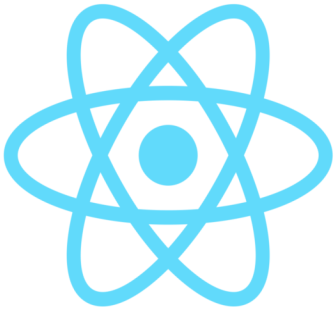
Adrien Vasseur
M2 Informatique (AL)



Rapport Individuel du projet React

Équipe :

- **Adrien Vasseur, M2 Informatique (AL)**
- Yassine Jrad, M2 Informatique (AL)
- Thierno Fall, SI5 (Web)

Url du projet : https://github.com/JYassine/si5-project-web-nodejs

Identifiant Github : adrien-vasseur

Table des matières

I. Taches effectuées

- La « Header Bar », la barre en haut du site qui contient le titre et le bouton pour dérouler ou replier le menu
  - Composant : Header
- Le menu, qui contient les accès à toutes les pages du site. Le menu peut être en mode « replié », dans ce cas il n'affiche que les icônes des pages vers lesquelles il renvoie, ou bien en mode « déroulé », dans ce cas il devient un peu plus large, et on peut voir également le nom des pages vers lesquelles il renvoie. Le reste de la page se décale également pour s'adapter au déroulement du menu.
  - Composants :
    - Menu
    - MenuItem
- La structure de la page, c'est-à-dire le découpage en lignes et colonnes pour uniformiser les pages entre elles. Pour cela je me suis aidé des composants reactstrap Container, Row et Col qui permettent l'utilisation plus facile des flexbox.
- La page **/list**, qui affiche dans un tableau les données que nous avons enregistrées dans a base de données. Comme il y a beaucoup d'entrées à afficher, j'ai fait un tableau « multi-pages » qui affiche 12 entrées par page, avec une pagination en bas du tableau qui permet de sélectionner la page. Pendant le chargement des données du tableau, le tableau est remplacé par une icône animée de chargement.
  - Composants :
    - TableComponent
    - TablePagination
- La page d'accueil (/) qui affiche quelques informations à la date d'aujourd'hui sur les chiffres du covid en France (nombre de décès, de guéris, d'hospitalisés…). Pour cela je fais appel à deux API (une seule à la base, mais elle a changé et renvoie désormais moins d'informations) et à partir des données reçues je crée plusieurs instances du composant général CovidInfoCard qui affiche les données et leur nom à l'écran. Seule l'instance pour le nombre de cas total est affiché sur toutes les pages. Pendant le chargement des données, le nombre dans ces composants est remplacé par une icône animée de chargement.
  - Composants :
    - HomePageComponent
    - CovidInfoCard
- La page de contact (**/contact**) qui contient un formulaire avec validation côté client (email, sujet, message) et permet d'envoyer un message aux développeurs. Ce message sera transformé en email grâce à EmailJS, donc de notre côté nous recevons un email avec toutes les informations entrées par l'utilisateur et nous pouvons le contacter en retour directement par mail.
  - Composants :
    - ContactPage
    - ContactForm
- La carte (**/map**) qui affiche sur un carte Google Maps le nombre de tests positifs additionnées pour chaque région sur un mois, filtrable par mois
  - Composants :
    - Heatmap
    - MapComponent

    - MonthFilter
- La possibilité de détecter la région de l'utilisateur, avec un bouton « Détecter région » dans les filtres sur /list et /graph. Utilise l'API HTML5 Geolocation et l'API Google Geocoding.
- Le routage
- Le déploiement sur Google Cloud Platform

II. Stratégie employée pour la gestion des versions avec Git

Nous avons travaillé comme sur nos autres projets avec les « feature branch ». Pour chaque nouvelle issue sur laquelle nous travaillons, nous ouvrons une nouvelle branche. Lorsque nous avons accompli une tache, nous ouvrons une pull request et les autres membres du groupe l'examinent avant de merge sur la branche dev, puis sur la branche principale lorsque nous voulons faire une nouvelle release.

III. Solutions choisies

Pour les composants nous avons choisi de nous aider de reactstrap. N'ayant que peu d'expérience dans le développement frontend, reactsrap a permis de beaucoup raccourcir le temps de développement car ils fournissent beaucoup de composants (Container, Card…) qui sont personnalisables avec les classes bootstrap.

Nous avons aussi utilisé Sass, qui permet de grandement fluidifier le développement du css lorsqu'il y a beaucoup de propriétés à écrire.

Pour les icônes nous avons utilisé Fontawesome qui est une grande librairie qui a fourni ce dont nous avions besoin (icônes d'accueil, des pages liste, carte, graph et contact, ainsi que les icônes animées de chargement).

Il existe beaucoup de librairies qui fournissent des composants « tout faits » semblables à reactstrap que nous aurions pu utiliser. Nous en avons utilisé certains (*availity-reactstrap-validation*) mais de manière générale reactstrap nous a fourni tout ce dont nous avions besoin pour nous aider.

IV. Difficultés rencontrées

J'ai eu du mal à rendre mes composants « responsive » qui s'adaptent à toutes les tailles d'écran. Cela m'a pris beaucoup de temps et ce n'est toujours pas parfait car selon l'appareil qui accède au site, il faut agencer les composants différemment, ou changer leur taille, etc. Une solution à ça était bootstrap (avec reactsrap) et notamment les propriétés de la flexbox avec le composant Container qui permet facilement d'agencer les composants selon la taille de l'écran. Mais pour le menu par exemple, j'aurais aimé qu'il s'ouvre automatiquement et prenne toute la largeur de

l'écran au dessus du reste de la page si c'est un smartphone qui accède au site, ou bien qu'il reste replié mais se mette en ligne au lieu d'une colonne. Malheureusement pour le peu de temps que nous avions pour le projet j'ai préféré me concentrer sur le reste des tâches à faire car cela m'aurait pris trop de temps.

J'ai également eu du mal à faire des composants « généraux » que l'on peut réutiliser dans certaines situations. Pour CovidInfoCard par exemple, c'était à la base un composant uniquement pour afficher le nombre de cas total sur toutes les pages. Lorsque j'ai fait la page d'accueil, j'ai fait un gros refactoring sur ce composant pour le rendre le plus général possible et pouvoir afficher toutes les informations liées au covid avec un seul et même composant, et non en créer plusieurs.

Enfin, j'ai eu beaucoup de mal à faire la pagination du tableau des données. Après quelques heures de travail j'ai finalement décidé d'utiliser un composant tout fait disponible sur npm (react-reactstrap-pagination).

V. Temps de développement par tâche

- Header : 5 %
- Menu : 20 %
- Structure de la page : 5 %
- Page /list : 20 %
- Page d'accueil : 15 %
- Page /contact : 10 %
- Page /map 15 %
- Routage : 5 %
- Déploiement : 5 %

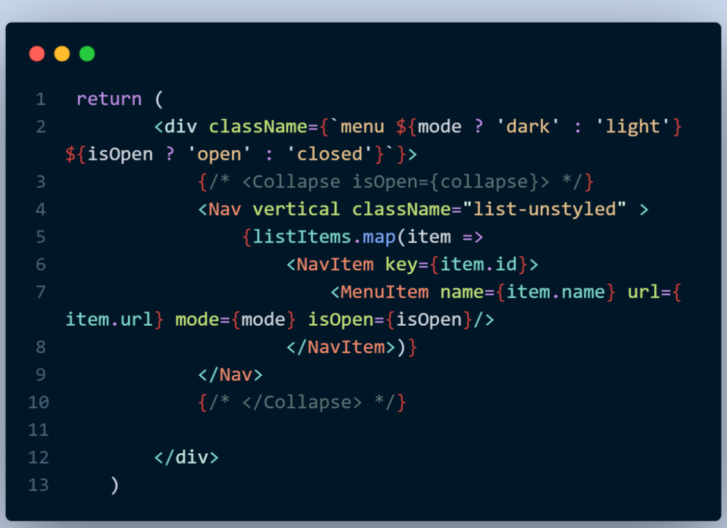
VI. Un composant élégant

C'est le composant **CovidInfoCard**. Comme dit plus haut c'est un composant que j'ai refactoré pour être le plus général possible. Ainsi, j'ai juste à passer en props un *name* et un *data*. Selon le nom, le titre du composant va changer, et afficher le nombre que nous lui donnons en props. Il y a une exception pour le nombre de cas total en France, car il est affiché sur toutes les pages. Ainsi, dans ce cas, c'est le composant lui-même qui va faire la requête à l'API, alors que pour les autres c'est le composant parent. De plus, pour ce cas, j'affiche la date d'aujourd'hui dans le composant pour afficher la date sur toutes les pages.

```jsx
export const CovidInfoCard = ({ mode, name, data }) => {
    const [dataNumber, setDataNumber] = useState(-1);
    const [title, setTitle] = useState();
    const urlToFetch = "
https://www.trackcorona.live/api/countries/FR";

    const renderLoadingOrNumber = () => {

// if (isLoaded === false) return <FontAwesomeIcon icon=
{faSpinner} className="fa fa-spinner fa-spin" />;
        if (dataNumber === -1 && data === -1) return <
FontAwesomeIcon icon={faSpinner} className="
fa fa-spinner fa-spin" />;
        else return name === 'totalCases' ?
            new Intl.NumberFormat('fr-FR').format(
dataNumber)
            :
            new Intl.NumberFormat('fr-FR').format(data);
    }

    useEffect(() => {
        switch (name) {
            case 'totalCases':
                fetch(urlToFetch)
                    .then(result => result.json())
                    .then(json => setDataNumber(json.
data[0].confirmed))
                    .catch(err => console.log(err));
                setTitle('Nombre de cas en France');
                break;
            case 'deces':
                setTitle('Nombre de décès');
                break;
            case 'decesEhpad':
                setTitle('Nombre de décès en Ehpad');
                break;
            case 'hospitalises':
                setTitle('Nombre d\'hospitalisations');
                break;
            case 'reanimation':
                setTitle('Nombre de cas en réanimation')
;
                break;
            case 'gueris':
                setTitle('Nombre de cas guéris');
                break;
            case 'casEhpad':
                setTitle('Nombre de cas en Ehpad');
                break;
            default:
                setDataNumber(-1);
        }
    }, [name]);


    return (
        <Card className={`
covid-info-card p-3 mb-3 text-center
        ${mode ? 'dark' : 'light'}`}>
            <CardTitle tag="h4">{title}</CardTitle>
            <CardText>
                {renderLoadingOrNumber()}
            </CardText>
            {name === 'totalCases' && <CardSubtitle>
                <em>Au {moment().format('LL')}</em>
            </CardSubtitle>}

        </Card>
    )
}
```

VI. Un composant moins élégant

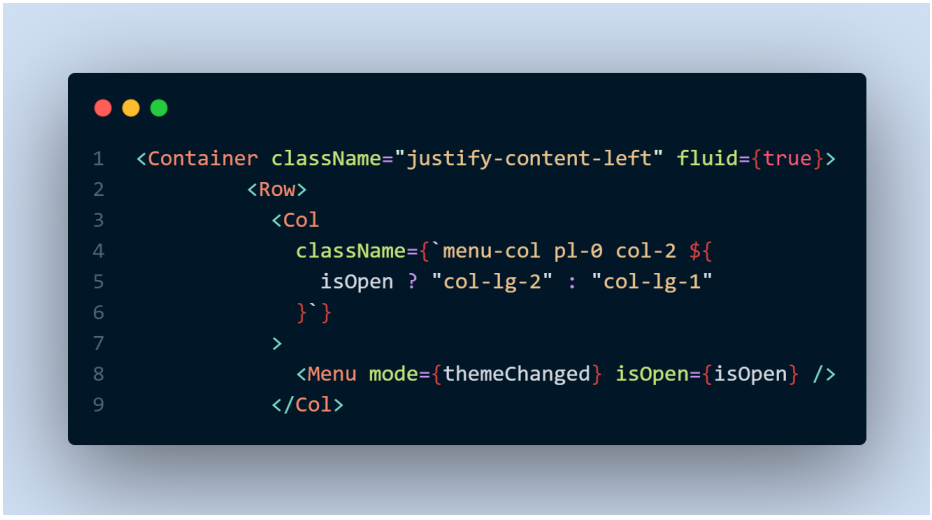
Le composant dont je suis moins fier est le composant du menu de gauche qui redirige vers toutes les pages du site. En effet il est très peu responsive, et ce n'est pas bien adapté lorsque l'utilisateur accède au site avec un smartphone ou un autre petit écran.



```
return (
        <div className={`menu ${mode ? 'dark' : 'light'} ${isOpen ? 'open' : 'closed'}`}>
            {/* <Collapse isOpen={collapse}> */}
            <Nav vertical className="list-unstyled" >
                {listItems.map(item =>
                    <NavItem key={item.id}>
                        <MenuItem name={item.name} url={item.url} mode={mode} isOpen={isOpen}/>
                    </NavItem>)}
            </Nav>
            {/* </Collapse> */}

        </div>
    )
```

Pour améliorer ce composant j'aurais du dynamiquement modifier la structure de la page selon la taille de l'écran de l'utilisateur.

Ainsi, lorsque l'utilisateur accède au site avec un grand écran, le menu s'affiche normalement à gauche, mais lorsqu'il utilise un smartphone, la colonne (flexbox) du menu deviendrait une ligne, et donc le menu pourrait s'afficher horizontalement en haut de l'écran, ce qui améliorerait grandement la lisibilité du site, et donc l'expérience utilisateur.



```
<Container className="justify-content-left" fluid={true}>
          <Row>
            <Col
              className={`menu-col pl-0 col-2 ${
                isOpen ? "col-lg-2" : "col-lg-1"
              }`}
            >
              <Menu mode={themeChanged} isOpen={isOpen} />
            </Col>
```