Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №3

з дисципліни «Технології розроблення програмного забезпечення»

Тема: «Основи проектування розгортання»

Варіант: E-mail клієнт

Виконав:
студент групи ІА-32
Карповець Роман

Київ - 2025

Тема: Основи проектування розгортання.

Мета: Навчитися проєктувати діаграми розгортання та компонентів для системи що проєктується, а також розробляти діаграми взаємодії, а саме діаграми послідовностей, на основі сценаріїв зроблених в попередній лабораторній роботі.

Тема проєкту: E-mail клієнт

Опис: Поштовий клієнт повинен нагадувати функціонал поштових програм Mozilla Thunderbird, The Bat і т.д. Він повинен сприймати і коректно обробляти pop3/smtp/imap протоколи, мати функції автонастройки основних поштових провайдерів для України (gmail, ukr.net, i.ua), розділяти повідомлення на папки/категорії/важливість, зберігати чернетки незавершених повідомлень, прикріплювати і обробляти прикріплені файли.

Зміст

Теоретичні відомості 3
Завдання. 4
Хід роботи. 5
Висновок. 11

Теоретичні відомості

Діаграми розгортання представляють фізичне розташування системи, показуючи, на якому фізичному обладнанні запускається та чи інша складова програмного забезпечення.

Головними елементами діаграми є вузли, пов'язані інформаційними шляхами. Вузол (node) – це те, що може містити програмне забезпечення. Вузли бувають двох типів. Пристрій (device) – це фізичне обладнання: комп'ютер або пристрій, пов'язаний із системою. Середовище виконання (execution environment) – це програмне забезпечення, яке саме може включати інше програмне забезпечення, наприклад операційну систему або процес-контейнер (наприклад, вебсервер).

Діаграма компонентів UML є представленням проєктованої системи, розбитої на окремі модулі [3]. Залежно від способу поділу на модулі розрізняють три види діаграм компонентів: логічні; фізичні; виконувані.

Коли використовують логічне розбиття на компоненти, то у такому разі проєктовану систему віртуально уявляють як набір самостійних, автономних модулів (компонентів), що взаємодіють між собою.

Діаграма послідовностей (Sequence Diagram) – це один із типів діаграм у моделюванні UML (Unified Modeling Language), який використовується для моделювання взаємодії між об'єктами системи у певній послідовності часу. Вона відображає, як об'єкти обмінюються повідомленнями, показуючи порядок і логіку виконання операцій.

Діаграма складається з таких основних елементів:

Актори (Actors): Зазвичай позначаються піктограмами або назвами. Це користувачі чи інші системи, які взаємодіють із системою. Актори можуть бути зовнішніми стосовно моделювання системи.

Об'єкти або класи: Розміщуються горизонтально на діаграмі. Вони позначаються прямокутниками з іменем об'єкта або класу під прямокутником. Кожен об'єкт має «життєвий цикл», який представлений вертикальною пунктирною лінією (лінія життя).

Повідомлення: Це лінії зі стрілками, які з'єднують об'єкти. Вони показують передачу повідомлень чи виклик методів. Стрілка може бути синхронною (звичайна стрілка) або асинхронною (лінія з відкритим трикутником) та з пунктирною лінією, що показує повернення результату.

Активності: Вказують періоди, протягом яких об'єкт виконує певну дію. На діаграмі це позначається прямокутником, накладеним на лінію життя.

Контрольні структури: Використовуються для відображення умов, циклів або альтернативних сценаріїв. Наприклад, блоки "alt" (альтернатива) або "loop" (цикл).

Діаграми послідовностей є корисними для моделювання бізнес-процесів, проєктування архітектури систем і тестування. Вони дають змогу візуалізувати логіку взаємодії компонентів та виявити потенційні проблеми ще на етапі проєктування.

Завдання

• Ознайомитись з короткими теоретичними відомостями.

• Проаналізувати діаграми створені в попередній лабораторній роботі а також тему системи та спроєктувати діаграму розгортання використання відповідно до обраної теми лабораторного циклу.

• Розробити діаграму компонентів для проєктованої системи.

• Розробити діаграму розгортання для проєктованої системи.

• Розробити як мінімум дві діаграми послідовностей для сценаріїв прописаних в попередній лабораторній роботі.

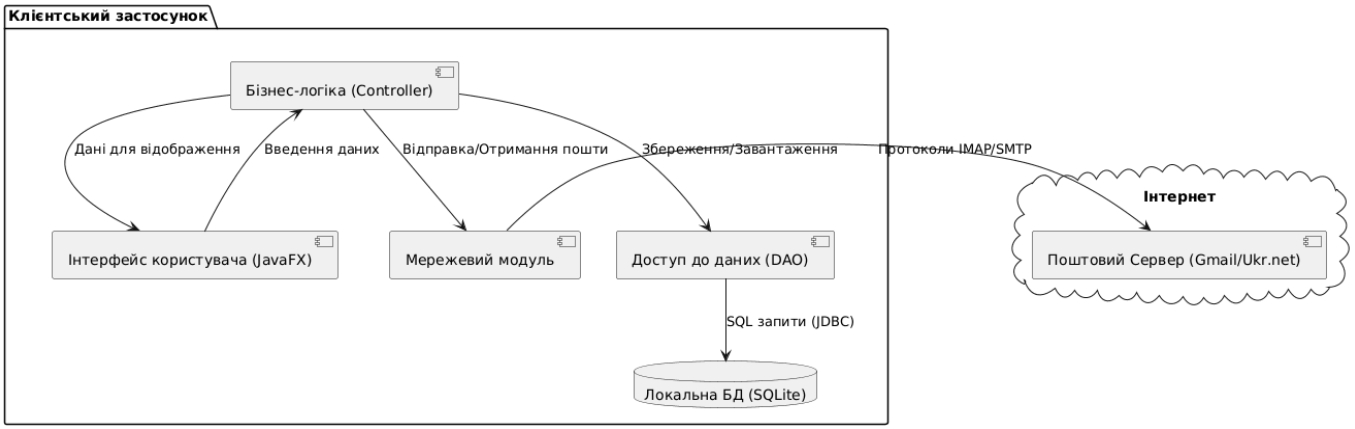
• На основі спроєктованих діаграм розгортання та компонентів доопрацювати програмну частину системи. Реалізація системи, додатково до попередньої реалізації, повинна містити як мінімум дві візуальні форми. В системі вже повинен бути повністю реалізована архітектура (повний цикл роботи з даними від вводу на формі до збереження їх в БД і подальшій виборці з БД та відображенням на UI).

• Підготувати звіт щодо виконання лабораторної роботи. Поданий звіт

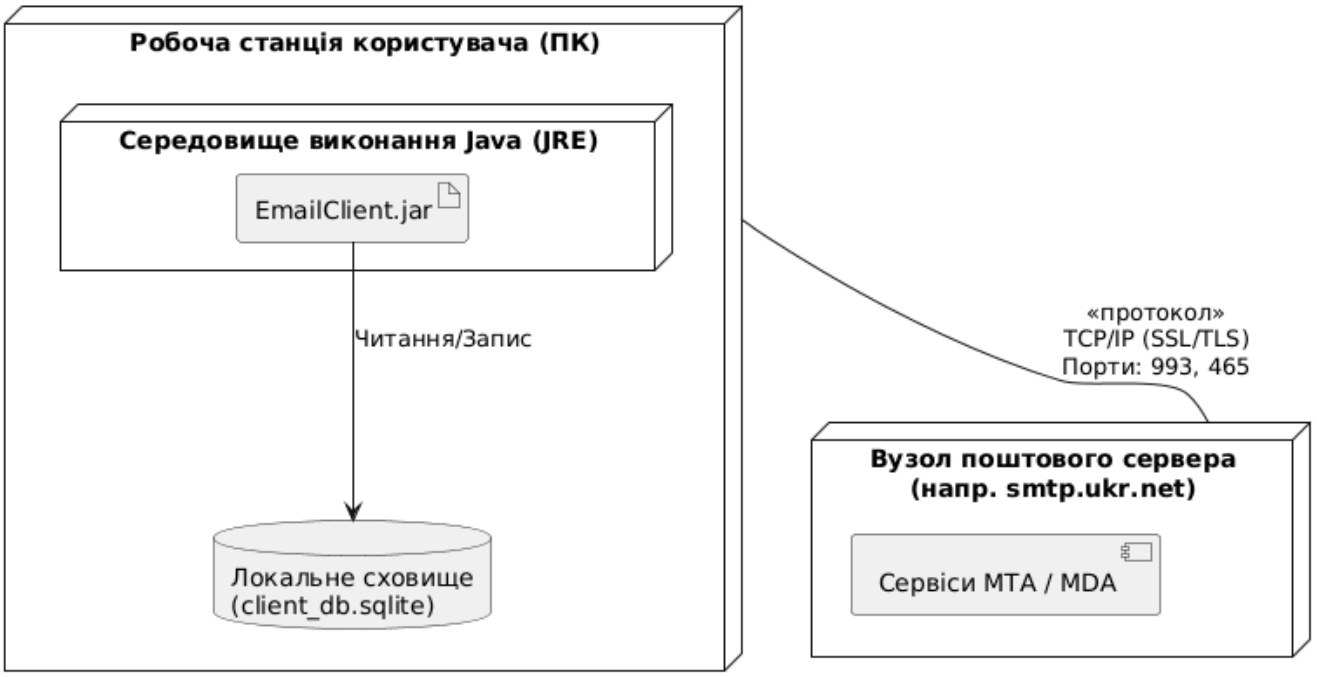
повинен містити: діаграму розгортання з описом, діаграму компонентів системи з описом, діаграми послідовностей, а також вихідний код системи, який було додано в цій лабораторній роботі.

Хід роботи

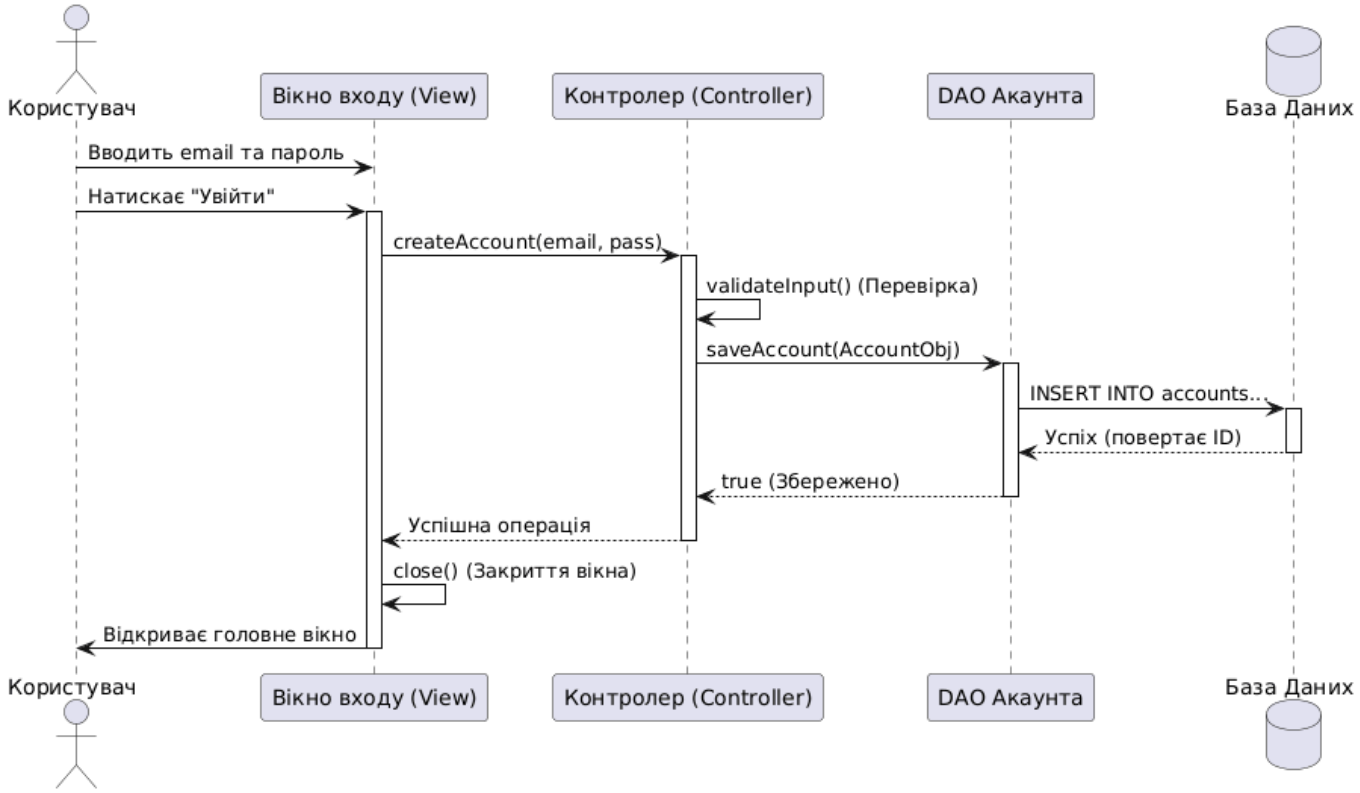
Діаграма компонентів



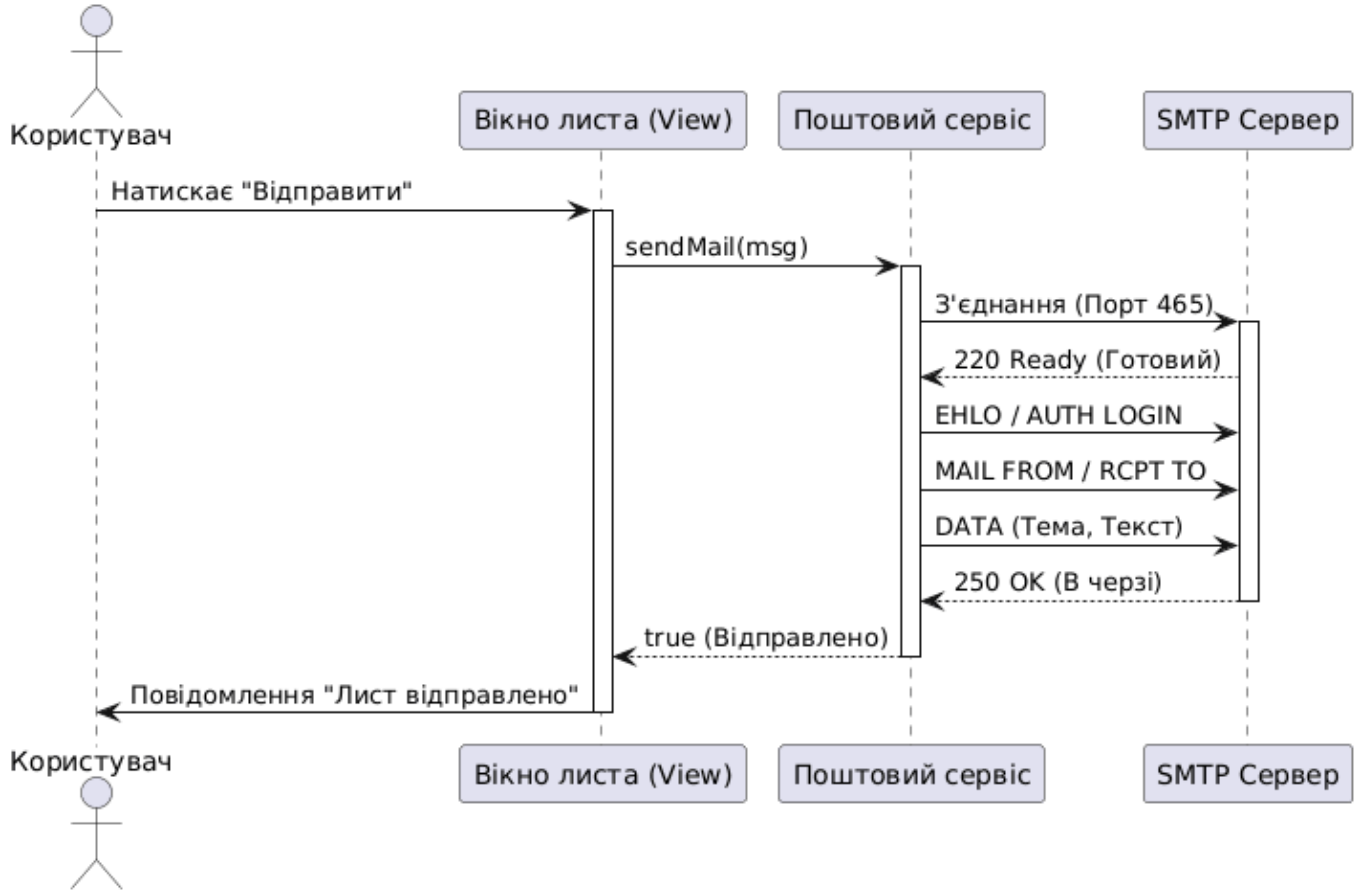
Діаграма розгортання



Діграма послідовностей для Сценарію 1: Налаштування облікового запису

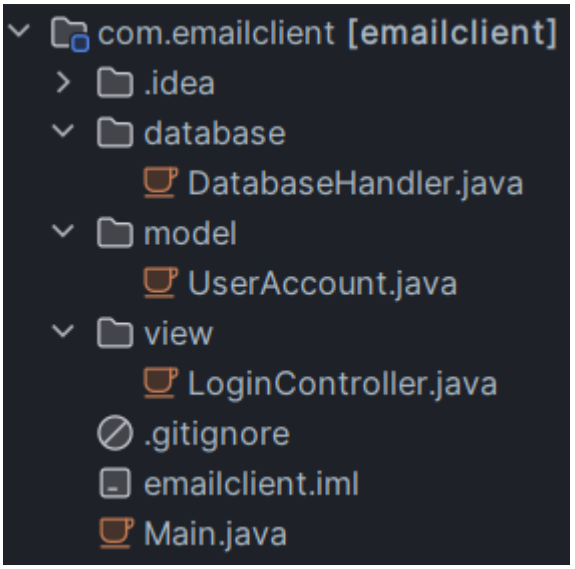


Діграма послідовностей для Сценарію 1: Відправлення листа



Вихідний код системи

Структура проєкту



Main.java

```java
package com.emailclient;

import com.emailclient.database.DatabaseHandler;
import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Scene;
import javafx.stage.Stage;

import java.io.IOException;

public class Main extends Application {
    @Override
    public void start(Stage stage) throws IOException {
        // Ініціалізація БД перед запуском UI
        DatabaseHandler.initDB();

        FXMLLoader fxmlLoader = new FXMLLoader(Main.class.getResource("/view/login.fxml"));
        Scene scene = new Scene(fxmlLoader.load(), 320, 240);
        stage.setTitle("E-mail Client - Login");
        stage.setScene(scene);
        stage.show();
    }

    public static void main(String[] args) {
        launch();
    }
```

```
}
```

DatabaseHandler.java

```
package com.emailclient.database;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;

public class DatabaseHandler {
   private static final String DB_URL = "jdbc:sqlite:email_client.db";
   private static Connection connection;

   public static Connection getConnection() throws SQLException {
       if (connection == null || connection.isClosed()) {
           connection = DriverManager.getConnection(DB_URL);
       }
       return connection;
   }

   // Ініціалізація таблиць при старті
   public static void initDB() {
       String sql = "CREATE TABLE IF NOT EXISTS accounts (" +
               "id INTEGER PRIMARY KEY AUTOINCREMENT, " +
               "email TEXT NOT NULL, " +
               "password TEXT NOT NULL);"; // В реальності пароль шифруємо!

       try (Connection conn = getConnection();
            Statement stmt = conn.createStatement()) {
           stmt.execute(sql);
           System.out.println("DB Initialized.");
       } catch (SQLException e) {
           e.printStackTrace();
       }
   }
}
```

UserAccount.java

```
package com.emailclient.view;

import com.emailclient.database.DatabaseHandler;
```

```
import javafx.fxml.FXML;
import javafx.scene.control.Alert;
import javafx.scene.control.PasswordField;
import javafx.scene.control.TextField;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;

public class LoginController {

    @FXML
    private TextField emailField;

    @FXML
    private PasswordField passwordField;

    @FXML
    protected void onLoginButtonClick() {
        String email = emailField.getText();
        String password = passwordField.getText();

        if (email.isEmpty() || password.isEmpty()) {
            showAlert("Помилка", "Заповніть всі поля!");
            return;
        }

        // Збереження в БД (Full Cycle implementation)
        if (saveAccountToDB(email, password)) {
            showAlert("Успіх", "Акаунт збережено в БД! Перехід до головного вікна...");
            // Тут код відкриття Main Window
        } else {
            showAlert("Помилка", "Не вдалося зберегти дані.");
        }
    }

    private boolean saveAccountToDB(String email, String pass) {
        String insertSQL = "INSERT INTO accounts(email, password) VALUES(?, ?)";

        try (Connection conn = DatabaseHandler.getConnection();
             PreparedStatement pstmt = conn.prepareStatement(insertSQL)) {

            pstmt.setString(1, email);
            pstmt.setString(2, pass);
```

```
            pstmt.executeUpdate();
            return true;

        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        }
    }

    private void showAlert(String title, String content) {
        Alert alert = new Alert(Alert.AlertType.INFORMATION);
        alert.setTitle(title);
        alert.setContentText(content);
        alert.showAndWait();
    }
}
```

LoginController.java

```
package com.emailclient.view;

import com.emailclient.database.DatabaseHandler;
import javafx.fxml.FXML;
import javafx.scene.control.Alert;
import javafx.scene.control.PasswordField;
import javafx.scene.control.TextField;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;

public class LoginController {

    @FXML
    private TextField emailField;

    @FXML
    private PasswordField passwordField;

    @FXML
    protected void onLoginButtonClick() {
        String email = emailField.getText();
        String password = passwordField.getText();

        if (email.isEmpty() || password.isEmpty()) {
```

```
            showAlert("Помилка", "Заповніть всі поля!");
            return;
        }

        // Збереження в БД (Full Cycle implementation)
        if (saveAccountToDB(email, password)) {
            showAlert("Успіх", "Акаунт збережено в БД! Перехід до головного вікна...");
            // Тут код відкриття Main Window
        } else {
            showAlert("Помилка", "Не вдалося зберегти дані.");
        }
    }

    private boolean saveAccountToDB(String email, String pass) {
        String insertSQL = "INSERT INTO accounts(email, password) VALUES(?, ?)";

        try (Connection conn = DatabaseHandler.getConnection();
             PreparedStatement pstmt = conn.prepareStatement(insertSQL)) {

            pstmt.setString(1, email);
            pstmt.setString(2, pass);
            pstmt.executeUpdate();
            return true;

        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        }
    }

    private void showAlert(String title, String content) {
        Alert alert = new Alert(Alert.AlertType.INFORMATION);
        alert.setTitle(title);
        alert.setContentText(content);
        alert.showAndWait();
    }
}
```

Висновок: я навчився проєктувати діаграми розгортання та компонентів для системи що проєктується, а також розробляти діаграми взаємодії, а саме діаграми послідовностей, на основі сценаріїв зроблених в попередній лабораторній роботі.