# Cahier de Conception : Application de Location de Vélos Urbains

## 1. Présentation du projet

Le projet consiste en la conception d'une solution logicielle mobile dédiée à la micromobilité urbaine. Inspirée des services de type BIXI, cette application vise à faciliter les déplacements citoyens en offrant un accès rapide et autonome à une flotte de vélos en libre-service.

L'objectif principal est de fluidifier l'expérience utilisateur grâce à une interface intuitive permettant de localiser, déverrouiller et régler une location en quelques secondes, tout en optimisant la gestion opérationnelle pour les équipes de maintenance.

## 2. Analyse du besoin

### Besoins Fonctionnels (Ce que l'application fait)

- **Géolocalisation :** Permettre aux utilisateurs de trouver des vélos et des stations à proximité.
- **Gestion des accès :** Système de déverrouillage sécurisé (via QR Code ou Bluetooth).
- **Gestion financière :** Calcul automatique des tarifs en fonction de la durée et prélèvement via une passerelle de paiement.
- **Suivi de flotte :** Permettre aux techniciens de localiser les vélos nécessitant une intervention ou une recharge (si électriques).

### Besoins Non-Fonctionnels (Qualités du système)

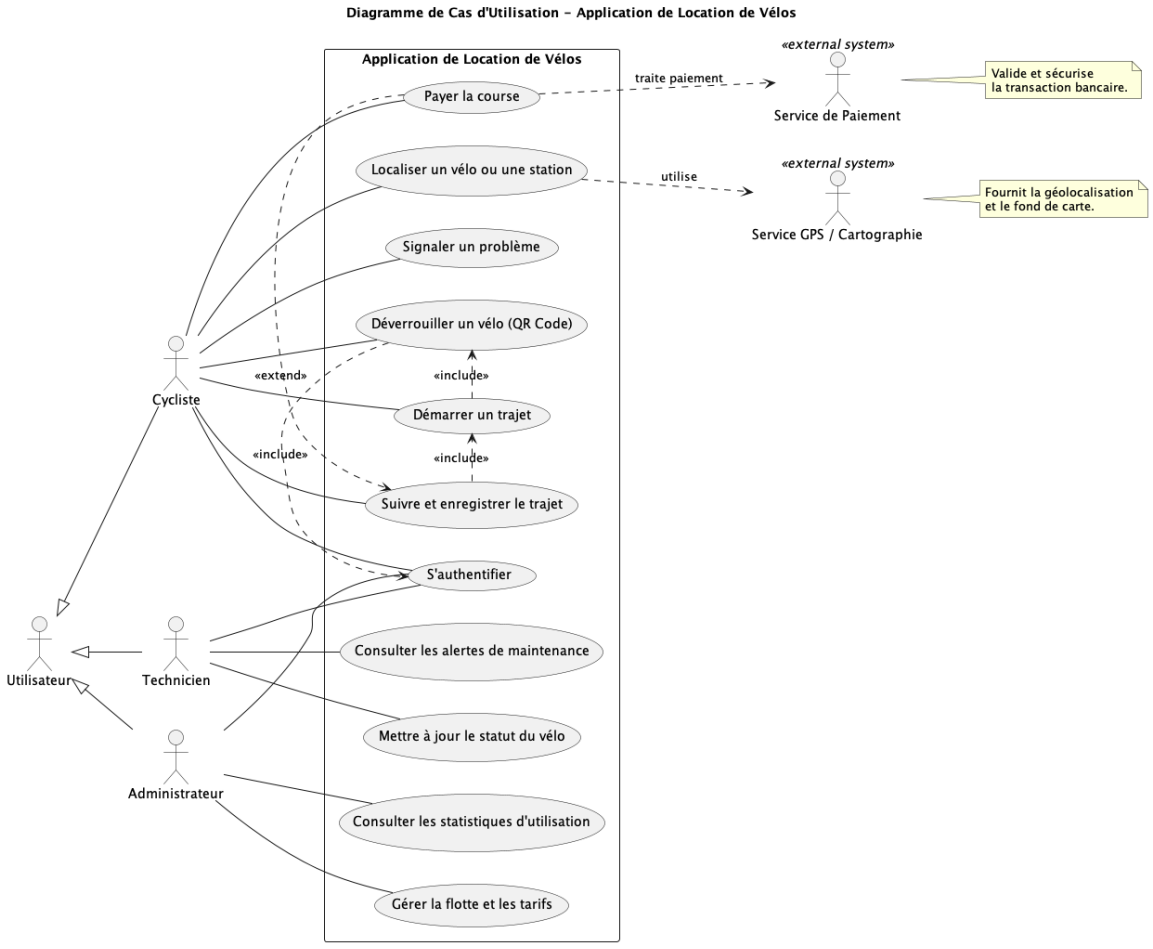
- **Disponibilité :** Le service doit être accessible 24h/24 et 7j/7.
- **Performance :** La mise à jour de la disponibilité des vélos sur la carte doit se faire en temps réel (latence minimale).
- **Sécurité :** Chiffrement des données bancaires et protection des données de localisation des utilisateurs (RGPD / Loi 25).

## 3. Description des utilisateurs et des cas d'utilisation (diagramme UML des cas d'utilisation)

### Identification des Acteurs

| Nom | Rôle |
|---|---|
| **Cycliste (Client)** | Utilise l'application pour louer un vélo, se déplacer et payer. |
| **Administrateur** | Supervise le système, gère les tarifs et les comptes utilisateurs. |
| **Technicien** | Assure la maintenance physique des vélos et des stations. |
| **Service de Paiement** | Gère les transactions financières (ex: PayPal). |
| **Service de Cartographie** | Fournit les données GPS et l'affichage de la carte (ex: Google Maps API). |

### Diagramme de Cas d'Utilisation (UML)



## Parcours Utilisateur (User Flow)

*Scénario Nominal : "Louer un vélo et terminer la course"*

Ce flux représente le chemin "idéal" sans erreur :

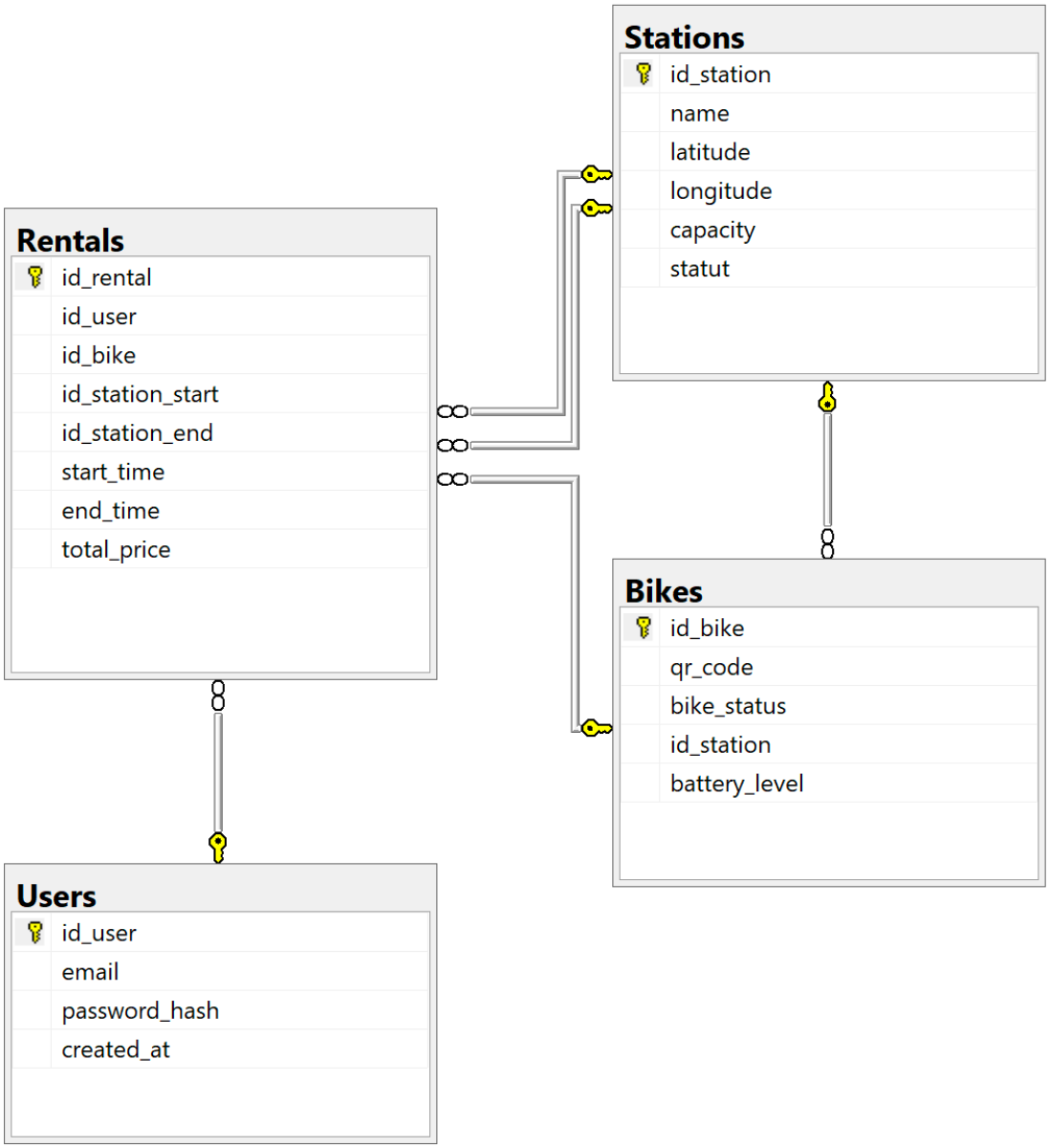
1. **Ouverture & Localisation :** L'utilisateur ouvre l'application. Sa position GPS est détectée et les vélos/stations environnants s'affichent sur la carte.
2. **Sélection :** L'utilisateur marche vers le vélo choisi et clique sur le bouton **"Déverrouiller"**.
3. **Authentification/Validation :** Si non connecté, l'app demande une authentification. Le système vérifie la validité du moyen de paiement.
4. **Action de déverrouillage :** L'appareil photo s'active, l'utilisateur scanne le **QR Code** du vélo. Le verrou électronique se libère.
5. **Suivi du trajet :** L'écran passe en mode "Course active", affichant la durée et le coût estimé en temps réel.
6. **Fin de trajet :** L'utilisateur dépose le vélo dans une station (ou zone autorisée) et verrouille manuellement ou via l'app.
7. **Paiement & Résumé :** L'application confirme la fin de la location, affiche le coût final et envoie une facture par courriel.

## Gestion des Exceptions (Scénarios Alternatifs)

Liste des imprévus :

- **Vélo défectueux :** Si l'utilisateur signale un bris dans les 2 premières minutes, la course est annulée sans frais et le vélo est marqué "En maintenance" pour le technicien.
- **Zone de dépôt interdite :** Si l'utilisateur tente de terminer sa course hors zone, une notification l'alerte et l'empêche de verrouiller le vélo tant qu'il n'est pas dans une zone valide.
- **Échec de paiement :** Si la transaction échoue, le compte est temporairement suspendu jusqu'à régularisation.

## 4. Modélisation de la base de données (modèle relationnel)




## 5. Maquettes et prototypes (lien Figma)

https://www.figma.com/proto/afZLEMTSP5FYR5raB7cmae/Location-de-V%C3%A9los-Urbains?node-id=10-15&t=LKfSvoRRcRrQpcQA-1&scaling=scale-down&content-scaling=fixed&page-id=0%3A1&starting-point-node-id=10%3A15

## 6. Contraintes techniques

**Disponibilité et Temps Réel :** L'application doit refléter l'état exact de la flotte.

**Précision du GPS :** Le système doit intégrer des algorithmes de lissage de position pour calculer les distances parcourues.

**Sécurité des données :** Conformité aux normes (Loi 25 au Québec / RGPD en Europe) pour la protection des données personnelles.

## 7. Choix technologiques

**Environnement :**

- Android / Kotlin

**Backend et API :**

- Langage : ASP.NET Core.
- Architecture : REST API pour la communication entre le mobile et le serveur.

**Base de données**

- BD local : SQLite
- BD distance : MS SQL Server

**Cartographie :**

- Google Maps

**Paiement :**

- PayPal API

**Hébergement :**

- Microsoft Azure (compatible avec SQL Server)