Famery_Valentin_2_dossier_technique_052023

Date de création	@1 mai 2023 18:06
Étiquettes	

OC Pizza

Mise en place d’un nouveau système informatique pour l’ensemble des pizzerias du groupe.

Dossier de conception technique

Version 1.0

Auteur

Famery Valentin

Lead Developer

Table des matières

1 - Versions

Auteur	Date	Description	Version
Famery Valentin	01/05/2023	Création du document	1.0

2 - Introduction

2.1 - Objet du document

Le présent document constitue le dossier de conception technique de l'application OC Pizza

Un dossier de conception technique (DCT) a pour objectif de décrire en détail la conception technique d'un système ou d'un produit. Il s'agit d'un document important pour les équipes de développement et de production car il fournit des informations détaillées sur la façon dont le système doit être conçu, développé et mis en œuvre.

Voici quelques-uns des objectifs spécifiques d'un DCT :

1. Décrire l'architecture technique du système : Le DCT doit fournir une description détaillée de l'architecture technique du système, y compris les composants, les modules, les couches, les interfaces et les interactions entre les différentes parties du système.

2. Présenter les technologies et les outils utilisés : Le DCT doit expliquer les technologies et les outils utilisés pour développer et mettre en œuvre le système, y compris les langages de programmation, les frameworks, les bibliothèques, les bases de données, les serveurs d'application, les outils de développement, etc.
3. Décrire les algorithmes et les mécanismes : Le DCT doit expliquer les algorithmes, les mécanismes et les techniques utilisés pour implémenter les fonctionnalités du système, y compris les règles de validation, les algorithmes de recherche, les méthodes de calcul, les règles de sécurité, etc.
4. Spécifier les normes et les conventions : Le DCT doit décrire les normes et les conventions utilisées pour le développement et la documentation du système, y compris les conventions de nommage, les normes de codage, les conventions de documentation, etc.
5. Fournir des instructions pour la maintenance et la mise à jour : Le DCT doit fournir des instructions pour la maintenance et la mise à jour du système, y compris les procédures de sauvegarde, les instructions pour le dépannage, les procédures de mise à jour, etc.

En somme, le DCT est un document essentiel pour garantir que la conception technique du système est bien documentée et que toutes les parties impliquées dans le développement et la production disposent d'une compréhension commune de l'architecture, des technologies et des processus nécessaires pour construire et maintenir le système.

Les éléments du présents dossiers découlent :

- des besoins exprimés et recolté aupres du client OC Pizza
- du document de conception fonctionnelle

2.2 - Références

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

1. **DCF -** Famery_Valentin_1_dossier_fonctionnelle_052023 : Dossier de conception fonctionnelle de l'application
2. **DE** - Famery_Valentin_3_dossier_d’exploitation_052023 : Dossier d’exploitation

3 - Architecture Technique

3.1 Introduction

Flutter est un framework open-source créé par Google pour faciliter la création d'applications mobiles et web de haute qualité. Il utilise un langage de programmation appelé Dart pour construire des interfaces utilisateur rapides et fluides, avec des fonctionnalités avancées telles que l'animation, la gestion des états, la gestion des thèmes, et bien plus encore.

L'une des principales caractéristiques de Flutter est sa capacité à fournir une expérience utilisateur cohérente sur différentes plateformes, y compris Android, iOS, Web, Desktop et même Embedded. Cela signifie que les développeurs peuvent créer une application pour plusieurs plates-formes en utilisant une seule base de code, ce qui réduit considérablement le temps et les coûts de développement.

Flutter offre également une riche bibliothèque de widgets personnalisables et préconçus pour la création d'interfaces utilisateur modernes et esthétiques. Les développeurs peuvent personnaliser ces widgets pour s'adapter à leur marque ou à leur application spécifique, ou créer leurs propres widgets pour des besoins plus avancés

3.2 Gestion de versions

Git est un système de contrôle de version décentralisé largement utilisé pour gérer les modifications de code source lors de la création de logiciels.

Git fonctionne en enregistrant les modifications apportées à un projet de manière incrémentielle, en créant des instantanés (commits) du code à différents stades de son développement. Ces instantanés peuvent être enregistrés localement sur l'ordinateur de l'utilisateur ou partagés avec d'autres développeurs via un serveur Git. Chaque instantané contient une référence à l'instantané précédent, ce qui permet de suivre facilement les modifications et de revenir en arrière si nécessaire.

Git dispose également de nombreuses fonctionnalités utiles pour les développeurs, telles que la gestion des branches (pour travailler sur plusieurs versions du code simultanément), la fusion de modifications de différents contributeurs et la résolution de conflits lors de la fusion.

3.3 Langage

Dart est un langage de programmation orienté objet développé par Google en 2011. Il est principalement utilisé pour le développement web et mobile, en particulier pour le développement d'applications Flutter.

Dart est un langage fortement typé, ce qui signifie que les variables et les objets doivent être déclarés avec un type spécifique. Il prend également en charge l'héritage de classes, les interfaces, les mixins et les extensions, offrant ainsi aux développeurs une grande flexibilité pour créer des objets complexes et réutilisables.

Dart dispose également d'une gestion automatique de la mémoire, ce qui signifie que les développeurs n'ont pas besoin de s'occuper de la gestion manuelle de la mémoire et que le langage prend en charge la collecte des déchets.

En plus de cela, **Dart** dispose d'une bibliothèque standard riche, qui comprend des fonctionnalités pour la gestion des collections, la gestion des chaînes, l'accès aux fichiers et les entrées-sorties, entre autres. Dart prend également en charge la programmation asynchrone et la gestion des futures, ce qui facilite le développement d'applications réactives et performantes.

3.4 Frontend

3.4.1 Flutter Architecture

Flutter ne suit pas une architecture spécifique telle que MVVM (Modèle-Vue-Vue Modèle), mais plutôt une architecture basée sur les widgets, appelée architecture de widgets.

Dans l'architecture de widgets, chaque élément de l'interface utilisateur est représenté par un widget, qui est un objet réutilisable et configurable qui peut être intégré dans d'autres widgets pour former l'interface utilisateur. Les widgets peuvent être des éléments d'interface utilisateur simples, tels que des boutons et des champs de texte, ou des widgets plus complexes, tels que des listes et des grilles.

Les widgets peuvent être répartis en deux catégories : les widgets Stateless, qui ne contiennent pas d'état interne et ne peuvent pas être modifiés une fois qu'ils ont été créés, et les widgets Stateful, qui contiennent un état interne et peuvent être mis à jour dynamiquement.

L'architecture de widgets est souvent considérée comme une approche plus légère et plus flexible que d'autres architectures plus lourdes, telles que MVVM ou MVC (Modèle-Vue-Contrôleur), et elle est particulièrement adaptée aux applications mobiles où la performance et la légèreté sont des préoccupations importantes.

3.4.2 Stripe

⚠️ Assurer d'avoir installé flutter_stripe : https://pub.dev/packages/flutter_stripe

⚠️ Assurer d'avoir installé flutter_stripe_web : https://pub.dev/packages/flutter_stripe_web

API REST de paiement Stripe : Stripe est une plateforme de paiement en ligne qui permet aux entreprises de collecter des paiements via leur application. L'API de paiement Stripe est utilisée pour intégrer la fonctionnalité de paiement dans l'application.

3.5 Backend

3.5.1 Architecture DartFrog

Dans un framework backend comme DartFrog , les composants routes et middlewares sont deux éléments essentiels pour la création d'API web.

Les routes sont des moyens de définir la structure de l'API et de spécifier comment les différentes ressources sont exposées. Elles permettent de répondre aux requêtes HTTP en associant une URL et une méthode HTTP à une action à exécuter. Les routes permettent ainsi de définir les points d'entrée de l'API, les ressources disponibles, les méthodes HTTP autorisées et les paramètres attendus.

Les middlewares sont des fonctions qui sont exécutées avant ou après une route, et qui permettent de manipuler la requête ou la réponse HTTP. Les middlewares sont souvent utilisés pour ajouter des fonctionnalités communes à l'ensemble de

l'application, telles que l'authentification, la gestion des erreurs, la compression de la réponse ou la validation des données. Les middlewares peuvent également être utilisés pour contrôler l'accès aux ressources, en vérifiant par exemple les autorisations de l'utilisateur.

En résumé, les routes et les middlewares sont deux composants importants pour la création d'API web dans un framework backend. Les routes permettent de définir la structure de l'API et les points d'entrée des ressources, tandis que les middlewares permettent de manipuler les requêtes et les réponses HTTP pour ajouter des fonctionnalités communes ou pour contrôler l'accès aux ressources.

Le principe des microservices consiste à diviser une application en plusieurs services indépendants et spécialisés, chacun d'entre eux étant développé, déployé et maintenu séparément. Chaque service est généralement déployé sur une infrastructure de cloud computing et peut être mis à l'échelle horizontalement pour répondre à la demande.

Dans le contexte des microservices, les packages sont souvent utilisés pour regrouper des fonctionnalités spécifiques en modules réutilisables. Chaque package peut contenir plusieurs classes ou fonctions qui peuvent être partagées entre les différents services. Les packages permettent ainsi d'organiser le code de manière modulaire et de faciliter la réutilisation du code dans différents contextes.

Lors de la conception d'un système basé sur des microservices, il est courant de diviser les différentes fonctionnalités en différents packages, en fonction de leur domaine d'application ou de leur spécialisation. Chaque package peut ensuite être développé, testé et déployé indépendamment des autres packages, ce qui permet une mise à jour plus rapide et plus flexible de l'application.

Very Good Ventures le createur du framework DartFrog nous fourni des outils pour diviser le code en plusieurs packages independant

- Un Template : Very Good Dart

> ⚠️ Assurer d'avoir installé mason_**cli** pour utilisé Very Good Dart : https://pub.dev/packages/mason_cli

```
├── .github
│   ├── ISSUE_TEMPLATE
│   │   ├── bug_report.md
│   │   ├── build.md
│   │   ├── chore.md
│   │   ├── ci.md
│   │   ├── config.yml
│   │   ├── documentation.md
│   │   ├── feature_request.md
│   │   ├── performance.md
│   │   ├── refactor.md
│   │   ├── revert.md
│   │   ├── style.md
│   │   └── test.md
│   ├── PULL_REQUEST_TEMPLATE.md
│   ├── dependabot.yaml
│   └── workflows
│       └── main.yaml
├── .gitignore
├── CHANGELOG.md
├── LICENSE
├── README.md
├── analysis_options.yaml
├── coverage_badge.svg
├── lib
│   ├── src
│   │   └── my_package.dart
│   └── my_package.dart
├── pubspec.yaml
└── test
    └── src
        └── my_package_test.dart
```

En résumé, le principe des microservices consiste à diviser une application en services indépendants et spécialisés, et les packages sont souvent utilisés pour organiser le code de manière modulaire et faciliter la réutilisation du code dans différents contextes. Chaque package peut être développé, testé et déployé indépendamment des autres packages, ce qui permet une mise à jour plus rapide et plus flexible de l'application.

3.5.2 Base de données

1. Backend DartFrog : DartFrog est un framework de développement backend en Dart qui peut être utilisé pour créer des applications web et mobiles. Le backend est responsable de la logique de l'application, de la gestion des requêtes des clients, de la manipulation des données et de la communication avec la base de données MySQL.
2. Base de données MySQL : MySQL est un système de gestion de base de données relationnelles open source. Il est utilisé pour stocker les données de l'application, telles que les informations des utilisateurs, les produits et les commandes.

Chaque composant de l'architecture remplit une fonction spécifique pour permettre à l'application de fonctionner de manière efficace et sans problème. Le backend gère les opérations de l'application, la base de données stocke les données nécessaires, le frontend offre une interface utilisateur agréable et facile à utiliser, et l'API de paiement Stripe permet aux utilisateurs de payer pour les services ou les produits offerts par l'application.

4 - Architecture de Déploiement

- Google Cloud Web Hosting : Il s'agit d'un service d'hébergement de sites web proposé par Google Cloud. Il permet aux entreprises et aux particuliers de déployer et de gérer leurs sites web de manière rapide et facile en utilisant les ressources informatiques de Google Cloud. Les utilisateurs peuvent choisir parmi plusieurs options de déploiement, y compris les conteneurs et les machines virtuelles.
- Google Cloud SQL : C'est un service de base de données géré proposé par Google Cloud. Il permet aux entreprises de créer, de configurer et de gérer facilement des bases de données MySQL et PostgreSQL sur le Cloud. Les utilisateurs peuvent se concentrer sur le développement de leurs applications sans avoir à se soucier des tâches de gestion de base de données, telles que la mise à l'échelle, la sauvegarde et la récupération.
- Google Cloud Run : C'est un service d'exécution de conteneurs géré proposé par Google Cloud. Il permet aux développeurs d'exécuter facilement leurs applications conteneurisées sur le Cloud, sans avoir à gérer l'infrastructure sous-jacente. Les utilisateurs peuvent déployer leurs conteneurs à l'aide de divers outils de développement populaires, tels que Docker et Kubernetes.
- Google Play Console :

permet aux développeurs de créer et distribuer des applications pour les appareils Android. La plateforme fournit des outils, des ressources et une documentation pour aider les développeurs à créer, tester et distribuer leurs applications sur Google Play, la boutique d'applications Android officielle.

Les développeurs peuvent utiliser la Google Play Console pour soumettre leurs applications à Google pour examen et approbation avant qu'elles ne soient mises à disposition des utilisateurs. La plateforme fournit également des statistiques détaillées sur les téléchargements, les avis et les revenus générés par les applications.

Les développeurs peuvent également utiliser la Google Play Console pour gérer les mises à jour, les prix, les paramètres de distribution et les stratégies de monétisation de leurs applications. La plateforme est un outil essentiel pour les développeurs Android qui souhaitent atteindre un large public et générer des revenus grâce à leurs applications.

- **Apple Developer/AppStore** :

Apple Developer est une plateforme fournie par Apple Inc. qui permet aux développeurs de créer et distribuer des applications pour les appareils iOS, iPadOS, macOS, watchOS et tvOS. La plateforme offre divers outils, ressources et documentation pour aider les développeurs à construire, tester et distribuer leurs applications.

L'App Store d'Apple est une plateforme de distribution numérique pour les applications mobiles sur les appareils iOS et les ordinateurs macOS. Elle est gérée par Apple Inc. et constitue le moyen principal de distribution des applications iOS aux utilisateurs. Les développeurs doivent soumettre leurs applications à l'App Store pour examen et approbation par Apple avant qu'elles ne puissent être téléchargées par les utilisateurs.

5 - Points particuliers

6 - Glossaire

Android	C'est un système d'exploitation mobile développé par Google. Il est utilisé par des milliards de personnes à travers le monde et est compatible avec de nombreux types d'appareils mobiles, tels que les smartphones, les tablettes et les télévisions. Les développeurs peuvent utiliser Android pour créer des applications mobiles pour une variété d'utilisations.
iOS	C'est un système d'exploitation mobile développé par Apple. Il est utilisé par des millions de personnes à travers le monde et est exclusif aux appareils mobiles d'Apple, tels que l'iPhone et l'iPad. Les développeurs peuvent utiliser iOS pour créer des applications mobiles pour les appareils Apple, en utilisant des outils de développement tels que Xcode et Swift.
SQL	SQL (Structured Query Language) est un langage informatique standardisé utilisé pour gérer et manipuler des bases de données relationnelles. Il a été développé dans les années 1970 et est devenu le langage de choix pour les applications de gestion de données, en raison de sa simplicité, de sa flexibilité et de sa compatibilité avec la plupart des systèmes de gestion de bases de données relationnelles.
Frontend	Le frontend fait référence à la partie visible de l'application ou du système avec laquelle les utilisateurs interagissent. Cela peut inclure l'interface utilisateur, les boutons, les formulaires, les graphiques, les animations, les icônes et tout autre élément visuel de l'application.
Backend	Le backend, quant à lui, fait référence à la partie invisible de l'application ou du système qui est responsable de la gestion des données, des calculs et des fonctionnalités. Cela peut inclure la gestion des bases de données, le traitement des transactions, la gestion des utilisateurs et des autorisations, et toutes les autres fonctions nécessaires pour faire fonctionner l'application.