CONCEPTION DE LA SOLUTION TECHNIQUE D'UN SYSTEME DE GESTION DE PIZZERIA

Définir le domaine fonctionnel et concevoir l'architecture technique de la solution répondant aux besoins du client

Spécifications techniques

Table des matières

1 Introduction

Contexte :

« OC Pizza » 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 responsables du groupe a pris contact avec vous afin de mettre en place un système informatique, 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'ils le souhaitent – 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 trouvé ne lui conviennent pas.

Objet du document *Livrables*

- *Un document (format PDF) de spécifications techniques comprenant :*
    - *une description du domaine fonctionnel*
    - *les différents composants du système et les composants externes utilisés par celui-ci et leur interaction*
    - *la description de l'organisation physique de ces composants (déploiement)*
- *Le modèle physique de données (MPD) ou modèle relationnel de la base de données au format PDF ou image (PNG)*
- *Base de données PostgreSQL avec un jeu de données de démo :*
    - *un fichier ZIP contenant un dump de votre base de données*
    - *un fichier ZIP contenant l'ensemble des scripts SQL de création de la base de données et du jeu de données de démo*

2 Cahier de charges

Objectif : Définir le domaine fonctionnel et concevoir l'architecture technique de la solution répondant aux besoins du client.

Objectifs opérationnels :

1. obtenir un outil permettant une **gestion efficace des commande**s de pizzas à tout niveau.
2. améliorer la **gestion du stock et son suivi en temps réel**.
3. permettre à tout moment de **consulter la liste des commandes** passées en préparation et à venir.
4. **consulter la recette** de la pizza.
5. permettre aux clients d' « OC Pizza » :
    1. de **passer leur commande via un site internet**.
    2. de **régler en ligne cette commande**.
    3. **annuler leur commande** tant que celle-ci n'est pas en phase de préparation.

Contraintes

1. La solution proposée doit donc tenir compte de la contrainte de multi établissement.
2. la conception d'un système de commande à distance et de livraison implique de définir des règles d'attribution des commandes à un lieu de préparation et à un livreur.

3 Domaine fonctionnel

a. Diagramme de Classe

La structure de base des données est modélisée selon le diagramme UML suivant (fig.),

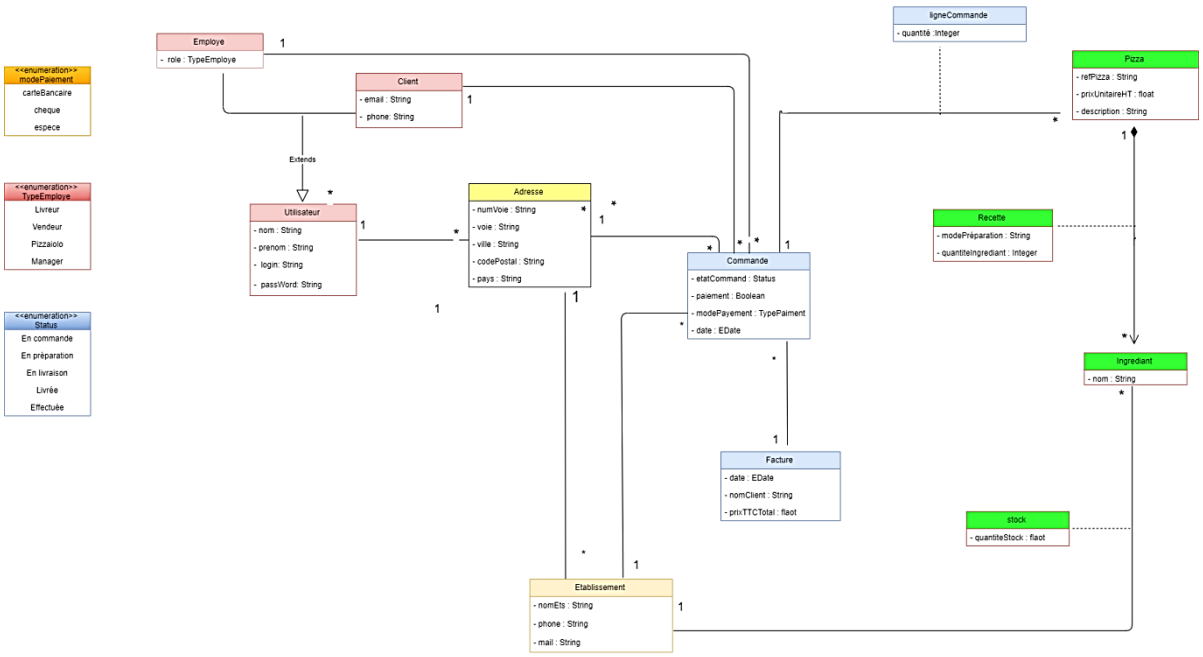

Fig 1 Diagramme des Classes OCPIZZA

b. Description des Tables et *Justification des cardinalités*

1. La classe Utilisateur

utilisateur (Physical Name: utilisateur)

Logical Column Name	Physical Column Name	Type	PK	Nullable
id (PK)	id	INTEGER	PK	NOT NULL
password	password	VARCHAR(10)		NOT NULL
nom	nom	VARCHAR(100)		NOT NULL
prenom	prenom	VARCHAR(100)		NOT NULL
login	login	VARCHAR(10)		NOT NULL
adresse_id (FK)	adresse_id	INTEGER		NOT NULL

References

- adresse through (adresse_id)

Referenced By

- employe referencing (id)
- client referencing (id)

Permet de stocker les données de tous les utilisateurs de l'application, c'est la classe dont hérite la classes client et la classe employé.
Utilisateur_id est clé primaire de type Integer.
Cardinalité **Un à plusieurs**.
Utilisateur 1—1..* Adresse.

2. La classe Client :

Classe extend de la Classe utilisateur

client (Physical Name: client)

Logical Column Name	Physical Column Name	Type	PK	Nullable
utilisateur_id (PK) (FK)	utilisateur_id	INTEGER	PK	NOT NULL
email	email	VARCHAR(100)		NOT NULL
phone	phone	VARCHAR(10)		NOT NULL

References

- utilisateur through (utilisateur_id)

Referenced By

- commande referencing (utilisateur_id)

3. La classe Employe:

Classe extend de la Classe utilisateur

employe (Physical Name: employe)

Logical Column Name	Physical Column Name	Type	PK	Nullable
utilisateur_id (PK) (FK)	utilisateur_id	INTEGER	PK	NOT NULL
role	role	VARCHAR(0)		NOT NULL

References

- utilisateur through (utilisateur_id)

Referenced By

- commande referencing (utilisateur_id)

4. La classe établissement:

Classe qui identifie les restaurants du groupe.
Cardinalité **un à plusieurs** .
Adresse **1 – 1 .. *** établissement.

etablissement (Physical Name: etablissement)

Logical Column Name	Physical Column Name	Type	PK	Nullable
id (PK)	id	INTEGER	PK	NOT NULL
commande_id (FK)	commande_id	INTEGER		NOT NULL
adresse_id (FK)	adresse_id	INTEGER		NOT NULL
nomets	nomets	VARCHAR(100)		NOT NULL
email	email	VARCHAR(100)		NOT NULL
phone	phone	VARCHAR(10)		NOT NULL

References

- adresse through (adresse_id)
- commande through (commande_id)

Referenced By

- stock referencing (id)

5. La classe Commande :

Classe qui identifie les commandes des clients.
Cardinalité **un à plusieurs** .
Command **1 – 1 .. *** produit.
Une commande peut contenir plusieurs produits, alors qu'un produit ne peut être motionner qu'une seule fois dans une ligne de commande pour un même client.

commande (Physical Name: commande)

Logical Column Name	Physical Column Name	Type	PK	Nullable
id (PK)	id	INTEGER	PK	NOT NULL
employe_id (FK)	employe_id	INTEGER		NOT NULL
client_id (FK)	client_id	INTEGER		NOT NULL
adresse_id (FK)	adresse_id	INTEGER		NOT NULL
etatcommande	etatcommande	VARCHAR(0)		NOT NULL
paiement	paiement	BOOLEAN		NOT NULL
modepaiement	modepaiement	VARCHAR(0)		NOT NULL
date	date	TIMESTAMP		NOT NULL

References

- client through (client_id)
- employe through (employe_id)
- adresse through (adresse_id)

Referenced By

- lignecommande referencing (id)
- facture referencing (id)

6. La classe LigneCommande :

Classe d'association entre classe command et classe Pizza. Elle permet de spécifier plusieurs lignes de commande.

lignecommande (Physical Name: lignecommande)

Logical Column Name	Physical Column Name	Type	PK	Nullable
command_id (PK) (FK)	command_id	INTEGER	PK	NOT NULL
produit_id (PK) (FK)	produit_id	INTEGER	PK	NOT NULL
quantite	quantite	VARCHAR(0)		NOT NULL

References

- commande through (command_id)
- pizza through (produit_id)

7. La classe facture :

Cette classe décrie l'objet facture.
Cardinalité **un à plusieurs**.
Facture **1 – 1 .. *** Commande

facture (Physical Name: facture)

Logical Column Name	Physical Column Name	Type	PK	Nullable
id (PK)	id	INTEGER	PK	NOT NULL
command_id (FK)	command_id	INTEGER		NOT NULL
date	date	TIME		NOT NULL
nomclient	nomclient	VARCHAR(100)		NOT NULL
prixtotalttc	prixtotalttc	DECIMAL(6,2)		NOT NULL

References

- commande through (command_id)

8. La classe Pizza :

Classe décrivant le produit pizza composite de la classe ingrédient.
Cardinalité **un à plusieurs** .
Pizza **1 – 1.. *** Ingrédient.

pizza (Physical Name: pizza)

Logical Column Name	Physical Column Name	Type	PK	Nullable
id (PK)	id	INTEGER	PK	NOT NULL
refpizza	refpizza	VARCHAR(100)		NOT NULL
prix_unitaire_ht	prix_unitaire_ht	DECIMAL(4,2)		NOT NULL

Referenced By

- lignecommande referencing (id)
- recette referencing (id)

9. La classe Recette :

Classe décrivant le mode de préparation du produit.
C'est une classe d'association entre la classe Pizza et la classe ingrédients.
Cardinalité **plusieurs à plusieurs.**
Recette * - * Pizza.
Recette * - * Ingrédient.

recette (Physical Name: recette)

Logical Column Name	Physical Column Name	Type	PK	Nullable
pizza_id (PK) (FK)	pizza_id	INTEGER	PK	NOT NULL
ingrediant_id (PK) (FK)	ingrediant_id	INTEGER	PK	NOT NULL
modepreparation	modepreparation	VARCHAR(1000)		NOT NULL
quantiteingrediant	quantiteingrediant	INTEGER		NOT NULL

References

- ingrediant through (ingrediant_id)
- pizza through (pizza_id)

10. La classe Ingrédient :

Cette Classe représente des ingrédients disponibles à un instant donné.
Cardinalité **un à plusieurs.**
Etablissent **1 – 1.. *** Ingrédient.

ingredient (Physical Name: ingredient)

Logical Column Name	Physical Column Name	Type	PK	Nullable
id (PK)	id	INTEGER	PK	NOT NULL
refingrediant	refingrediant	VARCHAR(100)		NOT NULL

Referenced By

- recette referencing (id)
- stock referencing (id)

11. La classe Stock :

Classe d'association entre Classe Ingrédient et Classe Etablissement.

stock (Physical Name: stock)

Logical Column Name	Physical Column Name	Type	PK	Nullable
ingrediant_id (PK) (FK)	ingrediant_id	INTEGER	PK	NOT NULL
etablissement_id (PK) (FK)	etablissement_id	INTEGER	PK	NOT NULL
quantitestock	quantitestock	INTEGER		NOT NULL

References

- etablissement through (etablissement_id)
- ingredient through (ingrediant_id)

c. Modèle Physique des Données

Ce diagramme présente les différentes tables de notre base de données relationnelle ainsi que les attributs de chacune de ses tables.

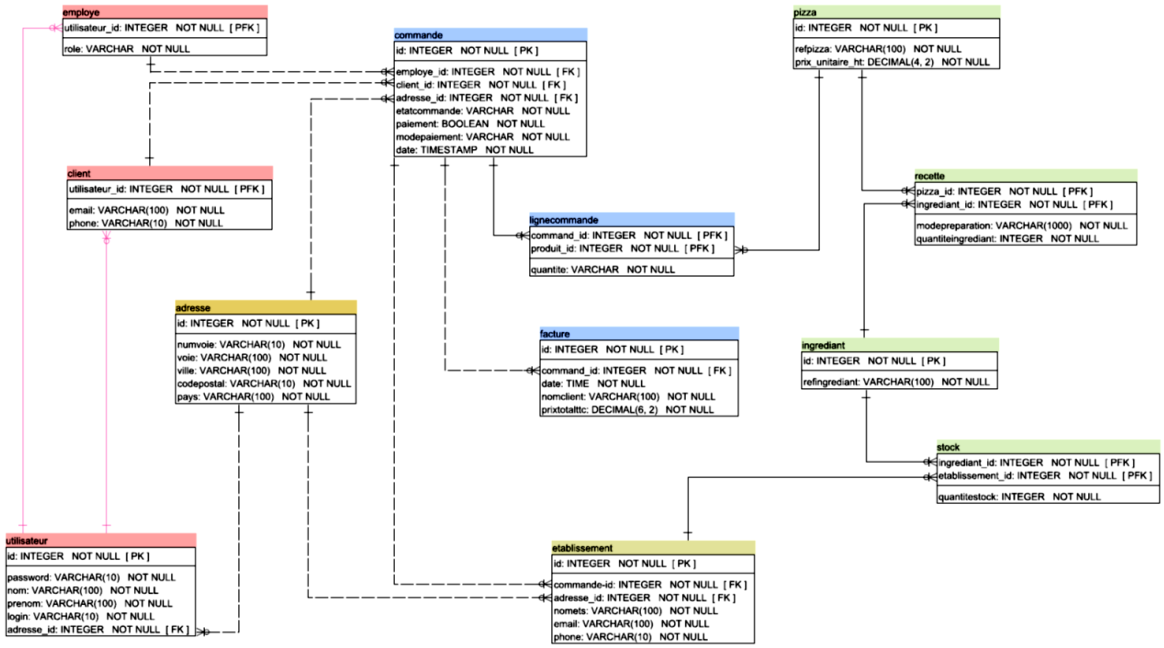

Fig 2 MPD_OCPIZZA

4.1 Diagramme de composants OCPIZZA

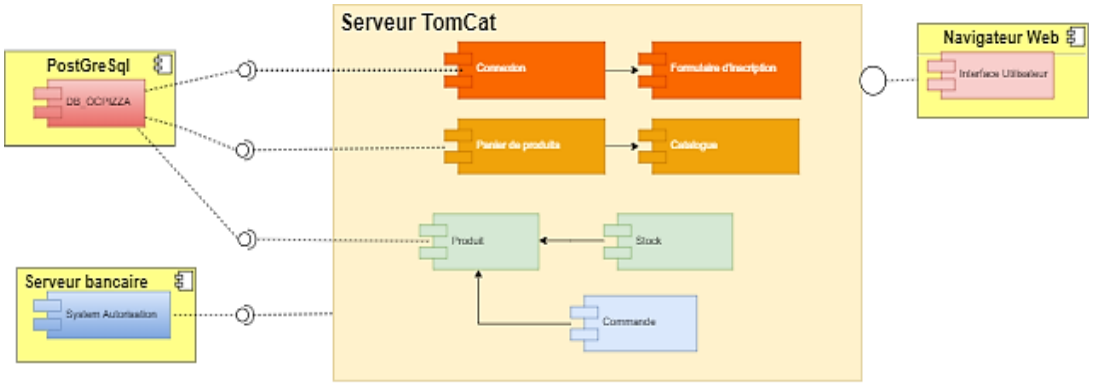

Fig 3 diagramme des composants OCPIZZA

Le **diagramme de composants** décrit l'organisation du système du point de vue des éléments logiciels. Il permet de mettre en évidence les dépendances entre les composants (qui utilisent quoi).

4.2 Diagramme de déploiement OCPIZZA

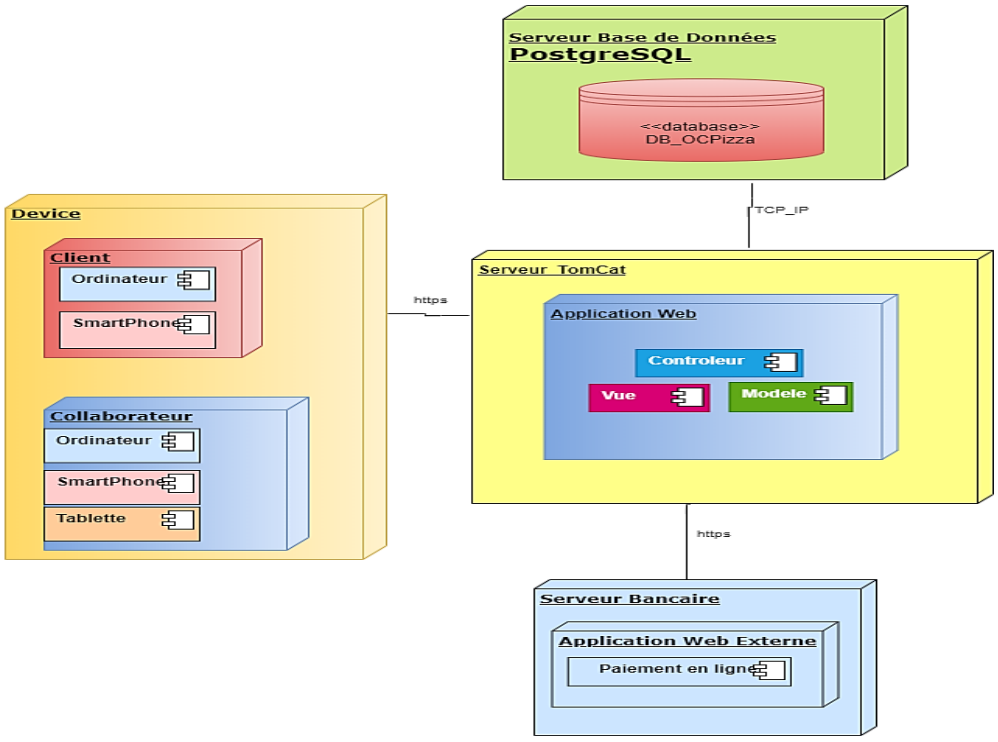

Fig 4 Diagramme de déploiement OCPIZZA

Ce **diagramme de déploiement** est une vue statique qui sert à représenter l'utilisation de l'infrastructure physique par le système et la manière dont les **composants** du système sont répartis ainsi que leurs relations entre eux.

5. Base De Données OCPIZZA

1. Script de création de la base de données OCPIZZA

Exemple Table pizza

```
CREATE SEQUENCE public.pizza_id_seq;

CREATE TABLE public.pizza (
                id INTEGER NOT NULL DEFAULT nextval('public.pizza_id_seq'),
                refpizza VARCHAR(100) NOT NULL,
                prix_unitaire_ht NUMERIC(4,2) NOT NULL,
                CONSTRAINT pizza_pk PRIMARY KEY (id)
);


ALTER SEQUENCE public.pizza_id_seq OWNED BY public.pizza.id;
```

2. Exemple de jeu de données.

```
--
-- TOC entry 2195 (class 0 OID 33220)
-- Dependencies: 185
-- Data for Name: recette; Type: TABLE DATA; Schema: public; Owner: postgres
--

COPY recette (pizza_id, ingrediant_id, modepreparation, quantiteingrediant) FROM stdin;
1	1	md1 10
2	1	md2 10
3	1	md3 10
3	3	md3 6
2	2	md2 8
3	2	md3 8
4	1	md4 10
4	2	md4 8
4	3	md4 6
4	4	md4 4
5	1	md5 10
5	2	md5 8
5	3	md5 6
5	4	md5 4
5	5	md5 2
\.
```

```
--
-- TOC entry 2192 (class 0 OID 33206)
-- Dependencies: 182
-- Data for Name: pizza; Type: TABLE DATA; Schema: public; Owner: postgres
--

COPY pizza (id, refpizza, prix_unitaire_ht) FROM stdin;
1	pizza1	10.00
2	pizza2	20.00
3	pizza3	30.00
4	pizza4	40.00
5	pizza5	50.00
6	pizza6	60.00
\.
```

Exemples de requêtes :

1 _ Affiche les pizza à prix superieur à 30

```
# Affiche les pizza  à prix superieur à 30
SELECT * FROM pizza
WHERE prix_unitaire_ht >30;
```

Sortie de données | Expliquer (Explain) | Messages | Historique

	id integer	refpizza character varying(100)	prix_unitaire_ht numeric(4,2)
1	4	pizza4	40.00
2	5	pizza5	50.00
3	6	pizza6	60.00

2 _ Afficher les ingrédients pour une pizza (avec quantité).

```
# Afficher les ingrédients pour une pizza (avec quantité).
SELECT * FROM recette
WHERE pizza_id IN (
SELECT pizza.id FROM pizza
WHERE pizza.refpizza = 'pizza3'
);
```

Panneau sortie

Sortie de données | Expliquer (Explain) | Messages | Historique

	pizza_id integer	ingrediant_id integer	modepreparation character varying(1000)	quantiteingrediant integer
1	3	1	md3	10
2	3	3	md3	6
3	3	2	md3	8

3_ Jointure entre la table commande et utilisateur

```
SELECT *
FROM commande
JOIN public.client
ON commande.client_id =utilisateur_id
;
```

Panneau sortie

Sortie de données | Expliquer (Explain) | Messages | Historique

	id integer	employe_id integer	client_id integer	adresse_id integer	etatcommande character varying	paiement boolean	modepaiement character varying	date timestamp without time zone	utilisateur_id integer	email character varying(100)	phone character varying(10)
1	2	12	2	2	liv	t	cb	2000-01-02 00:00:00	2	mail2	tel2
2	3	13	3	3	liv	t	esp	2000-01-03 12:00:00	3	mail3	tel3
3	1	11	1	1	liv	t	cb	2000-01-01 11:00:00	1	mail1	tel1