Projet Génie Logiciel
Système de parking de Carville

DANIEL Kevin
SAEBI Pardis
VIBERT Léa
RATSIANDAVANA Sonia

M1/M2 RS MIASHS DCISS
Année 2020-2021

TABLE DES MATIÈRES

.

A. Structure générale du système de parking

Remarque: les uses-cases, les diagrammes d'activité et de séquence présents dans ce rapport sont aussi disponibles en grand format dans le dossier de rendu. Le diagramme de classe étant trop imposant, il ne sera visible que dans le dossier de rendu.

Pour commencer, nous avons conçu le diagramme de classe et les uses-cases du système. Nous avons ainsi pu nous répartir les diagrammes de séquence et d'activité en ayant une base commune.

1.Diagrammes des cas d'utilisation (UC)

Nos diagrammes de cas d'utilisation capturent le comportement de système de contrôle de parking et ses deux sous-systèmes tel qu'un utilisateur extérieur le voit. Le système donné est un système complet et les diagrammes d'UC et de l'activité nous donnent un moyen simple d'exprimer les besoins définis par le client (Requirements). D'abord nous nous sommes intéressés aux acteurs (externes). Nous avons initialement identifié ces acteurs :

- Client
- Employé de parking
- Agent de maintenance
- Agent de surveillance et sécurité
- Senseur d'accès

1.1 Cas d'utilisation

Dans le diagramme de UC #1 affiché en annexes nous avons démontré quelques scénarios parmi de scénarios qui existent entre les acteurs et le système de contrôle des parkings qui représentent une conception initiale de ce système pendant son développement.

Tous les acteurs humains peuvent aussi être un client de ce système donc une relation de type généralisation existe entre client et les acteurs humains(pour la simplicité,on a décidé de ne pas la représenter ici).

- Le client:

  Le client de parking interagit avec le système de parking (le parking) en entrant, se garant, payant et sortant de parking. Il peut s'abonner aux parkings aussi.

- Employé de parking:

Les acteurs classiques ne se trouvent plus dans ce parking comme billeteur,responsable de paiement, secrétaire, etc parce que le système est semi automatique. Dans le cas d'employé de parking,ses rôles sont liés aux anomalies dans le système (enregistrement des fautes, gestion des paiements, notification de l'agent de maintenance, activer la barrière de sortie, et la validation de l'heure de passage des agents de surveillance.

- Agent de maintenance:

  On considère que les agents de maintenance s'authentifient lors d'arrivée sur place avec la borne de contrôle d'entrée et ceux sont eux qui ajoutent la date et l'heure de réparation des fautes dans le système chaque fois que cela arrive. On peut considérer un secrétaire comme un acteur externe qui insère ces données dans le système. Mais dans nos UC on n'a pas pris ce choix.

- Agent de surveillance et sécurité:

  Dans les spécifications, on ne sait pas qui insère les heures et les durées de présence des agents de surveillance dans le système. Mais il est très probable que ce sont eux-mêmes qui prennent en charge ce rôle et la validation de l'heure de passage de ces agents se fait par l'employé de parking.

- Senseur d'accès

  Lors d'arrivée d'une voiture, le sensor d'entrée déclenche l'affichage de signal "Appuyez sur le bouton".

1.2. Sous-systèmes

Nous pouvons aussi regarder d'une autre façon le système de contrôle des parkings en nous basant sur les deux sous-systèmes. Le sous-système opérationnel (quotidien) et le système de gestion (non-quotidien). Vous voyez dans le UC#2 les deux sous-systèmes emboîtés dans le système complet de contrôle de parking.

Nous avons montré quelques « use case » du diagramme UC#1 précédente dans ce diagramme. Nous avons ajouté dans le sous-système de gestion, le « use case » liés à la vente d'abonnement et mise à jour des abonnements ainsi que la demande d'abonnement par le client. Ici, les acteurs interagissent avec les deux sous-systèmes. Il y a un acteur qui s'appelle database server système qui met à jour des abonnements y compris suppression, ajout et modification des abonnements (comme ça se fait par un bibliothécaire pour les adhérents). Ce serveur va ensuite transmettre les infos à jour pour être utilisé par le système opérationnel quotidien dans les bornes de contrôle. Il est un acteur externe pour les deux sous-systèmes. Etant donnée que le but des diagrammes de cas d'utilisation est de se concentrer sur les acteurs externes, nous retournons sur notre diagramme de UC#1 avec plus de détails.

1.3. Diagramme des cas d'utilisation détaillés

Dans le diagramme UC#3, on a ajouté d'autres « use case » (requirements) pour l'acteur « employé de parking » intitulé « voir l'info d'abonné », « voir statut de borne de paiement et de borne de contrôle ». Ce système est un système automatisé d'où vient le fait de ne pas avoir un acteur classique externe pour vérifier par exemple la liste des abonnées de parking avec le client. Le rôle de cet acteur remplissant cette tâche se fait de manière automatique entre le sous-système de base de données et les bornes de système opérationnel (moins d'acteur humain égale moins employé) moins de charge pour l'entreprise de contrôle des parkings. Nous avons montré les données de cette base d'enregistrement comme un acteur externe.

2. Diagramme de classe

2.1 Parking

Nous avons commencé par émettre l'hypothèse qu'un parking était doté d'une ou plusieurs bornes de contrôle et de paiement, ainsi que d'un écran permettant de signaler la disponibilité des places aux clients. Ces éléments nous ont paru essentiels et indissociables de l'objet parking.

2.2 Employé de parking

Nous avons créé une classe pour les employés du parking. En tant qu'utilisateurs privilégiés du système, ils ont accès à certaines fonctionnalités :

- voirStatutBorne()
- lireCodeBarre() / calculerMontantDu() : ces deux méthodes leur permettent de lire le code barre du client et de calculer le montant dû.
- verifierDonneesTicket(): permet de valider le ticket pour lever la barrière.
- enregistrerFaille() / rapportSecurite() / notifierEntreprise() : notre hypothèse est que ce sont les employés du parking qui enregistrent les failles concernant les visites de sécurité. Ce sont également eux qui produisent le rapport et l'envoient, grâce aux données enregistrées durant le mois.
- enregistrerFaute / rapportFautes() / notifierMunicipalite() : ils enregistrent également les problèmes du parking et notifient le groupe de maintenance de la municipalité en cas de problème concernant l'équipement et ce qui concerne le système opérationnel..

2.3 Bornes

Les bornes de contrôle, de paiement et le pointeur automatique partagent des éléments communs : elles héritent donc de la classe borne qui possède les composants écran LCD et fente d'insertion.

L'écran LCD permet d'afficher les messages (afficherMessage()), et la fente d'insertion reçoit les tickets/cartes (recevoirTicket()) et les rejette en cas de problème (ejecterTicket()).

2.3.1 La borne de contrôle

La borne de contrôle possède plusieurs fonctionnalités qui lui sont propres:

- imprimer des tickets() : cette méthode permet d'imprimer des tickets pour les clients ordinaires.
- changerStatutDistribution() : permet de changer de statut de distribution, par exemple dans le cas où le parking est plein, arrêter de distribuer des tickets.
- verifierTicketSortie(): permet de vérifier le ticket du client ordinaire lors de la sortie du parking. Si le client est resté moins de 15mn après le paiement, la méthode renvoie "vrai", sinon elle renvoie "faux".
- vérifierValiditeAbonnement(): permet de vérifier si l'abonnement sur le ticket du client abonné est toujours valide.
- activerInterphone() : permet d'activer l'interphone de la borne de contrôle.
- enregistrerHeureDepart() : permet d'enregistrer l'heure de départ du client abonné.

2.3.2 La borne de paiement

La borne de paiement possède deux composants essentiels :

- le réceptacle pour la monnaie : il possède une méthode ejecterMonnaie() qui permet de rendre la monnaie au client.
- le réceptacle pour le reçu : sa méthode ejecterRecu() permet de délivrer le reçu au client

2.3.3 Pointeur automatique

Le pointeur automatique sert pour les visites de sécurité: il permet d'enregistrer les allées et venues des gardiens de sécurité. Il possède trois méthodes qui lui sont propre:

- incrementerVisiteParJour()
- remiseAZero()
- calculerTempsParVisite()

Le pointeur étant une borne, il peut recevoir une carte (la borne est marquée comme recevant un ticket, mais elle peut aussi recevoir une carte).

2.4 Tarif

Un tarif possède plusieurs attributs :

- tarifSemaine : entier
- tarifSoiree : entier
- tarifWeekend : entier

Nous avons identifié 2 types de tarifs qui héritent de cette classe Tarif:

- Les tarifs longs
- Les tarifs courts

Un parking choisit de mettre en place un de ces deux types de tarif avec des tarifSemaine, tarifSoiree et tarifWeekend spécifiques.

2.5 Ticket, reçu et code barre

Nous avons créé des classes spécifiques pour le ticket et le reçu : il s'agit de deux objets indépendants, possédant des attributs propres.

Le ticket est utilisé par les clients pour entrer et sortir sur parking : les clients ordinaires prennent un ticket à chacun de leur passage tandis que les clients abonnées possèdent un abonnement. Le ticket est délivré par la borne de contrôle et c'est également elle qui le scanne lors des différents passages du client (abonné ou non).

Le ticket possède des attributs lisibles par le client:

- numero: entier
- dateEntree: entier
- heureEntree: entier

Mais il possède également un code barre, lisible par la borne lors du départ du client. Ce code barre est indispensable, car il permet à la borne de contrôle de vérifier si moins de 15mn se sont écoulées entre le paiement et la sortie.Nous avons donc choisi de lier le code barre et le ticket par une relation de composition.

Le reçu est délivré par la borne de paiement si le client le souhaite. Il possède différents attributs qui sont lisibles par les clients:

- adresseMunicipalite: chaine
- adresseParking: chaine
- numeroTVA: entier
- datePaiement: entier
- montantPaye: entier

2.6 Senseur et barrière

Un senseur permet de possède une méthode permettant de détecter le passage d'une voiture, mais il en existe deux types: le premier, que nous avons appelé "senseur d'accès", s'active et lève la barrière lorsque la borne de contrôle lui notifie que le client est autorisé à passer. Cela peut se faire dans les deux sens : lors de l'entrée du client dans le parking et lors de sa sortie du parking. Le deuxième, que nous avons appelé "senseur de fermeture", s'active de manière autonome et permet d'abaisser la barrière lorsque le client est passé.

La barrière possède deux méthodes : l'une permet de lever la barrière (leverBarriere()) et l'autre permet de la baisser (baisserBarriere()). Il est difficile de

déterminer si la barrière fait partie intégrante de la borne de contrôle ou s'il s'agit d'un élément à part entière. Notre hypothèse est qu'une borne de contrôle sans moyen de restriction n'a pas de fondement: il serait inutile de contrôler le client sans l'empêcher de passer. Cependant, une barrière n'est pas nécessairement dépendante d'une borne de contrôle. Dans notre cas, elle peut être fermée par le senseur de fermeture. Lors de la suppression de l'instance de la borne de contrôle, on supprimerait également la barrière qui y est rattachée. Nous avons donc choisi de considérer la barrière comme un élément indépendant, mais rattaché à la borne de contrôle par une relation d'agrégation. Ainsi, le changement d'état de la borne de contrôle peut impliquer un changement d'état de la barrière.

2.7 Événement

2.7.1 Fautes et visites de sécurité

Les fautes et les visites de sécurité héritent de la classe Evènement car elles possèdent des attributs partagés:

- organisation: chaine
- date: entier
- heure: entier
- duree: entier

Les fautes possèdent des attributs supplémentaires:

- objetConcerne: chaine
- details: chaine

L'employé qui l'enregistre a la possibilité d'ajouter une description détaillée de la faute et de préciser quel était l'objet défectueux.

2.7.2 Rapport

Les évènements sont relatés dans des rapports mensuels, qui sont validés et envoyés par les employés du parking. Les rapports ne sont pas envoyés directement par le système, mais revus, et commentés humainement. Cela nous a paru important pour que la description des événements soit la plus détaillée possible.

Par exemple, le système n'enregistre que les données tirées du pointeur, mais l'humain peut noter les détails quant à un éventuel problème qui aurait pu perturber les visites de sécurité. De même, les fautes sont enregistrées par les employés du parking, car seuls eux sont à même de vérifier si une fenêtre est cassée.

B. Système opérationnel

1. Entrée dans le parking

Nous avons choisi de réaliser deux diagrammes d'activités ainsi que deux diagrammes de séquences pour modéliser l'entrée dans le parking.
En effet, il existe le cas normal où le client peut entrer dans le parking s'il reste de la place, et le cas exceptionnel où le parking est complet. Il est donc nécessaire de séparer les deux pour comprendre comment réagit le système dans un cas exceptionnel.

1.1 Déroulement principal

Nous allons donc commencer par le cas où le parking n'est pas complet. Lorsque le client approche de la barrière d'entrée du parking avec sa voiture, il est détecté par un senseur. C'est ce senseur qui permet d'enclencher tout le processus de la borne de contrôle. Cette dernière affiche donc un message "Appuyez sur le bouton", le client s'exécute et obtient donc un ticket. Si le client est abonné au parking, il lui suffit simplement d'insérer son ticket d'abonnement dans la fente de la borne de contrôle. Celle-ci prend maximum 5 secondes pour vérifier la validité de l'abonnement, qu'on est un jour de semaine, et que le client abonné n'a pas déjà été enregistré en tant qu'entrant ou sortant du parking.
Ensuite, quand le client ordinaire a pris son ticket, ou que les vérifications sont effectuées pour le ticket d'abonnement dans le cas du client abonné, la barrière se lève. Dans les deux cas, la borne de contrôle enregistre des données : l'heure d'entrée pour l'abonné; numéro de ticket, date d'impression, heure d'impression et machine distributrice pour le client ordinaire.
Dans le même temps, quand la voiture à passer la barrière, un senseur situé derrière celle-ci détecte que la voiture a passé la barrière, et la borne de contrôle incrémente de 1 le nombre de voitures présentes dans le parking. La capacité du parking diminue donc.

Nous avons donc réalisé un diagramme d'activité pour ce scénario où il reste de la place dans le parking, pour illustrer le déroulement de l'activité de chacun des acteurs et objets dans le système, l'ordre d'action, ainsi que le cheminement de l'activité selon que le client soit un client ordinaire ou abonné au parking.
Nous avons fait le choix de terminer ce diagramme par l'incrémentation du nombre de véhicules dans le parking car cet événement marque définitivement l'entrée du client dans le parking.
Dans le diagramme de séquence, nous avons différencié par un fragment combiné dit “alternatif” les évènements qui découle selon la condition que remplit le client : ordinaire ou abonné.

Ces deux alternatives débutent par les mêmes actions : la détection de la voiture arrivant devant la barrière d'entrée, ainsi que l'affichage d'un message sur la borne de contrôle. De même, elles terminent toutes deux avec les mêmes actions : la détection de la voiture passant la barrière d'entrée, et l'incrémentation du nombre de véhicules.

1.2 Déroulements exceptionnels

Abordons maintenant le cas exceptionnel où le parking serait complet.
Tout d'abord, un message est affiché à l'entrée du parking "Parking complet" sur un panneau d'affichage pour en informer les potentiels clients.
Les diagrammes d'activités et de séquence produits dans ce cas exceptionnels reprennent le même début de schémas que les précédents diagrammes, c'est-à-dire que le senseur détecte la voiture devant la barrière d'entrée.
Ensuite le message affiché est "Complet". Le client ordinaire ne peut pas appuyer sur le bouton, cela n'aura aucun effet. Et le client abonné ne peut pas insérer sa carte d'abonnement dans la fente qui sera fermée.

Dans le cas où un véhicule sort, la borne de contrôle affiche le message "Appuyez sur le bouton", et le scénario normal continue.

2. Paiement

Pour le paiement, nous avons décidé de présenter deux diagrammes de séquence, un diagramme de séquence dans le cas normal, celui où le ticket est lu correctement et celui dans le cas où le code barre du ticket n’est pas bien lu. Nous avons décidé de présenter un diagramme d’activité pour détailler le processus de paiement lorsque le ticket est valide.

1.1 Déroulement principal

Le déroulement du paiement se fait à la borne de paiement. Le procédé commence par l’introduction du ticket dans la borne de paiement par le client. C’est à cela que sert la fente d’insertion dont est muni la borne de paiement. La borne de paiement fait plusieurs actions sur le ticket, , lireCodeBarre(), calculerMontantDu(), verifierDonneeTicket(), qui permettent d’obtenir le montant que le client doit payer (montant dû sur les diagrammes). A la suite de cela, la borne de paiement affiche le montant sur l’écran LCD dont elle est munie et le client doit payer. Pour payer, le client doit insérer de la monnaie, billets ou pièces. Pendant que le client insère sa monnaie, la borne de contrôle vérifie. Si la monnaie n’est pas valide, elle est rejetée dans le réceptacle à monnaie de la borne de paiement avec la fonction ejecterMonnaie(). Si la monnaie est valide, la borne de paiement actualise le montant que le client doit introduire grâce à sa fonction actualiserMontantDu(). Lorsque le client a payé la somme afficher le ticket est validé et éjecté de la borne de paiement pour que le client le récupère. La borne de paiement propose alors au client un reçu. Si le client ne veut pas de reçu alors la borne de paiement affiche un dernier message (message de fin) sur son écran LCD pour informer le client de la suite du processus pour sortir du parking. Si

le client veut un reçu, alors un reçu est créé avec tous ses attributs et est imprimé et éjecté de la borne de paiement par le réceptacle à reçu que la borne de paiement a.

Ce processus est assez long et difficile à visualiser surtout pour les actions de la borne de contrôle c'est pour cela que nous avons décidé de proposer un diagramme d'activité pour ce processus. En effet, il nous a paru important de détailler les tâches qui sont effectuées par la borne de paiement lorsqu'un client veut payer.

De plus, nous avons aussi fait un diagramme de séquence ce qui permet de visualiser les interactions entre les acteurs et les étapes du processus.

1.2 Déroulements exceptionnels

Le déroulement exceptionnel du processus de paiement est lorsque la borne de contrôle n'arrivera pas à bien déchiffrer le code barre du ticket. Il faut alors se diriger vers l'employé du parking. L'employé du parking remplace alors les actions de la borne de paiement.

Dans ce cas, il nous a semblé moins important de détailler car le système du parking entre moins en facteur. C'est pour cela que nous n'avons pas fait de diagramme d'activités pour ce cas. Nous avons préféré mettre l'accent sur les tâches plus complexes pour le système.

Nous avons quand même fait un diagramme de séquence pour ce cas pour permettre de visualiser les échanges entre les différents acteurs.

3. Sortie du parking

Le scénario de sortie du parking étant assez bref, nous avons décidé de fusionner le déroulement principal et le déroulement exceptionnel.

Lorsque la voiture s'approche de la barrière de sortie, le senseur d'accès s'active (cette action correspond à la méthode héritée detecteVoiture() du senseur). Cela permet d'afficher le message "insérer ticket" sur l'écran LCD de la borne de contrôle (méthode afficheMessage() héritée de borne).

Une fois que le client a inséré son ticket, la borne de contrôle vérifie s'il s'agit d'un abonné ou d'un client de passage à l'aide de la méthode verifierTypeTicket(). Si c'est un abonné, la borne enregistre l'heure de son départ et lève automatiquement la barrière avec la méthode leverBarriere(). Si c'est un client de passage, il vérifie les informations de paiement sur le ticket (méthode verifierTicketSortie()): si le ticket peut être lu et validé, la barrière est levée, sinon le client est invité à échanger avec un employé du parking via l'interphone présent dans la borne de contrôle. L'employé du parking peut alors vérifier le ticket et activer la barrière manuellement.

Au moment du passage de la voiture sur le senseur de fermeture, qu'il s'agisse d'un client de passage ou d'un abonné, la barrière se ferme et le compteur du parking est décrémenté.

C. Système de gestion

1. Abonnement

Les abonnements sont enregistrés dans la classe abonnement. Le système d'abonnement va tous les jours vérifier si la date d'expiration est dans deux semaines ou non par rapport à la date du jour. Suite à cette vérification, le système envoie au client un formulaire de réabonnement si besoin. Nous avons considéré que le système pouvait le faire sans vérification d'un employé parce qu' avec une vérification pour chaque client cela serait une perte de temps qui est censé être annulé grâce au système d'abonnement.
En plus d'une vérification de date, le système d'abonnement gère la modification de la nouvelle date d'expiration suite à un réabonnement. Il fait cela grâce à sa fonction fixerDateExpiration().

2. Gestion des événements

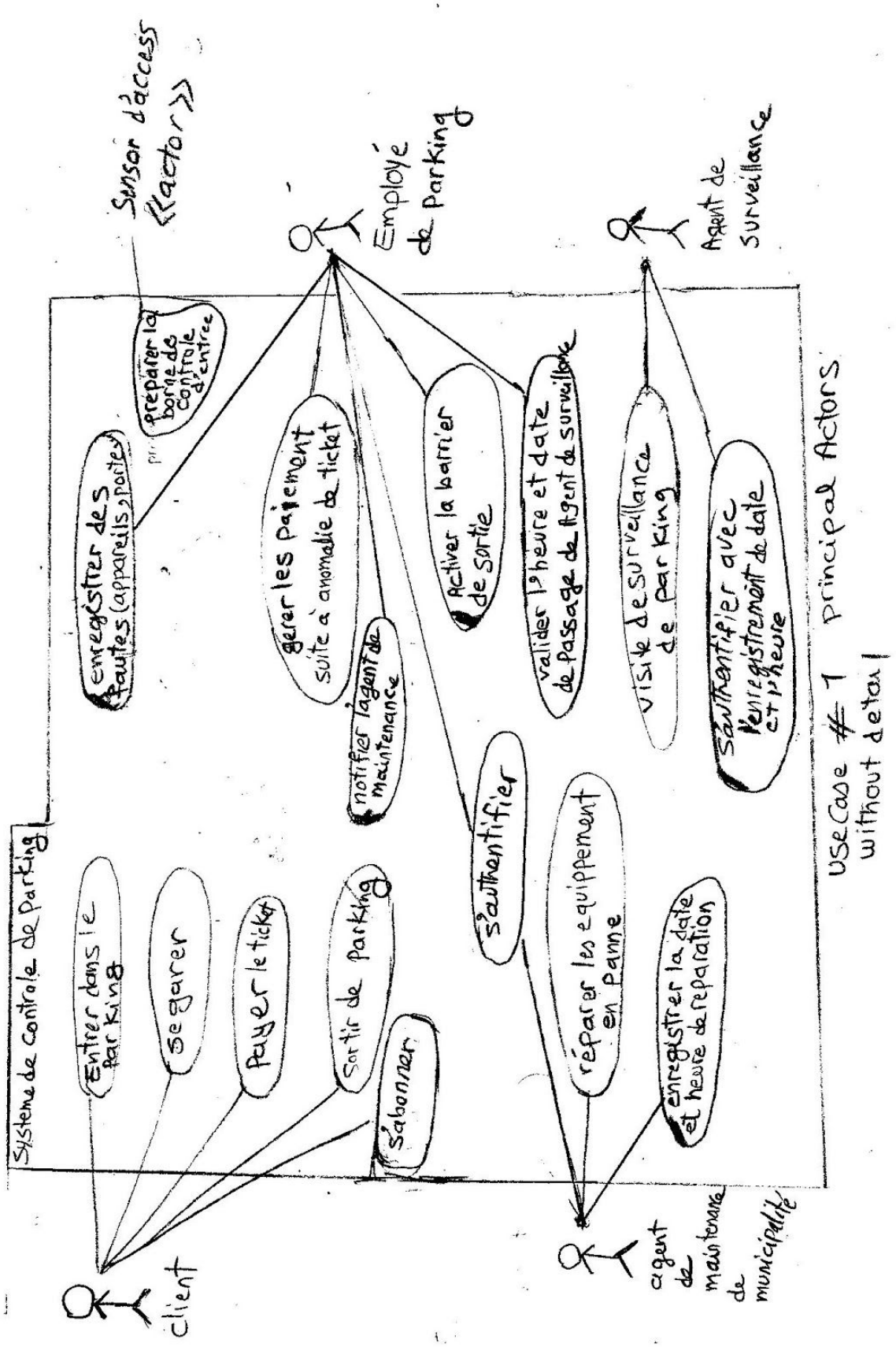
2.1 Gestion des fautes

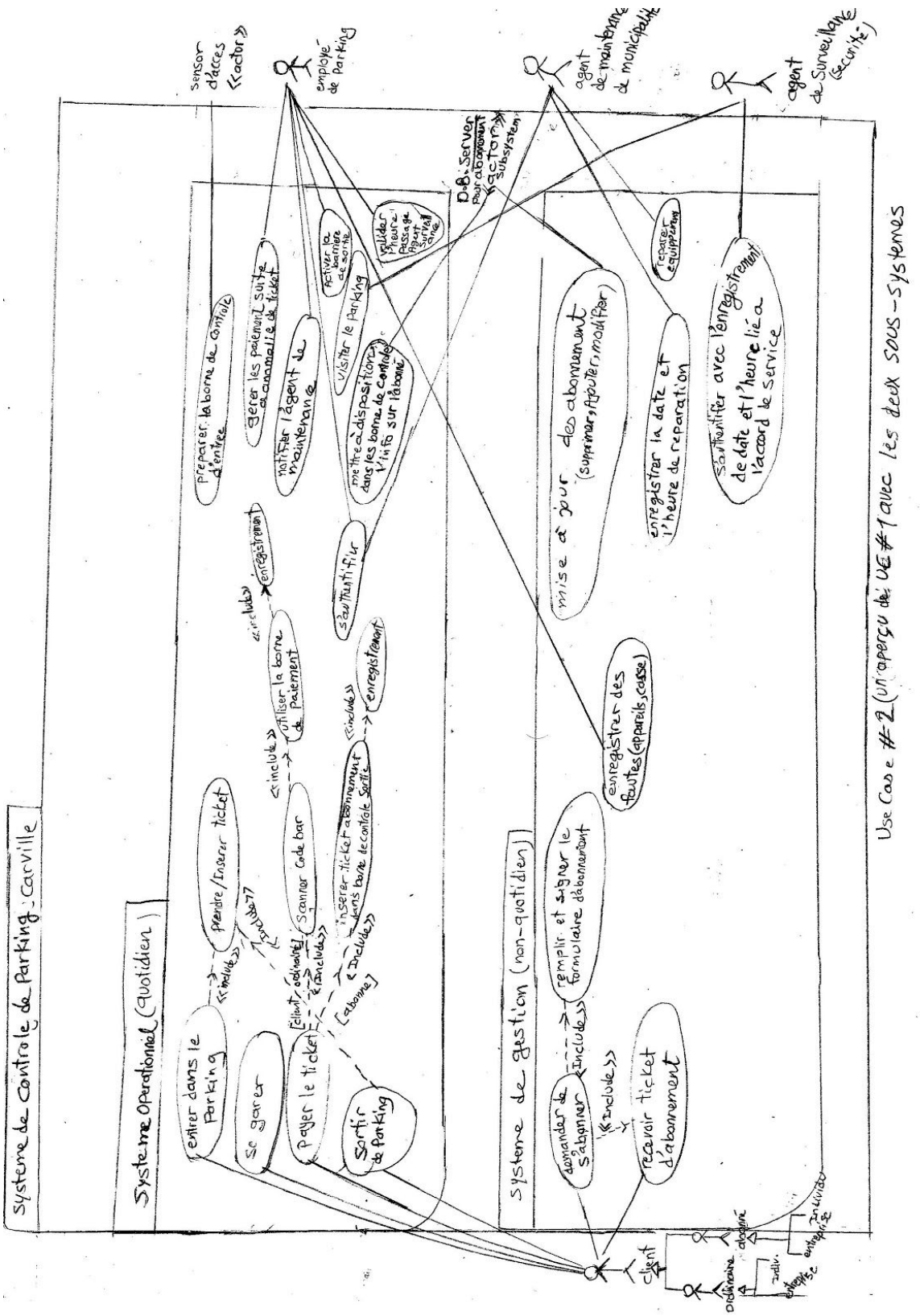
La gestion des fautes comprend les dégradations de matériel, nous avons donc décidé que les fautes sont repérées par les employés du parking et que les employés de parking s'occuperont de la procédure à suivre.
En effet, un employé de parking va repérer la faute qu'il y a dans le parking, l'enregistrer et contacter la municipalité. C'est pour cela que nous avons muni l'employé du parking de différentes fonctions comme notifierMunicipalité() ou bien enregistrerFaute().

2.2 Gestion des failles

La gestion des failles consiste à repérer les failles de contrat par rapport aux visites de sécurité. Ce processus consiste à vérifier que le contrat est bien respecté et enregistrer les cas où le contrat n'a pas été respecté.
Nous avons décidé de faire un diagramme de séquence pour ce processus car il nous a semblé important de détailler les vérifications que le pointeur automatique va devoir faire à chaque visite de sécurité et aussi en fin de journée.
Ce processus commence lorsque le garde de sécurité arrive. Il insère sa carte, dans le diagramme il n'est pas détaillé que lorsque le garde introduit sa carte c'est alors le pointeur automatique qui va utiliser recoitTicket(). Le pointeur enregistre donc l'heure d'arrivée. Pour le départ de l'agent c'est le même schéma qui se reproduit. Le garde de sécurité peut partir, cependant le processus continue. C'est à ce moment que le pointeur automatique vérifie que le temps de visite a été respecté, calculerTempsParViste(). S'il n'y a pas d'erreur à enregistrer, le processus s'arrête jusqu'à la prochaine visite de sécurité ou bien jusqu'à la fin de la journée. S'il y a une erreur, la faille est enregistrée. Lorsque la journée se finit, le nombre de visites de sécurité est vérifié. Lorsque le nombre de visites est inférieur au nombre stipulé dans le contrat, la faille est enregistrée, sinon le nombre de visite de la journée (VisiteParJour) est remis à zéro pour le lendemain.

ANNEXES


Sensor d'access
<<factor>>
Employé de Parking
Agent de surveillance
Système de contrôle de Parking
Entrer dans le parking
se garer
Payer le ticket
Sortir de parking
s'abonner
client
enregistrer des pautes (appareils, portes)
préparer la borne de contrôle d'entrée
gérer les paiement suite à anomalie de ticket
notifier l'agent de maintenance
s'authentifier
Activer la barrier de sortie
valider l'heure et date de passage de l'agent de surveillance
réparer les equippement en panne
enregistrer la date et heure de reparation
visite de surveillance de parking
s'authentifier avec l'enregistrement de date et l'heure
agent de maintenance de municipalité
UseCase # 1 principal Actors without detail



Use Case #2 (un aperçu de UC #1 avec les deux sous-systèmes

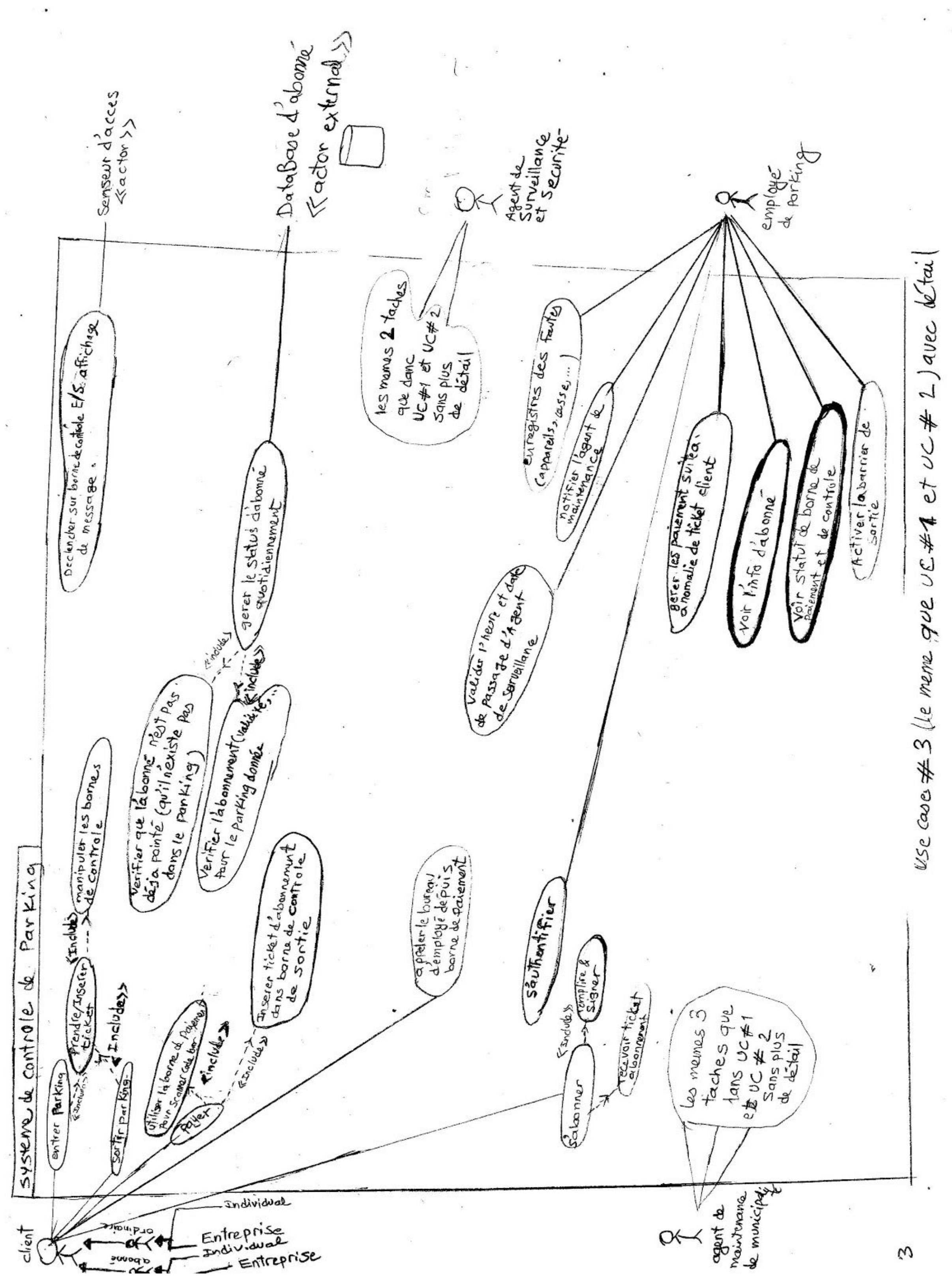

Systeme de controle de Parking
Senseur d'acces
«actor»
DataBase d'abonné
«actor external»
Agent de Surveillance et Securite
employé de Parking
client
agent de maintenance de municipalité
Individual
Entreprise
entrer Parking
sortir Parking
Prendre/Inserer ticket
manipuler les bornes de controle
«include»
payer
Verifier que l'abonné n'est pas déjà pointé (qu'il n'existe pas dans le parking)
Verifier l'abonnement (validité, ...) pour le parking donné
gerer le status d'abonné quotidiennement
Declencher sur borne de controle E/S affichage de message
Inserer ticket d'abonnement dans borne de controle de sortie
appeler le bureau d'employé depuis borne de paiement
s'authentifier
s'abonner
remplire & signer
recevoir ticket abonnement
Valider l'heure et date de passage d'Agent de Surveillane
notifier l'agent de maintenance
enregistres des fautes (appareils, casse, ...)
gerer les paiement suite à anomalie de ticket client
voir l'info d'abonné
voir statut de borne de paiement et de controle
Activer la barrier de sortie
les memes 2 taches que danc UC#1 et UC#2 sans plus de détail
les memes 3 taches que dans UC#1 et UC#2 sans plus de détail
Use cases #3 (le meme que UC#1 et UC#2) avec détail
3

Diagramme d’activité : Entrée
Scénario : places restantes

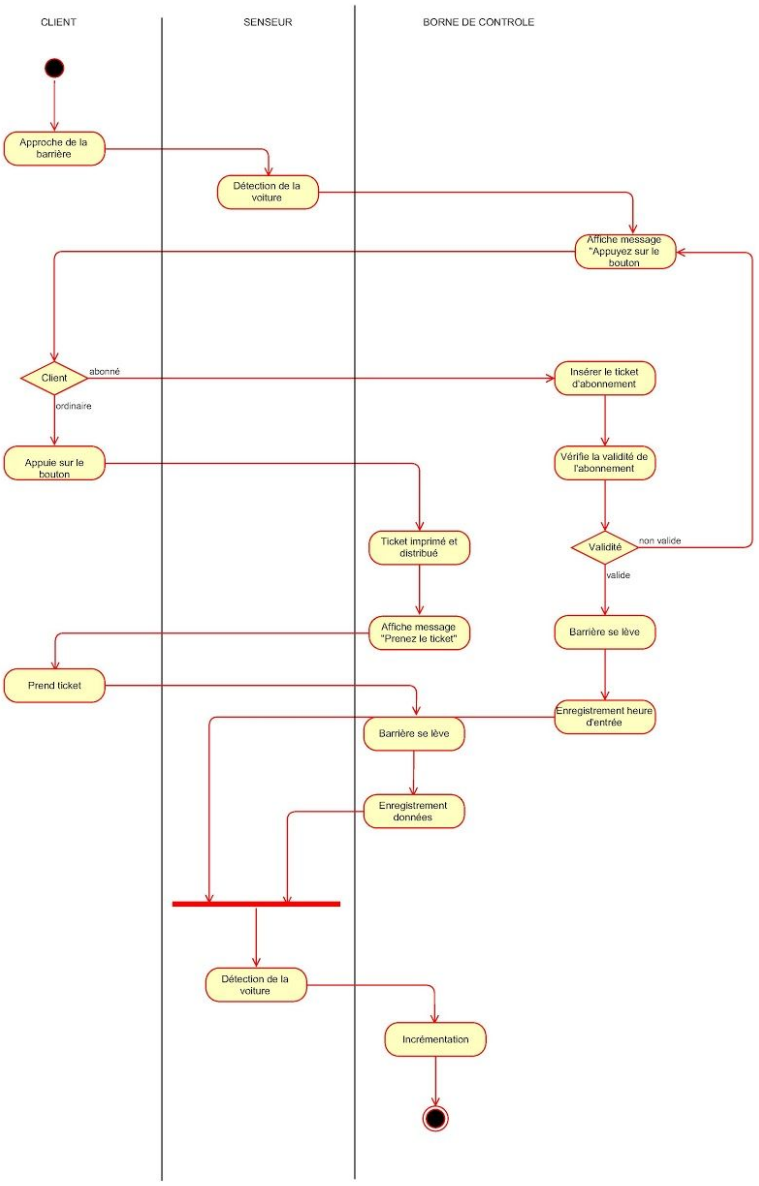


Diagramme d’activité : Entrée
Scénario : Parking complet

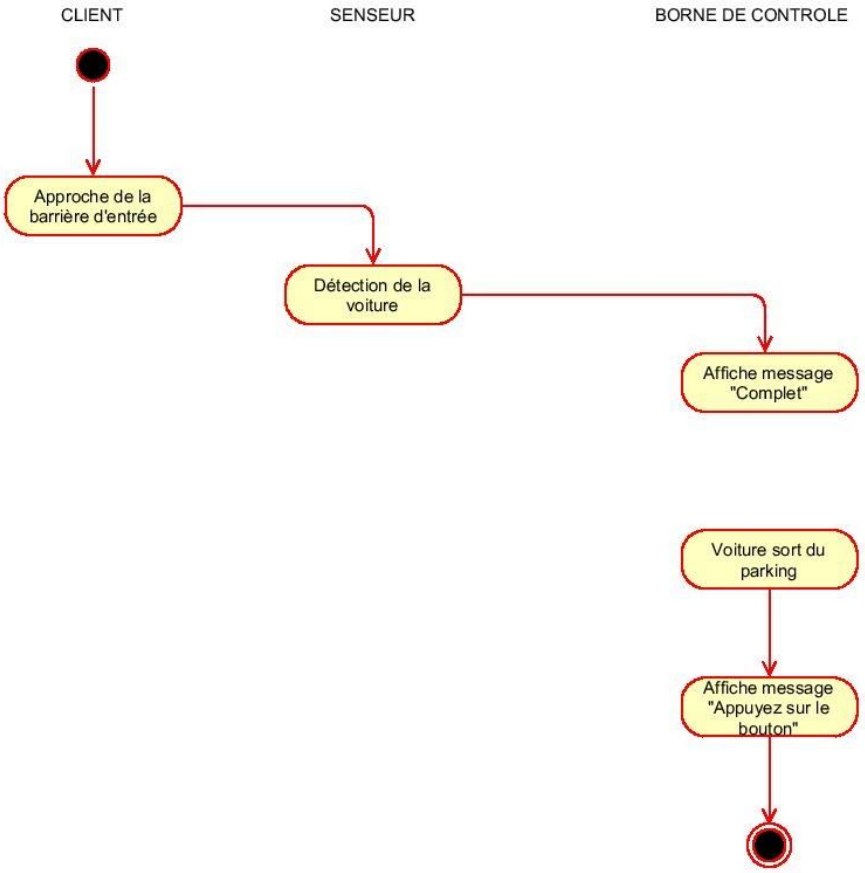


Diagramme de séquence : Entrée
Scénario : Places restantes

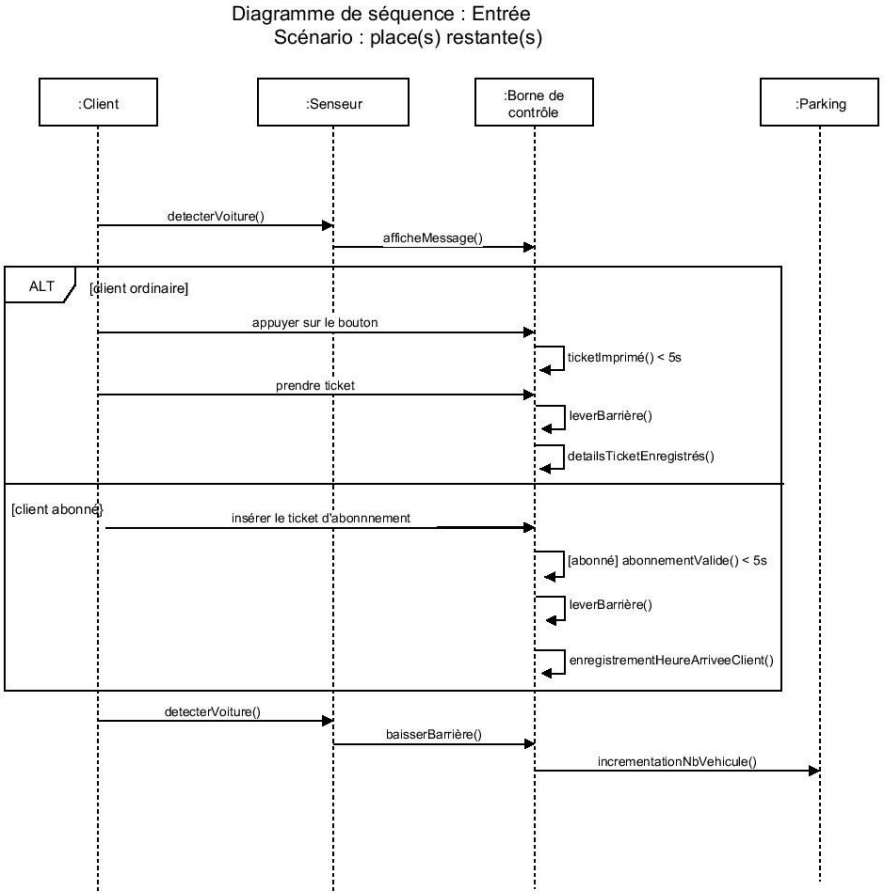


Diagramme de séquence : Entrée
Scénario : Parking complet

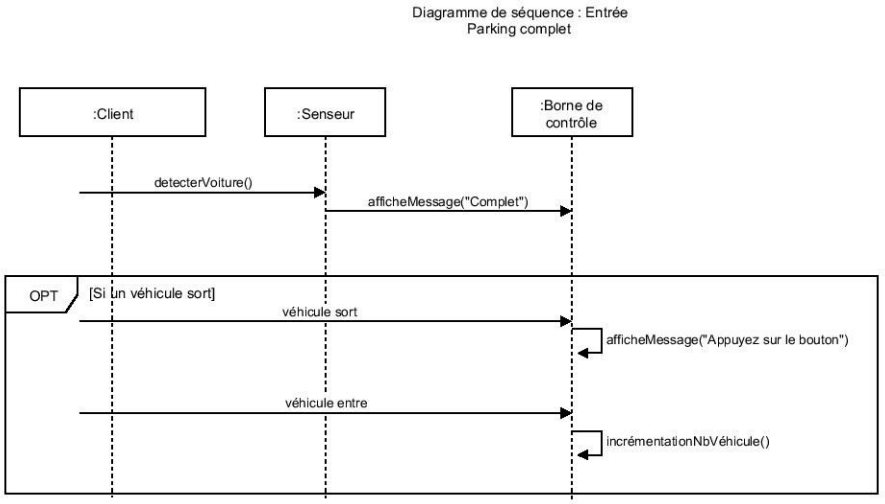


Diagramme de séquence du paiement
scénario : cas normal

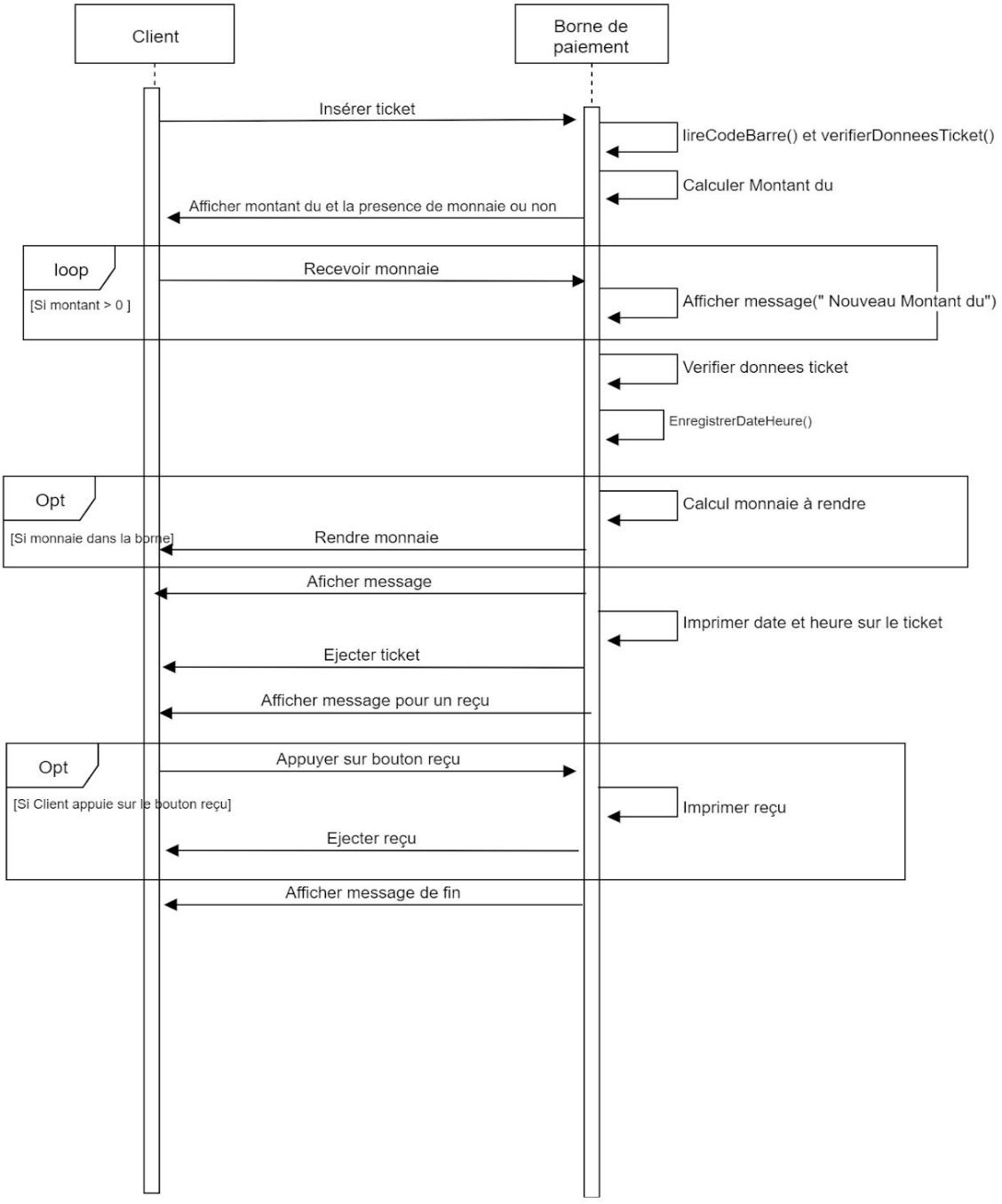


Diagramme de séquence de paiement
Scénario : cas où le ticket n’est pas bien lu et est rejeté

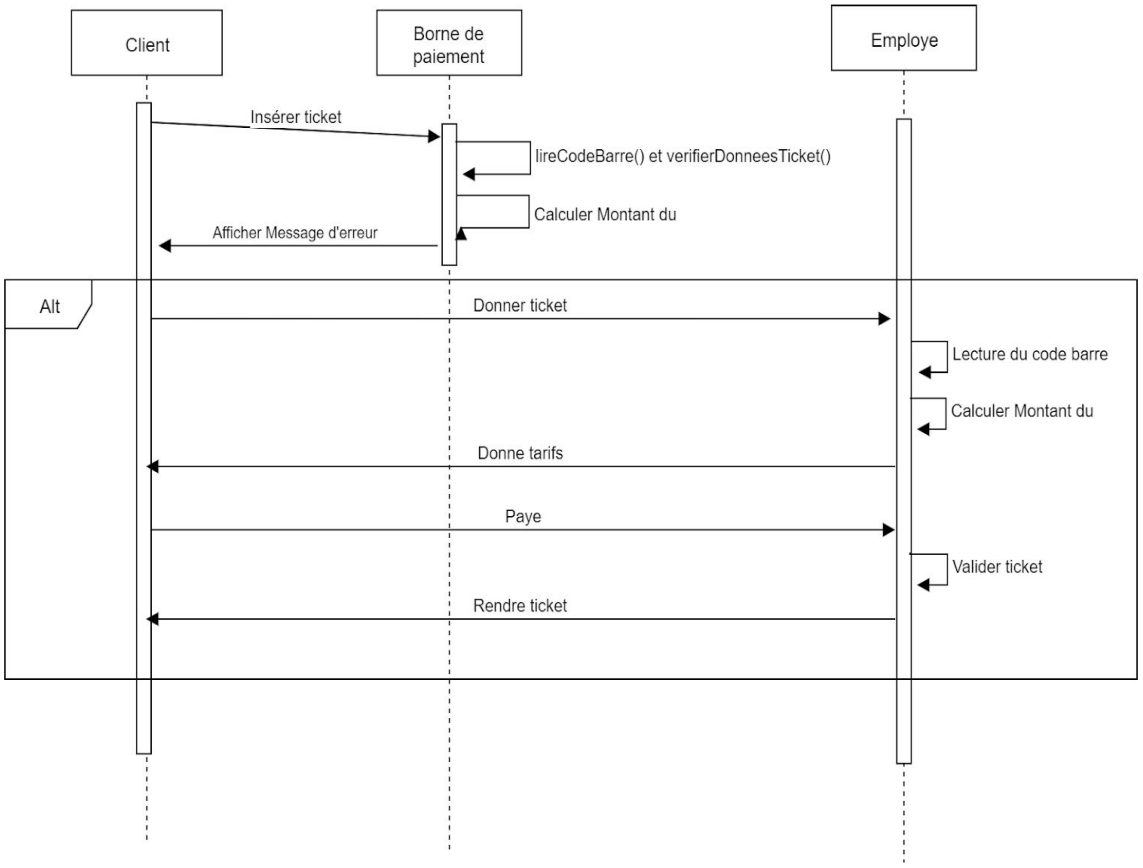


Diagramme d’activité du paiement

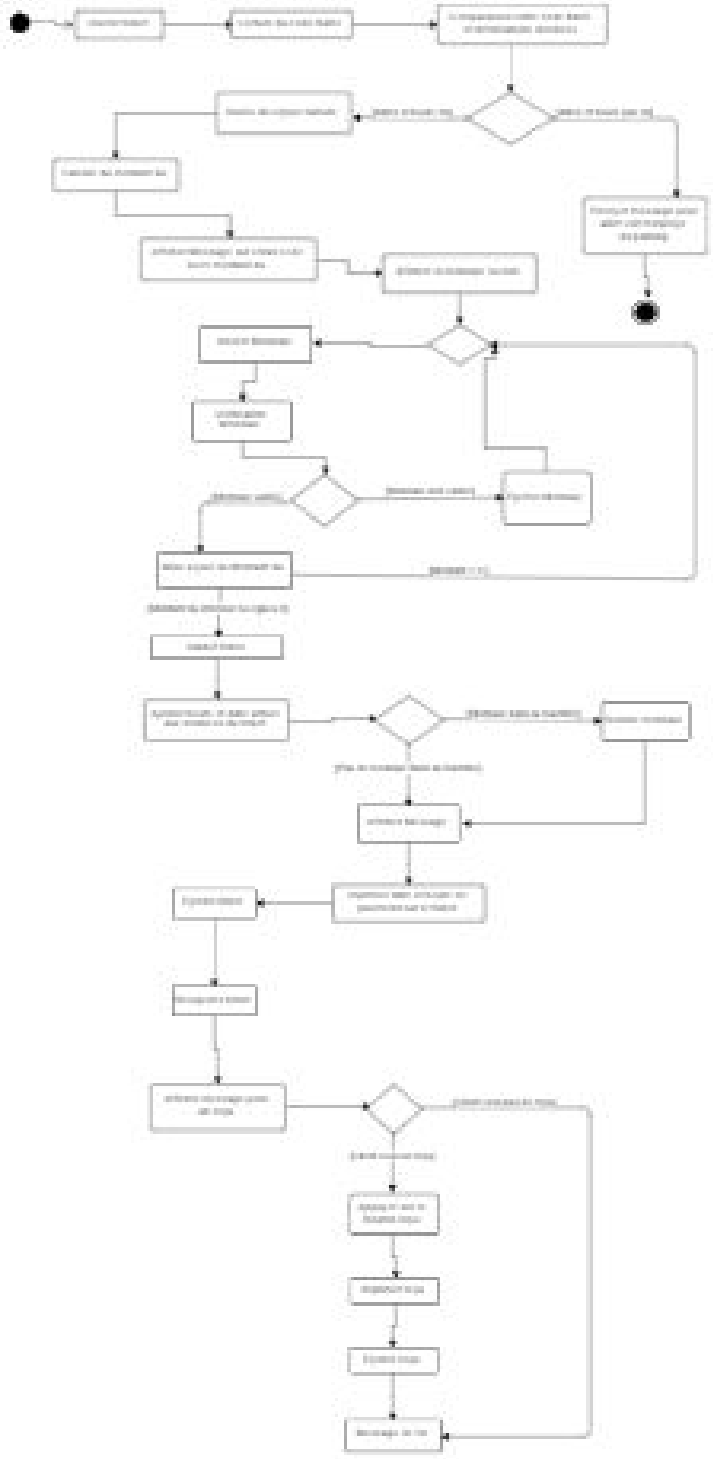


Diagramme d’activité de la sortie du parking

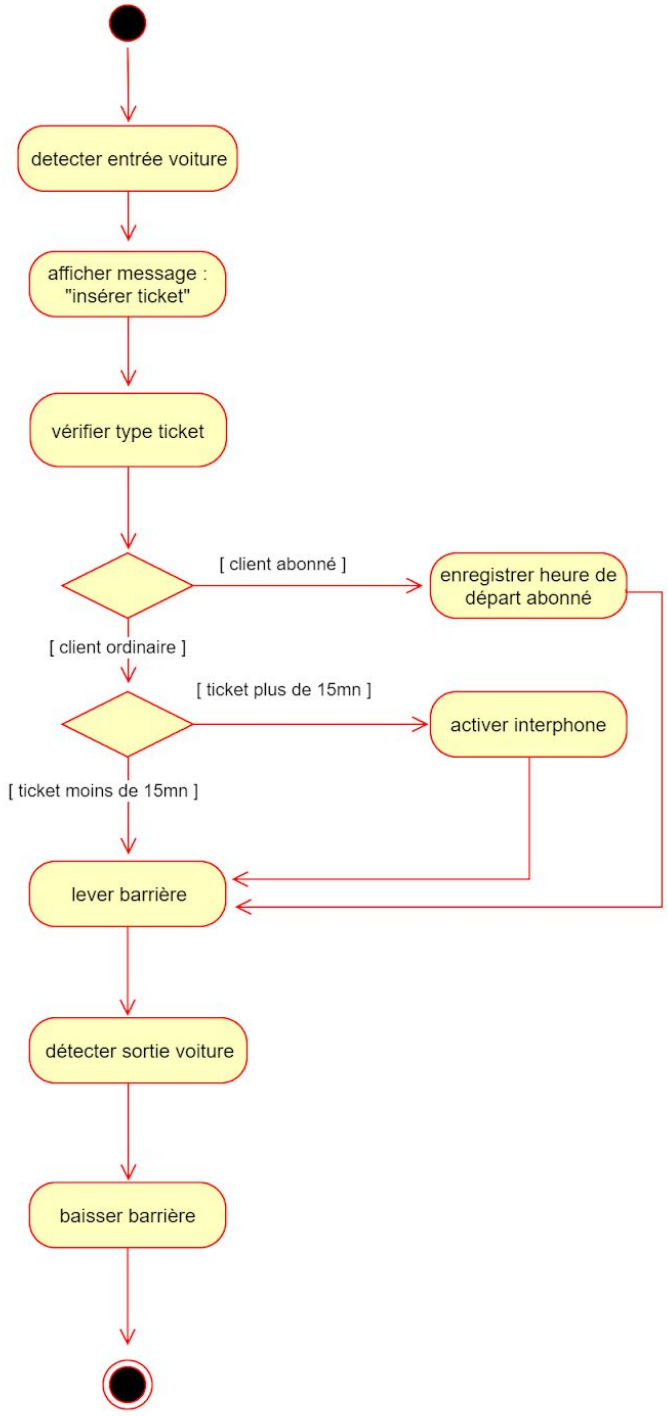


Diagramme d'activité de la sortie du parking

Diagramme de séquence de la sortie du parking

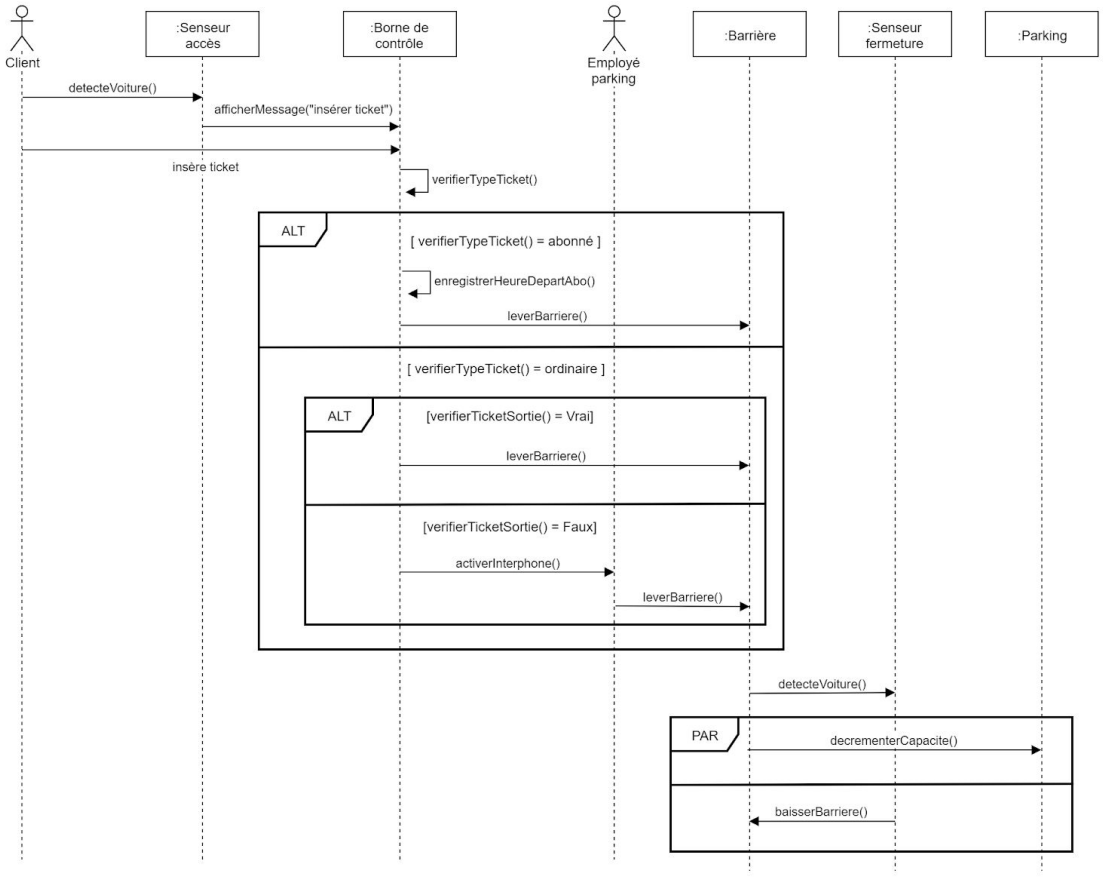


Diagramme de séquence du scénario de sortie du parking

Diagramme de séquence faille Visite de sécurité

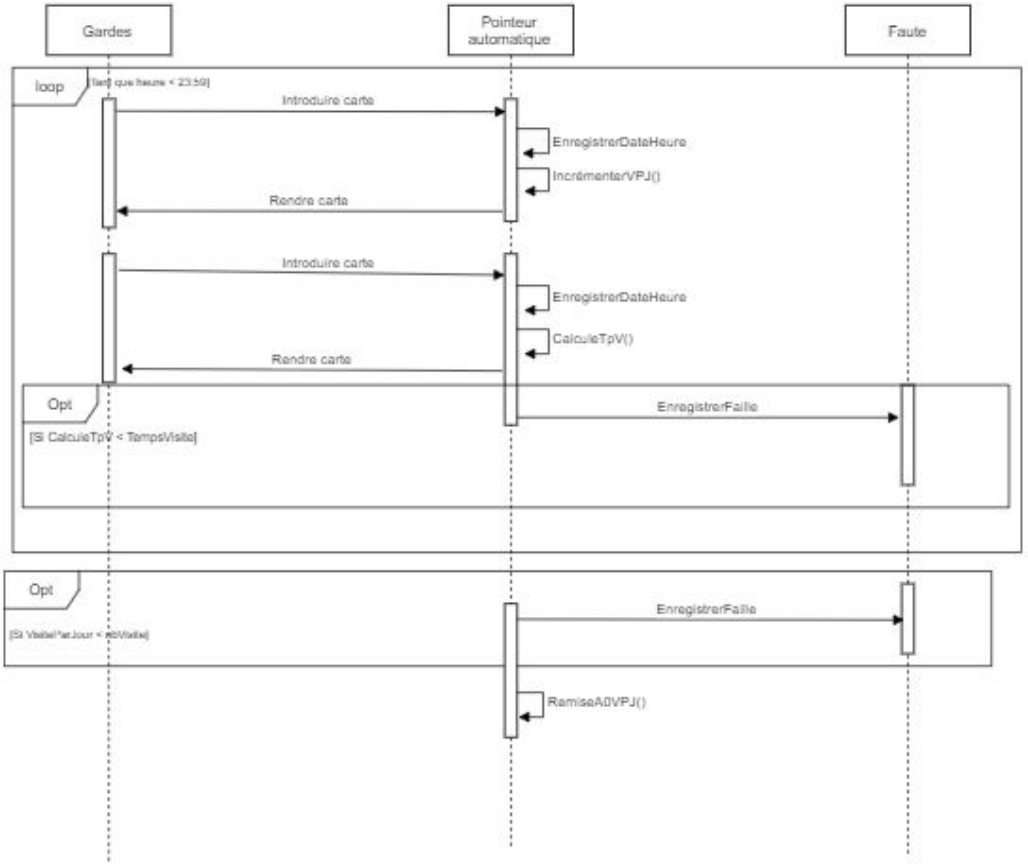


Diagramme d’activité Abonnement