Variance 1: booking

DÉFINITION DU SCOPE

Pour ce projet, il nous a été attribué la variance V1 du sujet qui consiste à créer deux applications: une web et une mobile au travers desquelles les utilisateurs auraient la possibilité d'effectuer des réservations de billets de train.

Acteurs:

- Utilisateurs de l'application
- Banque
- Système fournissant la liste des trains ainsi que les informations relatives au train
- Tout ingénieur voulant ajouter des modifications à la solution (Documentation)

Services externes :

- Liste des trains
  nombre de wagon (TGV / TER), nombre de places

- Banque (Confirmation ou non du paiement)

Fonctionnalités couvertes pour le POC:

- La possibilité offerte à l'utilisateur aussi bien sur l'application web que l'application mobile d'acheter un billet de train
- Nous considérons que la liste des trains disponibles est fournie par un système extérieur. Ces données seront sous forme de "mocks"
- Les places de trains n'auront que les états "payé" ou "non payé"
- Le positionnement dans le train ne sera pas pris en compte
- La répartition des places sera faite de façon aléatoire
- La notion de fidélité et d'avantage n'est pas prise en compte
- Un mail de confirmation sera reçu une fois le paiement validé par la banque
- La banque étant un service externe, on suppose dans un premier temps que toutes les opérations seront acceptées par la banque.
- Nous essaierons de prendre en compte la notion de compte client afin de donner la possibilité de revenir à une transaction en cours

Personas:

Antoine, âgé de 22 ans, est un étudiant en Sciences Informatiques à l'Université Polytech Nice et habitant à Antibes. Originaire de Paris, il rentre régulièrement dans sa ville pour les vacances. Il souhaiterait pouvoir acheter des billets de train depuis chez lui avec son téléphone portable.

Rosalie, âgé de 26 ans, est une ingénieur travaillant chez Atos à Sophia-Antipolis et habitant à Antibes. Originaire de Paris, elle rentre régulièrement dans sa ville voir sa famille. Elle souhaiterait pouvoir acheter des billets de train de n'importe où mais n'a pas beaucoup de temps et à souvent des empêchements de dernière minute

USER STORIES

1 - Création de compte utilisateur

Description
En tant qu'utilisateur (Antoine), je veux me créer un compte afin d'avoir une trace de mes activités et d'avoir une meilleure gestion de mes réservations

Critères d'acceptation
La possibilité de suivre mes opérations au travers d'un compte

Tests d'acceptance
Scénario 1:
Initialisation: J'arrive sur la page dédiée à l'inscription sur l'application
Action: Je remplis le formulaire d'informations puis le valide
Résultat: Je reçois un mail de prise en compte de mon inscription

2 - Consultation de la liste des trains en fonction d'un trajet

Description

En tant qu'utilisateur de train-booking (Antoine), je veux consulter la liste des trains vers une destination dans le but d'effectuer ma réservation

Critères d'acceptation

La possibilité d'obtenir une liste de trains correspondant aux trajets et à l'heure de train approximative recherchée

Tests d'acceptance

Scénario 1:
Initialisation: Je me rends sur l'application mobile ou le site
Action: Je remplis les champs nécessaires à la réservation d'un train (Date, trajet, départ , arrivée , heure)puis je valide les informations remplies
Résultat: J'obtiens la liste des trains correspondants à ma recherche

3 - Visualisation du récapitulatif de ma réservation

Description

En tant qu'utilisateur de train-booking (Antoine), je veux visualiser le récapitulatif de ma réservation de train avant de procéder au paiement

Critères d'acceptation

La possibilité de visualiser le récapitulatif suite à des choix effectués dans l'application

Tests d'acceptance

Scénario 1:
Initialisation: J'ai fini de remplir puis de valider le formulaire. L'application me propose donc une liste de train correspondant aux informations remplies
Action: Je choisis le train que dans lequel je souhaiterais avoir une réservation
Résultat: J'obtiens un récapitulatif de mon choix comportant notamment le numéro de mon train, l'heure et la gare de départ et d'arrivée ainsi que ma place dans le train (numéro de wagon et place assise)

4 - Paiement d'une réservation

Description

En tant qu'utilisateur de train-booking (Antoine), je veux procéder au paiement de mon billet de train

Critères d'acceptation

La possibilité de procéder au paiement par l'intermédiaire de la banque

Tests d'acceptance

Scénario 1:
Initialisation: J'ai visualisé le récapitulatif de mon billet de train
Action: Je remplis les informations relatives à ma carte de crédit et je valide le formulaire
Résultat: J'obtiens la réponse du système pour savoir si la banque a accepté ou rejeté mon paiement

5 - Visualisation de mon billet de train (réservation déjà effectuée)

Description

En tant qu'utilisateur de train-booking (Rosalie), je veux revoir les informations relatives à mon billet de train précédemment acheté

Critères d'acceptation

La possibilité de visualiser les informations relatives à un billet déjà acheté

Tests d'acceptance

Scénario 1:
Initialisation:Je vais sur la page dédiée à la consultation des billets de trains
Action: Je renseigne mon nom ainsi que le numéro de ma réservation dans le formulaire puis je le valide
Résultat: J'obtiens un récapitulatif de mon billet de train

6 - Annulation d'une réservation par un utilisateur

Description

En tant qu'utilisateur de train-booking (Rosalie), je voudrais annuler ma réservation déjà payée.

Critères d'acceptation

La possibilité d'annuler une réservation déjà payée avant l'heure de départ.

Tests d'acceptance

Scénario 1:

Initialisation: Je suis un utilisateur sans compte, j'ai payé mon billet mais je souhaiterais annulé ce dernier avant l'heure de départ

Action: Avec l'application train-booking, je vais dans la page 'voyages' puis je renseigne la référence de mon billet et mon nom puis j'annule mon billet.

Résultat: J'obtiens une notification qui dit que mon annulation est bien faite.

Scénario 2:

Initialisation: Je suis un utilisateur avec un compte, j'ai payé mon billet mais je souhaiterais annulé ce dernier avant l'heure de départ

Action: Avec l'application train-booking, je vais dans la page 'voyages' puis je me connecte après je sélectionne le voyage que je veux annuler puis j'annule mon billet.

Résultat: J'obtiens une notification qui dit que mon annulation est bien faite.

Scénario 3:
Initialisation: Je suis un utilisateur sans compte, j'ai payé mon billet mais je souhaiterais annulé ce dernier avant l'heure de départ
Action: Avec l'application train-booking, je vais dans la page 'voyages' puis je renseigne une mauvaise référence et mon nom.
Résultat: J'obtiens une notification qui dit que mon annulation est annulée car la référence est mauvaise .

7 - Finaliser sur mobile une réservation commencée sur l'application web

(ou vice-versa)

Description

En tant qu'utilisateur de train-booking (Rosalie), je veux pouvoir finaliser, sur mon application mobile, la réservation que j'ai commencé plus tôt sur la page web

Critères d'acceptation

Mise en place d'un système de mise à jour rapide des informations sur un compte utilisateur afin d'avoir la possibilité de continuer sur une autre plateforme

Tests d'acceptance

Scénario 1:
Initialisation: J'ai rempli le formulaire de recherche de train puis je le valide sur l'application web. Je suis contrainte d'aller en réunion mais je souhaiterais quand même finaliser ma réservation
Action: Je me connecte sur mon application mobile afin de la finaliser
Résultat: J'obtiens sur l'application mobile le résultat de ma recherche de train préremplie.

Scénario 2:
Initialisation: J'ai rempli le formulaire de recherche de train mais je n'ai pas eu le temps de le valider sur l'application web. Je suis contrainte d'aller en réunion mais je souhaiterais quand même finaliser ma réservation
Action: Je me connecte sur mon application mobile afin de la finaliser
Résultat: Je remplis de nouveau ce formulaire afin de continuer ma recherche

Scénario 3:
Initialisation: J'ai fini de faire mes choix relatifs au train que je voudrais prendre et obtenu un numéro de réservation sur l'application web. Je suis contrainte d'aller en réunion mais je souhaiterais quand même finaliser ma réservation
Action: Je me connecte sur mon application mobile afin de la finaliser
Résultat: En entrant mon numéro de réservation, je reviens à la page de paiement de la réservation

Diagramme de composants

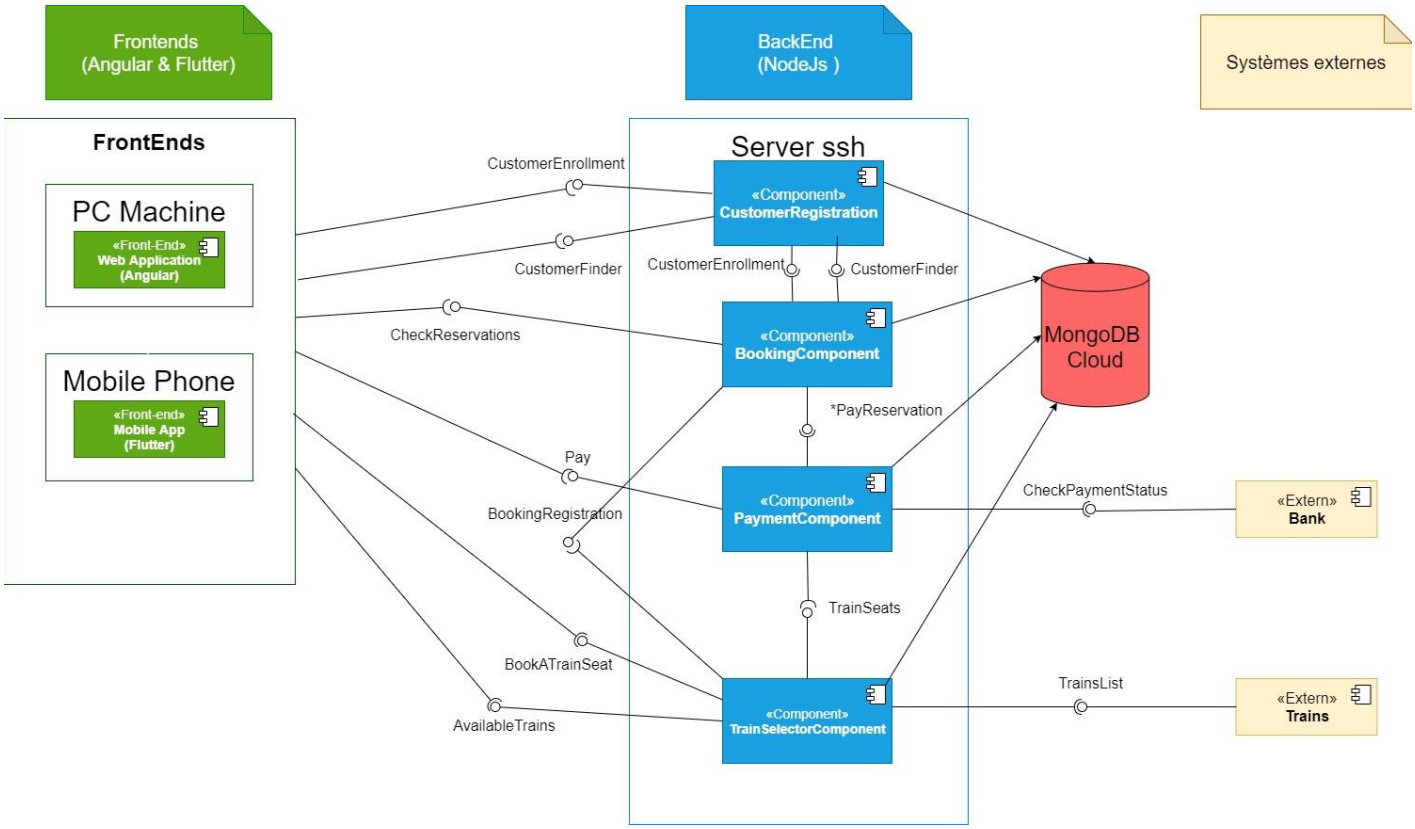

Figure 1 : Architecture des composants du système train-booking

(PS : Image également consultable sur https://github.com/wak-nda/train-booking-al-20-21-team-c/tree/master/deliverables/train-booking-components-diagram.jpg)

Diagramme d'architecture général du système

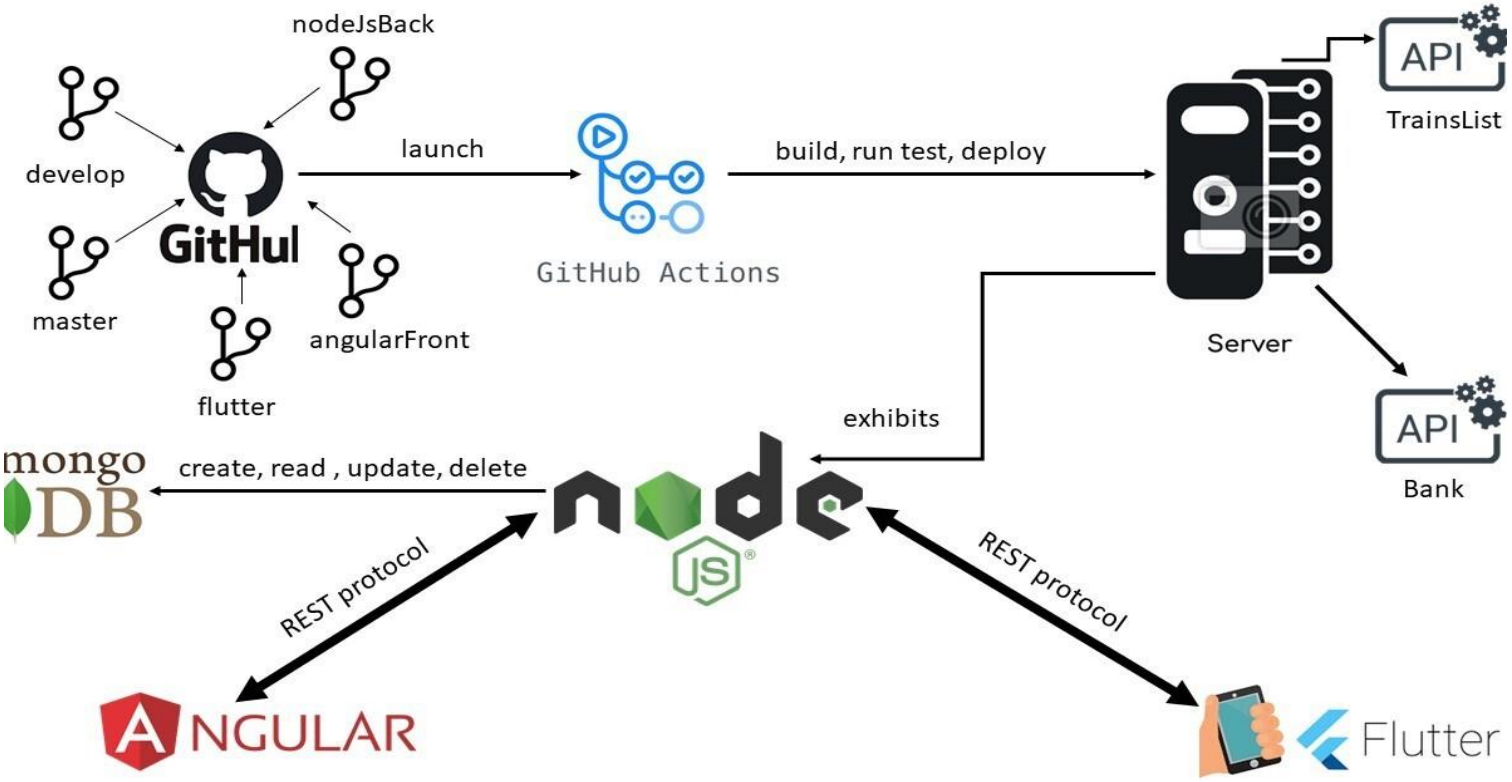

Figure 1 : Architecture général du système train-booking

(PS : Image également consultable sur https://github.com/wak-nda/train-booking-al-20-21-team-c/tree/master/deliverables/architecture.jpg)

CHOIX D'ARCHITECTURE ET DES TECHNOS DU SYSTÈME TRAIN-BOOKING

I. Description du diagramme de composants

Pour notre projet nous implémenterons deux clients.

Web Application : comme son nom l'indique, c'est une application web qui permet à l'utilisateur de réserver un train via un navigateur. Cette application sera implémentée en Angular.

Mobile application : Ce client permet à l'utilisateur de réserver un train à travers une application mobile qui sera implémentée en Flutter.

Nous avons opté pour 2 applications car cela permet de monter la communication, la synchronisation de données entre les 2 applications . Aussi, l'application web offrira des fonctionnalités que l'application mobile ne fera pas comme par exemple le login , voir la liste de toutes ses réservations, modifier les informations personnelles de son compte.

II. Description du BackEnd

1. Interne

Notre backend pour le mvp à rendre bientôt est composé de 4 composants qui sont :

- **CustomerRegistration** : ce composant se charge de la gestion des clients, donc création de compte client ou connection du client permettant à ceux qui veulent enregistrer un client (inscription) ou retrouver un client (connection/recherche) à travers les interfaces 'customerEnrollment' et 'customerFinder'

- **BookingComponent** : ce composant a la responsabilité de gérer les réservations (en ajouter principalement pour le mvp) et de permettre de voir les réservations d'un client donné. Ainsi il offre donc ces possibilités au travers de 2 interfaces 'checkReservations' et 'bookingRegistration'. Ce dernier a quand même besoin de connaître le client si ce dernier est en base de données ou sinon le créer de manière temporaire (je pense) pour associer la réservation à un client. Ainsi lorsque le client voudra regarder ces réservations en renseignant son nom et son email il verra la liste de ces réservations et si ce dernier donne son nom et la référence de sa réservation il aura accès au données spécifiques de sa réservation.

- **PaymentComponent** : ce dernier offre la possibilité au client de payer sa commande une fois le train choisi. Il est possible de faire d'effectuer cette tâche par le biais de l'interface 'Pay' et au moment du paiement le composant demande une vérification au

composant externe qu'est la 'bank' par le biais de l'interface offerte 'CheckPaymentStatus'. Une fois le paiement validé, le composant retire une place libre du train concerné par l'interface 'TrainSeats' offerte le trainSelectorComponent' et ensuite enregistre la réservation grâce l'interface 'bookingRegistration'

- **TrainSelectorComponent**: grâce à 'TrainSeats' , 'BookATrainSeat', 'AvailableTrains', le client peut regarder les trains disponibles et/ou réserver une place pour un train donnée en fonction des données renseignées (départ, destination voire peut être la date) ou aux autres composants de mettre à jour le nombre de place d'un train après un paiement validé. Il utilise l'interface 'TrainsList' du composant 'Trains' pour récupérer la liste de trains puis faire son filtrage pour renvoyer les trains disponibles en fonctions des filtres et aussi modifier le nombre de place libres du train concerné

2. Externe

- **Bank** : La banque nous permettra de valider le paiement des billets de train par un client ainsi que de fournir un historique des différentes transactions. La notion de paiement bien que fondamentale pour l'achèvement de l'opération de réservation, il n'est pas aussi proprement intégré à notre architecture. C'est un service externe qu'on suppose fourni par une banque réelle (ex de payload etc ..) qui nous permettra de conclure les opérations du client.

- **Trains** : Ce service n'est pas interne à notre architecture. Il s'agira d'une API distante REST qui nous permettra de récupérer divers informations qui sont au coeur de notre système de réservation de billets:
    - ❖ la liste des trains disponibles en fonction de plusieurs paramètres (horaires, journée, lieu, type de train TER/TGV , etc..)
    - ❖ les sièges disponibles
    - ❖ les classes de wagons etc

Ces 2 services externes seront fait avec une logique de code très simple.

3. Base de données

Pour la base de données, nous avons choisi une seule base de données dans laquelle chaque composant n'a accès qu'à sa table et en est le responsable pour sa gestion. Dans le temps imparti puisque nous avons opté pour une architecture monolithique, nous n'avons pas besoin notre application avec une base de données par composant. Quant aux services externes comme nous ne sommes pas chargé de les implémenter, nous avons choisi de faire des mocks.

Notre backend tournera sur un serveur en ligne, et donc nos frontends sur des machines différentes et notre base de données dans un cloud. Cette configuration nous permet de ne pas tout lancer en local et nous permets donc non seulement de lancer les frontends sur

différentes machines vue que le serveur est en ligne et puisque la base de données est sur le cloud alors le serveur peut y avoir accès grâce à l'adresse et les logs pour s'y connecter.

III. Justification du choix d'architecture et techniques

Application	Technologie	Critères de Choix
Front-end Web	Angular	● Angular profite de la **rigueur et flexibilité** du langage TypeScript ● Notre application sera très orienté composants notamment pour visualiser une liste de trains (composant unique répété plusieurs fois). Un cas plus complexe est la page de réservation de billets qui va être essentiellement composé d'un agencement de plusieurs composants et fait en monopage (angular nous permet de profiter d'un système de refresh des composants sans recharger toute la page, ce qui nous sera indispensable car on devra appliquer divers filtres (fonction de la date, de l'heure, départ, arrivée, escales ou pas, type de train, classe de voyage).
Front-end Mobile	Flutter	● Flutter est multi-plateforme (Android et IOS) ce qui permet tout d'abord de simplifier la tâche au développeur en n'ayant qu'un seul code à maintenir pour plusieurs plateformes. En effet notre application devra permettre à quiconque de réserver un billet de train sur une plateforme IOS ou Android. En outre, le rendu des interfaces est exactement le même sur les deux plateformes, ce qui évite au développeur de traiter des particularités propres à chaque langage. ● En raison du temps de développement assez court, nous avons préféré flutter car c'est une technologie qui demande moins de code. Étant à la version de mvp de l'application, on pourrait être amené à revoir l'interface plus tard ● Notre architecture est basée sur des composants et chaque composant est responsable de d'une logique métier. Ceci nous permet d'éviter le couplage et de responsabiliser les fonctionnalités. Ce mécanisme interne du système peut être représenté pareillement sur le front-end car flutter est également architecturé en composants et ces composants peuvent s'emboîter avec d'autres. ● Un autre bénéfice pour le développeur est la documentation du langage. Flutter dispose d'une documentation sur chaque composant de son sdk et dispose d'une communauté très active.

Back-end	Node JS	● Langage utilisé: JavaScript et est donc très rapide: pour une application qui doit faire des opérations de recherche sur éventuellement une très grosse liste de train, nodejs semble adapté ● La scalabilité: pour une application de vente de billet, il sera possible que des milliers de requêtes soient exécutées de façon simultanée. ● La modularité de ce langage nous permettra de nous assurer d'une réelle indépendance entre composants notamment entre le **TrainSelectorComponent et le PaymentComponent** si on souhaite augmenter les ressources au niveau d'un seul module ou composant
Base de Donnée	MongoDB	● Si on a choisi une base de données NoSql dans un premier temps , c'était pour tirer partie de la flexibilité car la structure de la plupart de nos modèles vont très vite évoluer notamment par rapport à la gestion des places de trains (Éventuellement la possibilité de stocker des tableaux plus tard)
Documentation	Swagger	-Documentation claire et générée automatiquement et testable

Auto-évaluation du travail réalisé

Avec une définition concise et précise de notre SCOPE, nous avons très bien démarré notre projet. La rédaction de nos US qui en découlent s'est faite aussi facilement. Nous étions donc sur les bons rails lors des 2 premières semaines du projet. Nous avons pêché ensuite dans la représentation de notre schéma d'archi (diagramme de composants) et dans la justification de nos choix technologiques ce qui a entraîné un léger retard pour la suite.

Ce contretemps a été rattrapé dès la première semaine où nous avons commencé l'implémentation de notre solution. Avec une bonne répartition des tâches :

(djotiham) Membre le plus à l'aise sur la techno Mobile Front se charge entièrement de l'app mobile
(paul) Membre le plus à l'aise sur la techno Web Front se charge entièrement de la plateforme Web
(florian & paul-marie) Les 2 autres membres se concentrent sur le back-End, les services externes ainsi que la dockerisation des test
La CI (github Action et le déploiement de nos services externes + backEnd) a été réalisée en parallèle par paul qui s'y connaissait déjà.

Nous avons avancé alors assez vite sur tous les plans (mobile, web, backend , CI) ce qui nous a permis d'avoir très tôt un scénario MVP qui partait du front web et se terminait dans le front mobile. Nous avons continué au fil des semaines à étoffer les justifications de nos choix technologiques ainsi que de notre architecture choisie. Chaque semaine, on faisait un point au début et à la fin pour se fixer les nouveaux objectifs et contrôler l'avancée sur toutes les parties du code.

Nous avons réalisé toutes les fonctionnalités prévues pour ce POC. Néanmoins, si le backEnd a été testé entièrement (unitaire + cucumber), nous avons négligé les tests côté FrontEnd que nous n'avons finalement pas réalisés et que nous avons prévu pour la suite du projet.

Chaque membre du groupe s'est sérieusement impliqué dans ce projet. De ce fait, nous nous répartissons les 400 points de la façon suivante :

Florian AINADOU : 100 points

Djotiham NABAGOU : 100 points

Paul KOFFI : 100 points

Paul-Marie DJEKINNOU : 100 points