SmartCafe

Documentation Technique

v1.0.0

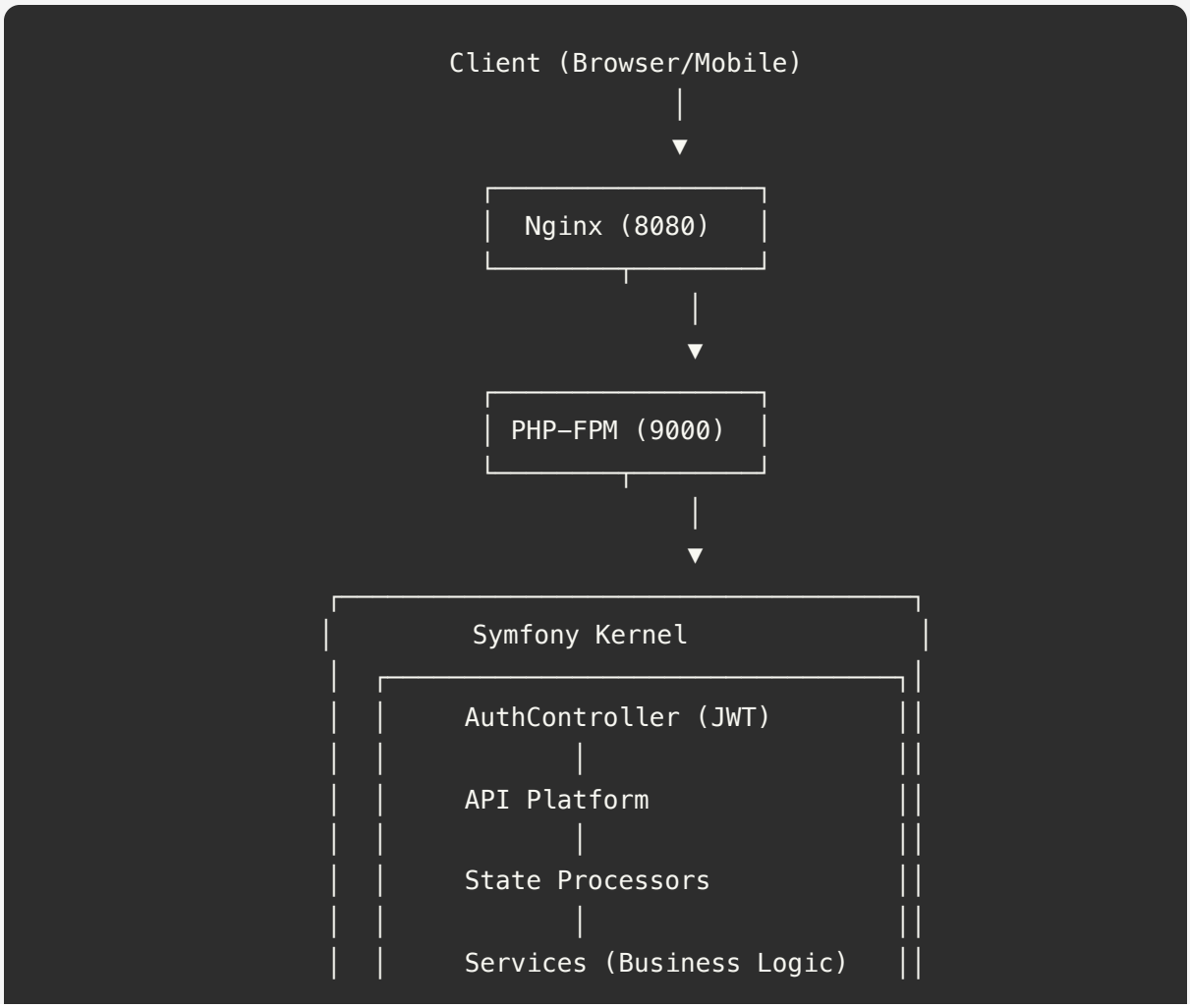
Architecture du Projet

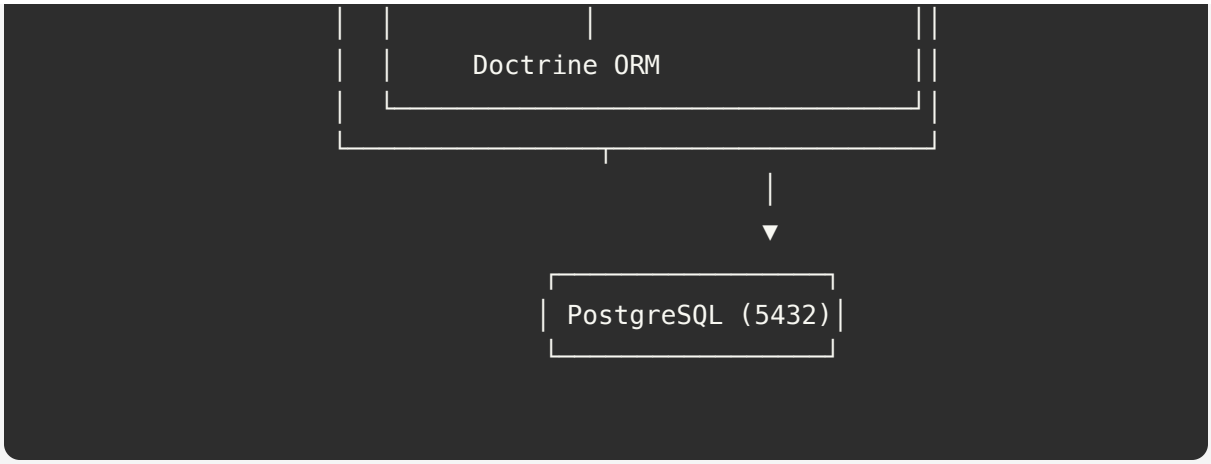
Structure des dossiers

```
SmartCafe/
├── api/                          # Backend API Symfony
│   ├── config/                   # Configuration Symfony
│   │   ├── packages/             # Bundles config (doctrine, sec
│   │   └── routes.yaml           # Routes personnalisees
│   ├── migrations/               # Migrations Doctrine
│   ├── src/
│   │   ├── Controller/           # Controllers (AuthController)
│   │   ├── DataFixtures/         # Donnees de test
│   │   ├── DTO/                  # Data Transfer Objects
│   │   ├── Entity/               # Entites Doctrine (11 entites)
│   │   ├── Enum/                 # Enumerations (OrderStatus, et
│   │   ├── EventSubscriber/      # Subscribers (JWT)
│   │   ├── Exception/            # Exceptions metier
│   │   ├── Repository/           # Repositories Doctrine
│   │   ├── Service/              # Services metier
```

```
│   │   │   ├── Auth/              # RefreshTokenService
│   │   │   ├── Loyalty/           # LoyaltyService
│   │   │   ├── Order/             # OrderService
│   │   │   ├── Stock/             # StockService
│   │   │   └── User/              # UserService
│   │   └── State/                 # Processors/Providers API Plat
│   └── tests/                     # Tests PHPUnit
├── docker/                        # Configuration Docker
│   ├── nginx/default.conf
│   └── php/Dockerfile
├── docs/                          # Documentation
├── docker-compose.yml             # Orchestration
├── Makefile                       # Commandes utiles
└── README.md
```

Architecture Globale





Patterns Utilises

Repository Pattern

Abstraction de l'acces aux donnees via Doctrine ORM

Service Layer Pattern

Logique metier encapsulee dans des services dedies

State Processor Pattern

API Platform processors pour les operations CRUD

DTO Pattern

Objets de transfert pour l'API

Technologies

Stack Principal

Composant	Technologie	Version
Framework	Symfony	8.0.*

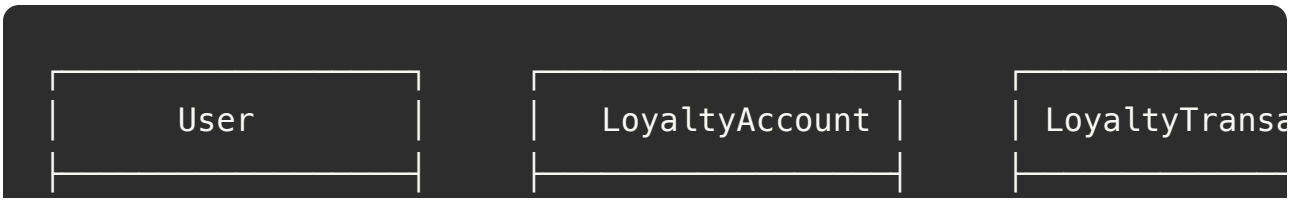
Langage	PHP	>= 8.2 (Docker: 8.4)
Base de donnees	PostgreSQL	15-alpine
API Framework	API Platform	4.2
ORM	Doctrine	3.6
Authentification	Lexik JWT Bundle	3.2
Serveur Web	Nginx	Alpine

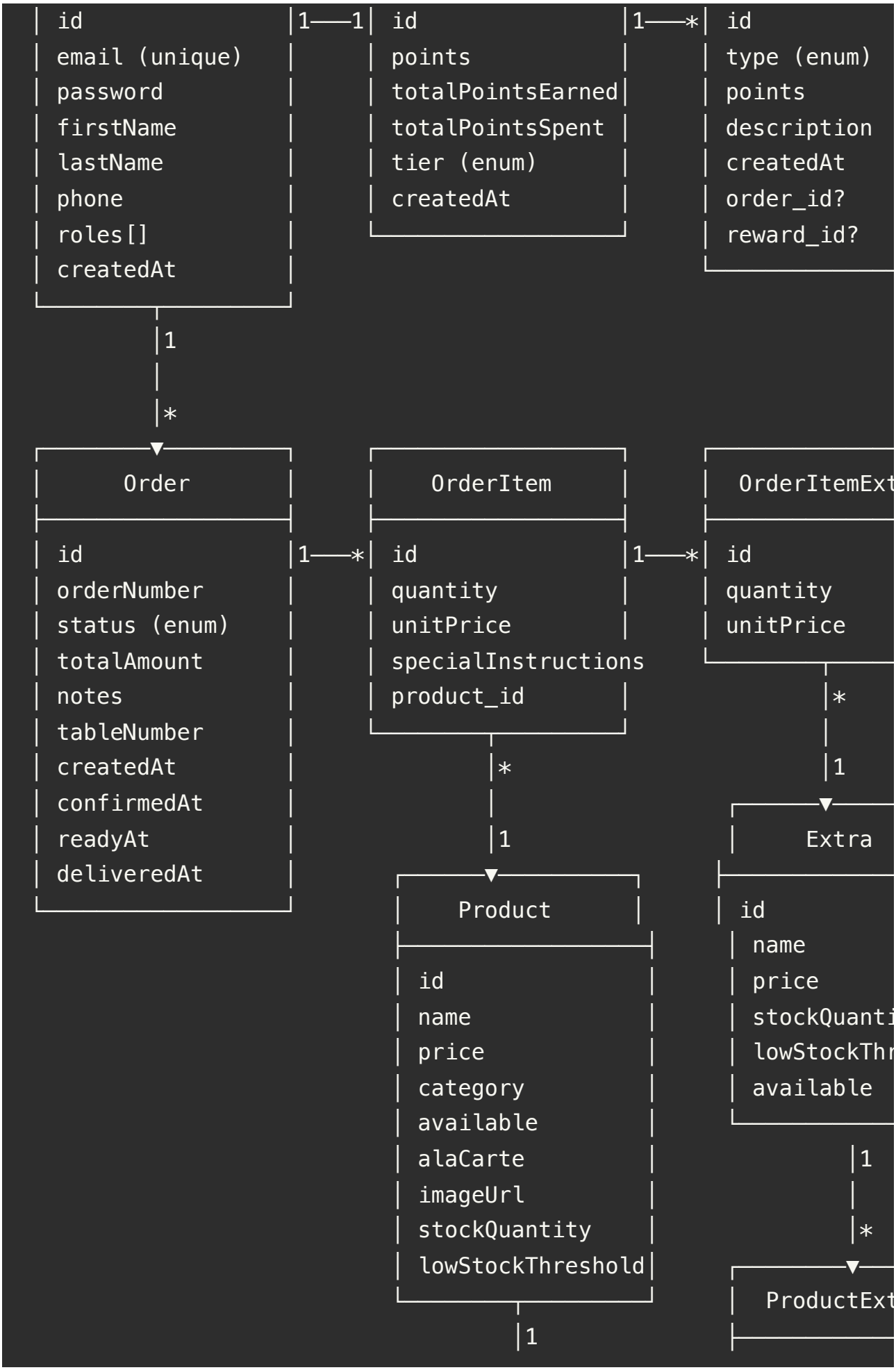
Outils de Qualite

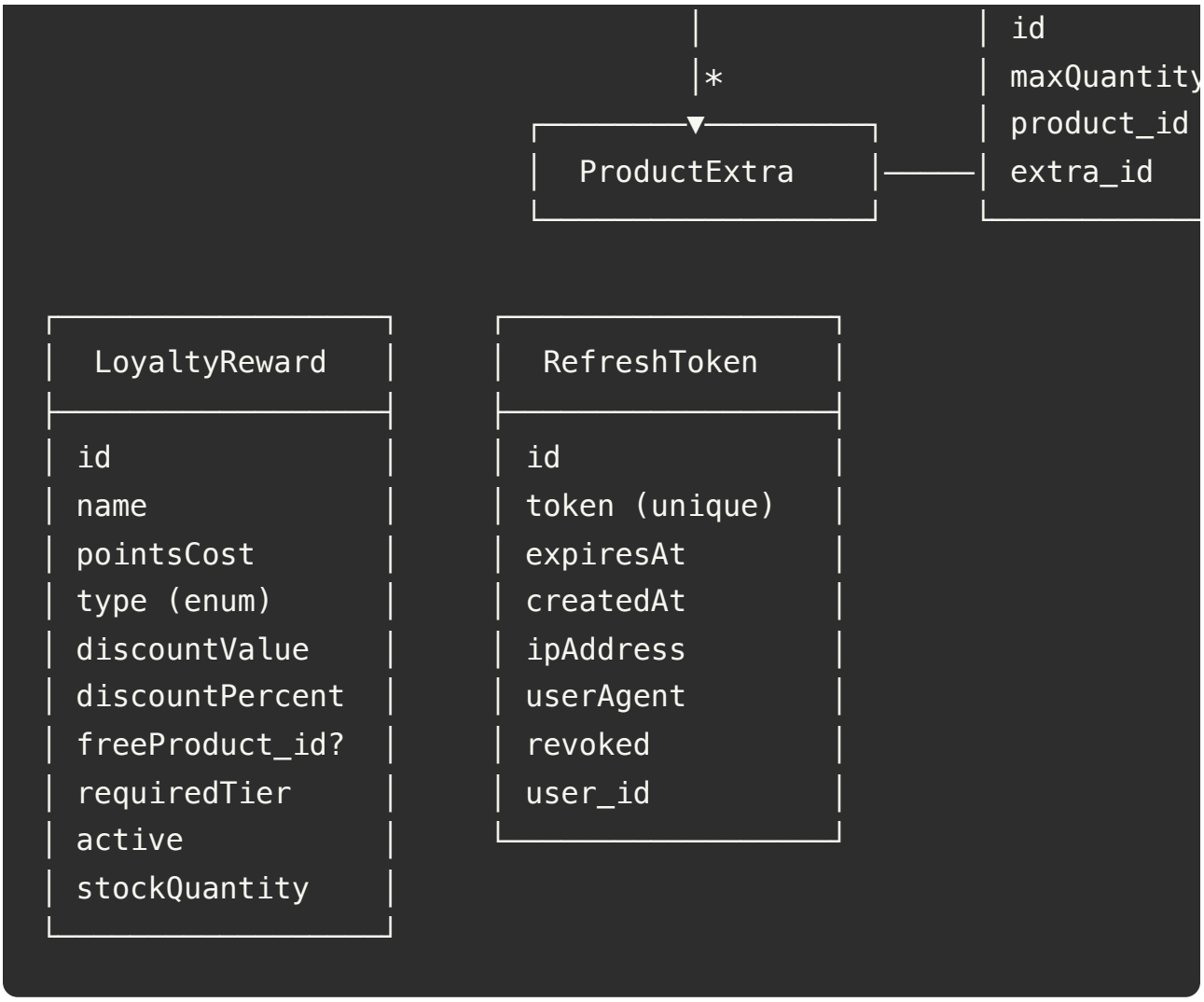
Outil	Usage
PHPUnit 11.0	Tests unitaires et fonctionnels
PHPStan 2.0	Analyse statique du code
PHP-CS-Fixer 3.64	Style de code
Doctrine Fixtures	Donnees de test
Zenstruck Foundry 2.0	Factories pour tests

Modele de Donnees

Diagramme des Entites




id
email (unique)
password
firstName
lastName
phone
roles[]
createdAt
1
1
id
points
totalPointsEarned
totalPointsSpent
tier (enum)
createdAt
1
*
id
type (enum)
points
description
createdAt
order_id?
reward_id?
1
*
Order
id
orderNumber
status (enum)
totalAmount
notes
tableNumber
createdAt
confirmedAt
readyAt
deliveredAt
1
*
OrderItem
id
quantity
unitPrice
specialInstructions
product_id
1
*
OrderItemExt
id
quantity
unitPrice
*
1
*
1
Product
id
name
price
category
available
alaCarte
imageUrl
stockQuantity
lowStockThreshold
1
Extra
id
name
price
stockQuanti
lowStockThr
available
1
*
ProductExt



Detail des Entites

User

Champ	Type	Contraintes
id	int	Auto-generated
email	string(180)	Unique, NotBlank, Email
password	string	NotBlank, Min 6 chars
firstName	string(100)	NotBlank
lastName	string(100)	NotBlank

phone	string(20)	Nullable
roles	array	Default: ROLE_USER
createdAt	datetime	Auto-set

Product

Champ	Type	Contraintes
id	int	Auto-generated
name	string(255)	NotBlank
price	decimal(10,2)	Positive
category	string(100)	NotBlank
available	boolean	Default: true
alaCarte	boolean	Default: true
imageUrl	string	Nullable, URL
stockQuantity	int	Nullable, PositiveOrZero
lowStockThreshold	int	Default: 10

Order

Champ	Type	Contraintes
id	int	Auto-generated
orderNumber	string	Unique, Auto-generated
status	OrderStatus	Enum

totalAmount	decimal(10,2)	Calculated
notes	text	Nullable
tableNumber	string(20)	Nullable
createdAt	datetime	Auto-set
confirmedAt	datetime	Nullable
readyAt	datetime	Nullable
deliveredAt	datetime	Nullable

LoyaltyAccount

Champ	Type	Contraintes
id	int	Auto-generated
points	int	Default: 0
totalPointsEarned	int	Default: 0
totalPointsSpent	int	Default: 0
tier	string	bronze/silver/gold/platinum
createdAt	datetime	Auto-set

Enumerations

OrderStatus

```
enum OrderStatus: string {
    case PENDING = 'pending';       // Commande creee, en attente
    case CONFIRMED = 'confirmed';   // Confirmee, stock deduit
```

```
    case PREPARING = 'preparing';   // En preparation
    case READY = 'ready';           // Prete a servir
    case DELIVERED = 'delivered';   // Livree, points attribues
    case CANCELLED = 'cancelled';   // Annulee, stock restaure
}
```

Transitions de statut autorisees:

- PENDING → CONFIRMED, CANCELLED
- CONFIRMED → PREPARING, CANCELLED
- PREPARING → READY, CANCELLED
- READY → DELIVERED, CANCELLED
- DELIVERED → (final)
- CANCELLED → (final)

LoyaltyTransactionType

```
enum LoyaltyTransactionType: string {
    case EARN = 'earn';           // Points gagnes sur commande
    case REDEEM = 'redeem';       // Points utilises pour recompe
    case BONUS = 'bonus';         // Points bonus
    case EXPIRED = 'expired';     // Points expires
    case ADJUSTMENT = 'adjustment'; // Ajustement manuel
}
```

RewardType

```
enum RewardType: string {
    case FREE_PRODUCT = 'free_product';       // Produit offert
    case DISCOUNT_AMOUNT = 'discount_amount'; // Reduction en eur
    case DISCOUNT_PERCENT = 'discount_percent'; // Reduction en %
    case FREE_EXTRA = 'free_extra';           // Extra offert
```

```
    case DOUBLE_POINTS = 'double_points';     // Points doubles
}
```

Regles de Gestion Metier

Gestion des Commandes

RG-CMD-001: Generation du numero de commande

Format: `ORD-YYYYMMDD-XXXXXXXX` (8 caracteres aleatoires)
Exemple: `ORD-20260121-a3f8b2c1`

RG-CMD-002: Calcul du total

Le total est calcule automatiquement:

```
totalAmount = SUM(item.quantity * item.unitPrice)
            + SUM(itemExtra.quantity * itemExtra.unitPrice)
```

RG-CMD-003: Verification du stock

Avant creation d'une commande, verifier que:

- Chaque produit a un stock suffisant (si gere)
- Chaque extra a un stock suffisant (si gere)

Si stock insuffisant: erreur 400 avec detail

RG-CMD-004: Deduction du stock

Le stock est deduit uniquement lors du passage au statut `CONFIRMED`

RG-CMD-005: Restauration du stock

Le stock est restaure lors du passage au statut `CANCELLED`

RG-CMD-006: Attribution des points

Les points de fidelite sont attribues uniquement lors du passage au statut `DELIVERED`

Systeme de Fidelite

RG-FID-001: Points par euro

Base: **10 points par euro depense**

RG-FID-002: Multiplicateurs par tier

Tier	Multiplicateur	Points requis
Bronze	1.0x	0 - 499
Silver	1.25x	500 - 1999
Gold	1.5x	2000 - 4999
Platinum	2.0x	5000+

RG-FID-003: Calcul des points

```
pointsGagnes = floor(totalAmount * POINTS_PAR_EURO * multipl
```

Exemple: Commande 15.50EUR, tier Silver

```
points = floor(15.50 * 10 * 1.25) = floor(193.75) = 193 poin
```

RG-FID-004: Mise a jour du tier

Le tier est recalcule apres chaque transaction basee sur `totalPointsEarned`

RG-FID-005: Echange de recompense

- L'utilisateur doit avoir suffisamment de points
- L'utilisateur doit avoir le tier requis (si applicable)
- La recompense doit etre active
- La recompense doit etre en stock (si applicable)

Gestion du Stock

RG-STK-001: Stock optionnel

Si `stockQuantity = null`, le produit/extra est considere comme toujours disponible

RG-STK-002: Seuil d'alerte

Un produit/extra est considere en stock faible si:

```
stockQuantity <= lowStockThreshold
```

RG-STK-003: Disponibilite automatique

Un produit/extra devient automatiquement indisponible si `stockQuantity = 0`

Gestion des Extras

RG-EXT-001: Association produit-extra

Un extra ne peut etre ajoute a un produit que s'il existe une association `ProductExtra`

RG-EXT-002: Quantite maximale

La quantite d'un extra sur un item de commande ne peut pas depasser le `maxQuantity` defini dans `ProductExtra`

Authentification

RG-AUTH-001: Creation de compte

A la creation d'un utilisateur:

- Un compte de fidelite est automatiquement cree
- Le role par defaut est `ROLE_USER`
- Le mot de passe doit faire minimum 6 caracteres

RG-AUTH-002: Refresh Token

- Duree de validite: 30 jours
- Stocke en cookie HttpOnly
- Tracking IP et User-Agent

- Possibilite de revoquer un ou tous les tokens

Routes API

Production: `http://152.228.131.67/api`
Local: `http://localhost:8080/api`
Swagger UI (local): `http://localhost:8080/api/docs`

Authentification

Methode	Route	Description	Acces
POST	`/api/login`	Connexion (retourne JWT + cookie refresh)	Public
POST	`/api/token/refresh`	Rafraichir le JWT	Public
POST	`/api/token/revoke`	Revoquer le token actuel	Auth
POST	`/api/token/revoke-all`	Revoquer tous les tokens	Auth
GET	`/api/auth/sessions`	Lister les sessions actives	Auth
GET	`/api/auth/me`	Profil utilisateur connecte	Auth

Utilisateurs

Methode	Route	Description	Acces

GET	`/api/users`	Liste des utilisateurs	Admin
POST	`/api/users`	Creer un utilisateur	Public
GET	`/api/users/{id}`	Detail utilisateur	Self/Admin
PATCH	`/api/users/{id}`	Modifier utilisateur	Self/Admin
DELETE	`/api/users/{id}`	Supprimer utilisateur	Admin

Produits

Methode	Route	Description	Acces
GET	`/api/products`	Liste des produits	Public
GET	`/api/products/{id}`	Detail produit	Public
GET	`/api/products/low-stock`	Produits en stock faible	Admin
POST	`/api/products`	Creer produit	Admin
PATCH	`/api/products/{id}`	Modifier produit	Admin
DELETE	`/api/products/{id}`	Supprimer produit	Admin

Extras

Methode	Route	Description	Acces
GET	`/api/extras`	Liste des extras	Public
GET	`/api/extras/low-stock`	Extras en stock faible	Admin

POST	`/api/extras/{id}/restock`	Restocker un extra	Admin

Commandes

Methode	Route	Description	Acces
GET	`/api/orders`	Liste des commandes	Admin
GET	`/api/auth/me/orders`	Mes commandes	Auth
POST	`/api/orders`	Creer commande	Auth
GET	`/api/orders/{id}`	Detail commande	Owner/Admin
PATCH	`/api/orders/{id}`	Modifier statut	Admin

Fidelite

Methode	Route	Description	Acces
GET	`/api/loyalty/rewards`	Liste des recompenses	Public
POST	`/api/loyalty/rewards/{id}/redeem`	Echanger une recompense	Auth
GET	`/api/auth/me/loyalty`	Mon compte fidelite	Auth
GET	`/api/auth/me/loyalty/transactions`	Historique transactions	Auth

Services Metier

OrderService

Fichier: `api/src/Service/Order/OrderService.php`

Methode	Description
`createOrder(User, CreateOrderDTO)`	Cree une commande avec validation du stock
`updateOrderStatus(Order, OrderStatus)`	Change le statut avec gestion stock/points
`calculateOrderTotal(Order)`	Recalcule le montant total
`generateOrderNumber()`	Genere un numero unique

LoyaltyService

Fichier: `api/src/Service/Loyalty/LoyaltyService.php`

Methode	Description
`awardPoints(LoyaltyAccount, Order)`	Attribue les points pour une commande
`redeemReward(LoyaltyAccount, LoyaltyReward)`	Echange des points contre une recompense
`calculatePoints(float, string)`	Calcule les points avec multiplicateur tier
`updateTier(LoyaltyAccount)`	Met a jour le tier selon les points

StockService

Fichier: `api/src/Service/Stock/StockService.php`

Methode	Description
`checkStock(Product/Extra, int)`	Verifie la disponibilite du stock
`deductStock(Product/Extra, int)`	Deduit le stock
`restoreStock(Product/Extra, int)`	Restaure le stock
`getLowStockProducts()`	Retourne les produits en stock faible

RefreshTokenService

Fichier: `api/src/Service/Auth/RefreshTokenService.php`

Methode	Description
`createRefreshToken(User, Request)`	Cree un token avec tracking
`refreshToken(string)`	Valide et renouvelle un token
`revokeToken(string)`	Revoque un token specifique
`revokeAllUserTokens(User)`	Revoque tous les tokens d'un user

State Processors (API Platform)

Processor	Operations	Description
`OrderStateProcessor`	POST, PATCH	Creation et mise a jour des commandes
`UserStateProcessor`	POST, PATCH	Creation utilisateur + compte fidelite

`ExtraStateProcessor`	POST (restock)	Restockage des extras
`RedeemRewardProcessor`	POST	Echange de recompenses

State Providers (API Platform)

Provider	Route	Description
`MyOrdersProvider`	/api/auth/me/orders	Commandes de l'utilisateur connecte
`MyLoyaltyProvider`	/api/auth/me/loyalty	Compte fidelite de l'utilisateur
`MeProvider`	/api/auth/me	Profil de l'utilisateur connecte

Securite

Configuration JWT

```
# config/packages/lexik_jwt_authentication.yaml
lexik_jwt_authentication:
    secret_key: '%env(resolve:JWT_SECRET_KEY)%'
    public_key: '%env(resolve:JWT_PUBLIC_KEY)%'
    pass_phrase: '%env(JWT_PASSPHRASE)%'
    token_ttl: 3600  # 1 heure
```

Firewall

```
# config/packages/security.yaml
security:
```

```
    firewalls:
        login:
            pattern: ^/api/login
            stateless: true
            json_login:
                check_path: /api/login
                success_handler: lexik_jwt_authentication.handler
                failure_handler: lexik_jwt_authentication.handler
        api:
            pattern: ^/api
            stateless: true
            jwt: ~
```

Controle d'acces

```
access_control:
    - { path: ^/api/docs, roles: PUBLIC_ACCESS }
    - { path: ^/api/login, roles: PUBLIC_ACCESS }
    - { path: ^/api/token/refresh, roles: PUBLIC_ACCESS }
    - { path: ^/api/users, methods: [POST], roles: PUBLIC_ACCESS
    - { path: ^/api/products, methods: [GET], roles: PUBLIC_ACCES
    - { path: ^/api/extras, methods: [GET], roles: PUBLIC_ACCESS
    - { path: ^/api/loyalty/rewards, methods: [GET], roles: PUBLI
    - { path: ^/api, roles: IS_AUTHENTICATED_FULLY }
```

Roles

Role	Description	Permissions
`ROLE_USER`	Utilisateur standard	Creer commandes, voir son profil, gerer sa fidelite
`ROLE_ADMIN`	Administrateur	Toutes les operations, gestion produits/commandes/users

Refresh Token (Cookie)

Configuration du cookie:

- **HttpOnly:** true (non accessible via JavaScript)
- **Secure:** true en production (HTTPS uniquement)
- **SameSite:** Lax
- **Duree:** 30 jours
- **Path:** /api/token

Tests

Structure

```
tests/
├── Unit/
│   ├── Entity/           # Tests des entites
│   │   ├── UserTest.php
│   │   ├── OrderTest.php
│   │   └── ...
│   └── Service/          # Tests des services
│       ├── OrderServiceTest.php
│       ├── LoyaltyServiceTest.php
│       └── ...
└── Functional/
    ├── Api/              # Tests d'endpoints
    │   ├── UserApiTest.php
    │   ├── ProductApiTest.php
    │   ├── OrderApiTest.php
    │   └── ...
    └── ApiTestCase.php   # Classe de base
```

Commandes

```
# Tous les tests
make test

# Tests unitaires uniquement
make test:unit

# Tests fonctionnels uniquement
make test:functional

# Avec couverture
./vendor/bin/phpunit --coverage-html coverage/
```

Exemple de test fonctionnel

```
class OrderApiTest extends ApiTestCase
{
    public function testCreateOrder(): void
    {
        $client = static::createClient();
        $this->loginAs($client, 'john.doe@example.com');

        $response = $client->request('POST', '/api/orders', [
            'json' => [
                'items' => [
                    ['productId' => 1, 'quantity' => 2]
                ],
                'tableNumber' => 'A1'
            ]
        ]);

        $this->assertResponseStatusCodeSame(201);
        $this->assertJsonContains(['status' => 'pending']);
    }
}
```

Qualite du code

```
# Analyse statique (PHPStan niveau 8)
make phpstan

# Verification style de code
make cs:check

# Correction automatique du style
make cs:fix

# Tous les checks
make qa
```

Commandes Makefile

Docker

Commande	Description
`make build`	Build les images Docker
`make up`	Demarre les containers
`make down`	Arrete les containers
`make restart`	Redemarre les containers
`make logs`	Affiche les logs
`make shell`	Ouvre un shell dans le container PHP

Base de donnees

Commande	Description
`make db-create`	Cree la base de donnees
`make db-migrate`	Execute les migrations
`make db-diff`	Genere une nouvelle migration
`make db-seed`	Charge les fixtures

Developpement

Commande	Description
`make init`	Initialisation complete du projet
`make composer-install`	Installe les dependances
`make jwt-generate`	Genere les cles JWT
`make cache-clear`	Vide le cache Symfony

Tests et Qualite

Commande	Description
`make test`	Lance tous les tests
`make test:unit`	Tests unitaires
`make test:functional`	Tests fonctionnels
`make phpstan`	Analyse statique

`make cs:check`	Verifie le style de code
`make cs:fix`	Corrige le style de code
`make qa`	Tous les checks qualite

Donnees de Test (Fixtures)

Utilisateurs

Email	Mot de passe	Role
admin@smartcafe.fr	admin123	ROLE_ADMIN
john.doe@example.com	password123	ROLE_USER
jane.doe@example.com	password123	ROLE_USER

Extras disponibles

- Chantilly
- Sirop Caramel
- Sirop Vanille
- Sirop Noisette
- Lait d'Avoine
- Lait d'Amande
- Lait de Soja
- Shot Espresso
- Copeaux Chocolat
- Cannelle

Charger les fixtures

```
make db-seed
```

SmartCafe - Documentation Technique v1.0.0
Generee le 22 janvier 2026