

OC Pizza



Spécifications techniques

Version du document :

Date	Auteur	Description	Version
21/05/19	Sandrine	Initial document	V1

Objet du document :

Ce document permet de dégager les règles de gestion fonctionnelles du projet “système de gestion de pizzas” pour la société OC Pizza, il comprend :

- les différents acteurs interagissant avec le système proposé
- le descriptif des fonctionnalités
- les règles de gestion fonctionnelles
- le cycle de vie des commandes

Contexte :

La société OC Pizza est un jeune groupe de pizzerias 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 nous afin de mettre en place un système informatique, déployé dans toutes ses pizzerias. OC Pizza a déjà fait une petite prospection et les logiciels existants qu'il a pu trouver ne lui conviennent pas.

Objectifs :

L'application que nous proposons permet d'être plus efficace dans la gestion des commandes, de leur réception à leur livraison en passant par leur préparation.

Notre outil permet :

- de suivre en temps réel les commandes passées et en préparation ;
- de suivre en temps réel le stock d'ingrédients restants pour permettre l'actualisation des cartes à pizzas pour n'afficher que celles qui 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

Acteurs/Rôles et fonctionnalités attendues :

Acteurs / Rôles :

- Le/La responsable de toutes les pizzerias
- Le/La manager d'une pizzeria
- Le/La pizzaiolo
- Le/La Standardiste d'une pizzeria
- Le/La livreur
- Le client (ou client potentiel)

Les fonctionnalités attendues côté client :

- Passer une commande
- Créer un compte "client"
- S'authentifier
- Gérer son compte "client"
- Modifier le statut d'une commande

Les fonctionnalités attendues côté personnel :

- S'authentifier :
- Passer une commande
- Modifier le statut d'une commande

Les fonctionnalités attendues côté administration :

- S'authentifier
- Gérer son personnel
- Afficher les statistiques
- Gérer la carte à pizza
- .Gérer le stock

Specifications techniques :

Modélisation du domaine fonctionnel :

- **N°1 Person**

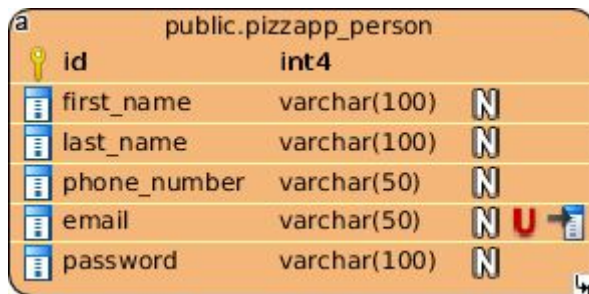


Diagram of the database table `public.pizzapp_person`. The table has the following columns:

Column	Type	Constraints
id	int4	Primary Key
first_name	varchar(100)	Nullable (N)
last_name	varchar(100)	Nullable (N)
phone_number	varchar(50)	Nullable (N)
email	varchar(50)	Nullable (N), Unique (U)
password	varchar(100)	Nullable (N)

Tout acteur est représenté par une personne. Ce modèle est composé d'un champ

- ❖ id (clé primaire),
- ❖ nom, pouvant être null, limité à 100 caractères, non obligatoire,
- ❖ prénom, pouvant être null, limité à 100 caractères, non obligatoire,
- ❖ téléphone, pouvant être null, limité à 50 caractères, non obligatoire,
- ❖ email de connection, pouvant être null, limité à 50 caractères, unique et non obligatoire,
- ❖ password de connection, pouvant être null, limité à 100 caractères et non obligatoire.

Le nom, prénom et téléphone ne sont pas obligatoire, car un client qui commande depuis l'application, et qui prend toujours ses pizzas à emporter, peut ne pas remplir ces informations.

L'email et le password sont également non obligatoire, car un client qui se déplace à la pizzeria pour commander ses pizzas, peut ne pas remplir ces informations.

Le choix de laisser les identifiants de connection sur le modèle personne, est stratégique, car dans toutes les situations, la personne peut fournir son email afin de recevoir les publicités et promotions.

- **N°2 Role**




Diagram of the database table `public.pizzapp_role`. The table has the following columns:

Column	Type	Constraints
id	int4	Primary Key
short_name	varchar(100)	Nullable (N), Unique (U)
description	text	Nullable (N)

Chaque acteur de l'application sera identifié par un rôle. Ce modèle est composé d'un champ

- ❖ nom court qui est unique et non null limité à 100 caractères et obligatoire,
- ❖ description qui peut être nulle et sans limite de caractère.

En fonction du rôle connecté, l'application filtrera les fonctionnalités afin de ne rendre accessible que ce que l'acteur pourra effectuer comme action.

• **N°3 Employed**

Diagram illustrating the structure of the `public.pizzapp_employed` table. The table contains the following fields:

Field	Type	Constraints
<code>id</code>	<code>int4</code>	Primary Key (PK)
<code>is_active</code>	<code>bool</code>	
<code>registration_date</code>	<code>timestamp(35) with time zone</code>	
<code>inactivity_date</code>	<code>timestamp(35) with time zone</code>	Nullable (N)
<code>comment</code>	<code>text</code>	Nullable (N)
<code>person_id</code>	<code>int4</code>	Unique (U)
<code>pizzeria_id</code>	<code>int4</code>	

L'employé peut être un responsable administrateur, un standardiste, pizzaiolo, livreur.

Chaque employé est constitué d'un champ

- ❖ `is_active`, boolean permettant de savoir si l'employé est actif dans la pizzeria, True par défaut
- ❖ date d'enregistrement, datetime avec timezone, défaut datetime de l'instance,
- ❖ date d'inactivation, datetime avec timezone non obligatoire,
- ❖ commentaire lié à l'employé, qui peut être null et sans limite de caractère.
- ❖ personne unique et obligatoire,
- ❖ pizzeria dans laquelle il travail, obligatoire.

L'employé est également constitué d'une

- ❖ liste de rôles, ce qui permet en fonction de ses rôles d'accéder à certaines fonctionnalités.

Diagram illustrating the structure of the `public.pizzapp_employed_roles` table. The table contains the following fields:

Field	Type	Constraints
<code>id</code>	<code>int4</code>	Primary Key (PK)
<code>employed_id</code>	<code>int4</code>	Unique (U)
<code>role_id</code>	<code>int4</code>	Unique (U)

• **N°4 Customer**

public.pizzapp_customer	
id	int4
registration_date	timestamp(35) with time zone
comment	text N
person_id	int4 U

Le customer est le client, il peut commander ses pizzas par téléphone, sur place ou avec l'application. Il est composé d'une

- ❖ date d'enregistrement, au format datetime avec prise en charge de la timezone, défaut datetime de l'instance,
- ❖ commentaire qui peut être nul et sans limite de caractère.
- ❖ personne, qui est unique et obligatoire.

• N°5 Address

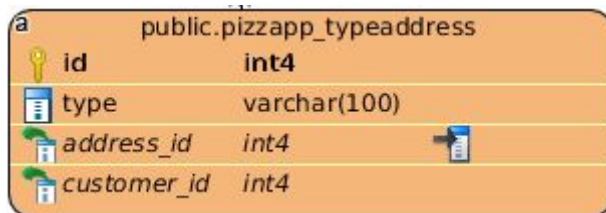
public.pizzapp_address	
id	int4
title	varchar(100) N
address_one	varchar(400)
address_two	varchar(400) N
zip_code	varchar(20)
city	varchar(100)
country	varchar(100) N
location	int4 N
google_place_id	text N U
loc_address	text N

L'adresse est un modèle de données qui peut être géo-localisé par l'API Google. Une adresse peut être utilisée par plusieurs objets (customer, supplier, pizzeria). Elle est composée d'un champ

- ❖ titre, limité à 100 caractère et non obligatoire,
- ❖ première adresse, correspondant au numéro et nom de la rue, obligatoire et limitée à 400 caractères,
- ❖ deuxième adresse, si complément de rue, non obligatoire et limitée à 400 caractères,
- ❖ code postal, obligatoire et limitée à 20 caractères,
- ❖ ville, obligatoire et limitée à 100 caractères,
- ❖ pays, non obligatoire et limitée à 100 caractères,
- ❖ location, qui est un champ point contenant latitude et longitude, élément fourni par l'API Google, non obligatoire
- ❖ identifiant google place, text sans limite de caractère, élément unique fourni par l'API Google, non obligatoire,

- ❖ loc adresse qui correspond à l'adresse complète envoyé à l'API Google contenant numéro de rue, rue, code postal et ville, il n'est pas obligatoire et sans limite de caractère.

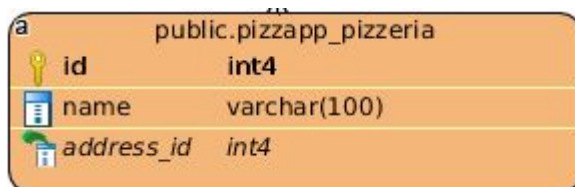
• **N°6 TypeAddress**



Le type d'adresse est un modèle de données, qui permet d'enregistrer plusieurs adresse client avec un type différent. Ce modèle contient un champ

- ❖ `type`, permettant de donner un titre à l'adresse, limité à 100 caractères et obligatoire
- ❖ une adresse, obligatoire,
- ❖ un client, obligatoire.

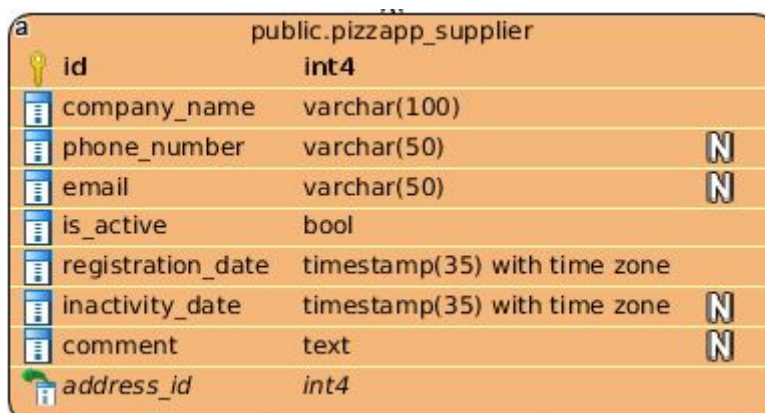
• **N°7 Pizzeria**



La pizzeria est modèle qui contient un champ

- ❖ nom, limité à 100 caractère et obligatoire,
- ❖ adresse, obligatoire.

• **N°8 Supplier**



Le fournisseur est le modèle contenant un champ

- ❖ nom de société limité à 100 caractère et obligatoire,
- ❖ numéro de téléphone, limité à 50 caractère et non obligatoire,
- ❖ email, limité à 50 caractère et non obligatoire,
- ❖ is_active, boolean permettant de savoir si le fournisseur est actif, True par défaut,
- ❖ date d'enregistrement, datetime avec timezone, défaut datetime de l'instance,
- ❖ date d'inactivité, datetime avec timezone, non obligatoire,
- ❖ commentaire, texte sans limite de caractère, non obligatoire,
- ❖ adresse.

• **N°9 Contact**

The diagram shows the structure of the `public.pizzapp_contact` table. It lists the following fields and their properties:

Field	Type	Constraints
<code>id</code>	<code>int4</code>	Primary Key (indicated by a key icon)
<code>is_active</code>	<code>bool</code>	
<code>registration_date</code>	<code>timestamp(35) with time zone</code>	
<code>inactivity_date</code>	<code>date</code>	Nullable (indicated by 'N')
<code>comment</code>	<code>text</code>	Nullable (indicated by 'N')
<code>person_id</code>	<code>int4</code>	Unique (indicated by 'U')
<code>supplier_id</code>	<code>int4</code>	

Le contact est le modèle correspondant aux contacts de personnes travaillant pour un fournisseur. Il contient un champ

- ❖ actif, boolean à True par défaut, permettant de savoir si le contact est actif,
- ❖ date d'enregistrement, datetime avec timezone, défaut datetime de l'instance,
- ❖ date d'inactivité, datetime avec timezone, non obligatoire,
- ❖ commentaire, texte sans limite de caractère, non obligatoire,
- ❖ adresse, obligatoire.
- ❖ personne unique et obligatoire,
- ❖ fournisseur, obligatoire.

• **N°10 UnitOfMeasure**

The diagram shows the structure of the `public.pizzapp_unitofmeasure` table. It lists the following fields and their properties:

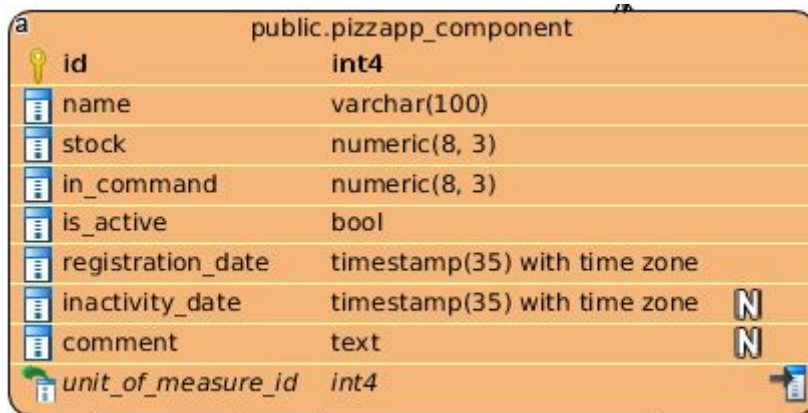
Field	Type	Constraints
<code>id</code>	<code>int4</code>	Primary Key (indicated by a key icon)
<code>short_name</code>	<code>varchar(20)</code>	Unique (indicated by 'U')
<code>description</code>	<code>varchar(100)</code>	Nullable (indicated by 'N')

L'unité de mesure est attribué à chaque composant afin de connaître l'unité dans laquelle on le comptabilise. Ce modèle est composé d'un champ

- ❖ nom court qui est unique et non null limité à 20 caractères,
- ❖ description qui peut être nulle et limité à 100 caractères.

Cette unité de mesure servira pour tous les mouvements de stocks et commandes de composants.

- **N°11 Component**



public.pizzapp_component	
id	int4
name	varchar(100)
stock	numeric(8, 3)
in_command	numeric(8, 3)
is_active	bool
registration_date	timestamp(35) with time zone
inactivity_date	timestamp(35) with time zone
comment	text
unit_of_measure_id	int4

Le composant, est le modèles de données des sous articles qui composent les pizzas mais également de tous les articles qui sont stockés dans les pizzérias en dehors des pizzas. Ce modèle contient un champ

- ❖ nom, limité à 100 caractères et obligatoire,
- ❖ quantité en stock, numérique de 8 chiffres dont 3 décimales, le stock est mis à jour à chaque mouvement de stock, numérique de 8 chiffres dont 3 décimales, obligatoire,
- ❖ quantité en commande, obligatoire, 0 par défaut,
- ❖ date d'enregistrement, datetime avec timezone, défaut datetime de l'instance,
- ❖ date d'inactivité, datetime avec timezone, non obligatoire,
- ❖ commentaire, texte sans limite de caractère, non obligatoire,
- ❖ unité de mesure, obligatoire.

- **N°12 ComponentPrice**

public.pizzapp_componentprice	
id	int4
cost_price_currency	varchar(3)
cost_price	numeric(7, 2)
is_active	bool
registration_date	timestamp(35) with time zone
inactivity_date	timestamp(35) with time zone N
comment	text N
component_id	int4
supplier_id	int4

Le composant prix est un modèle permettant de stocker le prix d'un composant en fonction d'un fournisseur. Un composant peut avoir plusieurs prix différents chez plusieurs fournisseurs. Ce modèle contient un champ

- ❖ currency du prix d'achat, par défaut EUR,
- ❖ prix d'achat, numérique de 7 chiffres dont 3 décimales, obligatoire,
- ❖ is_active, boolean permettant de savoir si le composant est actif, True par défaut,
- ❖ date d'enregistrement, datetime avec timezone, défaut datetime de l'instance,
- ❖ date d'inactivité, datetime avec timezone, non obligatoire,
- ❖ commentaire, texte sans limite de caractère, non obligatoire,
- ❖ composant, obligatoire,
- ❖ fournisseur, obligatoire.










• **N°13 Pizza**

public.pizzapp_pizza	
id	int4
name	varchar(100) U
price_currency	varchar(3)
price	numeric(7, 2)
is_active	bool
registration_date	timestamp(35) with time zone
inactivity_date	timestamp(35) with time zone N
comment	text N

- ❖ nom, limité à 100 caractère, unique et obligatoire,
- ❖ currency du prix de vente, par défaut EUR, obligatoire
- ❖ prix de vente, numérique de 7 chiffres dont 3 décimales, obligatoire,
- ❖ is_active, boolean permettant de savoir si le fournisseur est actif, True par défaut,
- ❖ date d'enregistrement, datetime avec timezone, défaut datetime de l'instance
- ❖ date d'inactivité, datetime avec timezone, non obligatoire,

- ❖ commentaire, texte sans limite de caractère, non obligatoire.









• **N°14 PizzaLine**

a public.pizzapp_pizzaline	
 id	int4
 quantity	numeric(8, 3)
 is_active	bool
 registration_date	timestamp(35) with time zone
 inactivity_date	timestamp(35) with time zone N
 comment	text N
 component_id	int4
 pizza_id	int4
 unit_of_measure_id	int4

Le modèle ligne pizza permet de composer les lignes des composant de la pizza avec les quantité ce qui permet les mouvement de stocks et la mise à jour en temp réel des composants et donc de la mise à jour de la carte à pizza. Ce modèle contient un champ

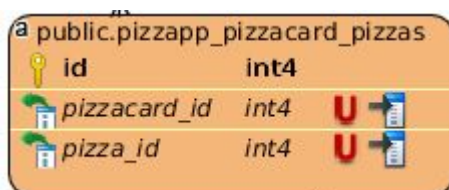
- ❖ quantité du composant, numérique de 8 chiffres dont 3 décimales, obligatoire,
- ❖ is_active, boolean permettant de savoir si la ligne de pizza est active, True par défaut,
- ❖ date d'enregistrement, datetime avec timezone, défaut datetime de l'instance
- ❖ date d'inactivité, datetime avec timezone, non obligatoire,
- ❖ commentaire, texte sans limite de caractère, non obligatoire,
- ❖ composant, obligatoire,
- ❖ pizza, obligatoire,
- ❖ unité de mesure du composant, obligatoire.

• **N°15 PizzaCard**

a public.pizzapp_pizzacard	
 id	int4
 name	varchar(100)
 is_active	bool
 registration_date	timestamp(35) with time zone
 inactivity_date	timestamp(35) with time zone N
 comment	text N
 employed_id	int4
 pizzeria_id	int4

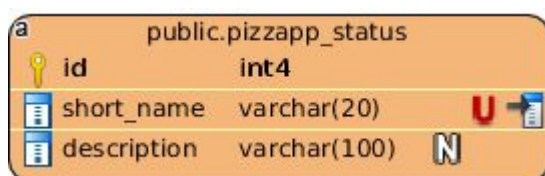
Le modèle de carte à pizza, permet de composer une ou plusieurs cartes à pizzas pour une pizzeria. Une validation métier limitera le nombre de cartes à pizzas active à 1, mais ceci laissera la possibilité aux pizzérias de garder en mémoire des cartes à pizzas non actives. Ce modèle contient un champ

- ❖ nom, limité à 100 caractère, unique et obligatoire,
- ❖ currency du prix de vente, par défaut EUR, obligatoire
- ❖ prix de vente, numérique de 7 chiffres dont 3 décimales, obligatoire,
- ❖ is_active, boolean permettant de savoir si le fournisseur est actif, True par défaut,
- ❖ date d'enregistrement, datetime avec timezone, défaut datetime de l'instance
- ❖ date d'inactivité, datetime avec timezone, non obligatoire,
- ❖ commentaire, texte sans limite de caractère, non obligatoire,
- ❖ employé, responsable de la création de la carte, obligatoire,
- ❖ pizzeria, obligatoire,



- ❖ une liste de pizzas.















• **N°16 Status**



Le statut constitue une table, car des énumérations limites l'ajouts de nouvelles instances. Ce modèle contient un champ

- ❖ nom court qui est unique et non null limité à 100 caractères et obligatoire,
- ❖ description qui peut être nulle et sans limite de caractère.











• **N°17 Order**

a public.pizzapp_order		
 id	int4	
 delay	timestamp(35) with time zone	
 is_active	bool	
 registration_date	timestamp(35) with time zone	
 inactivity_date	timestamp(35) with time zone	N
 comment	text	N
 customer_id	int4	
 employed_id	int4	N 
 pizzeria_id	int4	
 status_id	int4	

Le modèle commande représente les commandes des pizzas pour les clients. Le status de la commande sera mis à jour tout au long du processus de la commande, permettant ainsi de suivre en temps réel l'état des commandes. Ce modèle contient un champ

- ❖ délai de livraison, datetime avec timezone, obligatoire,
- ❖ is_active, boolean permettant de savoir si la commande est active, True par défaut,
- ❖ date d'enregistrement, datetime avec timezone, défaut datetime de l'instance
- ❖ date d'inactivité, datetime avec timezone, non obligatoire,
- ❖ commentaire, texte sans limite de caractère, non obligatoire,
- ❖ client, responsable de la création de la commande, obligatoire,
- ❖ employé, responsable de la création de la commande, non obligatoire,
- ❖ pizzeria, obligatoire,
- ❖ status de la commande, obligatoire.

• N°18 OrderLine

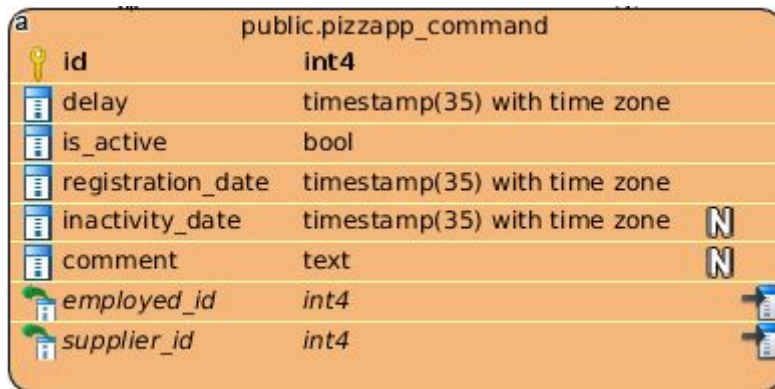
a public.pizzapp_orderline		
 id	int4	
 quantity	int4	
 is_active	bool	
 registration_date	timestamp(35) with time zone	
 inactivity_date	timestamp(35) with time zone	N
 comment	text	N
 order_id	int4	
 pizza_id	int4	

Les lignes de commandes permettent de gérer les pizzas commandées par les clients. Ce modèle contient un champ

- ❖ quantité de pizzas, integer, obligatoire,

- ❖ is_active, boolean permettant de savoir si le fournisseur est actif, True par défaut,
- ❖ date d'enregistrement, datetime avec timezone, défaut datetime de l'instance
- ❖ date d'inactivité, datetime avec timezone, non obligatoire,
- ❖ commentaire, texte sans limite de caractère, non obligatoire,
- ❖ commande, obligatoire,
- ❖ pizza, obligatoire.

• N°19 Command



public.pizzapp_command	
id	int4
delay	timestamp(35) with time zone
is_active	bool
registration_date	timestamp(35) with time zone
inactivity_date	timestamp(35) with time zone N
comment	text N
employed_id	int4
supplier_id	int4

La commande fournisseur, permet de gérer l'approvisionnement des composants. Ce modèle contient le champ

- ❖ délai de livraison, datetime avec timezone, obligatoire,
- ❖ is_active, boolean permettant de savoir si la commande est active, True par défaut,
- ❖ date d'enregistrement, datetime avec timezone, défaut datetime de l'instance
- ❖ date d'inactivité, datetime avec timezone, non obligatoire,
- ❖ commentaire, texte sans limite de caractère, non obligatoire,
- ❖ client, responsable de la création de la commande, obligatoire,
- ❖ employé, responsable de la création de la commande fournisseur, obligatoire,
- ❖ fournisseur, obligatoire.

• N°20 CommandLine

public.pizzapp_commandline		
id	int4	
quantity	numeric(8, 3)	N
delay	timestamp(35) with time zone	
is_active	bool	
registration_date	timestamp(35) with time zone	
inactivity_date	timestamp(35) with time zone	N
comment	text	N
command_id	int4	
component_id	int4	
unit_of_measure_id	int4	

La ligne de commande fournisseur, est le détails des lignes de commandes de composant, ce modèle sera utilisé directement pour les mouvements de stocks, il contient un champ

- ❖ quantité du composant, numérique de 8 chiffres dont 3 décimales, obligatoire,
- ❖ délai de livraison, datetime avec timezone, obligatoire,
- ❖ is_active, boolean permettant de savoir si la ligne de commande est active, True par défaut,
- ❖ date d'enregistrement, datetime avec timezone, défaut datetime de l'instance
- ❖ date d'inactivité, datetime avec timezone, non obligatoire,
- ❖ commentaire, texte sans limite de caractère, non obligatoire,
- ❖ commande, obligatoire,
- ❖ composant, obligatoire,
- ❖ unité de mesure, obligatoire.

• **N°21 StockMovement**

public.pizzapp_stockmovement		
id	int4	
quantity	numeric(8, 3)	
stock_before	numeric(8, 3)	
stock_after	numeric(8, 3)	
is_active	bool	
registration_date	timestamp(35) with time zone	
inactivity_date	timestamp(35) with time zone	N
comment	text	N
command_line_id	int4	N
component_id	int4	
order_line_id	int4	N
unit_of_measure_id	int4	

Le modèle mouvement de stock permet à chaque changement de statut des commandes (passage en “en attente de livraison”), de décrémenter les composants qui auront théoriquement servi à la fabrication de la commande. Le réajustement pourra se faire par une modification manuelle du stock. Ce modèle contient le champ

- ❖ quantité du composant, numérique de 8 chiffres dont 3 décimales, obligatoire,
- ❖ stock précédent du composant, numérique de 8 chiffres dont 3 décimales, défaut à zéro,
- ❖ stock après mouvement du composant, numérique de 8 chiffres dont 3 décimales, défaut à zéro,
- ❖ is_active, boolean permettant de savoir si le mouvement de stock est actif, True par défaut,
- ❖ date d'enregistrement, datetime avec timezone, défaut datetime de l'instance
- ❖ date d'inactivité, datetime avec timezone, non obligatoire,
- ❖ commentaire, texte sans limite de caractère, non obligatoire,
- ❖ ligne de commande fournisseur, non obligatoire,
- ❖ composant, obligatoire,
- ❖ ligne de commande client, non obligatoire,
- ❖ unité de mesure, obligatoire.

- **N°22 PaymentMethod**

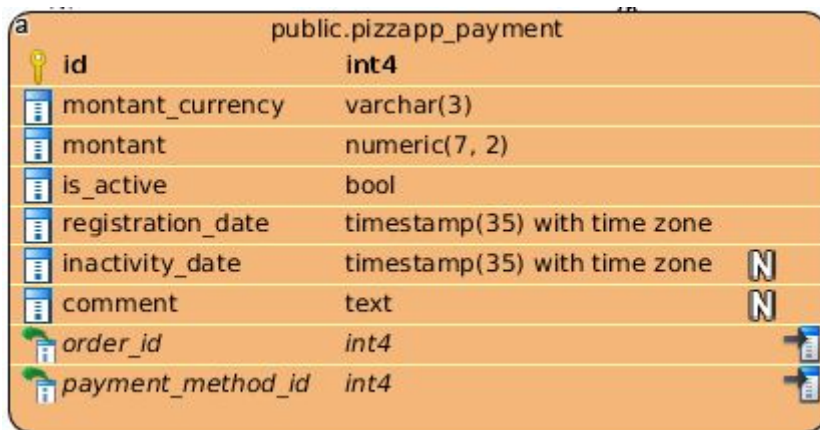


public.pizzapp_paymentmethod	
id	int4
short_name	varchar(20)
description	varchar(100)

Le modèle méthode de paiement existe afin de ne pas mettre d'énumération et se trouver limitée dans la création des instance. Ce modèle représente les différent moyens de paiements autorisés par les pizzérias. Ce modèle contient un champ

- ❖ nom court qui est unique et non null limité à 100 caractères et obligatoire,
- ❖ description qui peut être nulle et limité à 100 caractères.

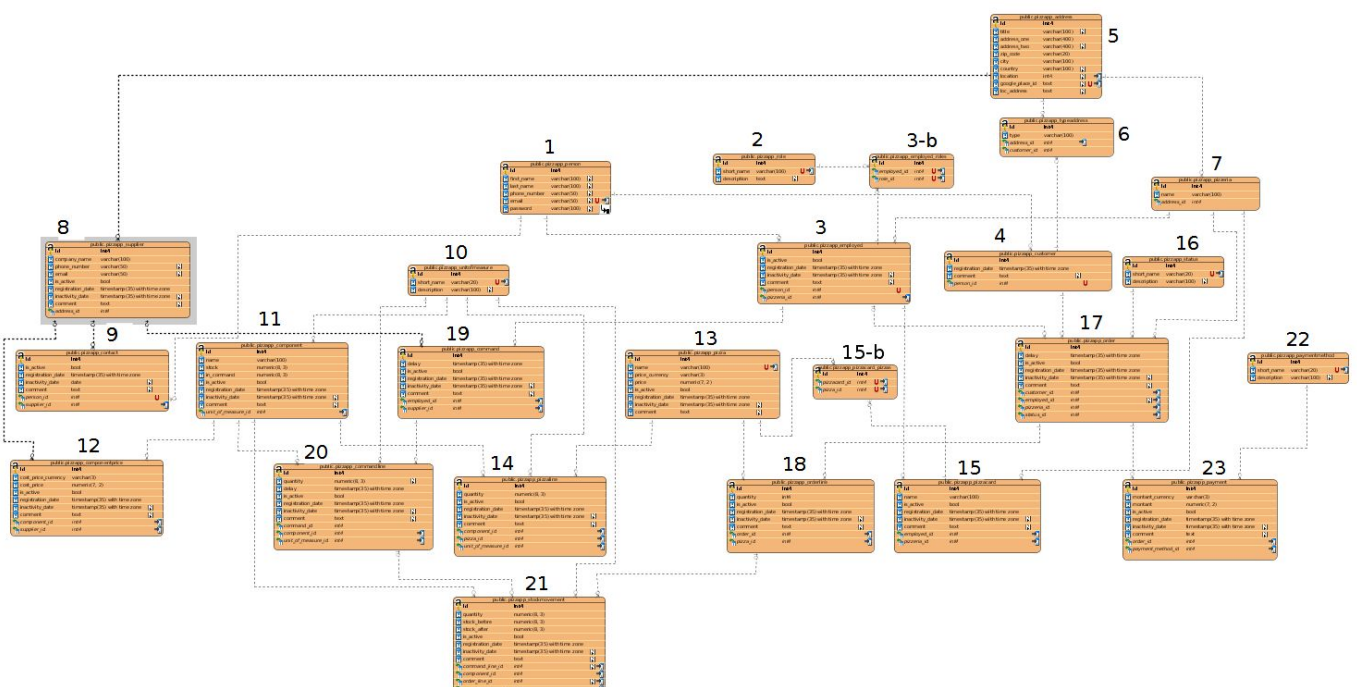
- **N°23 Payment**



Le modèle de données payment correspond à chaque moyen de paiement utilisé afin de régler une commande client. Ce modèle contient un champ

- ❖ currency du montant réglé, par défaut EUR, obligatoire
- ❖ montant réglé, numérique de 7 chiffres dont 3 décimales, obligatoire,
- ❖ is_active, boolean permettant de savoir si le paiement est actif, True par défaut,
- ❖ date d'enregistrement, datetime avec timezone, défaut datetime de l'instance
- ❖ date d'inactivité, datetime avec timezone, non obligatoire,
- ❖ commentaire, texte sans limite de caractère, non obligatoire,
- ❖ commande client, obligatoire,
- ❖ méthode de paiement, obligatoire.

Synthèse du diagramme de classe reprenant les spécifications fonctionnelles de l'application :



une version image est jointe aux livrable à des fins de lisibilités.