Projet MSPR — Voyage
“Projet de groupe : Préparation et présentation d’une application de voyage”

l’école d’ingénierie informatique

l’école [tech]
de l’expertise digitale

Participants :

Tom CLEMENT
William COLLE
Clément SALINGUE

Projet encadré par :

GABER Khalid

Préparation et présentation d’une application de voyage - Première Année, 2025

EPSI — Ecole de l’ingénierie informatique, Arras

Table des matières

1. Context.....03

2. Question.....03

1.1. MCD.....03

1.2. SQL.....04

3. Explication des spécifications techniques.....09

2.1. Les Modèles (src/Model/).....11

2.2. Les Contrôleurs (src/Controller/).....15

2.3. Le Routeur (src/Routing/).....23

2.4. Les Templates (templates/).....24

4. Annexe.....26

3.1.DICTIONNAIRE DE DONNÉE.....26

3.1.dépendance fonctionnelle.....28

3.2. MLD.....28

3.2. Conclusion.....29

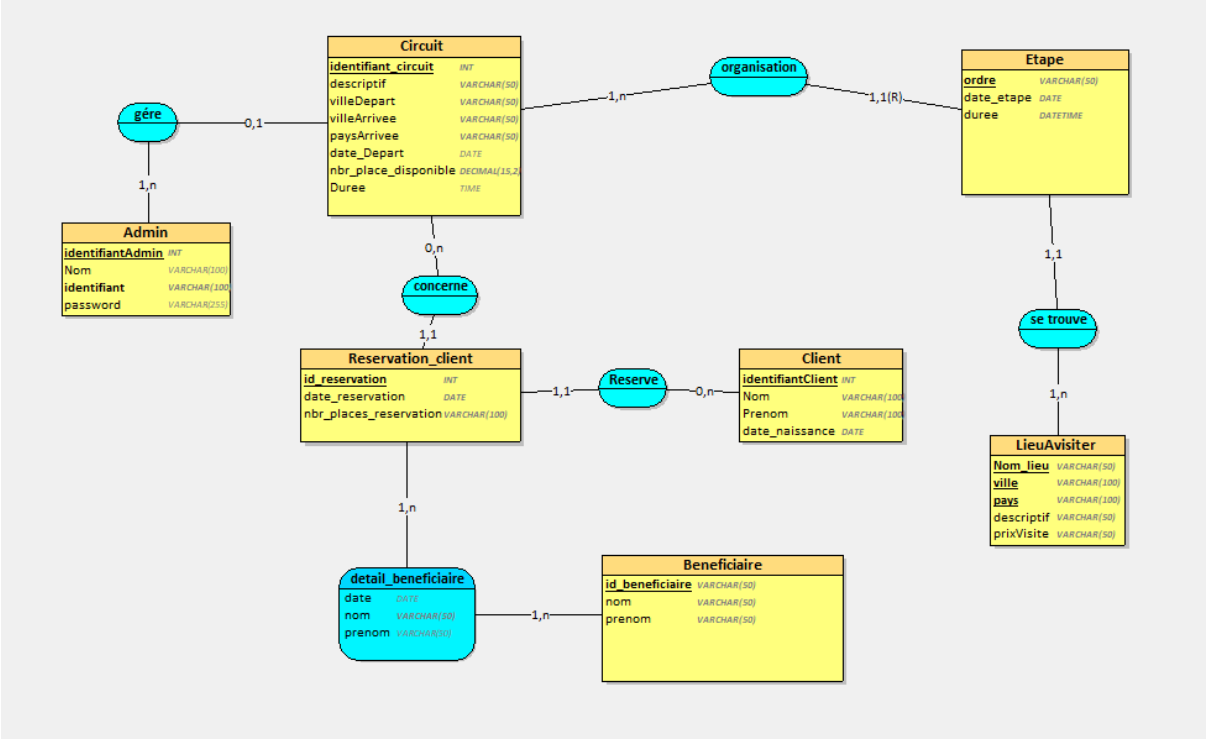
Context :

En premier lieu nous devons réaliser une base de données en MCD grâce au logiciel Looping puis un dictionnaire de données et les relations fonctionnelles.

En deuxième lieu on doit répondre à des questions de requête sql.

On souhaite développer une application en utilisant PHP qui permet aux clients de s'enregistrer puis de se connecter ce qui va permettre de réserver leurs circuits. En parallèle développer une partie administrateur avec une authentification dont seul lui peut avoir les droits pour ajouter, supprimer, modifier un circuit, une étape, ect.

1) MCD



On a créé ici nos 7 tables pour notre planning de voyages qui regroupe la table Admin qui est liée à la table circuit.

Une table circuit qui gère le circuit de voyage, cette table est reliée à la réservations client qui va réserver un circuit, et il est aussi lié à la table étape puisque l'étape du voyage appartient à un ou plusieurs circuits.

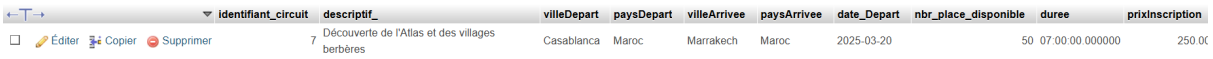
On a ensuite la table Étape qui est relié la table circuite et la table LieuAvisiter puisque chaque étape possède un ou plusieurs lieu à visiter

On ensuite justement cette table Lieu visiter lié à la table Étape

Ensuite nous avons la tables Reservations Client qui est lié à la table CIrcuit, la table CLient qui renseigne les infos personnels du client et la table bénéficiaires si un client offre un circuit de voyage

Remplissage des tables via les requêtes SQL :

2) Insérer les valeurs dans les tables

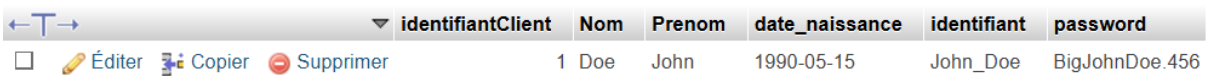


	identifiant_circuit	descriptif_	villeDepart	paysDepart	villeArrivee	paysArrivee	date_Depart	nbr_place_disponible	duree	prixInscription
Éditer Copier Supprimer	7	Découverte de l'Atlas et des villages berbères	Casablanca	Maroc	Marrakech	Maroc	2025-03-20	50	07:00:00.000000	250.0

INSERT INTO Circuit_ (identifiant_circuit, descriptif_, villeDepart, paysDepart, villeArrivee, paysArrivee, date_Depart, nbr_place_disponible, Duree, prixInscription)
VALUES (7, 'Découverte de l\'Atlas et des villages berbères', 'Casablanca','Maroc', 'Marrakech', 'Maroc', '2025-03-20', 50, '07:00:00', 250);

	Nom_lieu	ville	pays	descriptif_	prixVisite
Éditer Copier Supprimer	Jardin Majorelle	Marrakech	Maroc	Un jardin botanique unique	100.00

INSERT INTO LieuAvisiter_ (Nom_lieu, ville, pays, descriptif_, prixVisite)
VALUES ('Jardin Majorelle', 'Marrakech', 'Maroc', 'Un jardin botanique unique', 100);



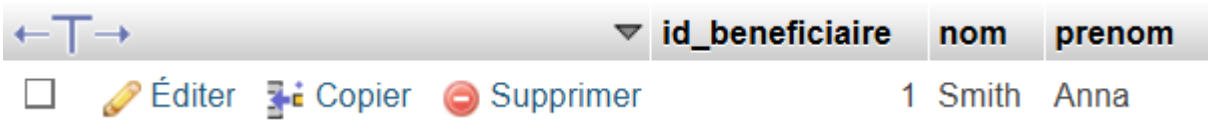
	identifiantClient	Nom	Prenom	date_naissance	identifiant	password
Éditer Copier Supprimer	1	Doe	John	1990-05-15	John_Doe	BigJohnDoe.456

INSERT INTO Client (identifiantClient, Nom, Prenom, date_naissance, identifiant, password)
VALUES (1, 'Doe', 'John', '1990-05-15', 'John_Doe', 'BigJohnDoe.456');

---

	id_reservation	date_reservation	nbr_places_reservation	identifiant_circuit	identifiantClient
Éditer Copier Supprimer	1	2025-03-01	2	7	1

INSERT INTO Reservation_client (id_reservation, date_reservation, nbr_places_reservation, identifiant_circuit, identifiantClient)
VALUES (1, '2025-03-01', '2', 7, 1);



INSERT INTO Beneficiaire (id_beneficiaire, nom, prenom)
VALUES (1, 'Smith', 'Anna');

---

	identifiant_circuit	ordre	date_etape_	duree	Nom_lieu	ville	pays
Éditer Copier Supprimer	7	1	2025-03-20	02:00:00.000000	Jardin Majorelle	Marrakech	Maroc

INSERT INTO Etape (identifiant_circuit, ordre, date_etape_, duree, Nom_lieu, ville, pays)
VALUES (7, '1', '2025-03-20', '02:00:00', 'Jardin Majorelle', 'Marrakech', 'Maroc');

Requetes SQL

3)

identifiant_circuit	descriptif_	villeDepart	villeArrivee	nombre_etapes
7	Découverte de l'Atlas et des villages berbères	Casablanca	Marrakech	1

```
SELECT
    Circuit_.identifiant_circuit,
    Circuit_.descriptif_,
    Circuit_.villeDepart,
    Circuit_.villeArrivee,
    COUNT(Etape.ordre) AS nombre_etapes
FROM
    Circuit_
JOIN
    Etape ON Circuit_.identifiant_circuit = Etape.identifiant_circuit
WHERE
    Circuit_.identifiant_circuit = 7
GROUP BY
    Circuit_.identifiant_circuit, Circuit_.descriptif_, Circuit_.villeDepart, Circuit_.villeArrivee;
```

---

4)


```
DELETE FROM LieuAvisiter_
WHERE Nom_lieu = 'Jardin Majorelle'
AND ville = 'Marrakech'
AND pays = 'Maroc'
AND (Nom_lieu, ville, pays) NOT IN
```

---

5)

prix_total
475.00

```
SELECT

    Circuit_.prixInscription + SUM(CAST(LieuAvisiter_.prixVisite AS DECIMAL(15,2))) AS
prix_total

FROM

    Circuit_

JOIN

    Etape ON Circuit_.identifiant_circuit = Etape.identifiant_circuit

JOIN

    LieuAvisiter_ ON Etape.Nom_lieu = LieuAvisiter_.Nom_lieu

    AND Etape.ville = LieuAvisiter_.ville

    AND Etape.pays = LieuAvisiter_.pays

WHERE

    Circuit_.identifiant_circuit = 7

GROUP BY

     Circuit_.prixInscription;
```

---

6)

identifiant_circuit	prix_total	date_Depart	Duree
7	5100.00	2025-03-20	07:00:00.000000

```
SELECT
    Circuit_.identifiant_circuit,
    Circuit_.nbr_place_disponible + SUM(CASE
        WHEN LieuAvisiter_.prixVisite IS NOT NULL THEN CAST(LieuAvisiter_.prixVisite AS
DECIMAL(15,2))
        ELSE 0
    END) AS prix_total,
    Circuit_.date_Depart,
    Circuit_.Duree
FROM
    Circuit_
JOIN
    Etape ON Circuit_.identifiant_circuit = Etape.identifiant_circuit
JOIN
    LieuAvisiter_ ON Etape.Nom_lieu = LieuAvisiter_.Nom_lieu
    AND Etape.ville = LieuAvisiter_.ville
    AND Etape.pays = LieuAvisiter_.pays
WHERE
    Circuit_.date_Depart BETWEEN '2025-02-01' AND '2025-06-15'  -- Plage de dates de
vacances
AND
    Circuit_.nbr_place_disponible >= 10  -- Nombre de places disponibles;
```

---

7)

Avant la requete

	identifiant_circuit	ordre	date_etape_	duree	Nom_lieu	ville	pays
☐ Éditer Copier Supprimer	7	1	2025-03-20	02:00:00.000000	Jardin Majorelle	Marrakech	Maroc
☐ Éditer Copier Supprimer	7	2	2025-03-21	01:30:00.000000	Montagne d'atlas	Marrakech	Maroc
☐ Éditer Copier Supprimer	7	3	2025-03-23	04:30:00.000000	Chefchaouen	Marrakech	Maroc

Après la requête

	identifiant_circuit	ordre	date_etape_	duree	Nom_lieu	ville	pays
Éditer Copier Supprimer	7	1	2025-03-20	02:00:00.000000	Jardin Majorelle	Marrakech	Maroc
Éditer Copier Supprimer	7	2	2025-03-23	04:30:00.000000	Chefchaouen	Marrakech	Maroc

Procédure:

```
DELIMITER $$
CREATE PROCEDURE SupprimerEtape(
    IN circuit_id INT,
    IN ordre_etape INT
)
BEGIN
    DELETE FROM Etape
    WHERE identifiant_circuit = circuit_id
    AND ordre = ordre_etape;
    UPDATE Etape
    SET ordre = ordre - 1
    WHERE identifiant_circuit = circuit_id
    AND ordre > ordre_etape;
END$$
DELIMITER ;
```

appel procédure:

```
 CALL SupprimerEtape(7, 2);
```

---

8)

	id_reservation	date_reservation	nbr_places_reservation	identifiant_circuit	identifiantClient
Éditer Copier Supprimer	1	2025-03-01	2	7	1
Éditer Copier Supprimer	2	2025-03-05	3	7	1

```
INSERT INTO Reservation_client (id_reservation, date_reservation, nbr_places_reservation,
identifiant_circuit, identifiantClient)
VALUES (2, '2025-03-05', '3', 7, 1);
```

Explication des spécifications techniques

Notre projet suit une architecture MVC (Modèle-Vue-Contrôleur) avec des CRUD pour chaque table coté admin .

1. Les Entités (src/Entity/)

Les entités représentent les tables de la base de données. Elles définissent la structure des données et leurs relations.

Administrateur.php

Beneficiaire.php

Circuit.php

Client.php

DetailBeneficiaire.php

```php
<?php
declare(strict_types=1);
namespace MyApp\Entity;

use MyApp\Entity\ReservationClient;
use MyApp\Entity\Beneficiaire;

class DetailBeneficiaire
{
    private ?int $idReservation;
    private ?int $idBeneficiaire;

    private ReservationClient $reservationClient;
    private Beneficiaire $beneficiaire;
```

`private ?int $idReservation;` : Stocke l'ID de la réservation associée (peut être `null`).

`private ?int $idBeneficiaire;` : Stocke l'ID du bénéficiaire associé (peut être `null`).

`private ReservationClient $reservationClient;` : Référence à un objet `ReservationClient`, qui représente la réservation liée.

`private Beneficiaire $beneficiaire;` : Référence à un objet `Beneficiaire`, qui représente le bénéficiaire associé.

```php
public function __construct(
    ?int $idReservation,
    ?int $idBeneficiaire,
    ReservationClient $reservationClient,
    Beneficiaire $beneficiaire
) {
    $this->idReservation = $idReservation;
    $this->idBeneficiaire = $idBeneficiaire;
    $this->reservationClient = $reservationClient;
    $this->beneficiaire = $beneficiaire;
}
```

Ce constructeur est utilisé pour créer une nouvelle instance de `DetailBeneficiaire`.

Il prend quatre paramètres :

1. `$idReservation` → L'ID de la réservation (peut être `null`).
2. `$idBeneficiaire` → L'ID du bénéficiaire (peut être `null`).
3. `$reservationClient` → Un objet `ReservationClient` (obligatoire).
4. `$beneficiaire` → Un objet `Beneficiaire` (obligatoire).

Il assigne les valeurs aux **propriétés privées** de la classe.

```php
public function getIdReservation(): ?int
{
    return $this->idReservation;
}

public function setIdReservation(?int $idReservation): void
{
    $this->idReservation = $idReservation;
}
```

`getIdReservation()` → Retourne l'ID de la réservation (ou `null` si non défini).

`setIdReservation()` → Définit un nouvel ID pour la réservation.

Etape.php

LieuAvisiter.php

ReservationClient.php

Explication du code dans entité :

2) Les Modèles (src/Model/)

Les modèles gèrent les interactions avec la base de données et implémentent la logique de manipulation des données.

AdministrateurModel.php

BeneficiaireModel.php

CircuitModel.php

ClientModel.php

DetailBeneficiaireModel.php

Ce fichier définit un modèle (`Model`) pour gérer les opérations liées à l'entité `DetailBeneficiaire`. Il permet d'effectuer les opérations CRUD (Create, Read, Delete) sur la table `DetailBeneficiaire`, qui fait le lien entre une réservation (`ReservationClient`) et un bénéficiaire (`Beneficiaire`).

```php
<?php
declare(strict_types=1);

namespace MyApp\Model;

use MyApp\Entity\DetailBeneficiaire;
use MyApp\Entity\ReservationClient;
use MyApp\Entity\Beneficiaire;
use PDO;
```

`use MyApp\Entity\ReservationClient;` et `use MyApp\Entity\Beneficiaire;`
→ Importent les entités associées (`ReservationClient` pour les réservations et `Beneficiaire` pour les bénéficiaires).

`use PDO;`
→ Importe PDO pour permettre l'interaction avec la base de données.

```php
    public function getAllDetailBeneficiaires(): array
    {
        $sql = "SELECT db.idReservation, db.idBeneficiaire,
        r.dateReservation, r.nbrPlacesReservation, r.identifiantCircuit, r.identifiantClient,
        b.nom, b.prenom
FROM DetailBeneficiaire db
INNER JOIN ReservationClient r ON db.idReservation = r.idReservation
INNER JOIN Beneficiaire b ON db.idBeneficiaire = b.idBeneficiaire";

        $stmt = $this->db->query($sql);
        $details = [];

        while ($row = $stmt->fetch(PDO::FETCH_ASSOC)) {
            $reservation = new ReservationClient(
                $row['idReservation'],
                $row['dateReservation'],
                $row['nbrPlacesReservation'],
                $row['identifiantCircuit'],
                $row['identifiantClient']
            );

            $beneficiaire = new Beneficiaire($row['idBeneficiaire'], $row['nom'], $row['prenom']);

            $details[] = new DetailBeneficiaire($reservation->getIdReservation(), $beneficiaire->getIdBeneficiaire());
        }

        return $details;
    }
```

Exécute une jointure SQL (`INNER JOIN`) pour récupérer les détails des bénéficiaires avec leurs réservations associées.

`$stmt = $this->db->query($sql);` → Exécute la requête SQL.

Parcours des résultats (`while`) :

- Créer un objet `ReservationClient` en utilisant les données récupérées.
- Créer un objet `Beneficiaire`.
- Ajoute un `DetailBeneficiaire` à `$details[]`.

Retourne un tableau de `DetailBeneficiaire`.

```php
public function getOneDetailBeneficiaire(int $idReservation, int $idBeneficiaire): ?DetailBeneficiaire
{
    $sql = "SELECT db.idReservation, db.idBeneficiaire,
                   r.dateReservation, r.nbrPlacesReservation,
                   b.nom, b.prenom
            FROM DetailBeneficiaire db
            INNER JOIN ReservationClient r ON db.idReservation = r.idReservation
            INNER JOIN Beneficiaire b ON db.idBeneficiaire = b.idBeneficiaire
            WHERE db.idReservation = :idReservation AND db.idBeneficiaire = :idBeneficiaire";

    $stmt = $this->db->prepare($sql);
    $stmt->bindValue(":idReservation", $idReservation, PDO::PARAM_INT);
    $stmt->bindValue(":idBeneficiaire", $idBeneficiaire, PDO::PARAM_INT);
    $stmt->execute();

    $row = $stmt->fetch(PDO::FETCH_ASSOC);
    if (!$row) {
        return null;
    }

    $reservation = new ReservationClient(
        $row['idReservation'],
        $row['dateReservation'],
        $row['nbrPlacesReservation'],
        $row['identifiantCircuit'],
        $row['identifiantClient']
    );

    $beneficiaire = new Beneficiaire($row['idBeneficiaire'], $row['nom'], $row['prenom']);

    return new DetailBeneficiaire($reservation->getIdReservation(), $beneficiaire->getIdBeneficiaire());
}
```

- Sélectionne un seul `DetailBeneficiaire` en fonction de l'ID de la réservation et de l'ID du bénéficiaire.
- Utilisation de `prepare()` et `bindValue()` → Protège contre les injections SQL.
- Si aucune donnée n'est trouvée → Retourne `null`.
- Sinon → Crée et retourne un objet `DetailBeneficiaire`.

```php
public function createDetailBeneficiaire(DetailBeneficiaire $detailBeneficiaire): bool
{
    $sql = "INSERT INTO DetailBeneficiaire (idReservation, idBeneficiaire) VALUES (:idReservation, :idBeneficiaire)";
    $stmt = $this->db->prepare($sql);
    $stmt->bindValue(':idReservation', $detailBeneficiaire->getIdReservation(), PDO::PARAM_INT);
    $stmt->bindValue(':idBeneficiaire', $detailBeneficiaire->getIdBeneficiaire(), PDO::PARAM_INT);
    return $stmt->execute();
}
```

- Insère une nouvelle relation entre une réservation et un bénéficiaire.
- Utilisation de `prepare()` et `bindValue()` pour éviter les injections SQL.
- Retourne `true` si l'ajout est réussi, sinon `false`.

```php
    public function deleteDetailBeneficiaire(int $idReservation, int $idBeneficiaire): bool
    {
        $sql = "DELETE FROM DetailBeneficiaire WHERE idReservation = :idReservation AND idBeneficiaire = :idBeneficiaire";
        $stmt = $this->db->prepare($sql);
        $stmt->bindValue(':idReservation', $idReservation, PDO::PARAM_INT);
        $stmt->bindValue(':idBeneficiaire', $idBeneficiaire, PDO::PARAM_INT);
        return $stmt->execute();
    }
}
```

Supprime un lien entre un bénéficiaire et une réservation.

Utilisation de `prepare()` pour éviter les injections SQL.

Retourne `true` si la suppression réussit, sinon `false`.

EtapeModel.php

LieuAvisiterModel.php

ReservationClientModel.php

-Contiennent des méthodes pour interagir avec la base (ex : récupérer un client, ajouter une réservation).

- Font appel aux entités pour obtenir ou sauvegarder des données.

Permettent d'effectuer des requêtes spécifiques (ex : récupérer les circuits d'un client).

3)Les Contrôleurs (src/Controller/)

Les contrôleurs gèrent la logique métier et répondent aux requêtes HTTP.

AdministrateurController.php

BeneficiaireController.php

CircuitController.php

ClientController.php

DefaultController.php

ReservationClientController.php

Explication du code dans AdministrateurController.php

```php
class AdministrateurController
{
    private $twig;
    private AdministrateurModel $administrateurModel;

    public function __construct(Environment $twig, DependencyContainer $dependencyContainer)
    {
        $this->twig = $twig;
        $this->administrateurModel = $dependencyContainer->get('AdministrateurModel');
    }


    public function listAdministrateurs()
    {
        $administrateurs = $this->administrateurModel->getAllAdministrateurs();
        echo $this->twig->render('administrateurController/listAdministrateurs.html.twig', ['administrateurs' => $administrateurs]);
    }
```

- Constructeur :

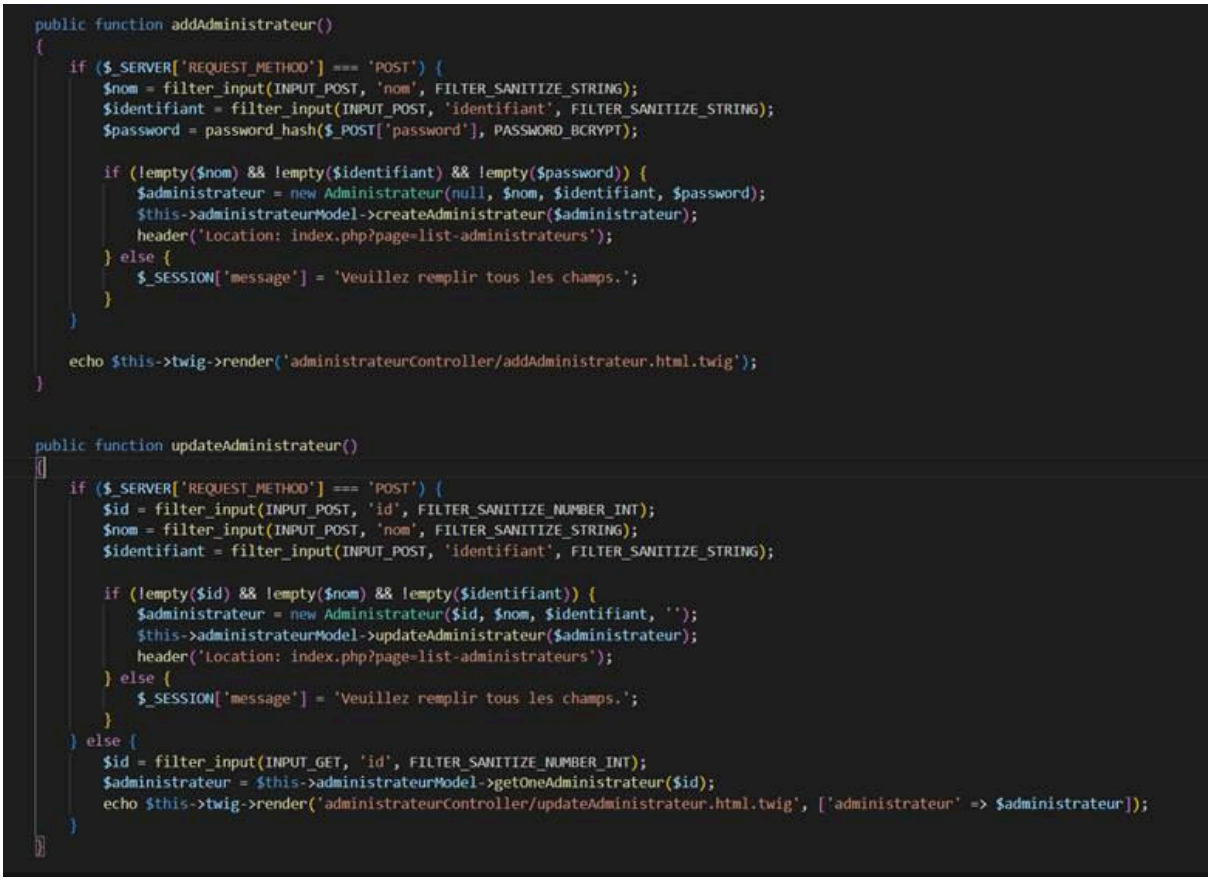
  $twig est injecté pour gérer l'affichage avec Twig.

  $dependencyContainer->get('AdministrateurModel') récupère l'instance de AdministrateurModel depuis le conteneur de dépendances.

Fonction listadministrateur

- Récupère tous les administrateurs via $this->administrateurModel->getAllAdministrateurs().
- Affiche la vue Twig listAdministrateurs.html.twig en lui passant les données.

Exécution :
Si un utilisateur accède à index.php?page=list-administrateurs, il verra la liste des administrateurs.



```php
public function addAdministrateur()
{
    if ($_SERVER['REQUEST_METHOD'] === 'POST') {
        $nom = filter_input(INPUT_POST, 'nom', FILTER_SANITIZE_STRING);
        $identifiant = filter_input(INPUT_POST, 'identifiant', FILTER_SANITIZE_STRING);
        $password = password_hash($_POST['password'], PASSWORD_BCRYPT);

        if (!empty($nom) && !empty($identifiant) && !empty($password)) {
            $administrateur = new Administrateur(null, $nom, $identifiant, $password);
            $this->administrateurModel->createAdministrateur($administrateur);
            header('Location: index.php?page=list-administrateurs');
        } else {
            $_SESSION['message'] = 'Veuillez remplir tous les champs.';
        }
    }

    echo $this->twig->render('administrateurController/addAdministrateur.html.twig');
}


public function updateAdministrateur()
{
    if ($_SERVER['REQUEST_METHOD'] === 'POST') {
        $id = filter_input(INPUT_POST, 'id', FILTER_SANITIZE_NUMBER_INT);
        $nom = filter_input(INPUT_POST, 'nom', FILTER_SANITIZE_STRING);
        $identifiant = filter_input(INPUT_POST, 'identifiant', FILTER_SANITIZE_STRING);

        if (!empty($id) && !empty($nom) && !empty($identifiant)) {
            $administrateur = new Administrateur($id, $nom, $identifiant, '');
            $this->administrateurModel->updateAdministrateur($administrateur);
            header('Location: index.php?page=list-administrateurs');
        } else {
            $_SESSION['message'] = 'Veuillez remplir tous les champs.';
        }
    } else {
        $id = filter_input(INPUT_GET, 'id', FILTER_SANITIZE_NUMBER_INT);
        $administrateur = $this->administrateurModel->getOneAdministrateur($id);
        echo $this->twig->render('administrateurController/updateAdministrateur.html.twig', ['administrateur' => $administrateur]);
    }
}
```

- Vérifie si le formulaire a été soumis en POST.

- Récupère et sécurise les données avec filter_input().

-Hache le mot de passe avec password_hash().

– Vérifie si tous les champs sont remplis.

- Crée un nouvel administrateur (new Administrateur(...)).

- Appelle createAdministrateur() pour l’ajouter à la base.

- Redirige l’utilisateur vers la liste des administrateurs.

```php
public function deleteAdministrateur()
{
    $id = filter_input(INPUT_GET, 'id', FILTER_SANITIZE_NUMBER_INT);
    $this->administrateurModel->deleteAdministrateur($id);
    header('Location: index.php?page=list-administrateurs');
}


public function showAdministrateurDetails()
{
    $id = filter_input(INPUT_GET, 'id', FILTER_SANITIZE_NUMBER_INT);
    $administrateur = $this->administrateurModel->getOneAdministrateur($id);
    echo $this->twig->render('administrateurController/showAdministrateurDetails.html.twig', ['administrateur' => $administrateur]);
}
```

La fonction deleteAdministrateur :

-Récupère l'ID depuis l'URL.

-Supprime l'administrateur en appelant deleteAdministrateur().

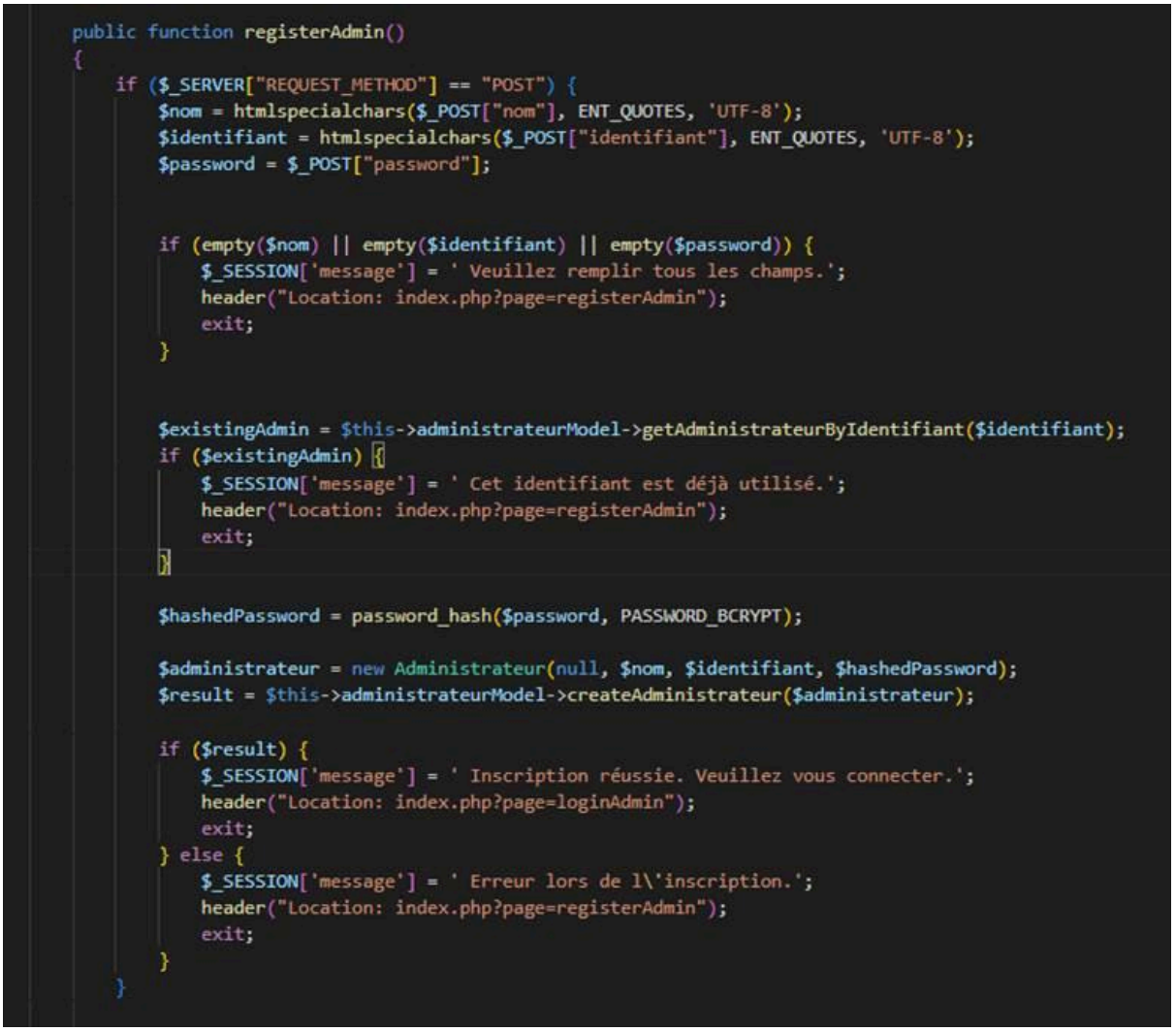
-Redirige l'utilisateur vers la liste après suppression.

La fonction Showadministrateur

-Récupère l'ID de l'administrateur.

-Charge ses détails via getOneAdministrateur().

-Affiche la vue Twig showAdministrateurDetails.html.twig.



```php
public function registerAdmin()
{
    if ($_SERVER["REQUEST_METHOD"] == "POST") {
        $nom = htmlspecialchars($_POST["nom"], ENT_QUOTES, 'UTF-8');
        $identifiant = htmlspecialchars($_POST["identifiant"], ENT_QUOTES, 'UTF-8');
        $password = $_POST["password"];


        if (empty($nom) || empty($identifiant) || empty($password)) {
            $_SESSION['message'] = ' Veuillez remplir tous les champs.';
            header("Location: index.php?page=registerAdmin");
            exit;
        }


        $existingAdmin = $this->administrateurModel->getAdministrateurByIdentifiant($identifiant);
        if ($existingAdmin) {
            $_SESSION['message'] = ' Cet identifiant est déjà utilisé.';
            header("Location: index.php?page=registerAdmin");
            exit;
        }

        $hashedPassword = password_hash($password, PASSWORD_BCRYPT);

        $administrateur = new Administrateur(null, $nom, $identifiant, $hashedPassword);
        $result = $this->administrateurModel->createAdministrateur($administrateur);

        if ($result) {
            $_SESSION['message'] = ' Inscription réussie. Veuillez vous connecter.';
            header("Location: index.php?page=loginAdmin");
            exit;
        } else {
            $_SESSION['message'] = ' Erreur lors de l\'inscription.';
            header("Location: index.php?page=registerAdmin");
            exit;
        }
    }
```

La fonction registerAdmin

- Vérifie si la requête HTTP est une soumission de formulaire via POST.

· Vérifie si les champs sont remplis.

· Si un champ est vide, un message d'erreur est stocké en session et l'utilisateur est redirigé vers la page d'inscription.

-Vérifie si l’identifiant est déjà utilisé en appelant la méthode getAdministrateurByIdentifiant().

- Si un administrateur existe déjà avec cet identifiant, un message d’erreur est affiché et l'utilisateur est redirigé.

-password_hash() est utilisé avec PASSWORD_BCRYPT pour sécuriser le mot de passe avant de l'enregistrer en base de données.

-Crée un nouvel objet Administrateur avec les données sécurisées.

-Appelle createAdministrateur() pour insérer l'administrateur en base.

- Si l'insertion en base réussit, redirige vers la page de connexion.

- Sinon, affiche un message d’erreur.

```php
public function loginAdmin()
{
    if ($_SERVER['REQUEST_METHOD'] === 'POST') {

        $identifiant = filter_input(INPUT_POST, 'identifiant', FILTER_SANITIZE_STRING);
        $password = $_POST['password'];

        if (!$identifiant || !$password) {
            $_SESSION['message'] = ' Identifiant ou mot de passe erroné.';
            header('Location: index.php?page=loginAdmin');
            exit;
        }


        $admin = $this->administrateurModel->getAdministrateurByIdentifiant($identifiant);

        if (!$admin) {
            $_SESSION['message'] = ' Identifiant incorrect.';
            header('Location: index.php?page=loginAdmin');
            exit;
        }


        if (!password_verify($password, $admin->getPassword())) {
            $_SESSION['message'] = ' Mot de passe erroné.';
            header('Location: index.php?page=loginAdmin');
            exit;
        }


        session_start();
        $_SESSION['admin'] = $admin->getIdentifiant();
        $_SESSION['admin_id'] = $admin->getIdentifiantAdmin();
        $_SESSION['message'] = ' Connexion réussie !';


        header('Location: index.php?page=adminDashboard');
        exit;
    }

    echo $this->twig->render("defaultController/loginAdmin.html.twig");
}
```

-Vérifie que la requête HTTP utilisée pour accéder à la fonction est bien une soumission de formulaire via POST.

filter_input(INPUT_POST, 'identifiant', FILTER_SANITIZE_STRING);

-Récupère l’identifiant et supprime les caractères spéciaux pour éviter les attaques

Si l’identifiant ou le mot de passe est vide :

-Stocke un message d’erreur dans $_SESSION['message'].

-Redirige l’utilisateur vers la page de connexion.

- Appelle getAdministrateurByIdentifiant($identifiant) pour chercher l’administrateur correspondant en base.
- Cette méthode doit retourner un objet Administrateur si l’utilisateur existe, sinon null
-Si aucun administrateur n’a été trouvé, un message d’erreur est affiché et l'utilisateur est redirigé vers index.php?page=loginAdmin.

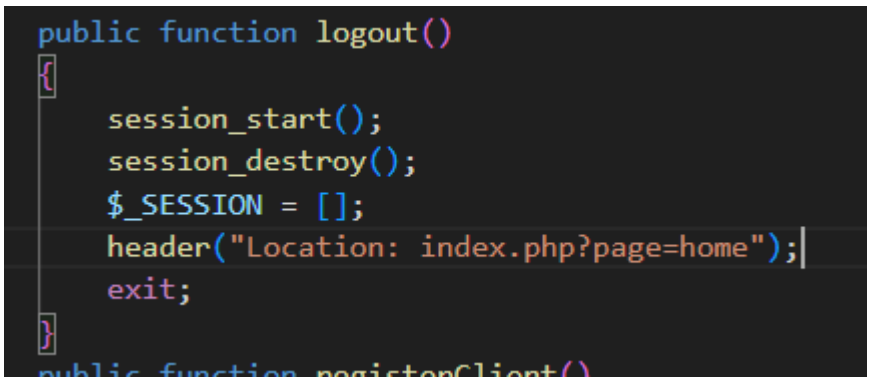
password_verify($password, $admin->getPassword()) :

- Compare le mot de passe entré avec celui stocké en base (haché).
- Retourne true si les deux correspondent.

Si le mot de passe est incorrect :

- Affiche un message d'erreur.
- Redirige l'utilisateur vers index.php?page=loginAdmin.

- Stocke les informations importantes en session :
  - $_SESSION['admin'] = $admin->getIdentifiant(); → Stocke l’identifiant de l'administrateur connecté.
  - $_SESSION['admin_id'] = $admin->getIdentifiantAdmin(); → Stocke son ID unique.
  - $_SESSION['message'] = 'Connexion réussie !'; → Message de confirmation.



```php
public function logout()
{
    session_start();
    session_destroy();
    $_SESSION = [];
    header("Location: index.php?page=home");
    exit;
}
public function registerClient()
```

- S'assure que la session est bien active avant de la manipuler.
- Si la session n'était pas déjà démarrée, cela l'initialise.
- Supprime la session en cours du serveur.
- Cela signifie que toutes les variables de session sont supprimées.
- Supprime la session en cours du serveur.
- Cela signifie que toutes les variables de session sont supprimées.

-Chaque contrôleur correspond à une entité et gère les actions associées (ex : ClientController.php gère les actions liées aux clients).

-Contiennent des méthodes qui reçoivent une requête HTTP, effectuent un traitement et retournent une réponse.

-Appellent les modèles pour récupérer des données et les envoient aux vues.

4. Le Routeur (src/Routing/)

Le routeur gère les URL et définit quel contrôleur doit être exécuté.

Ici le router est router.php

Associe chaque URL à un contrôleur spécifique.

```
'home' => [DefaultController::class, 'home'],
'404' => [DefaultController::class, 'error404'],
'500' => [DefaultController::class, 'error500'],
'register' => [DefaultController::class, 'register'],
'registerAdmin' => [DefaultController::class, 'registerAdmin'],
'adminDashboard' => [DefaultController::class, 'adminDashboard'],
```

Ce tableau définit les routes de l'application sous la forme :

- **Clé** → Nom de la page (récupérée via `$_GET['page']` dans l'URL).
- **Valeur** → Tableau contenant :
  - La classe du contrôleur (`DefaultController::class`).
  - La méthode à appeler (`home`, `error404`, etc.).

5. Les Templates (templates/)

Les templates contiennent le code HTML affiché aux utilisateurs.

-Affichent les données envoyées par les contrôleurs.

-Utilisent Twig pour générer du HTML dynamique qui se propage sur plusieurs pages comme la base.html.twig dont le footer et le header se propage sur les autres pages .

Explication de comment le tout marche :

-Un utilisateur fait une requête HTTP
→ /clients/1

Le routeur (Router.php) détermine quel contrôleur doit être appelé
→ ClientController.php

Le contrôleur exécute une action et appelle un modèle pour récupérer les données
→ ClientModel.php va chercher le client en base de données.

Les données sont envoyées à un template Twig (templates/)
→ show.html.twig

Twig génère la page HTML et l'affiche à l'utilisateur.

```
<title>
    {% block title %}{% endblock %}
</title>
```

```
>{% block content %}{% endblock %}
```

`{% block content %}` : Zone où le contenu spécifique de chaque page sera affiché.

```
<li class="nav-item">
    <div style="position: fixed; right: 500px; top: 10px; font-size: 16px;">
        <a class="nav-link" href="index.php?page=registerAdmin">S'inscrire en tant qu'admin</a>
    </div>
</li>
<li class="nav-item">
    <div style="position: fixed; right: 250px; top: 10px; font-size: 16px;">
        <a class="nav-link" href="index.php?page=loginAdmin">Se connecter en tant qu'admin</a>
    </div>
</li>

{%  if session.admin is defined and session.admin is not empty %}

    <li class="nav-item dropdown">
        <a class="nav-link dropdown-toggle" href="#" id="adminDropdown" role="button" data-bs-toggle="dropdown" aria-expanded="false">
            ⚙ Administration
        </a>
        <ul class="dropdown-menu" aria-labelledby="adminDropdown">
            <li>
                <a class="dropdown-item" href="index.php?page=adminCircuits">Gérer les circuits</a>
            </li>
            <li>
                <a class="dropdown-item" href="index.php?page=adminClients">Gérer les clients</a>
            </li>
            <li>
                <a class="dropdown-item" href="index.php?page=adminReservations">Gérer les réservations</a>
            </li>
            <li>
                <a class="dropdown-item" href="index.php?page=adminBeneficiaires">Gérer les bénéficiaires</a>
            </li>
            <li>
```

`{% if session.admin is defined and session.admin is not empty %}` : Vérifie si un administrateur est connecté pour afficher le menu admin.

ANNEXE

Dictionnaire de Donnée et dépendances Fonctionnelles

DICTIONNAIRE DE DONNÉE

Nom de l'Attribut	Type de Donnée	Description	Contraintes
Circuit			
Id	INT	Identifiant unique du circuit	Clé primaire
Descriptif	Varchar (255)	Descriptif du circuit	Not null
Ville_depart	Varchar (100)	Ville de départ du circuit	Not null
Ville_arrivée	Varchar (100)	Ville d'arrivée du circuit	Not null
Pays_depart	Varchar (100)	Pays de départ du circuit	Not null
Pays_arrivée	Varchar (100)	Pays d'arrivée du circuit	Not null
Date_depart	DATE	Date de départ	Not null
Nbr_place_dispo	Varchar (100)	Nbr place sur le circuit	Not null
Duree	TIME	Durée du circuit	En jour
prixInscription	Decimal (10,2)	prix pour l'inscription au circuit	
Etape			
Ordre	INT	Identifiant unique de l'etape	Clé primaire
Nom_lieu	Varchar (100)	Nom de l'étape	Not null
Date_etape	DATE		Not null

Duree	TIME		Not null
LieuAVisite			
Nom_lieux	INT	Identifiant lieu a visité	Clé primaire
Ville	Varchar (100)		
Pays	Varchar (100)		
Descriptif	Varchar (255)		Not null
Prix_visite	FLOAT	Prix visite du lieu visité (peut être gratuit)	Not null
Client			
Id	INT	Identifiant client	Clé primaire
Nom	Varchar (100)	Nom client	Not null
Prenom	Varchar (100)	Prenom client	Not null
Date_de_naissance	DATE	Date de naissance du client	Not null
Reservation_client			
Id_reservation	INT	Identifiant_reservation	Clé primaire
Date_reservation	DATE	Date de réservation	Not null
Nbr_place_reservation	Varchar (100)	Nombre de place de réservations	Not null
Beneficiaire			
Id_beneficiaire	INT	Identifiant du bénéficiaire	Clé primaire
Nom	Varchar (100)	Nom du bénéficiaire	Not null
Prenom	Varchar (100)	Prenom du bénéficiaire	Not null
Detail_beneficiaire			
Nbr_personne	Varchar (100)	Nombre de bénéficiaire	Not null

Date_reservation	DATE	Date de la réservation du bénéficiaire	Not null
Admin			
identifiantAdmin	INT	identifiant de la admin	clé primaire
Nom	Varchar (100)	nom de l'admin	Not null
identifiant	Varchar (100)	pseudo	unique, note null
password	Varchar (255)	mot de pass de l'admin	Not null

Dépendances fonctionnelles

Circuit

IdentifiantCircuit → (Descriptif, VilleDepart, PaysDepart, VilleArrivee, PaysArrivee, DateDepart, NbrPlaceDisponible, Duree, PrixInscription)

Etape

(identifiantCricuit, Ordre) →

(Nom_lieu, Ville, Pays, Date_etape, Duree)

Lieu à visiter

(Nom_lieu, Ville, Pays) →

(Descriptif, Prix_visite)

Client

Id_client →

(Nom, Prenom, Date de naissance)

Reservation_client

Id_reservation →

(Date_reservation, Nbr_de_place)

Bénéficiaire

Id_bénéficiaire →

(Nom, prénom)

Détail bénéficiaire

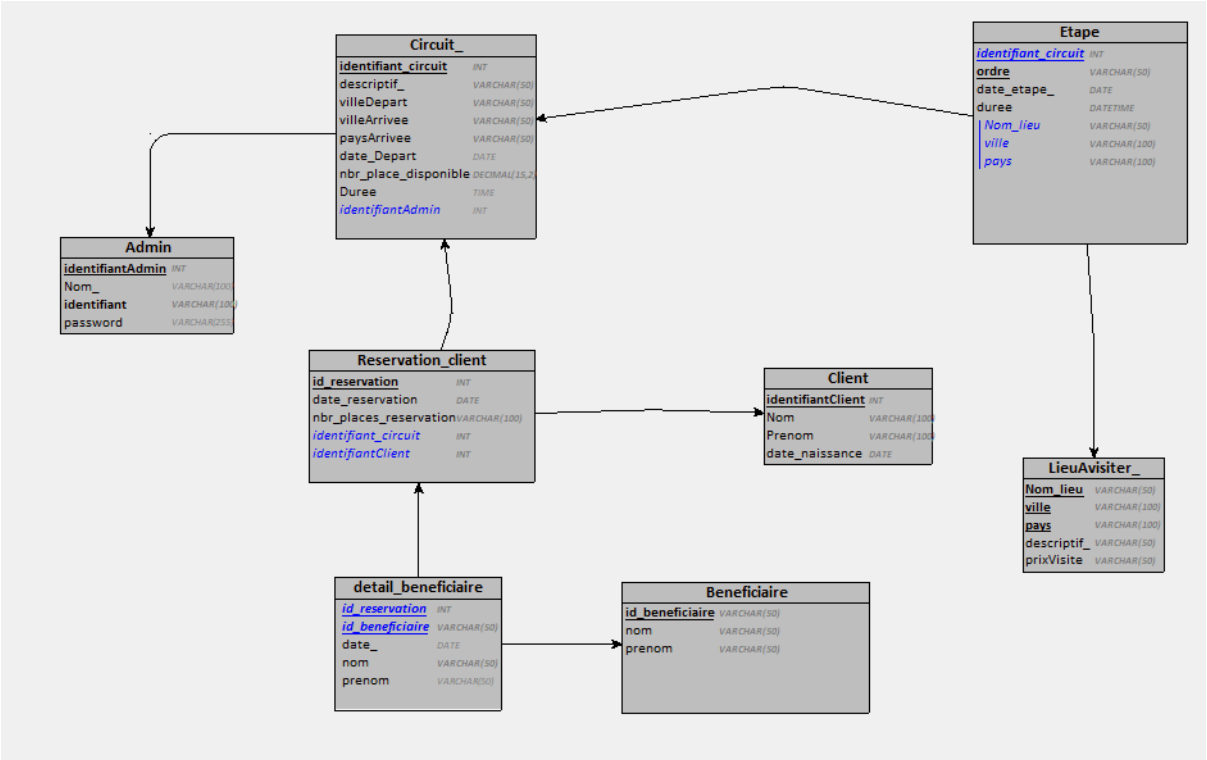
(Id_reservation, id beneficiaire) →

(nbr_personne, prix, Date_reservation)

Admin

Id_admin → (Nom, identifiant, password)

MLD



Conclusion :

Ce projet nous a appris à améliorer nos requêtes SQL, à gérer une base de données en lien avec une application de voyage. Le travail de groupe au niveau de l'organisation et le partage des connaissances nous a entrainer pour nos projets futurs.

Problèmes rencontrées :

- **1 dev et 2 réseaux (rééquilibrer les groupes sur les projets importants dans une spécialité en particulier)**
- **découvrir le javaScript pour les inscriptions pour améliorer notre application**
- **Manque de temp pour développer l'application**

Au niveau des axes d'amélioration :

- **Mettre au moins 2 dev pour mieux répartir les tâches (permet de faire le back-end et le front-end)**
- **La BDD peut être améliorer avec un héritage (on crée une table users qu'on lie avec admin / client / bénéficiaire)**
- **Possibilité du paiement du client (il peut réservé mais pas payer) pas de page de paiement et de sécurité**