AggregationCore.java

AuthenticationValidator.java

package com.alligator.alligator.client;
import com.alligator.alligator.database.IDatabase;
import java.util.List;
public class AuthenticationValidator {
private final IDatabase database;
public AuthenticationValidator(IDatabase database)
{
this.database = database;
}
public boolean isAuthDataValid(String login, String password)
{
User user = database.readUserByName(login);
String passwordHash = HashProvider.getStringHash(password);
return (passwordHash.equals(user.getPasswordHash()));
}
}

HashProvider.java

package com.alligator.alligator.client;
import com.google.common.base.Charsets;
import com.google.common.hash.HashCode;
import com.google.common.hash.Hasher;
import com.google.common.hash.Hashing;
public class HashProvider
{
public static String getStringHash(String inputString)
{
Hasher hasher = Hashing.sha256().newHasher();
hasher.putString(inputString, Charsets.UTF\_8);
HashCode sha256 = hasher.hash();
return sha256.toString();
}
}

User.java

UserRole.java

package com.alligator.alligator.client;
public enum UserRole
{
USER,
MANAGER
}

Database.java

IDatabase.java

package com.alligator.alligator.database;
import com.alligator.alligator.client.User;
import com.alligator.alligator.domain.Sprint;
import com.alligator.alligator.domain.Task;
import java.util.List;
public interface IDatabase {
public void insertTask(Task task);
public Task readTask(int id);
public List<Task> getAllTasks();
public void updateTask(int id, Task task);
public void deleteTask(int id);
public void insertSprint(Sprint sprint);
public Sprint readSprint(int id);
public List<Sprint> getAllSprints();
public void updateSprint(int id, Sprint sprint);
public void deleteSprint(int id);
public void insertUser(User user);
public User readUser(int id);
public User readUserByName(String name);
public List<User> getAllUsers();
public void updateUser(int id, User user);
public void deleteUser(int id);
}

MockDatabase.java

MockStorage.java

Sprint.java

package com.alligator.alligator.domain;
import jakarta.persistence.\*;
import java.io.Serializable;
import java.util.\*;
@Entity
public class Sprint implements Serializable
{
@Id
@GeneratedValue
private int id;
public int getId() {
return id;
}
public void setId(int id) {
this.id = id;
}
private String name;
public String getName() {
return name;
}
public void setName(String name) {
this.name = name;
}
@OneToMany(cascade = CascadeType.ALL)
private Set<Task> assignedTasks = new HashSet<Task>();
public Set<Task> getAssignedTasks() {
return assignedTasks;
}
public void setAssignedTasks(Set<Task> assignedTasks) {
for (Task t: assignedTasks)
{
this.assignedTasks.add(t);
t.setSprint(this);
}
}
private boolean isActive;
public void assignTask(Task task)
{
assignedTasks.add(task);
task.setSprint(this);
}
public boolean isEmpty()
{
return assignedTasks.isEmpty();
}
public boolean isActive()
{
return isActive;
}
public void toggleActive()
{
isActive = !isActive;
}
public int getTotalComplexity()
{
int complexity = 0;
for (Task task: assignedTasks)
{
complexity += task.getComplexity();
}
return complexity;
}
private Date dueDate;
public Date getDueDate()
{
return dueDate;
}
public void setDueDate(Date date)
{
dueDate = date;
}
}

SprintController.java

package com.alligator.alligator.domain;
import com.alligator.alligator.database.Database;
import com.alligator.alligator.database.IDatabase;
import java.util.ArrayList;
public class SprintController
{
private IDatabase database;
private ArrayList<Sprint> sprints = new ArrayList<Sprint>();
public SprintController(IDatabase db)
{
database = db;
}
public void createSprint()
{
}
public ArrayList<Sprint> getSprintList()
{
return sprints;
}
}

Task.java

package com.alligator.alligator.domain;
import com.alligator.alligator.client.User;
import jakarta.persistence.\*;
import java.io.Serializable;
import java.util.Set;
@Entity
public class Task implements Serializable
{
@Id
@GeneratedValue
private int id;
public int getId() {
return id;
}
public void setId(int id) {
this.id = id;
}
private String name;
public String getName() {
return name;
}
public void setName(String name) {
this.name = name;
}
private boolean completed = false;
public boolean isCompleted() {
return completed;
}
public void setCompleted(boolean completed) {
this.completed = completed;
}
private int complexity = 0;
private TaskPriority priority = TaskPriority.LOW;
public TaskPriority getPriority() {
return priority;
}
public void setPriority(TaskPriority priority) {
this.priority = priority;
}
@ManyToOne
private Sprint sprint = null;
@ManyToMany
private Set<User> assignees = null;
public Set<User> getAssignees()
{
return assignees;
}
public void setAssignees(Set<User> user)
{
assignees = user;
}
public Sprint getSprint() {
return sprint;
}
public void setSprint(Sprint sprint) {
this.sprint = sprint;
}
public int getComplexity()
{
return complexity;
}
public void setComplexity(int complexity){ this.complexity = complexity; }
public boolean isUnassigned()
{
return (sprint == null);
}
}

TaskController.java

TaskPriority.java

package com.alligator.alligator.domain;
public enum TaskPriority
{
LOW,
MEDIUM,
HIGH
}

HelloApplication.java

package com.alligator.alligator;
import com.alligator.alligator.automation.AggregationCore;
import com.alligator.alligator.client.User;
import com.alligator.alligator.client.UserRole;
import com.alligator.alligator.database.Database;
import com.alligator.alligator.database.IDatabase;
import com.alligator.alligator.database.MockDatabase;
import com.alligator.alligator.domain.Sprint;
import com.alligator.alligator.domain.Task;
import com.alligator.alligator.domain.TaskPriority;
import com.alligator.alligator.sceneControllers.\*;
import com.alligator.alligator.sceneControllers.main.AuthFormSceneController;
import com.alligator.alligator.sceneControllers.main.MainMenuSceneController;
import com.alligator.alligator.sceneControllers.main.RegistrationSceneController;
import com.alligator.alligator.sceneControllers.main.WorkerSceneController;
import com.alligator.alligator.sceneControllers.sprints.AddTasksSprintSceneController;
import com.alligator.alligator.sceneControllers.sprints.ListSprintsSceneController;
import com.alligator.alligator.sceneControllers.tasks.AddTasksSceneController;
import com.alligator.alligator.sceneControllers.tasks.AssignTaskSceneController;
import com.alligator.alligator.sceneControllers.tasks.ListTasksSceneController;
import javafx.application.Application;
import javafx.scene.image.Image;
import javafx.stage.Stage;
import java.util.HashMap;
import java.util.List;
public class HelloApplication extends Application
{
public Stage stage;
public User currentUser;
public IDatabase database;
public AggregationCore aggregationCore;
public Task currentTaskOnEdit = null;
@Override
public void start(Stage stage) throws Exception
{
database = new Database();
aggregationCore = new AggregationCore(database);
if (database instanceof MockDatabase)
{
System.out.println("INFO: Program is running in mock mode. Switch it");
}
this.stage = stage;
sceneControllers = new HashMap<>();
addSceneController("main", new MainSceneController(this));
addSceneController("next", new NextSceneController(this));
addSceneController("mainMenuScene", new MainMenuSceneController(this));
addSceneController("listTasksScene", new ListTasksSceneController(this));
addSceneController("listSprintsScene", new ListSprintsSceneController(this));
addSceneController("addSprintScene", new AddSprintSceneController(this));
addSceneController("addTasksScene", new AddTasksSceneController(this));
addSceneController("sprintInfoScreen", new SprintInfoScreenController(this));
addSceneController("authScene", new AuthFormSceneController(this));
addSceneController("registration", new RegistrationSceneController(this));
addSceneController("assignTask", new AssignTaskSceneController(this));
addSceneController("assignTaskInSprint", new AddTasksSprintSceneController(this));
addSceneController("assignmentOfTasks", new AssignmentOfTasksController(this));
addSceneController("workerScene", new WorkerSceneController(this));
setScene("authScene");
stage.setTitle("Alligator");
stage.setMinHeight(400);
stage.setMinWidth(300);
stage.getIcons().add(new Image("logo.png"));
stage.show();
}
public static void main(String[] args)
{
launch(args);
}
static HashMap<String, SceneController> sceneControllers;
public void addSceneController(String name, SceneController sceneController)
{
sceneControllers.put(name, sceneController);
}
public void setScene(String name)
{
SceneController sceneController = sceneControllers.get(name);
sceneController.update();
stage.setScene(sceneController.getScene());
}
}

HelloController.java

package com.alligator.alligator;
import com.alligator.alligator.automation.AggregationCore;
import com.alligator.alligator.database.Database;
import com.alligator.alligator.domain.Sprint;
import com.alligator.alligator.domain.SprintController;
import com.alligator.alligator.domain.Task;
import com.alligator.alligator.domain.TaskController;
import javafx.fxml.FXML;
import javafx.scene.control.Label;
public class HelloController
{
// public AggregationCore core = new AggregationCore();
public Database database = new Database();
@FXML
private Label welcomeText;
@FXML
protected void onHelloButtonClick()
{
Sprint sprint = new Sprint();
sprint.setName("test123");
Task task = new Task();
task.setName("test");
sprint.assignTask(task);
database.insertTask(task);
welcomeText.setText("deleted");
}
}

SceneController.java

package com.alligator.alligator;
import javafx.scene.Scene;
import javafx.stage.Stage;
public abstract class SceneController {
protected Stage stage;
protected Scene scene;
protected HelloApplication helloApplication;
public SceneController(HelloApplication helloApplication){
this.helloApplication = helloApplication;
this.stage = helloApplication.stage;
}
public Scene getScene(){
return scene;
}
public abstract void update();
}

AddSprintSceneController.java

package com.alligator.alligator.sceneControllers;
import com.alligator.alligator.HelloApplication;
import com.alligator.alligator.SceneController;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.VBox;
import javafx.geometry.Insets;
import javafx.scene.layout.GridPane;
public class AddSprintSceneController extends SceneController {
GridPane gridPane;
VBox vBox;
Label nameLabel, durationLabel, sceneLabel;
TextField nameField, durationField;
Button backButton, addButton;
public AddSprintSceneController(HelloApplication helloApplication) {
super(helloApplication);
addButton = new Button("Add");
addButton.setOnAction(new EventHandler<ActionEvent>(){
@Override
public void handle(ActionEvent event) {
String name = nameField.getText();
String duration = durationField.getText();
System.out.println("нужна база данных");
//addItemToFlowPane(items, name, duration);
nameField.clear();
durationField.clear();
}
});
backButton = new Button("Back");
backButton.setOnAction(event -> helloApplication.setScene("listSprintsScene"));
vBox = new VBox();
sceneLabel = new Label("Добавить спринт");
gridPane = new GridPane();
gridPane.setPadding(new Insets(20));
gridPane.setHgap(10);
gridPane.setVgap(10);
nameLabel = new Label("Имя");
nameField = new TextField();
nameField.setPromptText("Enter a name");
durationLabel = new Label("Длительность");
durationField = new TextField();
durationField.setPromptText("Enter a duration");
// Размещаем элементы в GridPane
gridPane.add(nameLabel, 0, 0);
gridPane.add(nameField, 1, 0);
gridPane.add(durationLabel, 0, 1);
gridPane.add(durationField, 1, 1);
//gridPane.add(backButton, 0, 2, 2, 1); // Кнопку "Back" размещаем на две ячейки в ширину
gridPane.add(addButton, 0, 2);
gridPane.add(backButton, 1, 2); // Кнопку "Back" размещаем на две ячейки в ширину
vBox.getChildren().addAll(sceneLabel, gridPane);
scene = new Scene(vBox, 400, 500);
}
@Override
public void update() {}
}

AssignmentOfTasksController.java

FlowListItem.java

package com.alligator.alligator.sceneControllers;
import javafx.event.EventHandler;
import javafx.geometry.Pos;
import javafx.scene.control.Label;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.VBox;
public class FlowListItem extends VBox {
Label captionLabel, descriptionLabel;
public FlowListItem(MainSceneController mainSceneController, String caption, String description){
captionLabel = new Label(caption);
captionLabel.setAlignment(Pos.BASELINE\_LEFT);
captionLabel.setStyle("-fx-font-size: 16px; -fx-font-weight: bold;");
descriptionLabel = new Label(description);
descriptionLabel.setAlignment(Pos.CENTER\_LEFT);
// Create a VBox to hold the caption and description labels
getChildren().addAll(captionLabel, descriptionLabel);
setOnMouseClicked(new EventHandler<MouseEvent>() {
@Override
public void handle(MouseEvent mouseEvent) {
mainSceneController.captionField.setText(captionLabel.getText());
mainSceneController.descriptionField.setText(descriptionLabel.getText());
}
});
}
public void edit(String caption, String description){
captionLabel.setText(caption);
descriptionLabel.setText(description);
}
}

AuthFormSceneController.java

package com.alligator.alligator.sceneControllers.main;
import com.alligator.alligator.HelloApplication;
import com.alligator.alligator.SceneController;
import com.alligator.alligator.client.AuthenticationValidator;
import com.alligator.alligator.client.User;
import com.alligator.alligator.client.UserRole;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
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.PasswordField;
import javafx.scene.control.TextField;
public class AuthFormSceneController extends SceneController {
Parent root;
TextField login;
PasswordField password;
Button submit, register;
Label wrongPassword;
public AuthFormSceneController(HelloApplication helloApplication){
super(helloApplication);
try {
root = FXMLLoader.load(helloApplication.getClass().getResource("auth-view.fxml"));
login = (TextField)root.lookup("#login");
password = (PasswordField) root.lookup("#password");
submit = (Button)root.lookup("#submit");
register = (Button) root.lookup("#register");
wrongPassword = (Label) root.lookup("#wrong");
wrongPassword.setVisible(false);
submit.setOnAction(new EventHandler<ActionEvent>() {
@Override
public void handle(ActionEvent actionEvent) {
System.out.println(login.getText() + ", " + password.getText());
if (isLoginDataValid(login.getText(), password.getText()))
{
helloApplication.currentUser = helloApplication.database.readUserByName(login.getText());
if (helloApplication.currentUser.getRole() == UserRole.MANAGER)
{
helloApplication.setScene("mainMenuScene");
}
else
{
helloApplication.setScene("workerScene");
}
wrongPassword.setVisible(false);
}
else
{
password.clear();
wrongPassword.setVisible(true);
}
}
});
register.setOnAction(new EventHandler<ActionEvent>() {
@Override
public void handle(ActionEvent actionEvent) {
helloApplication.setScene("registration");
}
});
}
catch (Exception ex){
ex.printStackTrace();
}
scene = new Scene(root);
}
@Override
public void update() {
}
private boolean isLoginDataValid(String login, String password)
{
AuthenticationValidator validator = new AuthenticationValidator(helloApplication.database);
return validator.isAuthDataValid(login, password);
}
}

MainMenuSceneController.java

package com.alligator.alligator.sceneControllers.main;
import com.alligator.alligator.HelloApplication;
import com.alligator.alligator.SceneController;
import com.alligator.alligator.client.UserRole;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
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.TextField;
import javafx.scene.layout.VBox;
public class MainMenuSceneController extends SceneController {
Parent root;
Button toTasksList, toSprintsList, assignButton, exitButton;
Label wrongLabel;
public MainMenuSceneController(HelloApplication helloApplication){
super(helloApplication);
try {
root = FXMLLoader.load(helloApplication.getClass().getResource("mainMenu-view.fxml"));
toTasksList = (Button) root.lookup("#tasks");
toSprintsList = (Button) root.lookup("#sprints");
exitButton = (Button)root.lookup("#exit");
assignButton = (Button) root.lookup("#autoAssign");
wrongLabel = (Label) root.lookup("#wrong");
toTasksList.setOnAction(new EventHandler<ActionEvent>(){
@Override
public void handle(ActionEvent event) {
helloApplication.setScene("listTasksScene");
}
});
toSprintsList.setOnAction(new EventHandler<ActionEvent>(){
@Override
public void handle(ActionEvent event) {
helloApplication.setScene("listSprintsScene");
}
});
exitButton.setOnAction(new EventHandler<ActionEvent>() {
@Override
public void handle(ActionEvent actionEvent) {
helloApplication.currentUser = null;
helloApplication.setScene("authScene");
}
});
assignButton.setOnAction(new EventHandler<ActionEvent>() {
@Override
public void handle(ActionEvent actionEvent) {
if (helloApplication.currentUser.getRole() == UserRole.MANAGER)
{
if (helloApplication.database.getAllSprints().size() == 0) {
wrongLabel.setVisible(true);
}
else {
helloApplication.aggregationCore.proceedTasksAssignation();
helloApplication.setScene("assignmentOfTasks");
}
}
else
{
helloApplication.setScene("workerScene");
}
}
});
}
catch (Exception ex){
ex.printStackTrace();
}
scene = new Scene(root);
}
@Override
public void update() {
wrongLabel.setVisible(false);
}
}

RegistrationSceneController.java

package com.alligator.alligator.sceneControllers.main;
import com.alligator.alligator.HelloApplication;
import com.alligator.alligator.SceneController;
import com.alligator.alligator.client.User;
import com.alligator.alligator.client.UserRole;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.\*;
public class RegistrationSceneController extends SceneController {
Parent root;
TextField name, login;
PasswordField password;
Button registerButton, backButton;
public RegistrationSceneController(HelloApplication helloApplication){
super(helloApplication);
try {
root = FXMLLoader.load(helloApplication.getClass().getResource("babyRegistration-view.fxml"));
name = (TextField) root.lookup("#name");
login = (TextField) root.lookup("#login");
password = (PasswordField) root.lookup("#password");
CheckBox isManager = (CheckBox) root.lookup("#isManager");
registerButton = (Button) root.lookup("#register");
backButton = (Button) root.lookup("#back");
registerButton.setOnAction(new EventHandler<ActionEvent>() {
@Override
public void handle(ActionEvent actionEvent)
{
String nameText = name.getText();
String loginText = login.getText();
String passwordPlaintext = password.getText();
User user = new User(nameText, loginText, passwordPlaintext);
if (isManager.isSelected())
{
user.setRole(UserRole.MANAGER);
}
helloApplication.database.insertUser(user);
helloApplication.setScene("authScene");
}
});
backButton.setOnAction(new EventHandler<ActionEvent>() {
@Override
public void handle(ActionEvent actionEvent) {
helloApplication.setScene("authScene");
}
});
}
catch (Exception ex){
ex.printStackTrace();
}
scene = new Scene(root);
}
@Override
public void update() {
}
}

WorkerSceneController.java

MainSceneController.java

NextSceneController.java

package com.alligator.alligator.sceneControllers;
import com.alligator.alligator.HelloApplication;
import com.alligator.alligator.SceneController;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.layout.VBox;
public class NextSceneController extends SceneController {
VBox root;
Label helloLabel;
Button addButton, editButton, deleteButton, clearButton, nextButton, backButton;
public NextSceneController(HelloApplication helloApplication){
super(helloApplication);
backButton = new Button("Back");
backButton.setOnAction(new EventHandler<ActionEvent>(){
@Override
public void handle(ActionEvent event) {
helloApplication.setScene("main");
}
});
root = new VBox();
helloLabel = new Label("Hello, world!\nwritten by Quartyom");
root.getChildren().addAll(helloLabel, backButton);
scene = new Scene(root, 400, 500);
}
@Override
public void update() {
}
}

SimpleFlowListItem.java

package com.alligator.alligator.sceneControllers;
import javafx.event.EventHandler;
import javafx.geometry.Pos;
import javafx.scene.control.Label;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.VBox;
public class SimpleFlowListItem extends VBox {
Label captionLabel, descriptionLabel;
public SimpleFlowListItem(String caption, String description){
captionLabel = new Label(caption);
captionLabel.setAlignment(Pos.BASELINE\_LEFT);
captionLabel.setStyle("-fx-font-size: 16px; -fx-font-weight: bold;");
descriptionLabel = new Label(description);
descriptionLabel.setAlignment(Pos.CENTER\_LEFT);
// Create a VBox to hold the caption and description labels
getChildren().addAll(captionLabel, descriptionLabel);
}
public void edit(String caption, String description){
captionLabel.setText(caption);
descriptionLabel.setText(description);
}
}

SprintInfoScreenController.java

package com.alligator.alligator.sceneControllers;
import com.alligator.alligator.HelloApplication;
import com.alligator.alligator.SceneController;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.layout.VBox;
public class SprintInfoScreenController extends SceneController {
VBox root;
Label sceneLabel;
Button backButton;
public SprintInfoScreenController(HelloApplication helloApplication) {
super(helloApplication);
backButton = new Button("Back");
backButton.setOnAction(new EventHandler<ActionEvent>() {
@Override
public void handle(ActionEvent event) {
helloApplication.setScene("listSprintsScene");
}
});
root = new VBox();
sceneLabel = new Label("Инфа о спринте");
root.getChildren().addAll(sceneLabel, backButton);
scene = new Scene(root, 300, 400);
}
@Override
public void update() {
}
}

SprintListItem.java

package com.alligator.alligator.sceneControllers;
import com.alligator.alligator.HelloApplication;
import javafx.event.EventHandler;
import javafx.geometry.Pos;
import javafx.scene.control.Label;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.VBox;
public class SprintListItem extends VBox {
Label captionLabel, descriptionLabel;
public SprintListItem(HelloApplication helloApplication, String caption, String description){
captionLabel = new Label(caption);
captionLabel.setAlignment(Pos.BASELINE\_LEFT);
captionLabel.setStyle("-fx-font-size: 16px; -fx-font-weight: bold;");
descriptionLabel = new Label(description);
descriptionLabel.setAlignment(Pos.CENTER\_LEFT);
// Create a VBox to hold the caption and description labels
getChildren().addAll(captionLabel, descriptionLabel);
setOnMouseClicked(new EventHandler<MouseEvent>() {
@Override
public void handle(MouseEvent mouseEvent) {
helloApplication.setScene("sprintInfoScreen");
}
});
}
public void edit(String caption, String description){
captionLabel.setText(caption);
descriptionLabel.setText(description);
}
}

AddTasksSprintSceneController.java

ListSprintsSceneController.java

AddTasksSceneController.java

package com.alligator.alligator.sceneControllers.tasks;
import com.alligator.alligator.HelloApplication;
import com.alligator.alligator.SceneController;
import com.alligator.alligator.domain.Task;
import com.alligator.alligator.domain.TaskPriority;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.fxml.FXMLLoader;
import javafx.geometry.Insets;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.ChoiceBox;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.GridPane;
import javafx.scene.layout.VBox;
public class AddTasksSceneController extends SceneController {
TextField nameField, complexityField;
ChoiceBox<TaskPriority> priorityChoice;
Button backButton, assignButton;
Parent root;
public AddTasksSceneController(HelloApplication helloApplication) {
super(helloApplication);
try {
root = FXMLLoader.load(helloApplication.getClass().getResource("addTaskToBacklog-view.fxml"));
nameField = (TextField) root.lookup("#name");
priorityChoice = (ChoiceBox<TaskPriority>) root.lookup("#priority");
complexityField = (TextField) root.lookup("#complexity");
backButton = (Button) root.lookup("#back");
assignButton = (Button) root.lookup("#assign");
backButton.setOnAction(event -> helloApplication.setScene("listTasksScene"));
assignButton.setOnAction(new EventHandler<ActionEvent>() {
@Override
public void handle(ActionEvent actionEvent) {
Task task = new Task();
task.setName(nameField.getText());
task.setComplexity(Integer.parseInt(complexityField.getText()));
TaskPriority priority = priorityChoice.getValue();
task.setPriority(priority != null ? priority : TaskPriority.LOW);
helloApplication.currentTaskOnEdit = task;
helloApplication.setScene("assignTask");
}
});
priorityChoice.getItems().addAll(TaskPriority.LOW, TaskPriority.MEDIUM, TaskPriority.HIGH);
}
catch (Exception ex){
ex.printStackTrace();
}
scene = new Scene(root);
}
@Override
public void update() {
}
}

AssignTaskSceneController.java

ListTasksSceneController.java

module-info.java

module com.alligator.alligator {
requires javafx.controls;
requires javafx.fxml;
requires com.dlsc.formsfx;
requires org.hibernate.orm.core;
requires jakarta.persistence;
requires java.naming;
requires com.google.common;
opens com.alligator.alligator.domain to org.hibernate.orm.core;
opens com.alligator.alligator to javafx.fxml;
opens com.alligator.alligator.client to org.hibernate.orm.core;
exports com.alligator.alligator;
exports com.alligator.alligator.domain;
}

AirConditioner.java

AirPurifier.java

package org.example.devices;
import com.google.gson.Gson;
import org.example.Menu;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.text.ParsePosition;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Random;
import java.util.Scanner;
public class AirPurifier extends SmartDevice {
// Additional (4)
private String mode;
private String speedOfFan;
private final int noiseLevel;
private String filterReplacementDate = null;
public AirPurifier(String \_manufacturer, String \_model, String \_powerSupply, String \_macAddress, int \_noiseLevel, String \_mode, String \_speedOfFan, String \_filterReplacementDate) {
super(\_manufacturer, \_model, \_powerSupply, \_macAddress);
type = "AirPurifier";
noiseLevel = \_noiseLevel;
mode = \_mode;
speedOfFan=\_speedOfFan;
filterReplacementDate = \_filterReplacementDate;
}
public String getMode() {
return mode;
}
public String getSpeedOfFan() {
return speedOfFan;
}
public String getFilterReplacementDate() {
return filterReplacementDate;
}
// from SmartDevice
@Override
public void turnOn() {
if (!getOnOrOff()) {
if (getConnectedToPowerSupply()) {
setOnOrOff(true);
System.out.println("Очиститель воздуха " + getMacAddress() + " включен.");
} else {
setOnOrOff(false);
System.out.println("Очиститель воздуха " + getMacAddress() + " не может быть включен, так как отключён от энергопитания.");
}
} else {
if (getConnectedToPowerSupply()) {
setOnOrOff(true);
System.out.println("Очиститель воздуха " + getMacAddress() + " уже включен.");
} else {
setOnOrOff(false);
System.out.println("Очиститель воздуха " + getMacAddress() + " не может быть включен, так как отключён от энергопитания.");
}
}
}
@Override
public void turnOff() {
if (getOnOrOff()) {
setOnOrOff(false);
System.out.println("Очиститель воздуха " + getMacAddress() + " выключен.");
} else {
System.out.println("Очиститель воздуха " + getMacAddress() + " уже выключен.");
}
}
@Override
public void connectToPowerSupply() {
if (getConnectedToPowerSupply()) {
System.out.println("Очиститель воздуха " + getMacAddress() + " уже подключен к электропитанию.");
} else {
setConnectedToPowerSupply(true);
System.out.println("Очиститель воздуха " + getMacAddress() + " подключен к электропитанию.");
}
}
@Override
public void disconnectFromPowerSupply() {
if (getConnectedToPowerSupply()) {
setOnOrOff(false);
setConnectedToPowerSupply(false);
System.out.println("Очиститель воздуха " + getMacAddress() + " отключен от электропитания.");
} else {
setOnOrOff(false);
setConnectedToPowerSupply(false);
System.out.println("Очиститель воздуха " + getMacAddress() + "уже отключен от электропитания.");
}
}
@Override
public void showStatus() {
System.out.println("Состояние очистителя воздуха ID" + id + ":");
if (getOnOrOff()) {
System.out.println("\tВключен.");
System.out.println("\tУровень шума: " + noiseLevel);
System.out.println("\tУстановленный режим: " + mode);
System.out.println("\tСкорость вентилятора: " + speedOfFan);
System.out.println("\tДата последней замены фильтров: " + filterReplacementDate);
showInternetConnectionStatus();
} else {
System.out.println("\tВыключен.");
if (getConnectedToPowerSupply()) System.out.println("\tПодключен к электропитанию.");
else System.out.println("\tОтключен от электропитания.");
System.out.println("\tДата последней замены фильтров: " + filterReplacementDate);
}
}
@Override
public void showInternetConnectionStatus() {
if (getOnOrOff()) {
if (getInternetConnection()) {
Random random = new Random();
int internetStatus = random.nextInt(3) + 1;
System.out.print("\tЕсть доступ к Интернету. Качество соединения: ");
switch (internetStatus) {
case 1: {
System.out.println("плохое.");
break;
}
case 2: {
System.out.println("хорошее.");
break;
}
case 3: {
System.out.println("отличное.");
break;
}
default: {
System.out.println("неопределено.");
break;
}
}
} else {
System.out.println("\tОтсутствует доступ к Интернету.");
}
} else {
System.out.println("Выключен.");
System.out.println("\tОтсутствует доступ к Интернету.");
}
}
@Override
public void configure() {
System.out.println("Выберите настройку: ");
System.out.println("\t0 - Подключить к сети Интернет");
System.out.println("\t1 - Изменить режим");
System.out.println("\t2 - Изменить скорость вентилятора");
System.out.println("\t3 - Заменить фильтр");
switch (Menu.getInt(3)) {
case 0: {
changeInternetConnection();
break;
}
case 1: {
changeMode();
break;
}
case 2: {
changeSpeedOfFan();
break;
}
case 3: {
changeFilterReplacementDate();
break;
}
default: {
}
}
}
@Override
public void showInfo() {
showStatus();
System.out.println("\tТип устройства: очиститель воздуха");
System.out.println("\tПроизводитель: " + getManufacturer());
System.out.println("\tМодель: " + getModel());
System.out.println("\tИсточник электропитания: " + getPowerSupply());
System.out.println("\tMAC-адрес: " + getMacAddress());
}
// Additional (4)
// Режим
private void changeMode() {
mode = Menu.readAirPurifierMode();
System.out.println("Режим успешно изменен!");
}
public void changeMode(String tempMode) {
if (tempMode.equals("обогрев") || tempMode.equals("вентиляция") || tempMode.equals("экономия энергии") || tempMode.equals("работа с влажностью") || tempMode.equals("другой")) {
mode = tempMode;
System.out.println("Режим успешно изменен!");
} else {
System.out.println("Режим не был изменен.");
}
}
// Скорость вентилятора
private void changeSpeedOfFan() {
speedOfFan = Menu.readFanSpeed();
System.out.println("Скорость вентилятора успешно изменена!");
}
public void changeSpeedOfFan(String tempSpeedOfFan) {
if (tempSpeedOfFan.equals("высокая") ||tempSpeedOfFan.equals("средняя") || tempSpeedOfFan.equals("низкая") || tempSpeedOfFan.equals("автоматическая") || tempSpeedOfFan.equals("другая")) {
speedOfFan = tempSpeedOfFan;
System.out.println("Скорость вентилятора успешно изменена!");
} else {
System.out.println("Скорость вентилятора не была изменена.");
}
}
// Заменить фильтры
private void changeFilterReplacementDate() {
System.out.println("Введите новую дату замены фильтра в формате дд.мм.гггг: ");
String scanned;
Scanner scanner = new Scanner(System.in);
SimpleDateFormat dateFormat = new SimpleDateFormat("dd.MM.yyyy");
while(true) {
scanned = scanner.nextLine();
if (Menu.isValidDate(scanned)) {
ParsePosition position1 = new ParsePosition(0);
ParsePosition position2 = new ParsePosition(0);
Date curDate = dateFormat.parse(filterReplacementDate, position1);
Date newDate = dateFormat.parse(scanned, position2);
if (!newDate.before(curDate)) {
filterReplacementDate = scanned;
System.out.println("Дата замены фильтра успешно изменена!");
return;
} else {
System.out.println("Попробуйте ввести ещё раз. Введённая дата меньше установленной ("+filterReplacementDate+").");
}
}
else {
System.out.println("Попробуйте ввести ещё раз.");
}
}
}
public void changeFilterReplacementDate(String tempDate) {
SimpleDateFormat dateFormat = new SimpleDateFormat("dd.MM.yyyy");
if (Menu.isValidDate(tempDate)) {
ParsePosition position1 = new ParsePosition(0);
ParsePosition position2 = new ParsePosition(0);
Date curDate = dateFormat.parse(filterReplacementDate, position1);
Date newDate = dateFormat.parse(tempDate, position2);
if (!newDate.before(curDate)) {
filterReplacementDate = tempDate;
System.out.println("Дата замены фильтра успешно изменена!");
} else {
System.out.println("Попробуйте ввести ещё раз. Введённая дата меньше установленной ("+filterReplacementDate+").");
}
}
else {
System.out.println("Попробуйте ввести ещё раз.");
}
}
private void changeInternetConnection(){
if (getOnOrOff()) {
if (getInternetConnection()) {
Random random = new Random();
int internetStatus = random.nextInt(3) + 1;
System.out.print("\tЕсть доступ к Интернету. Качество соединения: ");
switch (internetStatus) {
case 1: {
System.out.println("плохое.");
break;
}
case 2: {
System.out.println("хорошее.");
break;
}
case 3: {
System.out.println("отличное.");
break;
}
default: {
System.out.println("неопределено.");
break;
}
}
} else {
String scanned = null;
Scanner scanner = new Scanner(System.in);
System.out.println("Введите пароль для доступа в Интернет (для выхода введите пустую строку)");
while (scanned == null || !scanned.isEmpty()) {
scanned = scanner.nextLine();
if (connectToInternetMenu(scanned)) {
System.out.println("Интернет соединение успешно установлено!");
return;
} else {
System.out.println("Неправильный пароль. Попробуйте ещё раз.");
}
}
}
} else {
System.out.println("Включите устройство прежде чем проверить доступ в Интернет.");
}
}
@Override
public void saveToJSON(File file) {
Gson gson = new Gson();
String json = gson.toJson(this);
// Сохраняем JSON в файл
try (FileWriter writer = new FileWriter(file)) {
writer.write(json);
} catch (IOException e) {
e.printStackTrace();
}
}
}

Heater.java

SmartDevice.java

package org.example.devices;
import com.google.gson.Gson;
import java.io.File;
import java.io.Serializable;
import java.util.HashMap;
import java.util.Scanner;
abstract public class SmartDevice implements Serializable {
private final String type;
private final String manufacturer;
private final String model;
private final String powerSupply;
private boolean onOrOff; // True if ON
private boolean connectedToPowerSupply; //battery or powerSupplyNetwork
// Additional
protected int id; // для таблицы
static protected int maxId = 0;
protected int idRoom; // для таблицы
SmartDevice(String \_type, String \_manufacturer, String \_model, String \_powerSupply, int \_idRoom, boolean \_electricityInRoom) {
type = \_type;
manufacturer = \_manufacturer;
model = \_model;
powerSupply = \_powerSupply;
if (powerSupply.equals("сеть")) {
onOrOff = \_electricityInRoom;
} else onOrOff = true;
connectedToPowerSupply = true;
id = maxId+1;
maxId = id;
idRoom = \_idRoom;
}
public String getType(){
return type;
}
public String getManufacturer() {
return manufacturer;
}
public String getModel() {
return model;
}
public String getPowerSupply(){
return powerSupply;
}
public boolean getOnOrOff() { return onOrOff; }
public boolean getConnectedToPowerSupply(){
return connectedToPowerSupply;
}
public int getId() { return id; }
public int getIdRoom() { return idRoom; }
protected void setOnOrOff(boolean var){
onOrOff = var;
}
protected void setConnectedToPowerSupply(boolean var) {
connectedToPowerSupply = var;
}
abstract public boolean turnOn(boolean electricityInRoom);
abstract public void turnOff();
abstract public void connectToPowerSupply();
abstract public void disconnectFromPowerSupply();
abstract public void update(float \_roomTemperature, float \_roomHumidity, float \_roomAirParticleLevel, boolean \_electricityInRoom);
// Additional
abstract public HashMap<String, Object> getConfigurations();
abstract public void setConfigurations(HashMap<String, Object> configurations);
// abstract public String getContentForJSON();
// abstract public byte[] getContentForBinaryFile();
public void rebootId() {
this.id = ++maxId;
}
}

Main.java

package org.example;
public class Main {
public static void main(String[] args) {
System.out.println("Управление умными устройствами");
Menu.showMainMenu();
}
}

Menu.java

AirConditionerTest.java

package org.example.tests;
import org.example.devices.AirConditioner;
import static org.junit.jupiter.api.Assertions.\*;
public class AirConditionerTest {
@org.junit.jupiter.api.Test
public void turnOnOffTest() {
AirConditioner airC = new AirConditioner("Vector", "M1234", "сеть", "11:11:11:11:11:11", 25.0f, "Cooling", "высокая", "11.01.2022");
// initial
assertTrue(airC.getOnOrOff());
// turn ON + turn ON
airC.turnOn();
airC.turnOn();
assertTrue(airC.getOnOrOff());
// turn OFF + turn OFF
airC.turnOff();
airC.turnOff();
assertFalse(airC.getOnOrOff());
// turn ON after turn OFF
airC.turnOn();
assertTrue(airC.getOnOrOff());
}
@org.junit.jupiter.api.Test
public void turnOnOffAndPowerSupplyTest() {
AirConditioner airC = new AirConditioner("Vector", "M1234", "сеть", "11:11:11:11:11:11", 25.0f, "Cooling", "высокая", "11.01.2022");
// initial
assertTrue(airC.getConnectedToPowerSupply());
assertTrue(airC.getOnOrOff());
// powerSupply OFF
airC.disconnectFromPowerSupply();
assertFalse(airC.getConnectedToPowerSupply());
assertFalse(airC.getOnOrOff());
// try to turn ON
airC.turnOn();
assertFalse(airC.getOnOrOff());
// powerSupply ON
airC.connectToPowerSupply();
assertTrue(airC.getConnectedToPowerSupply());
assertFalse(airC.getOnOrOff());
// try to turn ON
airC.turnOn();
assertTrue(airC.getOnOrOff());
}
@org.junit.jupiter.api.Test
public void changeTemperatureTest() {
AirConditioner airC = new AirConditioner("Vector", "M1234", "сеть", "11:11:11:11:11:11", 25.0f, "Cooling", "высокая", "11.01.2022");
//init (25.0)
assertEquals(25.0f, airC.getTemperature());
// temperature 15.0
airC.changeTemperature(15.0f);
assertEquals(25.0f, airC.getTemperature());
// temperature 20
airC.changeTemperature(20.0f);
assertEquals(20.0f, airC.getTemperature());
// temperature 40
airC.changeTemperature(40.0f);
assertEquals(20.0f, airC.getTemperature());
}
@org.junit.jupiter.api.Test
public void changeModeTest() {
AirConditioner airC = new AirConditioner("Vector", "M1234", "сеть", "11:11:11:11:11:11", 25.0f, "Cooling", "высокая", "11.01.2022");
// init (Cooling)
assertEquals("Cooling", airC.getMode());
// Sun
airC.changeMode("Sun");
assertEquals("Cooling", airC.getMode());
// Auto
airC.changeMode("Auto");
assertEquals("Auto", airC.getMode());
}
@org.junit.jupiter.api.Test
public void changeSpeedOfFanTest() {
AirConditioner airC = new AirConditioner("Vector", "M1234", "сеть", "11:11:11:11:11:11", 25.0f, "Cooling", "высокая", "11.01.2022");
// init
assertEquals("высокая", airC.getSpeedOfFan());
// Cold
airC.changeSpeedOfFan("Cold");
assertEquals("высокая", airC.getSpeedOfFan());
// низкая
airC.changeSpeedOfFan("низкая");
assertEquals("низкая", airC.getSpeedOfFan());
}
@org.junit.jupiter.api.Test
public void changeFilterReplacementDateTest() {
AirConditioner airC = new AirConditioner("Vector", "M1234", "сеть", "11:11:11:11:11:11", 25.0f, "Cooling", "высокая", "11.01.2022");
// init
assertEquals("11.01.2022", airC.getFilterReplacementDate());
// 20.11.2023
airC.changeFilterReplacementDate("20.11.2024");
assertEquals("11.01.2022", airC.getFilterReplacementDate());
// 15.11.2023
airC.changeFilterReplacementDate("15.11.2023");
assertEquals("15.11.2023", airC.getFilterReplacementDate());
// 15 ноября
airC.changeFilterReplacementDate("15 ноября");
assertEquals("15.11.2023", airC.getFilterReplacementDate());
}
}

AirPurifierTest.java

package org.example.tests;
import org.example.devices.AirPurifier;
import static org.junit.jupiter.api.Assertions.\*;
import static org.junit.jupiter.api.Assertions.assertEquals;
public class AirPurifierTest {
@org.junit.jupiter.api.Test
public void turnOnOffTest() {
AirPurifier airP = new AirPurifier("Vector", "M1234", "сеть", "11:11:11:11:11:11", 40, "обогрев", "высокая", "11.01.2022");
// initial
assertTrue(airP.getOnOrOff());
// turn ON + turn ON
airP.turnOn();
airP.turnOn();
assertTrue(airP.getOnOrOff());
// turn OFF + turn OFF
airP.turnOff();
airP.turnOff();
assertFalse(airP.getOnOrOff());
// turn ON after turn OFF
airP.turnOn();
assertTrue(airP.getOnOrOff());
}
@org.junit.jupiter.api.Test
public void turnOnOffAndPowerSupplyTest() {
AirPurifier airP = new AirPurifier("Vector", "M1234", "сеть", "11:11:11:11:11:11", 40, "обогрев", "высокая", "11.01.2022");
// initial
assertTrue(airP.getConnectedToPowerSupply());
assertTrue(airP.getOnOrOff());
// powerSupply OFF
airP.disconnectFromPowerSupply();
assertFalse(airP.getConnectedToPowerSupply());
assertFalse(airP.getOnOrOff());
// try to turn ON
airP.turnOn();
assertFalse(airP.getOnOrOff());
// powerSupply ON
airP.connectToPowerSupply();
assertTrue(airP.getConnectedToPowerSupply());
assertFalse(airP.getOnOrOff());
// try to turn ON
airP.turnOn();
assertTrue(airP.getOnOrOff());
}
@org.junit.jupiter.api.Test
public void changeModeTest() {
AirPurifier airP = new AirPurifier("Vector", "M1234", "сеть", "11:11:11:11:11:11", 40, "обогрев", "высокая", "11.01.2022"); // init (Cooling)
// initial
assertEquals("обогрев", airP.getMode());
// Sun
airP.changeMode("Sun");
assertEquals("обогрев", airP.getMode());
// Auto
airP.changeMode("экономия энергии");
assertEquals("экономия энергии", airP.getMode());
}
@org.junit.jupiter.api.Test
public void changeSpeedOfFanTest() {
AirPurifier airP = new AirPurifier("Vector", "M1234", "сеть", "11:11:11:11:11:11", 40, "обогрев", "высокая", "11.01.2022"); // init
// initial
assertEquals("высокая", airP.getSpeedOfFan());
// Cold
airP.changeSpeedOfFan("Cold");
assertEquals("высокая", airP.getSpeedOfFan());
// низкая
airP.changeSpeedOfFan("низкая");
assertEquals("низкая", airP.getSpeedOfFan());
}
@org.junit.jupiter.api.Test
public void changeFilterReplacementDateTest() {
AirPurifier airP = new AirPurifier("Vector", "M1234", "сеть", "11:11:11:11:11:11", 40, "обогрев", "высокая", "11.01.2022"); // init
assertEquals("11.01.2022", airP.getFilterReplacementDate());
// 20.11.2023
airP.changeFilterReplacementDate("20.11.2024");
assertEquals("11.01.2022", airP.getFilterReplacementDate());
// 15.11.2023
airP.changeFilterReplacementDate("15.11.2023");
assertEquals("15.11.2023", airP.getFilterReplacementDate());
// 15 ноября
airP.changeFilterReplacementDate("15 ноября");
assertEquals("15.11.2023", airP.getFilterReplacementDate());
}
}

HeaterTest.java

package org.example.tests;
import org.example.devices.Heater;
import static org.junit.jupiter.api.Assertions.\*;
import static org.junit.jupiter.api.Assertions.assertEquals;
public class HeaterTest {
@org.junit.jupiter.api.Test
public void turnOnOffTest() {
Heater heater = new Heater("Vector", "M1234", "сеть", "11:11:11:11:11:11", 25.0f, 30.0f, 15);
// initial
assertTrue(heater.getOnOrOff());
// turn ON + turn ON
heater.turnOn();
heater.turnOn();
assertTrue(heater.getOnOrOff());
// turn OFF + turn OFF
heater.turnOff();
heater.turnOff();
assertFalse(heater.getOnOrOff());
// turn ON after turn OFF
heater.turnOn();
assertTrue(heater.getOnOrOff());
}
@org.junit.jupiter.api.Test
public void turnOnOffAndPowerSupplyTest() {
Heater heater = new Heater("Vector", "M1234", "сеть", "11:11:11:11:11:11", 25.0f, 30.0f, 15.0f);
// initial
assertTrue(heater.getConnectedToPowerSupply());
assertTrue(heater.getOnOrOff());
// powerSupply OFF
heater.disconnectFromPowerSupply();
assertFalse(heater.getConnectedToPowerSupply());
assertFalse(heater.getOnOrOff());
// try to turn ON
heater.turnOn();
assertFalse(heater.getOnOrOff());
// powerSupply ON
heater.connectToPowerSupply();
assertTrue(heater.getConnectedToPowerSupply());
assertFalse(heater.getOnOrOff());
// try to turn ON
heater.turnOn();
assertTrue(heater.getOnOrOff());
}
@org.junit.jupiter.api.Test
public void changeTemperatureTest() {
Heater heater = new Heater("Vector", "M1234", "сеть", "11:11:11:11:11:11", 25.0f, 30.0f, 15.0f);
//init (25.0 30.0)
assertEquals(25.0f, heater.getMinTemperature());
assertEquals(30.0f, heater.getMaxTemperature());
// min temperature 15.0
heater.changeMinTemperature(15.0f);
assertEquals(25.0f, heater.getMinTemperature());
// min temperature 20 max 35
heater.changeMinTemperature(20.0f);
heater.changeMaxTemperature(35.0f);
assertEquals(20.0f, heater.getMinTemperature());
assertEquals(35.0f, heater.getMaxTemperature());
// min temperature 40
heater.changeMinTemperature(40.0f);
assertEquals(20.0f, heater.getMinTemperature());
// max temperature 17
heater.changeMaxTemperature(17.0f);
assertEquals(35.0f, heater.getMaxTemperature());
}
@org.junit.jupiter.api.Test
public void changeAngleTest() {
Heater heater = new Heater("Vector", "M1234", "сеть", "11:11:11:11:11:11", 25.0f, 30.0f, 15.0f);
// init (Cooling)
assertEquals(15.0f, heater.getAngle());
assertTrue(heater.getOnOrOff());
// -10.0
heater.changeAngle(-10.0f);
assertEquals(15.0f, heater.getAngle());
assertTrue(heater.getOnOrOff());
// 10.0
heater.changeAngle(10.0f);
assertEquals(10.0f, heater.getAngle());
assertTrue(heater.getOnOrOff());
// 91.0
heater.changeAngle(91.0f);
assertEquals(10.0f, heater.getAngle());
assertTrue(heater.getOnOrOff());
// 50.0
heater.changeAngle(50.0f);
assertEquals(50.0f, heater.getAngle());
assertFalse(heater.getOnOrOff());
// turn ON
heater.turnOn();
assertFalse(heater.getOnOrOff());
// 16.0 turn ON
heater.changeAngle(16.0f);
assertEquals(16.0f, heater.getAngle());
assertFalse(heater.getOnOrOff());
heater.turnOn();
assertTrue(heater.getOnOrOff());
}
}

AirConditionerTest.java

import org.example.devices.AirConditioner;
import static org.junit.jupiter.api.Assertions.\*;
public class AirConditionerTest {
@org.junit.jupiter.api.Test
public void turnOnOffTest() {
AirConditioner airC = new AirConditioner("Vector", "M1234", "сеть", "11:11:11:11:11:11", 25.0f, "Cooling", "высокая", "11.01.2022");
// initial
assertTrue(airC.getOnOrOff());
// turn ON + turn ON
airC.turnOn();
airC.turnOn();
assertTrue(airC.getOnOrOff());
// turn OFF + turn OFF
airC.turnOff();
assertFalse(airC.getOnOrOff());
// turn ON after turn OFF
airC.turnOn();
assertTrue(airC.getOnOrOff());
}
@org.junit.jupiter.api.Test
public void turnOnOffAndPowerSupplyTest() {
AirConditioner airC = new AirConditioner("Vector", "M1234", "сеть", "11:11:11:11:11:11", 25.0f, "Cooling", "высокая", "11.01.2022");
// initial
assertTrue(airC.getConnectedToPowerSupply());
assertTrue(airC.getOnOrOff());
// powerSupply OFF
airC.disconnectFromPowerSupply();
assertFalse(airC.getConnectedToPowerSupply());
assertFalse(airC.getOnOrOff());
// try to turn ON
airC.turnOn();
assertFalse(airC.getOnOrOff());
// powerSupply ON
airC.connectToPowerSupply();
assertTrue(airC.getConnectedToPowerSupply());
assertFalse(airC.getOnOrOff());
// try to turn ON
airC.turnOn();
assertTrue(airC.getOnOrOff());
}
@org.junit.jupiter.api.Test
public void changeTemperatureTest() {
AirConditioner airC = new AirConditioner("Vector", "M1234", "сеть", "11:11:11:11:11:11", 25.0f, "Cooling", "высокая", "11.01.2022");
//init (25.0)
assertEquals(25.0f, airC.getTemperature());
// temperature 15.0
airC.changeTemperature(15.0f);
assertEquals(25.0f, airC.getTemperature());
// temperature 20
airC.changeTemperature(20.0f);
assertEquals(20.0f, airC.getTemperature());
// temperature 40
airC.changeTemperature(40.0f);
assertEquals(20.0f, airC.getTemperature());
}
@org.junit.jupiter.api.Test
public void changeModeTest() {
AirConditioner airC = new AirConditioner("Vector", "M1234", "сеть", "11:11:11:11:11:11", 25.0f, "Cooling", "высокая", "11.01.2022");
// init (Cooling)
assertEquals("Cooling", airC.getMode());
// Sun
airC.changeMode("Sun");
assertEquals("Cooling", airC.getMode());
// Auto
airC.changeMode("Auto");
assertEquals("Auto", airC.getMode());
}
@org.junit.jupiter.api.Test
public void changeSpeedOfFanTest() {
AirConditioner airC = new AirConditioner("Vector", "M1234", "сеть", "11:11:11:11:11:11", 25.0f, "Cooling", "высокая", "11.01.2022");
// init
assertEquals("высокая", airC.getSpeedOfFan());
// Cold
airC.changeSpeedOfFan("Cold");
assertEquals("высокая", airC.getSpeedOfFan());
// низкая
airC.changeSpeedOfFan("низкая");
assertEquals("низкая", airC.getSpeedOfFan());
}
@org.junit.jupiter.api.Test
public void changeFilterReplacementDateTest() {
AirConditioner airC = new AirConditioner("Vector", "M1234", "сеть", "11:11:11:11:11:11", 25.0f, "Cooling", "высокая", "11.01.2022");
// init
assertEquals("11.01.2022", airC.getFilterReplacementDate());
// 20.11.2023
airC.changeFilterReplacementDate("20.11.2024");
assertEquals("11.01.2022", airC.getFilterReplacementDate());
// 15.11.2023
airC.changeFilterReplacementDate("15.11.2023");
assertEquals("15.11.2023", airC.getFilterReplacementDate());
// 15 ноября
airC.changeFilterReplacementDate("15 ноября");
assertEquals("15.11.2023", airC.getFilterReplacementDate());
}
}

AirPurifierTest.java

import org.example.devices.AirPurifier;
import static org.junit.jupiter.api.Assertions.\*;
import static org.junit.jupiter.api.Assertions.assertEquals;
public class AirPurifierTest {
@org.junit.jupiter.api.Test
public void turnOnOffTest() {
AirPurifier airP = new AirPurifier("Vector", "M1234", "сеть", "11:11:11:11:11:11", 40, "обогрев", "высокая", "11.01.2022");
// initial
assertTrue(airP.getOnOrOff());
// turn ON + turn ON
airP.turnOn();
airP.turnOn();
assertTrue(airP.getOnOrOff());
// turn OFF + turn OFF
airP.turnOff();
airP.turnOff();
assertFalse(airP.getOnOrOff());
// turn ON after turn OFF
airP.turnOn();
assertTrue(airP.getOnOrOff());
}
@org.junit.jupiter.api.Test
public void turnOnOffAndPowerSupplyTest() {
AirPurifier airP = new AirPurifier("Vector", "M1234", "сеть", "11:11:11:11:11:11", 40, "обогрев", "высокая", "11.01.2022");
// initial
assertTrue(airP.getConnectedToPowerSupply());
assertTrue(airP.getOnOrOff());
// powerSupply OFF
airP.disconnectFromPowerSupply();
assertFalse(airP.getConnectedToPowerSupply());
assertFalse(airP.getOnOrOff());
// try to turn ON
airP.turnOn();
assertFalse(airP.getOnOrOff());
// powerSupply ON
airP.connectToPowerSupply();
assertTrue(airP.getConnectedToPowerSupply());
assertFalse(airP.getOnOrOff());
// try to turn ON
airP.turnOn();
assertTrue(airP.getOnOrOff());
}
@org.junit.jupiter.api.Test
public void changeModeTest() {
AirPurifier airP = new AirPurifier("Vector", "M1234", "сеть", "11:11:11:11:11:11", 40, "обогрев", "высокая", "11.01.2022"); // init (Cooling)
// initial
assertEquals("обогрев", airP.getMode());
// Sun
airP.changeMode("Sun");
assertEquals("обогрев", airP.getMode());
// Auto
airP.changeMode("экономия энергии");
assertEquals("экономия энергии", airP.getMode());
}
@org.junit.jupiter.api.Test
public void changeSpeedOfFanTest() {
AirPurifier airP = new AirPurifier("Vector", "M1234", "сеть", "11:11:11:11:11:11", 40, "обогрев", "высокая", "11.01.2022"); // init
// initial
assertEquals("высокая", airP.getSpeedOfFan());
// Cold
airP.changeSpeedOfFan("Cold");
assertEquals("высокая", airP.getSpeedOfFan());
// низкая
airP.changeSpeedOfFan("низкая");
assertEquals("низкая", airP.getSpeedOfFan());
}
@org.junit.jupiter.api.Test
public void changeFilterReplacementDateTest() {
AirPurifier airP = new AirPurifier("Vector", "M1234", "сеть", "11:11:11:11:11:11", 40, "обогрев", "высокая", "11.01.2022"); // init
assertEquals("11.01.2022", airP.getFilterReplacementDate());
// 20.11.2023
airP.changeFilterReplacementDate("20.11.2024");
assertEquals("11.01.2022", airP.getFilterReplacementDate());
// 15.11.2023
airP.changeFilterReplacementDate("15.11.2023");
assertEquals("15.11.2023", airP.getFilterReplacementDate());
// 15 ноября
airP.changeFilterReplacementDate("15 ноября");
assertEquals("15.11.2023", airP.getFilterReplacementDate());
}
}

HeaterTest.java

import org.example.devices.Heater;
import static org.junit.jupiter.api.Assertions.\*;
import static org.junit.jupiter.api.Assertions.assertEquals;
public class HeaterTest {
@org.junit.jupiter.api.Test
public void turnOnOffTest() {
Heater heater = new Heater("Vector", "M1234", "сеть", "11:11:11:11:11:11", 25.0f, 30.0f, 15);
// initial
assertTrue(heater.getOnOrOff());
// turn ON + turn ON
heater.turnOn();
heater.turnOn();
assertTrue(heater.getOnOrOff());
// turn OFF + turn OFF
heater.turnOff();
heater.turnOff();
assertFalse(heater.getOnOrOff());
// turn ON after turn OFF
heater.turnOn();
assertTrue(heater.getOnOrOff());
}
@org.junit.jupiter.api.Test
public void turnOnOffAndPowerSupplyTest() {
Heater heater = new Heater("Vector", "M1234", "сеть", "11:11:11:11:11:11", 25.0f, 30.0f, 15.0f);
// initial
assertTrue(heater.getConnectedToPowerSupply());
assertTrue(heater.getOnOrOff());
// powerSupply OFF
heater.disconnectFromPowerSupply();
assertFalse(heater.getConnectedToPowerSupply());
assertFalse(heater.getOnOrOff());
// try to turn ON
heater.turnOn();
assertFalse(heater.getOnOrOff());
// powerSupply ON
heater.connectToPowerSupply();
assertTrue(heater.getConnectedToPowerSupply());
assertFalse(heater.getOnOrOff());
// try to turn ON
heater.turnOn();
assertTrue(heater.getOnOrOff());
}
@org.junit.jupiter.api.Test
public void changeTemperatureTest() {
Heater heater = new Heater("Vector", "M1234", "сеть", "11:11:11:11:11:11", 25.0f, 30.0f, 15.0f);
//init (25.0 30.0)
assertEquals(25.0f, heater.getMinTemperature());
assertEquals(30.0f, heater.getMaxTemperature());
// min temperature 15.0
heater.changeMinTemperature(15.0f);
assertEquals(25.0f, heater.getMinTemperature());
// min temperature 20 max 35
heater.changeMinTemperature(20.0f);
heater.changeMaxTemperature(35.0f);
assertEquals(20.0f, heater.getMinTemperature());
assertEquals(35.0f, heater.getMaxTemperature());
// min temperature 40
heater.changeMinTemperature(40.0f);
assertEquals(20.0f, heater.getMinTemperature());
// max temperature 17
heater.changeMaxTemperature(17.0f);
assertEquals(35.0f, heater.getMaxTemperature());
}
@org.junit.jupiter.api.Test
public void changeAngleTest() {
Heater heater = new Heater("Vector", "M1234", "сеть", "11:11:11:11:11:11", 25.0f, 30.0f, 15.0f);
// init (Cooling)
assertEquals(15.0f, heater.getAngle());
assertTrue(heater.getOnOrOff());
// -10.0
heater.changeAngle(-10.0f);
assertEquals(15.0f, heater.getAngle());
assertTrue(heater.getOnOrOff());
// 10.0
heater.changeAngle(10.0f);
assertEquals(10.0f, heater.getAngle());
assertTrue(heater.getOnOrOff());
// 91.0
heater.changeAngle(91.0f);
assertEquals(10.0f, heater.getAngle());
assertTrue(heater.getOnOrOff());
// 50.0
heater.changeAngle(50.0f);
assertEquals(50.0f, heater.getAngle());
assertFalse(heater.getOnOrOff());
// turn ON
heater.turnOn();
assertFalse(heater.getOnOrOff());
// 16.0 turn ON
heater.changeAngle(16.0f);
assertEquals(16.0f, heater.getAngle());
assertFalse(heater.getOnOrOff());
heater.turnOn();
assertTrue(heater.getOnOrOff());
}
}

module-info.java

module org.example.demo {
requires javafx.controls;
requires javafx.fxml;
requires org.apache.logging.log4j;
requires java.sql;
requires com.google.gson;
opens org.example.demo to javafx.fxml;
opens org.example.demo.devices to com.google.gson;
exports org.example.demo;
exports org.example.demo.sceneControllers;
opens org.example.demo.sceneControllers to javafx.fxml;
exports org.example.demo.fileManagers;
opens org.example.demo.fileManagers to javafx.fxml;
}

Application.java

package org.example.demo;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;
import org.example.demo.devices.AirConditioner;
import org.example.demo.devices.Heater;
import org.example.demo.devices.SmartDevice;
import org.example.demo.fileManagers.BinaryFileManager;
import org.example.demo.fileManagers.JsonFileManager;
import org.example.demo.sceneControllers.ControllerListOfDevices;
import java.io.File;
import java.io.InputStream;
import java.io.IOException;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Properties;
public class Application extends javafx.application.Application {
private static final Logger logger = LogManager.getLogger("FileFilesLogger");
private static final Logger loggerUser = LogManager.getLogger("FileUserLogger");
private static void configureDB() {
Properties properties = new Properties();
HashMap<String, String> configInfo = new HashMap<>();
try (InputStream input = Application.class.getClassLoader().getResourceAsStream("application.properties")) {
if (input == null) {
logger.fatal("Ошибка чтения конфигурационного файла!");
return;
}
properties.load(input);
configInfo.put("dbName", properties.getProperty("application.dbName"));
configInfo.put("dbUser", properties.getProperty("application.dbUser"));
configInfo.put("dbPass", properties.getProperty("application.dbPass"));
} catch (IOException e) {
e.printStackTrace();
}
DatabaseManager.setValuesForConnection(configInfo.get("dbName"), configInfo.get("dbUser"), configInfo.get("dbPass"));
}
@Override
public void start(Stage stage) throws Exception {
ControllerListOfDevices controller = new ControllerListOfDevices();
controller.start(stage);
}
public static void main(String[] args) {
loggerUser.info("Программа начала работу");
configureDB();
launch();
loggerUser.info("Программа успешно закончила работу");
}
}

DatabaseManager.java

package org.example.demo;
import org.example.demo.devices.AirConditioner;
import org.example.demo.devices.AirPurifier;
import org.example.demo.devices.Heater;
import org.example.demo.devices.SmartDevice;
import org.example.demo.fileManagers.BinaryFileManager;
import java.io.ByteArrayInputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.sql.\*;
import java.util.ArrayList;
public class DatabaseManager {
static String url = "jdbc:mysql://localhost:3306/";
static String dbName;
static String username;
static String password;
public static void setValuesForConnection(String \_dbName, String \_username, String \_password) {
url += \_dbName;
dbName = \_dbName;
username = \_username;
password = \_password;
}
public static ArrayList<SmartDevice> getDevices() {
ArrayList<SmartDevice> devices = new ArrayList<>();
String query = "SELECT \* FROM devices";
try {
Connection connection = DriverManager.getConnection(url, username, password);
Statement statement = connection.createStatement();
ResultSet resultSet = statement.executeQuery(query);
while (resultSet.next()) {
int idDevice = resultSet.getInt("id");
Blob dataDevice = resultSet.getBlob("data");
byte[] data = dataDevice.getBytes(1, (int) dataDevice.length());
ByteArrayInputStream bais = new ByteArrayInputStream(data);
ObjectInputStream ois = new ObjectInputStream(bais);
SmartDevice obj = (SmartDevice) ois.readObject();
ois.close();
bais.close();
if (obj instanceof AirConditioner) {
SmartDevice resObj = (AirConditioner) obj;
resObj.setId(idDevice);
devices.add(resObj);
} else if (obj instanceof AirPurifier) {
SmartDevice resObj = (AirPurifier) obj;
resObj.setId(idDevice);
devices.add(resObj);
} else if (obj instanceof Heater) {
SmartDevice resObj = (Heater) obj;
resObj.setId(idDevice);
devices.add(resObj);
} else {
// loger
}
}
connection.close();
return devices;
} catch (SQLException | IOException | ClassNotFoundException e) {
e.printStackTrace();
}
return null;
}
public static SmartDevice getDevice(int id) {
String query = "SELECT \* FROM devices WHERE id = ?";
try {
Connection connection = DriverManager.getConnection(url, username, password);
PreparedStatement statement = connection.prepareStatement(query);
statement.setInt(1, id);
ResultSet resultSet = statement.executeQuery();
while (resultSet.next()) {
int idDevice = resultSet.getInt("id");
Blob dataDevice = resultSet.getBlob("data");
byte[] data = dataDevice.getBytes(1, (int) dataDevice.length());
ByteArrayInputStream bais = new ByteArrayInputStream(data);
ObjectInputStream ois = new ObjectInputStream(bais);
SmartDevice obj = (SmartDevice) ois.readObject();
ois.close();
bais.close();
if (obj instanceof AirConditioner) {
SmartDevice resObj = (AirConditioner) obj;
resObj.setId(idDevice);
return resObj;
} else if (obj instanceof AirPurifier) {
SmartDevice resObj = (AirPurifier) obj;
resObj.setId(idDevice);
return resObj;
} else if (obj instanceof Heater) {
SmartDevice resObj = (Heater) obj;
resObj.setId(idDevice);
return resObj;
} else {
// loger
}
}
connection.close();
} catch (SQLException | IOException | ClassNotFoundException e) {
e.printStackTrace();
}
return null;
}
public static void insertDevice(SmartDevice device) {
String query = "INSERT INTO devices (data) VALUES (?)";
try (Connection connection = DriverManager.getConnection(url, username, password);
PreparedStatement preparedStatement = connection.prepareStatement(query, Statement.RETURN\_GENERATED\_KEYS)) {
// вставка
byte[] serializedObject = BinaryFileManager.getByteData(device);
ByteArrayInputStream data = new ByteArrayInputStream(serializedObject);
preparedStatement.setBinaryStream(1, data, serializedObject.length);
int affectedRows = preparedStatement.executeUpdate();
// получение id
if (affectedRows > 0) {
ResultSet generatedKeys = preparedStatement.getGeneratedKeys();
if (generatedKeys.next()) {
int generatedId = generatedKeys.getInt(1); // Получение сгенерированного ключа
device.setId(generatedId);
}
} else {
}
connection.close();
} catch (SQLException e) {
e.printStackTrace();
}
}
public static boolean updateDevice(SmartDevice device) {
String updateQuery = "UPDATE devices SET data = ? WHERE id = ?";
try {
Connection connection = DriverManager.getConnection(url, username, password);
PreparedStatement preparedStatement = connection.prepareStatement(updateQuery);
byte[] serializedObject = BinaryFileManager.getByteData(device);
ByteArrayInputStream data = new ByteArrayInputStream(serializedObject);
preparedStatement.setInt(2, device.getId());
preparedStatement.setBinaryStream(1, data, serializedObject.length);
int rowsAffected = preparedStatement.executeUpdate();
if (rowsAffected > 0) {
// System.out.println("Запись успешно обновлена");
connection.close();
return true;
} else {
// System.out.println("Запись не была обновлена");
connection.close();
return false;
}
} catch (SQLException e) {
e.printStackTrace();
return false;
}
}
public static boolean deleteDevice(int id) {
String deleteQuery = "DELETE FROM devices WHERE id = ?";
try (Connection connection = DriverManager.getConnection(url, username, password);
PreparedStatement preparedStatement = connection.prepareStatement(deleteQuery)) {
preparedStatement.setInt(1, id);
int rowsAffected = preparedStatement.executeUpdate();
if (rowsAffected > 0) {
// System.out.println("Запись успешно обновлена");
connection.close();
return true;
} else {
// System.out.println("Запись не была обновлена");
connection.close();
return false;
}
} catch (SQLException e) {
e.printStackTrace();
return false;
}
}
}

AirConditioner.java

AirPurifier.java

Heater.java

SmartDevice.java

package org.example.demo.devices;
import java.io.Serializable;
import java.util.HashMap;
abstract public class SmartDevice implements Serializable {
private final String type;
private final String manufacturer;
//public void setMan(String \_manufacturer) {
// manufacturer = \_manufacturer;
// }
private final String model;
private final String powerSupply;
private boolean onOrOff; // True if ON
private boolean connectedToPowerSupply; //battery or powerSupplyNetwork
// Additional
transient protected int id; // для таблицы
static protected int maxId = 0;
SmartDevice(String \_type, String \_manufacturer, String \_model, String \_powerSupply, boolean \_electricityInRoom) {
type = \_type;
manufacturer = \_manufacturer;
model = \_model;
powerSupply = \_powerSupply;
if (powerSupply.equals("сеть")) {
onOrOff = \_electricityInRoom;
} else onOrOff = true;
connectedToPowerSupply = true;
}
public String getType(){
return type;
}
public String getManufacturer() {
return manufacturer;
}
public String getModel() {
return model;
}
public String getPowerSupply(){
return powerSupply;
}
public boolean getOnOrOff() { return onOrOff; }
public boolean getConnectedToPowerSupply(){
return connectedToPowerSupply;
}
public int getId() { return id; }
public void setId(int id) {
this.id = id;
}
protected void setOnOrOff(boolean var){
onOrOff = var;
}
protected void setConnectedToPowerSupply(boolean var) {
connectedToPowerSupply = var;
}
abstract public boolean turnOn(boolean electricityInRoom);
abstract public void turnOff();
abstract public void connectToPowerSupply();
abstract public void disconnectFromPowerSupply();
abstract public void update(float \_roomTemperature, float \_roomHumidity, float \_roomAirParticleLevel, boolean \_electricityInRoom);
// Additional
abstract public HashMap<String, Object> getConfigurations();
abstract public void setConfigurations(HashMap<String, Object> configurations);
// abstract public String getContentForJSON();
// abstract public byte[] getContentForBinaryFile();
public void rebootId() {
this.id = ++maxId;
}
}

BinaryFileManager.java

package org.example.demo.fileManagers;
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;
import org.example.demo.devices.AirConditioner;
import org.example.demo.devices.AirPurifier;
import org.example.demo.devices.Heater;
import org.example.demo.devices.SmartDevice;
import java.io.\*;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
public class BinaryFileManager {
private static final Logger logger = LogManager.getLogger("FileFilesLogger");
public static boolean isValidFileName(String fileName) {
String fileNameRegex = "^[^\\\\/:\*?\"<>|.]\*$";
Pattern pattern = Pattern.compile(fileNameRegex);
Matcher matcher = pattern.matcher(fileName);
return matcher.matches();
}
public static boolean writeIntoBAT(SmartDevice device, String fileName) {
try (FileOutputStream fileOutputStream = new FileOutputStream(fileName); ObjectOutputStream objectOutputStream = new ObjectOutputStream(fileOutputStream)) {
objectOutputStream.writeObject(device);
logger.info("Успешная запись PK=" + device.getId() + " в бинарный файл: " + fileName);
return true;
} catch (Exception e) {
e.printStackTrace();
}
return false;
}
public static SmartDevice readFromBAT(File file) {
try (ObjectInputStream objectInputStream = new ObjectInputStream(new FileInputStream(file))) {
SmartDevice device = (SmartDevice) objectInputStream.readObject();
if (device instanceof AirPurifier || device instanceof AirConditioner || device instanceof Heater) {
logger.info("Чтение из бинарного файла: информация об устройстве типа " + device.getType() + " успешно получена");
return device;
} else {
logger.error("Чтение из бинарного файла: информация получена, но объект не идентифицирован");
}
} catch (ClassNotFoundException | IOException e) {
e.printStackTrace();
logger.error("Чтение из бинарного файла: произошла ошибка чтения файла");
}
return null;
}
public static byte[] getByteData(SmartDevice device) {
try (ByteArrayOutputStream baos = new ByteArrayOutputStream()) {
ObjectOutputStream oos = new ObjectOutputStream(baos);
oos.writeObject(device);
byte[] serializedObject = baos.toByteArray();
return serializedObject;
} catch (IOException e) {
e.printStackTrace();
}
return null;
}
}

FileManager.java

package org.example.demo.fileManagers;
import java.util.regex.Pattern;
public class FileManager {
public static int getTypeOfFile(String fileName) {
if (fileName.matches("[a-zA-Z0-9\\-\_]+\\.bat")) {
return 1;
} else if (fileName.matches("[a-zA-Z0-9\\-\_]+\\.json")) {
return 2;
} else return 0;
}
}

JsonFileManager.java

package org.example.demo.fileManagers;
import com.google.gson.GsonBuilder;
import com.google.gson.JsonParseException;
import com.google.gson.JsonParser;
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;
import org.example.demo.devices.AirConditioner;
import org.example.demo.devices.AirPurifier;
import org.example.demo.devices.Heater;
import org.example.demo.devices.SmartDevice;
import com.google.gson.Gson;
import java.io.\*;
import java.time.LocalDate;
public class JsonFileManager {
private static final Logger logger = LogManager.getLogger("FileFilesLogger");
private static boolean isValidFileNameJSON(String fileName){
try {
JsonParser parser = new JsonParser();
parser.parse(fileName+".json");
return true;
} catch (JsonParseException e) {
return false;
}
}
public static boolean writeIntoJSON(SmartDevice device, String fileName) {
Gson gson = new GsonBuilder()
.registerTypeAdapter(LocalDate.class, new LocalDateAdapter())
.create();
try (FileWriter writer = new FileWriter(fileName)) {
gson.toJson(device, writer);
logger.info("Успешная запись PK="+ device.getId() +" в JSON файл: " + fileName);
return true;
} catch (IOException e) {
e.printStackTrace();
}
return false;
}
public static SmartDevice readFromJSON(File file) { // известно, что file существует
if (file.exists()) {
StringBuilder contentJSON = new StringBuilder();
try {
String line;
FileReader reader = new FileReader(file);
BufferedReader bufReader = new BufferedReader(reader);
while ((line = bufReader.readLine()) != null) {
contentJSON.append(line);
}
String jsonString = contentJSON.toString();
Gson gson = new GsonBuilder()
.registerTypeAdapter(LocalDate.class, new LocalDateAdapter())
.create();
SmartDevice jsonObject = null;
try {
jsonObject = gson.fromJson(jsonString, AirConditioner.class);
if (jsonObject.getType().equals("AirPurifier")) {
jsonObject = gson.fromJson(jsonString, AirPurifier.class);
} else if (jsonObject.getType().equals("Heater")) {
jsonObject = gson.fromJson(jsonString, Heater.class);
} else if (jsonObject.getType().equals("AirConditioner")) {
jsonObject = gson.fromJson(jsonString, AirConditioner.class);
} else {
logger.error("Чтение из JSON: содержимое JSON файла не соответствует ни одному устройству");
}
} catch (RuntimeException ex1) {
ex1.printStackTrace();
logger.error("Чтение из JSON: в ходе работы с JSON произошла ошибка");
}
if (jsonObject!=null){
if (jsonObject.getType().equals("AirPurifier") || jsonObject.getType().equals("AirConditioner") || jsonObject.getType().equals("Heater"))
logger.info("Чтение из JSON: информация об устройстве типа " + jsonObject.getType() + " успешно получена");
else logger.error("Чтение из JSON: информация получена, но объект не идентифицирован");
return jsonObject;
}
} catch (IOException e) {
e.printStackTrace();
logger.error("Чтение из JSON: в ходе работы произошла ошибка чтения файла");
}
} else {
logger.error("Чтение из JSON: файл с таким именем не существует");
}
return null;
}
}

LocalDateAdapter.java

package org.example.demo.fileManagers;
import com.google.gson.\*;
import java.lang.reflect.Type;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
public class LocalDateAdapter implements JsonSerializer<LocalDate>, JsonDeserializer<LocalDate> {
private static final DateTimeFormatter DATE\_FORMATTER = DateTimeFormatter.ISO\_LOCAL\_DATE;
@Override
public JsonElement serialize(LocalDate date, Type typeOfSrc, JsonSerializationContext context) {
return new JsonPrimitive(date.format(DATE\_FORMATTER));
}
@Override
public LocalDate deserialize(JsonElement json, Type typeOfT, JsonDeserializationContext context) throws JsonParseException {
return LocalDate.parse(json.getAsString(), DATE\_FORMATTER);
}
}

ControllerAddAirConditioner.java

ControllerAddAirPurifier.java

package org.example.demo.sceneControllers;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.Scene;
import javafx.scene.control.\*;
import javafx.stage.Stage;
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;
import org.example.demo.DatabaseManager;
import org.example.demo.SmartDevicesManager;
import org.example.demo.devices.AirConditioner;
import org.example.demo.devices.AirPurifier;
import org.example.demo.devices.SmartDevice;
import org.example.demo.fileManagers.BinaryFileManager;
import org.example.demo.fileManagers.FileManager;
import org.example.demo.fileManagers.JsonFileManager;
import java.io.File;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.util.HashMap;
public class ControllerAddAirPurifier {
public TextField fileName;
public Button importButton;
public Label fileMessage;
@FXML
private Label errorMessage;
@FXML
private ComboBox<String> speedOfFan;
private ToggleGroup modeHumidityGroup;
@FXML
public RadioButton modeHumidity0;
@FXML
public RadioButton modeHumidity1;
@FXML
public RadioButton modeHumidity2;
@FXML
public CheckBox modeAuto;
@FXML
public DatePicker filterReplacementDate;
@FXML
public TextField manufacture;
@FXML
public TextField model;
@FXML
public ComboBox<String> powerSupply;
@FXML
public CheckBox modeAirParticle;
@FXML
public Button cancel;
@FXML
public Button save;
private static final Logger loggerUser = LogManager.getLogger("FileUserLogger");
public void start(Stage stage) {
loggerUser.info("Добавление очистителя воздуха");
manufacture.setText("manufacture");
model.setText("model");
powerSupply.getItems().addAll("сеть", "аккумулятор", "батарейки");
powerSupply.setValue("сеть");
speedOfFan.getItems().addAll("высокая", "средняя", "низкая", "автоматическая");
speedOfFan.setValue("автоматическая");
String dateString = "2000-01-01";
DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd");
filterReplacementDate.setValue(LocalDate.parse(dateString, formatter));
modeHumidityGroup = new ToggleGroup();
modeHumidity0.setToggleGroup(modeHumidityGroup);
modeHumidity1.setToggleGroup(modeHumidityGroup);
modeHumidity2.setToggleGroup(modeHumidityGroup);
modeHumidity2.setSelected(true);
modeAuto.setSelected(true);
}
public void cancelClicked() {
loggerUser.info("Отмена добавления очистителя воздуха");
ControllerListOfDevices controller = new ControllerListOfDevices();
Scene currentScene = cancel.getScene();
Stage stage = (Stage) currentScene.getWindow();
controller.start(stage);
}
public void saveClicked() {
String manufacturerValue = manufacture.getText();
String modelValue = model.getText();
String powerSupplyValue = powerSupply.getValue();
String fanSpeed = speedOfFan.getValue();
LocalDate replacementDate = filterReplacementDate.getValue();
String modeHumidity;
boolean isHumidification = modeHumidity0.isSelected();
boolean isDehumidification = modeHumidity1.isSelected();
if (isHumidification) modeHumidity = "увлажнение";
else if (isDehumidification) modeHumidity = "осушение";
else modeHumidity = "отключен";
boolean isAirParticleMode = modeAirParticle.isSelected();
boolean isAutomaticMode = modeAuto.isSelected();
AirPurifier airP = new AirPurifier( manufacturerValue, modelValue, powerSupplyValue, SmartDevicesManager.getElectricityInRoom(), modeHumidity, isAirParticleMode, isAutomaticMode, fanSpeed, replacementDate);
if (airP.isFilterReplacementDateValid(replacementDate)) {
SmartDevice device = airP;
device.update(SmartDevicesManager.getRoomTemperature(), SmartDevicesManager.getRoomHumidity(), SmartDevicesManager.getRoomAirParticleLevel(), SmartDevicesManager.getElectricityInRoom());
DatabaseManager.insertDevice(device);
ControllerListOfDevices controller = new ControllerListOfDevices();
Scene currentScene = cancel.getScene();
Stage stage = (Stage) currentScene.getWindow();
loggerUser.info("Успешное сохранение очистителя воздуха");
controller.start(stage);
} else {
loggerUser.error("Введены неправильные данные");
errorMessage.setText("Дата замены фильтров не должна превышать сегодняшнюю дату!");
}
}
public void importButtonClicked(ActionEvent actionEvent) {
loggerUser.info("Попытка импортирования очистителя воздуха");
String fileNameValue = fileName.getText();
switch (FileManager.getTypeOfFile(fileNameValue)) {
case(1): {
fileMessage.setText("");
File file = new File(fileNameValue);
if (file.exists()) {
SmartDevice device = BinaryFileManager.readFromBAT(file);
if (device == null) {
fileMessage.setText("Произошла ошибка чтения файла!");
} else {
if (device.getType().equals("AirPurifier")) {
setFields((AirPurifier) device);
loggerUser.info("Успешное импортирование очистителя воздуха");
} else {
fileMessage.setText("В файле содержится не очиститель воздуха!");
}
}
} else {
fileMessage.setText("Такой файл не существует!");
}
break;
}
case(2):{
fileMessage.setText("");
File file = new File(fileNameValue);
if (file.exists()) {
SmartDevice device = JsonFileManager.readFromJSON(file);
if (device == null) {
fileMessage.setText("Произошла ошибка чтения файла!");
} else {
if (device.getType().equals("AirPurifier")) {
setFields((AirPurifier) device);
loggerUser.info("Успешное импортирование очистителя воздуха");
} else {
fileMessage.setText("В файле содержится не очиститель воздуха!");
}
}
} else {
fileMessage.setText("Такой файл не существует!");
}
break;
}
default: {
fileMessage.setText("Имя файла не соответствует .bat или .json");
}
}
}
public void setFields (AirPurifier device) {
manufacture.setText(device.getManufacturer());
model.setText(device.getModel());
powerSupply.setValue(device.getPowerSupply());
HashMap<String, Object> configurations = device.getConfigurations();
speedOfFan.getItems().addAll("высокая", "средняя", "низкая", "автоматическая");
speedOfFan.setValue((String) configurations.get("Скорость вентиляторов"));
filterReplacementDate.setValue((LocalDate) configurations.get("Дата замены фильтров"));
modeHumidityGroup = new ToggleGroup();
modeHumidity0.setToggleGroup(modeHumidityGroup);
modeHumidity1.setToggleGroup(modeHumidityGroup);
modeHumidity2.setToggleGroup(modeHumidityGroup);
switch ((String) configurations.get("Режим регуляции влажности")) {
case ("увлажнение"): {
modeHumidity0.setSelected(true);
break;
}
case ("осушение"):{
modeHumidity1.setSelected(true);
break;
}
case ("отключен"): {
modeHumidity2.setSelected(true);
}
}
modeAuto.setSelected((boolean)configurations.get("Автоматический режим"));
}
}

ControllerAddHeater.java

ControllerConfigureAirConditioner.java

package org.example.demo.sceneControllers;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.Scene;
import javafx.scene.control.\*;
import javafx.stage.Stage;
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;
import org.example.demo.DatabaseManager;
import org.example.demo.SmartDevicesManager;
import org.example.demo.devices.AirConditioner;
import org.example.demo.devices.SmartDevice;
import org.example.demo.fileManagers.BinaryFileManager;
import org.example.demo.fileManagers.FileManager;
import org.example.demo.fileManagers.JsonFileManager;
import java.io.File;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.util.HashMap;
public class ControllerConfigureAirConditioner {
@FXML
public Label manufacture;
@FXML
public Label model;
@FXML
public Label powerSupply;
public CheckBox connectedToPowerSupply;
public CheckBox working;
public TextField fileName;
public Button exportButton;
public Label fileMessage;
@FXML
private Spinner<Double> temperature;
@FXML
private ComboBox<String> speedOfFan;
private ToggleGroup modeTemperatureGroup;
@FXML
private RadioButton modeTemperature0;
@FXML
private RadioButton modeTemperature1;
@FXML
private RadioButton modeTemperature2;
private ToggleGroup modeHumidityGroup;
@FXML
private RadioButton modeHumidity0;
@FXML
private RadioButton modeHumidity1;
@FXML
private RadioButton modeHumidity2;
@FXML
private CheckBox modeAuto;
@FXML
private DatePicker filterReplacementDate;
@FXML
private Button cancel;
@FXML
private Button save;
@FXML
private Label errorMessage;
private SmartDevice device;
private static final Logger loggerUser = LogManager.getLogger("FileUserLogger");
public void start(Stage stage, SmartDevice \_device) {
loggerUser.info("Настройка кондиционера PK="+\_device.getId());
device = \_device;
manufacture.setText(device.getManufacturer());
model.setText(device.getModel());
powerSupply.setText(device.getPowerSupply());
HashMap<String, Object> configurations = device.getConfigurations();
SpinnerValueFactory<Double> valueFactory =
new SpinnerValueFactory.DoubleSpinnerValueFactory(16.0, 27.0, Double.valueOf((float)configurations.get("Температура")), 0.1);
temperature.setValueFactory(valueFactory);
speedOfFan.getItems().addAll("высокая", "средняя", "низкая", "автоматическая");
speedOfFan.setValue((String) configurations.get("Скорость вентиляторов"));
filterReplacementDate.setValue((LocalDate) configurations.get("Дата замены фильтров"));
modeTemperatureGroup = new ToggleGroup();
modeTemperature0.setToggleGroup(modeTemperatureGroup);
modeTemperature1.setToggleGroup(modeTemperatureGroup);
modeTemperature2.setToggleGroup(modeTemperatureGroup);
if (((String) configurations.get("Режим регуляции температуры")).equals("охлаждение")){
modeTemperature0.setSelected(true);
} else if (((String) configurations.get("Режим регуляции температуры")).equals("обогрев")) {
modeTemperature1.setSelected(true);
} else {
modeTemperature2.setSelected(true);
}
modeHumidityGroup = new ToggleGroup();
modeHumidity0.setToggleGroup(modeHumidityGroup);
modeHumidity1.setToggleGroup(modeHumidityGroup);
modeHumidity2.setToggleGroup(modeHumidityGroup);
switch ((String) configurations.get("Режим регуляции влажности")) {
case ("увлажнение"): {
modeHumidity0.setSelected(true);
break;
}
case ("осушение"):{
modeHumidity1.setSelected(true);
break;
}
case ("отключен"): {
modeHumidity2.setSelected(true);
}
}
modeAuto.setSelected((boolean)configurations.get("Автоматический режим"));
connectedToPowerSupply.setSelected((device.getConnectedToPowerSupply()));
working.setSelected((device.getOnOrOff()));
}
@FXML
private void cancelClicked() {
loggerUser.info("Отмена настроек кондиционера");
ControllerListOfDevices controller = new ControllerListOfDevices();
Scene currentScene = cancel.getScene();
Stage stage = (Stage) currentScene.getWindow();
controller.start(stage);
}
@FXML
private void saveClicked() {
String manufacturerValue = manufacture.getText();
String modelValue = model.getText();
String powerSupplyValue = powerSupply.getText();
double tempValue = temperature.getValue();
String fanSpeed = speedOfFan.getValue();
LocalDate replacementDate = filterReplacementDate.getValue();
String modeTemperature;
boolean cooling = modeTemperature0.isSelected();
boolean heating = modeTemperature1.isSelected();
if (cooling) modeTemperature = "охлаждение";
else if (heating) modeTemperature = "обогрев";
else modeTemperature = "отключен";
String modeHumidity;
boolean isHumidification = modeHumidity0.isSelected();
boolean isDehumidification = modeHumidity1.isSelected();
if (isHumidification) modeHumidity = "увлажнение";
else if (isDehumidification) modeHumidity = "осушение";
else modeHumidity = "отключен";
boolean isAutomaticMode = modeAuto.isSelected();
AirConditioner airC = new AirConditioner( manufacturerValue, modelValue, powerSupplyValue, SmartDevicesManager.getElectricityInRoom(), (float) tempValue , modeTemperature, modeHumidity, isAutomaticMode, fanSpeed, replacementDate);
if (working.isSelected()) {
airC.turnOn(SmartDevicesManager.getElectricityInRoom());
} else {
airC.turnOff();
}
if (connectedToPowerSupply.isSelected()) {
airC.connectToPowerSupply();
} else {
airC.disconnectFromPowerSupply();
}
if (airC.isFilterReplacementDateValid(replacementDate)) {
SmartDevice deviceTemp = airC;
deviceTemp.update(SmartDevicesManager.getRoomTemperature(), SmartDevicesManager.getRoomHumidity(), SmartDevicesManager.getRoomAirParticleLevel(), SmartDevicesManager.getElectricityInRoom());
deviceTemp.setId(device.getId());
DatabaseManager.updateDevice(deviceTemp);
ControllerListOfDevices controller = new ControllerListOfDevices();
Scene currentScene = cancel.getScene();
Stage stage = (Stage) currentScene.getWindow();
loggerUser.info("Успешное сохранение настроек кондиционера PK="+device.getId());
controller.start(stage);
} else {
loggerUser.warn("Введены неверные данные");
errorMessage.setText("Дата замены фильтров не должна превышать сегодняшнюю дату!");
}
}
public void exportButtonClicked(ActionEvent actionEvent) {
String fileNameValue = fileName.getText();
switch (FileManager.getTypeOfFile(fileNameValue)) {
case(1): { // .bat
fileMessage.setText("");
File file = new File(fileNameValue);
if (BinaryFileManager.writeIntoBAT(device, fileNameValue)) {
fileMessage.setText("Файл успешно записан!");
loggerUser.info("Успешное экспортирование настроек кондиционера PK="+device.getId());
} else {
loggerUser.error("Ошибка при экспортировании");
fileMessage.setText("Произошла ошибка!");
}
break;
}
case(2):{ // .json
fileMessage.setText("");
File file = new File(fileNameValue);
if (JsonFileManager.writeIntoJSON(device, fileNameValue)) {
fileMessage.setText("Файл успешно записан!");
loggerUser.info("Успешное экспортирование настроек кондиционера PK="+device.getId());
} else {
fileMessage.setText("Произошла ошибка!");
loggerUser.error("Ошибка при экспортировании");
}
break;
}
default: {
fileMessage.setText("Имя файла не \*.bat или \*.json");
}
}
}
}

ControllerConfigureAirPurifier.java

package org.example.demo.sceneControllers;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.Scene;
import javafx.scene.control.\*;
import javafx.stage.Stage;
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;
import org.example.demo.DatabaseManager;
import org.example.demo.SmartDevicesManager;
import org.example.demo.devices.AirPurifier;
import org.example.demo.devices.SmartDevice;
import org.example.demo.fileManagers.BinaryFileManager;
import org.example.demo.fileManagers.FileManager;
import org.example.demo.fileManagers.JsonFileManager;
import java.io.File;
import java.time.LocalDate;
import java.util.HashMap;
public class ControllerConfigureAirPurifier {
public CheckBox connectedToPowerSupply;
public CheckBox working;
public TextField fileName;
public Button exportButton;
public Label fileMessage;
@FXML
private ComboBox<String> speedOfFan;
private ToggleGroup modeHumidityGroup;
@FXML
private RadioButton modeHumidity0;
@FXML
private RadioButton modeHumidity1;
@FXML
private RadioButton modeHumidity2;
@FXML
private CheckBox modeAuto;
@FXML
private DatePicker filterReplacementDate;
@FXML
private CheckBox modeAirParticle;
@FXML
private Label errorMessage;
@FXML
private Label manufacture;
@FXML
private Label model;
@FXML
private Label powerSupply;
@FXML
private Button cancel;
@FXML
private Button save;
private SmartDevice device;
private static final Logger loggerUser = LogManager.getLogger("FileUserLogger");
public void start(Stage stage, SmartDevice \_device) {
loggerUser.info("Настройка очистителя воздуха PK="+\_device.getId());
device = \_device;
manufacture.setText(device.getManufacturer());
model.setText(device.getModel());
powerSupply.setText(device.getPowerSupply());
HashMap<String, Object> configurations = device.getConfigurations();
speedOfFan.getItems().addAll("высокая", "средняя", "низкая", "автоматическая");
speedOfFan.setValue((String) configurations.get("Скорость вентиляторов"));
filterReplacementDate.setValue((LocalDate) configurations.get("Дата замены фильтров"));
modeHumidityGroup = new ToggleGroup();
modeHumidity0.setToggleGroup(modeHumidityGroup);
modeHumidity1.setToggleGroup(modeHumidityGroup);
modeHumidity2.setToggleGroup(modeHumidityGroup);
switch ((String) configurations.get("Режим регуляции влажности")) {
case ("увлажнение"): {
modeHumidity0.setSelected(true);
break;
}
case ("осушение"):{
modeHumidity1.setSelected(true);
break;
}
case ("отключен"): {
modeHumidity2.setSelected(true);
}
}
modeAirParticle.setSelected((boolean) configurations.get("Режим регуляции уровня частиц в воздухе"));
modeAuto.setSelected((boolean)configurations.get("Автоматический режим"));
connectedToPowerSupply.setSelected((device.getConnectedToPowerSupply()));
working.setSelected((device.getOnOrOff()));
}
@FXML
private void cancelClicked() {
loggerUser.info("Отмена настроек очистителя воздуха");
ControllerListOfDevices controller = new ControllerListOfDevices();
Scene currentScene = cancel.getScene();
Stage stage = (Stage) currentScene.getWindow();
controller.start(stage);
}
public void saveClicked() {
String manufacturerValue = manufacture.getText();
String modelValue = model.getText();
String powerSupplyValue = powerSupply.getText();
String fanSpeed = speedOfFan.getValue();
LocalDate replacementDate = filterReplacementDate.getValue();
String modeHumidity;
boolean isHumidification = modeHumidity0.isSelected();
boolean isDehumidification = modeHumidity1.isSelected();
if (isHumidification) modeHumidity = "увлажнение";
else if (isDehumidification) modeHumidity = "осушение";
else modeHumidity = "отключен";
boolean isAirParticleMode = modeAirParticle.isSelected();
boolean isAutomaticMode = modeAuto.isSelected();
AirPurifier airP = new AirPurifier( manufacturerValue, modelValue, powerSupplyValue, SmartDevicesManager.getElectricityInRoom(), modeHumidity, isAirParticleMode, isAutomaticMode, fanSpeed, replacementDate);
if (working.isSelected()) {
airP.turnOn(SmartDevicesManager.getElectricityInRoom());
} else {
airP.turnOff();
}
if (connectedToPowerSupply.isSelected()) {
airP.connectToPowerSupply();
} else {
airP.disconnectFromPowerSupply();
}
if (airP.isFilterReplacementDateValid(replacementDate)) {
SmartDevice deviceTemp = airP;
deviceTemp.update(SmartDevicesManager.getRoomTemperature(), SmartDevicesManager.getRoomHumidity(), SmartDevicesManager.getRoomAirParticleLevel(), SmartDevicesManager.getElectricityInRoom());
deviceTemp.setId(device.getId());
DatabaseManager.updateDevice(deviceTemp);
ControllerListOfDevices controller = new ControllerListOfDevices();
Scene currentScene = cancel.getScene();
Stage stage = (Stage) currentScene.getWindow();
loggerUser.info("Успешное сохранение настроек очистителя воздуаха PK="+device.getId());
controller.start(stage);
} else {
loggerUser.warn("Введены неверные данные");
errorMessage.setText("Дата замены фильтров не должна превышать сегодняшнюю дату!");
}
}
public void exportButtonClicked(ActionEvent actionEvent) {
String fileNameValue = fileName.getText();
switch (FileManager.getTypeOfFile(fileNameValue)) {
case(1): { // .bat
fileMessage.setText("");
File file = new File(fileNameValue);
if (BinaryFileManager.writeIntoBAT(device, fileNameValue)) {
fileMessage.setText("Файл успешно записан!");
loggerUser.info("Успешное экспортирование настроек очистителя воздуха PK="+device.getId());
} else {
fileMessage.setText("Произошла ошибка!");
loggerUser.error("Ошибка при экспортировании");
}
break;
}
case(2):{ // .json
fileMessage.setText("");
File file = new File(fileNameValue);
if (JsonFileManager.writeIntoJSON(device, fileNameValue)) {
fileMessage.setText("Файл успешно записан!");
loggerUser.info("Успешное экспортирование настроек очистителя воздуха PK="+device.getId());
} else {
fileMessage.setText("Произошла ошибка!");
loggerUser.error("Ошибка при экспортировании");
}
break;
}
default: {
fileMessage.setText("Имя файла не \*.bat или \*.json");
}
}
}
}

ControllerConfigureHeater.java

ControllerConfigureRoom.java

ControllerListOfDevices.java

package org.example.demo.sceneControllers;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.\*;
import javafx.scene.layout.GridPane;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;
import org.example.demo.Application;
import org.example.demo.DatabaseManager;
import org.example.demo.SmartDevicesManager;
import org.example.demo.devices.SmartDevice;
import java.io.IOException;
import java.util.ArrayList;
public class ControllerListOfDevices {
@FXML
public ScrollPane listOfDevices;
@FXML
public Button addAirConditioner;
public Button addAirPurifier;
public Button addHeater;
@FXML
private VBox listOfDevicesContainer;
@FXML
private Button editRoom;
// Характеристики комнаты
@FXML
private Label temperature;
@FXML
private Label humidity;
@FXML
private Label airParticle;
@FXML
private CheckBox electricity;
private static final Logger loggerUser = LogManager.getLogger("FileUserLogger");
public void start(Stage stage) {
try {
FXMLLoader fxmlLoader = new FXMLLoader(Application.class.getResource("listOfDevices.fxml"));
Parent root = fxmlLoader.load();
ControllerListOfDevices controller = fxmlLoader.getController();
controller.setRoomCharacteristics();
controller.updateListOfDevice();
Scene scene = new Scene(root, 700, 400);
stage.setTitle("Менеджер умных устройств");
stage.setScene(scene);
stage.show();
} catch (Exception e) {
e.printStackTrace();
}
}
public void setRoomCharacteristics() {
electricity.setSelected(SmartDevicesManager.getElectricityInRoom());
temperature.setText(Float.toString(SmartDevicesManager.getRoomTemperature())+"°C");
humidity.setText(Float.toString(SmartDevicesManager.getRoomHumidity())+"%");
airParticle.setText(Float.toString(SmartDevicesManager.getRoomAirParticleLevel())+"%");
}
public void updateListOfDevice() {
try {
listOfDevicesContainer.getChildren().clear();
ArrayList<SmartDevice> devices = DatabaseManager.getDevices();
if (devices == null) return;
for (SmartDevice device : devices) {
device.update(SmartDevicesManager.getRoomTemperature(), SmartDevicesManager.getRoomHumidity(), SmartDevicesManager.getRoomAirParticleLevel(), SmartDevicesManager.getElectricityInRoom());
GridPane gridPane = createGridPane(device.getId(), device.getType(), device.getManufacturer(), device.getModel(), device.getOnOrOff());
listOfDevicesContainer.getChildren().add(gridPane);
}
listOfDevices.setFitToWidth(true);
listOfDevices.setFitToHeight(true);
} catch (RuntimeException e) {
e.printStackTrace();
}
}
private GridPane createGridPane(int id, String type, String manufacturer, String model, boolean onOrOff) {
try {
FXMLLoader fxmlLoader = new FXMLLoader(Application.class.getResource("itemDevice.fxml"));
GridPane gridPane = fxmlLoader.load();
Button deleteDeviceButton = new Button("Удалить");
deleteDeviceButton.setOnAction(event -> {
DatabaseManager.deleteDevice(id);
updateListOfDevice();
});
GridPane.setConstraints(deleteDeviceButton, 0, 0);
gridPane.getChildren().add(deleteDeviceButton);
Button configureDeviceButton = new Button("Изменить");
configureDeviceButton.setOnAction(event -> {
SmartDevice device = DatabaseManager.getDevice(id);
switch (device.getType()) {
case "AirConditioner": {
Scene currentScene = addAirConditioner.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("configureAirConditioner.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
ControllerConfigureAirConditioner controller = loader.getController();
controller.start(stage, device);
} catch (IOException e) {
e.printStackTrace();
}
break;
}
case "AirPurifier": {
Scene currentScene = addAirConditioner.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("configureAirPurifier.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
ControllerConfigureAirPurifier controller = loader.getController();
controller.start(stage, device);
} catch (IOException e) {
e.printStackTrace();
}
break;
}
case "Heater": {
Scene currentScene = addAirConditioner.getScene();
Stage stage = (Stage) currentScene.getWindow();
try {
FXMLLoader loader = new FXMLLoader(Application.class.getResource("configureHeater.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
ControllerConfigureHeater controller = loader.getController();
controller.start(stage, device);
} catch (IOException e) {
e.printStackTrace();
}
}
}
});
GridPane.setConstraints(configureDeviceButton, 0, 1);
gridPane.getChildren().add(configureDeviceButton);
Label labelType = new Label(type);
GridPane.setConstraints(labelType, 1, 0);
gridPane.getChildren().add(labelType);
Label labelManufacturer = new Label(manufacturer);
GridPane.setConstraints(labelManufacturer, 2, 0);
gridPane.getChildren().add(labelManufacturer);
Label labelModel = new Label(model);
GridPane.setConstraints(labelModel, 3, 0);
gridPane.getChildren().add(labelModel);
Label labelOnOrOff;
if (onOrOff) {
labelOnOrOff = new Label("Работает");
} else {
labelOnOrOff = new Label("Отключено");
}
GridPane.setConstraints(labelOnOrOff, 4, 0);
gridPane.getChildren().add(labelOnOrOff);
return gridPane;
} catch (IOException e) {
e.printStackTrace();
}
return null;
}
public void editRoomClicked() {
loggerUser.info("Нажата кнопка редактирование комнаты");
// Получаем сцену из кнопки
Scene currentScene = addAirConditioner.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("configureRoom.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
ControllerConfigureRoom controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
public void addAirConditionerClicked() {
loggerUser.info("Нажата кнопка добавления кондиционера");
// Получаем сцену из кнопки
Scene currentScene = addAirConditioner.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("addAirConditioner.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
ControllerAddAirConditioner controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
public void addAirPurifierClicked() {
loggerUser.info("Нажата кнопка добавления очистителя воздуха");
// Получаем сцену из кнопки
Scene currentScene = addAirPurifier.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("addAirPurifier.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
ControllerAddAirPurifier controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
public void addHeaterClicked() {
loggerUser.info("Нажата кнопка добавления обогревателя");
// Получаем сцену из кнопки
Scene currentScene = addAirConditioner.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("addHeater.fxml"));
Parent root = loader.load();
stage.setScene(new Scene(root));
ControllerAddHeater controller = loader.getController();
controller.start(stage);
} catch (IOException e) {
e.printStackTrace();
}
}
}

SmartDevicesManager.java

module-info.java

module org.example.demo {
requires javafx.controls;
requires javafx.fxml;
requires org.apache.logging.log4j;
requires java.sql;
requires com.google.gson;
opens org.example.demo to javafx.fxml;
opens org.example.demo.devices to com.google.gson;
exports org.example.demo;
exports org.example.demo.sceneControllers;
opens org.example.demo.sceneControllers to javafx.fxml;
exports org.example.demo.fileManagers;
opens org.example.demo.fileManagers to javafx.fxml;
}

Application.java

package org.example.demo;
import javafx.stage.Stage;
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;
import org.example.demo.sceneControllers.ControllerListOfDevices;
import java.io.InputStream;
import java.io.IOException;
import java.util.HashMap;
import java.util.Properties;
public class Application extends javafx.application.Application {
private static final Logger logger = LogManager.getLogger("FileFilesLogger");
private static final Logger loggerUser = LogManager.getLogger("FileUserLogger");
private static void configureDB() {
Properties properties = new Properties();
HashMap<String, String> configInfo = new HashMap<>();
String[] tables = new String[0];
try (InputStream input = Application.class.getClassLoader().getResourceAsStream("application.properties")) {
if (input == null) {
logger.fatal("Ошибка чтения конфигурационного файла!");
return;
}
properties.load(input);
configInfo.put("dbName", properties.getProperty("application.dbName"));
configInfo.put("dbUser", properties.getProperty("application.dbUser"));
configInfo.put("dbPass", properties.getProperty("application.dbPass"));
configInfo.put("table1", properties.getProperty("application.table1"));
configInfo.put("table2", properties.getProperty("application.table2"));
configInfo.put("table3", properties.getProperty("application.table3"));
tables = new String[]{configInfo.get("table1"), configInfo.get("table2"), configInfo.get("table3")};
} catch (IOException e) {
e.printStackTrace();
}
DatabaseConnector.setValuesForConnection(configInfo.get("dbName"), configInfo.get("dbUser"), configInfo.get("dbPass"), tables);
}
@Override
public void start(Stage stage) throws Exception {
ControllerListOfDevices controller = new ControllerListOfDevices();
controller.start(stage);
}
public static void main(String[] args) {
loggerUser.info("Программа начала работу");
configureDB();
launch();
loggerUser.info("Программа успешно закончила работу");
}
}

DatabaseConnector.java

package org.example.demo;
import org.example.demo.devices.AirConditioner;
import org.example.demo.devices.AirPurifier;
import org.example.demo.devices.Heater;
import org.example.demo.devices.SmartDevice;
import org.example.demo.fileManagers.BinaryFileManager;
import java.io.ByteArrayInputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.sql.\*;
import java.util.ArrayList;
public class DatabaseConnector {
static String url = "jdbc:mysql://localhost:3306/";
static String dbName;
static String username;
static String password;
static String[] tableNames;
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 void setValuesForConnection(String \_dbName, String \_username, String \_password, String[] \_tableNames) {
url += \_dbName;
dbName = \_dbName;
username = \_username;
password = \_password;
tableNames = \_tableNames;
}
public static ArrayList<SmartDevice> getDevices() {
ArrayList<SmartDevice> devices = new ArrayList<>();
String query = "SELECT \* FROM devices";
try {
Connection connection = DriverManager.getConnection(url, username, password);
Statement statement = connection.createStatement();
ResultSet resultSet = statement.executeQuery(query);
while (resultSet.next()) {
int idDevice = resultSet.getInt("id");
Blob dataDevice = resultSet.getBlob("data");
byte[] data = dataDevice.getBytes(1, (int) dataDevice.length());
ByteArrayInputStream bais = new ByteArrayInputStream(data);
ObjectInputStream ois = new ObjectInputStream(bais);
SmartDevice obj = (SmartDevice) ois.readObject();
ois.close();
bais.close();
if (obj instanceof AirConditioner) {
SmartDevice resObj = (AirConditioner) obj;
resObj.setId(idDevice);
devices.add(resObj);
} else if (obj instanceof AirPurifier) {
SmartDevice resObj = (AirPurifier) obj;
resObj.setId(idDevice);
devices.add(resObj);
} else if (obj instanceof Heater) {
SmartDevice resObj = (Heater) obj;
resObj.setId(idDevice);
devices.add(resObj);
} else {
// loger
}
}
connection.close();
return devices;
} catch (SQLException | IOException | ClassNotFoundException e) {
e.printStackTrace();
}
return null;
}
public static SmartDevice getDevice(int id) {
String query = "SELECT \* FROM devices WHERE id = ?";
try {
Connection connection = DriverManager.getConnection(url, username, password);
PreparedStatement statement = connection.prepareStatement(query);
statement.setInt(1, id);
ResultSet resultSet = statement.executeQuery();
while (resultSet.next()) {
int idDevice = resultSet.getInt("id");
Blob dataDevice = resultSet.getBlob("data");
byte[] data = dataDevice.getBytes(1, (int) dataDevice.length());
ByteArrayInputStream bais = new ByteArrayInputStream(data);
ObjectInputStream ois = new ObjectInputStream(bais);
SmartDevice obj = (SmartDevice) ois.readObject();
ois.close();
bais.close();
if (obj instanceof AirConditioner) {
SmartDevice resObj = (AirConditioner) obj;
resObj.setId(idDevice);
return resObj;
} else if (obj instanceof AirPurifier) {
SmartDevice resObj = (AirPurifier) obj;
resObj.setId(idDevice);
return resObj;
} else if (obj instanceof Heater) {
SmartDevice resObj = (Heater) obj;
resObj.setId(idDevice);
return resObj;
} else {
// loger
}
}
connection.close();
} catch (SQLException | IOException | ClassNotFoundException e) {
e.printStackTrace();
}
return null;
}
public static void insertDevice(SmartDevice device) {
String query = "INSERT INTO devices (data) VALUES (?)";
try (Connection connection = DriverManager.getConnection(url, username, password);
PreparedStatement preparedStatement = connection.prepareStatement(query, Statement.RETURN\_GENERATED\_KEYS)) {
// вставка
byte[] serializedObject = BinaryFileManager.getByteData(device);
ByteArrayInputStream data = new ByteArrayInputStream(serializedObject);
preparedStatement.setBinaryStream(1, data, serializedObject.length);
int affectedRows = preparedStatement.executeUpdate();
// получение id
if (affectedRows > 0) {
ResultSet generatedKeys = preparedStatement.getGeneratedKeys();
if (generatedKeys.next()) {
int generatedId = generatedKeys.getInt(1); // Получение сгенерированного ключа
device.setId(generatedId);
}
} else {
}
connection.close();
} catch (SQLException e) {
e.printStackTrace();
}
}
public static boolean updateDevice(SmartDevice device) {
String updateQuery = "UPDATE devices SET data = ? WHERE id = ?";
try {
Connection connection = DriverManager.getConnection(url, username, password);
PreparedStatement preparedStatement = connection.prepareStatement(updateQuery);
byte[] serializedObject = BinaryFileManager.getByteData(device);
ByteArrayInputStream data = new ByteArrayInputStream(serializedObject);
preparedStatement.setInt(2, device.getId());
preparedStatement.setBinaryStream(1, data, serializedObject.length);
int rowsAffected = preparedStatement.executeUpdate();
if (rowsAffected > 0) {
// System.out.println("Запись успешно обновлена");
connection.close();
return true;
} else {
// System.out.println("Запись не была обновлена");
connection.close();
return false;
}
} catch (SQLException e) {
e.printStackTrace();
return false;
}
}
public static boolean deleteDevice(int id) {
String deleteQuery = "DELETE FROM devices WHERE id = ?";
try (Connection connection = DriverManager.getConnection(url, username, password);
PreparedStatement preparedStatement = connection.prepareStatement(deleteQuery)) {
preparedStatement.setInt(1, id);
int rowsAffected = preparedStatement.executeUpdate();
if (rowsAffected > 0) {
// System.out.println("Запись успешно обновлена");
connection.close();
return true;
} else {
// System.out.println("Запись не была обновлена");
connection.close();
return false;
}
} catch (SQLException e) {
e.printStackTrace();
return false;
}
}
}

DatabaseReaderThreaded.java

package org.example.demo;
import org.example.demo.devices.AirConditioner;
import org.example.demo.devices.AirPurifier;
import org.example.demo.devices.Heater;
import org.example.demo.devices.SmartDevice;
import java.sql.\*;
import java.util.ArrayList;
public class DatabaseReaderThreaded extends Thread {
private final String tableName;
public DatabaseReaderThreaded(String tableName) {
this.tableName = tableName;
}
@Override
public void run() {
String query = "SELECT \* FROM " + tableName;
try (Connection connection = DriverManager.getConnection(DatabaseConnector.getUrl(), DatabaseConnector.getUsername(), DatabaseConnector.getPassword());
Statement statement = connection.createStatement();
ResultSet resultSet = statement.executeQuery(query)) {
while (resultSet.next()) {
int id = resultSet.getInt("id");
String type = resultSet.getString("type");
String manufacturer = resultSet.getString("manufacturer");
String model = resultSet.getString("model");
String powerSupply = resultSet.getString("powerSupply");
switch (type) {
case "AirConditioner": {
float temperature = resultSet.getFloat("temperature");
String modeTemperature = resultSet.getString("modeTemperature");
String modeHumidity = resultSet.getString("modeHumidity");
boolean modeAuto = resultSet.getBoolean("modeAuto");
String speedOfFan = resultSet.getString("speedOfFan");
String filterReplacementDate = resultSet.getString("filterReplacementDate");
SmartDevice device = new AirConditioner(id, manufacturer, model, powerSupply, Room.getElectricityInRoom(), temperature , modeTemperature, modeHumidity, modeAuto, speedOfFan, filterReplacementDate);
Room.addDevices(device);
break;
}
case "AirPurifier": {
String modeHumidity = resultSet.getString("modeHumidity");
boolean modeAuto = resultSet.getBoolean("modeAuto");
boolean modeAirParticle = resultSet.getBoolean("modeAirParticle");
String speedOfFan = resultSet.getString("speedOfFan");
String filterReplacementDate = resultSet.getString("filterReplacementDate");
SmartDevice device = new AirPurifier(id, manufacturer, model, powerSupply, Room.getElectricityInRoom(), modeHumidity , modeAirParticle, modeAuto, speedOfFan, filterReplacementDate);
Room.addDevices(device);
break;
}
case "Heater": {
float minTemperature = resultSet.getFloat("minTemperature");
float maxTemperature = resultSet.getFloat("maxTemperature");
boolean modeHeater = resultSet.getBoolean("modeHeater");
boolean modeAuto = resultSet.getBoolean("modeAuto");
float angle = resultSet.getFloat("angle");
SmartDevice device = new Heater(id, manufacturer, model, powerSupply, Room.getElectricityInRoom(), minTemperature, maxTemperature, modeHeater, modeAuto, angle);
Room.addDevices(device);
break;
}
default: {
System.err.println("Не распознана запись в таблице " + tableName);
}
}
}
connection.close();
} catch (SQLException e) {
e.printStackTrace();
}
}
}

DatabaseWriterThreaded.java

package org.example.demo;
import org.example.demo.devices.AirConditioner;
import org.example.demo.devices.AirPurifier;
import org.example.demo.devices.Heater;
import org.example.demo.devices.SmartDevice;
import java.sql.\*;
import java.util.ArrayList;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.TimeUnit;
public class DatabaseWriterThreaded {
public static void write(ArrayList<SmartDevice> oldDevices) {
ExecutorService executorService = Executors.newFixedThreadPool(oldDevices.size());
for (SmartDevice device : oldDevices) { // Проходимся по старым, потому что меняем Room.devices
Runnable task = () -> {
DatabaseWriterThreaded.insertDevice(device);
};
executorService.submit(task); // Добавление задачи в пул потоков
}
executorService.shutdown(); // Остановка пула потоков после завершения всех задач
try {
if (!executorService.awaitTermination(60, TimeUnit.SECONDS)) {
executorService.shutdownNow(); // Прерывание выполнения задач через 60 секунд, если они не завершены
}
} catch (InterruptedException e) {
executorService.shutdownNow(); // Прерывание выполнения задач в случае исключения
}
}
public static void insertDevice(SmartDevice device) {
switch (device.getType()) {
case "AirConditioner": {
System.out.println("AirConditioner1");
insertAirConditioner((AirConditioner) device);
System.out.println("AirConditioner2");
break;
}
case "AirPurifier": {
insertAirPurifier((AirPurifier) device);
System.out.println("AirPurifier");
break;
}
case "Heater": {
insertHeater((Heater) device);
System.out.println("Heater");
break;
}
default: {
System.err.println("Не распознана");
}
}
}
private static void insertAirConditioner(AirConditioner device) {
String query = "INSERT INTO " + DatabaseConnector.tableNames[0] + " (type, manufacturer, model, powerSupply, temperature, modeTemperature, modeHumidity, modeAuto, speedOfFan, filterReplacementDate) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
try (Connection connection = DriverManager.getConnection(DatabaseConnector.getUrl(), DatabaseConnector.getUsername(), DatabaseConnector.getPassword());
PreparedStatement preparedStatement = connection.prepareStatement(query, Statement.RETURN\_GENERATED\_KEYS)) {
// вставка
preparedStatement.setString(1, device.getType());
preparedStatement.setString(2, device.getManufacturer());
preparedStatement.setString(3, device.getModel());
preparedStatement.setString(4, device.getPowerSupply());
preparedStatement.setFloat(5, device.getTemperature());
preparedStatement.setString(6, device.getModeTemperature());
preparedStatement.setString(7, device.getModeHumidity());
preparedStatement.setBoolean(8, device.getModeAuto());
preparedStatement.setString(9, device.getSpeedOfFan());
preparedStatement.setString(10, device.getFilterReplacementDate().toString());
// выполнение запроса
int affectedRows = preparedStatement.executeUpdate();
// получение id
if (affectedRows > 0) {
ResultSet generatedKeys = preparedStatement.getGeneratedKeys();
if (generatedKeys.next()) {
int generatedId = generatedKeys.getInt(1); // Получение сгенерированного ключа
device.setId(generatedId);
Room.addDevices(device);
}
} else {
System.out.println("Error of connecting");
}
connection.close();
} catch (SQLException e) {
e.printStackTrace();
}
}
private static void insertAirPurifier(AirPurifier device) {
String query = "INSERT INTO " + DatabaseConnector.tableNames[1] + " (type, manufacturer, model, powerSupply, modeHumidity, modeAirParticle, modeAuto, speedOfFan, filterReplacementDate) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)";
try (Connection connection = DriverManager.getConnection(DatabaseConnector.getUrl(), DatabaseConnector.getUsername(), DatabaseConnector.getPassword());
PreparedStatement preparedStatement = connection.prepareStatement(query, Statement.RETURN\_GENERATED\_KEYS)) {
// вставка
preparedStatement.setString(1, device.getType());
preparedStatement.setString(2, device.getManufacturer());
preparedStatement.setString(3, device.getModel());
preparedStatement.setString(4, device.getPowerSupply());
preparedStatement.setString(5, device.getModeHumidity());
preparedStatement.setBoolean(6, device.getModeAirParticle());
preparedStatement.setBoolean(7, device.getModeAuto());
preparedStatement.setString(8, device.getSpeedOfFan());
preparedStatement.setString(9, device.getFilterReplacementDate().toString());
// выполнение запроса
int affectedRows = preparedStatement.executeUpdate();
// получение id
if (affectedRows > 0) {
ResultSet generatedKeys = preparedStatement.getGeneratedKeys();
if (generatedKeys.next()) {
int generatedId = generatedKeys.getInt(1); // Получение сгенерированного ключа
device.setId(generatedId);
Room.addDevices(device);
}
} else {
// err
}
connection.close();
} catch (SQLException e) {
e.printStackTrace();
}
}
private static void insertHeater(Heater device) {
String query = "INSERT INTO " + DatabaseConnector.tableNames[2] + " (type, manufacturer, model, powerSupply, minTemperature, maxTemperature, modeHeater, modeAuto, angle) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)";
try (Connection connection = DriverManager.getConnection(DatabaseConnector.getUrl(), DatabaseConnector.getUsername(), DatabaseConnector.getPassword());
PreparedStatement preparedStatement = connection.prepareStatement(query, Statement.RETURN\_GENERATED\_KEYS)) {
// вставка
preparedStatement.setString(1, device.getType());
preparedStatement.setString(2, device.getManufacturer());
preparedStatement.setString(3, device.getModel());
preparedStatement.setString(4, device.getPowerSupply());
preparedStatement.setFloat(5, device.getMinTemperature());
preparedStatement.setFloat(6, device.getMaxTemperature());
preparedStatement.setBoolean(7, device.getModeHeater());
preparedStatement.setBoolean(8, device.getModeAuto());
preparedStatement.setFloat(9, device.getAngle());
// выполнение запроса
int affectedRows = preparedStatement.executeUpdate();
// получение id
if (affectedRows > 0) {
ResultSet generatedKeys = preparedStatement.getGeneratedKeys();
if (generatedKeys.next()) {
int generatedId = generatedKeys.getInt(1); // Получение сгенерированного ключа
device.setId(generatedId);
Room.addDevices(device);
}
} else {
// err
}
connection.close();
} catch (SQLException e) {
e.printStackTrace();
}
}
}

AirConditioner.java

AirPurifier.java

Heater.java

SmartDevice.java

package org.example.demo.devices;
import java.io.Serializable;
import java.util.HashMap;
abstract public class SmartDevice implements Serializable {
private final String type;
private final String manufacturer;
//public void setMan(String \_manufacturer) {
// manufacturer = \_manufacturer;
// }
private final String model;
private final String powerSupply;
private boolean onOrOff; // True if ON
private boolean connectedToPowerSupply; //battery or powerSupplyNetwork
// Additional
transient protected int id; // для таблицы
static protected int maxId = 0;
SmartDevice(String \_type, String \_manufacturer, String \_model, String \_powerSupply, boolean \_electricityInRoom) {
type = \_type;
manufacturer = \_manufacturer;
model = \_model;
powerSupply = \_powerSupply;
if (powerSupply.equals("сеть")) {
onOrOff = \_electricityInRoom;
} else onOrOff = true;
connectedToPowerSupply = true;
}
public String getType(){
return type;
}
public String getManufacturer() {
return manufacturer;
}
public String getModel() {
return model;
}
public String getPowerSupply(){
return powerSupply;
}
public boolean getOnOrOff() { return onOrOff; }
public boolean getConnectedToPowerSupply(){
return connectedToPowerSupply;
}
public int getId() { return id; }
public void setId(int id) {
this.id = id;
}
protected void setOnOrOff(boolean var){
onOrOff = var;
}
protected void setConnectedToPowerSupply(boolean var) {
connectedToPowerSupply = var;
}
abstract public boolean turnOn(boolean electricityInRoom);
abstract public void turnOff();
abstract public void connectToPowerSupply();
abstract public void disconnectFromPowerSupply();
abstract public void update(float \_roomTemperature, float \_roomHumidity, float \_roomAirParticleLevel, boolean \_electricityInRoom);
// Additional
abstract public HashMap<String, Object> getConfigurations();
abstract public void setConfigurations(HashMap<String, Object> configurations);
public void rebootId() {
this.id = ++maxId;
}
}

BinaryFileManager.java

package org.example.demo.fileManagers;
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;
import org.example.demo.Room;
import org.example.demo.devices.AirConditioner;
import org.example.demo.devices.AirPurifier;
import org.example.demo.devices.Heater;
import org.example.demo.devices.SmartDevice;
import java.io.\*;
import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.CountDownLatch;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;
public class BinaryFileManager {
private static final Logger logger = LogManager.getLogger("FileFilesLogger");
public static boolean isValidFileName(String fileName) {
String fileNameRegex = "^[^\\\\/:\*?\"<>|.]\*$";
Pattern pattern = Pattern.compile(fileNameRegex);
Matcher matcher = pattern.matcher(fileName);
return matcher.matches();
}
public static boolean writeIntoBAT(SmartDevice device, String fileName) {
try (FileOutputStream fileOutputStream = new FileOutputStream(fileName); ObjectOutputStream objectOutputStream = new ObjectOutputStream(fileOutputStream)) {
objectOutputStream.writeObject(device);
logger.info("Успешная запись PK=" + device.getId() + " в бинарный файл: " + fileName);
return true;
} catch (Exception e) {
e.printStackTrace();
}
return false;
}
public static SmartDevice readFromBAT(File file) {
try (ObjectInputStream objectInputStream = new ObjectInputStream(new FileInputStream(file))) {
SmartDevice device = (SmartDevice) objectInputStream.readObject();
if (device instanceof AirPurifier || device instanceof AirConditioner || device instanceof Heater) {
logger.info("Чтение из бинарного файла: информация об устройстве типа " + device.getType() + " успешно получена");
return device;
} else {
logger.error("Чтение из бинарного файла: информация получена, но объект не идентифицирован");
}
} catch (ClassNotFoundException | IOException e) {
e.printStackTrace();
logger.error("Чтение из бинарного файла: произошла ошибка чтения файла");
}
return null;
}
public static byte[] getByteData(SmartDevice device) {
try (ByteArrayOutputStream baos = new ByteArrayOutputStream()) {
ObjectOutputStream oos = new ObjectOutputStream(baos);
oos.writeObject(device);
byte[] serializedObject = baos.toByteArray();
return serializedObject;
} catch (IOException e) {
e.printStackTrace();
}
return null;
}
private static final Lock lock = new ReentrantLock();
public static void saveDevicesToBinaryFile(String filePath) {
List<SmartDevice> devices = Room.getDevices();
try (ObjectOutputStream outputStream = new ObjectOutputStream(new FileOutputStream(filePath))) {
devices.parallelStream().forEach(device -> {
try {
lock.lock();
outputStream.writeObject(device);
} catch (IOException e) {
e.printStackTrace();
} finally {
lock.unlock();
}
});
} catch (IOException e) {
e.printStackTrace();
}
}
public static ArrayList<SmartDevice> readDevicesFromBinaryFile(String filePath) {
ArrayList<SmartDevice> devices = new ArrayList<>();
try (ObjectInputStream inputStream = new ObjectInputStream(new FileInputStream(filePath))) {
SmartDevice device;
while ((device = (SmartDevice) inputStream.readObject()) != null) {
devices.add(device);
}
} catch (EOFException e) {
} catch (IOException | ClassNotFoundException e) {
e.printStackTrace();
}
return devices;
}
}

FileNameChecker.java

package org.example.demo.fileManagers;
public class FileNameChecker {
public static int getTypeOfFile(String fileName) {
if (fileName.matches("[a-zA-Z0-9\\-\_]+\\.bat")) {
return 1;
} else if (fileName.matches("[a-zA-Z0-9\\-\_]+\\.json")) {
return 2;
} else return 0;
}
}

JsonFileManager.java

LocalDateAdapter.java

package org.example.demo.fileManagers;
import com.google.gson.\*;
import java.lang.reflect.Type;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
public class LocalDateAdapter implements JsonSerializer<LocalDate>, JsonDeserializer<LocalDate> {
private static final DateTimeFormatter DATE\_FORMATTER = DateTimeFormatter.ISO\_LOCAL\_DATE;
@Override
public JsonElement serialize(LocalDate date, Type typeOfSrc, JsonSerializationContext context) {
return new JsonPrimitive(date.format(DATE\_FORMATTER));
}
@Override
public LocalDate deserialize(JsonElement json, Type typeOfT, JsonDeserializationContext context) throws JsonParseException {
return LocalDate.parse(json.getAsString(), DATE\_FORMATTER);
}
}

Room.java

ControllerAddAirConditioner.java

ControllerAddAirPurifier.java

package org.example.demo.sceneControllers;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.Scene;
import javafx.scene.control.\*;
import javafx.stage.Stage;
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;
import org.example.demo.Room;
import org.example.demo.devices.AirPurifier;
import org.example.demo.devices.SmartDevice;
import org.example.demo.fileManagers.BinaryFileManager;
import org.example.demo.fileManagers.FileNameChecker;
import org.example.demo.fileManagers.JsonFileManager;
import java.io.File;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.util.HashMap;
public class ControllerAddAirPurifier {
public TextField fileName;
public Button importButton;
public Label fileMessage;
@FXML
private Label errorMessage;
@FXML
private ComboBox<String> speedOfFan;
private ToggleGroup modeHumidityGroup;
@FXML
public RadioButton modeHumidity0;
@FXML
public RadioButton modeHumidity1;
@FXML
public RadioButton modeHumidity2;
@FXML
public CheckBox modeAuto;
@FXML
public DatePicker filterReplacementDate;
@FXML
public TextField manufacture;
@FXML
public TextField model;
@FXML
public ComboBox<String> powerSupply;
@FXML
public CheckBox modeAirParticle;
@FXML
public Button cancel;
@FXML
public Button save;
private static final Logger loggerUser = LogManager.getLogger("FileUserLogger");
public void start(Stage stage) {
loggerUser.info("Добавление очистителя воздуха");
manufacture.setText("manufacture");
model.setText("model");
powerSupply.getItems().addAll("сеть", "аккумулятор", "батарейки");
powerSupply.setValue("сеть");
speedOfFan.getItems().addAll("высокая", "средняя", "низкая", "автоматическая");
speedOfFan.setValue("автоматическая");
String dateString = "2000-01-01";
DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd");
filterReplacementDate.setValue(LocalDate.parse(dateString, formatter));
modeHumidityGroup = new ToggleGroup();
modeHumidity0.setToggleGroup(modeHumidityGroup);
modeHumidity1.setToggleGroup(modeHumidityGroup);
modeHumidity2.setToggleGroup(modeHumidityGroup);
modeHumidity2.setSelected(true);
modeAuto.setSelected(true);
}
public void cancelClicked() {
loggerUser.info("Отмена добавления очистителя воздуха");
ControllerListOfDevices controller = new ControllerListOfDevices();
Scene currentScene = cancel.getScene();
Stage stage = (Stage) currentScene.getWindow();
controller.start(stage);
}
public void saveClicked() {
String manufacturerValue = manufacture.getText();
String modelValue = model.getText();
String powerSupplyValue = powerSupply.getValue();
String fanSpeed = speedOfFan.getValue();
LocalDate replacementDate = filterReplacementDate.getValue();
String modeHumidity;
boolean isHumidification = modeHumidity0.isSelected();
boolean isDehumidification = modeHumidity1.isSelected();
if (isHumidification) modeHumidity = "увлажнение";
else if (isDehumidification) modeHumidity = "осушение";
else modeHumidity = "отключен";
boolean isAirParticleMode = modeAirParticle.isSelected();
boolean isAutomaticMode = modeAuto.isSelected();
AirPurifier airP = new AirPurifier( manufacturerValue, modelValue, powerSupplyValue, Room.getElectricityInRoom(), modeHumidity, isAirParticleMode, isAutomaticMode, fanSpeed, replacementDate);
if (airP.isFilterReplacementDateValid(replacementDate)) {
SmartDevice device = airP;
device.update(Room.getRoomTemperature(), Room.getRoomHumidity(), Room.getRoomAirParticleLevel(), Room.getElectricityInRoom());
Room.insert(device);
ControllerListOfDevices controller = new ControllerListOfDevices();
Scene currentScene = cancel.getScene();
Stage stage = (Stage) currentScene.getWindow();
loggerUser.info("Успешное сохранение очистителя воздуха");
controller.start(stage);
} else {
loggerUser.error("Введены неправильные данные");
errorMessage.setText("Дата замены фильтров не должна превышать сегодняшнюю дату!");
}
}
public void importButtonClicked(ActionEvent actionEvent) {
loggerUser.info("Попытка импортирования очистителя воздуха");
String fileNameValue = fileName.getText();
switch (FileNameChecker.getTypeOfFile(fileNameValue)) {
case(1): {
fileMessage.setText("");
File file = new File(fileNameValue);
if (file.exists()) {
SmartDevice device = BinaryFileManager.readFromBAT(file);
if (device == null) {
fileMessage.setText("Произошла ошибка чтения файла!");
} else {
if (device.getType().equals("AirPurifier")) {
setFields((AirPurifier) device);
loggerUser.info("Успешное импортирование очистителя воздуха");
} else {
fileMessage.setText("В файле содержится не очиститель воздуха!");
}
}
} else {
fileMessage.setText("Такой файл не существует!");
}
break;
}
case(2):{
fileMessage.setText("");
File file = new File(fileNameValue);
if (file.exists()) {
SmartDevice device = JsonFileManager.readFromJSON(file);
if (device == null) {
fileMessage.setText("Произошла ошибка чтения файла!");
} else {
if (device.getType().equals("AirPurifier")) {
setFields((AirPurifier) device);
loggerUser.info("Успешное импортирование очистителя воздуха");
} else {
fileMessage.setText("В файле содержится не очиститель воздуха!");
}
}
} else {
fileMessage.setText("Такой файл не существует!");
}
break;
}
default: {
fileMessage.setText("Имя файла не соответствует .bat или .json");
}
}
}
public void setFields (AirPurifier device) {
manufacture.setText(device.getManufacturer());
model.setText(device.getModel());
powerSupply.setValue(device.getPowerSupply());
HashMap<String, Object> configurations = device.getConfigurations();
speedOfFan.getItems().addAll("высокая", "средняя", "низкая", "автоматическая");
speedOfFan.setValue((String) configurations.get("Скорость вентиляторов"));
filterReplacementDate.setValue((LocalDate) configurations.get("Дата замены фильтров"));
modeHumidityGroup = new ToggleGroup();
modeHumidity0.setToggleGroup(modeHumidityGroup);
modeHumidity1.setToggleGroup(modeHumidityGroup);
modeHumidity2.setToggleGroup(modeHumidityGroup);
switch ((String) configurations.get("Режим регуляции влажности")) {
case ("увлажнение"): {
modeHumidity0.setSelected(true);
break;
}
case ("осушение"):{
modeHumidity1.setSelected(true);
break;
}
case ("отключен"): {
modeHumidity2.setSelected(true);
}
}
modeAuto.setSelected((boolean)configurations.get("Автоматический режим"));
}
}

ControllerAddHeater.java

ControllerConfigureAirConditioner.java

package org.example.demo.sceneControllers;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.Scene;
import javafx.scene.control.\*;
import javafx.stage.Stage;
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;
import org.example.demo.Room;
import org.example.demo.devices.AirConditioner;
import org.example.demo.devices.SmartDevice;
import org.example.demo.fileManagers.BinaryFileManager;
import org.example.demo.fileManagers.FileNameChecker;
import org.example.demo.fileManagers.JsonFileManager;
import java.io.File;
import java.time.LocalDate;
import java.util.HashMap;
public class ControllerConfigureAirConditioner {
@FXML
public Label manufacture;
@FXML
public Label model;
@FXML
public Label powerSupply;
public CheckBox connectedToPowerSupply;
public CheckBox working;
public TextField fileName;
public Button exportButton;
public Label fileMessage;
@FXML
private Spinner<Double> temperature;
@FXML
private ComboBox<String> speedOfFan;
private ToggleGroup modeTemperatureGroup;
@FXML
private RadioButton modeTemperature0;
@FXML
private RadioButton modeTemperature1;
@FXML
private RadioButton modeTemperature2;
private ToggleGroup modeHumidityGroup;
@FXML
private RadioButton modeHumidity0;
@FXML
private RadioButton modeHumidity1;
@FXML
private RadioButton modeHumidity2;
@FXML
private CheckBox modeAuto;
@FXML
private DatePicker filterReplacementDate;
@FXML
private Button cancel;
@FXML
private Button save;
@FXML
private Label errorMessage;
private SmartDevice device;
private static final Logger loggerUser = LogManager.getLogger("FileUserLogger");
public void start(Stage stage, SmartDevice \_device) {
loggerUser.info("Настройка кондиционера PK="+\_device.getId());
device = \_device;
manufacture.setText(device.getManufacturer());
model.setText(device.getModel());
powerSupply.setText(device.getPowerSupply());
HashMap<String, Object> configurations = device.getConfigurations();
SpinnerValueFactory<Double> valueFactory =
new SpinnerValueFactory.DoubleSpinnerValueFactory(16.0, 27.0, Double.valueOf((float)configurations.get("Температура")), 0.1);
temperature.setValueFactory(valueFactory);
speedOfFan.getItems().addAll("высокая", "средняя", "низкая", "автоматическая");
speedOfFan.setValue((String) configurations.get("Скорость вентиляторов"));
filterReplacementDate.setValue((LocalDate) configurations.get("Дата замены фильтров"));
modeTemperatureGroup = new ToggleGroup();
modeTemperature0.setToggleGroup(modeTemperatureGroup);
modeTemperature1.setToggleGroup(modeTemperatureGroup);
modeTemperature2.setToggleGroup(modeTemperatureGroup);
if (((String) configurations.get("Режим регуляции температуры")).equals("охлаждение")){
modeTemperature0.setSelected(true);
} else if (((String) configurations.get("Режим регуляции температуры")).equals("обогрев")) {
modeTemperature1.setSelected(true);
} else {
modeTemperature2.setSelected(true);
}
modeHumidityGroup = new ToggleGroup();
modeHumidity0.setToggleGroup(modeHumidityGroup);
modeHumidity1.setToggleGroup(modeHumidityGroup);
modeHumidity2.setToggleGroup(modeHumidityGroup);
switch ((String) configurations.get("Режим регуляции влажности")) {
case ("увлажнение"): {
modeHumidity0.setSelected(true);
break;
}
case ("осушение"):{
modeHumidity1.setSelected(true);
break;
}
case ("отключен"): {
modeHumidity2.setSelected(true);
}
}
modeAuto.setSelected((boolean)configurations.get("Автоматический режим"));
connectedToPowerSupply.setSelected((device.getConnectedToPowerSupply()));
working.setSelected((device.getOnOrOff()));
}
@FXML
private void cancelClicked() {
loggerUser.info("Отмена настроек кондиционера");
ControllerListOfDevices controller = new ControllerListOfDevices();
Scene currentScene = cancel.getScene();
Stage stage = (Stage) currentScene.getWindow();
controller.start(stage);
}
@FXML
private void saveClicked() {
String manufacturerValue = manufacture.getText();
String modelValue = model.getText();
String powerSupplyValue = powerSupply.getText();
double tempValue = temperature.getValue();
String fanSpeed = speedOfFan.getValue();
LocalDate replacementDate = filterReplacementDate.getValue();
String modeTemperature;
boolean cooling = modeTemperature0.isSelected();
boolean heating = modeTemperature1.isSelected();
if (cooling) modeTemperature = "охлаждение";
else if (heating) modeTemperature = "обогрев";
else modeTemperature = "отключен";
String modeHumidity;
boolean isHumidification = modeHumidity0.isSelected();
boolean isDehumidification = modeHumidity1.isSelected();
if (isHumidification) modeHumidity = "увлажнение";
else if (isDehumidification) modeHumidity = "осушение";
else modeHumidity = "отключен";
boolean isAutomaticMode = modeAuto.isSelected();
AirConditioner airC = new AirConditioner( manufacturerValue, modelValue, powerSupplyValue, Room.getElectricityInRoom(), (float) tempValue , modeTemperature, modeHumidity, isAutomaticMode, fanSpeed, replacementDate);
if (working.isSelected()) {
airC.turnOn(Room.getElectricityInRoom());
} else {
airC.turnOff();
}
if (connectedToPowerSupply.isSelected()) {
airC.connectToPowerSupply();
} else {
airC.disconnectFromPowerSupply();
}
if (airC.isFilterReplacementDateValid(replacementDate)) {
SmartDevice deviceTemp = airC;
deviceTemp.update(Room.getRoomTemperature(), Room.getRoomHumidity(), Room.getRoomAirParticleLevel(), Room.getElectricityInRoom());
deviceTemp.setId(device.getId());
Room.updateDevice(deviceTemp);
ControllerListOfDevices controller = new ControllerListOfDevices();
Scene currentScene = cancel.getScene();
Stage stage = (Stage) currentScene.getWindow();
loggerUser.info("Успешное сохранение настроек кондиционера PK="+device.getId());
controller.start(stage);
} else {
loggerUser.warn("Введены неверные данные");
errorMessage.setText("Дата замены фильтров не должна превышать сегодняшнюю дату!");
}
}
public void exportButtonClicked(ActionEvent actionEvent) {
String fileNameValue = fileName.getText();
switch (FileNameChecker.getTypeOfFile(fileNameValue)) {
case(1): { // .bat
fileMessage.setText("");
File file = new File(fileNameValue);
if (BinaryFileManager.writeIntoBAT(device, fileNameValue)) {
fileMessage.setText("Файл успешно записан!");
loggerUser.info("Успешное экспортирование настроек кондиционера PK="+device.getId());
} else {
loggerUser.error("Ошибка при экспортировании");
fileMessage.setText("Произошла ошибка!");
}
break;
}
case(2):{ // .json
fileMessage.setText("");
File file = new File(fileNameValue);
if (JsonFileManager.writeIntoJSON(device, fileNameValue)) {
fileMessage.setText("Файл успешно записан!");
loggerUser.info("Успешное экспортирование настроек кондиционера PK="+device.getId());
} else {
fileMessage.setText("Произошла ошибка!");
loggerUser.error("Ошибка при экспортировании");
}
break;
}
default: {
fileMessage.setText("Имя файла не \*.bat или \*.json");
}
}
}
}

ControllerConfigureAirPurifier.java

package org.example.demo.sceneControllers;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.Scene;
import javafx.scene.control.\*;
import javafx.stage.Stage;
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;
import org.example.demo.Room;
import org.example.demo.devices.AirPurifier;
import org.example.demo.devices.SmartDevice;
import org.example.demo.fileManagers.BinaryFileManager;
import org.example.demo.fileManagers.FileNameChecker;
import org.example.demo.fileManagers.JsonFileManager;
import java.io.File;
import java.time.LocalDate;
import java.util.HashMap;
public class ControllerConfigureAirPurifier {
public CheckBox connectedToPowerSupply;
public CheckBox working;
public TextField fileName;
public Button exportButton;
public Label fileMessage;
@FXML
private ComboBox<String> speedOfFan;
private ToggleGroup modeHumidityGroup;
@FXML
private RadioButton modeHumidity0;
@FXML
private RadioButton modeHumidity1;
@FXML
private RadioButton modeHumidity2;
@FXML
private CheckBox modeAuto;
@FXML
private DatePicker filterReplacementDate;
@FXML
private CheckBox modeAirParticle;
@FXML
private Label errorMessage;
@FXML
private Label manufacture;
@FXML
private Label model;
@FXML
private Label powerSupply;
@FXML
private Button cancel;
@FXML
private Button save;
private SmartDevice device;
private static final Logger loggerUser = LogManager.getLogger("FileUserLogger");
public void start(Stage stage, SmartDevice \_device) {
loggerUser.info("Настройка очистителя воздуха PK="+\_device.getId());
device = \_device;
manufacture.setText(device.getManufacturer());
model.setText(device.getModel());
powerSupply.setText(device.getPowerSupply());
HashMap<String, Object> configurations = device.getConfigurations();
speedOfFan.getItems().addAll("высокая", "средняя", "низкая", "автоматическая");
speedOfFan.setValue((String) configurations.get("Скорость вентиляторов"));
filterReplacementDate.setValue((LocalDate) configurations.get("Дата замены фильтров"));
modeHumidityGroup = new ToggleGroup();
modeHumidity0.setToggleGroup(modeHumidityGroup);
modeHumidity1.setToggleGroup(modeHumidityGroup);
modeHumidity2.setToggleGroup(modeHumidityGroup);
switch ((String) configurations.get("Режим регуляции влажности")) {
case ("увлажнение"): {
modeHumidity0.setSelected(true);
break;
}
case ("осушение"):{
modeHumidity1.setSelected(true);
break;
}
case ("отключен"): {
modeHumidity2.setSelected(true);
}
}
modeAirParticle.setSelected((boolean) configurations.get("Режим регуляции уровня частиц в воздухе"));
modeAuto.setSelected((boolean)configurations.get("Автоматический режим"));
connectedToPowerSupply.setSelected((device.getConnectedToPowerSupply()));
working.setSelected((device.getOnOrOff()));
}
@FXML
private void cancelClicked() {
loggerUser.info("Отмена настроек очистителя воздуха");
ControllerListOfDevices controller = new ControllerListOfDevices();
Scene currentScene = cancel.getScene();
Stage stage = (Stage) currentScene.getWindow();
controller.start(stage);
}
public void saveClicked() {
String manufacturerValue = manufacture.getText();
String modelValue = model.getText();
String powerSupplyValue = powerSupply.getText();
String fanSpeed = speedOfFan.getValue();
LocalDate replacementDate = filterReplacementDate.getValue();
String modeHumidity;
boolean isHumidification = modeHumidity0.isSelected();
boolean isDehumidification = modeHumidity1.isSelected();
if (isHumidification) modeHumidity = "увлажнение";
else if (isDehumidification) modeHumidity = "осушение";
else modeHumidity = "отключен";
boolean isAirParticleMode = modeAirParticle.isSelected();
boolean isAutomaticMode = modeAuto.isSelected();
AirPurifier airP = new AirPurifier( manufacturerValue, modelValue, powerSupplyValue, Room.getElectricityInRoom(), modeHumidity, isAirParticleMode, isAutomaticMode, fanSpeed, replacementDate);
if (working.isSelected()) {
airP.turnOn(Room.getElectricityInRoom());
} else {
airP.turnOff();
}
if (connectedToPowerSupply.isSelected()) {
airP.connectToPowerSupply();
} else {
airP.disconnectFromPowerSupply();
}
if (airP.isFilterReplacementDateValid(replacementDate)) {
SmartDevice deviceTemp = airP;
deviceTemp.update(Room.getRoomTemperature(), Room.getRoomHumidity(), Room.getRoomAirParticleLevel(), Room.getElectricityInRoom());
deviceTemp.setId(device.getId());
Room.updateDevice(deviceTemp);
ControllerListOfDevices controller = new ControllerListOfDevices();
Scene currentScene = cancel.getScene();
Stage stage = (Stage) currentScene.getWindow();
loggerUser.info("Успешное сохранение настроек очистителя воздуаха PK="+device.getId());
controller.start(stage);
} else {
loggerUser.warn("Введены неверные данные");
errorMessage.setText("Дата замены фильтров не должна превышать сегодняшнюю дату!");
}
}
public void exportButtonClicked(ActionEvent actionEvent) {
String fileNameValue = fileName.getText();
switch (FileNameChecker.getTypeOfFile(fileNameValue)) {
case(1): { // .bat
fileMessage.setText("");
File file = new File(fileNameValue);
if (BinaryFileManager.writeIntoBAT(device, fileNameValue)) {
fileMessage.setText("Файл успешно записан!");
loggerUser.info("Успешное экспортирование настроек очистителя воздуха PK="+device.getId());
} else {
fileMessage.setText("Произошла ошибка!");
loggerUser.error("Ошибка при экспортировании");
}
break;
}
case(2):{ // .json
fileMessage.setText("");
File file = new File(fileNameValue);
if (JsonFileManager.writeIntoJSON(device, fileNameValue)) {
fileMessage.setText("Файл успешно записан!");
loggerUser.info("Успешное экспортирование настроек очистителя воздуха PK="+device.getId());
} else {
fileMessage.setText("Произошла ошибка!");
loggerUser.error("Ошибка при экспортировании");
}
break;
}
default: {
fileMessage.setText("Имя файла не \*.bat или \*.json");
}
}
}
}

ControllerConfigureHeater.java

ControllerConfigureRoom.java

ControllerListOfDevices.java

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;
}

Main.java

ComputerPeripheral.java

package com.codemaster.computer;
public abstract class ComputerPeripheral {
protected float cost; // цена
protected int yearDevelopment; // дата изготовления (4-х значное число)
private final int countOfLettersInClassNameCP;
protected float minCost;
ComputerPeripheral() {
this.cost = 0.0f;
this.minCost = this.cost;
this.yearDevelopment = 2023;
countOfLettersInClassNameCP = 18;
}
ComputerPeripheral(float newCost, int newYear) {
this.cost = newCost;
this.minCost = this.cost;
this.yearDevelopment = 2023;
countOfLettersInClassNameCP = 18;
}
public void setCost(float newCost) { // установить цену
try {
if (newCost > 0) {
if (minCost > newCost) minCost = newCost;
this.cost = newCost;
}
else throw new Throwable();
} catch (Throwable war) {
System.err.println("Заданная цена не соответсвует формату");
}
}
public float getCost() { // получить цену
return this.cost;
}
public void setYearDevelopment(int newDataDevelopment) { // установить дату изготовления
try {
if (newDataDevelopment > 1900 && newDataDevelopment <= 2025) {
this.yearDevelopment = newDataDevelopment;
} else {
throw new Throwable();
}
} catch (Throwable war) {
System.err.println("Заданный год не соответствует формату");
}
}
public int getYearDevelopment() { // получить дату изготовления
return this.yearDevelopment;
}
public int getCountOfLettersInClassName() {
return this.countOfLettersInClassNameCP;
}
public abstract String getTypeOfPeripheral(); //абстрактный метод для получения типа устройства
public float getMinCost() {
return this.minCost;
}
public abstract void showAllCharacteristics();
}

Monitor.java

package com.codemaster.computer;
public class Monitor extends ComputerPeripheral {
private float size; // собственное свойство
private final int countOfLettersInClassNameM;
public Monitor() {
super();
this.size = 0;
this.countOfLettersInClassNameM = 7;
}
public Monitor(float newCost, int newYear, float newSize) {
super(newCost, newYear);
this.size = newSize;
this.countOfLettersInClassNameM = 7;
}
public void setSize(float newSize) {
if (newSize < 0) newSize \*= -1;
this.size = newSize;
}
public float getSize() {
return this.size;
}
@Override
public int getCountOfLettersInClassName() {
return this.countOfLettersInClassNameM;
}
@Override
public String getTypeOfPeripheral() {
return "Monitor";
}
public void showAllCharacteristics() {
System.out.println(getTypeOfPeripheral() + ": ");
System.out.println("Цена: "+getCost()+" Год: "+getYearDevelopment()+" Размер: "+getSize());
}
}

Printer.java

package com.codemaster.computer;
public class Printer extends ComputerPeripheral {
private String developer = ""; // собственное свойство
private final int countOfLettersInClassNameP;
public Printer() {
super();
this.developer = "";
this.countOfLettersInClassNameP = 7;
}
public Printer(float newCost, int newYear, String newDeveloper) {
super(newCost, newYear);
this.developer = newDeveloper;
this.countOfLettersInClassNameP = 7;
}
public void setDeveloper(String newDeveloper) {
this.developer = newDeveloper;
}
public String getDeveloper() {
return this.developer;
}
@Override
public int getCountOfLettersInClassName() {
return this.countOfLettersInClassNameP;
}
@Override
public String getTypeOfPeripheral() {
return "Printer";
}
public void showAllCharacteristics() {
System.out.println(getTypeOfPeripheral() + ": ");
System.out.println("Цена: "+getCost()+" Год: "+getYearDevelopment()+" Производитель: "+getDeveloper());
}
}

Main.java

package com.codemaster;
import com.codemaster.computer.ComputerPeripheral;
import com.codemaster.menu.Menu;
import java.util.ArrayList;
public class Main {
public static void main(String[] args) {
System.out.println("Лабораторная работа №2\n");
ArrayList<ComputerPeripheral> array = new ArrayList<>();
Menu.showMenu(array);
}
}

Menu.java

Book.java

import java.io.Serializable;
public class Book implements Serializable {
public String author;
public String name;
public int count;
public float cost;
public Book(String \_author, String \_name, int \_count, float \_cost) {
author=\_author;
name=\_name;
count=\_count;
cost=\_cost;
}
}

Main.java

CSVObject.java

import java.io.FileWriter;
import java.io.IOException;
import java.util.Map;
import java.util.TreeMap;
public class CSVObject {
private Map<String, Integer> wordFrequency;
private int totalWords;
private String theMostUseble;
private int numberOfRepTheMostUseble;
private String theLeastUseble;
private int numberOfRepTheLeastUseble;
public CSVObject() {
wordFrequency = new TreeMap<>(); // слово, частота, частота%
theMostUseble = null; // самое частое слово
numberOfRepTheMostUseble = 0; // количество повторений самого частого слова
theLeastUseble = null; // самое редкое слово
numberOfRepTheLeastUseble = 0; // количество повторений самого редкого слова
totalWords = 0; // additional
}
public void putWordFrequency(String word) {
wordFrequency.put(word, wordFrequency.getOrDefault(word, 0) + 1);
this.totalWords+=1;
}
public void setFrequencyOfMostAndLeastUseble() {
for (String word : wordFrequency.keySet()) {
if (theLeastUseble == null || wordFrequency.get(word) < wordFrequency.get(theLeastUseble)) {
theLeastUseble = word;
}
if (theMostUseble == null || wordFrequency.get(word) > wordFrequency.get(theMostUseble)) {
theMostUseble = word;
}
}
}
public void outputToFileCSV(String fileName) {
try (FileWriter writer = new FileWriter(fileName)) {
writer.append("Слово,Частота,Частота (в %),Самое частое слово,Количество повторений,Самое редкое слово,Количество повторений\n");
for (String word : wordFrequency.keySet()) {
writer.append(word); // Слово
writer.append(",");
writer.append(wordFrequency.get(word).toString()); // Частота
writer.append(",");
int proc = wordFrequency.get(word)/totalWords\*100;
writer.append(Integer.toString(proc)); // Частота в %
writer.append(",");
writer.append(theMostUseble); // Самое частое слово
writer.append(",");
writer.append(Integer.toString(numberOfRepTheMostUseble));
writer.append(",");
writer.append(theLeastUseble);
writer.append(",");
writer.append(Integer.toString(numberOfRepTheLeastUseble));
}
} catch (IOException e) {
e.printStackTrace();
}
}
public String getTheMostUseble(){ return this.theMostUseble; }
public int getNumberOfRepTheMostUseble(){ return this.numberOfRepTheMostUseble; }
public String getTheLeastUseble(){ return this.theLeastUseble; }
public int getNumberOfRepTheLeastUseble(){ return this.numberOfRepTheLeastUseble; }
public int getTotalWords() { return this.totalWords; }
}

JsonObject.java

public class JsonObject {
private String fileName;
private int amountOfLetters;
private String theMostCommonLetter;
private int numberOfRepTheMostCommonLetter;
private String rareSymbol;
private int numberOfRepRareSymbol;
private String theMostCommonWord;
private int numberOfRepCommonWord;
private String rareWord;
private int numberOfRepRareWord;
public void setFileName(String \_fileName){ this.fileName = \_fileName; }
public void setAmountOfLetters(int amount){ this.amountOfLetters = amount; }
public void setMostCommonLetter(String letter){ this.theMostCommonLetter = letter; }
public void setNumberOfRepTheMostCommonLetter(int number){ this.numberOfRepTheMostCommonLetter = number; }
public void setRareSymbol(String symbol){ this.rareSymbol = symbol; }
public void setNumberOfRepRareSymbol(int number){ this.numberOfRepRareSymbol = number; }
public void setTheMostCommonWord(String word){ this.theMostCommonWord = word; }
public void setNumberOfRepCommonWord(int number){ this.numberOfRepCommonWord = number; }
public void setRareWord(String word){ this.rareWord = word; }
public void setNumberOfRepRareWord(int number){ this.numberOfRepRareWord = number; }
public String getFileName(){ return this.fileName; }
public int getAmountOfLetters(){ return this.amountOfLetters; }
public String getMostCommonLetter(){ return this.theMostCommonLetter; }
public int getNumberOfRepTheMostCommonLetter(){ return this.numberOfRepTheMostCommonLetter; }
public String getRareSymbol(){ return this.rareSymbol; }
public int getNumberOfRepRareSymbol(){ return this.numberOfRepRareSymbol; }
public String getTheMostCommonWord(){ return this.theMostCommonWord; }
public int getNumberOfRepCommonWord(){ return this.numberOfRepCommonWord; }
public String getRareWord(){ return this.rareWord; }
public int getNumberOfRepRareWord(){ return this.numberOfRepRareWord; }
}

Main.java

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.regex.Pattern;
public class Main {
public static void createCSV(String fileName, int idx) {
CSVObject objCSV = new CSVObject();
// Чтение
try (BufferedReader reader = new BufferedReader(new FileReader(fileName))) {
String line;
Pattern wordPattern = Pattern.compile("[^a-zA-Z0-9]+"); // не буквы, не цифры, от одного и более раза
while ((line = reader.readLine()) != null) {
String[] words = wordPattern.split(line);
for (String word : words) {
if (!word.isEmpty()) {
objCSV.putWordFrequency(word); // подсчёт частоты
}
}
}
} catch (IOException e) {
e.printStackTrace();
return;
}
// Расчёт
objCSV.setFrequencyOfMostAndLeastUseble();
// Запись
String outputFileName = "output" + Integer.toString(idx) + ".csv";
objCSV.outputToFileCSV(outputFileName);
}
public static void createJSON(String fileName) {
}
public static void main(String[] args) {
if (args.length == 0) {
System.out.println("Аргументы не распознаны. Для вызова помощи используйте следующее:");
System.out.println("java lab4 -help");
return;
}
String key = args[0];
switch(key) {
case "-csv": {
for (int i = 1; i < args.length; ++i) {
createCSV(args[i], i);
}
break;
}
case "-json": {
for (int i = 1; i < args.length; ++i) {
createJSON(args[i]);
}
break;
}
case "-help": {
System.out.println("Program abilities:");
System.out.println("-csv\tfilename : Write to CSV file");
System.out.println("-csv\tfilename1 [filename2...]: Write to CSV files for multiple text files");
System.out.println("-json\tfilename : Write to JSON file");
System.out.println("-json\tfilename1 [filename2...]: Write JSON files for multiple text files");
System.out.println("-help\t : Description of launch parameters");
}
}
}
}

AirConditioner.java

AirPurifier.java

package org.example.devices;
import com.google.gson.Gson;
import org.example.Menu;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.text.ParsePosition;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Random;
import java.util.Scanner;
public class AirPurifier extends SmartDevice {
// Additional (4)
private String mode;
private String speedOfFan;
private final int noiseLevel;
private String filterReplacementDate = null;
public AirPurifier(String \_manufacturer, String \_model, String \_powerSupply, String \_macAddress, int \_noiseLevel, String \_mode, String \_speedOfFan, String \_filterReplacementDate) {
super(\_manufacturer, \_model, \_powerSupply, \_macAddress);
type = "AirPurifier";
noiseLevel = \_noiseLevel;
mode = \_mode;
speedOfFan=\_speedOfFan;
filterReplacementDate = \_filterReplacementDate;
}
public String getMode() {
return mode;
}
public String getSpeedOfFan() {
return speedOfFan;
}
public String getFilterReplacementDate() {
return filterReplacementDate;
}
// from SmartDevice
@Override
public void turnOn() {
if (!getOnOrOff()) {
if (getConnectedToPowerSupply()) {
setOnOrOff(true);
System.out.println("Очиститель воздуха " + getMacAddress() + " включен.");
} else {
setOnOrOff(false);
System.out.println("Очиститель воздуха " + getMacAddress() + " не может быть включен, так как отключён от энергопитания.");
}
} else {
if (getConnectedToPowerSupply()) {
setOnOrOff(true);
System.out.println("Очиститель воздуха " + getMacAddress() + " уже включен.");
} else {
setOnOrOff(false);
System.out.println("Очиститель воздуха " + getMacAddress() + " не может быть включен, так как отключён от энергопитания.");
}
}
}
@Override
public void turnOff() {
if (getOnOrOff()) {
setOnOrOff(false);
System.out.println("Очиститель воздуха " + getMacAddress() + " выключен.");
} else {
System.out.println("Очиститель воздуха " + getMacAddress() + " уже выключен.");
}
}
@Override
public void connectToPowerSupply() {
if (getConnectedToPowerSupply()) {
System.out.println("Очиститель воздуха " + getMacAddress() + " уже подключен к электропитанию.");
} else {
setConnectedToPowerSupply(true);
System.out.println("Очиститель воздуха " + getMacAddress() + " подключен к электропитанию.");
}
}
@Override
public void disconnectFromPowerSupply() {
if (getConnectedToPowerSupply()) {
setOnOrOff(false);
setConnectedToPowerSupply(false);
System.out.println("Очиститель воздуха " + getMacAddress() + " отключен от электропитания.");
} else {
setOnOrOff(false);
setConnectedToPowerSupply(false);
System.out.println("Очиститель воздуха " + getMacAddress() + "уже отключен от электропитания.");
}
}
@Override
public void showStatus() {
System.out.println("Состояние очистителя воздуха ID" + id + ":");
if (getOnOrOff()) {
System.out.println("\tВключен.");
System.out.println("\tУровень шума: " + noiseLevel);
System.out.println("\tУстановленный режим: " + mode);
System.out.println("\tСкорость вентилятора: " + speedOfFan);
System.out.println("\tДата последней замены фильтров: " + filterReplacementDate);
showInternetConnectionStatus();
} else {
System.out.println("\tВыключен.");
if (getConnectedToPowerSupply()) System.out.println("\tПодключен к электропитанию.");
else System.out.println("\tОтключен от электропитания.");
System.out.println("\tДата последней замены фильтров: " + filterReplacementDate);
}
}
@Override
public void showInternetConnectionStatus() {
if (getOnOrOff()) {
if (getInternetConnection()) {
Random random = new Random();
int internetStatus = random.nextInt(3) + 1;
System.out.print("\tЕсть доступ к Интернету. Качество соединения: ");
switch (internetStatus) {
case 1: {
System.out.println("плохое.");
break;
}
case 2: {
System.out.println("хорошее.");
break;
}
case 3: {
System.out.println("отличное.");
break;
}
default: {
System.out.println("неопределено.");
break;
}
}
} else {
System.out.println("\tОтсутствует доступ к Интернету.");
}
} else {
System.out.println("Выключен.");
System.out.println("\tОтсутствует доступ к Интернету.");
}
}
@Override
public void configure() {
System.out.println("Выберите настройку: ");
System.out.println("\t0 - Подключить к сети Интернет");
System.out.println("\t1 - Изменить режим");
System.out.println("\t2 - Изменить скорость вентилятора");
System.out.println("\t3 - Заменить фильтр");
switch (Menu.getInt(3)) {
case 0: {
changeInternetConnection();
break;
}
case 1: {
changeMode();
break;
}
case 2: {
changeSpeedOfFan();
break;
}
case 3: {
changeFilterReplacementDate();
break;
}
default: {
}
}
}
@Override
public void showInfo() {
showStatus();
System.out.println("\tТип устройства: очиститель воздуха");
System.out.println("\tПроизводитель: " + getManufacturer());
System.out.println("\tМодель: " + getModel());
System.out.println("\tИсточник электропитания: " + getPowerSupply());
System.out.println("\tMAC-адрес: " + getMacAddress());
}
// Additional (4)
// Режим
private void changeMode() {
mode = Menu.readAirPurifierMode();
System.out.println("Режим успешно изменен!");
}
public void changeMode(String tempMode) {
if (tempMode.equals("обогрев") || tempMode.equals("вентиляция") || tempMode.equals("экономия энергии") || tempMode.equals("работа с влажностью") || tempMode.equals("другой")) {
mode = tempMode;
System.out.println("Режим успешно изменен!");
} else {
System.out.println("Режим не был изменен.");
}
}
// Скорость вентилятора
private void changeSpeedOfFan() {
speedOfFan = Menu.readFanSpeed();
System.out.println("Скорость вентилятора успешно изменена!");
}
public void changeSpeedOfFan(String tempSpeedOfFan) {
if (tempSpeedOfFan.equals("высокая") ||tempSpeedOfFan.equals("средняя") || tempSpeedOfFan.equals("низкая") || tempSpeedOfFan.equals("автоматическая") || tempSpeedOfFan.equals("другая")) {
speedOfFan = tempSpeedOfFan;
System.out.println("Скорость вентилятора успешно изменена!");
} else {
System.out.println("Скорость вентилятора не была изменена.");
}
}
// Заменить фильтры
private void changeFilterReplacementDate() {
System.out.println("Введите новую дату замены фильтра в формате дд.мм.гггг: ");
String scanned;
Scanner scanner = new Scanner(System.in);
SimpleDateFormat dateFormat = new SimpleDateFormat("dd.MM.yyyy");
while(true) {
scanned = scanner.nextLine();
if (Menu.isValidDate(scanned)) {
ParsePosition position1 = new ParsePosition(0);
ParsePosition position2 = new ParsePosition(0);
Date curDate = dateFormat.parse(filterReplacementDate, position1);
Date newDate = dateFormat.parse(scanned, position2);
if (!newDate.before(curDate)) {
filterReplacementDate = scanned;
System.out.println("Дата замены фильтра успешно изменена!");
return;
} else {
System.out.println("Попробуйте ввести ещё раз. Введённая дата меньше установленной ("+filterReplacementDate+").");
}
}
else {
System.out.println("Попробуйте ввести ещё раз.");
}
}
}
public void changeFilterReplacementDate(String tempDate) {
SimpleDateFormat dateFormat = new SimpleDateFormat("dd.MM.yyyy");
if (Menu.isValidDate(tempDate)) {
ParsePosition position1 = new ParsePosition(0);
ParsePosition position2 = new ParsePosition(0);
Date curDate = dateFormat.parse(filterReplacementDate, position1);
Date newDate = dateFormat.parse(tempDate, position2);
if (!newDate.before(curDate)) {
filterReplacementDate = tempDate;
System.out.println("Дата замены фильтра успешно изменена!");
} else {
System.out.println("Попробуйте ввести ещё раз. Введённая дата меньше установленной ("+filterReplacementDate+").");
}
}
else {
System.out.println("Попробуйте ввести ещё раз.");
}
}
private void changeInternetConnection(){
if (getOnOrOff()) {
if (getInternetConnection()) {
Random random = new Random();
int internetStatus = random.nextInt(3) + 1;
System.out.print("\tЕсть доступ к Интернету. Качество соединения: ");
switch (internetStatus) {
case 1: {
System.out.println("плохое.");
break;
}
case 2: {
System.out.println("хорошее.");
break;
}
case 3: {
System.out.println("отличное.");
break;
}
default: {
System.out.println("неопределено.");
break;
}
}
} else {
String scanned = null;
Scanner scanner = new Scanner(System.in);
System.out.println("Введите пароль для доступа в Интернет (для выхода введите пустую строку)");
while (scanned == null || !scanned.isEmpty()) {
scanned = scanner.nextLine();
if (connectToInternetMenu(scanned)) {
System.out.println("Интернет соединение успешно установлено!");
return;
} else {
System.out.println("Неправильный пароль. Попробуйте ещё раз.");
}
}
}
} else {
System.out.println("Включите устройство прежде чем проверить доступ в Интернет.");
}
}
@Override
public void saveToJSON(File file) {
Gson gson = new Gson();
String json = gson.toJson(this);
// Сохраняем JSON в файл
try (FileWriter writer = new FileWriter(file)) {
writer.write(json);
} catch (IOException e) {
e.printStackTrace();
}
}
}

Heater.java

SmartDevice.java

package org.example.devices;
import java.io.File;
import java.util.Scanner;
abstract public class SmartDevice {
protected String type;
public String getType(){
return type;
}
private String manufacturer;
private String model;
private String powerSupply;
private boolean onOrOff; // True if ON
private boolean connectedToPowerSupply; //battery or powerSupplyNetwork
// Additional
private static final String password = "1234";
private boolean internetConnection;
private String macAddress; // device identifier
protected int id;
static protected int maxId = 0;
SmartDevice(String \_manufacturer, String \_model, String \_powerSupply, String \_macAddress) {
manufacturer = \_manufacturer;
model = \_model;
powerSupply = \_powerSupply;
macAddress = \_macAddress;
onOrOff = true;
connectedToPowerSupply = true;
internetConnection = false;
id = maxId+1;
maxId = id;
}
public String getManufacturer() {
return manufacturer;
}
public String getModel() {
return model;
}
public String getPowerSupply(){
return powerSupply;
}
protected void setOnOrOff(boolean var){
onOrOff = var;
}
public boolean getOnOrOff(){
return onOrOff;
}
protected void setConnectedToPowerSupply(boolean var) {
connectedToPowerSupply = var;
}
public boolean getConnectedToPowerSupply(){
return connectedToPowerSupply;
}
public String getMacAddress() {
return macAddress;
}
public boolean getInternetConnection() {
return internetConnection;
}
public int getId() {return id;}
public void rebootId() {
this.id = ++maxId;
}
abstract public void turnOn();
abstract public void turnOff();
abstract public void connectToPowerSupply();
abstract public void disconnectFromPowerSupply();
abstract public void showStatus();
// Additional
abstract public void showInternetConnectionStatus();
abstract public void configure();
abstract public void showInfo();
public static boolean checkPassword(String \_password) {
return password.equals(\_password);
}
protected boolean connectToInternetMenu(String \_password){
if (checkPassword(\_password)) {
internetConnection = true;
return true;
} else return false;
}
abstract public void saveToJSON(File file);
}

Main.java

package org.example;
public class Main {
public static void main(String[] args) {
System.out.println("Управление умными устройствами");
Menu.showMainMenu();
}
}

Menu.java

AirConditionerTest.java

package org.example.tests;
import org.example.devices.AirConditioner;
import static org.junit.jupiter.api.Assertions.\*;
public class AirConditionerTest {
@org.junit.jupiter.api.Test
public void turnOnOffTest() {
AirConditioner airC = new AirConditioner("Vector", "M1234", "сеть", "11:11:11:11:11:11", 25.0f, "Cooling", "высокая", "11.01.2022");
// initial
assertTrue(airC.getOnOrOff());
// turn ON + turn ON
airC.turnOn();
airC.turnOn();
assertTrue(airC.getOnOrOff());
// turn OFF + turn OFF
airC.turnOff();
airC.turnOff();
assertFalse(airC.getOnOrOff());
// turn ON after turn OFF
airC.turnOn();
assertTrue(airC.getOnOrOff());
}
@org.junit.jupiter.api.Test
public void turnOnOffAndPowerSupplyTest() {
AirConditioner airC = new AirConditioner("Vector", "M1234", "сеть", "11:11:11:11:11:11", 25.0f, "Cooling", "высокая", "11.01.2022");
// initial
assertTrue(airC.getConnectedToPowerSupply());
assertTrue(airC.getOnOrOff());
// powerSupply OFF
airC.disconnectFromPowerSupply();
assertFalse(airC.getConnectedToPowerSupply());
assertFalse(airC.getOnOrOff());
// try to turn ON
airC.turnOn();
assertFalse(airC.getOnOrOff());
// powerSupply ON
airC.connectToPowerSupply();
assertTrue(airC.getConnectedToPowerSupply());
assertFalse(airC.getOnOrOff());
// try to turn ON
airC.turnOn();
assertTrue(airC.getOnOrOff());
}
@org.junit.jupiter.api.Test
public void changeTemperatureTest() {
AirConditioner airC = new AirConditioner("Vector", "M1234", "сеть", "11:11:11:11:11:11", 25.0f, "Cooling", "высокая", "11.01.2022");
//init (25.0)
assertEquals(25.0f, airC.getTemperature());
// temperature 15.0
airC.changeTemperature(15.0f);
assertEquals(25.0f, airC.getTemperature());
// temperature 20
airC.changeTemperature(20.0f);
assertEquals(20.0f, airC.getTemperature());
// temperature 40
airC.changeTemperature(40.0f);
assertEquals(20.0f, airC.getTemperature());
}
@org.junit.jupiter.api.Test
public void changeModeTest() {
AirConditioner airC = new AirConditioner("Vector", "M1234", "сеть", "11:11:11:11:11:11", 25.0f, "Cooling", "высокая", "11.01.2022");
// init (Cooling)
assertEquals("Cooling", airC.getMode());
// Sun
airC.changeMode("Sun");
assertEquals("Cooling", airC.getMode());
// Auto
airC.changeMode("Auto");
assertEquals("Auto", airC.getMode());
}
@org.junit.jupiter.api.Test
public void changeSpeedOfFanTest() {
AirConditioner airC = new AirConditioner("Vector", "M1234", "сеть", "11:11:11:11:11:11", 25.0f, "Cooling", "высокая", "11.01.2022");
// init
assertEquals("высокая", airC.getSpeedOfFan());
// Cold
airC.changeSpeedOfFan("Cold");
assertEquals("высокая", airC.getSpeedOfFan());
// низкая
airC.changeSpeedOfFan("низкая");
assertEquals("низкая", airC.getSpeedOfFan());
}
@org.junit.jupiter.api.Test
public void changeFilterReplacementDateTest() {
AirConditioner airC = new AirConditioner("Vector", "M1234", "сеть", "11:11:11:11:11:11", 25.0f, "Cooling", "высокая", "11.01.2022");
// init
assertEquals("11.01.2022", airC.getFilterReplacementDate());
// 20.11.2023
airC.changeFilterReplacementDate("20.11.2024");
assertEquals("11.01.2022", airC.getFilterReplacementDate());
// 15.11.2023
airC.changeFilterReplacementDate("15.11.2023");
assertEquals("15.11.2023", airC.getFilterReplacementDate());
// 15 ноября
airC.changeFilterReplacementDate("15 ноября");
assertEquals("15.11.2023", airC.getFilterReplacementDate());
}
}

AirPurifierTest.java

package org.example.tests;
import org.example.devices.AirPurifier;
import static org.junit.jupiter.api.Assertions.\*;
import static org.junit.jupiter.api.Assertions.assertEquals;
public class AirPurifierTest {
@org.junit.jupiter.api.Test
public void turnOnOffTest() {
AirPurifier airP = new AirPurifier("Vector", "M1234", "сеть", "11:11:11:11:11:11", 40, "обогрев", "высокая", "11.01.2022");
// initial
assertTrue(airP.getOnOrOff());
// turn ON + turn ON
airP.turnOn();
airP.turnOn();
assertTrue(airP.getOnOrOff());
// turn OFF + turn OFF
airP.turnOff();
airP.turnOff();
assertFalse(airP.getOnOrOff());
// turn ON after turn OFF
airP.turnOn();
assertTrue(airP.getOnOrOff());
}
@org.junit.jupiter.api.Test
public void turnOnOffAndPowerSupplyTest() {
AirPurifier airP = new AirPurifier("Vector", "M1234", "сеть", "11:11:11:11:11:11", 40, "обогрев", "высокая", "11.01.2022");
// initial
assertTrue(airP.getConnectedToPowerSupply());
assertTrue(airP.getOnOrOff());
// powerSupply OFF
airP.disconnectFromPowerSupply();
assertFalse(airP.getConnectedToPowerSupply());
assertFalse(airP.getOnOrOff());
// try to turn ON
airP.turnOn();
assertFalse(airP.getOnOrOff());
// powerSupply ON
airP.connectToPowerSupply();
assertTrue(airP.getConnectedToPowerSupply());
assertFalse(airP.getOnOrOff());
// try to turn ON
airP.turnOn();
assertTrue(airP.getOnOrOff());
}
@org.junit.jupiter.api.Test
public void changeModeTest() {
AirPurifier airP = new AirPurifier("Vector", "M1234", "сеть", "11:11:11:11:11:11", 40, "обогрев", "высокая", "11.01.2022"); // init (Cooling)
// initial
assertEquals("обогрев", airP.getMode());
// Sun
airP.changeMode("Sun");
assertEquals("обогрев", airP.getMode());
// Auto
airP.changeMode("экономия энергии");
assertEquals("экономия энергии", airP.getMode());
}
@org.junit.jupiter.api.Test
public void changeSpeedOfFanTest() {
AirPurifier airP = new AirPurifier("Vector", "M1234", "сеть", "11:11:11:11:11:11", 40, "обогрев", "высокая", "11.01.2022"); // init
// initial
assertEquals("высокая", airP.getSpeedOfFan());
// Cold
airP.changeSpeedOfFan("Cold");
assertEquals("высокая", airP.getSpeedOfFan());
// низкая
airP.changeSpeedOfFan("низкая");
assertEquals("низкая", airP.getSpeedOfFan());
}
@org.junit.jupiter.api.Test
public void changeFilterReplacementDateTest() {
AirPurifier airP = new AirPurifier("Vector", "M1234", "сеть", "11:11:11:11:11:11", 40, "обогрев", "высокая", "11.01.2022"); // init
assertEquals("11.01.2022", airP.getFilterReplacementDate());
// 20.11.2023
airP.changeFilterReplacementDate("20.11.2024");
assertEquals("11.01.2022", airP.getFilterReplacementDate());
// 15.11.2023
airP.changeFilterReplacementDate("15.11.2023");
assertEquals("15.11.2023", airP.getFilterReplacementDate());
// 15 ноября
airP.changeFilterReplacementDate("15 ноября");
assertEquals("15.11.2023", airP.getFilterReplacementDate());
}
}

HeaterTest.java

package org.example.tests;
import org.example.devices.Heater;
import static org.junit.jupiter.api.Assertions.\*;
import static org.junit.jupiter.api.Assertions.assertEquals;
public class HeaterTest {
@org.junit.jupiter.api.Test
public void turnOnOffTest() {
Heater heater = new Heater("Vector", "M1234", "сеть", "11:11:11:11:11:11", 25.0f, 30.0f, 15);
// initial
assertTrue(heater.getOnOrOff());
// turn ON + turn ON
heater.turnOn();
heater.turnOn();
assertTrue(heater.getOnOrOff());
// turn OFF + turn OFF
heater.turnOff();
heater.turnOff();
assertFalse(heater.getOnOrOff());
// turn ON after turn OFF
heater.turnOn();
assertTrue(heater.getOnOrOff());
}
@org.junit.jupiter.api.Test
public void turnOnOffAndPowerSupplyTest() {
Heater heater = new Heater("Vector", "M1234", "сеть", "11:11:11:11:11:11", 25.0f, 30.0f, 15.0f);
// initial
assertTrue(heater.getConnectedToPowerSupply());
assertTrue(heater.getOnOrOff());
// powerSupply OFF
heater.disconnectFromPowerSupply();
assertFalse(heater.getConnectedToPowerSupply());
assertFalse(heater.getOnOrOff());
// try to turn ON
heater.turnOn();
assertFalse(heater.getOnOrOff());
// powerSupply ON
heater.connectToPowerSupply();
assertTrue(heater.getConnectedToPowerSupply());
assertFalse(heater.getOnOrOff());
// try to turn ON
heater.turnOn();
assertTrue(heater.getOnOrOff());
}
@org.junit.jupiter.api.Test
public void changeTemperatureTest() {
Heater heater = new Heater("Vector", "M1234", "сеть", "11:11:11:11:11:11", 25.0f, 30.0f, 15.0f);
//init (25.0 30.0)
assertEquals(25.0f, heater.getMinTemperature());
assertEquals(30.0f, heater.getMaxTemperature());
// min temperature 15.0
heater.changeMinTemperature(15.0f);
assertEquals(25.0f, heater.getMinTemperature());
// min temperature 20 max 35
heater.changeMinTemperature(20.0f);
heater.changeMaxTemperature(35.0f);
assertEquals(20.0f, heater.getMinTemperature());
assertEquals(35.0f, heater.getMaxTemperature());
// min temperature 40
heater.changeMinTemperature(40.0f);
assertEquals(20.0f, heater.getMinTemperature());
// max temperature 17
heater.changeMaxTemperature(17.0f);
assertEquals(35.0f, heater.getMaxTemperature());
}
@org.junit.jupiter.api.Test
public void changeAngleTest() {
Heater heater = new Heater("Vector", "M1234", "сеть", "11:11:11:11:11:11", 25.0f, 30.0f, 15.0f);
// init (Cooling)
assertEquals(15.0f, heater.getAngle());
assertTrue(heater.getOnOrOff());
// -10.0
heater.changeAngle(-10.0f);
assertEquals(15.0f, heater.getAngle());
assertTrue(heater.getOnOrOff());
// 10.0
heater.changeAngle(10.0f);
assertEquals(10.0f, heater.getAngle());
assertTrue(heater.getOnOrOff());
// 91.0
heater.changeAngle(91.0f);
assertEquals(10.0f, heater.getAngle());
assertTrue(heater.getOnOrOff());
// 50.0
heater.changeAngle(50.0f);
assertEquals(50.0f, heater.getAngle());
assertFalse(heater.getOnOrOff());
// turn ON
heater.turnOn();
assertFalse(heater.getOnOrOff());
// 16.0 turn ON
heater.changeAngle(16.0f);
assertEquals(16.0f, heater.getAngle());
assertFalse(heater.getOnOrOff());
heater.turnOn();
assertTrue(heater.getOnOrOff());
}
}

AirConditionerTest.java

import org.example.devices.AirConditioner;
import static org.junit.jupiter.api.Assertions.\*;
public class AirConditionerTest {
@org.junit.jupiter.api.Test
public void turnOnOffTest() {
AirConditioner airC = new AirConditioner("Vector", "M1234", "сеть", "11:11:11:11:11:11", 25.0f, "Cooling", "высокая", "11.01.2022");
// initial
assertTrue(airC.getOnOrOff());
// turn ON + turn ON
airC.turnOn();
airC.turnOn();
assertTrue(airC.getOnOrOff());
// turn OFF + turn OFF
airC.turnOff();
assertFalse(airC.getOnOrOff());
// turn ON after turn OFF
airC.turnOn();
assertTrue(airC.getOnOrOff());
}
@org.junit.jupiter.api.Test
public void turnOnOffAndPowerSupplyTest() {
AirConditioner airC = new AirConditioner("Vector", "M1234", "сеть", "11:11:11:11:11:11", 25.0f, "Cooling", "высокая", "11.01.2022");
// initial
assertTrue(airC.getConnectedToPowerSupply());
assertTrue(airC.getOnOrOff());
// powerSupply OFF
airC.disconnectFromPowerSupply();
assertFalse(airC.getConnectedToPowerSupply());
assertFalse(airC.getOnOrOff());
// try to turn ON
airC.turnOn();
assertFalse(airC.getOnOrOff());
// powerSupply ON
airC.connectToPowerSupply();
assertTrue(airC.getConnectedToPowerSupply());
assertFalse(airC.getOnOrOff());
// try to turn ON
airC.turnOn();
assertTrue(airC.getOnOrOff());
}
@org.junit.jupiter.api.Test
public void changeTemperatureTest() {
AirConditioner airC = new AirConditioner("Vector", "M1234", "сеть", "11:11:11:11:11:11", 25.0f, "Cooling", "высокая", "11.01.2022");
//init (25.0)
assertEquals(25.0f, airC.getTemperature());
// temperature 15.0
airC.changeTemperature(15.0f);
assertEquals(25.0f, airC.getTemperature());
// temperature 20
airC.changeTemperature(20.0f);
assertEquals(20.0f, airC.getTemperature());
// temperature 40
airC.changeTemperature(40.0f);
assertEquals(20.0f, airC.getTemperature());
}
@org.junit.jupiter.api.Test
public void changeModeTest() {
AirConditioner airC = new AirConditioner("Vector", "M1234", "сеть", "11:11:11:11:11:11", 25.0f, "Cooling", "высокая", "11.01.2022");
// init (Cooling)
assertEquals("Cooling", airC.getMode());
// Sun
airC.changeMode("Sun");
assertEquals("Cooling", airC.getMode());
// Auto
airC.changeMode("Auto");
assertEquals("Auto", airC.getMode());
}
@org.junit.jupiter.api.Test
public void changeSpeedOfFanTest() {
AirConditioner airC = new AirConditioner("Vector", "M1234", "сеть", "11:11:11:11:11:11", 25.0f, "Cooling", "высокая", "11.01.2022");
// init
assertEquals("высокая", airC.getSpeedOfFan());
// Cold
airC.changeSpeedOfFan("Cold");
assertEquals("высокая", airC.getSpeedOfFan());
// низкая
airC.changeSpeedOfFan("низкая");
assertEquals("низкая", airC.getSpeedOfFan());
}
@org.junit.jupiter.api.Test
public void changeFilterReplacementDateTest() {
AirConditioner airC = new AirConditioner("Vector", "M1234", "сеть", "11:11:11:11:11:11", 25.0f, "Cooling", "высокая", "11.01.2022");
// init
assertEquals("11.01.2022", airC.getFilterReplacementDate());
// 20.11.2023
airC.changeFilterReplacementDate("20.11.2024");
assertEquals("11.01.2022", airC.getFilterReplacementDate());
// 15.11.2023
airC.changeFilterReplacementDate("15.11.2023");
assertEquals("15.11.2023", airC.getFilterReplacementDate());
// 15 ноября
airC.changeFilterReplacementDate("15 ноября");
assertEquals("15.11.2023", airC.getFilterReplacementDate());
}
}

AirPurifierTest.java

import org.example.devices.AirPurifier;
import static org.junit.jupiter.api.Assertions.\*;
import static org.junit.jupiter.api.Assertions.assertEquals;
public class AirPurifierTest {
@org.junit.jupiter.api.Test
public void turnOnOffTest() {
AirPurifier airP = new AirPurifier("Vector", "M1234", "сеть", "11:11:11:11:11:11", 40, "обогрев", "высокая", "11.01.2022");
// initial
assertTrue(airP.getOnOrOff());
// turn ON + turn ON
airP.turnOn();
airP.turnOn();
assertTrue(airP.getOnOrOff());
// turn OFF + turn OFF
airP.turnOff();
airP.turnOff();
assertFalse(airP.getOnOrOff());
// turn ON after turn OFF
airP.turnOn();
assertTrue(airP.getOnOrOff());
}
@org.junit.jupiter.api.Test
public void turnOnOffAndPowerSupplyTest() {
AirPurifier airP = new AirPurifier("Vector", "M1234", "сеть", "11:11:11:11:11:11", 40, "обогрев", "высокая", "11.01.2022");
// initial
assertTrue(airP.getConnectedToPowerSupply());
assertTrue(airP.getOnOrOff());
// powerSupply OFF
airP.disconnectFromPowerSupply();
assertFalse(airP.getConnectedToPowerSupply());
assertFalse(airP.getOnOrOff());
// try to turn ON
airP.turnOn();
assertFalse(airP.getOnOrOff());
// powerSupply ON
airP.connectToPowerSupply();
assertTrue(airP.getConnectedToPowerSupply());
assertFalse(airP.getOnOrOff());
// try to turn ON
airP.turnOn();
assertTrue(airP.getOnOrOff());
}
@org.junit.jupiter.api.Test
public void changeModeTest() {
AirPurifier airP = new AirPurifier("Vector", "M1234", "сеть", "11:11:11:11:11:11", 40, "обогрев", "высокая", "11.01.2022"); // init (Cooling)
// initial
assertEquals("обогрев", airP.getMode());
// Sun
airP.changeMode("Sun");
assertEquals("обогрев", airP.getMode());
// Auto
airP.changeMode("экономия энергии");
assertEquals("экономия энергии", airP.getMode());
}
@org.junit.jupiter.api.Test
public void changeSpeedOfFanTest() {
AirPurifier airP = new AirPurifier("Vector", "M1234", "сеть", "11:11:11:11:11:11", 40, "обогрев", "высокая", "11.01.2022"); // init
// initial
assertEquals("высокая", airP.getSpeedOfFan());
// Cold
airP.changeSpeedOfFan("Cold");
assertEquals("высокая", airP.getSpeedOfFan());
// низкая
airP.changeSpeedOfFan("низкая");
assertEquals("низкая", airP.getSpeedOfFan());
}
@org.junit.jupiter.api.Test
public void changeFilterReplacementDateTest() {
AirPurifier airP = new AirPurifier("Vector", "M1234", "сеть", "11:11:11:11:11:11", 40, "обогрев", "высокая", "11.01.2022"); // init
assertEquals("11.01.2022", airP.getFilterReplacementDate());
// 20.11.2023
airP.changeFilterReplacementDate("20.11.2024");
assertEquals("11.01.2022", airP.getFilterReplacementDate());
// 15.11.2023
airP.changeFilterReplacementDate("15.11.2023");
assertEquals("15.11.2023", airP.getFilterReplacementDate());
// 15 ноября
airP.changeFilterReplacementDate("15 ноября");
assertEquals("15.11.2023", airP.getFilterReplacementDate());
}
}

HeaterTest.java

import org.example.devices.Heater;
import static org.junit.jupiter.api.Assertions.\*;
import static org.junit.jupiter.api.Assertions.assertEquals;
public class HeaterTest {
@org.junit.jupiter.api.Test
public void turnOnOffTest() {
Heater heater = new Heater("Vector", "M1234", "сеть", "11:11:11:11:11:11", 25.0f, 30.0f, 15);
// initial
assertTrue(heater.getOnOrOff());
// turn ON + turn ON
heater.turnOn();
heater.turnOn();
assertTrue(heater.getOnOrOff());
// turn OFF + turn OFF
heater.turnOff();
heater.turnOff();
assertFalse(heater.getOnOrOff());
// turn ON after turn OFF
heater.turnOn();
assertTrue(heater.getOnOrOff());
}
@org.junit.jupiter.api.Test
public void turnOnOffAndPowerSupplyTest() {
Heater heater = new Heater("Vector", "M1234", "сеть", "11:11:11:11:11:11", 25.0f, 30.0f, 15.0f);
// initial
assertTrue(heater.getConnectedToPowerSupply());
assertTrue(heater.getOnOrOff());
// powerSupply OFF
heater.disconnectFromPowerSupply();
assertFalse(heater.getConnectedToPowerSupply());
assertFalse(heater.getOnOrOff());
// try to turn ON
heater.turnOn();
assertFalse(heater.getOnOrOff());
// powerSupply ON
heater.connectToPowerSupply();
assertTrue(heater.getConnectedToPowerSupply());
assertFalse(heater.getOnOrOff());
// try to turn ON
heater.turnOn();
assertTrue(heater.getOnOrOff());
}
@org.junit.jupiter.api.Test
public void changeTemperatureTest() {
Heater heater = new Heater("Vector", "M1234", "сеть", "11:11:11:11:11:11", 25.0f, 30.0f, 15.0f);
//init (25.0 30.0)
assertEquals(25.0f, heater.getMinTemperature());
assertEquals(30.0f, heater.getMaxTemperature());
// min temperature 15.0
heater.changeMinTemperature(15.0f);
assertEquals(25.0f, heater.getMinTemperature());
// min temperature 20 max 35
heater.changeMinTemperature(20.0f);
heater.changeMaxTemperature(35.0f);
assertEquals(20.0f, heater.getMinTemperature());
assertEquals(35.0f, heater.getMaxTemperature());
// min temperature 40
heater.changeMinTemperature(40.0f);
assertEquals(20.0f, heater.getMinTemperature());
// max temperature 17
heater.changeMaxTemperature(17.0f);
assertEquals(35.0f, heater.getMaxTemperature());
}
@org.junit.jupiter.api.Test
public void changeAngleTest() {
Heater heater = new Heater("Vector", "M1234", "сеть", "11:11:11:11:11:11", 25.0f, 30.0f, 15.0f);
// init (Cooling)
assertEquals(15.0f, heater.getAngle());
assertTrue(heater.getOnOrOff());
// -10.0
heater.changeAngle(-10.0f);
assertEquals(15.0f, heater.getAngle());
assertTrue(heater.getOnOrOff());
// 10.0
heater.changeAngle(10.0f);
assertEquals(10.0f, heater.getAngle());
assertTrue(heater.getOnOrOff());
// 91.0
heater.changeAngle(91.0f);
assertEquals(10.0f, heater.getAngle());
assertTrue(heater.getOnOrOff());
// 50.0
heater.changeAngle(50.0f);
assertEquals(50.0f, heater.getAngle());
assertFalse(heater.getOnOrOff());
// turn ON
heater.turnOn();
assertFalse(heater.getOnOrOff());
// 16.0 turn ON
heater.changeAngle(16.0f);
assertEquals(16.0f, heater.getAngle());
assertFalse(heater.getOnOrOff());
heater.turnOn();
assertTrue(heater.getOnOrOff());
}
}

AirConditioner.java

AirPurifier.java

Heater.java

SmartDevice.java

package org.example.devices;
import com.google.gson.Gson;
import java.io.File;
import java.io.Serializable;
import java.util.HashMap;
import java.util.Scanner;
abstract public class SmartDevice implements Serializable {
private final String type;
private final String manufacturer;
private final String model;
private final String powerSupply;
private boolean onOrOff; // True if ON
private boolean connectedToPowerSupply; //battery or powerSupplyNetwork
// Additional
protected int id; // для таблицы
static protected int maxId = 0;
protected int idRoom; // для таблицы
SmartDevice(String \_type, String \_manufacturer, String \_model, String \_powerSupply, int \_idRoom, boolean \_electricityInRoom) {
type = \_type;
manufacturer = \_manufacturer;
model = \_model;
powerSupply = \_powerSupply;
if (powerSupply.equals("сеть")) {
onOrOff = \_electricityInRoom;
} else onOrOff = true;
connectedToPowerSupply = true;
id = maxId+1;
maxId = id;
idRoom = \_idRoom;
}
public String getType(){
return type;
}
public String getManufacturer() {
return manufacturer;
}
public String getModel() {
return model;
}
public String getPowerSupply(){
return powerSupply;
}
public boolean getOnOrOff() { return onOrOff; }
public boolean getConnectedToPowerSupply(){
return connectedToPowerSupply;
}
public int getId() { return id; }
public int getIdRoom() { return idRoom; }
protected void setOnOrOff(boolean var){
onOrOff = var;
}
protected void setConnectedToPowerSupply(boolean var) {
connectedToPowerSupply = var;
}
abstract public boolean turnOn(boolean electricityInRoom);
abstract public void turnOff();
abstract public void connectToPowerSupply();
abstract public void disconnectFromPowerSupply();
abstract public void update(float \_roomTemperature, float \_roomHumidity, float \_roomAirParticleLevel, boolean \_electricityInRoom);
// Additional
abstract public HashMap<String, Object> getConfigurations();
abstract public void setConfigurations(HashMap<String, Object> configurations);
// abstract public String getContentForJSON();
// abstract public byte[] getContentForBinaryFile();
public void rebootId() {
this.id = ++maxId;
}
}

Main.java

package org.example;
public class Main {
public static void main(String[] args) {
System.out.println("Управление умными устройствами");
Menu.showMainMenu();
}
}

Menu.java

AirConditionerTest.java

//import org.example.devices.AirConditioner;
//
//import static org.junit.jupiter.api.Assertions.\*;
//
//public class AirConditionerTest {
//
// @org.junit.jupiter.api.Test
// public void turnOnOffTest() {
// AirConditioner airC = new AirConditioner("Vector", "M1234", "сеть", "11:11:11:11:11:11", 25.0f, "Cooling", "высокая", "11.01.2022");
// // initial
// assertTrue(airC.getOnOrOff());
// // turn ON + turn ON
// airC.turnOn();
// airC.turnOn();
// assertTrue(airC.getOnOrOff());
// // turn OFF + turn OFF
// airC.turnOff();
// assertFalse(airC.getOnOrOff());
// // turn ON after turn OFF
// airC.turnOn();
// assertTrue(airC.getOnOrOff());
// }
//
// @org.junit.jupiter.api.Test
// public void turnOnOffAndPowerSupplyTest() {
// AirConditioner airC = new AirConditioner("Vector", "M1234", "сеть", "11:11:11:11:11:11", 25.0f, "Cooling", "высокая", "11.01.2022");
// // initial
// assertTrue(airC.getConnectedToPowerSupply());
// assertTrue(airC.getOnOrOff());
// // powerSupply OFF
// airC.disconnectFromPowerSupply();
// assertFalse(airC.getConnectedToPowerSupply());
// assertFalse(airC.getOnOrOff());
// // try to turn ON
// airC.turnOn();
// assertFalse(airC.getOnOrOff());
// // powerSupply ON
// airC.connectToPowerSupply();
// assertTrue(airC.getConnectedToPowerSupply());
// assertFalse(airC.getOnOrOff());
// // try to turn ON
// airC.turnOn();
// assertTrue(airC.getOnOrOff());
// }
//
// @org.junit.jupiter.api.Test
// public void changeTemperatureTest() {
// AirConditioner airC = new AirConditioner("Vector", "M1234", "сеть", "11:11:11:11:11:11", 25.0f, "Cooling", "высокая", "11.01.2022");
// //init (25.0)
// assertEquals(25.0f, airC.getTemperature());
// // temperature 15.0
// airC.changeTemperature(15.0f);
// assertEquals(25.0f, airC.getTemperature());
// // temperature 20
// airC.changeTemperature(20.0f);
// assertEquals(20.0f, airC.getTemperature());
// // temperature 40
// airC.changeTemperature(40.0f);
// assertEquals(20.0f, airC.getTemperature());
// }
//
// @org.junit.jupiter.api.Test
// public void changeModeTest() {
// AirConditioner airC = new AirConditioner("Vector", "M1234", "сеть", "11:11:11:11:11:11", 25.0f, "Cooling", "высокая", "11.01.2022");
// // init (Cooling)
// assertEquals("Cooling", airC.getMode());
// // Sun
// airC.changeMode("Sun");
// assertEquals("Cooling", airC.getMode());
// // Auto
// airC.changeMode("Auto");
// assertEquals("Auto", airC.getMode());
// }
//
// @org.junit.jupiter.api.Test
// public void changeSpeedOfFanTest() {
// AirConditioner airC = new AirConditioner("Vector", "M1234", "сеть", "11:11:11:11:11:11", 25.0f, "Cooling", "высокая", "11.01.2022");
// // init
// assertEquals("высокая", airC.getSpeedOfFan());
// // Cold
// airC.changeSpeedOfFan("Cold");
// assertEquals("высокая", airC.getSpeedOfFan());
// // низкая
// airC.changeSpeedOfFan("низкая");
// assertEquals("низкая", airC.getSpeedOfFan());
// }
//
// @org.junit.jupiter.api.Test
// public void changeFilterReplacementDateTest() {
// AirConditioner airC = new AirConditioner("Vector", "M1234", "сеть", "11:11:11:11:11:11", 25.0f, "Cooling", "высокая", "11.01.2022");
// // init
// assertEquals("11.01.2022", airC.getFilterReplacementDate());
// // 20.11.2023
// airC.changeFilterReplacementDate("20.11.2024");
// assertEquals("11.01.2022", airC.getFilterReplacementDate());
// // 15.11.2023
// airC.changeFilterReplacementDate("15.11.2023");
// assertEquals("15.11.2023", airC.getFilterReplacementDate());
// // 15 ноября
// airC.changeFilterReplacementDate("15 ноября");
// assertEquals("15.11.2023", airC.getFilterReplacementDate());
// }
//}

AirPurifierTest.java

//import org.example.devices.AirPurifier;
//
//import static org.junit.jupiter.api.Assertions.\*;
//import static org.junit.jupiter.api.Assertions.assertEquals;
//
//public class AirPurifierTest {
// @org.junit.jupiter.api.Test
// public void turnOnOffTest() {
// AirPurifier airP = new AirPurifier("Vector", "M1234", "сеть", "11:11:11:11:11:11", 40, "обогрев", "высокая", "11.01.2022");
// // initial
// assertTrue(airP.getOnOrOff());
// // turn ON + turn ON
// airP.turnOn();
// airP.turnOn();
// assertTrue(airP.getOnOrOff());
// // turn OFF + turn OFF
// airP.turnOff();
// airP.turnOff();
// assertFalse(airP.getOnOrOff());
// // turn ON after turn OFF
// airP.turnOn();
// assertTrue(airP.getOnOrOff());
// }
//
// @org.junit.jupiter.api.Test
// public void turnOnOffAndPowerSupplyTest() {
// AirPurifier airP = new AirPurifier("Vector", "M1234", "сеть", "11:11:11:11:11:11", 40, "обогрев", "высокая", "11.01.2022");
// // initial
// assertTrue(airP.getConnectedToPowerSupply());
// assertTrue(airP.getOnOrOff());
// // powerSupply OFF
// airP.disconnectFromPowerSupply();
// assertFalse(airP.getConnectedToPowerSupply());
// assertFalse(airP.getOnOrOff());
// // try to turn ON
// airP.turnOn();
// assertFalse(airP.getOnOrOff());
// // powerSupply ON
// airP.connectToPowerSupply();
// assertTrue(airP.getConnectedToPowerSupply());
// assertFalse(airP.getOnOrOff());
// // try to turn ON
// airP.turnOn();
// assertTrue(airP.getOnOrOff());
// }
//
// @org.junit.jupiter.api.Test
// public void changeModeTest() {
// AirPurifier airP = new AirPurifier("Vector", "M1234", "сеть", "11:11:11:11:11:11", 40, "обогрев", "высокая", "11.01.2022"); // init (Cooling)
// // initial
// assertEquals("обогрев", airP.getMode());
// // Sun
// airP.changeMode("Sun");
// assertEquals("обогрев", airP.getMode());
// // Auto
// airP.changeMode("экономия энергии");
// assertEquals("экономия энергии", airP.getMode());
// }
//
// @org.junit.jupiter.api.Test
// public void changeSpeedOfFanTest() {
// AirPurifier airP = new AirPurifier("Vector", "M1234", "сеть", "11:11:11:11:11:11", 40, "обогрев", "высокая", "11.01.2022"); // init
// // initial
// assertEquals("высокая", airP.getSpeedOfFan());
// // Cold
// airP.changeSpeedOfFan("Cold");
// assertEquals("высокая", airP.getSpeedOfFan());
// // низкая
// airP.changeSpeedOfFan("низкая");
// assertEquals("низкая", airP.getSpeedOfFan());
// }
//
// @org.junit.jupiter.api.Test
// public void changeFilterReplacementDateTest() {
// AirPurifier airP = new AirPurifier("Vector", "M1234", "сеть", "11:11:11:11:11:11", 40, "обогрев", "высокая", "11.01.2022"); // init
// assertEquals("11.01.2022", airP.getFilterReplacementDate());
// // 20.11.2023
// airP.changeFilterReplacementDate("20.11.2024");
// assertEquals("11.01.2022", airP.getFilterReplacementDate());
// // 15.11.2023
// airP.changeFilterReplacementDate("15.11.2023");
// assertEquals("15.11.2023", airP.getFilterReplacementDate());
// // 15 ноября
// airP.changeFilterReplacementDate("15 ноября");
// assertEquals("15.11.2023", airP.getFilterReplacementDate());
// }
//}

HeaterTest.java

//import org.example.devices.Heater;
//
//import static org.junit.jupiter.api.Assertions.\*;
//import static org.junit.jupiter.api.Assertions.assertEquals;
//
//public class HeaterTest {
// @org.junit.jupiter.api.Test
// public void turnOnOffTest() {
// Heater heater = new Heater("Vector", "M1234", "сеть", "11:11:11:11:11:11", 25.0f, 30.0f, 15);
// // initial
// assertTrue(heater.getOnOrOff());
// // turn ON + turn ON
// heater.turnOn();
// heater.turnOn();
// assertTrue(heater.getOnOrOff());
// // turn OFF + turn OFF
// heater.turnOff();
// heater.turnOff();
// assertFalse(heater.getOnOrOff());
// // turn ON after turn OFF
// heater.turnOn();
// assertTrue(heater.getOnOrOff());
// }
//
// @org.junit.jupiter.api.Test
// public void turnOnOffAndPowerSupplyTest() {
// Heater heater = new Heater("Vector", "M1234", "сеть", "11:11:11:11:11:11", 25.0f, 30.0f, 15.0f);
// // initial
// assertTrue(heater.getConnectedToPowerSupply());
// assertTrue(heater.getOnOrOff());
// // powerSupply OFF
// heater.disconnectFromPowerSupply();
// assertFalse(heater.getConnectedToPowerSupply());
// assertFalse(heater.getOnOrOff());
// // try to turn ON
// heater.turnOn();
// assertFalse(heater.getOnOrOff());
// // powerSupply ON
// heater.connectToPowerSupply();
// assertTrue(heater.getConnectedToPowerSupply());
// assertFalse(heater.getOnOrOff());
// // try to turn ON
// heater.turnOn();
// assertTrue(heater.getOnOrOff());
// }
//
// @org.junit.jupiter.api.Test
// public void changeTemperatureTest() {
// Heater heater = new Heater("Vector", "M1234", "сеть", "11:11:11:11:11:11", 25.0f, 30.0f, 15.0f);
// //init (25.0 30.0)
// assertEquals(25.0f, heater.getMinTemperature());
// assertEquals(30.0f, heater.getMaxTemperature());
// // min temperature 15.0
// heater.changeMinTemperature(15.0f);
// assertEquals(25.0f, heater.getMinTemperature());
// // min temperature 20 max 35
// heater.changeMinTemperature(20.0f);
// heater.changeMaxTemperature(35.0f);
// assertEquals(20.0f, heater.getMinTemperature());
// assertEquals(35.0f, heater.getMaxTemperature());
// // min temperature 40
// heater.changeMinTemperature(40.0f);
// assertEquals(20.0f, heater.getMinTemperature());
// // max temperature 17
// heater.changeMaxTemperature(17.0f);
// assertEquals(35.0f, heater.getMaxTemperature());
//
// }
//
// @org.junit.jupiter.api.Test
// public void changeAngleTest() {
// Heater heater = new Heater("Vector", "M1234", "сеть", "11:11:11:11:11:11", 25.0f, 30.0f, 15.0f);
// // init (Cooling)
// assertEquals(15.0f, heater.getAngle());
// assertTrue(heater.getOnOrOff());
// // -10.0
// heater.changeAngle(-10.0f);
// assertEquals(15.0f, heater.getAngle());
// assertTrue(heater.getOnOrOff());
// // 10.0
// heater.changeAngle(10.0f);
// assertEquals(10.0f, heater.getAngle());
// assertTrue(heater.getOnOrOff());
// // 91.0
// heater.changeAngle(91.0f);
// assertEquals(10.0f, heater.getAngle());
// assertTrue(heater.getOnOrOff());
// // 50.0
// heater.changeAngle(50.0f);
// assertEquals(50.0f, heater.getAngle());
// assertFalse(heater.getOnOrOff());
// // turn ON
// heater.turnOn();
// assertFalse(heater.getOnOrOff());
// // 16.0 turn ON
// heater.changeAngle(16.0f);
// assertEquals(16.0f, heater.getAngle());
// assertFalse(heater.getOnOrOff());
// heater.turnOn();
// assertTrue(heater.getOnOrOff());
// }
//
//}

AirConditioner.java

AirPurifier.java

package org.example.devices;
import com.google.gson.Gson;
import org.example.Menu;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.text.ParsePosition;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Random;
import java.util.Scanner;
public class AirPurifier extends SmartDevice {
// Additional (4)
private String mode;
private String speedOfFan;
private final int noiseLevel;
private String filterReplacementDate = null;
public AirPurifier(String \_manufacturer, String \_model, String \_powerSupply, String \_macAddress, int \_noiseLevel, String \_mode, String \_speedOfFan, String \_filterReplacementDate) {
super(\_manufacturer, \_model, \_powerSupply, \_macAddress);
type = "AirPurifier";
noiseLevel = \_noiseLevel;
mode = \_mode;
speedOfFan=\_speedOfFan;
filterReplacementDate = \_filterReplacementDate;
}
public String getMode() {
return mode;
}
public String getSpeedOfFan() {
return speedOfFan;
}
public String getFilterReplacementDate() {
return filterReplacementDate;
}
// from SmartDevice
@Override
public void turnOn() {
if (!getOnOrOff()) {
if (getConnectedToPowerSupply()) {
setOnOrOff(true);
System.out.println("Очиститель воздуха " + getMacAddress() + " включен.");
} else {
setOnOrOff(false);
System.out.println("Очиститель воздуха " + getMacAddress() + " не может быть включен, так как отключён от энергопитания.");
}
} else {
if (getConnectedToPowerSupply()) {
setOnOrOff(true);
System.out.println("Очиститель воздуха " + getMacAddress() + " уже включен.");
} else {
setOnOrOff(false);
System.out.println("Очиститель воздуха " + getMacAddress() + " не может быть включен, так как отключён от энергопитания.");
}
}
}
@Override
public void turnOff() {
if (getOnOrOff()) {
setOnOrOff(false);
System.out.println("Очиститель воздуха " + getMacAddress() + " выключен.");
} else {
System.out.println("Очиститель воздуха " + getMacAddress() + " уже выключен.");
}
}
@Override
public void connectToPowerSupply() {
if (getConnectedToPowerSupply()) {
System.out.println("Очиститель воздуха " + getMacAddress() + " уже подключен к электропитанию.");
} else {
setConnectedToPowerSupply(true);
System.out.println("Очиститель воздуха " + getMacAddress() + " подключен к электропитанию.");
}
}
@Override
public void disconnectFromPowerSupply() {
if (getConnectedToPowerSupply()) {
setOnOrOff(false);
setConnectedToPowerSupply(false);
System.out.println("Очиститель воздуха " + getMacAddress() + " отключен от электропитания.");
} else {
setOnOrOff(false);
setConnectedToPowerSupply(false);
System.out.println("Очиститель воздуха " + getMacAddress() + "уже отключен от электропитания.");
}
}
@Override
public void showStatus() {
System.out.println("Состояние очистителя воздуха ID" + id + ":");
if (getOnOrOff()) {
System.out.println("\tВключен.");
System.out.println("\tУровень шума: " + noiseLevel);
System.out.println("\tУстановленный режим: " + mode);
System.out.println("\tСкорость вентилятора: " + speedOfFan);
System.out.println("\tДата последней замены фильтров: " + filterReplacementDate);
showInternetConnectionStatus();
} else {
System.out.println("\tВыключен.");
if (getConnectedToPowerSupply()) System.out.println("\tПодключен к электропитанию.");
else System.out.println("\tОтключен от электропитания.");
System.out.println("\tДата последней замены фильтров: " + filterReplacementDate);
}
}
@Override
public void showInternetConnectionStatus() {
if (getOnOrOff()) {
if (getInternetConnection()) {
Random random = new Random();
int internetStatus = random.nextInt(3) + 1;
System.out.print("\tЕсть доступ к Интернету. Качество соединения: ");
switch (internetStatus) {
case 1: {
System.out.println("плохое.");
break;
}
case 2: {
System.out.println("хорошее.");
break;
}
case 3: {
System.out.println("отличное.");
break;
}
default: {
System.out.println("неопределено.");
break;
}
}
} else {
System.out.println("\tОтсутствует доступ к Интернету.");
}
} else {
System.out.println("Выключен.");
System.out.println("\tОтсутствует доступ к Интернету.");
}
}
@Override
public void configure() {
System.out.println("Выберите настройку: ");
System.out.println("\t0 - Подключить к сети Интернет");
System.out.println("\t1 - Изменить режим");
System.out.println("\t2 - Изменить скорость вентилятора");
System.out.println("\t3 - Заменить фильтр");
switch (Menu.getInt(3)) {
case 0: {
changeInternetConnection();
break;
}
case 1: {
changeMode();
break;
}
case 2: {
changeSpeedOfFan();
break;
}
case 3: {
changeFilterReplacementDate();
break;
}
default: {
}
}
}
@Override
public void showInfo() {
showStatus();
System.out.println("\tТип устройства: очиститель воздуха");
System.out.println("\tПроизводитель: " + getManufacturer());
System.out.println("\tМодель: " + getModel());
System.out.println("\tИсточник электропитания: " + getPowerSupply());
System.out.println("\tMAC-адрес: " + getMacAddress());
}
// Additional (4)
// Режим
private void changeMode() {
mode = Menu.readAirPurifierMode();
System.out.println("Режим успешно изменен!");
}
public void changeMode(String tempMode) {
if (tempMode.equals("обогрев") || tempMode.equals("вентиляция") || tempMode.equals("экономия энергии") || tempMode.equals("работа с влажностью") || tempMode.equals("другой")) {
mode = tempMode;
System.out.println("Режим успешно изменен!");
} else {
System.out.println("Режим не был изменен.");
}
}
// Скорость вентилятора
private void changeSpeedOfFan() {
speedOfFan = Menu.readFanSpeed();
System.out.println("Скорость вентилятора успешно изменена!");
}
public void changeSpeedOfFan(String tempSpeedOfFan) {
if (tempSpeedOfFan.equals("высокая") ||tempSpeedOfFan.equals("средняя") || tempSpeedOfFan.equals("низкая") || tempSpeedOfFan.equals("автоматическая") || tempSpeedOfFan.equals("другая")) {
speedOfFan = tempSpeedOfFan;
System.out.println("Скорость вентилятора успешно изменена!");
} else {
System.out.println("Скорость вентилятора не была изменена.");
}
}
// Заменить фильтры
private void changeFilterReplacementDate() {
System.out.println("Введите новую дату замены фильтра в формате дд.мм.гггг: ");
String scanned;
Scanner scanner = new Scanner(System.in);
SimpleDateFormat dateFormat = new SimpleDateFormat("dd.MM.yyyy");
while(true) {
scanned = scanner.nextLine();
if (Menu.isValidDate(scanned)) {
ParsePosition position1 = new ParsePosition(0);
ParsePosition position2 = new ParsePosition(0);
Date curDate = dateFormat.parse(filterReplacementDate, position1);
Date newDate = dateFormat.parse(scanned, position2);
if (!newDate.before(curDate)) {
filterReplacementDate = scanned;
System.out.println("Дата замены фильтра успешно изменена!");
return;
} else {
System.out.println("Попробуйте ввести ещё раз. Введённая дата меньше установленной ("+filterReplacementDate+").");
}
}
else {
System.out.println("Попробуйте ввести ещё раз.");
}
}
}
public void changeFilterReplacementDate(String tempDate) {
SimpleDateFormat dateFormat = new SimpleDateFormat("dd.MM.yyyy");
if (Menu.isValidDate(tempDate)) {
ParsePosition position1 = new ParsePosition(0);
ParsePosition position2 = new ParsePosition(0);
Date curDate = dateFormat.parse(filterReplacementDate, position1);
Date newDate = dateFormat.parse(tempDate, position2);
if (!newDate.before(curDate)) {
filterReplacementDate = tempDate;
System.out.println("Дата замены фильтра успешно изменена!");
} else {
System.out.println("Попробуйте ввести ещё раз. Введённая дата меньше установленной ("+filterReplacementDate+").");
}
}
else {
System.out.println("Попробуйте ввести ещё раз.");
}
}
private void changeInternetConnection(){
if (getOnOrOff()) {
if (getInternetConnection()) {
Random random = new Random();
int internetStatus = random.nextInt(3) + 1;
System.out.print("\tЕсть доступ к Интернету. Качество соединения: ");
switch (internetStatus) {
case 1: {
System.out.println("плохое.");
break;
}
case 2: {
System.out.println("хорошее.");
break;
}
case 3: {
System.out.println("отличное.");
break;
}
default: {
System.out.println("неопределено.");
break;
}
}
} else {
String scanned = null;
Scanner scanner = new Scanner(System.in);
System.out.println("Введите пароль для доступа в Интернет (для выхода введите пустую строку)");
while (scanned == null || !scanned.isEmpty()) {
scanned = scanner.nextLine();
if (connectToInternetMenu(scanned)) {
System.out.println("Интернет соединение успешно установлено!");
return;
} else {
System.out.println("Неправильный пароль. Попробуйте ещё раз.");
}
}
}
} else {
System.out.println("Включите устройство прежде чем проверить доступ в Интернет.");
}
}
@Override
public void saveToJSON(File file) {
Gson gson = new Gson();
String json = gson.toJson(this);
// Сохраняем JSON в файл
try (FileWriter writer = new FileWriter(file)) {
writer.write(json);
} catch (IOException e) {
e.printStackTrace();
}
}
}

Heater.java

SmartDevice.java

package org.example.devices;
import java.io.File;
import java.util.Scanner;
abstract public class SmartDevice {
protected String type;
public String getType(){
return type;
}
private String manufacturer;
private String model;
private String powerSupply;
private boolean onOrOff; // True if ON
private boolean connectedToPowerSupply; //battery or powerSupplyNetwork
// Additional
private static final String password = "1234";
private boolean internetConnection;
private String macAddress; // device identifier
protected int id;
static protected int maxId = 0;
SmartDevice(String \_manufacturer, String \_model, String \_powerSupply, String \_macAddress) {
manufacturer = \_manufacturer;
model = \_model;
powerSupply = \_powerSupply;
macAddress = \_macAddress;
onOrOff = true;
connectedToPowerSupply = true;
internetConnection = false;
id = maxId+1;
maxId = id;
}
public String getManufacturer() {
return manufacturer;
}
public String getModel() {
return model;
}
public String getPowerSupply(){
return powerSupply;
}
protected void setOnOrOff(boolean var){
onOrOff = var;
}
public boolean getOnOrOff(){
return onOrOff;
}
protected void setConnectedToPowerSupply(boolean var) {
connectedToPowerSupply = var;
}
public boolean getConnectedToPowerSupply(){
return connectedToPowerSupply;
}
public String getMacAddress() {
return macAddress;
}
public boolean getInternetConnection() {
return internetConnection;
}
public int getId() {return id;}
public void rebootId() {
this.id = ++maxId;
}
abstract public void turnOn();
abstract public void turnOff();
abstract public void connectToPowerSupply();
abstract public void disconnectFromPowerSupply();
abstract public void showStatus();
// Additional
abstract public void showInternetConnectionStatus();
abstract public void configure();
abstract public void showInfo();
public static boolean checkPassword(String \_password) {
return password.equals(\_password);
}
protected boolean connectToInternetMenu(String \_password){
if (checkPassword(\_password)) {
internetConnection = true;
return true;
} else return false;
}
abstract public void saveToJSON(File file);
}

Main.java

package org.example;
public class Main {
public static void main(String[] args) {
System.out.println("Управление умными устройствами");
Menu.showMainMenu();
}
}

Menu.java

AirConditionerTest.java

import org.example.devices.AirConditioner;
import static org.junit.jupiter.api.Assertions.\*;
public class AirConditionerTest {
@org.junit.jupiter.api.Test
public void turnOnOffTest() {
AirConditioner airC = new AirConditioner("Vector", "M1234", "сеть", "11:11:11:11:11:11", 25.0f, "Cooling", "высокая", "11.01.2022");
// initial
assertTrue(airC.getOnOrOff());
// turn ON + turn ON
airC.turnOn();
airC.turnOn();
assertTrue(airC.getOnOrOff());
// turn OFF + turn OFF
airC.turnOff();
assertFalse(airC.getOnOrOff());
// turn ON after turn OFF
airC.turnOn();
assertTrue(airC.getOnOrOff());
}
@org.junit.jupiter.api.Test
public void turnOnOffAndPowerSupplyTest() {
AirConditioner airC = new AirConditioner("Vector", "M1234", "сеть", "11:11:11:11:11:11", 25.0f, "Cooling", "высокая", "11.01.2022");
// initial
assertTrue(airC.getConnectedToPowerSupply());
assertTrue(airC.getOnOrOff());
// powerSupply OFF
airC.disconnectFromPowerSupply();
assertFalse(airC.getConnectedToPowerSupply());
assertFalse(airC.getOnOrOff());
// try to turn ON
airC.turnOn();
assertFalse(airC.getOnOrOff());
// powerSupply ON
airC.connectToPowerSupply();
assertTrue(airC.getConnectedToPowerSupply());
assertFalse(airC.getOnOrOff());
// try to turn ON
airC.turnOn();
assertTrue(airC.getOnOrOff());
}
@org.junit.jupiter.api.Test
public void changeTemperatureTest() {
AirConditioner airC = new AirConditioner("Vector", "M1234", "сеть", "11:11:11:11:11:11", 25.0f, "Cooling", "высокая", "11.01.2022");
//init (25.0)
assertEquals(25.0f, airC.getTemperature());
// temperature 15.0
airC.changeTemperature(15.0f);
assertEquals(25.0f, airC.getTemperature());
// temperature 20
airC.changeTemperature(20.0f);
assertEquals(20.0f, airC.getTemperature());
// temperature 40
airC.changeTemperature(40.0f);
assertEquals(20.0f, airC.getTemperature());
}
@org.junit.jupiter.api.Test
public void changeModeTest() {
AirConditioner airC = new AirConditioner("Vector", "M1234", "сеть", "11:11:11:11:11:11", 25.0f, "Cooling", "высокая", "11.01.2022");
// init (Cooling)
assertEquals("Cooling", airC.getMode());
// Sun
airC.changeMode("Sun");
assertEquals("Cooling", airC.getMode());
// Auto
airC.changeMode("Auto");
assertEquals("Auto", airC.getMode());
}
@org.junit.jupiter.api.Test
public void changeSpeedOfFanTest() {
AirConditioner airC = new AirConditioner("Vector", "M1234", "сеть", "11:11:11:11:11:11", 25.0f, "Cooling", "высокая", "11.01.2022");
// init
assertEquals("высокая", airC.getSpeedOfFan());
// Cold
airC.changeSpeedOfFan("Cold");
assertEquals("высокая", airC.getSpeedOfFan());
// низкая
airC.changeSpeedOfFan("низкая");
assertEquals("низкая", airC.getSpeedOfFan());
}
@org.junit.jupiter.api.Test
public void changeFilterReplacementDateTest() {
AirConditioner airC = new AirConditioner("Vector", "M1234", "сеть", "11:11:11:11:11:11", 25.0f, "Cooling", "высокая", "11.01.2022");
// init
assertEquals("11.01.2022", airC.getFilterReplacementDate());
// 20.11.2023
airC.changeFilterReplacementDate("20.11.2024");
assertEquals("11.01.2022", airC.getFilterReplacementDate());
// 15.11.2023
airC.changeFilterReplacementDate("15.11.2023");
assertEquals("15.11.2023", airC.getFilterReplacementDate());
// 15 ноября
airC.changeFilterReplacementDate("15 ноября");
assertEquals("15.11.2023", airC.getFilterReplacementDate());
}
}

AirPurifierTest.java

import org.example.devices.AirPurifier;
import static org.junit.jupiter.api.Assertions.\*;
import static org.junit.jupiter.api.Assertions.assertEquals;
public class AirPurifierTest {
@org.junit.jupiter.api.Test
public void turnOnOffTest() {
AirPurifier airP = new AirPurifier("Vector", "M1234", "сеть", "11:11:11:11:11:11", 40, "обогрев", "высокая", "11.01.2022");
// initial
assertTrue(airP.getOnOrOff());
// turn ON + turn ON
airP.turnOn();
airP.turnOn();
assertTrue(airP.getOnOrOff());
// turn OFF + turn OFF
airP.turnOff();
airP.turnOff();
assertFalse(airP.getOnOrOff());
// turn ON after turn OFF
airP.turnOn();
assertTrue(airP.getOnOrOff());
}
@org.junit.jupiter.api.Test
public void turnOnOffAndPowerSupplyTest() {
AirPurifier airP = new AirPurifier("Vector", "M1234", "сеть", "11:11:11:11:11:11", 40, "обогрев", "высокая", "11.01.2022");
// initial
assertTrue(airP.getConnectedToPowerSupply());
assertTrue(airP.getOnOrOff());
// powerSupply OFF
airP.disconnectFromPowerSupply();
assertFalse(airP.getConnectedToPowerSupply());
assertFalse(airP.getOnOrOff());
// try to turn ON
airP.turnOn();
assertFalse(airP.getOnOrOff());
// powerSupply ON
airP.connectToPowerSupply();
assertTrue(airP.getConnectedToPowerSupply());
assertFalse(airP.getOnOrOff());
// try to turn ON
airP.turnOn();
assertTrue(airP.getOnOrOff());
}
@org.junit.jupiter.api.Test
public void changeModeTest() {
AirPurifier airP = new AirPurifier("Vector", "M1234", "сеть", "11:11:11:11:11:11", 40, "обогрев", "высокая", "11.01.2022"); // init (Cooling)
// initial
assertEquals("обогрев", airP.getMode());
// Sun
airP.changeMode("Sun");
assertEquals("обогрев", airP.getMode());
// Auto
airP.changeMode("экономия энергии");
assertEquals("экономия энергии", airP.getMode());
}
@org.junit.jupiter.api.Test
public void changeSpeedOfFanTest() {
AirPurifier airP = new AirPurifier("Vector", "M1234", "сеть", "11:11:11:11:11:11", 40, "обогрев", "высокая", "11.01.2022"); // init
// initial
assertEquals("высокая", airP.getSpeedOfFan());
// Cold
airP.changeSpeedOfFan("Cold");
assertEquals("высокая", airP.getSpeedOfFan());
// низкая
airP.changeSpeedOfFan("низкая");
assertEquals("низкая", airP.getSpeedOfFan());
}
@org.junit.jupiter.api.Test
public void changeFilterReplacementDateTest() {
AirPurifier airP = new AirPurifier("Vector", "M1234", "сеть", "11:11:11:11:11:11", 40, "обогрев", "высокая", "11.01.2022"); // init
assertEquals("11.01.2022", airP.getFilterReplacementDate());
// 20.11.2023
airP.changeFilterReplacementDate("20.11.2024");
assertEquals("11.01.2022", airP.getFilterReplacementDate());
// 15.11.2023
airP.changeFilterReplacementDate("15.11.2023");
assertEquals("15.11.2023", airP.getFilterReplacementDate());
// 15 ноября
airP.changeFilterReplacementDate("15 ноября");
assertEquals("15.11.2023", airP.getFilterReplacementDate());
}
}

HeaterTest.java

import org.example.devices.Heater;
import static org.junit.jupiter.api.Assertions.\*;
import static org.junit.jupiter.api.Assertions.assertEquals;
public class HeaterTest {
@org.junit.jupiter.api.Test
public void turnOnOffTest() {
Heater heater = new Heater("Vector", "M1234", "сеть", "11:11:11:11:11:11", 25.0f, 30.0f, 15);
// initial
assertTrue(heater.getOnOrOff());
// turn ON + turn ON
heater.turnOn();
heater.turnOn();
assertTrue(heater.getOnOrOff());
// turn OFF + turn OFF
heater.turnOff();
heater.turnOff();
assertFalse(heater.getOnOrOff());
// turn ON after turn OFF
heater.turnOn();
assertTrue(heater.getOnOrOff());
}
@org.junit.jupiter.api.Test
public void turnOnOffAndPowerSupplyTest() {
Heater heater = new Heater("Vector", "M1234", "сеть", "11:11:11:11:11:11", 25.0f, 30.0f, 15.0f);
// initial
assertTrue(heater.getConnectedToPowerSupply());
assertTrue(heater.getOnOrOff());
// powerSupply OFF
heater.disconnectFromPowerSupply();
assertFalse(heater.getConnectedToPowerSupply());
assertFalse(heater.getOnOrOff());
// try to turn ON
heater.turnOn();
assertFalse(heater.getOnOrOff());
// powerSupply ON
heater.connectToPowerSupply();
assertTrue(heater.getConnectedToPowerSupply());
assertFalse(heater.getOnOrOff());
// try to turn ON
heater.turnOn();
assertTrue(heater.getOnOrOff());
}
@org.junit.jupiter.api.Test
public void changeTemperatureTest() {
Heater heater = new Heater("Vector", "M1234", "сеть", "11:11:11:11:11:11", 25.0f, 30.0f, 15.0f);
//init (25.0 30.0)
assertEquals(25.0f, heater.getMinTemperature());
assertEquals(30.0f, heater.getMaxTemperature());
// min temperature 15.0
heater.changeMinTemperature(15.0f);
assertEquals(25.0f, heater.getMinTemperature());
// min temperature 20 max 35
heater.changeMinTemperature(20.0f);
heater.changeMaxTemperature(35.0f);
assertEquals(20.0f, heater.getMinTemperature());
assertEquals(35.0f, heater.getMaxTemperature());
// min temperature 40
heater.changeMinTemperature(40.0f);
assertEquals(20.0f, heater.getMinTemperature());
// max temperature 17
heater.changeMaxTemperature(17.0f);
assertEquals(35.0f, heater.getMaxTemperature());
}
@org.junit.jupiter.api.Test
public void changeAngleTest() {
Heater heater = new Heater("Vector", "M1234", "сеть", "11:11:11:11:11:11", 25.0f, 30.0f, 15.0f);
// init (Cooling)
assertEquals(15.0f, heater.getAngle());
assertTrue(heater.getOnOrOff());
// -10.0
heater.changeAngle(-10.0f);
assertEquals(15.0f, heater.getAngle());
assertTrue(heater.getOnOrOff());
// 10.0
heater.changeAngle(10.0f);
assertEquals(10.0f, heater.getAngle());
assertTrue(heater.getOnOrOff());
// 91.0
heater.changeAngle(91.0f);
assertEquals(10.0f, heater.getAngle());
assertTrue(heater.getOnOrOff());
// 50.0
heater.changeAngle(50.0f);
assertEquals(50.0f, heater.getAngle());
assertFalse(heater.getOnOrOff());
// turn ON
heater.turnOn();
assertFalse(heater.getOnOrOff());
// 16.0 turn ON
heater.changeAngle(16.0f);
assertEquals(16.0f, heater.getAngle());
assertFalse(heater.getOnOrOff());
heater.turnOn();
assertTrue(heater.getOnOrOff());
}
}

HelloApplication.java

package com.example.demo;
import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Scene;
import javafx.stage.Stage;
import java.io.IOException;
public class HelloApplication extends Application {
@Override
public void start(Stage stage) throws IOException {
FXMLLoader fxmlLoader = new FXMLLoader(HelloApplication.class.getResource("hello-view.fxml"));
Scene scene = new Scene(fxmlLoader.load(), 320, 240);
stage.setTitle("Hello!");
stage.setScene(scene);
stage.show();
}
public static void main(String[] args) {
launch();
}
}

HelloController.java

package com.example.demo;
import javafx.fxml.FXML;
import javafx.scene.control.Label;
public class HelloController {
@FXML
private Label welcomeText;
@FXML
protected void onHelloButtonClick() {
welcomeText.setText("Welcome to JavaFX Application!");
}
}

module-info.java

module com.example.demo {
requires javafx.controls;
requires javafx.fxml;
opens com.example.demo to javafx.fxml;
exports com.example.demo;
}