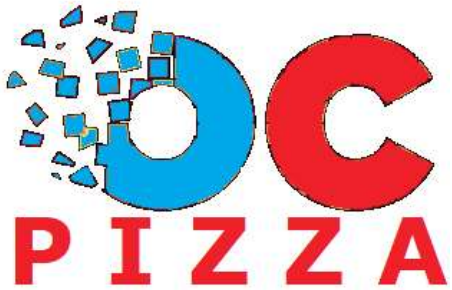


Spécifications techniques d'un système informatique.

Historique des versions :

La version en cours est la dernière affichée.

- Version 1.0 du 04/10/18

Plan :

I) Architecture du modèle de données.

1) Le diagramme de classes.

2) Le modèle physique de données.

II) Architecture des composants de l'application.

1) Le diagramme de composants.

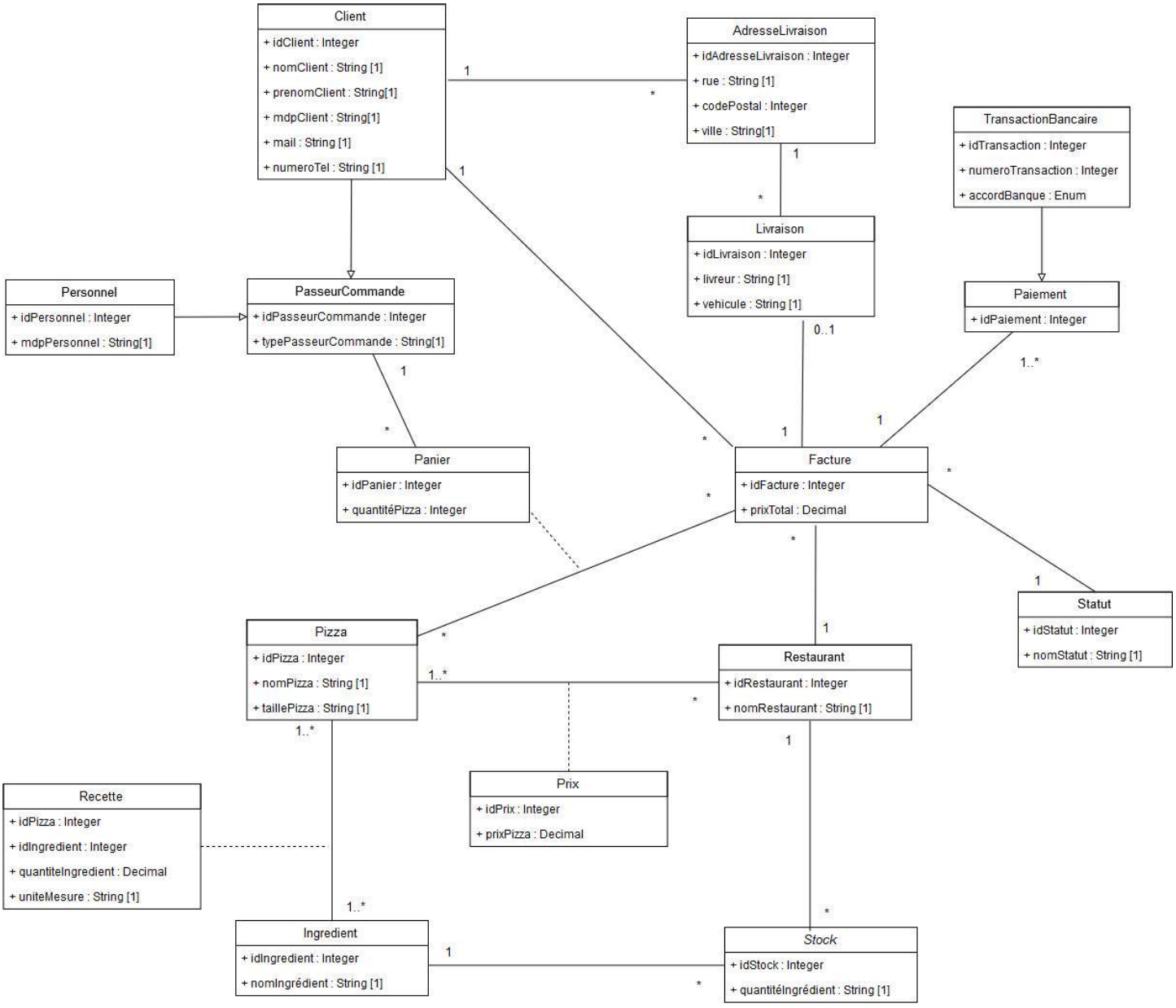
III) Architecture de déploiement.

1) Etude de solutions.

2) Le diagramme de déploiement.

I) Architecture du modèle de données.

1) Le diagramme de classes.



Sur ce diagramme, nous voyons toutes les classes que nous avons prévu de créer pour la construction de la base de données. Ainsi que les relations qu'elles entretiennent l'une avec l'autre, agrémentées de la cardinalité.

Explication des classes :

1) La classe "Client" représente l'ensemble des clients enregistrés sur la base de données, c'est-à-dire les clients ayant enregistré leur profil et informations personnelles depuis le site internet.

2) La classe "Personnel" représente les salariés du groupe OC Pizza qui enregistrent les commandes sur place, pour le compte des clients (enregistrés ou non).

3) La classe "PasseurCommande" est la classe mère des classes "Client" et "Personnel" et représente une personne effectuant une commande.

4) La classe "Pizza" représente toutes les pizzas théoriquement disponibles dans un restaurant.

5) La classe "Ingredient" liste tous les ingrédients nécessaires à la réalisation de toutes les pizzas.

6) La classe "Recette" est la rencontre des deux précédentes classes : quels ingrédients sont nécessaires pour quelle pizza.

7) La classe "Restaurant" représente les différentes pizzerias de l'enseigne.

8) La classe "Prix" est la jonction entre quel type de pizza et quel restaurant : les prix peuvent varier selon la position géographique, les marges calculées des pizzerias, les promotions actives, etc...

9) La classe "Stock" représente le stock physique d'ingrédients par restaurant.

10) La classe "Facture" est au centre et représente les commandes en cours de traitement ou qui ont déjà été traitées.

11) La classe "Panier" est l'intersection entre les classes "Pizza" et "Facture".

12) La classe "Paiement" indique le ou les modes de paiement(s) effectué(s) par facture.

13) La classe "TransactionBancaire" représente la liaison avec le service bancaire externe en cas de paiement par carte bancaire, et est une classe fille de "paiement".

14) La classe "Livraison" représente le fait de livrer une commande à domicile.

15) La classe "AdresseLivraison" permet au client de choisir une adresse où se faire livrer.

16) La classe "Statut" récapitule le statut d'une commande à l'instant t.

2) Le modèle physique de données.

Le modèle physique de données montre toutes les classes qui seront matérialisées par des tables dans la base de données, et les colonnes constituant ces tables. Pour des raisons de lisibilité, il est joint au présent document en annexe, sous format pdf.

Notes sur les mots de passe :

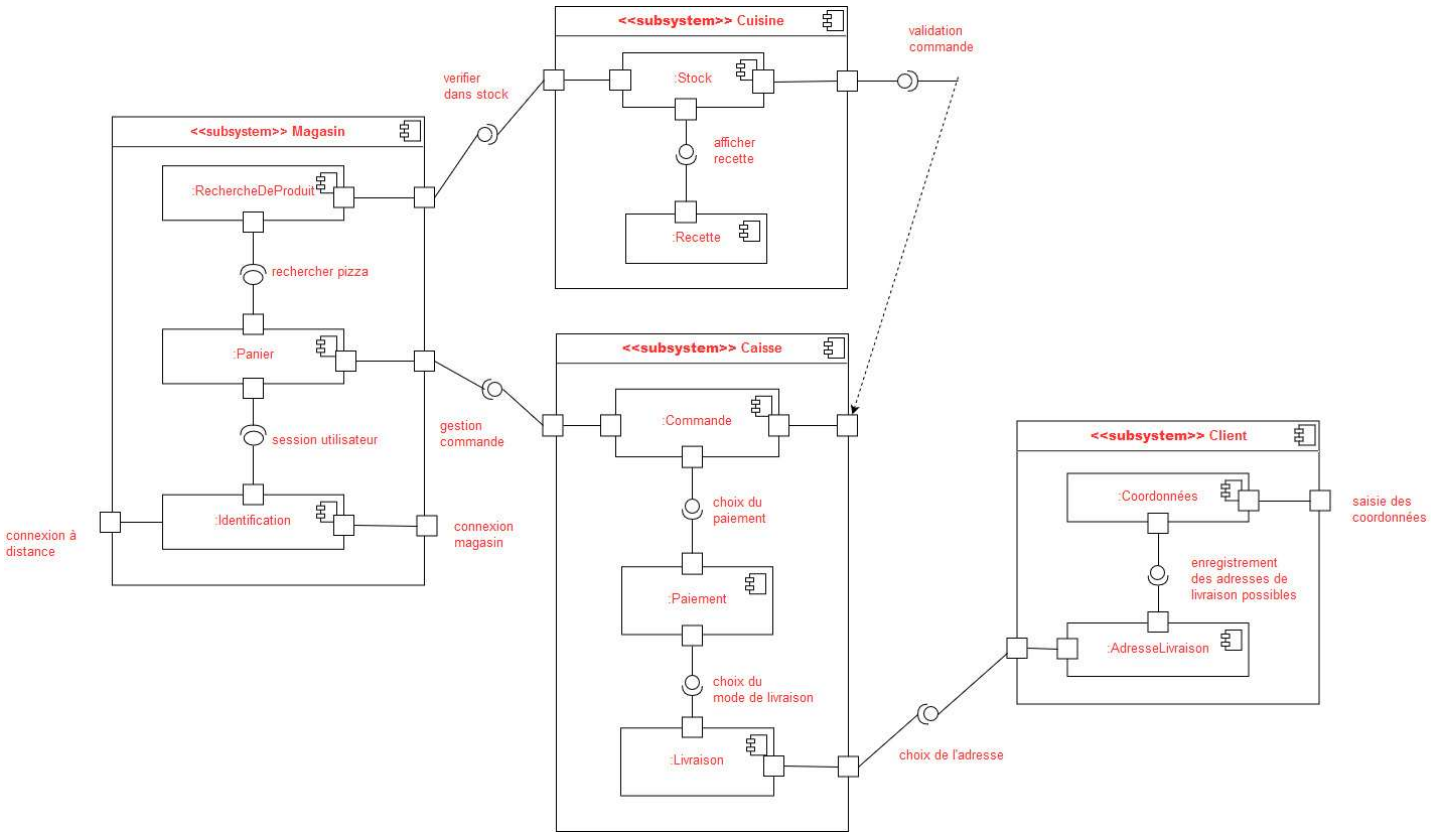
Concernant les classes "client" et "personnel", il est évident que dans la réalité, les mots de passe doivent être cryptés. Pour les besoins de la phase de test nous avons choisi de ne pas les crypter. Lors du déploiement en production, nous utiliserons une librairie python (passlib) pour crypter les mots de passe, et dont le hash sera une chaîne de longueur 255. Nous devrons installer la librairie passlib grâce à pip, et choisir un des algorithmes de hash (par exemple pbkdf2_sha256) pour permettre de crypter les mots de passe ; puis de vérifier qu'un mot de passe saisi correspond bien au hash d'un mot de passe enregistré grâce aux fonctions nom_algoritme.hash("motdepasse") et nom_algorithme.verify("motdepasse", hash). Notons un autre algorithme de hash, argon2, qui est plus récent (2013) mais qui tendra à devenir la référence pour le hash de mot de passe dans les prochaines années.

Note sur les données de test :

Pour nos besoins de test, nous avons choisi de générer des données de test grâce à mockaroo.com. Ce site permet de générer aléatoirement des données selon des catégories prédéfinies et construit le code SQL en fonction pour une insertion facilitée dans la base de données. Ainsi nous avons généré quelques dizaines de clients, membres du personnel, adresses, numéros de téléphone, mots de passe, etc... Cependant, pour certaines classes (pizza, statut, par exemple), les données de test ont été choisies par nos soins.

II) Architecture des composants de l'application.

1) Le diagramme de composants.



Ainsi modélisé, le diagramme de composants décrit quelles sont les relations entre les éléments d'un composant et les services offerts par les éléments d'un autre composant.

Explication des interfaces :

Chaque interface présente sur le diagramme de composants peut-être divisée en plusieurs fonctionnalités. On retrouve ces fonctionnalités dans les descriptions textuelles des cas d'utilisation du document présenté au client il y a quelques mois. A titre d'exemple, nous avons listé quelques cas d'utilisation précédemment décrits dans le document "Règles de gestion fonctionnelle d'un système informatique" pouvant se rapporter à ces interfaces.

1) Connexion à distance : "Créer un compte", "Se connecter", "Récupérer identifiants perdus"

2) Connexion magasin : "Créer un compte équipe", "Créer un compte responsable d'enseigne", "Gérer les accès interfaces"

3) Session utilisateur : "Se connecter", "Se déconnecter", "Changer de mot de passe"

4) Rechercher pizza : Cette interface devrait être traduite par un menu déroulant.

5) Vérifier dans stock : "Suivre les ingrédients disponibles", "Modifier la liste des pizzas"

6) Gestion commande : "Modifier statut de la commande", "Annuler ou modifier une commande", "Consulter une commande"

7) Afficher recette : "Consulter l'aide mémoire"

8) Choix du paiement : "Payer en ligne", "Payer à la livraison"

9) Choix du mode de livraison : "Attribuer un livreur", "Modifier statut de la commande au départ de la livraison"

10) Validation commande : "Constituer panier"

11) Choix de l'adresse : "Payer à la livraison", "Livrer une pizza"

12) Saisie des coordonnées : "Créer un compte",

13) Enregistrement des adresses de livraison possibles : "Enregistrer coordonnées personnelles complémentaires"

III) Architecture de déploiement.

1) Etudes de solutions.

Dans cette partie, nous abordons la solution technique que nous préconisons pour le déploiement en production de l'application.

La solution logicielle sera développée en Python avec le framework Django, comme expliqué précédemment. Pour le déploiement physique de l'application dans les pizzerias, il convient d'étudier plusieurs possibilités :

Solution 1 : achat de 2 serveurs dédiés par pizzeria.

Prix : à partir de 150€ le serveur.

Avantages : dépense maîtrisée. Pas de connexion internet obligatoire. Indépendant de la connexion à un serveur externe.

Inconvénients : nécessité d'avoir du personnel qualifié pour la maintenance, ou de prendre l'option payante de service après-vente de nos services (prix : cf proposition commerciale). Nécessité d'avoir 2 serveurs physiques dédiés en local par pizzeria pour faire de la réplication (avoir un backup en cas de panne). Il faut prévoir un emplacement physique de type salle fermée ou grand placard pour installer le matériel.

Le protocole wsgi est une norme qui permet à une application web python et un serveur web de communiquer.

a) Installation avec apache2 et mod_wsgi :

Il faut préalablement installer django et apache2, puis le module wsgi.

b) Installation avec nginx et gunicorn :

Solution légère et facile à paramétrer, utilisation de nginx comme un serveur web. D'abord lancer un serveur http (nginx), puis un serveur wsgi (gunicorn), qui est lié à l'application django en wsgi.

Solution 2 : location d'un serveur auprès d'un datacenter.

Prix : Ils sont très variables d'un prestataire à l'autre. Voici un comparatif de quelques datacenters les plus courants.

alwaysdata : de gratuit pour 100Mo de disque et 1 Go de trafic mensuel à 25€ par mois pour 50Go de stockage et 500Go de trafic mensuel sur un serveur mutualisé; à partir de 150€ par mois pour un serveur virtualisé (dans notre cas, il faut prévoir minimum 50Go de stockage).

heroku : à partir de 25$ par mois.

djangoeurope : de 5 à 25€ par mois. Spécialiste des projets Django.

ovh : 60€ HT par mois.

Avantages : Pas besoin de personnel qualifié pour la maintenance, c'est le prestataire qui s'en charge. Possibilité de modifier la capacité du serveur en contactant directement le prestataire.

Inconvénients : Dépense continue (au mois ou à l'année) et connexion internet obligatoire. Si le serveur central du prestataire est hors service, aucune pizzeria ne pourra faire tourner la solution logicielle, empêchant toute commande.

Solution 3 : location de services cloud (Amazon Web Services).

Cette solution consiste à passer par les services en ligne d'Amazon Web Services. Il est alors possible d'y déployer l'intégralité de la solution : base de données, applications, etc...

Avantages : Pas besoin de personnel qualifié pour la maintenance, les prestations sont fournies par AWS. Pas besoin de matériel spécifique (serveurs). La garantie d'avoir un service support de qualité toujours disponible, du matériel à disposition à la pointe de la technologie, et aucune coupure d'accès en cas de mise à jour ou de changement de serveurs. L'offre peut évoluer en fonction de la croissance (ou décroissance) du volume de données transférées.

Inconvénients : Le prix peut être très variable. Tout dépend de la quantité d'options choisies, et du volume de données transférées mensuellement, mais cela peut aller de quelques dizaines d'euros à plusieurs centaines d'euros par mois (voire bien plus pour les gros projets). Le choix des options est très fourni et il peut être compliqué d'y voir clair.

Solution 4 : location de services cloud (Digital Ocean).

C'est le même principe que pour la solution précédente. A partir de 5$ par mois pour un stockage de 25 Gb, un total de transfert n'excédant pas 1 Tb par mois avec 1 Gb de mémoire.

Avantages : Idem que pour Amazon Web Services. Le prix est cependant moindre : à partir de 5$ par mois. Soit environ l'équivalent de la solution 1, à durée comparable. Possibilité d'augmenter la taille du stockage, de la mémoire ou des transfert par tranche de 5$ supplémentaires.

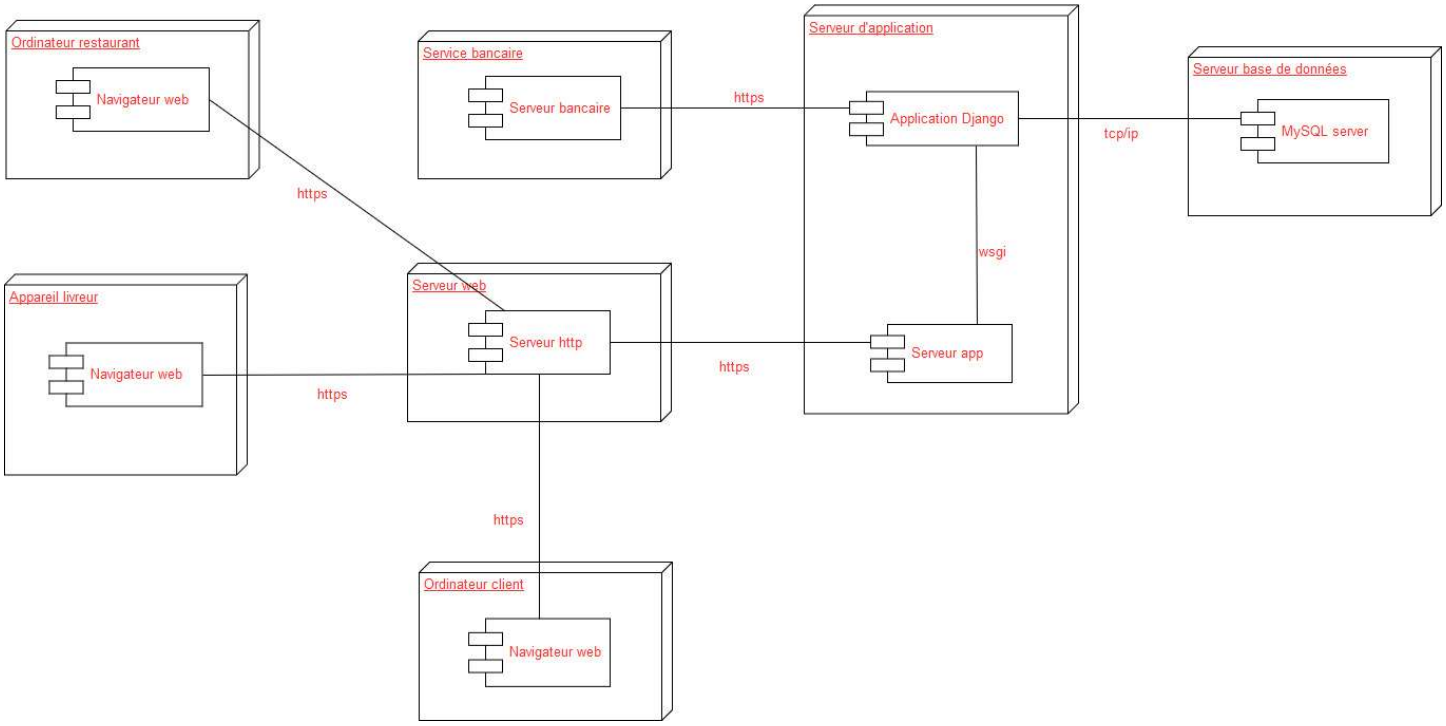
Inconvénients : Dépense continue et connexion à internet obligatoire. En cas de coupure (côté client), aucune pizzeria ne pourra faire tourner la solution logicielle. Pas de prestations après-vente de notre part, le support est assuré par le support de DO.

Solutions choisies :

Etant donné les contraintes physiques (les pizzerias sont éloignées les unes des autres) budgétaires et structurelles (une coupure d'internet empêcherait la solution de fonctionner), nous préconisons la solution d'achat de serveurs dédiés par pizzeria pour un budget minimal de 500€ de matériel par pizzeria. A remplacer de préférence tous les 5 ans. C'est également une solution plus économique que de payer une location mensuelle moyenne de 25€ pendant cette même période.

Nous préconisons également la solution d'installation avec nginx et gunicorn, qui est une installation relativement facile à mettre en place et à paramétrer (c'est une solution très souvent choisie de nos jours).

2) Le diagramme de déploiement.



Le diagramme de déploiement décrit la disposition physique des ressources matérielles constitutives du système. Chaque ressource est représentée par un nœud qui contient un ou plusieurs composants. Chaque association entre deux nœuds représente un chemin de communication qui permet l'échange d'informations. Ici nous reprenons les choix effectués ci-dessus, à savoir l'installation de serveurs dans le magasin et l'utilisation de nginx et gunicorn.