

---

# Trombi GI

---

**Étudiants:**

Julio Cesar CAMARGO AMARO

Rodrigo Uriel ZUNIGA TELLEZ

**Responsable UV:** Ahmed LOUNIS

**Responsable TD:** Ghada Jaber

# CHAPITRE 1

## RESUME

Ce Rapport a comme but de créer une application web en utilisant React. L'application fait des requêtes vers l'API des utilisateurs dans la branche GI à l'UTC. On utilise uniquement 2 pages principaux, la page de recherche et la page où on affiche la liste des utilisateurs. Tous les autres pages sont des modaux (Voir Section 4).

Afin de comprendre le fonctionnement de l'application on va partir pour expliquer l'architecture du single Page application avec React.js. Après on va montrer la conception de notre application, comment on fait les requêtes vers l'API de l'université, comment a été construit la page web etc.

Ensuite, une fois compris le fonctionnement interne, on dédiera une section à l'installation et l'utilisation pour finalement donner une brève conclusion de notre travail.

# TABLE DES MATIÈRES

<b>1</b>	<b>Resume</b>	<b>2</b>
<b>2</b>	<b>Introduction</b>	<b>4</b>
2.1	Single Page Application . . . . .	4
2.1.1	Comment fonctionne une Single Page Application ? . . . . .	4
	Un langage : le JavaScript . . . . .	5
	Frameworks et Librairies . . . . .	5
2.2	React . . . . .	5
2.3	API REST . . . . .	6
2.3.1	Qu'est-ce qu'une API REST ? . . . . .	6
2.3.2	Qu'est-ce que le standard d'API REST ? . . . . .	6
<b>3</b>	<b>Conception</b>	<b>8</b>
<b>4</b>	<b>Installation</b>	<b>11</b>
<b>5</b>	<b>Utilisation</b>	<b>12</b>
5.1	Lancement de requêtes avec filtres . . . . .	13
5.1.1	Exemple 1 - Un seul filtre . . . . .	14
5.1.2	Exemple 2 - Deux filtres . . . . .	14
5.1.3	Exemple 3 - Trois Filtres . . . . .	14
5.1.4	Développement écologique . . . . .	15
<b>6</b>	<b>Conclusion</b>	<b>17</b>

Tout d'abord il faut expliquer tous les concepts de base qu'on utilise dans le projet. Parlons sur l'architecture des *Single Page Application*.

## 2.1 Single Page Application

Une *Single Page Application* est une application web accessible via une page web unique. Le but est d'éviter le chargement d'une nouvelle page à chaque action demandée, et de fluidifier ainsi l'expérience utilisateur. Deux méthodes existent pour ce faire : soit l'ensemble des éléments de l'application est chargé (contenu, images, CSS et JavaScript) dans un unique fichier HTML, soit les ressources nécessaires sont récupérées et affichées dynamiquement en fonction des actions de l'utilisateur.

### 2.1.1 Comment fonctionne une Single Page Application ?

En clair, pour afficher une page web, il faut un serveur et un client, généralement un ordinateur ou smartphone, qui communiquent entre eux via internet :

- dans le cas classique, le serveur exécute un script pour envoyer aux clients la page web à afficher.
- dans le cas d'une SPA, le serveur envoie aux clients le script qui permet de générer la page.

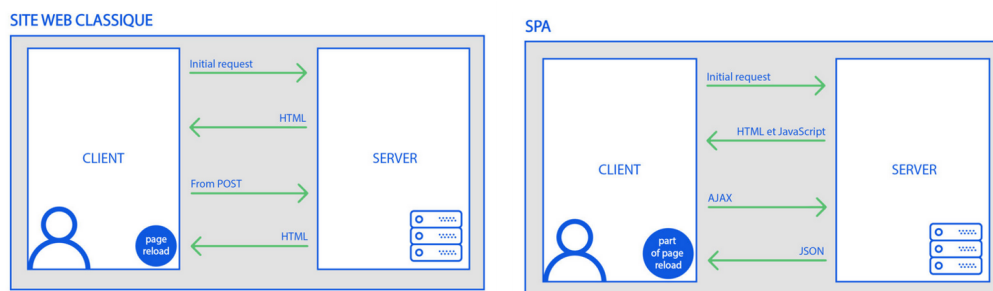


FIGURE 2.1 – Connexion Client

Une SPA est donc une application qui utilise le navigateur de l'utilisateur pour générer la page qu'il doit afficher. On dit que le code s'exécute côté client. Cela permet de naviguer sur toute l'application web sans rechargement de la page.

L'exécution du code côté client est le concept principalement utilisé par les SPA, et nous pouvons nous demander comment un script peut générer une page web en s'exécutant dans le navigateur ?

### Un language : le JavaScript

Pour expliquer plus en détail le concept des SPA (Single Page Application) il faut connaître les technologies qui leur permettent d'exister. Aujourd'hui, le meilleur langage de programmation qui peut s'exécuter côté client est le JavaScript. Auparavant, il était surtout utilisé pour ajouter des animations. Mais lors des années 2000 l'implémentation d'AJAX (Asynchronous JavaScript and XML) a totalement changé l'utilisation de ce langage.

En effet, l'AJAX permet au JavaScript de faire des requêtes HTTP de type asynchrone vers un serveur pour que celui-ci envoie les informations demandées. Ces informations sont ainsi affichées après le premier chargement de la page. Toutefois l'utilisation d'AJAX et de JavaScript seuls est fastidieuse, et complique le développement d'applications complexes.

### Frameworks et Librairies

JavaScript est un langage de programmation très utilisé dans le développement front-end lors de la création ou la refonte de site web ou d'application métier. Pour réaliser une SPA, les développeurs utilisent le plus souvent des frameworks et librairies. En effet, c'est un ensemble d'outils qui permet de faciliter l'usage du langage et ils ont plusieurs avantages comme :

- L'application d'une architecture plus ou moins stricte au code
- La réduction des temps de développement
- L'amélioration de la lecture et de la maintenabilité du code

## 2.2 React

Une fois expliqué l'architecture du *Single page Application* dans la section précédent on peut donc passer à expliquer qu'est que React.

React.js, est une techno Javascript, mais il ne s'agit pas d'un framework à proprement parler. En fait, il s'agit plus d'une librairie open source qui permet de construire des interfaces utilisateur dynamique. Et, de plus, cette librairie est maintenue un grand nombres de développeurs indépendants. Pour coder en React, vous devez avoir de bonnes bases en javascript, bien évidemment. C'est par ailleurs une techno très appréciée par développeurs web ces derniers temps. Elle permet de réaliser des applications web cross platform et ultra-performantes. Ce sont les équipes de Facebook qui sont à l'origine de cette plateforme. En effet, c'est en 2013 qu'est né React.

La syntaxe, basée sur la norme *ECMAScript* 6, elle permet de créer des composants JS pour

structurer votre application à votre convenance. Puis, ainsi, concevoir la manière dont vous allez développer. Ça fait bizarre quand on a commencé avec HTML/CSS/JS ! React s’est distingué par la simplicité de son starter kit et la souplesse de son architecture logicielle. La structure de projet est libre et vous permet de laisser libre court à votre imagination pour mettre en place le système le plus adapté à vos besoins.

## 2.3 API REST

Dans le cadre de notre projet, il est très important de savoir le concept d’une API REST puisque on fait toujours des requêtes vers l’API de l’UTC.

### 2.3.1 Qu’est-ce qu’une API REST ?

API est un acronyme pour “Application Programming Interface” ou Interface de programmation d’application en français. Il s’agit d’une interface permettant l’interaction entre différentes applications. Elle définit quels appels ou requêtes peuvent être réalisés et comment les réaliser : le format des données à utiliser, la structure de la réponse, les conventions à respecter etc.

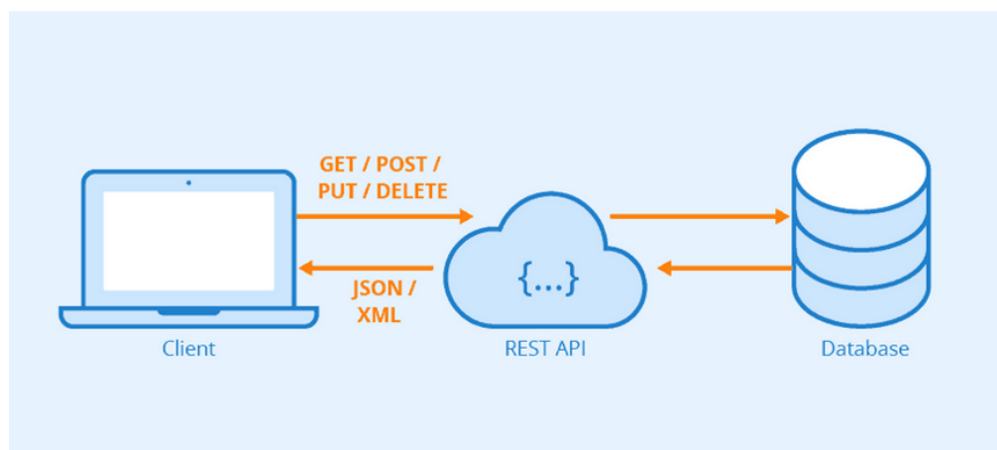


FIGURE 2.2 – API - REST

### 2.3.2 Qu’est-ce que le standard d’API REST ?

REST (pour REpresentational State Transfer) est une type d’architecture d’API qui fournit un certain nombre de normes et de conventions à respecter pour faciliter la communication entre applications. Les APIs qui respectent le standard REST sont appelées API REST ou API RESTful.

Le standard REST impose six contraintes architecturales qui doivent toutes être respectées par un système pour qu’il soit qualifiable de système RESTful. Le strict respect de ces six contraintes permet d’assurer une fiabilité, une scalabilité et une extensibilité optimales.

Les six principes de l’architecture REST sont :

- **Séparation CLient -Serveur** : Les responsabilités du côté serveur et du côté client sont séparées, si bien que chaque côté peut être implémenté indépendamment de l'autre. Le code côté serveur (l'API) et celui côté client peuvent chacun être modifiés sans affecter l'autre, tant que tous deux continuent de communiquer dans le même format. Dans une architecture REST, différents clients envoient des requêtes sur les mêmes endpoints, effectuent les mêmes actions et obtiennent les mêmes réponses.
- **Absence d'état de sessions (stateless)** : La communication entre client et serveur ne conserve pas l'état des sessions d'une requête à l'autre. Autrement dit, l'état d'une session est inclus dans chaque requête, ce qui signifie que ni le client ni le serveur n'a besoin de connaître l'état de l'autre pour communiquer.
- **Chaque requête est complète et se suffit à elle-même** : Pas besoin de maintenir une connexion continue entre client et serveur, ce qui implique une plus grande tolérance à l'échec. De plus, cela permet aux APIs REST de répondre aux requêtes de plusieurs clients différents sans saturer les ports du serveur. L'exception à cette règle est l'authentification, pour que le client n'ait pas à préciser ses informations d'authentification à chaque requête.
- **L'uniformité de l'interface** : Les différentes actions et/ou ressources disponibles avec leurs endpoints et leurs paramètres spécifiques doivent être décidés et respectés religieusement, de façon uniforme par le client et le serveur. Chaque réponse doit contenir suffisamment d'informations pour être interprétée sans que le client n'ait besoin d'autres informations au préalable. Les réponses ne doivent pas être trop longues et doivent contenir, si nécessaire, des liens vers d'autres endpoints.
- **La mise en cache** : Les réponses peuvent être mises en cache pour éviter de surcharger inutilement le serveur. La mise en cache doit être bien gérée : l'API REST doit préciser si telle ou telle réponse peut être mise en cache et pour combien de temps pour éviter que le client ne reçoive des informations obsolètes.
- **Architecture en couches** : Un client connecté à une API REST ne peut en général pas distinguer s'il est en communication avec le serveur final ou un serveur intermédiaire. Une architecture REST permet par exemple de recevoir les requêtes sur un serveur A, de stocker ses données sur un serveur B et de gérer les authentifications sur un serveur C.
- **Le code à la demande** : Cette contrainte est optionnelle. Elle signifie qu'une API peut retourner du code exécutable au lieu d'une réponse en JSON ou en XML par exemple. Cela signifie qu'une API RESTful peut étendre le code du client tout en lui simplifiant la vie en lui fournissant du code exécutable tel qu'un script JavaScript ou un applet Java.

# CHAPITRE 3

## CONCEPTION

Dans un premier temps on a commencé par créer une maquette de l'application (C'est vrai qu'à la fin le projet a évolué en fonction des besoins). Alors dans cette petite section on vous montrera la première phase de conception de l'application au niveau du design.

D'abord puisque React est un framework qui est dédié au single page applications pour notre application du trombinoscope on commence par la page suivante.

A screenshot of a web browser window titled "Firefox" displaying a web form titled "Trombi GI". The form contains three input fields: "Nom", "Prenom", and "Personnel" (which is a dropdown menu). Below these fields are three radio buttons labeled "All", "Heudiasyc", and "LMAC". At the bottom of the form is a green "Search" button.

FIGURE 3.1 – Maquette Index

Dans ce cas pour lancer une recherche on doit remplir au moins un des champs pour commencer la recherche, une fois la recherche a été lancé nous allons avoir comme résultat la page suivante :

Si on souhaite retourner vers la page de recherche il faudra cliquer sur le nom Trombi GI.



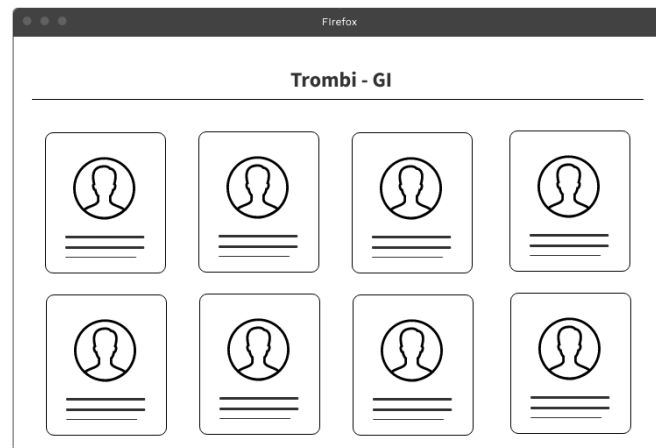


FIGURE 3.2 – Maquette List Page

Sur cette page, nous allons avoir tous les résultats liés à la recherche lancée précédemment, tous les résultats seront présentés sous la forme de cartes. Si on souhaite voir toutes les données concernées à une personne on devra cliquer sur sa carte, dès qu'on clique, un modal nous sera affiché comme l'exemple suivant :

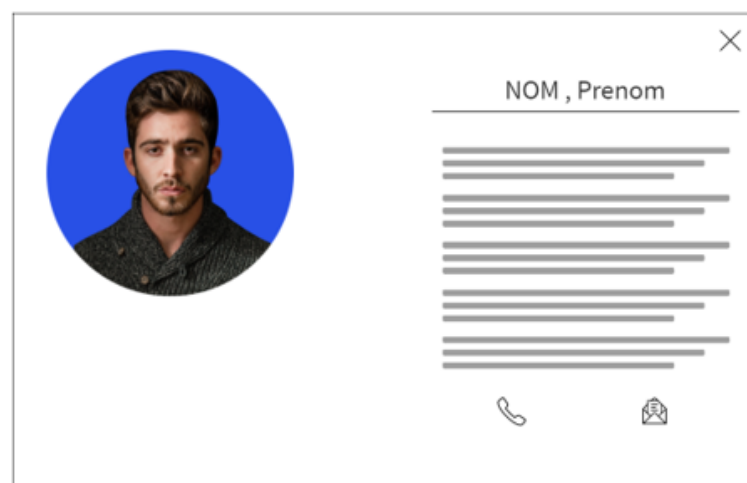


FIGURE 3.3 – Maquette Modal User

Sur ce modal on trouvera à gauche l'image de profil de l'utilisateur et à droite ses informations (nom, prénom, adresse mail, numéro de bureau, etc) et tout en bas on aura 2 icônes :

Le premier en forme de portable nous affichera un autre modal au-dessus avec un code QR, celui-ci pourra être scanné avec notre portable dans l'afin de faire un appel au sujet.

De même, le deuxième icône affichera un modal avec un code QR mais au moment de scanner celui-ci nous serons redirigé vers une autre fenêtre pour écrire un mail à l'individu.



FIGURE 3.4 – Maquette Modal Contact

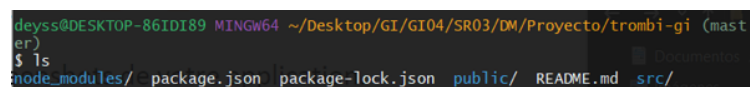
## CHAPITRE 4

## INSTALLATION

D'abord pour récupérer le projet nous devons exécuter la commande suivante dans le répertoire sur lequel on souhaite télécharger le projet.

```
git clone https://gitlab.utc.fr/jcamargo/trombi-gi.git
```

Une fois téléchargé vous devrez vous positionner sur le répertoire intern du projet et vous trouverez les fichiers et répertoires suivants :



```
deyss@DESKTOP-86IDI89 MINGW64 ~/Desktop/GI/GI04/SR03/DW/Proyecto/trombi-gi (master)
$ ls
node_modules/ package.json package-lock.json public/ README.md src/
```

FIGURE 4.1 – Git Clone

Suite à ça vous devrez écrire sur votre console la commande :

```
npm install
```

Cette commande vous permettra de récupérer tous les modules nécessaires pour lancer l'application react ainsi que tous les autres modules utilisés. Après cela suffit avec taper sur la console la commande .

```
npm start
```

# CHAPITRE 5

## UTILISATION

Cette dernière commande lance l'application sur votre navigateur par défaut. Vous devrez avoir sur votre navigateur la fenêtre suivante :

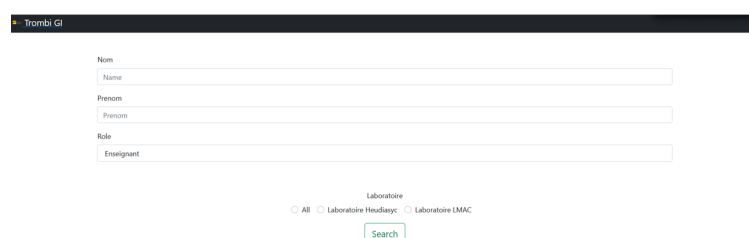


FIGURE 5.1 – Index Page

Une fois la page a été lancé on peut commencer à lancer des requêtes à l'API.

Si on souhaite trouver tous les personnes qui forment partie de GI (sauf étudiants) vous devez choisir par rôle le type "All" et sur laboratoire choisir "All" aussi ( ce dernière pas n'est pas nécessaire l'application un "All" comme valeur par défaut pour le champ Laboratoire ).

Pour lancer la requête, il suffit avec cliquer sur le bouton **Search**.

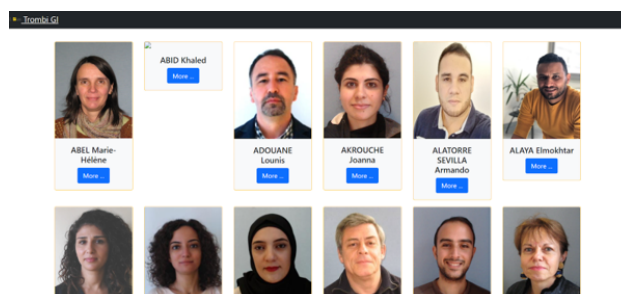


FIGURE 5.2 – List Utilisateurs

Sur cette nouvelle fenêtre on trouvera différentes cartes qui contiendront le nom et prénom de chaque individu. Pour voir les informations d'un certain utilisateur, vous pouvez cliquer sur le

bouton “More...”. Cette clique déclenche l’affichage d’un modal qui contiendra toutes les informations relatives à l’individu.

Prenons par exemple le professeur **LAKHLEF Hicham**.

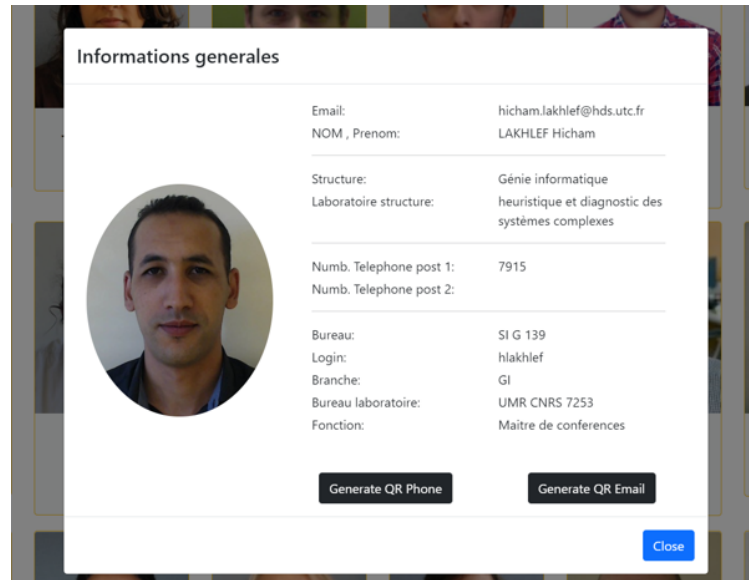


FIGURE 5.3 – Afficher un utilisateur par nom

Ensuite, nous avons l’option de générer un qr qui pourra être scanné soit pour envoyer un mail, soit pour se mettre en contact avec son bureau. Pour obtenir un des 2 Qrs on ne devra que cliquer sur le bouton “Generate QR Phone” ou “Generate QR Email”. Une fois un des deux boutons est cliqué il émerge un nouveau modal qui se superposera au premier modal.

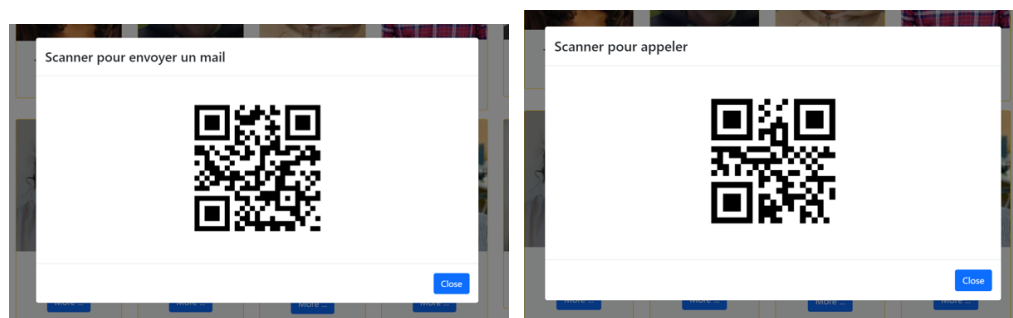


FIGURE 5.4 – Contact QR code

Dès que le QR est affiché, on ne doit que le scanner avec la caméra de nôtre portable.

## 5.1 Lancement de requêtes avec filtres

Dans le cadre de ce projet on a implémenté des filtres qui peuvent être lancés de manière simultanés ou séparés.

### 5.1.1 Exemple 1 - Un seul filtre

Par exemple, si on souhaite récupérer tous les utilisateurs qui forment partie du laboratoire LMAC on ne doit cliquer que sur l’option Laboratoire **LMAC** et le Role **All**.

FIGURE 5.5 – Filtre LMAC Recherche

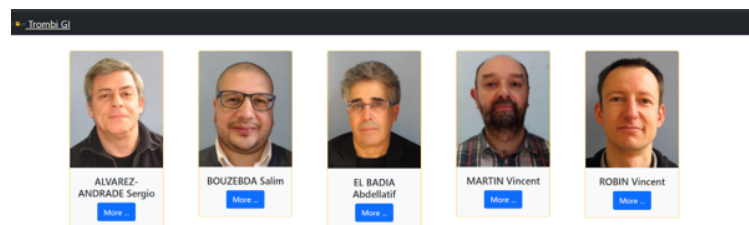


FIGURE 5.6 – Filtre LMAC Resultat

### 5.1.2 Exemple 2 - Deux filtres

Si on souhaite trouver tous les utilisateurs qui sont enseignants et qui forment partie du laboratoire LMAC , nous ne devons que mettre comme Rôle l’option **Enseignant** et sur laboratoire cliquer **Laboratoire LMAC**.

FIGURE 5.7 – Filtre LMAC Recherche 2 Filtres

### 5.1.3 Exemple 3 - Trois Filtres

Maintenant on souhaite trouver les utilisateurs qui ont comme nom “MARTIN” , qui sont enseignants et qui forment partie du laboratoire LMAC. Sur la page principal du site nous devons saisir les informations suivantes :

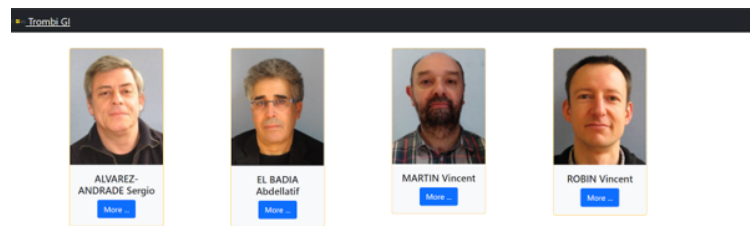


FIGURE 5.8 – Filtre LMAC Resultat 2 Filtres

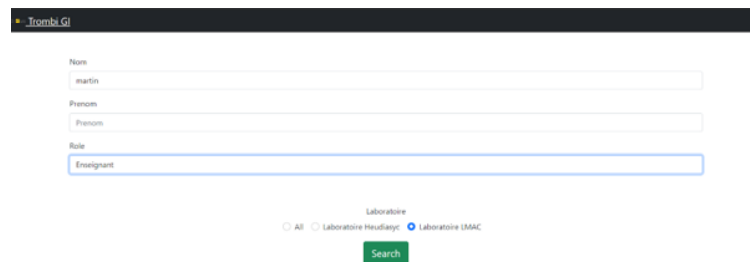


FIGURE 5.9 – Filtre LMAC Recherche 3 Filtres

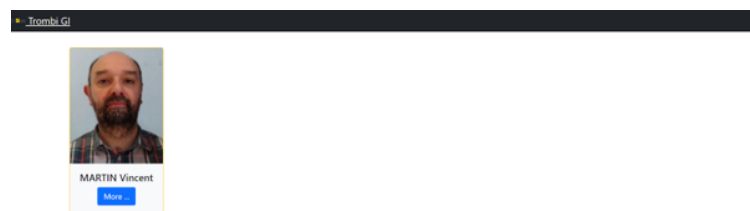


FIGURE 5.10 – Filtre LMAC Resultat 3 Filtres

**NB : Tous ces mêmes filtres marchent aussi sur le laboratoire Heudiasyc ainsi que pour tout l'ensemble des individus de GI.**

#### 5.1.4 Développement écologique

Grâce à l'utilisation du plugin EcoIndex on peut certainement dire que notre site est éco-responsable (notation A).

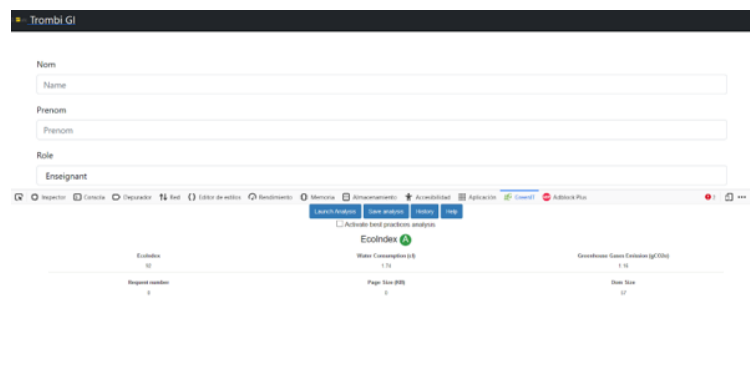


FIGURE 5.11 – Eco Test



## CHAPITRE 6

CONCLUSION

## TABLE DES FIGURES

2.1	Connexion Client . . . . .	4
2.2	API - REST . . . . .	6
3.1	Maquette Index . . . . .	8
3.2	Maquette List Page . . . . .	9
3.3	Maquette Modal User . . . . .	9
3.4	Maquette Modal Contact . . . . .	10
4.1	Git Clone . . . . .	11
5.1	Index Page . . . . .	12
5.2	List Utilisateurs . . . . .	12
5.3	Afficher un utilisateur par nom . . . . .	13
5.4	Contact QR code . . . . .	13
5.5	Filtre LMAC Recherche . . . . .	14
5.6	Filtre LMAC Resultat . . . . .	14
5.7	Filtre LMAC Recherche 2 Filtres . . . . .	14
5.8	Filtre LMAC Resultat 2 Filtres . . . . .	15
5.9	Filtre LMAC Recherche 3 Filtres . . . . .	15
5.10	Filtre LMAC Resultat 3 Filtres . . . . .	15
5.11	Eco Test . . . . .	16