

Projet Resto.fr – Itération 3

Réservation d'un restaurant

User story : En tant qu'utilisateur, je souhaite pouvoir réserver le restaurant que j'ai sélectionné dans la liste « détails » en indiquant mon nom, tel, date, heure et le nombre de personnes.

DAILY SCRUM DEBUT DE L'IT3 (25/11/2024) :	2
TÂCHE 01 : CREATION DE L'INTERFACE --> RESERVATION RESTAURANT	3
TÂCHE 02 : EDIT BDD	5
TÂCHE 03 : AJOUT DE NOUVELLES REQUETES DAO	7
TÂCHE 04 : CONSTRUCTEUR --> RESERVATION RESTAURANT	9
TÂCHE 05 : DEVELOPPEMENT DES INTERFACES	11
TÂCHE 06 : AMELIORATION DES CONSTRUCTEURS DE L'APP	12
TÂCHE 07 : TEST UNITAIRE (MODELE DAO)	14
TÂCHE 08 : TEST FONCTIONNEL	17
TÂCHE 09 : DEPLOIEMENT AVEC DOCKER	18
TÂCHE 10 : CREATION D'UN DOC UML, DIAGRAMME DE CLASSE... (OPTION)	20
RESULTAT FINAL :	21
DAILY SCRUM DEBUT DE L'IT3 (09/12/2024) :	22

DAILY SCRUM DEBUT DE L'IT3 (25/11/2024) :

GitLab : https://gitlab.com/KKG_Ambbussh/p2_g9_siteresto2024.git

- La branche correspondante à l'itération sur le dépôt est celle avec son numéro, si celle-ci n'est pas présente alors elle correspond aux mains
- Le README possède normalement le mode opératoire de test

Pendant la réunion :

- ➔ Renommage de tous les tickets en tâche comme demandé
- ➔ Création des tâches de l'IT3
- ➔ Attribution des tâches de l'IT3
- ➔ Mise en place d'un système d'importance & durée des tâches avec des labels majeur (majeur = +4h, moyen = 1h-2h, mineur = -1h)
- ➔ Mise en place de date d'échéance sur les tâches pour éviter les retards

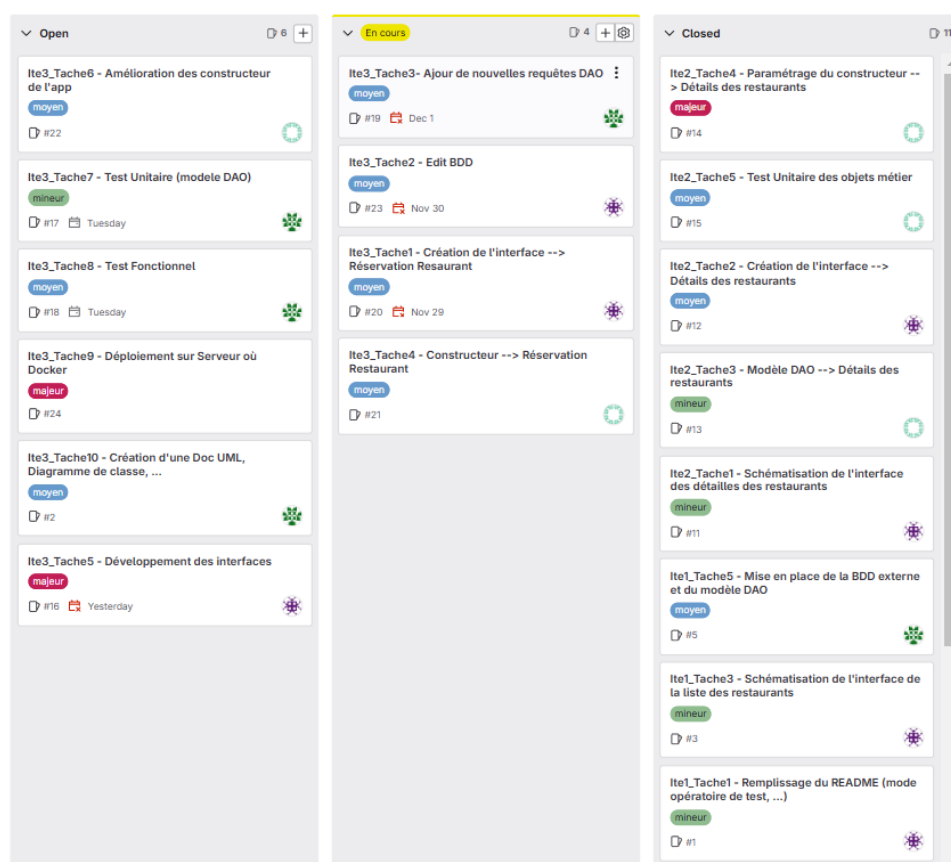
Répartition des tâches du début :

Thibault à Tâche 4, 6

Pierre à Tâche 3, 7, 8, 10

Thomas à Tâche 1, 2, 5

Tableau de Kanban du début :

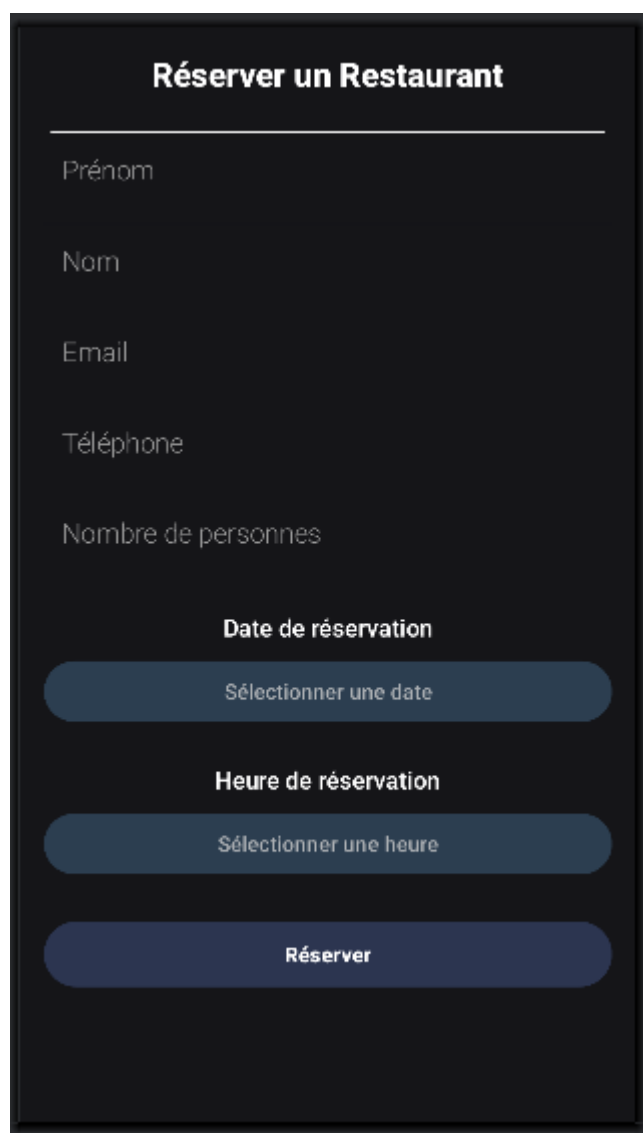


TÂCHE 01 : CREATION DE L'INTERFACE --> RESERVATION RESTAURANT

Pour répondre aux besoins de réservation, j'ai intégré les champs suivants dans le formulaire :

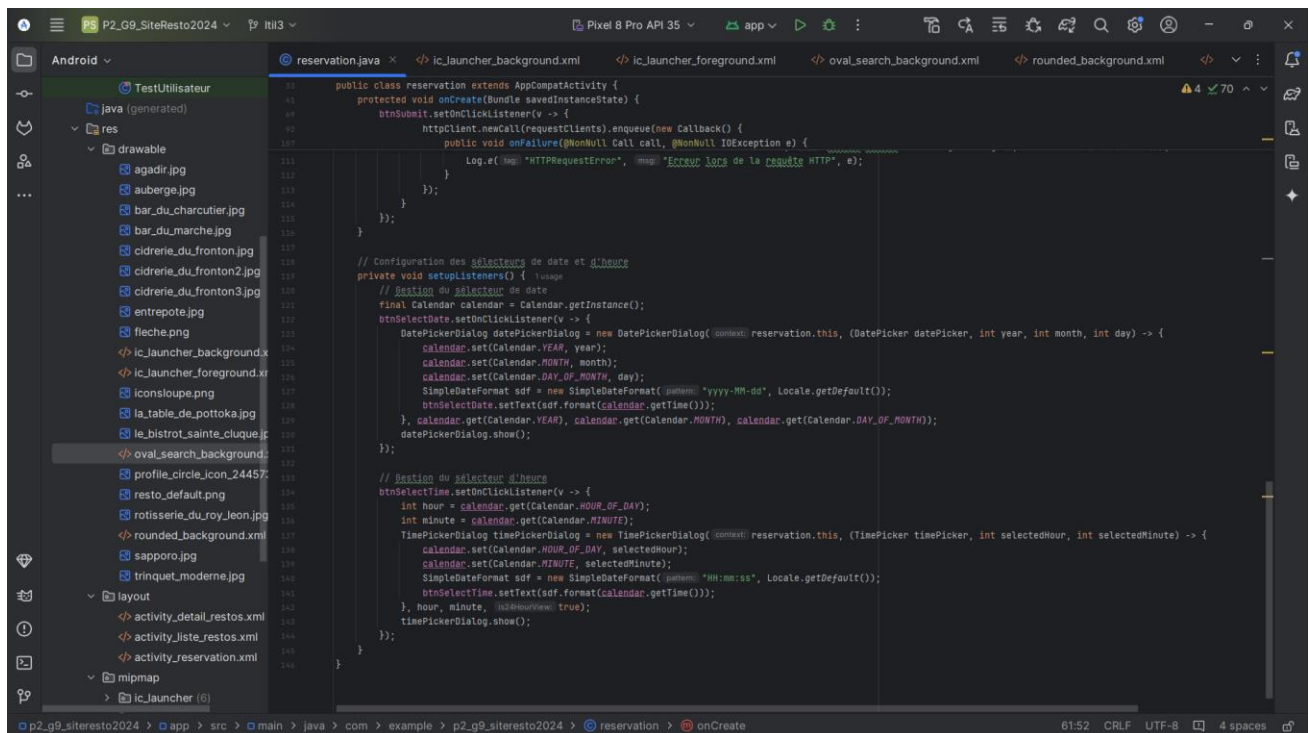
- **Nom** : permet d'identifier la personne qui réserve.
- **Prénom** : permet d'identifier la personne qui réserve.
- **Téléphone** : pour contacter la personne en cas de besoin.
- **Date et heure** : afin que l'utilisateur puisse choisir quand il souhaite réserver.
- **Nombre de personnes** : pour indiquer le nombre de convives.

Voici l'interface :



The image shows a dark-themed mobile application interface for restaurant reservations. At the top, the title 'Réserver un Restaurant' is displayed in white. Below the title, there are five input fields: 'Prénom', 'Nom', 'Email', 'Téléphone', and 'Nombre de personnes'. These fields are separated by horizontal lines. Below the input fields, there are two sections for date and time selection. The first section is titled 'Date de réservation' and contains a button labeled 'Sélectionner une date'. The second section is titled 'Heure de réservation' and contains a button labeled 'Sélectionner une heure'. At the bottom of the form, there is a large button labeled 'Réserver'.

Pour réaliser la sélection de la date et de l'heure, j'ai réalisé le code nécessaire :



J'ai eu des difficultés à réaliser les formulaires à cause du fait que je ne peux pas tester mon code car ma machine virtuelle (téléphone) ne se lance pas.

TÂCHE 02 : EDIT BDD

Deux modifications principales ont été apportées au script SQL :

1. **Création d'une table de réservation**
2. **Complétion des descriptions et des valeurs manquantes dans la table resto**

Ces ajustements ont pour but de répondre aux besoins de l'application en matière de gestion des réservations et d'améliorer les données existantes pour une meilleure utilisation.

1. Création de la table "réservation"

J'ai ajouté une nouvelle table, appelée `reservation`, pour gérer les réservations. Cette table est vide par défaut, mais elle contient les attributs nécessaires pour enregistrer les informations des utilisateurs. Voici le script SQL que j'ai ajouté pour créer cette table :

```
CREATE TABLE reservation (  
    numReservation INT AUTO_INCREMENT PRIMARY KEY,  
    nom VARCHAR(100) NOT NULL,  
    tel VARCHAR(15) NOT NULL,  
    date DATE NOT NULL,  
    heure TIME NOT NULL,  
    nbrPersonne INT NOT NULL  
);
```

Explication des colonnes :

- `numReservation` : Identifiant unique pour chaque réservation.
- `nom` : Nom de la personne effectuant la réservation.
- `tel` : Numéro de téléphone pour le contact.
- `date` et `heure` : Informations sur la date et l'heure de la réservation.
- `nbrPersonne` : Nombre de personnes pour la réservation.
-

2. Complétion des données de la table resto

La table `resto` contenait des champs incomplets, notamment des valeurs `NULL` dans les colonnes relatives à la description des restaurants. J'ai modifié le script pour remplir ces champs avec des données fictives ou réalistes.

Voici un exemple des modifications ajoutées :

```
INSERT INTO resto (id, nom, ville, voie, description)  
VALUES  
(1, 'Le Gourmet', 'Lyon', '15 Rue des Fleurs', 'Un restaurant raffiné proposant  
une cuisine française de qualité.'),
```

```
(2, 'Chez Marie', 'Paris', '12 Avenue de la République', 'Une ambiance  
chaleureuse et des plats faits maison.');
```

Voici la bdd après réalisation des modifications :



TÂCHE 03 : AJOUT DE NOUVELLES REQUETES DAO

```
<?php
require 'connexion.php';
```

On commence par une inclusion du fichier connexion.php qui permet la connexion à la BDD

```
try {
    // Vérification que la donnée est fournie
    if (isset($_GET['nom'], $_GET['prenom'], $_GET['tel'], $_GET['email'], $_GET['date'], $_GET['heure'], $_GET['nbrPersonne'], $_GET['idResto'])) {
        $nom = $_GET['nom'];
        $prenom = $_GET['prenom'];
        $tel = $_GET['tel'];
        $email = $_GET['email'];
        $date = $_GET['date']; // Format attendu : YYYY-MM-DD
        $heure = $_GET['heure']; // Format attendu : HH:MM:SS
        $nbrPersonne = $_GET['nbrPersonne'];
        $idResto = $_GET['idResto'];
    }
```

Ensuite nous faisons une vérification de chaque paramètre n'est pas null, et la récupération de chaque données. Si un paramètre est null cela saute le if automatiquement et l'erreur est gérée avec un catch.

```
// Préparer la requête SQL
$reponse = $connexion->prepare(
    "INSERT INTO reservation (`nom`, `prenom`, `tel`, `email`, `date`, `heure`, `nbrPersonne`, `idResto`)
    VALUES (:nom, :prenom, :tel, :email, :dateR, :heure, :nbrPersonne, :idResto)"
);
```

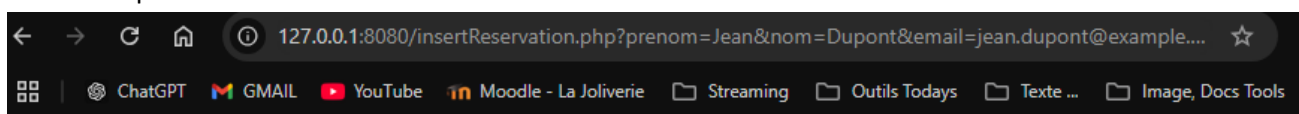
Ici nous avons nous avons une requête préparer ce qui rend la requête SQL sécurisé.

```
// Lier les paramètres
$reponse->bindParam(':nom', $nom);
$reponse->bindParam(':prenom', $prenom);
$reponse->bindParam(':tel', $tel);
$reponse->bindParam(':email', $email);
$reponse->bindParam(':dateR', $date);
$reponse->bindParam(':heure', $heure);
$reponse->bindParam(':nbrPersonne', $nbrPersonne);
$reponse->bindParam(':idResto', $idResto);
```

Le bindParam permet de lier chaque placeholder dans la requête SQL à une variable PHP

```
// Exécuter la requête
if ($reponse->execute()) {
    echo "Réservation enregistrée avec succès.";
} else {
    echo "Erreur lors de l'insertion de la réservation.";
}
} else {
    echo "Paramètres manquants.";
}
} catch (PDOException $e) {
    echo "Erreur : " . $e->getMessage();
}
?>
```

Si tout jusqu'à là à fonctionner cela exécute la requête sinon cela gère l'erreur et affiche Paramètres manquants



Réservation enregistrée avec succès.

Et si à l'avenir on souhaite supprimer une réservation, j'ai fait un programme pour préparer ceci dans l'api

```
<?php
require 'connexion.php';

try {
    // Vérification que l'ID de la réservation est fourni
    if (isset($_GET['idReservation'])) {
        $idReservation = $_GET['idReservation'];

        // Préparer la requête SQL pour supprimer la réservation
        $reponse = $connexion->prepare(
            query: "DELETE FROM reservation WHERE idReservation = :idReservation"
        );

        // Lier les paramètres
        $reponse->bindParam(param: ':idReservation', var: &$idReservation);

        // Exécuter la requête
        if ($reponse->execute()) {
            if ($reponse->rowCount() > 0) {
                echo "Réservation supprimée avec succès.";
            } else {
                echo "Aucune réservation trouvée avec cet ID.";
            }
        } else {
            echo "Erreur lors de la suppression de la réservation.";
        }
    } else {
        echo "Paramètre idReservation manquant.";
    }
} catch (PDOException $e) {
    echo "Erreur : " . $e->getMessage();
}
?>
```


TÂCHE 04 : CONSTRUCTEUR --> RESERVATION RESTAURANT

Dans cette tâche ma mission était de paramétrer le constructeur de l'interface réservation pour qu'il envoie à la bdd une réservation, avec une heure, une date, un nom, prénom, ...

Pour réaliser ceci j'ai de la premièrement préparé dans la table liste_resto, un intent qui envoyait l'idR du resto sélectionné précédemment et nous redirigeait vers la réservation.

```
View.OnClickListener ecouteur = new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent = new Intent( packageContext: detail_restos.this, reservation.class);
        intent.putExtra( name: "idR", idR);
        startActivity(intent);
    }
};

btnReservation.setOnClickListener(ecouteur);
```

Ensuite au niveau de reservation.java, il fallait dans un premier temps déclarer les views de l'interface ainsi que de récupérer l'idResto du Intent

```
// Déclaration des widgets de l'interface utilisateur
private EditText etFirstName, etLastName, etEmail, etTel, etNbrPersonnes; 2 usages
private Button btnSelectDate, btnSelectTime, btnSubmit; 4 usages

// Récupération des données transmises via l'Intent
Intent intent = getIntent();
int idR = intent.getIntExtra( name: "idR", defaultValue: -1);

// Initialisation des widgets en récupérant leur référence via les ID
etFirstName = findViewById(R.id.etFirstName);
etLastName = findViewById(R.id.etLastName);
etEmail = findViewById(R.id.etEmail);
etTel = findViewById(R.id.etTel);
etNbrPersonnes = findViewById(R.id.etNbrPersonnes);
btnSelectDate = findViewById(R.id.btnSelectDate);
btnSelectTime = findViewById(R.id.btnSelectTime);
btnSubmit = findViewById(R.id.btnSubmit);
```

Après j'ai dû préparer le déroulement des étapes en cas de clic sur le bouton :

- Récupération des champs saisis par l'utilisateur

```
// Ajout d'une action au bouton de soumission
btnSubmit.setOnClickListener(v -> {
    // Récupération des valeurs saisies par l'utilisateur
    String firstName = etFirstName.getText().toString();
    String lastName = etLastName.getText().toString();
    String email = etEmail.getText().toString();
    String tel = etTel.getText().toString();
    String nbrPersonnes = etNbrPersonnes.getText().toString();
    String date = btnSelectDate.getText().toString();
    String time = btnSelectTime.getText().toString();
```

- Vérification du remplissage des champs

```
// Vérification si tous les champs sont remplis
if (firstName.isEmpty() || lastName.isEmpty() || email.isEmpty() || tel.isEmpty() || nbrPersonnes.isEmpty() || date!="0000-00-00" || time!="00:00:00"){
    Toast.makeText(context, reservation.this, text: "Veuillez remplir tous les champs", Toast.LENGTH_SHORT).show();
} else {
```

- Préparation de la requête insert vers la bdd

```
} else {
    // Construction de l'URL avec les paramètres attendus par le serveur
    String url = "http://192.168.56.1/API_Resto/insertReservation.php?" +
        "prenom=" + firstName + "&nom=" + lastName + "&email=" + email + "&tel=" + tel + "&nbrPersonne=" + nbrPersonnes + "&date=" + date + "&heure=" + time + "&idResto=" + idR;

    // Préparation de la requête HTTP
    Request requestClients = new Request.Builder().url(url).build();
    OkHttpClient httpClient = new OkHttpClient();
```

- Envoi de la requête et préparation de la réponse (en cas de réussite redirection vers le detail_restaurant)

```
// Envoi de la requête HTTP en arrière-plan
httpClient.newCall(requestClients).enqueue(new Callback() {
    @Override 4 usages
    public void onResponse(@NonNull Call call, @NonNull Response response) throws IOException {
        // Vérification si la réponse du serveur est réussie
        if (response.isSuccessful()) {
            runOnUiThread() -> Toast.makeText(context, reservation.this, text: "Réservation effectuée pour " + firstName + " " + lastName, Toast.LENGTH_LONG).show();
            finish();
        } else {
            // Gestion des erreurs côté serveur
            final String errorResponse = response.body() != null ? response.body().string() : "Aucune réponse";
            runOnUiThread() -> Toast.makeText(context, reservation.this, text: "Erreur serveur : " + response.code() + " - " + errorResponse, Toast.LENGTH_LONG).show();
        }
    }
}
```

Cas d'erreur

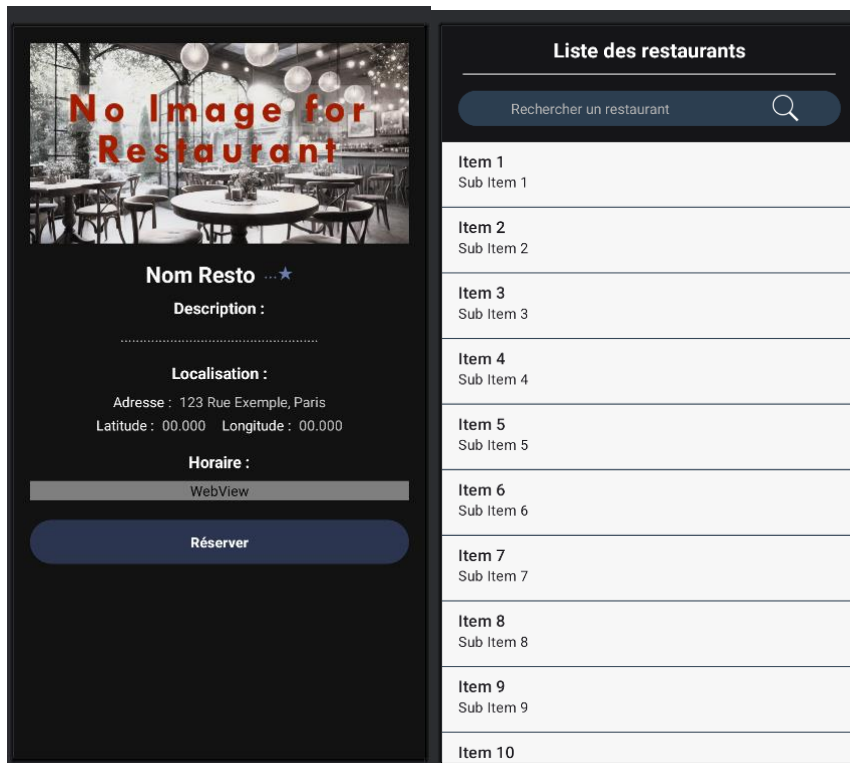
Cas réussie

numReservation	nom	prenom	tel	email	date	heure	nbrPersonne	idResto
1	Grimaud	Thibault	01 02 03 04 05	tgrimaud@joliverie.com	2024-12-15	19:49:18	2	11

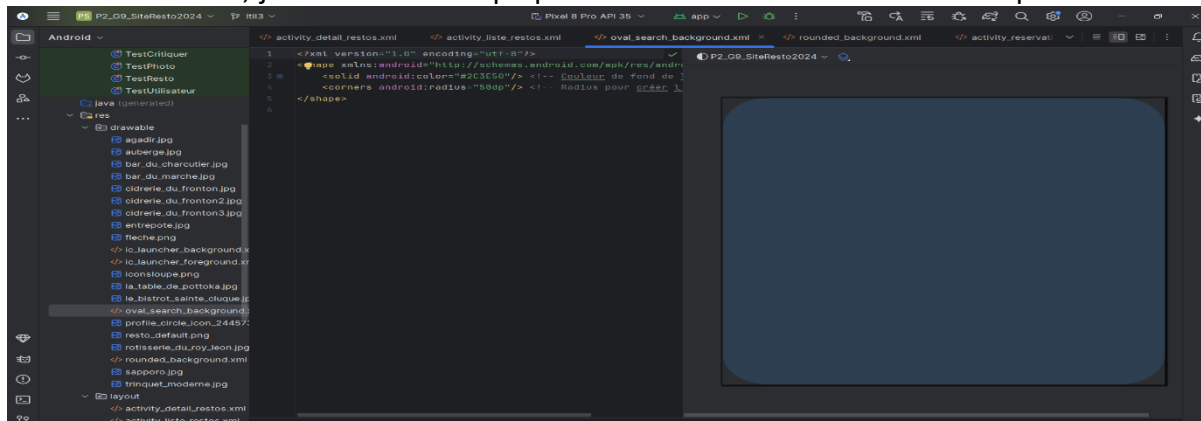
TÂCHE 05 : DEVELOPPEMENT DES INTERFACES

J'ai mis à jours les autres interfaces pour mettre toutes les interfaces sur la même DA (direction artistique) pour améliorer l'utilisation de l'app des clients :

Voici l'interface de la liste des restos et celle des détails restos :



J'ai créé d'autre fichier .xml que j'ai appelé dans le fichier principal afin de créer le visuel des boutons. En effet, j'ai créé un .xml qui permet de faire les boutons ovales pour réserver :



De plus, j'ai réalisé un bouton de retour en arrière sur la page réservation (.xml et .class) vers la page détails resto. Cependant, cela a créé des conflits avec le code. On a donc choisi de ne pas mettre cette fonctionnalité pour éviter les conflits, de plus les appareils androides permette directement de retourner en arrière. Donc cette fonctionnalité n'est vraiment utile.

TÂCHE 06 : AMELIORATION DES CONSTRUCTEURS DE L'APP

Dans cette tâche bonus, je me suis donné pour mission d'optimiser le code de l'application afin de la rendre moins lourde, plus compréhensible, et d'ajouter des fonctionnalités supplémentaires. Cela comprenait également la réalisation de correctifs applicatifs et l'intégration d'une meilleure gestion des erreurs, tâches que nous n'avions pas pu effectuer auparavant. Voici les principales actions réalisées :

1. Amélioration de la lisibilité du code

- **Commentaires** : J'ai corrigé, supprimé ou reformulé certains commentaires pour mieux expliquer le fonctionnement du code.
- **Structure** : J'ai réorganisé le code pour qu'il ne ressemble plus à un brouillon, mais suive une structure logique et cohérente.
- **Nommage** : Les variables, fonctions et autres éléments mal nommés ont été renommés pour respecter les conventions de développement et améliorer la lisibilité.

```
// Récupération des données
// Description des données
TextView textViewNomResto = (TextView) findViewById(R.id.nom_resto);
TextView textViewNoteResto = (TextView) findViewById(R.id.note_resto);
TextView textViewDescription = (TextView) findViewById(R.id.description_resto);
TextView textViewAdresse = (TextView) findViewById(R.id.adresse_resto);
TextView textViewLatitude = (TextView) findViewById(R.id.latitude_resto);
TextView textViewLongitude = (TextView) findViewById(R.id.longitude_resto);
TextView textViewHoraires = (TextView) findViewById(R.id.horaires_resto);
Button btnReservation = (Button) findViewById(R.id.btnReservation);

// Récupération des données transmises via l'intent
Intent intent = getIntent();
int id = intent.getIntExtra("id", -1);

// Création de la requête HTTP vers le serveur PHP
Request request = new Request.Builder().url("http://10.0.2.15:8080/getRestoById.php?id=" + id).build();
OkHttpClient httpClient = new OkHttpClient();

// Envoi de la requête HTTP en arrière-plan
httpClient.newCall(request).enqueue(new Callback() {
    @Override public void onResponse(Call call, Response response) throws IOException {
        // Traitement de la réponse HTTP reçue
        final String myResponse = response.body().string();
        detail_resto.this.runOnUiThread() -> {
            try {
                // Conversion de la réponse en un objet JSON
                JSONObject jsonObject = new JSONObject(myResponse);
            }
        }
    }
});
```

2. Optimisation des performances

- **Nettoyage** : J'ai supprimé les éléments inutilisés, tels que des *imports*, des ressources *drawable* non utilisées, ou encore du code redondant.

```
8 8 import android.webkit.WebView;
9 - import android.widget.AdapterView;
10 9 import android.widget.Button;
11 - import android.widget.EditText;
12 10 import android.widget.ImageView;
13 - import android.widget.ListView;
14 11 import android.widget.TextView;
15 12
16 13 import androidx.activity.EdgeToEdge;
17 ... @ -20,14 +17,10 @ import androidx.core.graphics.Insets;
18 20 import androidx.core.view.ViewCompat;
19 21 import androidx.core.view.WindowInsetsCompat;
20 22
21 23 - import com.example.p2_g2_siteresto2024.net.Resto;
22 24
23 25 - import org.json.JSONArray;
24 26 import org.json.JSONException;
25 27 import org.json.JSONObject;
26 28
27 29 import java.io.IOException;
28 30 import java.util.ArrayList;
29 31
30 32
```

- **Amélioration des échanges de données** : L'optimisation principale consistait à simplifier les échanges entre les pages *liste* et *détail*. Au lieu de transmettre chaque donnée via des *intents*, seul l'ID du restaurant est désormais envoyé. Les informations sont ensuite récupérées à l'aide de la méthode *getRestoById* de l'API (implémentée lors de l'itération 1). Cette modification réduit la quantité de données stockées en local et permet d'afficher des informations supplémentaires sur la page *détail*.

```
// Création de l'Intent pour passer à l'activité de détail
Intent intent = new Intent(liste_restos.this, detail_restos.class);
intent.putExtra("id", selectedResto.getId());
intent.putExtra("nomR", selectedResto.getNomR());
intent.putExtra("numAdrR", selectedResto.getNumAdr());
intent.putExtra("voieAdrR", selectedResto.getVoieAdr());
intent.putExtra("cpR", selectedResto.getCpR());
intent.putExtra("villeR", selectedResto.getVilleR());
intent.putExtra("latitudeDegR", selectedResto.getLatitudeDegR());
intent.putExtra("longitudeDegR", selectedResto.getLongitudeDegR());
intent.putExtra("descR", selectedResto.getDescR());
intent.putExtra("horairesR", selectedResto.getHorairesR());
//intent.putExtra("PhotoR", selectedResto.getPhotos());
startActivity(intent);
});

// Récupération des données
// Description des données
TextView textViewNomResto = (TextView) findViewById(R.id.nom_resto);
TextView textViewNoteResto = (TextView) findViewById(R.id.note_resto);
TextView textViewDescription = (TextView) findViewById(R.id.description_resto);
TextView textViewAdresse = (TextView) findViewById(R.id.adresse_resto);
TextView textViewLatitude = (TextView) findViewById(R.id.latitude_resto);
TextView textViewLongitude = (TextView) findViewById(R.id.longitude_resto);
TextView textViewHoraires = (TextView) findViewById(R.id.horaires_resto);
Button btnReservation = (Button) findViewById(R.id.btnReservation);

// Récupération des données transmises via l'intent
Intent intent = getIntent();
int id = intent.getIntExtra("id", -1);

// Création de la requête HTTP vers le serveur PHP
Request request = new Request.Builder().url("http://10.0.2.15:8080/getRestoById.php?id=" + id).build();
OkHttpClient httpClient = new OkHttpClient();

// Envoi de la requête HTTP en arrière-plan
httpClient.newCall(request).enqueue(new Callback() {
    @Override public void onResponse(Call call, Response response) throws IOException {
        // Traitement de la réponse HTTP reçue
        final String myResponse = response.body().string();
        detail_resto.this.runOnUiThread() -> {
            try {
                // Conversion de la réponse en un objet JSON
                JSONObject jsonObject = new JSONObject(myResponse);
            }
        }
    }
});
```

3. Correctifs applicatifs

- **Affichage des horaires** : J'ai corrigé un bug où les horaires affichés dans le tableau ne correspondaient pas à ceux de la base de données, mais provenaient d'un tableau fixe initialisé en local.

Ouverture	Semaine	Week-end
Midi	de 11h45 à 14h30	de 11h45 à 15h00
Soir	de 18h45 à 22h30	de 18h45 à 1h
À emporter	de 11h30 à 23h	de 11h30 à 2h

```

71 + String horairesR = intent.getStringExtra("horairesR");
72 //String noteR = intent.getStringExtra("noteR");
73 // Remplissage des champs avec les données
74
75 -
76 -
77 - //Drawable drawable = Drawable.createFromPath("imageR");
78 - //ImageResto.setImageDrawable(drawable);
79 -
80 + //Chargement des TextView
81 + textViewNomResto.setText(nomR);
82 + textViewAdresse.setText(adresse);
83 + textViewLongitude.setText(String.valueOf(Longitude));
84 + textViewLatitude.setText(String.valueOf(Latitude));
85 + textViewDescription.setText(descR);
86
87 - // Code HTML pour les horaires
88 - String htmlContent = "<table border='1' style='width:100%; text-align:left;'>"
89 - + "<thead>"
90 - + "<tr><th>Ouverture</th><th>Semaine</th><th>Week-end</th></tr>"
91 - + "</thead>"
92 - + "<tbody>"
93 - + "<tr><td>Midi</td><td>de 11h45 à 14h30</td><td>de 11h45 à 15h00</td></tr>"
94 - + "<tr><td>Soir</td><td>de 18h45 à 22h30</td><td>de 18h45 à 1h</td></tr>"
95 - + "<tr><td>À emporter</td><td>de 11h30 à 23h</td><td>de 11h30 à 2h</td></tr>"
96 - + "</tbody>"
97 - + "</table>";
98 -
99 - // Charger le contenu HTML dans la WebView
100 + // Chargement du contenu HTML dans la WebView
101 + webViewHoraires.setJavaScriptEnabled(true); // Si nécessaire pour le HTML
102 + webViewHoraires.loadData(htmlContent, "text/html", "UTF-8");
103 + webViewHoraires.loadData(horairesR, "text/html", "UTF-8");

```

4. Ajout de nouvelles options

- **Affichage d'une image du restaurant** : Une image de la devanture du restaurant est désormais affichée sur la page *détail*, en fonction du restaurant sélectionné dans la liste.



```

String nomImage = cheminP.substring(0, cheminP.lastIndexOf( str: ".")); // Extraire le nom
int imageResId = getResources().getIdentifier(nomImage, defType: "drawable", getPackageName());
imageResto.setImageResource(imageResId);

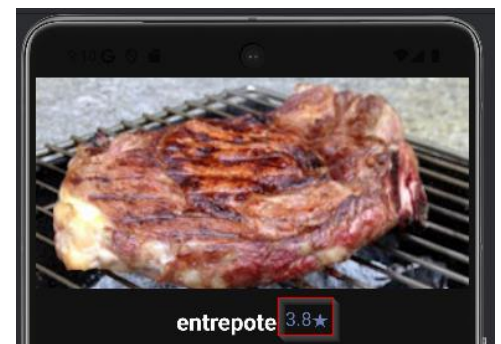
```

- **Note moyenne** : J'ai ajouté l'affichage de la note moyenne attribuée par les utilisateurs. Pour cela, il a fallu modifier l'API afin de récupérer les évaluations des utilisateurs et calculer leur moyenne.

```

SELECT r.*, p.*, AVG(c.note)
if (noteMoy == 0.0){
    textViewNoteResto.setText("...");
} else {
    textViewNoteResto.setText(String.valueOf(noteMoy));
}

```



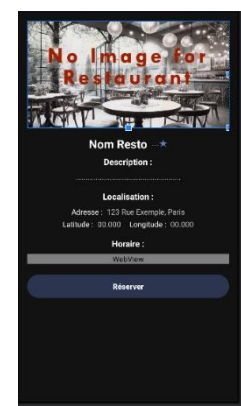
5. Gestion des erreurs

- **Fallback pour les images manquantes** : Si aucune image n'est trouvée pour un restaurant, une image par défaut est affichée.

```

//Chargement de l'image dans le ImageView
if (cheminP == "null") {
    imageResto.setImageResource(R.drawable.resto_default);
} else {
    String nomImage = cheminP.substring(0, cheminP.lastIndexOf( str: ".")); // Extraire le nom du
    int imageResId = getResources().getIdentifier(nomImage, defType: "drawable", getPackageName());
    imageResto.setImageResource(imageResId);
}

```



Ces différentes optimisations et ajouts ont permis de rendre l'application plus performante, user-friendly et stable tout en facilitant sa maintenance future.

TÂCHE 07 : TEST UNITAIRE (MODELE DAO)

Dans l'API, on a au total 4 méthode php pour communiquer avec la bdd et récupérer ou envoyer des données, les voici et comment il fonctionne :

Le premier est Connexion.php permet la connexion à la BDD, pour les autre API, on s'en servira dans chaque API de façon redondante

```

184 / www / API_Resto / connexion.php
<?php
$db="resto2";
$dbhost="localhost";
$dbport=3306;
$dbuser="resto_util";
$dbpasswd="secret";

try {
    $connexion = new PDO(dsn: 'mysql:host='.$dbhost.';port='.$dbport.';dbname='.$db.'', username: $dbuser, password: $dbpasswd);
    $connexion->exec(statement: "SET CHARACTER SET utf8");
} catch (PDOException $e) {
    echo "Erreur de connexion : " . $e->getMessage();
    die();
}
?>

```

La seconde méthode est getAllResto.php qui sert pour la liste des restos et dans ce cas-ci, à récupérer tous les restos de la bdd et les stocker sous format json :

```

Projet_SiteRestoAndroid > API_Resto > getAllResto.php > ...
1 <?php
2 require 'connexion.php';
3
4 try {
5     $reponse = $connexion->prepare(query: "SELECT * FROM resto r INNER JOIN photo p ON r.idR = p.idR GROUP BY r.idR ORDER BY r.idR ASC;");
6     $reponse->execute();
7     $datas = array();
8
9     while($res=$reponse->fetch(mode: PDO::FETCH_ASSOC)) {
10         $datas['resto'][]=$res;
11     }
12     echo json_encode(value: $datas);
13 } catch (PDOException $e) {
14     echo json_encode(value: ["error" => $e->getMessage()]);
15 }
16 }
17 ?>

```

```

[{"idR":1,"nomR":"entrepot","nomAdR":"","voiesAdR":"rue Maurice Ravel","cpR":"33000","villeR":"Bordeaux","latitudeDegR":44.7948,"longitudeDegR":-0.58754,"descR":"Un restaurant chaleureux à Bordeaux, proposant une cuisine française traditionnelle dans un cadre moderne et d'atmosphère conviviale","imagesR":["a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p","q","r","s","t","u","v","w","x","y","z"]},
{"idR":2,"nomR":"le bar du charcutier","nomAdR":"","voiesAdR":"rue du Parlement Sainte Catherine","cpR":"33000","villeR":"Bordeaux","latitudeDegR":44.8378,"longitudeDegR":-0.5792,"descR":"Un endroit incontournable pour les amateurs de charcuterie traditionnelle bordelaise"},
{"idR":3,"nomR":"Sapporo","nomAdR":"","voiesAdR":"rue Saint Rémi","cpR":"33000","villeR":"Bordeaux","latitudeDegR":44.8378,"longitudeDegR":-0.5768,"descR":"Le Sapporo propose à ses clients de délicieux plats japonais dans un cadre moderne et d'atmosphère conviviale"},
{"idR":4,"nomR":"Café du fronton","nomAdR":"","voiesAdR":"Place du Fronton","cpR":"64110","villeR":"Arbonne","latitudeDegR":43.4264,"longitudeDegR":-1.4826,"descR":"Un restaurant rustique à Arbonne, spécialisé dans la cuisine traditionnelle bordelaise"},
{"idR":5,"nomR":"Agade","nomAdR":"","voiesAdR":"Rue Sainte Catherine","cpR":"64100","villeR":"Bayonne","latitudeDegR":43.4329,"longitudeDegR":-1.4748,"descR":"Restaurant marocain à Bayonne, offrant une expérience culinaire unique"},
{"idR":6,"nomR":"Le bistrot Sainte Claire","nomAdR":"","voiesAdR":"Rue Hugues","cpR":"64100","villeR":"Bayonne","latitudeDegR":43.4327,"longitudeDegR":-1.4733,"descR":"Un bistrot cosy à Bayonne, idéal pour déguster des plats traditionnels"}]

```

Test Méthode

BDD

idR	nomR	nomAdR	voiesAdR	cpR	villeR	latitudeDegR	longitudeDegR	descR	imagesR
1	entrepot	2	rue Maurice Ravel	33000	Bordeaux	44.7948	-0.58754	Un restaurant chaleureux à Bordeaux, proposant une...	<table><thead><tr><th></th></tr></thead><tbody><tr><td></td></tr></tbody></table>
2	le bar du charcutier	30	rue du Parlement Sainte Catherine	33000	Bordeaux	44.8378	-0.5792	Un endroit incontournable pour les amateurs de cha...	<table><thead><tr><th></th></tr></thead><tbody><tr><td></td></tr></tbody></table>
3	Sapporo	33	rue Saint Rémi	33000	Bordeaux	44.8378	-0.5768	Le Sapporo propose à ses clients de délicieux plats...	<table><thead><tr><th></th></tr></thead><tbody><tr><td></td></tr></tbody></table>
4	Café du fronton	1	Place du Fronton	64110	Arbonne	43.4264	-1.4826	Un restaurant rustique à Arbonne, spécialisé dans ...	<table><thead><tr><th></th></tr></thead><tbody><tr><td></td></tr></tbody></table>
5	Agade	3	Rue Sainte Catherine	64100	Bayonne	43.4329	-1.4748	Restaurant marocain à Bayonne, offrant une expérie...	<table><thead><tr><th></th></tr></thead><tbody><tr><td></td></tr></tbody></table>
6	Le bistrot Sainte Claire	9	Rue Hugues	64100	Bayonne	43.4327	-1.4733	Un bistrot cosy à Bayonne, idéal pour déguster des ...	<table><thead><tr><th></th></tr></thead><tbody><tr><td></td></tr></tbody></table>

```

1  <?php
2  require 'connexion.php';
3
4  try {
5      // Vérification que la donnée est fournie
6      if (isset($_GET['idR'])) {
7          $idR = $_GET['idR'];
8          $reponse=$connexion->prepare(query: "SELECT r.*, p.*, AVG(c.note) as noteMoy FROM resto r LEFT JOIN photo p ON r.idR = p.idR LEFT JOIN critiquer c ON c.idR = r.idR WHERE r.idR = :idR GROUP BY r.idR;");
9          // Préparer la requête SQL d
10         $reponse->bindParam(param: ':idR', var: &$idR);
11
12         // Exécuter la requête
13         if ($reponse->execute()) {
14             $data = $reponse->fetch(mode: PDO::FETCH_ASSOC);
15             echo json_encode(value: $data);
16
17         } else {
18             echo json_encode(value: ['error' => 'Erreur lors de la sélection de données']);
19         }
20     } else {
21         echo json_encode(value: ['error' => 'Paramètres manquants.']);
22     }
23 } catch (PDOException $e) {
24     echo json_encode(value: ["error" => $e->getMessage()]);
25 }
26 >

```

Test Méthode

idR	nomR	numAdr	voieAdr	cpR	villeR	latitudeDegR	longitudeDegR	descR	horairesR	idP	cheminP	idR	noteMo
1	enseigne	4	100	20000	GORREAU	44.7260	-1.50729	une...		0	enseigne.jpg	1	2.0000
2	le bar du charcutier	30	rue Parlement Sainte-Catherine	33000	Bordeaux	44.8378	-0.5792	Un endroit incontournable pour les amateurs de cha...	<th>O...</th><thead><tr><tr></thead></tr>	4	bar_du_charcutier.jpg	2	2.2500
3	Sapporo	33	rue Saint Rémi	33000	Bordeaux	44.8378	-0.5768	Le Sapporo propose à ses clients de délicieux plat...	<thead><tr><tr></thead></tr>	2	sapporo.jpg	3	NULL
4	Cidriele du fronton	1	Place du Fronton	64210	Arbonne	43.4264	-1.4826	Un restaurant rustique à Arbonne, spécialisé dans ...	<thead><tr><tr></thead></tr>	6	cidriele_du_fronton.jpg	4	5.0000
5	Agadir	3	Rue Sainte-Catherine	64100	Bayonne	43.4929	-1.4748	Restaurant marocain à Bayonne, offrant une expérie...	<thead><tr><tr></thead></tr>	7	agadir.jpg	5	3.0000
6	Le Bistrot Sainte Cluque	9	Rue Hugues	64100	Bayonne	43.4927	-1.4733	Un bistrot cosy à Bayonne, idéal pour déguster des...	<thead><tr><tr></thead></tr>	8	le_bistrot_sainte_cluque.jpg	6	4.0000
7	la petite auberge	15	rue des cordeliers	64100	Bayonne	43.4931	-1.4761	Un charmant restaurant proposant une cuisine du te...	<thead><tr><tr></thead></tr>	9	la_petite_auberge.jpg	7	4.5000
8	La table de POTTOKA	21	Quai Amiral Dubourdieu	64100	Bayonne	43.4898	-1.4742	Une table gastronomique à Bayonne, mettant en avan...	<thead><tr><tr></thead></tr>	10	la_table_de_pottoka.jpg	8	2.5000
9	La Rotisserie du Roy Léon	8	rue de coursic	64100	Bayonne	43.4939	-1.4767	Spécialisée dans les viandes rôties, cette rôtièr...	<thead><tr><tr></thead></tr>	11	rotisserie_du_roy_leon.jpg	9	4.0000

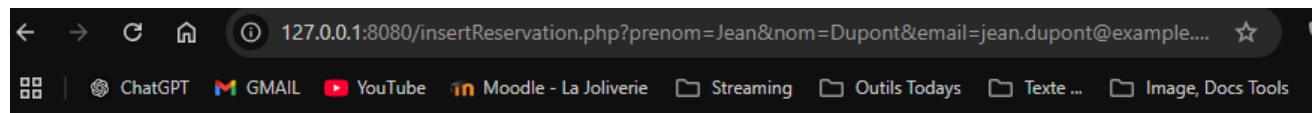
BDD

La quatrième méthode est insertReservation.php qui sert pour les reservation des restos et dans ce cas-ci, à ajouter une réservation d'un resto dans la bdd :

```

Projet_SiteRestoAndroid > API_Resto > insertReservation.php > ...
4  try {
5      // Vérification que la donnée est fournie
6      if (isset($_GET['nom'], $_GET['prenom'], $_GET['tel'], $_GET['email'], $_GET['date'], $_GET['heure'], $_GET['nbrPersonne'], $_GET['idResto'])) {
7          $nom = $_GET['nom'];
8          $prenom = $_GET['prenom'];
9          $tel = $_GET['tel'];
10         $email = $_GET['email'];
11         $date = $_GET['date']; // Format attendu : YYYY-MM-DD
12         $heure = $_GET['heure']; // Format attendu : HH:MM:SS
13         $nbrPersonne = $_GET['nbrPersonne'];
14         $idResto = $_GET['idResto'];
15
16         // Préparer la requête SQL
17         $reponse = $connexion->prepare(
18             query: "INSERT INTO reservation ('nom', 'prenom', 'tel', 'email', 'date', 'heure', 'nbrPersonne', 'idResto')
19                 VALUES (:nom, :prenom, :tel, :email, :dateR, :heure, :nbrPersonne, :idResto)"
20         );
21
22         // Lier les paramètres
23         $reponse->bindParam(param: ':nom', var: &$nom);
24         $reponse->bindParam(param: ':prenom', var: &$prenom);
25         $reponse->bindParam(param: ':tel', var: &$tel);
26         $reponse->bindParam(param: ':email', var: &$email);
27         $reponse->bindParam(param: ':dateR', var: &$date);
28         $reponse->bindParam(param: ':heure', var: &$heure);
29         $reponse->bindParam(param: ':nbrPersonne', var: &$nbrPersonne);
30         $reponse->bindParam(param: ':idResto', var: &$idResto);
31
32         // Exécuter la requête
33         if ($reponse->execute()) {
34             echo "Réservation enregistrée avec succès.";
35         } else {
36             echo "Erreur lors de l'insertion de la réservation.";
37         }
38     } else {
39         echo "Paramètres manquants.";
40     }
41 } catch (PDOException $e) {
42     echo "Erreur : " . $e->getMessage();
43 }
44 >>

```



Réservation enregistrée avec succès.

Test Méthode

2	Dupont	Jean	0612345678	jean.dupont@example.com	2024-12-24 19:30:00	4	1
---	--------	------	------------	-------------------------	---------------------	---	---

BDD

TÂCHE 08 : TEST FONCTIONNEL

TÂCHE 09 : DEPLOIEMENT AVEC DOCKER

Dans cette tâche bonus, afin de créer un environnement de test plus favorable et de préparer un éventuel déploiement sur un serveur à l'avenir, j'ai configuré l'application pour qu'elle puisse être testée via Docker. Cette solution permet aux testeurs de lancer simplement le conteneur et l'application Android sans avoir à effectuer les ajustements habituels, tels que modifier l'adresse IP des constructeurs pour l'API ou installer la base de données manuellement.

Voici ce que j'ai fait :

Tout d'abord j'ai fait un premier DockerFile destiné à l'API PHP

```
Projet_SiteRestoAndroid > Dockerfile > FROM
1 FROM php:8.1-apache
2
3 # Installer les extensions nécessaires pour PHP et MySQL
4 RUN docker-php-ext-install pdo pdo_mysql
5
6 # Copier l'API dans le conteneur /var/www/html/
7 COPY ./API_Resto /var/www/html/
8
9 EXPOSE 80
```

Ensuite j'ai fait docker compose, pour différencier les différents services de l'app. Ils sont au nombre de 3, mais en réalité seul 2 sont nécessaires :

- Service API →

```
services:
  php:
    build: . # Utilise le DockerFile
    depends_on:
      - db # Indique que le service PHP dépend du service db pour démarrer.
    ports:
      - "8080:80" # Expose le service PHP sur le port 8080
    env_file:
      - ./env # Utilise le fichier .env pour récupérer les var d'env
    environment:
      DB_HOST: ${BDD_HOST}
      DB_NAME: ${BDD_NAME}
      DB_USER: ${BDD_USER}
      DB_PASSWORD: ${BDD_PASSWORD}
    volumes:
      - ./API_Resto:/var/www/html
    networks:
      - resto_network
```

- Service base de données →

```
db:
  image: mysql:8.0
  restart: always # Redémarre le conteneur automatiquement en cas d'échec.
  env_file:
    - ./env
  environment:
    MYSQL_ROOT_PASSWORD: ${ROOT_PASSWORD}
    MYSQL_DATABASE: ${BDD_NAME}
    MYSQL_USER: ${BDD_USER}
    MYSQL_PASSWORD: ${BDD_PASSWORD}
  volumes:
    - db_data:/var/lib/mysql
    - ./sql:/docker-entrypoint-initdb.d
  command: --sql-mode="STRICT_TRANS_TABLES,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION"
  networks:
    - resto_network
```

```
phpmyadmin:
  image: phpmyadmin/phpmyadmin:latest
  restart: always
  ports:
    - "8081:80"
  env_file:
    - ./env
  environment:
    PMA_HOST: ${BDD_HOST}
    PMA_USER: ${BDD_USER}
    PMA_PASSWORD: ${BDD_PASSWORD}
  networks:
    - resto_network
```

J'ai essayé d'inclure un 3^{ème} service pour avoir l'application android sur un émulateur, mais cela n'a pas fonctionné et après y avoir passé beaucoup de temps j'ai préféré laisser les utilisateurs utiliser Android Studio en local

J'ai également suite à ceci modifier le mode opératoire de test disponible sur le README du projet, pour qu'il s'adapte à cette nouvelle option pour faciliter le déploiement

```

Projet_SiteRestoAndroid > README.md > # Application Android de Réservation de Restaurants > ## Mode opératoire de test > ### Installation de l'application >
14
15  ## **Mode opératoire de test**
16
17  ### **Installation de l'application**
18  ### **Pré-requis :**
19  - Un terminal Android version **10** ou supérieure.
20  - Une connexion internet pour accéder à l'API REST.
21  - Docker et Docker Compose installés sur votre machine.
22
23  ### **Étapes d'installation :**
24  1. **Cloner le projet :**
25  | - Clonez ce dépôt sur votre machine locale à l'aide des commande
26  |   -> git clone https://gitlab.com/KKG_Ambush/p2_g9_siteresto2024.git
27  |   -> cd P2_G9_SiteResto2024"
28  2. **Lancement du service Docker :**
29  | - Dans le répertoire racine de votre projet, exécutez la commande suivante pour lancer le service docker :
30  |   -> docker-compose up --build
31  | - Cela va construire et démarrer deux services :
32  |   -> api_resto : la partie DAO qui fait le lien entre l'app et la base de donnée accessible sur ce port (localhost:8080)
33  |   -> db : une base de donnée MySQL accessible sur ce port (localhost:8081)
34
35  3. **Lancement de l'application Android :**
36  | - Connectez vous à un terminal Android ou un émulateur et lancez l'application.

```

```

PS D:\Etu\BTS 1SIO\Informatique\Projet\Projet_SiteRestoAndroid> docker compose up
time="2024-12-15T21:28:18+01:00" level=warning msg="D:\\Etu\\BTS 1SIO\\Informatique\\Projet\\Projet_SiteRestoAndroid\\docker-compose.yml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential confusion"
[*] Running /4/4
✔Network projet_siterestoandroid_resto_network Created 0.1s
✔Container projet_siterestoandroid-phpmyadmin-1 Created 0.2s
✔Container projet_siterestoandroid-db-1 Created 0.2s
✔Container projet_siterestoandroid-php-1 Created 0.1s
Attaching to db-1, php-1, phpmyadmin-1
db-1 | 2024-12-15 21:28:20+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.0.40-1.el9 started.
phpmyadmin-1 | AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.19.0.3. Set the 'ServerName' directive globally to suppress this message
phpmyadmin-1 | AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.19.0.3. Set the 'ServerName' directive globally to suppress this message
phpmyadmin-1 | [Sun Dec 15 21:28:20.521479 2024] [mpm_prefork:notice] [pid 1] AH00163: Apache/2.4.57 (Debian) PHP/8.1.31 configured -- resuming normal operations
phpmyadmin-1 | [Sun Dec 15 21:28:20.521560 2024] [core:notice] [pid 1] AH00094: Command line: 'apache2 -D FOREGROUND'
db-1 | 2024-12-15 21:28:20+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'
db-1 | 2024-12-15 21:28:20+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.0.40-1.el9 started.
db-1 | '/var/lib/mysql/mysql.sock' -> '/var/run/mysqld/mysqld.sock'
php-1 | AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.19.0.4. Set the 'ServerName' directive globally to suppress this message
php-1 | AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.19.0.4. Set the 'ServerName' directive globally to suppress this message
php-1 | [Sun Dec 15 21:28:20.872257 2024] [mpm_prefork:notice] [pid 1] AH00163: Apache/2.4.62 (Debian) PHP/8.1.31 configured -- resuming normal operations
php-1 | [Sun Dec 15 21:28:20.873365 2024] [core:notice] [pid 1] AH00094: Command line: 'apache2 -D FOREGROUND'
db-1 | 2024-12-15T21:28:21.875872Z 0 [Warning] [MY-011868] [Server] The syntax '--skip-host-cache' is deprecated and will be removed in a future release. Please use SET GLOBAL host_cache_size=0 instead.
db-1 | 2024-12-15T21:28:21.875137Z 0 [Warning] [MY-010915] [Server] 'NO_ZERO_DATE', 'NO_ZERO_IN_DATE' and 'ERROR_FOR_DIVISION_BY_ZERO' sql modes should be used with strict mode. They will be merged with strict mode in a future release.
db-1 | 2024-12-15T21:28:21.883925Z 0 [System] [MY-010116] [Server] /usr/sbin/mysqld (mysqld 8.0.40) starting as process 1
db-1 | 2024-12-15T21:28:21.926570Z 1 [System] [MY-013576] [InnoDB] InnoDB initialization has started.
db-1 | 2024-12-15T21:28:22.085847Z 1 [System] [MY-013577] [InnoDB] InnoDB initialization has ended.
db-1 | 2024-12-15T21:28:23.084960Z 0 [Warning] [MY-010068] [Server] CA certificate ca.pem is self signed.
db-1 | 2024-12-15T21:28:23.085849Z 0 [System] [MY-013602] [Server] Channel mysql_main configured to support TLS. Encrypted connections are now supported for this channel.
db-1 | 2024-12-15T21:28:23.091578Z 0 [Warning] [MY-011810] [Server] Insecure configuration for --pid-file: Location '/var/run/mysqld' in the path is accessible to all OS users. Consider choosing a different directory.
db-1 | 2024-12-15T21:28:23.145749Z 0 [System] [MY-011323] [Server] X Plugin ready for connections. Bind-address: '::' port: 33060, socket: /var/run/mysqld/mysqld.sock
db-1 | 2024-12-15T21:28:23.145916Z 0 [System] [MY-010931] [Server] /usr/sbin/mysqld: ready for connections. Version: '8.0.40' socket: '/var/run/mysqld/mysqld.sock' port: 3306 MySQL Community Server - GPL.

```

Lancement de l'app avec Docker

TÂCHE 10 : CREATION D'UN DOC UML, DIAGRAMME DE CLASSE... (OPTION)

RESULTAT FINAL :

Liste des restaurants

Rechercher un restaurant

nom : entrepote, ville : Bordeaux

nom : le bar du charcutier, ville : Bordeaux

nom : Sapporo, ville : Bordeaux

nom : Cidrerie du fronton, ville : Arbonne

nom : Agadir, ville : Bayonne

nom : Le Bistrot Sainte Cluque, ville : Bayonne

nom : la petite auberge, ville : Bayonne

nom : La table de POTTOKA, ville : Bayonne

nom : La Rotisserie du Roy Léon, ville : Bayonne

nom : Bar du Marché, ville : Bayonne

nom : Trinquet Moderne, ville : Bayonne

Liste Restaurant

Trinquet Moderne

Description :

Localisation :

Horaire :

Ouverture

Semaine

Week-end

Un lieu emblématique de Bayonne, combinant sport et gastronomie basque pour une expérience unique.

Adresse : 60 Avenue Dubrocq, Bayonne 64100

Latitude : 43.4933 Longitude : -1.4729

Midi de 11h45 à 14h30

Soir de 18h45 à 22h30

À emporter de 11h30 à 23h

Réserver

Réservez maintenant

Détail Restaurant

Réserver un Restaurant

Thibault

Grimaud

tgrimaud@joliverie.com

01 02 03 04 05

2

Date de réservation

2024-12-15

Heure de réservation

Sélectionner une heure

Réserver

Reservez maintenant

Réservation Restaurant

numReservation	nom	prenom	tel	email	date	heure	nbrPersonne	idResto
1	Grimaud	Thibault	01 02 03 04 05	tgrimaud@joliverie.com	2024-12-15	19:49:18	2	11

Remplissage de la bdd avec réservation

DAILY SCRUM DEBUT DE L'IT3 (09/12/2024) :

GitLab : https://gitlab.com/KKG_Ambussh/p2_g9_siteresto2024.git

Drive avec documentation : [.Compte-Rendu RestoAndroid](#)

- La branche correspondante à l'itération sur le dépôt est celle avec son numéro, si celle-ci n'est pas présente alors elle correspond au MAIN
- Le README possède le mode opératoire de test

Répartition des tâches finalement :

Thibault à Tâche 4, 6, 7, 9

Pierre à Tâche 3, 8, 10

Thomas à Tâche 1, 2, 5

Bilan (problématique rencontré et ce qui a été réalisé) :

Pour cette fin de Projet et d'IT3 nous avons réussi à finir le projet entièrement, avec la liste des restaurant, le détail, l'interface pour réserver. Nous sommes tous très satisfait du résultat final au niveau esthétique, fonctionnel.

A l'avenir sur cette application on pourra faire des améliorations pour la rendre meilleur, par exemple cela peu passé par l'ajout d'autre information des restos dans le détail, par l'amélioration de l'esthétisme de la liste des restos avec des images et les notes ou encore par l'ajout de fonction pour supprimer une réservation, s'authentifier, ...

Cette fin de projet aura été compliqué et réalisé sur le bout du fil. Avec aujourd'hui, le rendu fini mais avec un retard uniquement sur le test fonctionnel du côté de pierre et son compte-rendu UML qui n'a pas peu être fait mais est disponible sur le lien OneDrive du projet GIT

Tableau de Kanban de fin :

