JUNIA ISEN

2023/2024 – ISEN 3 – Projet de Fin d'Année

RAPPORT D'ACTIVITE

(Département CSM)

DANSE CLUB

Groupe 4

NOM Prénom étudiant 1	Hugo MANY	CSI3 ☐	CIR3 ☒	CNB3 ☐
NOM Prénom étudiant 2	Eliott PERES	CSI3 ☐	CIR3 ☒	CNB3 ☐
NOM Prénom étudiant 3	Charles ESPRIET	CSI3 ☐	CIR3 ☒	CNB3 ☐
NOM Prénom étudiant 4	Adam VIDAL	CSI3 ☐	CIR3 ☒	CNB3 ☐
NOM Prénom étudiant 5	Paul MAZAINGUE	CSI3 ☐	CIR3 ☒	CNB3 ☐
Date de la soutenance	27 juin 2024			

Sommaire

Introduction

Les associations doivent s'adapter dans un monde de plus en plus numérique pour rester compétitives et répondre aux attentes de leurs membres. L'association de danse est une exception à cette règle. Afin de simplifier et d'optimiser la gestion des inscriptions, des paiements, des réservations de cours ainsi que la coordination des activités entre les professeurs et les élèves, nous avons entrepris le développement d'une application dédiée.

Cette application a pour objectif de proposer une solution complète et intuitive pour gérer efficacement les différentes facettes de l'association. L'application automatisera les tâches administratives et centralisera les informations, ce qui permettra non seulement de gagner du temps, mais aussi d'améliorer la communication et l'expérience utilisateur pour tous les membres de l'association.

Le présent rapport traite des différentes étapes du développement de cette application, depuis la conception initiale jusqu'à la mise en œuvre et les tests. Nous allons également étudier les fonctionnalités principales, les technologies utilisées, ainsi que les bénéfices attendus pour l'association et ses membres.

1. Etat de l’art et problématique

Actuellement, de nombreuses associations de danse gèrent leurs activités de manière traditionnelle, en s'appuyant sur des outils disparates tels que les feuilles de calcul, les carnets de notes et les communications par email ou téléphone. Bien que ces méthodes aient prouvé leur utilité, elles présentent des limitations significatives. Par exemple, la gestion manuelle des inscriptions et des paiements peut entraîner des erreurs et des retards, en plus d'imposer une charge administrative importante. La communication au sein de l'association est souvent fragmentée, avec des informations cruciales diffusées par divers canaux, ce qui rend la transmission des messages peu efficace et susceptible de provoquer des malentendus. De plus, le suivi des progrès des élèves et la gestion des plannings des professeurs manquent souvent de centralisation, compliquant ainsi l'organisation et la planification des cours.

Après ces constats, il est évident que plusieurs problèmes doivent être adressés. Il est important d'améliorer les processus administratifs pour automatiser et centraliser la gestion des inscriptions et des paiements, réduisant ainsi les erreurs et la charge administrative. L'amélioration de la communication est également cruciale pour garantir que tous les membres recevront les informations importantes en temps voulu. L'intégration des plannings des professeurs et des élèves permettrait une organisation optimale et un suivi personnalisé des progrès des élèves. La mise en place d'un système de réservation en ligne est indispensable pour éviter les problèmes de sureffectif ou de sous-effectif et garantir une meilleure planification des cours.

La mise en place d'une application destinée à l'association de danse est cruciale pour répondre à ces besoins et résoudre les problématiques actuelles. L'application, en centralisant et en informatisant la gestion des différentes facettes de l'association, permettra d’accroître l’efficacité administrative en automatisant les tâches répétitives et en réduisant les risques d’erreurs humaines. Elle favorisera également la communication entre les membres de l'association grâce à des notifications instantanées et des informations centralisées. En outre, l'application permettra d'améliorer la gestion des plannings en fournissant une vue d'ensemble claire et accessible à tous les intervenants, tout en facilitant les réservations de cours via une plateforme en ligne intuitive pour les élèves.

Ce projet est donc important pour moderniser l'association de danse, améliorer l'expérience de ses membres et assurer une gestion plus fluide et efficace.

2. Solution envisagée

2.1. Présentation de la solution

2.1.1. Exposition de la solution

Cette application aide les associations de danse à gérer leurs cours, leurs entrées et leurs finances. Même pour les utilisateurs non techniques, elle est conçue pour être ergonomique et simple à utiliser.

La solution permet de suivre les entrées, d'informer sur les activités, de respecter certaines réglementations RGPD, de visualiser les cours en fonction de la participation et de comptabiliser les entrées pour rétribuer les professeurs.

Les profils des visiteurs, des élèves, des professeurs et des administrateurs sont gérés par le système. Les visiteurs peuvent accéder aux informations publiques et s'inscrire. Les élèves peuvent accéder aux vidéos et aux photos, vérifier leurs abonnements et gérer leurs crédits. Les enseignants ont le pouvoir de superviser les élèves, les cours, les participations et les paiements en espèces. Toutes ces fonctionnalités sont accessibles aux administrateurs, qui peuvent configurer l'application, créer des cours et des abonnements et suivre les règles.

L'application enregistre les recettes des cours et exporte les données en format CSV ou Excel. Une option de gestion de caisse permet de garder un historique des crédits, des modifications et des débits. Pour faciliter la gestion des accès et des achats sur place, le site Web est compatible avec les appareils mobiles et tablettes.

2.1.2. Périmètre fonctionnel

Cette application doit permettre une meilleure gestion des cours, des entrées et des finances tout en étant ergonomique et facile d'utilisation pour les utilisateurs non techniques.

Notre solution vise à simplifier le suivi des entrées, à informer les utilisateurs sur l'ensemble des activités de l'association, à respecter dans la mesure du possible certaines réglementations RGPD, à proposer une visualisation des cours en fonction de la participation et à comptabiliser les entrées selon les cours pour rétribuer les professeurs.

Les profils tels que les visiteurs, les élèves, les professeurs et les administrateurs peuvent être gérés par le système. Un visiteur peut accéder aux données publiques telles que les contacts, les lieux, les horaires et les prix et créer un compte pour devenir élève. Les élèves peuvent s'inscrire, gérer leurs crédits, vérifier la durée de leur abonnement, consulter l'historique des cours auxquels ils ont participé et accéder aux vidéos et aux photos. En cas d'oubli, ils peuvent également réinitialiser leur mot de passe.

Les enseignants peuvent organiser les cours, ajouter des participations, gérer les paiements en espèces et ajouter, modifier et supprimer des élèves. De plus, ils peuvent détecter les places achetées par période et annuler les cours avant leur début. Les administrateurs ont accès à toutes les fonctionnalités des professeurs, mais ils peuvent également configurer l'ensemble de l'application, créer des cours, des professeurs, des types de cartes et d'abonnements et suivre les règles.

Une image, des professeurs associés, une durée, une date et une heure de début, une récurrence, un titre, un type et une liste de liens constituent les informations d'un cours. Les cours peuvent être payants ou gratuits sous forme d'abonnements, de cartes ou d'unités.

De plus, l'application offre la possibilité de surveiller les bénéfices des cours et offre la possibilité d'exporter les données en format CSV ou Excel. Les différents types de cours peuvent être divisés en différentes catégories de recettes. Il est possible, en option, de configurer une gestion de caisse pour surveiller le contenu de la caisse, avec des actions de crédit, des modifications automatiques avec les achats en espèces et des débit, toutes datées pour un suivi précis.

L'ergonomie du site est conçue pour les utilisateurs non techniques, ce qui rend l'interface simple et rapide. Pour gérer les entrées et les achats sur place, l'affichage est compatible avec les appareils mobiles et tablettes.

Nous n'avions pas de nom de domaine, ce qui a également limité notre capacité à respecter toutes les exigences de la réglementation RGPD. De plus, étant donné que nous ne sommes pas affiliés, nous n'avons pas pu connecter notre application à la plateforme de gestion des paiements en ligne HelloAsso.

En conclusion, cette application vise à offrir une solution complète et facile à utiliser pour la gestion des activités et des finances d'une association tout en prenant en compte les limites techniques et administratives auxquelles nous avons été confrontés.

2.1.3. Choix techniques

Nous avons pris des décisions techniques stratégiques pour assurer la robustesse, l'évolutivité et la facilité d'utilisation de ce projet. Nous avons choisi React, une bibliothèque JavaScript très populaire qui permet de créer des interfaces utilisateur rapides et réactives grâce à son modèle de composants, pour le front-end. La maintenabilité et la scalabilité de notre application sont améliorées grâce à la réutilisation des composants. React bénéficie d'une grande communauté et d'un riche écosystème de bibliothèques complémentaires et offre des performances optimisées grâce au Virtual DOM.

Nous avons choisi d'utiliser Node. Grâce à son modèle d'E/S non bloquant, Node.js est une plateforme JavaScript qui permet l'utilisation du langage JavaScript côté serveur, ce qui le rend particulièrement adapté aux applications en temps réel. Cela nous permet d'utiliser le même langage (JavaScript) sur le front-end et le back-end tout en ayant des performances élevées et une gestion efficace des connexions concurrentes.

De plus, nous avons choisi des bases de données relationnelles mySQL car elles sont robustes, conformes aux normes ACID (atomicité, cohérence, isolation et durabilité) et capables de traiter des requêtes complexes. Les bases de données mySQL garantissent l'intégrité des données, offrent des capacités avancées de requêtage pour des analyses de données détaillées et bénéficient d'une large adoption et d'un soutien communautaire.

Nous utilisons Swagger et Selenium pour améliorer nos processus de développement et de tests. Swagger est un outil de développement et de documentation d'API. Selenium, un framework pour les tests automatisés des applications web, permet de simuler les interactions utilisateur et de vérifier le bon fonctionnement de l'application. Cela nous permet d'automatiser les tests pour une détection précoce des bugs et des régressions, tout en supportant divers langages de programmation pour les scripts de test et en offrant la capacité de tester sur divers navigateurs et plateformes.

Nous pouvons créer des applications performantes, évolutives et maintenables grâce à ces choix techniques. Notre structure solide et flexible est assurée par l'utilisation de React pour le front-end, de Node.js pour le back-end et de bases de données mySQL, tandis que Swagger et Selenium améliorent nos processus de développement et de tests.

2.1.4. Pilotage du projet

2.1.4.1. Estimation des charges

Pour notre projet, nous avons réalisé une estimation détaillée des charges pour chaque tâche. Le système d'authentification via JWT a pris 10 heures pour une personne, tandis que l'intégration du Captcha a également nécessité 10 heures pour une personne. Le hachage du mot de passe a été complété en 2 heures par une personne, et la gestion des erreurs Cors a également pris 2 heures pour une personne.

La mise en place du serveur sur une machine virtuelle Ubuntu a demandé 20 heures pour une personne, tandis que l'installation du serveur NodeJS a pris 4 heures pour une personne. La mise en place de la base de données a requis 6 heures pour une personne. L'envoi de mails automatisés pour la réinitialisation de mot de passe a également pris 6 heures pour une personne.

Pour la documentation Swagger, 30 heures ont été nécessaires, mobilisant deux personnes. Les tests et corrections ont également pris 30 heures pour deux personnes. La gestion des cas d'erreur a demandé 14 heures pour deux personnes. La création du diagramme de la base de données a pris 2 heures pour une personne. L'implémentation des requêtes et des changements demandés par l'équipe front-end a nécessité 15 heures pour deux personnes. La rédaction de points quotidiens pour tenir le reste de l'équipe au courant de l'avancée du back-end a pris 10 heures pour deux personnes.

Les requêtes ont nécessité 40 heures pour deux personnes.

Le développement en CSS a pris 20 heures pour une personne. La création de designs sur Canva a pris 8 heures pour deux personnes, tandis que l'utilisation de Miro pour la collaboration a nécessité 2 heures pour une personne. La création des pages web a été la tâche la plus intensive, nécessitant 60 heures pour trois personnes. Enfin, la gestion des erreurs a pris 20 heures pour trois personnes.

Au total, l'équipe a consacré 618 heures au projet back-end, réparties sur différentes tâches et impliquant plusieurs membres de l'équipe pour assurer la réalisation efficace et en temps voulu de chaque composant du projet.

2.1.4.2. Communication

Pour faciliter la collaboration et la gestion de notre projet, notre équipe a utilisé plusieurs outils de communication et de gestion. Microsoft Teams a été notre principal outil pour les réunions en ligne, le chat et le partage de fichiers. Trello nous a aidés à organiser et à suivre les tâches grâce à ses tableaux de bord visuels et ses cartes. GitHub a été indispensable pour le contrôle de version et la gestion du code, permettant à chacun de collaborer efficacement sur le développement du projet.

Canva a été utilisé pour créer des visuels attrayants et des présentations, ajoutant une touche professionnelle à nos documents. Enfin, Miro nous a permis de réaliser des brainstormings et de collaborer visuellement sur des mind maps et des diagrammes, facilitant la planification et la mise en commun des idées. L'utilisation de ces outils a grandement amélioré notre coordination et notre productivité tout au long du projet.

2.2. Analyse fonctionnelle

2.2.1. Détail des fonctionnalités

L'application comprend une variété de fonctionnalités bien organisées en plusieurs modules. L'application offre des outils pour gérer les finances générales et comprend des fonctions dédiées à la récupération de données essentielles comme les étudiants, les professeurs, les cours et les paiements pour la gestion de la comptabilité.

En ce qui concerne les abonnements, il existe des interfaces spécifiques pour générer des cartes d'abonnement. La gestion des cours est très complète et comprend des fonctionnalités pour gérer les cours, telles que la création, la modification et la suppression des cours. Les interfaces sont faciles à comprendre et permettent aux utilisateurs de gérer ces aspects.

L'application offre des outils pour gérer et suivre les élèves, gérer leurs crédits, ajouter de nouveaux élèves, modifier leurs informations et les supprimer si nécessaire, pour la gestion des élèves. De plus, les enseignants bénéficient de fonctionnalités de gestion complètes, qui incluent des outils pour ajouter de nouveaux enseignants, modifier leurs données et les supprimer.

Des informations détaillées sur divers types de cours, tels que les cours de danse classique, moderne et rock, sont fournies dans le module de blog. Les enseignants peuvent gérer des contacts et des présences, planifier des cours, ajouter des liens et des tags à un cours et accéder aux informations des élèves. Ils peuvent également accéder à un tableau de bord spécialisé dans la gestion centralisée.

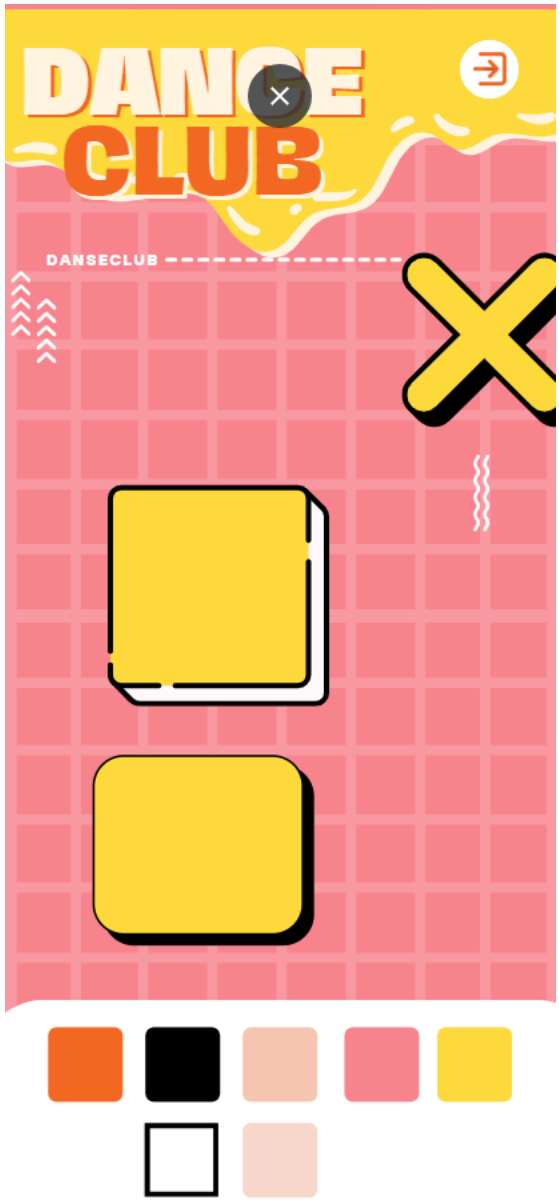
De plus, l'application comprend des fonctionnalités importantes telles que le retour des profils des étudiants, la gestion des crédits des utilisateurs, une boutique pour l'achat de biens ou de services et la gestion des QR codes pour diverses fonctionnalités comme l'accès et la présence. Les utilisateurs peuvent se connecter à l'application via la page de connexion, tandis que la gestion des cours dynamiques permet de configurer des horaires variables. Il existe des outils pour récupérer les ID des étudiants, s'inscrire, consulter la liste des cours disponibles et gérer les entrées pour la présence, et les erreurs d'accès non autorisé sont gérées de manière appropriée.

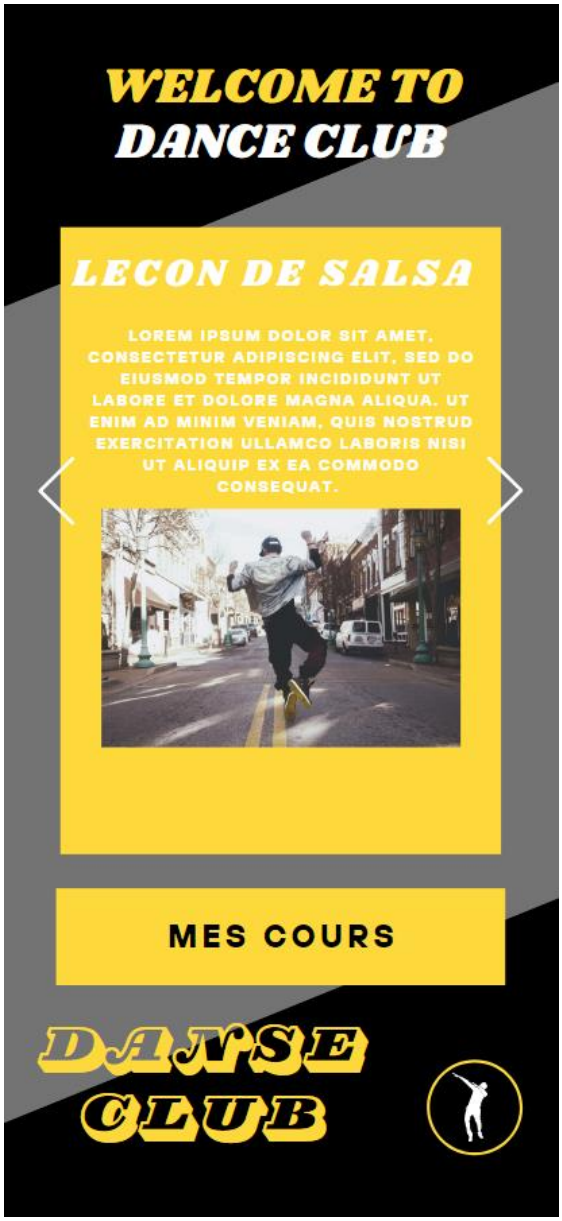
De plus, l'application offre des fonctionnalités de recherche, des pages de profil utilisateur, une planification des cours pour les élèves et des historiques de cours pour les professeurs et les étudiants. Une interface spécifique permet de réserver des cours ou d'autres services. Enfin, il est pris en compte de respecter le RGPD (Règlement Général sur la Protection des Données).

Une feuille de style principale et un composant principal de l'application React font partie des fichiers de configuration principaux. Cette structure modulaire offre une interface utilisateur simple et conforme aux réglementations en vigueur, tout en permettant une gestion complète des cours, des élèves, des professeurs et des abonnements.

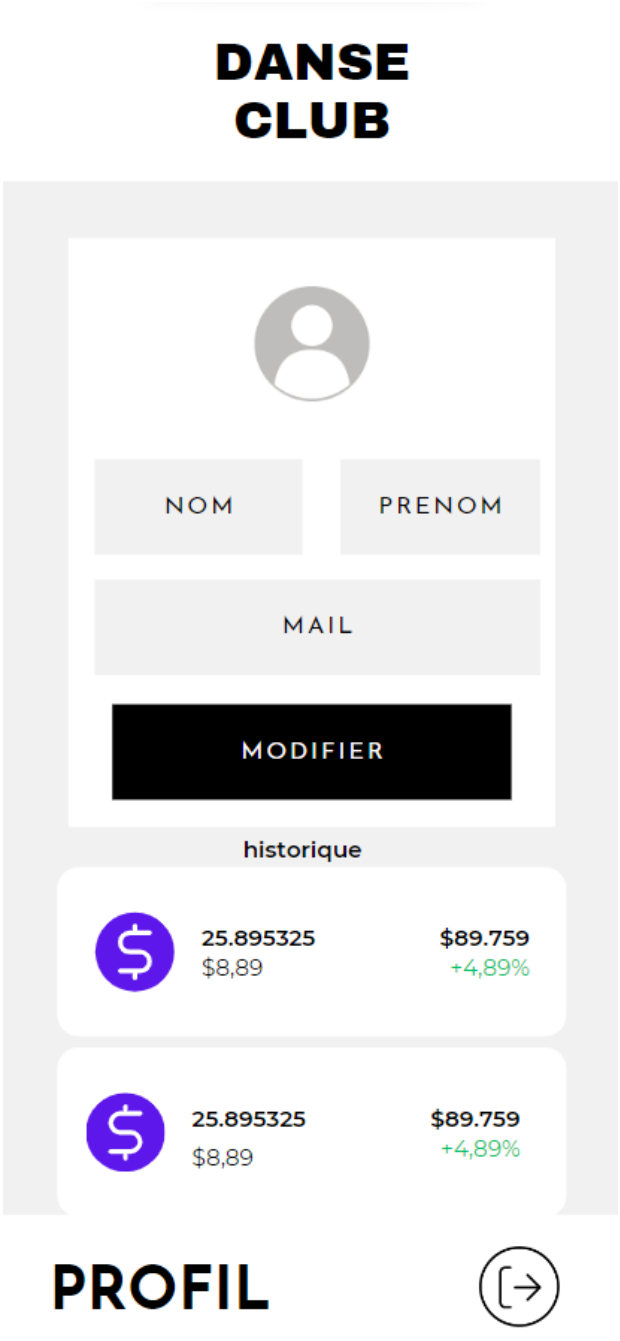
2.2.2. Maquettes

On a eu plusieurs maquettes, tout d'abord quelque chose de très coloré, puis une autre plus sombre mais plus simple :





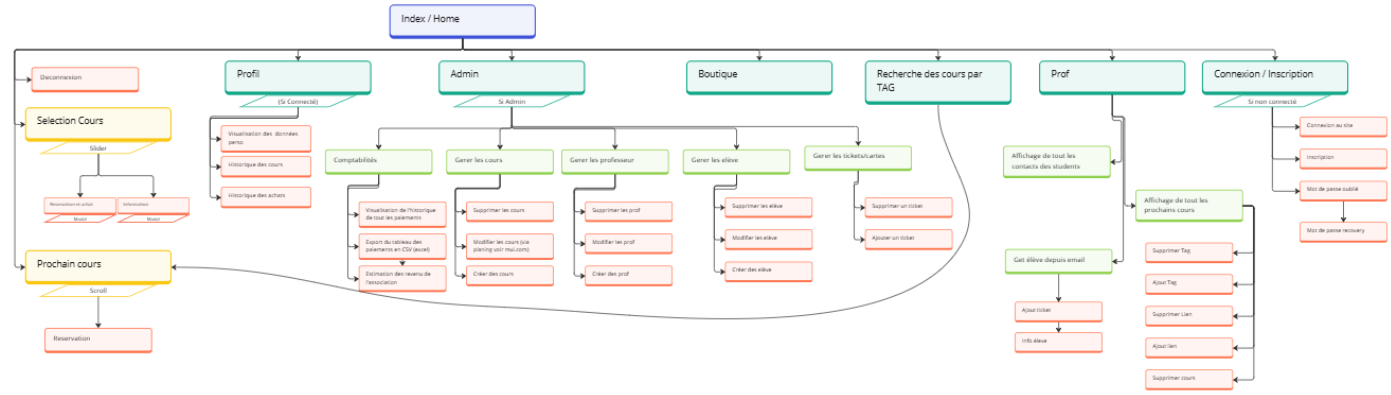
Enfin, nous avons réalisé une maquette plus sobre, simple et pratique :



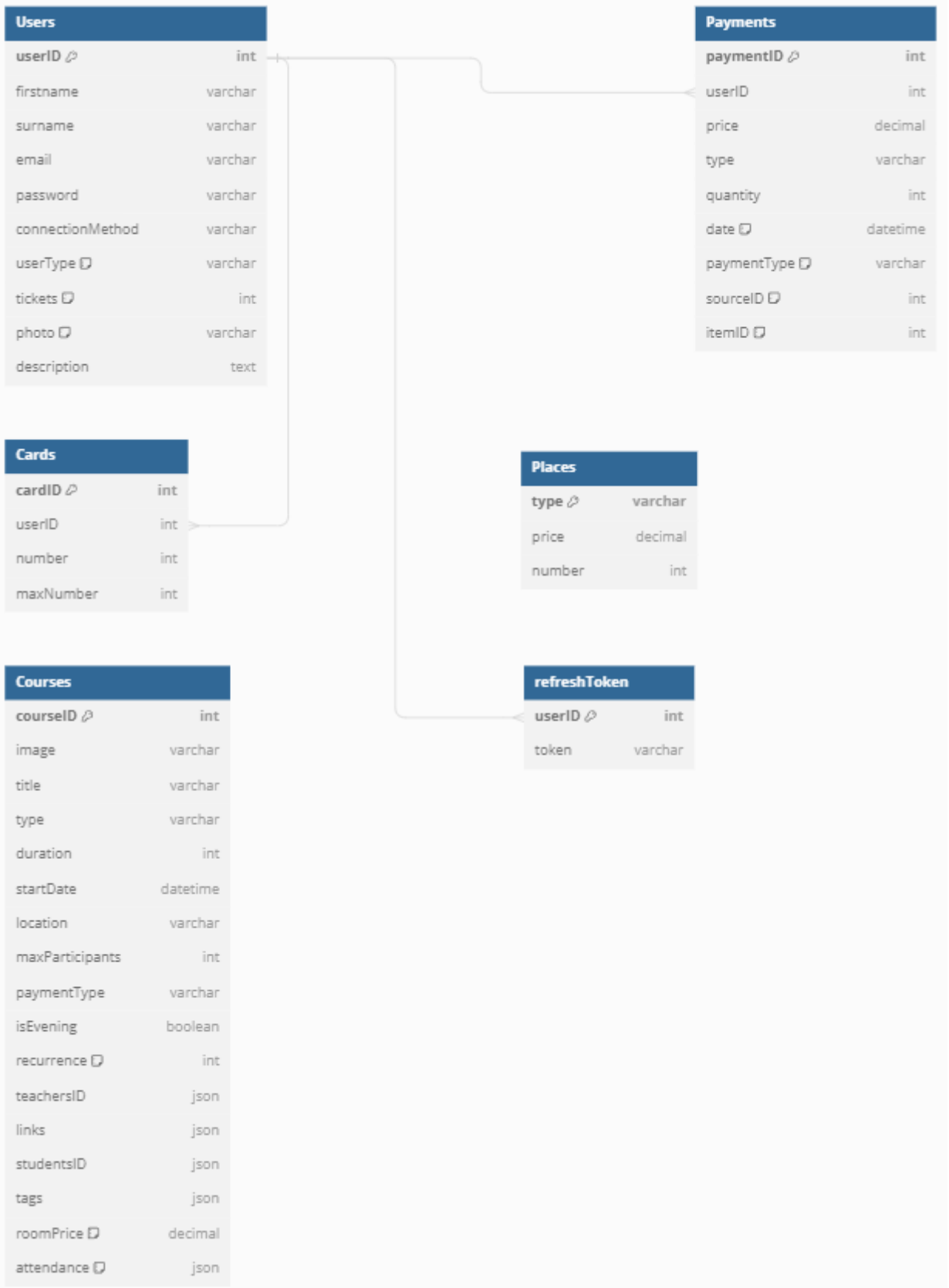
2.3. Analyse fonctionnelle

2.3.1. Architecture technique

https://miro.com/app/board/uXjVKCmCc_0=/



2.3.2. Modélisation de la data



2.4. Réalisation

Pour le frontend, nous avons choisi React. Cette bibliothèque a été sélectionnée pour sa flexibilité, sa performance et sa capacité à créer des interfaces utilisateur dynamiques et réactives. L'écosystème riche de React, avec de nombreux outils et une communauté active, a également été un facteur décisif. Nous avons utilisé des composants React pour structurer l'application de manière modulaire et réutilisable, permettant une gestion efficace des fonctionnalités principales telles que la gestion des cours, des élèves et des professeurs.

Pour le backend, nous avons opté pour Node.js. Node.js permet de construire des applications rapides et scalables grâce à son modèle non-bloquant et à sa gestion efficace des requêtes simultanées.

Une base de données mySQL a été choisie pour sa fiabilité, sa capacité à gérer des transactions complexes et ses fonctionnalités robustes de requêtage. Nous avons configuré la base de données pour stocker les informations sur les cours, les utilisateurs, les paiements, etc.

Pour documenter l'API, nous avons utilisé Swagger. Cet outil a facilité la communication et l'intégration avec d'autres systèmes. Pour les tests end-to-end, nous avons employé Selenium, assurant que l'application fonctionne correctement dans différents scénarios utilisateurs. Nous avons intégré Swagger pour générer automatiquement la documentation de l'API et utilisé Selenium pour automatiser les tests de l'interface utilisateur.

Voici un extrait de la page App.js ou sont importé et où l'on définit nos routes :

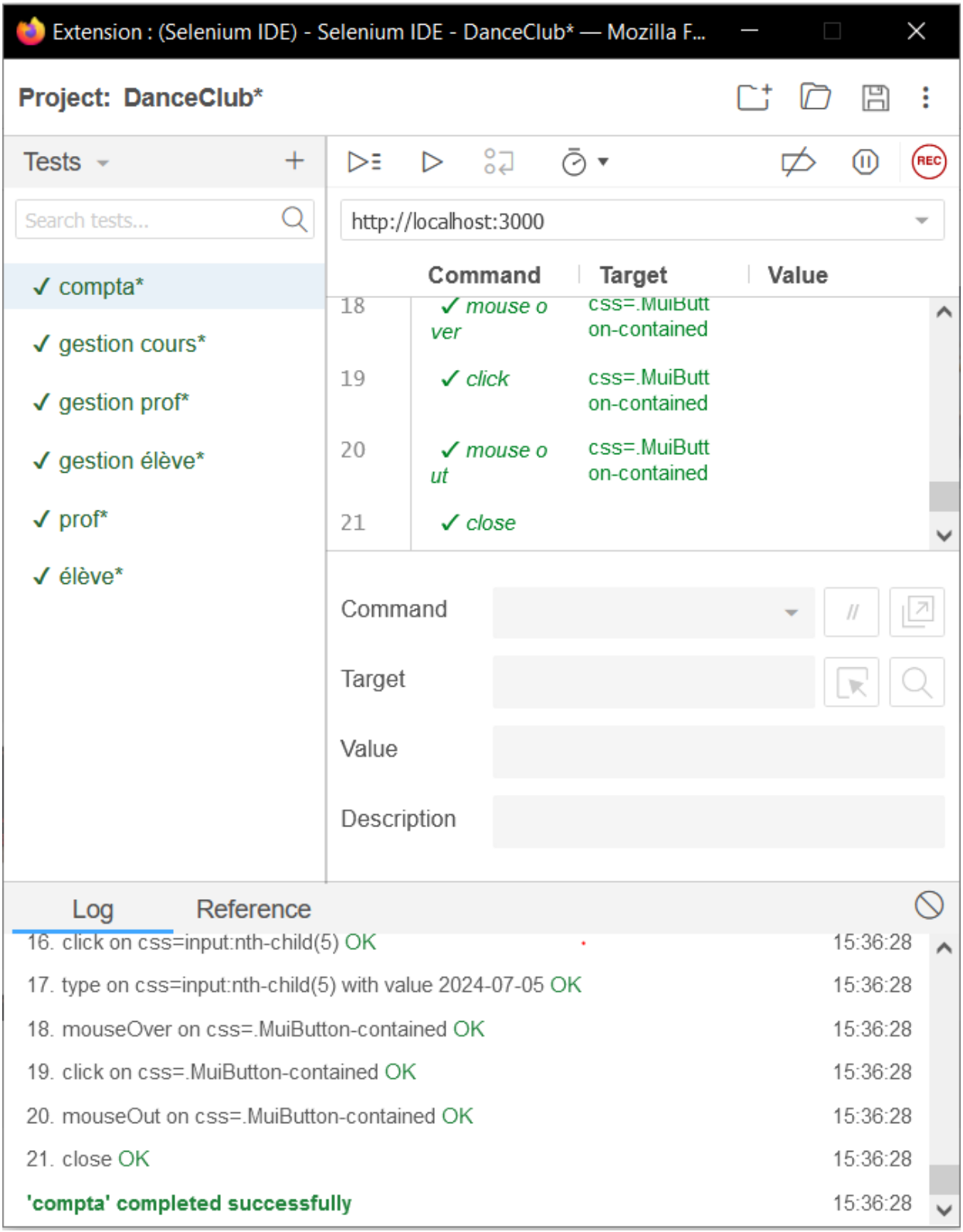
```
import AllContacts from './PAGES/prof/allContacts';
import InfoClassiqueDance from './PAGES/blog/infoClassiqueDance';
import MdpForget from './PAGES/mdpForget.jsx';
import
        (alias) const Revenu: () => React.JSX.Element
        import Revenu
                                                    niqueProf.jsx';
import Revenu from './PAGES/admin/Compta/revenu.jsx';
import ProfilStudentFromProf from './PAGES/prof/profilStudentFromProf.jsx';

function App() {
  return (
    <div className="App">
      <Helmet>
        <title>Dance Club</title>
        <link rel="icon" href={logo} />
      </Helmet>
      <Router>
        <Routes>
          <Route path="/" element={<Home />} />
          <Route path="/profil" element={<Profil />} />
          <Route path="/rgpd" element={<Rgpd />} />
          <Route path="/connexion/" element={<ConnexionEtInscriptionSlider />} />

          <Route path="/admin/prof" element={<AdminProf />} />
          <Route path="/admin/cours" element={<AdminCours />} />
          <Route path="/admin/cours" element={<AdminCours />} />
          <Route path="/admin/abo" element={<CreateCardPage />} />
```

Et voici un lien vers le swagger avec nos routes API :

http://90.110.227.143/api-docs/

http://localhost:3000/api-docs/

2.5. Tests


Extension : (Selenium IDE) - Selenium IDE - DanceClub* — Mozilla F...
Project: DanceClub*
Tests
Search tests...
✓ compta*
✓ gestion cours*
✓ gestion prof*
✓ gestion élève*
✓ prof*
✓ élève*
http://localhost:3000
Command
Target
Value
18 ✓ mouse over css=.MuiButton-contained
19 ✓ click css=.MuiButton-contained
20 ✓ mouse out css=.MuiButton-contained
21 ✓ close
Command
Target
Value
Description
Log
Reference
16. click on css=input:nth-child(5) OK 15:36:28
17. type on css=input:nth-child(5) with value 2024-07-05 OK 15:36:28
18. mouseOver on css=.MuiButton-contained OK 15:36:28
19. click on css=.MuiButton-contained OK 15:36:28
20. mouseOut on css=.MuiButton-contained OK 15:36:28
21. close OK 15:36:28
'compta' completed successfully 15:36:28

2.6. Documentation

Nous avons mis à disposition un guide d'utilisation.

Un Swagger a également été mis en place afin d'avoir une documentation des requêtes API accessible sur le serveur :

http://90.110.227.143/api-docs/

http://localhost:3000/api-docs/

3. Résultats obtenus

3.1. Difficultés rencontrées et solutions

Dans le cadre de la gestion de projet, diviser les équipes en différentes spécialités, comme front-end et back-end, est courant pour optimiser les compétences et la productivité. Cependant, cette séparation peut entraîner des problèmes de dépendance et de coordination. L'équipe front-end peut se retrouver bloquée en attendant que l'équipe back-end développe et expose des API nécessaires. Inversement, l'équipe back-end peut attendre des retours du front-end pour ajuster ses développements. Les deux équipes peuvent également avoir des visions différentes de la finalité du produit, entraînant des incohérences et des malentendus. De plus, la communication entre les équipes peut être insuffisante ou inefficace, exacerbant les problèmes de dépendance et de synchronisation. Les plannings de développement peuvent ne pas être parfaitement alignés, causant des retards.

Pour pallier ces problèmes, plusieurs solutions peuvent être mises en place pour améliorer l'autonomie des équipes tout en favorisant une meilleure collaboration. L'équipe front-end peut utiliser des mockups ou des simulateurs d'API pour continuer à développer et tester leur interface indépendamment du développement back-end. Cela permet de valider l'interface utilisateur et les interactions sans attendre l'API finale. De leur côté, l'équipe back-end peut utiliser des outils de test pour simuler les appels front-end et tester leurs API de manière indépendante.

Organiser des réunions régulières de synchronisation, telles que des réunions quotidiennes ou hebdomadaires, permet d'aligner les priorités et les avancées de chaque équipe. Utiliser des outils de collaboration comme Trello facilite une communication asynchrone efficace et une meilleure gestion des tâches. Créer et maintenir une documentation détaillée et partagée des API et des interfaces permet aux équipes de travailler en connaissance de cause et de réduire les malentendus.

Encourager le prototypage rapide et les feedbacks fréquents entre les équipes permet d'ajuster les développements en cours de route et d'éviter les retours coûteux en fin de cycle de développement. En mettant en place ces solutions, votre équipe pourra mieux gérer les interdépendances entre les développements front-end et back-end, tout en optimisant la collaboration et l'efficacité globale du projet.

3.2. Respect des délais

Les délais ont toujours été respectés. Cependant, ils ont été serrés à la fin du projet lorsque le client est intervenu avec quelques demandes supplémentaires.

3.3. Mise en production

Pour le projet que nous avons réalisé, nous ne pensons pas qu'il sera utilisé dans son ensemble. Cependant, nous espérons que certaines fonctionnalités développées seront conservées et intégrées dans d'autres projets futurs. Ces éléments spécifiques pourraient apporter une réelle valeur ajoutée et répondre à des besoins identifiés au cours de ce projet.

4. Conclusion

4.1. Montée en compétence de l’équipe

Charles : Personnellement, ce projet m’a permis de découvrir une nouvelle méthode de travail à savoir séparer l’équipe Back-end et l’équipe Front-end, c’est une méthode intéressante qui a ses limites. J’estime avoir amélioré mes compétences, comme à chaque projet qui est chaque fois un peu plus technique que le précédent. Enfin, j’estime pouvoir mettre en pratique certaines connaissances acquises en compétence relationnelle.

Eliott : D'un point de vue individuel, j'ai appris à me concentrer plus longtemps et à résoudre mes problèmes tout seul, sur le travail en équipe, j'ai réalisé que c'était sympa aussi, mais on pouvait rencontrer aussi des désaccords. Dans ce projet, j'ai amélioré mes connaissances en React et en JavaScript notamment en utilisant des nouvelles bibliothèques. J'ai aussi réalisé l'importance de la communication au sein d'une équipe surtout quand les opinions divergent.

Hugo : J'ai appris à travailler dans un projet plus professionnalisant pour un client, avec des demandes que l'on abordé pas du tout lors de nos projets de CIR1 et CIR2, sur des questions de sécurité, de documentation, ou encore dans l'aspect technique avec un cahier des charges strict. Le travail en collaboration avec tout le monde, la maitrise des moments de crise ou de conflits pour éviter l'impact sur la production.

Adam : J’ai pu apprendre une multitude de compétences très utiles en travaillant sur ce projet. En travaillant sur la partie back-end de ce dernier j’ai pu développer mes compétences techniques : j’ai découvert comment mettre en place des requêtes d’API et j’ai aussi pu améliorer mes compétences en documentation notamment avec le Swagger. De plus, j’ai également pu apprendre à intégrer des ressources externes comme ReCaptcha. Et au niveau du travail en équipe, j’ai pu développer mes capacités de communication tout en réalisant l’importance cruciale de bien partager les informations et d’adopter une bonne organisation pour éviter de rencontrer des problèmes pendant le projet.

Paul : En travaillant sur le Back-End, j'ai développé de nombreuses compétences techniques, telles que la mise en place de serveurs nodeJS, l'automatisation de l'envoi de mails, et l'organisation de la structure des fichiers et du code, entre autres. De plus, le travail en équipe m'a appris l'importance de partager les informations afin que chaque membre soit bien informé, évitant ainsi les malentendus.

4.2. Et si c’était à refaire

Si nous devions recommencer ce projet, nous pensons qu'il aurait été préférable d'avoir une personne dédiée pour faire le lien entre l'équipe front-end et l'équipe back-end. Cette personne aurait pu jouer un rôle crucial en facilitant la communication et en coordonnant les efforts des deux équipes, ce qui aurait amélioré l'avancée globale du projet.

Un tel rôle aurait permis d'identifier rapidement les points de discorde et les blocages potentiels pouvant survenir en séparant un groupe en deux petites équipes spécialisées. En ayant une vue d'ensemble et en harmonisant les priorités des deux équipes, cette personne aurait pu anticiper les besoins de chaque côté et réduire les délais d'attente, assurant ainsi une meilleure synchronisation et une efficacité accrue.