💡 Un système de gestion de la billetterie d'un réseau ferroviaire

Auteurs (trice) : Iloniavo RANDRIAMANGA - Van Trang DANG - William PLAYERS

1. Contexte, Glossaire

1.1. Contexte général du projet

Le présent projet s'inscrit dans la conception d'un système de billetterie numérique permettant la gestion complète de titres de transport pour un réseau ferroviaire simplifié.

Dans un contexte où la dématérialisation des services publiques et la sécurisation des intéractions client - systèmes deviennent des enjeux majeurs, il est nécessaire de disposer d'un service fiable, cohérent et vérifiable permettant:

```
- la recherche, l'achat et l'émission de billets électroniques
- la gestion d'un réseau fixe de services de transport
- la vérification locale des billets électroniques par une unité de contrôle
- l'identification d'un client via les informations obtenues via le code
- la réduction des risques de fraude et de duplications abusives
```

Le système vise principalement un environnement pédagogique et expérimental, mais doit refléter les contraintes essentielles d’un système réel : cohérence fonctionnelle, intégrité des données, traçabilité des actions et robustesse face aux usages courants.

Les **utilisateurs cibles** sont : les clients achetant des billets, les agents de contrôle vérifiant la validité des titres de transport, l’administrateur système, responsable de la configuration statique du réseau, des services et de la base client.

1.2. Contexte métier

Le projet vise à développer un système simple de billetterie numérique pour un réseau ferroviaire fixe comprenant au moins dix villes, un ensemble de services planifiés et un groupe de clients enregistré. Le système permet la recherche, l’achat et l’émission de billets électroniques associés à un code optique, ainsi que leur vérification par une unité de contrôle utilisée par les agents à bord.

Le projet est réalisé dans un cadre pédagogique et vise à reproduire les exigences essentielles d’un système réel : cohérence fonctionnelle, vérifiabilité des billets et réduction des risques de fraude.

1.3. Vocabulaire spécifique (Glossaire métier)

- **Billet** : Titre de transport électronique émis pour un client donné et associé à un service de transport spécifique (train, date, heure, trajet).
- **Billet valide** : Billet dont les conditions métier sont remplies (paiement effectué ou simulé, service existant, date et heure dans le créneau de validité, non expiré).

- **Billet validé** : Billet valide pour lequel une validation a été enregistrée par le système (après contrôle par une unité de contrôle et confirmation par le serveur central).
- **Service de transport** : Instance de trajet planifiée correspondant à un train donné à une date et une heure précises, reliant un point A à un point B.
- **Trajet** : Itinéraire entre une ville de départ et une ville d'arrivée à l'intérieur du réseau de transport (peut être composé d'un ou plusieurs services, selon les choix de conception).
- **Réseau de transport** : Ensemble fixe de villes et de liaisons ferroviaires définies dans le système.
- **Contrôleur (ou unité de contrôle)** : Agent (et/ou application) chargé de vérifier la validité des billets présentés par les clients.
- **Code optique (code QR)** : Représentation graphique (par exemple un code QR) permettant d'encoder un identifiant de billet, lisible par un terminal de contrôle. Le code optique ne doit pas contenir directement de données personnelles.
- **Serveur central** : Composant applicatif principal hébergeant la logique métier, la base de données et l'API exposée aux clients (interface web, unité de contrôle, etc.).
- **Mode dégradé** : Mode de fonctionnement de l'unité de contrôle en l'absence de connexion réseau, limité au contrôle local des billets à partir des données en cache, sans modification de l'état global sur le serveur central.
- **Contrôle local** : Vérification effectuée par l'unité de contrôle à partir des données disponibles localement (cache de billets).
- **Validation globale** : Décision finale de validation d'un billet, enregistrée sur le serveur central.
- **Cache local** : Ensemble de données stockées temporairement sur l'unité de contrôle (par exemple, les billets d'une journée donnée) pour permettre un contrôle local en cas de perte de connexion réseau.
- **Journal de contrôle** : Historique des contrôles effectués par une unité de contrôle, comprenant au minimum l'identifiant du billet, la date et l'heure du contrôle, le terminal utilisé et le résultat du contrôle (positif ou négatif).
- **Créneau de validité** : Intervalle de temps pendant lequel un billet est considéré comme utilisable pour un service donné .
- **API REST** : Interface de programmtion permettant l'échange de données entre les clients (Mobile, Unité de contrôle) et le serveur via le protocole HTTP et le format JSON.
- **Idempotence** : Propriété garantissant qu'une opération peut être répétée plusieurs fois sans changer le résultat au-delà de la première application, éviter les erreurs lorsde synchronisations multiples.

---

2. Objectifs

2.1 Objectifs fonctionnels

L'objectif est de gérer le cycle de vie dématérialisé du billet : **Recherche → Achat → Émission → Validation → Expiration.**

Acteur	Capacités du système
Voyageur	Création de compte, recherche A/B, achat, stockage, consultation et affichage QR, notifications.
Contrôleur	Scan QR, vérification (Online/Offline) et identification des statuts.
Système	Gestion (User/Billet/Trajet), unicité des titres, arbitrage centralisé et anti-fraude.

2.2 Objectifs non fonctionnels

Catégorie	Objectifs non fonctionnels
Sécurité	Intégrité des billets, prévention de la falsification, protection des données personnelles, authentification fiable
Fiabilité	Fonctionnement cohérent en cas de réseau instable, synchronisation correcte, absence d'incohérences
Simplicité d'utilisation	Interface intuitive, actions minimales pour l'achat et le contrôle, expérience fluide
Performance	Temps de réponse rapide, gestion d'un nombre raisonnable de connexions simultanées
Maintenabilité & évolutivité	Usage de technologies standards et open-source, possibilités d'évolution du système (paiements réels, intégrations futures)

3. Contraintes

3.1. Contraintes techniques

Le système devra reposer sur **une architecture client–serveur** : C'est une architecture classique pour les services qui disposent d'interactions avec des données. Le client envoie une requête au serveur qui est éventuellement traitée.

Dans notre cas, les exemples de telles interactions sont la créations du compte, la validation des billets ou l'achat d'un ticket en ligne.

Le backend devra être implémenté sous forme de service **web (API REST)** : Une contrainte nécessaire est de faire les services fonctionner et de les faire communiquer entre eux. Pour cela, on implementera une API pour nos services.

Les données devront être stockées dans une **base de données relationnelle** : Comme les données doivent être stockées quelque part, on va implémenter une base de données qui sauvegardera les données des utilisateurs, les billets achetés et les trajets. Chaque fois qu'on aura besoin d'accéder aux données, on enverra une requête à cette base, qui nous transmettra éventuellement les données demandées.

La validation définitive d'un billet nécessitera un accès au **serveur central** : Le système de validation des billets est l'un des aspects les plus durs de ce projet. Il faut penser à beaucoup de choses, comme la fraude ou les billets simplement invalides.

Cela n'est possible qu'avec une connexion au réseau stable, un requis non négligeable pour acceder à la base de données afin de valider les informations avec la meilleure précision.

Le système devra **prévoir un mode dégradé en cas d'indisponibilité du réseau**, limité à la lecture du billet et au contrôle local : Il faut penser au fait que la validation définitive n'est possible que avec la connexion au réseau, ce qui n'est pas toujours le cas (dans le tunnels, sur des stations loin de la civilisation). Pour résoudre ce probléme, une proposition consiste à implementer une validation "partielle" sans réseau, qui permettra

valider au moins l'integralité du billet, la cohérence crypto et de marquer que le ticket a été validé localement. Cette contrainte est nécessaire et est utilisée par plusieurs systèmes comme SNCF.

3.2. Contraintes de sécurité

Les billets devront être protégés **contre la falsification** :

Il faut toujours penser à la fraude. On propose d'introduire une solution classique assez simple, qui consiste à générer un code QR unique pour chaque billet existant. Cette solution nous permettra d'assurer l'unicité des billets, tandis que la vérification côté serveur permet de détécter les tentatives de fraude.

Comment?

Chaque billet sera associé à un compte utilisateur qui l'a acheté. Il sera possible d'en obtenir un uniquement après un paiement succesif. Les tickets sont gardés sur le serveur, donc si le ticket n'y est pas présent ou s'il ne correspond pas au compte depuis duquel il a été scanné, il y a alors une possibilité de fraude. C'est aussi un aspect discutable. On n'a pas envie de forcer chaque utilisateur à télécharger l'application et à passer la vérification (ce qui est éffectivement la meilleure solution possible), mais de pouvoir retrouver le ticket dans la boîte aux mails. Dans ce projet, pour simplifier un peu la vie, nous allons rester avec la prémiere idée, qui consiste à associer les billets au comptes physiques validés.

Chaque billet devra être identifié de manière unique :Contrainte d'unicité assez typique qui consiste à faire de sorte que chaque identifiant de tickets soit unique. C'est assez réalisable et n'est pas le piége le plus difficile de ce projet.

Un même billet ne devra pas pouvoir être validé plusieurs fois au niveau global :Comme déjà précisé, il y aura 2 niveaux de validation (locale et globale). La validation globale ne pourra se faire qu'une seule fois. Qu'est-ce-qu'on va faire si le billet sera validé la deuxieme fois? Après la validation, on le marque valide et on garde les informations suivantes : par qui il a été validé et quand. Lors les vérifications suivantes, ces informations seront affichées pour les controlleurs. Il faut également penser au cas où le ticket a été bien validé, mais où l'utilisateur souhaite l'utiliser une deuxième fois pour un autre trajet.

Comment on fait?

On peut résoudre ce probléme grâce à une validité globale du ticket. Pour chaque trajet on aura une heure approximative d'arrivée. Chaque minute, lorsque le serveur actualise les données, il marquera invalides tous les billets dont l'heure d'arrivée + 10 minutes (temps approximative pour sortir du quai et où pendant lequel une validation peut encore avoir lieu) est déjà dépassée. Ce n'est pas la solutions la plus sécurisé, mais cela permet quand-même de réduire fortement les tentatives de fraude.

Architecture "Zero-Trust" : Le code optique asssocié à chaque billet ne devra contenir aucune donnée personnelle. ***Toutefois***, pour éviter toute **tentative de falsification reposant sur la génération artificielle de codes séquentiels** (ex: ticket_id = 1,2,3,...), le système devra contenir un identifiant de billet ainsi qu'un mécanisme d'authentification garantissant son intégrité.

L'authentification sera requise pour toute opération sensible (achat, validation, administration) : Cette contrainte consiste à vérifier les droits nécessaires à gérer le système ou des operations sur un billet.

3.3. Contraintes économiques

Le projet devra utiliser exclusivement des **technologies open-source** : Évidemment, on n'a pas accès aux technologies privées des entreprises. On va se contenter avec des outils en accés libre, par exemple la génération des codes QR de Google.

L'infrastructure devra être **déployable sur des ressources limitées** : Cette contrainte implique que le système puisse fonctionner sur une infrastructure serveur légère, disposant de ressources matérielles limitées (CPU, mémoire, stockage).

L'architecture proposée repose sur une **API REST** monolithique et une base de données relationnelle unique (PostgreSQL), afin de limiter la complexité du déploiement et la consommation de ressources. Aucun service externe coûteux ou fortement consommateur de ressources ne sera requis. Cette approche permet un déploiement sur un serveur mutualisé ou une machine de capacité modeste, tout en assurant les fonctionnalités essentielles du système.

Les coûts liés aux services externes devront être évités ou simulés : Il est évident que, pour un projet étudiant à présenter, aucun paiement réel ne sera éffectué. Tous les paiements seront simulés, ainsi que la validation des comptes (pas de connexion avec FranceConnect). Ça va quand-même rester une possibilité si on va penser à vendre notre produit.

3.4. Contraintes opérationnelles

Le système devra fonctionner **malgré une connexion réseau instable** :Cette contrainte est déjà décrite précédemment, avec l'introduction d'une validation locale si le réseau n'est pas accessible.

En cas de perte de connexion, les opérations critiques devront être reportées : Un exemple est la validation locale, qui peut être reportée jusqu'au moment où la connexion est rétablie. Ça peut être implémentée sous forme de requêtes mises en attente, envoyées automatiquement lorsque la connexion sera établie.

La synchronisation avec le serveur devra permettre la résolution de conflits liés aux validations multiples : Cette contrainte a également été décrite précédemment, ainsi que les solutions possibles.

---

4. Exigences fonctionnelles

4.1. Fonctionnalités principales

4.1.1. Recherche de trajet

Le système devra permettre à un utilisateur de rechercher un trajet entre deux points A et B.

Le calcul du trajet reposera sur un réseau ferroviaire prédéfini.

L'intégration de services cartographiques externes est optionnelle mais trés souhaitable à introduire.

4.1.2. Achat et émission de billet

Le système devra permettre l'achat d'un billet pour un trajet sélectionné (s'il y a des places disponibles).

Le processus de paiement sera simulé.

À l'issue de l'achat, un billet électronique unique devra être émis et sauvegardé dans le compte de l'utilisateur.

4.1.3. Génération et gestion des codes QR

Chaque billet devra être associé à un code QR unique. Ce code, dont le contenu ne devra pas permettre l'accès direct aux données personnelles, devra permettre l'identification du billet par le système.

Le système devra empêcher la validation multiple d'un même billet au niveau du serveur si le billet est déjà validé, ou prévenir le contrôleur que ce billet est déjà validé.

4.1.4. Validation des billets

La validation définitive d'un billet nécessitera une communication avec le serveur central.

En cas d'**indisponibilité du réseau**, le système devra permettre : **la lecture du code optique** ; **un contrôle local du billet** à partir des données disponibles en cache; **l'enregistrement du résultat** de ce contrôle dans un journal local pour synchronisation ultérieure et **la prévention du rejeu** (pour éviter qu'une même requête de validation ne soit traitée deux fois par erreur lors du rétablissement de connexion)

Toute décision de validation globale d'un billet restera de la responsabilité du serveur central. En cas de conflit (plusieurs contrôles pour le même billet), la première validation enregistrée par le serveur fera foi.

4.1.5. Notifications

Le système devra notifier l'utilisateur : de l'émission d'un billet; de sa validation et de son annulation ou expiration.

4.1.6. Gestion des erreurs (Error Handling)

Le système devra gérer et signaler de manière cohérente les erreurs suivantes:

Erreurs serveur (5xx): L'application cliente devra afficher un message indiquant une indisponibilité temporaire du service, sans valider ou annuler d'opérations.

Blocage ou indisponibilité de la base de données : Le serveur devra renvoyer un état explicite "service indisponible" et ne modifier aucune donnée.

Perte de connexion pendant l'achat d'un billet : Si le paiement n'a pas été confirmé, aucun billet n'est émis. Par contre, si la confirmation a été envoyée mais le client n'a pas reçu la réponse, une opération d'"idempotence" devra permettre au client de récupérer le billet déjà émis.

Erreurs de lecture du code optique: Code illisible, altéré, le système notifie: "Billet non authentique"

4.2. Scénarios d'utilisation

Seuls les scénarios les plus significatifs ont été retenus afin de décrire de manière concise les usages principaux du système.

Scénario 1 - Achat d'un billet

```
- Acteur principal : Client
- Pré-conditions : Le client dispose d'un compte valide.
```

```
- Post-conditions : Un billet électronique unique est émis et associé au compte.
```

Déroulement :

Le client se connecte à son compte, recherche un trajet entre deux villes, puis sélectionne un service parmi ceux proposés. Après confirmation de son achat, le système simule le paiement, enregistre la transaction, génère un billet électronique unique avec son code optique et l'ajoute automatiquement au compte du client où il devient consultable.

Scénario 2 - Validation d'un billet (en ligne)

```
- Acteur principal : Contrôleur
- Pré-conditions : Le billet est valide et non encore validé au niveau global.
- Post-conditions : Le billet devient validé au niveau global.
```

Déroulement :

Le contrôleur scanne le code optique ; l'unité de contrôle envoie alors une requête de validation au serveur central, qui vérifie l'authenticité et la validité du billet. Si tout est conforme, le serveur enregistre la validation globale et renvoie une confirmation explicite du statut au contrôleur

Scénario 3 - Validation d'un billet (hors ligne / mode dégradé)

```
- Acteur principal : Contrôleur
- Pré-conditions : Le billet est valide ; la connexion réseau est indisponible.
- Post-conditions : Un contrôle local est enregistré, sans modifier l'état global.
```

Déroulement :

Le contrôleur scanne le billet ; l'unité de contrôle détecte l'absence de réseau et consulte les données disponibles en cache. Elle affiche alors "Présenté comme valide" ou "Présenté comme invalide", puis enregistre localement le contrôle dans le journal.

Scénario 4 - Synchronisation après reconnection

```
- Acteur principal : Unité de contrôle + serveur central
- Pré-conditions : Des contrôles locaux sont en attente et la connexion réseau est
rétablie
- Post-conditions : Les billets concernés sont mis à jour au niveau global ; les
conflits sont résolus.
```

Déroulement :

Lorsque la connexion réseau est rétablie, l'unité de contrôle envoie au serveur central l'ensemble des **validations locales horodatées**. Le serveur traite alors chaque contrôle : il **enregistre la première validation chronologiquement** si aucune validation globale n'existe, et marque comme **conflit** toute validation tardive provenant d'une autre unité. L'unité de contrôle met ensuite à jour l'état affiché des billets, puis le journal local est vidé ou marqué comme synchronisé.

5. Hypothèses & Limitations

Dans le cadre de ce projet, plusieurs hypothèses et limitations sont acceptées afin de réduire la complexité de conception et de mise en œuvre

Hypothèses	Limitations
Réseau : Fixe et prédéfini (villes/horaires).	**Offline :** Pré-validation locale uniquement.
Paiement : Entièrement simulé (pas de banque).	**Fraude :** Détection décalée si mode offline.
Supports : 100% numérique (Smartphone/QR).	**Service :** Pas de remboursement ni d'abonnement.
Utilisateurs : Terminaux compatibles avec caméra.	**Hors Périmètre :** Cartographie et FranceConnect.
Contrôle : Application dédiée aux agents.	**Performance :** Sensible aux pics et réseaux instables.

6. Critères de validation

Les critères de validation définissent l'ensemble des conditions permettant d'évaluer objectivement la conformité du système final aux exigences formulées dans le Cahier de Charges.

6.1. Validation des exigences fonctionnelles

ID	Exigence	Indicateur de succès
VAL-01	**Réseau & Trajet**	Gestion de 10 villes et calcul d'itinéraire A -> B fonctionnel.
VAL-02	**Achat & Émission**	Génération d'un billet unique avec QR Code après simulation de paiement.
VAL-03	**Contrôle Online**	Mise à jour instantanée du statut en `VALIDÉ` via API.
VAL-04	**Mode Dégradé**	Lecture QR, vérification via cache local et stockage des logs hors-ligne.
VAL-05	**Synchronisation**	Envoi automatique des logs au serveur et résolution des conflits (fraude).

6.2. Validation des critères non fonctionnels

Le système doit satisfaire des exigences de qualité minimales.

En matière de sécurité, le code QR ne doit pas être exploitable par un tiers sans accès aux mécanismes internes du système.

La fiabilité doit être garantie lors des transitions entre les modes en ligne et hors-ligne, sans perte ni corruption de données.

Les performances attendues imposent un temps de réponse de l'API inférieur à 1000 ms pour les opérations de recherche et de validation dans des conditions normales d'utilisation.

Enfin, l'ergonomie de l'interface doit permettre à un agent de contrôle d'effectuer la vérification complète d'un billet en moins de 10 secondes.

6.3. Conditions de conformité finale

Le produit est jugé conforme lorsque l'ensemble des scénarios d'usage définis en section 4.2 peut être exécuté de bout en bout sans erreur bloquante.

L'état global des billets enregistré sur le serveur central doit constituer l'unique source de vérité du système.

Tout écart technique ou limitation rencontrée lors de l'implémentation doit être explicitement documenté et justifié.