

DOSSIER DE PROJET

Clément Ramos

Développeur Web et Web mobile

Studi

Site original Dev' up!



Remerciements

Je tiens à remercier particulièrement Mr SALL Abdou qui m'a aidé constamment chaque jour à mener à bien ce projet. Il m'a donné l'opportunité de faire un projet intéressant malgré sa complexité, qui m'a permis de me confronter à la réalité du métier de développeur, de mettre en pratique les connaissances acquises lors de la formation et d'en acquérir beaucoup d'autres.

Sommaire

Remerciements.....	2
Sommaire.....	3
Introduction.....	4
1. Compétences du référentiel couvertes par le projet.....	4
a. Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité.....	4
b. Développer la partie back-end d'une application web ou web mobile en intégrant les recommandations de sécurité.....	4
2. Résumé du projet.....	5
3. Environnement humain et technique.....	6
a. Environnement humain.....	6
b. Environnement technique.....	6
Les projets.....	7
1. Le cahier des charges « Dev' up ! ».....	7
2. Spécifications techniques « Dev' up ! ».....	10
3. Spécifications techniques « Terrablock ».....	10
La réalisation du projet « Dev' up ! ».....	11
1. Choix du langage.....	11
2. Mise en place de l'environnement de développement.....	13
3. Développement de la partie front-end.....	16
a. Différentes intégrations	16
b. Extraits de code.....	18
c. Sécurité.....	
d. Le déploiement de l'application.....	35
Fonctionnalité la plus représentative : La récupération des informations GitHub.....	35
1. Explications et jeu d'essai.....	36
2. Recherches anglophones.....	38
Conclusion.....	40
Annexes.....	41

Introduction

1. Compétences du référentiel couvertes par le projet

a. Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité

- Maquetter une application
- Réaliser une interface utilisateur web statique et adaptable
- Développer une interface utilisateur web dynamique

La page web doit être accessible sur mobile et ordinateur, d'où le choix de commencer par la réalisation de plusieurs maquettes pour imaginer le site en version responsive. J'ai ensuite décidé d'utiliser React Native avec le framework Next.js pour plus de facilités. Elle doit proposer une expérience utilisateur simple et fluide quelque soit la taille de l'écran. J'ai donc fait en sorte que l'interface s'adapte aux différentes tailles d'écrans.

b. Développer la partie back-end d'une application web ou web mobile en intégrant les recommandations de sécurité

- Développer les composants d'accès aux données
- Développer la partie back-end d'une application web ou web mobile

La partie front de mon site web fait appel à un formulaire de contact qui envoie un mail à l'utilisateur et à moi-même. Cependant, selon moi, cela ne couvre pas assez le référentiel de l'activité type 2. C'est pour cela que je préfère développer cette partie en utilisant la partie back-end d'un autre projet, qui est développé en PHP avec le modèle VMC (Vue Modèle Contrôleur).

2. Résumé du projet

Dans le cadre de ma formation Développeur Web – Web Mobile, je réalise une application web pour un projet de NFT.

Ce projet m'a fait réaliser l'importance de posséder en tant que développeur Web, un site internet attractif, dynamique et sobre pour présenter ses projets et pouvoir être contacté par des personnes désireuses de se démarquer sur le web.

Je décide donc de créer mon site « Dev'up ! » une contraction de « development is up ! »

L'utilisateur arrive sur le site où lui sont présentées des informations me concernant, ainsi que des liens pratiques et simples d'utilisation pour accéder facilement aux différentes rubriques du site.

L'utilisateur peut voir les projets sur lesquels j'ai travaillé et me contacter facilement pour avoir un devis gratuit.

Le déploiement d'une première version du site est prévu pour la fin du mois de mai 2022. Les technologies utilisées sont les suivantes :

- React avec le framework Nextjs pour un rendu côté serveur
- Tailwind CSS, un framework CSS pour une facilité d'intégration accrue
- L'outil Git
- Vercel pour le déploiement de l'application

Travaillant seul sur mon projet, je le réalise en toute autonomie du début à la fin.

Cela me permet de mettre en pratique les compétences acquises lors de ma formation mais également d'en gagner de nouvelles très rapidement en me référant aux différentes documentations et forums dédiés.

3. Environnement humain et technique

a. Environnement humain

J'ai passé beaucoup de temps depuis le début de mon cursus chez Studi, à chercher des sites web de développeurs, du style portfolio, avec l'espoir d'avoir à la fin de mon cursus, le mien.

J'ai alors demandé de l'aide sur différents forums (StackOverflow entres autres), mais surtout à un ami de longue date et collègue, Mr Abdou SALL. Il est comme moi assistant d'éducation spécialisé TICE (explication) et passionné par le codage. J'ai pu lui demander conseil et utiliser son expérience dans le développement informatique pour m'inciter à prendre du recul lorsque j'ai rencontré certains obstacles. Il m'a beaucoup aidé dans le raisonnement qui m'a souvent amené à parvenir à trouver la solution.

Début avril, j'ai fait la rencontre d'un informaticien de l'académie de Montpellier, Jean, qui était en intervention dans notre collège pour une installation de nouveaux logiciels. Son travail à un impact sur le mien. J'ai grâce à lui compris l'importance d'une diversification dans l'apprentissage des langages de programmation et dans les différentes technologies pour être un développeur compétent et réfléchi.

Enfin, début mai, j'ai été contacté via les réseaux sociaux pour créer un site internet. J'ai été embauché par Mathieu Dumas, un thérapeute réflexologue plantaire en Haute-Loire pour passer d'un site WordPress, à un site dynamique codé de A à Z.

b. Environnement technique

Je travaille sur mon ordinateur portable personnel car j'ai besoin d'être mobile et de travailler sur mon temps libre. J'ai également chez moi un ordinateur de bureau qui est connecté à mon compte OneDrive pour un accès facile à mes applications via les différentes plateformes.

J'ai beaucoup de temps libre lorsque je suis sur mon lieu de travail, quand je ne suis pas en mission. En étant la majeure partie de mon temps avec mon collègue, j'ai beaucoup d'occasions pour lui demander conseil.

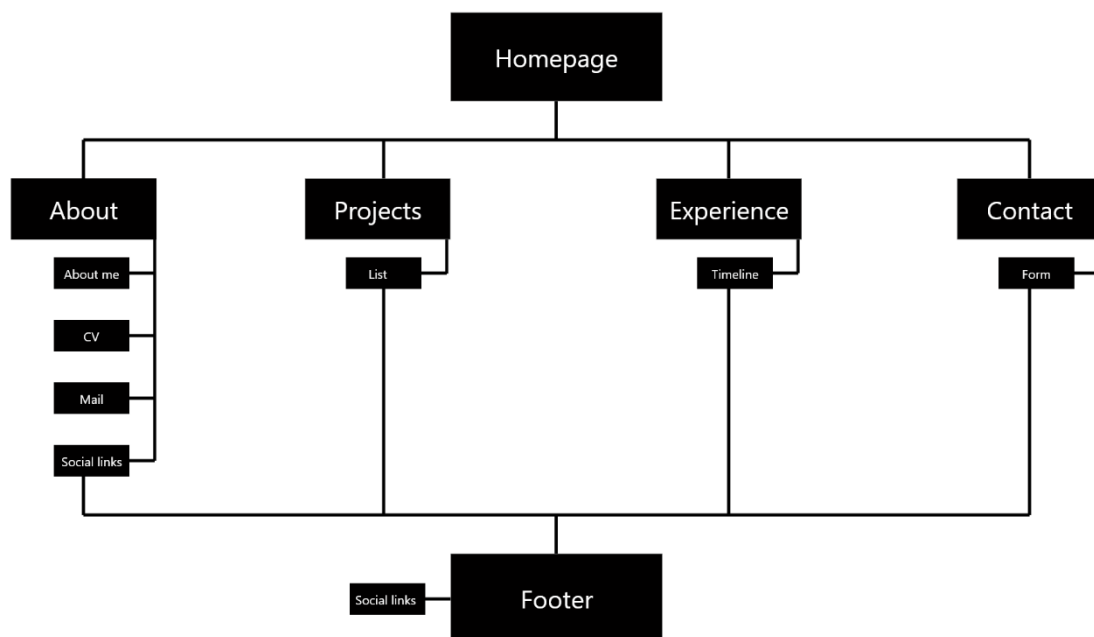
Les projets

1. Le cahier des charges « Dev' up ! »

Je me suis inspiré de plusieurs site web portfolio présentant le travail de plusieurs développeurs (mettre sites).

Mon cahier des charges s'est donc inspiré de ces sites web pour faire le mien, afin qu'il soit le plus représentatif possible de ce que je sais faire. J'ai donc décidé de faire mon site avec React Native et le framework Next.js. La partie développement fût relativement rapide et le déploiement du site à pu se faire rapidement. Le site est hébergé via Vercel.

J'ai travaillé en mode Agile avec une méthodologie Scrum allégée. J'ai ainsi adapté l'avancement du développement en fonction des fonctionnalités et écrans que je valide au fur et à mesure.




Le parcours type de l'utilisateur est le suivant :

- A son arrivée sur le site web, l'utilisateur voit un écran qui s'anime avec une photo de moi. Il peut ainsi défiler la page vers le bas pour voir les derniers projets sur lesquels j'ai travaillé et sur mes futurs projets. Il accède ensuite à mes 6 derniers « Repositories » Git Hub pour voir mon travail. Il arrive ensuite en pied de page avec quelques informations à mon sujet.

- Via la barre de navigation, il peut accéder à plusieurs parties du site.
 - o Une première partie « About » qui me présente et présente les technologies que j'utilise, ainsi que le projet sur lequel je travaille. Il a une partie avec quelques informations pour me contacter.
 - o Une seconde partie « Projects » dans laquelle l'utilisateur voit tous les projets sur lesquels j'ai travaillé, ainsi que mes futurs projets. Il peut cliquer sur les cartes pour en savoir plus.
 - o Une troisième partie « Expérience » sur laquelle il voit toutes mes expériences professionnelles
 - o Une quatrième partie « Contact » dans laquelle se trouve mes informations de contact ainsi qu'un formulaire de contact pour entrer en contact avec moi.
- Une prochaine partie du site est en développement, qui sera un blog personnel.

Pour l'analyse des besoins utilisateur, j'ai imaginé deux types d'utilisateur différent :

- Marie, Directrice Marketing dans une entreprise moyenne, qui souhaite engager un développeur web freelance.

	Marie souhaite m'engager pour créer/modifier le site web de leur entreprise.
Nom Marie	Elle peut donc accéder à mon site, le visiter rapidement, mais ce qu'elle recherche c'est un moyen facile et rapide de me contacter. S'offre alors à elle plusieurs possibilités :
Intitulé de poste Chargé de communication.	<ul style="list-style-type: none"> - Les réseaux sociaux qui sont affichés dans la barre de navigation et le pied de page de chaque page - Le formulaire de contact - Le mail affiché sur le formulaire de contact - Le numéro de téléphone affiché également sur le formulaire de contact.
Âge Entre 18 et 24 ans	
Niveau d'études Licence ou diplôme équivalent	
Secteur d'activité Investissement	
Taille de l'entreprise 51 à 200 salariés	

- Paul, CEO d'une Start-up, qui souhaite engager un développeur web full-stack.


Nom Paul
Intitulé de poste CEO
Âge Entre 18 et 24 ans
Niveau d'études Master ou diplôme équivalent
Taille de l'entreprise 51 à 200 salariés

Paul souhaite m'engager pour créer/modifier le site web de leur entreprise.

Il peut donc accéder à mon site, le visiter et prendre le temps sur chaque page pour voir ce dont je suis capable de produire, et de voir les technologies avec lesquelles je suis familier.

Il peut également s'attarder sur mon expérience professionnelle et télécharger mon CV.

Mais ce qu'il recherche c'est un moyen facile et rapide de me contacter. S'offre alors à lui plusieurs possibilités :

- Les réseaux sociaux qui sont affichés dans la barre de navigation et le pied de page de chaque page
- Le formulaire de contact
- Le mail affiché sur le formulaire de contact
- Le numéro de téléphone affiché également sur le formulaire de contact.

2. Spécifications techniques « Dev' up ! »

Voici les différentes technologies que j'ai utilisé pour ce projet :

- Environnement de travail :
 - o L'éditeur de code Visual Studio Code 2019 avec les extensions suivantes :
 - Prettier, un formater de code automatique
 - GitLens, pour développer les fonctionnalités sur de nouvelles branches, et « push » le code sur GitHub après chaque commit.
 - Nodemailer pour tester l'envoi de mail.
 - Nextjs snippets
- Partie front et back :
 - o Next.js : framework React orienté Server Side Rendering. Il permet d'avoir le backend et frontend sous le même toit. Il se base sur le file system node pour définir les routes en frontend et backend de l'application. Pour chaque fichier dans le répertoire Pages, Next.js crée la route frontend du même nom. Next.js à une simplicité de déploiement. Des services comme Netlify ou Vercel permettent de déployer l'application via le terminal et de bénéficier d'un CDN et d'un certificat SSL gratuitement.

3. Spécifications techniques « Terrablock »

Voici les différentes technologies que j'ai utilisé pour ce projet :

- Environnement de travail :
 - o L'éditeur de code Visual Studio Code 2019 avec les extensions suivantes :
 - Prettier, qui formate le code
 - GitLens, pour développer les fonctionnalités sur de nouvelles branches, et « push » le code sur GitHub après chaque commit.
 - PHP Intelephense
 - SQL Tools
- Partie back :
 - o PHP : J'ai décidé de créer un formulaire de contact pour permettre à un client de pouvoir effectuer une réclamation ou demander un service supplémentaire.

Une fois le formulaire rempli et envoyé, un mail est alors envoyé à l'administrateur du site pour prendre connaissance de la réclamation. À la suite du maquetage que j'ai effectué avec des Wireframes et grâce à Adobe XD, il m'était facile de réaliser la partie statique de ce formulaire de contact.

J'ai ensuite procédé à un maquetage de la base de données et ai créé un Modèle Physique de Données en utilisant le langage de modélisation UML.

PHP m'a servi à générer la page de contact, pour ensuite récupérer les informations rentrées par l'utilisateur pour ensuite les insérer en base de données.

Une fois les informations complétées et sans champs vides dans le formulaire, le code PHP permet d'envoyer un mail à l'administrateur du site web.

J'ai également dû créer des scripts SQL à la main pour les intégrer dans les scripts PHP afin de récupérer les informations et les insérer en base de données.

- Une fois que le formulaire est rempli et que l'utilisateur à cliquer sur « envoyer », un message de confirmation apparaît.

La réalisation du projet « Dev' up ! »

1. Choix du langage

Après de nombreuses discussions avec des collègues de travail et de lecture sur les différents forums, je décide de me lancer dans la production de mon site web en utilisant Next.js. Oui, mais à quelles fins ?

Next JS, ce framework React orienté Server Side Rendering, s'est imposé parmi les frameworks JavaScript les plus populaires.



Classement 2020 des frameworks Backend JavaScript les plus populaires

Pour un projet web de très petite envergure tel que celui-ci, toutes les technos auraient pu être adaptés. Je n'avais pas d'avantage à utiliser un framework en particulier.

Sur le marché Français, React est le framework frontend le plus populaire.

L'intérêt des frameworks JS tels que Next.js, Nuxt ou Gatsby est d'avoir le backend et le frontend sous le même toit.

Système de routes basé sur les fichiers

Next JS se base sur le file system node pour définir les routes frontend et backend de votre application.

Pour chaque fichier JavaScript présent dans le répertoire *Pages*, Next JS crée la route frontend du même nom.

Déploiement en CLI et hébergement gratuit

L'avantage que procure Next.js est la simplicité de déploiement. Des services comme Vercel ou Netlify nous permettent de déployer l'application via le terminal et de bénéficier d'un CDN et d'un certificat SSL gratuitement.

L'intérêt du Static Site Generation et Server Side Rendering via Next JS

Un des critères les plus importants aux yeux de Google et des internautes est la vitesse de chargement d'un site.

Dans un projet comme le mien, on peut se demander l'intérêt d'utiliser Next.js au lieu de faire directement tout le site en Node.js, PHP, Python ou tout autre langage de développement web pour renvoyer le HTML au navigateur.

Tout est une question d'optimisation et de trouver comment gratter des millisecondes de chargement. En effet, un des atouts de Next JS et de ses confrères est de pouvoir générer un site statique.

En utilisant du Server Side Rendering, via du PHP par exemple, les variables de mon code sont remplacées au moment de la requête par le serveur. Il n'y a qu'une seule page "index.php" et en fonction du lien sur lequel vous avez cliqué le serveur va remplacer les variables par les bons noms en dur pour l'affichage de la page.

Cela veut dire qu'à chaque requête, votre client interroge le serveur, celui-ci interroge la base de données, construit une page HTML et la retourne prête à l'emploi au navigateur.

Grâce au Static Site Generation, Next JS va créer toutes les pages que peut avoir le site en dur dès le *build*, c'est-à-dire au moment où je code le site. Il y aura une page par section qui sera déployée. L'avantage est que le serveur n'aura pas à construire la page à chaque requête, elle sera déjà prête à l'emploi.

Elle pourra être mise en cache pour être servie d'autant plus rapidement. C'est la solution la plus rapide pour livrer une page web.

En plus de Next.js, j'utilise tailwindcss, un framework CSS qui permet de prototyper rapidement et styliser une application web.

J'utilise également Rough Notation, une librairie qui permet de surligner un texte important.

2. Mise en place de l'environnement de développement

Avant de commencer à coder mon site, j'ai dû installer mon environnement de développement. J'ai donc eu besoin de deux outils :

- Node.js (version 10.13)
- Un éditeur (Visual Studio Code)
- Git

Vient ensuite la création d'une application Next.js via le terminal :

```
C:\Users\ramos>npx create-next-app nextjs-blog --use-npm --example "https://github.com/vercel/next-learn/tree/master/basics/learn-starter"

.next/
components/
lib/
node_modules/
package-lock.json
package.json
pages/
posts/
public/
README.md
styles/
```

Dans mon fichier que je viens de créer (« nextjs-blog ») je vais me positionner dessus :

```
C:\Users\ramos>cd nextjs-blog
C:\Users\ramos\nextjs-blog>_
```

Et lancer en localhost :

```
C:\Users\ramos\nextjs-blog>npm run dev
> dev
> next dev
ready - started server on 0.0.0.0:3000, url: http://localhost:3000
```

Voici la page sur laquelle j'atterris :

Welcome to Next.js!

Get started by editing `pages/index.js`

Documentation →

Find in-depth information about Next.js features and API.

Learn →

Learn about Next.js in an interactive course with quizzes!

Examples →

Deploy →

La page d'accueil Next.js en local host

3. Développement de la partie front-end

a. Différentes intégrations

Le site inclus ce que je considère être le minimum nécessaire dans un site portfolio, avec quelques fonctionnalités supplémentaires.

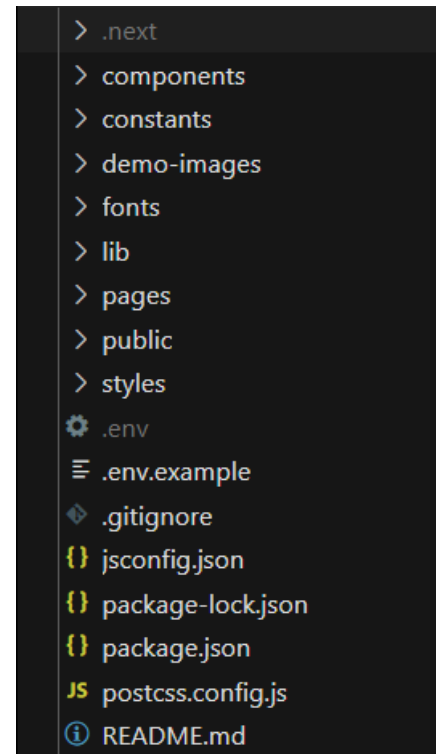
- Dark Mode, un bouton est présent sur la barre de navigation pour passer du mode sombre au mode éclairé.
- Next.js, intervient pour l'optimisation d'images et le support SEO
- TailwindCSS, permet de personnaliser entièrement le site selon mes besoins
- Composant Meta Personnalisé, chaque page contient un composant Meta pour des informations supplémentaires sur la page
- Design Responsive, les pages du site ont un rendu exceptionnel sur toutes les plateformes, mobiles, tablettes et web.

Il inclut également toutes les pages nécessaires dans un portfolio.

- Page Principale, une page pour le visiteur, la première chose qu'il verra en venant sur le site.
- About, une introduction brève sur ce que je fais, mes compétences techniques et mes liens vers les réseaux sociaux.
- Experience, toute mon expérience que j'ai pu accumuler au cours des années
- Projects, une grille avec tous mes projets.
- Contact, un formulaire de contact pour que quelqu'un puisse me joindre.

Le site entier est divisé en plusieurs composants, des bouts de code que je peux ré-utiliser à souhait dans une autre partie du site.

- components, la où se trouvent tous mes composants.
- public, là où se trouvent mes images, polices d'écritures, ect...
- styles, c'est ici que vont se trouver les scripts CSS et la librairie Tailwindcss.
- pages, toutes mes routes de l'applications sont présentes dans ce dossier, et c'est une des meilleures fonctionnalités de Next.js. Il suffit de créer un nouveau fichier dans pages et il servira de nouvelle route.



b. Extraits de code

Container Block

`<ContainerBlock />` est le parent de tous mes composants. Il permet d'avoir une balise meta personnalisable pour chaque page. J'ai désigné l'agencement pour qu'il accepte des props comme `children` et amène à chaque page une `Navbar`, des tags `<meta>` et un `Footer`.

```
1  import React from "react";
2  import Head from "next/head";
3  import { useRouter } from "next/router";
4  import Navbar from "../Navbar";
5  import Footer from "../Footer";
6
7  export default function ContainerBlock({ children, ...customMeta }) {
8    const router = useRouter();
9
10   const meta = {
11     title: "Clément Ramos - Developer, Student.",
12     description: "I've been developing websites for 1 years straight. Get in touch with me to know more.",
13     image: "/avatar.png",
14     type: "website",
15     ...customMeta,
16   };
17   return (
18     <div>
19       <Head>
20         <title>{meta.title}</title>
21         <meta name="robots" content="follow, index" />
22         <meta content={meta.description} name="description" />
23         <meta
24           property="og:url"
25           content={`https://yourwebsite.com${router.asPath}`}
26         />
27         <link
28           rel="canonical"
29           href={`https://yourwebsite.com${router.asPath}`}
30         />
31         <meta property="og:type" content={meta.type} />
32         <meta property="og:site_name" content="Dev' up!" />
33         <meta property="og:description" content={meta.description} />
34         <meta property="og:title" content={meta.title} />
35         <meta property="og:image" content={meta.image} />
36         <meta name="twitter:card" content="summary_large_image" />
37         <meta name="twitter:site" content="@ " />
38         <meta name="twitter:title" content={meta.title} />
39         <meta name="twitter:description" content={meta.description} />
40         {meta.date && (
41           <meta property="article:published_time" content={meta.date} />
42         )}
43       </Head>
44       <main className="dark:bg-gray-800 w-full">
45         <Navbar />
46         <div>{children}</div>
47         <Footer />
48       </main>
49     </div>
50   );
51 }
52
53
```

Le script ContainerBlock.js

Après avoir créé ContainerBlock.js, je peux mettre tous mes composant de ma page dans un bloc ContainerBlock :

```
1  import Head from "next/head";
2  import styles from "../styles/Home.module.css";
3  import ContainerBlock from "../components/ContainerBlock";
4  import FavouriteProjects from "../components/FavouriteProjects";
5  import LatestCode from "../components/LatestCode";
6  import Hero from "../components/Hero";
7  import getLatestRepos from "@lib/getLatestRepos";
8  import userData from "@constants/data";
9
10 export default function Home({ repositories }) {
11   return (
12     <ContainerBlock
13       title="Clément Ramos - Student, Developer, Creator"
14       description="This is my website. A portfolio for future jobs."
15     >
16       <Hero />
17       <FavouriteProjects />
18       <LatestCode repositories={repositories} />
19     </ContainerBlock>
20   );
21 }
22
23 export const getServerSideProps = async () => {
24   console.log(process.env.GITHUB_AUTH_TOKEN);
25   let token = process.env.GITHUB_AUTH_TOKEN;
26
27   const repositories = await getLatestRepos(userData, token);
28   // console.log("REPOSITORIES", repositories);
29
30   return {
31     props: {
32       repositories,
33     },
34   };
35 };
36
```

Exportation des composant d'application dans d'autres pages

Mode sombre

Le mode sombre est fourni par un paquetage npm appelé « next-themes ». L'objectif est d'envelopper le conteneur parent avec un fournisseur ThemeProvider via lequel le thème est disponible pour tous les enfants à tout moment.

Dans _app.js :

```
1  import "../styles/globals.css";
2  import { ThemeProvider } from "next-themes";
3
4  function MyApp({ Component, pageProps }) {
5    return (
6      <ThemeProvider defaultTheme="light" attribute="class">
7        <Component {...pageProps} />
8      </ThemeProvider>
9    );
10 }
11
12 export default MyApp;
13
```

Clément Ramos

Student - Full-Stack Developer

[About](#)

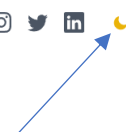
[Projects](#)

[Experience](#)

[Contact](#)



Le mode sombre, une fonctionnalité de plus en plus recherchée



Le bouton pour changer de thème entre sombre et éclairé est réutilisable partout dans tout le site web. Il est intégré à la NavBar.

Navbar.js :

```
188 <button
189   aria-label="Toggle Dark Mode"
190   type="button"
191   className="w-10 h-10 p-3 rounded focus:outline-none"
192   onClick={() => setTheme(theme === "dark" ? "light" : "dark")}
193 >
194   {mounted && (
195     <svg
196       xmlns="http://www.w3.org/2000/svg"
197       viewBox="0 0 24 24"
198       fill="currentColor"
199       stroke="currentColor"
200       className="w-4 h-4 text-yellow-500 dark:text-yellow-500"
201     >
202       {theme === "dark" ? (
203         <path
204           strokeLinecap="round"
205           strokeLinejoin="round"
206           strokeWidth={2}
207           d="M12 3v1m0 16v1m9-9h-1M4 12H3m15.364 6.364l-.707-.707"
208         />
209       ) : (
210         <path
211           strokeLinecap="round"
212           strokeLinejoin="round"
213           strokeWidth={2}
214           d="M20.354 15.354A9 9 0 018.646 3.646 9.003 9.003 0 000 0"
215         />
216       )}
217     </svg>
218   )}
219 </button>
```

La section Hero

La section Hero permet d'attirer l'attention des gens. J'ai utilisé react-rough-notation, une bibliothèque qui met dynamiquement en évidence le texte avec différentes couleurs et animation.

L'avantage principal est que l'utilisateur final prête immédiatement attention au texte mis en surbrillance.

Le code est simple : j'enveloppe le texte à mettre en surbrillance dans les balises

<RoughNotationGroup> et <RoughNotation> avec des paramètres supplémentaires telles que les couleurs et délais d'animation.

J'ai donc créé un composant RainbowHighlight qui prend une couleur et met en surbrillance le texte joint qui peut être utilisé partout.

RainbowHighlight.js :

```
1 import React from "react";
2 import { RoughNotation } from "react-rough-notation";
3
4 export const RainbowHighlight = ({ color, children }) => {
5   // Change the animation duration depending on length of text we're animating (speed = distance / time)
6   const animationDuration = Math.floor(30 * children.length);
7
8   return (
9     <RoughNotation
10       type="highlight"
11       multiline={true}
12       padding={[0, 2]}
13       iterations={1}
14       animationDuration={animationDuration}
15       color={color}
16     >
17       {children}
18     </RoughNotation>
19   );
20 };
21
```

Les couleurs dans Hero.js :

```
1 import React from "react";
2 import { RoughNotation, RoughNotationGroup } from "react-rough-notation";
3 import { RainbowHighlight } from "../RainbowHighlight";
4 import userData from "@constants/data";
5
6 export default function Hero() {
7   const colors = ["#F59E0B", "#84CC16", "#10B981", "#3B82F6"];
8   return (
9     <div className="flex flex-row justify-center items-start overflow-hidden">
10       {/* Text container */}
11
12       <div className="w-full md:w-1/2 mx-auto text-center md:text-left lg:p-20">
13         <RoughNotationGroup show={true}>
14           <RainbowHighlight color={colors[0]}>
15             <h1 className="text-4xl md:text-8xl font-bold text-gray-700 dark:text-gray-200 my-2">
16               Student.
17             </h1>
18           </RainbowHighlight>
19           <RainbowHighlight color={colors[1]}>
20             <h1 className="text-4xl md:text-8xl font-bold text-gray-700 dark:text-gray-200 my-2">
21               Developer.
22             </h1>
23           </RainbowHighlight>
24           <RainbowHighlight color={colors[2]}>
25             <h1 className="text-4xl md:text-8xl font-bold text-gray-700 dark:text-gray-200 my-2">
26               Programmer.
27             </h1>
28           </RainbowHighlight>
29           <RainbowHighlight color={colors[3]}>
30             <h1 className="text-4xl md:text-8xl font-bold text-gray-700 dark:text-gray-200 my-2">
31               Autodidact.
32             </h1>
33           </RainbowHighlight>
34         </RoughNotationGroup>
35       </div>
36       {/* Image container */}
37       <div className="hidden lg:block relative w-full md:w-1/2 -mr-40 mt-20">
38         <div className="w-3/4">
39           <img src={userData.avatarUrl} alt="avatar" className="shadow" />
40         </div>
41       </div>
42     </div>
43   );
44 }
```

Inclusion des projets dans mon Portfolio

J'ai gardé la section des projets aussi simple que possible avec une énorme zone d'image, car le recruteur / l'utilisateur final est le plus intéressé à voir cette dernière.

J'ai divisé la page en grilles Tailwind de deux colonnes, qui se divisent sur les écrans mobiles en 1 colonne.

Le conteneur d'image contient un texte d'en-tête qui affiche le nom du projet et un numéro en bas. Les animations de survol sur les images sont subtiles.

L'image se redimensionne lentement pour attirer l'attention de l'utilisateur.

Au clic, l'utilisateur accède au site Web en direct / au référentiel GitHub du projet.

Projects.js :

```
1  import React from "react";
2  import userData from "@constants/data";
3
4  export default function Projects() {
5    return (
6      <section className="bg-white dark:bg-gray-800">
7        <div className="max-w-6xl mx-auto h-48 bg-white dark:bg-gray-800">
8          <h1 className="text-5xl md:text-9xl font-bold py-20 text-center md:text-left">
9            Projects
10          </h1>
11        </div>
12        { /* Grid starts here */ }
13        <div className="bg-[#F1F1F1] dark:bg-gray-900">
14          <div className="max-w-6xl mx-auto grid grid-cols-1 md:grid-cols-2 gap-8 py-20 pb-40">
15            {userData.projects.map((proj, idx) => (
16              <ProjectCard
17                title={proj.title}
18                link={proj.link}
19                imgUrl={proj.imgUrl}
20                number={` ${idx + 1}` }
21              />
22            ))}
23          </div>
24        </div>
25      </section>
26    );
27  }
28
29  const ProjectCard = ({ title, link, imgUrl, number }) => {
30    return (
31      <a href={link} className="w-full block shadow-2xl">
32        <div className="relative overflow-hidden">
33          <div className="h-72 object-cover">
34            <img
35              src={imgUrl}
36              alt="portfolio"
37              className="transform hover:scale-125 transition duration-2000 ease-out object-cover h-full w-full"
38            />
39          </div>
40          <h1 className="absolute top-10 left-10 text-gray-50 font-bold text-xl bg-green-500 rounded-md px-2">
41            {title}
42          </h1>
43          <h1 className="absolute bottom-10 left-10 text-gray-50 font-bold text-xl ">
44            {number.length === 1 ? "0" + number : number}
45          </h1>
46        </div>
47      </a>
48    );
49  };
50
```

On récupère ensuite les informations dans data.js :

```
9   projects: [  
10     {  
11       title: "Terrablock",  
12       link: "https://terrablock.fr",  
13       imgUrl: "/terrablock.png"  
14     },  
15     {  
16       title: "ECF Studi",  
17       link: "https://hypnoshotels.online",  
18       imgUrl: "/ecf.png"  
19     },  
20     {  
21       title: "Tower Blocks",  
22       link: "",  
23       imgUrl: "/towerblocks.png"  
24     },  
25     {  
26       title: "Weather Station",  
27       link: "",  
28       imgUrl: "/meteo.png"  
29     },  
30     {  
31       title: "Mathieu Dumas - Thérapeute",  
32       link: "",  
33       imgUrl: "/comingsoon.jpg"  
34     },  
35   ]
```

Création de la page de contact :

J'ai pris la section de contact directement du Tailwind Master Kit, qui est un marché de composants et de modèles pour les projets d'applications Web Tailwind.

Je ne voulais pas passer plus de temps à créer moi-même un formulaire de contact et j'ai utilisé de l'aide.

Le composant est absolument gratuit et facilement intégrable aux sites Web liés à Tailwind.

Contact.js :

```
1  import React from "react";
2  import userData from "@constants/data";
3  import { useState } from "react";
4
5  export default function Contact() {
6    const [name, setName] = useState("");
7    const [enterprise, setEnterprise] = useState("");
8    const [email, setEmail] = useState("");
9    const [message, setMessage] = useState("");
10   const [submitted, setSubmitted] = useState(false);
11   const handleSubmit = (e) => {
12     e.preventDefault()
13     console.log("Sending")
14     let data = {
15       name,
16       enterprise,
17       email,
18       message
19     }
20     fetch('/api/contact', {
21       method: 'POST',
22       headers :{
23         'Accept': 'application/json, text/plain, */*',
24         'Content-Type': 'application/json'
25       },
26       body: JSON.stringify(data)
27     }).then((res) => {
28       console.log("Received!")
29       if (res.status === 200) {
30         console.log("Reponse Succeeded!")
31         setSubmitted(true)
32         setName('')
33         setEnterprise('')
34         setEmail('')
35         setMessage('')
36       }
37     })
38   }
```

Dans l'ordre : il s'agit de variables d'état contenant le nom, l'entreprise, l'e-mail et le message de l'utilisateur, ainsi qu'une valeur booléenne indiquant si le message a été soumis ou non.

Nous devons maintenant transmettre ces valeurs des champs d'entrée aux variables d'état.

```

<label htmlFor="name" className="text-sm text-gray-600 mx-4">
  {""}
  Name
</label>
<input
  type="text"
  className="font-light rounded-md border focus:outline-none py-2 mt-2 px-1 mx-4 focus:ring-2 focus:border-none ring-blue-500"
  name="name"
  required
  placeholder="Your beautiful name"
  value={name}
  onChange={(e) => {setName(e.target.value);}}
/>
<label htmlFor="text" className="text-sm text-gray-600 mx-4 mt-4">

```

Commençons par la fonction onChange.

'onChange' est un attribut que nous définissons à l'intérieur des balises HTML, telles que les champs de saisie, qui exécutera une fonction chaque fois que la valeur de cette balise change - comme, par exemple, lorsque l'utilisateur tape son nom dans une zone vide.

À l'intérieur de cette fonction onChange, nous pouvons utiliser nos fonctions de définition de variable d'état pour transmettre la valeur du champ d'entrée modifié à l'état, capturant ainsi l'entrée de l'utilisateur.

Le 'onChange={(e)=>{setName(e.target.value)}}

 est la ligne importante ici.

Il indique à Next.js que, lorsque ce champ de saisie enregistre un changement, il doit exécuter cette fonction anonyme.

Notre variable 'e' contient des informations sur notre événement, que nous devons transmettre au setter afin de capturer la valeur du champ d'entrée.

La valeur e.target.value prend cette valeur et la définit comme la nouvelle valeur de notre variable name dans state.

Traitement des données envoyées :

Lorsque l'utilisateur appuie sur le bouton d'envoi, nous devons prendre les informations qu'il nous a fournies et en FAIRE quelque chose.

Un formulaire qui prend des données puis les vide sans action est un formulaire plutôt inutile.

Maintenant, nous devons faire appel à Fetch afin d'acheminer ces données quelque part.

Tout ce dont nous avons besoin est, lorsque l'utilisateur soumet, de prendre ses données et de les publier sur un API.

```

11  const handleSubmit = (e) => {
12    e.preventDefault()

```

Nous n'avons pas besoin de transmettre quoi que ce soit à cette fonction en dehors de l'événement - tout ce dont elle a besoin, elle le tirera de State.

Je crée un e.preventDefault(), pour que la page ne se recharge pas à chaque fois.

Maintenant, avec la requête Fetch :

```
14     let data = {
15         name,
16         enterprise,
17         email,
18         message
19     }
20     fetch('/api/contact', {
21         method: 'POST',
22         headers :{
23             'Accept': 'application/json, text/plain, */*',
24             'Content-Type': 'application/json'
25         },
26         body: JSON.stringify(data)
```

Qu'est-ce que tout cela fait ?

Il s'agit d'un appel Fetch à l'url `"/api/contact"`, auquel je reviendrais plus tard.

Nous en avons besoin pour publier sur cette URL, et nous devons envoyer le JSON de notre objet de données, qui contient des informations sur l'utilisateur telles que le nom et l'e-mail.

Enfin, nous voudrions gérer ce qui devrait se passer lorsque cet appel est renvoyé, afin que nous puissions montrer à l'utilisateur que son action a abouti :

```
    fetch('/api/contact', {
        method: 'POST',
        headers :{
            'Accept': 'application/json, text/plain, */*',
            'Content-Type': 'application/json'
        },
        body: JSON.stringify(data)
    }).then((res) => {
        console.log("Received!")
        if (res.statusCode === 200) {
            console.log("Reponse Succeeded!")
            setSubmitted(true)
            setName('')
            setEnterprise('')
            setEmail('')
            setMessage('')
        }
    })
}
```

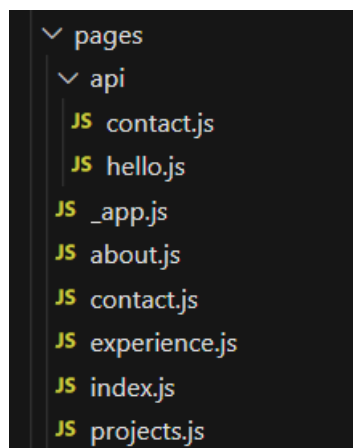
Avec un `.then` enchaîné à la fin de l'appel Fetch, nous réinitialisons nos valeurs à null et définissons la valeur true.

Maintenant, enveloppons notre fonction `handleSubmit` dans un `onClick` dans le bouton de soumission.

```
<button
  type="submit"
  className="bg-blue-500 rounded-md w-1/2 mx-4 mt-8 py-2 text-gray-50 text-xs font-bold"
  onClick={(e) => {handleSubmit(e)}}
>
  Send Message
</button>
```

Cela exécutera la fonction `handleSubmit` chaque fois que nous cliquons sur soumettre, de sorte que l'entrée de l'utilisateur sera collectée et publiée par l'appel `Fetch` à notre URL `/api/contact`.

Création de la route de l'API



Maintenant, à l'intérieur de `contact.js`, je crée une nouvelle fonction d'exportation par défaut, avec `(req, res)` comme paramètres.

Maintenant, je teste que les composants d'application communiquent comme ils le devraient.

```
event - compiled client and server successfully in 5.2s (133 modules)
wait - compiling / (client and server)...
wait - compiling...
event - compiled client and server successfully in 998 ms (225 modules)
undefined
/
/
undefined
  accepted: [ 'ramos.clement@outlook.fr' ],
  rejected: [],
  envelopeTime: 616,
  messageTime: 1175,
  messageSize: 713,
  response: '250 2.0.0 OK 1652373137 n6-20020a1c270600000b003942a244ec0sm191980wmn.5 - gmailtp',
  envelope: {
    from: 'ramosclement.pro@gmail.com',
    to: [ 'ramos.clement@outlook.fr' ]
  },
  messageId: '<02903e22-b6e2-2822-2b51-cde4060caad0@gmail.com>'
}
```

Installation de Nodemailer

J'utilise la commande `npm i nodemailer` pour installer le paquetage.

```
PS C:\Users\ramos\OneDrive\Bureau\Dev\ up> npm i nodemailer
```

Maintenant, dans ma route `api contact.js`, j'importe Nodemailer et crée un objet appelé "transporter" avec `nodemailer.createTransport()`

```
1 export default function(req, res) {
2   require('dotenv').config()
3
4   let nodemailer = require('nodemailer')
5   const transporter = nodemailer.createTransport({
```

L'objet transporteur est, essentiellement, l'objet qui stocke toutes les informations sur la façon dont nous voulons envoyer nos e-mails - quel compte utilisons-nous ? Quel fournisseur ? Quel port ? Nous définirons tout cela sur l'objet transporteur, comme suit :

```
5   const transporter = nodemailer.createTransport({
6     port: 465,
7     host: "smtp.gmail.com",
8     auth: {
9       user: 'ramosclement.pro@gmail.com',
10      pass: process.env.password,
11    },
12    secure: true,
13  });
```

Voici les principales caractéristiques de ce transporteur : port, hôte et auth.

Le port fait référence au « canal » de communication que l'e-mail utilisera - 465 est pour la communication SMTP, que nous désignons dans la ligne « hôte » en dessous.

Ce qui importe, c'est **l'authentification**.

Notez qu'il contient des propriétés pour 'user' et 'pass' - il s'agit du nom d'utilisateur et du mot de passe du compte à partir duquel nous enverrons nos e-mails.

Oui — **Pour des raisons de sécurité évidentes**, je ne peux pas envoyer d'e-mail depuis mon compte directement.

Ce que je peux faire, cependant, c'est prendre leur message et nous l'envoyer par e-mail via un compte fictif que j'ai créé.

Mais n'y a-t-il pas une autre mesure de sécurité que je devrais prendre ?

Le mot de passe : c'est un compte fictif, bien sûr, mais je n'ai pas vraiment envie de mettre mon mot de passe dans mon code, en plus en **texte brut**.

Pour donner à mon compte un peu plus de sécurité, je dois configurer la route API pour extraire le mot de passe en tant que variable d'environnement. C'est possible avec dotenv.

```
PS C:\Users\ramos\OneDrive\Bureau\Dev\ up> npm i dotenv
```

Je crée un fichier d'environnement appelé .env et je stocke mon mot de passe à l'intérieur. Je l'ajoute ensuite dans mon fichier .gitignore.

Ensuite, au sein de mon fichier contact.js de mon API, j'appelle le fichier :

```
require('dotenv').config()
```

L'envoi de mail

Donc, j'ai mon compte fictif tout configuré, le formulaire, et l'API envoyant cette entrée côté serveur. La dernière chose à faire est de tout faire fonctionner.

Pour en revenir au code contact.js, il nous reste deux dernières choses à faire : créer l'e-mail que nous aimerions envoyer, puis l'envoyer !

```
const mailData = {
  from: 'ramosclement.pro@gmail.com',
  to: 'ramos.clement@outlook.fr',
  subject: `Nouveau message de ${req.body.name}`,
  text: req.body.message + " | Envoyé depuis: " + req.body.email,
  html: `

${req.body.message}</div><p>Envoyé depuis : ${req.body.email}</p>`
}


```

Nous créons un nouvel objet, appelé « mailData », qui contient cinq champs : un From, un To, un Subject, Text et HTML.

Ce sont les champs de saisie d'un e-mail.

From est l'adresse qui l'envoie (notre compte fictif).

To est le destinataire.

Subject est l'objet avec lequel l'e-mail sera envoyé.

Et Text est le corps en texte brut du message, avec HTML permettant le formatage avec des balises <div> et autres.

Grâce à la route API, nous pouvons prendre les informations saisies par l'utilisateur sur la page du formulaire de contact et les injecter dans l'e-mail que nous souhaitons envoyer. J'utilise simplement req.body.name, req.body.email et req.body.message pour chaque variable respective.

Nous pouvons également styliser le contenu comme nous le souhaitons sous la propriété HTML, afin de le rendre plus lisible.

Maintenant, nous avons configuré l'expéditeur et l'e-mail qu'il doit envoyer. Le dernier élément majeur est en fait l'envoi de cet e-mail !

Heureusement, ce n'est qu'un simple appel de fonction.

```
29     transporter.sendMail(mailData, function(err, info) {
30         if (err)
31             console.error(err)
32         else
33             console.log(info)
34     })
35 }
```

Cette fonction enverra l'e-mail, et si une sorte d'erreur survient, elle enverra cette erreur à la console pour le debug.

Et enfin, puisque tout cela est rendu possible grâce à un appel API, nous devons terminer la route par un `.send`, afin que notre frontal sache si l'appel a réussi ou non.

```
res.status(200)
```

Terrablock : formulaire de contact (partie backend)

Pour couvrir le référentiel de l'activité type 2, je vais maintenant expliquer mes scripts d'un formulaire de contact en PHP ainsi qu'un stockage des informations personnelles en base de données.

```
<div class="hero__p-contain">
  <form id="contact-form" style="font-size: xx-large;" method="post">
    <label for="first_name" class="fade-up tricks">First Name</label>
    <input type="text" class="hero__button w-inline-block" id="first_name" name="first_name" required>
    <label for="name" class="fade-up tricks">Name</label>
    <input type="text" class="hero__button w-inline-block" id="name" name="name" required>
    <label for="company" class="fade-up tricks">Company</label>
    <input type="text" class="hero__button w-inline-block" id="company" name="company" required>
    <label for="phone" class="fade-up tricks">Phone</label>
    <input type="tel" class="hero__button w-inline-block" id="phone" name="phone" required>
    <label for="email" class="fade-up tricks">Email</label>
    <input type="email" class="hero__button w-inline-block" id="email" name="email" required>
    <label for="subject" class="fade-up tricks">Write us</label>
    <textarea name="subject" id="subject" class="hero__button w-inline-block" required></textarea>
    <br>
    <button class="hero__button-contain" id="submit" type="submit" name="send" value="SEND"><img src="h
  </form>
</div>
```

Dans ma page `index.php`, je commence par créer un formulaire de contact basique dynamique avec des ids bien précis qui me seront utiles pour récupérer les informations rentrées par l'utilisateur.

**PLEASE GET IN TOUCH AND OUR TEAM
WILL ANSWER YOU ASAP**

FIRST NAME

NAME

```
1  <?php
2  //Base de donnée
3  if(!empty($_POST["send"])) {
4      $first_name = $_POST["first_name"];
5      $name = $_POST["name"];
6      $company = $_POST["company"];
7      $phone = $_POST["phone"];
8      $email = $_POST["email"];
9      $subject = $_POST["subject"];
10
```

Via cette partie du script PHP, je vérifie que l'utilisateur ai bien rentré les informations demandées avant de poursuivre l'exécution du script.

Si tout est bon, je crée une variable pour me connecter à ma base de données précédemment créée et j'injecte une requête SQL pour envoyer les données personnelles de l'utilisateur dans ma base de données que j'aurais choisie.

La création de la base de données :

```
CREATE TABLE CONTACT(
    first_name VARCHAR(255) NOT NULL ,
    name VARCHAR(255) NOT NULL ,
    company VARCHAR(255) NOT NULL ,
    phone VARCHAR(15) PRIMARY KEY NOT NULL,
    email VARCHAR(255),
    subject TEXT
);
```


La connexion et l'insertion dans la base de données :

```
11 $connexion = mysqli_connect("localhost", "root", "", "terrablock") or die("Erreur de connexion: " . mysqli_error($connexion));
12 $subject_esc = $connexion->real_escape_string($subject);
13 $result = mysqli_query($connexion, "INSERT INTO contact (first_name, name, company, phone, email, subject) VALUES
14 ('" . $first_name . "', '" . $name . "', '" . $company . "', '" . $phone . "', '" . $email . "', '" . $subject_esc . "')");
```

De plus, pour les besoins du projet, j'ai dû créer un second script pour permettre au mailer PHP d'envoyer un courriel aux administrateurs du site web pour leur informer d'une demande de contact.

```
if(!empty($_POST["send"])){
    $first_name = $_POST["first_name"];
    $name = $_POST["name"];
    $company = $_POST["company"];
    $phone = $_POST["phone"];
    $email = $_POST["email"];
    $subject = $_POST["subject"];
    $object = "Nouvelle demande de contact";
    $headers .= "MIME-Version: 1.0\r\n";
    $headers .= "Content-Type: text/html; charset=ISO-8859-1\r\n";

    $toEmail = "ramos.clementjoseph@gmail.com";
    $message = "<div style='text-align: center;'><h1>Nouvelle demande de contact.</h1></div><br><img src='https://i.ibb.
    if(mail($toEmail, $object, $message, $headers)) {
        $mail_msg = "<h4><i>Thanks, a team member will be notified and come back to you soon.</i></h4>";
        $type_mail_msg = "success";
    }else{
        $mail_msg = "<h3 style='color: red;'><i>Sorry, there's a server maintenance, please try again later.</i></h3>";
        $type_mail_msg = "error";
    }
}
```

J'affiche ensuite à l'utilisateur un message de succès ou d'erreur :

```
if(mail($toEmail, $object, $message, $headers)) {
    $mail_msg = "<h4><i>Thanks, a team member will be notified and come back to you soon.</i></h4>";
    $type_mail_msg = "success";
}else{
    $mail_msg = "<h3 style='color: red;'><i>Sorry, there's a server maintenance, please try again later.</i></h3>";
    $type_mail_msg = "error";
}
```

```
<label for="statusMessage" class="fade-up tricks">
    <?php if(! empty($mail_msg)) { ?>
        <p><?php echo $mail_msg; ?></p>
    <?php } ?>
</label>
```

Le message de succès :

THANKS, A TEAM MEMBER WILL BE NOTIFIED AND COME BACK TO YOU SOON.

Ensuite, les administrateurs reçoivent un mail :

Nouvelle demande de contact.



Bonjour, vous avez reçu une nouvelle demande de contact de Clément RAMOS .

Cette personne travaille pour l'entreprise Fenix .

Voici ses informations personnelles :

Prénom : Clément

Nom de famille : RAMOS

Compagnie : Fenix

Numéro de téléphone : 0752635269

Adresse Mail : ramosantoinette@outlook.fr

Sujet du message :

Bonjour, Je souhaite avoir un renseignement ! Merci de me recontacter.

Exemple de mail reçu

c. Les protocoles de sécurité

Utilisant Next.js pour ce projet, voici les options de sécurité qui s'offrent à moi.

Pour améliorer la sécurité de mon application, je peux utiliser des en-têtes dans next.config.js pour appliquer des en-têtes de réponse HTTP à toutes les routes de mon application.

Par exemple, pour une X-XSS-Protection :

Cet en-tête arrête le chargement des pages lorsqu'elles détectent des attaques de type cross-site scripting (XSS).

```
{  
  key: 'X-XSS-Protection',  
  value: '1; mode=block'  
}
```

d. Le déploiement de l'application « Dev' up ! »

C'est assez simple et ne nécessite que 2 étapes simples.

1. « Push » le code sur GitHub

```
git add *  
git commit -m "first commit"  
git push
```

2. Depuis le compte Vercel, ajouter le repo GitHub et Vercel va automatiquement déployer le site et donner un lien pour le visiter.

La fonctionnalité la plus représentative : La récupération des informations GitHub

1. Explications et jeu d'essai

Pour récupérer mes données GitHub (mes « repositories »), j'ai dû me servir du GitHub API. Cette API permet de récupérer toutes sortes de données sur un utilisateur GitHub. La seule fonctionnalité de cette API qui m'intéressait était le fait de pouvoir récupérer les données de mes derniers « repositories » pour les inclure dans mon site web.

Pour cela, j'ai dû utiliser OAuth. OAuth utilise des tokens avec deux fonctionnalités :

- Accès révocables : l'utilisateur peut révoquer un accès à une application tierce à tout moment.
- Accès limité : l'utilisateur peut revoir l'accès spécifique qu'un token va produire avant d'autoriser une application spécifique.

Un token OAuth est comme un mot de passe.

Par exemple, pour récupérer, GET, les informations de mes repositories, je peux utiliser la commande suivante :

```
$ curl -i https://api.github.com/repos/clementramos/dev-up
```

Pour ensuite voir les informations via un utilisateur authentifié :

```
$ curl -i -H "Authorization: token ghp_16C7e42F292c6912E7710c838347Ae178B4a" \> https://api.github.com/user/repos
```

Dans mon cas, dans mon script getLatestRepos.js :

```
const username = data.githubUsername;

// let token = `token ${process.env.GITHUB_AUTH_TOKEN}`;
// console.log("TOKEN", token);
```

Je récupère mon nom d'utilisateur GitHub se trouvant dans dossier data.

```
if (token) {
  const res = await axios.get(
    `https://api.github.com/search/repositories?q=user:${username}+sort:author-date-asc`,
    {
      headers: {
        Authorization: `token ${token}`,
      },
    },
  );
}
```

Ensuite, via le GitHub API, nous pouvons récupérer les informations dans l'ordre que nous voulons.

Une fois que j'ai récupéré les informations voulus, je vais couper l'array que j'ai obtenu, afin d'obtenir uniquement mes 6 derniers repositories.

```
let repos = res.data.items;
let latestSixRepos = repos.splice(0, 6);
// console.log("LATEST 6 repos", latestSixRepos);
return latestSixRepos;
```

Une fois toutes les données récupérées, je peux les utiliser dans mon composant React `<GetReposCard />` et les passer en paramètres :

```
<div className="grid grid-cols-1 md:grid-cols-2 lg:grid-col
  { /* Single github Repo */ }

  {repos &&
    repos.map((latestRepo, idx) => (
      <GithubRepoCard latestRepo={latestRepo} key="idx" />
    ))}
</div>
```

```
const GithubRepoCard = ({ latestRepo }) => {
  return (
    <div className="github-repo">
      <h1 className="font-semibold text-xl dark:text-gray-200 text-gray-700">
        {latestRepo.name}
      </h1>
      <p className="text-base font-normal my-4 text-gray-500">
        {latestRepo.description}
      </p>
      <a
        href={latestRepo.clone_url}
        className="font-semibold group flex flex-row space-x-2 w-full items-center"
      >
        <p>View Repository </p>
        <div className="transform group-hover:translate-x-2 transition duration-300">
          &rarr;
        </div>
      </a>
    </div>
  );
};
```

Je récupère les informations et les affiche à l'utilisateur

2. Recherches anglophones sur les requêtes au sein de l'API GitHub

Etant donné que je n'avais que très peu abordé la notion d'authentification et de sécurité d'une API lors de ma formation, je me suis donc documenté sur le fait de pouvoir récupérer des requêtes d'un compte GitHub. Voici un extrait du site officiel Github.com, ainsi que la traduction.

Requests from personal accounts

Direct API requests that you authenticate with a personal access token are user-to-server requests. An OAuth App or GitHub App can also make a user-to-server request on your behalf after you authorize the app. For more information, see "[Creating a personal access token](#)," "[Authorizing OAuth Apps](#)," and "[Authorizing GitHub Apps](#)."

GitHub associates all user-to-server requests with the authenticated user. For OAuth Apps and GitHub Apps, this is the user who authorized the app. All user-to-server requests count toward the authenticated user's rate limit.

User-to-server requests are limited to 5,000 requests per hour and per authenticated user. All requests from OAuth applications authorized by a user or a personal access token owned by the user, and requests authenticated with any of the user's authentication credentials, share the same quota of 5,000 requests per hour for that user.

User-to-server requests are subject to a higher limit of 15,000 requests per hour and per authenticated user in the following scenarios.

- The request is from a GitHub App that's owned by a GitHub Enterprise Cloud organization.
- The request is from an OAuth App that's owned or approved by a GitHub Enterprise Cloud organization.

For unauthenticated requests, the rate limit allows for up to 60 requests per hour. Unauthenticated requests are associated with the originating IP address, and not the person making requests.

Demandes provenant de comptes personnels

Les requêtes API directes que vous authentifiez avec un jeton d'accès personnel sont des requêtes utilisateur-serveur. Une application OAuth ou une application GitHub peut également effectuer une demande utilisateur-serveur en votre nom après avoir autorisé l'application. Pour plus d'informations, consultez « Créer un jeton d'accès personnel », « Autoriser les applications OAuth » et « Autoriser les applications GitHub ».

GitHub associe toutes les requêtes utilisateur-serveur à l'utilisateur authentifié. Pour les applications OAuth et GitHub, il s'agit de l'utilisateur qui a autorisé l'application. Toutes les requêtes utilisateur-serveur sont prises en compte dans la limite de débit de l'utilisateur authentifié.

Les requêtes utilisateur-serveur sont limitées à 5 000 requêtes par heure et par utilisateur authentifié. Toutes les demandes provenant d'applications OAuth autorisées par un utilisateur ou un jeton d'accès personnel appartenant à l'utilisateur, et les demandes authentifiées avec l'un des identifiants d'authentification de l'utilisateur, partagent le même quota de 5 000 demandes par heure pour cet utilisateur.

Les requêtes utilisateur-serveur sont soumises à une limite supérieure de 15 000 requêtes par heure et par utilisateur authentifié dans les scénarios suivants.

- La demande provient d'une application GitHub appartenant à une organisation GitHub Enterprise Cloud.
- La demande provient d'une application OAuth détenue ou approuvée par une organisation GitHub Enterprise Cloud.

Pour les demandes non authentifiées, la limite de débit autorise jusqu'à 60 demandes par heure. Les demandes non authentifiées sont associées à l'adresse IP d'origine, et non à la personne qui fait les demandes.

Conclusion

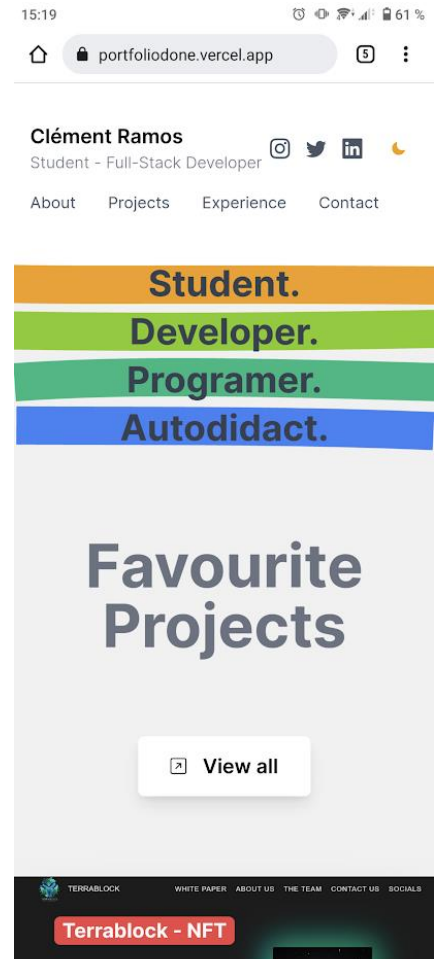
Concernant le projet « Dev' up ! », les principales fonctionnalités sont développées. J'estime être à 70% de l'achèvement complet du site web. Je me donne encore jusqu'à fin juin pour continuer le développement afin de proposer une version stable et finie dont le déploiement complet sera prévu à la rentrée 2023.

En commençant ce site web, je savais qu'il s'agirait d'un défi personnel étant donné que je n'avais jamais utilisé Next.js pour créer un site web ou TailwindCSS en tant que framework CSS. Je n'étais habitué qu'à utiliser React Native et le framework Bootstrap. Je suis finalement satisfait du travail que j'aurais pu accomplir malgré mon manque de connaissances du sujet.

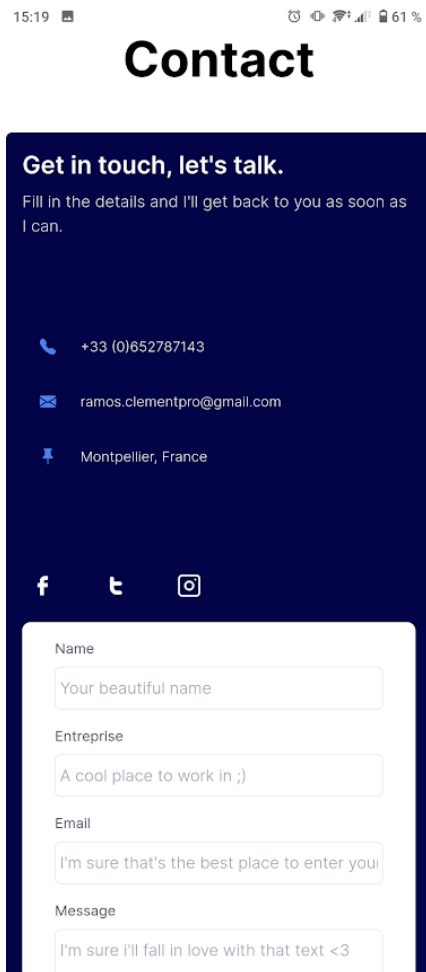
En effet, le développement web est un défi de tous les jours, il faut apprendre constamment les nouveaux langages, lire la documentation officielle, s'aider des forums dédiés, de tutoriels vidéo sur YouTube, pour pouvoir reproduire ce que j'ai pu apprendre sur mon projet « Dev' up ! »

Je pense que c'est la meilleure méthode de travail afin de pouvoir progresser rapidement, c'est également une bonne façon d'entrevoir ce qu'est le métier de développeur web : un apprentissage constant.

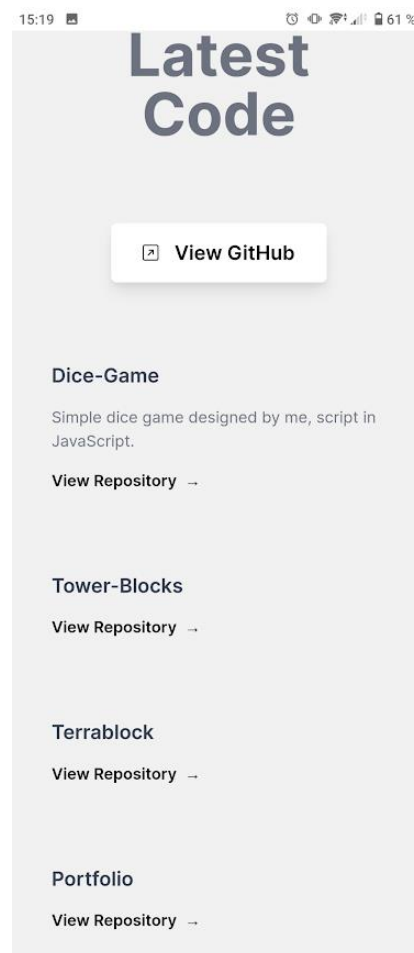
Annexes



La page d'accueil en version sombre et éclairée



La page de contact



La page des dépôts github

About Me.

I'm a student in software development that loves building products and web applications. I constantly want to learn new stuff and develop new applications.

Currently working on **Mathieu Dumas - Thérapeute** 📧

Contact

For any sort of help / enquiry, shoot a [mail](#) and I'll get back. I swear.

Job Opportunities

I'm looking for an internship currently, If you see me as a good fit, check my [CV](#) and I'd love to work for you.

Social Links

- Facebook
- Twitter
- GitHub
- LinkedIn
- Instagram

I've been developing full-stack application since I was 16 years old. I didn't know what full-stack meant at that time because the term was not coined back then.

After learning HTML and struggling with CSS, I came up with a brilliant idea of using bootstrap so that I don't have to style everything by myself and - for obvious reasons - if you knew bootstrap, you were cool.

After that, I decided to start learning JavaScript and it was really hard for me in the beginning, but now I'm learning Next.js, Node.js and React Native !

But now, I'm coding in TypeScript, React, Vue, Node.js, Next.js, Express, MongoDB, MySQL, Bootstrap, Tailwind, C++, C#, PHP, Ruby on Rails and I'm a beginner in AWS infrastructure. Although I barely have time to learn other languages, I'm loving what I'm doing.

Tech Stack



La page « About »

Experience

2018 - Now

Assistant d'éducation (TICE)

Collège Simone Veil

Contributed to merging the old website to a new one from a CMS designed by the Academy of Montpellier

2022

Developer

Terrablock

Developed a landing page for an NFT project. Smart contract on the blockchain to come later. Techno used : Javascript, PHP.

2022

Creator and Developer

Hypnos Hotels

Created a website for an evaluation. The website is still under construction. Techno used : HTML and CSS with Bootstrap, PHP and MongoDB for the database

2021

Graduation

UM3 - Montpellier

Bachelor's degree in clinical Psychology and Psycho-crime. Brief on cybercrimes : new issues in data protection.

La page « Experience »