



DOSSIER-PROJET

Nom de naissance ▶

Nom d'usage ▶ LAFORGUE

Prénom ▶ Arnaud

Adresse ▶ 5-7 rue Sidonie Jacolin 38100 Grenoble

Titre professionnel visé

Concepteur développeur d'application

Sommaire

(à adapter selon le projet)

Remerciements.....	.3
Résumé du projet en anglais.....	.4
Liste des compétences du référentiel.....	.5
Introduction, contexte du projet, résumé exécutif.....	.8
Description de l'application.....	.9
Partie 1 : compréhension du besoin et traduction technique de la réponse apporté.....	10
Partie 2 : enjeux de la mise en œuvre du projet : justification des choix et arbitrages réalisés, problèmes rencontrés et solution apportée.....	.19
Partie 3 : bilan de projet et améliorations envisagées.....	.31
Conclusion : apprentissages, perspectives pour le professionnel envisagé.....	.32

Remerciements

REMERCIEMENTS

Je tiens particulièrement à remercier Thomas Laforge, Anthony Youssef et Kévin Wolff, ainsi que Typhaine Hanry et Leslie Bitene-Verrier

RÉSUMÉ DU PROJET EN ANGLAIS

Good morning, The masterpiece project that I am presenting to you provides additional support for the knowledge acquired during student training. It reassures the user about the prospects of their course. Environmental perspectives are the challenge of mastering science in the face of pollution. The user of the application will be able to take a multiple choice test and obtain a personalized CV by email or by downloading a file. By studying the questions and answers, the user will be able to find out about the blocking points.

LISTE DES COMPÉTENCES DU RÉFÉRENTIEL

A1. Amélioration du processus

de développement

d'applications à l'aide des

principes de l'intégration

continue

C1. Assurer le versionnement d'un

code source d'une application

organisée en fonctionnalités et lots à

l'aide d'un logiciel de contrôle de

version de manière à garantir la

fiabilité du code source dans un

environnement multi-contributeurs

C2. Contrôler l'exécution du code

source à l'aide de tests et d'outils

d'analyses statiques* du code source

afin de minimiser le risque d'erreur

dans un contexte de livraison continue

C3. Automatiser les phases de tests

unitaires et d'analyses statiques du

code source lors du partage des

sources à l'aide d'un outil

d'intégration continue* de manière à

prévenir les erreurs potentielles

A2. Mise en oeuvre des
conditions préalables à la
livraison continue

C4. Concevoir un processus de
livraison continue à l'aide d'outils
d'automatisation de manière à
l'intégrer au processus de
développement

C5. Développer l'architecture d'une
application en micro-services à l'aide
d'outils et de bibliothèques logicielles
adaptées afin de réduire la complexité
globale du système

—

A3. Conception et mise en
oeuvre d'un système de veille
technologique pour aider à la
prise de décision

C6. Concevoir un système de veille
technologique permettant la collecte,
la classification, l'analyse et la
diffusion de l'information aux
différents acteurs de l'organisation
afin d'améliorer la prise de décisions
techniques

A4. Diffusion des méthodes

DevOps auprès des équipes de
l'entreprise/organisation

C7. Accompagner les collaborateurs
au sein de l'équipe projet dans la
sensibilisation et l'acculturation des
méthodes d'organisation et de
production DevOps de manière à
optimiser le cycle de livraison d'un
projet

INTRODUCTION, CONTEXTE DU PROJET, RÉSUMÉ EXÉCUTIF

Après avoir étudié une série de technologies de développement WEB et Mobile et les méthodologies Agile et DevOps, le projet chef d'oeuvre vient compléter l'application des connaissances. Cette application permet à un utilisateur de se rassurer sur ses connaissances scientifiques. Les domaines abordés lors des QCM sont choisis en correspondance avec la couleur verte qui représente la nature et la couleur violette qui représente l'innovation. Une interface web et mobile propose à un utilisateur de se connecter, répondre à des qcm d'informatique et de physique et télécharger un CV personnalisé. Une spécification supplémentaire permettra de se documenter sur les mauvaises réponses. Le parcours d'un utilisateur sur le site se fait à l'aide de boutons et permet de réaliser un qcm de façon connectée ou non avec l'obligation d'être connecté pour télécharger un CV personnalisé. Ainsi ce projet fera beaucoup de bien à ceux qui souhaitent obtenir encore plus de connaissances.

DESCRIPTION DE L'APPLICATION

L'application marche sur smartphone et ordinateur de bureau. Une image docker permettra de faire fonctionner l'application. Une mise en production sur Vercel permet d'accéder au site web via une url. Les utilisateurs intéressés peuvent conseiller l'application à d'autres scientifiques.

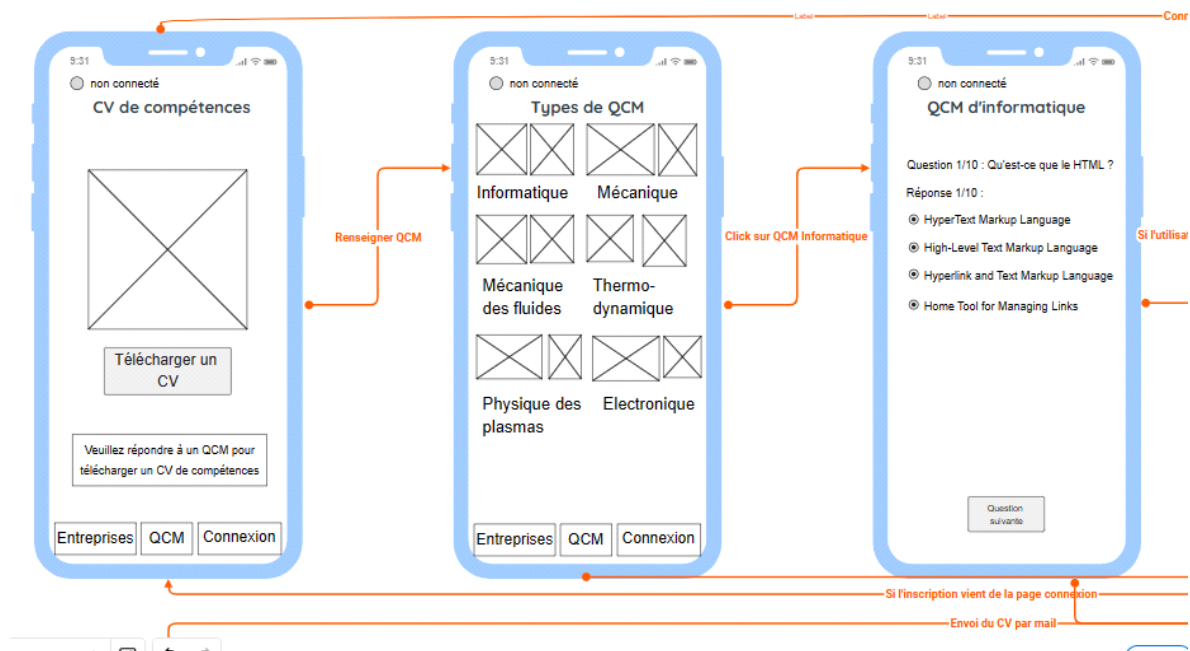
PARTIE 1 : COMPRÉHENSION DU BESOIN ET TRADUCTION TECHNIQUE DE LA RÉPONSE APPORTÉ

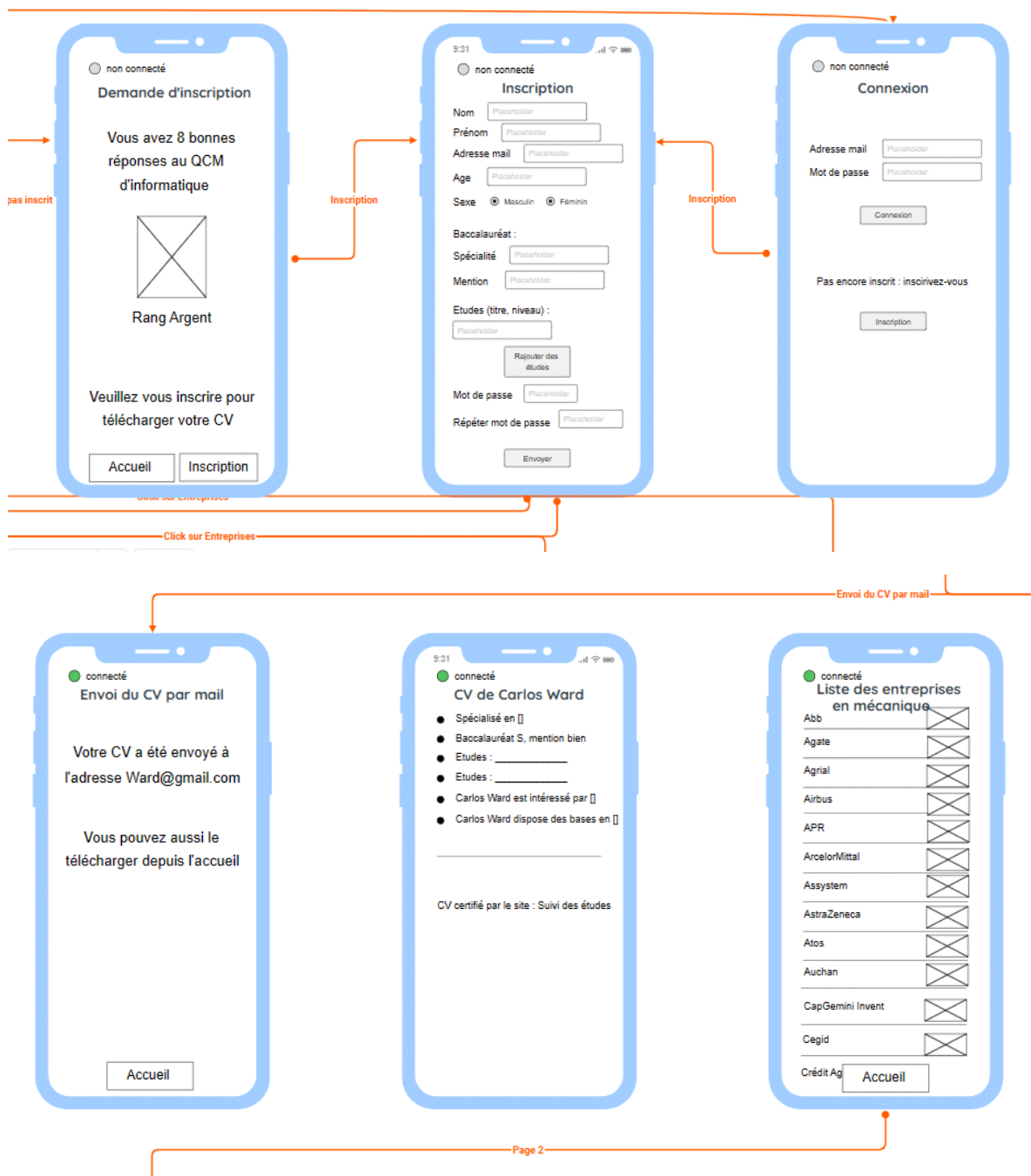
La compréhension du besoin s'est faite après de nombreuses discussions. Dans un premier temps, l'utilisateur répondait à un qcm autant de fois qu'il voulait. Puis le besoin s'est précisé. L'utilisateur peut répondre à un qcm de façon non connecté puis peut éventuellement se connecter à la fin du qcm. S'il est connecté, il dispose de trois essais et peut télécharger son CV personnalisé dans une matière.

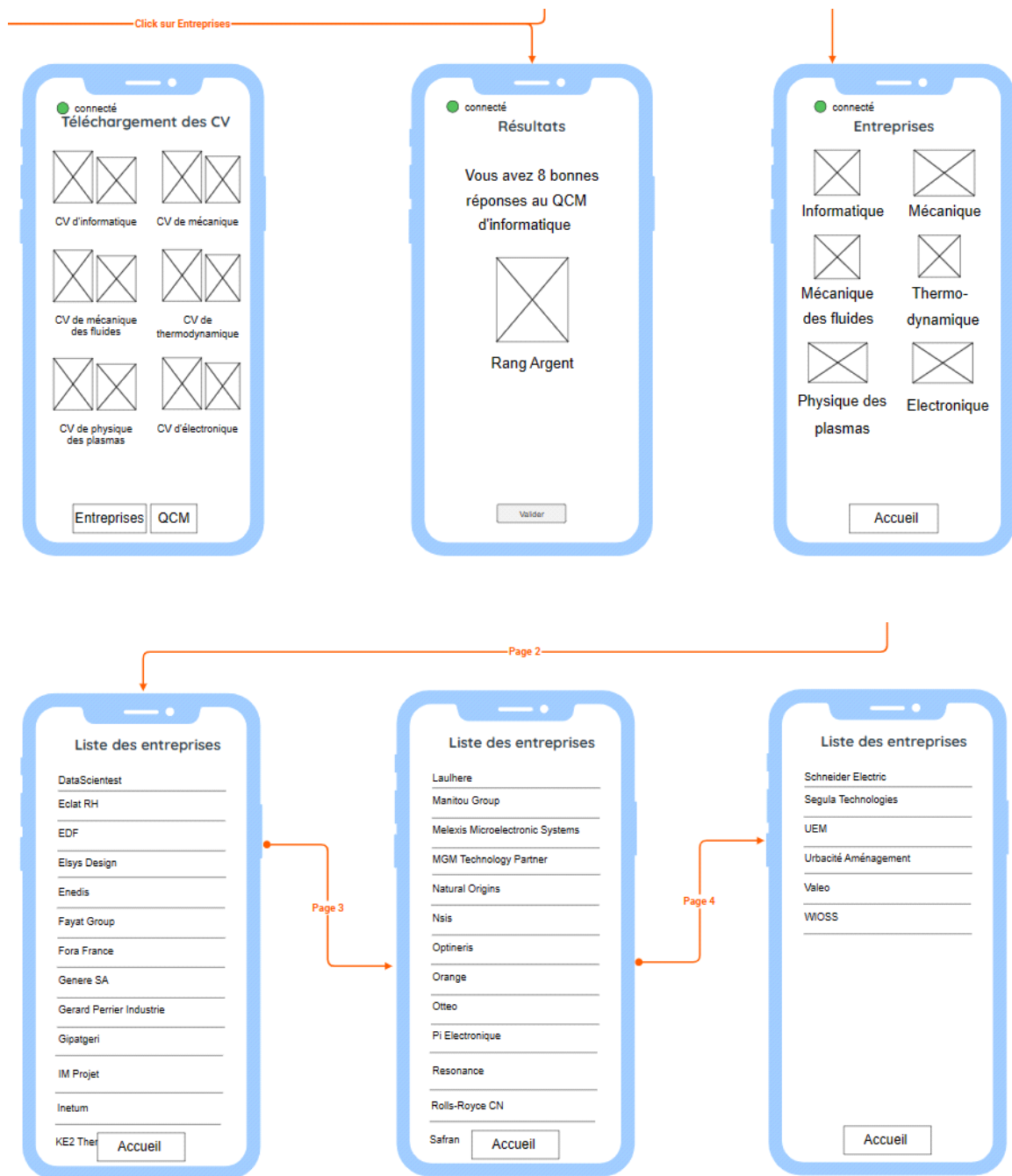
Une fonctionnalité supplémentaire a été ajoutée en cours de projet. Les questions où l'utilisateur a eu des mauvaises réponses peuvent être étudiées pour garantir un apprentissage.

1 Wireframe et Maquette

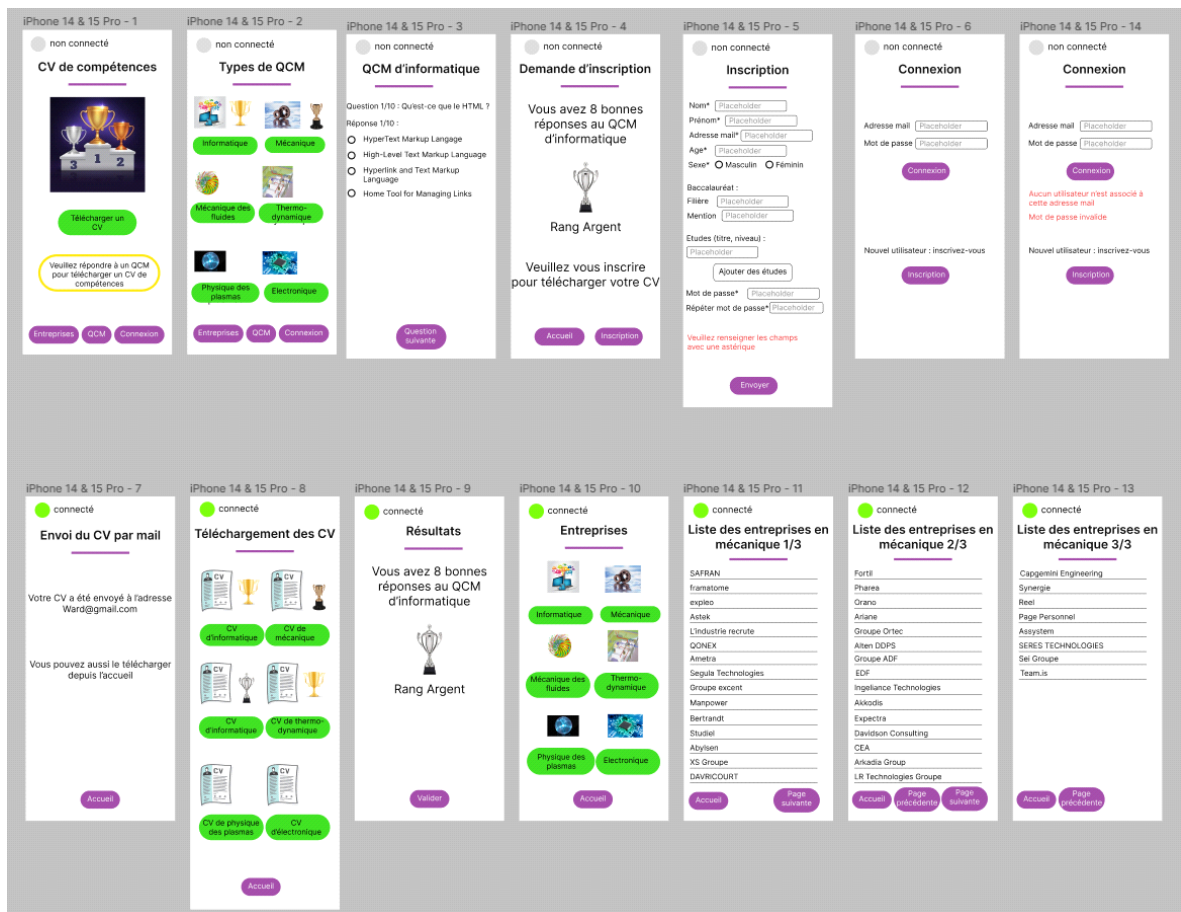
Le wireframe du projet est le suivant :



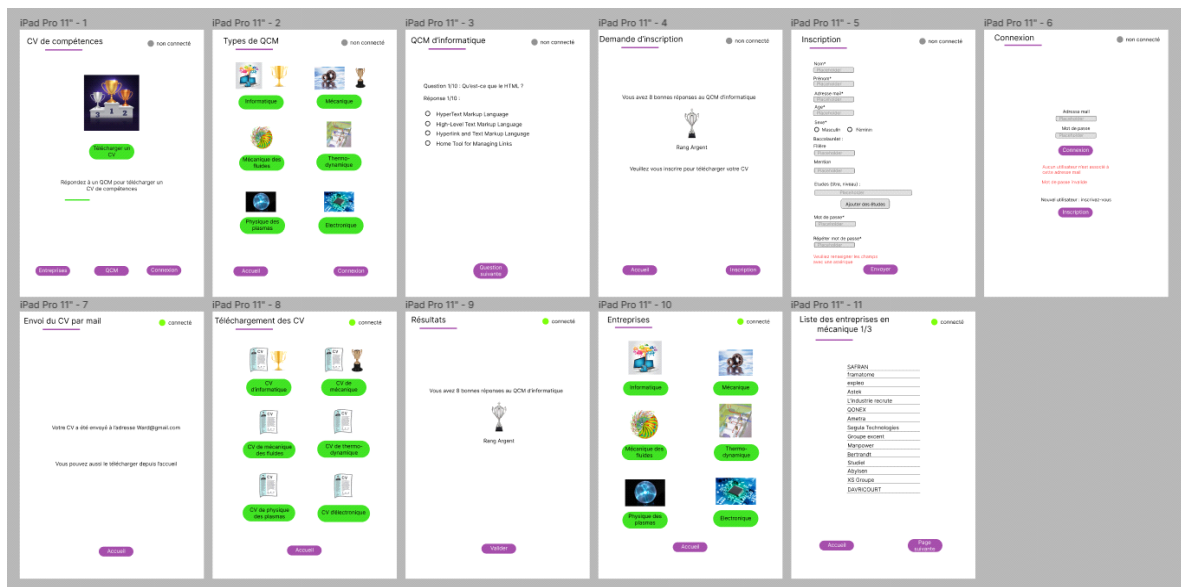




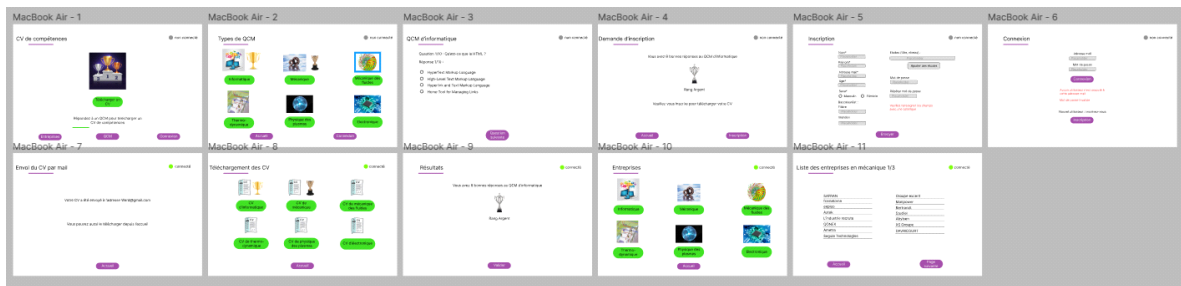
La maquette du projet en affichage mobile est la suivante :



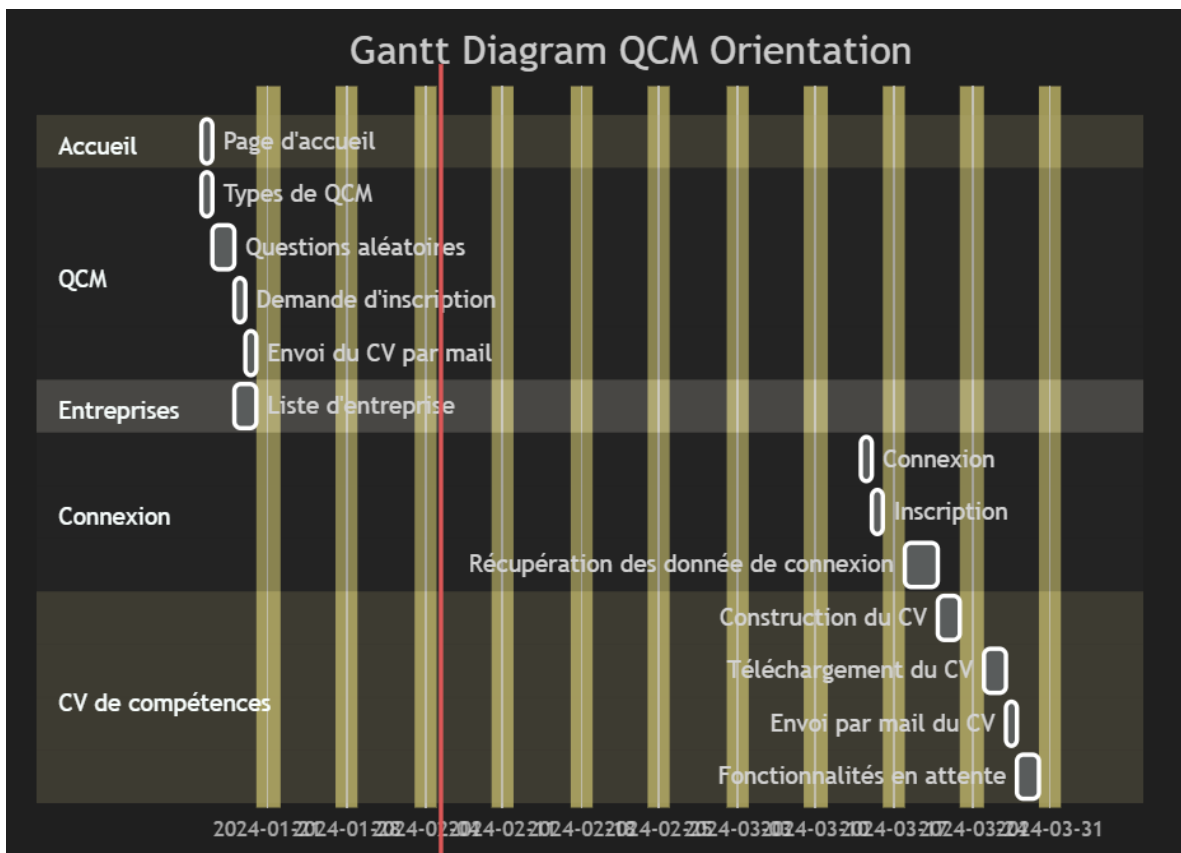
La maquette du projet en affichage tablette est la suivante :



La maquette du projet en affichage desktop est la suivante :



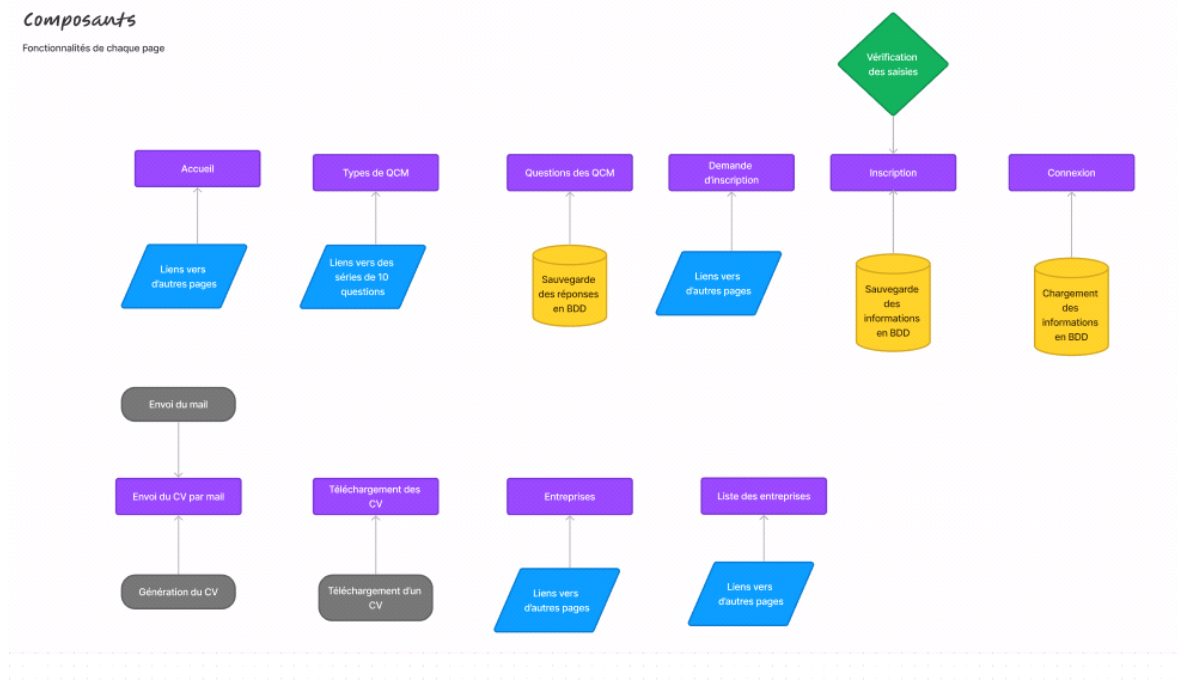
2 Diagramme de Gantt



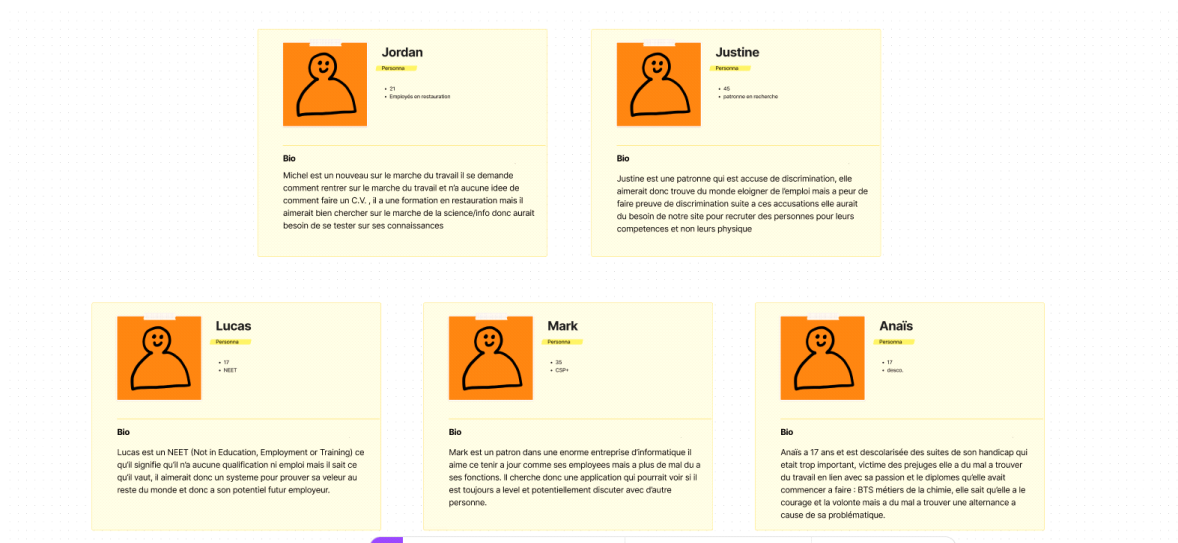
3 Composants

Composants

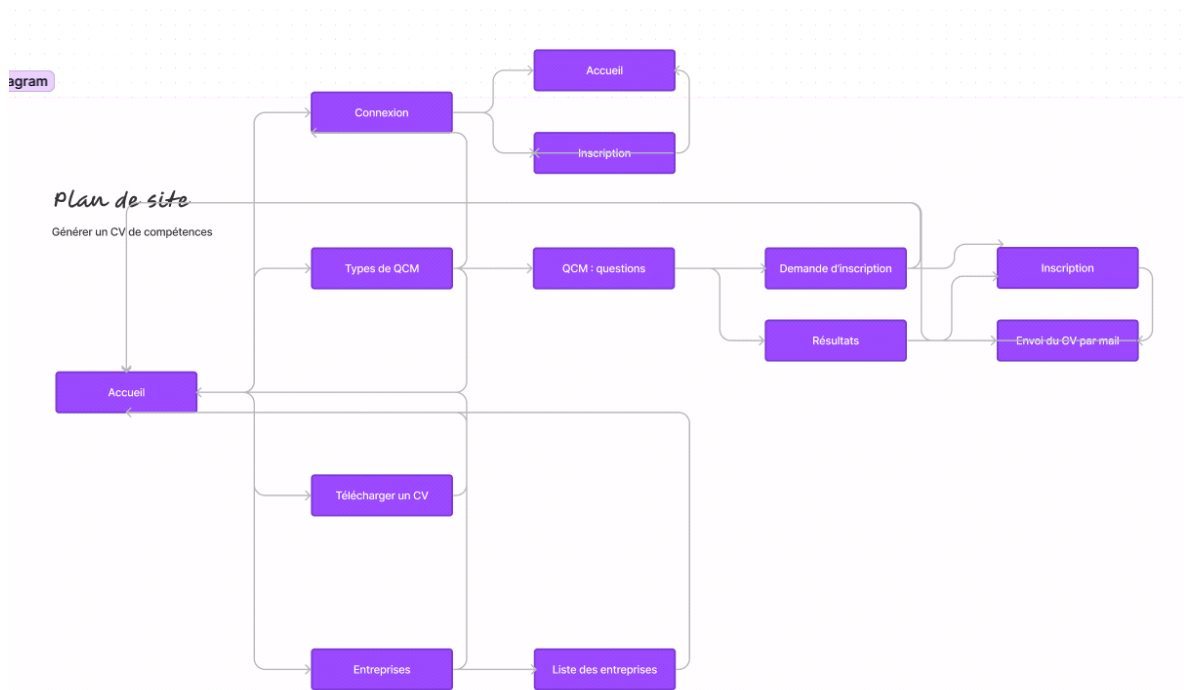
Fonctionnalités de chaque page



4 Personas

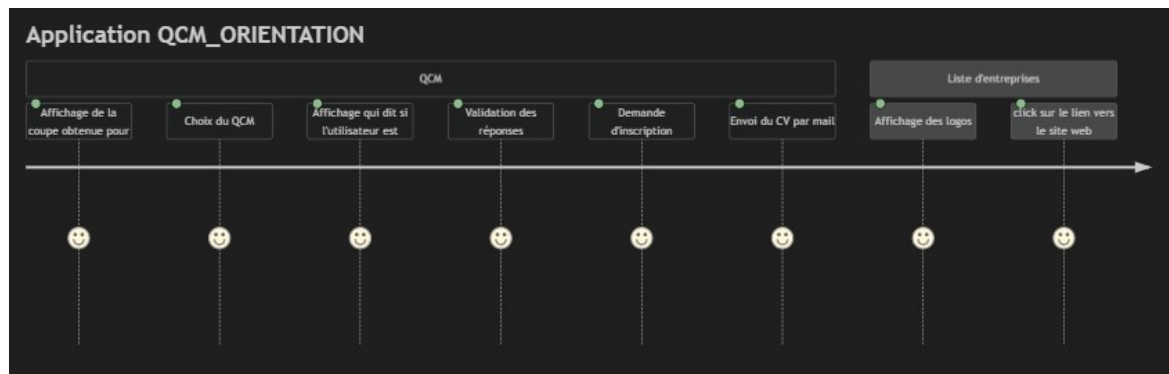
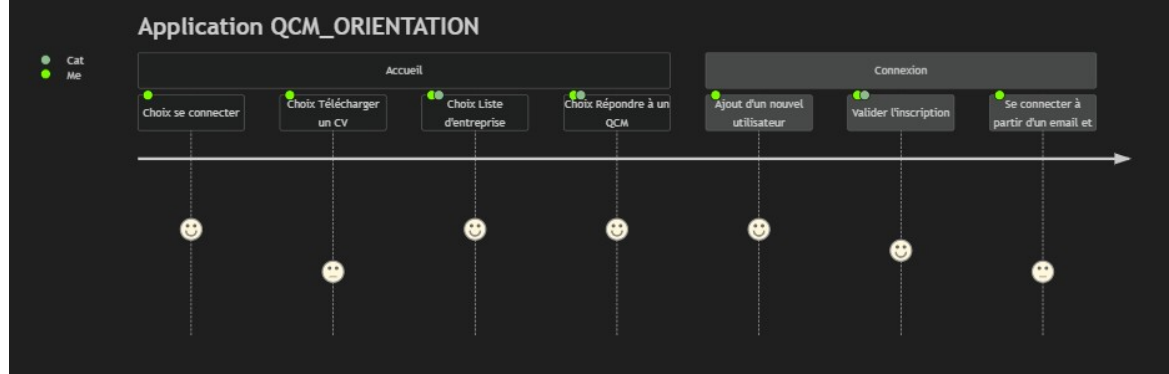


5 Plan de site

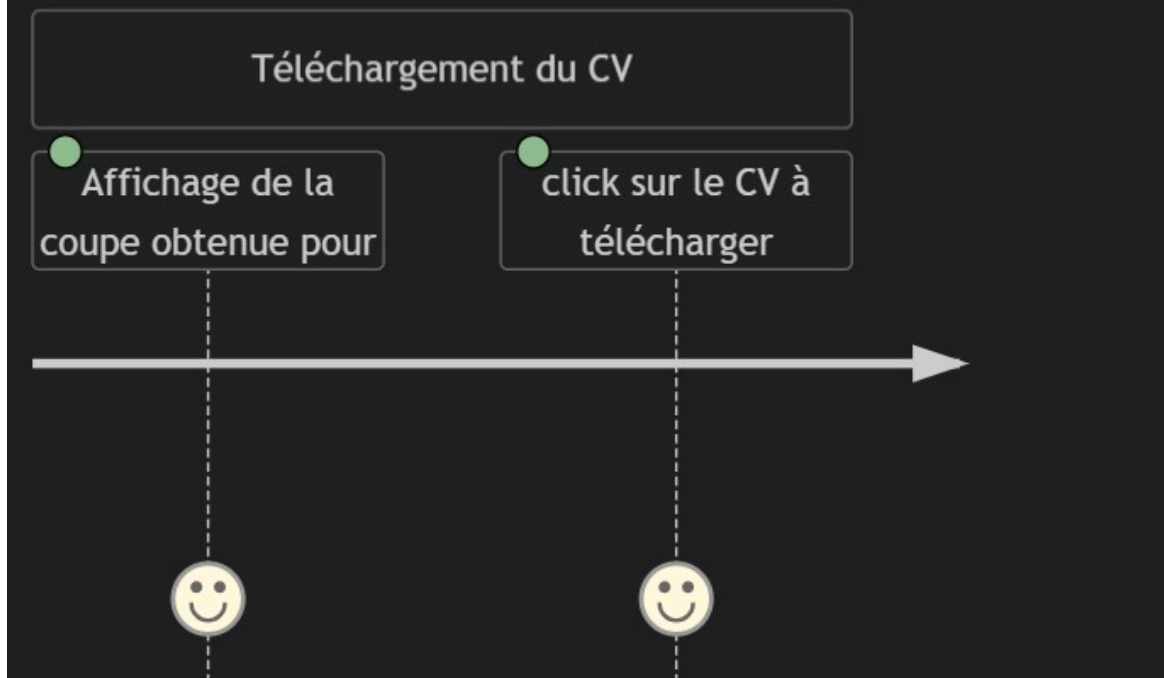


6 User Journey

Projet QCM_Orientation



Application QCM_ORIENTATION



7 Gitflow

7.1 Branches

Les tickets sont utilisés dans Github. Il y a deux branches principales :

_ la branche prod

_ la branche develop

La branche develop dispose d'autres branches qui participent à la mise en place de fonctionnalités.

Les commits sont fréquents

Pour créer une branche : `git branch <nom de la branche>`

Pour se positionner sur une branche : `git checkout <nom de la branche>`

Pour créer une branche et se positionner dessus : `git checkout -b <nom de la branche>`

Pour fusionner la branche develop avec la branche prod en étant positionné sur la branche prod : `git merge develop`

7.2 Pour réaliser un push

git init : pour créer un dépôt git

git status : pour voir l'état du répertoire

git add . : pour ajouter les modifications

git push : pour pousser les modifications vers github

d'autres méthodes comme git push --set-upstream permet de spécifier l'origine

7.3 Pull Request (PR) :

Lors du travail de groupe github permet de notifier des pull request.

Une fois les modifications validées par les associés du projet, les méthodes git sont :

git checkout time-accueil

git add .

git commit -m « nom_du_commit »

git checkout main

git pull

git checkout time-accueil

git rebase main

git push origin time-accueil

(force with lease)

7.4 Récupérer un dépôt distant

git clone <url du depot>

Sur github, il est aussi possible de forker un projet

PARTIE 2 : ENJEUX DE LA MISE EN ŒUVRE DU PROJET : JUSTIFICATION DES CHOIX ET ARBITRAGES RÉALISÉS, PROBLÈMES RENCONTRÉS ET SOLUTION APPORTÉE

Les wireframes et les maquettes présentés dans la partie 1 ne prennent pas en compte la vérification des mauvaises réponses. Lors de la mise en oeuvre du projet un arbitrage a été réalisé. L'utilisateur pourra se renseigner sur ses mauvaises réponses et continuer d'apprendre. Si une entreprise recrute dans un secteur, un utilisateur pourra présenter son CV personnalisé validé par l'application. L'effet papillon permettra à un utilisateur de conseiller l'application à d'autres utilisateur.

1 Frontend Création des routes

Dans la partie Frontend, la navigation se fait à l'aide de routes :

```
const router = createBrowserRouter([
  {
    path: "/",
    element: <Home />,
  },
  {
    path: "/connexion",
    element: <Connexion />,
  },
])
```

Pour cela il faut utiliser RouterProvider :

```
return (  
  <>  
  <div>  
    <RouterProvider router={router} />  
  </div>  
</>  
)
```

2 Backend Utilisation des routes

La meilleure utilisation des routes se fait à l'aide d'un API Router :

```
const apiRouter = express.Router();  
apiRouter.use('/qcm/informatique', qcmInfoRouter);  
apiRouter.use('/auth', authRouter);  
apiRouter.use('/users', userRouter);  
apiRouter.use('/results', resultRouter);  
  
app.use("/api", apiRouter);
```

Dans le fichier auth.ts rangé dans le dossier src/Router/auth.ts, la méthode post permet d'enregistrer un utilisateur :

```

export const authRouter = Router();

authRouter.post("/local/register", async(req, res) => {
  //const id = req.body.id;
  const nom=req.body.nom;
  const prenom=req.body.prenom;
  const email = req.body.email;
  const ismasculin = req.body.ismasculin;
  const filiere = req.body.filiere;
  const mention = req.body.mention;
  const etudes = req.body.etudes;

  const userEmail = await User.findOne({ where: {email:email}});
  console.log("userEmail : ",userEmail);
  if (userEmail===null) {
    const password = req.body.password;

    const saltRounds = 10;
    const hash = await bcrypt.hash(password, saltRounds);
    const monUser = { nom,prenom,email,ismasculin,filiere,mention,etudes, password:hash };

    const newUser = await User.create(monUser);
    const newUserData = newUser.dataValues
    delete newUserData.password
    //res.status(200).json(newUserData);


    const tokenJWT = jwt.sign({ data: 'foobar'}, 'secret', { expiresIn: '1h' });
    | res.status(200).send(tokenJWT);
    //res.status(200).send(hash);
  }
  else {
    | res.status(400).send("l'email que vous avez saisi est déjà utilisé");
  }
  console.log("userEmail : ",userEmail);
})

```

3 Affichage des résultats

La figure suivante présente la page d'accueil en affichage mobile. Un bouton permet de télécharger un CV personnalisé. Les boutons du bas permettent de :

- _ se renseigner sur une liste d'entreprise
- _ choisir un type de qcm
- _ se connecter à l'application
- _ se déconnecter

 Connecté

CV de compétences



Télécharger
un CV

Répondez à un QCM pour
télécharger un CV de
compétences


Entreprises

QCM

Connexion

Déconnexion

La page types de QCM permet de choisir un QCM parmi 6 matières proposées.

 Connecté

Types de QCM



Informatique



Mécanique



Mécanique
des fluides



Thermo-
dynamique



Physique
des plasmas




Electronique

Accueil

Connexion

Si l'utilisateur choisit le QCM d'informatique, 10 questions aléatoires parmi 30 sont posées et il faut choisir une réponse parmi 4.

 Connecté

QCM d'informatique

Question 1 :

Quelle est l'utilité d'une boucle dans la programmation ?

- ☐ Répéter une série d'instructions
- ☐ Créer une interface utilisateur
- ☐ Choisir entre deux options
- ☐ Interrompre l'exécution du programme

Question
suivante

Si une fille a envie de se perfectionner en informatique, elle répond à ce qcm et se documente sur les questions auxquelles elle a mal répondu.

4 Méthodologie CI/CD

4.1 Docker pour le frontend

Un fichier Dockerfile est présent à la racine du projet. L'exemple suivant a été utilisé dans un premier temps mais produisait une erreur 400

```
FROM node:18-alpine
WORKDIR /app
COPY package.json .
COPY package-lock.json .
RUN npm install
COPY . .
EXPOSE 1337
CMD [ "npm", "run", "dev" ]
```

La résolution du problème a été d'installer serve pour le projet front.

```
FROM node:18-alpine
WORKDIR /app
COPY package.json .
COPY package-lock.json .
RUN npm install
RUN npm install serve
COPY . .
RUN npm run build
EXPOSE 5173
CMD npx serve dist -p 5173
```

4.2 Dockerisation du backend

4.2.1 Création d'image et lancement de conteneur

Pour construire une image dont le nom est backend on utilise la commande suivante :

docker build -t backend .

```
arnaud@DESKTOP-BB67L0V:~/dev/Projet-back/projet-back$ docker build -t backend .
[+] Building 2.5s (12/12) FINISHED
    docker:default
=> [internal] load build definition from Dockerfile
    0.3s
=> => transferring dockerfile: 188B
    0.0s
=> [internal] load .dockerignore
    0.2s
=> => transferring context: 73B
    0.0s
=> [internal] load metadata for docker.io/library/node:18-alpine
    0.9s
=> [1/7] FROM docker.io/library/node:18-alpine@sha256:0085670310d2879621f96a4216c893f92e2ded827e9e6ef8437672e1bd72f437
    0.0s
=> [internal] load build context
    0.1s
=> => transferring context: 10.37kB
    0.0s
=> CACHED [2/7] WORKDIR /app
    0.0s
=> CACHED [3/7] COPY package.json .
    0.0s
=> CACHED [4/7] COPY package-lock.json .
    0.0s
=> CACHED [5/7] RUN npm install
    0.0s
=> CACHED [6/7] COPY . .
    0.0s
=> CACHED [7/7] RUN npx tsc
    0.0s
=> exporting to image
    0.1s
=> => exporting layers
    0.0s
=> => writing image sha256:11afedfe2c130ae8066573d836f55fbf2db5245aba52fb0ccc8ce937f7590d5b
    0.1s
=> => naming to docker.io/library/backend
    0.1s
```

Pour lancer un conteneur on utilise la commande :

docker run -dp 3000:3000 backend

4.2.2 Dockerdesktop et Docker hub

Il faut :

- _ s'identifier sur dockerdesktop*
- _ réaliser un build de l'application pour Dockerhub :*
docker build -t arnaudmd/backend
- _ réaliser un push de l'application :*
docker push arnaudmd/backend


```

arnaud@DESKTOP-BB67L0V:~/dev/Projet-back/projet-back$ docker push arnaudmd/backend
Using default tag: latest
The push refers to repository [docker.io/arnaudmd/backend]
5adca014cee4: Pushed
4e947d35aff8: Pushed
5ba45cf590df: Pushed
785fe5e68914: Pushed
5044e61c1f62: Pushed
c77b7e7979de: Pushed
5325b33b9813: Mounted from library/node
4a0d315ad53e: Mounted from library/node
29e213bad130: Mounted from library/node
44fc045c9e3a: Mounted from library/node
latest: digest: sha256:d76524ebbab11c9a4d94bc9dd794d14f74c78d9106ae68ad67374a83f3da12b0 size: 2410

```

_Pour tager autre que latest :
docker build -t arnaudmd/backend:mardi
Puis
docker push arnaudmd/backend:mardi

4.2.3 Render


Dashboard
Blueprints
Env Groups
Docs
Community
Help
New +
Arnaud Laforgue

WEB SERVICE

backend-chef-oeuvre

Image Free Upgrade your instance →

Connect

Manual Deploy

arnaudmd / backend latest

https://backend-chef-oeuvre.onrender.com

Events

Your free instance will spin down with inactivity, which can delay requests by 50 seconds or more. [Upgrade now.](#)

Logs

February 13, 2024 at 2:56 PM

Live

Deploy for d76524e latest

Disks

Environment

Shell

All logs

Search

Live tail

GMT+1

↑

⌵

Feb 13 02:56:28 PM

⇒ Starting service...

Feb 13 02:56:42 PM

serveur running on port : 10000

Feb 13 02:56:42 PM

Executing (default): SELECT name FROM sqlite_master WHERE type='table' AND name='Users';

Feb 13 02:56:42 PM

Executing (default): PRAGMA INDEX_LIST('Users')

Feb 13 02:56:42 PM

Executing (default): SELECT name FROM sqlite_master WHERE type='table' AND name='QCM_contents';

Feb 13 02:56:42 PM

Executing (default): PRAGMA INDEX_LIST('QCM_contents')

Feb 13 02:56:42 PM

Executing (default): SELECT name FROM sqlite_master WHERE type='table' AND name='Results';

Feb 13 02:56:42 PM

Executing (default): PRAGMA INDEX_LIST('Results')

Feb 13 02:56:42 PM

Executing (default): SELECT name FROM sqlite_master WHERE type='table' AND name='token-black-lists';

Feb 13 02:56:42 PM

Executing (default): PRAGMA INDEX_LIST('token-black-lists')

Feb 13 02:56:49 PM

Your service is live 🎉

4.3 Tests d'intégration end2end

Les tests unitaires seront écrits à la fin du projet. Les tests end2end sont réalisés avec Playwright.

```
test('test_connexion_inscr_elts', async ({ page }) => {
  await page.goto('http://localhost:5173/');
  await page.getByRole('button', { name: 'Connexion', exact: true }).click();
  await expect(page.getByRole('heading', { name: 'Connexion' })).toHaveText('Connexion');
  await expect(page.getByRole('button', { name: 'Connexion' })).toBeVisible();
  await page.getByRole('button', { name: 'Inscription' }).click();
  await expect(page.locator('div').filter({ hasText: /^Masculin$/ })).getByRole('radio').toBeVisible();
  await expect(page.locator('div').filter({ hasText: /^Féminin$/ })).getByRole('radio').toBeVisible();
  await page.locator('div').filter({ hasText: /^Féminin$/ }).getByRole('radio').click();
  await expect(page.locator('div').filter({ hasText: /^Féminin$/ })).getByRole('radio').toBeChecked();
  await expect(page.getByText('Mot de passe*', { exact: true })).toBeEditable();
  await expect(page.getByText('Répéter mot de passe*')).toBeEditable();
});

test('test_light_types_qcm', async ({ page }) => {
  await page.goto('http://localhost:5173/');
  await page.getByRole('button', { name: 'QCM' }).click();
  const color = "rgb(236, 59, 59)"
  await expect(page.locator('.light_off')).toHaveCSS('background-color', color);
});
```

5 Fonctionnement des boutons

Dans le composant Connexion.tsx et dans le code html du return, le bouton pour se connecter est géré par la fonction handleClick. Le code est le suivant :

```
<button onClick={handleClick} className="button_purple connect_button">Connexion</button>
{incorrect?
<div></div>:
<div className="errormsg">
  <p>Adresse email ou mot de passe invalide</p>
</div>}
```

En haut de ce composant Connexion.tsx, la définition de la fonction handleClick fait appel à une promesse qui utilise la route <http://localhost:8887/api/auth/local> avec la méthode POST. Dans la constante data on stocke la réponse puis dans le LocalStorage on stocke le token donné par la réponse. Puis on se dirige vers la page home une fois connecté.

```

const handleClick = useCallback(async() => {
  // Lancez une requête POST vers l'API avec les données de connexion
  const response = await fetch('http://localhost:8887/api/auth/local', {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json'
    },
    body: JSON.stringify({
      email: email,
      password: mdp
    })
  })
  const data = await response.text()
  console.log("rtte", data)
  if (data!=="le mot de passe n'est pas le bon" && data!="l'email saisi n'est pas le bon") {
    localStorage.setItem('token', data)
    localStorage.setItem('isconnected',"true")
    navigate("/home");
  }

  else {
    setIsCorrect(false);
    console.log("incorrect false");
    navigate("/connexion")
  }

  // Si la connexion est réussie, stockez le token dans le localStorage
  // Et redirigez l'utilisateur vers la page d'accueil

  // Si la connexion est échouée, affichez un message d'erreur

}, [email, mdp, navigate])

```

Dans la constante email, on récupère le corps email de la requête. Dans la constante userEmail on récupère l'objet JSON d'un utilisateur où l'email correspond à la constante email. Puis dans la constante match on compare le password récupéré dans le corps de la requête au password des valeurs de userEmail. Si le password match on signe le token et on l'envoie en réponse avec un statut 200. Le code de la méthode post est le suivant.

```

authRouter.post("/local", async(req, res) => {
  const email = req.body.email;
  const password = req.body.password;
  const userEmail = await User.findOne({ where: {email:email}});
  if (userEmail!==null) {
    const userEmailData = userEmail.dataValues;
    const match = await bcrypt.compare(password, userEmailData.password);
    if(!match) {
      res.status(400).send("le mot de passe n'est pas le bon");
    }
    else {
      const tokenJWT = jwt.sign(userEmailData, process.env.JWT_SECRET!, { expiresIn: '1h' });
      res.status(200).send(tokenJWT);
    }
  }
  else {
    res.status(400).send("l'email saisi n'est pas le bon");
  }
})

```

6 Utilisation des useState

L'utilisation des états permet de stocker un état à partir de la fonction setNom et d'avoir sa valeur dans nom. La fonction handleChangeNom récupère à chaque saisie une chaîne de caractère et la stocke dans l'état nom. L'input est de type text et appelle la fonction à chaque saisie.

```
const [nom,setNom]=useState("");
```

```

const handleChangeNom = useCallback((e: React.ChangeEvent<HTMLInputElement>) => {
  console.log('new value', e.target.value)
  setNom(e.target.value);
}, []);

```

```

<div className="placeholder">
  <p>Nom* </p>
  <input onChange={handleChangeNom} type="text" className="ph"></input>
</div>

```

PARTIE 3 : BILAN DE PROJET ET AMÉLIORATIONS ENVISAGÉES

Ce projet a été réalisé avec un frontend utilisant Vite et un backend utilisant Typescript et React.

Les couleurs utilisées pour les boutons sont la couleur verte qui représentent la nature et la couleur violette qui représente l'innovation. La couleur verte est indispensable pour contrer la fin du monde et la couleur violette représente la créativité. Les innovations ne seront possibles qu'après l'appropriation d'un savoir. Ainsi l'utilisateur pourra se documenter sur les mauvaises réponses et se renseigner sur une liste d'entreprise. Le silence dans le choix des entreprises pourra compléter une formation et enrichir son savoir d'autres connaissances par une recherche personnelle. Une suite pourra être donné à ce projet en écoutant et espérant les attentes des utilisateurs.

La multiplication des espérances personnelles conduit à une utilisation de la société qui mène au désastre. La complétude organisée par cette application propose de renouer avec le socle du savoir scientifique et au final d'avoir une meilleure espérance de vie.

Les compétences validées sont C1 avec le versionnement du code source et l'utilisation d'une partie frontend et backend. ESLINT est le linter utilisé pour valider la compétence C2. Les phases de tests unitaires sont réalisées avec vitest et valident la compétence C3. La dockerisation avec un fichier Dockerfile et les « run » et « build » de l'application permettent de concevoir un processus de livraison continue à l'aide d'outils d'automatisation et valident la compétence C4. Les outils et bibliothèques adaptées afin de réduire la complexité globale du système sont « react » et « react-router-dom » pour la partie front. Pour la partie backend, il y a « express », « dotenv », « body-parser », « sequelize », « cors », « jsonwebtoken », les modèles, et les routers. Ces outils et bibliothèques valident la compétence C5. La veille technologique sur les frameworks, les méthodes et les outils se fait à partir des mots clés et un résumé en markdown. Elle valide la compétence C6. Pour accompagner les collaborateurs au sein de l'équipe projet dans la sensibilisation et l'acculturation des méthodes d'organisation et de production DevOps il faut un gitflow avec des pull-requests déclenchés par GithubAction. Il faut aussi établir de la communication sur chaque document de conception.

Les améliorations envisagées sont :

- _ l'affichage des logos des entreprises*
- _ la réécriture des qcm ou l'écriture de nouveaux qcm*
- _ la fonctionnalité de faire payer les entreprises pour apparaître sur la liste de l'application*

CONCLUSION : APPRENTISSAGES, PERSPECTIVES POUR LE PROFESSIONNEL ENVISAGÉ

Les apprentissages réalisés pour ce projet sont :

- 1 L'utilisation de la méthode devops avec l'intégration continue et le déploiement continu*
- 2 L'utilisation de la méthode Agile avec les documents de conception, les sprints et les daylies scrums*
- 3 L'utilisation de crochets en React, des useState, des fonctions « handle », la gestion de l'inscription et de la connexion*
- 4 L'utilisation d'un Modèle Conceptuel des Données*
- 5 La création de quizz avec des réponses à choix multiples.*

Les perspectives pour le professionnel envisagé :

- 1 Construire une application avec une inscription et une connexion*
- 2 Construire un application Web et Mobile en utilisant un affichage responsive*
- 3 Construire une application React Native sur smartphone avec géolocalisation*