Rapport Architecture Logicielle #1

CastexSki

Semestre 8 - 2021

Équipe C

BERTOLOTTO Loïc
FACQ Antoine
LECAVELIER Maëva
MAZURIER Alexandre
POUEYTO Clément

Université Côte d'Azur
Polytech Nice Sophia
SI4 - ISA

Sommaire

Introduction 3
Diagramme de cas d'utilisation 4
Les clients 5
Les acteurs professionnels de la station 5
Perchiste 5
Caissier 5
Employé 5
Les services externes au système 5
Précisions sur l'achat de forfaits/cartes 5
Diagramme de classe des objets métiers 6
Diagramme de composants 8
Interfaces pseudo-code 10

Introduction

Les différents diagrammes ont été réalisé en fonction de ces scénarios :

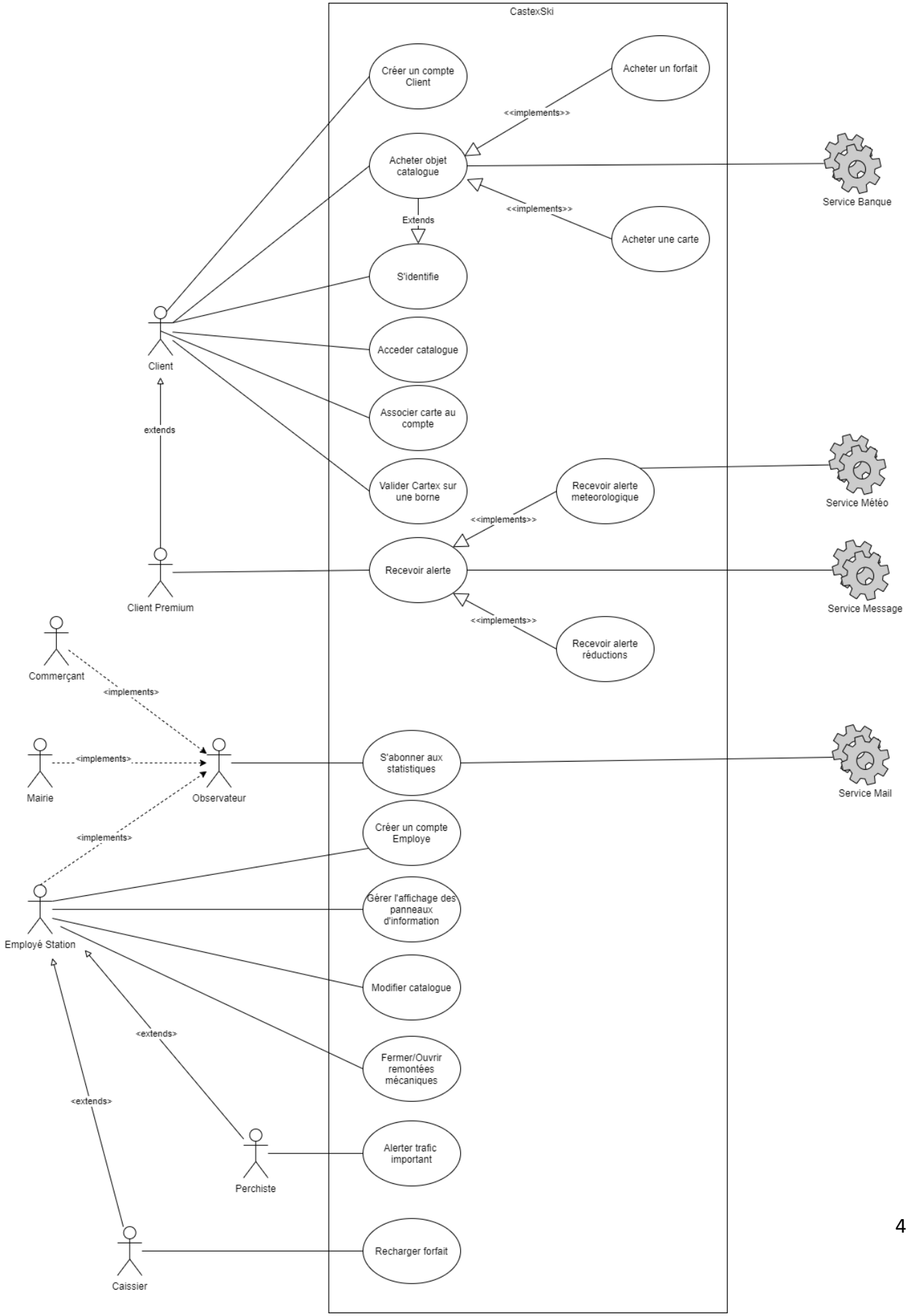
- Un client achète une carte sur Internet
- Un client achète un forfait sur Internet
- Un client achète un forfait et une carte en borne/caisse physique
- Un client accède au catalogue
- Un client crée son compte
- Un client s'identifie
- Un client associe une carte à son compte
- Un client devient premium
- Un client premium reçoit les alertes météorologiques
- Un client premium reçoit les alertes promotions

- Un caissier recharge un forfait pour un client
- Un employé de la station change le message d'un panneau
- Un perchiste alerte d'un trafic important
- Un employé crée son compte
- Un employé modifie le catalogue
- Un employé gère l'état des stations

- Les observateurs (mairie, employés, commerçants) s'abonnent et reçoivent le mail des statistiques

Notre réflexion et nos choix se sont orientés afin de répondre à ces scénarios de la façon la plus simple possible.

Diagramme de cas d'utilisation



Notre diagramme de cas d'utilisation permet de visualiser les différentes interactions entre les utilisateurs, et notre système d'information, Castex Ski. Parmi eux, on trouve :

Les clients

Ils peuvent dans un premier temps créer un compte, avec lequel ils peuvent acheter ou recharger des forfaits de ski. Une fois qu'ils disposent d'une Cartex, ils peuvent la valider aux bornes présentes au départ des remontées mécaniques de la station. Un client peut souscrire au programme premium (SuperCartex), lui permettant notamment de recevoir des notifications quant à la météo en station, et d'éventuelles réductions.

Les acteurs professionnels de la station

Ils peuvent, grâce à notre système, recevoir des statistiques sur la fréquentation de la station.

Perchiste

Il peut alerter sur la fréquentation de la remontée de laquelle il est responsable.

Caissier

Les actions d'achat / recharge qui peuvent être réalisées par les clients à travers un compte en ligne, peuvent aussi être réalisées par un caissier.

Employé

Un employé peut modifier les objets du catalogue comme le prix ou la disponibilité. Il peut modifier le message affiché pour un panneau donné et gérer l'ouverture d'une remontée mécanique.

Les services externes au système

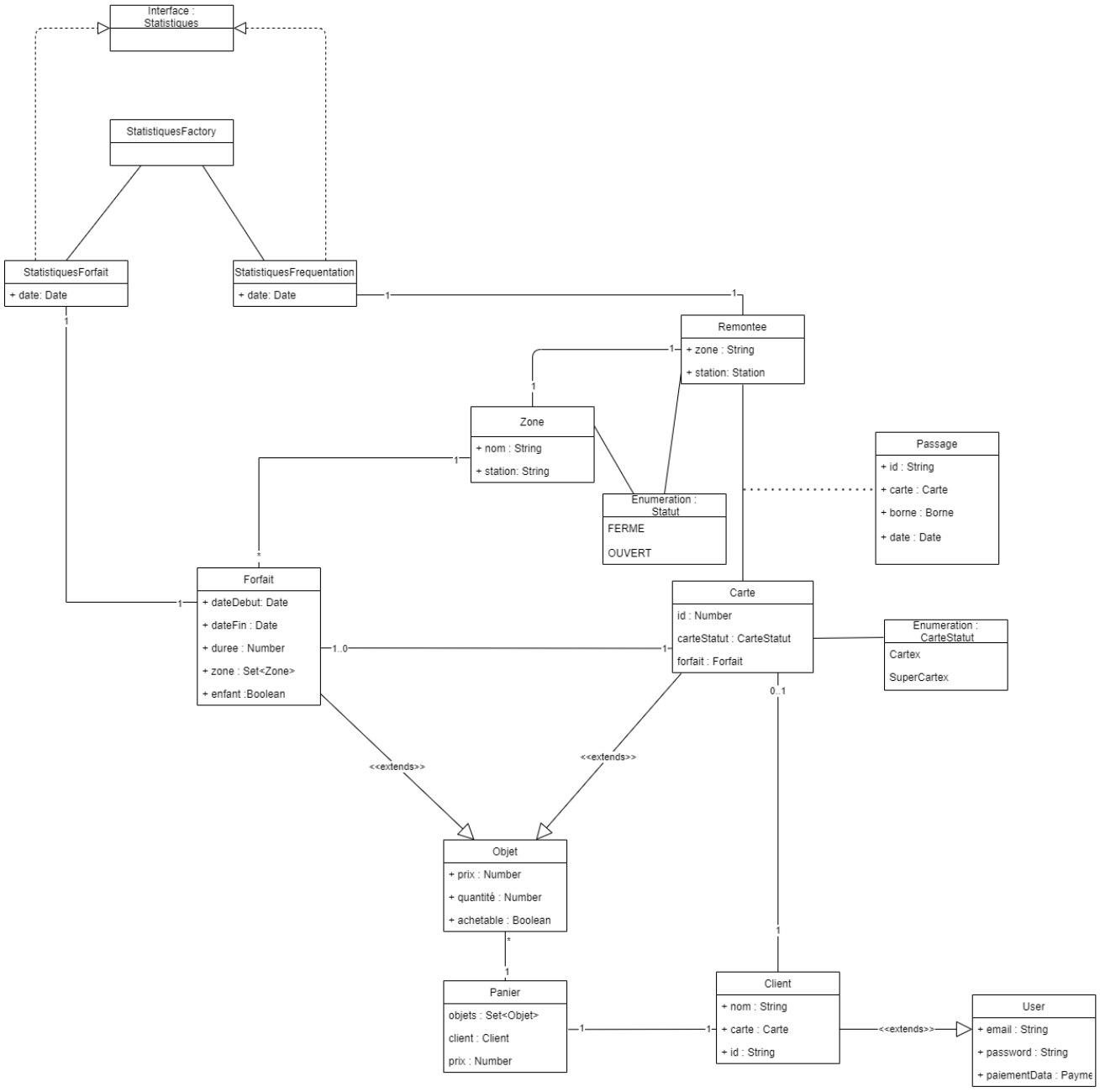
Notre système fera appel à différents systèmes extérieurs. On compte parmi eux un service bancaire, afin de valider les achats, une API de météo afin d'envoyer un bulletin météorologique aux utilisateurs premium. Enfin, tous les envois de messages seront gérés grâce à un service de messagerie.

Précisions sur l'achat de forfaits/cartes

L'achat de cartes et de forfaits, pouvant être réalisé depuis l'interface Web et physiquement aux caisses des stations, le scénario "Acheter objet catalogue" **étend** le scénario "s'identifier" dans le cas où l'on passe par le web (et donc ne l'**inclut** pas).

Le scénario « Acheter une carte » comprend notamment le scénario de l'achat une CarteX et celui de « devenir premium ».

Diagramme de classe des objets métiers



Pour la gestion et la création des statistiques, nous avons choisi d'implémenter un pattern *Factory.* En effet, par la nature différente des statistiques que nous souhaitons créer et exposer aux différents acteurs, ce pattern est intéressant car il va nous permettre de construire les statistiques en fonction du type de statistiques souhaité. De plus, ce pattern est idéal pour l'extensibilité si on souhaite ultérieurement rajouter de nouveaux types de statistiques.
Ici nous avons les statistiques suivantes :

- *StatistiquesFrequentation* : ces statistiques correspondent à l'affluence aux seins de la station (nombre de passages par jour pour une remontée mécanique). Celles-ci sont envoyées automatiquement par mail aux différents acteurs qui y sont abonnés (commerçants, mairie, …).
- *StatistiquesForfait* : ces statistiques correspondent aux statistiques de vente de forfait au sein de la station. Ces statistiques sont utilisés en interne pour adapter la stratégie marketing.

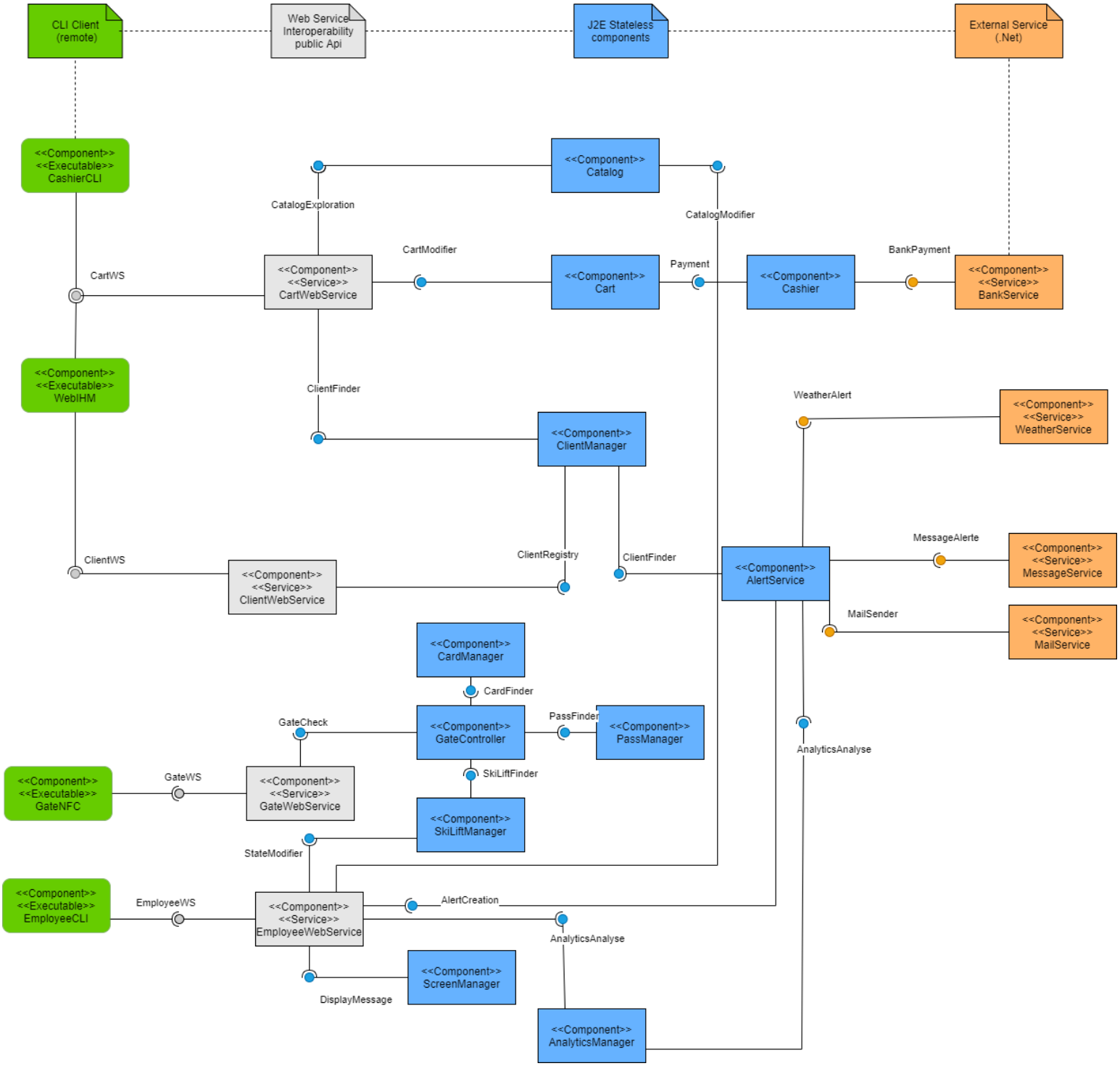
Pour la gestion de l'achat de *Forfait* et de *Carte* (pouvant être au statut *Cartex* ou *SuperCartex*) par des clients (identifié grâce à la classe *Client* ou non), les *Objets* à acheter sont stockés dans le *Panier*.

Les différents *Forfaits* sont dépendants d'une ou plusieurs *Zones* géographiques, qui peuvent être *Fermée* ou *Ouvertes* dans lesquelles sont regroupées plusieurs *Remontée* mécaniques. Lorsqu'un skieur scanne sa *Carte* au niveau d'une remontée mécanique, son *Passage* (association entre une *Carte* et une *Remontée*) est enregistré et par la suite traité dans les *StatistiquesFréquentation*.

Piste d'automatisation :

Actuellement nous avons choisi que des acteurs humains envoient les messages à afficher sur les différents panneaux d'affichages présents sur les bords de pistes. Dans une optique d'automatisation des messages à afficher sur ces différents panneaux, nous avons pensé à implémenter un pattern Strategy, avec une stratégie qui affiche des Itinéraires conseillés ou au contraire une stratégie qui va afficher des itinéraires déconseillés.

Diagramme de composants



Notre diagramme de composants est découpé en quatre types d'entités :

- En vert, ce sont les différents intervenants externes qui vont utiliser notre système. Certains de ces clients seront représentés par des CLI, d'autres par des IHM.
- En gris, ce sont les WebServices qui sont l'interface accessible depuis l'extérieur et qui vont être utilisés par les clients en vert.
- En bleu, ce sont les composants qui forment la logique métier de notre système.
- En orange, les services externes à CastexSki utilisés par les composants.

L'interface CatalogExploration permet au client Web ou au caissier d'accéder à l'intégralité du catalogue de forfaits/cartes. Le CatalogModifier relie le Catalog et l'interface Web des employés, leur permettant de mettre à jour les objets du Catalog.

Le client ou le caissier pourront également modifier le panier de la commande courante via l'interface CartModifier, et valider et payer celui-ci via l'interface Payment. ClientFinder permettra de trouver le client associé à l'achat d'un forfait s'il existe. Le rechargement d'un forfait n'est donc pas mentionné dans ce diagramme puisqu'il revient à acheter un forfait avec une carte existante.

Pour l'accès aux remontées, le passage d'une carte devant une borne consulte le GateController via l'interface GateCheck. Celui-ci est en charge de retrouver la carte correspondante, de vérifier la validité du forfait et l'état de la remontée que le client souhaite emprunter, respectivement via les interfaces CardFinder, PassFinder et SkiLiftFinder.

L'état de la remontée peut être défini par un employé via StateModifier. Les actions des employés solliciteront aussi les interfaces AlertCreation, AnalyticsAnalyse et DisplayMessage respectivement pour lancer une alerte de distanciation, consulter les données d'analyse de la station, et afficher un message sur un panneau.

Dans notre modélisation de nos composants, les interfaces BankPayment, WeatherAlert, MessageAlerte et MailSender constituent nos moyens de communication avec les services externes

Interfaces pseudo-code

CartModifier

void addItem(client: Client, item: Item)

void removeItem(client: Client, item: Item)

Payment

void payByCb(client: Client)

BankPayment

boolean pay(paymentData: PaymentData)

ClientRegistry

void authenticateClient(user: User)

void registerClient(client: Client)

void associateCardToClient(client: Client, card: Card)

ClientFinder

void findById(id: String)

CatalogExploration

List<Item> getAllItems()

Item getItemById(id: String)

CatalogModifier

void modifyCatalogItem(id: String, newItem: Item)

void addCatalogItem(newItem: Item)

void deleteCatalogItem(id: String)

WeatherAlert

WeatherAlert getWeatherAlert()

MessageAlerte

void sendMessageAlert(alert: Alert, phoneNumbers: List<String>)

MailSender

void sendMailAnalytics(analytics: List<Analytics>, emails: List<String>)

CardFinder

Card getCardById(id: String)

PassFinder

Pass getPassById(id: String)

GateCheck

boolean isPassageValid(idCard : String, lift: SkiLift) // Piste spécifique (en fonction du forfait)

DisplayMessage

void showMessage(screenIds: List<String>, message: String)

AnalyticsAnalyse

List<Analytics> getAnalytics()

List<TrafficAnalytics> getTrafficAnalyticsByDay(day: Date)

List<PassAnalytics> getPassAnalyticsByDay(day: Date)

StateModifier

void setZoneState(idZone: String, statut: Statut)

void setSkiLiftState(idLift: String, statut: Statut)

SkiLiftFinder

SkiLift getSkiLiftById(id : String)

AlertCreation

void createTrafficAlert(lift: SkiLift, alert: TrafficAlert)