OCPIZZA

Système de gestion de pizzerias

Dossier de conception technique

Version 1.0

Auteur
Rudy Lepretre
Analyste programmeur

EVOLUX
Evolux.com
27 Rue du Faubourg Saint-Antoine, 75011 Paris – +33 1 48 04 67 56 – evolux@gmail.com
S.A.R.L. au capital de 1 000,00 € enregistrée au RCS de Xxxx – SIREN 999 999 999 – Code APE : 6202A

OC PIZZA

EVOLUX

TABLE DES MATIERES

1 - VERSIONS

Auteur	Date	Description	Version
R.Lepretre	25/02/2023	Création du document	1.0.0

2 - INTRODUCTION

2.1 - Objet du document

Les éléments du présent dossier découlent :

- de l'analyse fonctionnelle détaillée de l'application OC Pizza ;
- des choix d'implémentation retenus ;
- des contraintes techniques et fonctionnelles à respecter ;
- des normes et standards en vigueur dans l'entreprise ;
- des recommandations et bonnes pratiques en matière de développement logiciel.

Ce dossier s'adresse principalement aux développeurs qui seront chargés de mettre en place l'application OC Pizza, mais il pourra également être utile à toute personne impliquée dans le projet, telle que les chefs de projet ou les architectes logiciels.

2.2 - Références

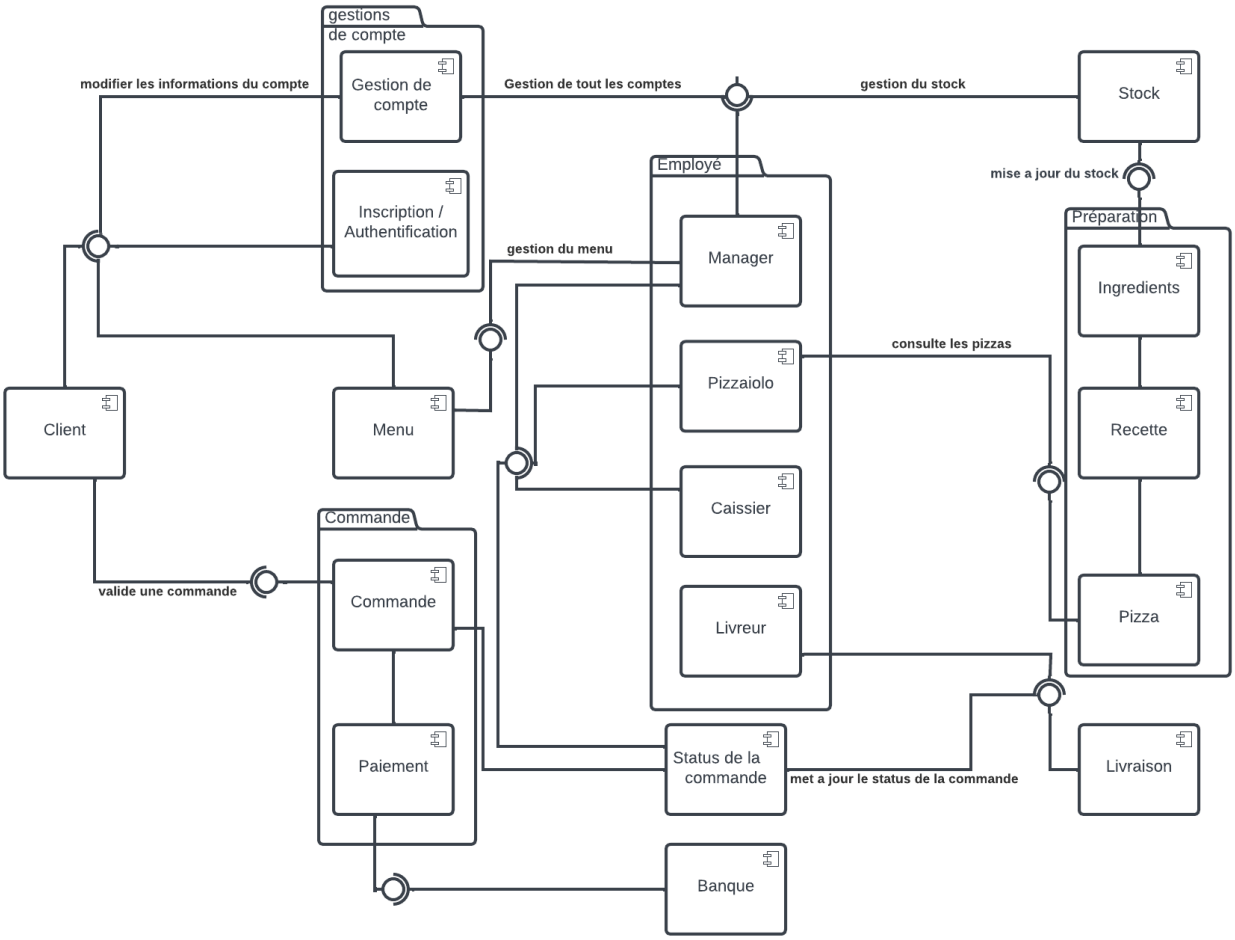
Pour de plus amples informations, se référer également aux éléments suivants :

1. **DCF - 1.0.0** : Dossier de conception fonctionnelle de l'application
2. **DE – 1.0.0** : Dossier d'exploitation de l'application
3. **PVL – 1.0.0** : Procès-verbal de livraison de l'application

3 - Architecture Technique

3.1 - Composants généraux

3.1.1 - Diagramme de composant



3.1.2 - Commande

3.1.2.1 - Commande

Ce composant gère le processus de commande des clients de la pizzeria, depuis la sélection des articles jusqu'à la confirmation et la validation. Il permet aux clients de passer des commandes en ligne ou via l'application mobile, et permet également aux employés de saisir des commandes par téléphone ou en personne.

3.1.2.2 - Paiement

Ce composant gère les paiements associés aux commandes, y compris les paiements en ligne, les paiements en personne et les remboursements. Il s'assure que les paiements sont correctement traités et que les transactions sont sécurisées.

3.1.3 - Gestion de compte

3.1.3.1 - Gestion de compte

Ce composant permet aux clients de créer et de gérer leur compte, y compris la gestion des informations personnelles, l'historique des commandes et les préférences de notification. Il fournit également aux employés de la pizzeria des outils pour gérer les comptes clients.

3.1.3.2 - Inscription / Authentification

Ce sous-composant gère le processus d'inscription et d'authentification des clients. Il permet aux clients de créer un compte, de se connecter en toute sécurité et de récupérer leur mot de passe en cas de besoin.

3.1.4 - Préparation

3.1.4.1 - Ingrédients

Ce sous-composant gère la gestion des ingrédients utilisés dans la fabrication des pizzas. Il s'assure que les ingrédients sont stockés correctement et en quantité suffisante pour répondre aux demandes de commande.

3.1.4.2 - Recette

Ce sous-composant gère la création et la modification des recettes de pizza. Il assure la mise à jour des listes d'ingrédients nécessaires à la préparation des pizzas et s'assure que les

recettes sont suivies de manière cohérente pour garantir la qualité des pizzas.

3.1.4.3 - Pizza

Ce sous-composant gère la fabrication et la cuisson des pizzas en fonction des commandes passées. Il assure la cohérence de la qualité et du goût des pizzas à chaque commande.

3.1.5 - Employé

3.1.5.1 - Manager

Ce sous-composant gère les responsabilités du manager de la pizzeria, y compris la supervision des opérations quotidiennes, la gestion des employés et la résolution des problèmes.

3.1.5.2 - Pizzaiolo

Ce sous-composant gère les responsabilités du pizzaiolo, y compris la préparation des pizzas en fonction des recettes et des commandes, la gestion de l'inventaire et le nettoyage de la cuisine.

3.1.5.3 - Livreur

Ce sous-composant gère les responsabilités du livreur, y compris la récupération des commandes et la livraison aux clients. Il s'assure que les commandes sont livrées en temps voulu et en bon état, et il peut également être chargé de collecter les paiements en personne.

3.1.5.4 - Caissier

Ce sous-composant gère les responsabilités du caissier, y compris la gestion des paiements en personne, la tenue de la caisse enregistreuse et l'interaction avec les clients.

3.1.6 - Autres composants

3.1.6.1 - Banque

Ce sous-composant gère la liaison avec la banque pour les transactions de paiement en ligne et les remboursements. Il s'assure que les paiements sont correctement traités et que les transactions sont sécurisées.

3.1.6.2 - Stock

Ce sous-composant gère la gestion des stocks de la pizzeria, y compris la gestion des commandes et des livraisons d'ingrédients, ainsi que la gestion des stocks de fournitures et d'équipements.

3.1.6.3 - Menu

Ce sous-composant gère le menu de la pizzeria, y compris la liste des pizzas disponibles, les ingrédients utilisés dans chaque pizza, les prix et les descriptions.

3.1.6.4 - Client

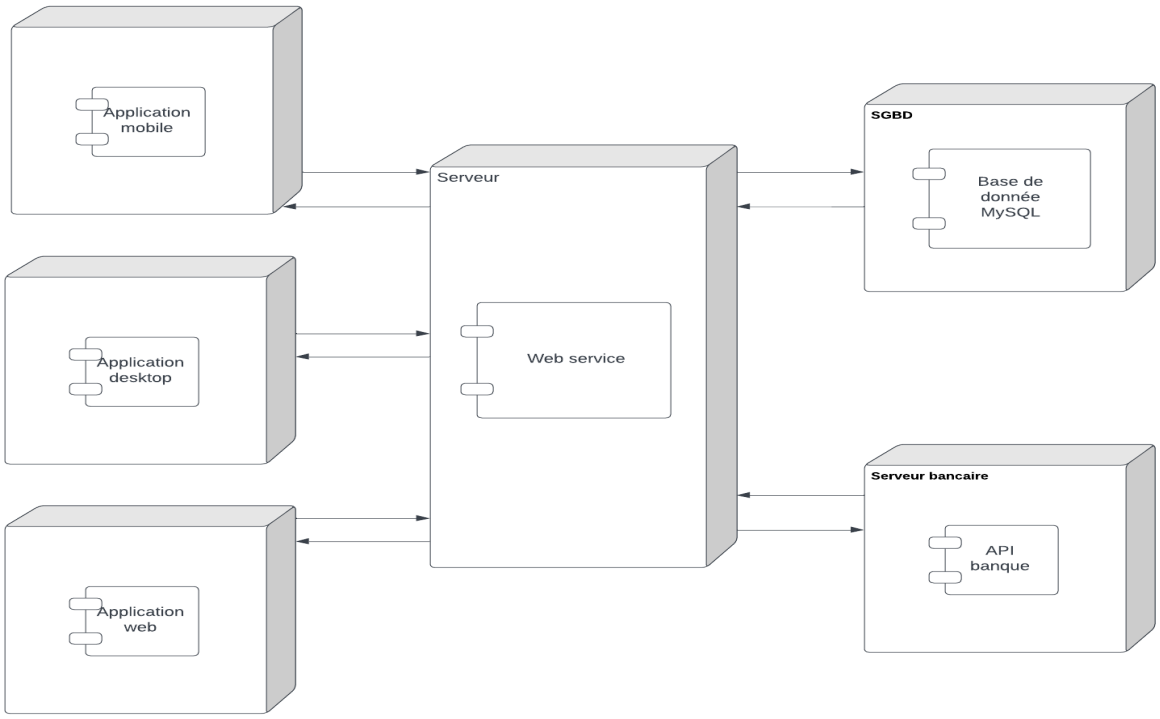
Ce sous-composant gère les interactions avec les clients, y compris la collecte d'informations personnelles, la communication des promotions et des offres spéciales, et la gestion des commentaires et des plaintes.

3.1.6.5 - Livraison

Ce sous-composant gère le processus de livraison des pizzas aux clients, y compris la planification des itinéraires de livraison, la communication avec les clients et la gestion des problèmes liés à la livraison

4 - Architecture de Deploiement

Le diagramme de déploiement ci-dessous représente la topologie matérielle du système. Il montre comment les composants logiciels sont déployés sur les différents matériels physiques et comment ils communiquent entre eux.

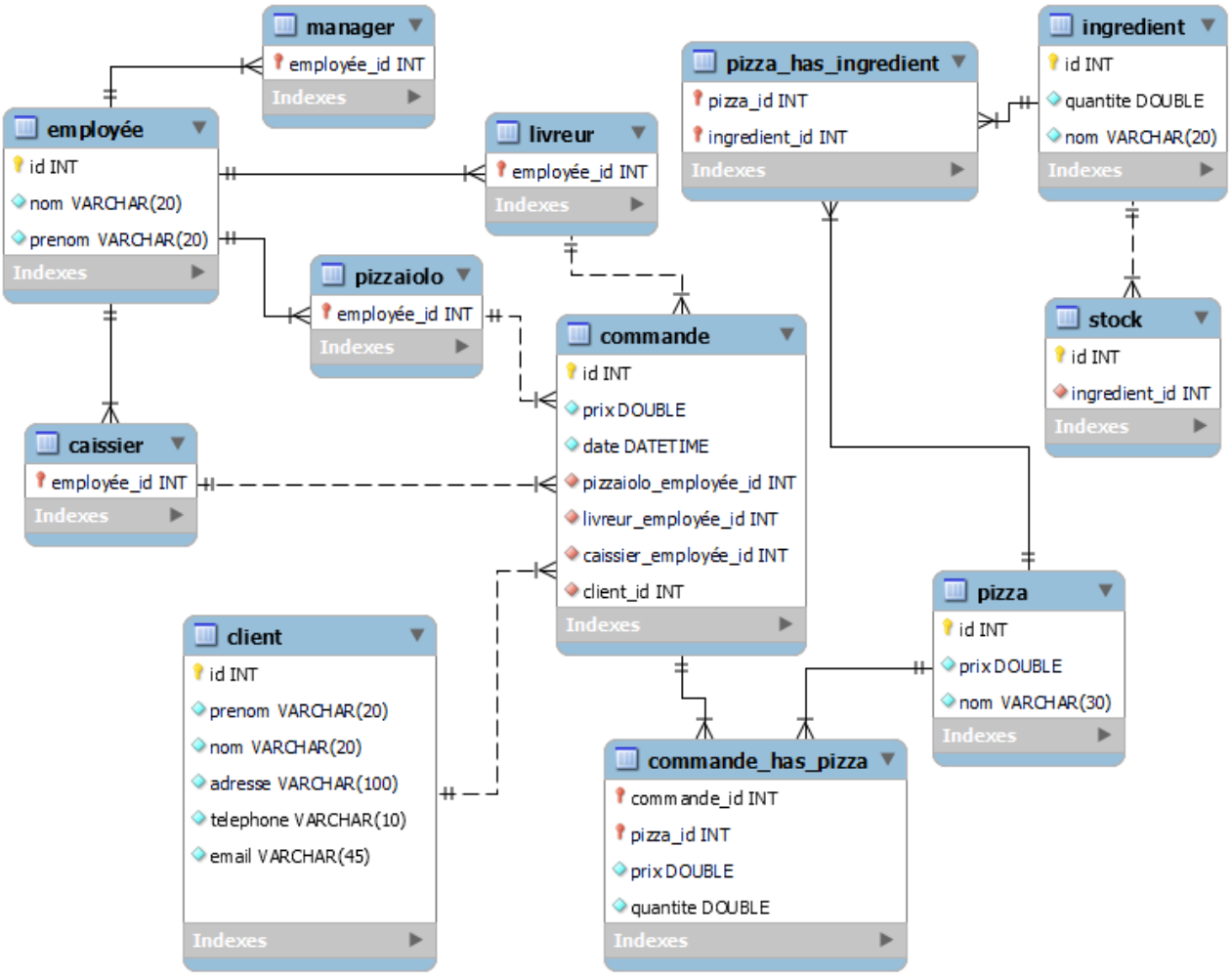


4.1 - Serveur de Base de données

Le serveur de base de données sera un serveur physique hébergé dans les locaux techniques du siège de OC Pizza. Le système de base données relationnelle sera un serveur MySQL.

4.1.1 - Modèle physique de données

4.1.1.1 - Diagramme



4.1.1.2 - Définition

Le modèle physique de données représente la structure physique de la base de données. Il se base sur le diagramme de classes et permet de définir les clés primaires et secondaires pour chaque table.

- Dans la table "Client", l'identifiant (id) est la clé primaire de type long. Elle est unique pour chaque client et sert à identifier de manière unique chaque enregistrement dans la table. Pour les autres champs tels que "prénom", "nom", "adresse", "téléphone" et "email", j'ai défini des types de données appropriés pour chaque champ, par exemple "VARCHAR(20)" pour le prénom et le nom qui limite les caractères à 20.
- Dans la table "Commande", l'identifiant (id) est également la clé primaire de type long. La table "Commande" est liée à la table "Pizza" par une relation many-to-many, car une commande peut contenir plusieurs pizzas et une pizza peut être commandée plusieurs fois. J'ai créé une table de jointure "Commande_has_Pizza" qui contient deux clés étrangères : l'identifiant de la commande (id_commande) et l'identifiant de la pizza (id_pizza). Ces clés font référence aux clés primaires correspondantes dans les tables "Commande" et "Pizza". La présence des champs "prix" et "quantité" dans la table de jointure "Commande_has_Pizza" est due au fait qu'une commande peut inclure plusieurs pizzas différentes, et chaque pizza peut avoir un prix et une quantité différents. Ainsi, en stockant ces informations dans la table de jointure, on peut enregistrer le prix et la quantité pour chaque pizza commandée dans une commande donnée, plutôt que d'avoir à les stocker dans la table "Pizza" ou dans la table "Commande". Cela permet de représenter plus précisément les informations relatives à chaque commande et de faciliter la gestion des transactions financières liées aux commandes.
- Pour la table "Pizza", l'identifiant (id) est la clé primaire de type long. La table "Pizza" est liée à la table "Ingrédient" par une relation many-to-many, car une pizza peut contenir plusieurs ingrédients et un ingrédient peut être utilisé dans plusieurs pizzas. J'ai créé une table de jointure "Pizza_has_Ingredient" qui contient deux clés étrangères : l'identifiant de la pizza (id_pizza) et l'identifiant de l'ingrédient (id_ingredient). Ces clés font référence aux clés primaires correspondantes dans les tables "Pizza" et "Ingrédient".

Cette modélisation assure l'intégrité des données et permet de gérer efficacement les relations many-to-many entre les tables de la base de données.

5 - ARCHITECTURE LOGICIELLE

5.1 - Principes généraux

La gestion du projet de développement de l'application Web et Android sera effectuée à l'aide de la méthode Agile Scrum. Les sources et les versions du projet seront gérées par Git pour assurer une collaboration efficace entre les membres de l'équipe et une traçabilité de l'historique des modifications. Les dépendances et le packaging des différentes composantes seront gérés par Apache Maven pour l'application Web et Gradle pour l'application Android. La mise en place d'une intégration continue avec Jenkins permettra de garantir une qualité de code optimale ainsi qu'une gestion efficace des tests et de la livraison. L'utilisation de JIRA pour le suivi des tâches et des anomalies facilitera la communication et le suivi de l'avancement du projet. Enfin, une documentation complète du projet sera maintenue à jour dans Confluence pour une meilleure gestion des connaissances et une facilitation de la maintenance.

5.1.1 - Les couches

L'architecture applicative est la suivante :

- Une couche **Service** : responsable de la logique métier du composant
- Une couche **Model** : implémentation du modèle des objets métiers
- Une couche **ViewModel** : responsable de la liaison entre les couches View et Service et de la gestion des données affichées dans l'interface utilisateur.
- Une couche **View** : responsable de l'affichage de l'interface utilisateur et de la gestion des interactions de l'utilisateur.
- Une couche **Repository** : responsable de la persistance des données dans la base de données ou dans des fichiers

5.1.2 - Les modules

Dans le cas d'une application multi-module, la structure est basée sur les modules Maven. Chaque module est indépendant et peut être compilé et déployé séparément, avec ses propres dépendances.

5.1.3 - Structure des sources

La structuration des répertoires du projet suit la logique suivante :

```
racine
├─ pom.xml
├─ repository
│  ├─ pom.xml
│  ├─ src
│  │  ├─ main
│  │  │  └─ java
│  │  └─ test
│  │  └─ java
│  └─ target
├─ service
│  ├─ pom.xml
│  ├─ src
│  │  ├─ main
│  │  │  ├─ java
│  │  │  └─ resources
│  │  └─ test
│  │  └─ java
│  └─ target
├─ model
│  ├─ pom.xml
│  └─ src
│  └─ main
│  └─ java
├─ viewModel
│  ├─ pom.xml
│  └─ src
│  └─ main
│  ├─ java
│  └─ resources
├─ view
│  ├─ pom.xml
│  └─ src
│  ├─ main
│  │  ├─ java
│  │  └─ resources
│  └─ test
│  ├─ java
│  └─ ressources
└─ src
   └─ lib
```

- Le répertoire "lib" contient les bibliothèques externes du projet, tandis que le répertoire "main" contient les sources principales du projet. Le répertoire "java" contient les sources Java, tandis que le répertoire "ressources" contient les fichiers de configuration et les ressources nécessaires à l'application.

5.2 - Application Web

L'application Web proposera une interface utilisateur permettant aux managers et aux clients d'accéder aux fonctionnalités suivantes :

- Pour les managers : suivre l'activité des points de vente, consulter et modifier les stocks, et gérer les comptes utilisateur.
- Pour les clients : créer ou modifier leur compte client, et créer et suivre des commandes.

En parallèle, le serveur Web mettra en place plusieurs web services pour permettre à l'application Android d'interagir avec l'ensemble des composants définis dans le système.

5.3 - Application Android

L'architecture de l'application sera de type MVVM, et sera compatible avec Android 6.0 et les versions ultérieures.

Les données seront échangées avec le serveur Web via des web services, et une base SQLite locale sera utilisée pour stocker les commandes en cours en cas de perte de connexion. Les données saisies localement seront fusionnées avec les données du serveur lors de la reconnexion, en accordant une priorité à la saisie la plus récente en cas de conflit.

L'application sera orientée données, avec l'utilisation de DAO, et sera conçue conformément aux maquettes réalisées par l'UX/UI Designer, tout en respectant les exigences de la charte graphique du client et les recommandations du Material Design de Google.

6 - POINTS PARTICULIERS

6.1 - Gestion des logs

La gestion des logs est un élément crucial du système de gestion de pizzeria. Elle permet d'enregistrer et de stocker les événements système pour une analyse ultérieure. Les objectifs de la gestion des logs comprennent la détection des erreurs système, la conformité aux exigences réglementaires et la protection du système contre les attaques malveillantes. Pour assurer une gestion efficace des logs, nous déterminerons les types d'événements à enregistrer, les méthodes de stockage et d'analyse, les politiques de rétention, la sécurité des logs, les responsabilités et les processus. Les logs seront stockés dans des fichiers de logs sur les serveurs de la pizzeria, et les logs seront régulièrement sauvegardés pour assurer la disponibilité des données en cas de défaillance du système. Les détails de ces éléments seront décrits dans la section sur la gestion des logs pour assurer une gestion efficace et sécurisée des logs système.

6.2 - Ressources

Les ressources nécessaires pour le développement et le déploiement du système de gestion de pizzeria incluent les équipements matériels, les logiciels et les licences requises. Pour le développement, les développeurs auront besoin d'ordinateurs équipés de suffisamment de RAM et de processeurs pour exécuter les outils de développement de manière fluide. Pour le déploiement, nous aurons besoin de serveurs, de stockage et de bande passante suffisants pour prendre en charge le trafic utilisateur. Les licences pour les logiciels et les outils de développement seront acquises conformément aux politiques de licence du client.

6.3 - Environnement de développement

L'environnement de développement pour le système de gestion de pizzeria inclura les outils de développement, les processus de développement et les normes de codage. Nous utiliserons les outils de développement recommandés pour la plateforme choisie et nous respecterons les meilleures pratiques en matière de développement de logiciels pour assurer la qualité et la maintenabilité du code. Le processus de développement sera basé sur une approche agile et itérative, avec des revues de code régulières pour garantir la qualité du code. Les normes de codage suivront les directives de codage du client et les bonnes pratiques de codage recommandées pour la plateforme.

6.4 - Procédure de packaging / livraison

La procédure de déploiement et de livraison est une étape critique du projet de développement de l'application de gestion de pizzeria. Elle vise à garantir la mise en production de l'application dans un environnement stable et contrôlé. Pour cela, nous définirons une procédure de déploiement qui prendra en compte les différents environnements (développement, recette, production) ainsi que les différentes étapes de déploiement (installation, configuration, tests, validation). Nous mettrons également en place un système de livraison automatisé pour faciliter la mise en production de nouvelles versions de l'application. Cette procédure de déploiement et de livraison sera documentée et régulièrement mise à jour pour assurer sa pertinence tout au long du projet.