Dossier Technique Informatique 4
CyCruise

Antonin PERALTA

Alexandre Halhoute

Sujet ClickJourneY

PréIng-2 Math-Info options Physiques

MI-1 Tri-J

Année 2024/2025

Sommaire

Comment utiliser l’application

a) Démarrage avec docker

Le moyen principal pour utiliser l’application CyCruise est via Docker et Docker Compose. Pour cela, il suffit de cloner le repository (git clone https://github.com/topeestla/cycruise). Une fois cette étape réalisée, lancez le script bash “run-cycruise-app.sh”. Ce script va créer l’image Docker du projet et la sauvegarder localement. Après la création de l’image, il lance Docker Compose (dev-compose.yml). Docker Compose démarre deux services : le premier utilise l’image Docker de notre projet (qui vient donc d’être créée) et le deuxième correspond à un serveur MariaDB. Dès que vous voyez apparaître la ligne (✔ **Container cycruise_php Started**), vous pouvez accéder au service via le port 4000 (localhost:4000). Si besoin, le service MariaDB est exposé sur le port 3838.

Pour arrêter le service, un script “stop-cycruise-app.sh” est disponible, et il arrête à la fois le service CyCruise et MariaDB. Pour supprimer complètement l’application de votre machine, le script “clear-cycruise-app.sh” est également disponible : il permet de supprimer l’image et le volume Docker.

b) Démarrage sans docker (non recommandé)

Même si Docker est recommandé pour utiliser l'application, il est tout de même possible de l'utiliser sans Docker. Pour cela, assurez-vous d'abord de disposer de tous les prérequis pour l'application (PHP 8, mysqli, pdo_mysql et un serveur MariaDB). Une fois ces prérequis remplis, créez une base de données nommée “cycruise” sur le serveur MariaDB. Ensuite, indiquez à l'application les informations de connexion au serveur MariaDB en vous rendant dans le fichier **src/Models/Database.php** à la ligne 12, ainsi que dans le fichier **src/Service/InvoiceService.php** pour remplacer `getenv("APP_URL")` à la ligne 22 par `”localhost:4000”`. Une fois ces modifications effectuées, vous pouvez lancer votre serveur PHP de test via la commande suivante (à la racine de votre dépôt) : “php -S localhost:4000 -t public/”. Et voilà, l'application CyCruise est lancée !

Système de données

a) Choix technique

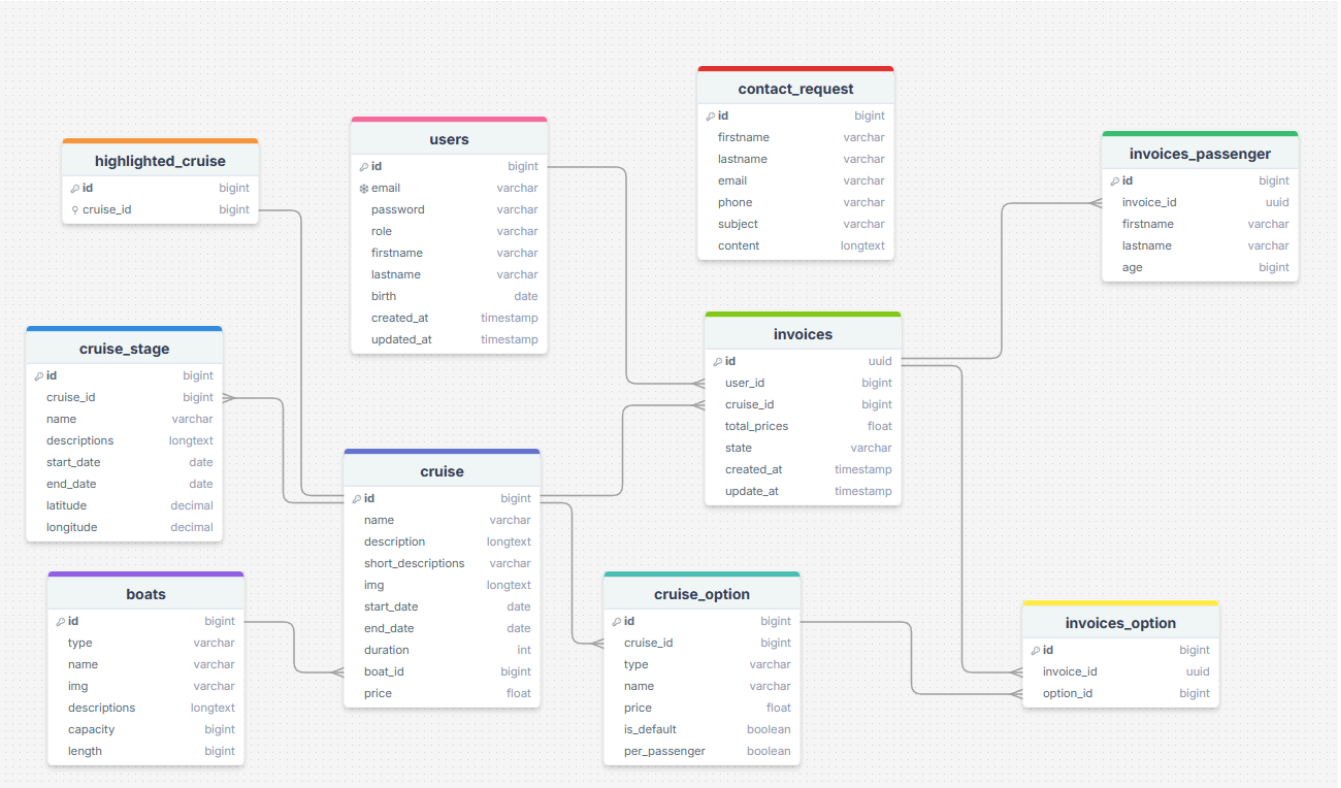
Pour notre projet, nous avons choisi d'utiliser PHP en version 8. Cela nous permet de bénéficier du typage des variables, ce qui rend le code beaucoup plus lisible, notamment sur la partie "Class".

Concernant nos données, nous avons opté pour une base de données MariaDB (dont nous expliquerons le schéma un peu plus tard).

Notre projet est structuré autour de trois principaux dossiers : **"src"**, **"public"** et **"component"**.

- **src** : Contient tout le système lié au serveur (Backend), notamment la gestion de la base de données, les objets et les services d'authentification.
- **public** : Comme son nom l'indique, ce dossier regroupe toute la partie PHP correspondant aux pages du site web accessibles au public.
- **component** : Ce dossier contient les éléments réutilisables dans les pages publiques, qui ne doivent pas être affichés séparément.

b) Explication du schéma Sql



(image aussi disponible ici lien alternative ici)

Notre schéma SQL repose sur trois grandes tables centrales : **“cruise”**, **“user”** et **“invoices”**. Ces tables représentent les trois fonctionnalités principales de notre site.

1. La table “user”

Cette table contient toutes les informations importantes des utilisateurs, notamment l’**email**, le **mot de passe**, ainsi que d’autres informations personnelles telles que **nom**, **prénom** et **date de naissance**.

Elle possède un champ **“id”**, de type **INT**, qui est généré automatiquement par la base de données et **auto-incrémenté** au fil du temps. Chaque utilisateur dispose ainsi d’un identifiant unique qui s’incrémente de 1 en 1 (exemple : si le dernier utilisateur enregistré a l’ID **300**, le prochain aura l’ID **301**).

Les champs **“created_at”** et **“updated_at”** sont automatiquement gérés par la base de données :

- **“created_at”** enregistre automatiquement la date d’insertion.
- **“updated_at”** est mis à jour automatiquement à chaque modification.

2. La table “cruise”

Comme la table **“user”**, elle possède un **ID auto-incrémenté** ainsi que ses informations principales.

Cependant, pour structurer une croisière, nous avons dû créer **deux autres tables** liées à celle-ci via une relation **"one-to-many"** :

- Une table pour les **options passager** (cabine, etc.).
- Une table pour les **étapes de la croisière**.

De plus, la table **“cruise”** fait référence à une autre table, **“bateaux”**, afin d’afficher les informations du bateau utilisé pour la croisière.

3. La table “invoices”

Cette table fait le lien entre un **utilisateur** et une **croisière**. Elle représente donc les **réservations** effectuées par les utilisateurs. Une table supplémentaire est également utilisée pour **enregistrer les options choisies par l’utilisateur** lors de la réservation.

4. Autres tables

Certaines autres tables, comme **“highlighted_cruise”** (croisières mises en avant) et **“contact_request”** (demandes de contact), sont plus anecdotiques et utilisées pour des fonctionnalités secondaires.

Les croisière

a) Les croisière par défaut

Dans sa forme de base, le projet ajoute automatiquement 25 croisières d'exemple. Celles-ci sont stockées dans le fichier "cruises_sql.sql". Si vous ne souhaitez pas les inclure, il vous suffit simplement de supprimer les lignes de ce fichier. Informations importantes à noter : ces croisières sont uniquement des exemples et ont été générées aléatoirement via ce script. Il est donc possible que certaines croisières n'aient pas de sens, notamment au niveau des étapes, ainsi qu'au niveau des prix des options proposées à l'utilisateur.

b) Crée une croisière via le site

Si vous souhaitez créer une croisière la manière la plus simple est via le panel administrateur du site "admin/create-cruise.php"

c) Crée une croisière par la base de donnée

Si vous souhaitez créer une croisière mais directement à la main via les requête sql, dans ce cas il est conseillé de s'inspirer du fichier "cruise_sql.sql" mais aussi du fichier CruiseRepository situé dans src/repository