:filterType/:rechercheEvenement", (req, res, next) => getEvenementsSearch(req, res, next));
apiAdminRouter.patch("/api/admin/evenements/evenement/:code", (req, res, next) => patchEvenement(req, res, next));
apiAdminRouter.delete("/api/admin/evenements/evenement/:code", (req, res, next) => deleteEvenement(req, res, next));
const apiAdminRouter: Router = /evenements/evenement", (req, res, next) => postEvenement(req, res, next));
apiAdminRouter.get("/api/admin/evenements/annees", (req, res, next) => getEvenementYears(req, res, next));
apiAdminRouter.get("/api/admin/evenements/evenements/annee/:ordreAffichage/:annee/:rechercheEvenement", (req, res, next) => getEvenementsSearchByYear(req, res, next));
apiAdminRouter.get("/api/admin/evenements/typeevenement", (req, res, next) => getTypeEvenements(req, res, next));
```

## Fonctions dans l'api :

## imports

```
import { Request, Response, NextFunction } from 'express';
import { pool } from '../../index';
import { ResultSetHeader, FieldPacket } from 'mysql2';
import { returnErreur, Erreur, returnSucces } from '../../function/returnMessage';
import { Evenement } from '../../interfaceType/interfaces/tEvenements';
```

```

export async function postEvenement(req: Request, res: Response, next: NextFunction) {
    console.log("postEvenement");

    // Récupère le body de la requête
    const body = req.body;
    if (!body) return returnErreur(res, next, Erreur.noBody);

    // Extraction des champs attendus depuis le body
    const { DESIGNATION, TYPEEVENEMENT, ANNEE, VALEUR, COMMENTAIRES } = body;

    // Validation des types et de la présence des champs obligatoires
    if (typeof DESIGNATION !== "string" || DESIGNATION.trim() === "")
        return returnErreur(res, next, Erreur.badData, "DESIGNATION (string attendu)");
    if (isNaN(Number(TYPEEVENEMENT)))
        return returnErreur(res, next, Erreur.badData, "TYPEEVENEMENT (number attendu)");
    if (isNaN(Number(ANNEE)))
        return returnErreur(res, next, Erreur.badData, "ANNEE (number attendu)");
    if (isNaN(Number(VALEUR)))
        return returnErreur(res, next, Erreur.badData, "VALEUR (number attendu)");

    // Requête SQL pour insérer un nouvel événement
    const createQuery = `
        INSERT INTO TEVENEMENTS (DESIGNATION, TYPEEVENEMENT, ANNEE, VALEUR, COMMENTAIRES)
        VALUES (?, ?, ?, ?, ?)
    `;

    try {

```

```

export async function postEvenement(req: Request, res: Response, next: NextFunction) {
    console.log("postEvenement");

    // Récupère le body de la requête
    const body = req.body;
    if (!body) return returnErreur(res, next, Erreur.noBody);

    // Extraction des champs attendus depuis le body
    const { DESIGNATION, TYPEEVENEMENT, ANNEE, VALEUR, COMMENTAIRES } = body;

    // Validation des types et de la présence des champs obligatoires
    if (typeof DESIGNATION !== "string" || DESIGNATION.trim() === "")
        return returnErreur(res, next, Erreur.badData, "DESIGNATION (string attendu)");
    if (isNaN(Number(TYPEEVENEMENT)))
        return returnErreur(res, next, Erreur.badData, "TYPEEVENEMENT (number attendu)");
    if (isNaN(Number(ANNEE)))
        return returnErreur(res, next, Erreur.badData, "ANNEE (number attendu)");
    if (isNaN(Number(VALEUR)))
        return returnErreur(res, next, Erreur.badData, "VALEUR (number attendu)");

    // Requête SQL pour insérer un nouvel événement
    const createQuery = `
        INSERT INTO TEVENEMENTS (DESIGNATION, TYPEEVENEMENT, ANNEE, VALEUR, COMMENTAIRES)
        VALUES (?, ?, ?, ?, ?)
    `;

    try {

```

```
function getOrderByFor(ordreAffichage: string = ""): string {
  switch (ordreAffichage) {
    case 'designation': return 'ORDER BY TEVENEMENTS.DESIGNATION ASC';
    case 'typeevenement': return 'ORDER BY TEVENEMENTS.TYPEREVENEMENT ASC';
    case 'annee': return 'ORDER BY TEVENEMENTS.ANNEE ASC';
    case 'valeur': return 'ORDER BY TEVENEMENTS.VEAUER ASC';
    default: return '';
  }
}
```

```
export async function getEvenements(req: Request, res: Response, next: NextFunction) {
  console.log("getEvenements");
  const query = `
    SELECT CODE, DESIGNATION, TYPEEVENEMENT, ANNEE, VALEUR, COMMENTAIRES
    FROM TEVENEMENTS
    ORDER BY DESIGNATION ASC
  `;

  try {
    const [results] = await pool.execute(query);
    if (!Array.isArray(results))
      return returnErreurs(res, next, Erreur.notArray);
    const evenements: Evenement[] = results as Evenement[];
    return returnSucces(res, next, evenements);
  } catch (error) {
    console.error("Erreur getEvenements:", error);
    return returnErreurs(
      res,
      next,
      Erreur.exception,
      error instanceof Error ? error.message : ''
    );
  }
}
```

```

export async function _getEvenementByCode(code: number) {
    try {
        const query = `SELECT * FROM TEVENEMENTS
        WHERE CODE = ?`;
        const [results] = await pool.execute(query, [code]);
        if (!Array.isArray(results) || results.length === 0) {
            throw new Error(`Aucun événement trouvé pour le code ${code}`);
        }
        return results[0] as Evenement;
    } catch (error) {
        console.error("Erreur dans _getEvenementByCode :", error);
        throw error;
    }
}

```

```

export async function getEvenementsSearch(req: Request, res: Response, next: NextFunction) {
    const ordreAffichage = req.params.ordreAffichage;
    const filtreType = req.params.filtreType;
    const rechercheEvenement = req.params.rechercheEvenement;

    if (!ordreAffichage)
        return returnErreur(res, next, Erreur.noParam, 'ordreAffichage');
    if (!filtreType)
        return returnErreur(res, next, Erreur.noParam, 'filtreType');
    if (rechercheEvenement === undefined)
        return returnErreur(res, next, Erreur.noParam, 'rechercheEvenement');

    // Si rechercheEvenement vaut "*" (joker pour vide) on considère la chaîne vide
    const searchPattern = `${rechercheEvenement === '*' ? '' : rechercheEvenement}`;

    const query = `SELECT CODE, DESIGNATION, TYPEEVENEMENT, ANNEE, VALEUR, COMMENTAIRES
    FROM TEVENEMENTS
    WHERE TYPEEVENEMENT = ?
    AND DESIGNATION LIKE ?
    + getOrderByFor(ordreAffichage);

    console.log("Query EvenementsSearch:", query);
    try {
        const [results] = await pool.execute(query, [filtreType, searchPattern]);
        if (!Array.isArray(results))
            return returnErreur(res, next, Erreur.notArray);
        const listeEvenements: Evenement[] = results as Evenement[];
        return returnSucces(res, next, listeEvenements);
    } catch (error) {
        if (error instanceof Error) {
            console.error("Erreur getEvenementsSearch:", error.name, error.message);
            return returnErreur(res, next, Erreur.exception, `${error.name} = ${error.message}`);
        } else {
            return returnErreur(res, next, Erreur.exception, "");
        }
    }
}

```

```

export async function getEvenement(req: Request, res: Response, next: NextFunction) {
    console.log("getEvenement");

    // Récupération du paramètre CODE dans req.params
    const codeStr = req.params.code;
    if (!codeStr || isNaN(Number(codeStr))) {
        return returnErreur(res, next, Erreur.noParam, 'code');
    }

    const query = `
        SELECT * FROM TEVENEMENTS WHERE CODE = ?
    `;

    try {
        const [results] = await pool.execute(query, [Number(codeStr)]);
        if (!Array.isArray(results) || results.length === 0) {
            return returnErreur(res, next, Erreur.notArray, "Événement non trouvé");
        }
        return returnSucces(res, next, results[0]);
    } catch (error) {
        if (error instanceof Error) {
            console.log("Erreur getEvenement :", error.name, error.message);
            return returnErreur(res, next, Erreur.exception, error.name + " = " + error.message);
        } else {
            return returnErreur(res, next, Erreur.exception, "");
        }
    }
}

```

```

export async function patchEvenement(req: Request, res: Response, next: NextFunction) {
    console.log("patchEvenement invoked");

    // Récupération du paramètre code
    const codeStr = req.params.code;
    console.log("Paramètre code reçu:", codeStr);
    if (!codeStr || isNaN(Number(codeStr))) {
        console.error("Code invalide:", codeStr);
        return returnErreur(res, next, Erreur.noParam, 'code');
    }

    // Récupération des champs attendus depuis le body
    const { DESIGNATION, TYPEEVENEMENT, ANNEE, VALEUR, COMMENTAIRES } = req.body;
    console.log("Données du body:", { DESIGNATION, TYPEEVENEMENT, ANNEE, VALEUR, COMMENTAIRES });
    if (!DESIGNATION || !TYPEEVENEMENT || !ANNEE || !VALEUR) {
        console.error("Champs obligatoires manquants:", { DESIGNATION, TYPEEVENEMENT, ANNEE, VALEUR });
        return returnErreur(res, next, Erreur.noParam, 'DESIGNATION, TYPEEVENEMENT, ANNEE et VALEUR sont requis');
    }

    // Requête SQL pour récupérer l'événement existant
    const selectQuery = `
        SELECT CODE, DESIGNATION, TYPEEVENEMENT, ANNEE, VALEUR, COMMENTAIRES
        FROM TEVENEMENTS
        WHERE CODE = ?
    `;
    console.log("Exécution de la requête de sélection:", selectQuery);

    try {

```

```

try {
  const [results] = await pool.execute(selectQuery, [Number(codestr)]);
  if (!Array.isArray(results)) {
    console.error("Résultat invalide (non tableau) de la requête de sélection");
    return returnErreur(res, next, Erreur.notArray);
  }
  if (results.length <= 0) {
    console.error("Aucun événement trouvé pour CODE =", codeStr);
    return returnErreur(res, next, Erreur.noData);
  }

  // Récupération de l'événement existant
  const evenementExistant = results[0] as Evenement;
  console.log("Événement existant:", evenementExistant);

  // Comparaison et collecte des modifications
  const modifications: { field: string; value: any }[] = [];
  if (DESIGNATION && DESIGNATION !== evenementExistant.DESIGNATION) {
    modifications.push({ field: "DESIGNATION = ?", value: DESIGNATION });
    console.log(`Modification pour DESIGNATION: ${evenementExistant.DESIGNATION} -> ${DESIGNATION}`);
  }
  if (TYPEEVENEMENT && Number(TYPEEVENEMENT) !== evenementExistant.TYPEEVENEMENT) {
    modifications.push({ field: "TYPEEVENEMENT = ?", value: Number(TYPEEVENEMENT) });
    console.log(`Modification pour TYPEEVENEMENT: ${evenementExistant.TYPEEVENEMENT} -> ${TYPEEVENEMENT}`);
  }
  if (ANNEE && Number(ANNEE) !== evenementExistant.ANNEE) {
    modifications.push({ field: "ANNEE = ?", value: Number(ANNEE) });
    console.log(`Modification pour ANNEE: ${evenementExistant.ANNEE} -> ${ANNEE}`);
  }
  if (VALEUR && Number(VALEUR) !== evenementExistant.VALEUR) {
    modifications.push({ field: "VALEUR = ?", value: Number(VALEUR) });
    console.log(`Modification pour VALEUR: ${evenementExistant.VALEUR} -> ${VALEUR}`);
  }
  // COMMENTATIRES est facultatif et peut être null
}

```

```

// COMMENTAIRES est facultatif et peut être null
if (COMMENTAIRES !== undefined && COMMENTAIRES !== evenementExistant.COMMENTAIRES) {
    modifications.push({ field: "COMMENTAIRES =?", value: COMMENTAIRES || null });
    console.log(`Modification pour COMMENTAIRES: ${evenementExistant.COMMENTAIRES} -> ${COMMENTAIRES}`);
}

if (modifications.length <= 0) {
    console.error("Aucune modification détectée");
    return returnErreur(res, next, Erreur.noDataModified);
}

// Construction de la requête de mise à jour dynamique
const updateQuery =
    UPDATE TEVENEMENTS
    SET ${modifications.map(mod => mod.field).join(", ")}
    WHERE CODE = ?
;
const updateParams = modifications.map(mod => mod.value);
updateParams.push(Number(codeStr));

console.log("Requête de mise à jour:", updateQuery);
console.log("Paramètres de mise à jour:", updateParams);

const [updateResults]: [ResultSetHeader, FieldPacket[]} = await pool.execute(updateQuery, updateParams);
if (updateResults.affectedRows === 0) {
    console.error("Échec de la mise à jour pour CODE =", codeStr);
    return returnErreur(res, next, Erreur.echecUpdate, "Échec de la mise à jour de l'événement");
}
console.log("Mise à jour effectuée avec succès pour CODE =", codeStr);

// Relecture de l'événement mis à jour
const [updatedResults] = await pool.execute(selectQuery, [Number(codeStr)]);
if (!Array.isArray(updatedResults) || updatedResults.length === 0) {
    console.error("Erreur lors de la relecture de l'événement mis à jour");
    return returnErreur(res, next, Erreur.noData, "Erreur lors de la relecture de l'événement mis à jour");
}

const updatedEvenement = updatedResults[0] as Evenement;
console.log("Événement mis à jour:", updatedEvenement);

return returnSuccess(res, next, updatedEvenement);

} catch (error) {
    if (error instanceof Error) {
        console.error("Erreur dans patchEvenement :", error.name, error.message);
        return returnErreur(res, next, Erreur.exception, error.name + " = " + error.message);
    } else {
        console.error("Erreur inconnue dans patchEvenement");
        return returnErreur(res, next, Erreur.exception, "");
    }
}
}

```

```

export async function deleteEvenement(req: Request, res: Response, next: NextFunction) {
    console.log("deleteEvenement");
    const code = req.params.code;
    if (!code || isNaN(Number(code)))
        return returnErreur(res, next, Erreur.noParam, 'code');

    const query = ` 
        DELETE FROM TEVENEMENTS
        WHERE CODE = ?;
    `;
    try {
        const [results] = await pool.execute(query, [code]);
        if ((results as any).affectedRows === 0)
            return returnErreur(res, next, Erreur.echecDelete);
        return returnSucces(res, next, { message: 'Événement supprimé', ok: true });
    } catch (error) {
        if (error instanceof Error) {
            (alias) enum Erreur
            import Erreur
            console.error(`Erreur deleteEvenement : ${error.message}`);
            return returnErreur(res, next, Erreur.exception, `${error.name} = ${error.message}`);
        } else {
            return returnErreur(res, next, Erreur.exception, "");
        }
    }
}

```

```

export async function getEvenementYears(req: Request, res: Response, next: NextFunction) {
    console.log("getEvenementYears");
    const query = ` 
        SELECT DISTINCT ANNEE
        FROM TEVENEMENTS
        ORDER BY ANNEE DESC
    `;
    try {
        const [results] = await pool.execute(query);
        if (!Array.isArray(results))
            return returnErreur(res, next, Erreur.notArray);
        // Extraire les années depuis les résultats
        const annees = (results as any[]).map(row => row.ANNEE);
        return returnSucces(res, next, annees);
    } catch (error) {
        console.error(`Erreur getEvenementYears : ${error.message}`);
        return returnErreur(
            res,
            next,
            Erreur.exception,
            error instanceof Error ? error.message : ''
        );
    }
}

```

```

export async function getEvenementsSearchByYear(req: Request, res: Response, next: NextFunction) {
    const ordreAffichage = req.params.ordreAffichage;
    const selectedYear = req.params.annee;
    const rechercheEvenement = req.params.rechercheEvenement;

    if (!ordreAffichage)
        return returnErreur(res, next, Erreur.noParam, 'ordreAffichage');
    if (!selectedYear)
        return returnErreur(res, next, Erreur.noParam, 'annee');
    if (rechercheEvenement === undefined)
        return returnErreur(res, next, Erreur.noParam, 'rechercheEvenement');

    // Si rechercheEvenement vaut "*" (joker pour vide) on considère la chaîne vide
    const searchPattern = `${rechercheEvenement === '*' ? '' : rechercheEvenement}%`;

    const query = `
        SELECT CODE, DESIGNATION, TYPEEVENEMENT, ANNEE, VALEUR, COMMENTAIRES
        FROM TEVENEMENTS
        WHERE ANNEE = ?
        AND DESIGNATION LIKE ?
        + getOrderByFor(ordreAffichage);

    console.log("Query EvenementsSearchByYear:", query);
    try {
        const [results] = await pool.execute(query, [selectedYear, searchPattern]);
        if (!Array.isArray(results))
            return returnErreur(res, next, Erreur.notArray);
        const listeEvenements: Evenement[] = results as Evenement[];
        return returnSuccess(res, next, listeEvenements);
    } catch (error) {
        if (error instanceof Error) {
            console.error(`Erreur getEvenementsSearchByYear: ${error.name} - ${error.message}`);
            return returnErreur(res, next, Erreur.exception, ` ${error.name} = ${error.message}` );
        } else {
            return returnErreur(res, next, Erreur.exception, "");
        }
    }
}

```

Code Affichage web :

Pour Tout les Evenements :

```

import { useEffect, useState, useRef } from "react";
import { useNavigate } from "react-router-dom";
import { getFetch } from "../../../../../functions/request";
import { Evenement } from "../../../../../functions/interface/Evenements/Evenement";

import pictovoir from '../../../../../assets/svg/pictovoir.svg';
import pictoPencilsSquare from '../../../../../assets/svg/pictoPencilsSquare.svg';

import "./First_Layer.css";

function EvenementsFirstLayer() {
  const navigate = useNavigate();

  const savedFiltre = localStorage.getItem("EvenementsFirstLayer.filtreEtat");
  const savedRecherche = localStorage.getItem("EvenementsFirstLayer.rechercheValue");

  // Filtre par type (déjà présent) avec l'option "Tous"
  const [eventTypes, setEventTypes] = useState<{ CODE: number; DESIGNATION: string }[]>([]);
  const [selectedFilter, setSelectedFilter] = useState<string>(savedFiltre ? savedFiltre : "Tous");

  // Nouvel état pour les années
  const [annees, setAnnees] = useState<number[]>([]);
  const [selectedYear, setSelectedYear] = useState<string>("Tous");

  const [evenements, setEvenements] = useState<Evenement[]>([]);
  const [rechercheEvenement, setRechercheEvenement] = useState<string>(savedRecherche || "");

  const inputFiltre = useRef<HTMLInputElement>(null);

  useEffect(() => {
    if (inputFiltre.current) {
      inputFiltre.current.select();
      console.log("Input de recherche sélectionné");
    }
  }, []);

  function EvenementsFirstLayer() {
    const inputFiltre = useRef<HTMLInputElement>(null);

    useEffect(() => {
      if (inputFiltre.current) {
        inputFiltre.current.select();
        console.log("Input de recherche sélectionné");
      }
    }, []);

    // Charger la liste des types d'événement
    useEffect(() => {
      getFetch("api/admin/evenements/typeevenement")
        .then((data) => {
          if (Array.isArray(data)) {
            setEventTypes(data);
            // Ajout de l'option "Tous" n'est pas nécessaire dans le mapping puisque nous l'ajoutons dans le select
            if (selectedFilter === "Tous" && data.length > 0) {
              // Vous pouvez décider de sélectionner le premier type par défaut si besoin
              setSelectedFilter(data[0].CODE.toString());
            }
          } else {
            throw new Error("La réponse du serveur n'est pas conforme pour les types d'événement");
          }
        })
        .catch((err) =>
          console.error("Erreur lors de la récupération des types d'événement", err)
        );
    }, []);
  }
}

```

```
// Charger la liste des années
useEffect(() => {
  getFetch("api/admin/evenements/annees")
    .then((data) => {
      if (Array.isArray(data)) {
        setAnnees(data);
        // Si aucun filtre n'est encore défini, on peut par défaut sélectionner la première année
        if (selectedYear === "Tous" && data.length > 0) {
          setSelectedYear(data[0].toString());
        }
      } else {
        throw new Error("La réponse du serveur n'est pas conforme pour les années");
      }
    })
    .catch((err) =>
      console.error("Erreur lors de la récupération des années", err)
    );
}, [ ]);
```

```
// Fonction de mise à jour de la recherche
function handleChangeInput(value: string) {
  const cleanedValue = value.replace(/[\\\]/g, "");
  localStorage.setItem("EvenementsFirstLayer.rechercheValue", cleanedValue);
  setRechercheEvenement(cleanedValue);
  console.log("Recherche modifiée :", cleanedValue);
}

// Mise à jour du filtre par type
const filterEvenements = (filter: string) => {
  localStorage.setItem("EvenementsFirstLayer.filtreEtat", filter);
  setSelectedFilter(filter);
  console.log("Filtre type modifié :", filter);
};

// Mise à jour du filtre par année
const filterEvenementsByYear = (year: string) => {
  setSelectedYear(year);
  console.log("Filtre année modifié :", year);
};
```

```
// Récupérer les événements dès qu'un filtre ou la recherche change
useEffect(() => {
  const rechercheParam = rechercheEvenement.trim().length === 0 ? "*" : rechercheEvenement.trim();
  const ordre = "designation"; // ordre d'affichage par défaut

  let url = "";

  if (selectedYear === "Tous" && selectedFilter === "Tous") {
    // Aucun filtre n'est appliqué, on récupère tous les événements
    url = "api/admin/evenements/evenements";
  } else if (selectedYear !== "Tous") {
    // Si une année est sélectionnée, on utilise l'endpoint par année
    url = `api/admin/evenements/evenements/annee/${ordre}/${selectedYear}/${rechercheParam}`;
  } else if (selectedFilter !== "Tous") {
    // Sinon, si un type est sélectionné, on utilise l'endpoint par type
    url = `api/admin/evenements/evenements/${ordre}/${selectedFilter}/${rechercheParam}`;
  }

  console.log("Fetching événements avec URL :", url);
  getFetch(url)
    .then((dataEvenements) => {
      if (Array.isArray(dataEvenements)) {
        console.log("Données reçues :", dataEvenements);
        setEvenements(dataEvenements as Evenement[]);
      } else {
        throw new Error("La réponse du serveur n'est pas conforme");
      }
    })
    .catch((err) =>
      console.error("Erreur lors de la récupération des événements", err)
    );
}, [rechercheEvenement, selectedFilter, selectedYear]);
```

```

function EvenementsFirstLayer() {
<div>
  <div className="Entete">
    <p className="Titre">Liste des bénévoles</p>
    <div className="controls">
      <div className="filter-container">
        <div className="titreFiltre">Filtre :</div>
      </div>
      /* Filtre par type */
      <div className="EtiquetteEtInput">
        <span>Type Événement : </span>
        <select
          onChange={(e) => filterEvenements(e.target.value)}
          value={selectedFilter}
        >
          <option value="Tous">Tous</option>
          {eventTypes.map((type) => (
            <option key={type.CODE} value={type.CODE.toString()}>
              {type.DESIGNATION}
            </option>
          )))
        </select>
      </div>
      /* Filtre par année */
      <div className="EtiquetteEtInput">
        <span>Année : </span>
        <select
          onChange={(e) => filterEvenementsByYear(e.target.value)}
          value={selectedYear}
        >
          <option value="Tous">Tous</option>
          {annees.map((annee, idx) => (
            <option key={idx} value={annee.toString()}>
              {annee}
            </option>
          )))
        </select>
      </div>
    </div>
  </div>
}

```

```
        </select>
    </div>
    <div className="EtiquetteEtInput">
        <span> Désignation : </span>
        <input
            className="searchBar"
            type="text"
            autoFocus
            ref={inputFiltre}
            value={rechercheEvenement}
            placeholder="Rechercher un événement"
            onChange={(e) => handleChangeInput(e.target.value)}
        />
    </div>

    </div>
    <div className="add-btn-container">
        <button
            className="btn-edit"
            onClick={() => navigate(`/Evenement/${-1}`)}
        >
            &#10010; Créer un·e événement
        </button>
    </div>
</div>

<div className="table-container">
    <table>
        <thead>
            <tr>
                <th>
                    <div className="headBlock">
                        <div className="title">Désignation</div>
                    </div>
                </th>
            </tr>
        <tbody>
```

```
    <div className="headBlock">
      <div className="title">Désignation</div>
    </div>
  </th>
  <th>
    <div className="headBlock">
      <div className="title">Type</div>
    </div>
  </th>
  <th>
    <div clas (property) React.HTMLAttributes<HTMLDivElement>.className
      <div className="title">Année</div>
    </div>
  </th>
  <th>
    <div className="headBlock">
      <div className="title">Valeur</div>
    </div>
  </th>
  <th>Commentaires</th>
  <th className="case_centrees">Action</th>
</tr>
</thead>
<tbody>
  {evenements.map((evt, index) => (
    <tr key={index}>
```

```

        </thead>
        <tbody>
            {evenements.map((evt, index) => (
                <tr key={index}>
                    <td>{evt.DESIGNATION}</td>
                    <td>
                        {eventTypes.find((type) => type.CODE === evt.TYPEEVENEMENT)
                         ?.DESIGNATION || evt.TYPEEVENEMENT}
                    </td>
                    <td>{evt.ANNEE}</td>
                    <td>{evt.VALEUR}</td>
                    <td>{evt.COMMENTAIRES}</td>
                    <td className="case_centrees">
                        <img
                            src={pictoVoir}
                            alt=""
                            className="iconeAction"
                            title="Ouvrir la fiche de l'événement"
                            onClick={() => navigate(`/evenement/${evt.CODE}`)}
                        />
                        <img
                            src={pictoPencilSquare}
                            alt=""
                            className="iconeAction"
                            title="Modifier l'événement"
                            onClick={() => navigate(`/evenement/${evt.CODE}/${true}`)}
                        />
                    </td>
                </tr>
            )))
        </tbody>
    </table>
</div>
</div>

```

Page pour afficher l'événement que l'on souhaite (infos) :

```
src/app/components/pages/Evenement/infosLayer/EvenementInfosLayer.js
```

```
import { useEffect, useState } from "react";
import { useParams, useNavigate, } from "react-router-dom";
import [getFetch] from "../../functions/request";
import { Evenement } from "../../functions/interface/Evenements/Evenement";
import pictoPencilSquare from "../../assets/svg/pictoPencilSquare.svg";
// import pictoTrash from "../../assets/svg/pictoTrash.svg"; // Ajoutez votre icône de suppression
import "./Infos_Layer.css";

function EvenementInfosLayer() {
  const { code } = useParams<{ code: string }>();
  const [evenement, setEvenement] = useState<Evenement | null>(null);
  const navigate = useNavigate();
  // const navigate = useNavigate();

  useEffect(() => {
    if (code) {
      getFetch(`api/admin/evenements/evenement/${code}`)
        .then((data) => {
          if (data) {
            setEvenement(data as Evenement);
          } else {
            console.error("Aucun événement trouvé");
          }
        })
        .catch((err) => console.error("Erreur lors de la récupération de l'événement", err));
    }
  }, [code]);

  if (!evenement) {
    return <div>Chargement...</div>;
  }

  return (
    <div className="FormatTableauEvenement">
      <table>
```

```

function EvenementInfosLayer() {
    </div>
    <div className="part infoEvenement">
        <table>
            <thead>
                <tr>
                    <th>Code</th>
                    <th>Désignation</th>
                    <th>Type</th>
                    <th>Année</th>
                    <th>Valeur</th>
                    <th>Commentaires</th>
                    <th>Action</th>
                </tr>
            </thead>
            <tbody>
                <tr>
                    <td>{evenement.CODE}</td>
                    <td>{evenement.DESIGNATION}</td>
                    <td>{evenement.TYPEEVENEMENT}</td>
                    <td>{evenement.ANNEE}</td>
                    <td>{evenement.VALEUR}</td>
                    <td>{evenement.COMMENTAIRES}</td>
                    <td className="case_centrees">
                        <img
                            src={pictoPencilSquare}
                            alt="Modifier l'événement"
                            className="iconeAction"
                            title="Modifier l'événement"
                            onClick={() => navigate(`/evenement/${evenement.CODE}/true`)}
                        />
                    </td>
                </tr>
            </tbody>
        </table>
    </div>
}

```

Affichage des actions (à gauche, supprimer, éditer ...):

```
function EvenementFirstLayer() {
    console.log("==> EvenementFirstLayer mounted");
    const { code, forModif } = useParams<{ code: string; forModif?: string }>();
    console.log("URL params:", { code, forModif });
    const isForModif: boolean = forModif === "true";
    const navigate = useNavigate();

    const [evenement, setEvenement] = useState<Evenement | undefined>(undefined);
    const [oldEvenement, setOldEvenement] = useState<Evenement | undefined>(undefined);
    const [modif, setModif] = useState<boolean>(isForModif);
    const textareaRef = useRef<HTMLTextAreaElement>(null);

    // Chargement ou initialisation de l'événement
    useEffect(() => {
        console.log("useEffect: code =", code);
        if (code && isNaN(Number(code))) {
            const numericCode = Number(code);
            if (numericCode < 0) {
                console.log("Initialisation d'un nouvel événement");
                const newEvt: Evenement = {
                    CODE: -1,
                    DESIGNATION: "",
                    TYPEEVENTEMENT: 0,
                    ANNEE: new Date().getFullYear(),
                    VALEUR: 0,
                    COMMENTAIRES: ""
                };
                setEvenement(newEvt);
                setOldEvenement(newEvt);
                setModif(true);
            } else {
        
```

```

    } else {
      console.log(`Récupération de l'événement existant avec code ${numericCode}`);
      getFetch(`api/admin/evenements/evenement/${numericCode}`)
        .then((data) => {
          console.log("Réponse de getFetch:", data);
          if (data && typeof data === "object") {
            setEvenement(data as Evenement);
            setOldEvenement(data as Evenement);
          } else {
            console.error("Aucun événement trouvé ou réponse invalide");
          }
        })
        .catch((err) => {
          console.error("Erreur lors de la récupération de l'événement:", err);
        });
    }
  } else {
    console.error("Paramètre code invalide :", code);
  }
}, [code]);

// Mise à jour d'un champ de l'événement
const handleChangeInput = (value: string, key: keyof Evenement) => {
  console.log(`handleChangeInput: ${key} => ${value}`);
  if (evenement) {
    setEvenement({
      ...evenement,
      [key]: typeof evenement[key] === "number" ? Number(value) : value
    });
  }
};

```

```
// Validation minimale des entrées
const validateInputs = (): boolean => {
  if (!evenement) return false;
  const errors: string[] = [];
  if (!evenement.DESIGNATION || evenement.DESIGNATION.trim() === "") {
    errors.push("La désignation est obligatoire");
  }
  if (!evenement.TYPEEVENEMENT || Number(evenement.TYPEEVENEMENT) <= 0) {
    errors.push("Le type d'événement doit être une valeur positive");
  }
  if (!evenement.ANNEE || Number(evenement.ANNEE) <= 0) {
    errors.push("L'année doit être valide");
  }
  if (errors.length > 0) {
    console.error("Validation errors:", errors.join(" - "));
    return false;
  }
  console.log("Validation inputs ok");
  return true;
};

// Sauvegarder l'événement (création ou modification)
const save = (e: React.MouseEvent<HTMLButtonElement>) => {
  e.preventDefault();
  console.log("Appel de save()");
  if (!evenement) {
    console.error("evenement est undefined");
    return;
  }
  if (validateInputs()) {
```

```
    if (validateInputs()) {
      if (Number(code) < 0) {
        console.log("Appel de postFetch pour créer un nouvel événement", evenement);
        postFetch("api/admin/evenements/evenement", evenement)
          .then((resp) => {
            console.log("Réponse de postFetch:", resp);
            const newEvt = resp as Evenement;
            setOldEvenement(newEvt);
            setEvenement(newEvt);
            setModif(false);
            if (isForModif) {
              navigate(-1);
            } else {
              navigate(`/evenement/${newEvt.CODE}`);
            }
          })
          .catch((err) => console.error("Erreur lors de l'insertion de l'événement:", err));
      } else {
        console.log("Appel de patchFetch pour modifier l'événement", evenement);
        patchFetch(`api/admin/evenements/evenement/${evenement.CODE}`, evenement)
          .then((resp) => {
            console.log("Réponse de patchFetch:", resp);
            const updatedEvt = resp as Evenement;
            setOldEvenement(updatedEvt);
            setEvenement(updatedEvt);
            setModif(false);
            if (isForModif) {
              navigate(-1);
            } else {
              navigate(`/evenement/${updatedEvt.CODE}`);
            }
          })
          .catch((err) => console.error("Erreur lors de la modification de l'événement:", err));
      }
    }
  }
}
```

```
// Passage en mode modification (lecture seule => édition)
const clickModif = (e: React.MouseEvent<HTMLButtonElement>) => {
  e.preventDefault();
  console.log("Passage en mode modification");
  setModif(true);
};

const clickDelete = (e: React.MouseEvent<HTMLButtonElement>) => {
  e.preventDefault();
  if (!window.confirm("Confirmer la suppression de l'événement ?")) return;
  if (!evenement) {
    console.error("Erreur: evenement is undefined");
    return;
  }
  // Utilisez deleteFetch ou une fonction similaire pour appeler l'API
  deleteFetch(`api/admin/événements/événement/${evenement.CODE}`, {})
    .then(() => {
      navigate(`/événements`);
    })
    .catch((err) => console.error("Erreur lors de la suppression de l'événement:", err));
};

// Bouton Retour vers la liste des événements
const clickRetour = (e: React.MouseEvent<HTMLButtonElement>) => {
  e.preventDefault();
  console.log("Click retour");
  navigate(`/événements`);
};

if (!événement) return <div>Chargement...</div>;
```

```
return [
  <div className="EvenementPage">
    <form>
      <div className="part">
        <div className="element">
          <p className="Titre">Formulaire Événement</p>
          <div className="buttons-bar">
            <button type="button" className="btn-nav" onClick={clickRetour}>
              &#9665; Retour à la liste des événements
            </button>
            {modif ? null : (
              <button type="button" className="btn-edit" onClick={() => navigate(`/evenement/${-1}`)}>
                &#10010; Créer un nouvel événement
              </button>
            )}
          </div>
        </div>
      <div className="part">
        <div className="element">Code : {evenement.CODE}</div>
        <div className="element">
          <label htmlFor="designation">Désignation</label>
          <input
            id="designation"
            type="text"
            value={evenement.DESIGNATION}
            placeholder="Désignation de l'événement"
            onChange={(e) => handleChangeInput(e.target.value, "DESIGNATION")}
            readOnly={!modif}
            required
          />
        </div>
        <div className="element">
```

```
function EVENEMENTEVITELEAVENTURE() {
    <label htmlFor="typeEvenement">Type d'événement</label>
    <input
        id="typeEvenement"
        type="number"
        value={evenement.TYPEEVENEMENT}
        placeholder="Type d'événement"
        onChange={(e) => handleChangeInput(e.target.value, "TYPEEVENEMENT")}
        readOnly={!modif}
        required
    />
</div>
<div className="element">
    <label htmlFor="annee">Année</label>
    <input
        id="annee"
        type="number"
        value={evenement.ANNEE}
        placeholder="Année"
        onChange={(e) => handleChangeInput(e.target.value, "ANNEE")}
        readOnly={!modif}
        required
    />
</div>
<div className="element">
    <label htmlFor="valeur">Valeur</label>
    <input
        id="valeur"
        type="number"
        value={evenement.VALEUR}
        placeholder="Valeur"
        onChange={(e) => handleChangeInput(e.target.value, "VALEUR")}
        readOnly={!modif}
        required
    />
</div>
```

```
</div>
<div className="element">
  <label htmlFor="commentaires">Commentaires</label>
  <textarea
    id="commentaires"
    ref={textareaRef}
    value={evenement.COMMENTAIRES || ""}
    placeholder="Commentaires"
    onChange={(e) => handleChangeInput(e.target.value, "COMMENTAIRES")}
    readOnly={!modif}
  />
</div>
</div>
<div className="part">
  <div className="element buttons-bar">
    {modif ? (
      <>
        <button type="button" className="btn-nav" onClick={clickRetour}>
          &#9665; Retour
        </button>
        <button type="button" className="btn-save" onClick={save}>
          &#10003; Valider
        </button>
      </>
    ) : (
      <>
        <button type="button" className="btn-edit" onClick={clickModif}>
          &#9997; Modifier
        </button>
        <button type="button" className="btn-del" onClick={clickDelete}>
          &#10008; Supprimer
        </button>
      </>
    )}
  </div>
</div>
```

Screen du site :

Menu >

- [Les bénévoles](#)
- [Les opérations](#)
- [Les plannings](#)
- [Les événements](#)
- [Messagerie](#)
- [À traiter...](#)
- [Paramètres](#)
  
- [Déconnexion](#)

localhost:8100/admin/Evenements

HERCULE

Hello Gwenael !

Liste des bénévoles

Filtre : Type Événement : APV Année : 2025 Désignation : Rechercher un événement

+ Créer un événement

Désignation	Type	Année	Valeur	Commentaires	Action
Evenement 1	JB	2025	100	Premier événement	
Evenement 10	JB	2025	300	Dixième événement	
Evenement 2	BSV	2025	200	Deuxième événement	
Evenement 3	JB	2025	150	Troisième événement	
Evenement 8	APV	2025	180	Huitième événement	
teffffdff	BSV	2025	2	ttt	

Premier tri de recherche

Type Événement : APV Année : 2025 Désignation : Rechercher un événement

Tous

APV

BSV

JB

Exemple :

**Filtre :**

Type Événement : APV Année : Tous Désignation : Recherche

Désignation	Type	Année
Evenement 4	APV	2024
Evenement 8	APV	2025

Deuxième tri de recherche :

Type Événement : APV Année : 2025

Tous

2025

2024

Exemple :

**Filtre :**

Type Événement : Tous Année : 2024 Désignation : Recherche

Désignation	Type	Année
Evenement 4	APV	2024
Evenement 9	BSV	2024

## Recherche globale (par nom d'événement)

Filtre : Type Événement : APV Année : 2025 Désignation : 10

+ Créer un e événement

Désignation	Type	Année	Valeur	Commentaires	Action
Evenement 10	JB	2025	300	Dixième événement	

## Boutons



## Exemple page 9 eme evenemnt :

url :

localhost:8100/admin/evenement/9

HERCULE  
Hello Gwenaël !

Formulaire Événement

Détails de l'Événement

Code	Désignation	Type	Année	Valeur	Commentaires	Action
9	Evenement 10	3	2025	300	Dixième événement	

Désignation  
Evenement 10

Type d'événement  
3

Année  
2025

Valeur  
300

Commentaires  
Dixième événement

Modifier Supprimer

## Création d'un événement :

URL :

localhost:8100/admin/evenement/-1

 HERCULE  
Hello Gwenaël !

## Formulaire Événement

[!\[\]\(9903737e913933d8ce799b73542be4e6\_img.jpg\) Retour à la liste des événements](#)

Code : -1

**Désignation**  
Désignation de l'événement

**Type d'événement**  
0

**Année**  
2025

**Valeur**  
0

**Commentaires**  
Commentaires

[!\[\]\(18dbb04d2a029a220600e16a594549b4\_img.jpg\) Retour](#) [!\[\]\(12149ce9b24ccbdaf0a6cd74d2bb1d2f\_img.jpg\) Valider](#)

Edit de l'événement :

## Formulaire Événement

[!\[\]\(b95805d2b7fc6bef10576db0580256f1\_img.jpg\) Retour à la liste des événements](#)

Code : 9

### Désignation

Evenement 10

### Type d'événement

3

### Année

2025

### Valeur

300

### Commentaires

Dixième événement

[!\[\]\(c8609112592430a975878c9d50a0dbf2\_img.jpg\) Retour](#)

 Valider

## 11 : Implémentation d'une fonction de calcul des points de participation

### Route de l'API :

```
apiExterneRouter.post("/api/externe/participation/cumulPointsPersonne", (req, res, next) => getCumulPointsPersonne(req, res, next, curr))
```

```
export async function _getCumulPointsPersonne(personneCode: number): Promise<PersonneCumulePoint> {
    console.log('_getCumulPointsPersonne');
    console.log(`personneCode: ${personneCode}`);

    // D'abord, vérifions si la personne existe
    const checkPersonneQuery = `SELECT CODE FROM TPERSONNES WHERE CODE = ?`;
    ;

    const [personneExists] = await pool.execute(checkPersonneQuery, [personneCode]);

    if (!Array.isArray(personneExists) || personneExists.length === 0) {
        throw new Error(`Aucune personne trouvée avec le code ${personneCode}`);
    }

    const query = `SELECT
        d.NUMDOSSIER,
        p.NOMPrenom,
        COALESCE(SUM(part.VALEUR), 0) as TOTALVALEUR
    FROM TPERSONNES p
    INNER JOIN TDOSSEIERS d ON p.DOSSIER = d.CODE
    LEFT JOIN TPARTICIPATIONS part ON part.PERSONNE = p.CODE
    WHERE p.CODE = ?`;
    ;

    try {
        const [results] = await pool.execute(query, [personneCode]);

        if (!Array.isArray(results) || results.length === 0) {
            throw new Error(`Erreur lors de la récupération des données pour la personne ${personneCode}`);
        }

        const result = results[0] as any;
```

```

const [results] = await pool.execute(query, [personneCode]);

if (!Array.isArray(results) || results.length === 0) {
    throw new Error(`Erreur lors de la récupération des données pour la personne ${personneCode}`);
}

const result = results[0] as any;

if (!result.NUMDOSSIER || !result.NOMPrenom) {
    throw new Error(`Données incomplètes pour la personne ${personneCode}`);
}

const personneCumul: PersonneCumulePoint = {
    NUMDOSSIER: result.NUMDOSSIER,
    NOMPrenom: result.NOMPrenom,
    TOTALVALEUR: Number(result.TOTALVALEUR),
    COMMENTAIRE: null
};

console.log("Cumul des points calculé:", personneCumul);
return personneCumul;
} catch (error) {
    console.error("Erreur lors du calcul du cumul des points:", error);
    throw error;
}
}

```

```

export async function getCumulPointsPersonne(req: Request, res: Response, next: NextFunction, currentSession: CurrentSession) {
    try {
        console.log("==> Début getCumulPointsPersonne ==>");
        const { personnes } = req.body;

        if (!personnes || !Array.isArray(personnes)) {
            return returnErreurs(
                res,
                next,
                Erreur.noParam,
                "Le corps de la requête doit contenir un tableau de personnes"
            );
        }

        const resultsPromises = personnes.map(async (personne: { NUMDOSSIER: number, NOMPrenom: string }) => {
            try {
                const personneInfo = await _getPersonneByDossierNomPrenom(
                    personne.NUMDOSSIER,
                    personne.NOMPrenom
                );

                if (!personneInfo) {
                    return {
                        NUMDOSSIER: personne.NUMDOSSIER,
                        NOMPrenom: personne.NOMPrenom,
                        TOTALVALEUR: 0,
                        error: `Aucune personne trouvée avec le dossier ${personne.NUMDOSSIER} et le nom ${personne.NOMPrenom}`
                    };
                }

                const cumulPoints = await _getCumulPointsPersonne(personneInfo.CODE);
                return {
                    NUMDOSSIER: cumulPoints.NUMDOSSIER,
                    NOMPrenom: cumulPoints.NOMPrenom,
                    TOTALVALEUR: cumulPoints.TOTALVALEUR
                };
            }
        });
    }
}

```

```

        };
    } catch (error) {
        return {
            NUMDOSSIER: personne.NUMDOSSIER,
            NOMPRENOM: personne.NOMPRENOM,
            TOTALVALEUR: 0,
            error: `Erreur : ${error instanceof Error ? error.message : "Erreur inconnue"}`
        };
    });
}

const results = await Promise.all(resultsPromises);
return returnSuccess(res, next, results);

} catch (error) {
    console.error("==> Erreur générale dans getCumulPointsPersonne ==>");
    return returnError(
        res,
        next,
        Erreur.exception,
        `Erreur système: ${error instanceof Error ? error.message : "Erreur inconnue"}`

    );
}
}

```

## 12 : Développement d'un système de retour de données pour les points dans l'environnement externe

① [localhost:8100/externe/totalpoint](http://localhost:8100/externe/totalpoint)

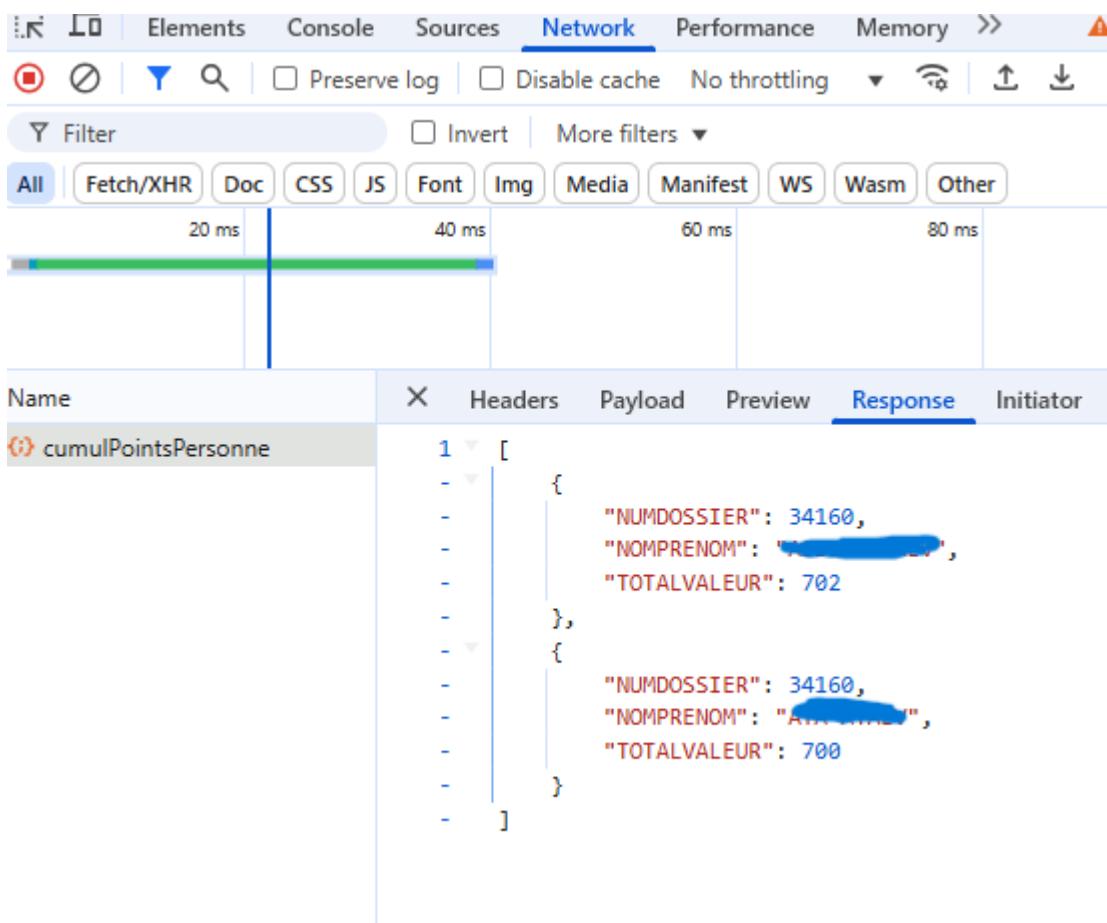
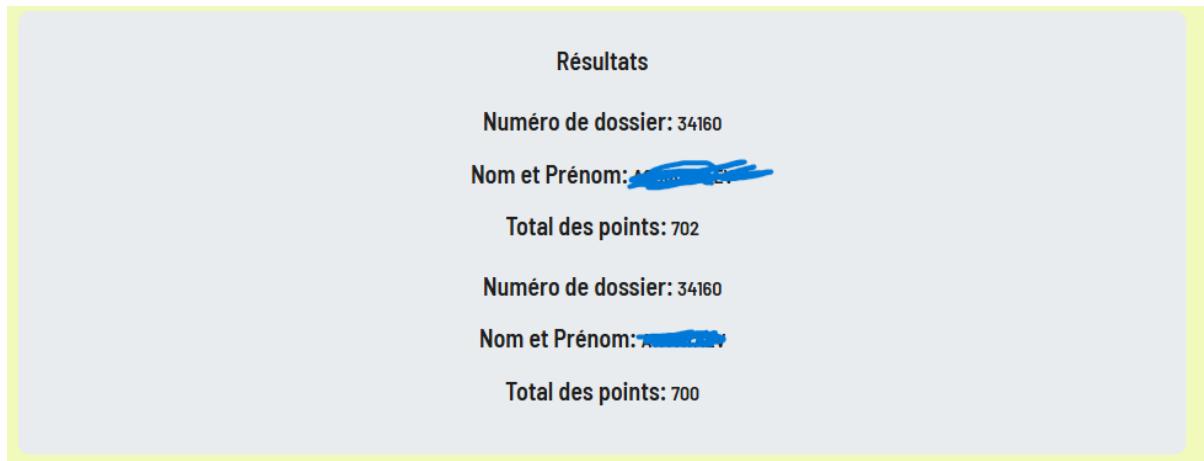


The screenshot shows a web page titled "Consultation des Points Cumulés". It displays two entries for dossier numbers 34160, each with a redacted name. Each entry includes a "Supprimer" button. At the bottom, there are "Ajouter une personne" and "Rechercher les points" buttons.

Numéro de dossier:	Nom et Prénom:	Action
34160	[REDACTED]	Supprimer
34160	[REDACTED]	Supprimer

**Ajouter une personne**    **Rechercher les points**

Retour :



Network

Preserve log

Invert

More filters

All Fetch/XHR Doc CSS JS Font Img Media Manifest WS Wasm Other

20 ms 40 ms 60 ms 80 ms

Name	Headers	Payload	Preview	Response	Initiator
cumulPointsPersonne				[{"NUMDOSSIER": 34160, "NOMPrenom": "A. [REDACTED]", "TOTALVALEUR": 702}, {"NUMDOSSIER": 34160, "NOMPrenom": "A. [REDACTED]", "TOTALVALEUR": 700}]	



Code pour l'affichage web de la page des points :

```
import { useState, useEffect } from 'react';
import { postFetch, postFetchAuth } from '../../../../../functions/request';
import './First_Layer.css';

interface AuthResponse {
  token: string;
}

interface PersonneRequest {
  NUMDOSSIER: number;
  NOMPRENOM: string;
}

interface CumulPoints {
  NUMDOSSIER: number;
  NOMPRENOM: string;
  TOTALVALEUR: number;
  error?: string;
}

const FirstLayer = () => {
  const [loading, setLoading] = useState<boolean>(true);
  const [error, setError] = useState<string | null>(null);
  const [tokenApi, setTokenApi] = useState<string | null>(null);
  const [cumulPoints, setCumulPoints] = useState<CumulPoints[]>([{}]);
  // Remplacer le state nomPrenom par un tableau
  const [personnes, setPersonnes] = useState<PersonneRequest[]>([
    {
      NUMDOSSIER: 0,
      NOMPRENOM: ''
    }
  ]);

  // Gestion de l'authentification
  useEffect(() => {
    const existingToken = localStorage.getItem('tokenApi');
    if (existingToken) {
      setTokenApi(existingToken);
      setLoading(false);
      return;
    }

    const requestBody = {
      identifiant: 'gbauche',
      password: 'Hercule35'
    };

    postFetchAuth(`api/cyberpunk/auth/login`, requestBody)
      .then(async (response: unknown) => {
        if (typeof response === 'object' && response !== null && 'token' in response) {
          localStorage.setItem('tokenApi', (response as AuthResponse).token);
          setTokenApi((response as AuthResponse).token);
        } else {
          throw new Error('Réponse invalide du serveur');
        }
      })
      .catch((err) => {
        console.error(`Erreur d'authentification:`, err);
        setError(err.message);
      })
      .finally(() => {
        setLoading(false);
      });
  }, []);
}
```

```
// Gestion de l'authentification
useEffect(() => {
  const existingToken = localStorage.getItem('tokenApi');
  if (existingToken) {
    setTokenApi(existingToken);
    setLoading(false);
    return;
  }

  const requestBody = {
    identifiant: 'gbauche',
    password: 'Hercule35'
  };

  postFetchAuth(`api/cyberpunk/auth/login`, requestBody)
    .then(async (response: unknown) => {
      if (typeof response === 'object' && response !== null && 'token' in response) {
        localStorage.setItem('tokenApi', (response as AuthResponse).token);
        setTokenApi((response as AuthResponse).token);
      } else {
        throw new Error('Réponse invalide du serveur');
      }
    })
    .catch((err) => {
      console.error(`Erreur d'authentification:`, err);
      setError(err.message);
    })
    .finally(() => {
      setLoading(false);
    });
}, []);
```

```

const handlePersonneChange = (index: number, field: keyof PersonneRequest, value: string) => {
    const newPersonnes = [...personnes];
    newPersonnes[index] = {
        ...newPersonnes[index],
        [field]: field === 'NUMDOSSIER' ? Number(value) : value
    };
    setPersonnes(newPersonnes);
};

const addPersonne = () => {
    setPersonnes([...personnes, { NUMDOSSIER: 0, NOMPRENOM: '' }]);
};

const removePersonne = (index: number) => {
    if (personnes.length > 1) {
        setPersonnes(personnes.filter((_, i) => i !== index));
    }
};

const fetchCumulPoints = async (e: React.FormEvent) => {
    e.preventDefault();
    setError(null);
    setCumulPoints([]);
    setLoading(true);

    if (personnes.some(p => !p.NOMPRENOM.trim() || p.NUMDOSSIER === 0)) {
        setError("Tous les champs sont requis");
        setLoading(false);
        return;
    }
}

```

```

try {
    const response = await postFetch('api/externe/participation/cumulPointsPersonne', {
        personnes: personnes
    });

    console.log('Response points cumulés:', JSON.stringify(response));
    setCumulPoints(response as CumulPoints[]);
} catch (err) {
    console.error('Erreur:', err);
    setError(err instanceof Error ? err.message : 'Erreur lors de la récupération des points');
} finally {
    setLoading(false);
}

if (!tokenApi) {
    return <div>Chargement...</div>;
}

```

```

if (!tokenApi) {
    return <div>Chargement...</div>;
}

return (
    <div className="total-points-containner">
        <h1>Consultation des Points Cumulés</h1>

        <form onSubmit={fetchCumulPoints} className="search-form">
            {personnes.map((personne, index) => (
                <div key={index} className="form-group">
                    <div className="input-group">
                        <div className="input-field">
                            <label htmlFor={`dossier-${index}`}>Numéro de dossier:</label>
                            <input
                                type="number"
                                id={`dossier-${index}`}
                                value={personne.NUMDOSSIER || ''}
                                onChange={(e) => handlePersonneChange(index, 'NUMDOSSIER', e.target.value)}
                                placeholder="Numéro de dossier"
                                required
                            />
                        </div>
                        <div className="input-field">
                            <label htmlFor={`personne-${index}`}>Nom et Prénom:</label>
                            <input
                                type="text"
                                id={`personne-${index}`}
                                value={personne.NOMPRENOM}
                                onChange={(e) => handlePersonneChange(index, 'NOMPRENOM', e.target.value)}
                                placeholder="Nom et prénom"
                                required
                            />
                        </div>
                </div>
            ))
        )
    </div>
)

```

```

        <button
            type="button"
            onClick={() => removePersonne(index)}
            className="remove-button"
        >
            Supprimer
        </button>
    </div>
)
)

<div className="buttons-container">
    <button
        type="button"
        onClick={addPersonne}
        className="add-button"
    >
        Ajouter une personne
    </button>
    <button
        type="submit"
        className="search-button"
        disabled={loading}
    >
        {loading ? 'Recherche...' : 'Rechercher les points'}
    </button>
</div>
</form>

```

```

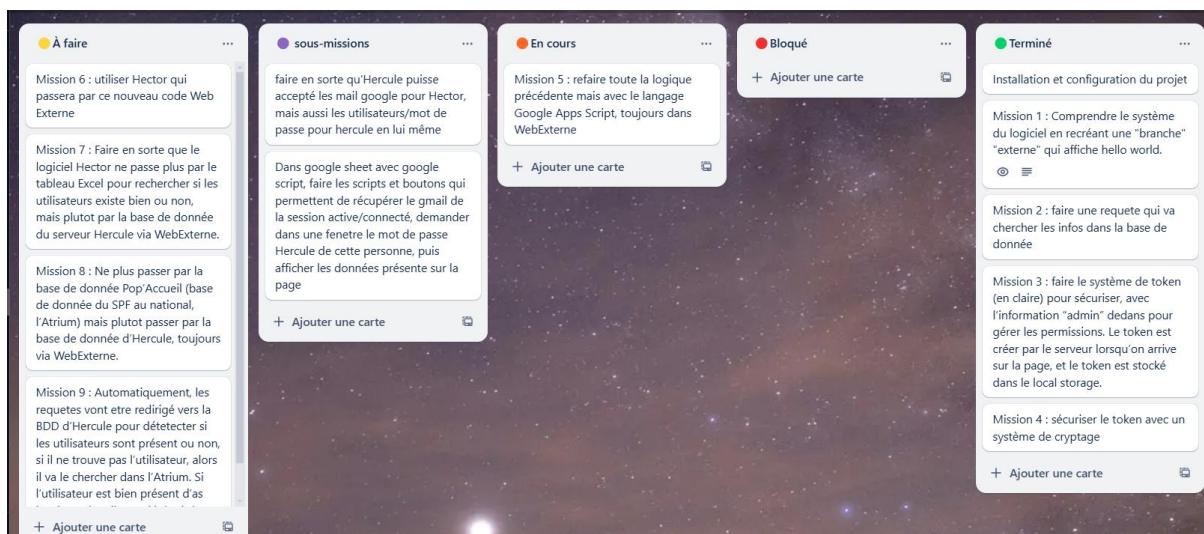
    {error && (
      <div className="error-message">
        {error}
      </div>
    )}

    {cumulPoints.length > 0 && (
      <div className="results-container">
        <h2>Résultats</h2>
        {cumulPoints.map((cumul, index) => (
          <div key={index} className="results-details">
            <p><strong>Numéro de dossier:</strong> {cumul.NUMDOSSIER}</p>
            <p><strong>Nom et Prénom:</strong> {cumul.NOMPRENOM}</p>
            {cumul.error ? (
              <p className="error-message"><strong>Erreur:</strong> {cumul.error}</p>
            ) : (
              <p><strong>Total des points:</strong> {cumul.TOTALVALEUR}</p>
            )}
          </div>
        )));
      </div>
    );
}

export default FirstLayer;

```

## Utilisation de Trello pour l'organisation des tâches :



## Organisation faite par mon tuteur pour les tâches :

TAF ULYSSE > HERCULE

	A	B	C	D	E	F	G	H	I	J	K	L
	next:	1 phase 1	0 Phase 2	0-Autres	1 à faire	2 à étudier	3 à tester	Version courante	#REF!	Version en prod	#REF!	
								priorité	1-forêt,5	Facile	Complexion	Tr A faire par
	N°	Thème	Module	Type	Quoi			Date MAJ		Commentaires / suivi		
4					Externaliser la fonction recupererInformationsAtrium(numeros) sous HERCULE :							
5	1	ATRIUM	INTERFACE	Technique	JB synchroniserPersonnes() => ULYSSEINTERFACE.validerInfoAtriumPersonne(...) => EXT.recupererInformationsAtrium(...)			1		A faire	Enzo	
6	2	ATRIUM	INTERFACE	Cas d'utilisation	Créer un cache des données dossiers atrium "PERSONNES" avec un timeout de 10 jours Prévoir un mode "force" qui ne va pas chercher dans le cache (mais le met à jour quand même si nécessaire)			1		A étudier	Enzo	
7	3	BDD	INTERFACE	Technique	Développer les fonctions de données d'ULYSSE dans HERCULE "ÉVÉNEMENTS" - (à définir) Prévoir et router les fonctions CRUD de la BDD vers Hercule			1		A étudier	Enzo	
8												
9												
10												

Organisation sur Notion pour stocker les notes et mettre à plat les tâches :



## Stage

- [Lettre de motivation](#)
- [CV](#)
- [Boîtes](#)
- [Image](#)
- [message](#)
- [Stage Secours Populaire](#)
- [cache Hercule/atrium](#)
- [Déplacement BDD Ulysse dans Hercule](#)

PASS :

La synchronisation se fait d'Atrium vers le cache :

Les données reçues d'Atrium doivent mettre à jour le cache. Le flux est unidirectionnel : Atrium → Cache

Faire la recherche uniquement sur les personnes ayant un flag false

- 1 / Recherche par DATENAISSANCE : Si une date de naissance correspond, mettre à jour automatiquement la personne concernée et définir son flag à true
- 2 / Recherche par DATENAISSANCE : Si deux dates de naissance correspondent mais sans correspondance de NOMPRENOM, définir le flag à false
- 3 / Recherche par DATENAISSANCE puis NOMPRENOM : Si deux DATENAISSANCE correspondent et un NOMPRENOM correspond à une personne, mettre à jour cette personne et définir son flag à true.

L'autre personne est mise à jour selon le cas 1

- 4 / Recherche par NOMPRENOM : Si le NOMPRENOM correspond mais pas la date de naissance, mettre à jour la personne et définir son flag à true
- 5 / Si ni la DATENAISSANCE ni le NOMPRENOM ne correspondent : insérer dans la base de données (POST) et définir le flag à true

Si après la boucle, le nombre de personnes dans le cache est supérieur au nombre de personnes dans le dossier Atrium concerné, supprimer toutes les personnes ayant un flag false

N
Notion de Enzo
Notion de Enzo
← → +
Stage / NOTE
Dernière modification : à l'instant Partager
...

### NOTE

Nouvelle table Tpersonne dans la base de donnée hercule  
ajouter toute les colonnes NON CALCULABLE (exemple : date prochain entretien)  
ajout time stamp

fonction interne dans l'API : `_getPersonnes()` : va récupérer la data sous la forme d'un tableau

transformer ce retour en tableau en retour json

horodatage de 3 mois :

- si en dessous de 3 mois et n'existe pas dans hercule —> ajouter dans hercule
- si au dessus de 3 mois passé —> retiré d'Hercule

Récupérer en priorité la data coté Hercule si elle existe, sinon la récupérer dans Atrium et l'ajouter dans la bdd Hercule

récupérer l'identifiant du dossier dans la page html  
créer nouvelle table Tevenement