

Rapport du « Technologie WEB » (S1)

Atelier Site WEB de CV

Réalisé par :

Etudiant1 ASSIAR Ilyas

Etudiant2 EL BOUR Othmane

Encadré par :

Mme. Asmaâ RETBI



Table of Contents

01 *Introduction*

02 *Analyse des besoins et cahier de charge*

03 *Conception et Structure de l'application*

04 *Réalisation technique*

05 *Les données du CV*

06 *Profile*

07 *Resume body*

08 *Footer*

09 *Hébergement*

Introduction

Au cours de ces 6 semaines de travaux sur le projets web, et après avoir appris les base de l'HTML5, CSS, et JavaScript. L'étape suivante est le React. Ce framework permet de simplifier le développement web. Rendant les pages plus dynamique que jamais. C'est ainsi que nous nous somme penché sur la réalisation de deux pages web. Un CV et un Formulaire. Et ce rapport détail les différentes fonctionnalités de ce projet de développement web.

Analyse des besoins et cahier de charges

Le site web de CV est une plateforme moderne et intuitive, conçue pour présenter au mieux les compétences, les réalisations et les expériences professionnelles. Voici une description détaillée de ses principales fonctionnalités:

La page d'accueil du site web est conçue pour donner une impression rapide et positive de la personne. Elle présente:

- Un visuel accrocheur ou une photo de profil.

- Un titre (par exemple dans notre cas le nom, le métier).

- Des liens rapides vers l'Email ,les page de LinkedIn ,et GitHub.

- Une partie "About me " présentant un résumé court et efficace des compétences, réalisations.

- Une partie " Expériences" présentant les expériences professionnelles.

- Une partie " Formation" présentant les formations scolaires et Universitaires.

- Une partie " Skills " présentant les compétences de la personne.

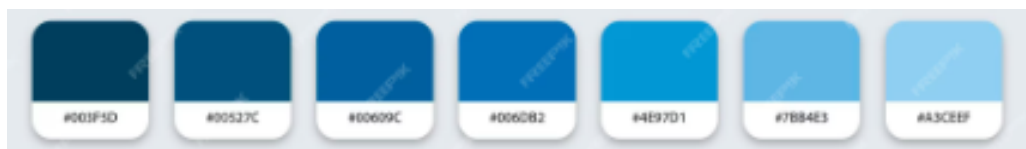
- Un Footer présentant l'appartenance su site le copyright et la date de création

Analyse des besoins et cahier de charges

. La conception graphique et ergonomique:

nous avons respecter certaines règles de nommage et de codage que nous nous sommes définis. Un jargon propre à chaque partie du site, il permet une compréhension plus rapide et une utilisation plus simple de l'application.

Il y'avait une exigence concernant le design et les couleurs de l'application. Quand on dit Computer science il existe certains couleurs qui nous viennent en tete et la on commence par la couleur bleu, pour cela nous allons essayer de rester dans le thème et jouer sur la palette de couleur bleu.



structure de l'application:

Afin de produire une application Cv facilement lisible et compréhensible on a opté pour une structure bien définis qui met en valeur chaque élément du Cv pour que toutes les parties soient cohérentes

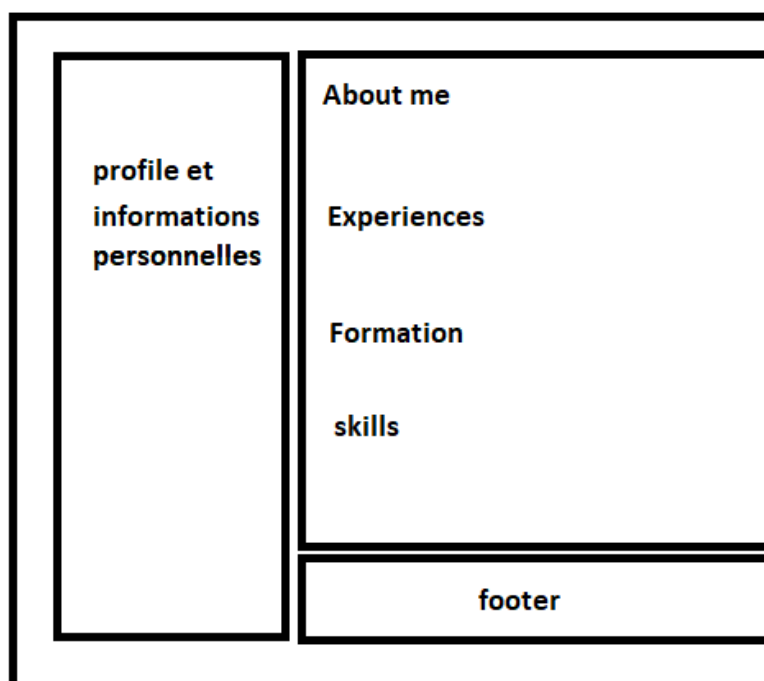


figure: structure du Cv

Analyse des besoins et cahier de charges

Le site web de Formulaire de son côté est une plateforme simple et interactive, conçue pour avoir une bonne lisibilité via un design simpliste mais pour pouvoir gagner une performance et ainsi assurer la collecte efficace des données.

La page du site web est conçue pour donner une impression rapide et positive de la personne. Elle présente:

- Un titre en exposition.

- Une partie "Informations personnelles" constituée des informations sur le nom, le prénom, l'âge etc...

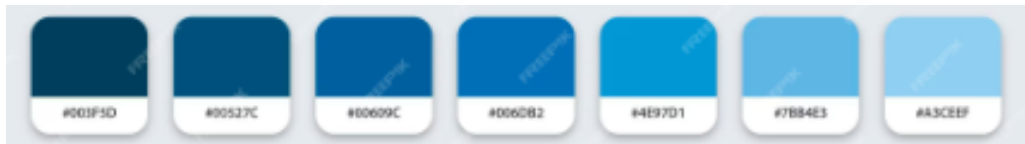
- Une partie compétences exposants le parcours scolaire, l'expérience professionnelles ainsi que les différentes compétences.

Analyse des besoins et cahier de charges

. La conception graphique et ergonomique:

nous avons encore la du respecter certaines règles de nommage et de codage que nous nous sommes définis.

Il y'avait une exigence concernant le design et les couleurs de l'application. Nous avons mis en place une palette de couleurs similaire au CV



Structure de l'application:

Afin de produire un formulaire facilement lisible et intuitif d'utilisation, on a opté pour le structure structure suivante:

TITLE
<div>Informations Personnelles</div> <div>Nom</div> <div>Prénom</div> <div>.</div> <div>.</div> <div>.</div>
<div>Formations et Compétences</div> <div>Formation</div> <div>Experience professionnelle</div> <div>Langes</div> <div>.</div> <div>.</div> <div>.</div>
FOOTER

Figure: Structure du formulaire

Réalisation technique

Les exigences de l'application et les besoins impliquent l'utilisation de certains langages:

- **JAVASCRIPT/TYPESCRIPT:**

JavaScript peut être utilisé afin de mettre à jour le contenu ou l'apparence d'une page sans avoir à recharger la page.

- **Css:**

Afin de manipuler la présentation, nous avons utilisé des feuilles de style CSS. Voici les raisons :

D'une part, il permet d'alléger le code source , puisque tout ce qui est relatif à la présentation est géré dans un fichier séparé. Et d'autre part, il permet de nous retrouver plus facilement dans notre code et ainsi facilite les modifications à effectuer, puisqu'au lieu d'avoir à modifier toutes les pages unes à unes, nous avons juste à modifier le fichier CSS.

- **Material-Ui:**

La bibliothèque Material-ui a été utilisée afin de pouvoir créer des interfaces utilisateur adaptatives, qui s'ajustent automatiquement à différents types d'appareils et de dispositifs d'entrée. Ainsi que pour pouvoir faire appel à des Composants prédéfinis et réutilisables tels que Grid ,les icons ect ...

- **React-hook-form:**

La bibliothèque open-source est populaire en JavaScript, spécifiquement conçue pour faciliter la gestion des formulaires dans les applications React. Elle se distingue par son utilisation des hooks, introduits dans React 16.8, pour simplifier la gestion de l'état des formulaires.

- **TailwindCSS:**

Tailwind CSS est un framework CSS open-source qui simplifie le processus de création d'interfaces utilisateur (UI) élégantes et réactives pour les applications web. Elle permet d'éviter l'utilisation de CSS pure mais plutôt de pouvoir styliser son site en introduisant seulement des nom de classe spécifique que l'on peut trouver dans la documentation.

LE CV

Les données du CV

les données du CV ont été représenté à travers le fichier ResumeData.js .Ce code est une définition d'un objet JSON qui contient les informations personnelles et professionnelles d'e l'individu. L'objet est utilisé pour remplir un profil en ligne.

Voici un aperçu des différentes propriétés de l'objet:

1. 'title' - Le titre professionnel de la personne.
2. 'birthday' - La date de naissance de la personne.
3. 'email' - L'adresse e-mail de la personne.
4. 'address' - L'adresse postale de la personne.
5. 'phone' - Le numéro de téléphone de la personne.
6. 'socials' - Un objet contenant des liens vers les profils de la personne sur les réseaux sociaux, tels que Facebook, LinkedIn et e-mail.
7. 'about ' - Une brève description de la personne.
8. 'experiences' - Un tableau d'objets, chacun représentant une expérience professionnelle de la personne.
9. 'Formation' - Un tableau d'objets, chacun représentant une formation ou une qualification de la personne.
10. 'skills' - Un tableau d'objets, chacun représentant les compétences technique spécifique de la personne.

L'objet JSON est exporté par défaut, pour être utilisé dans d'autres fichiers JavaScript en utilisant la syntaxe import .

```
import FacebookIcon from '@mui/icons-material/Facebook';
import LinkedInIcon from '@mui/icons-material/LinkedIn';
import MailIcon from '@mui/icons-material/Mail';

export default {
  name: 'Ilyas Assiar',
  title:'Data Scientist',
  birthday: ' 16 th September 2003',
  email:'voiceilyas@gmail.com',
  address:'Bloc J B CYM',
  phone : '0659050692',
```

```
  experiences:[
    {
      title:'work 1',
      date:'2018 - present',
      description:'and enthusiastic computer science engineer, with a pass
```

Le profile

Le profil est divisé en trois sections principales : le nom, le titre et les informations.

La première section du profil, qui contient le nom et le titre, est représentée par la div "profile_name". Les informations sont affichées en utilisant les composants Typography de Material-UI.

La deuxième section du profil, qui contient l'image de profil, est représentée par la figure "profile_image". L'image est chargée à partir d'un fichier local en utilisant la balise img et le module require de Node.js.

La troisième section du profil, qui contient les informations personnelles, est représentée par la div "profile_information". Les informations sont affichées en utilisant le composant CustomTimeline du composant Timeline/Timeline.js.

Dans cette section, le nom, le titre, la date de naissance et les liens sociaux sont affichés. Les informations sur les liens sociaux sont tirées de l'objet ResumeData et affichées en utilisant la méthode map de JavaScript.

ILYAS ASSIAR

Data Scientist



- Name: Ilyas Assiar
- Title: Data Scientist
- Birthday: 16 th September 2003
- facebook: Ilyasassiar
- LinkedIn: @Ilyas_assiar
- Mail: voiceilyas@gmail.com

```
const Pro = () => {
  return (
    <div className="profile_container_shadow">
      <div className="profile_name">
        <Typography className="name">{ResumeData.name}</Typography>           /*afficher les données a trave
        <Typography className="title">{ResumeData.title}</Typography>
      </div>

      <figure className="profile_image">
        <img src={require("../Assets/images/img2.jpg")} alt="photoprofil" /> /*la photo de profile */
      </figure>

      <div className="profile_information">
        <CustomTimeline icon={<PersonOutlineIcon/>}>
          <CustomTimelineItem title='Name' text={ResumeData.name}/>
          <CustomTimelineItem title='Title' text={ResumeData.title}/>
          <CustomTimelineItem title='Birthday' text={ResumeData.birthday}/>

          {Object.keys(ResumeData.socials).map((key)=>{                               /*on parcourt la liste des liens */
            <CustomTimelineItem title={key}text={ResumeData.socials[key].text} link={ResumeData.socials[key].li
          })}
        </CustomTimeline>
      </div>
    </div>
  )
}
```

Resume body

Dans ce code, on crée une composante React pour le corps du curriculum vitale en utilisant les données stockées dans le fichier **ResumeData.js**. Le fichier CSS associé est **Resume.css**.

On commence par afficher la section "About me" en utilisant la grille MUI (Material-UI). La grille permet de disposer les éléments en ligne et de les diviser en différentes colonnes selon la taille de l'écran.

Ensuite, on crée la section "Experiences" et "Formation" en utilisant la composante **CustomTimeline**. Cette composante affiche un chronos horizontalement avec des événements en utilisant les composantes **TimelineItem**, **CustomTimelineSeparator**, **TimelineContent** et **TimelineDot** de Material-UI.

Pour la section "Skills", on crée un grid qui contient plusieurs cartes avec les compétences . Les cartes sont construites en utilisant la composante Paper de Material-UI.

Le fichier ResumeData.js contient toutes les informations à afficher dans le curriculum vitale, telles que les compétences, les expériences professionnelles et les formations.

Projects

Outcome

Enfin, le fichier Resume.css permet de styliser le curriculum vitale en définissant des règles CSS pour les différentes classes utilisées dans le code.

le code source de la partie Timeline

```
const CustomTimeline = ({title , icon, children }) => {
  return (
    <Timeline className={"timeline"}>
      { /* Header item*/ }
      <TimelineItem className={"timeline_firstItem"}>
        <TimelineSeparator>
          <TimelineDot className={"timeline_dot_header"}>
            {icon}
          </TimelineDot>
          <TimelineConnector />
        </TimelineSeparator>
        <TimelineContent>
          <Typography variant='h6' className={"timeline_header"}>
            {title}
          </Typography>
        </TimelineContent>
      </TimelineItem>
      {children} { /*Other items*/ }
    </Timeline>
  )
}
```

Resume body

Exemple de code pour la partie Experiences

```
<Grid item xs={12}>
  <Grid Container className="Resume_timeline">
    { /* Working History*/ }
    <Grid item sm={12} md={6}>
      <CustomTimeline title="Experiences" icon={ <WorkIcon/> }>
        {ResumeData.experiences.map((experience) => (
          <TimelineItem className='timeline_content'>
            <CustomTimelineSeparator />
            <TimelineContent>
              <Typography className='timeline_title'>{experience.title}</Typography>
              <Typography variant='caption' className='timeline_date'>{experience.date}</Typography>
              <Typography variant='body2' className='timeline_description'>{experience.description}</Typography>
            </TimelineContent>
          </TimelineItem>
        ))}
      </CustomTimeline>
    </Grid>
  </Grid>
</Grid>
```

Projects

Partie Experiences su site web



Experiences

Outcome

Stage KPMG

2023 - present

and enthusiastic computer science engineer, with a passion for innovation and technology

Stage deloitte

2021 - 2022

and enthusiastic computer science engineer, with a passion for innovation and technology

Stage Oracle

2019 - 2020

and enthusiastic computer science engineer, with a passion for innovation and technology

Resume body

Exemple de code pour la partie skills

```
<Grid container
spacing={3}
justify="space-between"
className="section graybg pb_45 p_50">
  {ResumeData.skills.map((skill) =>(
    <Grid item xs={12} sm={6} md={4}>
      <Paper elevation={0} className="skill">
        <Typography variant='h6' className="skills_title">
          {skill.title}
          {skill.description.map((element) =>(
            <Typography variant="body2" className='skill_description'>
              <TimelineDot variant={'outlined'} className='timeline_dot' />
              {element}
            </Typography>
          ))}
        </Typography>
      </Paper>
    </Grid>
  ))}
</Grid>
```

Projects

Outcome

Partie Skills su site web

Skills

FRONT-END

- ReactJS
- Javascript
- Typescript
- Bootstrap
- Material Ui

BACK-END

- NodeJS
- Java
- Python
- C

DATABASE

- Oracle
- MySQL

Resume body

About me

I am a dedicated and enthusiastic computer science engineer, with a passion for innovation and technology. I am committed to developing cutting-edge software solutions that optimize efficiency, user experience, and overall system performance. My academic background includes a bachelor degree in computer science and a minor in software engineering, providing me with a solid foundation in computer science theory and practical programming.



Experiences

- Stage KPMG**
2023 - present
and enthusiastic computer science engineer, with a passion for innovation and technology
- Stage deloitte**
2021 - 2022
and enthusiastic computer science engineer, with a passion for innovation and technology
- Stage Oracle**
2019 - 2020
and enthusiastic computer science engineer, with a passion for innovation and technology

Projects



Formation

- EMI**
2023 - present
and enthusiastic computer science engineer, with a passion for innovation and technology
- CPGE**
2021 - 2023
and enthusiastic computer science engineer, with a passion for innovation and technology
- Bacallauréat Science math**
2021
and enthusiastic computer science engineer, with a passion for innovation and technology

Outcome

Skills

FRONT-END

- ReactJS
- Javascript
- Typescript
- Bootstrap
- Material Ui

BACK-END

- NodeJS
- Java
- Python
- C

DATABASE

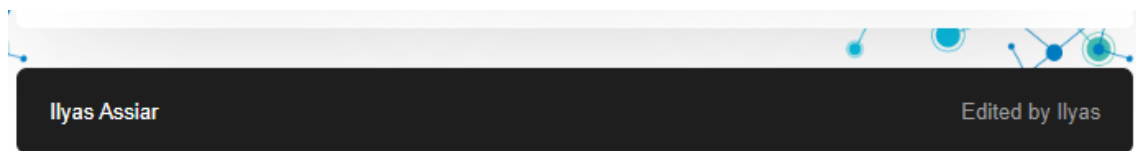
- Oracle
- MySQL

Footer

ce code crée un composant de pied de page qui affiche le nom de la personne et un message indiquant que le profil a été édité par Ilyas.

```
const Footer = () => {  
  return (  
    <div className='footer'>  
      <div className='Footer_left'>  
        <Typography className="footer_name">{ResumeData.name}</Typography>  
      </div>  
      <div className="footer_right">  
        <Typography className='footer_copyright'>  
          Edited by Ilyas  
        </Typography>  
      </div>  
    </div>  
  )  
}
```

le code source de Skills



La partie Footer sur le site WEb

LE FORMULAIRE

Les données du formulaire

Les données du formulaire sont simplement les différents champs de saisis qui sont:

Pour les informations personnelles constituées de:

1. Nom: Qui doit être en majuscule.
2. Prénom: Dont la première lettre seulement doit être majuscule.
3. Age : entre 18 et 60 ans.
4. Situation familiale: Célibataire ou Marié.
5. Email: Doit respecter la structure d'une email.
6. Une photo d'identité
7. Objectifs professionnels

Tout ces champs sont requis

Pour l'espace 'Formation et Competences':

1. Parcours scolaire: Requis
2. Experiences Professionnelles: Requis
3. Langues parlées: Optionel
4. Qualités: Ajout de qualité à la demande, requis.

Nous avons aussi recours a la bibliotheque react-hook-form ainsi que les le Titre et le Footer qui sont dans des composantes à part.

```
1 import { useState } from 'react'
2 import { Title } from "../components/FORM COMP/Title";
3 + import { Footer } from "../components/FORM COMP/Footer";
4 import { useForm, SubmitHandler } from "react-hook-form";
```

Le Titre

Le titre est simplement une divition qui contient le titre du formulaire “Fomulaire pour CV”. Un design tres simpliste rentrant dans la palette de couleurs du cahier de charges. Décomposons un peu le code ci dessous:

- Le tout est contenue dans un header pour signifier le début de la page
- Dans la classe on peut retrouver:
 - w-full: pour signifier que le header prendra toute la longueur
 - bg-gradient-to-r from-cyan-300 to-blue-300: détermine la couleur de fond
 - p-4: Concerne le padding à l’interieur
 - text-center: pour centrer le titre au milieu de la page
- Puis pour le titre on appliquera
 - text-3xl: Pour donner au titre une plus grande taill car tailwind ne donne pas les tailles des balises h1.....h6
 - font-bold: pour mettre le texte en gras

Formulaire pour CV

```
1 export const Title = () => {
2   return (
3     <header className="w-full bg-gradient-to-r from-cyan-300 to-blue-300 p-4 text-center">
4       <div className="container mx-auto">
5         <h1 className="text-3xl font-bold">Formulaire pour CV</h1>
6       </div>
7     </header>
8   );
9 }
```

La forme du formulaire

En ce qui concerne la forme du corps du formulaire, le design est toujours simpliste avec la même palette avec différentes nuances de bleu.

Commençons par les informations personnelles: le code étant volumineux prenons l'exemple du Nom qui va plus ou moins se répéter:

- Nous avons une div qui va englober deux autres div:
 - La première va contenir le label ainsi que l'input
 - La deuxième concerne le potentiel message d'erreur. Nous y reviendrons pour la partie script
- La div va être sous forme de flex-box pour pouvoir aligner le message d'erreur avec l'input.
- Pour le label rien à part le `htmlFor` pour le lier à son input.
- Pour la stylisation de l'input on a:
 - `focus`: Pour donner une bordure quand le champ est sélectionné
 - `px`, `py`: Pour mettre en place un espace à l'intérieur du champ de saisie
 - `w-80`: pour gérer la largeur
 - `mb-3`: pour le décaler du label
 - `rounded-md border-1`: Vont générer une bordure arrondie d'épaisseur 1
- Pour la div contenant le message d'erreur on le décalera à droite (`ml-5`) et le texte sera en rose souligné (`underline text-pink-800`)

```
65 <form className="mx-60" onSubmit={handleSubmit(onSubmit)}>
66   <fieldset className="w-full rounded-xl pl-5 mt-10 mb-5 border-4 bg-gradient-to-r from-cyan-200 to-blue-200">
67     <legend className="text-2xl underline"><h2>Informations personnelles: </h2></legend>
68     <div className="flex items-center">
69       <div>
70         <label htmlFor="lName">*Nom :</label><br />
71         <input {...register("lName", {required: 'Champs obligatoire', validate: (value) => isUppercase(value) || 'Nom
          doit être en majuscule'})} className="focus:outline-none focus:ring focus:ring-slate-600 px-2 py-2 w-80 mb-3
          rounded-md border-1" type="text" name="lName" id="lName" placeholder="Ex: EL BOUR" />
72       </div>
73       <div className="ml-5 underline text-pink-800">
74         {errors.lName && <p>{errors.lName.message}</p>}
75       </div>
76     </div>
77     <div className="flex items-center">
78       <div>
79         <label htmlFor="fName">*Prénom :</label><br />
80         <input {...register("fName", {required: 'Champs obligatoire', validate: (value) => isFirstUppercase(value) ||
          'Début doit être majuscule'})} className="focus:outline-none focus:ring focus:ring-slate-600 px-2 py-2 w-80
          mb-3 rounded-md border-1" type="text" name="fName" id="fName" placeholder="Ex: Othmane" />
81       </div>
```

La forme du formulaire

Petite exception pour le textarea pour lesquelles la bordure à dû être ajoutée dans le CSS sans passer par TailwindCSS.

```
129 <div className="flex items-center">
130 <div>
131   <label htmlFor="objectifs">*Objectifs professionnels :</label><br />
132   <textarea {...register("objectifs", { required: 'Champs obligatoire' })} className="mb-3 focus:outline-none focus:ring focus:ring-slate-600
133     px-2 py-2 resize-none" placeholder="Entrez vos objectifs professionnels..." name="objectifs" id="objectifs" cols={60} rows={5} />
134 </div>
135 <div className="ml-5 underline text-pink-800">
136   {errors.objectifs && <p>{errors.objectifs.message}</p>}
137 </div>
</div>
```

Pour la selection, pour une meilleur gestion des imprévus, nous avons rendu la première option désactivé pour éviter les erreurs involontaires:

```
98 <div className="flex items-center">
99 <div>
100   <label htmlFor="picture">*Situation Familiale :</label><br />
101   <select {...register("situation", { required: 'Champs obligatoire', validate: (value) => isValidSelect(value) || 'Veuillez selectionner une
102     option' })} className="border-2 w-80 py-2" name="situation" id="situation">
103     <option value="..." selected disabled>...</option>
104     <option value="Célibataire">Célibataire</option>
105     <option value="Marié(e)">Marié(e)</option>
106   </select>
107 </div>
108 <div className="ml-5 underline text-pink-800">
109   {errors.situation && <p>{errors.situation.message}</p>}
110 </div>
</div>
```

La forme du formulaire

Pour la partie Formation et compétences nous avons 3 types principaux:

- Les textarea: qui ont les memes spécificités que les textarea de la section précédente
- Les checkbox: qui on juste un espacement pour plus de lisibilité

```
140 + <fieldset className="w-full rounded-xl pl-5 mt-10 mb-5 border-4 bg-gradient-to-r from-cyan-200 to-blue-200">
141   <legend className="text-2xl underline"><h2>Formation et compétences : </h2></legend>
142   <div className="flex items-center">
143     <div>
144       <label htmlFor="ParcoursScolaire">*Parcours scolaire: </label><br />
145       <textarea {...register("ParcoursScolaire", { required: 'Champs obligatoire' })} className="mb-3 focus:outline-none focus:ring
        focus:ring-slate-600 px-2 py-2 resize-none" name="ParcoursScolaire" id="ParcoursScolaire" cols={60} rows={5} placeholder="Decrivez vos
        objectifs professionnels"></textarea>
146     </div>
147     <div className="ml-5 underline text-pink-800">
148       {errors.ParcoursScolaire && <p>{errors.ParcoursScolaire.message}</p>}
149     </div>
150 + </div>
151   <div className="flex items-center">
152     <div>
153       <label htmlFor="Langues">Langues parlées: (<i>Optionnelle</i>)</label><br />
154       <label htmlFor="Anglais">Anglais: </label>
155       <input className="ml-2" name="Eng" type="checkbox" />Oral
156       <input className="ml-2" name="Eng" type="checkbox" />Écrit
157       <br />
158       <label htmlFor="Francais">Français: </label>
159       <input className="ml-2" name="Fr" type="checkbox" />Oral
160       <input className="ml-2" name="Fr" type="checkbox" />Écrit
161       <br />
162       <label htmlFor="Arabe">Arabe: </label>
163       <input className="ml-2" name="Ar" type="checkbox" />Oral
164       <input className="ml-2" name="Ar" type="checkbox" />Écrit
165       <br />
166       <label htmlFor="Espagnol">Espagnol: </label>
167       <input className="ml-2" name="Esp" type="checkbox" />Oral
168       <input className="ml-2" name="Esp" type="checkbox" />Écrit
169     </div>
170   </div>
171 </div>
172 </div>
```

Formation et compétences :

*Parcours scolaire:

Decrivez vos objectifs professionnels

*Expériences professionnelles:

Decrivez votre experience professionnels

Langues parlées: (Optionnelle)

Anglais: ☐ Oral ☐ Écrit

Français: ☐ Oral ☐ Écrit

Arabe: ☐ Oral ☐ Écrit

Espagnol: ☐ Oral ☐ Écrit

*Qualité 1

Entrez votre compétence

+ Ajouter un champ

La forme du formulaire

Pour la dernière partie concernant les Qualités, nous avons recours à un script qui sera développé dans la partie suivante.

Cette div est constituée d'un label et d'un input ainsi que d'un bouton qui permet d'ajouter un input en cliquant dessus.

```
181 <div className="flex items-center">
182   <div>
183     {inputs.map((input, index) => (
184       <div className="flex items-center" key={index}>
185         <div>
186           <label htmlFor={`skill${index + 1}`} >Qualité {index + 1} </label><br />
187           <input {...register("skill", { required: 'Champs obligatoire' })} className="focus:outline-none focus:ring
              focus:ring-slate-600 px-2 py-2 w-80 mb-3 rounded-md border-1" type="text" name="skill" id={`skill${index + 1}`} value=
              {input} placeholder="Entrez votre compétence" onChange={(e) => handleInputChange(index, e.target.value)} />
188         </div>
189         <div className="ml-5 underline text-pink-800">
190           {errors.skill && <p>{errors.skill.message}</p>}
191         </div>
192       </div>
193     ))}
194   </div>
```

```
195   <button className="rounded-full px-4 mt-3 mb-3 bg-gradient-to-r from-cyan-300 to-blue-300
              hover:bg-slate-400 active:bg-slate-700 focus:outline-none focus:ring focus:ring-violet-300" onClick=
              {addInput}>+ Ajouter un champ</button>
196 </div>
197 </div>
```

Et enfin le bouton pour envoyer le formulaire: rien de remarquable à part la fonction onSubmit qu'il exécutera au clique. Fonction que nous verrons dans la partie suivante

ENVOYER

```
200 <div className="text-center w-full">
201   <button type="submit" className="rounded-full w-40 h-18 px-4 mt-3 mb-3 bg-gradient-to-r from-cyan-300 to-blue-300
              hover:bg-violet-400 focus:outline-none focus:ring focus:ring-cyan-700">ENVOYER</button>
202 </div>
203 </form>
```

Le script

La partie la plus imposante de notre code est en faite un script.

Decomposons le de facon à rester le plus bref possible:

1. La creation d'un type contenant toutes les informations collectée

```
6 + v type FormInput = {
7     lName: string,
8     fName: string,
9     email: string,
10    age: number,
11    situation: string,
12    file: File,
13    objectifs: string,
14    ParcoursScolaire: string,
15    ExpPro: string,
16    skill: string
17 }
```

C'est ici que sont regroupées les fonction de gestion des erreurs du formulaire. Dans l'ordre:

- Verifie que les caractères sont en majuscule
- Verifie que le premier est majuscule
- Verifie que le nombre est entre 18 et 60
- Verifie que l'email adhère à la forme réguliaire d'une email
- Verifie que la selection n'est pas restée inchangée

```
33 const { register, handleSubmit, reset, formState: { errors } } = useForm<FormInput>()
34
35 + v const isUppercase = (value: string) => {
36     return value === value.toUpperCase();
37 }
38 v const isFirstUppercase = (value: string) => {
39     return value[0] === value[0].toUpperCase();
40 }
41 v const isDigit = (value: number) => {
42     return value >= 18 && value <= 60;
43 }
44 v const isEmail = (value: string) => {
45     const emailRegex = /^[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,}$/i;
46     return emailRegex.test(value);
47 }
48
49
50 v const isValidSelect = (value: string) => {
51     return value !== '...';
52 }
53
```


Le script

Pour ce qui est de l'application de ces fonctions, comme avant nous prendrons l'exemple du premier input du NOM qui va résumer la procédure.

```
65 <form className="mx-60" onSubmit={handleSubmit(onSubmit)}>
66   <fieldset className="w-full rounded-xl pl-5 mt-10 mb-5 border-4 bg-gradient-to-r from-cyan-200 to-blue-200">
67     <legend className="text-2xl underline"><h2>Informations personnelles: </h2></legend>
68     <div className="flex items-center">
69       <div>
70         <label htmlFor="lName">*Nom :</label><br />
71         <input {...register("lName", {required: 'Champs obligatoire', validate: (value) => isUppercase(value) || 'Nom
72           doit être en majuscule'})} className="focus:outline-none focus:ring focus:ring-slate-600 px-2 py-2 w-80 mb-3
73             rounded-md border-1" type="text" name="lName" id="lName" placeholder="Ex: EL BOUR" />
74       </div>
75       <div className="ml-5 underline text-pink-800">
76         {errors.lName && <p>{errors.lName.message}</p>}
77       </div>
78     </div>
79     <div className="flex items-center">
80       <div>
81         <label htmlFor="fName">*Prénom :</label><br />
82         <input {...register("fName", { required: 'Champs obligatoire', validate: (value) => isFirstUppercase(value) ||
83           'Debut doit être majuscule' })} className="focus:outline-none focus:ring focus:ring-slate-600 px-2 py-2 w-80
84             mb-3 rounded-md border-1" type="text" name="fName" id="fName" placeholder="Ex: Othmane" />
85       </div>
```

Tout ce qui va faire partie de la verification fait partie de ...register(). Dans l'ordre:

- "lName": indique le nom du champs à traiter
- required: qui indique que le champs est obligatoire et s'il n'est pas rempli le message "Champs obligatoire"
- validate: qui va gérer la validation ou non de ce champs. Il contient une fonction dont le paramètre est la valeur d'entrée du champs et qui va si le texte est en majuscule. Si ce n'est pas le cas il sera affiché: "Nom doit être en majuscule"

Voici le rendu des 3 cas de figure:

*Nom :

Champs obligatoire

Cas: champs vide

*Nom :

Nom doit être en majuscule

Cas: champs pas en majuscule

*Nom :

Cas: champs conforme

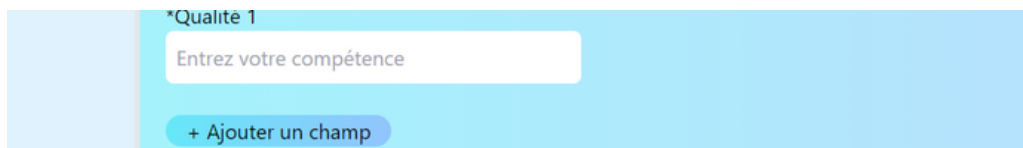
Le script

Passons maintenant au script d'ajout de champs d'insertion:

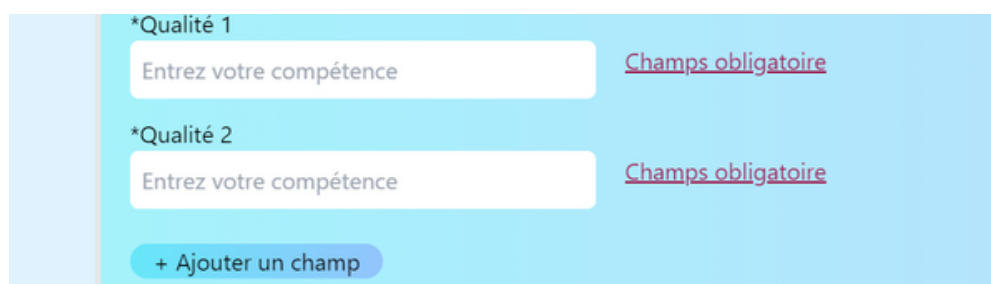
- Tout d'abord nous avons défini les variable dont nous aurons besoin pour la fonction
- Au changement, on executera la fonction `handleInputChange`
- Et le bouton au clique déclanchera la fonction `addInput`

```
20 +
21   const [inputs, setInputs] = useState<string[]>(['']); // Initial
    state with one input
22
23   const addInput = () => {
24     setInputs([...inputs, '']); // Add a new input to the array
25   };
26
27   const handleInputChange = (index: number, value: string) => {
28     const updatedInputs = [...inputs];
29     updatedInputs[index] = value;
30     setInputs(updatedInputs);
31   }
32
```

```
182
183   {inputs.map((input, index) => (
184     <div className="flex items-center" key={index}>
185       <div>
186         <label htmlFor={`skill${index + 1}`} *Qualité {index + 1} </label><br />
187         <input {...register("skill", { required: 'Champs obligatoire' })} className="focus:outline-none focus:ring
          focus:ring-slate-600 px-2 py-2 w-80 mb-3 rounded-md border-1" type="text" name="skill" id={`skill${index + 1}`} value=
          {input} placeholder="Entrez votre compétence" onChange={(e) => handleInputChange(index, e.target.value)} />
188       </div>
189       <div className="ml-5 underline text-pink-800">
190         {errors.skill && <p>{errors.skill.message}</p>}
191       </div>
192     </div>
193   ))}
194   <button className="rounded-full px-4 mt-3 mb-3 bg-gradient-to-r from-cyan-300 to-blue-300 hover:bg-slate-400 active:bg-slate-700
    focus:outline-none focus:ring focus:ring-violet-300" onClick={addInput}>+ Ajouter un champ</button>
196 </div>
```



Avant d'appuyé sur le bouton



Après avoir cliqué

Footer

En bas de la page, le composant Footer va afficher un 'Copyright' ainsi que l'heure actuelle grâce au script qui permet à la fois de récupérer l'heure actuelle mais aussi de la formater pour ensuite l'afficher

```
1 import React, { useEffect, useState } from 'react';
2
3 export const Footer: React.FC = () => {
4   const [currentDate, setCurrentDate] = useState(new Date());
5   useEffect(() => {
6     const intervalId = setInterval(() => {
7       setCurrentDate(new Date());
8     }, 1000); // Mettez à jour la date toutes les secondes
9
10    return () => clearInterval(intervalId); // Nettoyez l'intervalle lors du démontage du composant
11  }, []);
12
13  // Options de formatage pour la date et l'heure
14  const dateTimeFormatOptions: Intl.DateTimeFormatOptions = {
15    year: 'numeric',
16    month: 'numeric',
17    day: 'numeric',
18    hour: 'numeric',
19    minute: 'numeric',
20    second: 'numeric',
21    hour12: false, // Utiliser le format 24 heures
22  };
23
24  const formattedDate = new Intl.DateTimeFormat('fr-FR', dateTimeFormatOptions).format(currentDate);
25
26  return (
27    <footer className='ml-15'>
28      <div className="text-right">
29        <strong>Copyright © 2023</strong><br />
30        <p>{formattedDate}</p>
31      </div>
32    </footer>
33  );
34 }
```

le code source de Skills

Réception des données

Nous avons d'abord pensé à stocker les données dans un fichier texte mais l'opération à échoué, nous nous sommes alors contenté de les afficher dans la console grace a 'console.log()'

```
55     const onSubmit: SubmitHandler<FormInput> = (data) => {  
56         console.log(data);  
57         // Reset le formulaire apres envoie  
58         reset();  
59     }
```

▼ Object  [Formulaire.tsx:56](#)

```
  ExpPro: "MON EXPERIENCE"  
  ParcoursScolaire: "MON PARCOURS"  
  age: "20"  
  email: "elbour.othmane@gmail.com"  
  fName: "Othmane"  
  ▶ file: FileList {length: 0}  
  lName: "EL BOUR"  
  objectifs: "MES OBJECTIFS"  
  situation: "Célibataire"  
  skill: "SKILL 1"  
  ▶ [[Prototype]]: Object
```

AFFICHAGE DE LA CONSOLE DEPUIS LE NAVIGATEUR

Hébergement

Afin d'héberger notre site web ,on a opté pour GitHub, comme étant source gratuite:

1-Creation d'un repertoire.

2-Au niveau de notre code sur Vs ,on a implémenté les commandes suivantes:

```
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/Assiar1/APPCV.git
git push -u origin main
```

on a pu ainsi accéder a notre site web via le lien: <https://assiar1.github.io/CvAPP/>

on peut aussi accéder au code source sur github a travers le lien : <https://github.com/Assiar1/CvAPP>