

Rapport Projet

Développement, données, exploitation

2023 / 2024

I-Boardgamemanager

II-Gestion Audience

Rapport rédigé par :

CAMARA Mamadou cire

BARRY Saikou yaya

Tables des matières

I Boardgamemanager

3.1 Phase d'analyse2-3-4
3.2 Pour chaque conteneur prévu pour votre déploiement, décrivez.....	4
3.3 Phase technique4-5
4 Schéma visuel5

II Choix libre : Gestion Audience

5 Explication projet choix6-7
6 Explication déploiement7-8
7 Cas d'utilisation8

3.1 - Phase d'analyse:

1. **Combien de conteneurs sont nécessaires pour le déploiement du logiciel? À quoi va servir chaque conteneur dans ce déploiement?**

Pour le déploiement complet de l'application, nous prévoyons d'utiliser trois conteneurs distincts afin de répartir les responsabilités de manière efficace.

Chacun des conteneurs jouera un rôle spécifique :

- a. Conteneur Frontend Angular : Ce conteneur sera dédié au lancement de l'interface utilisateur frontend développée en Angular.
- b. Conteneur Backend Spring Boot : Ce conteneur sera dédié au backend de l'application, développé en utilisant Spring Boot.
- c. Conteneur Base de Données : Ce conteneur sera dédié à la base de données nécessaire au fonctionnement de l'application.

L'utilisation d'un réseau virtuel facilitera la communication entre ces conteneurs, comme décrit dans les travaux pratiques.

2. **Devez-vous construire des images de conteneurs pour ce déploiement?**

Oui, pour ce déploiement, la création de deux images de conteneurs est prévue. Une image sera spécifiquement dédiée au frontend, tandis que l'autre sera destinée au backend. Cela permettra une gestion plus efficace des dépendances et des configurations propres à chaque partie de l'application.

Frontend :

- A. Nous avons utilisé l'image de base `node:21-alpine` et `nginx:latest`. L'image `node` nous permet d'installer les dépendances nécessaires pour le fonctionnement de l'application web, tandis que `nginx` est utilisé pour héberger l'application, permettant ainsi une consultation directe dans un navigateur web.
- B. Après l'installation des dépendances requises, nous construisons l'application à l'aide de la commande `npm run build`. Ensuite, nous copions les éléments de l'application dans le conteneur, et après la construction, nous déplaçons le répertoire `target` dans le dossier de base de `nginx`.
- C. Pour le frontend, nous avons ajouté dans l'image les fichiers nécessaires pour le lancement de l'application ainsi que les fichiers de configuration pour `nginx`, nommés `default.config`.

- D. Par exemple, il est possible de construire l'application en local et simplement copier les fichiers dans l'image.
- E. Nous n'avons pas utilisé de point d'entrée explicite pour le frontend, car il est directement hébergé dans nginx.
- F. Pour utiliser l'image correctement, il est important de s'assurer qu'elle tourne sur le même réseau que la partie backend, ainsi que le conteneur backend. De plus, l'utilisation d'une variable d'environnement pour transmettre le nom du conteneur backend est nécessaire dans ce cas de figure le nom du conteneur backend doit être <back>. Le frontend écoute sur le port 80 et communique avec le backend via le port 8080. Pour accéder à l'application en dehors du conteneur, une publication de port est nécessaire, par exemple, `docker run -p <votre_port>:80`.

Backend :

- A. Nous avons utilisé l'image de base `maven:3-eclipse-temurin-17` pour le backend, étant réalisé en Java avec Maven. Nous avons opté pour cette image afin de simplifier l'installation des dépendances et la construction de l'application.
- B. Ce script comprend trois étapes principales : installation des dépendances avec Maven (`package`), construction du projet avec Maven en mode production (`maven -Pprod`). Cela génère un répertoire `target`, dans lequel le fichier de configuration `application.yml` est copié, suivi du lancement de l'application.
- C. Au début du `containerfile`, nous avons effectué une copie des éléments nécessaires pour la construction de l'application intégralement dans le conteneur .

- D. Par exemple, il serait possible de construire localement l'application puis simplement copier les fichiers dans l'image.
- E. Le point d'entrée de l'image permet d'exécuter l'application qui a été construite après l'option `-Pprod`.
- F. Pour bien utiliser cette image, il est essentiel de s'assurer qu'elle fonctionne sur le même réseau que le conteneur PostgreSQL. De plus, l'utilisation de variables d'environnement est nécessaire pour fournir les informations relatives à PostgreSQL, telles que le nom du conteneur, la base de données, le nom d'utilisateur et le mot de passe. Par exemple : `–env db_host=database –env db_name=camara –env user=Camara –env password=CamaraPwds`.

3.2 Pour chaque conteneur prévu pour votre déploiement, décrivez :

Pour le déploiement intégral de l'application, trois conteneurs seront utilisés, tous connectés au même réseau. Dans le cadre de mes tests, j'ai créé un réseau nommé "gameNetwork", que les trois conteneurs utilisent pour assurer une communication fluide entre eux.

- Postgres :
- Backend :
 - le container doit utiliser le même réseaux virtuelle que la base de donnée enfin de pouvoir communiquer
 - Aucun montage de répertoire n'est requis, et aucun port n'a besoin d'être exposé. Les informations de connexion à la base de données seront configurées via des variables d'environnement, comme suit
`–env db_host=database –env db_name=camara –env user=Camara –env password=CamaraPwds`
- Frontend :
 - Il utilisera le même réseau virtuel que le back pour la communication avec les autres composants.
 - Aucun montage de répertoire n'est nécessaire. Cependant, pour consulter l'application web en dehors du conteneur, une publication de port est requise, par exemple : `–publish 3000:80`.

3.3 Phase technique :

Pour assurer une exécution optimale de ces conteneurs, quatre étapes doivent être suivies scrupuleusement :

- a) Créer un réseau virtuel :

```
podman network create gameNetwork
```

- b) Lancement de la base de données PostgreSQL avec un montage de répertoires pour la récupération des données lors des prochains lancements, et ce, sur le réseau virtuel gameNetwork :

```
$podman container run - -name database
- -detach
- -network gameNetwork
- -env POSTGRES_PASSWORD=CamaraPwds
- -env POSTGRES_USER=Camara
- -env POSTGRES_DB=camara
- -volume/comptes/E224740H/postgres/
db-data:/var/lib/postgresql/data docker.io/postgres
```

- c) Lancement du backend sur le même réseau virtuel avec les informations de la base de données. L'image du backend est déjà présente sur le serveur GitLab de l'Université de Nantes. Dans cet exemple, les variables d'environnement sont utilisées pour saisir les données de configuration de PostgreSQL :

```
$podman container run --rm --name back
--env db_host=database
--env db_name=camara
--env user=Camara
--env password=CamaraPwds
--network gameNetwork
docker-registry.univ-nantes.fr/e224740h/
boardgamemanager/back:v1
```

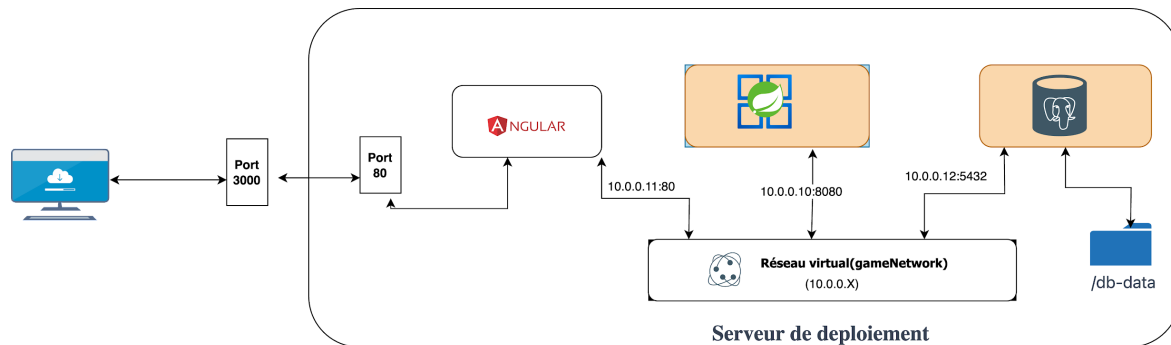
NB : Il est crucial de nommer ce conteneur "back" pour que la prochaine étape fonctionne.

- d) Dernière étape consistant à lancer le front, de la même manière que les autres conteneurs, tous devant être sur le même réseau :

```
$podman container run --rm --name game
--env back_name=back
--network gameNetwork
--publish 8220:80
--docker-registry.univ-nantes.fr/e224740h/
```

```
--network gameNetwork  
docker-registry.univ-nantes.fr/e224740h/boardgamemanager/game:v2
```

4 Schéma visuel



Choix libre : Gestion Audience

5 Explication projet choix

Un procès doit se tenir sur 4 audiences où l'on souhaite appeler différents témoins à la barre. 8 témoins ont été identifiés et certains ont notifié qu'ils seraient absents à certaines audiences. L'objectif est de produire un (mini) site web permettant de convoquer à chaque audience différents témoins choisis parmi les convocables,

i.e. parmi les non-absents. Comme illustré en figure 1, chaque convocation partitionne les témoins en trois listes disjointes : les absents, les convoqués et les exempts (i.e. ni absents, ni convoqués). Une convocation est pré-enregistrée en base de données pour chaque audience et celle de la première audience s'affiche par défaut (figure 1). Le visiteur peut choisir un autre numéro d'audience (bouton central) et en extraire la convocation stockée en base de données en cliquant sur le bouton EXTRAIRE : les figures 2, 3 et 4 illustrent les convocations pré-enregistrées pour les 3 autres audiences. Le visiteur peut ensuite modifier chaque convocation en convoquant de nouveaux témoins (les figures 5 et 6 illustrent l'ajout de CALIGULA), ou en exemptant (les figures 7 et 8 illustrent le retrait de CLAUDE puis HADRIEN)

EXTRAIRE 1 METTRE A JOUR		
CONVOQUES	EXEMPTS	ABSENTS
AUGUSTE	CALIGULA	NERON
CESAR		TIBERE
CLAUDE		
HADRIEN		
TRAJAN		












figure 1

EXTRAIRE 2 METTRE A JOUR		
CONVOQUES	EXEMPTS	ABSENTS
HADRIEN	AUGUSTE	CESAR
NERON	CALIGULA	CLAUDE
TIBERE		
TRAJAN		






figure 2

Bien sûr, voici une version corrigée et améliorée de votre texte :

6. Explication du déploiement

Pour déployer ce logiciel, nous avons utilisé deux conteneurs et établi une connexion entre eux en utilisant un réseau virtuel.

Pour la partie front-end, nous avons choisi l'image de base php:7.4-apache, qui intègre déjà Nginx, facilitant ainsi son utilisation et permettant des tests plus aisés des résultats. Nous avons également utilisé des variables d'environnement pour la configuration de la base de données, telles que :

- db_host=dataaudience (correspondant au nom du conteneur MySQL),
- db_user=sabarry (identifiant de la base de données),
- db_pass=saikou1993 (mot de passe de la base de données).

Le deuxième conteneur est dédié au lancement de la base de données MySQL. Pour garantir son bon fonctionnement, il est essentiel de s'assurer qu'il opère sur le même réseau que le conteneur front-end. De plus, l'utilisation de variables d'environnement est indispensable pour fournir les informations nécessaires à MySQL, telles que le nom du conteneur, le nom de la base de données, le nom d'utilisateur et le mot de passe. Par exemple :

- db_host=dataaudience,
- db_name=dataaudience,

- user=sabarry,
- password=saikou1993.

NB : Il est important de monter un répertoire afin que la base de données soit opérationnelle. Vous trouverez à la racine du projet "Audience" un fichier nommé "audience.sql", qui contient les tables nécessaires au bon fonctionnement de ce logiciel. Assurez-vous d'exécuter ce script SQL pour créer les tables requises avant de démarrer l'application.

7 Cas d'utilisation

- a) `podman run --name dataaudience`
 `--network gameNetwork`
 `--env MYSQL_ROOT_PASSWORD=saikou1993`
 `--env MYSQL_DATABASE=audience --env`
 `MYSQL_USER=sabarry`
 `--env MYSQL_PASSWORD=saikou1993`
 `--volume/comptes/E224740H/audience/Audiencede/audience.sql:/doc`
 `ker-entrypoint-initdb.d/init.sql`
 `docker.io/mysql:latest`
- b)
- `$podman container run --name audience`
 `--publish 8220:80`
 `--network gameNetwork`
 `--env db_host=dataaudience`
 `--env db_user=sabarry`
 `--env db_pass=saikou1993`
 `docker-registry.univ-nantes.fr/e224740h/audience/ audience:v1`