Plan type du rapport d'architecture ISA

Equipe M

Farineau Thomas
Kitabdjian Léo
Chelgham Zinedine
Bassy Ayman
ELFIGUIGUI Aymane

1. Introduction

Rappel rapide du sujet

Pour renforcer la concurrence des marchés locaux face aux grandes surfaces de la région, une initiative a été lancée consistant en une carte de fidélité multi-enseignes qui permet aux clients membres de cumuler des points lors de leurs achats dans les magasins partenaires, leur offrant ainsi des avantages supplémentaires en fonction de leurs besoins. L'objectif est d'inciter les consommateurs locaux à privilégier les marchés locaux et de favoriser les avantages des clients et des commerçants partenaires.

Points clés :

- Développer un système de carte multi-fidélités pour les commerçants
- Architecture logicielle réutilisable dans différentes villes
- Objectif de devenir leader sur le marché des cartes multi-fidélités en 18 mois
- Incitation à l'achat dans les commerces partenaires avec des avantages en contrepartie
- Gain de points proportionnels aux montants dépensés chez les partenaires
- Possibilité de virer un petit montant sur la carte pour des achats
- Déblocage du statut de Very Faithful Person (VFP) pour les clients ayant une utilisation hebdomadaire de la carte et perte du statut si une semaine passe sans avoir utilisé la carte
- Avantages institutionnels (venant de la collectivité) pour les clients VFP

Liste des hypothèses par rapport au sujet :

- N'importe qui peut consulter les offres, mais il faut s'inscrire pour en bénéficier.
- Possibilité d'être notifié en cas de changement d'horaire des magasins favoris (par e-mail ou par téléphone).
- Après 5 achats avec sa carte de fidélité, un client devient VFP dès qu'il réalise au moins une consommation par semaine.
- Liste d'avantages liés au VFP : ticket de transport en commun offert, réduction sur les places de parking.
- Scanner : périphérique utilisé par l'application pour lire le code barre, permettant son interprétation en éléments traitables par l'application.
- Pour bénéficier d'un avantage institutionnel, il faut l'activer sur l'application. Pour obtenir un cadeau en magasin, il faut également payer en points sur l'application, puis présenter la validation au magasin lors du paiement d'un bien.
- Un agent de la DSI peut créer et publier des sondages, et gérer les offres de la collectivité (suppression, ajout, modification).
- Les comptes dans agents de la DSI sont préenregistrés par un administrateur du système à la demande de la mairie. Les agents doivent passer par le portail de connexion de leur mairie pour accéder à l’application.

- Les commerçants doivent remplir un formulaire dans l'application pour que l'agent puisse créer leurs comptes après vérification de la légitimité et validité de la demande.

2. Cas d'utilisation

Acteurs primaires

- Utilisateur : il s'agit d'une personne quelconque qui ouvre notre application, celui-ci peut-être un futur Client, Commerçant. Le but est de donner envie à ces personnes de s'abonner au système en les laissant consulter tous ce qui a à gagner.

- Commerçant partenaire : un commerçant inscrit qui peut donc gérer la présentation de son commerce dans l'application (offres, horaires) et qui valide la carte fidélité des clients pour leur permettre de gagner des points avec leurs achats.

- Client : Utilisateur qui s'est inscrit sur l'application, il peut bénéficier des avantages de la carte de fidélité auprès de sa ville, et des commerces partenaires.

- Client VFP : Client ayant obtenu un statut particulier lui permettant de bénéficier d'avantages supplémentaires, celui-ci bénéficie des avantages institutionnels mis en place par les agents de la DSI qui sont plus attrayants.

- Chef du Département : Pilote le déploiement de la carte de fidélité, supervise la DSI

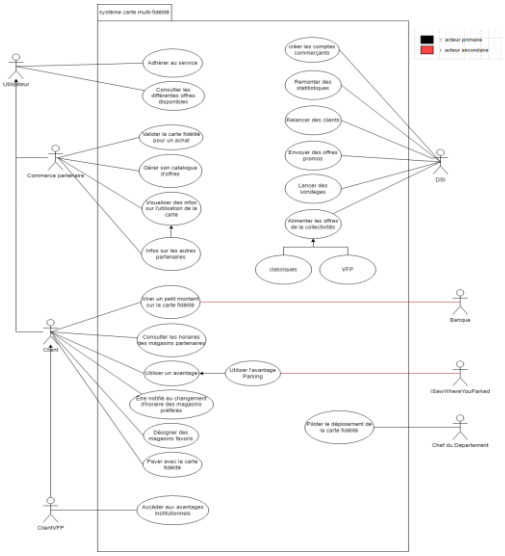
- DSI : Agent de la mairie chargé créer les comptes commerçants et de faire l'interface entre la mairie et l'application en gérant les offres de la collectivité, les relances, les sondages.

Acteurs Secondaires

- Banque : représente la banque du client lorsqu'il virera de l'argent vers sa carte de fidélité.
- ISawWhereYouParked : API partenaire utilisé pour gérer les stationnements places de parking

Diagramme de cas d'utilisation

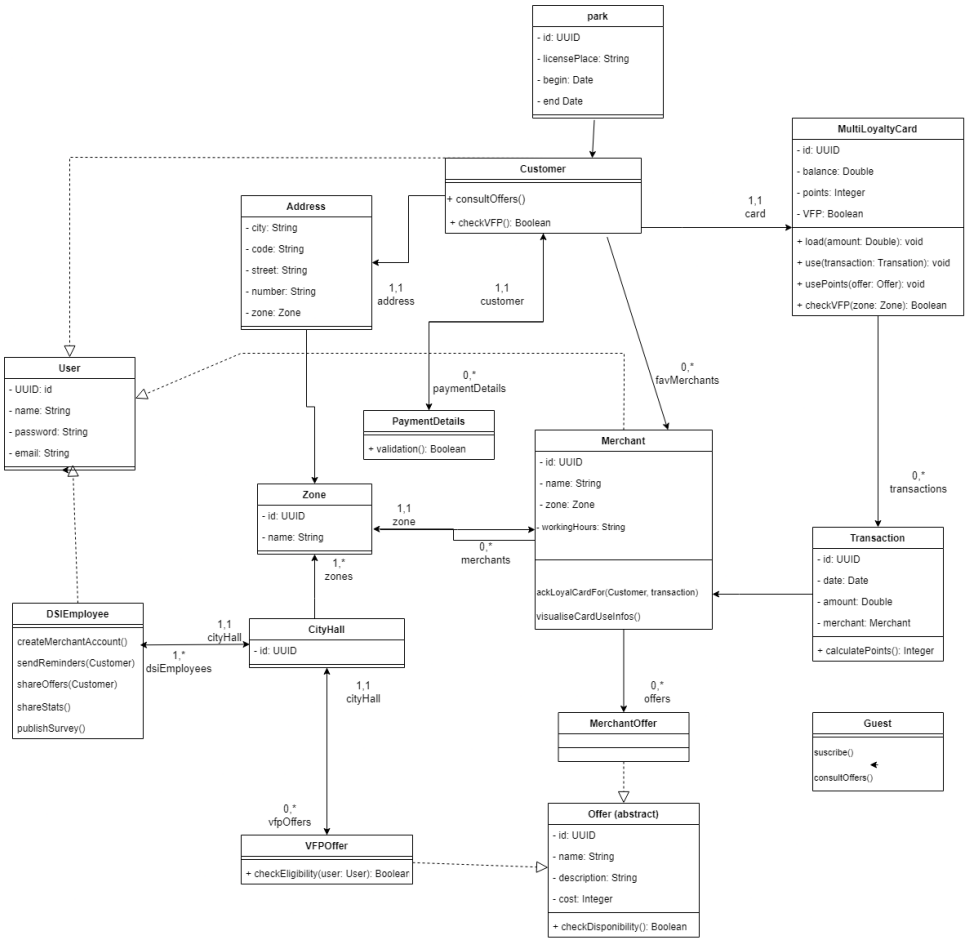
Diagramme de cas d'utilisation + explications de justification du découpage des cas.



https://app.diagrams.net/#G15XsJTiLfctq7XaztdOESxYBbRsVvG-v4

3. Objets métiers

Diagramme de classe :



https://app.diagrams.net/#G15XsJTiLfctq7XaztdOESxYBbRsVvG-v4

Explications :

Dans cette première analyse, on a essayé de faire le moins d'abstractions possibles en nous concentrant sur l'essentielle pour couvrir les fonctionnalités de notre MVP.

Pour les associations,

- MultiLoyaltyCard est liée à User, ce qui signifie qu'une carte multi-fidélité est détenue par un utilisateur spécifique.
- MultiLoyaltyCard est liée à Transaction, ce qui signifie qu'une carte multi-fidélité peut être utilisée pour effectuer des achats.
- Transaction est lié à Merchant, ce qui signifie qu'un achat est effectué auprès d'un commerçant spécifique.

- Merchant est lié à MerchantOffer, ce qui signifie qu'un commerçant peut offrir des avantages spécifiques aux clients fidèles.
- Merchant est lié à Zone, cela signifie qu'un commerçant est situé dans une zone spécifique.
- Offer est lié à MerchantOffer et VFPOffer, ce qui signifie qu'il existe deux types d'avantages différents, proposés par les commerçants et les institutions respectivement.
- VFPOffer est lié à CityHall, ce qui signifie que les avantages institutionnels sont offerts par une mairie spécifique.
- CityHall est liée à Zone, cela signifie qu'une mairie peut gérer plusieurs zones commerciales.
- User est lié à Address, ce qui signifie qu'un utilisateur a une adresse spécifique.
- Address est liée à Zone, ce qui signifie qu'une adresse est située dans une zone spécifique.
- DSIEmployee est liée à CityHall, ce qui signifie que plusieurs employés peuvent travailler dans une seule mairie

Ces relations permettent de mieux comprendre comment les différentes parties du système interagissent entre elles et comment les différents cas d'utilisation sont gérés. Par exemple, comment les utilisateurs peuvent obtenir des points en fonction de leur zone de résidence, comment les avantages sont vérifiés pour la disponibilité et l'éligibilité et comment les avantages commerçants et institutionnels sont gérés.

4. Interfaces

CustomerConnectionRegistration : Cette interface permet à un utilisateur de s'enregistrer ou se connecter.

```
public interface CustomerConnectionRegistration {

    boolean register(String name, String fullname, String email, String     mdp)
throws VerifFailedException;

    boolean connection (String email, String mdp) throws BadInformationException;
}
```

DsiConnectionRegistration : Cette interface permet l'enregistrement et la connexion d'un membre de la DSI. Elle va venir récupérer des informations concernant les membres de la mairie pour faire vérifier l'identité du membre de la DSI avec la clé de connexion.

```
public interface DsiRegistration {

    boolean register (String name, String fullname, String email, String mdp, int
connectionKey) throws VerifFailedException;

    boolean connection (String email, String mdp) throws BadInformationException;
}
```

Bank : cette interface du BankProxy va renseigner le BankService des données fournies par l'utilisateur et ainsi validité d'une transaction.

```
public interface Bank {

     boolean pay (Customer customer, double value) throws PaymentException;
}
```

ReloadedProcessor : cette interface permet au client de recharger sa carte de fidélité.

```
public interface ReloadedProcessor {

    boolean reloadCard (Customer customer, double value) throws
PaymentException;
}
```

Reload : cette interface dessert l'interface ReloadedProcessor, elle va enclencher l'action de paiement dans la procédure de paiement dans la recharge.

```
public interface Reload {

    boolean reload (Customer customer, double value) throws PaymentException;
}
```

PayementProcessor : cette interface permet au client de payer avec sa carte de fidélité.

```
public interface PayementProcessor {

    boolean validatePayement(MultiLoyalityCarte carte, Transaction transaction)
throws InternePaymentException;
}
```

Payement : dessert l'interface PayementProcessor en enclenchant l'action de payement dans la procédure de paiement avec la carte de fidélité, retirant ainsi le montant requis de la carte de fidélité du client ciblé.

```
public interface Payement {

	boolean pay (Customer customer, double price) throws
InternePaymentException;
}
```

CustomerFinder : permet de retrouver un client dans le repository des Clients

```
public interface CustomerFinder {

	Customer findById (double id) throws NotFoundException;

	 Customer findByName (String name) throws NotFoundException;
}
```

ShowStoreOffer : permet à un utilisateur invité ou non de voir les offres de magasin

```
public interface ShowStoreOffer {

	List<Offer> ShowStoreOffers (Store store) throws NotFoundException;
}
```

ShowStore : permet de voir les magasins partenaires

```
public interface ShowStores {

	List<Store> ShowStores (CityHall town) throws NotFoundException;
}
```

ShowInstitutionalAdvantage : permet de voir les offres institutionnelles de la collectivité

```
public interface ShowInstitutionalAdvantage {

	List<Offer> ShowInstitutionalAdv (CityHall town) throws NotFoundException;
}
```

MerchantCheck : fait le lien entre le composant du marchant et la base de données des marchands par le composant de la DSI, fait la vérification pour permettre ou non à la personne de devenir un marchant partenaire.

```
public interface MerchantCheck {

	boolean check(Zone adress) throws RejectedDemandException;
}
```

BecomePartApp: permet à un marchand de faire la demande pour devenir marchant partenaire

```
public interface BecomePartApp {

	boolean demand(Zone locationStore, CityHall town) throws
RejectedDemandException;
}
```

OfferManagement : permet à un marchand de mettre à jour sa liste d'offre.

```
public interface OfferManagement {
```

```
    boolean addOffer(Offer offer, Store store);

     boolean delOffer(Offer offer, Store store);
}
```

ScanBill : permet à un marchand de scanner la transaction d'un client

```
public interface ScanBill {

    boolean scan (Transaction bill) throws BadBillException;
}
```

ImpactAccount : cette interface permet de gagner des points sur son compte après un achat en utilisant sa carte de fidélité.

```
public interface ImpactAccount{

    boolean gainPoints(double amountPayed, Customer customer);
}
```

PayPointOffer: cette interface traite les cas de perte de point sur un compte après achat d'une offre.

```
public interface PayPointOffer{

    boolean payPoints(double amountPayed, Customer customer, Offer offre);
}
```

CityOfferManagement: cette interface permet à la DSI d'ajouter ou retiré des offres lier à la ville.

```
public interface CityOffersManagement{

    boolean addOffer(VFPOffer offer, CityHall town);

     boolean delOffer(VFPOffer offer, CityHall town);
}
```

StatManager : cette interface permet à la DSI de faire des calcules Statistique, comme par exemple récupéré les offres n'ayant pas eu de succès ou inversement via le nombre d'achat lier à celle-ci.

```
public interface PayPointOffer{

    boolean getWorstOffer(double maxValue);
     boolean getBetterOffer(double minValue);
}
```

BenefitAdvantage : cette interface permet au client de beneficier d'un avantage contre des points

```
public interface BenefitAdvantage{

    boolean useStoreAdvantage(Offer choisedAdv, Store store);
     boolean useCityAdvantage(Offer choisedAdv, CityHall town);
}
```

BenefitVFPAdvantage : cette interface permet au client de beneficier d'un avantage institutionnel contre des points

```
public interface BenefitVFPAdvantage{

	boolean UseVFPAdvantage(VFPOffer choisedAdv, CityHall town);
}
```

BenefitGift : cette interface permet au client de bénéficier d'un cadeau contre des points

```
public interface BenefitGift{

	boolean unlockGift(Offer choisedGift, Store store);
}
```

ParkAdvantage : interface qui permet d'activer sur le compte client les avantages liés au stationnement dans un parking de la ville

```
public interface ParkAdvantage{

	boolean parkFree(VFPOffer choisedGift);

	 boolean getReducPricePark(VFPOffer choisedGift);
}
```

TransportAdvantage: interface qui permet d'activer sur le compte les avantages liés au transport dans la ville

```
public interface TransportAdvantage{

	boolean FreeVoucher(VFPOffer choisedGift);
}
```

5. Composants

DSIManagementComposent : le DSIManagement gère les actions possibles par un agent de la DSI, il reçoit des demandes de modification sur les offres de la ville, (traité à l'aide de l'interface CityOfferManagement) et des demandes sur les statistiques utilisateur via l'interface StatManager. Le composant permet aussi de faire la vérification sur un marchant qui existe bel et bien dans la ville afin de l'ajouter au répertoire correspondant.

MerchantManagementComposent : le MerchantManagement gère les actions d'un commerçant partenaire, il reçoit des demandes de modification sur les offres du commerce, ou des informations liées au scan d'un ticket client et reçoit les demandes pour devenir commerçant partenaire utilisant l'interface MerchantCheck du composant de la DSI.

RegistryComposent : Un composant gérant les Connections et enregistrement suivant les demandes du RegistrationConnectionController pour les Utilisateurs et la DSI, dans le cas de la DSI, le composant fait une requête à une base de données des membres de la mairie fourni pour vérifier qu'il s'agit bien d'un membre de la DSI. Le composant reçoit aussi des demandes de récupération de client depuis le CreditCarteCustomerController

MoneyManagerComposent : Composant responsable de la gestion de l'argent sur un compte, reçoit des demandes de CreditCarteCustomerController afin de recharger une carte de fidélité via l'appel du service externe lié à la banque via le contrôler PayementComposant ou pour payer avec la carte de fidélité directement via l'interface PaymentProcessor qui sollicite le composant de payement via Payement. Il communique aussi avec le composant AccountManager dans l'appel de l'interface ImpactAccount pour attribuer des points au client dans le cas d'un payement avec la carte de fidélité.

PayementComposant : Composant qui gère le paiement, que ce soit dans le cadre du versement d'argent dans la carte de fidélité via le BankProxy, ou dans le payement en magasin avec la carte de fidélité.

DisplayOffer : Composant répondant à la demande d'affichage d'informations liées à l'offre en général par VisitAppDisplay. Représente la première interface commune à tous les utilisateurs qui leur permet de se renseigner sur les différentes offre (cadeaux, avantages commercial, avantages institutionnel) de l'application et sur les magasins (emplacement, horaires)

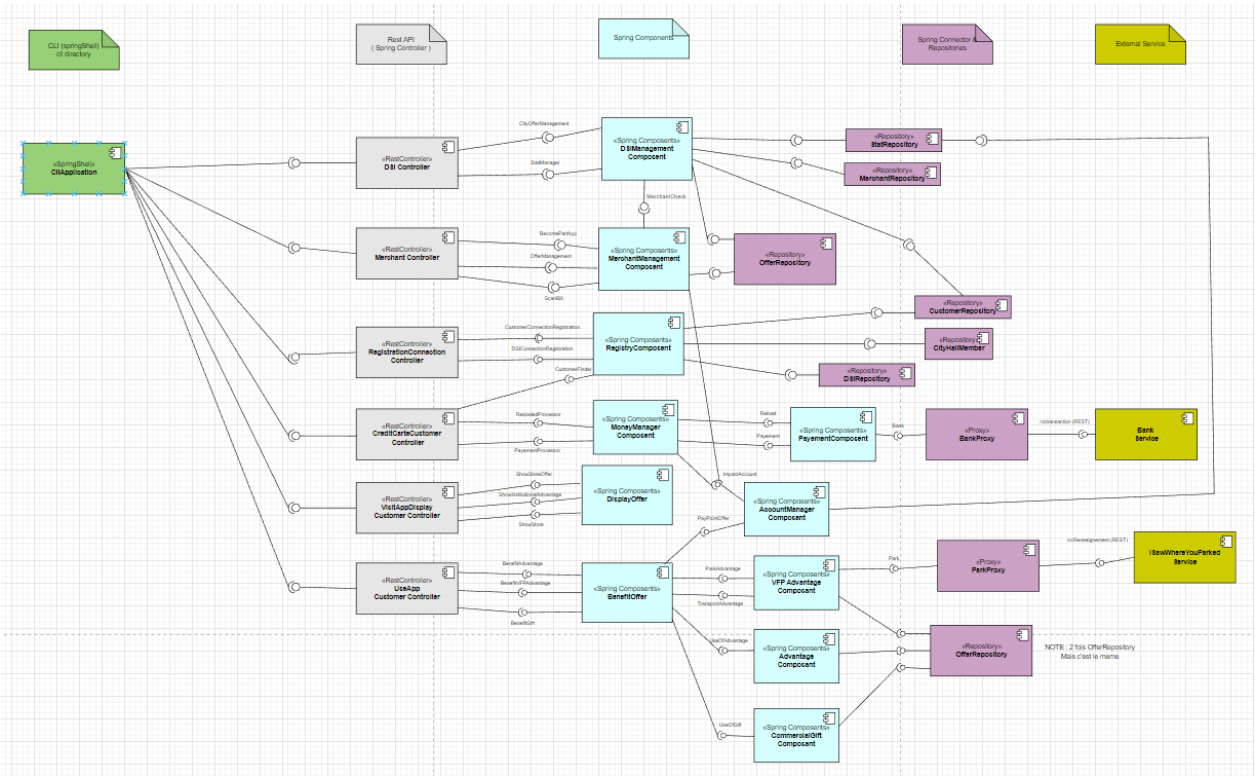
AccountManagerComposant: Représente le composant représentant les impacts des différentes actions sur le compte d'un client, par la communication avec MerchantManagement et MoneyManager, il répond à la demande d'ajout de points sur le compte d'un client suite à un achat effectuer, avec la carte de fidélité ou via le scan d'un marchand partenaire. Il Permet aussi le payement des offres en utilisant les points de la carte.

BenefitOfferComposant : Composant répondant aux demandes UseAppCustomerController, lorsqu'un client souhaite activer via son compte des offres contre des points, avec la demande vers le composant AccountManager pour le retrait de points.

VFPAdvantageComposant : Composant en lien avec BenefitOfferComposant dans la gestion de la logique des avantages institutionnel pour les clients avec le statut VFP. En lien avec ParkProxy via l'interface Park dans l'envoie de données à L'API ISawWhereYouParked.

AdvantageComposant : Composant en lien avec BenefitOfferComposant dans la gestion de la logique des avantages institutionnels ou commercial.

CommercialGiftComposant : Composant en lien avec BenefitOfferComposant dans la gestion de la logique des cadeaux.



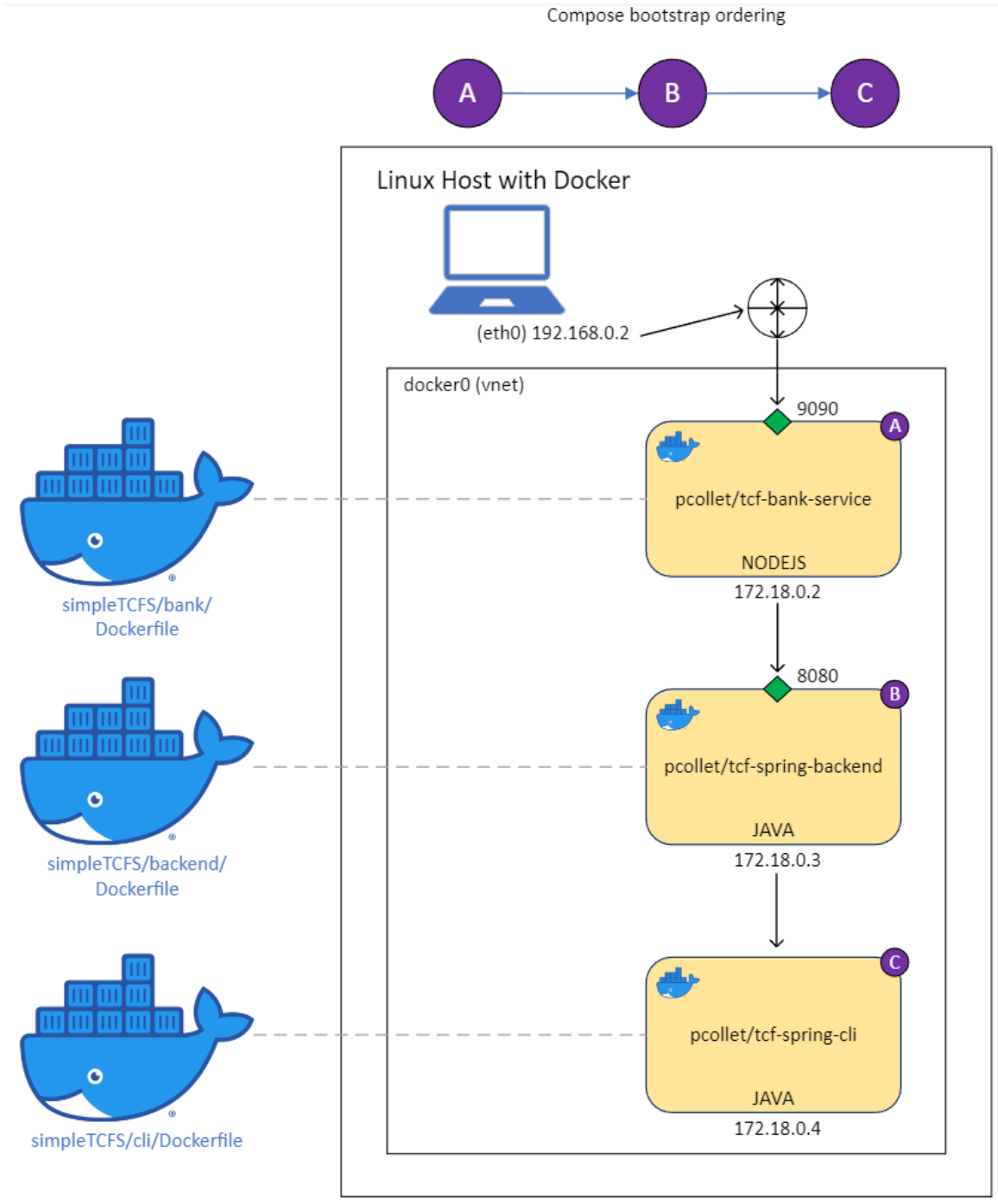
https://app.diagrams.net/#G1cNK55CGQYeLbuVM0bw3Gt_LwBwqxnl2g

6. Scénarios MVP

Nous envisageons un ensemble de scénarios pour construire un MVP et ensuite l'étendre pour couvrir l'ensemble des fonctionnalités de la carte multi-fidélité :

- S'inscrire ou se connecter pour le client et le commerçant, se connecter pour la DSI :
  (CLI -> RegistrationConnectionController -> RegistryComposent -> CustomerRepository)
- S'inscrire pour la DSI :
  (CLI -> RegistrationConnectionController -> RegistryComposent -> CityHallMember), puis après réponse valide, (RegistryComposent->DSIRepository)
- Consulter l'application :
  (CLI -> VisitAppDisplay -> DisplayOffer)
- Client recharge sa carte de fidélité :
  (CLI->CreditCarteCustomer->MoneyManagerComposent->PayementComposant->BankProxy-> BankService externe) (retour ok BankService->BankProxy->PayementComposant->MoneyManager)
- Client qui paye avec sa carte de fidélité :
  (CLI-> CreditCarteCustomer->MoneyManagerComposent->PayementComposant)
- Client qui gagne des points avec une transaction
  (CLI->MerchantController->MerchantManagementComposant->AccountManagerComposant)-
- Utiliser les points pour débloquer un avantage
  (CLI->UseAppCustomerController->BenefitOffer->AvantageComposant->OfferRepository)(retour jusqu'au front pour afficher le ticket de preuve d'achat en point en ligne à présenter)
- Utiliser les points pour acheter un cadeau
- (CLI->UseAppCustomerController->BenefitOffer-> CommercialGiftComposant ->OfferRepository) (retour jusqu'au front pour afficher le ticket de preuve d'achat en point en ligne à présenter)
- Utiliser les points pour débloquer un avantage de VFP, ici 30 min de parking gratuit
- (CLI->UseAppCustomerController->BenefitOffer->VFPAdvantageComposant->ParkProxy- >ISawWhereYouParked Service, renseignement sur les modalités lié au client envoyer à l'application) (retour positif ISawWhereYouParked Service->ParkProxy->VFPAdvantageComposant jusqu'au front pour afficher le ticket et validation lier à la place de parking gratuite)
- Le Commerçant souhaite modifier son catalogue d'offre
  (CLI->MerchantController->MerchantManagementComposent->OfferRepository)
- L'agent de la DSI accède au statistique utilisateur voulu.
  (CLI->DSIController-> DSIManagementComposent->StatRepository)
- L'agent de la DSI souhaite modifier le catalogue d'offre de commune ou VFP
- (CLI->DSIController->DSIManagementComposent->OfferRepository)

7. Docker Chart de la CookieFactory fournie (DevOps)



Le tcf-spring-backend se sert du tcf-bank-service et le tcf-spring-cli se sert du tcf-spring-backend, ce qui explique les liaisons ainsi que l'ordonnancement du bootstrap. Malgré le docker-compose qui les lance dans l'ordre suivant : server, cli et bank, via le script wait-for-it, ils attendent chacun que leur depends_on soit lancé pour se démarrer.