OC Pizza

Système de gestion pour pizzeria

Dossier d'exploitation

Version 1.2

Auteur
Brice NIATEL
Développeur d'application android

TABLE DES MATIERES

1 - VERSIONS

Auteur	Date	Description	Version
Brice NIATEL	28/07/2021	Création du document	1.0
Brice NIATEL	10/09/2021	Finition de la rédaction du document	1.1
Brice NIATEL	17/09/2021	Relecture	1.2

2 - Introduction

2.1 - Objet du document

Le présent document constitue le dossier d'exploitation de l'application OC Pizza pour les l'équipe technique, les développeurs ainsi que les personnes qui maintiendront le système.

L'objectif du document est de présenter les informations dont l'équipe d'exploitation a besoin pour pouvoir assurer une exploitation du système et pouvoir réagir de manière appropriée lorsqu'un problème surgit.

2.2 - Références

Pour de plus amples informations, se référer :

1. **Projet 10 - Dossier de conception fonctionnelle** : Dossier de conception technique de l'application.
2. **Projet 10 - Dossier de conception technique** : Dossier de conception technique de l'application.
3. **Projet 10 - PV Livraison** : Procès-verbal de la livraison.

3 - PREREQUIS

3.1 - Système

L'application sera hébergée sur Clever Cloud, une solution Française, nous utiliserons son ***CLI*** pour le déploiement sur Clever Cloud sur un ordinateur. Seront installés avec l'application les add-ons ***PostreSQL*** et ***Cellar S3 Storage*** pour la gestion des données ainsi que les sauvegardes des logs, facture, etc.

Sources :
https://www.clever-cloud.com/en/pricing
https://www.clever-cloud.com/doc/reference/clever-tools/getting_started/
https://github.com/CleverCloud/clever-tools

Le site internet sera accessible sous le nom de domaine « ocpizza.fr » qui sera enregistré sous OVH.

Source :
https://bit.ly/3kjyYTr
https://www.ovh.com/fr/

3.1.1 - Base de données

Nous utiliserons une base de données ***PostgreSQL*** en version 13.4 directement héberger avec ***Clever Cloud*** avec comme l'option « L SMALL SPACE », qui aura des sauvegardes automatiques tous les jours et un maximum de sept seront gardées en mémoire.

Cette option nous permet d'avoir 40GB de place, 500 connexions simultanées ce qui est intéressant pour notre besoin.

Sources :
https://www.clever-cloud.com/en/pricing
https://www.clever-cloud.com/doc/deploy/addon/postgresql/postgresql/

3.1.2 - Serveur Web

Un serveur hébergeant l'application web, ***Apache TOMCAT*** sera configuré pour ce système en version 8.5.70. Ce logiciel est open source, est régulièrement maintenu, fiable, stable avec une facilité de configuration mais attention à ne pas se perdre avec les configurations complexes qu'il peut y avoir.

Source :

https://tomcat.apache.org/whichversion.html

https://tomcat.apache.org/tomcat-8.5-doc/index.html

3.2 - Bases de données

Les bases de données et schémas suivants doivent être accessibles et à jour :

- **oc_pizza_db :** version 1.1

3.3 - Web-services

Les web services suivants doivent être accessibles et à jour :

3.3.1 - API Stripe

Stripe est une API de type ***REST*** permettant de créer une page de paiement ainsi que toute la gestion de celui-ci. Sa version est « 2020-08-27 ».

Il sera important d'effectuer une veille sur cette API pour des raisons de cybersécurité car le sujet bancaire en sensible.

Sources :
https://stripe.com/docs
https://stripe.com/docs/api
https://www.clever-cloud.com/doc/extend/add-ons-api/

3.3.2 - API Google Map

L'API Google Map permettra à l'utilisateur de choisir et situer les restaurants, de plus les livreurs s'en serviront aussi en couplant l'API Google Direction. Sa version est « 3.46 ».

Source :
https://developers.google.com/maps/documentation/javascript/overview

3.3.3 - API Google Direction

L'API Google Direction permettra au livreur de lui indiquer son parcours en fonction du moyen de locomotion choisi. Sa version est « 3.12 ».

https://developers.google.com/maps/documentation/directions/quickstart#next_steps

3.4 - Autres Ressources

Les autres services seront les modules développés pour les besoins de l'application présent au sein de ***GitHub***.

4 - PROCEDURE DE DEPLOIEMENT

4.1 - Déploiement de l'application

4.1.1 - Back-office

4.1.1.1 - Déploiement

Le déploiement du back-office se fera via ***Clever Cloud CLI,*** voici la méthode ci-dessous :

1. Créer le projet sur Clever Cloud avec la commande « clever create –type <maven> <nom application> --region <par> »
2. La page « Informations » de votre application vous donne votre URL de déploiement git. Cela ressemble à « *git+ssh://git@push.clever-cloud.com/<application id>.git* »
3. Ajoutez l'URL de déploiement avec « git *remote add <nom> <url déploiement de git>.*
4. Ajoutez vos fichiers via « *git add <chemin des fichiers>* » et validez-les via « *git commit -m <message de validation>* ».
5. Poussez l'application sur Clever Cloud avec « *git push <nom> master* ».
6. Déployez l'application avec « clever deploy »

Source :
https://www.clever-cloud.com/doc/deploy/application/java/java-maven/

4.1.2 - Front-office

4.1.2.1 - Déploiement

Le front-office se fera sur une nouvelle application ***Clever Cloud*** via ligne de commande, voici la méthode ci-dessous :

1. Créer le projet sur Clever Cloud avec la commande « clever create –type <node> <nom application> --region <par> »
2. La page « Informations » de votre application vous donne votre URL de déploiement git. Cela ressemble à « *git+ssh://git@push.clever-cloud.com/<application id>.git* »
3. Ajoutez l'URL de déploiement avec « git *remote add <nom> <url déploiement de git>.*
4. Ajoutez vos fichiers via « *git add <chemin des fichiers>* » et validez-les via « *git commit -m <message de validation>* ».

5. Poussez l'application sur Clever Cloud avec « *git push <nom> master* ».
6. Déployer l'application avec éclever deploy »

Sources :
https://www.clever-cloud.com/doc/deploy/application/javascript/by-framework/nodejs/

4.1.2.2 - Variable d'environnement

Le serveur d'application sera exécuté avec la variable d'environnement « NODE_ENV=production ». La commande à effectuer est :

1. clever env set NODE_ENV production
2. clever env set JAVA_VERSION 8

Sources :
https://www.clever-cloud.com/doc/deploy/application/javascript/by-framework/nodejs/

4.2 - Add-ons

4.2.1.1 - Lier une base de données à l'application

Sur l'écran « *Applications* », vous pourrez ajouter des add-ons via le bouton « *Add an add-on* » et sélectionner « *PostgreSQL* » pour ensuite finir la configuration à votre guise avec le fichier de configuration de l'application « application.yml ».

Source :
https://www.clever-cloud.com/doc/deploy/application/javascript/by-framework/nodejs/
https://www.clever-cloud.com/doc/deploy/addon/postgresql/postgresql/

4.2.1.2 - Configuration des modules

Le module ***config*** gère la configuration de tous les modules avec ***Spring Initializr***, les fichiers de configuration des différents modules seront rendus visible grâce à un serveur ***Eureka***.

Source :
https://openclassrooms.com/fr/courses/4668216-optimisez-votre-architecture-microservices/5176545-externalisez-la-configuration-de-vos-microservices

4.3 - Gestions des Batches

La gestion des différents batches se fait avec le module ***task-manager*** qui permet aussi de faire la planification de jobs tel que l'envoie de mails, lancement de procédure de sauvegarde, traitement des logs etc.

4.4 - Vérifications

Le CLI de Clever Cloud ou le site web retournera toujours le bon déploiement ou non de l'application.

4.5 - Logs

Les logs sont accessibles directement via le site web de Clever Cloud en sélectionnant l'application sinon, via la ligne de commande avec « clever log ».

5 - PROCEDURE DE DEMARRAGE / ARRET

5.1 - Base de données

La gestion des démarrages et arrêts de la base de données se fera directement au sein de l'application web de Clever Cloud.

5.2 - Batches

Les différents fichiers batches peuvent être démarrés et stoppés en paramétrant le fichier le batch à « true/enable » grave au module ***config*** et en redémarrant le module ***task-manager***.

5.3 - Application

L'application peut être entièrement gérée depuis le site internet de Clever Cloud, arrêter, redémarrer, reconstruire et redémarrer, redémarrer au dernier commit poussé.

5.4 - Module

Les modules sont gérés via le serveur Eureka avec les deux commandes ci-dessous :

- restart_service <nom service>
- stop_service <nom service>

6 - PROCEDURE DE MISE A JOUR

6.1 - Base de données

Pour mettre à jour et gérer la base de données dans son ensemble, un logiciel sera nécessaire du type pgAdmin ou bien le logiciel web de Clever Cloud.

Sources :

https://www.clever-cloud.com/doc/deploy/addon/postgresql/postgresql/
https://dbms-adminer.clever-cloud.com/

6.2 - Application et application web

L'application et l'application web se mettent à jour avec la même méthode, via le site internet de Clever Cloud sur la page de l'application, sélectionner « redémarrer au dernier commit poussé » après avoir effectué les changements sur github.

6.3 - Module

Pour mettre à jour un module, il suffit d'effectuer les changements sur github, ensuite de redémarrer l'application sur Clever Cloud avec le bouton « redémarrer au dernier commit poussé ».

6.4 - Clever Cloud

Il est possible d'améliorer les machines louées ainsi que les services de Clever Cloud pour mieux s'adapter aux besoins et aux charges du site.

Changer d'offre en augmentant la ram / CPU, plus de stockage etc.

7 - SUPERVISION/MONITORING

7.1 - Supervision de l'application web

Les différents objectifs des tests peuvent inclure :

- Vérifier si toutes les exigences spécifiées ont été satisfaites,
- Valider si l'objet de test est complet et fonctionne comme attendu par les utilisateurs et autres parties prenantes,
- Construire la confiance dans le niveau de qualité de l'objet de test,
- Prévenir des défauts,
- Trouver des défaillances et défauts,
- Fournir suffisamment d'information aux parties prenantes pour leur permettre de prendre des décisions éclairées, en particulier en ce qui concerne le niveau de qualité de l'objet de test,
- Réduire le niveau de risque d'une qualité logicielle inadéquate (p. ex. des défaillances non détectées auparavant se produisant en opération),
- Se conformer aux exigences ou aux normes contractuelles, légales ou réglementaires, et/ou vérifier la compatibilité de l'objet de test avec de telles exigences ou normes.

Afin de tester que l'application web est toujours fonctionnelles, une stratégie de test est adaptée.

Nous aurons différents niveaux de test à effectuer tout au long du développement du projet :

- **Test de composant / unitaire :** Tester le développement des différents modules et du back-end de l'application.
- **Test d'intégration :** Tester les interactions entre les différents composants tel que les modules, entre l'application web, android et iOS avec l'application back-end et la base de données.
- **Test système :** Tester les capacités du système ainsi qu'analyser ses comportements en exécutant des taches de bout en bout. Permet de voir le fonctionnement du système dans son ensemble.
- **Test d'acceptation :** comme les tests système, se concentrent généralement sur le comportement et les capacités d'un système, mais ses objectifs seront de :
    - Établir la confiance dans la qualité du système dans son ensemble,
    - Valider que le système est complet et qu'il fonctionnera comme attendu,
    - Vérifier que les comportements fonctionnels et non-fonctionnels du système sont tels que spécifiés.

Pour les tests unitaires, les tests de non-régressions, les tests de parcours utilisateurs aussi appelés test de bout en bout, les plus importants et les plus utilisés seront automatisés dans le but de gagner du temps.

A la fin de chaque sprint, sera donc écrit de nouveau test pour les nouvelles fonctionnalités, de plus, il sera testé le système dans son ensemble avec tous les tests automatisés déjà écrit.

Ces tests seront lancés sur différentes plateformes avec différentes versions :

- **Explorateur web :** Chrome/ Safari / Firefox / Samsung Internet / Edge Legacy / Internet Explorer et Opera.
- **Android :** Google déclare une compatibilité ascendante à 99%. Il sera utilisé alors un appareil Android avec une version dite « stock » c'est à dire sans surcouche constructeur, la version sera 5.0. En fonction des différentes librairies utilisées nous serons amenés à effectuer ces tests sur d'autres versions. Plusieurs appareils avec diverses tailles pour des tests non fonctionnels sur l'aspect visuel des applications
- **iOS :** Avec un panel de mobile restreint sur peu de version, il suffira de tester pour différentes versions et sur différent modèle pour les tous les tests.

Pour l'automatisation des tests, **Appium** est un logiciel fiable et peut être utilisé sur les trois plateformes Android, iOS et Explorateur Web.

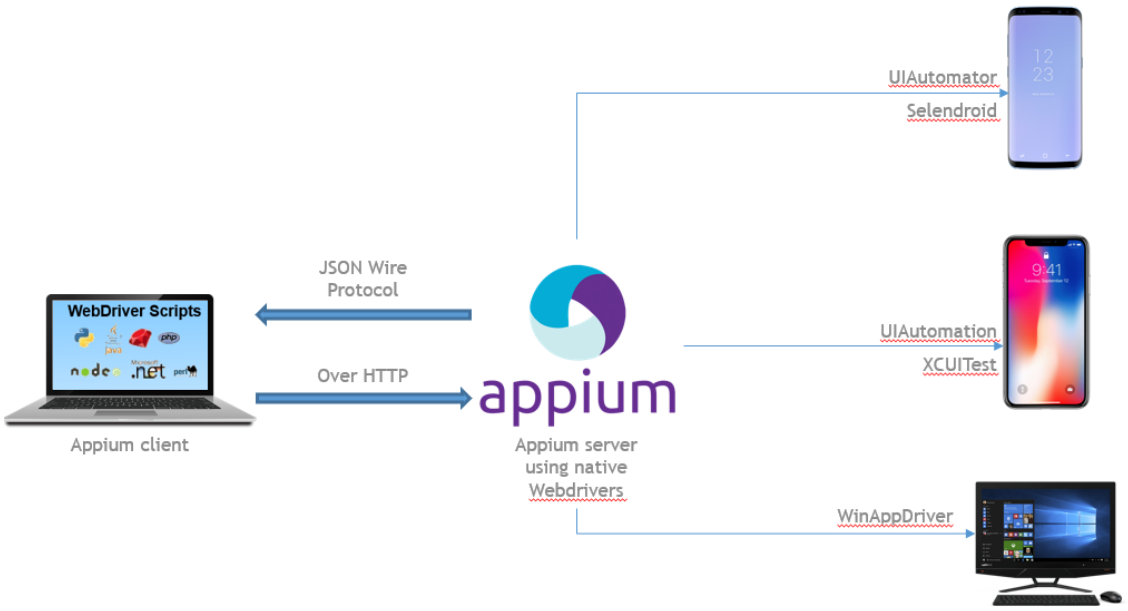

Image 1 : Appium

Sources :

Syllabus CTFL Version 2018
Android Developer

8 - PROCEDURE DE SAUVEGARDE ET RESTAURATION

PgAdmin sera l'outil permettant d'effectuer toute commande et gestion de l'ensemble de la base de données donc de réaliser les sauvegardes de même que la restauration de l'ensemble des données. Il est possible également de gérer les sauvegardes et restauration directement depuis Clever Cloud.

Il est important de tester régulièrement les sauvegarde de la base données pour vérifier si tout fonctionne bien.

8.1 - Sauvegarde SQL

La méthode pour effectuer une sauvegarde d'une base de données :

1. Se connecter à la base de données,
2. Produire un fichier texte de commande SQL « pg_dump base_de_donnees > fichier_sauvegarde » ou « pg_dump base_de_donnees | gzip > fichier_sauvegarde.gz » pour les bases de données plus grandes.

Sources :
https://docs.postgresql.fr/9.6/backup.html

8.2 - Restauration

La méthode pour effectuer une restauration d'une base de données :

1. Se connecter à la base de données,
2. Utiliser un fichier texte de commande SQL « psql base_de_donnees < fichier_sauvegarde » ou « gunzip -c fichier_sauvegarde.gz | psql base_de_donnees » pour les bases de données plus grandes.

Sources :
https://docs.postgresql.fr/9.6/backup.html

9 - GLOSSAIRE

Clever Cloud : C'est une plateforme d'automatisation informatique. Nous gérons tout le travail opérationnel pendant que vous vous concentrez sur la valeur de votre entreprise.

CLI : Interface Ligne de Commande.

REST : Signifie REpresentational State Transfer, constitue un style architectural et un mode de communication fréquemment utilisé dans le développement de services Web.

GitHub : est un service web d'hébergement et de gestion de développement de logiciels, utilisant le logiciel de gestion de versions Git.
Add-on : Extension d'un programme informatique qui lui apporte de nouvelles fonctionnalités.