

Dossier projet:

PC Compare



**Table des matières :**

**"**

**"**

Table des matières

**Tapez le titre du chapitre (niveau 1)1**

Tapez le titre du chapitre (niveau 2)2

Tapez le titre du chapitre (niveau 3)3

**Tapez le titre du chapitre (niveau 1)4**

Tapez le titre du chapitre (niveau 2)5

Tapez le titre du chapitre (niveau 3)6

**Remerciements :**

**"**

**"**

**Aujourd'hui,** je prends le temps de vous exprimer ma gratitude et ma reconnaissance pour le soutien inestimable que vous m'avez apporté tout au long de ma formation en tant que Concepteur Développeur d'Applications. Cette expérience a été transformative pour moi, et je ne serais pas arrivé jusqu'ici sans votre précieuse contribution.

**Tout d'abord,** je tiens à remercier mon formateur **Laurent BÉDU**. Vous avez été les piliers de ma réussite. Vos connaissances approfondies, votre passion pour le domaine du développement d'applications et votre dévouement à notre apprentissage ont été une source d'inspiration constante. Grâce à vous, j'ai acquis des compétences techniques solides, j'ai appris à résoudre des problèmes complexes et j'ai compris les principes fondamentaux de la conception logicielle. Votre patience, votre disponibilité et votre capacité à expliquer des concepts difficiles ont fait toute la différence dans ma compréhension et ma progression. Je suis profondément reconnaissant pour l'impact que vous avez eu sur ma carrière émergente.

**À mes proches,** je suis reconnaissant pour votre soutien inconditionnel tout au long de ce parcours. Votre confiance en moi et vos encouragements constants ont été essentiels pour surmonter les moments de doute et de difficulté. Votre présence à mes côtés m'a donné la force nécessaire pour persévérer et atteindre mes objectifs. Vos mots d'encouragement et votre croyance en mon potentiel m'ont inspiré à donner le meilleur de moi-même chaque jour. Je suis fier de vous avoir à mes côtés et je suis reconnaissant pour l'amour et le soutien indéfectibles que vous m'avez apporté.

**Enfin,** je souhaite adresser mes remerciements à mes collègues. Nous avons partagé des moments intenses et stimulants tout au long de cette formation. Vos compétences, votre engagement et votre esprit d'équipe ont créé une atmosphère propice à l'apprentissage et à la collaboration. Les défis auxquels nous avons été confrontés ensemble nous ont poussés à repousser nos limites et à nous améliorer continuellement. Votre soutien mutuel et nos échanges fructueux resteront gravés dans ma mémoire. J'espère que notre amitié et notre collaboration se poursuivront au-delà de cette formation.

**En conclusion,** je tiens à exprimer ma gratitude envers chacun d'entre vous. Votre rôle dans ma réussite est inestimable, et je ne pourrai jamais assez-vous remercier pour tout ce que vous avez fait. Je suis enthousiasmé par les perspectives qui s'offrent à moi en tant que Concepteur Développeur d'Applications, et je suis convaincu que cette formation me servira de base solide pour une carrière épanouissante. Votre influence restera à jamais présente dans ma vie professionnelle.

**Avec toute ma gratitude,**

**Qui suis-je ?**

**"**

**"**

Je m'appelle **Maxence ALLART** et je suis ravi de me présenter devant vous aujourd'hui. Depuis mon plus jeune âge, je suis passionné par l'informatique et la programmation. Ma fascination pour ce domaine a commencé au collège, lorsque j'ai découvert les calculatrices **CASIO** et le langage **BASIC**. J’ai passé un nombre incalculable d’heures à programmer mes propres petits jeux ! C'est à ce moment-là que j'ai su que je voulais poursuivre une carrière dans le **développement** de logiciels.

Après avoir obtenu mon **baccalauréat**, j'ai décidé de suivre un parcours en sciences et technologies de l'industrie (STI) qui comportait une composante importante de **programmation**. C'est là que j'ai vraiment eu l'opportunité de découvrir la **programmation** avec comme premier langage le **C** (oui c’était il y a plus de 15ans !). Mais également de plonger dans le monde du développement web, en apprenant des langages tels que **HTML**, **CSS**, **JavaScript**, **PHP** et **SQL**.

Malheureusement, en raison de problèmes familiaux, j'ai dû interrompre mes études en deuxième année de BTS. Cette période a été difficile pour moi, et j'ai dû trouver des emplois "alimentaires" pour subvenir à mes besoins financiers. Malgré ces circonstances, ma **passion** pour l'informatique n'a jamais faibli. J'ai continué à me former de manière autodidacte, en suivant des tutoriels en ligne et en participant à des projets personnels. Mais aussi en développant des outils internes dans mon entreprise (sur EXCELL, en **VBA**.

Aujourd'hui, je me tiens devant vous, devant ce jury, pour valider mon diplôme de **Concepteur Développeur d'Applications** suite à ma reconversion professionnelle. Cette opportunité représente un nouveau départ pour moi, une chance de concrétiser ma **passion** et d'atteindre mes **objectifs.**

Je suis impatient de partager avec vous mon travail et mes réalisations lors de cette **présentation**. Je suis reconnaissant pour cette **opportunité** et je suis déterminé à prouver ma valeur en tant que **Concepteur Développeur d'Applications**.

**Merci de votre attention.**

**Cordialement,**

**Who am I ?**

**"**

**"**

**Hello everyone,**

My name is **Maxence ALLART**, and I am delighted to introduce myself to you today. Since a young age, I have been passionate about computers and programming. My fascination with this field began in middle school when I discovered CASIO calculators and the BASIC language. I spent countless hours programming my own little games! It was at that moment that I knew I wanted to pursue a career in software development.

After completing my baccalaureate, I decided to pursue a path in industrial sciences and technologies (STI), which had a significant programming component. It was there that I had the opportunity to delve into programming, starting with the C language (yes, that was over 15 years ago!). I also dived into the world of web development, learning languages such as HTML, CSS, JavaScript, PHP, and SQL.

Unfortunately, due to family issues, I had to interrupt my studies during the second year of my BTS program. This period was challenging for me, and I had to take on "survival jobs" to meet my financial needs. Despite these circumstances, my passion for computer science never waned. I continued to self-educate by following online tutorials and engaging in personal projects. I also developed internal tools within my company using Excel and VBA.

Today, I stand before you, before this jury, to validate my degree as a Concepteur Développeur d'Applications (Application Developer) following my career change. This opportunity represents a fresh start for me, a chance to materialize my passion and achieve my goals. I am proud of the progress I have made so far, and I am confident that my determination and acquired skills will lead to success in this new stage of my career.

I am eager to share my work and achievements with you during this presentation. I am grateful for this opportunity and determined to prove my worth as a Software engineer.

**Thank you for your attention.**

**Best regards,**

**Présentation du projet**

**"**

**"**

Vous avez en annexe l’ébauche de cahier des charges. Je ne vais pas vous présenter en détails la totalité de celui-ci, mais je vais vous présenter les grandes lignes du projet.

Notez que le projet a été conduit par 2 personnes, Mickael et moi-même.

Vous l’aurez compris, le projet « **PcCompare** », est un **comparateur** de prix de matériel informatique : il s’agit d’une idée que j’ai toujours eu, en effet, tous ceux qui ont déjà fait leur propre config. fixe par eux-mêmes le savent, trouver les meilleurs prix pour chaque composant, est assez fastidieux et on se retrouve à jongler avec plusieurs sites en fonction des promos, des prix différents ...

**Ensuite**, la seconde partie du projet est de proposer un « builder » de config. Qui sera chargé de vérifier la compatibilité entre les produits sélectionnés.

**Enfin**, ultimement, cela pourrait se rapprocher d’un réseau social, avec la possibilité de partager ses config, ses bons-plans etc…

Il a été décidé via le cahier des charges, de découper le projet en trois grands « sprint » :

* **1ère version** qui contient toute la partie affichage (page liste des produits, page produit en détails, système de login, système de note et de commentaire).
* **2nd version** qui contient le « scrapping » automatique des données.
* **3ème version** qui contient un module de configuration (sauvegarder des pièces + vérifications de la compatibilité entre eux.

**Outils utilisés :**

**"**

**"**

L’IDE utilisé est « **VScode** », un incontournable qui n’est pas nécessaire de présenter. Notez que ***SonarLINT*** a été utilisé pour éviter les erreurs grossières.

« **Figma** », comme le **papier** et le **crayon** ont été utilisé pour effectuer les premières maquettes !

Pour s’organiser dans une équipe de 2, il n’était pas nécessaire d’utiliser « **Jira** », « **Trello** » a été choisi pour désigner « qui fait quoi » journée par journée.

« **Wamp** » a été choisi pour le développement en local, cela permet d’avoir un server Apache en local d’une Système de Gestion de Base de Données Relationnelles (qui sera décrite plus tard).

« **Mockaroo** » permet d’ajouter des « fake data », cela a été utilisé pour le développement de la première version.

« **Looping** » a été utilisé pour modéliser la base de donnée (MCD, MLD etc…)

Bien entendu, « **GIT** » est un classique pour gérer le « versionning ».

**Stack utilisée : REACT**

**"**

**"**

**Performance élevée** : utilise le DOM **virtuel** pour minimiser les mises à jour réelles du DOM. Lorsqu'un état change, calcule les différences entre le DOM **virtuel** et le DOM **réel**, puis met à jour uniquement les parties affectées, ce qui améliore considérablement les **performances** de l'application.

**Composants réutilisables** : basé sur les composants, qui sont des blocs de construction indépendants et auto-suffisants. Les composants encapsulent le HTML, la logique et les styles associés, ce qui permet de les réutiliser facilement dans différentes parties de l'application. Cela favorise la modularité, la maintenabilité et l'efficacité du développement.

**Large communauté de support** : une communauté active et dynamique. Cette communauté fournit une vaste gamme de ressources, telles que des tutoriels, des exemples de code, des bibliothèques tierces et des solutions aux problèmes courants. Cette abondance de connaissances offre un support précieux lors du développement.

**Mises à jour régulières** : constamment mis à jour avec de nouvelles fonctionnalités, des améliorations de performance et des corrections de bugs. Les mises à jour régulières garantissent que les développeurs ont accès aux dernières améliorations et peuvent bénéficier des nouvelles fonctionnalités dès qu'elles sont disponibles.

**En résumé**, React JS offre des avantages significatifs pour le développement d'applications modernes. Il permet de créer des interfaces utilisateur performantes, réutilisables et maintenables, tout en offrant une large communauté de support, un écosystème riche et une adaptabilité à différentes plateformes. Que ce soit pour le développement d'applications web, mobiles ou de bureau, c‘est un choix populaire et puissant.

**Stack utilisée : Node/Express**

**"**

**"**

**Hautes performances** : Node.js est basé sur le moteur JavaScript V8 de Chrome, ce qui lui confère des performances élevées et une grande efficacité en termes de gestion des requêtes entrantes. Il utilise un modèle d'E/S asynchrone et non bloquant, ce qui lui permet de gérer simultanément un grand nombre de connexions sans provoquer de blocages.

**Évolutivité** : Node.js est hautement évolutif. Grâce à son modèle de traitement asynchrone et à sa capacité à gérer efficacement les connexions persistantes, il peut facilement gérer un grand nombre de requêtes simultanées avec une utilisation efficace des ressources système.

**Large écosystème** : Node.js bénéficie d'un vaste écosystème de modules et de bibliothèques open-source disponibles via le gestionnaire de packages npm. Cela facilite le développement en permettant aux développeurs d'utiliser des packages préexistants pour des fonctionnalités telles que **l'authentification**, la **manipulation des images**, l'envoi de **courriers électroniques**, etc.

**Framework minimaliste** : Express offre une couche d'abstraction légère au-dessus de Node.js, ce qui facilite le développement d'applications web en fournissant des fonctionnalités essentielles pour la gestion des **routes**, la gestion des **requêtes** et la création de **middleware**.

En résumé, **Node.js + Express** offre une flexibilité et une gestion puissante des requêtes. Ensemble, ils constituent une combinaison puissante pour le développement d'applications web rapides, évolutives et modulaires.

**Stack utilisée : mySQL**

**"**

**"**

**Fiabilité et stabilité** : MySQL est connu pour sa fiabilité et sa stabilité. Il est largement utilisé dans des environnements de production à grande échelle, offrant une robustesse et une disponibilité élevées pour la gestion des données.

**Historique** : MySQL bénéficie d'une large communauté de développeurs et d'utilisateurs qui fournissent un support, des ressources et des mises à jour régulières.

**Facilité d'utilisation** : MySQL est réputé pour sa facilité d'installation, de configuration et de gestion. Avec **PHPmyAdmin**, les tâches courantes telles que la création de tables, la gestion des utilisateurs et la surveillance des performances sont simplifiées et visuelles.

**Accessibilité** : En tant que système de gestion de base de données open-source, MySQL est accessible à tous gratuitement. Cela signifie que les développeurs et les entreprises peuvent l'utiliser sans frais de licence initiaux ou de coûts de déploiement élevés, ce qui le rend économiquement avantageux, en particulier pour les petites entreprises et les projets à budget limité.

**En résumé**, MySQL offre une combinaison de fiabilité, de performances élevées, de support communautaire, de flexibilité et de sécurité avancée, en faisant un choix solide pour la gestion de bases de données dans une variété d'applications, allant des petites applications web aux systèmes critiques à grande échelle.

**Librairies externes React**

**"**

**"**

(Voir Annexe XXXXXXXXX)

**Redux Toolkit** est un ensemble d'outils officiels de Redux pour faciliter la gestion de l'état dans les applications React. Il fournit des fonctions et des utilitaires qui simplifient la configuration du store Redux, la définition des actions, la création des reducers, etc.

**Axios** est une bibliothèque populaire pour effectuer des requêtes HTTP. Elle facilite l'envoi de requêtes GET, POST, PUT, DELETE, etc., et la gestion des réponses et des erreurs. Elle prend en charge les fonctionnalités telles que les intercepteurs, les annulations de requêtes et la gestion des cookies.

**Date-fns** est une bibliothèque légère pour la manipulation et la gestion des dates dans JavaScript. Elle offre une API simple et fonctionnelle pour effectuer des opérations courantes telles que le formatage, le calcul, la comparaison, la conversion de fuseaux horaires, etc...

**Dotenv** permet de charger des variables d'environnement à partir d'un fichier .env dans votre application. Cela permet de stocker des informations sensibles (comme des clés d'API) en dehors du code source, ce qui facilite la configuration de l'application lors du déploiement sur différents environnements.

**"react-google-charts"** : Cette bibliothèque permet d'intégrer facilement des graphiques interactifs de Google Charts dans une application React. Elle fournit des composants React qui facilitent la création de graphiques tels que les graphiques en barres, les graphiques en ligne, les graphiques circulaires, etc., en utilisant l'API Google Charts.

**React Hook Form** est une bibliothèque de gestion de formulaires légère et performante pour React. Elle permet de gérer facilement les états et les validations des formulaires, de gérer les erreurs, les soumissions, etc., en utilisant des hooks React.

**React Toastify** est une bibliothèque de notifications légère et facile à utiliser pour React. Elle permet d'afficher des notifications, des alertes ou des messages d'erreur à l'utilisateur de manière élégante et réactive. Vous pouvez personnaliser l'apparence des notifications, définir des options de temporisation, gérer les événements de fermeture, etc. Elle offre une API simple pour afficher des notifications de manière programmée ou en réponse à des événements spécifiques.

**Styled Components** est une bibliothèque populaire qui permet de styliser les composants React en utilisant des styles CSS déclaratifs. Elle permet d'écrire des styles en tant que composants React, ce qui facilite la gestion des styles au sein de votre application. Vous pouvez définir des styles spécifiques à un composant, utiliser des interpolations de variables et de fonctions, gérer les pseudo-classes et les médias queries, et bénéficier d'une isolation des styles pour éviter les conflits.

**Exemple : styled-component**

**"**

**"**

(Voir ANNEXE XXXXXXXX)

**Dans le JSX**, nous n’utilisons plus de <div>, <form> ou autre, mais des composants !

**Dans ces composants**, on utilise du CSS classique (voir annexe XXX.XXX). Et comme nous sommes dans REACT, nous pouvons passer des « props » et donc faire du style conditionnel en utilisant directement les « state » du composant dans lequel on est !

**Par exemple**, (voir annexe XXXX.XXX), on passe un state « acceptedCookies » en « props » dans le styled-component, afin d’afficher, ou pas le pop-up d’acceptation des cookies.

**Ensuite**, cela permet également de se créer facilement une petite bibliothèque de composant afin d’harmoniser les styles dans toute l’application (voir annexe XXXX.XXX) et en combinant les « props », nous avons une bibliothèque de composants modulables !

**Enfin,** (voir annexe XXXX.XXX), les styles sont encapsulés dans les composants où ils sont utilisés. Cela signifie que les styles ne risquent pas d'entrer en conflit avec d'autres parties de votre application. Chaque composant a ses propres styles isolés, ce qui facilite le développement et la gestion des styles.

**Bien sûr** tout cela est possible sans cette librairie, mais étant des composants REACT, on est en lien direct avec l'écosystème React. En effet, les styles définis avec Styled Components sont automatiquement optimisés pour React, ce qui améliore les performances de l'application.

**Exemple : react-hook-form**

**"**

**"**

(Voir annexe XXXXX)

TODO

**Exemple: react-router-dom**

**"**

**"**

(Voir annexe XXXXX)

TODO

**Exemple : Redux/Toolkit**

**"**

**"**

(Voir annexe XXXXXX)

TODO

**Librairies externes NodeJS**

**"**

**"**

(Voir annexe XXXXXX)

**Bcrypt** est une bibliothèque utilisée pour le hachage sécurisé des mots de passe. Elle permet de stocker les mots de passe de manière sécurisée en les hachant avec un algorithme de hachage fort, rendant difficile la récupération du mot de passe en clair.

**Cookie-parser** est une bibliothèque middleware pour Express qui permet de lire et de traiter les cookies dans les requêtes HTTP. Elle facilite la manipulation des cookies

**CORS (***Cross-Origin Resource Sharing***)** est une bibliothèque middleware pour Express qui gère les problèmes de sécurité liés aux requêtes effectuées à partir d'un domaine différent. Elle permet de contrôler les autorisations d'accès aux ressources de votre application en fonction de l'origine de la demande.

**Date-fns** est une bibliothèque légère et fonctionnelle permet la manipulation et la gestion des dates dans JavaScript.

**Dotenv** est utilisée pour charger les variables d'environnement à partir d'un fichier .env. Cela permet de stocker des informations sensibles, telles que des clés d'API ou des identifiants de base de données, en dehors du code source et de les charger en tant que variables d'environnement.

**Express Rate Limit** est un middleware pour Express qui permet de limiter le nombre de requêtes par adresse IP ou par utilisateur dans un laps de temps donné. Cela aide à prévenir les attaques par force brute et les abus de l'API en limitant le nombre de requêtes.

**Express Validator** est une bibliothèque middleware qui facilite la validation des données d'entrée dans les requêtes HTTP. Elle offre des fonctionnalités pour valider les champs de formulaire, les paramètres d'URL, les en-têtes de requête, etc., et facilite la gestion des erreurs de validation.

**Jsonwebtoken** est une bibliothèque utilisée pour générer et vérifier les tokens JSON Web (JWT) dans les applications Node.js. Elle permet de créer des tokens sécurisés pour l'authentification et l'autorisation des utilisateurs.

**Multer** est une bibliothèque middleware pour Express qui facilite la gestion des fichiers téléchargés dans les requêtes HTTP. Elle permet de recevoir des fichiers depuis les formulaires et de les enregistrer sur le serveur.

**MySQL2** est un pilote MySQL pour Node.js. Il permet d'interagir avec une base de données MySQL en exécutant des requêtes, en récupérant et en insérant des données, en gérant les transactions, etc.

**Nodemailer** est une bibliothèque pour l'envoi d'e-mails depuis Node.js. Elle facilite l'envoi de courriers électroniques en utilisant différents services de messagerie et protocoles tels que SMTP, Sendmail, etc.

**Nodemon** est un outil de développement pour Node.js qui redémarre automatiquement le serveur chaque fois que des modifications de fichiers sont détectées. Cela permet un processus de développement plus fluide et évite d'avoir à redémarrer manuellement le serveur à chaque fois.

**UUID** est une bibliothèque utilisée pour générer des identifiants uniques universels (Universally Unique Identifier) dans les applications Node.js. Elle fournit des fonctions pour créer des identifiants aléatoires uniques qui peuvent être utilisés pour identifier de manière unique des entités dans votre application.

**Exemple : Bcrypt**

**"**

**"**

(Voir annexe XXXXXX)

TODO

**Exemple : Multer**

**"**

**"**

(Voir annexe XXXXXX)

TODO

**Exemple : Nodemailer**

**"**

**"**

(Voir annexe XXXXXX)

TODO

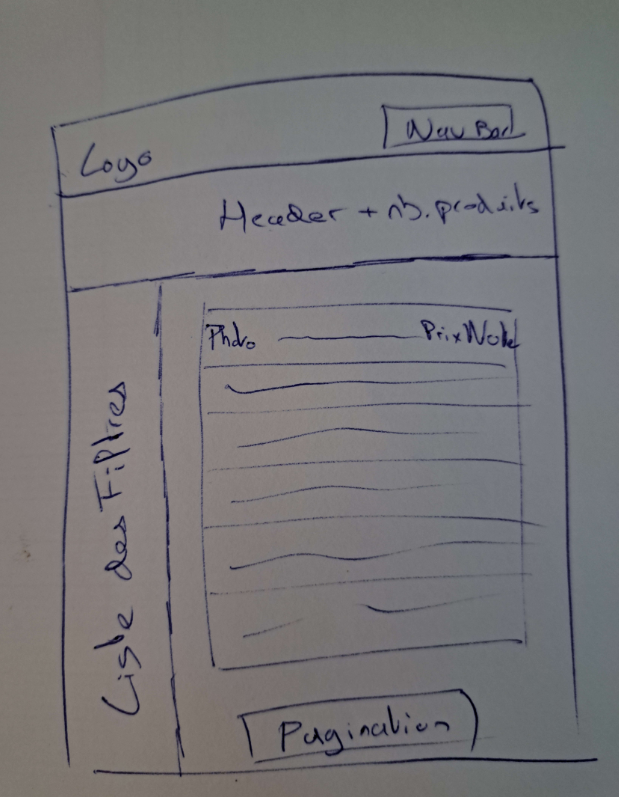
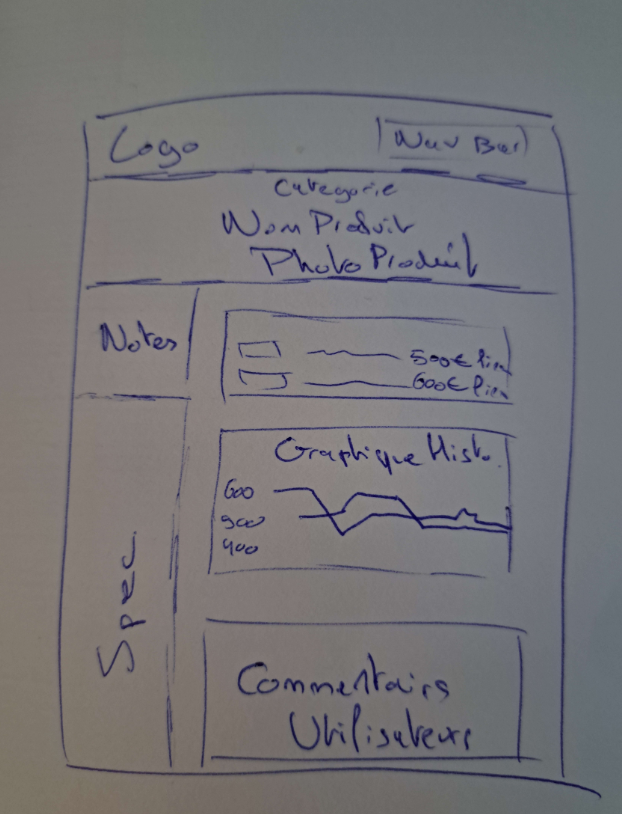
**Conception de l’application**

**"**

**"**

La première étape de la conception, a été de faire une maquette afin d’avoir une première idée d’un rendu final.

La toute première ébauche a été faite avec du papier et un crayon ☺ :

Voici la seconde ébauche de la page principale du site version mobile, faite sur **FIGMA**, qui permet de se diriger vers les différentes parties du site :

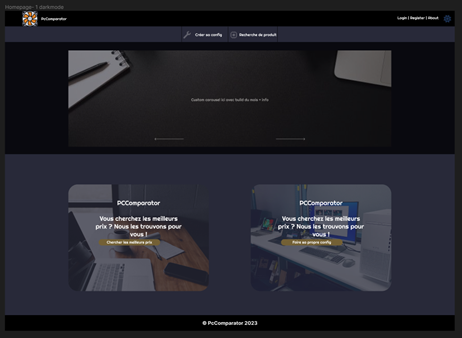
(Voir annexe XXX)

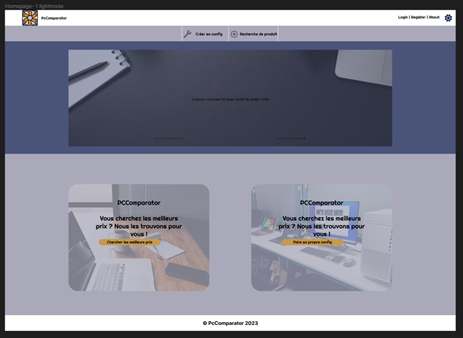
 

*On peut noter un design assez classique, une navbar responsive. Un carousel pour y afficher les dernières informations du site (notez que ce carousel sera manipulable directement par les admins…)*

Voici cette même page mais en version desktop :

(Voir annexe XXXXX)





*On peut noter que le design est responsive !*

**Charte Graphique**

**"**

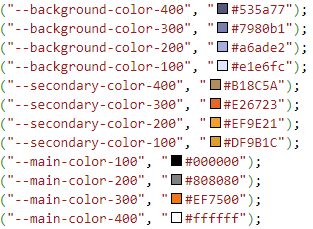
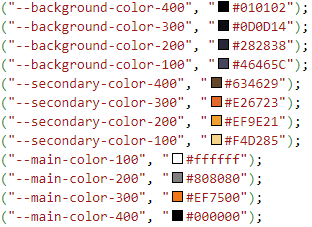
**"**

(Voir annexe XXXXX)

Comme vu dans le cahier des charges, le design du site doit rester sobre et « classique ». La police et les couleurs utilisées doivent respecter la norme d’accessibilité WCAG 2.0.

Actuellement, nous proposons 2 thèmes, light & dark.

**Voici notre charte graphique :**



**Notre logo :**



**La police utilisée :**

'Oxygen Mono'

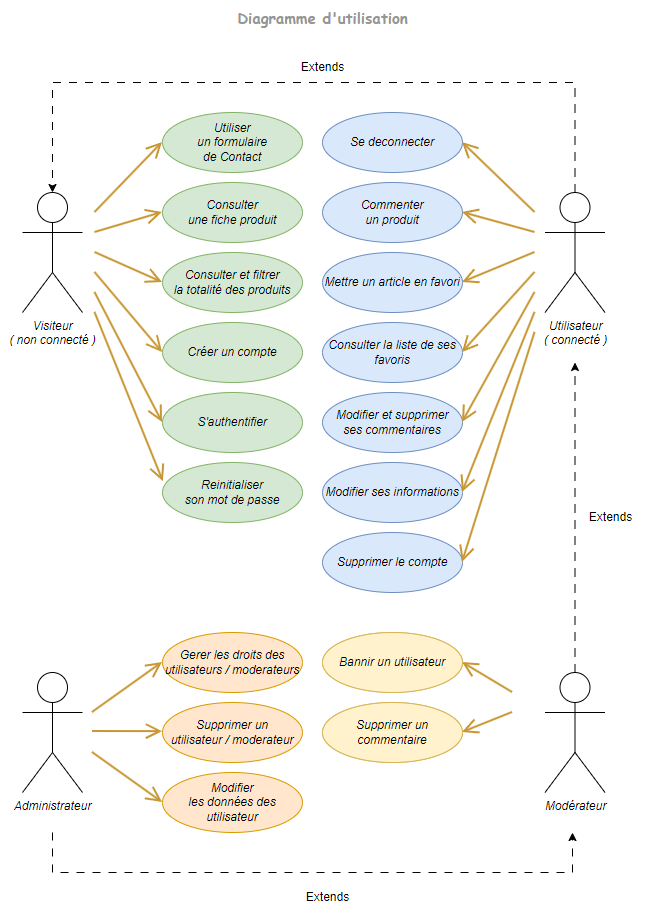
**Cas d’utilisations :**

**"**

**"**

Il est maintenant temps de définir les actions qui seront possibles pour chaque type d’utilisateur, nous aurons 4 types d’utilisateurs :

* **Visiteur**, qui n’est pas identifié (non-loggé) :
  + Qui peut utiliser le site de manière classique.
* **Utilisateur**, qui est identifié (loggé) :
  + Qui a les mêmes pouvoirs qu’un Visiteur.
  + Peut en plus commenter les produits, les mettre en favoris, consulter ses favoris et ses commentaires.
  + Peut en plus supprimer son compte.
  + Peut également modifier ses informations.
* **Modérateur** :
  + Qui a les mêmes pouvoirs qu’un Utilisateur.
  + Peut en plus bannir un utilisateur (empêcher de faire des commentaires).
  + Peut en plus supprimer un commentaire.
* **Administrateur** :
  + Qui a les mêmes pouvoirs qu’un Modérateur.
  + Peut en plus ajouter/supprimer les droits administrateur.
  + Supprimer un utilisateur/modérateur.
  + Modifier totale de toutes les données d’un utilisateur.



**Conception de la base de donnée**

**"**

**"**

J’utilise la méthode Merise afin de concevoir ma base de données.

Nous avons vu précédemment, via le cahier des charges et les cas d’utilisations, les besoins et les objectifs.

Avant de concevoir le Modèle Conceptuel des Données, il faut identifier les différentes entités et les relations entre-elles :

L’élément principal du site sera donc l’entité « Article » :

* Pourra être : une carte graphique, un processeur etc…
* Sera d’un model particulier : une RTX 4090, un Ryzen 4600 etc...
* Appartiendra à une catégorie : une carte graphique, un processeur etc…
* Possèdera un historique de prix
* Sera lié à des commentaires.
* Possiblement dans les favoris d’un utilisateur.

On regroupera toutes les informations communes à 1 article dans l’entité article et on déportera toutes les autres informations dans d’autres entités ! (à voir plus tard)

Un autre élément important sera **l’utilisateur**, il faut donc créer les entités qui représenteront un utilisateur dans la base de données.

Faisons un focus sur cette **entité** et ses **relations** :

Un compte utilisateur sera composé des éléments suivants :

* Adresse mail -
* Mot de passe -
* Pseudo -
* Nom -
* Prénom -
* Avatar -
* Date de dernière connexion -
* Rôle -

Cela donne donc 3 entités :

**customer (Client) :**

* Nom (Nom du client)
* Prénom (Prénom du client)
* Pseudo (Pseudonyme du client)
* Avatar (Image/avatar du client)
* Date de dernière connexion (Date de la dernière connexion du client)

**account (Compte) :**

* Adresse mail (Adresse e-mail associée au compte)
* Mot de passe (Mot de passe du compte)

**role (Rôle) :**

* Titre du rôle (Nom ou titre du rôle)

Maintenant, décrivons les relations entre ces entités :

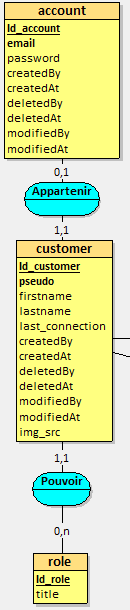
* **Relation entre Customer et Account :**
* Un Customer peut avoir un seul Account (relation 1:1).
* Un Account est associé à un seul Customer (relation 1:1).

*Cette relation est représentée par une ligne reliant les entités Customer et Account.*

* **Relation entre Customer et Role :**
* Un Customer peut avoir un seul Role (relation 1:1).
* Un Role peut être associé à plusieurs Customers (relation 1:N).

*Cette relation est représentée par une ligne reliant les entités Customer et Role.*

Voici ce que cela donne : (Voir annexe XXXX)

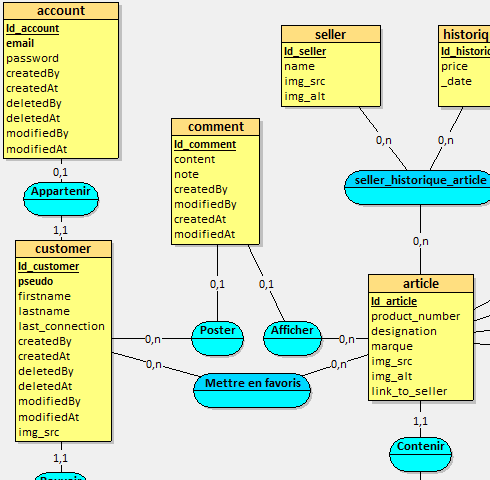


*Nb : On ajoute des attributs createdBy / deletedBy etc… pour garder une trace d’un compte supprimé et savoir qui l’a supprimé !*

**Maintenant ajoutons la possibilité d’ajouter des articles en favoris !**

* Relation entre Customer et Article :
* Un Customer peut mettre en favoris plusieurs Articles (N:N).
* Un Article peut être mis en favoris par plusieurs Customers (N:N).

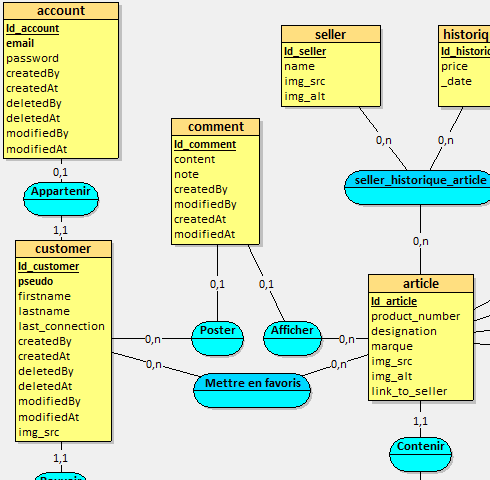
(Voir annexe XXXXXX)



**Maintenant ajoutons la possibilité d’ajouter un commentaire par article.**

* Relation entre Customer et Comment :
* Un Customer peut faire plusieurs Commentaires (relation 1:N).
* Un Commentaire est fait par un seul Customer (relation 1:1).
* Relation entre Comment et Article :
* Un Article peut contenir plusieurs Commentaires (relation 1:N).
* Un Commentaire est lié à un seul Article (relation 1:1).

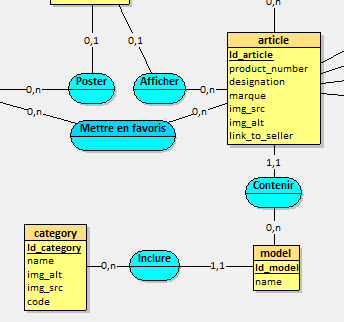
(Voir annexe XXXXXX)



Repassons sur l’entité **Article**, décrivons les relations avec **Model**, et **Category** :

* Relation entre **Article** et **Model** :
* Un **Article** *contient* forcément Un **Model**.
* Un **Model** peut *contenir* plusieurs **Article**.
* Relation entre **Model** et **Category** :
* Un **Model** *inclus* forcément Une **Category**. (relation N:1)
* Une **Category** peut *inclure* plusieurs **Model**. (relation 1:N)

(Voir annexe XXXXXX)



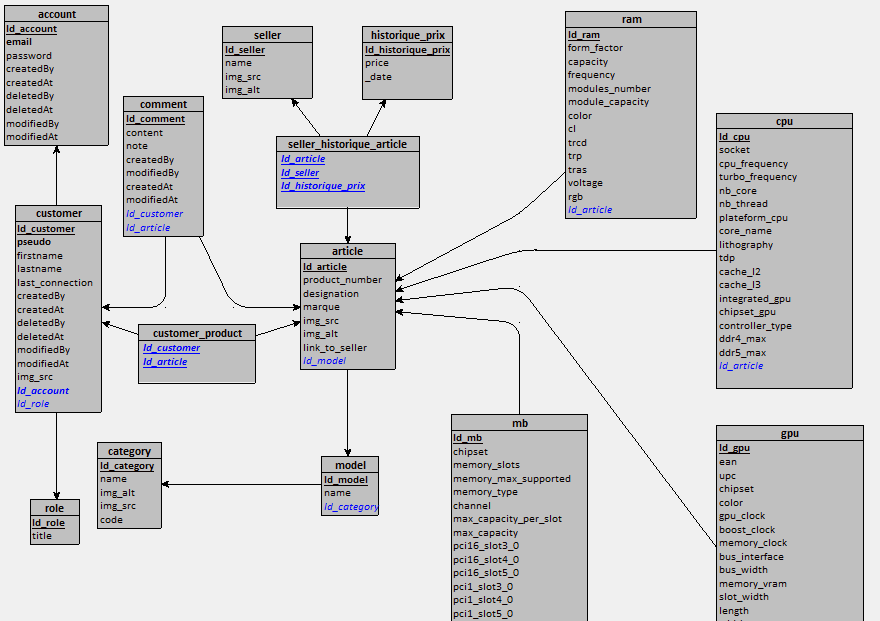
*Notez que* ***Category*** *contient un attribut “code”, ce qui me permet de savoir quelle entité choisir pour afficher les spécifications associées. Cela permet également d’éviter une référence circulaire entre* ***category*** */* ***model*** */* ***article*** */* ***ram*** *par exemple.*

TODO : Pareil pour relation seller / historique\_prix / article.

Il est donc maintenant temps de faire le Modèle Logique de Données : il s’agit de convertir les entités qu’on vient de voir, en forme exploitable par une base de donnée relationnelle.

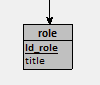
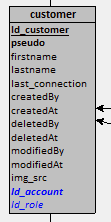
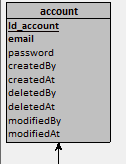
Il faut en effet convertir les entités en table et les relations entre-elles en clé primaire et clé étrangère. Nous utilisons looping qui effectue cette conversion automatiquement et facilement !

Voici ce que cela donne : (Voir annexe XXXXXX)



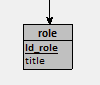
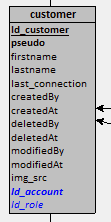
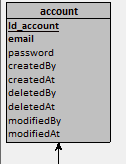
**Analysons** pour l’instant la partie **Account** / **Customer** / **Role** ; Nous avons créé 3 tableaux :

* **Customer (Client) :**
* Id\_customer (Identifiant du client)
* Nom (Nom du client)
* Prénom (Prénom du client)
* Pseudo (Pseudonyme du client)
* Avatar (Image/avatar du client)
* Date de dernière connexion (Date de la dernière connexion du client)
* **Clé primaire** *Id\_Customer*
* **Clé étrangère** : *Id\_account* (fait référence à Customer.Id\_account)
* **Clé étrangère** : *Id\_role* (fait référence à Role.Id\_role)
* **Account (Compte) :**
* Id\_account (Identifiant du compte)
* Adresse mail (Adresse e-mail associée au compte)
* Mot de passe (Mot de passe du compte)
* **Clé primaire** : *Id\_account*
* **Role (Rôle) :**
* Id\_role (Identifiant du rôle)
* Titre du rôle (Nom ou titre du rôle)
* **Clé primaire** : *Id\_role*



**Maintenant**, décrivons les relations entre ces entités :

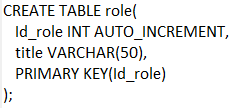
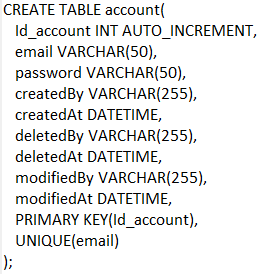
* Relation entre Customer et Account :
* Un Customer peut avoir un seul Account (relation 1:1).
* Un Account est associé à un seul Customer (relation 1:1).
* ***Clé étrangère :*** *Id\_account (dans la table Customer) référence la clé primaire Id\_account (dans la table Account).*
* Relation entre Customer et Role :
* Un Customer peut avoir un seul Role (relation 1:1).
* Un Role peut être associé à plusieurs Customers (relation 1:N).
* ***Clé étrangère :*** *Id\_role (dans la table Customer) référence la clé primaire Id\_role (dans la table Role).*

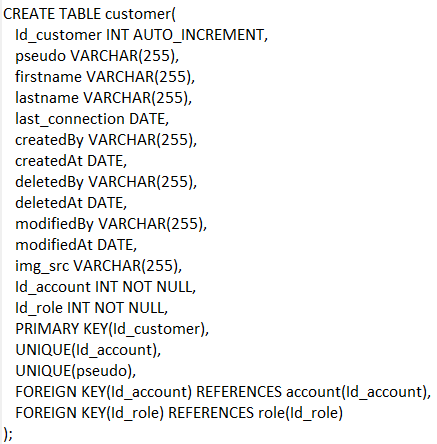


**Maintenant**, il faut passer à l’étape du Modèle Physique des Données (**MPD**), cela consiste à définir le type de chaque attribut, ses valeurs de base, sa taille maximum etc… Comme pour le **MLD**, nous utilisons **Looping** qui nous donne le strict **mySQL** en fonction de notre définition des types dans son interface.

*Pour voir la totalité du MPD, rendez-vous à l’annexe XXXXXX.*

Voici juste un petit aperçu pour les tables : **Account** / **Customer** / **Role** :





**Diagrammes de séquence !**

**"**

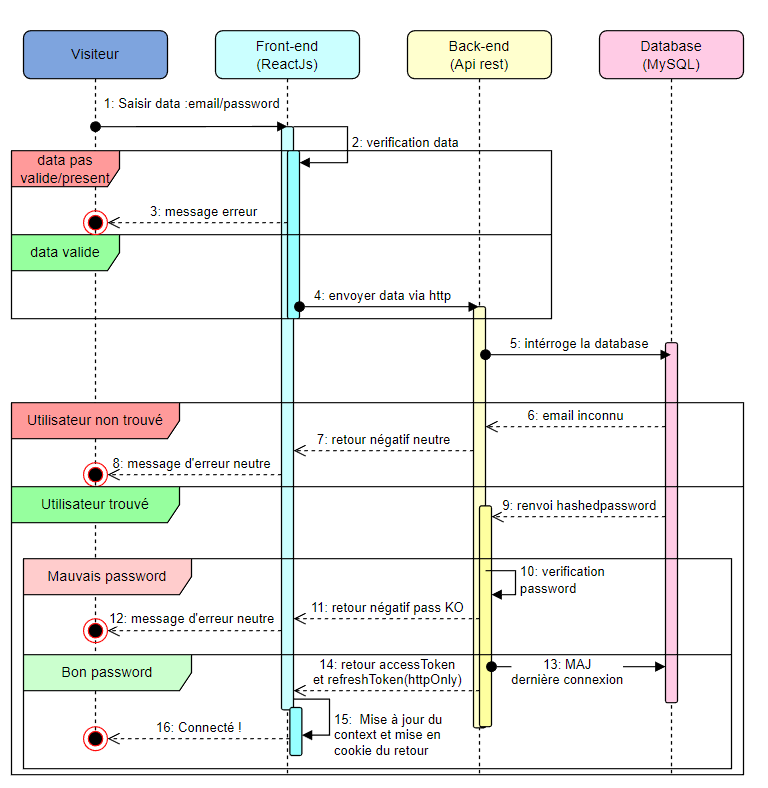
**"**

**Maintenant** que la base de données est modélisée, il est temps de **modéliser** le système de login/register !

Voici le cheminement d’un utilisateur qui tente de s’identifier sur le site :



(Voir Annexe XXXXXX)



1. L’utilisateur saisi sur le site son email et son mot de passe.
2. Le Front-end vérifie le format/présence des données sur le formulaire.
3. Message d’erreur envoyé à l’écran pour indiquer une erreur de saisie / format. (Voir Annexe XXXXX)
4. Il y a bien des données sur le formulaire, on transmet en requête HTTPs à l’API.
5. L’api interroge la database via une requête SQL.
6. La requête SQL retourne aucun résultat, l’API prévient le front via un message neutre (éviter le brute-force/ empêche un pirate de savoir si une personne possède ou non un compte).
7. Le Front-end est prévenu et doit envoyer à l’utilisateur un message.
8. Le Front-end prévient l’utilisateur qu’il y a un souci lors de son authentification. (neutre) (Voir Annexe XXXX)
9. La requête SQL trouve l’utilisateur et on va chercher en database, le mot de passe hashé.
10. L’api vérifie l’authenticité du mot de passe (il compare la saisie avec celui en database via bcrypt).
11. La vérification n’est pas authentifiée, l’api prévient le front qu’il y a un souci lors de son authentification.
12. Le Front-end prévient l’utilisateur avec un message d’erreur neutre (éviter le brute-force/ empêche un pirate de savoir si une personne possède ou non un compte). (Voir Annexe XXXX)
13. Si le mot de passe est vérifié, l’api modifie la database via une requête SQL afin de mettre à jour la « dernière connexion ».
14. L’api retourne au Front-end, un accessToken et un refreshToken(httpOnly)
15. Le Front reçoit ces tokens et peut mettre à jour ses states (contexte)
16. Le Front-end prévient l’utilisateur qu’il s’est identifié avec succès. (Voir Annexe XXXXX)

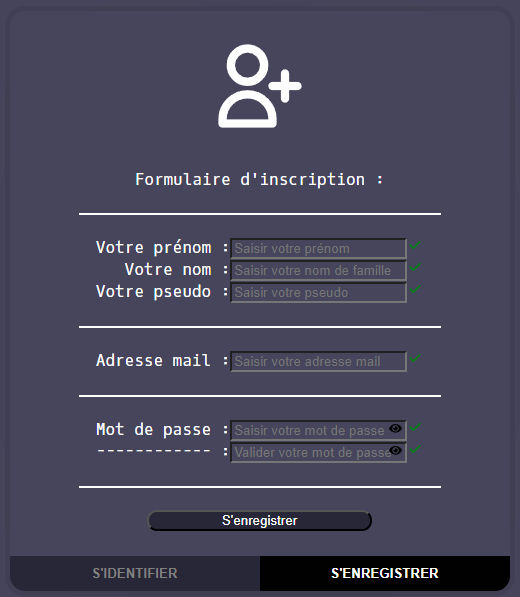
**Diagrammes de séquence !**

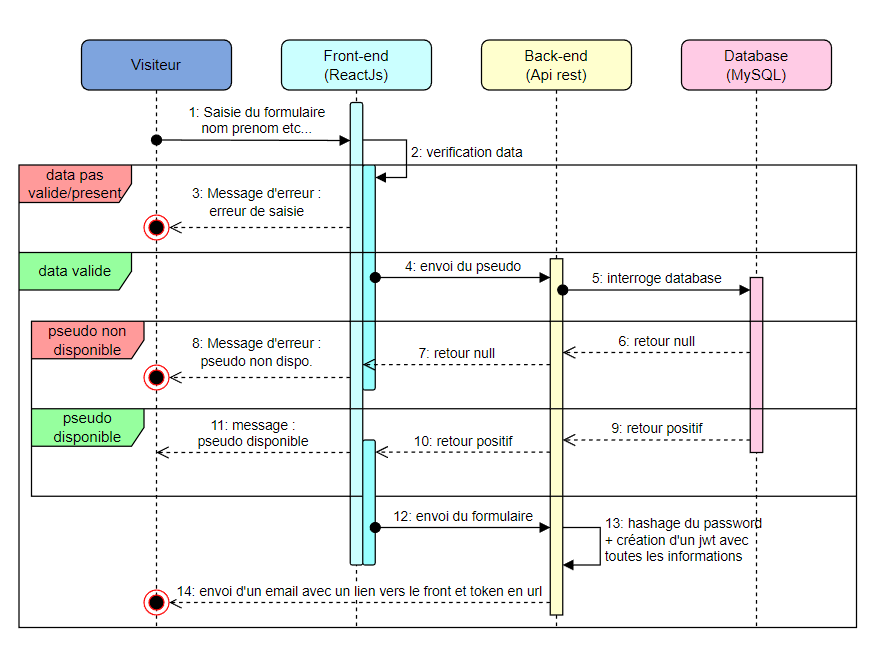
**"**

**"**

**Passons** à la modélisation du système de register !

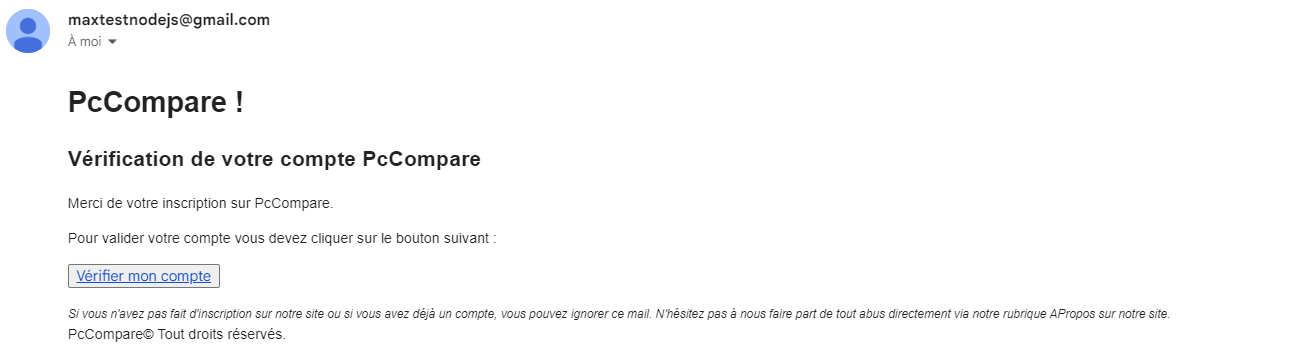
Voici le cheminement d’un utilisateur qui tente de s’enregistrer sur le site :



(Voir Annexe XXXX)

1. L’utilisateur saisi sur le site son email et son mot de passe.
2. Le Front-end vérifie le format/présence des données sur le formulaire.
3. Message d’erreur envoyé à l’écran pour indiquer une erreur de saisie / format. (Voir Annexe XXXXX)
4. Le Front-end transmet le pseudo saisi à l’api afin de vérifier la disponibilité de celui-ci.
5. L’api interroge la database via une requête SQL si ce pseudo est déjà connu ou pas.
6. Le retour indique que ce pseudo est déjà utilisé.
7. L’api prévient le Front-end que ce pseudo n’est pas disponible.
8. Le Front-end affiche à l’écran que le pseudo n’est pas disponible. (Voir Annexe XXXXX)
9. Le retour de la requête est négatif (pseudo inconnu)
10. L’api prévient le Front-end que le pseudo est disponible. (Voir Annexe XXXXX)
11. Le Front-end indique au client que le pseudo est disponible !
12. Le Front-end transmet maintenant le formulaire contenant la totalité des informations du client à l’api.
13. L’api hash le password et crée un JsonWebToken avec toutes les informations
14. L’api via nodemailer envoi un e-mail au client pour qu’il clique sur un bouton de validation de compte. (Voir Annexe XXXXX)

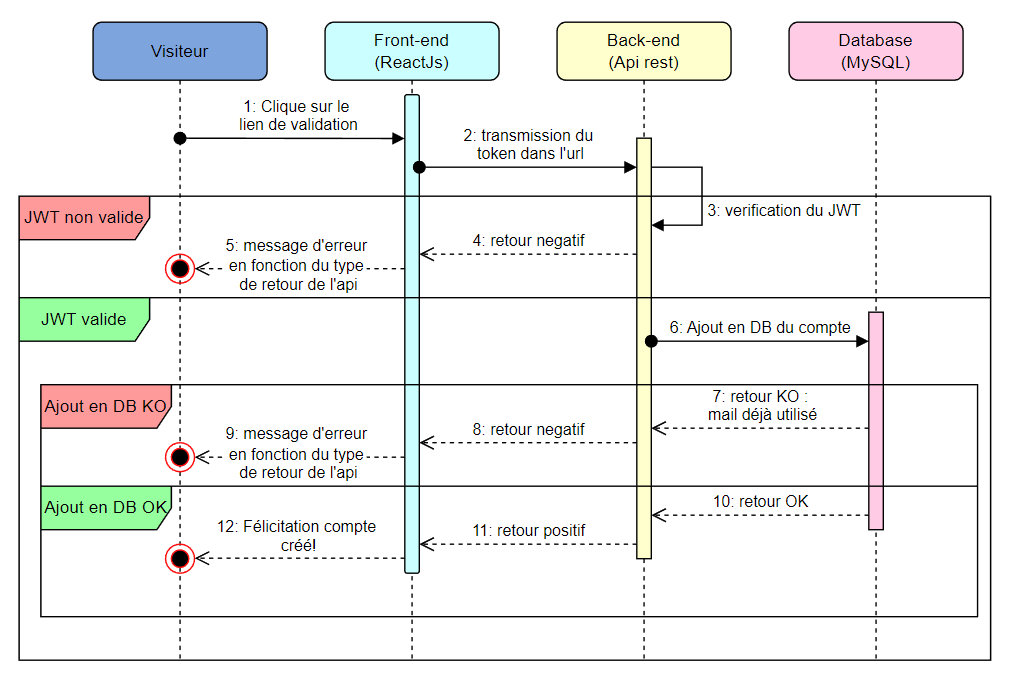
Après avoir reçu le mail que voici :



Si l’utilisateur clique sur ce bouton, il est renvoyé vers cette page :



Voici donc la suite du diagramme de séquence pour l’enregistrement sur le site après avoir reçu et cliqué sur le bouton du mail de validation :



1. L’utilisateur clique sur le bouton de validation
2. Le Front-end transmet via une requête Https à l’api avec le token contenant les informations d’inscription
3. L’api vérifie la validité du JWT
4. La vérification est KO, l’api prévient le Front-End.
5. Le Front-end affiche à l’écran de l’utilisateur un message d’erreur basé sur le retour de l’api.
6. Le JWT est valide, l’api tente d’ajouter via une requête SQL, les informations du customer, et de l’account.
7. Si le retour de la requête indique que l’e-mail est déjà utilisé.
8. L’api prévient le Front-end.
9. Le Front-end affiche à l’écran un message d’erreur en fonction du retour de l’api.
10. L’api a pu ajouter avec succès en database les données.
11. L’api prévient le Front-end.
12. Le Front-end affiche que le compte est créé avec succès !