**ÉPREUVE SYNTHÈSE DE PROGRAMME**

**Hiver 2025**

**Nelson Junior Yimou Noubissi**

**#DA :2130331**

**Hiver 2025**



**CAHIER DE CONCEPTION**

**SYSTÈME DE GESTION DE TICKETS DE STATIONNEMENT**

Table des matières

[RAPPEL DU PROJET 2](#_Toc190036846)

[RAPPEL 2](#_Toc190036847)

[BesoinS 3](#_Toc190036848)

[SOLUTION RETENUE 3](#_Toc190036849)

[ÉVALUATION DES COMPOSANTS 4](#_Toc190036850)

[SPÉCIFICATIONS TECHNIQUES 4](#_Toc190036851)

[Langage de développement 4](#_Toc190036852)

[Patron de conception 4](#_Toc190036853)

[Sauvegarde des données 4](#_Toc190036854)

[TECHNOLOGIE EXPÉRIMENTÉES 4](#_Toc190036855)

[MODÉLISATION DE LA SOLUTION 5](#_Toc190036856)

[DIAGRAMME DE BASE DE DONNÉES de L’API 5](#_Toc190036857)

[FONCTIONNALITÉS DU SYSTÈME 6](#_Toc190036858)

[DIAGRAMME DE CAS D’UTILISATION 8](#_Toc190036859)

[DIAGRAMME DE SÉQUENCE POUR L’UTILISATION DE LA BORNE DE PAIEMENT 9](#_Toc190036860)

[DESCRIPTION DES CAS D’UTILISATION 10](#_Toc190036861)

[MAQUETTES DE LA SOLUTION 19](#_Toc190036862)

[MAQUETTES DES ÉCRANS DU LOGICIEL D’ADMINISTRATION 19](#_Toc190036863)

[MAQUETTE(s) DE L’ÉCRAN DE LA BORNE DE PAIEMENT 23](#_Toc190036864)

[PLan de tests 24](#_Toc190036865)

[TESTS UNITAIRTES (BOITE BLANCHE) 24](#_Toc190036866)

[TESTS DE FONCTIONNALITÉS (BOITE NOIRE) 24](#_Toc190036867)

# RAPPEL DU PROJET

## RAPPEL

Le client ici est le CIUSSS Saguenay-Lac-St-Jean. Il souhaite se doter d’un logiciel de gestion des tickets de stationnement. L’application se décline en 4 programmes, qui s’exécuteront chacun sur un équipement différent : La borne d’entrée, la borne de paiement, la borne de sortie et le logiciel d’administration. Cette application a pour but la gestion simplifiée et l’automatisation des tickets de stationnement.

## BesoinS

1. **Usagers/Bénéficiaires** :

* La génération de tickets à la **borne d’entrée**.
* Le payement de facture associée au ticket à la **borne de paiement**.
* La validation de ticket à la **borne de sortie**.

1. **Administrateur** :

* Le suivie des revenus (hebdomadaires ou journaliers) et l’occupation en temps réel via un **tableau de bord**.
* Des ajustements sur les utilisateurs de l’application et les paramètres de tarification via une **console de gestion**.
* La gestion des rapports de revenus avec des outils d’analyse modernes via la **section de consultation des rapports**.

## SOLUTION RETENUE

Une solution native avec API REST et base de données spécifique aux besoins et qui respectes les contraintes du cahier de charges.

# ÉVALUATION DES COMPOSANTS

## SPÉCIFICATIONS TECHNIQUES

### Langage de développement

L’application sera écrite dans le langage **C#** utilisant les interfaces de développement **Visual Studio 2022** et/ou Visual Studio Code. Nous utiliserons .net 6.0 et suivons les règles de programmation telle que les normes de nommage et de structure habituelles. Et nous utiliserons le Framework **WPF**. Le code source sera formaté automatiquement en utilisant CSharpier. Le code et les commentaires seront écris en français.

### Patron de conception

MVVM (Model-View-ViewModel) en utilisant une bibliothèque graphique moderne telle que **MaterialDesignInXAML**.

### Sauvegarde des données

Les données seront sauvegardées dans une BD MySQL sur le serveur de BD sql.decinfo-cchic.ca du Cégep.

### TECHNOLOGIE EXPÉRIMENTÉES

Pour la génération des codes-barres, il est prévu d’utiliser les packages NutGet : **ZXing.Net** et **System.Drawing.Common**. Une preuve de concept est fournie dans la remise de ces documents de conception. En effet, il s’agit d’un petit projet WPF (patron de conception MVVM) qui démontre bien que ces technologies permettent bien de générer des code-barres dont les lecteurs de codes-barres peuvent lire.

# MODÉLISATION DE LA SOLUTION

## DIAGRAMME DE BASE DE DONNÉES de L’API

Une image contenant texte, diagramme, capture d’écran, Police

Description générée automatiquement

## FONCTIONNALITÉS DU SYSTÈME

* 1. **Borne d’entrée** :
* Génération et impression de tickets avec QR code.
* Envoie d’un signal et d’un message d’ouverture respectivement à la barrière et au bénéficiaire.
  1. **Borne de paiement** :
* Scanne du code barre du ticket pour **lecture** afin de calculer le montant à payer.
* Simulation de paiement par carte de crédit.
* Impression de reçus.
  1. **Borne de sortie** :
* Scanne du code barre du ticket pour vérification/validation du ticket payé.
* Envoie d’un signal pour ouvrir la barrière.
* Notification au bénéficiaire de l’ouverture de la barrière
  1. **Logiciel d’administration** :
* Connexion sécurisée et protégée au système.
* Tableau de bord en temps réel (occupation et revenus hebdomadaires ou journaliers).
* Gestion des utilisateurs.
* Modification des paramètres de tarification (tarifs, taxes, etc.).
* Génération et analyse de rapports.
  1. **Fonctionnalités supplémentaires (retenues par l’enseignant)**
* Choix d’un abonnement mensuel ou hebdomadaire. (**Borne de paiement**)
* Basculement de la langue (fr/en) du système. (**Possible à tous les niveaux de l’application**)
* Consultation locale des transactions à chaque borne en cas de problème de synchronisation. (**Borne d’entrée, borne de paiement, Borne de sortie**) (possibilité de savoir dans le logiciel d’administration les synchronisations faites)

# DIAGRAMME DE CAS D’UTILISATION

Le diagramme ci-dessous représente l’utilisation globale de l’application.

Une image contenant texte, diagramme, dessin, croquis

Description générée automatiquement

# DIAGRAMME DE SÉQUENCE POUR L’UTILISATION DE LA BORNE DE PAIEMENT

Une image contenant texte, diagramme, Plan, Parallèle

Description générée automatiquement

# DESCRIPTION DES CAS D’UTILISATION

Les éléments de cette section décrivent en détail les actions effectuées par les utilisateurs et le comportement du système pour réaliser certains scénarios d’utilisation du système. **La liste des scénarios présentés n’est pas exhaustive**, mais résulte d’une sélection de scénarios pour lesquels des précisions ont été jugées importantes à mettre en évidence.

Génération d’un ticket (Borne d’entrée)

|  |
| --- |
| **Nom du cas :** Génération d’un ticket  **But :** Générer un ticket contenant les informations de stationnement et ouvrir la barrière d’entrée.  **Acteur principal :** Utilisateur  **Acteur secondaire :** Borne d’entrée et l’API |
| Séquencement  **Précondition(s)**   * La borne d’entrée est connectée au service de donnée ou un fonctionnement autonome de la borne en cas de problème de synchronisation * Il reste des places disponibles dans le stationnement   **Séquence nominale**  1. L’utilisateur appuie sur le bouton « **Générer Ticket** »  2. La borne d’entrée envoie une requête à l’API pour générer un identifiant unique et enregistrer l’entrée.  3. l’API retourne l’identifiant.  4. La borne génère le ticket.  5. La borne envoie un signal à la barrière pour l’ouverture.  **Postcondition(s)**   * Le ticket est stocké dans la BD. * L’utilisateur entrer dans le stationnement.   **Séquences alternatives**   * *Le stationnement est plein : Affichage d’un message indiquant cela* * *La borne est incapable de fonctionner de manière autonome en cas de problème se synchronisation.* |

Scan du ticket (Borne de paiement)

|  |
| --- |
| **Nom du cas :** Scan du ticket  **But :** Lire le ticket pour calculer le montant  **Acteur principal :** Utilisateur  **Acteur secondaire :** Borne de paiement et l’API |
| Séquencement  **Précondition(s)**   * L’utilisateur doit avoir un **ticket valide** généré à l’entrée.   **Séquence nominale**  1. L’utilisateur **scanne son ticket** via le lecteur de code-barres.  2. La borne envoie une requête à l’API pour récupérer les informations du ticket.  3. l’API renvoie les informations du ticket.  4. La borne calcule le **montant dû**.  5. la borne affiche le montant à payer.  **Postcondition(s)**   * Le montant est affiché à l’utilisateur.   **Séquences alternatives**   * ***Ticket invalide****: Un message indiquant cela s’affiche.* * ***Ticket déjà payé :*** *Un message indiquant cela s’affiche.* |

Simulation de paiement (Borne de paiement)

|  |
| --- |
| **Nom du cas :** Simulation de paiement  **But :** Effectuer le paiement du stationnement par carte de crédit  **Acteur principal :** Utilisateur  **Acteur secondaire :** Borne de paiement et l’API |
| Séquencement  **Précondition(s)**   * L’utilisateur a scanné un **ticket valide.**   **Séquence nominale**  1.L’utilisateur insère sa carte de crédit et valide le paiement.  2. La borne de paiement envoie la transaction au système bancaire.  3. Une fois payé la borne envoie une requête à l’API pour mettre à jour le ticket(EstPaye = true)  4. La borne affiche « **Paiement réussi »**  5. La borne envoie un signal à la barrière pour l’ouverture.  **Postcondition(s)**   * Le ticket est marqué comme **payé** dans la BD. * L’utilisateur peut sortir du stationnement.   **Séquences alternatives**   * ***Problème de paiement****: Un message indiquant cela s’affiche.* |

Validation du paiement (Borne de sortie)

|  |
| --- |
| **Nom du cas :** Validation du paiement  **But :** Vérifier si un ticket a été payé avant d’ouvrir la barrière  **Acteur principal :** Utilisateur  **Acteur secondaire :** Borne de sortie et l’API |
| Séquencement  **Précondition(s)**   * L’utilisateur doit posséder un **ticket valide (déjà payé).**   **Séquence nominale**  1.L’utilisateur **scanne son ticket** à la borne de sortie.  2. La borne envoie une requête à l’API pour **vérifier si le paiement a été effectué**.  3. Si le ticket est payé, la borne **envoie un signal à la barrière** pour l’ouverture.  4. La borne affiche **un message d’aurevoir**.  **Postcondition(s)**   * L’utilisateur peut quitter le stationnement.   **Séquences alternatives**   * ***Ticket non payé :*** *Un message indiquant cela s’affiche.* |

Connexion sécurisée (Logiciel d’administration)

|  |
| --- |
| **Nom du cas :** Connexion sécurisée  **But :** Permettre à un administrateur de se connecter au système de gestion  **Acteur principal :** Administrateur  **Acteur secondaire :** logiciel d’administration et l’API |
| Séquencement  **Précondition(s)**   * L’administrateur possède un **compte valide** avec un **mot de passe sécurisée**.   **Séquence nominale**  1.L’utilisateur entre les informations de connexion.  2. Le logiciel envoie une requête à l’API pour authentification.  3. Si les informations sont correctes, accès accordé au tableau de bord.  **Postcondition(s)**   * L’administrateur est connecté et peut gérer le système.   **Séquences alternatives**   * *Mot de passe incorrect : Un message indiquant cela s’affiche* * *Mot de passe oublié : Un mot de passe temporaire est affiché pour se connecter.* |

Génération de rapports financiers (Logiciel d’administration)

|  |
| --- |
| **Nom du cas :** Génération de rapports financiers.  **But :** Générer un rapport détaillé des revenus sur une période donnée.  **Acteur principal :** Administrateur  **Acteur secondaire :** Logiciel d’administration et l’API |
| Séquencement  **Précondition(s)**   * L’administrateur est connecté.   **Séquence nominale**  1.L’adminstrateur accède à la section **Les rapports**.  2.L’adminstrateur sélectionne une période.  3. L’API récupère les **transactions correspondantes**.  4. Le logiciel affiche le rapport et les informations correspondantes.  5. Un fichier PDF du rapport est généré.  **Postcondition(s)**   * Le rapport est disponible pour téléchargement.   **Séquences alternatives**   * ***Aucune donnée disponible****: Un message indiquant cela s’affiche.* |

Consultation des statistiques en temps réel (Logiciel d’administration)

|  |
| --- |
| **Nom du cas : Consultation** des statistiques en temps réel.  **But :** Afficher le nombre de places disponibles et le nombre de places occupés en temps réel.  **Acteur principal :** Administrateur  **Acteur secondaire :** Logiciel d’administration et l’API |
| Séquencement  **Précondition(s)**   * L’administrateur est connecté.   **Séquence nominale**  1.L’adminstrateur accède au **tableau de bord**.  2.L’API récupère les **places occupées / disponibles**.  3.Le logiciel génère un graphique circulaire.  4.Les statistiques sont affichées.  **Postcondition(s)**   * L’administrateur visualise les performances du stationnement.   **Séquences alternatives**   * ***Problème de connexion à l’API/BD****: Un message indiquant cela s’affiche.* |

Modification d’un tarif / abonnement (Logiciel d’administration)

|  |
| --- |
| **Nom du cas :** Modification d’un tarif / abonnement**.**  **But :** ajuster les paramètres d’un tarif ou d’un abonnement.  **Acteur principal :** Administrateur  **Acteur secondaire :** Logiciel d’administration et l’API |
| Séquencement  **Précondition(s)**   * L’administrateur est connecté.   **Séquence nominale**  1.L’adminstrateur accède à **la console de gestion**.  2.L’administrateur effectue les modifications.  3. L’administrateur clique sur le bouton **Enregistrer**.  4.Le logiciel envoie une requête à l’API pour mettre à jour dans la BD.  **Postcondition(s)**   * Les mises à jour sont appliquées dans tous le système.   **Séquences alternatives**   * *L’administrateur clique sur* ***Annuler****: Et l’action est annulé ou pas enregistrée.* |

# MAQUETTES DE LA SOLUTION

## MAQUETTES DES ÉCRANS DU LOGICIEL D’ADMINISTRATION

Une image contenant texte, capture d’écran, affichage, nombre

Description générée automatiquement

***Maquette de la page de connexion au logiciel d’administration***

Une image contenant texte, capture d’écran, logiciel, Parallèle

Description générée automatiquement

**Maquette représentant le tableau de bord du logiciel d’administration**

Une image contenant texte, capture d’écran, Parallèle, logiciel

Description générée automatiquement

MaquetteMxdddxxddddfacfffffffffMasssdddd

***Maquette représentant la console de gestion du logiciel d’administration***

Une image contenant texte, capture d’écran, logiciel, Page web

Description générée automatiquementHFEKPOL;SDDDQUEDDDDDDDTTEHHHDDDDDD

***Maquette représentant la section « Consultation de rapports » du logiciel d’administration***

## MAQUETTE(s) DE L’ÉCRAN DE LA BORNE DE PAIEMENT

Une image contenant texte, diagramme, logiciel, capture d’écran

Description générée automatiquement

# PLan de tests

## TESTS UNITAIRTES (BOITE BLANCHE)

En parallèle au développement de l’application, des tests unitaires seront mis en place afin de vérifier le bon fonctionnement de chacune des classes tout au long du développement. Ces tests unitaires seront exécutés minimalement à la fin de chaque sprint et avant toute livraison au client, que ce soit pour une démonstration, un livrable d’étape (prototype) ou pour la livraison finale. Un rapport sur les résultats des tests unitaires sera fourni lors de la livraison du produit final.

## TESTS DE FONCTIONNALITÉS (BOITE NOIRE)

Les tests de fonctionnalités suivants seront réalisés lors de la période de tests et débogage ainsi qu’avant la livraison du produit final. Un rapport sur les résultats des tests de fonctionnalités sera fourni lors de la livraison du produit final.

***La liste suivante des tests n’est pas exhaustive, elle est en lien avec la description des cas effectuée précédemment***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Description | Préconditions | Scénario | Résultats attendus |
| 1 | Génération d’un ticket | 🡪La borne d’entrée est fonctionnelle.  🡪Il reste des places disponibles. | 1. L’utilisateur appuie sur le bouton "Générer Ticket".  2. Le système enregistre l’heure d’entrée et génère un identifiant unique.  3. Un ticket avec un code-barre est imprimé.  4. La borne envoie un signal à la barrière pour l’ouverture. | * Un Ticket avec Id unique est généré-imprimé. * La barrière s’ouvre automatiquement. |
| 2 | Scan d’un ticket (Borne de paiement) | 🡪L’utilisateur possède un ticket valide.  🡪La borne de paiement est fonctionnelle et/ou connectée à l’API | 1. L’utilisateur scanne son ticket.  2. La borne envoie une requête à l’API pour récupérer l’heure d’arrivée.  3. Le montant est calculé et affiché à l’utilisateur. | * Le montant correct s’affiche. * Un message d’erreur apparait si le ticket est invalide. |
| 3 | Simulation de paiement (Borne de paiement) | 🡪Le montant à payer a été affiché.  🡪L’utilisateur insère une carte de crédit. | 1. L’utilisateur confirme le paiement.  2. La borne envoie une requête au simulateur de paiement.  3. Si validé, l’API met à jour le statut du ticket (EstPaye = True).  4. L’utilisateur peut imprimer un reçu. | * Le paiement est accepté et le ticket est marqué comme « **Payé »** * **Un reçu s’imprime.** * Message d’erreur si le paiement échoue. |
| 4 | Validation du paiement (Borne de sortie) | 🡪L’utilisateur possède un ticket payé  🡪La borne de sortie est fonctionnelle et/ou connectée à l’API | 1. L’utilisateur scanne son ticket à la borne de sortie.  2. L’API vérifie que le ticket est payé.  3. Si oui, un signal est envoyé pour ouvrir la barrière. | * La barrière s’ouvre si le paiement est confirmé. * Message « **Veuillez payer votre ticket** » si le paiement est manquant |
| 5 | Connexion sécurisée (Logiciel d’administration) | L’administration possède un compte actif. | 1. L’administrateur entre ses informations de connexion  2. L’API vérifie les informations.  3. Si valides, l’accès est autorisé et la session démarre. | * L’accès est accordé et l’admin accède au tableau de bord. * Un message d’erreur en cas d’échec de connexion. |
| 6 | Génération de rapports financiers | 🡪L’administrateur est connecté.  🡪Des transactions/rapports existent dans la BD | 1. L’admin accède à "Rapports financiers".  2. Il sélectionne une période et génère un rapport.  3. L’API compile les données.  4. le logiciel génère le rapport et propose de le télécharger. | * Un rapport PDF est généré et téléchargeable. * Message « Aucune donnée pour cette période »Si aucune revenu n’a été enregistré. |
| 7 | Consultation des statistiques en temps réel. | 🡪L’administrateur est connecté.  🡪Le système suit les transactions en temps réel. | 1. L’admin accède au tableau de bord.  2. L’API récupère le nombre de places disponibles et le chiffre d’affaires du jour.  3. Un graphique des revenus s’affiche. | * Les statistiques s’affichent correctement. * Un Message spécifique d’erreur en cas d’erreur. |
| 8 | Modification d’un tarif/abonnement | L’administrateur est connecté. | 1. L’admin accède à "Gestion des Tarifs".  2. Il sélectionne un tarif et effectue ses modifications.  3. L’API met à jour la base de données. | * Le tarif/l’abonnement est mis à jour et la mise à jour est visible. * Message d’erreur si la mise à jour échoue. |