OC PIZZERIA

Gestion de commande

Dossier de conception fonctionnelle

Version 2.0

My_solution – 0456565656 – mysolution@hotmail.fr
https://mysolution.com
S.A.R.L. au capital de 1 000,00 € enregistrée au RCS de Xxxx – SIREN 999 999 999 – Code APE : 6202A

Auteur
Jean Baptiste Servais
Analyste programmeur

Table des matières

1 - VERSIONS

Auteur	Date	Description	Version
Servais	02/04/19	Création du document	02/04/19

2 - INTRODUCTION

2.1 -Objet du document

Le présent document constitue le dossier de conception fonctionnelle de l'application de gestion de commande Oc Pizzéria.

L'objectif du document est de concevoir l'aspect des fonctionnalités de l'outil de gestion de commande de la pizzéria.

Les éléments du présent dossiers découlent :

- de la réunion avec le directeur de la pizzéria,
- des reunions avec l'ensemble des developpeurs (font et back).

2.2 -Références

Pour de plus amples informations, se référer également aux éléments suivants :

1. **DCT – 2.0** : Dossier de conception technique de l'application
2. **DCF – 2.0** : Dossier de conteption fonctionnelle
3. **DE – 2.0 :** Dossier d'exploitation
4. **PV – 2.0 :** PV livraison

2.3 -Besoin du client

2.3.1 - Contexte

OC Pizzéria est un jeune groupe de pizzeria en plein essor et spécialisé dans les pizzas livrées ou à emporter. Il compte déjà 5 points de vente et prévoit d'en ouvrir au moins 3 de plus d'ici la fin de l'année.

Un des responsable du groupe a pris contact avec vous afin de mettre en place un système informatique sur-mesures, déployé dans toutes ses pizzerias et qui lui permettrait notamment : d'être plus efficace dans la gestion des commandes, de leur réception à leur livraison en passant par leur préparation ; de suivre en temps réel les commandes passées et en préparation ; de suivre en temps réel le stock d'ingrédients restants pour savoir quelles pizzas sont encore réalisables ; de proposer un site internet pour que les clients puissent : passer leurs commandes, en plus de la prise de commande par téléphone ou sur place payer en ligne leur commande s'il le souhaite, sinon, ils paieront directement à la livraison modifier ou annuler leur commande tant que celle-ci n'a pas été préparée de proposer un aide mémoire aux pizzaiolos indiquant la recette de chaque pizza d'informer ou notifier les clients sur l'état de leur commande. Le client a déjà fait une petite prospection et les logiciels existants qu'il a pu trouver ne lui conviennent pas.

2.3.2 - Enjeux et Objectifs

L'enjeu principal est de proposer une solution technique et fonctionnelle qui permettrait d'améliorer la gestion de commande. Pour cela, nous devons proposer des outils simples et complets. Que ce soit pour les utilisateurs du coté du client mais aussi vis à vis du membre du personnel. En effet, l'utilisateur ne doit pas chercher pour s'inscrire, se connecter, visualier les produits et surtout pas pour pouvoir commander et donc payer. Le membre du personnel ne doit pas passer par des étapes fastidieuses et difficiles afin de faire une commande, la terminer ou meme s'inscrire ou se connecter.

Les applications devront alors avoir des fonctionnalités accessibles afin de pouvoir permettre une gestion de commande plus efficace.

Les objectifs est d'améliorer la gestion de commande de la pizzéria à travers différents points. Le premier est la réception de la commande. Pour cela il nous faut des outils adéquats pour le serveur qui la recoit, pour le cuisinier qui la prépare, et pour le livreur qui la livre. Le directeur aussi doit avoir des outils adaptés pour la gestion de son équipe.

Nous proposons alors un site web pour le client et une application logicielle pour le membre du personnel liée par une base de donnée. C'est sur la base de donnée que tout le site repose. Elle devra donc etre fiable, dynamique et rapide. Afin de pouvoir afficher les commandes le plus rapidement possible, les effacer les modifier... Mais aussi de pouvoir ou non afficher la liste des produits qui sont disponibles aux consommateurs.

Les deux apllications devront etre disponibles sur tout type d'écran (téléphone portable, tablette et ordinateur). Les deux applications devront etre accessibles et esthétiquements correctes.

Afin d'assurer une optimisation de la gestion de commande nous devront donc faire un site accessible, simple et complet. Au niveau du design les applications devront etre avenants sans surcharges. Pour cela nous pourront faire appel à une personne possèdant des qualités de neuromarketing.

3 - DESCRIPTION GÉNÉRALE DE LA SOLUTION

3.1 -Les principe de fonctionnement

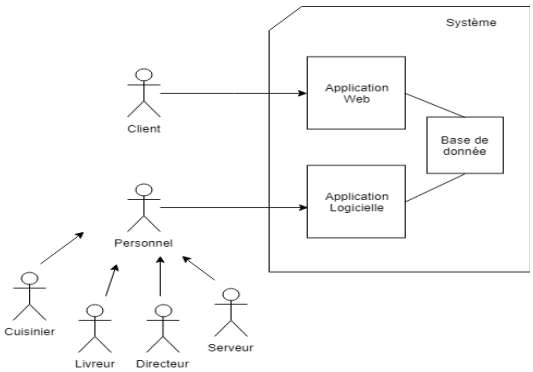
Nous allons donc devoir faire 2 applications. Une application web pour le client et une application logicielle pour le membre du personnel. Le tout relié par une base de donnée de type relationnelle en sql. Le client s'inscrit, accède à la liste des produits, commande, paie et selon son type de commande recoit sa pizza. Le personnel doit pouvoir répondre à la demande. Pour cela, nous devons diviser les fonctionnalités de l'application. Le cuisinier doit pouvoir gérer et préparer les commandes. Le serveur et le livreur doivent pouvoir faire payer le client, et le directeur doit pouvoir faire le coté plus administratif.

Les produits seront gérés automatiquement par la base de donnée. Cela affiche les produits en boutique (la liste des pizza par exemple), renseigne le directeur sur l'état du stock, le cuisinier sur les commandes en attente et le livreur et serveur sur les commandes à payer.

3.2 -Les acteurs

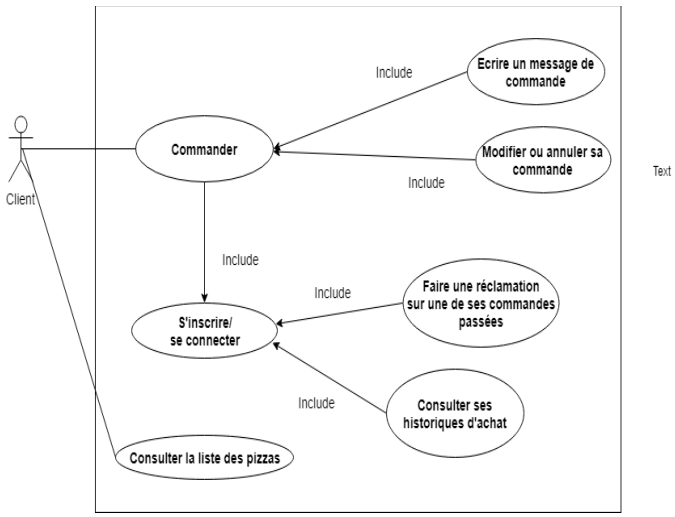
Nous avons défini 2 types d'acteur susceptibles d'être en interaction avec le système informatique de gestion des commandes à travers une application Web et à travers une application logicielle.

-> Le premier type d'acteur est le client qui veut acheter une pizza rapidement.
-> Le deuxième type d'acteur est l'employé de la pizzeria, sans formation en informatique qui va devoir utiliser le système informatique.
- Le cuisinier se servira de l'application logicielle afin de pouvoir faire les pizzas.
- Le serveur utilisera l'application mobile afin de faire payer la commande au client.
- Le livreur pourra livrer la pizza à l'adresse du client grâce à l'application mobile.
  - Le directeur va gérer les réclamations, l'espace du personnel ainsi que le stock par l'application logicielle.



Découpage en package

3.3 -Les cas d'utilisation généraux



Cas d'utilisation client

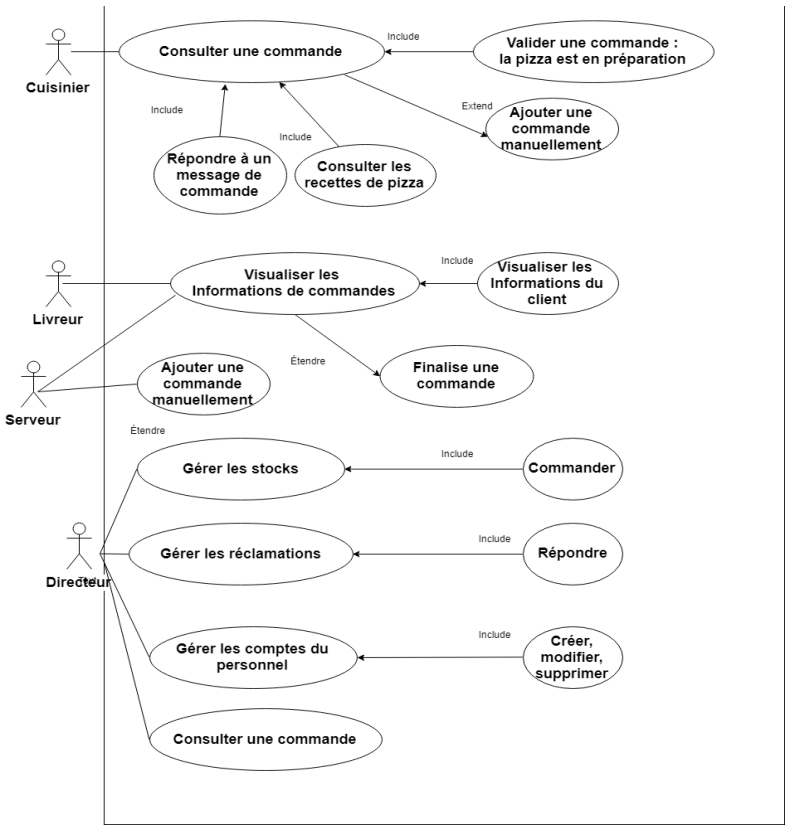
Que doit pouvoir faire le client ?

Le client dispose de plusieurs fonctionnalités. Le client peut visualiser la liste des pizza sans connexion ni inscription au préalable. De ce fait il peut alors commencer a faire son choix et s'inscrire. Il faut noter que cette fonctionnalité est très importante.

Il peut s'inscrire et se connecter. Cela assure la sécurité de la commande et le bon fonctionement du site. Il peut aussi consulter son panier afin de pouvoir ajuster ses besoins. Ensuite il peut commander. La commande prend en compte le paiment et le type de commande (sur place ou à emporter cf).

Le client doit pouvoir nous contacter en cas de besoin (question hors commande). Il doit aussi pouvoir consulter son historique de commande et en cas de nécéssité il doit pouvoir faire une réclamation.

Le client dispose donc de plusieurs fonctionnalités. Que ce soit au niveau de la pré-vente comme la visualisation des pizzas, a l'inscription, connexion, choix de ces produits, commande jusqu'au niveau du suivis après vente par la réclamation.



Cas d'utilisation du Personnel

Que doit pouvoir faire le personnel ?

Nous avons choisis d'attribuer des fonctionnalités selon le type de personnel. Tout d'abord tous les acteurs doivent pouvoir se connecter. Le premier type d'acteur que nous allons voir est le cuisinier. Le cuisinier est le role principale du fonctionnement de la gestion de commande. Il doit donc pouvoir visionner la liste de commande . Il doit gérer les commandes qui se traduit par la prise en charge ou non d'une commande à un instant t. De plus il doit pouvoit visualiser la recette d'une pizza au cas ou. Enfin il peut ajouter une commande manuellement. Le cuisinier peut donc pouvoir voir et gérer les commandes en attente pour les faire passer à en cours. Il peut visualiser la recette d'un produit et ajouter une commande manuellement. Mais qui va donc faire payer le client ?

Le cuisinier est le role le plus important dans la gestion de commande. Mais sans le role du serveur l'enseigne ne marcherait pas. Le livreur possède plusieurs fonctionnalités au sein de l'application logicielle. Ce dernier peut accéder aux informations du client. Il peut aussi ajouter une commande manuellement. Et c'est ce dernier qui va pouvoir terminer une commande en cours en prenant le paiement du client. Et si la commande est à livrer ?

Si la commande est à livrer la suite de la commande revient au livreur. Il peut également voir les informations du client afin de connaitre ses informations. Il termine lui aussi la commande en faisant payer le client. Le livreur est donc le role de liaison entre l'enseigne et les clients qui ne souhaitent pas se déplacer.

Enfin le dernier acteur est le directeur. Il possède de nombreuses fonctionnalités. Il peut comme le cuisinier gérer les commandes. Il peut aussi faire la gestion du stock en vérifiant son état en passant

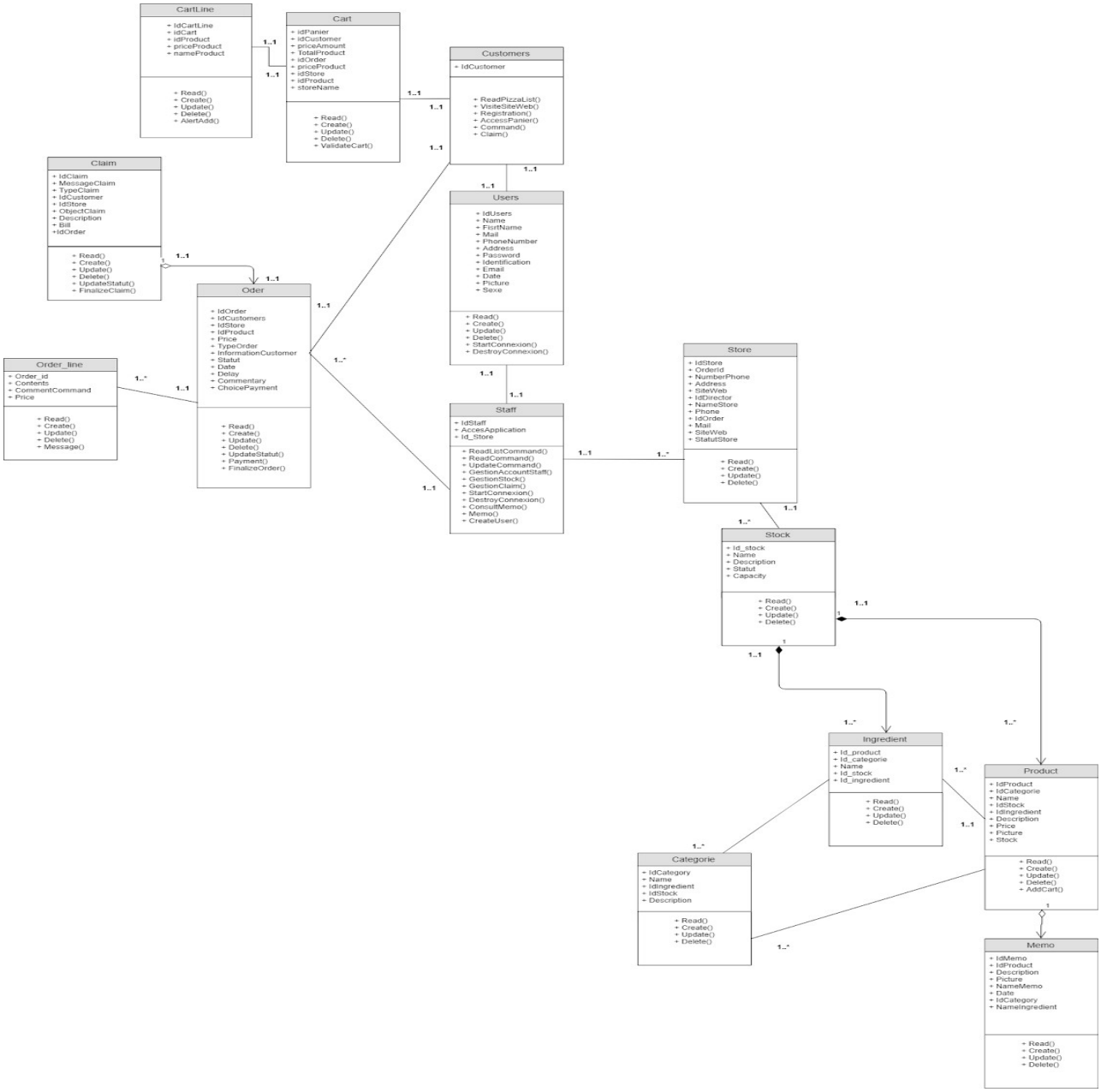
commande. Il peut aussi faire la gestion de profil du personnel en leur attribuant leur accès à l'outils mais aussi gérer d'éventuelles réclamations. Le directeur à donc le plus d'outils à travers cette application. Car il peut gérer les commandes mais aussi la partie plus administrative de l'application logicielle.

Nous avons donc vu les acteurs divisé par le livreur, le serveur le cuisinier et le directeur. Tous ces acteurs ont différent role et donc différentes fonctionnalités qui assurent la cohésion de l'enseigne et la gestion de commande.

4 - LE DOMAINE FONCTIONNEL

4.1 -Référentiel

Diagramme UML de classes



Nous commencerons par la classe centrale Users. Cette classe est composée de deux autres classes qui nous amèneront dans la logique suivante: 1- les classes centrées clients et 2- les classes centrées pizzéria.

La classe Users dispose de différentes méthodes de classe. Les deux premières permettent de lancer ou de stopper une connexion avec les méthodes: destroyConnexion et StartConnexion(). Elle possède aussi les méthodes CRUD: créer un utilisateur en base de donnée, voir les informations d'un utilisateur, modifier les informations d'un utilisateur et supprimer un utilisateur.
La classe Users est composé de plusieurs attributs de classe. Le premier attribut est IdUsers qui va nous permettre de pouvoir identifier un utilisateur de manière unique en base de donnée. Il y a aussi Name, FirstName, Mail, PhoneNumber, Address, Email, Picture et Sexe qui vont nous donner l'accès aux informations essentielles sur le client.
Par exemple à partir de ces informations on va pouvoir répondre à d'éventuelles réclamations de la part d'un client ou pouvoir envoyer des papiers administratifs aux employés. De plus il y a d'autres attributs de classe: l'identifiant, le password, et la date de création de compte. L'utilisateur pourra ainsi accéder à son compte en toute sécurité.

Nous avons donc vu que la classe User avait plusieurs méthodes de classe de connexion et elle qu'elle était composée des méthodes CRUD. Nous avons aussi vu qu'elle avait plusieurs attributs de classe comme les informations sur l'utilisateur et sur le compte.
La classe Users est composée de deux autres classes. La classe Client et la classe du personnel. Dans un premier temps nous allons voir la classe client puis dans un second temps nous verrons la classe du personnel.

Tout d'abord il y a la classe Customer qui est reliée à la classe Users via une relation one-to-one. La classe client hérite de la classe Users.

Cette classe à un attribut de classe (en plus des attributs de classe de Users dont elle hérite) qui est IdCustomer. IdCustomer permet de pouvoir identifier un client de façon unique en base de donnée.
La classe client possède plusieurs méthodes de classe (en plus des méthodes de Users) : VisiteSiteWeb() pour que le client puisse visiter le site web, ReadPizzaList() qui permet de voir la liste des pizzas du siteWeb, Registration() afin de s'enregistrer, AccesPanier() pour pouvoir accéder à son panier, Command() pour commander et Claim() afin d'effectuer des réclamations.

Nous avons donc vu que la classe Client est composée de la classe Users. La classe client dispose donc de plusieurs attributs de classe (cf attribut de classe d'Users) et de plusieurs méthodes centrées sur la navigation du site web et sur la commande. Cette classe est reliée à la classe Panier par une relation one-to-one. En effet, un client doit avoir un panier pour commander.

La classe Panier est composé de la classe CartLine (que nous verrons dans le paragraphe suivant) et qu'elle est relié à la classe customers. Cette classe est relié à ces deux classes via des relations, one-to-one (Cart-Customers) et one-to-many (Cart-CartLine).

La classe Panier est composée: des méthodes CRUD: créer un panier, afficher le panier, modifier le panier et supprimer le panier. En plus de ValidateCart() qui permet de valider son panier pour pouvoir passer à la commande.

La classe Panier est composé de plusieurs attributs de classe. L'idPanier va pouvoir permettre l'identification unique du panier en base de donnée. L'idcustomer établit la relation entre le client et la panier. PriceProduct est la variable qui est le prix d'un produit. Id_product permet d'identifier un produit. PriceAmount et TotalProduct permettent d'afficher la totalité du panier en terme de prix et de produit. L'IdOrder permet d'établir la liaison entre le panier et la commande. Enfin, IdStore et StoreName permettent d'établir la connexion entre un panier et un magasin.

Nous avons donc vu que la classe Cart était composée de plusieurs méthodes de classe sur la modification du panier et sur la poursuite du processus de commande, ainsi que de plusieurs attributs de classe portant sur le panier mais aussi sur sa constitution. Nous avons aussi vu qu'elle était reliée à deux autres classes: Customer et CartLine. CarteLine par exemple permet de pouvoir remplir son panier. En effet, elle permet d'ajouter des produits de les modifier, de les supprimer ect…

Les classes CartLine et Cart sont reliées par une relation one-to-many car un panier est composé de plusieurs lignes de panier et les lignes de panier composent un panier.

La classe CartLine est composée des méthodes CRUD: create() pour créer une ligne au panier, read() pour afficher une ligne de son panier, update() pour modifier une ligne de son panier et delete() pour supprimer une ligne au panier. Elle est également composée d'alertAdd() qui va permettre d'alerter le client qu'il a modifié son panier (par exemple un ajout).

La classe CartLine est reliée au panier. En effet elle permet de pouvoir modifier ces lignes. Elle dispose de plusieurs attributs de classe. L'idCartLine permet d'identifier de façon unique une ligne du panier. L'attribut idCart qui permet d'établir la connexion entre CartLine et Cart. IdProduct va pouvoir identifier un produit qui va être ajouté, modifié ou supprimé au panier. PriceProduct qui va nous renseigner sur le prix du produit qui va être ajouté, supprimé ou modifié du panier. Enfin nous avons NameProduct qui va nous informer sur le nom du produit. qui va être ajouté, supprimé ou modifié d'une ligne du panier.

Nous avons donc vu que CartLine permettait de modifier les lignes du panier par différentes méthodes et qu'elle était constituée de plusieurs attributs de classes. En résumé, la classe Users est composée de la

classe Customers qui est elle même reliée à la classe Cart composée de la classe CartLine. Une fois son panier constitué, le client va vouloir commander. C'est ce que nous allons voir dans le paragraphe suivant.

Maintenant nous allons voir que la classe Customers est aussi reliée à la classe Order. Cela va nous permettre de pouvoir faire le pont entre le côté client et le côté du personnel. En effet, La classe client et la classe du personnel que nous verrons un peu plus tard sont reliées à une même classe, la classe Order par une relation one-to-one(Client-Order, un client a une commande) et une relation one-to-many(Staff-Order, un membre du personnel peut effectuer plusieurs commandes à la fois).

La classe Order est composée des méthodes CRUD: créer une commande en base de donnée, voir une commande en base de donnée, modifier une commande et supprimer une commande.
Mais elle est également composée de la méthode updateSatut() qui permet de mettre à jour le statut de la commande. Composée aussi de la méthode payment() qui sert à pouvoir faire payer une commande et finalizeOrder() qui permet de terminer une commande.

Elle dispose aussi de plusieurs attributs de classe qui sont: IdOrder qui permet l'identification unique de la commande en base de donnée. Idcustomers qui va faire la relation entre la commande et l'id du client, et également informationCustomer qui va permettre d'avoir accès aux informations du client. L'idStore va relier la commande à son restaurant. Id product va permettre la relation entre les produits et la commande la composant. Price qui va nous renseigner le prix de la commande, typeOrder qui nous indique le type de la commande (livraison, en magasin...). Statut qui permet de connaître le statut de la commande (en cours, en attente...). Date qui va pouvoir permettre de connaître la date et l'heure de la commande (pour une réclamation, pour une commande en cours ect...) Delay qui va pouvoir nous donner une idée du temps de préparation au client et au membre du personnel. Commentary pour d'éventuelle commentaire à propos de la commande et ChoicePayment pour le type de paiement (carte, espèce...).

Nous avons donc vu que la classe Order était composée de plusieurs méthodes de classe comme les méthodes Crud ainsi que l'actualisation du statut de la commande, le paiement et la finalisation de la commande. Puis qu'elle était composée de plusieurs attributs de classe reliant ainsi le client, le magasin, le personnel et les produits à la commande.

Comme le panier qui est composé de ligne de panier, Order est composé de la classe Order_line via une relation one-to-many. En effet, une order_line compose une commande et une commande est composée de plusieurs ligne de commande.

La classe Order_line permet d'écrire des messages par sa méthode Message(). Elle est également constituée des méthodes CRUD: créer une spécificité à la commande, lire une spécificité, mettre à jour une

spécificité et de supprimer une spécificité.
La classe Order_line est composée de plusieurs attributs de classe. Order_id permet l'identification unique en base de donnée de la ligne de commande. Contents est le contenu de la ligne ajoutée, supprimée ou modifiée. CommentCommand permet d'éventuels commentaires et Price est le prix de la ligne de commande (un produit par exemple).

Nous avons donc vu qu'Order_line permettait d'ajouter des spécificités, de supprimer ou de modifier des lignes de la commande via une interaction one-to-many d'Order_line à Order. Il est important de pouvoir faire des réclamations pour d'éventuels problèmes de commande.
C'est pour cela qu'Order est relié à la classe Claim via une connection de type agrégation one-to-one. En effet, il faut qu'il y est une commande pour qu'il puisse y avoir une réclamation mais la commande existe sans la réclamation.

La classe Claim est composée des méthode CRUD create() qui permet la création d'une réclamation, read() qui permet la lecture de la réclamation, update() qui permet de modifier la réclamation, et delete() qui permet de supprimer une réclamation. De plus, Claim est constituée d'updatestatut() afin de mettre à jour le statut de la réclamation et finalizeClaim() qui termine une réclamation.
La classe Claim a plusieurs attributs de classe. IdClaim pour pouvoir identifier de manière unique une réclamation en base de donnée, MessageClaim() permet de faire les messages de réclamation, TypeClaim nous renseigne sur le type de la réclamation. IdCustomer nous permet de faire la connexion entre le client et la réclamation. Id Store fait la connexion entre un magasin et la réclamation. ObjectClaim nous renseigne sur l'objet de la réclamation. Description va nous permettre d'avoir une description de la réclamation. Bill qui est la facture et est un élément indispensable pour la réclamation. Et pour terminer IdOrder fait la connexion entre la commande et la réclamation.
Nous avons donc vu la classe Claim qui est reliée à la classe Order par une relation de type agrégation one-to-one. Cette classe Claim dispose de plusieurs méthodes de classe ainsi que de plusieurs attributs de classe vis à vis de l'identité de la réclamation. Nous avons aussi vu que la classe Order était reliée à la classe Customers et à la classe Order_line.

Cette partie était accès sur le client, son panier et sur la commande. Nous allons maintenant voir la partie accès sur le fonctionnement de la Pizzéria.
Pour rappel, la classe Order est en interaction avec la classe Customers mais elles est aussi reliée à la classe Staff car il faut pouvoir faire la commande d'un client. Il faut aussi noter que la classe Staff est elle aussi est un composant d'Users.

Voici la classe du personnel qui est aussi reliée à la classe Order par une relation one-to-many et est également reliée à la classe Users par une relation one-to-one.

Cette classe est composée de plusieurs méthodes: ReadListCommand() permet de pouvoir voir la liste des commandes en attente et en cours. UpdateCommand() permet de pouvoir mettre à jour le statut d'une commande. CreateUser() permet de gérer les comptes du personnel. GestionStock() permet de pouvoir gérer les stocks (cf Stock()). GestionClaim() permet de gérer les réclamations(cf Claim()). ConsultMemo() pour pouvoir faire des mémos (cf Memo()), CreateUser() permet de créer des comptes des membres du personnel et ConsultMemo pour consulter au besoin un mémo. En plus de tout cela, la classe Staff hérite de la classe Users et possède donc toutes ces méthodes de classe comme StartConnexion() pour se connecter, DestroyConnexion() pour se déconnecter, et des méthodes CRUD.
La classe Staff a aussi plusieurs attributs de classe comme IdStaff afin de pouvoir identifier de manière unique l'id du personnel en base de donnée. AccesApplication afin de pouvoir utiliser l'application (par exemple le livreur et le cuisinier n'auront pas le même accès à l'application). Et id_store afin de pouvoir établir la connection entre le personnel et le magasin. Héritant de la classe Users, la classe Staff possède aussi ces attributs de classe comme toutes les informations du compte (email, nom, addresse...).

Nous avons donc vu que la classe Staff était composée de méthode permettant la gestion de commande, qu'elle héritait de la classe Users et qu'elle était constituée de plusieurs attributs de classe portant sur les données personnelles de l'utilisateur.

La classe Staff est donc reliée aux deux classes précédemments vu (Order et Users). Mais elle est aussi en interaction avec la classe Store via une relation one-to-many(un magasin a plusieurs employés et un employé travail dans un magasin).

Cette dernière est composée des méthodes CRUD: créer une base donnée pour un magasin, voir une base donnée d'un magasin, de modifier une base de donnée d'un magasin et de supprimer() pour pouvoir supprimer une base de donnée d'un magasin.
Elle possède aussi plusieurs attributs de classe. IdStore est l'identifiant unique d'un magasin en base de donnée. OderId est la relation entre une commande et son magasin, NumerPhone, address, SiteWeb, NameStore, Mail sont les informations du magasin. IdDirector permet d'identifier le directeur du magasin. StatutStore permet de savoir quel est le statut du magasin(fermé, ouvert…).

Nous avons donc vu que la classe Store avait plusieurs méthodes et attributs de classe. Et qu'elle était en relation avec la classe Staff.
Mais elle aussi en interaction avec la classe stock avec une relation one-to-many (store-cart car un magasin a plusieurs stock et un stock n'a qu'un seul magasin).

La classe Stock est composée des méthodes CRUD: create() qui permet de créer un stock, read() qui

permet d'afficher un stock, update() qui permet de modifier un stock et delete() qui permet de supprimer un stock.
Elle possède aussi des attributs de classe: id_stock qui permet de d'identifier un stock de manière unique en base de donnée, Name qui est le nom du stock, Statut qui nous renseigne sur l'état du stock(plein, vide...) et capacity qui nous renseigne sur sa taille maximum.

Nous avons donc vu la classe Stock à travers ses méthodes et attributs de classe. Mais aussi qu'elle était en relation avec la classe Store.

Cette classe est aussi en relation de type composition avec deux autres classes. La classe Ingrédient et la classe Product. De fait, afin qu'un stock existe il faut qu'il y est des ingrédients et des produits.

La première relation est une relation one-to-many avec la classe Ingrédient(Stock-Ingrédient un stock a plusieurs ingrédients et un ingrédient n'a qu'un stock). Cette classe est constituée des méthodes CRUD: read() permet de voir un ingrédient, create() permet d'ajouter un ingrédient, update() permet de modifier un ingrédient et delete() qui permet de supprimer un ingrédient.
Elle est aussi composée d'attributs de classe. Id_Ingredient permet d'identifier de manière unique un ingrédient en base de donnée. Id_categorie permet de faire la relation entre l'ingrédient et la classe catégorie. Name permet de connaître le nom d'un ingrédient. Id_stock permet de savoir à quel stock est relié l'ingrédient.
Nous avons donc vu la classe ingrédient était composée de méthodes et d'attributs de classe. Mais aussi qu'elle était en relation avec la classe Stock.

Cependant, la classe Stock est aussi reliée à la classe Product via une relation one-to-many de type composition (Stock-Product un produit n'a qu'un stock et un stock a plusieurs produits).

La classe stock est composée de méthode de classe CRUD: read() qui permet de voir un produit, create() qui permet de créer un produit, update() qui permet de modifier un produit et delete() qui permet de supprimer un produit. Elle est aussi constituée de la méthode AddCart() qui permet de mettre un produit dans le panier.
Cette classe a aussi des attributs de classe. IdProduct qui permet d'identifier de manière unique un produit en base de donnée. IdCategory qui permet de faire la relation entre la catégorie et le produit. Name qui permet de savoir le nom d'un produit. IdStock qui permet de faire la liaison entre le stock et un produit. Ingrédient qui permet de faire la relation entre un produit et les ingrédients le constituant. Description qui permet de faire une description d'un produit. Price qui permet de savoir le prix d'un produit. Et picture qui permet de voir une image du produit.

Nous avons donc vu la classe product à travers ces méthodes et ces attributs de classe. Et qu'elle

était en relation avec la classe stock. Il faut aussi préciser que la classe Ingrédient et Product sont en relation one-to-many (ingredient-produit un produit est constitué de plusieurs ingrédients et un ingrédient constitue un produit)

La classe Product est aussi en relation de type agrégation one-to-one avec la classe Mémo.

La classe mémo est constituée de méthode de classe CRUD: read() qui permet de voir un mémo, create() qui permet de créer un mémo, update() qui permet de modifier un mémo et delete() qui permet de supprimer un mémo.
Cette classe a aussi plusieurs attributs de classe. IdMemo qui permet d'identifier un mémo de manière unique en base de donnée. IdProduct qui permet de faire la relation entre un produit et son mémo. Description qui permet de faire la description d'un produit. Picture qui permet de voir l'image du produit. NameMemo qui permet de connaître le nom du mémo. Date qui permet de savoir la date de création du mémo. IdCategory qui permet de faire la relation entre la catégorie du produit et du mémo. Et NameIngrédient qui permet de savoir quels ingrédients composent le produit.

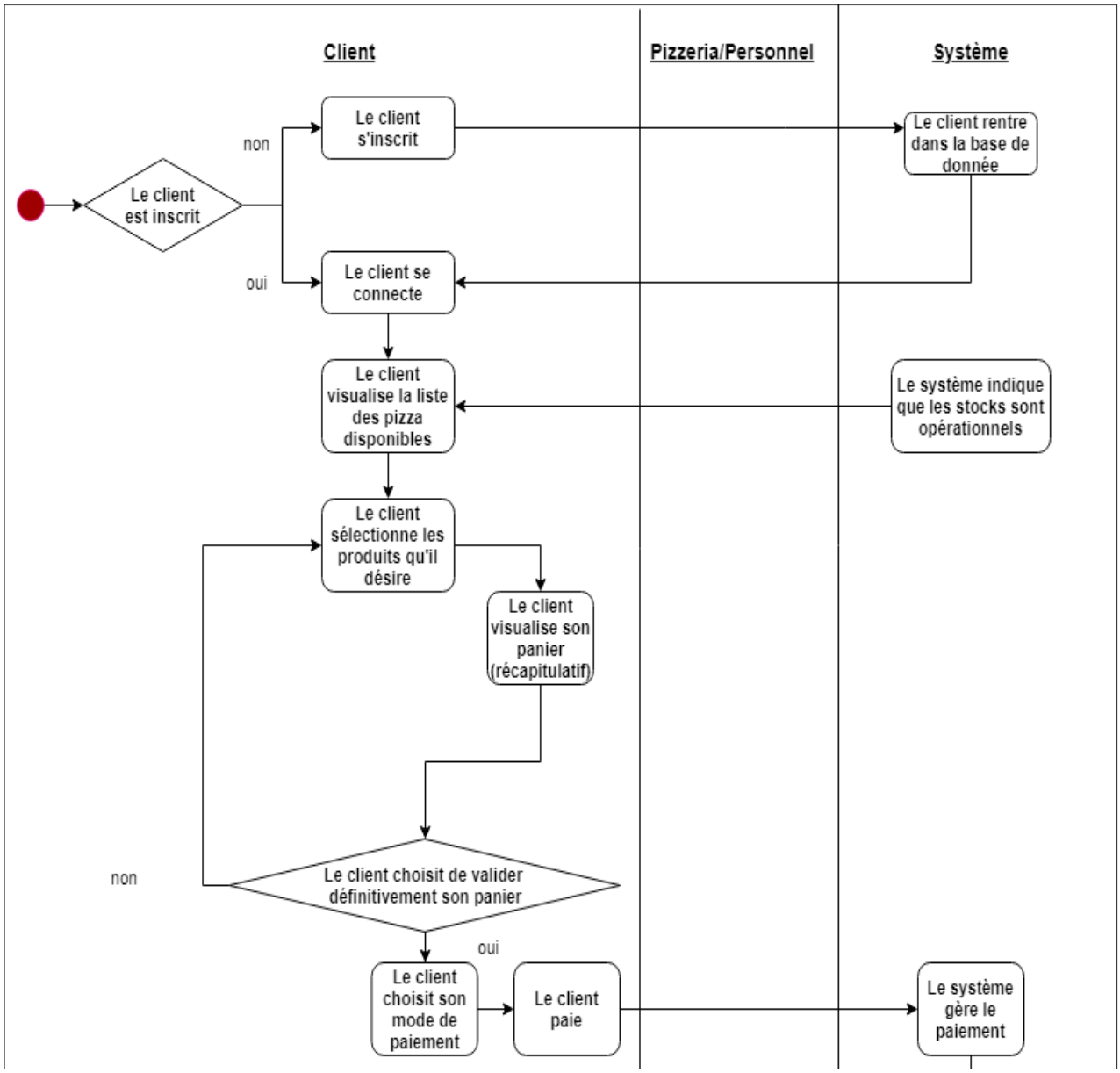
Nous avons donc vu que la classe Mémo était en relation avec la classe Produit. Nous avons aussi vu ces méthodes et les attributs de classe.

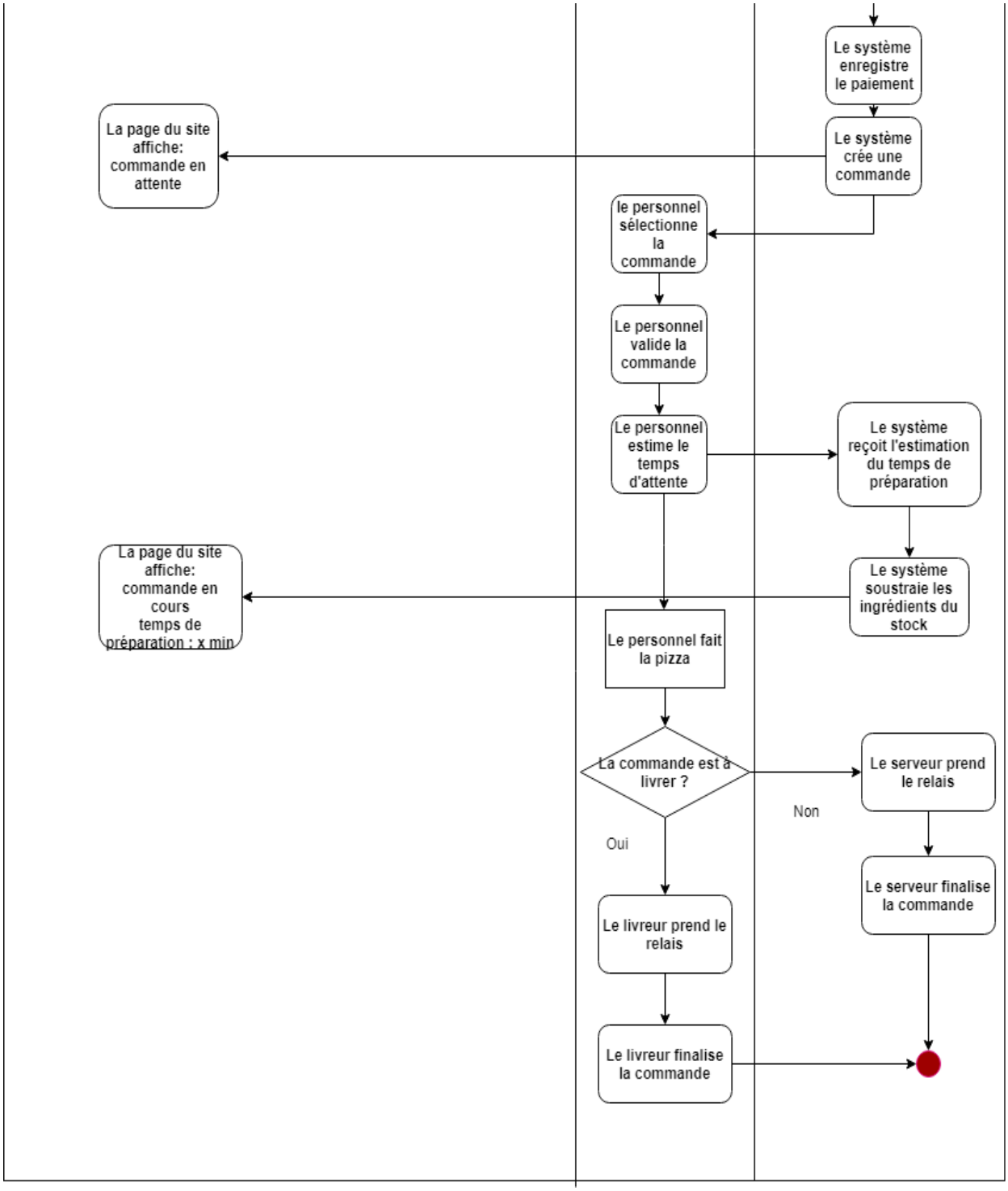
Pour finir, Nous allons voir la classe Catégorie qui est en relation avec la classe Ingrédient et la classe Product via des relations many-to-many (catégorie-ingrédient un ingrédient a plusieurs catégories et une catégorie a plusieurs ingrédient. Product-catégorie un produit a plusieurs catégorie et une catégorie a plusieurs produits).

La classe catégorie est composée des méthodes de classe CRUD: read() qui permet de voir une catégorie, create() qui permet de créer une catégorie, update() qui permet de modifier une catégorie et delete() qui permet de supprimer une catégorie.
Cette dernière est aussi composée d'attribut de classe. IdCategory qui permet d'identifier de manière unique la catégorie en base de donnée. Name qui permet de savoir le nom d'une catégorie. Ingrédient qui permet de faire la relation avec la classe ingrédient. IdStock qui permet de faire la relation avec la classe stock. Et la description qui permet de connaître la description d'une catégorie.

5 - LES WORKFLOWS

5.1 -Le workflow




Le système enregistre le paiement
Le système crée une commande
La page du site affiche: commande en attente
le personnel sélectionne la commande
Le personnel valide la commande
Le personnel estime le temps d'attente
Le système reçoit l'estimation du temps de préparation
La page du site affiche: commande en cours temps de préparation : x min
Le système soustraie les ingrédients du stock
Le personnel fait la pizza
La commande est à livrer ?
Non
Oui
Le serveur prend le relais
Le serveur finalise la commande
Le livreur prend le relais
Le livreur finalise la commande

6 - LES CAS D'UTILISATION

6.1 -Les cas d'utilisation

Cas numéro 1 : visualiser la liste des pizzas

Nom : visualiser la liste des pizzas

Acteur : Un utilisateur

Description : L'utilisateur doit pouvoir voir la liste des pizzas.

Auteur : JBS

Date : 02/10/2018

Pré-conditions : Aucunes.

Démarrage : Etre allé sur le site Web.

Le scénario nominal

1 - L'utilisateur est sur la page d'accueil.

2 - Il clique sur liste des pizzas.

3 - Il visualise la liste des produits.

Le scénario alternatif

2 - Il clique sur une autre rubrique.

Cas numéro 2 : s'inscrire

Nom : S'inscrire

Acteur : Le client

Description : Le client doit s'inscrire pour commander.

Auteur : JBS

Date : 02/10/2018

Pré-conditions : Aucunes.

Démarrage : Il faut avoir cliqué sur «s'inscrire».

Le scénario nominal

1 - Le client clique sur la commande « s'inscrire ».

2 - Le système affiche un formulaire: nom, prénom, âge, adresse, téléphone fixe, mobile, adresse, mail …

3 - Le client reçoit un message mail et valide son inscription en appuyant sur le lien du courriel.

4 - Le client est envoyé sur la page d'accueil

Le scénario alternatif

2a - Le client retourne sur la page d'accueil, l'inscription est annulée.

2b - Le client quitte la page web.

2c - Le client ne remplit pas les champs obligatoires, un message d'erreur apparaît, il doit alors les re saisir.

3a - Le client ne reçoit pas de courriel, on lui renvoie alors le message.

4a - Le client quitte la page web.

Les posts-conditions

Le client est enregistré dans la base de donnée.

Le client peut se connecter.

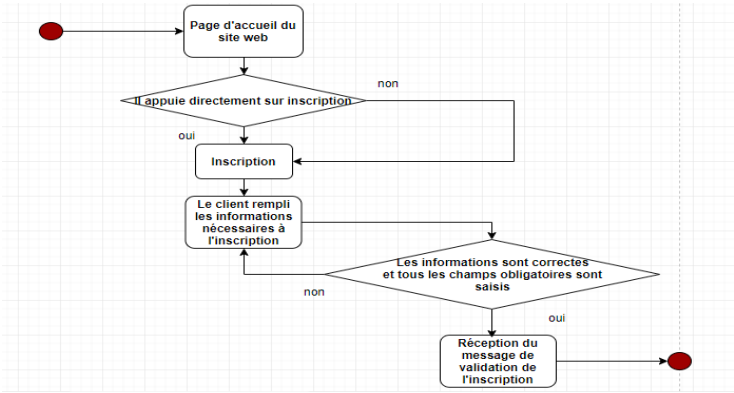


Diagramme d'activité du cas d'utilisation "s'inscrire"

Cas numéro 3 : se connecter

Nom : Se connecter

Acteur : Le client
Description : Le client doit pouvoir se connecter pour pouvoir commander.
Auteur : JBS
Date : 02/10/2018
Pré-conditions : S'être inscrit
Démarrage : Avoir rentré ses identifiants.

Le scénario nominal

1 - Le client clique sur se connecter.
2 - Il rentre ses identifiants.

Le scénario alternatif

1 - Il clique sur une autre rubrique.

2 - Il se trompe dans ces identifiants, il doit alors les re-rentrer.

3 - Il est sur la liste des pizzas. Il clique sur sélectionner un article. On l'envoie sur l'inscription s'il n'est pas inscrit. Ou on le fait se connecter.

Les posts-conditions

Il se connecte et il peut sélectionner des articles.

cas numéro 4 : Ajouter des produits à son panier

Nom : Ajouter des produits à son panier.

Acteur : Le client
Description : Le client doit pouvoir ajouter des produits à son panier pour pouvoir commander.
Auteur : JBS
Date : 02/10/2018
Pré-conditions : S'être inscrit et connecté. Être allé sur la liste des pizza.
Démarrage : Avoir cliqué sur: "liste des pizzas".

Le scénario nominal

1- Le client clique sur: “Ajouter au panier” avec la quantité souhaité.

Les posts-conditions

Un message s’affiche: “Produit ajouté”.

Ergonomie

La page s’agrandit au fur et à mesure que le client va vers le bas. Les produits peuvent être organisés par catégorie.

Cas numéro 5 : consulter son panier

Nom : Consulter son panier.
Acteur : Le client
Description : Le client doit pouvoir consulter son panier afin de pouvoir commander.
Auteur : JBS
Date : 02/10/2018
Pré-conditions : Avoir sélectionné des produits depuis la liste des pizzas;
Démarrage : Avoir cliqué sur son panier.

Le scénario nominal

1 - L’utilisateur clique sur son panier.
2 - Le système affiche les produits, leur quantité, leur prix et le prix totale.

Le scénario alternatif

1 - Le client continue sa sélection.

Les posts-conditions

Des produits sont présents dans le panier.

cas numéro 6: commander

Nom : Commander

Acteur : Le client

Description : Le client doit pouvoir passer une commande.

Auteur : JBS

Date : 02/10/2018

Pré-conditions : S'être inscrit, s'être connecté, avoir sélectionné des produits depuis la liste des pizzas et avoir cliqué sur son panier.

Démarrage : Avoir sélectionné ses produits et avoir cliqué sur commander.

Le Scénario nominal

1 - Le système lui demande s'il veut une dernière fois modifier son panier.

2- Il clique sur payer.

3 - Le système affiche une nouvelle page (format de mini-page) avec différents types de paiement: "espèce" ou "carte bancaire". Le client choisit de payer avec la carte bancaire.

4 - Le système affiche le système de paiement par carte bancaire. Le client y met ses informations et appuie sur "valider".

5 - La mini-page se ferme. Le système renouvelle la page en une page qui récapitule les produits commandés. Un message en rouge indique:

"commande en cours"

6 - La commande passe de "commande en cours" à "commande en préparation" avec le temps estimé.

Le scénario alternatif

1 - Il clique sur modifier il est retourné à son panier.

4a - La saisie d'information est erronée.

Posts-conditions

La pizzeria reçoit la commande du client. Les stocks se mettent à jour. Le client reçoit sa commande

Performance attendue

Tout doit être assez rapide, assez fluide. Aucun lag ne doit être signalé.

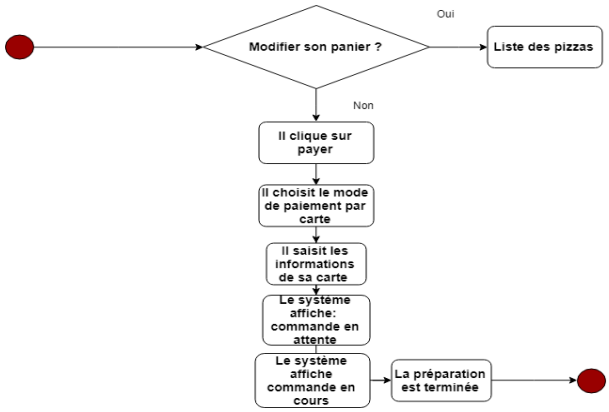


Diagramme d'activité
du cas d'utilisation Commander

Cas numéro 7: nous contacter

Nom : Nous contacter.

Acteur : Le client

Description : Le client doit pouvoir nous contacter si besoin.

Auteur : JBS

Date : 12/10/2018

Pré-conditions : Il faut être inscrit.

Démarrage : Il faut appuyer sur « nous contacter » depuis l'accueil.

Le scénario nominal

1 - Le client est sur la page d'accueil.

2 - Il appuie sur « nous contacter ».

3 - le système affiche une page. Sur cette page se trouve: un encadré intitulé “objet” et un champs de saisie où il y a: « écrire un message » (qui disparaît lorsque l’on écrit dedans).

5 - Il client clique sur envoyer.

6 - Un message apparaît : « message envoyé ».

7 - Le client est renvoyé sur la page d'accueil.

Le scénario alternatif_

3a- Le client ne s'est pas connecté. Il doit se connecter et sera envoyé en 4.

3b- Il retourne sur la page d'accueil.

3c- Il quitte la page.

4a - Il quitte la page.

4b - Il retourne sur la page d'accueil.

5a - Il n'a pas mis d'objet, un message en rouge apparaît, “aucun objet ” L’objet est obligatoire.

5b – Il n'a pas écrit de message, il ne peut alors rien envoyer.

5c – Il retourne sur la page d'accueil.

5d – Il quitte le site web.

Les Posts conditions

Le personnel plus qualifié reçoit un message.

Performance attendue

Tout doit être assez rapide, assez fluide. Aucun lag ne doit être signalé.

Ergonomie

Les messages d'erreur sont en rouges et visibles.

Le message « envoyé » s'affiche au milieu de l’écran.

Le message ne comporte pas plus de 500 caractères

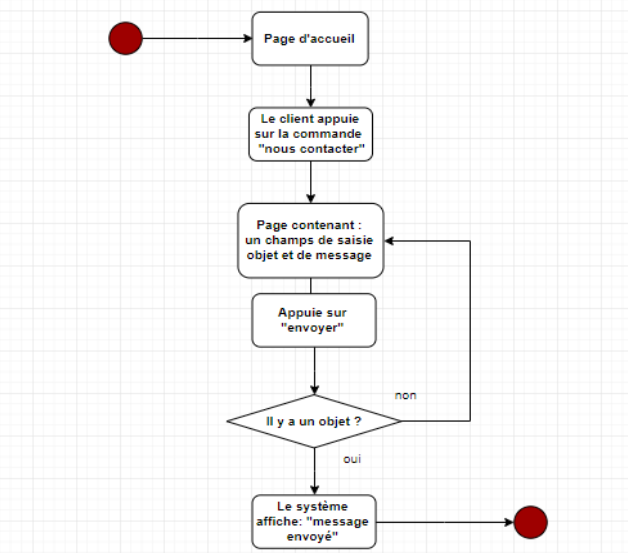


Diagramme d'activité
du cas d'utilisation Nous contacter

Cas numéro 8 : consulter ses factures

Nom : Consulter ses factures.

Acteur : Le client

Description : Le client doit pouvoir voir ses anciens achats.

Auteur : JBS

Date : 12/10/2018

Pré-conditions : s'être connecté et avoir appuyé sur la commande historique achat
Démarrage : Il faut s'être connecté et avoir appuyé sur la commande: "historique d'achat" depuis "mon compte".

Le scénario nominal

1 - Le client a cliqué sur 'anciens achats'
2 - Une nouvelle page s'ouvre sous une forme de liste contenant l'historique des achats et se présente sous la forme de: date/prix.
3 - Le client clique sur l'une des commandes effectuée.
4 - L'historique de la commande apparaît sous forme de mini page.

.

Le scénario alternatif

3a- Il se déconnecte ou quitte le site web.
4a - Il se déconnecte ou quitte le site web.

Les posts-conditions

Le client peut voir sa commande.

Performance attendue

L'historique de l'achat s'affiche 10 minutes après avoir payé.

Ergonomie

L'historique des achats est une liste de 5 par 5 de la plus récente à la plus vieille.

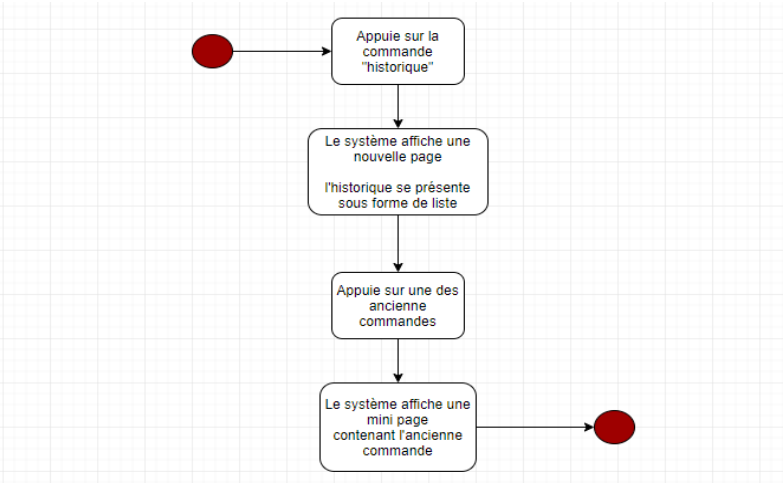


Diagramme d'activité du cas
d'utilisation "consulter ses factures"

Cas numéro 9 : gérer les commandes

Nom : Gérer les commandes

Acteur : Le personnel

Description : Le personnel doit voir les commandes afin de pouvoir les effectuer.

Auteur : JBS

Date : 02/10/2018

Pré-conditions : Il faut avoir un compte personnel et s'être connecté à son compte.

Démarrage : La liste des commandes s'affiche.

Le scénario nominal

1 - Le système affiche la liste des commandes.

2 - Le personnel à 3 choix: il peut "ajouter une commande manuellement", "supprimer ou modifier une commande" et "cliquer sur une des commande". Il appuie sur une des commandes.

3 - Le personnel est sur la commande d'un client où il peut voir les produits commandés sous forme de liste.
En dessous de chaque pizza se trouve un encadré : «recette pizza ». Il y a aussi deux boutons: « message commande » et "informations du compte client".
Enfin il clique sur le bouton "valider" en bas au centre.

4 - Le système affiche une page avec un encadré: « temps estimé ». Il clique dessus. "Temps estimé" s'efface. Il saisit l'encadré par le temps qu'il pense nécessaire à la production de la commande.

5 - Il appuie sur valider en bas au centre.

6 - L'utilisateur est retourné à la liste des commandes.

7 - la commande passe en vert, plus rien n'est modifiable. On peut cependant consulter l'étape 4.

8 - La commande est en préparation.

Le scénario alternatif

2a - L'utilisateur clique sur "ajouter une commande" (qui a été fait par téléphone ou en magasin). La commande s'ajoute à la liste des commandes.

2b - Le personnel vient de recevoir un appel, le client ne veut plus de sa commande ou il a oublié un produit, il peut alors la supprimer ou la modifier.

2c - Il peut se déconnecter.

3a - Il appuie sur "recette pizza": une mini page s'affiche avec les conseils de préparation (Il peut cliquer sur la croix ou sur les cotés, ce qu'il le ramène à la page 3).

3b - Il se déconnecte.

3c - Il fait un retour en arrière qui le ramène à la liste des commandes.

3d - Il se déconnecte.

3f - Il appuie sur ”information du client”. Une mini page s'ouvre avec les informations du client.

5a - La commande est à livrer. La commande est faite. Le relais passe au livreur. Les informations du client (coordonnées, téléphone…) apparaissent. La commande est livrée.

Les posts-conditions

Le client voit sa commande valider et en attente.

La commande passe du statut “commande en attente” à “commande en cours”. Les commandes finies sont dans l'historique. Les ingrédients qui ont été utilisé se soustraient du stock.

Ergonomie

Les listes de commande sont disposées de 10 en 10 sur de nouvelles pages.

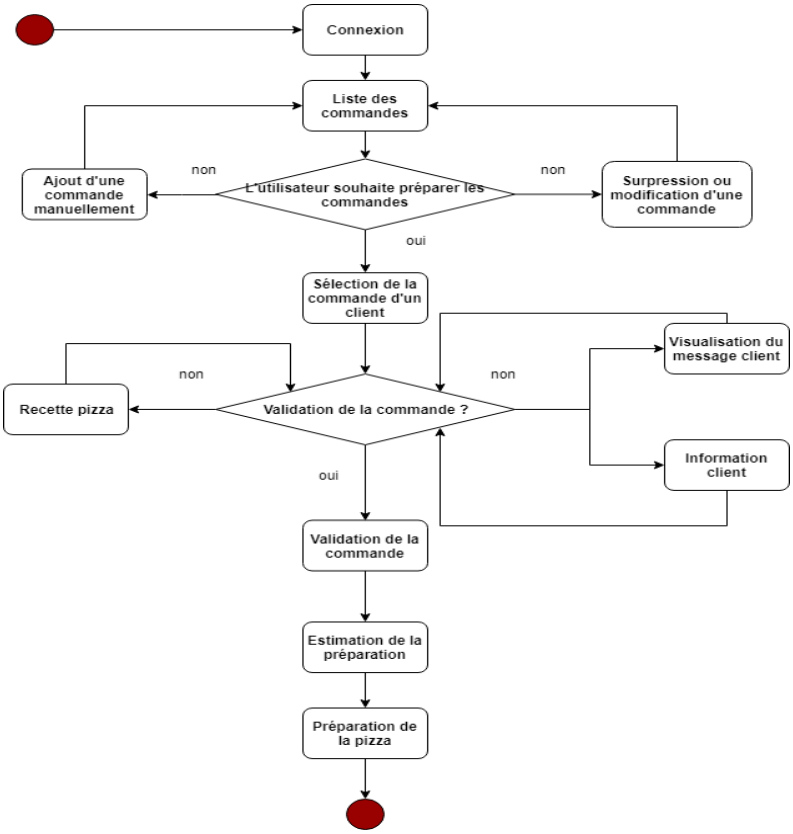


Diagramme d’activité

du cas d’utilisation “Gérer les commandes”

cas numéro 10: visualiser les informations de commande

Nom : visualiser Les informations de commande
Acteur : Le serveur
Description : L'utilisateur doit pouvoir voir les informations de la commande
Auteur : JBS
Date : 02/10/2018
Pré-conditions :Une commande vient d'être finie en cuisine.
Démarrage : Appuyer sur une des commandes finies.

Le scénario nominal

1 - Le serveur appuie sur une des commande finies.

2 - Un résumé de la commande apparaît avec le prix.

3 - Il appuie sur Information client.

4 - Toutes les informations apparaissent.

5 - L'utilisateur retourne sur la liste des commandes finies et clique sur terminer commande.

cas numéro 11: visualiser les informations de commande

Nom : visualiser Les informations de commande
Acteur : Le livreur
Description : L'utilisateur doit pouvoir voir les informations de la commande
Auteur : JBS
Date : 02/10/2018
Pré-conditions :Une commande vient d'être finie en cuisine.
Démarrage : Appuyer sur une des commandes finies.

Le scénario nominal

1 - Le livreur appuie sur une des commande finies.

2 - Les Informations du client apparaissent ainsi que la facture.

3 - L'utilisateur retourne sur la liste des commandes finies et clique sur terminer commande.

Le scénario alternatif

1a - Il s'est trompé de commande, il appuie sur retour.

3a - L'utilisateur est sur mobile. Il retourne sur la liste des commandes finies et appuie quelques secondes dessus et clique sur terminer commande.

Les posts-conditions

Des commandes finies s'affichent.

Ergonomie

La liste des commandes finies sont affichées 10 par 10 de la plus vieille a la plus récente Avec l'id de commande.

cas numéro 12 : connexion

Nom : Connexion

Acteur : Le personnel plus

Description : "Le personnel plus" doit pouvoir se connecter afin de pouvoir accéder aux différentes fonctionnalités de l'employé plus qualifié.

Auteur : JBS

Date : 15/10/2018

Pré-conditions : Il faut s'être connecté.

Le scénario nominal

1 - L'utilisateur appuie sur "se connecter".

2 - Le système affiche une page avec quatre commandes: "*gestion du personnel*", "*gestion de commande*", "gestion du *stock*" ou "gestion des *réclamations*".

Le scénario alternatif

2a - Il ne rentre pas les bons identifiants. Il doit alors les re-saisir.

Les posts-conditions

L'utilisateur se connecte et accède aux fonctionnalités citées ci dessus.

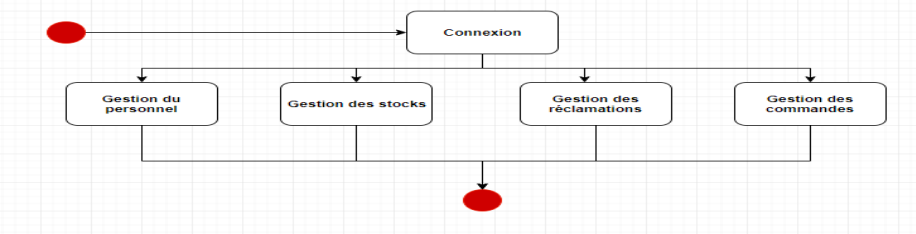


Diagramme d'activité
du cas d'utilisation "connexion du personnel"

cas numéro 13: gestion de profil

Nom : Gestion de profil

Acteur : Le personnel

Description : "Le personnel plus" doit pouvoir gérer les profils des employés.

Auteur : JBS

Date : 15/10/2018

Pré-conditions : Il faut s'être connecté et avoir appuyé sur la commande "gestion de profil".

Le scénario nominal

1 - Le système affiche une page ou il y a trois commandes : "*Création de compte*", "*modification de compte*" et "*suppression de compte*".
Il appuie sur création de compte.
2 - Une nouvelle page apparaît, il saisit les informations du nouvel employé puis les valide.
3 - Une nouvelle page apparaît avec un message: "Création validée" en vert.

Le scénario alternatif

1a - Il appuie sur une des 2 autres fonctionnalités.
2b - Il se déconnecte ou quitte le site.
3a - Il clique sur modification de compte. La liste des comptes du personnel s'affiche. Il peut appuyer sur un des

comptes. Il peut alors modifier les informations du compte.
3b - Il clique sur supprimer un compte. La liste des comptes du personnel s'affiche, il peut alors supprimer un compte.

Les posts-conditions
Un membre du personnel à un compte et peut se connecter.

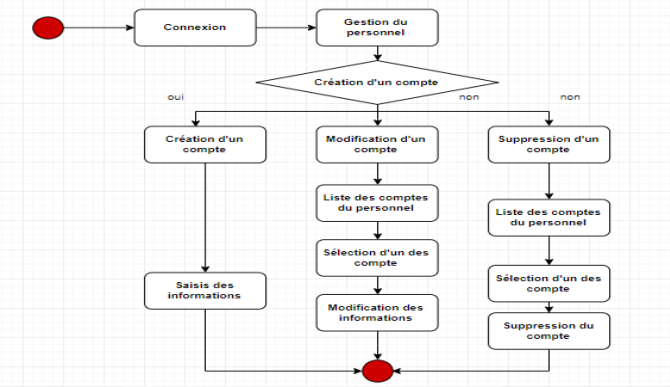


Diagramme d'activité
du cas d'utilisation "gestion de profil du personnel"

cas numéro 14: gestion du stock

Nom : Gestion du stock

Acteur : Le personnel

Description : "Le personnel plus" doit pouvoir gérer les stocks.

Auteur : JBS

Date : 15/10/2018

Pré-conditions : Il faut s'être connecté avec un compte "personnel plus" et avoir cliqué sur "gestion du stock".

Le scénario nominal
1 - Le stock apparait sous forme de liste. Il appuie sur "*passer une commande*" ou consulter les stocks. Il clique sur passer une commande
2 - Il est dirigé vers le site de l'entreprise fournissant.

3 - Il commande.

Le scénario alternatif

1a - Il consulte le stock.

2b - Il se déconnecte, quitte.

3a - Il se déconnecte, quitte.

Problème non résolu

Je ne connais pas le fonctionnement des commandes. Faire appel à une personne qui connaît le système de commande inter-entreprise.

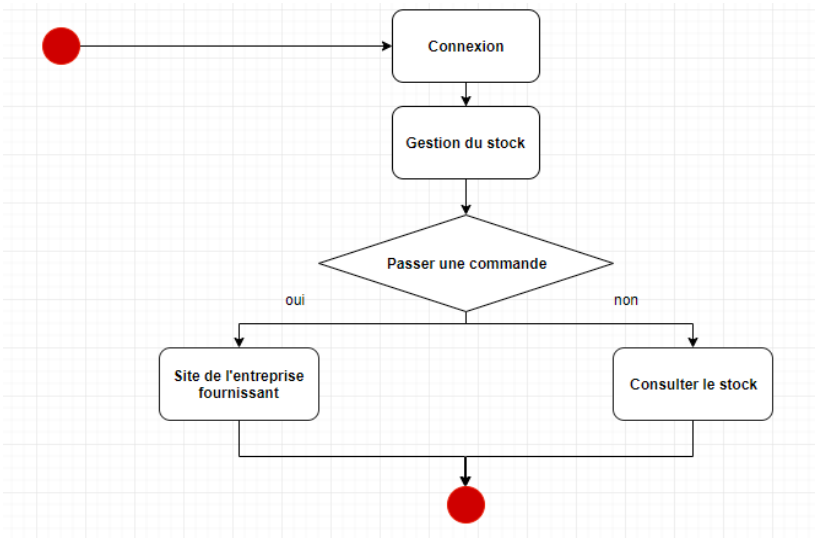


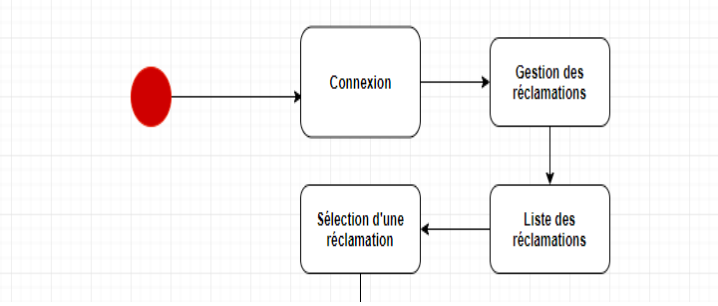
Diagramme d'activité
du cas d'utilisation Gestion du stock

cas numéro 15 : gestion des réclamations

Nom : Gestion des réclamations

Acteur : Le personnel

Description : “Le personnel plus” doit pouvoir gérer les réclamations.



Auteur : JBS

Date : 15/10/2018

Pré-conditions : Il faut s'être connecté et avoir appuyé sur “réclamation”.

Le scénario nominal

1 - Le système affiche une page avec les réclamations qui sont sous forme de liste: *numéro commande, nom,objet, montant*... Il appuie sur une des réclamations.

2 - Une page s’ouvre avec le message de réclamation (A noter qu’il peut accéder aux informations sur client depuis cette page).

3 - Il y répond.

4 - Il clique sur envoyer.

Le scénario alternatif

1a - Il quitte, retourne en arrière ou se déconnecte.

2a - Il quitte, retourne en arrière ou se déconnecte.

2b - Il clique sur Information du compte. Il est dirigé vers les informations du client sous forme de mini page.

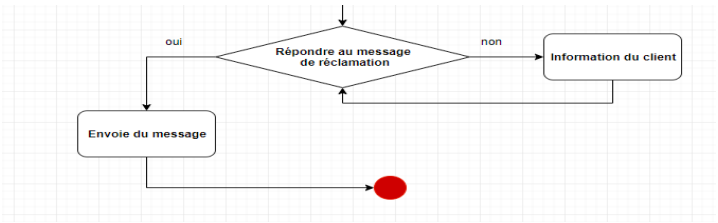
3a - Il oublie de rentrer un objet, alors un message d’alerte en rouge apparaît.

Les Posts-conditions

Le compte client reçoit un message.

Diagramme d'activité

du cas d'utilisation Gestion des réclamations



7 - APPLICATION GESTION DE COMMANDE

Il y a donc une application web centrée sur le besoin du client et une application logicielle pour le cuisinier, le directeur et le serveur s'il est à la caisse. Le serveur à table et le livreur auront tous les deux une application mobile. Pour cela nous aurons besoin de:

Matériel: *ou hardware* n'importe quel appareil ayant un navigateur et ayant accès à internet pour la partie site web.
Pour la partie "application", un ordinateur sera nécessaire afin d'installer l'application dédiée aux employés et leurs permettant de gérer l'ensemble de l'activité des pizzerias.L'application logicielle pourra aussi être fait en version mobile, un mobile sera nécessaire et pour les serveurs et pour les livreurs. Ces applications ne seront pas téléchargeables, et nous les installerons sur l'ensemble des ordinateurs et mobiles nécessaires. A noter qu'un ordinateur minimum devra être présent par pizzeria, et plusieurs peuvent être mis en place, en fonction du nombre d'employés qui vont être amenés à travailler sur l'application.

Site Web: Le site internet sera codé en python 3. Les pages seront faites en HTML. Le framework sera supporté par Django.

Application: Les applications seront codés en python 3.

La base de donnée : Les données (liste des pizzas, prix, ingrédients) devront être introduites par le personnel qualifié d'OC PIZZERIA. Le système informatique traitera lui même la gestion du stock. La base de donnée sera mise en place avec MySQL, sera relationnelle, et sera commune, à la fois à l'application pour les employés, et au site web pour les clients.

Humain: Le personnel devra suivre une formation d'au moins un jour pour se familiariser avec le système informatique.
Il y aura plusieurs types de formations selon le métier afin de pouvoir se servir des applications sur mobile et sur ordinateur.

A noter que les zonings ne sont en rien représentatifs du produit final. Ils sont là afin que vous puissiez imaginer le produit final.

8 - GLOSSAIRE
