Sujet : tou-tou (garder pour le nom d'appli)

Idées : - web service - postgre sql pour la base de données - outis pour gestion en temmps réel - interface utilisateur intuitive - système de notifications pour les tickets, updates - gestion des codes qr (penser à la gestion de validation multiple) - authentification - trouver un trajet entre deux points A et B (implémenter les maps pour ça?)

Qui s'occupe de quoi : - William : interface - Lia : backend - Van trang : base de données, web service

Cdc:

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

## 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 un code optique unique // un photo plut^t
- 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.
```

Le produit final doit garantir un fonctionnement fiable, une communication claire entre composants et une utilisation conforme au périmètre défini.

## 1.2. Contexte métier

Le domaine de la billetterie numérique implique une série de notions métiers centrales :

```
- **Un réseau de transport**, composé d'au moins dix villes reliées par des
services ferroviaires planifiés. Ce réseau est déterminé à l'avance et ne peut
être modifié dynamiquement dans le cadre du projet.

- **Des clients enregistrés**, disposant d'identifiants permettant l'émission de
titres personnalisés. Ces clients constituent un ensemble fixe au démarrage du
système.

- **Des services de transport** définis par un train, une date, une heure et un
trajet allant d'un point A à un point B. Chaque service correspond à un événement
unique et constitue une unité facturable.

- **Une mécanisme de tarification** rudimentaire, reflétant le coût d'un service
sans nécessiter l'intégration de solutions de paiement réelles.

- **L'émission de titres client-dépendants**, comportant **un code optique**
servant d'identifiant unique du billet. Ce code sert de support à la vérification
locale et doit garantir la non-répudiation du titre.

- **Un contrôle localisé**, assuré par une application spécifique capable de lire
le code optique et d'interroger le serveur pour déterminer si le billet présenté
est valide pour le service en cours.
```

Ce contexte délimite clairement les responsabilités du système, les interactions essentielles entre les acteurs et les flux d'information critiques permettant la validation correcte d'un titre.

## 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 la fenêtre 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. Ce réseau est configuré statiquement et ne peut pas être modifié dynamiquement pendant l'exécution.
- **Client** : Utilisateur final achetant et utilisant des billets pour voyager sur le réseau de transport.
- **Contrôleur (ou unité de contrôle)** : Agent (et/ou application) chargé de vérifier la validité des billets présentés par les clients, à l'aide d'un terminal capable de lire le code optique et de communiquer avec le serveur.

- **Administrateur système** : Utilisateur disposant de droits élevés, responsable de la configuration initiale du réseau (villes, services, tarifs) et de la gestion de la base de 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.). Il constitue l'unique source de vérité pour l'état global des billets.
- **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), permettant de déterminer si un billet est présenté comme valide ou invalide, sans changer l'état global du billet côté serveur.
- **Validation globale** : Décision finale de validation d'un billet, enregistrée sur le serveur central. C'est cette validation globale qui fait foi en cas de conflit ou de tentative de fraude.
- **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 (par exemple depuis une heure donnée jusqu'à 10 minutes après l'heure d'arrivée prévue).
- **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 principal du projet est de concevoir une application de billetterie ferroviaire numérique permettant à des utilisateurs finaux d'acheter, stocker et utiliser des billets de train de manière dématérialisée sur un réseau ferroviaire fixe.

Le système devra permettre :

- à un **voyageur** :
    - de créer un compte utilisateur ;
    - de rechercher un trajet entre deux points A et B du réseau ferroviaire ;
    - d'acheter un billet pour un trajet donné ;
    - de consulter ses billets achetés ;
    - de présenter un billet sous forme de code QR lors d'un contrôle ;
    - de recevoir des notifications liées à ses billets (achat, validation, expiration).

- à un **contrôleur** :
    - de scanner un code QR associé à un billet ;
    - de vérifier la validité d'un billet (en ligne ou hors ligne) ;
    - d'identifier un billet déjà validé ou invalide.
- au **système central** :
    - de gérer les utilisateurs, billets et trajets ;
    - d'assurer l'unicité et la traçabilité des billets ;
    - de centraliser et arbitrer les validations de billets ;
    - de limiter les risques de fraude.

Le projet vise donc à couvrir l'ensemble du cycle de vie d'un billet ferroviaire :
**recherche → achat → émission → validation → expiration**.

---

## 2.2 Objectifs non fonctionnels

En complément des fonctionnalités métier, le système devra respecter plusieurs objectifs non fonctionnels essentiels à sa qualité et à sa viabilité.

**Sécurité**

- Garantir l'intégrité des billets électroniques.
- Empêcher la falsification ou la duplication des billets.
- Protéger les données personnelles des utilisateurs.
- Assurer une authentification fiable pour les opérations sensibles.

**Fiabilité**

- Garantir un fonctionnement cohérent même en cas de connexion réseau instable.
- Assurer la synchronisation correcte entre validations locales et serveur central.
- Prévenir les incohérences liées aux validations multiples.

**Simplicité d'utilisation**

- Proposer une interface utilisateur intuitive et accessible.
- Limiter les actions nécessaires pour acheter ou contrôler un billet.
- Rendre l'expérience fluide aussi bien pour les voyageurs que pour les contrôleurs.

**Performance**

- Permettre des temps de réponse rapides pour la recherche de trajets et la validation des billets.
- Supporter un nombre raisonnable de connexions simultanées sans dégradation notable du service.

**Maintenabilité et évolutivité**

- Reposer sur des technologies standards et open-source.

- Permettre l’évolution future du système (paiements réels, intégration à des services tiers, extension du réseau).

---

# 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.
- Le code QR devra permettre l’identification du billet par le système.
- Le contenu du code QR ne devra pas permettre l’accès direct aux données
personnelles.
- 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. Ici on fait une distinction entre être validé et être valide :
```

```
     * un billet valide est un billet qui a une vraie validité au niveau du
paiement, de la date, du temps et du trajet.

     * un billet validé est un billet VALIDE qui a été validé par un controlleur.
```

#### 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.
```

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. - la prévention du rejeu (Anti-replay): Un jeton d'unicité est intégré au processus de synchronisation pour éviter qu'une même requête de validation ne soit traitée deux fois par erreur lors du rétablissement de connexion

#### 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 la billet déjà émis.
- Erreurs de lecture du code optique:  COde illisible, altéré, le système notifie:
"Billet non authentique"
```

### 4.2. Scénarios d'utilisation

**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.

Il recherche un trajet entre deux villes du réseau.

Le système affiche les services disponibles correspondant au trajet.

Le client sélectionne un service et confirme son intention d’achat.

Le système simule le paiement et enregistre la transaction.

Le système génère un billet électronique unique avec son code optique.

Le billet est ajouté au compte du client et devient consultable.

**Scénario 2 - Consultation des billets par un client**

```
- Acteur principal : Client
- Pré-conditions : Le client possède au moins un billet émis.
- Post-conditions : Aucun changement d’état.
```

***Déroulement***:

Le client accède à son espace personnel.

Le système affiche la liste de ses billets classés par date.

Le client sélectionne un billet.

Le système affiche :les informations du service, la fenêtre de validité, le code optique, le statut actuel (valide / validé / expiré).

**Scénario 3 - 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 une requête de validation au serveur central.

Le serveur vérifie l’authenticité et la validité du billet.

Si tout est conforme, le serveur enregistre une validation globale.

Le contrôleur reçoit une confirmation explicite du statut du billet.

**Scénario 4 - 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.

Le système consulte les données disponibles en cache.

Le système affiche : “Présenté comme valide” ou “Présenté comme invalide”.

Un enregistrement local du contrôle est ajouté au journal.

**Scénario 5 - 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*** :

L’unité de contrôle détecte le retour de la connexion réseau.

Elle envoie au serveur central l’ensemble des validations locales, horodatées

Le serveur traite chaque contrôle :

si aucune validation globale n’existe donc le serveur **enregistre la première validation chronologiquement**,

si le billet a déjà été validé et il provient d'une autre unité: le serveur marque la validation tardive comme **conflit**, potentiellement frauduleuse.

L’unité de contrôle met à jour l’état affiché de chaque billet.

Le journal local est vidé ou marqué comme synchronisé.

**Scénario 6 - Expiration automatique d'un billet**

```
- Acteur principal : Système central
- Pré-conditions : La fenêtre de validité du billet est dépassée.
- Post-conditions : Le billet passe à l'état "expiré".
```

***Déroulement*** :

Le serveur exécute périodiquement la vérification des billets.

Le serveur identifie les billets dont la fenêtre de validité est dépassée.

L'état de ces billets passe à "expiré".

Le client est notifié de l'expiration.

---

# 5. Hypothèses & Limitations

## 5.1 Hypothèses retenues

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

- Le réseau ferroviaire est **fixe et prédéfini** (pas de gestion dynamique des lignes ou horaires).
- Les **paiements sont simulés** et ne font pas intervenir de prestataire bancaire réel.
- Les utilisateurs disposent d'un **terminal compatible** (smartphone ou appareil de contrôle avec caméra).
- Les contrôleurs utilisent une version dédiée de l'application ou un module spécifique.
- Les identités des utilisateurs sont validées via un système interne simplifié (pas d'intégration FranceConnect).
- Les billets sont exclusivement **numériques** (aucune gestion de billets papier).

## 5.2 Limitations du système

Certaines limites fonctionnelles et techniques sont assumées dans le cadre du projet :

- En mode hors ligne, seule une **pré-validation locale** est possible :
    - la validation définitive dépend d'une synchronisation ultérieure avec le serveur ;
    - en cas de conflit, la décision du serveur prévaut.
- Le système ne garantit pas une prévention absolue de la fraude hors ligne :
    - une tentative de double validation peut être détectée uniquement lors de la synchronisation.
- Les performances du système peuvent être impactées :
    - en cas de réseau fortement dégradé ;

    - lors de pics d’utilisation simultanée.

- Le système ne gère pas :
    - les remboursements complexes ;
    - les changements de trajet après achat ;
    - les abonnements longue durée.

- Les services externes (cartographie, paiement réel, identité numérique) sont **hors périmètre** du projet, mais pourront être envisagés dans une version ultérieure.

---

# 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. Ils constituent la base permettant au client, aux utilisateurs et à l’équipe de développement de juger si le produit livré répond correctement aux besoins fonctionnels et non fonctionnels.

## 6.1. Validation des exigences fonctionnelles

Le système sera considéré conforme si les opérations suivantes peuvent être réalisées de manière correcte, cohérente et reproductible :

```
- Gestion d’un réseau fixe d’au moins dix villes.
- Gestion d’un ensemble de clients prédéfinis, correctement identifiés.
- Achat et émission d’un billet unique, associé à un service spécifique et à un
client déterminé.
- Génération d’un code optique unique et décodable pour chaque billet.
- Authentification localisée d’un billet à partir de l’unité de contrôle.
- Gestion explicite des cas d’erreur (billet inexistant, service incorrect,
duplication, format invalide).
```

## 6.2. Validation des critères non fonctionnels

Le système devra satisfaire les exigences suivantes :

```
- Fiabilité des processus critiques (émission, authentification, communication
entre composants).
- Performance raisonnable, notamment pour les réponses d’authentification.
- Sécurité des données clients et des liens client–billet.
- Intégrité des données, sans mise à jour partielle ou incohérente.
- Simplicité d’usage, notamment pour l’unité de contrôle.
```

## 6.3. Critères d’acceptation par les parties prenantes

Le système est jugé acceptable lorsque :

```
- Les clients peuvent acheter un billet et obtenir un code optique utilisable sans
assistance.
- Les agents de contrôle peuvent vérifier un billet rapidement et sans ambiguïté.
- L’administrateur peut gérer la configuration initiale sans intervention
technique complexe.
- Tous les scénarios d’usage définis sont exécutables de bout en bout.
```

### 6.4. Validation en conditions de connectivité limitée (usage à bord du train)

Dans un contexte ferroviaire réel, l’unité de contrôle peut être utilisée dans un environnement à connectivité faible, instable ou inexistante. Le système devra donc respecter les critères suivants :

```
- Tolérance à l’absence de réseau : En cas de non-disponibilité de Wi-Fi ou de
données mobiles, l’unité de contrôle doit afficher un message explicite indiquant
que la validation en temps réel auprès du serveur central est impossible, tout en
restant pleinement utilisable pour effectuer un contrôle local du billet (lecture
du code optique et consultation des données en cache).
Contrôle local sans modification de l’état global
- En l’absence de connexion réseau, l’application de contrôle doit pouvoir : lire
le code optique du billet ; vérifier, à partir des données locales en cache, si le
billet est valide (service,date, heure, fenêtre de validité, statut non-utilisé) ;
et retourner un résultat de contrôle local au contrôleur (par exemple : « présenté
comme valide » / « présenté comme invalide »). Dans ce mode, l’application ne doit
en aucun cas modifier l’état global du billet sur le système central : un billet
ne peut pas être marqué comme validé au niveau global tant que la communication
avec le serveur n’a pas eu lieu.
- Journalisation des contrôles locaux : Chaque contrôle réalisé hors ligne doit
être enregistré dans un journal local, incluant au minimum : l’identifiant du
billet, la date et l’heure du contrôle, l’identifiant du terminal de contrôle et
le résultat du contrôle local. Ces informations seront utilisées lors de la
synchronisation ultérieure avec le serveur central.
-  Non-altération des données locales : L’absence de réseau ne doit entraîner
aucune corruption, perte ou duplication des données stockées localement (cache de
billets, journal de contrôles). Les opérations de lecture et d’écriture locales
doivent rester atomiques et robustes face aux coupures de connexion.- Reprise
automatique et synchronisation des contrôle: Dès que la connectivité est rétablie,
l’application doit retrouver un fonctionnement normal sans nécessiter de
redémarrage manuel. Les contrôles enregistrés localement doivent être synchronisés
automatiquement avec le serveur central, qui :met à jour l’état global des billets
concernés (première validation globale faisant foi) ; signale les éventuels
conflits (billet déjà validé auparavant) comme cas de suspicion de fraude.
- Cohérence après synchronisation : Le rétablissement du réseau ne doit pas créer
d’état incohérent entre l’unité de contrôle et le serveur central. Après
synchronisation, l’application de contrôle doit refléter l’état global effectif de
chaque billet (valide, validé, expiré, refusé, en conflit) de manière non ambiguë
pour le contrôleur.
```

### 6.5. Conditions de conformité finale

Le produit final est conforme lorsque :

```
- Toutes les exigences du Cahier de Charges sont satisfaites.
- Aucun comportement contradictoire ou non spécifié ne subsiste dans les fonctions
principales.
- Les scénarios fonctionnels sont exécutables sans correction manuelle.
- Les éventuels écarts sont documentés, justifiés et validés par le client.
```