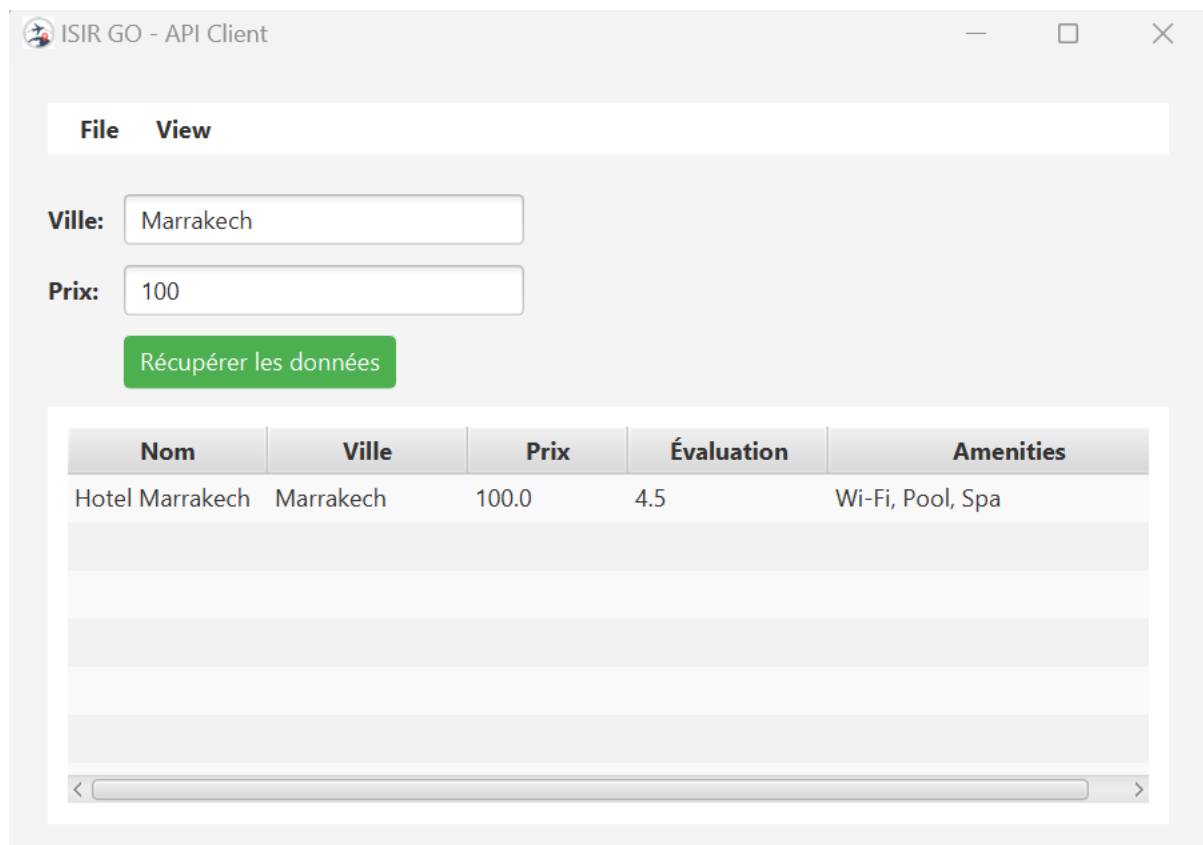


## TP10 WEB SERVICES -RESTFul-

- ① **L'Objectif :** créer un service web d'hôtellerie pour voir les valables dans certaines conditions :



The screenshot shows a web application titled "ISIR GO - API Client". It features a search form with two input fields: "Ville:" (containing "Marrakech") and "Prix:" (containing "100"). Below these fields is a green button labeled "Récupérer les données". Below the button is a table with the following data:

Nom	Ville	Prix	Évaluation	Amenities
Hotel Marrakech	Marrakech	100.0	4.5	Wi-Fi, Pool, Spa

- ② **Environnement & Languages :**

- (a) Outils de developpement :
  - Eclipse
- (b) Languages utilisées :
  - Jersey
  - JavaFX
- (c) Environnement :
  - le gestionnaire de dépendance maven
  - Tomcat

**Avertissement:** apres de voir le manuel de configuration et de tester votre 1 ere projet

### ③ Implémentation :

#### (a) organise l'architecture de l'application

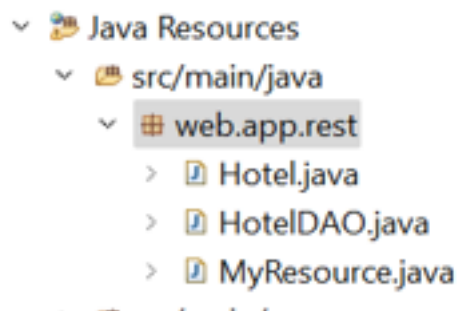
##### MVC

- Le MVC(Model-View-Controller) est un design pattern
- Permet d'avoir un code bien structuré
- Le Modèle : Chargé de la discussion avec la base de données
- La Vue : Affichage et présentation de l'interface
- Le Contrôleur : Agit comme intermédiaire entre le Modèle et la Vue.

##### DAO

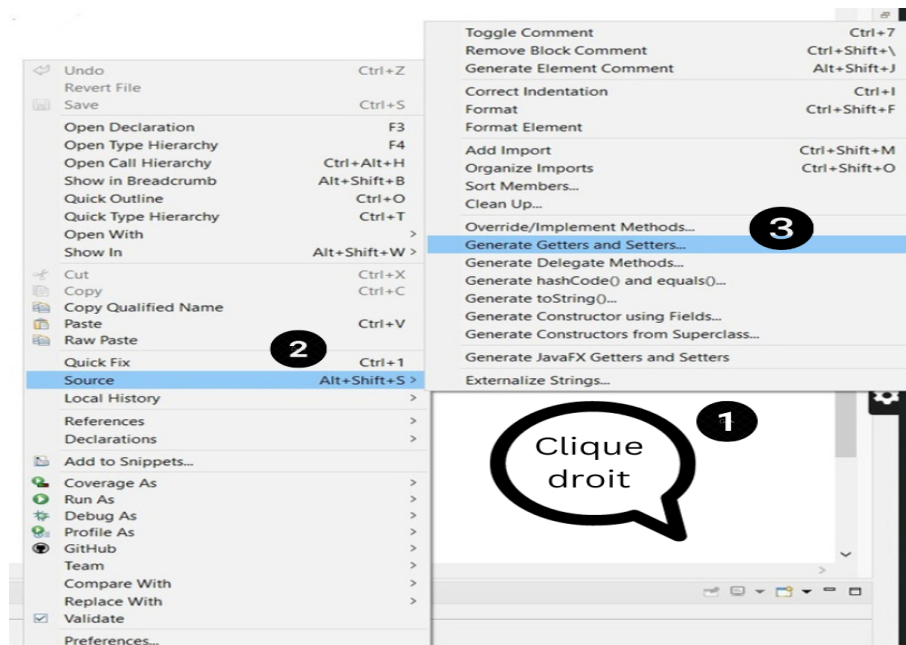
- Le DAO (Data Access Object) est un design pattern :
- Il a pour objectif de séparer la logique d'accès aux données de la logique métier de l'application.
- Il centralise toutes les opérations de lecture et d'écriture de données vers et depuis la base de données.
- Réduire la duplication de code liée à l'accès à la base de données en centralisant ces opérations.

- (b) Pour ce faire, on doit respecter les design patterns MVC et DAO, conformément aux bonnes pratiques, et notre projet doit être organisé comme suit



- i) Définir la classe Hotel qui représente un hôtel.
- ii) Définir les attributs d'un hôtel.
  - L'identifiant unique de l'hôtel.
  - Le nom de l'hôtel.
  - La ville où se situe l'hôtel.
  - Le prix moyen d'une nuitée dans l'hôtel.
  - La note moyenne attribuée à l'hôtel par les clients.
  - La liste des services offerts par l'hôtel.
- iii) Définir le constructeur par défaut de la classe Hotel.
- iv) Définir le constructeur paramétré de la classe Hotel.
- v) Définir les getters et les setters pour les attributs de la classe Hotel.

**Avertissement:** pour gérer les getters et les setters il suffit de



```

1 package web.app.rest;
2
3 //Importer les classes n cessaires
4 import java.util.ArrayList;
5 import java.util.List;
6
7 //D finir la classe Hotel qui repr sente un h tel
8 public class Hotel {
9
10 // D finir les attributs d'un h tel
11 private int id; // L'identifiant unique de l'h tel
12 private String name; // Le nom de l'h tel
13 private String city; // La ville o se situe l'h tel
14 private double price; // Le prix moyen d'une nuit e dans l'h tel
15 private double rating; // La note moyenne attribu e l'h tel par les
    clients
16 private List<String> amenities; // La liste des services offerts par l'h tel
17
18 // D finir le constructeur par d faut de la classe Hotel
19 public Hotel() {
20     // Initialiser les attributs avec des valeurs par d faut
21     this.id = 0;
22     this.name = "";
23     this.city = "";
24     this.price = 0.0;
25     this.rating = 0.0;
26     this.amenities = new ArrayList<>();
27 }
28
29 // D finir le constructeur param tr de la classe Hotel
30 public Hotel(int id, String name, String city, double price, double rating,
    List<String> amenities) {
31     // Initialiser les attributs avec les valeurs pass es en param tre
32     this.id = id;
33     this.name = name;
34     this.city = city;
35     this.price = price;
36     this.rating = rating;
37     this.amenities = amenities;
38 }
39
40 // D finir les getters et les setters pour les attributs de la classe Hotel

```

```

41 public int getId() {
42     return id;
43 }
44
45 public void setId(int id) {
46     this.id = id;
47 }
48
49 public String getName() {
50     return name;
51 }
52
53 public void setName(String name) {
54     this.name = name;
55 }
56
57 public String getCity() {
58     return city;
59 }
60
61 public void setCity(String city) {
62     this.city = city;
63 }
64
65 public double getPrice() {
66     return price;
67 }
68
69 public void setPrice(double price) {
70     this.price = price;
71 }
72
73 public double getRating() {
74     return rating;
75 }
76
77 public void setRating(double rating) {
78     this.rating = rating;
79 }
80
81 public List<String> getAmenities() {
82     return amenities;
83 }
84
85 public void setAmenities(List<String> amenities) {
86     this.amenities = amenities;
87 }
88 }

```

Listing 1: Hotel class

(a) **HotelDAO :**

- i) Définir la classe HotelDAO qui gère l'accès aux données des hôtels.
- ii) Définir une liste qui simule une base de données des hôtels.
- iii) Définir le constructeur de la classe HotelDAO.
- iv) Définir une méthode qui renvoie la liste des hôtels dans une ville.
- v) Définir une méthode qui renvoie les hôtels

```

1 package web.app.rest;
2
3 import java.util.ArrayList;
4 import java.util.List;
5
6 //D finir la classe HotelDAO qui g re l'acc s aux donn es des h tels
7 public class HotelDAO {
8
9     // D finir une liste qui simule une base de donn es des h tels
10    private List<Hotel> hotels;
11
12    // D finir le constructeur de la classe HotelDAO
13    public HotelDAO() {
14        // Initialiser la liste des h tels avec quelques exemples
15        hotels = new ArrayList<>();
16        hotels.add(new Hotel(1, "Hotel Marrakech", "Marrakech", 100.0, 4.5, List.
of("Wi-Fi", "Pool", "Spa")));
17        hotels.add(new Hotel(2, "Hotel Paris", "Paris", 150.0, 4.0, List.of("Wi-
Fi", "Parking", "Gym")));
18        hotels.add(new Hotel(3, "Hotel London", "London", 200.0, 3.5, List.of("Wi
-Fi", "Breakfast", "")));
19    }
20
21    // D finir une m thode qui renvoie la liste des h tels dans une ville
22    public List<Hotel> getHotelsByCity(String city, double price) {
23        // Cr er une liste vide pour stocker les h tels trouv s
24        List<Hotel> result = new ArrayList<>();
25        // Parcourir la liste des h tels
26        for (Hotel hotel : hotels) {
27            // V rifier si l'h tel se situe dans la ville recherch e
28            if (hotel.getCity().equalsIgnoreCase(city)&& hotel.getPrice()<=price)
{
29                // Si oui, ajouter l'h tel la liste des r sultats
30                result.add(hotel);
31            }
32        }
33        // Renvoyer la liste des r sultats
34        return result;
35    }
36
37    public List<Hotel> getAllHotels() {
38        List<Hotel> allHotels = new ArrayList<>();
39
40        // Parcourir la liste des h tels
41        for (Hotel hotel : hotels) {
42            // Ajouter chaque h tel la liste
43            allHotels.add(hotel);
44        }
45
46        // Renvoyer la liste compl te des h tels
47        return allHotels;
48    }
49
50    // D finir une m thode qui renvoie un h tel par son identifiant
51    public Hotel getHotelById(int id) {
52        // Parcourir la liste des h tels
53        for (Hotel hotel : hotels) {
54            // V rifier si l'h tel a l'identifiant recherch
55            if (hotel.getId() == id) {
56                // Si oui, renvoyer l'h tel
57                return hotel;
58            }
59        }

```

```

60 // Si aucun h tel n'a t trouv , renvoyer null
61 return null;
62 }
63
64 }

```

Listing 2: DAO class

(b) **Ressource :**

- i) Définir le chemin de base du service web.
- ii) Définir un objet qui gère l'accès aux données des hôtels.

```

1 package web.app.rest;
2
3 //Importer les classes n cessaires
4
5
6 import jakarta.ws.rs.*;
7 import jakarta.ws.rs.core.MediaType;
8 import jakarta.ws.rs.core.Response;
9
10 import java.util.List;
11
12 //D finir le chemin de base du service web
13 @Path("hotels")
14 public class MyResource {
15
16 // D finir un objet qui g re l'acc s aux donn es des h tels
17 private HotelDAO hotelDAO = new HotelDAO();
18
19 // D finir une m thode qui r pond une requ te HTTP GET
20 // et qui renvoie la liste des h tels dans une ville
21 @GET
22 @Path("/{city}")
23 @Consumes(MediaType.APPLICATION_JSON)
24 @Produces(MediaType.APPLICATION_JSON)
25 public Response getHotelsByCityAndPrice(
26     @PathParam("city") String city,
27     @QueryParam("price") double price) {
28     // R cup rer la liste des h tels dans la ville avec le prix sp cifi
29     List<Hotel> hotels = hotelDAO.getHotelsByCity(city, price);
30     // Construire la r ponse HTTP avec le code 200 (OK) et le format JSON
31     return Response.ok(hotels).build();
32 }
33
34
35 // D finir une m thode qui r pond une requ te HTTP GET
36 // et qui renvoie les d tails d'un h tel par son identifiant
37 @GET
38 @Path("/all")
39 @Produces(MediaType.APPLICATION_JSON)
40 public Response getHotels() {
41     // R cup rer l'h tel par son identifiant
42     List<Hotel> hotel = hotelDAO.getAllHotels();
43     // V rifier si l'h tel existe
44     if (hotel == null) {
45         // Si les h tels n'existent pas, renvoyer le code 404 (Not Found)
46         return Response.status(Response.Status.NOT_FOUND).build();
47     } else {
48         // Si les h tels existent, renvoyer le code 200 (OK) et le format
JSON

```

```

49         return Response.ok(hotel).build();
50     }
51 }
52
53 }

```

Listing 3: Ressource implementation

(c) l'interprétation des résultats a partir le service web :

```

1     package application;
2
3     import javafx.application.Application;
4     import javafx.beans.property.SimpleObjectProperty;
5     import javafx.collections.FXCollections;
6     import javafx.collections.ObservableList;
7     import javafx.geometry.Insets;
8     import javafx.scene.Scene;
9     import javafx.scene.control.*;
10    import javafx.scene.image.Image;
11    import javafx.scene.layout.GridPane;
12    import javafx.scene.layout.VBox;
13    import javafx.stage.Stage;
14    import org.json.JSONArray;
15    import org.json.JSONObject;
16
17    import java.io.BufferedReader;
18    import java.io.InputStreamReader;
19    import java.net.HttpURLConnection;
20    import java.net.URL;
21    import java.util.List;
22    import java.util.stream.Collectors;
23
24    public class Main extends Application {
25
26        private TableView<JSONObject> tableView;
27        private ObservableList<JSONObject> data = FXCollections.
observableArrayList();
28
29        public static void main(String[] args) {
30            launch(args);
31        }
32
33        @Override
34        public void start(Stage primaryStage) {
35
36            Image icon = new Image(getClass().getResource("icon.png").toExternalForm
());
37            primaryStage.getIcons().add(icon);
38            primaryStage.setTitle("ISIR GO - API Client");
39            // si vous utiliser un chemin absolu fait sinn relative comme nous avonz
fait.
40            //     Image icon = new Image("file:/chemin/absolu/vers/votre/projet/icon.
png");
41            //     primaryStage.getIcons().add(icon);
42
43
44            // Layout
45            VBox vbox = new VBox();
46            vbox.setSpacing(10);
47            vbox.setPadding(new Insets(20, 20, 20, 20));
48
49            // Menu

```

```

50     MenuBar menuBar = new MenuBar();
51     Menu fileMenu = new Menu("File");
52     MenuItem exitMenuItem = new MenuItem("Exit");
53     exitMenuItem.setOnAction(e -> System.exit(0));
54     fileMenu.getItems().add(exitMenuItem);
55
56     Menu viewMenu = new Menu("View");
57     MenuItem allHotelsMenuItem = new MenuItem("All Hotels");
58     allHotelsMenuItem.setOnAction(e -> fetchAllHotels());
59     viewMenu.getItems().add(allHotelsMenuItem);
60
61     menuBar.getMenus().addAll(fileMenu, viewMenu);
62
63     // Formulaire de connexion
64     GridPane loginForm = new GridPane();
65     loginForm.setHgap(10);
66     loginForm.setVgap(10);
67
68     TextField apiUrlField = new TextField("http://localhost:8080/rest/
webapi/hotels/");
69     TextField cityField = new TextField();
70     TextField priceField = new TextField();
71     Button fetchButton = new Button("R cup rer les donn es");
72
73     //
74     //     loginForm.add(new Label("API URL:"), 0, 0);
75     loginForm.add(new Label("Ville:"), 0, 1);
76     loginForm.add(cityField, 1, 1);
77     loginForm.add(new Label("Prix:"), 0, 2);
78     loginForm.add(priceField, 1, 2);
79     loginForm.add(fetchButton, 1, 3);
80
81     // Action du bouton
82     fetchButton.setOnAction(e -> {
83         String apiUrl = apiUrlField.getText();
84         String city = cityField.getText();
85         String priceStr = priceField.getText();
86
87         // Valider que le prix est un nombre
88         double price;
89         try {
90             price = Double.parseDouble(priceStr);
91         } catch (NumberFormatException ex) {
92             System.out.println("Le prix doit tre un nombre.");
93             return;
94         }
95
96         // R cup rer les donn es de l'API
97         JSONArray apiData = getHotelDataFromAPI(apiUrl, city, price);
98
99         // Mettre jour la liste de donn es
100        updateDataList(apiData);
101
102        // Mettre jour la TableView
103        tableView.setItems(data);
104    });
105
106    // Ajout du menu et du formulaire la VBox
107    vbox.getChildren().addAll(menuBar, loginForm);
108
109    // Cr er et configurer la TableView
110    tableView = createTableView();
111

```



```

112 // Ajout de la TableView la VBox
113 vbox.getChildren().add(tableView);
114
115 Scene scene = new Scene(vbox, 600, 400);
116 scene.getStylesheets().add(getClass().getResource("application.css").
toExternalForm());
117 primaryStage.setScene(scene);
118 primaryStage.show();
119 }
120
121 // Methode pour recuperer les donnees de l'API
122 private JSONArray getHotelDataFromAPI(String apiUrl, String city, double
price) {
123     JSONArray result = new JSONArray();
124     try {
125         String fullUrl = apiUrl + city + "?price=" + price;
126         URL url = new URL(fullUrl);
127         HttpURLConnection connection = (HttpURLConnection) url.
openConnection();
128         connection.setRequestMethod("GET");
129
130         try (BufferedReader reader = new BufferedReader(new
InputStreamReader(connection.getInputStream())) {
131             StringBuilder response = new StringBuilder();
132             String line;
133
134             while ((line = reader.readLine()) != null) {
135                 response.append(line);
136             }
137
138             // Convertir la rponse en tableau JSON
139             result = new JSONArray(response.toString());
140         }
141     } catch (Exception e) {
142         e.printStackTrace();
143     }
144     return result;
145 }
146
147 private TableView<JSONObject> createTableView() {
148     TableView<JSONObject> tableView = new TableView<>();
149
150     // Dfinir les colonnes
151     TableColumn<JSONObject, String> nameCol = new TableColumn<>("Nom");
152     nameCol.setCellValueFactory(cellData -> new SimpleObjectProperty<>(
cellData.getValue().getString("name")));
153     nameCol.getStyleClass().add("column-name");
154
155     TableColumn<JSONObject, String> cityCol = new TableColumn<>("Ville");
156     cityCol.setCellValueFactory(cellData -> new SimpleObjectProperty<>(
cellData.getValue().getString("city")));
157     cityCol.getStyleClass().add("column-city");
158
159     TableColumn<JSONObject, Number> priceCol = new TableColumn<>("Prix");
160     priceCol.setCellValueFactory(cellData -> new SimpleObjectProperty<>(
cellData.getValue().getDouble("price")));
161     priceCol.getStyleClass().add("column-price");
162
163     TableColumn<JSONObject, Number> ratingCol = new TableColumn<>("
valuation ");
164     ratingCol.setCellValueFactory(cellData -> new SimpleObjectProperty<>(
cellData.getValue().getDouble("rating")));
165     ratingCol.getStyleClass().add("column-rating");

```

```

166     TableColumn<JSONObject, String> amenitiesCol = new TableColumn<>("
167     Amenities");
168     amenitiesCol.setCellValueFactory(cellData -> {
169         List<String> amenities = cellData.getValue()
170             .getJSONArray("amenities")
171             .toList()
172             .stream()
173             .map(Object::toString)
174             .collect(Collectors.toList());
175         return new SimpleObjectProperty<>(String.join(", ", amenities));
176     });
177     amenitiesCol.getStyleClass().add("column-amenities");
178
179     // Ajouter les colonnes la TableView
180     tableView.getColumns().addAll(nameCol, cityCol, priceCol, ratingCol,
181     amenitiesCol);
182
183     return tableView;
184 }
185
186 // Mettre jour la liste de donn es
187 private void updateDataList(JSONArray apiData) {
188     data.clear();
189
190     for (int i = 0; i < apiData.length(); i++) {
191         data.add(apiData.getJSONObject(i));
192     }
193
194     // Methode pour recuperer et afficher tous les hotels
195     private void fetchAllHotels() {
196         // Recuperer les donn es de l'API pour tous les hotels
197         JSONArray apiData = getHotelDataFromAPI("http://localhost:8080/rest/
198         webapi/hotels/all", "", 0);
199
200         // Mettre jour la liste de donn es
201         updateDataList(apiData);
202
203         // Mettre jour la TableView
204         tableView.setItems(data);
205     }
206 }

```