T-WEB-800

EPIC ROAD TRIP

Table des matières

Contexte

C’est une application de réservation de voyage est composée de plusieurs micro-services :

- Un micro-service pour la gestion des hôtels
- Un micro-service pour la gestion des vols
- Un micro-service pour la gestion des événements
- Un micro-service pour la gestion des bars
- Un micro-service pour la gestion des utilisateurs

Ces micro-services sont orchestrés par un cluster Docker Swarm. Nous avons également mis en place un API Gateway pour permettre l'accès à ces micro-services de manière centralisée.
Pour assurer la scalabilité et la résilience de votre application, nous avons défini des seuils de surcharge pour le Docker Swarm et des tests de charge sont effectués avec JMeter pour valider la capacité de l’application à gérer des charges élevées. Les instances sont dupliquées afin d’alléger le nombre de requêtes.

L’application est également sécurisée grâce à l'utilisation de certificats SSL générés pour Nginx, qui sert de proxy inverse pour l'API Gateway.

Le service API Gateway agit comme point d'entrée dans l’application. Nous utilisons Traefik comme proxy inverse pour router les requêtes entrantes vers les micro-services appropriés.

Le micro-service Users est responsable de la gestion des utilisateurs et de leurs informations de profil. Il utilise une base de données MySQL pour stocker les données des utilisateurs et des voyages qui leur sont attribués.
Le micro-service Vols est responsable de la récupération des informations sur les vols auprès de l’API Amadeus

Le micro-service Hotels est responsable de la récupération des informations sur les hôtels auprès de l’api Google (ainsi que de !!booking)
Le micro-service Evenements est responsable de la récupération des informations sur les événements auprès de l’api Google
Le micro-service Bars est responsable de la récupération des informations sur les bars auprès de l’api Google

Chacun de ces micro-services est déployé dans un conteneur Docker et géré par Docker Swarm. Les conteneurs sont connectés à un réseau Docker Overlay attachable pour permettre la communication entre eux.

Pour la sécurité, les middlewares d'authentification par JWT sont utilisés pour protéger les routes sensibles du micro-service Users. Les utilisateurs doivent être authentifiés avec un token JWT valide pour accéder aux fonctionnalités protégées (Users & Voyages).
Enfin, nous avons mis en place un pipeline CI/CD avec GitHub Actions pour automatiser le déploiement de l'application en production.

Personas

1. Julie, la voyageuse occasionnelle : Julie est une jeune professionnelle qui adore voyager. Elle n'a pas beaucoup de temps pour planifier ses voyages, donc elle recherche une application simple et facile à utiliser pour réserver ses vols, ses hôtels et ses activités. Elle a un budget limité et elle est toujours à la recherche de bonnes offres. Elle utilise principalement son téléphone portable pour naviguer sur internet.

2. Marc, le voyageur d'affaires fréquent : Marc est un homme d'affaires qui voyage beaucoup pour son travail. Il a besoin d'une application qui lui permet de réserver rapidement ses vols et ses hôtels, ainsi que de suivre son emploi du temps de voyage. Il voyage principalement en classe affaires et a besoin de services de conciergerie pour faciliter son séjour. Il utilise principalement son ordinateur portable pour effectuer ses réservations.

3. Sophie, la mère de famille : Sophie est une mère de famille qui adore voyager avec ses enfants. Elle recherche une application qui lui permet de trouver des vacances familiales abordables, ainsi que des activités et des attractions pour enfants. Elle a besoin d'un système de réservation flexible qui lui permet de modifier

Users stories

1. En tant qu'utilisateur, je souhaite pouvoir rechercher des vols en fonction de ma destination, de mes dates de voyage et de mon budget, afin de trouver les meilleures offres de vol.
2. En tant qu'utilisateur, je souhaite pouvoir réserver un vol avec mes informations personnelles et de paiement, afin de confirmer ma réservation et de recevoir une confirmation.
3. En tant qu'utilisateur, je souhaite pouvoir rechercher des hôtels dans ma destination de voyage en fonction de mes dates de voyage et de mon budget, afin de trouver un hébergement approprié.
4. En tant qu'utilisateur, je souhaite pouvoir réserver un hôtel avec mes informations personnelles et de paiement, afin de confirmer ma réservation et de recevoir une confirmation.
5. En tant qu'utilisateur, je souhaite pouvoir rechercher des événements dans ma destination de voyage, afin de planifier des activités pendant mon voyage.

6. En tant qu'utilisateur, je souhaite pouvoir ajouter des événements à mon itinéraire de voyage, afin de planifier mon emploi du temps pendant mon voyage.
7. En tant qu'utilisateur, je souhaite pouvoir rechercher des bars et des restaurants dans ma destination de voyage, afin de trouver des endroits où manger et boire pendant mon voyage.
8. En tant qu'utilisateur, je souhaite pouvoir réserver des places dans les bars et les restaurants, afin d'assurer une place pendant les périodes de pointe.
9. En tant qu'utilisateur, je souhaite pouvoir afficher mon itinéraire de voyage complet, afin de voir tous les détails de mon voyage en un seul endroit.
10. En tant qu'utilisateur, je souhaite pouvoir modifier mon itinéraire de voyage, afin de faire des changements à mes réservations existantes.

Études de risques

Voici donc une étude de potentiel risques orchestrés par Docker Swarm :

1. Risques de sécurité : Étant donné que l'application gère les informations personnelles des utilisateurs, il est important de prendre des mesures pour éviter les violations de données et les cyberattaques. Nous avons mis en place des certificats SSL et des middlewares d'authentification pour protéger l'application, mais il est important de rester vigilant et de continuer à mettre à jour les mesures de sécurité.

2. Risques de performance : Comme nous utilisons des micro-services, il est important de s'assurer que les différents services communiquent correctement et rapidement les uns avec les autres. Des temps de réponse lents ou des problèmes de connectivité peuvent entraîner des perturbations dans l'application, et peuvent impacter négativement l'expérience utilisateur.

3. Risques de surcharge : Si le nombre d'utilisateurs de l'application augmente rapidement, cela peut entraîner des problèmes de surcharge et de temps de réponse. Nous avons mis en place des tests de charge pour assurer que l'application est capable de gérer des charges élevées, mais il est important de rester attentif aux signes de surcharge et de mettre en place des mesures pour y remédier notamment la duplication des instances afin d'alléger le nombre de requêtes.

4. Risques de compatibilité : Comme nous utilisons différents micro-services provenant de différents fournisseurs, il est possible que des problèmes de compatibilité puissent survenir. Il est important de s'assurer que les différents services fonctionnent ensemble sans heurts.

5. Risques de gestion de projet : Un projet de cette ampleur peut être complexe à gérer, surtout avec une architecture basée sur des micro-services. Il est important de veiller à ce que la communication entre les différentes équipes impliquées soit efficace.

Études de risques (Cyber sécurité)

1. Risque de violation de données utilisateur : Les informations personnelles des utilisateurs telles que leur nom, adresse et informations de paiement peuvent être stockées dans les bases de données des micro-services. Si ces données sont compromises, cela peut causer des dommages à la vie privée et à la réputation des utilisateurs. Pour minimiser ce risque, il est important d'assurer la sécurité des données avec des techniques de cryptage.

2. Risque de déni de service (DoS) : Les attaques DoS sont une menace potentielle pour notre application. Les attaques DoS sont conçues pour surcharger les serveurs et empêcher les utilisateurs légitimes d'accéder aux ressources. Pour minimiser ce risque, il est important de mettre en place des mécanismes de surveillance des attaques et de limiter le trafic entrant.

3. Risque de vulnérabilité de la plateforme : Les plates-formes utilisées pour héberger les microservices peuvent avoir des vulnérabilités connues ou inconnues. Les pirates peuvent exploiter ces vulnérabilités pour accéder aux serveurs et causer des dommages. Pour minimiser ce risque, il est important de mettre à jour régulièrement les plates-formes et de mettre en place des mécanismes de surveillance pour détecter toute activité suspecte.

4. Risque de sécurité des API : Les API utilisées pour communiquer entre les micro-services peuvent être exposées aux attaques de sécurité. Les pirates peuvent exploiter des vulnérabilités dans les API pour accéder aux données sensibles. Pour minimiser ce risque, il est important de mettre en place des mesures de sécurité telles que l'authentification et l'autorisation pour garantir que seules les parties autorisées ont accès aux API.

5. Risque de fraude de carte de crédit (Bonus) : L'application de réservation de voyage nécessite l'utilisation d'informations de carte de crédit pour effectuer des transactions. Si ces informations sont compromises, cela peut causer des pertes financières importantes aux utilisateurs. Pour minimiser ce risque, il est important de s'assurer que toutes les transactions sont sécurisées et protégées par des protocoles de sécurité tels que le cryptage et l'authentification à deux facteurs.

Études de faisabilités

1. Faisabilité technique : La faisabilité technique de votre projet repose sur la mise en place d'un cluster Docker Swarm pour orchestrer les différents micro-services de l'application et la création d'un API Gateway pour permettre l'accès centralisé à ces micro-services. La faisabilité technique de ces deux éléments a été démontrée et est largement utilisée dans l'industrie. De plus, les micro-services eux-mêmes ont été développés en utilisant des technologies robustes et éprouvées, telles que MySQL pour la gestion des utilisateurs et des voyages, et les API Google et

Amadeus pour la récupération des informations sur les hôtels, les événements, les vols et les bars. En outre, plusieurs tests ont été effectués :
Des tests de charge avec JMeter pour valider la capacité de l'application à gérer des charges élevées.
Des tests sur les apis avec JestJs pour valider la capacité fonctionner et se cordonner ensemble.

2. Faisabilité économique : -

3. Faisabilité organisationnelle : La faisabilité organisationnelle du projet repose sur la capacité du groupe à gérer le développement, le déploiement et la maintenance de l'application. Nous avons mis en place un pipeline CI/CD avec GitHub Actions pour automatiser le déploiement de l'application en production, ce qui devrait faciliter la gestion de la maintenance de l'application. Cependant, il sera important de s'assurer que les développeurs possèdent les compétences techniques nécessaires pour maintenir l'application et qu'ils sont en mesure de gérer efficacement les mises à jour.