Université de Caen Normandie
UFR des Sciences
Département Informatique

3ème année de licence d'informatique

Rapport de stage

Développement du site web d'un logiciel.

Laboratoire de recherche GREYC

Tolga SAHIN

Maître de stage : Emmanuel Giguet
Tuteur du projet ou du stage : Emmanuel Giguet, Christophe Rosenberger

JURY : Emmanuel Giguet, Bruno Zanuttini

Année universitaire : 2020 / 2021
Stage effectué du 01/04 au 28/05

Soutenu le mardi 08 juin 2021

Table des matières

1. Introduction **1**

2. Environnement de stage **2**
2.1. Présentation du laboratoire GREYC 2
2.2. Présentation de l'équipe de travail 2
2.3. Environnement de travail 2
2.4. Outils utilisés 3

3. Travail réalisé **4**
3.1. Site web Beyond Rankings 4
3.1.1. Préambule : configuration du serveur 4
3.1.2. Les missions 5
3.1.3. Analyse de l'architecture du site 5
3.1.4. Analyse de la structure du site 5
3.1.5. Filtre de Widgets 7
3.1.6. Widget avec un slider et histogramme. 8
3.2. Création de pages 8
3.2.1. Page d'accueil 9
3.2.2. Page de nouveautés 9
3.2.3. Page de structures 10
3.2.4. Page de chercheur 10
3.2.5. Page de crédits 11
3.3. Page de profil 12
3.3.1. Structure de la page de profil 12
3.3.2. Architecture pour la page de profil 12
3.3.3. Conception d'une API REST 13
3.3.4. Les requêtes sur l'API 13
3.3.5. Génération de la page 14
3.3.6. Card de la personne 15
3.3.7. Datatable 15
3.3.8. Graph de réseaux entre les chercheurs 15
3.3.9. Chart doughnut sur les années. 16

4. Conclusion **17**

I. Annexes **I**

Remerciements

Je tenais à remercier mes tuteurs M. Emmanuel Giguet et M. Christophe Rosenberger, pour m'avoir offert cette opportunité. Ils m'ont offert un investissement et une bienveillance exemplaire, de leur aide et support régulier afin de réaliser leurs différentes missions.

Ainsi que M. Yann Mathet et la structure universitaire de Caen, qui nous ont permis d'effectuer ce stage en fin d'année de licence informatique à Caen.

Je remercie également les stagiaires du GREYC avec lesquels nos échanges furent constructifs, dans le partage et la bonne humeur.

1. Introduction

Dans ce rapport de stage seront présentés les différentes étapes de mon travail sur Beyond Rankings : une application web developpé par le GREYC. Beyond Rankings permet de récuperer les collaborations entre chercheurs au sein d'un laboratoire. De nombreux widgets permettent de filtrer et d'afficher des données de manière ergonomique. Par exemple, on affiche un graphe de collaboration entre chercheurs et une carte des pays collaboratifs en fonction des filtres actifs. On m'a proposer plusieurs missions tout au long de ce stage. J'expliquerai durant ce rapport le déroulement nécessaire à l'accomplissement de mes missions.

2. Environnement de stage

2.1. Présentation du laboratoire GREYC

Le GREYC (Groupe de Recherche en Informatique, Image et Instrumentation de Caen) est un laboratoire de recherche en sciences du numérique. C'est une unité de recherche mixte associée au CNRS, à l'Université de Caen Normandie (UNICAEN) et à l'École Nationale Supérieure d'Ingénieurs de Caen (ENSICAEN). Il est membre de l'école doctorale MIIS, du réseau d'intérêt normand « Normandie digitale », et est partenaire de la fédération de recherche Normastic.

2.2. Présentation de l'équipe de travail

Lors de ce stage, mon équipe de travail au sein du GREYC a été le SAFE (Security, Architecture, Forensics, biomEtrics). L'équipe a des activités de recherches autour de la sécurité (biomètrie, architecture et modèles, forensics...) et en parallèle réalise le développement de certaines applications. Par exemple, j'ai pu voir Greyc Star qui permet la reconnaissance d'une star ressemblant le plus à vous en fonction d'une analyse faciale.

M. Christophe Rosenberger et M. Emmanuel Giguet sont membres de cette équipe. Ils sont aussi les chefs et développeurs du projet : Beyond Rankings. Donc ce sont eux que j'ai principalement côtoyés durant mon stage.

2.3. Environnement de travail

FIGURE 2.1. – Logo Discord

Le stage s'est deroulé entièrement en distanciel à l'aide du logiciel Discord. Discord nous a permis de communiquer et d'organiser le déroulement de notre stage.

Le dépot du projet est sur la forge du GREYC (un dépot git). Je le mettais à jour le plus souvent possible pour que le travail fait puisse être voir par les tuteurs.

2.4. Outils utilisés

Les outils utilisés lors de ce stage sont :
Apache : ou aussi appellé Apache HTTP Server est un serveur web (HTTP) qui consiste à établir une connexion fluide et securisé (client-server) entre différent fichiers.

Php : PHP est un langage de programmation utilisé pour le backend de notre application web.

Javascript : Javascript est aussi un langage de programmation (orienté script) qui a été utilisé un peu pour du backend et du frontend. Sous sa forme de module en objet, il a été utilisé pour concevoir des widgets.

Jquery : Jquery est une librairie javascript qui facilite l'écriture de code javascript côté client. Cette librairie intègre aussi certaines fonctionnalités si elle est importé avec des sous librairies comme ci-dessous.

DataTable : Une sous librairie JQuery qui permet la transformation ou la définition d'un tableau dans le DOM (bien structuré) en DataTable : (un tableau qui dispose de fonctionnalités : pagination, barre de recherche, filtres...)

Json : Format de données utilisé pour stocker les collections de données HAL et les parser côté backend.

D3.js : Librairie graphique utilisé pour des affichages complexes en fonction de données. Cette librairie peut aussi manipuler le DOM à sa manière, en ajoutant ou transformant des éléments disposant de caractéristiques propre aux fonctions intégrés dans d3.

Bootstrap (v4 et v5 Beta) : Librairie graphique utilisé pour la manipulation de style plus avancé et ergonomique.

Chart.js : Librairie graphique proche de d3.js qui permet aussi des affichages complexes en fonction de données.

3. Travail réalisé

3.1. Site web Beyond Rankings

3.1.1. Préambule : configuration du serveur

Avant toute chose, il a fallu configurer un serveur web sur ma machine. Lors de mon stage, j'ai réalisé deux installations d'Apache et de PHP.

Une avec WAMPServer qui intègre php, il faut juste bien veiller à configurer le fichier httpd.conf avec les règles de réecriture propre au projet.

Une autre installation avec une VM Linux dans laquelle, j'ai suivi un tutoriel via une vidéo de 37 minutes (https ://www.youtube.com/watch ?v=TrLAx27Npns) pour réaliser une configuration fraiche d'un serveur web.

Lorsque le serveur est prêt, on réalise à l'intérieur du répertoire "www" d'apache. Un lien symbolique qui pointe sur le répertoire web dans notre projet. (Commande mklink sur windows ou ln -s sur Linux)

On veille à ce que chaque règle de réecriture soit bien fonctionnel afin que les URL du site soit réecrite correctement. Donc soit directement dans la configuration du serveur, soit avec le fichier .htaccess dans le dossier web.

Enfin, on configure le fichier 'aura.ini' à l'intérieur du projet avec les paramètres adéquats propre à chaque configuration web. Ce fichier permet de paramétrer l'URL du site et le répertoire vers les collections de laboratoires (les données).

3.1.2. Les missions

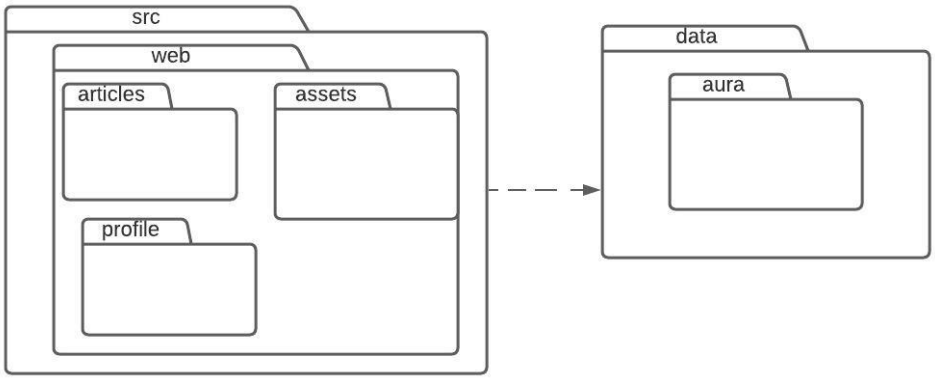
Lors de ma première réunion, mon sujet de stage a été mis au clair sur le fait qu'il sera principalement orienté sur de la vue (du front-end). Je devais travailler sur du style. J'avais aussi le choix entre trois missions, soit travailler sur des widgets. Migrer la version de d3.js de la v3 à la v6. Coder le front-end en Vue.js ou react. J'ai choisi de travailler avec la mission sur les widgets. On m'a proposer de concevoir un filtre de widgets, puis un filtre slider+histogramme animé sur les années de documents. Côté style et un peu backend, j'ai voulu changer certaines choses du côté des pages du site web.

Sur la V1 du projet, j'ai pas mal modifier le style du dashboard. Le passage à la v2 a fait que mes modifications n'été plus trop en accord avec le projet surtout à cause de Bootstrap5 qui posait des soucis. J'avais décidé de revoir un peu le layout, le header, les div. On m'a aussi proposé d'ajouter du zoom/unzoom sur les svg.

Enfin, j'avais une mission qui me proposait de trouver un chercheur et lui générer une page de profil. Pour cette partie, je partais d'une base vide. Je me suis permis de coder du backend (parser, API rest...) car j'avais pas vraiment les données en main. Sur cette partie, j'ai aussi pris le risque d'apprendre et d'utiliser des outils dont j'ai pas l'habitude. (requêtes asynchrones, bootstrap, chart.js...).

3.1.3. Analyse de l'architecture du site

Brièvement, l'architecture du site web se décompose ainsi :



— Avec web, le package principal du site web.
— articles : le package pour les pages d'articles.
— profile : le package pour la page de profil.
— assets : Package pour stocker tout ce qui est javascript, le CSS et les modules.
— data : Package pour stocker les fichiers de données (les collections HAL).

3.1.4. Analyse de la structure du site

Mon tuteur ayant déjà eu une idée précise de la structure. J'ai suivi ce qu'il avait fait en améliorant ce que je pouvais.

Donc nous avions un site de ce type :

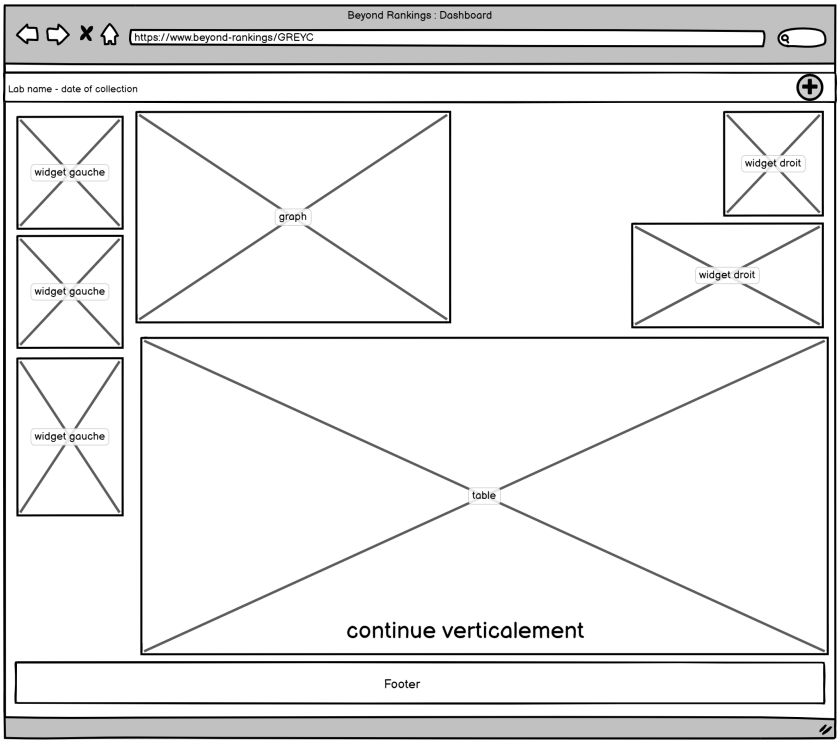


FIGURE 3.1. – Wireframe de la structure du Dashboard en v1

Le passage du projet en v2 + des modifications feront que la structure deviendra comme ca :

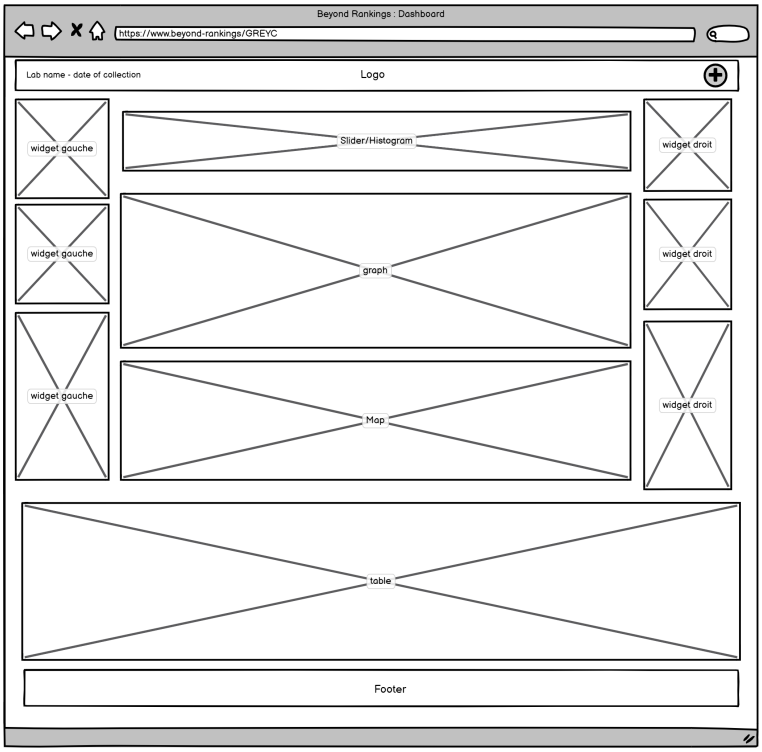


FIGURE 3.2. – Wireframe de la structure du Dashboard actuellement.

Le layout suit la même logique d'avoir 3 colonnes principales et un tableau en dessous. On a minimiser l'espace utilisé du côté du bas de la page. (le tableau ne continue plus verticalement). On a maximiser l'espace non utilisé du côté central de la page.

3.1.5. Filtre de Widgets

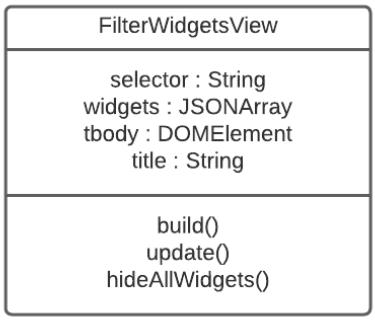


FIGURE 3.3. – UML de la classe pour filtrer les widgets.

Le filtre de widgets est un widget que j'ai conçu qui permet comme son nom l'indique de filtrer la présence ou non de widget dans la page. Je me suis inspiré de la structure des autres modules qui contiennent une construction avec comme paramètre un sélécteur. Puis une méthode build() pour construire l'élément DOM et update() pour la mise à jour événementiel.

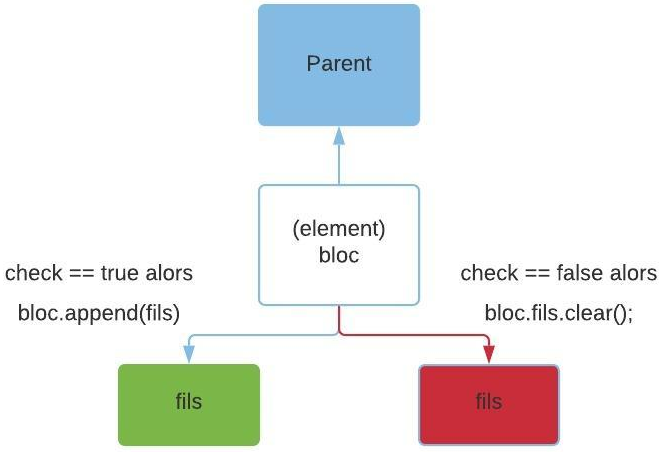


FIGURE 3.4. – Arbre schématisant l'idée générale du filtrage de widget.

J'ai décidé de construire un attribut avec le type JSON qui permettra de stocker des JSONobject avec comme attribut l'objet d'un widget et un autre pour son état (présent pour true et inactif pour false). Si ce widget est présent alors on le place dans son bloc respectif dans le DOM. Au contraire si il est inactif, on supprime sa présence du DOM. On peut mettre un EventListener "change" sur la liste de checkbox et ainsi modifier l'état des blocs.

(PS : En cours de route, on a passé le projet en v2, je n'ai pas activé le filtre de widgets dû à une nouvelle mise en page sur cette version qui rendait mal. J'ai pas passé trop de temps dessus mais ca devrait être fixable en revoyant le style des layouts des panels de gauche de la V2)

3.1.6. Widget avec un slider et histogramme.

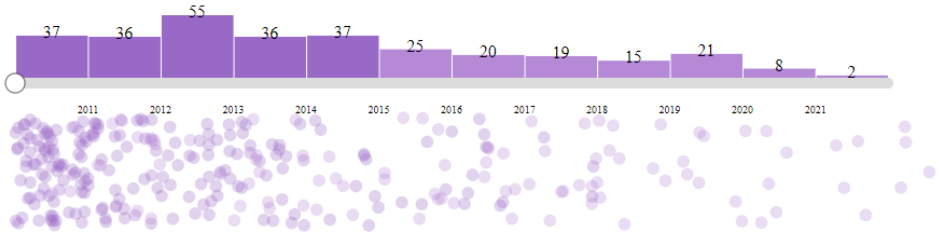


FIGURE 3.5. – Slider et histogramme sur les années des documents.

Ce widget permet l'affichage d'un slider sur les années ainsi que d'un histogramme et de cercles animées en fonction du nombre de documents par année.

Les dates sont parser du format d-b-y à un format plus textuel 02 Janvier 2021 00 :00 :00. Ce sera le format attendu pour la réalisation de cet affichage avec d3. Donc, nous avions seulement les années donc on peut initialiser nos dates au 02 janvier pour éviter une ambiguïté avec le 01 janvier de d3.js.

Puis, on peut déjà calculer l'intervalle de notre slider et histogramme. On calcule aussi la médiane du nombre de documents pour pouvoir avoir une logique de 2 couleurs (une basse et une forte) en fonction de la médiane des données.

```
1  let firstYear = data[0].year;
2  let lastYear = data[data.length - 1].year;
3  let sum = data.reduce((sum, val) => (sum += val.count), 0);
4  this.colorMedian = parseInt(sum / data.length);
```

Pour la partie d3.js, on configure l'échelle de l'histogramme avec scaleLinear() et scaleTime(). On initialise l'histogramme et les blocs nécessaire à la répresentation (svg, g...). Ainsi que l'espace d'un shape pour le "plot" qui sera la zone où nous dessineront nos cercles. On déclare l'histogramme avec les années. Pour le reste, on paramètre l'affichage de l'histogramme et du slider.

La méthode drawPlot permet l'affichage des cercles. D'où la raison pour laquelle nous la lions à la méthode updateSlider qui permet de mettre à jour l'évènementiel et les couleurs des barres de l'histogramme. La méthode eventHandler(activeYears) met à jour les années actives du projet. Un appel au controleur du projet mettra à jour tous les autres widgets.

On lie la méthode updateSlider avec un équivalent d'EventListener en d3 qui est nommé .on pour le paramètre "start drag".

3.2. Création de pages

J'avais décidé de développer le site avec de nouvelles pages pour des raisons d'ergonomie. Dans cette idée-là on pourrait séparer les idées et avoir une plus grande ouverture pour de future fonctionnalités.

Donc le site qui n'été au départ constitué que d'une page d'accueil, sera organisé de tel sorte à contenir de nouvelles pages.

Vu que je découvrais Bootstrap, j'ai voulu sortir de mes aises et prendre le risque d'utiliser principalement cette librarie. Ce qui m'a permis de découvrir de nombreuses fonctionnalités.

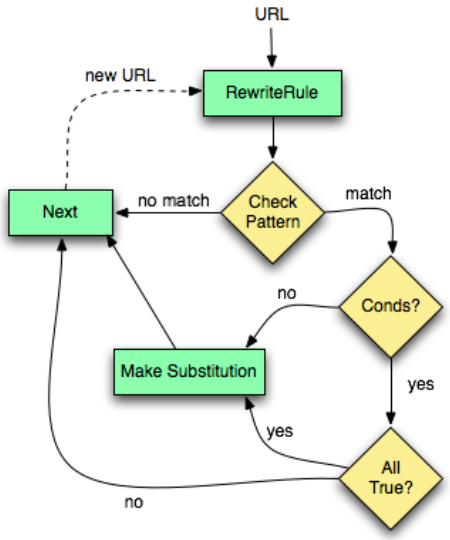


FIGURE 3.6. – Génération d'une URL avec une RewriteRule

Je me suis inspiré de la logique du projet qui consistait à ce que chaque page pour se générer utilise une sorte de "routing" via l'index.php, c'est-à-dire que la création de la page s'effectue en fonction d'un paramètre GET ou SERVER. Puis on inclus une template en fonction de certaine données propre à une page. Puis, c'est-là où le .htaccess va intervenir pour éviter d'avoir une URL sous une forme longue et parametrée. On la réecrit de telle manière à ce que soit plus ergonomique et lisible.

Ci-dessous seront cités brièvement le contenu essentiel des pages réalisés notamment grace à Bootstrap.

3.2.1. Page d'accueil

La page d'acccueil contient la template général (header avec logo, navbar, footer ...) Le contenu principal est un Carousel : qui est une liste défilante d'image ou vidéo accompagné si on le souhaite d'un titre et d'une description. Cette liste défile automatiquement environ toutes les 5 secondes. On peut aussi la défiler par nous-même grâce aux flèches.

Un carousel se concevoit grace à la classe carousel slide sur un div. Un sous div avec la classe carousel-item représente un item. Un lien avec la classe carousel-control-prev représente la flèche de gauche et avec le suffixe next la flèche de droite. Dans un item, on peut ajouter un sous bloc avec une classe carousel-caption si l'on veut légender, ou ajouter une description.

3.2.2. Page de nouveautés

La page de nouveautés est la page d'accueil de la seconde version du projet. Le contenu de la page est une liste formé d'une sous template d'item. Puis on génére la template général de la page.

Chaque item décrivent introductivement l'article et est titré de son lien. L'article lui se construit d'un contenu par une page généré sur la template général.

3.2.3. Page de structures

La page structures contient des cards pour chaque structures. Une card est une sorte de panel qui peut contenir une image, un titre, un bloc descriptif dans notre cas. Comme il pourrait contenir un bloc entier dans d'autres circonstances. Les cards doivent utiliser la classe card sur un bloc. Les sous-blocs possibles sont card-img : l'image (qui peut être dirigé avec un suffixe -top,-left...), card-body : le bloc du contenu, card-title : pour le titre, card-text : pour la description

Pour les images, j'ai choisi de mettre les logos des laboratoires respectifs. Le titre, la description sont aussi orienté pour les laboratoires. Sinon, il y'a un bouton qui redirige vers la page principal du site "Dashboard" généré du laboratoire. Ou un autre bouton qui va nous générer un modal. Un modal est une pop-up Bootstrap. Dans laquelle, j'ai mis une liste Bootstrap qui redirige vers les équipes du laboratoire.

Le modal se génére grace à la classe modal-dialog. Les sous-blocs modal-content, modal-header, modal-body, modal-footer. Les éléments sont implicitements ressemblant à ceux de HTML. Donc, on construit les sous-blocs par ordre logique avec le contenu souhaité dans notre cas une liste bootstrap formé avec la classe list-group sur un ul et list-group-item sur les li. Pour ouvrir un modal, il faut un déclencheur comme un bouton avec comme paramètre data-toggle="modal" data-target="idDuModal".

3.2.4. Page de chercheur

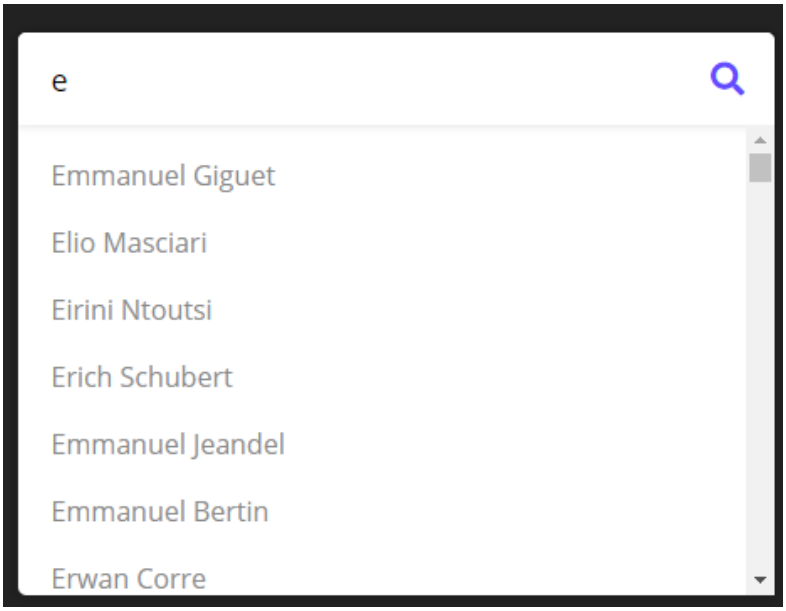


FIGURE 3.7. – Barre de recherche avec suggestions.

Cette page contient une barre de recherche avec suggestions. Pour le style, Bootstrap a été utilisé, ainsi que font-awesome (librarie qui contient des icones et des polices) pour l'icone de recherche.

En ce qui concerne l'idée générale de la partie technique : L'idée pour moi été un peu de

m'inspirer d'une barre de recherche comme Google/Youtube. De ce fait, des suggestions sont nécessaire. Les suggestions sont les noms des chercheurs existant dans la base de données. Je les récupère grâce à une méthode asynchrone dans laquelle on va utiliser une récente requête côté JS en combinant l'utilisation d'await et fetch(). Je ne détaillerai pas maintenant les requêtes, ils le sont plus tard dans le rapport. En tout cas, on récupère une réponse JSON avec la liste de toutes les personnes. Puis, un callback sur la méthode main() de notre script avec comme paramètre la réponse de l'API va nous permettre de construire la barre de recherche en fonction des suggestions.

Les suggestions vont s'afficher grace à un Wrapper (sorte de sous bloc dynamique lorsque l'on écrit). Dans laquelle on va filtrer les bonnes données notamment grâce à la méthode .filter et .startsWith ainsi que de .map pour écrire nos listes d'auteurs trouvées. Sur chaque liste de la suggestion on ajoute un eventListener "click" pour pouvoir choisir une suggestion.

3.2.5. Page de crédits

La page de crédit contient l'équipe du projet sous forme de flip-cards (card qui se retourne avec une animation lorsque la souris est dessus). Sur une card, on a l'image, le nom et le rôle sur le recto. Le verso contient le nom et une mini-description.

La conception a été inspiré d'un tutoriel et de la documentation sur les flip-card. Il faut avoir comme bloc mère, une classe flip. Un sous-bloc frontside et backside qui vont contenir les 2 faces d'une card, sachant que l'on créer une card avec la même logique que pour la page des structures. Dans notre cas, l'équipe est composé de 3 personnes donc on répète ses étapes 3 fois avec le contenu différent.

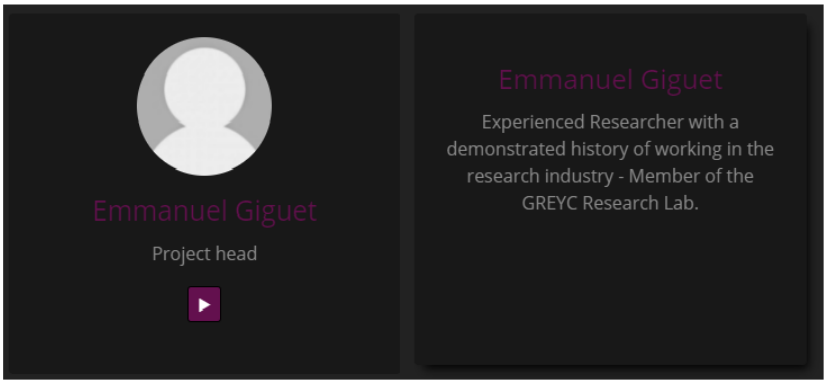


FIGURE 3.8. – Card qui est de côté face à gauche et retourner à droite.

3.3. Page de profil

3.3.1. Structure de la page de profil

La structure de la page de profil est celle-ci :

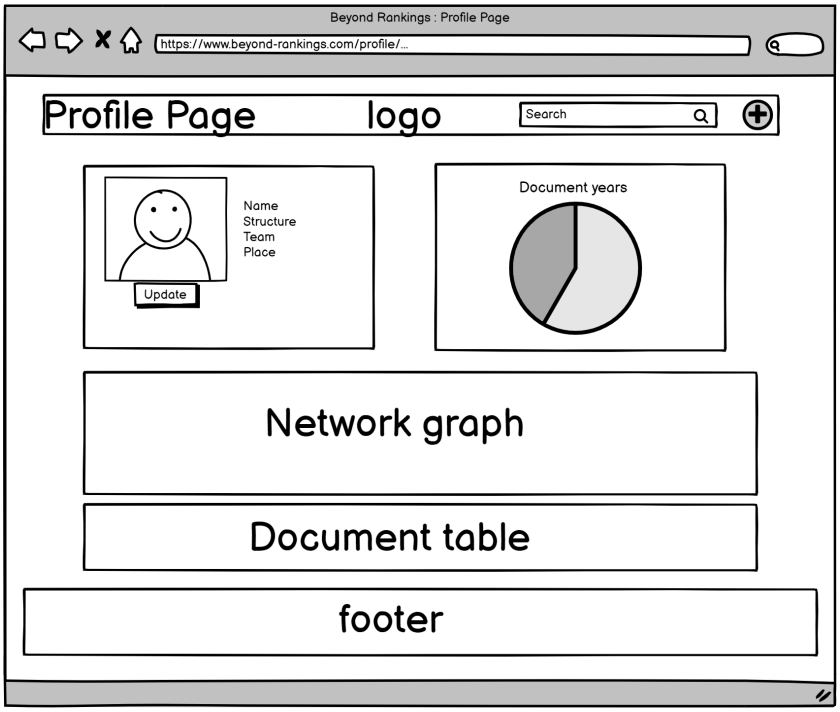


FIGURE 3.9. – Maquette de la page de profil

J'ai suivi la charte graphique du site (noir et gris) notamment avec un mode dark et light. La structure de la page suit aussi la même logique.

3.3.2. Architecture pour la page de profil

Les packages nécessaires pour la page de profil se décomposent ainsi :

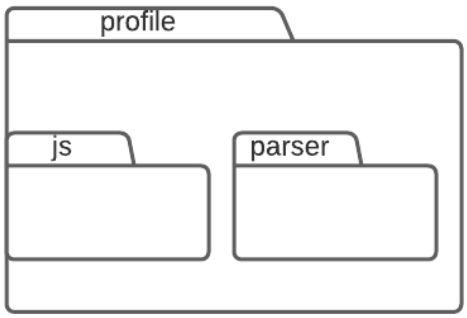


FIGURE 3.10. – Packages pour la page de profil.

3.3.3. Conception d'une API REST

J'ai conçu une API REST basique en php pour le traitement des données des différentes collections. Il y'a 3 étapes essentielles pour la conception de cet API. Il faut un Client donc une requête quelconque sur l'API (navigateur, requête fetch, ajax...). L'API elle-même doit exposer les ressources sur une URL. Enfin, la dernière est d'avoir une base de données et des requêtes l'exploitant.

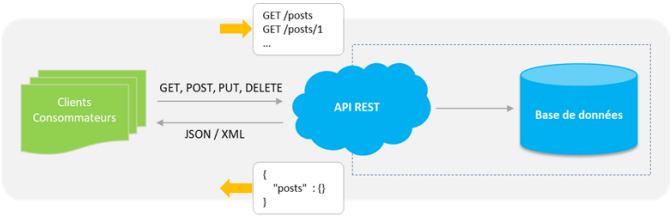


FIGURE 3.11. – Conception d'une API Rest.

La partie client sera expliqué dans la prochaine partie sur les requêtes. La base de données est réalisé via des étapes basiques de parsing sur du JSON. Le package parser contient : ParseDatabase : classe qui va récuperer les collections JSON décoder en liste manipulable par php avec json decode. J'ai aussi tester des librairies github comme json-streaming, JPOPHP pour gagner en performance. Mais bon, j'ai pas eu les résultats que je voulais donc il est compliqué de faire mieux qu'avec la méthode json decode.
DataBuilder : est la classe abstraite construit avec la base de donnée et une clé json. Elle permet d'offrir une récuperation de donnée identique pour chaque objet de la collection.

Authors, Bibrefs, Doctypes, Profile, Structures sont des sous-classes de DataBuilder construit avec leur clé respectif ("authors pour Authors") donc une récuperation de données avec une méthode hérité de la classe DataBuilder s'effectuera sur une liste collection["authors"].
Il y'a eu quelques différence sur la structure de certains objets JSON donc on peut réaliser des méthodes dans les sous-classes comme c'est le cas pour la classe People : qui a une méthode findPerson(name) ou findPersons() pour trouver une ou les personnes dans les collections.
L'API expose les données avec ses étapes : En php, il faut déclarer un header avec comme Content-type le format de donnée voulue (ici le JSON). Il faut choisir le paramètre voulu, dans notre cas ce sera un paramètre sur l'URL en GET, notre API est dans l'intention de récuperer des données. On peut eventuellement ajouter des requêtes PUT, DELETE, POST si on va plus loin avec cette API.
Lorsque l'un paramètre est valide on encode avec la fonction json encode (méthode php qui met au format json une liste de données) depuis notre base de données. Sinon, on renvoie une erreur HTTP (code 4xx) pour les requêtes.
On utilise exit() dès la fin de la requête pour terminer le script courant.

3.3.4. Les requêtes sur l'API

Pour effectuer une requête sur l'API, il faut faire de l'asynchrone. J'ai utiliser les dernières méthodes de Javascript notamment await et fetch(). Await permet d'attendre la fin d'une requête, donc son résultat : une promesse. Je l'ai notamment utilisé pour déclarer mes variables de réponses au format converti en json. Fetch() permet de réaliser une requête HTTP sur une URL, avec des paramètres comme GET, POST ... On récupère la réponse qui est une promesse. La promesse contient un statut (2xx pour

succès, 4xx pour un échec) mais surtout une réponse. La réponse peut-être converti vers un format comme le json.

Voici l'exemple d'une requête et de la conversion de sa réponse au format json :

```
// id = parametre de main
const response = await fetch('./collectionsAPI.php?data=people&name=' +
    id)
        .then((resp) => {
            if (resp.status === 400 || resp.status === 404) {
                throw 'Erreur : Chercheur introuvable';
            } else {
                console.log("Chercheur trouv  ! Chargement des donn es
                    en cours...");
                return resp;
            }
        })
        .catch((error) => {
            console.log(error);
        });
    const data = await response.json();
```

Voici à quoi ressemble la réponse d'une requête sur le nom d'un chercheur :


```
// 20210604152714
// http://127.0.0.1/profile/collectionsAPI.php?data=people&name=Emmanuel%20Giguet

{
  "id": "people-0",
  "type": "people",
  "name": "Emmanuel Giguet",
  "member": {
    "fullname": "Emmanuel Giguet",
    "team": "Hultech",
    "position": "Permanent",
    "end": "",
    "start": "",
    "struct": "struct-388300"
  },
  "authors": [
    "231226"
  ],
  "position": "Permanent",
  "status": "member",
  "team": "Hultech"
}
```


FIGURE 3.12. – Résultat d'une requête.

Dans mon cas, j'ai utiliser une méthode main(id) pour gérer tout appel asynchrone et des méthodes asynchrones pour des requêtes qui requiert du filtrage appélé dans le main. On réalise une fonction callback : build() qui permet de traiter toutes les données des réponses asynchrones.

3.3.5. Génération de la page

Afin de générer notre fonction main(), il a fallu vérifier les paramètres de l'URL depuis notre script.

```
const searchParams = new URLSearchParams(window.location.search.toString
    ());
document.addEventListener("DOMContentLoaded", function () {
    if (searchParams.get('searcher') !== null) {
        main(searchParams.get('searcher'));
```

```
    } else {
        errorPage();
    }
});
```

La méthode main(id) précèdemment indiqué est appelée en fonction d'un paramètre d'URL grace à l'objet URLSearchParams sur l'URL de la fenêtre. On peut récuperer le paramètre voulu grace à cet objet. Si le paramètre est valide alors on génére le script main() avec comme paramètre le nom du chercheur. Sinon on renvoie la page d'erreur.

3.3.6. Card de la personne

La card de la personne contient son image, son nom/prénom, sa structure, son équipe et son pays/ville. Si une information n'existe pas pour une personne, on le précise. Un bouton Update est mis pour une future ouverture sur la mise à jour de l'image d'un chercheur depuis la page.

3.3.7. Datatable

La conception du dataTable est semblable à celle pour la page de Beyond Rankings. On a créer une table avec un module PublicationsCard. Dans laquelle, la méthode build() construit en fonction d'une liste de documents une table. La table est ensuite transformé avec jquery grace à la méthode

```
$('#table_people').dataTable({
            "pageLength": 10,
        });
$('.dataTables_length').addClass('bs-select');
```

Un style existe avec Bootstrap. Il suffit d'importer ce lien sur notre page.

```
<link href="https://cdn.datatables.net/1.10.24/css/dataTables.bootstrap5.
    min.css" rel="stylesheet" crossorigin="anonymous">
```

J'y ai apporté mes modifications par-dessus en réecrivant dans mon CSS car je voulais modifier essentiellement les couleurs.

3.3.8. Graph de réseaux entre les chercheurs

Le graph contient comme noeuds tous les auteurs des documents. Il contient comme arête toutes les collaborations entre chercheurs.

La conception de ce graph a été effectué avec Chart.js. Pour les noeuds, il faut avoir une liste de JSONObject avec comme clé id : nom du noeud, et facultativement une position x ou y. Pour les arêtes, il faut avoir une liste de JSONObject aussi avec comme clé from : noeudA et une autre to : noeudB.

Donc, j'ai effectué un algorithme qui va récuperer tous les auteurs et ceux qui collaborent entre eux. L'algorithme réalisé quelques duplications pour les collaborations donc une solution a été de les supprimer car Chart.js ne veut qu'une seule arête entre 2 noeuds.

Ensuite, on stock nos données comme ceci :

```
let data = {
    "nodes": nodes,
    "edges": edges
    }
```

On déclare la chart :

```
var chart = anychart.graph(data);
```

Le reste de mon code permet de configurer l'affichage des noeuds grace à chart.nodes(). A la fin, on dessine le graph grace à cette commande :

```
chart.container("graph-bloc").draw();
```

3.3.9. Chart doughnut sur les années.

Le module YearsChart requiert une liste avec une liste de légendes sur l'index 0 et une liste d'années sur index 1. Avec nos données sous forme de liste, le passage en Doughnut est assez simplifié. Il suffit de créer un objet Chart avec comme type le doughnut avec labels la liste de légende d'index 0 et data la liste d'années d'index 1.

```
const myChart = new Chart(ctx, {
            type: 'doughnut',
            data: {
                labels: this.years[0],
                datasets: [{
                    label: 'Years statistics',
                    data: this.years[1],
                    ...
```

Le résultat ressemble à ca :

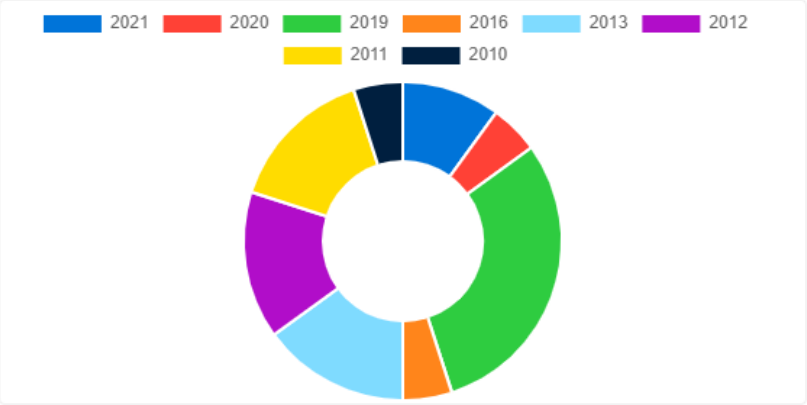


FIGURE 3.13. – Résultat de l'objet Chart typé en Doughnut.

4. Conclusion

Durant ce stage, je me suis fixé comme objectif de réaliser les différentes missions qui m'ont été proposés. Pour ce faire, j'ai dû inclure et modifier des widgets sur la page dashboard du site web. Surtout j'ai fais pas mal de retouches sur les styles et la mise en page de l'application. J'ai pu concevoir des nouvelles pages. Surtout la page de profil et tout ce qui va avec pour trouver un chercheur (API Rest, Parser...).

Ce que j'ai trouvé intéressant lors de ce stage est l'autonomie que l'on nous a offert. Ce que je regrette c'est le distanciel sur la fin d'année qui devenait fatiguant pour ma part. Mais j'ai fais de mon mieux pour garder un rythme de travail correct.

Des pistes de travaux futurs serait de continuer à enrichir de manière plus ergonomique le style des pages et d'approfondir certains des widgets integrés. Les technologies utilisés dans le projet peuvent aussi être modernisés.

J'ai pu acquérir de l'expérience en front-end (CSS, design, affichage de données...) et découvrir certaines librairies. Je n'avais jamais utilisé auparavant ni d3, ni bootstrap et ni chart.js... Ainsi que la partie sur l'API qui m'a permis de tester des nouvelles fonctions et d'intégrer pour la première fois ce genre d'architecture dans une application. Donc, j'ai pris des risques sur mes déploiements mais ca m'a permis d'en apprendre d'avantage.

Première partie

Annexes

Bibliographie

[1] ExperiencesProjets - La réecriture et la redirection d'URL avec Apache. *Source de l'image.* Lien : `http://experiences.projets.free.fr/index.php?tg=articles&idx=Print&topics=38&article=202`

[2] Blog de Nicolas Hachet - Exemple d'API Rest. *Source de l'image.* Lien : `https://blog.nicolashachet.com/developpement-php/exemples-api-rest-en-php/`

[3] DataLab Normandie - Description du laboratoire GREYC. *Source du paragraphe descriptif du GREYC.* Lien : `https://www.datalab-normandie.fr/laboratoire-greyc/`

[4] Discord OpenHive, le membre "Senpai" en M2 Dop que je remercie pour le partage de sa template latex. *Certains éléments de sa template m'ont servi notamment pour la page de garde.*

Table des figures

2.1. Logo Discord 2

3.1. Wireframe de la structure du Dashboard en v1 6
3.2. Wireframe de la structure du Dashboard actuellement. 6
3.3. UML de la classe pour filtrer les widgets. 7
3.4. Arbre schématisant l'idée générale du filtrage de widget. 7
3.5. Slider et histogramme sur les années des documents. 8
3.6. Génération d'une URL avec une RewriteRule 9
3.7. Barre de recherche avec suggestions. 10
3.8. Card qui est de côté face à gauche et retourner à droite. 11
3.9. Maquette de la page de profil 12
3.10. Packages pour la page de profil. 12
3.11. Conception d'une API Rest. 13
3.12. Résultat d'une requête. 14
3.13. Résultat de l'objet Chart typé en Doughnut. 16

Résumé

Au sein de ce stage au laboratoire GREYC, j'ai eu plusieurs petites missions sur une application web. Ce projet intitulé Beyond Rankings est au sujet des collaborations entre chercheurs au sein d'un laboratoire. J'ai developpé plusieurs fonctionnalités comme des widgets, de la mise en page ou de la conception de pages comme une page de profil pour un chercheur.

Abstract

Within this internship at the laboratory of GREYC. I had multiple little missions about a web application. This project is named Beyond Rankings. It talks about collaboration between researchers within a laboratory. I had developped some functionalities like widgets, layout, or some pages conception like a profile page for a researcher.

Keywords :
GREYC : Laboratoire de recherche à CAEN. / Research laboratory in CAEN.
Beyond Rankings : Projet web du GREYC / Web project made by GREYC