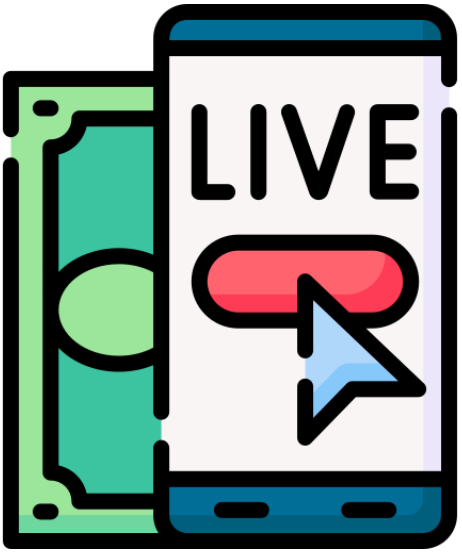
Atelier TRD - Etape 02

Infrastructure et implémentation



Cette suite d'étapes est donnée à titre indicatif pour vous guider dans la réalisation progressive de votre projet.

01 - Infrastructure Docker

L'objectif de cette étape est d'initier l'infrastructure *Docker* qui permettra à chaque service du projet de fonctionner au sein d'un conteneur *Docker* distinct (1 service = 1 conteneur).

- Pour commencer, mettez en place une infrastructure avec ***Docker*** et ***Docker Compose*** regroupant au moins 2 services :
    - un **service applicatif** avec l'environnement d'exécution adapté à la technologie de votre choix (ex: *Java, PHP, JS…*),
    - un **service de base de données** (ex: *SQL ou NoSQL*).
- Faites en sorte que le service applicatif communique avec le service de base de données au sein du **réseau** défini par votre infrastructure *Docker*.
- Pour vous aider à configurer votre infrastructure *Docker*, consultez la documentation et les exemples suivants :
    - https://docs.docker.com/compose/
    - https://github.com/docker/awesome-compose

étape 02 - page suivante

02 - Service CRUD template

L'objectif de cette étape est de disposer d'un template de service CRUD facilement réplicable et adaptable pour la suite du projet.

- Sélectionnez un service au sein de votre projet, proposant des fonctionnalités classiques, de type ***CRUD*** (*Create Read Update Delete*). A ce stade, pour des raisons pratiques, la logique et les différentes dépendances liées à ce service peuvent être simplifiées / simulées.
- Employez la stack technologique de votre choix (Langage de programmation, framework, type de persistance de données…). L'emploi d'un micro framework est recommandé afin de disposer d'une maîtrise plus fine.
- Programmez un service proposant des fonctionnalités de type ***CRUD*** sur une collection de données (ex: commandes).
- Architecturez votre service avec une approche adaptée au périmètre fonctionnel (découpage fin façon ***MVC / Architecture Hexagonale*** ou association basique routeur / contrôleur).
- Associez votre service à un port disponible sur votre machine hôte (ex: 3000).
- Exposez une ***API REST*** permettant à un client externe d'exécuter les actions proposées par le service en effectuant des requêtes *HTTP*.

  Respectez les normes définies par le style d'Architecture d'***API REST*** (https://fr.wikipedia.org/wiki/Representational_state_transfer) :

  - *URI* normés,
  - Emploi des méthodes *HTTP* pour définir le type d'action sur la collection ou la ressource identifiée par l'*URI*,
  - Emploi des headers *HTTP* pour renseigner les données méta (ex: identifiant, mot de passe, token, format de données attendues en retour, origine de la requête…),
  - Codes de statut *HTTP* pour préciser le type de résultat généré par le système (positif / négatif),
  - Données structurées au format *JSON*,

    - Les données retournées sont des représentations qui peuvent différer des données internes stockées en base de données,
    - Système sans état (stateless)...
- Testez votre *API REST* avec un outil tel que *Bruno* (https://www.usebruno.com/) ou *Postman* (https://www.postman.com/).

étape 03 - page suivante

03 - Service Gateway

L'objectif de cette étape est de masquer les services du système à l'extérieur du réseau, tout en les rendant accessibles par l'intermédiaire d'un service Gateway.

- Mettez en place un nouveau service Gateway dans votre infrastructure Docker avec la technologie de votre choix (identique ou non à celle employée précédemment).
- L'objectif du service Gateway est d'exposer une API REST en façade du système, qui fera le lien avec les services internes. Ainsi, l'API de chaque service, à l'exception de celle de la Gateway, ne sera plus directement exposée au client (situé à l'extérieur du réseau). La Gateway est un orchestrateur qui réceptionne les requêtes HTTP des clients externes et fait le relais avec le service interne visé, en réalisant une nouvelle requête HTTP auprès de son API puis en retournant le résultat obtenu au client d'origine. Le service Gateway n'est pas un simple Reverse Proxy, mais bien un client intermédiaire.
- La Gateway associe des routes exposées par sa propre API, aux routes exposées par l'API du 1er service CRUD (ex: la route Get /commandes de l'API du Gateway est associée à la route Get /commandes du 1er service CRUD).
- La Gateway récupère les valeurs utiles en provenance de la requête HTTP d'origine (ex: URI, headers et body, Token…) afin de les ré-utiliser pour effectuer la ou les requêtes intermédiaires au sein du réseau interne.
- N'exposez plus votre service 1er service CRUD au réseau externe. Exposez uniquement votre service Gateway en associant un port disponible sur votre machine hôte à celui de l'API Gateway.
- Testez les différentes routes de votre 1er service CRUD via le Service Gateway.

étape 4 - page suivante

04 - Authentification

L'objectif de cette étape est de disposer d'un service d'authentification.

- Programmez un service permettant d'effectuer les opérations suivantes :
    - **Sign Up** : création de compte,
    - **Sign In** : connexion (Basic Auth), création d'un access token et d'un refresh token,
    - **Sign Out** : déconnexion, suppression du refresh token conservé en base de données, afin de bloquer le renouvellement automatique des connexions.
    - **Verify** : vérification de l'access token (Bearer Auth),
    - **Refresh Token** : création d'un nouveau token JWT à partir du refresh token fourni.
    - **Me** : obtention des données à l'utilisateur courant en fonction de son id intégré au payload du token JWT fourni.
- Associez au service d'authentification un service de base de données afin de stocker les comptes utilisateurs et les refresh tokens associés.

étape 05 - page suivante

05 - Sécurisation des routes

L'objectif de cette étape est de sécuriser l'accès aux routes de l'API exposée par le service CRUD en employant le service d'authentification.

- Chaque route privée, tous services confondus, doit vérifier au moyen de sa clé publique que le token JWT est valide. Le token JWT doit être accessible dans le header HTTP authorization en mode bearer. Si le token est invalide, le service concerné retourne une réponse HTTP avec le code de statut 401 (*Unauthorized)* et le service Gateway retourne cette réponse au client.

étape 06 - page suivante

06 - Message Broker

L'objectif de cette étape est de disposer d'un Message Broker tel que RabbitMQ afin de réceptionner et diffuser des messages, entre services, sur un modèle événementiel.

- Mettez en place un 2ème service de type CRUD en vous inspirant du 1er service CRUD.
- L'objectif est de disposer de 2 services partageant des données et interagissant indirectement par l'intermédiaire du Message Broker : une action effectuée sur le 1er service CRUD, doit faire réagir le 2ème service CRUD (ex: Service Commande + Service Livraison).
    - Le 1er service CRUD est diffuseur de messages (publisher),
    - Le 2ème service est consommateur de messages (subscriber).
    - Une action réalisée sur le 1er service (ex: passage de commande) doit être suivie de l'envoi d'un message (contenant les données utiles au format JSON) auprès du Message Broker.
    - Le 1er service stocke les données liées à l'action.
    - La réception du message par le Message Broker doit déclencher un événement qui permettra au 2ème service CRUD de réagir.
    - Le 2ème service, abonné à la Message Queue qui vient de recevoir le message, réagit.
    - Le 2ème service consomme le message et utilise les données utiles contenues au format JSON.
    - Il est recommandé de répliquer les données contenues dans le message fourni par le Message Broker, dans le système de persistance du 2ème service (pour un traitement ultérieur, en cas de panne ou de surcharge ponctuelle).
- Ajoutez un service basé sur RabbitMQ.
- Configurez des queues de messages correspondant aux événements du Domaine (ex: order_placed, order_ready, order_delivered…).
- Connectez les 2 services au Message Broker, et faites les communiquer en fonction de leur rôle (émission / réception).

- Vérifiez que le scénario se déroule comme prévu en effectuant une action sur le 1er service (via le Gateway) et en vérifiant que les données ont bien été répliquées dans le 2ème service (via le Message Broker).