Application.java

package com.example.client;
import javafx.fxml.FXMLLoader;
import javafx.scene.Scene;
import javafx.scene.layout.AnchorPane;
import javafx.stage.Stage;
public class Application extends javafx.application.Application {
public static void main(String[] args) {
launch(args);
}
@Override
public void start(Stage primaryStage) {
String hostName = "localhost"; // путь к серверу
int portNumber = 5000;
try {
AnchorPane root = FXMLLoader.load(getClass().getResource("start.fxml"));
//Parent root = FXMLLoader.load(getClass().getResource("start.fxml"));
primaryStage.setTitle("Электронное меню");
primaryStage.setWidth(910); // Установка ширины окна
primaryStage.setHeight(630); // Установка высоты окна
Scene scene = new Scene(root);
primaryStage.setScene(scene);
//primaryStage.setFullScreen(true); // Устанавливаем полноэкранный режим
//primaryStage.setFullScreenExitHint(""); // Пустая строка, чтобы не отображался подсказка о выходе из полноэкранного режима
primaryStage.show();
//root.layoutXProperty().bind(scene.widthProperty().subtract(root.prefWidthProperty()).divide(2));
//root.layoutYProperty().bind(scene.heightProperty().subtract(root.prefHeightProperty()).divide(2));
} catch (Exception e) {
e.printStackTrace();
}
}
}

Client.java

package com.example.client.client;
import java.util.HashMap;
public class Client {
public static int tableNumber;
// выбранные блюда для заказа
// номер столика
public static String position = "client";
public static boolean accepted = false;
public static HashMap<Integer, Integer> selectedDishes = new HashMap<>(); // выбранные блюда - id блюда и количество
public static HashMap<Integer, Float> selectedDishesPrice = new HashMap<>(); // выбранные блюда - id блюда и цена
public static void clean() {
tableNumber = -1;
accepted = false;
selectedDishes.clear();
selectedDishesPrice.clear();
}
public static void addDish(int id, int count, float price) {
selectedDishes.put(id, count);
selectedDishesPrice.put(id, price);
}
public static void delete(int dish) {
selectedDishes.remove(dish);
selectedDishesPrice.remove(dish);
}
}

ClientAboutDishSceneController.java

package com.example.client.client;
import com.example.client.Application;
import com.example.client.MethodsSceneController;
import com.example.client.ServerCommunicator;
import com.example.client.items.Category;
import com.example.client.items.Dish;
import com.example.client.items.Ingredient;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.ScrollPane;
import javafx.scene.text.Text;
import javafx.stage.Stage;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Map;
import java.util.StringJoiner;
public class ClientAboutDishSceneController {
@FXML
private Button aboutRestaurat;
@FXML
private Text category;
@FXML
private Text description;
@FXML
private Button dishesList;
@FXML
private Text errorMessage;
@FXML
private Button exit;
@FXML
private Text ingredients;
@FXML
private Text name;
@FXML
private Text price;
@FXML
private ScrollPane scrollPane;
@FXML
private Button selectedDishes;
@FXML
private Text table;
@FXML
private Text telephone;
@FXML
private Text weight;
private Dish curDish;
private Stage stage;
public void start(Stage stage, Dish curDish) {
this.stage = stage;
table.setText("Стол: " + Integer.toString(Client.tableNumber));
int id = curDish.id;
Dish dish = ServerCommunicator.getDishClient(id);
ArrayList<Category> list = new ArrayList<>();
ServerCommunicator.getAllCategoriesClient(list);
if (dish == null) {
errorMessage.setText("Произошла ошибка");
} else {
curDish = dish;
// устанавливаю значения
name.setText(curDish.name);
description.setText(curDish.description);
//category.setValue();
category.setText("без категории");
for (Category categoryItem : list) {
if (categoryItem.id == curDish.id) {
category.setText(categoryItem.name);
}
}
price.setText(Float.toString(curDish.price));
weight.setText(Float.toString(curDish.weight));
// вывод сохранённых ингредиентов
showIngredients(dish);
}
}
private void showIngredients(Dish dish) {
StringJoiner joiner = new StringJoiner(", ");
for (Map.Entry<Integer, Ingredient> entry : dish.ingredientsList.entrySet()) {
Ingredient value = entry.getValue();
joiner.add(value.simpleName);
}
ingredients.setText(joiner.toString());
}
@FXML
void aboutRestauratButtonClicked(ActionEvent event) {
errorMessage.setText("");
Scene currentScene = errorMessage.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("clientRestaurant.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
ClientRestaurantSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
@FXML
void dishesListButtonClicked(ActionEvent event) {
errorMessage.setText("");
Scene currentScene = errorMessage.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("clientDishes.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
ClientDishesSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
@FXML
void exitButtonClicked(ActionEvent event) {
errorMessage.setText("");
MethodsSceneController.logoutAction(stage);
}
@FXML
void selectedDishesButtonClicked(ActionEvent event) {
errorMessage.setText("");
if (!Client.accepted) {
Scene currentScene = errorMessage.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("clientChoose.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
ClientChooseSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
}
}

ClientAcceptedSceneController.java

ClientChooseSceneController.java

ClientDishesSceneController.java

ClientLoginSceneController.java

package com.example.client.client;
import com.example.client.Application;
import com.example.client.ServerCommunicator;
import com.example.client.employee.editor.EditorSuppliesMenuSceneController;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.TextField;
import javafx.stage.Stage;
import javafx.scene.text.Text;
import java.io.IOException;
public class ClientLoginSceneController {
@FXML
private Button enter;
@FXML
private Button exit;
@FXML
private TextField tableNumber;
@FXML
private Text errorMessage;
private Stage stage;
public void start(Stage \_stage) {
this.stage = \_stage;
tableNumber.setPromptText("Введите номер стола");
tableNumber.textProperty().addListener((observable, oldValue, newValue) -> {
if (!newValue.matches("\\d\*")) {
tableNumber.setText(newValue.replaceAll("[^\\d]", ""));
}
});
}
@FXML
void enterButtonClicked(ActionEvent event) {
errorMessage.setText("");
if (tableNumber.getText().isEmpty() || (Integer.parseInt(tableNumber.getText()) < 0)) {
errorMessage.setText("Введите номер корректный стола");
} else {
if (ServerCommunicator.client().equals("success")) {
} else {
errorMessage.setText("Произошла ошибка.");
return;
}
Client.tableNumber = Integer.parseInt(tableNumber.getText());
Scene currentScene = errorMessage.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("clientDishes.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
ClientDishesSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
}
public void exitButtonClicked(ActionEvent actionEvent) {
errorMessage.setText("");
FXMLLoader loader = new FXMLLoader(Application.class.getResource("start.fxml"));
try {
Parent root = loader.load();
Scene newScene = new Scene(root);
stage.setScene(newScene);
} catch (IOException e) {
e.printStackTrace();
}
}
}

ClientRestaurantSceneController.java

package com.example.client.client;
import com.example.client.Application;
import com.example.client.MethodsSceneController;
import com.example.client.ServerCommunicator;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.ScrollPane;
import javafx.scene.text.Text;
import javafx.stage.Stage;
import java.io.IOException;
import java.util.HashMap;
public class ClientRestaurantSceneController {
@FXML
private Button aboutRestaurat;
@FXML
private Text address;
@FXML
private Text description;
@FXML
private Button dishesList;
@FXML
private Text errorMessage;
@FXML
private Button exit;
@FXML
private Text name;
@FXML
private ScrollPane scrollPane;
@FXML
private Button selectedDishes;
@FXML
private Text table;
@FXML
private Text telephone;
private Stage stage;
public void start(Stage stage) {
this.stage = stage;
table.setText("Стол: " + Integer.toString(Client.tableNumber));
HashMap<String, String> restInfo = ServerCommunicator.getRestaurantInfo();
name.setText(restInfo.get("name"));
description.setText(restInfo.get("description"));
telephone.setText(restInfo.get("telephone"));
address.setText(restInfo.get("address"));
}
@FXML
void aboutRestauratButtonClicked(ActionEvent event) {
errorMessage.setText("");
Scene currentScene = errorMessage.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("clientRestaurant.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
ClientRestaurantSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
@FXML
void dishesListButtonClicked(ActionEvent event) {
errorMessage.setText("");
Scene currentScene = errorMessage.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("clientDishes.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
ClientDishesSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
@FXML
void exitButtonClicked(ActionEvent event) {
errorMessage.setText("");
MethodsSceneController.logoutAction(stage);
}
@FXML
void selectedDishesButtonClicked(ActionEvent event) {
errorMessage.setText("");
if (!Client.accepted) {
Scene currentScene = errorMessage.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("clientChoose.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
ClientChooseSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
}
}

ConfirmationBox.java

package com.example.client;
import javafx.scene.control.Alert;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.control.ButtonType;
import java.util.Optional;
public class ConfirmationBox {
private static boolean displayConfirmation(String title, String message) {
Alert alert = new Alert(AlertType.CONFIRMATION);
alert.setTitle(title);
alert.setHeaderText(null); // Отключает текст заголовка
alert.setContentText(message);
// Создание кнопок "Принять" и "Отмена"
ButtonType acceptButton = new ButtonType("Принять");
ButtonType cancelButton = new ButtonType("Отмена");
// Добавление кнопок в диалоговое окно
alert.getButtonTypes().setAll(acceptButton, cancelButton);
// Отображение окна и ожидание выбора
Optional<ButtonType> result = alert.showAndWait();
// Обработка выбора пользователя
if (result.isPresent() && result.get() == acceptButton) {
// Обработка нажатия кнопки "Принять"
//System.out.println("Принято");
return true;
} else {
// Обработка нажатия кнопки "Отмена" или закрытия окна
//System.out.println("Отменено");
return false;
}
}
public static boolean showAlertBox(String title, String message) {
return displayConfirmation(title, message);
}
}

AdminEmployeeAddSceneController.java

package com.example.client.employee.administrator;
import com.example.client.Application;
import com.example.client.ConfirmationBox;
import com.example.client.ServerCommunicator;
import com.example.client.employee.CurEmployee;
import com.example.client.employee.Employee;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.ComboBox;
import javafx.scene.control.DatePicker;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.text.Text;
import javafx.stage.Stage;
import java.io.IOException;
import java.time.LocalDate;
public class AdminEmployeeAddSceneController {
private Stage stage;
@FXML
private Button addEmployee;
@FXML
private Button employees;
@FXML
private Text errorMessage;
@FXML
private Button exit;
@FXML
private TextField fio;
@FXML
private TextField login;
@FXML
private ComboBox<String> position;
@FXML
private Button restaurantInfo;
@FXML
private Button save;
@FXML
private TextField telephone;
@FXML
private Text curEmployeeFIO;
@FXML
private TextField password;
@FXML
void addEmployeeButtonClicked(ActionEvent event) {
}
@FXML
void employeesButtonClicked(ActionEvent event) {
errorMessage.setText("");
if (!forgetChanges()) return;
Scene currentScene = addEmployee.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("adminMain.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
AdminMainSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
@FXML
void exitButtonClicked(ActionEvent event) {
errorMessage.setText("");
if (!forgetChanges()) return;
if (!ConfirmationBox.showAlertBox("Выход из аккаунта", "Вы уверены, что хотите выйти из аккаунта?")) {
return;
}
FXMLLoader loader = new FXMLLoader(Application.class.getResource("start.fxml"));
try {
Parent root = loader.load();
Scene newScene = new Scene(root);
stage.setScene(newScene);
} catch (IOException e) {
e.printStackTrace();
}
}
@FXML
void restaurantInfoButtonClicked(ActionEvent event) {
errorMessage.setText("");
if (!forgetChanges()) return;
Scene currentScene = addEmployee.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("adminRestaurantInfo.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
AdminRestaurantInfoSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
@FXML
void saveButtonClicked(ActionEvent event) {
errorMessage.setText("");
if (!ConfirmationBox.showAlertBox("Сохранение", "Вы уверены, что хотите сохранить изменения?")) {
return;
}
String fioValue = fio.getText();
if (fioValue.isEmpty()) {
errorMessage.setText("Заполните ФИО");
return;
}
String positionValue;
switch (position.getValue()) {
case "редактор": {
positionValue = "editor";
break;
}
case "администратор": {
positionValue = "administrator";
break;
}
default: {
positionValue = position.getValue();
}
}
String telephoneValue = telephone.getText();
String loginValue = login.getText();
String passwordValue = password.getText();
if (loginValue.isEmpty() || passwordValue.isEmpty()) {
errorMessage.setText("Логин и пароль должны быть установлены.");
return;
}
Employee newEmp = new Employee(fioValue, loginValue, passwordValue, positionValue, telephoneValue);
String message;
switch (ServerCommunicator.addEmployee(newEmp)) {
case "success": {
Scene currentScene = addEmployee.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("adminMain.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
AdminMainSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
return;
}
case "denied": {
message = "Отказано в доступе.";
break;
}
case "notUnicLogin": {
message = "Логин должен быть уникальным.";
break;
}
case "error": {
message = "Произошла ошибка.";
break;
}
default: message = "";
}
errorMessage.setText(message);
}
public boolean forgetChanges() {
if (!fio.getText().isEmpty() || !telephone.getText().isEmpty() || !login.getText().isEmpty() || !password.getText().isEmpty()) {
if (ConfirmationBox.showAlertBox("Предупреждение", "Вы уверены, что хотите завершить добавление сотрудника? Несохраненные данные будут потеряны.")) {
return true;
} else return false;
}
return true;
}
public void start(Stage stage) {
this.stage = stage;
curEmployeeFIO.setText(CurEmployee.getFio());
position.getItems().addAll("редактор", "администратор");
position.setValue("редактор");
}
}

AdminEmployeeEditSceneController.java

package com.example.client.employee.administrator;
import com.example.client.Application;
import com.example.client.ConfirmationBox;
import com.example.client.ServerCommunicator;
import com.example.client.employee.CurEmployee;
import com.example.client.employee.Employee;
import com.example.client.MethodsSceneController;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.ComboBox;
import javafx.scene.control.TextField;
import javafx.stage.Stage;
import javafx.scene.text.Text;
import java.io.IOException;
import java.util.HashMap;
public class AdminEmployeeEditSceneController {
private Stage stage;
private int id;
@FXML
private Button addEmployee;
@FXML
private Text curEmployeeFIO;
@FXML
private Button employees;
@FXML
private Text errorMessage;
@FXML
private Button exit;
@FXML
private TextField fio;
@FXML
private TextField login;
@FXML
private TextField password;
@FXML
private ComboBox<String> position;
@FXML
private Button restaurantInfo;
@FXML
private Button save;
@FXML
private TextField telephone;
public void start(int id, Stage \_stage) {
this.stage = \_stage;
curEmployeeFIO.setText(CurEmployee.getFio());
this.id = id;
// получение информации о сотруднике
HashMap<String, String> empMap = ServerCommunicator.getEmployeeById(id);
// заполнение полей
switch (empMap.get("message")) {
case "success": {
fio.setText(empMap.get("fio"));
telephone.setText(empMap.get("telephone"));
login.setText(empMap.get("login"));
position.setValue(empMap.get("position"));
break;
}
case "denied": {
errorMessage.setText("Отказано в доступе.");
break;
}
case "error": {
errorMessage.setText("Произошла ошибка.");
}
default: {
errorMessage.setText("Произошла ошибка. Попробуйте ещё раз");
return;
}
}
position.getItems().addAll("редактор", "администратор");
}
@FXML
void addEmployeeButtonClicked(ActionEvent event) {
errorMessage.setText("");
if (!forgetChanges()) return;
Scene currentScene = addEmployee.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("adminEmployeeAdd.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
AdminEmployeeAddSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
@FXML
void employeesButtonClicked(ActionEvent event) {
errorMessage.setText("");
if (!forgetChanges()) return;
Scene currentScene = addEmployee.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("adminMain.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
AdminMainSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
public boolean forgetChanges() {
return ConfirmationBox.showAlertBox("Предупреждение", "Вы уверены, что хотите завершить редактирование сотрудника? Несохраненные данные будут потеряны.");
}
@FXML
void exitButtonClicked(ActionEvent event) {
errorMessage.setText("");
MethodsSceneController.logoutAction(stage);
}
@FXML
void restaurantInfoButtonClicked(ActionEvent event) {
errorMessage.setText("");
if (!forgetChanges()) return;
Scene currentScene = addEmployee.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("adminRestaurantInfo.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
AdminRestaurantInfoSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
@FXML
void saveButtonClicked(ActionEvent event) {
errorMessage.setText("");
if (!ConfirmationBox.showAlertBox("Сохранение", "Вы уверены, что хотите сохранить изменения?")) {
return;
}
String positionValue;
switch (position.getValue()) {
case "редактор": {
positionValue = "editor";
break;
}
case "администратор": {
positionValue = "administrator";
break;
}
default: {
positionValue = position.getValue();
}
}
String fioValue = fio.getText();
String telephoneValue = telephone.getText();
String loginValue = login.getText();
String passwordValue = password.getText();
if (passwordValue.equals("Заполните поле, если хотите поменять пароль") || passwordValue.isEmpty()) {
passwordValue = null;
}
if (loginValue.isEmpty()) {
errorMessage.setText("Логин должен быть установлены.");
return;
}
switch (ServerCommunicator.setEmployee(id, fioValue, positionValue, telephoneValue, loginValue, passwordValue)) {
case "error": {
errorMessage.setText("Ошибка.");
break;
}
case "denied": {
errorMessage.setText("Отказано в доступе.");
break;
}
case "notUnicLogin": {
errorMessage.setText("Логин должен быть уникальным");
break;
}
case "noSuchItem": {
errorMessage.setText("Пользователь не найден.");
break;
}
case "success": {
errorMessage.setText("Данные успешно сохранены!");
break;
}
}
}
}

AdminMainSceneController.java

package com.example.client.employee.administrator;
import com.example.client.Application;
import com.example.client.ConfirmationBox;
import com.example.client.MethodsSceneController;
import com.example.client.ServerCommunicator;
import com.example.client.employee.CurEmployee;
import com.example.client.employee.Employee;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.ScrollPane;
import javafx.scene.layout.GridPane;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;
import javafx.scene.text.Text;
import java.io.IOException;
import java.util.ArrayList;
public class AdminMainSceneController {
private Stage stage;
//ArrayList<Employee> listOfEmployees = new ArrayList<>();
@FXML
private Button addEmployee;
@FXML
private Button employees;
@FXML
private VBox employeesContainer;
@FXML
private Button exit;
@FXML
private Text curEmployeeFIO;
@FXML
private Button restaurantInfo;
@FXML
private Text errorMessage;
@FXML
private ScrollPane listOfEmployees;
public void start(Stage \_stage) {
this.stage = \_stage;
// получаем фамилию сотрудника
curEmployeeFIO.setText(CurEmployee.getFio());
// выводим список всех сотрудников
showListOfEmployees();
}
public void showListOfEmployees() {
// try {
employeesContainer.getChildren().clear();
ArrayList<Employee> employeesList = new ArrayList<>();
// сообщение и список сотрудников
String message = ServerCommunicator.getEmployeesList(employeesList);
switch (message) {
case "error": {
errorMessage.setText("Произошла ошибка.");
break;
}
case "deny": {
errorMessage.setText("Недостаточно прав. Перезайдите в аккаунт.");
break;
}
default: {
// вывод всех сотрудников
if (employeesList == null) return;
for (Employee employeeItem : employeesList) {
GridPane gridPane = createGridPane(employeeItem.id, employeeItem.fio, employeeItem.position, employeeItem.telephone);
employeesContainer.getChildren().add(gridPane);
}
listOfEmployees.setFitToWidth(true);
listOfEmployees.setFitToHeight(true);
}
}
}
private GridPane createGridPane(int id, String fio, String position, String telephone) {
try {
FXMLLoader fxmlLoader = new FXMLLoader(Application.class.getResource("itemEmployee.fxml"));
GridPane gridPane = fxmlLoader.load();
// кнопка удаления (себя удалить нельзя)
if (id != CurEmployee.getId()) {
Button deleteEmployeeButton = new Button("Удалить");
deleteEmployeeButton.setOnAction(event -> {
if (ConfirmationBox.showAlertBox("Удаление аккаунта сотрудника", "Вы уверены, что хотите удалить сотрудника?")) {
errorMessage.setText(ServerCommunicator.deleteEmployeeById(id)); // сотрудник переносится в архив
showListOfEmployees();
}
});
GridPane.setConstraints(deleteEmployeeButton, 2, 0); // столбцы и строки
gridPane.getChildren().add(deleteEmployeeButton);
} else {
Label itIsYou = new Label("Это вы");
GridPane.setConstraints(itIsYou, 2, 0);
gridPane.getChildren().add(itIsYou);
}
Button configureEmployeeButton = new Button("Изменить");
configureEmployeeButton.setOnAction(event -> {
Scene currentScene = addEmployee.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("adminEmployeeEdit.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
AdminEmployeeEditSceneController controller = loader.getController();
controller.start(id, stage);
} catch (IOException e) {
e.printStackTrace();
}
});
GridPane.setConstraints(configureEmployeeButton, 2, 1);
gridPane.getChildren().add(configureEmployeeButton);
// ФИО
Label labelFio = new Label(fio);
GridPane.setConstraints(labelFio, 0, 0);
gridPane.getChildren().add(labelFio);
// должность
Label labelPosition = new Label(position);
GridPane.setConstraints(labelPosition, 1, 0);
gridPane.getChildren().add(labelPosition);
// телефон
Label labelTelephone = new Label(telephone);
GridPane.setConstraints(labelTelephone, 1, 1);
gridPane.getChildren().add(labelTelephone);
return gridPane;
} catch (IOException e) {
e.printStackTrace();
}
return null;
}
@FXML
void addEmployeeButtonClicked(ActionEvent event) {
errorMessage.setText("");
Scene currentScene = addEmployee.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("adminEmployeeAdd.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
AdminEmployeeAddSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
@FXML
void employeesButtonClicked(ActionEvent event) {
errorMessage.setText("");
Scene currentScene = addEmployee.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("adminMain.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
AdminMainSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
@FXML
void exitButtonClicked(ActionEvent event) {
errorMessage.setText("");
MethodsSceneController.logoutAction(stage);
}
@FXML
void restaurantInfoButtonClicked(ActionEvent event) {
errorMessage.setText("");
Scene currentScene = addEmployee.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("adminRestaurantInfo.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
AdminRestaurantInfoSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
}

AdminRestaurantInfoSceneController.java

package com.example.client.employee.administrator;
import com.example.client.Application;
import com.example.client.ConfirmationBox;
import com.example.client.MethodsSceneController;
import com.example.client.ServerCommunicator;
import com.example.client.employee.CurEmployee;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextArea;
import javafx.scene.control.TextField;
import javafx.scene.text.Text;
import javafx.stage.Stage;
import java.io.IOException;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.util.HashMap;
import java.util.Locale;
public class AdminRestaurantInfoSceneController {
private Stage stage;
@FXML
private Button addEmployee;
public boolean forgetChanges() {
return ConfirmationBox.showAlertBox("Предупреждение", "Вы уверены, что хотите завершить редактирование информации о ресторане? Несохраненные данные будут потеряны.");
}
@FXML
private TextField address;
@FXML
private TextArea description;
@FXML
private Button employees;
@FXML
private Button exit;
@FXML
private TextField name;
@FXML
private Button restaurantInfo;
@FXML
private Button save;
@FXML
private TextField telephone;
@FXML
private Text curEmployeeFIO;
@FXML
private Text errorMessage;
@FXML
void addEmployeeButtonClicked(ActionEvent event) {
errorMessage.setText("");
if (!forgetChanges()) return;
Scene currentScene = addEmployee.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("adminEmployeeAdd.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
AdminEmployeeAddSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
@FXML
void employeesButtonClicked(ActionEvent event) {
errorMessage.setText("");
if (!forgetChanges()) return;
Scene currentScene = addEmployee.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("adminMain.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
AdminMainSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
@FXML
void exitButtonClicked(ActionEvent event) {
errorMessage.setText("");
if (!forgetChanges()) return;
MethodsSceneController.logoutAction(stage);
}
@FXML
void restaurantInfoButtonClicked(ActionEvent event) {
errorMessage.setText("");
}
@FXML
void saveButtonClicked(ActionEvent event) {
errorMessage.setText("");
if (!ConfirmationBox.showAlertBox("Сохранение", "Вы уверены, что хотите сохранить изменения?")) {
return;
}
String nameValue = name.getText();
String descriptionValue = description.getText();
String telephoneValue = telephone.getText();
String addressValue = address.getText();
switch (ServerCommunicator.setRestaurantInfo(nameValue, descriptionValue, telephoneValue, addressValue)) {
case "error": {
errorMessage.setText("Ошибка.");
break;
}
case "denied": {
errorMessage.setText("Отказано в доступе.");
break;
}
case "success": {
errorMessage.setText("Данные успешно сохранены!");
break;
}
}
}
public void start(Stage stage) {
this.stage = stage;
curEmployeeFIO.setText(CurEmployee.getFio());
// загрузка инфы из базы
HashMap<String, String> restaurantInfo = ServerCommunicator.getRestaurantInfo();
// заполняю поля
switch (restaurantInfo.get("message")) {
case "success": {
name.setText(restaurantInfo.get("name"));
telephone.setText(restaurantInfo.get("telephone"));
description.setText(restaurantInfo.get("description"));
address.setText(restaurantInfo.get("address"));
break;
}
case "error": {
errorMessage.setText("Произошла ошибка.");
}
default: {
errorMessage.setText("Произошла ошибка. Попробуйте ещё раз");
}
}
}
}

AuthSceneController.java

package com.example.client.employee;
import com.example.client.Application;
import com.example.client.ServerCommunicator;
import com.example.client.StartSceneController;
import com.example.client.employee.administrator.AdminMainSceneController;
import com.example.client.employee.editor.EditorMainSceneController;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.PasswordField;
import javafx.scene.text.Text;
import javafx.stage.Stage;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.TextField;
import java.io.IOException;
public class AuthSceneController {
private StartSceneController startSceneController;
private Stage stage;
public void start(Stage \_stage) {
this.stage = \_stage;
}
@FXML
private Button enter;
@FXML
private Button exit;
@FXML
private TextField login;
@FXML
private PasswordField password;
@FXML
private Text errorMessage;
@FXML
void exitButtonClicked(ActionEvent event) {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("start.fxml"));
try {
Parent root = loader.load();
Scene newScene = new Scene(root);
stage.setScene(newScene);
} catch (IOException e) {
e.printStackTrace();
}
}
@FXML
void enterButtonClicked(ActionEvent event) {
String loginValue = login.getText();
String passwordValue = password.getText();
//CurEmployee.setLoginAndPass(loginValue, passwordValue)
String position = ServerCommunicator.authCheck(loginValue, passwordValue);
switch (position) {
case "editor": {
Scene currentScene = enter.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("editorMain.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
EditorMainSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
break;
}
case "administrator": {
Scene currentScene = enter.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("adminMain.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
AdminMainSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
break;
}
case "connectionError": {
errorMessage.setText("Произошла ошибка соединения. Попробуйте ещё раз.");
break;
}
case "failedAuth": {
errorMessage.setText("Несоответствие логина и пароля.");
break;
}
case "noSuchUser": {
errorMessage.setText("Пользователь с таким логином не найден!");
break;
}
default: {
errorMessage.setText("Произошла ошибка соединения. Отказано в соединении с сервером.");
break;
}
}
}
}

CurEmployee.java

package com.example.client.employee;
public class CurEmployee {
// public static String login;
// public static String password;
public static String position; // редактор или админ
public static String fio;
private static int id = -1;
private static String sessionKey;
// public static void setLoginAndPass(String \_login, String \_password) {
// login = \_login;
// password = \_password;
// }
public static void setPosition(String position) {
CurEmployee.position = position;
}
public static void setSessionKey(String sessionKey) {
CurEmployee.sessionKey = sessionKey;
}
// public static String getLogin() {
// return login;
// }
//
// public static String getPassword() {
// return password;
// }
public static void setFio(String fio) {
CurEmployee.fio = fio;
}
public static String getFio() {
return fio;
}
public static void setId(int id) {
CurEmployee.id = id;
}
public static int getId() {
return id;
}
public static String getSessionKey() {
return sessionKey;
}
public static void clean() {
// login = null;
// password = null;
position = null; // редактор или админ
fio = null;
sessionKey = null;
id = -1;
}
}

EditorCategoriesEditSceneController.java

package com.example.client.employee.editor;
import com.example.client.Application;
import com.example.client.ConfirmationBox;
import com.example.client.MethodsSceneController;
import com.example.client.ServerCommunicator;
import com.example.client.employee.CurEmployee;
import com.example.client.employee.Employee;
import com.example.client.items.Category;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.TextArea;
import javafx.scene.control.TextField;
import javafx.scene.text.Text;
import java.io.IOException;
public class EditorCategoriesEditSceneController {
private Stage stage;
private int id;
private int archived;
public void start(Stage stage, int id, String name, String description, int archived, String titleValue) {
this.stage = stage;
this.id = id;
this.archived = archived;
title.setText(titleValue);
this.name.setText(name);
this.description.setText(description);
curEmployeeFIO.setText(CurEmployee.getFio());
}
@FXML
private Button categoriesMenu;
@FXML
private Text curEmployeeFIO;
@FXML
private TextArea description;
@FXML
private Button dishesMenu;
@FXML
private Text errorMessage;
@FXML
private Button exit;
@FXML
private TextField name;
@FXML
private Button save;
@FXML
private Button suppliesMenu;
@FXML
private Text title;
@FXML
void categoriesMenuButtonClicked(ActionEvent event) {
errorMessage.setText("");
if (!forgetChanges()) return;
Scene currentScene = categoriesMenu.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("editorCategoriesList.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
EditorCategoriesListSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
@FXML
void dishesMenuButtonClicked(ActionEvent event) {
errorMessage.setText("");
if (!forgetChanges()) return;
Scene currentScene = dishesMenu.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("editorMain.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
EditorMainSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
@FXML
void exitButtonClicked(ActionEvent event) {
errorMessage.setText("");
if (!forgetChanges()) return;
MethodsSceneController.logoutAction(stage);
}
@FXML
void saveButtonClicked(ActionEvent event) {
errorMessage.setText("");
if (!ConfirmationBox.showAlertBox("Сохранение", "Вы уверены, что хотите сохранить изменения?")) {
return;
}
// сохранить
String nameValue = name.getText();
String descriptionValue = description.getText();
String message;
if (id == -1) {
message = ServerCommunicator.addCategory(nameValue, descriptionValue, 0);
} else {
message = ServerCommunicator.changeCategory(id, nameValue, descriptionValue, archived);
}
switch (message) {
case "success": {
Scene currentScene = categoriesMenu.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("editorCategoriesList.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
EditorCategoriesListSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
return;
}
case "denied": {
message = "Отказано в доступе";
break;
}
case "error": {
message = "Произошла ошибка";
break;
}
default:
message = "Произошла ошибка";
}
errorMessage.setText(message);
}
@FXML
void suppliesMenuButtonClicked(ActionEvent event) {
errorMessage.setText("");
if (!forgetChanges()) return;
Scene currentScene = suppliesMenu.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("editorSuppliesMenu.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
EditorSuppliesMenuSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
public boolean forgetChanges() {
return ConfirmationBox.showAlertBox("Предупреждение", "Вы уверены, что хотите завершить изменение категории? Несохраненные данные будут потеряны.");
}
}

EditorCategoriesListSceneController.java

package com.example.client.employee.editor;
import com.example.client.Application;
import com.example.client.ConfirmationBox;
import com.example.client.MethodsSceneController;
import com.example.client.ServerCommunicator;
import com.example.client.employee.CurEmployee;
import com.example.client.employee.administrator.AdminEmployeeEditSceneController;
import com.example.client.items.Category;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.scene.layout.GridPane;
import javafx.stage.Stage;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.ScrollPane;
import javafx.scene.layout.VBox;
import javafx.scene.text.Text;
import java.io.IOException;
import java.util.ArrayList;
public class EditorCategoriesListSceneController {
private Stage stage;
public void start(Stage stage) {
this.stage = stage;
curEmployeeFIO.setText(CurEmployee.getFio());
showListOfCategories();
}
public void showListOfCategories() {
categoriesContainer.getChildren().clear();
ArrayList<Category> categoriesList = new ArrayList<>();
// сообщение и список категорий
String message = ServerCommunicator.getAllCategories(categoriesList);
switch (message) {
case "error": {
errorMessage.setText("Произошла ошибка.");
break;
}
case "deny": {
errorMessage.setText("Недостаточно прав. Перезайдите в аккаунт.");
break;
}
default: {
for (Category categoryItem : categoriesList) {
GridPane gridPane = createGridPane(categoryItem.id, categoryItem.name, categoryItem.description, categoryItem.archived);
categoriesContainer.getChildren().add(gridPane);
}
listOfEmployees.setFitToWidth(true);
listOfEmployees.setFitToHeight(true);
}
}
}
private GridPane createGridPane(int id, String name, String description, int archived) {
try {
FXMLLoader fxmlLoader = new FXMLLoader(Application.class.getResource("itemCategory.fxml"));
GridPane gridPane = fxmlLoader.load();
// архивировать
String buttonName = "Архивировать";
if (archived != 0) buttonName = "Показать";
Button archiveCategoryButton = new Button(buttonName);
archiveCategoryButton.setOnAction(event -> {
if (ConfirmationBox.showAlertBox("Изменение свойства категории", "Вы уверены, что хотите изменить данное свойство? Оно приводит к изменению видимости блюд данной категории.")) {
int newArchived;
if (archived == 0) newArchived = 1;
else newArchived = 0;
String message = ServerCommunicator.changeCategory(id, name, description, newArchived);
switch (message) {
case "error": {
errorMessage.setText("Произошла ошибка.");
break;
}
case "deny": {
errorMessage.setText("Недостаточно прав. Перезайдите в аккаунт.");
break;
}
}
showListOfCategories();
}
});
GridPane.setConstraints(archiveCategoryButton, 1, 0); // столбцы и строки
gridPane.getChildren().add(archiveCategoryButton);
// удалить
Button deleteCategoryButton = new Button("Удалить");
deleteCategoryButton.setOnAction(event -> {
if (ConfirmationBox.showAlertBox("Удаление категории", "Вы уверены, что хотите удалить категорию?")) {
String message = ServerCommunicator.deleteCategory(id);
switch (message) {
case "error": {
errorMessage.setText("Произошла ошибка.");
break;
}
case "deny": {
errorMessage.setText("Недостаточно прав. Перезайдите в аккаунт.");
break;
}
}
showListOfCategories();
}
});
GridPane.setConstraints(deleteCategoryButton, 2, 0); // столбцы и строки
gridPane.getChildren().add(deleteCategoryButton);
// редактировать
Button configureCategoryButton = new Button("Изменить");
configureCategoryButton.setOnAction(event -> {
Scene currentScene = addCategory.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("editorCategoriesEdit.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
EditorCategoriesEditSceneController controller = loader.getController();
controller.start(stage, id, name, description, archived, "Изменение категории");
} catch (IOException e) {
e.printStackTrace();
}
});
GridPane.setConstraints(configureCategoryButton, 2, 1);
gridPane.getChildren().add(configureCategoryButton);
// название
Label labelFio = new Label(name);
GridPane.setConstraints(labelFio, 0, 0);
gridPane.getChildren().add(labelFio);
// описание
Label labelPosition = new Label(description);
GridPane.setConstraints(labelPosition, 0, 1);
gridPane.getChildren().add(labelPosition);
// архивация
if (archived != 0) {
Label labelArchived = new Label("Скрыто");
GridPane.setConstraints(labelArchived, 1, 1);
gridPane.getChildren().add(labelArchived);
}
return gridPane;
} catch (IOException e) {
e.printStackTrace();
}
return null;
}
@FXML
private Button addCategory;
@FXML
private VBox categoriesContainer;
@FXML
private Button categoriesMenu;
@FXML
private Text curEmployeeFIO;
@FXML
private Button dishesMenu;
@FXML
private Text errorMessage;
@FXML
private Button exit;
@FXML
private ScrollPane listOfEmployees;
@FXML
private Button suppliesMenu;
@FXML
void addCategoryButtonClicked(ActionEvent event) {
errorMessage.setText("");
Scene currentScene = dishesMenu.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("editorCategoriesEdit.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
EditorCategoriesEditSceneController controller = loader.getController();
controller.start(stage, -1, "", "", 0, "Добавить категорию");
} catch (IOException e) {
e.printStackTrace();
}
}
@FXML
void categoriesMenuButtonClicked(ActionEvent event) {
errorMessage.setText("");
showListOfCategories();
}
@FXML
void dishesMenuButtonClicked(ActionEvent event) {
errorMessage.setText("");
Scene currentScene = dishesMenu.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("editorMain.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
EditorMainSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
@FXML
void exitButtonClicked(ActionEvent event) {
errorMessage.setText("");
MethodsSceneController.logoutAction(stage);
}
@FXML
void suppliesMenuButtonClicked(ActionEvent event) {
errorMessage.setText("");
Scene currentScene = suppliesMenu.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("editorSuppliesMenu.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
EditorSuppliesMenuSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
}

EditorDishEditSimpleSceneController.java

EditorIngredientEditSceneController.java

EditorMainSceneController.java

package com.example.client.employee.editor;
import com.example.client.Application;
import com.example.client.ConfirmationBox;
import com.example.client.MethodsSceneController;
import com.example.client.ServerCommunicator;
import com.example.client.employee.CurEmployee;
import com.example.client.items.Category;
import com.example.client.items.Dish;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.ButtonBar;
import javafx.scene.control.Label;
import javafx.scene.control.ScrollPane;
import javafx.scene.layout.GridPane;
import javafx.scene.layout.VBox;
import javafx.scene.text.Text;
import javafx.stage.Stage;
import java.io.IOException;
import java.util.ArrayList;
public class EditorMainSceneController {
private Stage stage;
private String curCategory;
public void start(Stage stage) {
errorMessage.setText("");
this.stage = stage;
showCategories();
curEmployeeFIO.setText(CurEmployee.getFio());
showListOfDishes("все");
curCategory = "все";
}
private void showCategories() {
ArrayList<Category> list = new ArrayList<>();
switch (ServerCommunicator.getAllCategories(list)) {
case "error": {
errorMessage.setText("Произошла ошибка");
}
case "denied": {
errorMessage.setText("Отказано в доступе");
}
}
for (int i = list.size() - 1; i >= 0; i--) {
Category category = list.get(i);
addButtonToBar(category.name);
}
addButtonToBar("все");
}
public void addButtonToBar(String buttonText) {
Button button = new Button(buttonText);
button.setOnAction(event -> {
showListOfDishes(buttonText);
curCategory = buttonText;
});
categories.getButtons().add(button);
}
public void showListOfDishes(String category) {
dishesContainer.getChildren().clear();
ArrayList<Dish> dishesList = new ArrayList<>();
// сообщение и список блюд выбранной категории
String message = ServerCommunicator.getDishesByCategory(dishesList, category);
switch (message) {
case "error": {
errorMessage.setText("Произошла ошибка.");
break;
}
case "deny": {
errorMessage.setText("Недостаточно прав. Перезайдите в аккаунт.");
break;
}
default: {
for (Dish dishItem : dishesList) {
GridPane gridPane = createGridPane(dishItem);
dishesContainer.getChildren().add(gridPane);
}
listOfEmployees.setFitToWidth(true);
listOfEmployees.setFitToHeight(true);
}
}
}
private GridPane createGridPane(Dish dish) {
try {
FXMLLoader fxmlLoader = new FXMLLoader(Application.class.getResource("itemDishSimple.fxml"));
GridPane gridPane = fxmlLoader.load();
// удалить
Button deleteCategoryButton = new Button("Удалить");
deleteCategoryButton.setOnAction(event -> {
switch (deleteDish(dish.id)) {
case "success": {
showListOfDishes(curCategory);
break;
}
case "denied": {
errorMessage.setText("Отказано в доступе");
break;
}
case "error": {
errorMessage.setText("Произошла ошибка");
break;
}
default:
errorMessage.setText("Произошла ошибка");
}
});
GridPane.setConstraints(deleteCategoryButton, 1, 0); // столбцы и строки
gridPane.getChildren().add(deleteCategoryButton);
// архивировать
String buttonName = "Архивировать";
if (dish.archived != 0) buttonName = "Показать";
Button archiveCategoryButton = new Button(buttonName);
archiveCategoryButton.setOnAction(event -> {
if (ConfirmationBox.showAlertBox("Изменение свойств блюда", "Вы уверены, что хотите изменить видимость блюда?")) {
switch (archiveDish(dish.id, dish.archived)) {
case "success": {
showListOfDishes(curCategory);
break;
}
case "denied": {
errorMessage.setText("Отказано в доступе");
break;
}
case "error": {
errorMessage.setText("Произошла ошибка");
break;
}
default:
errorMessage.setText("Произошла ошибка");
}
}
});
GridPane.setConstraints(archiveCategoryButton, 1, 1); // столбцы и строки
gridPane.getChildren().add(archiveCategoryButton);
// нет в наличии
String buttonNameStock = "Изъять из наличия";
if (dish.inStock != 1) buttonNameStock = "Вернуть в наличие";
Button stockCategoryButton = new Button(buttonNameStock);
stockCategoryButton.setOnAction(event -> {
if (ConfirmationBox.showAlertBox("Изменение свойств блюда", "Вы уверены, что хотите изменить свойство блюда?")) {
switch (inStockDish(dish.id, dish.inStock)) {
case "success": {
showListOfDishes(curCategory);
break;
}
case "denied": {
errorMessage.setText("Отказано в доступе");
break;
}
case "error": {
errorMessage.setText("Произошла ошибка");
break;
}
default:
errorMessage.setText("Произошла ошибка");
}
}
});
GridPane.setConstraints(stockCategoryButton, 1, 2); // столбцы и строки
gridPane.getChildren().add(stockCategoryButton);
// редактировать
Button configureCategoryButton = new Button("Изменить");
configureCategoryButton.setOnAction(event -> {
Scene currentScene = addDish.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("editorDishEditSimple.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
EditorDishEditSimpleSceneController controller = loader.getController();
controller.start(stage, dish.id, "Изменение блюда");
} catch (IOException e) {
e.printStackTrace();
}
});
GridPane.setConstraints(configureCategoryButton, 1, 3);
gridPane.getChildren().add(configureCategoryButton);
// название
Label labelName = new Label(dish.name);
GridPane.setConstraints(labelName, 0, 0);
gridPane.getChildren().add(labelName);
// количество
Label labelCount = new Label("Количество: " + dish.count);
GridPane.setConstraints(labelCount, 0, 1);
gridPane.getChildren().add(labelCount);
// статус
String status = "активно";
switch (dish.getStatus()) {
case "archived": {
status = "в архиве";
break;
}
case "notInStock": {
status = "нет в наличии";
break;
}
}
Label labelStatus = new Label("Статус: " + status);
GridPane.setConstraints(labelStatus, 0, 2);
gridPane.getChildren().add(labelStatus);
// цена
Label labelPrice = new Label("Цена: " + dish.price);
GridPane.setConstraints(labelPrice, 0, 3);
gridPane.getChildren().add(labelPrice);
return gridPane;
} catch (IOException e) {
e.printStackTrace();
}
return null;
}
private String inStockDish(int id, int inStock) {
return ServerCommunicator.inStockDish(id, inStock);
}
private String archiveDish(int id, int archived) {
return ServerCommunicator.archiveDish(id, archived);
}
private String deleteDish(int id) {
if (ConfirmationBox.showAlertBox("Удаление блюда", "Вы уверены, что хотите удалить блюдо без возможности восстановления?")) {
return ServerCommunicator.deleteDish(id);
}
return "";
}
@FXML
private ButtonBar categories;
@FXML
private Button categoriesMenu;
@FXML
private Text curEmployeeFIO;
@FXML
private VBox dishesContainer;
@FXML
private Button dishesMenu;
@FXML
private Text errorMessage;
@FXML
private Button exit;
@FXML
private ScrollPane listOfEmployees;
@FXML
private Button suppliesMenu;
@FXML
private Button addDish;
@FXML
void addDishButtonClicked(ActionEvent event) {
errorMessage.setText("");
Scene currentScene = categories.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("editorDishEditSimple.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
EditorDishEditSimpleSceneController controller = loader.getController();
controller.start(stage, -1, "Добавление блюда");
} catch (IOException e) {
e.printStackTrace();
}
}
@FXML
void categoriesMenuButtonClicked(ActionEvent event) {
errorMessage.setText("");
Scene currentScene = categories.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("editorCategoriesList.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
EditorCategoriesListSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
@FXML
void dishesMenuButtonClicked(ActionEvent event) {
errorMessage.setText("");
// обновление списка блюд
showListOfDishes("все");
}
@FXML
void exitButtonClicked(ActionEvent event) {
errorMessage.setText("");
MethodsSceneController.logoutAction(stage);
}
@FXML
void suppliesMenuButtonClicked(ActionEvent event) {
errorMessage.setText("");
Scene currentScene = suppliesMenu.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("editorSuppliesMenu.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
EditorSuppliesMenuSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
}

EditorSuppliesMenuSceneController.java

package com.example.client.employee.editor;
import com.example.client.Application;
import com.example.client.ConfirmationBox;
import com.example.client.MethodsSceneController;
import com.example.client.ServerCommunicator;
import com.example.client.employee.CurEmployee;
import com.example.client.items.Ingredient;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.scene.layout.GridPane;
import javafx.stage.Stage;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.layout.VBox;
import javafx.scene.text.Text;
import javafx.scene.control.ScrollPane;
import java.io.IOException;
import java.util.ArrayList;
public class EditorSuppliesMenuSceneController {
private Stage stage;
public void start(Stage stage) {
this.stage = stage;
curEmployeeFIO.setText(CurEmployee.getFio());
showListOfIngredients();
}
private void showListOfIngredients() {
ingredientsContainer.getChildren().clear();
ArrayList<Ingredient> ingredientList = new ArrayList<>();
// сообщение и список категорий
String message = ServerCommunicator.getAllIngridients(ingredientList);
switch (message) {
case "error": {
errorMessage.setText("Произошла ошибка.");
break;
}
case "deny": {
errorMessage.setText("Недостаточно прав. Перезайдите в аккаунт.");
break;
}
default: {
for (Ingredient ingredientItem : ingredientList) {
GridPane gridPane = createGridPane(ingredientItem.id, ingredientItem.fullName, ingredientItem.simpleName, ingredientItem.count, ingredientItem.unit);
ingredientsContainer.getChildren().add(gridPane);
}
listOfIngredients.setFitToWidth(true);
listOfIngredients.setFitToHeight(true);
}
}
}
private GridPane createGridPane(int id, String fullName, String simpleName, float count, String unit) {
try {
FXMLLoader fxmlLoader = new FXMLLoader(Application.class.getResource("itemIngredient.fxml"));
GridPane gridPane = fxmlLoader.load();
// удалить
Button deleteButton = new Button("Удалить");
deleteButton.setOnAction(event -> {
if (ConfirmationBox.showAlertBox("Удаление ингредиента", "Вы уверены, что хотите удалить ингредиент?")) {
String message = ServerCommunicator.deleteIngredient(id);
switch (message) {
case "error": {
errorMessage.setText("Произошла ошибка.");
break;
}
case "deny": {
errorMessage.setText("Недостаточно прав. Перезайдите в аккаунт.");
break;
}
}
showListOfIngredients();
}
});
GridPane.setConstraints(deleteButton, 4, 0); // столбцы и строки
gridPane.getChildren().add(deleteButton);
// редактировать
Button configureButton = new Button("Изменить");
configureButton.setOnAction(event -> {
Scene currentScene = addIngredient.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("editorIngredientEdit.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
EditorIngredientEditSceneController controller = loader.getController();
controller.start(stage, id, "Изменение ингредиента на складе");
} catch (IOException e) {
e.printStackTrace();
}
});
GridPane.setConstraints(configureButton, 4, 1); // столбцы и строки
gridPane.getChildren().add(configureButton);
// название в составе
Label labelSimpleName = new Label(simpleName);
GridPane.setConstraints(labelSimpleName, 0, 0); // столбцы и строки
gridPane.getChildren().add(labelSimpleName);
Label labelFullName = new Label(fullName);
GridPane.setConstraints(labelFullName, 1, 0); // столбцы и строки
gridPane.getChildren().add(labelFullName);
Label labelCount = new Label(Float.toString(count));
GridPane.setConstraints(labelCount, 2, 0); // столбцы и строки
gridPane.getChildren().add(labelCount);
Label labelUnit = new Label(unit);
GridPane.setConstraints(labelUnit, 3, 0); // столбцы и строки
gridPane.getChildren().add(labelUnit);
return gridPane;
} catch (IOException e) {
e.printStackTrace();
}
return null;
}
@FXML
private ScrollPane listOfIngredients;
@FXML
private Text curEmployeeFIO;
@FXML
private Button addIngredient;
@FXML
private Button categoriesMenu;
@FXML
private Button dishesMenu;
@FXML
private Text errorMessage;
@FXML
private Button exit;
@FXML
private VBox ingredientsContainer;
@FXML
private Button suppliesMenu;
@FXML
private Text title;
@FXML
void addIngredientButtonClicked(ActionEvent event) {
errorMessage.setText("");
Scene currentScene = addIngredient.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("editorIngredientEdit.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
EditorIngredientEditSceneController controller = loader.getController();
controller.start(stage, -1,"Добавление ингредиента");
} catch (IOException e) {
e.printStackTrace();
}
}
@FXML
void categoriesMenuButtonClicked(ActionEvent event) {
errorMessage.setText("");
Scene currentScene = categoriesMenu.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("editorCategoriesList.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
EditorCategoriesListSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
@FXML
void dishesMenuButtonClicked(ActionEvent event) {
errorMessage.setText("");
Scene currentScene = dishesMenu.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("editorMain.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
EditorMainSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
@FXML
void exitButtonClicked(ActionEvent event) {
errorMessage.setText("");
MethodsSceneController.logoutAction(stage);
}
@FXML
void suppliesMenuButtonClicked(ActionEvent event) {
errorMessage.setText("");
Scene currentScene = suppliesMenu.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("editorSuppliesMenu.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
EditorSuppliesMenuSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
}

Employee.java

package com.example.client.employee;
public class Employee {
public int id;
public String login;
public String password;
public String position;
public String fio;
public String telephone;
public Employee(int id, String fio, String position, String telephone) {
this.id = id;
this.fio = fio;
this.position = position;
this.telephone = telephone;
}
public Employee(String fio, String login, String password, String position, String telephone) {
this.fio = fio;
this.position = position;
this.telephone = telephone;
this.login = login;
this.password = password;
}
}

Category.java

package com.example.client.items;
public class Category {
public int id;
public String name;
public String description;
public int archived; // 0 or 1
public Category(int \_id, String \_name, String \_description, int \_archived) {
this.id = \_id;
this.name = \_name;
this.description = \_description;
this.archived = \_archived;
}
public Category(int \_id, String \_name, String \_description) {
this.id = \_id;
this.name = \_name;
this.description = \_description;
this.archived = 0;
}
public Category(int \_id, String \_name) {
this.id = \_id;
this.name = \_name;
}
}

Dish.java

package com.example.client.items;
import java.util.HashMap;
public class Dish {
public int id;
public String name;
public String description;
public HashMap<Integer, Ingredient> ingredientsList;
public float price;
public float weight;
public int id\_category;
public int count;
public int archived;
public int inStock;
public Dish(int id, String name, String description, HashMap<Integer, Ingredient> ingredientsList, float price, float weight, int id\_category, int count, int archived, int inStock) {
this.id = id;
this.name = name;
this.description = description;
this.ingredientsList = new HashMap<>(ingredientsList);
this.price = price;
this.weight = weight;
this.id\_category = id\_category;
this.count = count;
this.archived = archived;
this.inStock = inStock;
}
public Dish(int id, String name, String description, float price, float weight, int id\_category, int count, int archived, int inStock) {
this.id = id;
this.name = name;
this.description = description;
this.ingredientsList = new HashMap<>();
this.price = price;
this.weight = weight;
this.id\_category = id\_category;
this.count = count;
this.archived = archived;
this.inStock = inStock;
}
public void addIngredient(Ingredient ingredient) {
this.ingredientsList.put(ingredient.id, ingredient);
}
public void deleteIngredient(Ingredient ingredient) {
this.ingredientsList.remove(ingredient.id);
}
/\*\*
\* @return active, notInStock, archived
\*/
public String getStatus() {
if (archived == 1) {
return "archived";
}
if (inStock == 0 || count < 1) {
return "notInStock";
} else return "active";
}
}

Ingredient.java

package com.example.client.items;
public class Ingredient {
public int id;
public String simpleName;
public String fullName;
public float count;
public String unit; // единица измерения
/\*\*
\*
\* @param id
\* @param simpleName
\* @param fullName
\* @param count
\* @param unit
\*/
public Ingredient(int id, String simpleName, String fullName, float count, String unit) {
this.id = id;
this.simpleName = simpleName;
this.fullName = fullName;
this.count = count;
this.unit = unit;
}
}

Units.java

package com.example.client.items;
public class Units {
public int id;
public String name;
}

MethodsSceneController.java

package com.example.client;
import com.example.client.client.Client;
import com.example.client.employee.CurEmployee;
import com.example.client.employee.Employee;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;
import java.io.IOException;
public class MethodsSceneController {
public static void logoutAction(Stage stage) {
if (!ConfirmationBox.showAlertBox("Выход из аккаунта", "Вы уверены, что хотите выйти из аккаунта?")) {
return;
}
CurEmployee.clean();
Client.clean();
FXMLLoader loader = new FXMLLoader(Application.class.getResource("start.fxml"));
try {
Parent root = loader.load();
Scene newScene = new Scene(root);
stage.setScene(newScene);
} catch (IOException e) {
e.printStackTrace();
}
}
}

ProtocolBuilder.java

package com.example.client;
import com.example.client.client.Client;
import com.example.client.employee.Employee;
import com.example.client.items.Dish;
import com.example.client.items.Ingredient;
import com.google.gson.\*;
import java.util.HashMap;
import java.util.Map;
public class ProtocolBuilder {
public static String authEmployee(String login, String password) {
Gson gson = new Gson();
String action = "login";
int employee = 1;
JsonObject jsonObject = new JsonObject();
jsonObject.addProperty("action", action);
jsonObject.addProperty("login", login);
jsonObject.addProperty("password", password);
String json = gson.toJson(jsonObject);
return json;
}
public static String getEmployeesListRequest(String key) {
Gson gson = new Gson();
String action = "getEmployeesList";
JsonObject jsonObject = new JsonObject();
jsonObject.addProperty("action", action);
jsonObject.addProperty("key", key);
String json = gson.toJson(jsonObject);
return json;
}
public static String addEmployee(String key, Employee newEmp) {
Gson gson = new Gson();
String action = "addEmployee";
JsonObject jsonObject = new JsonObject();
jsonObject.addProperty("action", action);
jsonObject.addProperty("key", key);
jsonObject.addProperty("login", newEmp.login);
jsonObject.addProperty("password", newEmp.password);
jsonObject.addProperty("fio", newEmp.fio);
jsonObject.addProperty("position", newEmp.position);
jsonObject.addProperty("telephone", newEmp.telephone);
String json = gson.toJson(jsonObject);
return json;
}
public static String getEmployee(String key, int id) {
Gson gson = new Gson();
String action = "getEmployee";
JsonObject jsonObject = new JsonObject();
jsonObject.addProperty("action", action);
jsonObject.addProperty("key", key);
jsonObject.addProperty("id", id);
String json = gson.toJson(jsonObject);
return json;
}
public static String setEmployee(String key, int id, String fio, String position, String telephone, String login, String password) {
Gson gson = new Gson();
String action = "setEmployee";
JsonObject jsonObject = new JsonObject();
jsonObject.addProperty("action", action);
jsonObject.addProperty("key", key);
jsonObject.addProperty("id", Integer.toString(id));
jsonObject.addProperty("login", login);
if (password != null) jsonObject.addProperty("password", password);
jsonObject.addProperty("fio", fio);
jsonObject.addProperty("position", position);
jsonObject.addProperty("telephone", telephone);
String json = gson.toJson(jsonObject);
return json;
}
public static String getRestaurantInfo() {
Gson gson = new Gson();
JsonObject jsonObject = new JsonObject();
jsonObject.addProperty("action", "getRestaurantInfo");
String json = gson.toJson(jsonObject);
return json;
}
public static String setRestaurantInfo(String key, String name, String description, String telephone, String address) {
Gson gson = new Gson();
String action = "setRestaurantInfo";
JsonObject jsonObject = new JsonObject();
jsonObject.addProperty("action", action);
jsonObject.addProperty("key", key);
jsonObject.addProperty("name", name);
jsonObject.addProperty("description", description);
jsonObject.addProperty("telephone", telephone);
jsonObject.addProperty("address", address);
String json = gson.toJson(jsonObject);
return json;
}
public static String deleteEmployee(String sessionKey, int id) {
Gson gson = new Gson();
String action = "deleteEmployee";
JsonObject jsonObject = new JsonObject();
jsonObject.addProperty("action", action);
jsonObject.addProperty("key", sessionKey);
jsonObject.addProperty("id", id);
String json = gson.toJson(jsonObject);
return json;
}
//--------------------------------------//
public static String getAllCategories(String key) {
Gson gson = new Gson();
String action = "getAllCategories";
JsonObject jsonObject = new JsonObject();
jsonObject.addProperty("action", action);
jsonObject.addProperty("key", key);
String json = gson.toJson(jsonObject);
return json;
}
public static String getAllCategories() {
Gson gson = new Gson();
String action = "getAllCategories";
JsonObject jsonObject = new JsonObject();
jsonObject.addProperty("action", action);
String json = gson.toJson(jsonObject);
return json;
}
public static String getCategory(String key, int id) {
Gson gson = new Gson();
String action = "getCategory";
JsonObject jsonObject = new JsonObject();
jsonObject.addProperty("action", action);
jsonObject.addProperty("key", key);
jsonObject.addProperty("id", id);
String json = gson.toJson(jsonObject);
return json;
}
public static String changeCategory(String key, int id, String name, String description, int archived) {
Gson gson = new Gson();
String action = "changeCategory";
JsonObject jsonObject = new JsonObject();
jsonObject.addProperty("action", action);
jsonObject.addProperty("key", key);
jsonObject.addProperty("id", id);
jsonObject.addProperty("name", name);
jsonObject.addProperty("description", description);
jsonObject.addProperty("archived", archived);
String json = gson.toJson(jsonObject);
return json;
}
public static String addCategory(String key, String name, String description) {
Gson gson = new Gson();
String action = "addCategory";
JsonObject jsonObject = new JsonObject();
jsonObject.addProperty("action", action);
jsonObject.addProperty("key", key);
jsonObject.addProperty("name", name);
jsonObject.addProperty("description", description);
String json = gson.toJson(jsonObject);
return json;
}
public static String deleteCategory(String key, int id) {
Gson gson = new Gson();
String action = "deleteCategory";
JsonObject jsonObject = new JsonObject();
jsonObject.addProperty("action", action);
jsonObject.addProperty("key", key);
jsonObject.addProperty("id", id);
String json = gson.toJson(jsonObject);
return json;
}
public static String getAllUnits(String sessionKey) {
Gson gson = new Gson();
String action = "getAllUnits";
JsonObject jsonObject = new JsonObject();
jsonObject.addProperty("action", action);
jsonObject.addProperty("key", sessionKey);
String json = gson.toJson(jsonObject);
return json;
}
public static String getAllIngredients(String sessionKey) {
Gson gson = new Gson();
String action = "getAllIngredients";
JsonObject jsonObject = new JsonObject();
jsonObject.addProperty("action", action);
jsonObject.addProperty("key", sessionKey);
String json = gson.toJson(jsonObject);
return json;
}
public static String getIngredient(String sessionKey, int id) {
Gson gson = new Gson();
String action = "getIngredient";
JsonObject jsonObject = new JsonObject();
jsonObject.addProperty("action", action);
jsonObject.addProperty("key", sessionKey);
jsonObject.addProperty("id", id);
String json = gson.toJson(jsonObject);
return json;
}
public static String addIngredient(String sessionKey, String simpleNameValue, String fullNameValue, String countValue, int unitValue) {
Gson gson = new Gson();
String action = "addIngredient";
JsonObject jsonObject = new JsonObject();
jsonObject.addProperty("action", action);
jsonObject.addProperty("key", sessionKey);
jsonObject.addProperty("simple\_name", simpleNameValue);
jsonObject.addProperty("full\_name", fullNameValue);
jsonObject.addProperty("count", countValue);
jsonObject.addProperty("unit", unitValue);
String json = gson.toJson(jsonObject);
return json;
}
public static String changeIngredient(String sessionKey, int id, String simpleNameValue, String fullNameValue, String countValue, int unitValue) {
Gson gson = new Gson();
String action = "changeIngredient";
JsonObject jsonObject = new JsonObject();
jsonObject.addProperty("action", action);
jsonObject.addProperty("key", sessionKey);
jsonObject.addProperty("id", id);
jsonObject.addProperty("simple\_name", simpleNameValue);
jsonObject.addProperty("full\_name", fullNameValue);
jsonObject.addProperty("count", countValue);
jsonObject.addProperty("unit", unitValue);
String json = gson.toJson(jsonObject);
return json;
}
public static String deleteIngredient(String sessionKey, int id) {
Gson gson = new Gson();
String action = "deleteIngredient";
JsonObject jsonObject = new JsonObject();
jsonObject.addProperty("action", action);
jsonObject.addProperty("key", sessionKey);
jsonObject.addProperty("id", id);
String json = gson.toJson(jsonObject);
return json;
}
public static String getAllDishes(String sessionKey, String category) {
Gson gson = new Gson();
String action = "getAllDishes";
JsonObject jsonObject = new JsonObject();
jsonObject.addProperty("action", action);
jsonObject.addProperty("key", sessionKey);
jsonObject.addProperty("category", category);
String json = gson.toJson(jsonObject);
return json;
}
public static String addOrChangeDish(String sessionKey, Dish dish) {
Gson gson = new Gson();
String action = "addOrChangeDish";
JsonObject jsonObject = new JsonObject();
jsonObject.addProperty("action", action);
jsonObject.addProperty("key", sessionKey);
jsonObject.addProperty("id", dish.id);
jsonObject.addProperty("name", dish.name);
jsonObject.addProperty("description", dish.description);
JsonArray categoryArray = new JsonArray();
for (Ingredient ingredient : dish.ingredientsList.values()) {
JsonObject unitJson = new JsonObject();
unitJson.addProperty("id", ingredient.id);
unitJson.addProperty("count", ingredient.count);
categoryArray.add(unitJson);
}
jsonObject.addProperty("ingredients", categoryArray.toString());
jsonObject.addProperty("price", dish.price);
jsonObject.addProperty("weight", dish.weight);
jsonObject.addProperty("id\_category", dish.id\_category);
jsonObject.addProperty("archived", dish.archived);
jsonObject.addProperty("in\_stock", dish.inStock);
String json = gson.toJson(jsonObject);
return json;
}
public static String getDish(String sessionKey, int id) {
Gson gson = new Gson();
String action = "getDish";
JsonObject jsonObject = new JsonObject();
jsonObject.addProperty("action", action);
jsonObject.addProperty("key", sessionKey);
jsonObject.addProperty("id", id);
String json = gson.toJson(jsonObject);
return json;
}
public static String deleteDish(String sessionKey, int id) {
Gson gson = new Gson();
String action = "deleteDish";
JsonObject jsonObject = new JsonObject();
jsonObject.addProperty("action", action);
jsonObject.addProperty("key", sessionKey);
jsonObject.addProperty("id", id);
String json = gson.toJson(jsonObject);
return json;
}
public static String archiveDish(String sessionKey, int id, int archived) {
Gson gson = new Gson();
String action = "archiveDish";
JsonObject jsonObject = new JsonObject();
jsonObject.addProperty("action", action);
jsonObject.addProperty("key", sessionKey);
jsonObject.addProperty("id", id);
jsonObject.addProperty("archived", archived);
String json = gson.toJson(jsonObject);
return json;
}
public static String inStockDish(String sessionKey, int id, int inStock) {
Gson gson = new Gson();
String action = "inStockDish";
JsonObject jsonObject = new JsonObject();
jsonObject.addProperty("action", action);
jsonObject.addProperty("key", sessionKey);
jsonObject.addProperty("id", id);
jsonObject.addProperty("in\_stock", inStock);
String json = gson.toJson(jsonObject);
return json;
}
public static String getDishesByCategoryClient(String category) {
Gson gson = new Gson();
String action = "getAllDishes";
JsonObject jsonObject = new JsonObject();
jsonObject.addProperty("action", action);
jsonObject.addProperty("category", category);
String json = gson.toJson(jsonObject);
return json;
}
public static String client() {
Gson gson = new Gson();
String action = "client";
JsonObject jsonObject = new JsonObject();
jsonObject.addProperty("action", action);
String json = gson.toJson(jsonObject);
return json;
}
public static String getDishClient(int id) {
Gson gson = new Gson();
String action = "getDish";
JsonObject jsonObject = new JsonObject();
jsonObject.addProperty("action", action);
jsonObject.addProperty("id", id);
String json = gson.toJson(jsonObject);
return json;
}
}

ProtocolDecoder.java

ServerCommunicator.java

ServerConnection.java

package com.example.client;
import javafx.application.Platform;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;
import java.net.UnknownHostException;
import java.util.Timer;
import java.util.TimerTask;
public class ServerConnection {
private static Socket socket;
private static final String hostName = "localhost"; // путь к серверу
private static final int portNumber = 5000;
public static Socket connectToServer() {
try {
socket = new Socket(hostName, portNumber);
//System.out.println("Успешно подключено к серверу!");
return socket;
} catch (IOException e) {
//System.err.println("Не удалось подключиться к серверу: " + e.getMessage());
return null;
}
}
public static void disconnectFromServer() {
try {
if (socket != null && !socket.isClosed()) {
socket.close();
//System.out.println("Сокет успешно закрыт");
}
} catch (IOException e) {
//System.err.println("Ошибка при закрытии сокета: " + e.getMessage());
}
}
}

StartSceneController.java

package com.example.client;
import com.example.client.client.Client;
import com.example.client.client.ClientLoginSceneController;
import com.example.client.employee.AuthSceneController;
import com.example.client.employee.CurEmployee;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.AnchorPane;
import javafx.stage.Stage;
import java.io.IOException;
public class StartSceneController {
@FXML
private Button client;
@FXML
private Button employee;
@FXML
private AnchorPane scene;
@FXML
void clientButtonClicked(ActionEvent event) {
// Получаем сцену из кнопки
Scene currentScene = scene.getScene();
// Получаем окно (Stage) из текущей сцены
Stage stage = (Stage) currentScene.getWindow();
// Загружаем новую сцену из FXML-файла
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("clientLogin.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
ClientLoginSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
@FXML
void employeeButtonClicked(ActionEvent event) {
// Получаем сцену из кнопки
Scene currentScene = scene.getScene();
// Получаем окно (Stage) из текущей сцены
Stage stage = (Stage) currentScene.getWindow();
//stage.setFullScreen(true);
// Загружаем новую сцену из FXML-файла
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("auth.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
AuthSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
public void exitButtonClicked(ActionEvent actionEvent) {
}
public void start(Stage stage) {
CurEmployee.clean();
Client.clean();
}
}

module-info.java

module com.example.client {
requires javafx.controls;
requires javafx.fxml;
requires com.google.gson;
exports com.example.client;
exports com.example.client.employee;
exports com.example.client.employee.administrator;
exports com.example.client.employee.editor;
exports com.example.client.client;
exports com.example.client.items;
opens com.example.client to javafx.fxml;
opens com.example.client.client to javafx.fxml;
opens com.example.client.employee to javafx.fxml;
opens com.example.client.employee.administrator to javafx.fxml;
opens com.example.client.employee.editor to javafx.fxml;
}

Application.java

package com.example.client;
import javafx.fxml.FXMLLoader;
import javafx.scene.Scene;
import javafx.scene.layout.AnchorPane;
import javafx.stage.Stage;
public class Application extends javafx.application.Application {
public static void main(String[] args) {
launch(args);
}
@Override
public void start(Stage primaryStage) {
String hostName = "localhost"; // путь к серверу
int portNumber = 5000;
try {
AnchorPane root = FXMLLoader.load(getClass().getResource("start.fxml"));
//Parent root = FXMLLoader.load(getClass().getResource("start.fxml"));
primaryStage.setTitle("Электронное меню");
primaryStage.setWidth(910); // Установка ширины окна
primaryStage.setHeight(630); // Установка высоты окна
Scene scene = new Scene(root);
primaryStage.setScene(scene);
//primaryStage.setFullScreen(true); // Устанавливаем полноэкранный режим
//primaryStage.setFullScreenExitHint(""); // Пустая строка, чтобы не отображался подсказка о выходе из полноэкранного режима
primaryStage.show();
//root.layoutXProperty().bind(scene.widthProperty().subtract(root.prefWidthProperty()).divide(2));
//root.layoutYProperty().bind(scene.heightProperty().subtract(root.prefHeightProperty()).divide(2));
} catch (Exception e) {
e.printStackTrace();
}
}
}

Client.java

package com.example.client.client;
import java.util.HashMap;
public class Client {
public static int tableNumber;
// выбранные блюда для заказа
// номер столика
public static String position = "client";
public static boolean accepted = false;
public static HashMap<Integer, Integer> selectedDishes = new HashMap<>(); // выбранные блюда - id блюда и количество
public static HashMap<Integer, Float> selectedDishesPrice = new HashMap<>(); // выбранные блюда - id блюда и цена
public static void clean() {
tableNumber = -1;
accepted = false;
selectedDishes.clear();
selectedDishesPrice.clear();
}
public static void addDish(int id, int count, float price) {
selectedDishes.put(id, count);
selectedDishesPrice.put(id, price);
}
public static void delete(int dish) {
selectedDishes.remove(dish);
selectedDishesPrice.remove(dish);
}
}

ClientAboutDishSceneController.java

package com.example.client.client;
import com.example.client.Application;
import com.example.client.MethodsSceneController;
import com.example.client.ServerCommunicator;
import com.example.client.items.Category;
import com.example.client.items.Dish;
import com.example.client.items.Ingredient;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.ScrollPane;
import javafx.scene.text.Text;
import javafx.stage.Stage;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Map;
import java.util.StringJoiner;
public class ClientAboutDishSceneController {
@FXML
private Button aboutRestaurat;
@FXML
private Text category;
@FXML
private Text description;
@FXML
private Button dishesList;
@FXML
private Text errorMessage;
@FXML
private Button exit;
@FXML
private Text ingredients;
@FXML
private Text name;
@FXML
private Text price;
@FXML
private ScrollPane scrollPane;
@FXML
private Button selectedDishes;
@FXML
private Text table;
@FXML
private Text telephone;
@FXML
private Text weight;
private Dish curDish;
private Stage stage;
public void start(Stage stage, Dish curDish) {
this.stage = stage;
table.setText("Стол: " + Integer.toString(Client.tableNumber));
int id = curDish.id;
Dish dish = ServerCommunicator.getDishClient(id);
ArrayList<Category> list = new ArrayList<>();
ServerCommunicator.getAllCategoriesClient(list);
if (dish == null) {
errorMessage.setText("Произошла ошибка");
} else {
curDish = dish;
// устанавливаю значения
name.setText(curDish.name);
description.setText(curDish.description);
//category.setValue();
category.setText("без категории");
for (Category categoryItem : list) {
if (categoryItem.id == curDish.id) {
category.setText(categoryItem.name);
}
}
price.setText(Float.toString(curDish.price));
weight.setText(Float.toString(curDish.weight));
// вывод сохранённых ингредиентов
showIngredients(dish);
}
}
private void showIngredients(Dish dish) {
StringJoiner joiner = new StringJoiner(", ");
for (Map.Entry<Integer, Ingredient> entry : dish.ingredientsList.entrySet()) {
Ingredient value = entry.getValue();
joiner.add(value.simpleName);
}
ingredients.setText(joiner.toString());
}
@FXML
void aboutRestauratButtonClicked(ActionEvent event) {
errorMessage.setText("");
Scene currentScene = errorMessage.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("clientRestaurant.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
ClientRestaurantSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
@FXML
void dishesListButtonClicked(ActionEvent event) {
errorMessage.setText("");
Scene currentScene = errorMessage.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("clientDishes.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
ClientDishesSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
@FXML
void exitButtonClicked(ActionEvent event) {
errorMessage.setText("");
MethodsSceneController.logoutAction(stage);
}
@FXML
void selectedDishesButtonClicked(ActionEvent event) {
errorMessage.setText("");
if (!Client.accepted) {
Scene currentScene = errorMessage.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("clientChoose.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
ClientChooseSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
}
}

ClientAcceptedSceneController.java

ClientChooseSceneController.java

ClientDishesSceneController.java

ClientLoginSceneController.java

package com.example.client.client;
import com.example.client.Application;
import com.example.client.ServerCommunicator;
import com.example.client.employee.editor.EditorSuppliesMenuSceneController;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.TextField;
import javafx.stage.Stage;
import javafx.scene.text.Text;
import java.io.IOException;
public class ClientLoginSceneController {
@FXML
private Button enter;
@FXML
private Button exit;
@FXML
private TextField tableNumber;
@FXML
private Text errorMessage;
private Stage stage;
public void start(Stage \_stage) {
this.stage = \_stage;
tableNumber.setPromptText("Введите номер стола");
tableNumber.textProperty().addListener((observable, oldValue, newValue) -> {
if (!newValue.matches("\\d\*")) {
tableNumber.setText(newValue.replaceAll("[^\\d]", ""));
}
});
}
@FXML
void enterButtonClicked(ActionEvent event) {
errorMessage.setText("");
if (tableNumber.getText().isEmpty() || (Integer.parseInt(tableNumber.getText()) < 0)) {
errorMessage.setText("Введите номер корректный стола");
} else {
if (ServerCommunicator.client().equals("success")) {
} else {
errorMessage.setText("Произошла ошибка.");
return;
}
Client.tableNumber = Integer.parseInt(tableNumber.getText());
Scene currentScene = errorMessage.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("clientDishes.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
ClientDishesSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
}
public void exitButtonClicked(ActionEvent actionEvent) {
errorMessage.setText("");
FXMLLoader loader = new FXMLLoader(Application.class.getResource("start.fxml"));
try {
Parent root = loader.load();
Scene newScene = new Scene(root);
stage.setScene(newScene);
} catch (IOException e) {
e.printStackTrace();
}
}
}

ClientRestaurantSceneController.java

package com.example.client.client;
import com.example.client.Application;
import com.example.client.MethodsSceneController;
import com.example.client.ServerCommunicator;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.ScrollPane;
import javafx.scene.text.Text;
import javafx.stage.Stage;
import java.io.IOException;
import java.util.HashMap;
public class ClientRestaurantSceneController {
@FXML
private Button aboutRestaurat;
@FXML
private Text address;
@FXML
private Text description;
@FXML
private Button dishesList;
@FXML
private Text errorMessage;
@FXML
private Button exit;
@FXML
private Text name;
@FXML
private ScrollPane scrollPane;
@FXML
private Button selectedDishes;
@FXML
private Text table;
@FXML
private Text telephone;
private Stage stage;
public void start(Stage stage) {
this.stage = stage;
table.setText("Стол: " + Integer.toString(Client.tableNumber));
HashMap<String, String> restInfo = ServerCommunicator.getRestaurantInfo();
name.setText(restInfo.get("name"));
description.setText(restInfo.get("description"));
telephone.setText(restInfo.get("telephone"));
address.setText(restInfo.get("address"));
}
@FXML
void aboutRestauratButtonClicked(ActionEvent event) {
errorMessage.setText("");
Scene currentScene = errorMessage.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("clientRestaurant.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
ClientRestaurantSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
@FXML
void dishesListButtonClicked(ActionEvent event) {
errorMessage.setText("");
Scene currentScene = errorMessage.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("clientDishes.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
ClientDishesSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
@FXML
void exitButtonClicked(ActionEvent event) {
errorMessage.setText("");
MethodsSceneController.logoutAction(stage);
}
@FXML
void selectedDishesButtonClicked(ActionEvent event) {
errorMessage.setText("");
if (!Client.accepted) {
Scene currentScene = errorMessage.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("clientChoose.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
ClientChooseSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
}
}

ConfirmationBox.java

package com.example.client;
import javafx.scene.control.Alert;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.control.ButtonType;
import java.util.Optional;
public class ConfirmationBox {
private static boolean displayConfirmation(String title, String message) {
Alert alert = new Alert(AlertType.CONFIRMATION);
alert.setTitle(title);
alert.setHeaderText(null); // Отключает текст заголовка
alert.setContentText(message);
// Создание кнопок "Принять" и "Отмена"
ButtonType acceptButton = new ButtonType("Принять");
ButtonType cancelButton = new ButtonType("Отмена");
// Добавление кнопок в диалоговое окно
alert.getButtonTypes().setAll(acceptButton, cancelButton);
// Отображение окна и ожидание выбора
Optional<ButtonType> result = alert.showAndWait();
// Обработка выбора пользователя
if (result.isPresent() && result.get() == acceptButton) {
// Обработка нажатия кнопки "Принять"
//System.out.println("Принято");
return true;
} else {
// Обработка нажатия кнопки "Отмена" или закрытия окна
//System.out.println("Отменено");
return false;
}
}
public static boolean showAlertBox(String title, String message) {
return displayConfirmation(title, message);
}
}

AdminEmployeeAddSceneController.java

package com.example.client.employee.administrator;
import com.example.client.Application;
import com.example.client.ConfirmationBox;
import com.example.client.ServerCommunicator;
import com.example.client.employee.CurEmployee;
import com.example.client.employee.Employee;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.ComboBox;
import javafx.scene.control.DatePicker;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.text.Text;
import javafx.stage.Stage;
import java.io.IOException;
import java.time.LocalDate;
public class AdminEmployeeAddSceneController {
private Stage stage;
@FXML
private Button addEmployee;
@FXML
private Button employees;
@FXML
private Text errorMessage;
@FXML
private Button exit;
@FXML
private TextField fio;
@FXML
private TextField login;
@FXML
private ComboBox<String> position;
@FXML
private Button restaurantInfo;
@FXML
private Button save;
@FXML
private TextField telephone;
@FXML
private Text curEmployeeFIO;
@FXML
private TextField password;
@FXML
void addEmployeeButtonClicked(ActionEvent event) {
}
@FXML
void employeesButtonClicked(ActionEvent event) {
errorMessage.setText("");
if (!forgetChanges()) return;
Scene currentScene = addEmployee.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("adminMain.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
AdminMainSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
@FXML
void exitButtonClicked(ActionEvent event) {
errorMessage.setText("");
if (!forgetChanges()) return;
if (!ConfirmationBox.showAlertBox("Выход из аккаунта", "Вы уверены, что хотите выйти из аккаунта?")) {
return;
}
FXMLLoader loader = new FXMLLoader(Application.class.getResource("start.fxml"));
try {
Parent root = loader.load();
Scene newScene = new Scene(root);
stage.setScene(newScene);
} catch (IOException e) {
e.printStackTrace();
}
}
@FXML
void restaurantInfoButtonClicked(ActionEvent event) {
errorMessage.setText("");
if (!forgetChanges()) return;
Scene currentScene = addEmployee.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("adminRestaurantInfo.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
AdminRestaurantInfoSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
@FXML
void saveButtonClicked(ActionEvent event) {
errorMessage.setText("");
if (!ConfirmationBox.showAlertBox("Сохранение", "Вы уверены, что хотите сохранить изменения?")) {
return;
}
String fioValue = fio.getText();
if (fioValue.isEmpty()) {
errorMessage.setText("Заполните ФИО");
return;
}
String positionValue;
switch (position.getValue()) {
case "редактор": {
positionValue = "editor";
break;
}
case "администратор": {
positionValue = "administrator";
break;
}
default: {
positionValue = position.getValue();
}
}
String telephoneValue = telephone.getText();
String loginValue = login.getText();
String passwordValue = password.getText();
if (loginValue.isEmpty() || passwordValue.isEmpty()) {
errorMessage.setText("Логин и пароль должны быть установлены.");
return;
}
Employee newEmp = new Employee(fioValue, loginValue, passwordValue, positionValue, telephoneValue);
String message;
switch (ServerCommunicator.addEmployee(newEmp)) {
case "success": {
Scene currentScene = addEmployee.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("adminMain.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
AdminMainSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
return;
}
case "denied": {
message = "Отказано в доступе.";
break;
}
case "notUnicLogin": {
message = "Логин должен быть уникальным.";
break;
}
case "error": {
message = "Произошла ошибка.";
break;
}
default: message = "";
}
errorMessage.setText(message);
}
public boolean forgetChanges() {
if (!fio.getText().isEmpty() || !telephone.getText().isEmpty() || !login.getText().isEmpty() || !password.getText().isEmpty()) {
if (ConfirmationBox.showAlertBox("Предупреждение", "Вы уверены, что хотите завершить добавление сотрудника? Несохраненные данные будут потеряны.")) {
return true;
} else return false;
}
return true;
}
public void start(Stage stage) {
this.stage = stage;
curEmployeeFIO.setText(CurEmployee.getFio());
position.getItems().addAll("редактор", "администратор");
position.setValue("редактор");
}
}

AdminEmployeeEditSceneController.java

package com.example.client.employee.administrator;
import com.example.client.Application;
import com.example.client.ConfirmationBox;
import com.example.client.ServerCommunicator;
import com.example.client.employee.CurEmployee;
import com.example.client.employee.Employee;
import com.example.client.MethodsSceneController;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.ComboBox;
import javafx.scene.control.TextField;
import javafx.stage.Stage;
import javafx.scene.text.Text;
import java.io.IOException;
import java.util.HashMap;
public class AdminEmployeeEditSceneController {
private Stage stage;
private int id;
@FXML
private Button addEmployee;
@FXML
private Text curEmployeeFIO;
@FXML
private Button employees;
@FXML
private Text errorMessage;
@FXML
private Button exit;
@FXML
private TextField fio;
@FXML
private TextField login;
@FXML
private TextField password;
@FXML
private ComboBox<String> position;
@FXML
private Button restaurantInfo;
@FXML
private Button save;
@FXML
private TextField telephone;
public void start(int id, Stage \_stage) {
this.stage = \_stage;
curEmployeeFIO.setText(CurEmployee.getFio());
this.id = id;
// получение информации о сотруднике
HashMap<String, String> empMap = ServerCommunicator.getEmployeeById(id);
// заполнение полей
switch (empMap.get("message")) {
case "success": {
fio.setText(empMap.get("fio"));
telephone.setText(empMap.get("telephone"));
login.setText(empMap.get("login"));
position.setValue(empMap.get("position"));
break;
}
case "denied": {
errorMessage.setText("Отказано в доступе.");
break;
}
case "error": {
errorMessage.setText("Произошла ошибка.");
}
default: {
errorMessage.setText("Произошла ошибка. Попробуйте ещё раз");
return;
}
}
position.getItems().addAll("редактор", "администратор");
}
@FXML
void addEmployeeButtonClicked(ActionEvent event) {
errorMessage.setText("");
if (!forgetChanges()) return;
Scene currentScene = addEmployee.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("adminEmployeeAdd.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
AdminEmployeeAddSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
@FXML
void employeesButtonClicked(ActionEvent event) {
errorMessage.setText("");
if (!forgetChanges()) return;
Scene currentScene = addEmployee.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("adminMain.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
AdminMainSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
public boolean forgetChanges() {
return ConfirmationBox.showAlertBox("Предупреждение", "Вы уверены, что хотите завершить редактирование сотрудника? Несохраненные данные будут потеряны.");
}
@FXML
void exitButtonClicked(ActionEvent event) {
errorMessage.setText("");
MethodsSceneController.logoutAction(stage);
}
@FXML
void restaurantInfoButtonClicked(ActionEvent event) {
errorMessage.setText("");
if (!forgetChanges()) return;
Scene currentScene = addEmployee.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("adminRestaurantInfo.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
AdminRestaurantInfoSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
@FXML
void saveButtonClicked(ActionEvent event) {
errorMessage.setText("");
if (!ConfirmationBox.showAlertBox("Сохранение", "Вы уверены, что хотите сохранить изменения?")) {
return;
}
String positionValue;
switch (position.getValue()) {
case "редактор": {
positionValue = "editor";
break;
}
case "администратор": {
positionValue = "administrator";
break;
}
default: {
positionValue = position.getValue();
}
}
String fioValue = fio.getText();
String telephoneValue = telephone.getText();
String loginValue = login.getText();
String passwordValue = password.getText();
if (passwordValue.equals("Заполните поле, если хотите поменять пароль") || passwordValue.isEmpty()) {
passwordValue = null;
}
if (loginValue.isEmpty()) {
errorMessage.setText("Логин должен быть установлены.");
return;
}
switch (ServerCommunicator.setEmployee(id, fioValue, positionValue, telephoneValue, loginValue, passwordValue)) {
case "error": {
errorMessage.setText("Ошибка.");
break;
}
case "denied": {
errorMessage.setText("Отказано в доступе.");
break;
}
case "notUnicLogin": {
errorMessage.setText("Логин должен быть уникальным");
break;
}
case "noSuchItem": {
errorMessage.setText("Пользователь не найден.");
break;
}
case "success": {
errorMessage.setText("Данные успешно сохранены!");
break;
}
}
}
}

AdminMainSceneController.java

package com.example.client.employee.administrator;
import com.example.client.Application;
import com.example.client.ConfirmationBox;
import com.example.client.MethodsSceneController;
import com.example.client.ServerCommunicator;
import com.example.client.employee.CurEmployee;
import com.example.client.employee.Employee;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.ScrollPane;
import javafx.scene.layout.GridPane;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;
import javafx.scene.text.Text;
import java.io.IOException;
import java.util.ArrayList;
public class AdminMainSceneController {
private Stage stage;
//ArrayList<Employee> listOfEmployees = new ArrayList<>();
@FXML
private Button addEmployee;
@FXML
private Button employees;
@FXML
private VBox employeesContainer;
@FXML
private Button exit;
@FXML
private Text curEmployeeFIO;
@FXML
private Button restaurantInfo;
@FXML
private Text errorMessage;
@FXML
private ScrollPane listOfEmployees;
public void start(Stage \_stage) {
this.stage = \_stage;
// получаем фамилию сотрудника
curEmployeeFIO.setText(CurEmployee.getFio());
// выводим список всех сотрудников
showListOfEmployees();
}
public void showListOfEmployees() {
// try {
employeesContainer.getChildren().clear();
ArrayList<Employee> employeesList = new ArrayList<>();
// сообщение и список сотрудников
String message = ServerCommunicator.getEmployeesList(employeesList);
switch (message) {
case "error": {
errorMessage.setText("Произошла ошибка.");
break;
}
case "deny": {
errorMessage.setText("Недостаточно прав. Перезайдите в аккаунт.");
break;
}
default: {
// вывод всех сотрудников
if (employeesList == null) return;
for (Employee employeeItem : employeesList) {
GridPane gridPane = createGridPane(employeeItem.id, employeeItem.fio, employeeItem.position, employeeItem.telephone);
employeesContainer.getChildren().add(gridPane);
}
listOfEmployees.setFitToWidth(true);
listOfEmployees.setFitToHeight(true);
}
}
}
private GridPane createGridPane(int id, String fio, String position, String telephone) {
try {
FXMLLoader fxmlLoader = new FXMLLoader(Application.class.getResource("itemEmployee.fxml"));
GridPane gridPane = fxmlLoader.load();
// кнопка удаления (себя удалить нельзя)
if (id != CurEmployee.getId()) {
Button deleteEmployeeButton = new Button("Удалить");
deleteEmployeeButton.setOnAction(event -> {
if (ConfirmationBox.showAlertBox("Удаление аккаунта сотрудника", "Вы уверены, что хотите удалить сотрудника?")) {
errorMessage.setText(ServerCommunicator.deleteEmployeeById(id)); // сотрудник переносится в архив
showListOfEmployees();
}
});
GridPane.setConstraints(deleteEmployeeButton, 2, 0); // столбцы и строки
gridPane.getChildren().add(deleteEmployeeButton);
} else {
Label itIsYou = new Label("Это вы");
GridPane.setConstraints(itIsYou, 2, 0);
gridPane.getChildren().add(itIsYou);
}
Button configureEmployeeButton = new Button("Изменить");
configureEmployeeButton.setOnAction(event -> {
Scene currentScene = addEmployee.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("adminEmployeeEdit.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
AdminEmployeeEditSceneController controller = loader.getController();
controller.start(id, stage);
} catch (IOException e) {
e.printStackTrace();
}
});
GridPane.setConstraints(configureEmployeeButton, 2, 1);
gridPane.getChildren().add(configureEmployeeButton);
// ФИО
Label labelFio = new Label(fio);
GridPane.setConstraints(labelFio, 0, 0);
gridPane.getChildren().add(labelFio);
// должность
Label labelPosition = new Label(position);
GridPane.setConstraints(labelPosition, 1, 0);
gridPane.getChildren().add(labelPosition);
// телефон
Label labelTelephone = new Label(telephone);
GridPane.setConstraints(labelTelephone, 1, 1);
gridPane.getChildren().add(labelTelephone);
return gridPane;
} catch (IOException e) {
e.printStackTrace();
}
return null;
}
@FXML
void addEmployeeButtonClicked(ActionEvent event) {
errorMessage.setText("");
Scene currentScene = addEmployee.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("adminEmployeeAdd.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
AdminEmployeeAddSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
@FXML
void employeesButtonClicked(ActionEvent event) {
errorMessage.setText("");
Scene currentScene = addEmployee.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("adminMain.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
AdminMainSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
@FXML
void exitButtonClicked(ActionEvent event) {
errorMessage.setText("");
MethodsSceneController.logoutAction(stage);
}
@FXML
void restaurantInfoButtonClicked(ActionEvent event) {
errorMessage.setText("");
Scene currentScene = addEmployee.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("adminRestaurantInfo.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
AdminRestaurantInfoSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
}

AdminRestaurantInfoSceneController.java

package com.example.client.employee.administrator;
import com.example.client.Application;
import com.example.client.ConfirmationBox;
import com.example.client.MethodsSceneController;
import com.example.client.ServerCommunicator;
import com.example.client.employee.CurEmployee;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextArea;
import javafx.scene.control.TextField;
import javafx.scene.text.Text;
import javafx.stage.Stage;
import java.io.IOException;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.util.HashMap;
import java.util.Locale;
public class AdminRestaurantInfoSceneController {
private Stage stage;
@FXML
private Button addEmployee;
public boolean forgetChanges() {
return ConfirmationBox.showAlertBox("Предупреждение", "Вы уверены, что хотите завершить редактирование информации о ресторане? Несохраненные данные будут потеряны.");
}
@FXML
private TextField address;
@FXML
private TextArea description;
@FXML
private Button employees;
@FXML
private Button exit;
@FXML
private TextField name;
@FXML
private Button restaurantInfo;
@FXML
private Button save;
@FXML
private TextField telephone;
@FXML
private Text curEmployeeFIO;
@FXML
private Text errorMessage;
@FXML
void addEmployeeButtonClicked(ActionEvent event) {
errorMessage.setText("");
if (!forgetChanges()) return;
Scene currentScene = addEmployee.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("adminEmployeeAdd.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
AdminEmployeeAddSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
@FXML
void employeesButtonClicked(ActionEvent event) {
errorMessage.setText("");
if (!forgetChanges()) return;
Scene currentScene = addEmployee.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("adminMain.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
AdminMainSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
@FXML
void exitButtonClicked(ActionEvent event) {
errorMessage.setText("");
if (!forgetChanges()) return;
MethodsSceneController.logoutAction(stage);
}
@FXML
void restaurantInfoButtonClicked(ActionEvent event) {
errorMessage.setText("");
}
@FXML
void saveButtonClicked(ActionEvent event) {
errorMessage.setText("");
if (!ConfirmationBox.showAlertBox("Сохранение", "Вы уверены, что хотите сохранить изменения?")) {
return;
}
String nameValue = name.getText();
String descriptionValue = description.getText();
String telephoneValue = telephone.getText();
String addressValue = address.getText();
switch (ServerCommunicator.setRestaurantInfo(nameValue, descriptionValue, telephoneValue, addressValue)) {
case "error": {
errorMessage.setText("Ошибка.");
break;
}
case "denied": {
errorMessage.setText("Отказано в доступе.");
break;
}
case "success": {
errorMessage.setText("Данные успешно сохранены!");
break;
}
}
}
public void start(Stage stage) {
this.stage = stage;
curEmployeeFIO.setText(CurEmployee.getFio());
// загрузка инфы из базы
HashMap<String, String> restaurantInfo = ServerCommunicator.getRestaurantInfo();
// заполняю поля
switch (restaurantInfo.get("message")) {
case "success": {
name.setText(restaurantInfo.get("name"));
telephone.setText(restaurantInfo.get("telephone"));
description.setText(restaurantInfo.get("description"));
address.setText(restaurantInfo.get("address"));
break;
}
case "error": {
errorMessage.setText("Произошла ошибка.");
}
default: {
errorMessage.setText("Произошла ошибка. Попробуйте ещё раз");
}
}
}
}

AuthSceneController.java

package com.example.client.employee;
import com.example.client.Application;
import com.example.client.ServerCommunicator;
import com.example.client.StartSceneController;
import com.example.client.employee.administrator.AdminMainSceneController;
import com.example.client.employee.editor.EditorMainSceneController;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.PasswordField;
import javafx.scene.text.Text;
import javafx.stage.Stage;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.TextField;
import java.io.IOException;
public class AuthSceneController {
private StartSceneController startSceneController;
private Stage stage;
public void start(Stage \_stage) {
this.stage = \_stage;
}
@FXML
private Button enter;
@FXML
private Button exit;
@FXML
private TextField login;
@FXML
private PasswordField password;
@FXML
private Text errorMessage;
@FXML
void exitButtonClicked(ActionEvent event) {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("start.fxml"));
try {
Parent root = loader.load();
Scene newScene = new Scene(root);
stage.setScene(newScene);
} catch (IOException e) {
e.printStackTrace();
}
}
@FXML
void enterButtonClicked(ActionEvent event) {
String loginValue = login.getText();
String passwordValue = password.getText();
//CurEmployee.setLoginAndPass(loginValue, passwordValue)
String position = ServerCommunicator.authCheck(loginValue, passwordValue);
switch (position) {
case "editor": {
Scene currentScene = enter.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("editorMain.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
EditorMainSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
break;
}
case "administrator": {
Scene currentScene = enter.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("adminMain.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
AdminMainSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
break;
}
case "connectionError": {
errorMessage.setText("Произошла ошибка соединения. Попробуйте ещё раз.");
break;
}
case "failedAuth": {
errorMessage.setText("Несоответствие логина и пароля.");
break;
}
case "noSuchUser": {
errorMessage.setText("Пользователь с таким логином не найден!");
break;
}
default: {
errorMessage.setText("Произошла ошибка соединения. Отказано в соединении с сервером.");
break;
}
}
}
}

CurEmployee.java

package com.example.client.employee;
public class CurEmployee {
// public static String login;
// public static String password;
public static String position; // редактор или админ
public static String fio;
private static int id = -1;
private static String sessionKey;
// public static void setLoginAndPass(String \_login, String \_password) {
// login = \_login;
// password = \_password;
// }
public static void setPosition(String position) {
CurEmployee.position = position;
}
public static void setSessionKey(String sessionKey) {
CurEmployee.sessionKey = sessionKey;
}
// public static String getLogin() {
// return login;
// }
//
// public static String getPassword() {
// return password;
// }
public static void setFio(String fio) {
CurEmployee.fio = fio;
}
public static String getFio() {
return fio;
}
public static void setId(int id) {
CurEmployee.id = id;
}
public static int getId() {
return id;
}
public static String getSessionKey() {
return sessionKey;
}
public static void clean() {
// login = null;
// password = null;
position = null; // редактор или админ
fio = null;
sessionKey = null;
id = -1;
}
}

EditorCategoriesEditSceneController.java

package com.example.client.employee.editor;
import com.example.client.Application;
import com.example.client.ConfirmationBox;
import com.example.client.MethodsSceneController;
import com.example.client.ServerCommunicator;
import com.example.client.employee.CurEmployee;
import com.example.client.employee.Employee;
import com.example.client.items.Category;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.TextArea;
import javafx.scene.control.TextField;
import javafx.scene.text.Text;
import java.io.IOException;
public class EditorCategoriesEditSceneController {
private Stage stage;
private int id;
private int archived;
public void start(Stage stage, int id, String name, String description, int archived, String titleValue) {
this.stage = stage;
this.id = id;
this.archived = archived;
title.setText(titleValue);
this.name.setText(name);
this.description.setText(description);
curEmployeeFIO.setText(CurEmployee.getFio());
}
@FXML
private Button categoriesMenu;
@FXML
private Text curEmployeeFIO;
@FXML
private TextArea description;
@FXML
private Button dishesMenu;
@FXML
private Text errorMessage;
@FXML
private Button exit;
@FXML
private TextField name;
@FXML
private Button save;
@FXML
private Button suppliesMenu;
@FXML
private Text title;
@FXML
void categoriesMenuButtonClicked(ActionEvent event) {
errorMessage.setText("");
if (!forgetChanges()) return;
Scene currentScene = categoriesMenu.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("editorCategoriesList.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
EditorCategoriesListSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
@FXML
void dishesMenuButtonClicked(ActionEvent event) {
errorMessage.setText("");
if (!forgetChanges()) return;
Scene currentScene = dishesMenu.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("editorMain.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
EditorMainSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
@FXML
void exitButtonClicked(ActionEvent event) {
errorMessage.setText("");
if (!forgetChanges()) return;
MethodsSceneController.logoutAction(stage);
}
@FXML
void saveButtonClicked(ActionEvent event) {
errorMessage.setText("");
if (!ConfirmationBox.showAlertBox("Сохранение", "Вы уверены, что хотите сохранить изменения?")) {
return;
}
// сохранить
String nameValue = name.getText();
String descriptionValue = description.getText();
String message;
if (id == -1) {
message = ServerCommunicator.addCategory(nameValue, descriptionValue, 0);
} else {
message = ServerCommunicator.changeCategory(id, nameValue, descriptionValue, archived);
}
switch (message) {
case "success": {
Scene currentScene = categoriesMenu.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("editorCategoriesList.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
EditorCategoriesListSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
return;
}
case "denied": {
message = "Отказано в доступе";
break;
}
case "error": {
message = "Произошла ошибка";
break;
}
default:
message = "Произошла ошибка";
}
errorMessage.setText(message);
}
@FXML
void suppliesMenuButtonClicked(ActionEvent event) {
errorMessage.setText("");
if (!forgetChanges()) return;
Scene currentScene = suppliesMenu.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("editorSuppliesMenu.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
EditorSuppliesMenuSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
public boolean forgetChanges() {
return ConfirmationBox.showAlertBox("Предупреждение", "Вы уверены, что хотите завершить изменение категории? Несохраненные данные будут потеряны.");
}
}

EditorCategoriesListSceneController.java

package com.example.client.employee.editor;
import com.example.client.Application;
import com.example.client.ConfirmationBox;
import com.example.client.MethodsSceneController;
import com.example.client.ServerCommunicator;
import com.example.client.employee.CurEmployee;
import com.example.client.employee.administrator.AdminEmployeeEditSceneController;
import com.example.client.items.Category;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.scene.layout.GridPane;
import javafx.stage.Stage;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.ScrollPane;
import javafx.scene.layout.VBox;
import javafx.scene.text.Text;
import java.io.IOException;
import java.util.ArrayList;
public class EditorCategoriesListSceneController {
private Stage stage;
public void start(Stage stage) {
this.stage = stage;
curEmployeeFIO.setText(CurEmployee.getFio());
showListOfCategories();
}
public void showListOfCategories() {
categoriesContainer.getChildren().clear();
ArrayList<Category> categoriesList = new ArrayList<>();
// сообщение и список категорий
String message = ServerCommunicator.getAllCategories(categoriesList);
switch (message) {
case "error": {
errorMessage.setText("Произошла ошибка.");
break;
}
case "deny": {
errorMessage.setText("Недостаточно прав. Перезайдите в аккаунт.");
break;
}
default: {
for (Category categoryItem : categoriesList) {
GridPane gridPane = createGridPane(categoryItem.id, categoryItem.name, categoryItem.description, categoryItem.archived);
categoriesContainer.getChildren().add(gridPane);
}
listOfEmployees.setFitToWidth(true);
listOfEmployees.setFitToHeight(true);
}
}
}
private GridPane createGridPane(int id, String name, String description, int archived) {
try {
FXMLLoader fxmlLoader = new FXMLLoader(Application.class.getResource("itemCategory.fxml"));
GridPane gridPane = fxmlLoader.load();
// архивировать
String buttonName = "Архивировать";
if (archived != 0) buttonName = "Показать";
Button archiveCategoryButton = new Button(buttonName);
archiveCategoryButton.setOnAction(event -> {
if (ConfirmationBox.showAlertBox("Изменение свойства категории", "Вы уверены, что хотите изменить данное свойство? Оно приводит к изменению видимости блюд данной категории.")) {
int newArchived;
if (archived == 0) newArchived = 1;
else newArchived = 0;
String message = ServerCommunicator.changeCategory(id, name, description, newArchived);
switch (message) {
case "error": {
errorMessage.setText("Произошла ошибка.");
break;
}
case "deny": {
errorMessage.setText("Недостаточно прав. Перезайдите в аккаунт.");
break;
}
}
showListOfCategories();
}
});
GridPane.setConstraints(archiveCategoryButton, 1, 0); // столбцы и строки
gridPane.getChildren().add(archiveCategoryButton);
// удалить
Button deleteCategoryButton = new Button("Удалить");
deleteCategoryButton.setOnAction(event -> {
if (ConfirmationBox.showAlertBox("Удаление категории", "Вы уверены, что хотите удалить категорию?")) {
String message = ServerCommunicator.deleteCategory(id);
switch (message) {
case "error": {
errorMessage.setText("Произошла ошибка.");
break;
}
case "deny": {
errorMessage.setText("Недостаточно прав. Перезайдите в аккаунт.");
break;
}
}
showListOfCategories();
}
});
GridPane.setConstraints(deleteCategoryButton, 2, 0); // столбцы и строки
gridPane.getChildren().add(deleteCategoryButton);
// редактировать
Button configureCategoryButton = new Button("Изменить");
configureCategoryButton.setOnAction(event -> {
Scene currentScene = addCategory.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("editorCategoriesEdit.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
EditorCategoriesEditSceneController controller = loader.getController();
controller.start(stage, id, name, description, archived, "Изменение категории");
} catch (IOException e) {
e.printStackTrace();
}
});
GridPane.setConstraints(configureCategoryButton, 2, 1);
gridPane.getChildren().add(configureCategoryButton);
// название
Label labelFio = new Label(name);
GridPane.setConstraints(labelFio, 0, 0);
gridPane.getChildren().add(labelFio);
// описание
Label labelPosition = new Label(description);
GridPane.setConstraints(labelPosition, 0, 1);
gridPane.getChildren().add(labelPosition);
// архивация
if (archived != 0) {
Label labelArchived = new Label("Скрыто");
GridPane.setConstraints(labelArchived, 1, 1);
gridPane.getChildren().add(labelArchived);
}
return gridPane;
} catch (IOException e) {
e.printStackTrace();
}
return null;
}
@FXML
private Button addCategory;
@FXML
private VBox categoriesContainer;
@FXML
private Button categoriesMenu;
@FXML
private Text curEmployeeFIO;
@FXML
private Button dishesMenu;
@FXML
private Text errorMessage;
@FXML
private Button exit;
@FXML
private ScrollPane listOfEmployees;
@FXML
private Button suppliesMenu;
@FXML
void addCategoryButtonClicked(ActionEvent event) {
errorMessage.setText("");
Scene currentScene = dishesMenu.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("editorCategoriesEdit.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
EditorCategoriesEditSceneController controller = loader.getController();
controller.start(stage, -1, "", "", 0, "Добавить категорию");
} catch (IOException e) {
e.printStackTrace();
}
}
@FXML
void categoriesMenuButtonClicked(ActionEvent event) {
errorMessage.setText("");
showListOfCategories();
}
@FXML
void dishesMenuButtonClicked(ActionEvent event) {
errorMessage.setText("");
Scene currentScene = dishesMenu.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("editorMain.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
EditorMainSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
@FXML
void exitButtonClicked(ActionEvent event) {
errorMessage.setText("");
MethodsSceneController.logoutAction(stage);
}
@FXML
void suppliesMenuButtonClicked(ActionEvent event) {
errorMessage.setText("");
Scene currentScene = suppliesMenu.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("editorSuppliesMenu.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
EditorSuppliesMenuSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
}

EditorDishEditSimpleSceneController.java

EditorIngredientEditSceneController.java

EditorMainSceneController.java

package com.example.client.employee.editor;
import com.example.client.Application;
import com.example.client.ConfirmationBox;
import com.example.client.MethodsSceneController;
import com.example.client.ServerCommunicator;
import com.example.client.employee.CurEmployee;
import com.example.client.items.Category;
import com.example.client.items.Dish;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.ButtonBar;
import javafx.scene.control.Label;
import javafx.scene.control.ScrollPane;
import javafx.scene.layout.GridPane;
import javafx.scene.layout.VBox;
import javafx.scene.text.Text;
import javafx.stage.Stage;
import java.io.IOException;
import java.util.ArrayList;
public class EditorMainSceneController {
private Stage stage;
private String curCategory;
public void start(Stage stage) {
errorMessage.setText("");
this.stage = stage;
showCategories();
curEmployeeFIO.setText(CurEmployee.getFio());
showListOfDishes("все");
curCategory = "все";
}
private void showCategories() {
ArrayList<Category> list = new ArrayList<>();
switch (ServerCommunicator.getAllCategories(list)) {
case "error": {
errorMessage.setText("Произошла ошибка");
}
case "denied": {
errorMessage.setText("Отказано в доступе");
}
}
for (int i = list.size() - 1; i >= 0; i--) {
Category category = list.get(i);
addButtonToBar(category.name);
}
addButtonToBar("все");
}
public void addButtonToBar(String buttonText) {
Button button = new Button(buttonText);
button.setOnAction(event -> {
showListOfDishes(buttonText);
curCategory = buttonText;
});
categories.getButtons().add(button);
}
public void showListOfDishes(String category) {
dishesContainer.getChildren().clear();
ArrayList<Dish> dishesList = new ArrayList<>();
// сообщение и список блюд выбранной категории
String message = ServerCommunicator.getDishesByCategory(dishesList, category);
switch (message) {
case "error": {
errorMessage.setText("Произошла ошибка.");
break;
}
case "deny": {
errorMessage.setText("Недостаточно прав. Перезайдите в аккаунт.");
break;
}
default: {
for (Dish dishItem : dishesList) {
GridPane gridPane = createGridPane(dishItem);
dishesContainer.getChildren().add(gridPane);
}
listOfEmployees.setFitToWidth(true);
listOfEmployees.setFitToHeight(true);
}
}
}
private GridPane createGridPane(Dish dish) {
try {
FXMLLoader fxmlLoader = new FXMLLoader(Application.class.getResource("itemDishSimple.fxml"));
GridPane gridPane = fxmlLoader.load();
// удалить
Button deleteCategoryButton = new Button("Удалить");
deleteCategoryButton.setOnAction(event -> {
switch (deleteDish(dish.id)) {
case "success": {
showListOfDishes(curCategory);
break;
}
case "denied": {
errorMessage.setText("Отказано в доступе");
break;
}
case "error": {
errorMessage.setText("Произошла ошибка");
break;
}
default:
errorMessage.setText("Произошла ошибка");
}
});
GridPane.setConstraints(deleteCategoryButton, 1, 0); // столбцы и строки
gridPane.getChildren().add(deleteCategoryButton);
// архивировать
String buttonName = "Архивировать";
if (dish.archived != 0) buttonName = "Показать";
Button archiveCategoryButton = new Button(buttonName);
archiveCategoryButton.setOnAction(event -> {
if (ConfirmationBox.showAlertBox("Изменение свойств блюда", "Вы уверены, что хотите изменить видимость блюда?")) {
switch (archiveDish(dish.id, dish.archived)) {
case "success": {
showListOfDishes(curCategory);
break;
}
case "denied": {
errorMessage.setText("Отказано в доступе");
break;
}
case "error": {
errorMessage.setText("Произошла ошибка");
break;
}
default:
errorMessage.setText("Произошла ошибка");
}
}
});
GridPane.setConstraints(archiveCategoryButton, 1, 1); // столбцы и строки
gridPane.getChildren().add(archiveCategoryButton);
// нет в наличии
String buttonNameStock = "Изъять из наличия";
if (dish.inStock != 1) buttonNameStock = "Вернуть в наличие";
Button stockCategoryButton = new Button(buttonNameStock);
stockCategoryButton.setOnAction(event -> {
if (ConfirmationBox.showAlertBox("Изменение свойств блюда", "Вы уверены, что хотите изменить свойство блюда?")) {
switch (inStockDish(dish.id, dish.inStock)) {
case "success": {
showListOfDishes(curCategory);
break;
}
case "denied": {
errorMessage.setText("Отказано в доступе");
break;
}
case "error": {
errorMessage.setText("Произошла ошибка");
break;
}
default:
errorMessage.setText("Произошла ошибка");
}
}
});
GridPane.setConstraints(stockCategoryButton, 1, 2); // столбцы и строки
gridPane.getChildren().add(stockCategoryButton);
// редактировать
Button configureCategoryButton = new Button("Изменить");
configureCategoryButton.setOnAction(event -> {
Scene currentScene = addDish.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("editorDishEditSimple.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
EditorDishEditSimpleSceneController controller = loader.getController();
controller.start(stage, dish.id, "Изменение блюда");
} catch (IOException e) {
e.printStackTrace();
}
});
GridPane.setConstraints(configureCategoryButton, 1, 3);
gridPane.getChildren().add(configureCategoryButton);
// название
Label labelName = new Label(dish.name);
GridPane.setConstraints(labelName, 0, 0);
gridPane.getChildren().add(labelName);
// количество
Label labelCount = new Label("Количество: " + dish.count);
GridPane.setConstraints(labelCount, 0, 1);
gridPane.getChildren().add(labelCount);
// статус
String status = "активно";
switch (dish.getStatus()) {
case "archived": {
status = "в архиве";
break;
}
case "notInStock": {
status = "нет в наличии";
break;
}
}
Label labelStatus = new Label("Статус: " + status);
GridPane.setConstraints(labelStatus, 0, 2);
gridPane.getChildren().add(labelStatus);
// цена
Label labelPrice = new Label("Цена: " + dish.price);
GridPane.setConstraints(labelPrice, 0, 3);
gridPane.getChildren().add(labelPrice);
return gridPane;
} catch (IOException e) {
e.printStackTrace();
}
return null;
}
private String inStockDish(int id, int inStock) {
return ServerCommunicator.inStockDish(id, inStock);
}
private String archiveDish(int id, int archived) {
return ServerCommunicator.archiveDish(id, archived);
}
private String deleteDish(int id) {
if (ConfirmationBox.showAlertBox("Удаление блюда", "Вы уверены, что хотите удалить блюдо без возможности восстановления?")) {
return ServerCommunicator.deleteDish(id);
}
return "";
}
@FXML
private ButtonBar categories;
@FXML
private Button categoriesMenu;
@FXML
private Text curEmployeeFIO;
@FXML
private VBox dishesContainer;
@FXML
private Button dishesMenu;
@FXML
private Text errorMessage;
@FXML
private Button exit;
@FXML
private ScrollPane listOfEmployees;
@FXML
private Button suppliesMenu;
@FXML
private Button addDish;
@FXML
void addDishButtonClicked(ActionEvent event) {
errorMessage.setText("");
Scene currentScene = categories.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("editorDishEditSimple.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
EditorDishEditSimpleSceneController controller = loader.getController();
controller.start(stage, -1, "Добавление блюда");
} catch (IOException e) {
e.printStackTrace();
}
}
@FXML
void categoriesMenuButtonClicked(ActionEvent event) {
errorMessage.setText("");
Scene currentScene = categories.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("editorCategoriesList.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
EditorCategoriesListSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
@FXML
void dishesMenuButtonClicked(ActionEvent event) {
errorMessage.setText("");
// обновление списка блюд
showListOfDishes("все");
}
@FXML
void exitButtonClicked(ActionEvent event) {
errorMessage.setText("");
MethodsSceneController.logoutAction(stage);
}
@FXML
void suppliesMenuButtonClicked(ActionEvent event) {
errorMessage.setText("");
Scene currentScene = suppliesMenu.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("editorSuppliesMenu.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
EditorSuppliesMenuSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
}

EditorSuppliesMenuSceneController.java

package com.example.client.employee.editor;
import com.example.client.Application;
import com.example.client.ConfirmationBox;
import com.example.client.MethodsSceneController;
import com.example.client.ServerCommunicator;
import com.example.client.employee.CurEmployee;
import com.example.client.items.Ingredient;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.scene.layout.GridPane;
import javafx.stage.Stage;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.layout.VBox;
import javafx.scene.text.Text;
import javafx.scene.control.ScrollPane;
import java.io.IOException;
import java.util.ArrayList;
public class EditorSuppliesMenuSceneController {
private Stage stage;
public void start(Stage stage) {
this.stage = stage;
curEmployeeFIO.setText(CurEmployee.getFio());
showListOfIngredients();
}
private void showListOfIngredients() {
ingredientsContainer.getChildren().clear();
ArrayList<Ingredient> ingredientList = new ArrayList<>();
// сообщение и список категорий
String message = ServerCommunicator.getAllIngridients(ingredientList);
switch (message) {
case "error": {
errorMessage.setText("Произошла ошибка.");
break;
}
case "deny": {
errorMessage.setText("Недостаточно прав. Перезайдите в аккаунт.");
break;
}
default: {
for (Ingredient ingredientItem : ingredientList) {
GridPane gridPane = createGridPane(ingredientItem.id, ingredientItem.fullName, ingredientItem.simpleName, ingredientItem.count, ingredientItem.unit);
ingredientsContainer.getChildren().add(gridPane);
}
listOfIngredients.setFitToWidth(true);
listOfIngredients.setFitToHeight(true);
}
}
}
private GridPane createGridPane(int id, String fullName, String simpleName, float count, String unit) {
try {
FXMLLoader fxmlLoader = new FXMLLoader(Application.class.getResource("itemIngredient.fxml"));
GridPane gridPane = fxmlLoader.load();
// удалить
Button deleteButton = new Button("Удалить");
deleteButton.setOnAction(event -> {
if (ConfirmationBox.showAlertBox("Удаление ингредиента", "Вы уверены, что хотите удалить ингредиент?")) {
String message = ServerCommunicator.deleteIngredient(id);
switch (message) {
case "error": {
errorMessage.setText("Произошла ошибка.");
break;
}
case "deny": {
errorMessage.setText("Недостаточно прав. Перезайдите в аккаунт.");
break;
}
}
showListOfIngredients();
}
});
GridPane.setConstraints(deleteButton, 4, 0); // столбцы и строки
gridPane.getChildren().add(deleteButton);
// редактировать
Button configureButton = new Button("Изменить");
configureButton.setOnAction(event -> {
Scene currentScene = addIngredient.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("editorIngredientEdit.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
EditorIngredientEditSceneController controller = loader.getController();
controller.start(stage, id, "Изменение ингредиента на складе");
} catch (IOException e) {
e.printStackTrace();
}
});
GridPane.setConstraints(configureButton, 4, 1); // столбцы и строки
gridPane.getChildren().add(configureButton);
// название в составе
Label labelSimpleName = new Label(simpleName);
GridPane.setConstraints(labelSimpleName, 0, 0); // столбцы и строки
gridPane.getChildren().add(labelSimpleName);
Label labelFullName = new Label(fullName);
GridPane.setConstraints(labelFullName, 1, 0); // столбцы и строки
gridPane.getChildren().add(labelFullName);
Label labelCount = new Label(Float.toString(count));
GridPane.setConstraints(labelCount, 2, 0); // столбцы и строки
gridPane.getChildren().add(labelCount);
Label labelUnit = new Label(unit);
GridPane.setConstraints(labelUnit, 3, 0); // столбцы и строки
gridPane.getChildren().add(labelUnit);
return gridPane;
} catch (IOException e) {
e.printStackTrace();
}
return null;
}
@FXML
private ScrollPane listOfIngredients;
@FXML
private Text curEmployeeFIO;
@FXML
private Button addIngredient;
@FXML
private Button categoriesMenu;
@FXML
private Button dishesMenu;
@FXML
private Text errorMessage;
@FXML
private Button exit;
@FXML
private VBox ingredientsContainer;
@FXML
private Button suppliesMenu;
@FXML
private Text title;
@FXML
void addIngredientButtonClicked(ActionEvent event) {
errorMessage.setText("");
Scene currentScene = addIngredient.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("editorIngredientEdit.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
EditorIngredientEditSceneController controller = loader.getController();
controller.start(stage, -1,"Добавление ингредиента");
} catch (IOException e) {
e.printStackTrace();
}
}
@FXML
void categoriesMenuButtonClicked(ActionEvent event) {
errorMessage.setText("");
Scene currentScene = categoriesMenu.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("editorCategoriesList.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
EditorCategoriesListSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
@FXML
void dishesMenuButtonClicked(ActionEvent event) {
errorMessage.setText("");
Scene currentScene = dishesMenu.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("editorMain.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
EditorMainSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
@FXML
void exitButtonClicked(ActionEvent event) {
errorMessage.setText("");
MethodsSceneController.logoutAction(stage);
}
@FXML
void suppliesMenuButtonClicked(ActionEvent event) {
errorMessage.setText("");
Scene currentScene = suppliesMenu.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("editorSuppliesMenu.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
EditorSuppliesMenuSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
}

Employee.java

package com.example.client.employee;
public class Employee {
public int id;
public String login;
public String password;
public String position;
public String fio;
public String telephone;
public Employee(int id, String fio, String position, String telephone) {
this.id = id;
this.fio = fio;
this.position = position;
this.telephone = telephone;
}
public Employee(String fio, String login, String password, String position, String telephone) {
this.fio = fio;
this.position = position;
this.telephone = telephone;
this.login = login;
this.password = password;
}
}

Category.java

package com.example.client.items;
public class Category {
public int id;
public String name;
public String description;
public int archived; // 0 or 1
public Category(int \_id, String \_name, String \_description, int \_archived) {
this.id = \_id;
this.name = \_name;
this.description = \_description;
this.archived = \_archived;
}
public Category(int \_id, String \_name, String \_description) {
this.id = \_id;
this.name = \_name;
this.description = \_description;
this.archived = 0;
}
public Category(int \_id, String \_name) {
this.id = \_id;
this.name = \_name;
}
}

Dish.java

package com.example.client.items;
import java.util.HashMap;
public class Dish {
public int id;
public String name;
public String description;
public HashMap<Integer, Ingredient> ingredientsList;
public float price;
public float weight;
public int id\_category;
public int count;
public int archived;
public int inStock;
public Dish(int id, String name, String description, HashMap<Integer, Ingredient> ingredientsList, float price, float weight, int id\_category, int count, int archived, int inStock) {
this.id = id;
this.name = name;
this.description = description;
this.ingredientsList = new HashMap<>(ingredientsList);
this.price = price;
this.weight = weight;
this.id\_category = id\_category;
this.count = count;
this.archived = archived;
this.inStock = inStock;
}
public Dish(int id, String name, String description, float price, float weight, int id\_category, int count, int archived, int inStock) {
this.id = id;
this.name = name;
this.description = description;
this.ingredientsList = new HashMap<>();
this.price = price;
this.weight = weight;
this.id\_category = id\_category;
this.count = count;
this.archived = archived;
this.inStock = inStock;
}
public void addIngredient(Ingredient ingredient) {
this.ingredientsList.put(ingredient.id, ingredient);
}
public void deleteIngredient(Ingredient ingredient) {
this.ingredientsList.remove(ingredient.id);
}
/\*\*
\* @return active, notInStock, archived
\*/
public String getStatus() {
if (archived == 1) {
return "archived";
}
if (inStock == 0 || count < 1) {
return "notInStock";
} else return "active";
}
}

Ingredient.java

package com.example.client.items;
public class Ingredient {
public int id;
public String simpleName;
public String fullName;
public float count;
public String unit; // единица измерения
/\*\*
\*
\* @param id
\* @param simpleName
\* @param fullName
\* @param count
\* @param unit
\*/
public Ingredient(int id, String simpleName, String fullName, float count, String unit) {
this.id = id;
this.simpleName = simpleName;
this.fullName = fullName;
this.count = count;
this.unit = unit;
}
}

Units.java

package com.example.client.items;
public class Units {
public int id;
public String name;
}

MethodsSceneController.java

package com.example.client;
import com.example.client.client.Client;
import com.example.client.employee.CurEmployee;
import com.example.client.employee.Employee;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;
import java.io.IOException;
public class MethodsSceneController {
public static void logoutAction(Stage stage) {
if (!ConfirmationBox.showAlertBox("Выход из аккаунта", "Вы уверены, что хотите выйти из аккаунта?")) {
return;
}
CurEmployee.clean();
Client.clean();
FXMLLoader loader = new FXMLLoader(Application.class.getResource("start.fxml"));
try {
Parent root = loader.load();
Scene newScene = new Scene(root);
stage.setScene(newScene);
} catch (IOException e) {
e.printStackTrace();
}
}
}

ProtocolBuilder.java

package com.example.client;
import com.example.client.client.Client;
import com.example.client.employee.Employee;
import com.example.client.items.Dish;
import com.example.client.items.Ingredient;
import com.google.gson.\*;
import java.util.HashMap;
import java.util.Map;
public class ProtocolBuilder {
public static String authEmployee(String login, String password) {
Gson gson = new Gson();
String action = "login";
int employee = 1;
JsonObject jsonObject = new JsonObject();
jsonObject.addProperty("action", action);
jsonObject.addProperty("login", login);
jsonObject.addProperty("password", password);
String json = gson.toJson(jsonObject);
return json;
}
public static String getEmployeesListRequest(String key) {
Gson gson = new Gson();
String action = "getEmployeesList";
JsonObject jsonObject = new JsonObject();
jsonObject.addProperty("action", action);
jsonObject.addProperty("key", key);
String json = gson.toJson(jsonObject);
return json;
}
public static String addEmployee(String key, Employee newEmp) {
Gson gson = new Gson();
String action = "addEmployee";
JsonObject jsonObject = new JsonObject();
jsonObject.addProperty("action", action);
jsonObject.addProperty("key", key);
jsonObject.addProperty("login", newEmp.login);
jsonObject.addProperty("password", newEmp.password);
jsonObject.addProperty("fio", newEmp.fio);
jsonObject.addProperty("position", newEmp.position);
jsonObject.addProperty("telephone", newEmp.telephone);
String json = gson.toJson(jsonObject);
return json;
}
public static String getEmployee(String key, int id) {
Gson gson = new Gson();
String action = "getEmployee";
JsonObject jsonObject = new JsonObject();
jsonObject.addProperty("action", action);
jsonObject.addProperty("key", key);
jsonObject.addProperty("id", id);
String json = gson.toJson(jsonObject);
return json;
}
public static String setEmployee(String key, int id, String fio, String position, String telephone, String login, String password) {
Gson gson = new Gson();
String action = "setEmployee";
JsonObject jsonObject = new JsonObject();
jsonObject.addProperty("action", action);
jsonObject.addProperty("key", key);
jsonObject.addProperty("id", Integer.toString(id));
jsonObject.addProperty("login", login);
if (password != null) jsonObject.addProperty("password", password);
jsonObject.addProperty("fio", fio);
jsonObject.addProperty("position", position);
jsonObject.addProperty("telephone", telephone);
String json = gson.toJson(jsonObject);
return json;
}
public static String getRestaurantInfo() {
Gson gson = new Gson();
JsonObject jsonObject = new JsonObject();
jsonObject.addProperty("action", "getRestaurantInfo");
String json = gson.toJson(jsonObject);
return json;
}
public static String setRestaurantInfo(String key, String name, String description, String telephone, String address) {
Gson gson = new Gson();
String action = "setRestaurantInfo";
JsonObject jsonObject = new JsonObject();
jsonObject.addProperty("action", action);
jsonObject.addProperty("key", key);
jsonObject.addProperty("name", name);
jsonObject.addProperty("description", description);
jsonObject.addProperty("telephone", telephone);
jsonObject.addProperty("address", address);
String json = gson.toJson(jsonObject);
return json;
}
public static String deleteEmployee(String sessionKey, int id) {
Gson gson = new Gson();
String action = "deleteEmployee";
JsonObject jsonObject = new JsonObject();
jsonObject.addProperty("action", action);
jsonObject.addProperty("key", sessionKey);
jsonObject.addProperty("id", id);
String json = gson.toJson(jsonObject);
return json;
}
//--------------------------------------//
public static String getAllCategories(String key) {
Gson gson = new Gson();
String action = "getAllCategories";
JsonObject jsonObject = new JsonObject();
jsonObject.addProperty("action", action);
jsonObject.addProperty("key", key);
String json = gson.toJson(jsonObject);
return json;
}
public static String getAllCategories() {
Gson gson = new Gson();
String action = "getAllCategories";
JsonObject jsonObject = new JsonObject();
jsonObject.addProperty("action", action);
String json = gson.toJson(jsonObject);
return json;
}
public static String getCategory(String key, int id) {
Gson gson = new Gson();
String action = "getCategory";
JsonObject jsonObject = new JsonObject();
jsonObject.addProperty("action", action);
jsonObject.addProperty("key", key);
jsonObject.addProperty("id", id);
String json = gson.toJson(jsonObject);
return json;
}
public static String changeCategory(String key, int id, String name, String description, int archived) {
Gson gson = new Gson();
String action = "changeCategory";
JsonObject jsonObject = new JsonObject();
jsonObject.addProperty("action", action);
jsonObject.addProperty("key", key);
jsonObject.addProperty("id", id);
jsonObject.addProperty("name", name);
jsonObject.addProperty("description", description);
jsonObject.addProperty("archived", archived);
String json = gson.toJson(jsonObject);
return json;
}
public static String addCategory(String key, String name, String description) {
Gson gson = new Gson();
String action = "addCategory";
JsonObject jsonObject = new JsonObject();
jsonObject.addProperty("action", action);
jsonObject.addProperty("key", key);
jsonObject.addProperty("name", name);
jsonObject.addProperty("description", description);
String json = gson.toJson(jsonObject);
return json;
}
public static String deleteCategory(String key, int id) {
Gson gson = new Gson();
String action = "deleteCategory";
JsonObject jsonObject = new JsonObject();
jsonObject.addProperty("action", action);
jsonObject.addProperty("key", key);
jsonObject.addProperty("id", id);
String json = gson.toJson(jsonObject);
return json;
}
public static String getAllUnits(String sessionKey) {
Gson gson = new Gson();
String action = "getAllUnits";
JsonObject jsonObject = new JsonObject();
jsonObject.addProperty("action", action);
jsonObject.addProperty("key", sessionKey);
String json = gson.toJson(jsonObject);
return json;
}
public static String getAllIngredients(String sessionKey) {
Gson gson = new Gson();
String action = "getAllIngredients";
JsonObject jsonObject = new JsonObject();
jsonObject.addProperty("action", action);
jsonObject.addProperty("key", sessionKey);
String json = gson.toJson(jsonObject);
return json;
}
public static String getIngredient(String sessionKey, int id) {
Gson gson = new Gson();
String action = "getIngredient";
JsonObject jsonObject = new JsonObject();
jsonObject.addProperty("action", action);
jsonObject.addProperty("key", sessionKey);
jsonObject.addProperty("id", id);
String json = gson.toJson(jsonObject);
return json;
}
public static String addIngredient(String sessionKey, String simpleNameValue, String fullNameValue, String countValue, int unitValue) {
Gson gson = new Gson();
String action = "addIngredient";
JsonObject jsonObject = new JsonObject();
jsonObject.addProperty("action", action);
jsonObject.addProperty("key", sessionKey);
jsonObject.addProperty("simple\_name", simpleNameValue);
jsonObject.addProperty("full\_name", fullNameValue);
jsonObject.addProperty("count", countValue);
jsonObject.addProperty("unit", unitValue);
String json = gson.toJson(jsonObject);
return json;
}
public static String changeIngredient(String sessionKey, int id, String simpleNameValue, String fullNameValue, String countValue, int unitValue) {
Gson gson = new Gson();
String action = "changeIngredient";
JsonObject jsonObject = new JsonObject();
jsonObject.addProperty("action", action);
jsonObject.addProperty("key", sessionKey);
jsonObject.addProperty("id", id);
jsonObject.addProperty("simple\_name", simpleNameValue);
jsonObject.addProperty("full\_name", fullNameValue);
jsonObject.addProperty("count", countValue);
jsonObject.addProperty("unit", unitValue);
String json = gson.toJson(jsonObject);
return json;
}
public static String deleteIngredient(String sessionKey, int id) {
Gson gson = new Gson();
String action = "deleteIngredient";
JsonObject jsonObject = new JsonObject();
jsonObject.addProperty("action", action);
jsonObject.addProperty("key", sessionKey);
jsonObject.addProperty("id", id);
String json = gson.toJson(jsonObject);
return json;
}
public static String getAllDishes(String sessionKey, String category) {
Gson gson = new Gson();
String action = "getAllDishes";
JsonObject jsonObject = new JsonObject();
jsonObject.addProperty("action", action);
jsonObject.addProperty("key", sessionKey);
jsonObject.addProperty("category", category);
String json = gson.toJson(jsonObject);
return json;
}
public static String addOrChangeDish(String sessionKey, Dish dish) {
Gson gson = new Gson();
String action = "addOrChangeDish";
JsonObject jsonObject = new JsonObject();
jsonObject.addProperty("action", action);
jsonObject.addProperty("key", sessionKey);
jsonObject.addProperty("id", dish.id);
jsonObject.addProperty("name", dish.name);
jsonObject.addProperty("description", dish.description);
JsonArray categoryArray = new JsonArray();
for (Ingredient ingredient : dish.ingredientsList.values()) {
JsonObject unitJson = new JsonObject();
unitJson.addProperty("id", ingredient.id);
unitJson.addProperty("count", ingredient.count);
categoryArray.add(unitJson);
}
jsonObject.addProperty("ingredients", categoryArray.toString());
jsonObject.addProperty("price", dish.price);
jsonObject.addProperty("weight", dish.weight);
jsonObject.addProperty("id\_category", dish.id\_category);
jsonObject.addProperty("archived", dish.archived);
jsonObject.addProperty("in\_stock", dish.inStock);
String json = gson.toJson(jsonObject);
return json;
}
public static String getDish(String sessionKey, int id) {
Gson gson = new Gson();
String action = "getDish";
JsonObject jsonObject = new JsonObject();
jsonObject.addProperty("action", action);
jsonObject.addProperty("key", sessionKey);
jsonObject.addProperty("id", id);
String json = gson.toJson(jsonObject);
return json;
}
public static String deleteDish(String sessionKey, int id) {
Gson gson = new Gson();
String action = "deleteDish";
JsonObject jsonObject = new JsonObject();
jsonObject.addProperty("action", action);
jsonObject.addProperty("key", sessionKey);
jsonObject.addProperty("id", id);
String json = gson.toJson(jsonObject);
return json;
}
public static String archiveDish(String sessionKey, int id, int archived) {
Gson gson = new Gson();
String action = "archiveDish";
JsonObject jsonObject = new JsonObject();
jsonObject.addProperty("action", action);
jsonObject.addProperty("key", sessionKey);
jsonObject.addProperty("id", id);
jsonObject.addProperty("archived", archived);
String json = gson.toJson(jsonObject);
return json;
}
public static String inStockDish(String sessionKey, int id, int inStock) {
Gson gson = new Gson();
String action = "inStockDish";
JsonObject jsonObject = new JsonObject();
jsonObject.addProperty("action", action);
jsonObject.addProperty("key", sessionKey);
jsonObject.addProperty("id", id);
jsonObject.addProperty("in\_stock", inStock);
String json = gson.toJson(jsonObject);
return json;
}
public static String getDishesByCategoryClient(String category) {
Gson gson = new Gson();
String action = "getAllDishes";
JsonObject jsonObject = new JsonObject();
jsonObject.addProperty("action", action);
jsonObject.addProperty("category", category);
String json = gson.toJson(jsonObject);
return json;
}
public static String client() {
Gson gson = new Gson();
String action = "client";
JsonObject jsonObject = new JsonObject();
jsonObject.addProperty("action", action);
String json = gson.toJson(jsonObject);
return json;
}
public static String getDishClient(int id) {
Gson gson = new Gson();
String action = "getDish";
JsonObject jsonObject = new JsonObject();
jsonObject.addProperty("action", action);
jsonObject.addProperty("id", id);
String json = gson.toJson(jsonObject);
return json;
}
}

ProtocolDecoder.java

ServerCommunicator.java

ServerConnection.java

package com.example.client;
import javafx.application.Platform;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;
import java.net.UnknownHostException;
import java.util.Timer;
import java.util.TimerTask;
public class ServerConnection {
private static Socket socket;
private static final String hostName = "localhost"; // путь к серверу
private static final int portNumber = 5000;
public static Socket connectToServer() {
try {
socket = new Socket(hostName, portNumber);
//System.out.println("Успешно подключено к серверу!");
return socket;
} catch (IOException e) {
//System.err.println("Не удалось подключиться к серверу: " + e.getMessage());
return null;
}
}
public static void disconnectFromServer() {
try {
if (socket != null && !socket.isClosed()) {
socket.close();
//System.out.println("Сокет успешно закрыт");
}
} catch (IOException e) {
//System.err.println("Ошибка при закрытии сокета: " + e.getMessage());
}
}
}

StartSceneController.java

package com.example.client;
import com.example.client.client.Client;
import com.example.client.client.ClientLoginSceneController;
import com.example.client.employee.AuthSceneController;
import com.example.client.employee.CurEmployee;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.AnchorPane;
import javafx.stage.Stage;
import java.io.IOException;
public class StartSceneController {
@FXML
private Button client;
@FXML
private Button employee;
@FXML
private AnchorPane scene;
@FXML
void clientButtonClicked(ActionEvent event) {
// Получаем сцену из кнопки
Scene currentScene = scene.getScene();
// Получаем окно (Stage) из текущей сцены
Stage stage = (Stage) currentScene.getWindow();
// Загружаем новую сцену из FXML-файла
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("clientLogin.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
ClientLoginSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
@FXML
void employeeButtonClicked(ActionEvent event) {
// Получаем сцену из кнопки
Scene currentScene = scene.getScene();
// Получаем окно (Stage) из текущей сцены
Stage stage = (Stage) currentScene.getWindow();
//stage.setFullScreen(true);
// Загружаем новую сцену из FXML-файла
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("auth.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
AuthSceneController controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
public void exitButtonClicked(ActionEvent actionEvent) {
}
public void start(Stage stage) {
CurEmployee.clean();
Client.clean();
}
}

module-info.java

module com.example.client {
requires javafx.controls;
requires javafx.fxml;
requires com.google.gson;
exports com.example.client;
exports com.example.client.employee;
exports com.example.client.employee.administrator;
exports com.example.client.employee.editor;
exports com.example.client.client;
exports com.example.client.items;
opens com.example.client to javafx.fxml;
opens com.example.client.client to javafx.fxml;
opens com.example.client.employee to javafx.fxml;
opens com.example.client.employee.administrator to javafx.fxml;
opens com.example.client.employee.editor to javafx.fxml;
}

ConfigFile.java

package com.example.server;
import java.io.\*;
import java.util.HashMap;
import java.util.Properties;
public class ConfigFile {
private static String folderPath = "config";
private static String filePath = folderPath + "/config.properties";
public static HashMap<String, String> configurations = new HashMap<>();
/\*\*
\* Открывает файл конфигурации для чтения и записи
\*
\* @return
\*/
public static File openConfigFile() {
File folder = new File(folderPath);
File configFile = new File(filePath);
if (!folder.exists()) {
if (folder.mkdir()) {
System.out.println("Папка 'config' создана");
} else {
System.out.println("Не удалось создать папку 'config'");
return null;
}
}
if (!configFile.exists()) {
try {
createConfigFile(configFile);
System.out.println("Конфигурационный файл создан: " + filePath);
return configFile;
} catch (IOException e) {
System.out.println("Не удалось создать конфигурационный файл: " + e.getMessage());
return null;
}
} else {
return configFile;
}
}
/\*\*
\* Читает конфигурации из файла конфигурации
\*
\* @return
\*/
public static HashMap<String, String> readConfigFile() {
if (openConfigFile() != null) {
Properties properties = new Properties();
try (InputStream input = new FileInputStream(filePath)) {
properties.load(input);
return convertToMap(properties);
} catch (IOException ex) {
ex.printStackTrace();
}
return null;
}
return null;
}
private static HashMap<String, String> convertToMap(Properties properties) {
for (String key : properties.stringPropertyNames()) {
configurations.put(key, properties.getProperty(key));
}
return configurations;
}
/\*\*
\* Изменяет файл конфигурации.
\*
\* @param newConfigurations
\*/
public static void changeConfigureFile(HashMap<String, String> newConfigurations) {
if (newConfigurations.get("dbHost") != null)
configurations.put("dbHost", newConfigurations.get("dbHost"));
if (newConfigurations.get("dbPort") != null)
configurations.put("dbPort", newConfigurations.get("dbPort"));
if (newConfigurations.get("dbName") != null)
configurations.put("dbName", newConfigurations.get("dbName"));
if (newConfigurations.get("dbUser") != null)
configurations.put("dbUser", newConfigurations.get("dbUser"));
if (newConfigurations.get("dbPass") != null)
configurations.put("dbPass", newConfigurations.get("dbPass"));
if (openConfigFile() != null) {
Properties properties = new Properties();
properties.putAll(configurations);
try (OutputStream output = new FileOutputStream(filePath)) {
properties.store(output, "Updated Config");
System.out.println("Файл конфигурации обновлен успешно!");
} catch (IOException e) {
e.printStackTrace();
}
}
}
/\*\*
\* Создаёт файл конфигурации.
\*
\* @param file
\* @throws IOException
\*/
private static void createConfigFile(File file) throws IOException {
configurations.put("dbHost", "localhost");
configurations.put("dbPort", "3306");
configurations.put("dbName", "mydb");
configurations.put("dbUser", "root");
configurations.put("dbPass", "");
Properties properties = new Properties();
properties.setProperty("dbHost", configurations.get("dbHost"));
properties.setProperty("dbPort", configurations.get("dbPort"));
properties.setProperty("dbName", configurations.get("dbName"));
properties.setProperty("dbUser", configurations.get("dbUser"));
properties.setProperty("dbPass", configurations.get("dbPass"));
OutputStream outputStream = null;
try {
outputStream = new FileOutputStream(file);
properties.store(outputStream, "Пример конфигурационного файла");
} catch (IOException e) {
e.printStackTrace();
} finally {
if (outputStream != null) {
try {
outputStream.close();
} catch (IOException e) {
e.printStackTrace();
}
}
}
}
}

Database.java

package com.example.server.database;
import java.sql.\*;
public class Database {
static String url = "jdbc:mysql://"; //"jdbc:mysql://localhost:3306/";
static String host;
static String port;
static String dbName;
static String username;
static String password;
static String[] tableNames = {"Users", "Employees", "Clients", "Restaurant", "Dishes", "Categories", "Orders", "FinishedOrders"};
public static String getUrl() {
return url;
}
public static String getPassword() {
return password;
}
public static String getUsername() {
return username;
}
public static String[] getTableNames() {
return tableNames;
}
public static boolean setValuesForConnection(String \_dbHost, String \_dbPort, String \_dbName, String \_username, String \_password) {
url += \_dbHost + ":" + \_dbPort + "/" + \_dbName;
host = \_dbHost;
port = \_dbPort;
dbName = \_dbName;
username = \_username;
password = \_password;
try (Connection connection = DriverManager.getConnection(url, username, password)) {
DatabaseMetaData metaData = connection.getMetaData();
/\*for (String tableName : tableNames) {
ResultSet resultSet = metaData.getTables(null, null, tableName, null);
if (!resultSet.next()) {
// Таблица не существует, создание таблицы
createTableUsers(tableName, connection);
} else {
System.out.println("Таблица " + tableName + " уже существует.");
}
}\*/
return true;
} catch (SQLException e) {
//e.printStackTrace();
return false;
}
}
}

DatabaseAdmin.java

DatabaseAuth.java

package com.example.server.database;
import com.example.server.PasswordHashing;
import java.io.ByteArrayInputStream;
import java.sql.\*;
import java.util.HashMap;
public class DatabaseAuth extends Database {
public static HashMap<String, String> checkLoginPassword(String login, String password) {
String sql = "SELECT id\_user, password FROM users WHERE login = ?";
HashMap<String, String> messageKey = new HashMap<>();
String hashedPassword = null;
String key = null;
String position = null;
String fio = null;
int id = -1;
try (Connection connection = DriverManager.getConnection(Database.url, Database.username, Database.password)) {
PreparedStatement statement = connection.prepareStatement(sql);
statement.setString(1, login);
ResultSet resultSet = statement.executeQuery();
// ищем пользователя с таким логином
if (resultSet.next()) {
hashedPassword = resultSet.getString("password");
id = resultSet.getInt("id\_user");
}
if (id == -1) { // Пользователя с таким логином нет
messageKey.put("message", "noSuchUser");
return messageKey;
}
// проверяем соответствие логина и пароля
if (PasswordHashing.verifyPassword(password, hashedPassword)) {
key = PasswordHashing.hashPassword(password);
// узнаём должность
if (isEditor(id, connection)) {
position = "editor";
} else if (isAdministrator(id, connection)) {
position = "administrator";
}
if (position != null) fio = getFIOById(position, id, connection);
// устанавливаем session key
if (updateSessionKey(position, id, key, connection)) {
messageKey.put("key", key);
messageKey.put("message", position);
messageKey.put("id", Integer.toString(id));
messageKey.put("fio", fio);
connection.close();
return messageKey;
} else {
connection.close();
messageKey.put("message", "failedAuth");
return messageKey;
}
} else { // Несоответствие логина и пароля
messageKey.put("message", "failedAuth");
return messageKey;
}
} catch (SQLException e) {
e.printStackTrace();
messageKey.put("message", "error");
return messageKey;
}
}
private static boolean updateSessionKey(String table, int id, String key, Connection connection) throws SQLException {
if (id < 0) {
return false;
}
String sql = "UPDATE " + table + "s SET session\_key = ? WHERE id\_user = ?";
PreparedStatement preparedStatement = connection.prepareStatement(sql);
preparedStatement.setString(1, key);
preparedStatement.setInt(2, id);
int rowsAffected = preparedStatement.executeUpdate();
if (rowsAffected > 0) {
//System.out.println("Запись успешно обновлена");
return true;
} else {
//System.out.println("Запись не была обновлена");
return false;
}
}
public static String getFIOById(String table, int id, Connection connection) throws SQLException {
String sql = "SELECT fio FROM " + table + "s WHERE id\_user = ?";
String fio = null;
PreparedStatement statement = connection.prepareStatement(sql);
statement.setInt(1, id);
ResultSet resultSet = statement.executeQuery();
if (resultSet.next()) { // обнаружен в таблице редакторов
fio = resultSet.getString("fio");
}
return fio;
}
public static boolean isEditor(int id, Connection connection) throws SQLException {
String sql = "SELECT fio FROM editors WHERE id\_user = ?";
PreparedStatement statement = connection.prepareStatement(sql);
statement.setInt(1, id);
ResultSet resultSet = statement.executeQuery();
if (resultSet.next()) { // обнаружен в таблице редакторов
return true;
}
return false;
}
public static boolean isAdministrator(int id, Connection connection) throws SQLException {
String sql = "SELECT fio FROM administrators WHERE id\_user = ?";
PreparedStatement statement = connection.prepareStatement(sql);
statement.setInt(1, id);
ResultSet resultSet = statement.executeQuery();
if (resultSet.next()) { // обнаружен в таблице администраторов
return true;
}
return false;
}
}

DatabaseClients.java

DatabaseEditor.java

Menu.java

package com.example.server.menu;
import com.example.server.ConfigFile;
import java.util.HashMap;
public class Menu {
public static boolean menu() {
while (true) {
int action = MenuView.showMenu();
switch (action) {
case 0: {
return true;
}
case 1: {
HashMap <String, String> configurations = new HashMap<>();
while (true) {
configurations.put("dbHost", MenuView.getHost());
int port;
if ((port = MenuView.getPort()) == -1) {
break;
}
configurations.put("dbPort", Integer.toString(port));
configurations.put("dbName", MenuView.getDBName());
configurations.put("dbUser", MenuView.getLogin());
configurations.put("dbPass", MenuView.getPassword());
if (MenuView.getAnswerToSave()){
ConfigFile.changeConfigureFile(configurations);
return true;
} else {
break;
}
}
break;
}
case 2: {
return false;
}
default: {
return false;
}
}
}
}
}

MenuView.java

package com.example.server.menu;
import java.util.Scanner;
public class MenuView {
public static int showMenu() {
System.out.print("\033[H\033[2J");
System.out.flush();
System.out.println("Добро пожаловать в систему управления меню и заказов");
System.out.println("Выберите действие:");
System.out.println("0 - Продолжить запуск сервера");
System.out.println("1 - Подключиться к базе данных (необходимы адрес, порт, имя базы данных, логин и пароль)");
System.out.println("2 - Завершить работу");
return getIntInRange(0, 2);
}
public static String getString() {
Scanner scanner = new Scanner(System.in);
return scanner.nextLine();
}
private static int getIntInRange(int minValue, int maxValue) {
Scanner scanner = new Scanner(System.in);
int number = 0;
boolean isValid = false;
while (true) {
if (scanner.hasNextInt()) {
number = scanner.nextInt();
if (number >= minValue && number <= maxValue) {
return number;
} else {
System.out.println("Введено некорректное значение.");
if (!getAnswerToContinue()) {
return minValue-1;
}
}
} else {
System.out.println("Введено некорректное значение.");
scanner.next();
if (!getAnswerToContinue()) {
return minValue-1;
}
}
}
}
public static String getHost() {
System.out.print("Введите имя хоста или IP-адрес базы данных: ");
return getString();
}
public static int getPort() {
System.out.print("Введите номер порта (от 0 до 65535): ");
return getIntInRange(0, 65535);
}
public static String getDBName() {
System.out.print("Введите имя базы данных: ");
return getString();
}
public static String getLogin() {
System.out.print("Введите логин: ");
return getString();
}
public static String getPassword() {
System.out.print("Введите пароль: ");
return getString();
}
public static boolean getAnswerToContinue() {
while (true) {
System.out.println("Желаете попробовать снова? [д/н]");
String decision = getString();
if (decision != null) {
if (decision.equals("д") || decision.equals("Д")) {
return true;
} else if (decision.equals("н") || decision.equals("Н")) {
return false;
}
}
System.out.println("Ответ неопределён. Попробуйте снова.");
}
}
public static boolean getAnswerToShutDown() {
while (true) {
System.out.println("Вы уверены, что хотите завершить работу сервера? [д/н]");
String decision = getString();
if (decision != null) {
if (decision.equals("д") || decision.equals("Д")) {
return true;
} else if (decision.equals("н") || decision.equals("Н")) {
return false;
}
}
System.out.println("Ответ неопределён. Попробуйте снова.");
}
}
public static boolean getAnswerToSave() {
while (true) {
System.out.println("Желаете применить внесённые данные? [д/н]");
String decision = getString();
if (decision != null) {
if (decision.equals("д") || decision.equals("Д")) {
return true;
} else if (decision.equals("н") || decision.equals("Н")) {
return false;
}
}
System.out.println("Ответ неопределён. Попробуйте снова.");
}
}
}

Category.java

package com.example.server.models;
public class Category {
public int id;
public String name;
public String description;
public int archived; // 0 or 1
public Category(int \_id, String \_name, String \_description, int \_archived) {
this.id = \_id;
this.name = \_name;
this.description = \_description;
this.archived = \_archived;
}
public Category(int \_id, String \_name, String \_description) {
this.id = \_id;
this.name = \_name;
this.description = \_description;
this.archived = 0;
}
}

Dish.java

package com.example.server.models;
import java.util.HashMap;
public class Dish {
public int id;
public String name;
public String description;
public HashMap<Integer, Ingredient> ingredientsList;
public float price;
public float weight;
public int id\_category;
public int count;
public int archived;
public int inStock;
public Dish() {
this.id = 0;
this.name = "";
this.description = "";
this.ingredientsList = new HashMap<>();
this.price = 0;
this.weight = 0;
this.id\_category = -1;
this.count = 0;
this.archived = 0;
this.inStock = 0;
}
public Dish(int id, String name, String description, HashMap<Integer, Ingredient> ingredientsList, float price, float weight, int id\_category, int count, int archived, int inStock) {
this.id = id;
this.name = name;
this.description = description;
this.ingredientsList = new HashMap<>(ingredientsList);
this.price = price;
this.weight = weight;
this.id\_category = id\_category;
this.count = count;
this.archived = archived;
this.inStock = inStock;
}
public Dish(int id, String name, String description, float price, float weight, int id\_category, int count, int archived, int inStock) {
this.id = id;
this.name = name;
this.description = description;
this.ingredientsList = new HashMap<>();
this.price = price;
this.weight = weight;
this.id\_category = id\_category;
this.count = count;
this.archived = archived;
this.inStock = inStock;
}
/\*\*
\* @return active, notInStock, archived
\*/
public String getStatus() {
if (archived == 1) {
return "archived";
}
if (inStock == 0) {
return "notInStock";
} else return "active";
}
}

Employee.java

package com.example.server.models;
public class Employee {
public int id;
public String login;
public String password;
public String position;
public String fio;
public String telephone;
// u.id\_user, u.login, a.fio, a.telephone
public Employee(int id, String fio, String position, String telephone) {
this.id = id;
this.fio = fio;
this.position = position;
this.telephone = telephone;
}
}

Ingredient.java

package com.example.server.models;
public class Ingredient {
public int id;
public String simpleName;
public String fullName;
public float count;
public String unit; // единица измерения
public Ingredient(int id, String simpleName, String fullName, float count, String unit) {
this.id = id;
this.simpleName = simpleName;
this.fullName = fullName;
this.count = count;
this.unit = unit;
}
public Ingredient(int id, float count) {
this.id = id;
this.count = count;
}
}

PasswordHashing.java

package com.example.server;
import org.mindrot.jbcrypt.BCrypt;
public class PasswordHashing {
public static String hashPassword(String password) {
return BCrypt.hashpw(password, BCrypt.gensalt());
}
public static boolean verifyPassword(String password, String hashedPassword) {
return BCrypt.checkpw(password, hashedPassword);
}
}

ProtocolReceiver.java

ProtocolSender.java

Server.java

package com.example.server;
import com.example.server.database.Database;
import com.example.server.database.DatabaseAdmin;
import com.example.server.menu.Menu;
import com.example.server.menu.MenuView;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.HashMap;
public class Server {
private static boolean isServerRunning = true;
private static ServerSocket serverSocket;
private static boolean connectToDB() {
HashMap<String, String> configs = ConfigFile.readConfigFile();
return Database.setValuesForConnection(configs.get("dbHost"), configs.get("dbPort"), configs.get("dbName"), configs.get("dbUser"), configs.get("dbPass"));
}
public static void main(String[] args) {
ConfigFile.openConfigFile(); // создание файла конфигурации для работы с сервером
if (!Menu.menu()) {
System.out.println("Работа программы завершена.");
return;
}
if (!connectToDB()) { // настройка подключения с базой данных (чтение данных из конфигурационного файла)
System.out.println("Ошибка соединения с базой данных. Поменяйте данные для подключения при следующем запуске программы.");
return;
}
// Поток для чтения с консоли
Thread consoleThread = new Thread(() -> {
BufferedReader consoleReader = new BufferedReader(new InputStreamReader(System.in));
try {
while (true) {
String userInput = consoleReader.readLine();
if (userInput.equals("2")) {
if (MenuView.getAnswerToShutDown()) {
isServerRunning = false;
break;
}
}
}
serverSocket.close();
} catch (IOException e) {
//e.printStackTrace();
}
});
consoleThread.start();
int portNumber = 5000;
try (ServerSocket \_serverSocket = new ServerSocket(portNumber)) {
System.out.println("Сервер запущен и готов к подключению клиентов.");
serverSocket = \_serverSocket;
while (isServerRunning) {
Socket clientSocket = serverSocket.accept(); // Ожидание подключения клиента
System.out.println("Клиент подключен: " + clientSocket);
// Запуск обработки соединения в отдельном потоке
Thread clientThread = new Thread(() -> handleClient(clientSocket));
clientThread.start();
}
} catch (IOException e) {
//e.printStackTrace();
}
}
private static void handleClient(Socket clientSocket) {
try {
PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true);
BufferedReader in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
String inputLine;
while ((inputLine = in.readLine()) != null) {
System.out.println("Клиент" + clientSocket + ": " + inputLine);
ProtocolReceiver.identify(inputLine, clientSocket);
}
} catch (IOException e) {
//e.printStackTrace();
}
}
}

module-info.java

module com.example.server {
requires javafx.controls;
requires javafx.fxml;
requires java.sql;
requires jbcrypt;
requires com.google.gson;
opens com.example.server to javafx.fxml;
exports com.example.server;
exports com.example.server.database;
exports com.example.server.models;
opens com.example.server.database to javafx.fxml;
exports com.example.server.menu;
opens com.example.server.menu to javafx.fxml;
}

ConfigFile.java

package com.example.server;
import java.io.\*;
import java.util.HashMap;
import java.util.Properties;
public class ConfigFile {
private static String folderPath = "config";
private static String filePath = folderPath + "/config.properties";
public static HashMap<String, String> configurations = new HashMap<>();
/\*\*
\* Открывает файл конфигурации для чтения и записи
\*
\* @return
\*/
public static File openConfigFile() {
File folder = new File(folderPath);
File configFile = new File(filePath);
if (!folder.exists()) {
if (folder.mkdir()) {
System.out.println("Папка 'config' создана");
} else {
System.out.println("Не удалось создать папку 'config'");
return null;
}
}
if (!configFile.exists()) {
try {
createConfigFile(configFile);
System.out.println("Конфигурационный файл создан: " + filePath);
return configFile;
} catch (IOException e) {
System.out.println("Не удалось создать конфигурационный файл: " + e.getMessage());
return null;
}
} else {
return configFile;
}
}
/\*\*
\* Читает конфигурации из файла конфигурации
\*
\* @return
\*/
public static HashMap<String, String> readConfigFile() {
if (openConfigFile() != null) {
Properties properties = new Properties();
try (InputStream input = new FileInputStream(filePath)) {
properties.load(input);
return convertToMap(properties);
} catch (IOException ex) {
ex.printStackTrace();
}
return null;
}
return null;
}
private static HashMap<String, String> convertToMap(Properties properties) {
for (String key : properties.stringPropertyNames()) {
configurations.put(key, properties.getProperty(key));
}
return configurations;
}
/\*\*
\* Изменяет файл конфигурации.
\*
\* @param newConfigurations
\*/
public static void changeConfigureFile(HashMap<String, String> newConfigurations) {
if (newConfigurations.get("dbHost") != null)
configurations.put("dbHost", newConfigurations.get("dbHost"));
if (newConfigurations.get("dbPort") != null)
configurations.put("dbPort", newConfigurations.get("dbPort"));
if (newConfigurations.get("dbName") != null)
configurations.put("dbName", newConfigurations.get("dbName"));
if (newConfigurations.get("dbUser") != null)
configurations.put("dbUser", newConfigurations.get("dbUser"));
if (newConfigurations.get("dbPass") != null)
configurations.put("dbPass", newConfigurations.get("dbPass"));
if (openConfigFile() != null) {
Properties properties = new Properties();
properties.putAll(configurations);
try (OutputStream output = new FileOutputStream(filePath)) {
properties.store(output, "Updated Config");
System.out.println("Файл конфигурации обновлен успешно!");
} catch (IOException e) {
e.printStackTrace();
}
}
}
/\*\*
\* Создаёт файл конфигурации.
\*
\* @param file
\* @throws IOException
\*/
private static void createConfigFile(File file) throws IOException {
configurations.put("dbHost", "localhost");
configurations.put("dbPort", "3306");
configurations.put("dbName", "mydb");
configurations.put("dbUser", "root");
configurations.put("dbPass", "");
Properties properties = new Properties();
properties.setProperty("dbHost", configurations.get("dbHost"));
properties.setProperty("dbPort", configurations.get("dbPort"));
properties.setProperty("dbName", configurations.get("dbName"));
properties.setProperty("dbUser", configurations.get("dbUser"));
properties.setProperty("dbPass", configurations.get("dbPass"));
OutputStream outputStream = null;
try {
outputStream = new FileOutputStream(file);
properties.store(outputStream, "Пример конфигурационного файла");
} catch (IOException e) {
e.printStackTrace();
} finally {
if (outputStream != null) {
try {
outputStream.close();
} catch (IOException e) {
e.printStackTrace();
}
}
}
}
}

Database.java

package com.example.server.database;
import java.sql.\*;
public class Database {
static String url = "jdbc:mysql://"; //"jdbc:mysql://localhost:3306/";
static String host;
static String port;
static String dbName;
static String username;
static String password;
static String[] tableNames = {"Users", "Employees", "Clients", "Restaurant", "Dishes", "Categories", "Orders", "FinishedOrders"};
public static String getUrl() {
return url;
}
public static String getPassword() {
return password;
}
public static String getUsername() {
return username;
}
public static String[] getTableNames() {
return tableNames;
}
public static boolean setValuesForConnection(String \_dbHost, String \_dbPort, String \_dbName, String \_username, String \_password) {
url += \_dbHost + ":" + \_dbPort + "/" + \_dbName;
host = \_dbHost;
port = \_dbPort;
dbName = \_dbName;
username = \_username;
password = \_password;
try (Connection connection = DriverManager.getConnection(url, username, password)) {
DatabaseMetaData metaData = connection.getMetaData();
/\*for (String tableName : tableNames) {
ResultSet resultSet = metaData.getTables(null, null, tableName, null);
if (!resultSet.next()) {
// Таблица не существует, создание таблицы
createTableUsers(tableName, connection);
} else {
System.out.println("Таблица " + tableName + " уже существует.");
}
}\*/
return true;
} catch (SQLException e) {
//e.printStackTrace();
return false;
}
}
}

DatabaseAdmin.java

DatabaseAuth.java

package com.example.server.database;
import com.example.server.PasswordHashing;
import java.io.ByteArrayInputStream;
import java.sql.\*;
import java.util.HashMap;
public class DatabaseAuth extends Database {
public static HashMap<String, String> checkLoginPassword(String login, String password) {
String sql = "SELECT id\_user, password FROM users WHERE login = ?";
HashMap<String, String> messageKey = new HashMap<>();
String hashedPassword = null;
String key = null;
String position = null;
String fio = null;
int id = -1;
try (Connection connection = DriverManager.getConnection(Database.url, Database.username, Database.password)) {
PreparedStatement statement = connection.prepareStatement(sql);
statement.setString(1, login);
ResultSet resultSet = statement.executeQuery();
// ищем пользователя с таким логином
if (resultSet.next()) {
hashedPassword = resultSet.getString("password");
id = resultSet.getInt("id\_user");
}
if (id == -1) { // Пользователя с таким логином нет
messageKey.put("message", "noSuchUser");
return messageKey;
}
// проверяем соответствие логина и пароля
if (PasswordHashing.verifyPassword(password, hashedPassword)) {
key = PasswordHashing.hashPassword(password);
// узнаём должность
if (isEditor(id, connection)) {
position = "editor";
} else if (isAdministrator(id, connection)) {
position = "administrator";
}
if (position != null) fio = getFIOById(position, id, connection);
// устанавливаем session key
if (updateSessionKey(position, id, key, connection)) {
messageKey.put("key", key);
messageKey.put("message", position);
messageKey.put("id", Integer.toString(id));
messageKey.put("fio", fio);
connection.close();
return messageKey;
} else {
connection.close();
messageKey.put("message", "failedAuth");
return messageKey;
}
} else { // Несоответствие логина и пароля
messageKey.put("message", "failedAuth");
return messageKey;
}
} catch (SQLException e) {
e.printStackTrace();
messageKey.put("message", "error");
return messageKey;
}
}
private static boolean updateSessionKey(String table, int id, String key, Connection connection) throws SQLException {
if (id < 0) {
return false;
}
String sql = "UPDATE " + table + "s SET session\_key = ? WHERE id\_user = ?";
PreparedStatement preparedStatement = connection.prepareStatement(sql);
preparedStatement.setString(1, key);
preparedStatement.setInt(2, id);
int rowsAffected = preparedStatement.executeUpdate();
if (rowsAffected > 0) {
//System.out.println("Запись успешно обновлена");
return true;
} else {
//System.out.println("Запись не была обновлена");
return false;
}
}
public static String getFIOById(String table, int id, Connection connection) throws SQLException {
String sql = "SELECT fio FROM " + table + "s WHERE id\_user = ?";
String fio = null;
PreparedStatement statement = connection.prepareStatement(sql);
statement.setInt(1, id);
ResultSet resultSet = statement.executeQuery();
if (resultSet.next()) { // обнаружен в таблице редакторов
fio = resultSet.getString("fio");
}
return fio;
}
public static boolean isEditor(int id, Connection connection) throws SQLException {
String sql = "SELECT fio FROM editors WHERE id\_user = ?";
PreparedStatement statement = connection.prepareStatement(sql);
statement.setInt(1, id);
ResultSet resultSet = statement.executeQuery();
if (resultSet.next()) { // обнаружен в таблице редакторов
return true;
}
return false;
}
public static boolean isAdministrator(int id, Connection connection) throws SQLException {
String sql = "SELECT fio FROM administrators WHERE id\_user = ?";
PreparedStatement statement = connection.prepareStatement(sql);
statement.setInt(1, id);
ResultSet resultSet = statement.executeQuery();
if (resultSet.next()) { // обнаружен в таблице администраторов
return true;
}
return false;
}
}

DatabaseClients.java

DatabaseEditor.java

Menu.java

package com.example.server.menu;
import com.example.server.ConfigFile;
import java.util.HashMap;
public class Menu {
public static boolean menu() {
while (true) {
int action = MenuView.showMenu();
switch (action) {
case 0: {
return true;
}
case 1: {
HashMap <String, String> configurations = new HashMap<>();
while (true) {
configurations.put("dbHost", MenuView.getHost());
int port;
if ((port = MenuView.getPort()) == -1) {
break;
}
configurations.put("dbPort", Integer.toString(port));
configurations.put("dbName", MenuView.getDBName());
configurations.put("dbUser", MenuView.getLogin());
configurations.put("dbPass", MenuView.getPassword());
if (MenuView.getAnswerToSave()){
ConfigFile.changeConfigureFile(configurations);
return true;
} else {
break;
}
}
break;
}
case 2: {
return false;
}
default: {
return false;
}
}
}
}
}

MenuView.java

package com.example.server.menu;
import java.util.Scanner;
public class MenuView {
public static int showMenu() {
System.out.print("\033[H\033[2J");
System.out.flush();
System.out.println("Добро пожаловать в систему управления меню и заказов");
System.out.println("Выберите действие:");
System.out.println("0 - Продолжить запуск сервера");
System.out.println("1 - Подключиться к базе данных (необходимы адрес, порт, имя базы данных, логин и пароль)");
System.out.println("2 - Завершить работу");
return getIntInRange(0, 2);
}
public static String getString() {
Scanner scanner = new Scanner(System.in);
return scanner.nextLine();
}
private static int getIntInRange(int minValue, int maxValue) {
Scanner scanner = new Scanner(System.in);
int number = 0;
boolean isValid = false;
while (true) {
if (scanner.hasNextInt()) {
number = scanner.nextInt();
if (number >= minValue && number <= maxValue) {
return number;
} else {
System.out.println("Введено некорректное значение.");
if (!getAnswerToContinue()) {
return minValue-1;
}
}
} else {
System.out.println("Введено некорректное значение.");
scanner.next();
if (!getAnswerToContinue()) {
return minValue-1;
}
}
}
}
public static String getHost() {
System.out.print("Введите имя хоста или IP-адрес базы данных: ");
return getString();
}
public static int getPort() {
System.out.print("Введите номер порта (от 0 до 65535): ");
return getIntInRange(0, 65535);
}
public static String getDBName() {
System.out.print("Введите имя базы данных: ");
return getString();
}
public static String getLogin() {
System.out.print("Введите логин: ");
return getString();
}
public static String getPassword() {
System.out.print("Введите пароль: ");
return getString();
}
public static boolean getAnswerToContinue() {
while (true) {
System.out.println("Желаете попробовать снова? [д/н]");
String decision = getString();
if (decision != null) {
if (decision.equals("д") || decision.equals("Д")) {
return true;
} else if (decision.equals("н") || decision.equals("Н")) {
return false;
}
}
System.out.println("Ответ неопределён. Попробуйте снова.");
}
}
public static boolean getAnswerToShutDown() {
while (true) {
System.out.println("Вы уверены, что хотите завершить работу сервера? [д/н]");
String decision = getString();
if (decision != null) {
if (decision.equals("д") || decision.equals("Д")) {
return true;
} else if (decision.equals("н") || decision.equals("Н")) {
return false;
}
}
System.out.println("Ответ неопределён. Попробуйте снова.");
}
}
public static boolean getAnswerToSave() {
while (true) {
System.out.println("Желаете применить внесённые данные? [д/н]");
String decision = getString();
if (decision != null) {
if (decision.equals("д") || decision.equals("Д")) {
return true;
} else if (decision.equals("н") || decision.equals("Н")) {
return false;
}
}
System.out.println("Ответ неопределён. Попробуйте снова.");
}
}
}

Category.java

package com.example.server.models;
public class Category {
public int id;
public String name;
public String description;
public int archived; // 0 or 1
public Category(int \_id, String \_name, String \_description, int \_archived) {
this.id = \_id;
this.name = \_name;
this.description = \_description;
this.archived = \_archived;
}
public Category(int \_id, String \_name, String \_description) {
this.id = \_id;
this.name = \_name;
this.description = \_description;
this.archived = 0;
}
}

Dish.java

package com.example.server.models;
import java.util.HashMap;
public class Dish {
public int id;
public String name;
public String description;
public HashMap<Integer, Ingredient> ingredientsList;
public float price;
public float weight;
public int id\_category;
public int count;
public int archived;
public int inStock;
public Dish() {
this.id = 0;
this.name = "";
this.description = "";
this.ingredientsList = new HashMap<>();
this.price = 0;
this.weight = 0;
this.id\_category = -1;
this.count = 0;
this.archived = 0;
this.inStock = 0;
}
public Dish(int id, String name, String description, HashMap<Integer, Ingredient> ingredientsList, float price, float weight, int id\_category, int count, int archived, int inStock) {
this.id = id;
this.name = name;
this.description = description;
this.ingredientsList = new HashMap<>(ingredientsList);
this.price = price;
this.weight = weight;
this.id\_category = id\_category;
this.count = count;
this.archived = archived;
this.inStock = inStock;
}
public Dish(int id, String name, String description, float price, float weight, int id\_category, int count, int archived, int inStock) {
this.id = id;
this.name = name;
this.description = description;
this.ingredientsList = new HashMap<>();
this.price = price;
this.weight = weight;
this.id\_category = id\_category;
this.count = count;
this.archived = archived;
this.inStock = inStock;
}
/\*\*
\* @return active, notInStock, archived
\*/
public String getStatus() {
if (archived == 1) {
return "archived";
}
if (inStock == 0) {
return "notInStock";
} else return "active";
}
}

Employee.java

package com.example.server.models;
public class Employee {
public int id;
public String login;
public String password;
public String position;
public String fio;
public String telephone;
// u.id\_user, u.login, a.fio, a.telephone
public Employee(int id, String fio, String position, String telephone) {
this.id = id;
this.fio = fio;
this.position = position;
this.telephone = telephone;
}
}

Ingredient.java

package com.example.server.models;
public class Ingredient {
public int id;
public String simpleName;
public String fullName;
public float count;
public String unit; // единица измерения
public Ingredient(int id, String simpleName, String fullName, float count, String unit) {
this.id = id;
this.simpleName = simpleName;
this.fullName = fullName;
this.count = count;
this.unit = unit;
}
public Ingredient(int id, float count) {
this.id = id;
this.count = count;
}
}

PasswordHashing.java

package com.example.server;
import org.mindrot.jbcrypt.BCrypt;
public class PasswordHashing {
public static String hashPassword(String password) {
return BCrypt.hashpw(password, BCrypt.gensalt());
}
public static boolean verifyPassword(String password, String hashedPassword) {
return BCrypt.checkpw(password, hashedPassword);
}
}

ProtocolReceiver.java

ProtocolSender.java

Server.java

package com.example.server;
import com.example.server.database.Database;
import com.example.server.database.DatabaseAdmin;
import com.example.server.menu.Menu;
import com.example.server.menu.MenuView;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.HashMap;
public class Server {
private static boolean isServerRunning = true;
private static ServerSocket serverSocket;
private static boolean connectToDB() {
HashMap<String, String> configs = ConfigFile.readConfigFile();
return Database.setValuesForConnection(configs.get("dbHost"), configs.get("dbPort"), configs.get("dbName"), configs.get("dbUser"), configs.get("dbPass"));
}
public static void main(String[] args) {
ConfigFile.openConfigFile(); // создание файла конфигурации для работы с сервером
if (!Menu.menu()) {
System.out.println("Работа программы завершена.");
return;
}
if (!connectToDB()) { // настройка подключения с базой данных (чтение данных из конфигурационного файла)
System.out.println("Ошибка соединения с базой данных. Поменяйте данные для подключения при следующем запуске программы.");
return;
}
// Поток для чтения с консоли
Thread consoleThread = new Thread(() -> {
BufferedReader consoleReader = new BufferedReader(new InputStreamReader(System.in));
try {
while (true) {
String userInput = consoleReader.readLine();
if (userInput.equals("2")) {
if (MenuView.getAnswerToShutDown()) {
isServerRunning = false;
break;
}
}
}
serverSocket.close();
} catch (IOException e) {
//e.printStackTrace();
}
});
consoleThread.start();
int portNumber = 5000;
try (ServerSocket \_serverSocket = new ServerSocket(portNumber)) {
System.out.println("Сервер запущен и готов к подключению клиентов.");
serverSocket = \_serverSocket;
while (isServerRunning) {
Socket clientSocket = serverSocket.accept(); // Ожидание подключения клиента
System.out.println("Клиент подключен: " + clientSocket);
// Запуск обработки соединения в отдельном потоке
Thread clientThread = new Thread(() -> handleClient(clientSocket));
clientThread.start();
}
} catch (IOException e) {
//e.printStackTrace();
}
}
private static void handleClient(Socket clientSocket) {
try {
PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true);
BufferedReader in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
String inputLine;
while ((inputLine = in.readLine()) != null) {
System.out.println("Клиент" + clientSocket + ": " + inputLine);
ProtocolReceiver.identify(inputLine, clientSocket);
}
} catch (IOException e) {
//e.printStackTrace();
}
}
}

module-info.java

module com.example.server {
requires javafx.controls;
requires javafx.fxml;
requires java.sql;
requires jbcrypt;
requires com.google.gson;
opens com.example.server to javafx.fxml;
exports com.example.server;
exports com.example.server.database;
exports com.example.server.models;
opens com.example.server.database to javafx.fxml;
exports com.example.server.menu;
opens com.example.server.menu to javafx.fxml;
}