

**Réaliser Par :**

**OTHMAN MABROUR / NAIFI Youssef / ABRIL Yahia**

**5IIR G3**

Table des matières

[Introduction Générale 5](#_Toc156676147)

[**1. Introduction** 6](#_Toc156676148)

[**2. Architecture Microservices** 6](#_Toc156676149)

[**3. Conception des Microservices** 9](#_Toc156676150)

[**4. Conteneurisation avec Docker** 9](#_Toc156676151)

[**5. CI/CD avec Jenkins** 16](#_Toc156676152)

[**6. Déploiement Automatique** 18](#_Toc156676153)

[**7. Intégration de SonarQube** 21](#_Toc156676154)

[**Pipline :** 24](#_Toc156676155)

[**8. Conclusion** 26](#_Toc156676156)

# **Introduction Générale**

Les systèmes informatiques complexes requièrent une architecture pensée, évolutive et capable de répondre aux demandes croissantes de performance, de flexibilité et de facilité de gestion. Le développement de solutions logicielles, telles que les plateformes de réservation pour des événements, des spectacles et des cinémas, nécessite une approche architecturale solide. Dans ce contexte, l'architecture microservices émerge comme une solution innovante pour répondre à ces défis. Cette approche divise une application en composants autonomes et indépendants, appelés microservices, permettant une évolution et une maintenance agiles.

Notre projet se focalise sur la création d'une plateforme de réservation de billets, s'appuyant sur une architecture microservices bien pensée. Cette plateforme se compose de plusieurs microservices tels que le serveur, la passerelle (gateway), l'administration, le client, les microservices dédiés aux événements et aux réservations. La communication entre ces microservices est orchestrée avec l'utilisation du modèle Rest Template, favorisant ainsi une intégration légère et efficace basée sur les principes REST (Representational State Transfer).

Pour assurer la portabilité, la gestion des dépendances simplifiée, et faciliter le déploiement et la mise à l'échelle des microservices, nous avons choisi d'implémenter la conteneurisation avec Docker. Cette approche permet également une répétabilité des environnements, garantissant que chaque instance fonctionne de manière cohérente, indépendamment de l'environnement d'exécution.

Le déploiement automatique, facilité par l'utilisation de Jenkins en conjonction avec Ngrok et des webhooks, représente un pilier essentiel de la disponibilité continue de notre plateforme. Cette combinaison offre un déploiement automatisé à chaque mise à jour du code source, garantissant une réactivité accrue aux changements et une expérience utilisateur sans interruption.

Enfin, l'intégration de SonarQube avec Jenkins assure la qualité du code, détectant et résolvant les problèmes potentiels, contribuant ainsi à la création d'une base de code propre, maintenable et conforme aux normes établies.

Ce rapport explore en détail chaque composant de notre architecture microservices, du choix de l'architecture à la mise en œuvre des technologies clés, tout en mettant en lumière les avantages et les perspectives de cette approche novatrice.

**1. Introduction**

* **Aperçu du projet :**

Le projet vise à développer une application web de réservation de billets pour des événements, des spectacles, des cinémas... Notre application permettra aux utilisateurs de rechercher, consulter des informations sur les événements, et effectuer des réservations en ligne. L'objectif est de créer une solution robuste, scalable et facile à maintenir.

* **Importance de l'architecture microservices :**

L'architecture microservices a été choisie pour ce projet en raison de ses nombreux avantages. Elle favorise la scalabilité, la flexibilité et la facilité de déploiement. Chaque microservice est indépendant, ce qui permet une évolution et une maintenance aisées. De plus, cette architecture facilite la gestion des différentes fonctionnalités de la plateforme.

**2. Architecture Microservices**

* **Architecture :**

L'architecture microservices est composée des éléments suivants : serveur, gateway, admin, client, événements et réservation.

**Une image contenant texte, diagramme, capture d’écran, ligne

Description générée automatiquement**

* **Description des services :**

Serveur : Héberge tous les microservices et assure la communication entre eux.

Gateway : Routage des requêtes clients vers les services appropriés.

Admin : Gestion des fonctionnalités d'administration telles que la gestion des utilisateurs, des événements et des réservations.

Client : Interface utilisateur fournissant la recherche et la réservation de billets.

Événements : Stockage et gestion des informations relatives aux événements tels que les détails, les horaires et les emplacements.

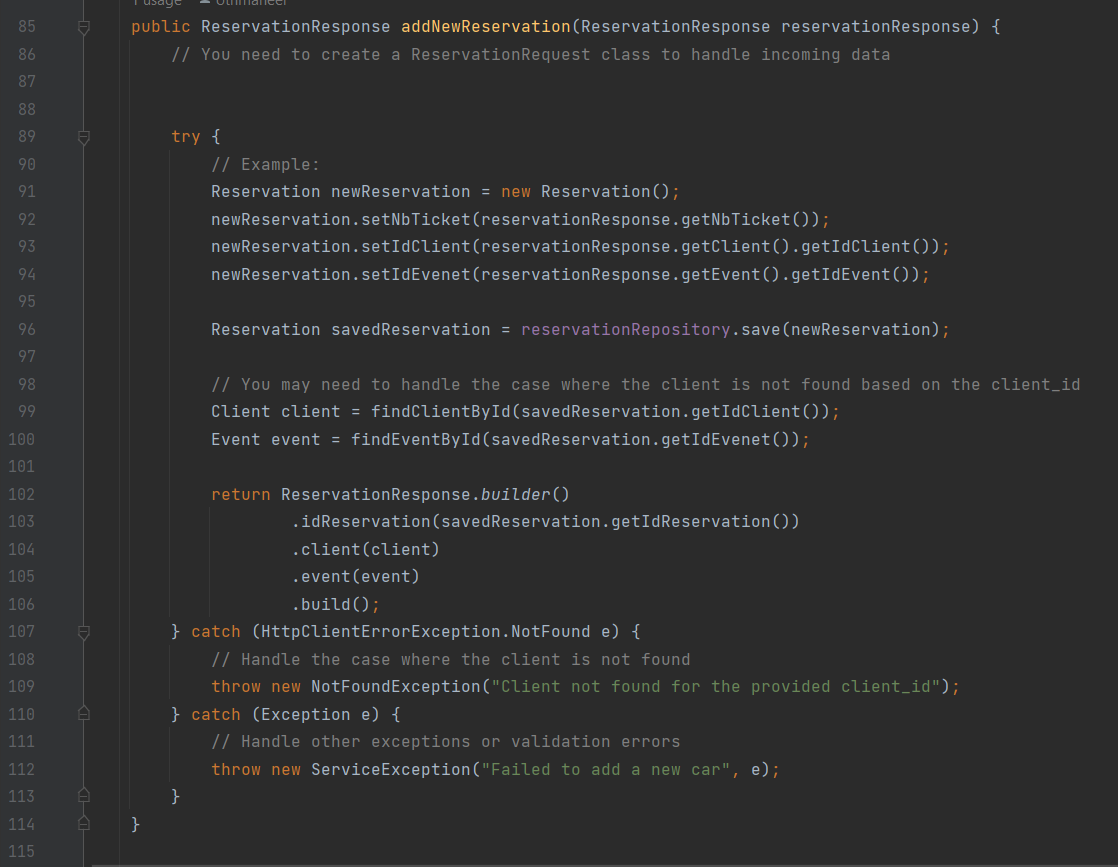
Réservation : Gestion des opérations de réservation, y compris la disponibilité des billets et la confirmation des réservations.

* **Mécanismes de communication : (Rest Template) :**

La communication entre les microservices est réalisée à l'aide du modèle Rest Template, basé sur les principes REST (Representational State Transfer).

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement



Un niveau de la couche service du microservice on trouve la communication entre microservice ou utilisant le gateway. Un exemple concret de cette interconnexion se produit lors de l'ajout d'une réservation.

Lorsqu'une réservation doit être effectuée, la couche service du microservice responsable de la gestion des réservations intervient. Pour ce faire, elle communique avec d'autres microservices, en l'occurrence le microservice client et le microservice événements.

Tout d'abord, pour associer une réservation à un client spécifique, le service de réservation effectue une requête vers le microservice client pour récupérer les informations du client concerné. Cette étape permet de garantir que la réservation est correctement liée au profil du client, assurant ainsi une traçabilité et une personnalisation appropriées.

En parallèle, pour déterminer les détails de l'événement pour lequel la réservation est effectuée, la couche service de réservation communique avec le microservice événements. Cette communication permet de récupérer les informations nécessaires sur l'événement en question, telles que la date, l'emplacement et d'autres détails pertinents.

L'ajout de la réservation ne peut aboutir si le cas de réservation est déjà enregistré. Cela garantit l'intégrité du processus de réservation en évitant les doublons et en assurant une gestion efficace des réservations.

**3. Conception des Microservices**

* **Approche de conception pour chaque service :**

Chaque microservice est conçu de manière indépendante, avec une approche spécifique pour ses fonctionnalités.

Serveur : Déployé en tant qu'hôte principal, gérant la communication entre les services.

Gateway : Gère le routage des requêtes clients vers les microservices appropriés.

Admin : Fonctionnalités d'administration avec des modules distincts pour les utilisateurs, les événements et les réservations.

Client : Interface utilisateur simple pour une expérience utilisateur optimale.

Événements : Stockage des données sur les événements avec des API pour la récupération et la modification.

Réservation : Gestion des opérations de réservation.

**4. Conteneurisation avec Docker**

* **Implémentation et avantages :**

La solution est conteneurisée à l'aide de Docker pour assurer une portabilité et une gestion des dépendances simplifiées. Cela facilite le déploiement et la mise à l'échelle des microservices de manière efficace.

Étapes d'Implémentation :

1. Création des Dockerfiles :

Pour chaque microservice, un fichier Dockerfile est créé pour définir les dépendances, les étapes d'installation et la configuration nécessaire.

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

1. Configuration du Docker Compose :

Un fichier docker-compose.yml est créé pour définir les services, les dépendances et les configurations réseau.

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

Une image contenant texte, capture d’écran, logiciel, ordinateur

Description générée automatiquement

1. Exécution de la construction avec Maven :

Dans chaque répertoire de microservice, on execute la commande Maven pour construire le fichier JAR.



1. Exécution de Docker Compose :

À la racine du projet, on exécute la commande Docker Compose pour démarrer tous les microservices.



Résultat d’implémentation :

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

Une image contenant texte, capture d’écran, logiciel, Icône d’ordinateur

Description générée automatiquement

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquementUne image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquementUne image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquementUne image contenant capture d’écran, texte, logiciel, Logiciel multimédia

Description générée automatiquement

Avantages :

* Portabilité :

Les conteneurs Docker encapsulent toutes les dépendances et configurations nécessaires, assurant ainsi la portabilité entre différents environnements.

* Gestion Simplifiée des Dépendances :

Docker permet d'isoler les dépendances de chaque microservice, évitant les conflits et simplifiant la gestion des dépendances.

* Facilité de Déploiement :

La conteneurisation simplifie le déploiement en garantissant que chaque microservice fonctionne dans son propre conteneur, indépendamment des autres.

* Mise à l'Échelle Efficace :

La nature légère des conteneurs Docker permet une mise à l'échelle efficace en ajoutant ou en retirant des instances de microservices selon les besoins.

* Répétabilité des Environnements :

Les conteneurs Docker garantissent que chaque environnement, du développement à la production, est cohérent et répétable.

**5. CI/CD avec Jenkins**

* **Processus et configuration :**

Le processus CI/CD est mis en place à l'aide de Jenkins, automatisant le processus de construction, de test et de déploiement. Cela garantit une intégration continue et une livraison rapide des nouvelles fonctionnalités.

Le pipeline va être affiché dans le 7eme chapitre.

Une image contenant texte, logiciel, Icône d’ordinateur, Page web

Description générée automatiquement

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

Une image contenant texte, capture d’écran, logiciel, Icône d’ordinateur

Description générée automatiquement

**6. Déploiement Automatique**

* **Utilisation de Ngrok :**

L'utilisation de Ngrok permet un déploiement automatique et efficace des microservices, assurant ainsi une disponibilité constante de la plateforme.

L'utilisation de Ngrok est particulièrement bénéfique lorsqu'elle est associée à des webhooks. Les webhooks sont des mécanismes permettant à un système d'informer automatiquement un autre système lorsqu'un événement spécifique se produit.

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

Une image contenant texte, logiciel, Logiciel multimédia, Icône d’ordinateur

Description générée automatiquement

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquementUne image contenant texte, logiciel, Logiciel multimédia, Icône d’ordinateur

Description générée automatiquementUne image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

Une image contenant texte, capture d’écran, logiciel, Icône d’ordinateur

Description générée automatiquement

**7. Intégration de SonarQube**

* **Configuration et bénéfices pour la qualité du code :**

SonarQube est intégré pour assurer la qualité du code. La configuration de SonarQube permet de détecter et de résoudre les problèmes de code, garantissant ainsi une base de code propre et maintenable.

Une image contenant texte, logiciel, Icône d’ordinateur, Page web

Description générée automatiquement

Une image contenant texte, logiciel, Icône d’ordinateur, Logiciel multimédia

Description générée automatiquement

Une image contenant texte, logiciel, Icône d’ordinateur, Logiciel multimédia

Description générée automatiquementUne image contenant texte, logiciel, Icône d’ordinateur, Logiciel multimédia

Description générée automatiquement

Une image contenant logiciel, texte, Logiciel multimédia, Icône d’ordinateur

Description générée automatiquement

Une image contenant texte, capture d’écran, logiciel, Icône d’ordinateur

Description générée automatiquement

Une image contenant texte, logiciel, capture d’écran, Icône d’ordinateur

Description générée automatiquement

**Pipline :**

Une image contenant texte, capture d’écran

Description générée automatiquement

Une image contenant texte, capture d’écran

Description générée automatiquement

Une image contenant texte, capture d’écran

Description générée automatiquement

Une image contenant texte, capture d’écran, Logiciel multimédia, logiciel

Description générée automatiquement

Une image contenant capture d’écran, noir, obscurité

Description générée automatiquement

**8. Conclusion**

* **Résumé des accomplissements :**

Le projet a abouti à la création d'une plateforme de réservation de billets robuste, basée sur une architecture microservices. L'utilisation de Docker, Jenkins et SonarQube a permis d'assurer une mise en œuvre efficace, une intégration et déploiement continue et une qualité de code élevée.

* **Perspectives :**

Pour l'avenir, des améliorations continues, l'ajout de nouvelles fonctionnalités et l'exploration de technologies émergentes sont envisagés. L'évolutivité de l'architecture microservices permettra une adaptation aisée aux besoins changeants de la plateforme.