Scope du projet - sujet V2

Groupe E :

- Rania Fekih
- Anis Khalili
- Benjamin Vouillon
- Robin Lambert

Les utilisateurs

Avant de définir le scope de ce projet, il convient d'identifier les utilisateurs. Voici ceux qui vont utiliser notre solution :

- **Le contrôleur** : Celui qui contrôle les billets dans le train et applique les sanctions aux fraudeurs.
- **L'analyste** : Celui qui consulte et analyse les données collectées par les contrôleurs à travers un dashboard.

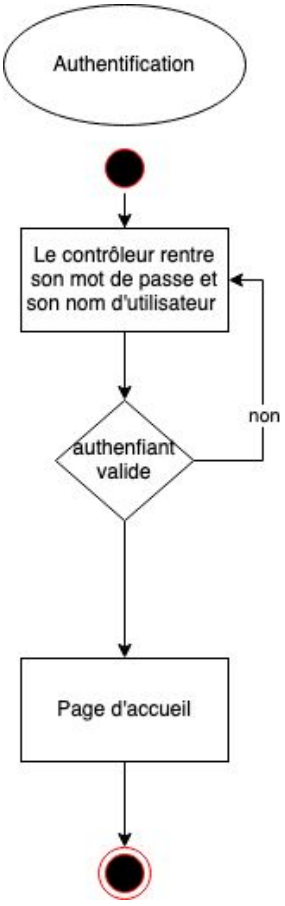
Description du projet :

Dans un premier temps le projet du sujet V2 est de réaliser une solution permettant de contrôler le billet d'un passager dans un train (Ce billet est en fait un QR Code et un numéro qui a été remis au client à l'achat du billet.) et dans le cas où le client ne possède pas de billet valide alors il faut grâce au logiciel remonter une fraude au système. Pour cela voilà les différentes fonctionnalités de ce logiciel :

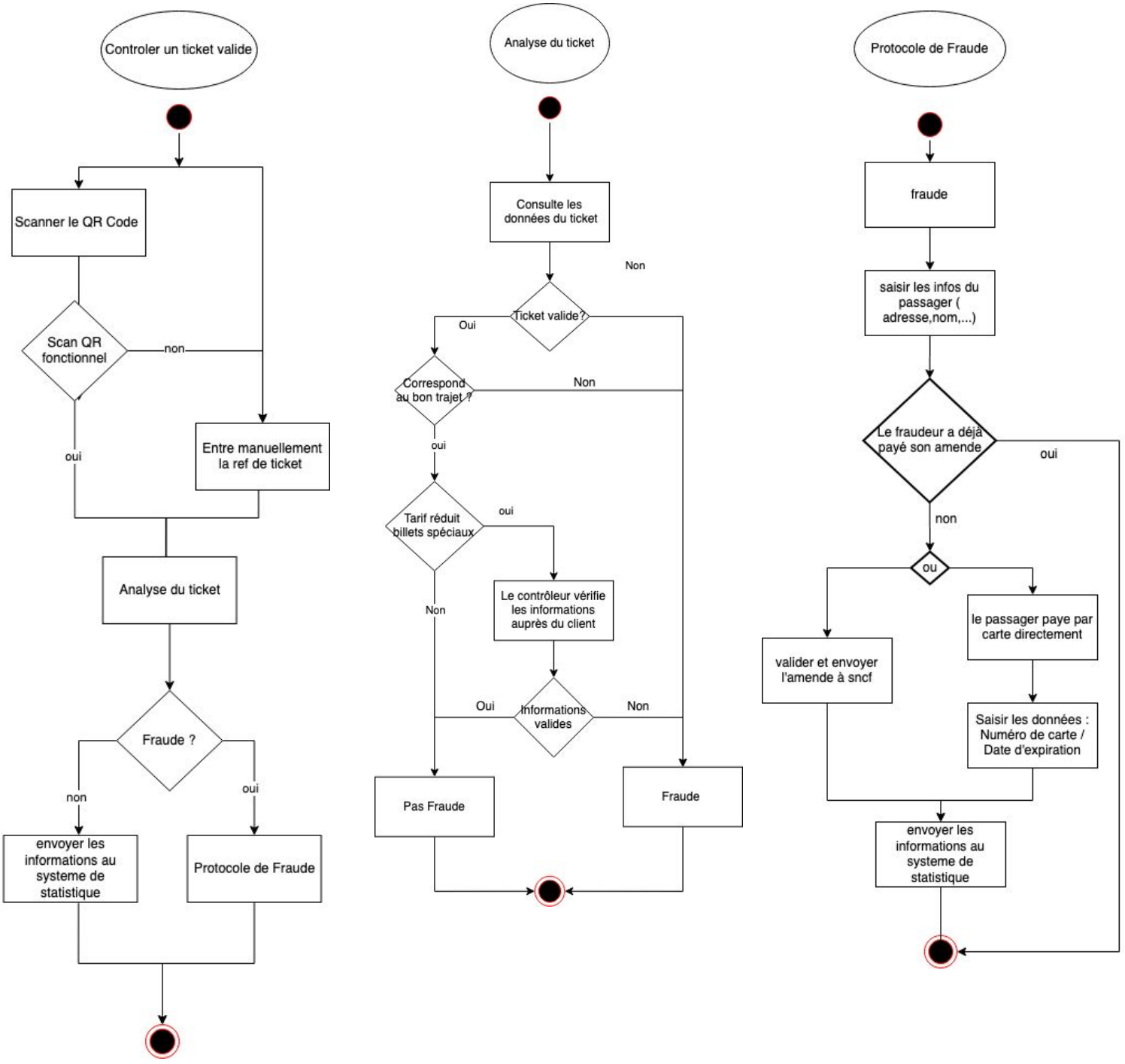
- Il faut pouvoir contrôler les billets en scannant le QR code, mais si le QRCode n'arrive pas à être lu le contrôleur doit pouvoir rentrer le numéro client à la main.
- Le logiciel permet d'ouvrir un dossier de fraude dans le cas où l'utilisateur le client n'a pas de billet ou son billet n'est pas valide.
- Le logiciel doit renvoyer les données de billet en cas de tarif réduit pour que le contrôleur s'assure que la réduction appliquée sur le ticket est bien applicable.
- Ce logiciel doit prendre en charge le paiement par carte bancaire et les paiements en espèces. Dans le cas où le client ne peut ou ne veut pas payer son amende directement il doit y avoir un système de demande de paiement qui s'enclenche.
- Les informations de validité des billets et le nombre de fraudes doivent êtres stockées sur un serveur pour qu'il puisse les analyser.
- Le fait de vérifier le ticket se fait depuis un module qui appelle des services externe. Il appelle un service du projet v1 "booking" qui renvoie les informations qui correspondent au ticket (tarif réduit ou non, nom du passager, trajet, ...) et un service qui donne des informations sur le train (position en direct, trajet effectué,

...). Si le billet est un faux, ou bien que le trajet n'est pas le bon, le contrôleur est informé et peut lancer le protocole de fraude.

- Le protocole de fraude est géré par un service à part qui se connecte à un module externe de paiement et à un autre qui permet de tarifer les fraudes (dépend de l'infraction : tarif de réduction non applicable, billet pour une portion du trajet seulement, pas le bon nom de voyageur, pas de billet, etc.)



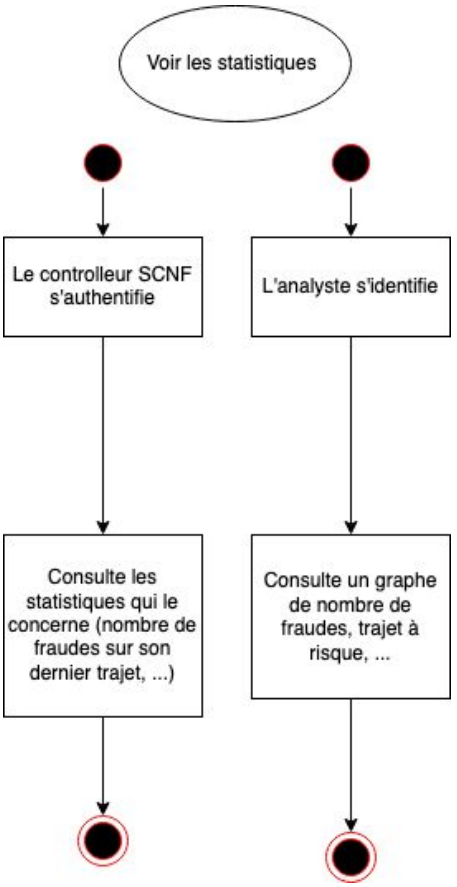
Scénario d'authentification du contrôleur



Scénarios de contrôle du ticket, de l'analyse du ticket et de l'application de la sanction pour les fraudeurs.

Dans un second temps une autre solution doit-être mise à disposition des analystes de la compagnie de train et de permettre aux contrôleurs de consulter leurs statistiques de travail. Voilà les fonctions à réaliser:

- Le contrôleur a la possibilité de consulter différentes statistiques réalisées à partir de ses propres contrôles :
  - le nombre de voyageurs contrôlés
  - le nombre d'amendes rédigées
  - les nombres de trajets effectués
- L'analyste a accès à une api / dashboard pour voir toutes les statistiques des trajets :
  - le nombre des trains contrôlés
  - le nombre de fraudes par train
  - les trajets et les horaires où il y a un nombre important de fraudes



Scénarios liés aux statistiques, pour le contrôleur et l'analyste.

Ordre de réalisation

L'objectif pour la POC de début Novembre est de se focaliser sur le scénario le plus important de l'application : le contrôle de billets par le contrôleur. Il pourra également signaler une fraude (gestion des différents cas de fraude décrits dans les scénarios). Le module de statistiques pourra être utilisé par l'analyste : en revanche, il n'affiche que les informations qui sont stockées, sans calcul ni analyse de données.

Un scénario qui n'est pas prioritaire est la possibilité pour le contrôleur de voir les statistiques qui le concernent.

Diagramme de composants

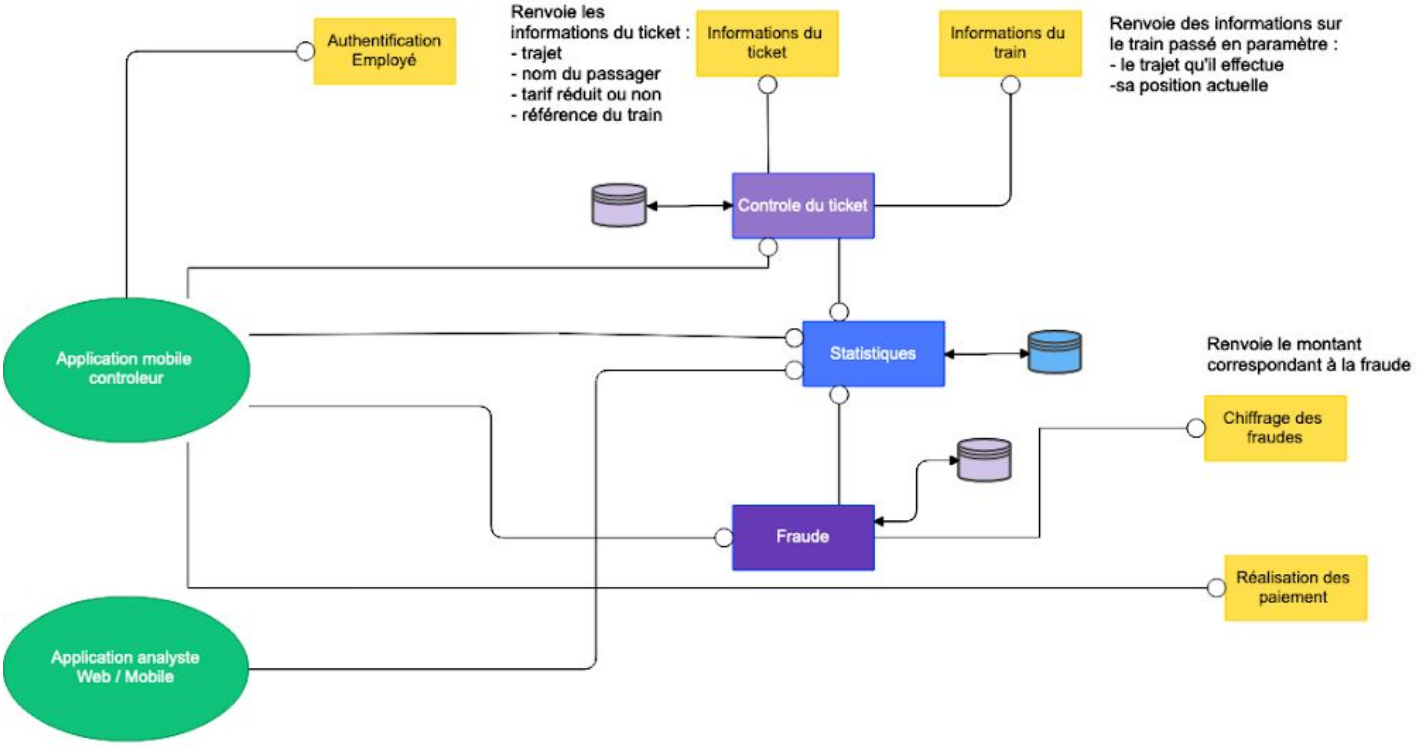
L'avantage en terme d'architecture de séparer la gestion des fraudes du contrôle du ticket est de découpler ces deux actions métiers : le contrôleur peut sanctionner quelqu'un qui n'a pas son billet, et si l'un des deux services ne répond pas, ce n'est pas bloquant pour l'application (pas de single point of failure). Le module de statistique peut être partagé entre les deux applications, mais ne renverra pas les mêmes informations.

- Authentification employé : il permet au contrôleur de se connecter à son profil pour lui donner accès à l'application et aux statistiques de contrôle qui le concerne, et permet au module de statistique d'identifier qui effectue le contrôle.
- Informations du ticket renvoie le trajet, le nom du passager, le tarif réduit, la référence du train, ...
- Informations du train : renvoie le trajet effectué
- Contrôle du ticket : vérifie les données du ticket pour renvoyer à la fin si le ticket est valide ou non.
- Statistiques : stocke et affiche les informations statistiques aux analystes
- Fraude : génère et affiche les réclamations de fraudes, renvoie les tarifs et communique avec le module paiement
- Chiffrages de fraude : Calcule le montant de fraude
- Paiement : Un service externe de banque

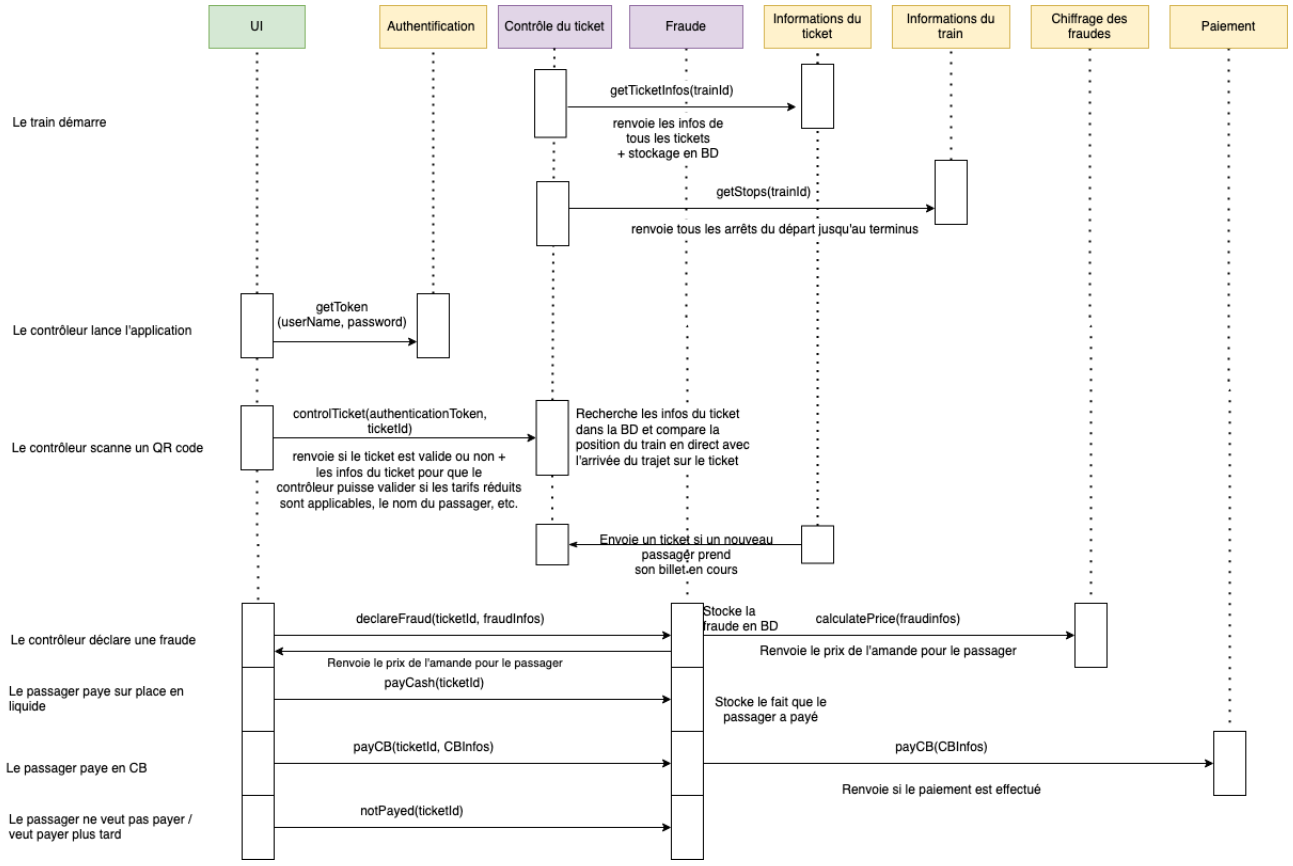
L'utilisateur de l'**app mobile** s'authentifie en communiquant avec **Authentification employé**.

L'utilisateur scanne le QR code, l'**app mobile** va envoyer les informations récupérés au **control ticket**.
Le composant **control ticket** va communiquer avec **informations ticket** et **informations train** pour récupérer tous les données et les comparer.
Le **control ticket** va retourner à l'**app mobile** si le ticket est valide, si c'est un tarif réduit à vérifier ou bien c'est si c'est une fraude (faux ticket, pas le bon train, trajet payé trop court si la personne est restée dans le train après la station de destination de son ticket).

En cas de fraude l'**application mobile** va communiquer avec le composant **fraude** pour remplir les données nécessaires qui va communiquer à son tour avec **chiffrages de fraude** pour donner le montant à payer.
L'**app mobile** va renvoyer au **paiement** le montant récupéré et les données du paiement (cash / carte bancaire) pour finaliser la procédure



Sur ce schéma d'architecture, les composants en violet sont les composants qui vont être physiquement présents dans les trains (1 par train). Cela nous permet de nous assurer que ces services sont joignables partout dans le train depuis l'application mobile pour les contrôleurs (par réseau wifi interne avec des répéteurs pour chaque wagon, par exemple).
Le service de statistique (en bleu) est interne à notre système mais n'est pas embarqué : il est commun à tous les trains, et ce n'est pas critique si il n'est pas disponible instantanément depuis l'application mobile.
Les composants en jaune sont les services externes, qui ne font pas partie du scope de notre projet et que nous allons implémenter pour simuler l'intéraction avec d'autres services indspensables.



Choix de technologies :

- **Front - end :** Android
  Après une recherche profonde nous avons trouvé que la plupart des applications dédiés à ce genre d'utilisation sont implémentés avec Android vu que sur le marché, il existe une large gamme de périphériques matériels alimentés par le système d'exploitation Android, y compris de nombreux téléphones et tablettes différents. Même le développement d'applications mobiles Android peut se produire sous Windows, Mac OS ou Linux.

- **Back-end :** NodeJs
  Ce choix a été fait parce que ce framework possède différents avantages qui le rende avantageux dans certains cas d'utilisation comme le nôtre:Grande disponibilité de modules faciles à installer grâce au gestionnaire NPM (Node Package Manager) spécifiquement dédié à Node.js. Les modules nous permettent de s'appuyer sur l'existant et ajouter de la valeur client.
    - Utilisation de Node en tant que serveur web le rend par ailleurs capable de traiter un gros volume de requêtes simultanément de manière efficace grâce à sa conception asynchrone. On pourra satisfaire l'ensemble des requêtes provenant des nombreux utilisateurs simultanés.

- Déploiement fortement facilité par le moteur V8 dont les sources en C peuvent être compilées pour tous types de système d'exploitation

- **BD :** MangoDB
  L'architecture évolutive horizontale de MongoDB peut prendre en charge d'énormes volumes de données et de trafic et c'est ça ce qu'on a besoin dans notre application . On aura un grand flux de tickets à controlés dans SNCF il y a 10 000 contrôleurs et chacun va contrôler minimum 300 tickets par jour donc là on parle d'un énorme flux de données.

Risks :

- Un nombre énorme de requêtes qui peut entraîner un surcharge de serveur
- Un réseau instable qui peut perturber la récupération des données.
- Le composant paiement est un composant sensible aux attaques sécurité

Roadmap

Voici un schéma qui présente ce que l'on a prévu de faire pour ces prochaines semaines :