Министерство образования Калининградской области

государственное бюджетное учреждение Калининградской области

профессиональная образовательная организация

«Колледж информационных технологий и строительства»

(ГБУ КО ПОО «КИТиС»)

**Отчет по учебной практике**

УП.01 Разработка модулей программного обеспечения для компьютерных систем

по ПМ.01 Разработка модулей программного обеспечения для компьютерных систем

Специальность 09.02.07 «Информационные системы и программирование»

Сроки прохождения практики:

с «09» сентября 2024 г. по «09» ноября 2024 г.

Место практики ГБУ КО ПОО «КИТиС»

|  |  |
| --- | --- |
| Выполнил: | студент 4 курса,  группы ИСп 21-1  Колыванова Елизавета Яковлевна  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |
| Проверила: | Большакова-Стрекалова Анна Викторовна  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (оценка)  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) |

Содержание

[Введение 3](#_Toc182483533)

[1. Решение задач 4](#_Toc182483534)

[1.1 Задача 1 60](#_Toc182483535)

[1.2 Задача 2 62](#_Toc182483536)

[2. Разработка АИС «Электронная почта» 65](#_Toc182483537)

[2.1 Диаграммы 65](#_Toc182483538)

[2.2 Техническое задание 68](#_Toc182483539)

[2.3 Тестирование 71](#_Toc182483540)

[3. Разработка мобильного приложения 75](#_Toc182483541)

[4. Разработка сайта 82](#_Toc182483542)

[5. Руководство пользователя 85](#_Toc182483543)

[Список использованной литературы 89](#_Toc182483544)

# Введение

Отчет подготовлен в рамках выполнения индивидуального задания по учебной практике (УП). В ходе практики были рассмотрены и реализованы задачи, связанные с программированием на языке Java, алгоритмизацией, а также проектированием и разработкой автоматизированных информационных систем (АИС) и мобильных приложений.

Основной целью прохождения учебной практики является закрепление теоретических знаний, полученных в процессе обучения, и развитие практических навыков в области разработки программного обеспечения. Выполнение практических заданий направлено на освоение таких ключевых аспектов, как работа с базами данных, обработка исключений, проектирование программных систем, применение алгоритмов и паттернов проектирования, а также оценка и оптимизация программных продуктов.

На протяжении всех этапов практики решались задачи по разработке программных решений на Java, включая управление потоками ввода/вывода и обработку исключений. Также было выполнено проектирование и реализация АИС в соответствии с требованиями второй нормализованной формы, создание диаграмм для системного анализа, а также тестирование и документирование разработанных решений.

В завершающем этапе практики было выполнено задание по разработке мобильного приложения, созданию сайта с использованием системы управления контентом (CMS), а также оценка эффективности программного продукта с применением паттернов проектирования.

Отчет по УП отражает выполненные задания, этапы проектирования и разработки программных решений, а также их тестирование и документирование.

1. Решение задач

Задание №1

5 Транспонировать квадратную матрицу.

Таблица 1 – таблица компонентов задания №1

|  |  |  |
| --- | --- | --- |
| Наименование переменной | Тип данных | Назначение |
| n | int | Входная переменная |
| matrix | int[][] | Входная матрица |
| transposedMatrix | int[][] | Вспомогательная матрица |
| i | int | Вспомогательная переменная |
| j | int | Вспомогательная переменная |

Блок – схема



Код программы

import java.util.Scanner;  
  
public class MatrixTranspose {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.in);  
  
 System.out.print("Введите размер матрицы: ");  
 int n = scanner.nextInt();  
  
 int[][] matrix = new int[n][n];  
 System.out.println("Введите элементы матрицы: ");  
 for (int i = 0; i < n; i++) {  
 for (int j = 0; j < n; j++) {  
 matrix[i][j] = scanner.nextInt();  
 }  
 }  
  
 int[][] transposedMatrix = new int[n][n];  
 for (int i = 0; i < n; i++) {  
 for (int j = 0; j < n; j++) {  
 transposedMatrix[i][j] = matrix[j][i];  
 }  
 }  
  
 System.out.println("Транспонированная матрица: ");  
 for (int i = 0; i < n; i++) {  
 for (int j = 0; j < n; j++) {  
 System.out.print(transposedMatrix[i][j] + " ");  
 }  
 System.out.println();  
 }  
  
 scanner.close();  
 }  
}

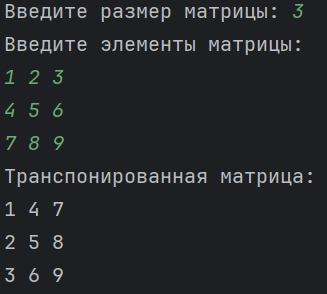


Рисунок 1 – результат задания №1

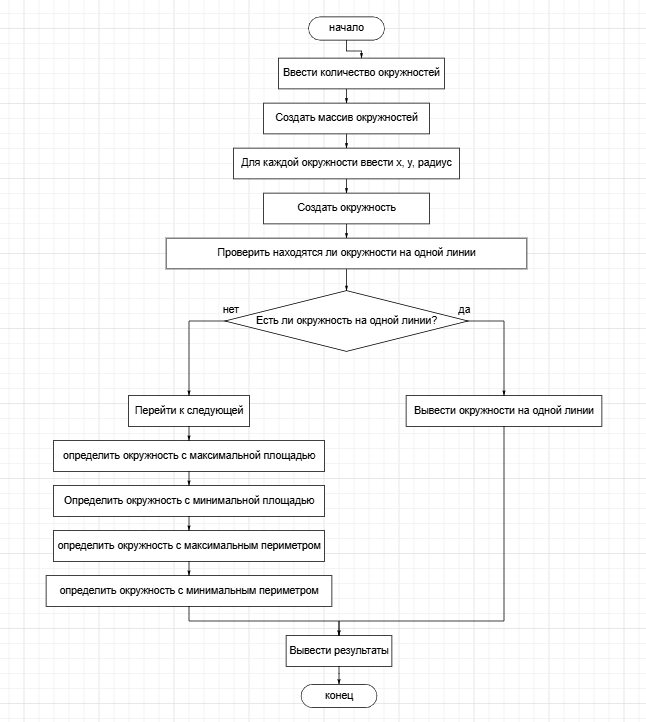
Задание №2

9 Определить класс Окружность на плоскости. Определить площадь и периметр. Создать массив/список/множество объектов и определить группы окружностей, центры которых лежат на одной прямой. Определить наибольший и наименьший по площади (периметру) объект.

Таблица 2 – таблица компонентов задания №2

|  |  |  |
| --- | --- | --- |
| Наименование переменной | Тип данных | Назначение |
| х | double | Координата центра окружности по оси X |
| y | double | Координата центра окружности по оси Y |
| circleList | ArrayList<Circle> | Массив объектов, содержащий все окружности |

Блок – схема



Код программы

import java.util.\*;  
  
class Circle {  
 private double x;   
 private double y;   
 private double radius;   
  
 public Circle(double x, double y, double radius) {  
 if (radius <= 0) {  
 throw new IllegalArgumentException("Радиус должен быть положительным числом.");  
 }  
 this.x = x;  
 this.y = y;  
 this.radius = radius;  
 }  
  
 public double getArea() {  
 return Math.PI \* radius \* radius;  
 }  
  
 public double getPerimeter() {  
 return 2 \* Math.PI \* radius;  
 }  
  
 public double getX() {  
 return x;  
 }  
  
 public double getY() {  
 return y;  
 }  
  
 @Override  
 public String toString() {  
 return "Окружность с центром (" + x + ", " + y + ") и радиусом " + radius;  
 }  
  
 public boolean isOnSameLine(Circle other) {  
 return this.x == other.x || this.y == other.y;  
 }  
}  
  
public class CircleMain {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.in);  
  
 System.out.print("Введите количество окружностей: ");  
 int n = scanner.nextInt();  
  
 Circle[] circles = new Circle[n];  
 for (int i = 0; i < n; i++) {  
 System.out.print("Введите координаты центра (x, y) и радиус окружности " + (i + 1) + ": ");  
 double x = scanner.nextDouble();  
 double y = scanner.nextDouble();  
 double radius = scanner.nextDouble();  
 circles[i] = new Circle(x, y, radius);  
 }  
  
 System.out.println("\nГруппы окружностей с центрами на одной прямой:");  
 for (int i = 0; i < n; i++) {  
 for (int j = i + 1; j < n; j++) {  
 if (circles[i].isOnSameLine(circles[j])) {  
 System.out.println(circles[i] + " и " + circles[j] + " лежат на одной прямой.");  
 }  
 }  
 }  
  
 Circle maxAreaCircle = circles[0];  
 Circle minAreaCircle = circles[0];  
 for (Circle circle : circles) {  
 if (circle.getArea() > maxAreaCircle.getArea()) {  
 maxAreaCircle = circle;  
 }  
 if (circle.getArea() < minAreaCircle.getArea()) {  
 minAreaCircle = circle;  
 }  
 }  
  
 Circle maxPerimeterCircle = circles[0];  
 Circle minPerimeterCircle = circles[0];  
 for (Circle circle : circles) {  
 if (circle.getPerimeter() > maxPerimeterCircle.getPerimeter()) {  
 maxPerimeterCircle = circle;  
 }  
 if (circle.getPerimeter() < minPerimeterCircle.getPerimeter()) {  
 minPerimeterCircle = circle;  
 }  
 }  
  
 System.out.println("\nОкружность с наибольшей площадью: " + maxAreaCircle);  
 System.out.println("Окружность с наименьшей площадью: " + minAreaCircle);  
 System.out.println("Окружность с наибольшим периметром: " + maxPerimeterCircle);  
 System.out.println("Окружность с наименьшим периметром: " + minPerimeterCircle);  
  
 scanner.close();  
 }  
}

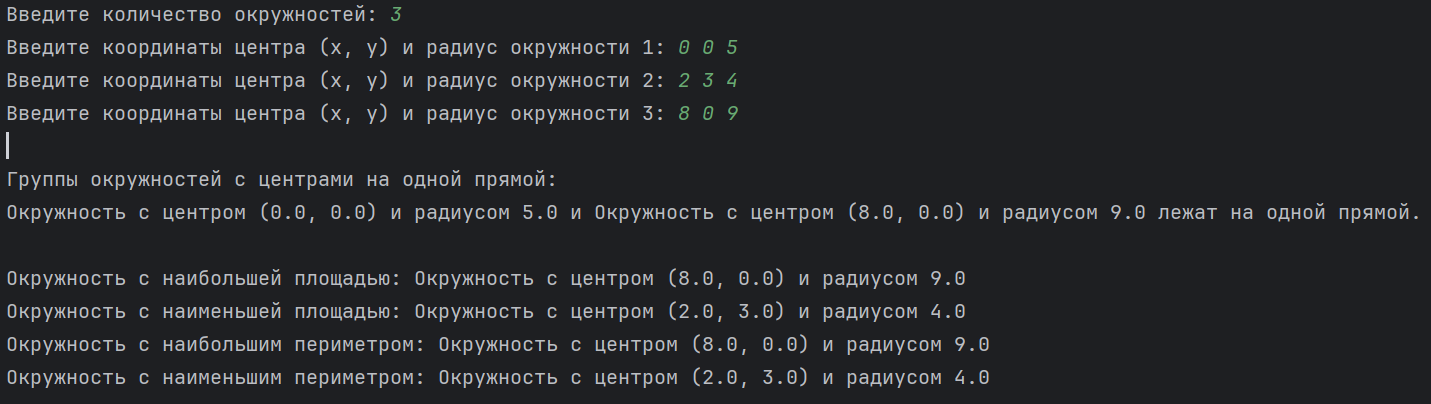


Рисунок 2 – результат задания №2

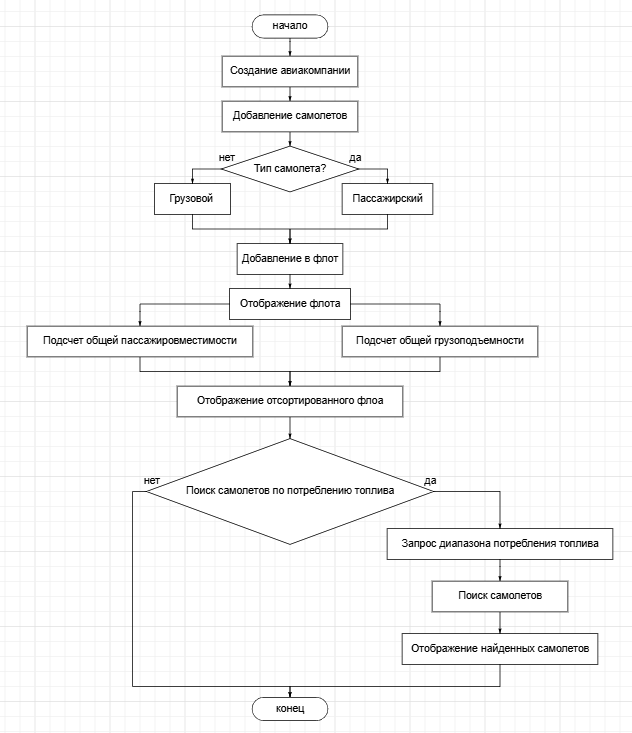
Задание №3

9 Авиакомпания. Определить иерархию самолетов. Создать авиакомпанию. Посчитать общую вместимость и грузоподъемность. Провести сортировку самолетов компании по дальности полета. Найти самолет в компании, соответствующий заданному диапазону параметров потребления горючего.

Таблица 3 – таблица компонентов задания №3

|  |  |  |
| --- | --- | --- |
| Наименование переменной | Тип данных | Назначение |
| |  | | --- | | capacity | | Int | Пассажировместимость самолета |
| cargoCapacity | double | Грузоподъемность самолета в тоннах |
| range | double | Дальность полета самолета в километрах |
| fuelConsumption | double | Потребление топлива самолетом в литрах на 100 км |
| fleet | List<Airplane> | Список самолетов авиакомпании |
| minFuelConsumption | double | Минимальное значение диапазона потребления топлива |
| maxFuelConsumption | double | Максимальное значение диапазона потребления топлива |
| airplane | Airplane | Объект класса, представляющий конкретный самолет |
| foundPlanes | List<Airplane> | Список самолетов, соответствующих критериям потребления топлива |

Блок – схема



Код программы

import java.util.ArrayList;  
import java.util.Comparator;  
import java.util.List;  
  
abstract class Airplane {  
 protected String model;  
 protected int capacity;   
 protected double cargoCapacity;   
 protected double range;   
 protected double fuelConsumption;   
  
 public Airplane(String model, int capacity, double cargoCapacity, double range, double fuelConsumption) {  
 this.model = model;  
 this.capacity = capacity;  
 this.cargoCapacity = cargoCapacity;  
 this.range = range;  
 this.fuelConsumption = fuelConsumption;  
 }  
  
 public String getModel() {  
 return model;  
 }  
  
 public int getCapacity() {  
 return capacity;  
 }  
  
 public double getCargoCapacity() {  
 return cargoCapacity;  
 }  
  
 public double getRange() {  
 return range;  
 }  
  
 public double getFuelConsumption() {  
 return fuelConsumption;  
 }  
  
 @Override  
 public String toString() {  
 return String.format("%s: Пассажировместимость=%d, Грузоподъемность=%.2f тонн, Дальность=%.2f км, Потребление топлива=%.2f л/100 км",  
 model, capacity, cargoCapacity, range, fuelConsumption);  
 }  
}  
  
class PassengerPlane extends Airplane {  
 public PassengerPlane(String model, int capacity, double range, double fuelConsumption) {  
 super(model, capacity, 0, range, fuelConsumption);  
 }  
}  
  
class CargoPlane extends Airplane {  
 public CargoPlane(String model, double cargoCapacity, double range, double fuelConsumption) {  
 super(model, 0, cargoCapacity, range, fuelConsumption);  
 }  
}  
  
class Airline {  
 private List<Airplane> fleet;  
  
 public Airline() {  
 this.fleet = new ArrayList<>();  
 }  
  
 public void addAirplane(Airplane airplane) {  
 fleet.add(airplane);  
 }  
  
 public int totalCapacity() {  
 return fleet.stream().mapToInt(Airplane::getCapacity).sum();  
 }  
  
 public double totalCargoCapacity() {  
 return fleet.stream().mapToDouble(Airplane::getCargoCapacity).sum();  
 }  
  
 public void sortByRange() {  
 fleet.sort(Comparator.comparingDouble(Airplane::getRange));  
 }  
  
 public List<Airplane> findPlanesByFuelConsumption(double minFuel, double maxFuel) {  
 List<Airplane> result = new ArrayList<>();  
 for (Airplane airplane : fleet) {  
 if (airplane.getFuelConsumption() >= minFuel && airplane.getFuelConsumption() <= maxFuel) {  
 result.add(airplane);  
 }  
 }  
 return result;  
 }  
  
 public void displayFleet() {  
 for (Airplane airplane : fleet) {  
 System.out.println(airplane);  
 }  
 }  
}  
  
public class Main {  
 public static void main(String[] args) {  
   
 Airline airline = new Airline();  
  
 airline.addAirplane(new PassengerPlane("Boeing 737", 160, 3000, 2500));  
 airline.addAirplane(new PassengerPlane("Airbus A320", 180, 3500, 2700));  
 airline.addAirplane(new CargoPlane("Boeing 747 Cargo", 100, 8000, 5000));  
 airline.addAirplane(new CargoPlane("Antonov AN-124", 150, 5000, 6000));  
  
 System.out.println("Флот авиакомпании:");  
 airline.displayFleet();  
  
 System.out.println("\nОбщая пассажировместимость: " + airline.totalCapacity());  
 System.out.println("Общая грузоподъемность: " + airline.totalCargoCapacity() + " тонн");  
  
 airline.sortByRange();  
 System.out.println("\nФлот, отсортированный по дальности полета:");  
 airline.displayFleet();  
  
 double minFuelConsumption = 2500;  
 double maxFuelConsumption = 6000;  
 System.out.println("\nСамолеты с потреблением топлива от " + minFuelConsumption + " до " + maxFuelConsumption + " л/100 км:");  
 List<Airplane> foundPlanes = airline.findPlanesByFuelConsumption(minFuelConsumption, maxFuelConsumption);  
 for (Airplane airplane : foundPlanes) {  
 System.out.println(airplane);  
 }  
 }  
}

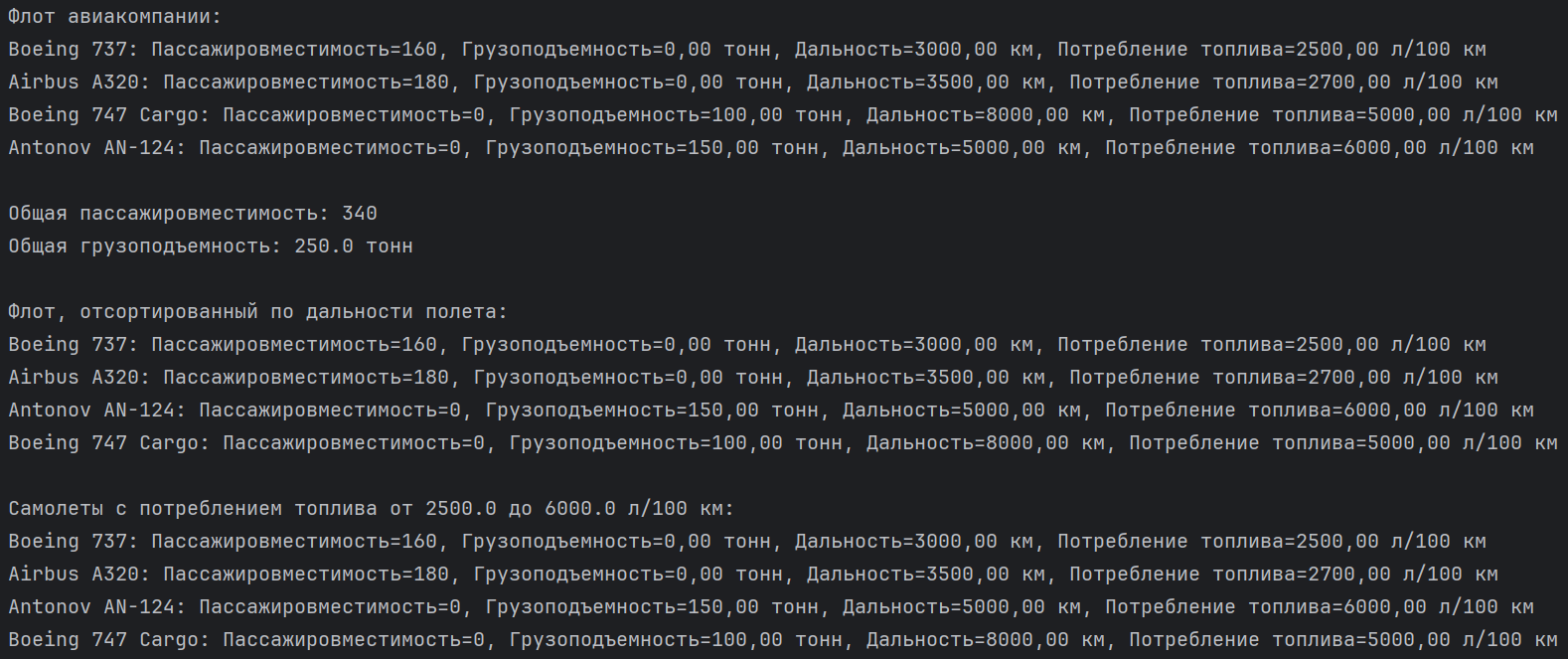


Рисунок 3 – результат задания №3

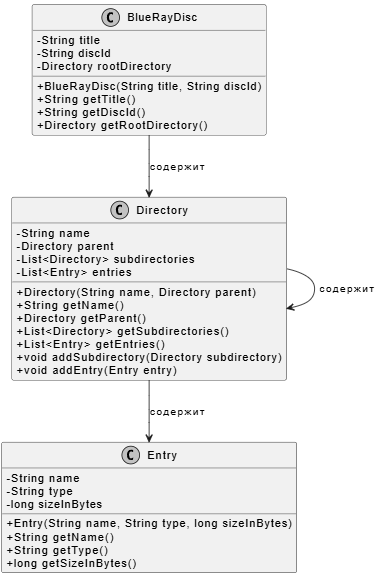
Задание №4

Создать класс BlueRayDisc с внутренним классом, с помощью объектов которого можно хранить информацию о каталогах, подкаталогах и записях.

Таблица 4 – таблица компонентов задания №4

|  |  |  |
| --- | --- | --- |
| Наименование переменной | Тип данных | Назначение |
| disc | BlueRayDisc | Объект класса BlueRayDisc, представляющий Blu – Ray диск |
| rootDirectory | Directory | Корневой каталог диска |
| subDirectory | Directory | Переменная, представляющая подкаталог текущего каталога |
| entry | int | Переменная, представляющая записи в текущем каталоге |

Диаграмма классов



Код программы

import java.util.ArrayList;  
import java.util.List;  
  
public class BlueRayDisc {  
  
 private String title;  
 private String discId;  
 private Directory rootDirectory;  
  
 public BlueRayDisc(String title, String discId) {  
 this.title = title;  
 this.discId = discId;  
 this.rootDirectory = new Directory("/", null);  
 }  
  
 public String getTitle() {  
 return title;  
 }  
  
 public String getDiscId() {  
 return discId;  
 }  
  
 public Directory getRootDirectory() {  
 return rootDirectory;  
 }  
  
 public static class Directory {  
 private String name;  
 private Directory parent;  
 private List<Directory> subdirectories;  
 private List<Entry> entries;  
  
 public Directory(String name, Directory parent) {  
 this.name = name;  
 this.parent = parent;  
 this.subdirectories = new ArrayList<>();  
 this.entries = new ArrayList<>();  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public Directory getParent() {  
 return parent;  
 }  
  
 public List<Directory> getSubdirectories() {  
 return subdirectories;  
 }  
  
 public List<Entry> getEntries() {  
 return entries;  
 }  
  
 public void addSubdirectory(Directory subdirectory) {  
 subdirectories.add(subdirectory);  
 }  
  
 public void addEntry(Entry entry) {  
 entries.add(entry);  
 }  
 }  
  
 public static class Entry {  
 private String name;  
 private String type; // "movie", "music", "photo", etc.  
 private long sizeInBytes;  
  
 public Entry(String name, String type, long sizeInBytes) {  
 this.name = name;  
 this.type = type;  
 this.sizeInBytes = sizeInBytes;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public String getType() {  
 return type;  
 }  
  
 public long getSizeInBytes() {  
 return sizeInBytes;  
 }  
 }  
  
 public static void main(String[] args) {  
 BlueRayDisc disc = new BlueRayDisc("Movie Collection", "BR19929");  
  
 Directory movies = new Directory("Movies", disc.getRootDirectory());  
 Directory action = new Directory("Action", movies);  
 Directory comedy = new Directory("Comedy", movies);  
  
 disc.getRootDirectory().addSubdirectory(movies);  
 movies.addSubdirectory(action);  
 movies.addSubdirectory(comedy);  
  
 Entry theMatrix = new Entry("ЯЛегенда", "movie", 1024 \* 1024 \* 1024);  
 Entry dieHard = new Entry("Трон", "movie", 800 \* 1024 \* 1024);  
 Entry theHangover = new Entry("Падингтон", "movie", 950 \* 1024 \* 1024);  
  
 action.addEntry(theMatrix);  
 action.addEntry(dieHard);  
 comedy.addEntry(theHangover);  
  
 System.out.println("Title: " + disc.getTitle());  
 System.out.println("Disc ID: " + disc.getDiscId());  
  
 System.out.println("\nDirectory Structure:");  
 printDirectoryStructure(disc.getRootDirectory(), "");  
 }  
  
 private static void printDirectoryStructure(Directory directory, String indent) {  
 System.out.println(indent + directory.getName());  
 for (Directory subdirectory : directory.getSubdirectories()) {  
 printDirectoryStructure(subdirectory, indent + " ");  
 }  
 for (Entry entry : directory.getEntries()) {  
 System.out.println(indent + " " + entry.getName() + " (" + entry.getType() + ")");  
 }  
 }  
}

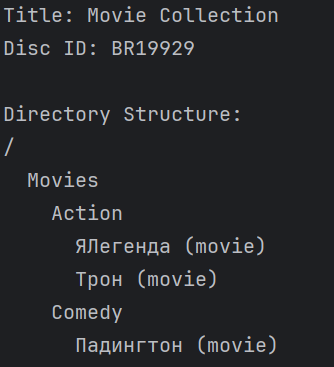


Рисунок 4 – результат задания №4

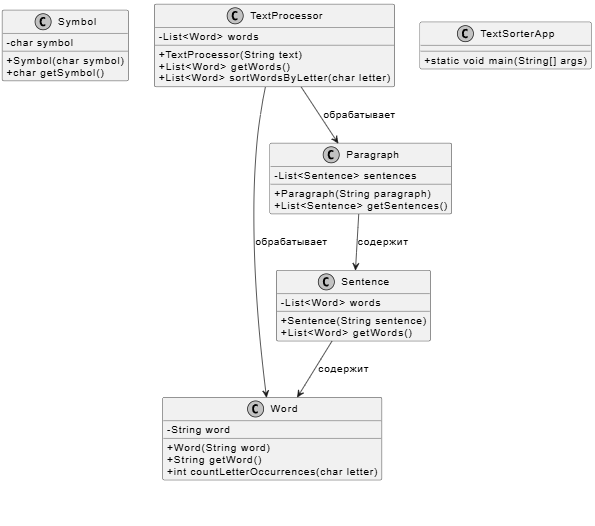
Задание №5

9 Все слова текста рассортировать по возрастанию количества заданной буквы в слове. Слова с одинаковым количеством расположить в алфавитном порядке.

Таблица 5 – таблица компонентов задания №5

|  |  |  |
| --- | --- | --- |
| Наименование переменной | Тип данных | Назначение |
| count | int | Хранит количество вхождений буквы в слове. |
| words | List<Word> | Список объектов Word, представляющий слова в предложении. |
| sentenceStr | String | Хранит строку, представляющую предложение. |
| paragraphStr | String | Хранит строку, представляющую абзац. |
| letter | char | Хранит букву, по которой будет происходить сортировка слов. |
| sortedWords | List<Word> | Хранит отсортированные слова по количеству вхождений буквы и алфавиту. |
| textBuilder | StringBuilder | Используется для накопления строк текста, вводимого пользователем. |

Диаграмма классов



Код программы

import java.util.ArrayList;  
import java.util.Collections;  
import java.util.Comparator;  
import java.util.List;  
import java.util.Scanner;  
  
class Symbol {  
 private char symbol;  
  
 public Symbol(char symbol) {  
 this.symbol = symbol;  
 }  
  
 public char getSymbol() {  
 return symbol;  
 }  
}  
  
class Word {  
 private String word;  
  
 public Word(String word) {  
   
 this.word = word.trim().replaceAll("\\s+", " ");  
 }  
  
 public String getWord() {  
 return word;  
 }  
  
 public int countLetterOccurrences(char letter) {  
 int count = 0;  
 for (char c : word.toCharArray()) {  
 if (Character.toLowerCase(c) == Character.toLowerCase(letter)) {  
 count++;  
 }  
 }  
 return count;  
 }  
}  
  
class Sentence {  
 private List<Word> words;  
  
 public Sentence(String sentence) {  
 words = new ArrayList<>();  
 String[] wordArray = sentence.split("\\s+"); // Разделение предложения на слова  
 for (String wordStr : wordArray) {  
 if (!wordStr.isEmpty()) {  
 words.add(new Word(wordStr));  
 }  
 }  
 }  
  
 public List<Word> getWords() {  
 return words;  
 }  
}  
  
class Paragraph {  
 private List<Sentence> sentences;  
  
 public Paragraph(String paragraph) {  
 sentences = new ArrayList<>();  
 String[] sentenceArray = paragraph.split("(?<=[.!?])\\s+"); // Разделение на предложения  
 for (String sentenceStr : sentenceArray) {  
 if (!sentenceStr.trim().isEmpty()) {  
 sentences.add(new Sentence(sentenceStr.trim()));  
 }  
 }  
 }  
  
 public List<Sentence> getSentences() {  
 return sentences;  
 }  
}  
  
class TextProcessor {  
 private List<Word> words;  
  
 public TextProcessor(String text) {  
 words = new ArrayList<>();  
 String[] paragraphArray = text.split("\\n\\n"); // Разделение на абзацы  
 for (String paragraphStr : paragraphArray) {  
 Paragraph paragraph = new Paragraph(paragraphStr.trim());  
 for (Sentence sentence : paragraph.getSentences()) {  
 words.addAll(sentence.getWords());  
 }  
 }  
 }  
  
 public List<Word> getWords() {  
 return words;  
 }  
  
 public List<Word> sortWordsByLetter(char letter) {  
 Collections.sort(words, new Comparator<Word>() {  
 @Override  
 public int compare(Word w1, Word w2) {  
 int count1 = w1.countLetterOccurrences(letter);  
 int count2 = w2.countLetterOccurrences(letter);  
  
 if (count1 == count2) {  
 return w1.getWord().compareToIgnoreCase(w2.getWord());  
 }  
 return Integer.compare(count1, count2); // Сортировка по количеству  
 }  
 });  
 return words;  
 }  
}  
  
public class TextSorterApp {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.in);  
  
 System.out.println("Введите текст (для завершения ввода введите 'exit'):");  
 StringBuilder textBuilder = new StringBuilder();  
 String line;  
 while (!(line = scanner.nextLine()).equals("exit")) {  
 textBuilder.append(line).append("\n");  
 }  
 String text = textBuilder.toString().trim();  
  
 System.out.println("Введите букву для сортировки:");  
 char letter = scanner.nextLine().charAt(0);  
  
 TextProcessor processor = new TextProcessor(text);  
 List<Word> sortedWords = processor.sortWordsByLetter(letter);  
  
 System.out.println("Сортированные слова:");  
 for (Word word : sortedWords) {  
 System.out.println(word.getWord());  
 }  
  
 scanner.close();  
 }  
}

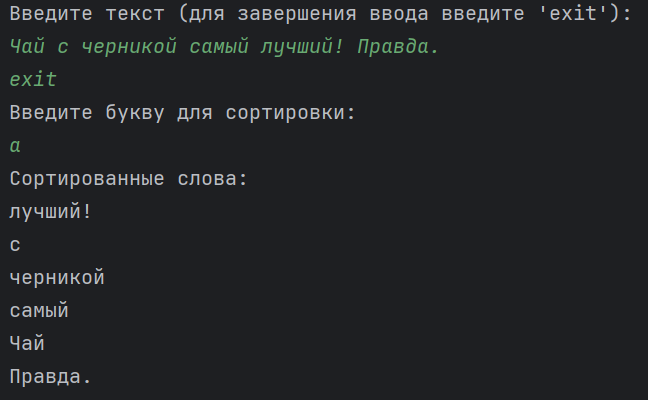


Рисунок 5 – результат задания №5

Задание №6

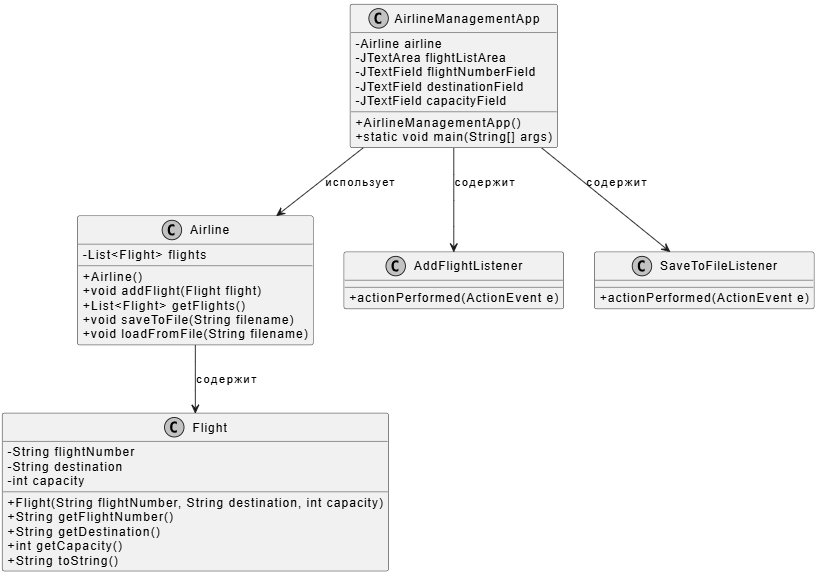
Выполнить задания на основе задачи № 3, контролируя состояние потоков ввода/вывода. При возникновении ошибок, связанных с корректностью выполнения математических операций, генерировать и обрабатывать исключительные ситуации. Предусмотреть обработку исключений, возникающих при нехватке памяти, отсутствии требуемой записи (объекта) в файле, недопустимом значении поля и т. д.

Выполнить задания из задачи № 3, реализуя собственные обработчики исключений и исключения ввода/вывода.

Таблица 6 – таблица компонентов задания №6

|  |  |  |
| --- | --- | --- |
| Наименование переменной | Тип данных | Назначение |
| flightNumber | String | Хранит номер рейса в формате двух заглавных букв и трех цифр |
| destination | String | Хранит направление рейса |
| capacity | int | Хранит вместимость рейса (количество мест). |
| flights | List<Flight> | Список объектов Flight, представляющих все рейсы авиакомпании. |
| filename | String | Хранит имя файла для сохранения или загрузки рейсов. |
| flight | Flight | Временный объект класса Flight для добавления рейса в список. |
| ex | Exception | Объект для обработки различных исключений. |

Диаграмма классов



Код программы

import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.io.\*;  
import java.util.ArrayList;  
import java.util.List;  
  
class Flight {  
 private String flightNumber;  
 private String destination;  
 private int capacity;  
  
 public Flight(String flightNumber, String destination, int capacity) throws InvalidFlightNumberException {  
 if (!flightNumber.matches("^[A-Z]{2}\\d{3}$")) {  
 throw new InvalidFlightNumberException("Недопустимый номер рейса: " + flightNumber);  
 }  
 this.flightNumber = flightNumber;  
 this.destination = destination;  
 this.capacity = capacity;  
 }  
  
 public String getFlightNumber() {  
 return flightNumber;  
 }  
  
 public String getDestination() {  
 return destination;  
 }  
  
 public int getCapacity() {  
 return capacity;  
 }  
  
 @Override  
 public String toString() {  
 return "Рейс: " + flightNumber + ", Направление: " + destination + ", Вместимость: " + capacity;  
 }  
}  
  
class InvalidFlightNumberException extends Exception {  
 public InvalidFlightNumberException(String message) {  
 super(message);  
 }  
}  
  
class InsufficientMemoryException extends Exception {  
 public InsufficientMemoryException(String message) {  
 super(message);  
 }  
}  
  
class FileNotFoundCustomException extends Exception {  
 public FileNotFoundCustomException(String message) {  
 super(message);  
 }  
}  
  
class Airline {  
 private List<Flight> flights;  
  
 public Airline() {  
 flights = new ArrayList<>();  
 }  
  
 public void addFlight(Flight flight) throws InsufficientMemoryException {  
 if (flights.size() >= 100) { // Предположим, что максимальное количество рейсов - 100  
 throw new InsufficientMemoryException("Недостаточно памяти для добавления рейса.");  
 }  
 flights.add(flight);  
 }  
  
 public List<Flight> getFlights() {  
 return flights;  
 }  
  
 public void saveToFile(String filename) throws IOException {  
 try (BufferedWriter writer = new BufferedWriter(new FileWriter(filename))) {  
 for (Flight flight : flights) {  
 writer.write(flight.toString());  
 writer.newLine();  
 }  
 }  
 }  
  
 public void loadFromFile(String filename) throws FileNotFoundCustomException, IOException {  
 File file = new File(filename);  
 if (!file.exists()) {  
 throw new FileNotFoundCustomException("Файл не найден: " + filename);  
 }  
 try (BufferedReader reader = new BufferedReader(new FileReader(file))) {  
 String line;  
 while ((line = reader.readLine()) != null) {  
 String[] parts = line.split(", ");  
 String flightNumber = parts[0].split(": ")[1];  
 String destination = parts[1].split(": ")[1];  
 int capacity = Integer.parseInt(parts[2].split(": ")[1]);  
 flights.add(new Flight(flightNumber, destination, capacity));  
 }  
 } catch (NumberFormatException e) {  
 throw new IOException("Ошибка формата числа при чтении файла.", e);  
 } catch (InvalidFlightNumberException e) {  
 throw new RuntimeException(e);  
 }  
 }  
}  
  
public class AirlineManagementApp extends JFrame {  
 private Airline airline;  
 private JTextArea flightListArea;  
 private JTextField flightNumberField;  
 private JTextField destinationField;  
 private JTextField capacityField;  
  
 public AirlineManagementApp() {  
 airline = new Airline();  
 setTitle("Управление Авиакомпанией");  
 setSize(500, 400);  
 setDefaultCloseOperation(EXIT\_ON\_CLOSE);  
 setLayout(new BorderLayout());  
  
 JPanel inputPanel = new JPanel();  
 inputPanel.setLayout(new GridLayout(4, 2));  
  
 inputPanel.add(new JLabel("Номер рейса:"));  
 flightNumberField = new JTextField();  
 inputPanel.add(flightNumberField);  
  
 inputPanel.add(new JLabel("Направление:"));  
 destinationField = new JTextField();  
 inputPanel.add(destinationField);  
  
 inputPanel.add(new JLabel("Вместимость:"));  
 capacityField = new JTextField();  
 inputPanel.add(capacityField);  
  
 JButton addButton = new JButton("Добавить рейс");  
 addButton.addActionListener(new AddFlightListener());  
 inputPanel.add(addButton);  
  
 JButton saveButton = new JButton("Сохранить в файл");  
 saveButton.addActionListener(new SaveToFileListener());  
 inputPanel.add(saveButton);  
  
 add(inputPanel, BorderLayout.NORTH);  
  
 flightListArea = new JTextArea();  
 flightListArea.setEditable(false);  
 add(new JScrollPane(flightListArea), BorderLayout.CENTER);  
 }  
  
 private class AddFlightListener implements ActionListener {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 String flightNumber = flightNumberField.getText();  
 String destination = destinationField.getText();  
 int capacity;  
 try {  
 capacity = Integer.parseInt(capacityField.getText());  
 Flight flight = new Flight(flightNumber, destination, capacity);  
 airline.addFlight(flight);  
 flightListArea.append(flight.toString() + "\n");  
 flightNumberField.setText("");  
 destinationField.setText("");  
 capacityField.setText("");  
 } catch (InvalidFlightNumberException ex) {  
 JOptionPane.showMessageDialog(null, ex.getMessage(), "Ошибка", JOptionPane.ERROR\_MESSAGE);  
 } catch (InsufficientMemoryException ex) {  
 JOptionPane.showMessageDialog(null, ex.getMessage(), "Ошибка", JOptionPane.ERROR\_MESSAGE);  
 } catch (NumberFormatException ex) {  
 JOptionPane.showMessageDialog(null, "Введите корректное значение вместимости.", "Ошибка", JOptionPane.ERROR\_MESSAGE);  
 }  
 }  
 }  
  
 private class SaveToFileListener implements ActionListener {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 String filename = "flights.txt";  
 try {  
 airline.saveToFile(filename);  
 JOptionPane.showMessageDialog(null, "Рейсы успешно сохранены в " + filename);  
 } catch (IOException ex) {  
 JOptionPane.showMessageDialog(null, "Ошибка при сохранении файла: " + ex.getMessage(), "Ошибка", JOptionPane.ERROR\_MESSAGE);  
 }  
 }  
 }  
  
 public static void main(String[] args) {  
 SwingUtilities.invokeLater(() -> {  
 new AirlineManagementApp().setVisible(true);  
 });  
 }  
}

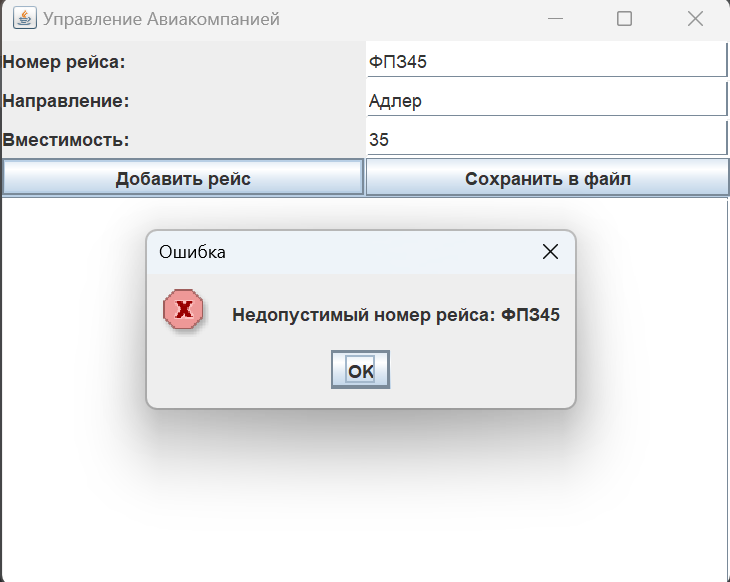
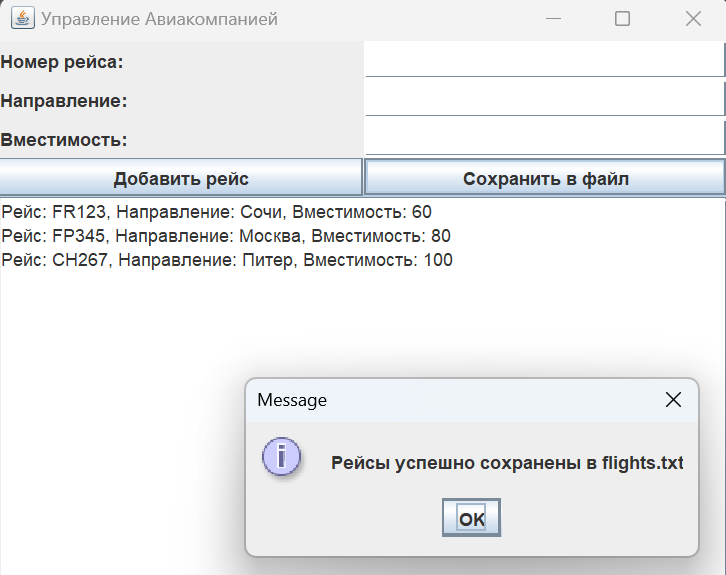


Рисунок 6 – результат задания №6

Задание №7

Составить программу получения на экране рисунка

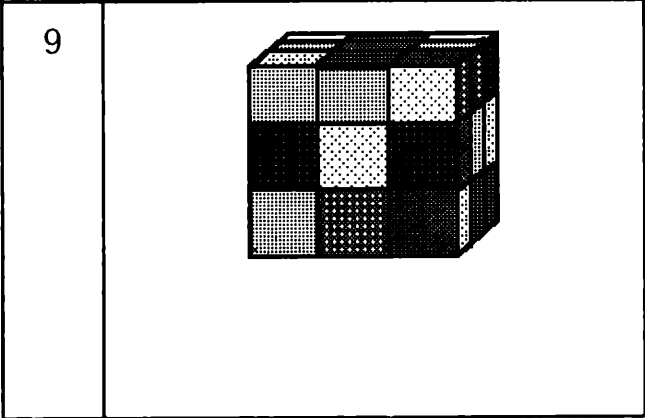
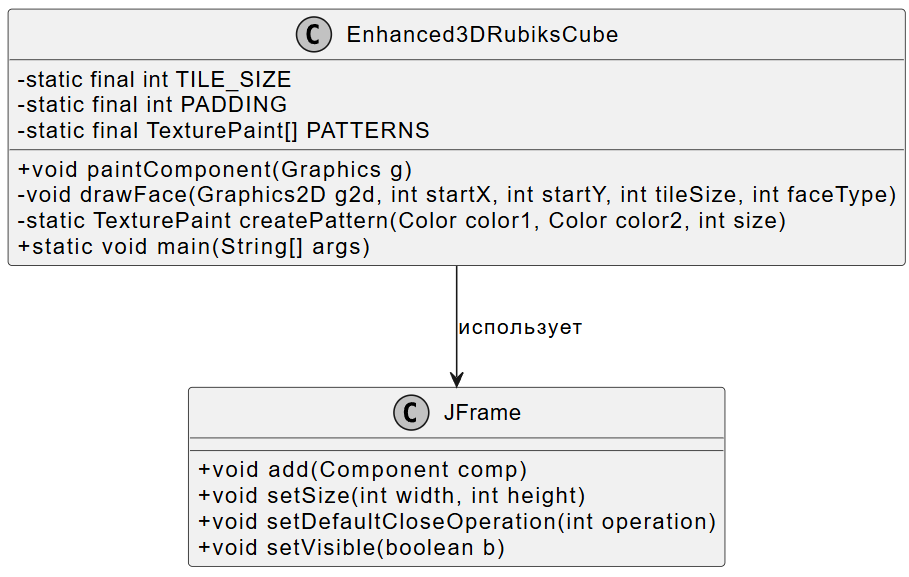


Диаграмма классов



Код программы

import javax.swing.\*;

import java.awt.\*;

import java.awt.geom.Rectangle2D;

import java.awt.image.BufferedImage;

public class Enhanced3DRubiksCube extends JPanel {

private static final int TILE\_SIZE = 50; // Размер каждой плитки

private static final int PADDING = 4; // Отступ между плитками

private static final TexturePaint[] PATTERNS = {

createPattern(Color.BLACK, Color.WHITE, 10), // Черный с белыми точками

createPattern(Color.WHITE, Color.BLACK, 10), // Белый с черными точками

createPattern(Color.DARK\_GRAY, Color.LIGHT\_GRAY, 10), // Серый с черными точками

createPattern(Color.LIGHT\_GRAY, Color.DARK\_GRAY, 15), // Светло-серый с черными точками

createPattern(Color.BLACK, Color.GRAY, 5) // Черный с серыми точками

};

@Override

protected void paintComponent(Graphics g) {

super.paintComponent(g);

Graphics2D g2d = (Graphics2D) g;

g2d.setRenderingHint(RenderingHints.KEY\_ANTIALIASING, RenderingHints.VALUE\_ANTIALIAS\_ON);

drawFace(g2d, 150, 150, TILE\_SIZE, 0); // Передняя грань

drawFace(g2d, 120, 110, TILE\_SIZE, 1); // Верхняя грань (светлее для эффекта объема)

drawFace(g2d, 190, 110, TILE\_SIZE, 2); // Правая грань (темнее для эффекта объема)

}

private void drawFace(Graphics2D g2d, int startX, int startY, int tileSize, int faceType) {

for (int row = 0; row < 3; row++) {

for (int col = 0; col < 3; col++) {

int x = startX + col \* (tileSize + PADDING);

int y = startY + row \* (tileSize + PADDING);

if (faceType == 1) { // Верхняя грань

x -= col \* 10;

y -= row \* 10;

} else if (faceType == 2) { // Правая грань

x += col \* 10;

y -= row \* 10;

}

g2d.setPaint(PATTERNS[(row + col) % PATTERNS.length]);

g2d.fill(new Rectangle2D.Double(x, y, tileSize, tileSize));

g2d.setColor(Color.BLACK);

g2d.setStroke(new BasicStroke(2));

g2d.draw(new Rectangle2D.Double(x, y, tileSize, tileSize));

if (faceType == 1) { // Тень для верхней грани (светлее)

g2d.setColor(new Color(220, 220, 220, 120));

} else if (faceType == 2) { // Тень для правой грани (темнее)

g2d.setColor(new Color(0, 0, 0, 80));

} else {

g2d.setColor(new Color(100, 100, 100, 80));

}

g2d.fillRect(x + tileSize - 5, y + tileSize - 5, 10, 10); // Угловая тень

}

}

}

private static TexturePaint createPattern(Color color1, Color color2, int size) {

BufferedImage bi = new BufferedImage(size, size, BufferedImage.TYPE\_INT\_ARGB);

Graphics2D g2 = bi.createGraphics();

g2.setColor(color1);

g2.fillRect(0, 0, size, size);

g2.setColor(color2);

for (int i = 0; i < size; i += 4) {

for (int j = 0; j < size; j += 4) {

g2.fillRect(i, j, 2, 2);

}

}

g2.dispose();

return new TexturePaint(bi, new Rectangle(0, 0, size, size));

}

public static void main(String[] args) {

JFrame frame = new JFrame("Enhanced 3D Rubik's Cube");

Enhanced3DRubiksCube cubePanel = new Enhanced3DRubiksCube();

frame.add(cubePanel);

frame.setSize(400, 400);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setVisible(true);

}

}

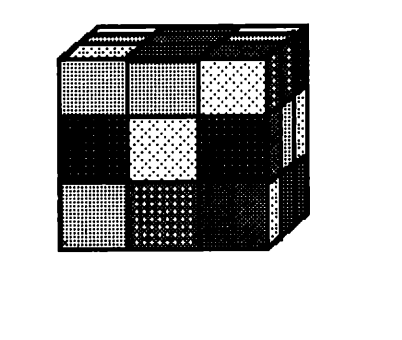


Рисунок 7 – результат задания №7

Задание №8

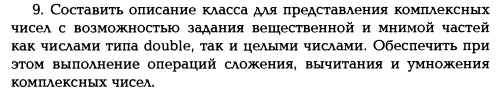
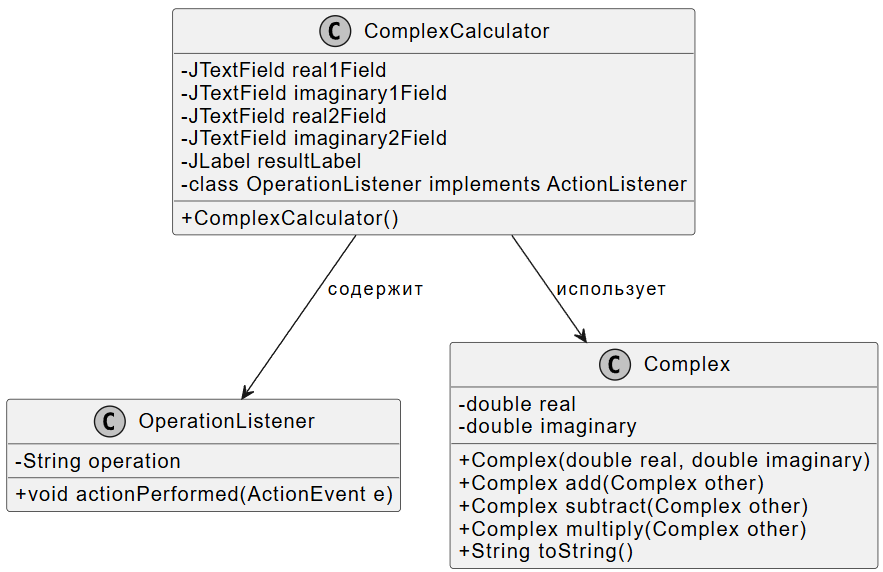


Диаграмма классов



Код программы

import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
  
class ComplexCalculator extends JFrame {  
  
 private JTextField real1Field, imaginary1Field, real2Field, imaginary2Field;  
 private JLabel resultLabel;  
  
 public ComplexCalculator() {  
 setTitle("Калькулятор комплексных чисел");  
 setSize(500, 350);  
 setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  
 setLayout(new GridBagLayout());  
  
 Color backgroundColor = new Color(240, 248, 255); // Нежно-голубой фон  
 Color buttonColor = new Color(173, 216, 230); // Пастельно-синий для кнопок  
 Font mainFont = new Font("SansSerif", Font.PLAIN, 14);  
  
 getContentPane().setBackground(backgroundColor);  
  
 GridBagConstraints gbc = new GridBagConstraints();  
 gbc.insets = new Insets(5, 5, 5, 5);  
  
 resultLabel = new JLabel("Результат: ");  
 resultLabel.setFont(new Font("SansSerif", Font.BOLD, 16));  
 gbc.gridx = 0;  
 gbc.gridy = 0;  
 gbc.gridwidth = 4;  
 gbc.anchor = GridBagConstraints.CENTER;  
 add(resultLabel, gbc);  
  
 gbc.gridwidth = 1;  
 gbc.anchor = GridBagConstraints.WEST;  
  
 gbc.gridx = 0;  
 gbc.gridy = 1;  
 JLabel firstNumberLabel = new JLabel("Первое комплексное число");  
 firstNumberLabel.setFont(mainFont);  
 add(firstNumberLabel, gbc);  
  
 gbc.gridx = 0;  
 gbc.gridy = 2;  
 JLabel real1Label = new JLabel("Вещественная часть:");  
 real1Label.setFont(mainFont);  
 add(real1Label, gbc);  
  
 real1Field = new JTextField(10);  
 real1Field.setFont(mainFont);  
 gbc.gridx = 1;  
 add(real1Field, gbc);  
  
 gbc.gridx = 0;  
 gbc.gridy = 3;  
 JLabel imaginary1Label = new JLabel("Мнимая часть:");  
 imaginary1Label.setFont(mainFont);  
 add(imaginary1Label, gbc);  
  
 imaginary1Field = new JTextField(10);  
 imaginary1Field.setFont(mainFont);  
 gbc.gridx = 1;  
 add(imaginary1Field, gbc);  
  
 gbc.gridx = 2;  
 gbc.gridy = 1;  
 JLabel secondNumberLabel = new JLabel("Второе комплексное число");  
 secondNumberLabel.setFont(mainFont);  
 add(secondNumberLabel, gbc);  
  
 gbc.gridx = 2;  
 gbc.gridy = 2;  
 JLabel real2Label = new JLabel("Вещественная часть:");  
 real2Label.setFont(mainFont);  
 add(real2Label, gbc);  
  
 real2Field = new JTextField(10);  
 real2Field.setFont(mainFont);  
 gbc.gridx = 3;  
 add(real2Field, gbc);  
  
 gbc.gridx = 2;  
 gbc.gridy = 3;  
 JLabel imaginary2Label = new JLabel("Мнимая часть:");  
 imaginary2Label.setFont(mainFont);  
 add(imaginary2Label, gbc);  
  
 imaginary2Field = new JTextField(10);  
 imaginary2Field.setFont(mainFont);  
 gbc.gridx = 3;  
 add(imaginary2Field, gbc);  
  
 JPanel buttonPanel = new JPanel(new FlowLayout(FlowLayout.CENTER, 10, 0));  
 buttonPanel.setBackground(backgroundColor); // Цвет панели кнопок  
  
 JButton addButton = new JButton("Сложение");  
 JButton subtractButton = new JButton("Вычитание");  
 JButton multiplyButton = new JButton("Умножение");  
  
 addButton.setBackground(buttonColor);  
 subtractButton.setBackground(buttonColor);  
 multiplyButton.setBackground(buttonColor);  
  
 addButton.setFont(mainFont);  
 subtractButton.setFont(mainFont);  
 multiplyButton.setFont(mainFont);  
  
 addButton.addActionListener(new OperationListener("+"));  
 subtractButton.addActionListener(new OperationListener("-"));  
 multiplyButton.addActionListener(new OperationListener("\*"));  
  
 buttonPanel.add(addButton);  
 buttonPanel.add(subtractButton);  
 buttonPanel.add(multiplyButton);  
  
 gbc.gridx = 0;  
 gbc.gridy = 4;  
 gbc.gridwidth = 4;  
 gbc.anchor = GridBagConstraints.CENTER;  
 add(buttonPanel, gbc);  
  
 setVisible(true);  
 }  
  
 private class OperationListener implements ActionListener {  
 private String operation;  
  
 public OperationListener(String operation) {  
 this.operation = operation;  
 }  
  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 try {  
 // Получаем вещественные и мнимые части  
 double real1 = Double.parseDouble(real1Field.getText());  
 double imaginary1 = Double.parseDouble(imaginary1Field.getText());  
 double real2 = Double.parseDouble(real2Field.getText());  
 double imaginary2 = Double.parseDouble(imaginary2Field.getText());  
  
 Complex c1 = new Complex(real1, imaginary1);  
 Complex c2 = new Complex(real2, imaginary2);  
 Complex result;  
  
 switch (operation) {  
 case "+":  
 result = c1.add(c2);  
 break;  
 case "-":  
 result = c1.subtract(c2);  
 break;  
 case "\*":  
 result = c1.multiply(c2);  
 break;  
 default:  
 throw new IllegalArgumentException("Некорректная операция");  
 }  
  
 resultLabel.setText("Результат: " + result);  
 } catch (NumberFormatException ex) {  
 JOptionPane.showMessageDialog(null, "Пожалуйста, введите корректные числа.", "Ошибка ввода", JOptionPane.ERROR\_MESSAGE);  
 }  
 }  
 }  
  
 public static void main(String[] args) {  
 SwingUtilities.invokeLater(ComplexCalculator::new);  
 }  
}  
  
class Complex {  
 private double real;  
 private double imaginary;  
  
 public Complex(double real, double imaginary) {  
 this.real = real;  
 this.imaginary = imaginary;  
 }  
  
 public Complex add(Complex other) {  
 return new Complex(this.real + other.real, this.imaginary + other.imaginary);  
 }  
  
 public Complex subtract(Complex other) {  
 return new Complex(this.real - other.real, this.imaginary - other.imaginary);  
 }  
  
 public Complex multiply(Complex other) {  
 double newReal = this.real \* other.real - this.imaginary \* other.imaginary;  
 double newImaginary = this.real \* other.imaginary + this.imaginary \* other.real;  
 return new Complex(newReal, newImaginary);  
 }  
  
 @Override  
 public String toString() {  
 if (imaginary >= 0) {  
 return real + " + " + imaginary + "i";  
 } else {  
 return real + " - " + (-imaginary) + "i";  
 }  
 }  
}

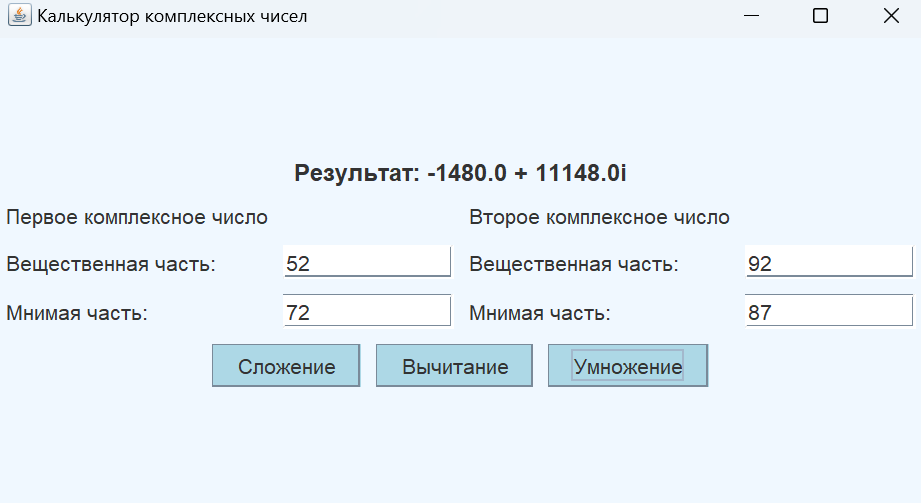


Рисунок 8 – результат задания №8

Задание №9

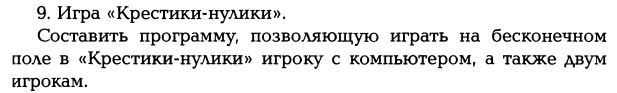
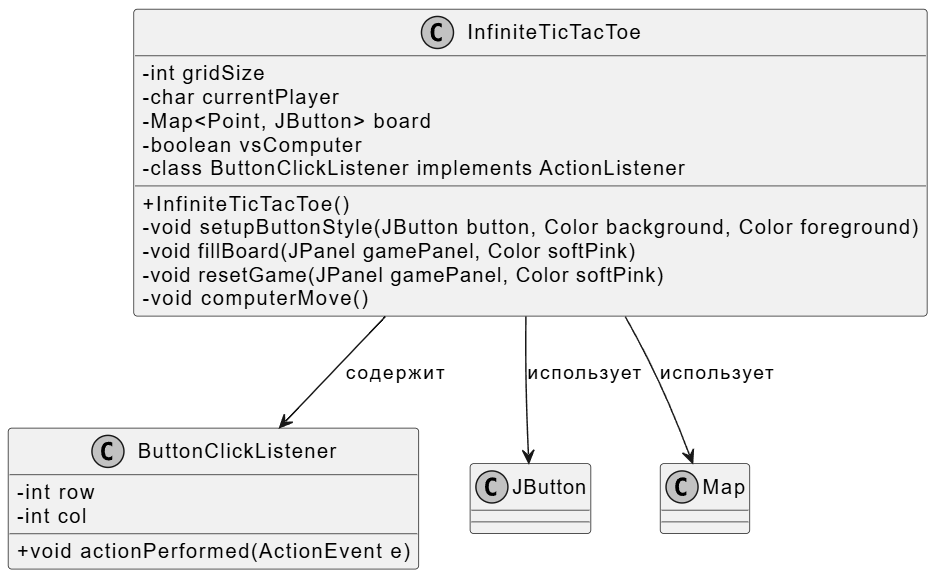


Диаграмма классов



Код программы

import javax.swing.\*;

import javax.swing.border.LineBorder;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.HashMap;

import java.util.Map;

public class InfiniteTicTacToe extends JFrame {

private int gridSize = 5; // Начальный размер сетки

private char currentPlayer = 'X'; // 'X' ходит первым

private Map<Point, JButton> board = new HashMap<>();

private boolean vsComputer = false; // Режим против компьютера

public InfiniteTicTacToe() {

setTitle("Крестики-Нолики на бесконечном поле");

setSize(700, 700);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

Color mintGreen = new Color(189, 236, 182); // Нежно-мятный

Color softPink = new Color(255, 210, 224); // Нежно-розовый

JPanel mainPanel = new JPanel();

mainPanel.setLayout(new BorderLayout());

mainPanel.setBackground(mintGreen);

JPanel gamePanel = new JPanel(new GridLayout(gridSize, gridSize));

mainPanel.add(gamePanel, BorderLayout.CENTER);

gamePanel.setBackground(mintGreen);

fillBoard(gamePanel, softPink);

JPanel controlPanel = new JPanel();

controlPanel.setBackground(mintGreen);

JButton newGameButton = new JButton("Новая Игра");

JButton playerVsPlayerButton = new JButton("2 Игрока");

JButton playerVsComputerButton = new JButton("Против Компьютера");

setupButtonStyle(newGameButton, mintGreen, softPink);

setupButtonStyle(playerVsPlayerButton, mintGreen, softPink);

setupButtonStyle(playerVsComputerButton, mintGreen, softPink);

controlPanel.add(newGameButton);

controlPanel.add(playerVsPlayerButton);

controlPanel.add(playerVsComputerButton);

mainPanel.add(controlPanel, BorderLayout.SOUTH);

add(mainPanel);

newGameButton.addActionListener(e -> resetGame(gamePanel, softPink));

playerVsPlayerButton.addActionListener(e -> {

vsComputer = false;

resetGame(gamePanel, softPink);

});

playerVsComputerButton.addActionListener(e -> {

vsComputer = true;

resetGame(gamePanel, softPink);

if (vsComputer && currentPlayer == 'O') {

computerMove();

}

});

setVisible(true);

}

private void setupButtonStyle(JButton button, Color background, Color foreground) {

button.setFont(new Font("Serif", Font.BOLD, 20));

button.setBackground(foreground);

button.setForeground(Color.BLACK);

button.setFocusPainted(false);

button.setBorderPainted(false);

}

private void fillBoard(JPanel gamePanel, Color softPink) {

gamePanel.removeAll();

board.clear();

gamePanel.setLayout(new GridLayout(gridSize, gridSize));

for (int row = 0; row < gridSize; row++) {

for (int col = 0; col < gridSize; col++) {

JButton button = new JButton("");

button.setFont(new Font("Serif", Font.BOLD, 30)); // Красивый шрифт

button.setBackground(softPink); // Нежно-розовый фон

button.setFocusPainted(false);

button.setBorder(new LineBorder(Color.BLACK, 1)); // Черная граница для создания сетки

button.addActionListener(new ButtonClickListener(row, col));

board.put(new Point(row, col), button);

gamePanel.add(button);

}

}

gamePanel.revalidate();

gamePanel.repaint();

}

private void resetGame(JPanel gamePanel, Color softPink) {

currentPlayer = 'X';

gridSize = 5;

fillBoard(gamePanel, softPink);

}

private class ButtonClickListener implements ActionListener {

private final int row;

private final int col;

public ButtonClickListener(int row, int col) {

this.row = row;

this.col = col;

}

@Override

public void actionPerformed(ActionEvent e) {

JButton button = (JButton) e.getSource();

if (!button.getText().equals("")) return; // Ячейка уже занята

button.setText(String.valueOf(currentPlayer));

if (checkWin(row, col)) {

JOptionPane.showMessageDialog(null, "Игрок " + currentPlayer + " выиграл!");

resetGame((JPanel) getContentPane().getComponent(0), new Color(255, 210, 224));

return;

}

currentPlayer = (currentPlayer == 'X') ? 'O' : 'X';

if (vsComputer && currentPlayer == 'O') {

computerMove();

}

if (isBorderFilled()) {

expandBoard((JPanel) getContentPane().getComponent(0));

}

}

}

private boolean checkWin(int row, int col) {

String symbol = String.valueOf(currentPlayer);

return (checkLine(row, 0, 0, 1, symbol) || // Проверка строки

checkLine(0, col, 1, 0, symbol) || // Проверка столбца

checkLine(0, 0, 1, 1, symbol) || // Диагональ слева направо

checkLine(0, gridSize - 1, 1, -1, symbol)); // Диагональ справа налево

}

private boolean checkLine(int startRow, int startCol, int rowIncrement, int colIncrement, String symbol) {

for (int i = 0; i < gridSize; i++) {

JButton button = board.get(new Point(startRow + i \* rowIncrement, startCol + i \* colIncrement));

if (button == null || !button.getText().equals(symbol)) {

return false;

}

}

return true;

}

private boolean isBorderFilled() {

for (int i = 0; i < gridSize; i++) {

if (board.get(new Point(0, i)).getText().equals("") ||

board.get(new Point(gridSize - 1, i)).getText().equals("") ||

board.get(new Point(i, 0)).getText().equals("") ||

board.get(new Point(i, gridSize - 1)).getText().equals("")) {

return false;

}

}

return true;

}

private void expandBoard(JPanel gamePanel) {

gridSize += 2;

fillBoard(gamePanel, new Color(255, 210, 224));

}

private void computerMove() {

for (Map.Entry<Point, JButton> entry : board.entrySet()) {

JButton button = entry.getValue();

if (button.getText().equals("")) {

button.setText(String.valueOf(currentPlayer));

currentPlayer = 'X';

return;

}

}

}

public static void main(String[] args) {

SwingUtilities.invokeLater(InfiniteTicTacToe::new);

}

}

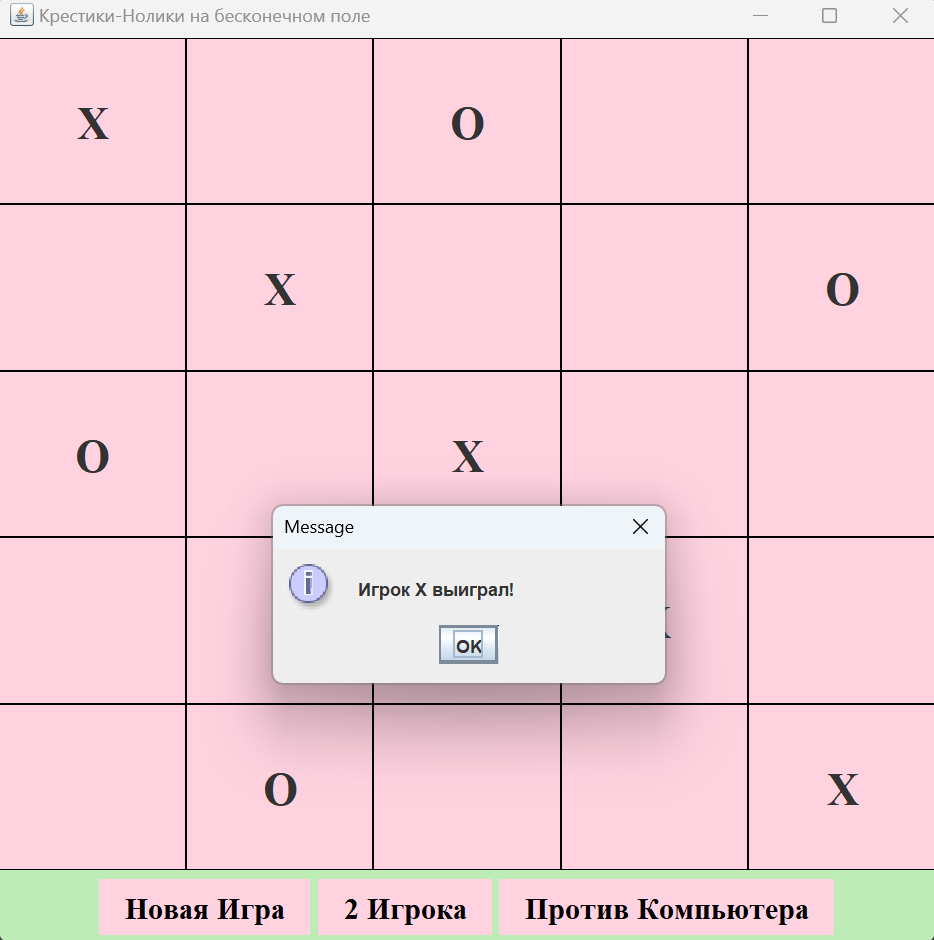


Рисунок 9 – результат задания №9

Ассемблер

* 1. Задача 1

Код программы

section .data

N dq 8 ; Исходное значение N (64-битное число)

msg db "Результат: ", 0 ; Сообщение для вывода

len equ $ - msg ; Длина сообщения

buffer db "0000000000000000", 0 ; Буфер для преобразованного числа (16 символов для 64-битного числа)

section .text

global \_start

\_start:

; Загружаем N в регистр RAX и умножаем на 2

mov rax, [N] ; Загружаем значение N в регистр RAX

shl rax, 1 ; Умножаем на 2 сдвигом влево

mov [N], rax ; Сохраняем результат обратно в N

; Преобразуем результат в строку

mov rdi, buffer + 15 ; Указатель на конец буфера

mov rcx, 10 ; Делитель (10 для десятичного представления)

convert\_to\_string:

xor rdx, rdx ; Очищаем регистр RDX для деления

div rcx ; Делим RAX на 10, результат в RAX, остаток в RDX

add dl, '0' ; Преобразуем остаток в ASCII

mov [rdi], dl ; Сохраняем символ в буфере

dec rdi ; Перемещаем указатель назад

test rax, rax ; Проверяем, закончилось ли деление

jnz convert\_to\_string ; Если нет, продолжаем деление

; Рассчитываем длину строки для вывода

mov rsi, rdi ; Сохраняем начальный указатель в rsi

mov rdi, buffer + 16 ; Конец буфера

sub rdi, rsi ; Вычисляем длину значащих цифр в rdi

; Выводим сообщение "Результат: " на экран

mov eax, 1 ; Системный вызов для записи (sys\_write)

mov edi, 1 ; Номер файла (1 = stdout)

mov rsi, msg ; Указатель на сообщение

mov edx, len ; Длина сообщения

syscall ; Вызов системного прерывания

; Выводим результат (содержимое буфера) на экран

mov eax, 1 ; Системный вызов для записи (sys\_write)

mov edi, 1 ; Номер файла (1 = stdout)

mov rsi, rsi ; Указатель на начало значащих цифр

mov edx, rdi ; Длина значащих цифр

syscall ; Вызов системного прерывания

; Завершаем программу

mov eax, 60 ; Код системного вызова для выхода (sys\_exit)

xor edi, edi ; Код завершения 0

syscall

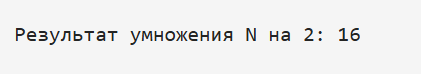


Рисунок 10 – результат задачи 1

* 1. Задача 2

Код программы

section .data

msg db "Результат выполнения x := x - a: ", 0

len equ $ - msg ; Длина строки сообщения

buffer db "00000000", 0 ; Буфер для вывода результата (макс 8 символов)

section .text

global \_start

\_start:

; Пример: вызываем psub с x = 10 и a = 3

; Пушим параметры на стек: сначала x, затем a

mov rax, 10 ; Исходное значение x

push rax

mov rax, 3 ; Значение a

push rax

; Вызов процедуры psub

call psub

; Результат вычитания остается в стеке, извлекаем его

pop rax ; Результат теперь в rax

; Преобразуем результат в строку для вывода

mov rdi, buffer + 7 ; Указатель на конец буфера

mov rcx, 10 ; Делитель для преобразования в строку

convert\_to\_string:

xor rdx, rdx ; Очищаем rdx для div

div rcx ; Делим rax на 10, результат в rax, остаток в rdx

add dl, '0' ; Преобразуем остаток в ASCII

mov [rdi], dl ; Сохраняем символ в буфере

dec rdi ; Смещаемся влево по буферу

test rax, rax ; Проверяем, закончилось ли деление

jnz convert\_to\_string ; Продолжаем, если есть еще цифры

mov rsi, rdi ; Начало строки с числом

mov rdi, buffer + 8 ; Конец строки

sub rdi, rsi ; Длина строки

mov eax, 1 ; sys\_write

mov edi, 1 ; дескриптор stdout

mov rsi, msg ; указатель на сообщение

mov edx, len ; длина сообщения

syscall ; Вызов

mov eax, 1 ; sys\_write

mov edi, 1 ; дескриптор stdout

mov rsi, rsi ; указатель на строку числа

mov edx, rdi ; длина строки числа

syscall ; Вызов

mov eax, 60 ; sys\_exit

xor edi, edi ; Код возврата 0

syscall

; Процедура psub: выполняет x := x - a

psub:

; Сохраняем значения регистров, если необходимо

push rbp

mov rbp, rsp

; Загружаем параметры из стека

mov rax, [rbp+16] ; Параметр x (первый)

mov rcx, [rbp+24] ; Параметр a (второй)

; Вычитание x - a

sub rax, rcx ; x := x - a

; Возвращаем результат в стек

push rax

; Восстанавливаем и выходим

pop rbp

ret



Рисунок 11 – результат задачи 2

1. Разработка АИС «Электронная почта»

Введение

Автоматизированная информационная система (АИС) «Электронная почта» предназначена для организации и автоматизации обмена сообщениями между пользователями в рамках одной организации. Система позволяет пользователям обмениваться электронными письмами с возможностью указывать тему сообщения и содержание, обеспечивая структурированный и безопасный канал коммуникации.

Цель создания: Обеспечить удобное, надежное и защищенное средство обмена текстовыми сообщениями между сотрудниками для упрощения деловой переписки и координации действий.

* 1. Диаграммы

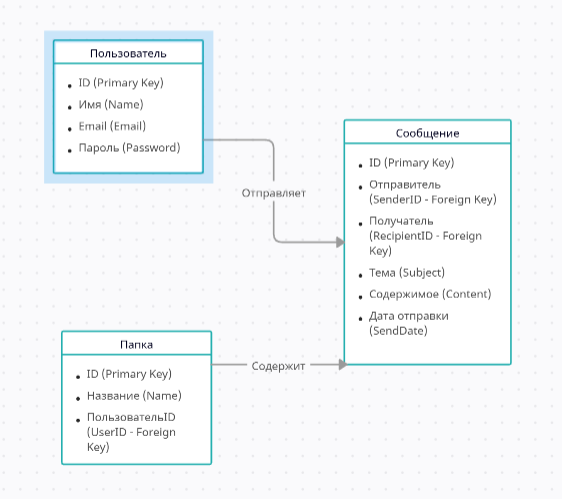


Рисунок 12 – ER диаграмма

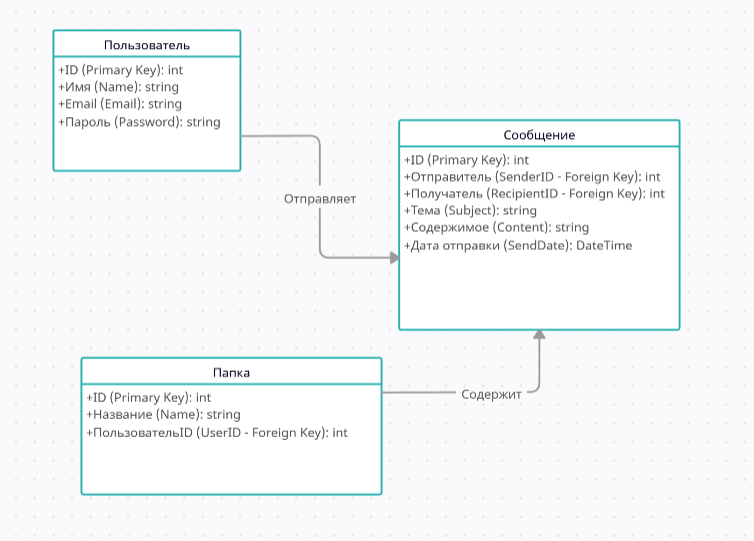


Рисунок 13 – диаграмма классов

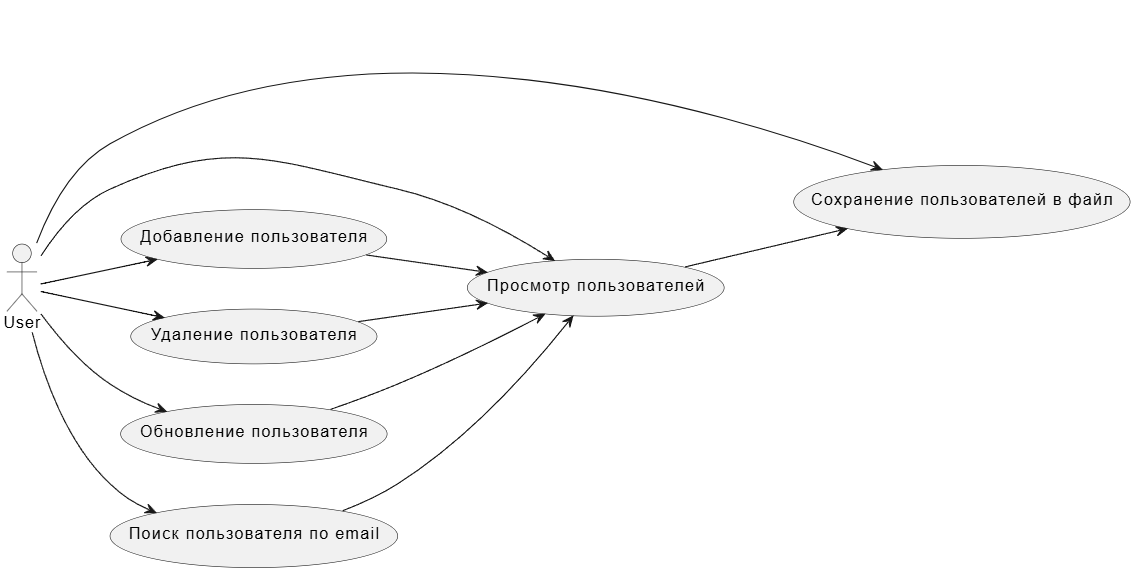


Рисунок 14 – диаграмма вариантов использования



Рисунок 15 – диаграмма последовательностей



Рисунок 16 – диаграмма состояний

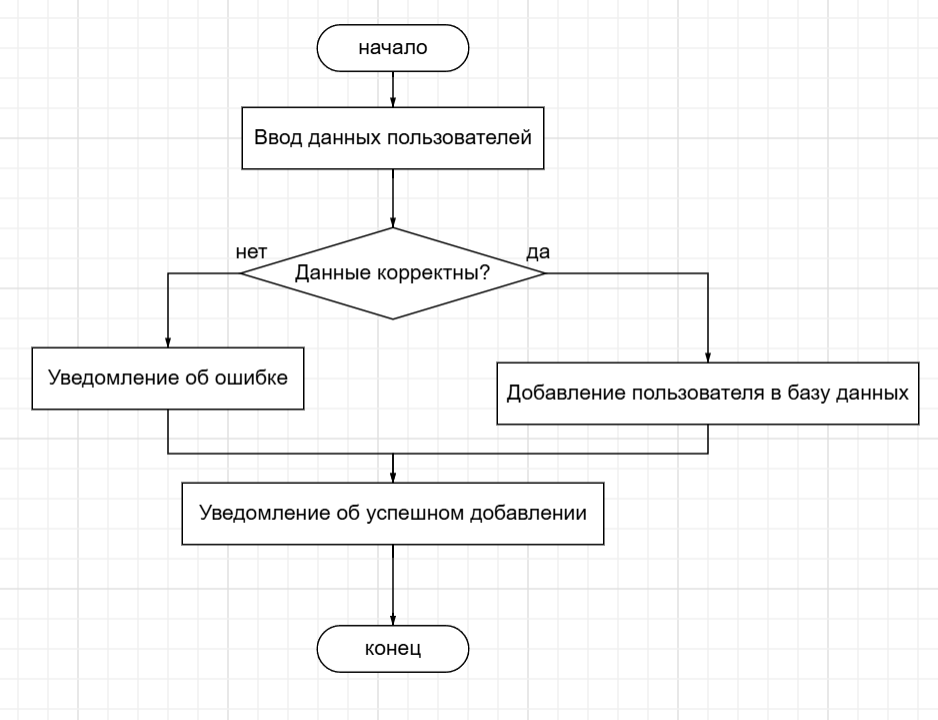


Рисунок 17 – блок схема алгоритма

* 1. Техническое задание

Область применения

АИС «Электронная почта» может быть использована в различных организациях, таких как:

Корпоративная среда: для внутренней деловой переписки между сотрудниками, рассылки уведомлений и координации совместных задач.

Образовательные учреждения: для связи между преподавателями и студентами, а также для административных уведомлений.

Клиентский сервис: для обработки входящих запросов, консультаций и общения с клиентами.

Государственные и общественные организации: для оповещений, координации мероприятий и взаимодействия между подразделениями.

Функциональные требования

Регистрация пользователей:

* Возможность создания нового пользователя с указанием имени пользователя, email и пароля.
* Поле email должно идти сразу после поля Имя пользователя.
* Система должна уведомлять о успешной регистрации пользователя.

Авторизация и доступ:

* Осуществлять проверку учетных данных при входе в систему для предоставления доступа.
* Возможность разграничения прав доступа (опционально для будущих версий).

Отправка сообщений:

* Пользователь может отправить сообщение другому зарегистрированному пользователю, указав отправителя, получателя, тему и текст сообщения.
* При успешной отправке сообщения должно отображаться подтверждение об отправке.

Просмотр сообщений:

* Пользователь может просматривать полученные сообщения.
* Сообщения должны отображаться с указанием отправителя, темы и текста, а также времени отправки.

Управление пользователями и сообщениями:

* Возможность хранения и управления списком пользователей и сообщений в рамках приложения.
* Возможность удаления сообщений (опционально для будущих версий).

Нефункциональные требования

Интерфейс:

* Окна интерфейса должны быть выполнены в пастельных тонах, с использованием нежного розового и мятного цветов.
* Удобная навигация между разделами (регистрация, отправка сообщений, просмотр сообщений).

Безопасность:

* Доступ к отправке сообщений и просмотру почты должен быть ограничен только зарегистрированными пользователями.
* Хранение паролей пользователей в зашифрованном виде (опционально для будущих версий).

Масштабируемость:

* Код должен быть написан таким образом, чтобы его можно было расширить для реализации дополнительных функций, таких как вложения в письмах, статус прочтения, фильтры и т. д.

Производительность:

Операции отправки и получения сообщений должны происходить быстро, без задержек.

Кроссплатформенность:

Программа должна корректно работать на системах Windows, MacOS и Linux с установленной JVM.

Структура и модульное деление

Главное окно: основное окно, предоставляющее доступ к функциям системы.

Регистрация: модуль для создания новых пользователей.

Отправка сообщения: модуль для формирования и отправки писем.

Просмотр сообщений: модуль для отображения полученных сообщений.

Основные алгоритмы

Регистрация пользователя:

* Проверка введенных данных.
* Добавление нового пользователя в список зарегистрированных пользователей и подтверждение об успешной регистрации.

Отправка сообщения:

* Проверка наличия получателя в базе данных пользователей.
* Сохранение отправленного сообщения в список сообщений.

Просмотр сообщений:

* Извлечение всех сообщений, предназначенных для текущего пользователя.
* Отображение списка сообщений с возможностью просмотреть детали.
* Требования к интерфейсу
* Интерфейс должен быть разработан с помощью Swing.
* Главные элементы (кнопки, поля ввода) должны быть оформлены в пастельных тонах, с основными цветами: нежно-розовый и мятный.
* Важные кнопки, такие как «Регистрация», «Отправить» и «Просмотр», должны быть заметными и интуитивно понятными.
  1. Тестирование

Провести функциональное тестирование для проверки работы всех основных функций: регистрация, отправка сообщений, просмотр сообщений.

Проверить корректность отображения элементов интерфейса.

Провести тестирование с разными пользователями, включая тесты на ввод невалидных данных.

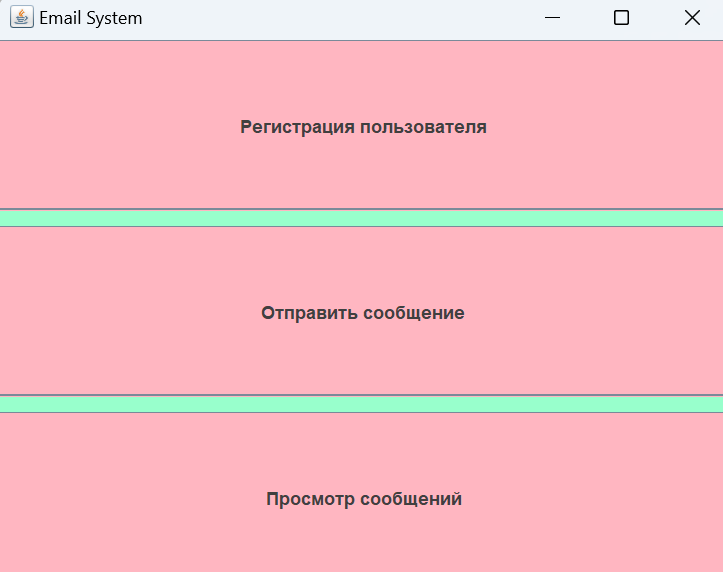


Рисунок 18 – меню АИС «Электронная почта»

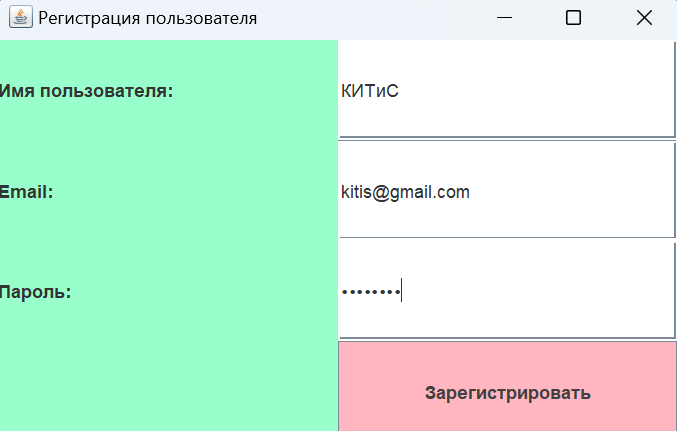


Рисунок 19 – регистрация пользователя

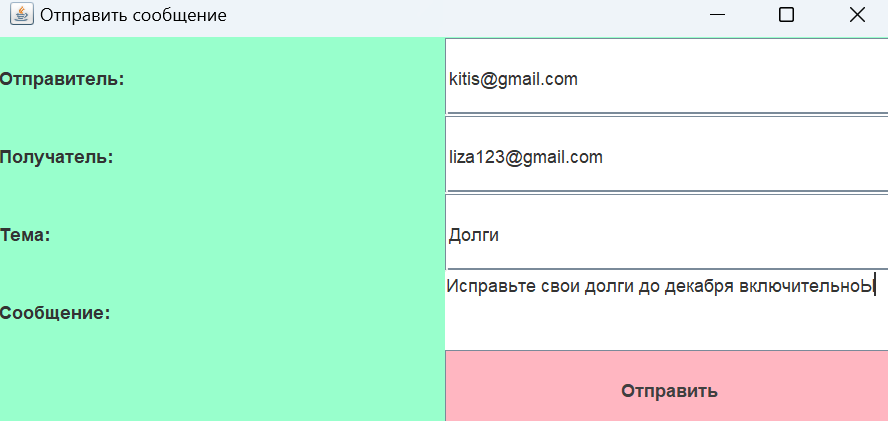


Рисунок 20 – отправка сообщения

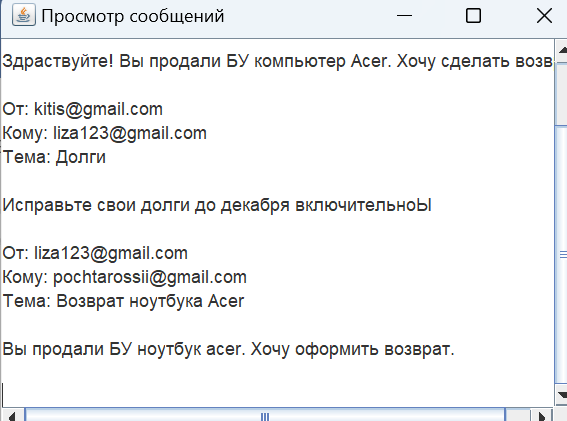


Рисунок 21 – просмотр сообщений

Также, разработка АИС «Электронная почта» должна быть проведена на 1С: Предприятие. И вот что получилось:

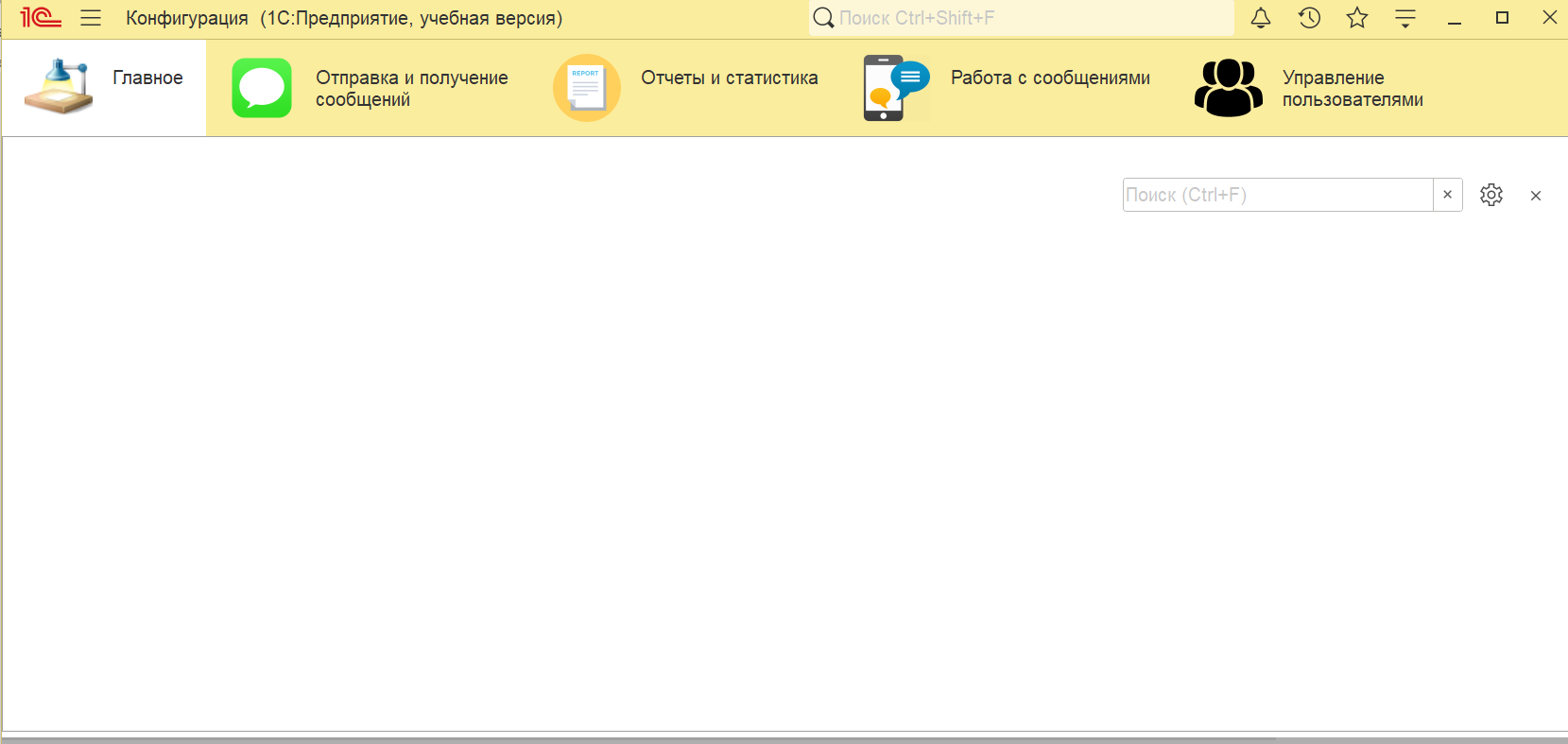


Рисунок 22 – Главный экран

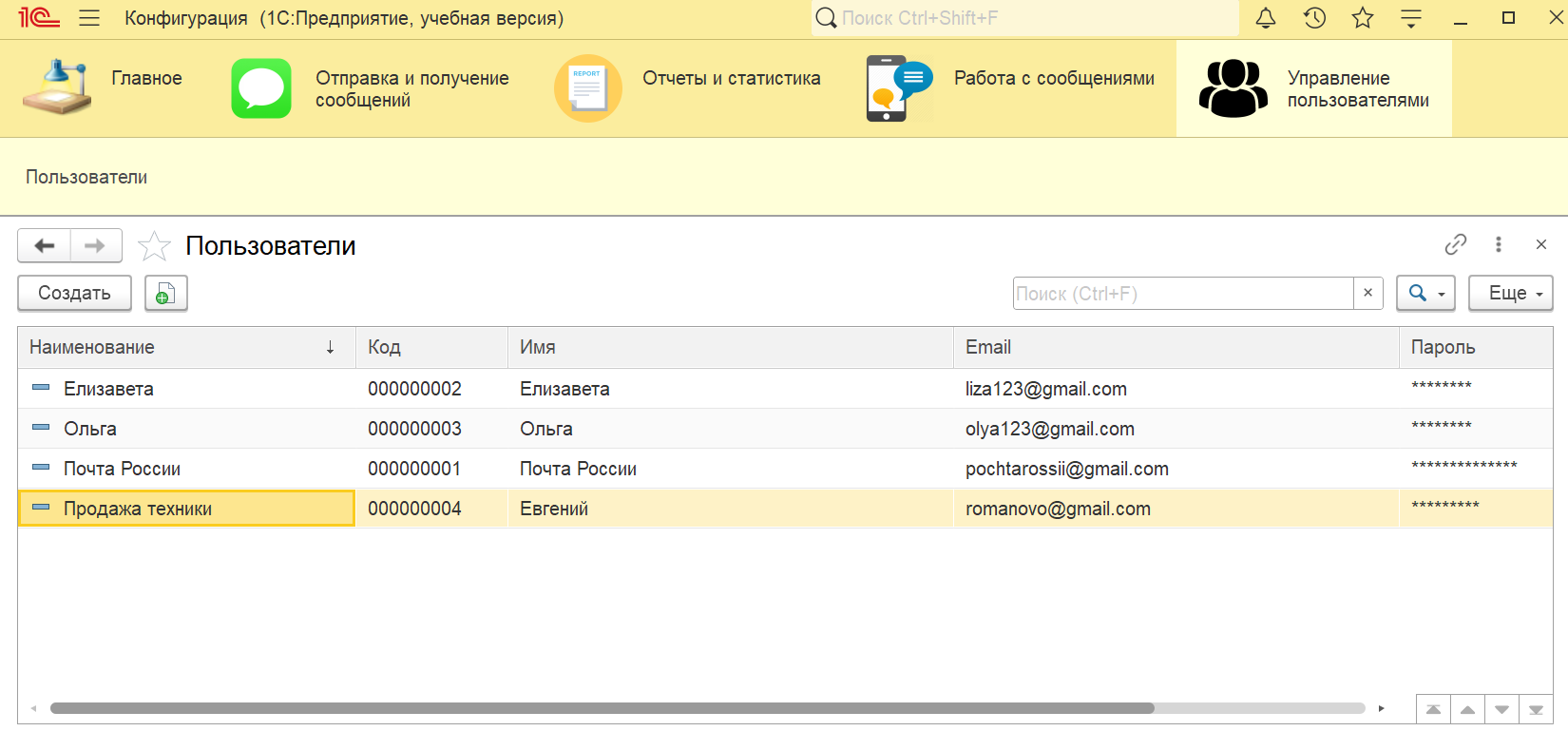


Рисунок 23 – Вкладка «Управление пользователями»

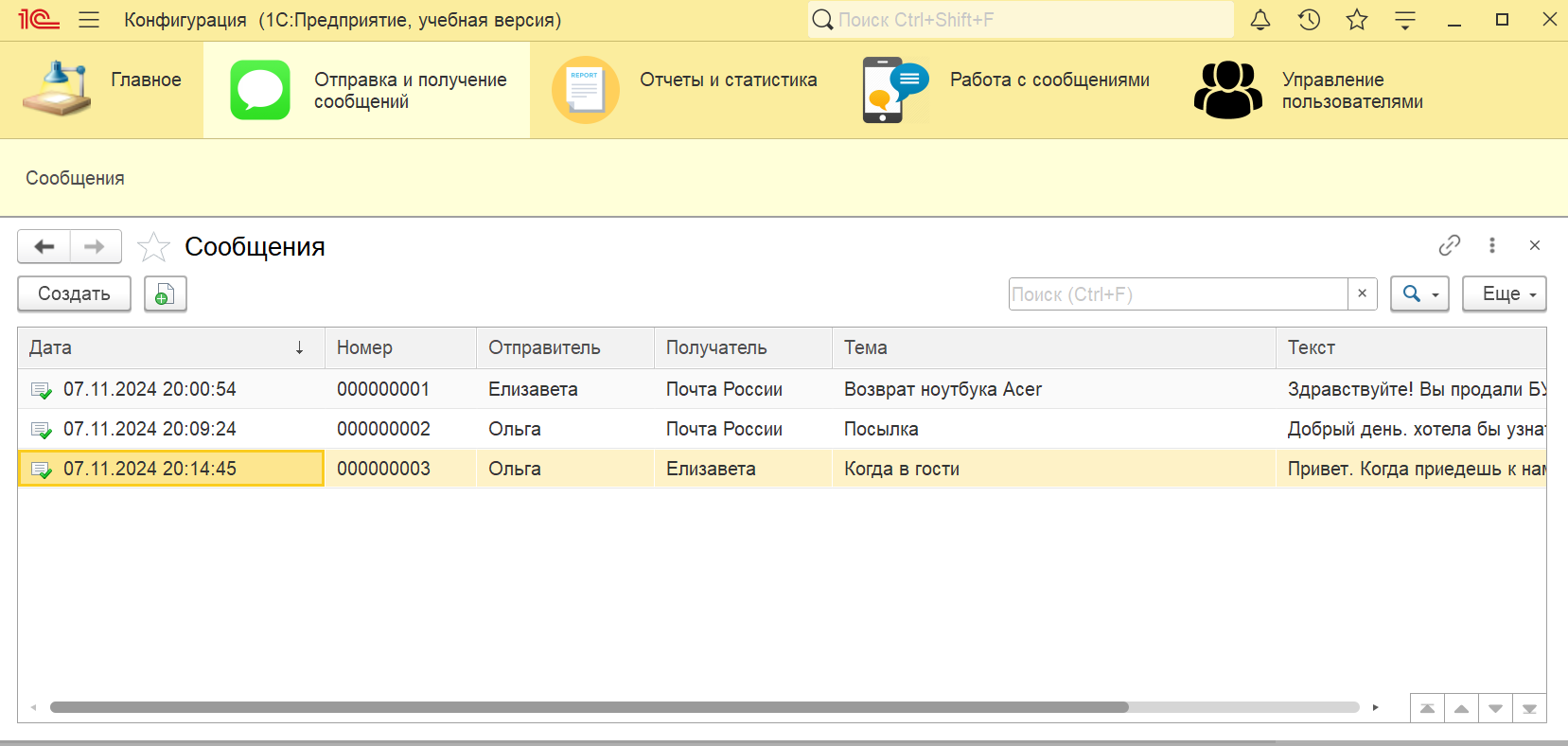


Рисунок 24 – Вкладка «Отправка и получение сообщений»

1. Разработка мобильного приложения

Мобильное приложение «Электронная почта» будет включать следующие основные функциональные возможности:

Регистрация пользователей:

* Пользователи смогут зарегистрироваться, указав свое имя, адрес электронной почты и пароль.
* Регистрация обеспечит доступ к личному почтовому ящику и всем функциям приложения.

Вход в почтовый ящик:

* Зарегистрированные пользователи смогут входить в свой почтовый ящик, используя адрес электронной почты и пароль.
* Вход в систему предоставит доступ ко всем полученным и отправленным письмам.

Отправка сообщений:

* Пользователи смогут создавать и отправлять новые электронные письма.
* Для отправки письма потребуется указать получателя, тему и текст сообщения.

Просмотр сообщений:

* Пользователи смогут просматривать полученные электронные письма.
* Будет доступен список всех входящих и отправленных сообщений.
* Пользователи смогут читать содержимое каждого письма.

Основные параметры:

Название приложения: Электронная почта  
 Платформа: Android Studio  
 Язык программирования: Java  
 Поддерживаемые устройства: Смартфоны и планшеты на Android

Об Android Studio:  
Android Studio является официальной интегрированной средой разработки (IDE) для Android, предоставляемой Google. Она основана на IntelliJ IDEA и предлагает мощные инструменты для создания, тестирования и отладки приложений. Android Studio поддерживает различные функции, такие как редактор кода с подсветкой синтаксиса, инструменты для визуального проектирования интерфейса, эмулятор Android для тестирования приложений на виртуальных устройствах и интеграцию с системами контроля версий. Это позволяет разработчикам эффективно управлять проектами и оптимизировать процесс разработки.

Разрешения:  
Приложение требует следующих разрешений:

Доступ в интернет: Необходим для отправки и получения электронных писем, а также для подключения к удалённым серверам и API.

Доступ к хранилищу: Позволяет пользователям загружать файлы, прикреплять их к письмам и сохранять вложения на устройстве.

Функции безопасности:

Защита данных: Вся информация о пользователях и их задачах хранится локально в базе данных SQLite на устройстве. Это предотвращает несанкционированный доступ к данным без физического доступа к устройству. Для повышения безопасности можно использовать шифрование данных.

Аутентификация: Пользователи проходят процесс аутентификации, вводя имя пользователя и пароль. Возможна реализация двухфакторной аутентификации для дополнительной защиты.

Реализация регистрации и авторизации:

Регистрация: Разработан экран для регистрации, где пользователи могут создать учетную запись, указав имя пользователя, email, пароль. При успешной регистрации данные сохраняются в базе данных SQLite, что обеспечивает надежное хранение информации.

Авторизация: После регистрации пользователи могут войти в систему, используя свои учетные данные. Реализована возможность восстановления пароля через электронную почту, что упрощает процесс восстановления доступа к учетной записи.

Дополнительные функции:

Интуитивно понятный интерфейс: Приложение имеет простой и удобный интерфейс, который позволяет пользователям легко навигировать между функциями.

Поддержка нескольких учетных записей: Пользователи могут добавлять и управлять несколькими учетными записями электронной почты в одном приложении.

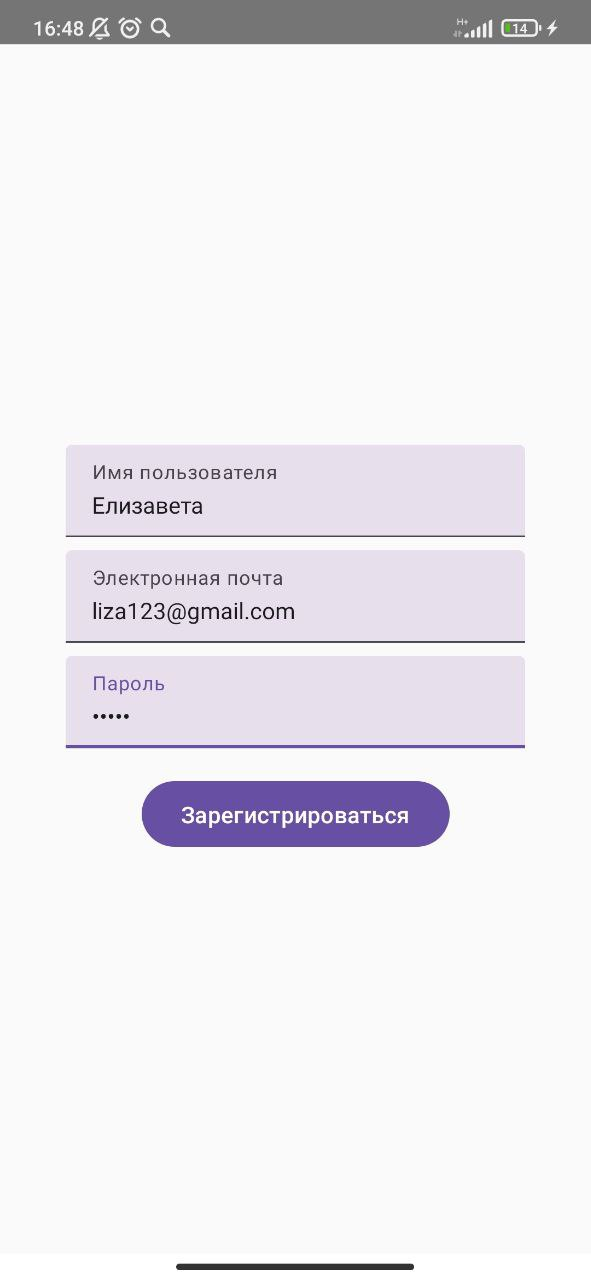


Рисунок 25 – регистрация пользователя

Это страница регистрации для мобильного приложения электронной почты. На ней пользователь может ввести свое имя, адрес электронной почты и пароль, а затем нажать кнопку «Зарегистрироваться» для создания учетной записи.

Основные функциональные возможности этого приложения будут включать:

* Регистрация пользователей с указанием имени, электронной почты и пароля.
* Вход в почтовый ящик с использованием зарегистрированных учетных данных.
* Отправка новых электронных писем.
* Просмотр полученных и отправленных сообщений.

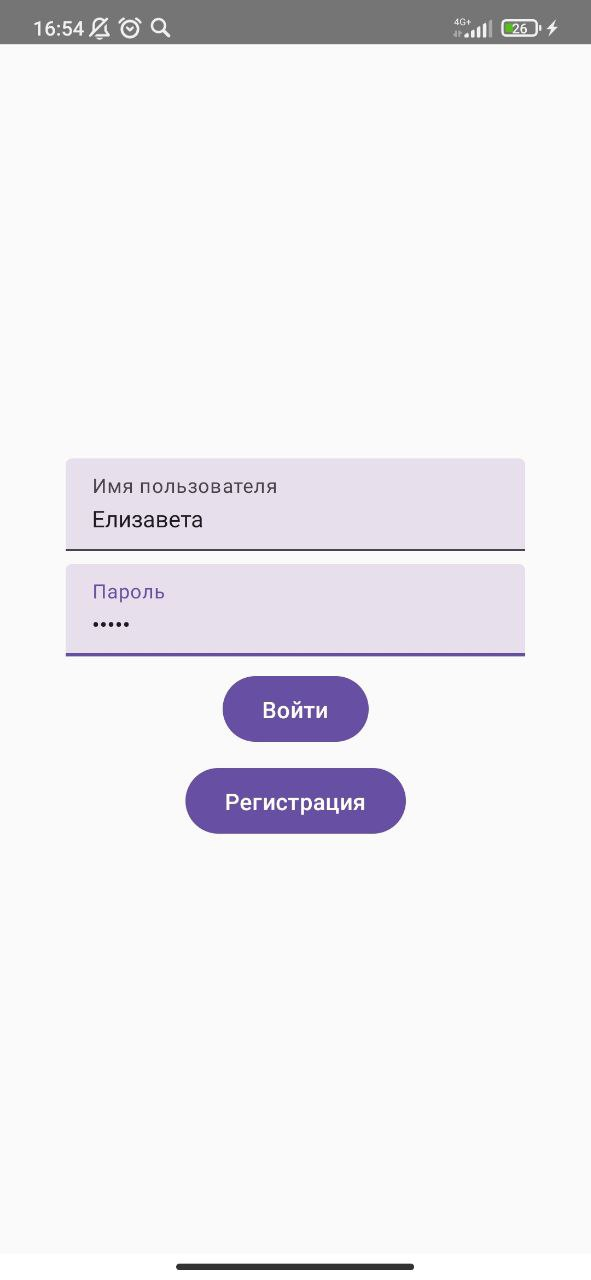


Рисунок 26 – вход на почту пользователя

Это страница входа в мобильное приложение электронной почты. На ней пользователь может ввести свое имя пользователя и пароль, а затем нажать кнопку «Войти» для входа в свой почтовый ящик.

Также на этой странице есть кнопка «Регистрация», которая позволит новым пользователям создать учетную запись в приложении.

Основные функции этого мобильного приложения для электронной почты будут включать:

* Вход в почтовый ящик с использованием зарегистрированных учетных данных.
* Просмотр полученных и отправленных сообщений.
* Создание и отправка новых электронных писем.
* Регистрация новых пользователей.

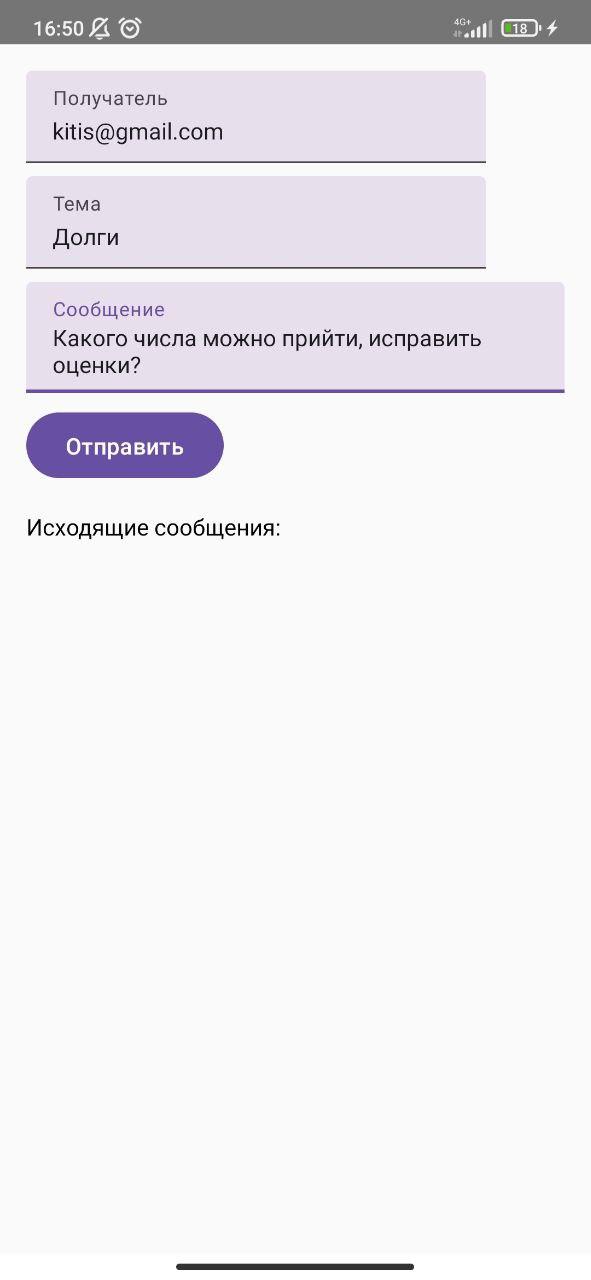


Рисунок 27 – отправка сообщений

Это черновик электронного письма, который пользователь собирается отправить.

Основные детали письма:

* Получатель
* Тема
* Сообщение

1. Разработка сайта

Для создания сайта использовалась система управления содержимым CMS

Система управления содержимым (CMS, от английского Content Management System) — это программное обеспечение, которое позволяет пользователям создавать, редактировать, управлять и публиковать цифровой контент на веб-сайтах без необходимости глубоких технических знаний в области программирования. CMS значительно упрощает процесс разработки и управления веб-контентом, предоставляя интуитивно понятный интерфейс и инструменты для работы с данными.

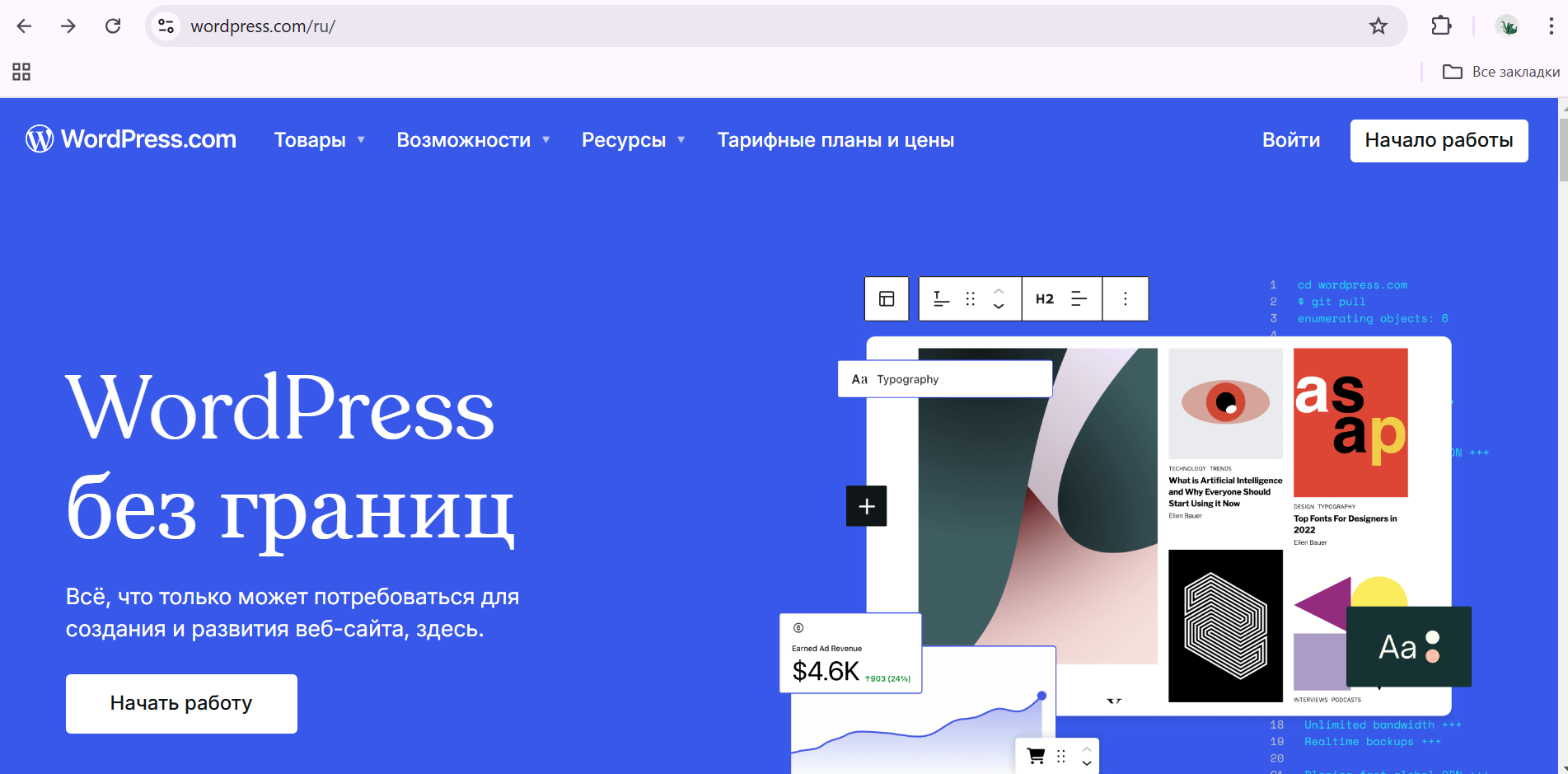


Рисунок 28 – Главная страница сайта WordPress

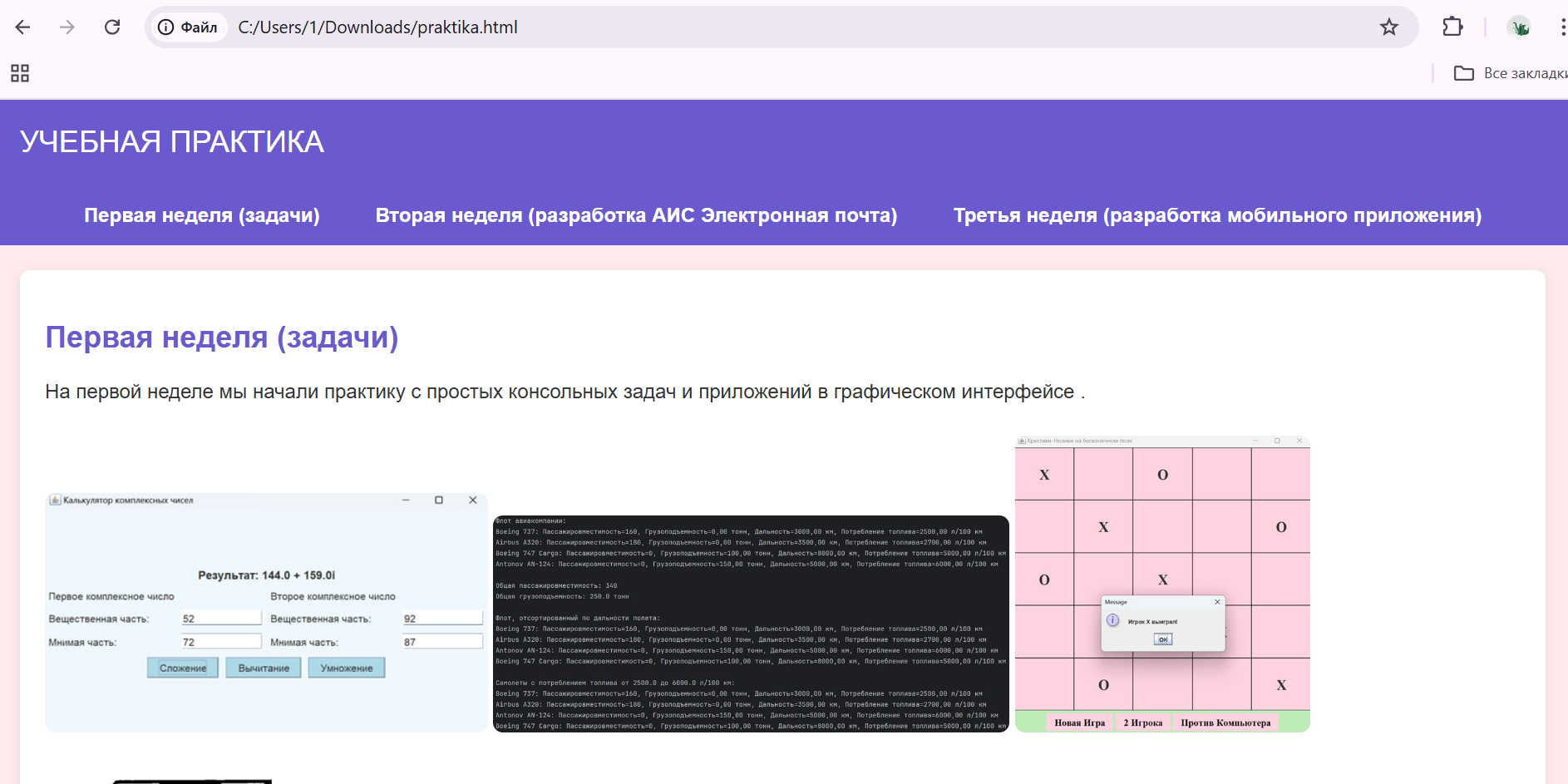


Рисунок 29 – Вкладка первая неделя

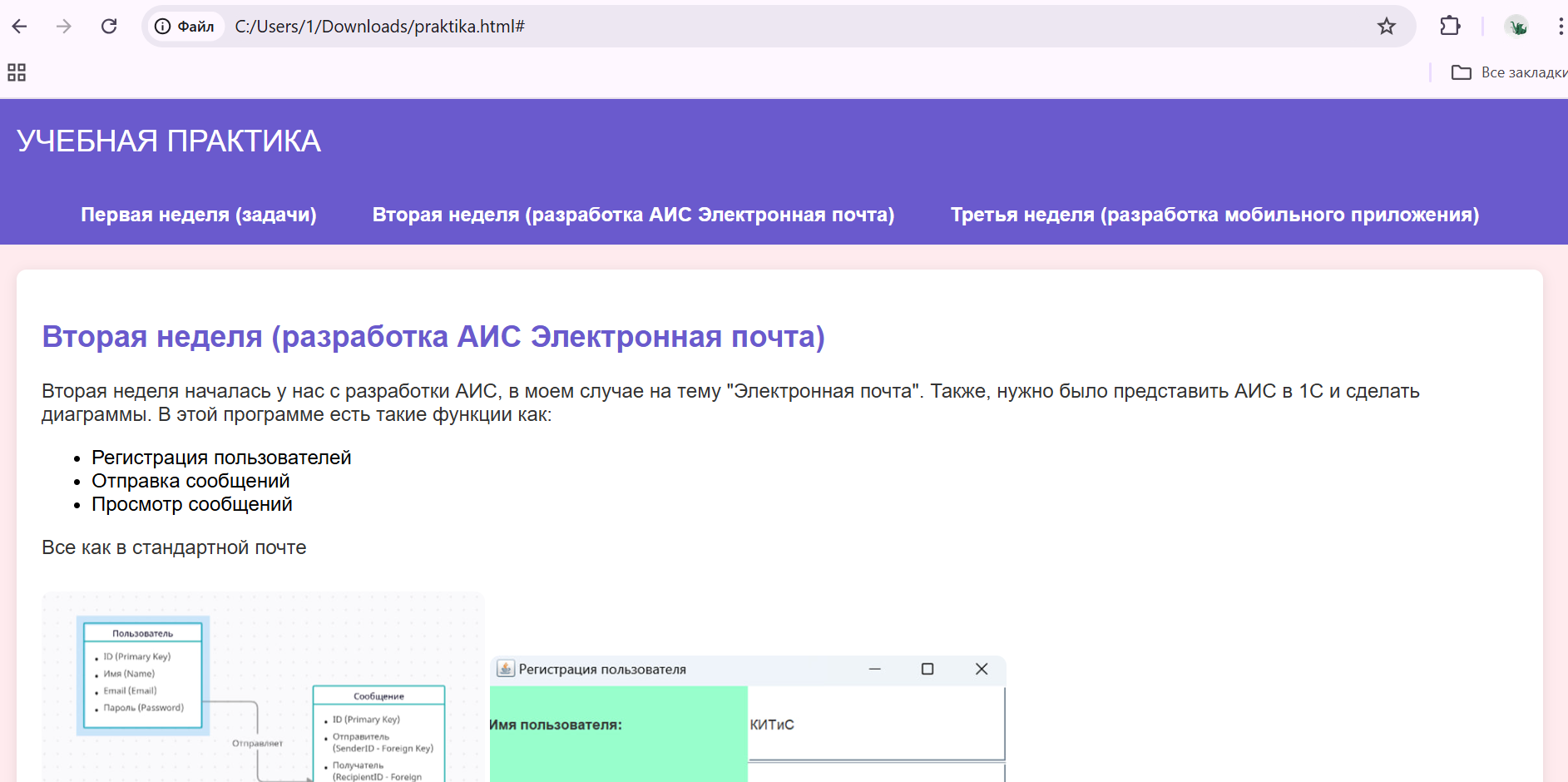


Рисунок 30 – Вкладка вторая неделя

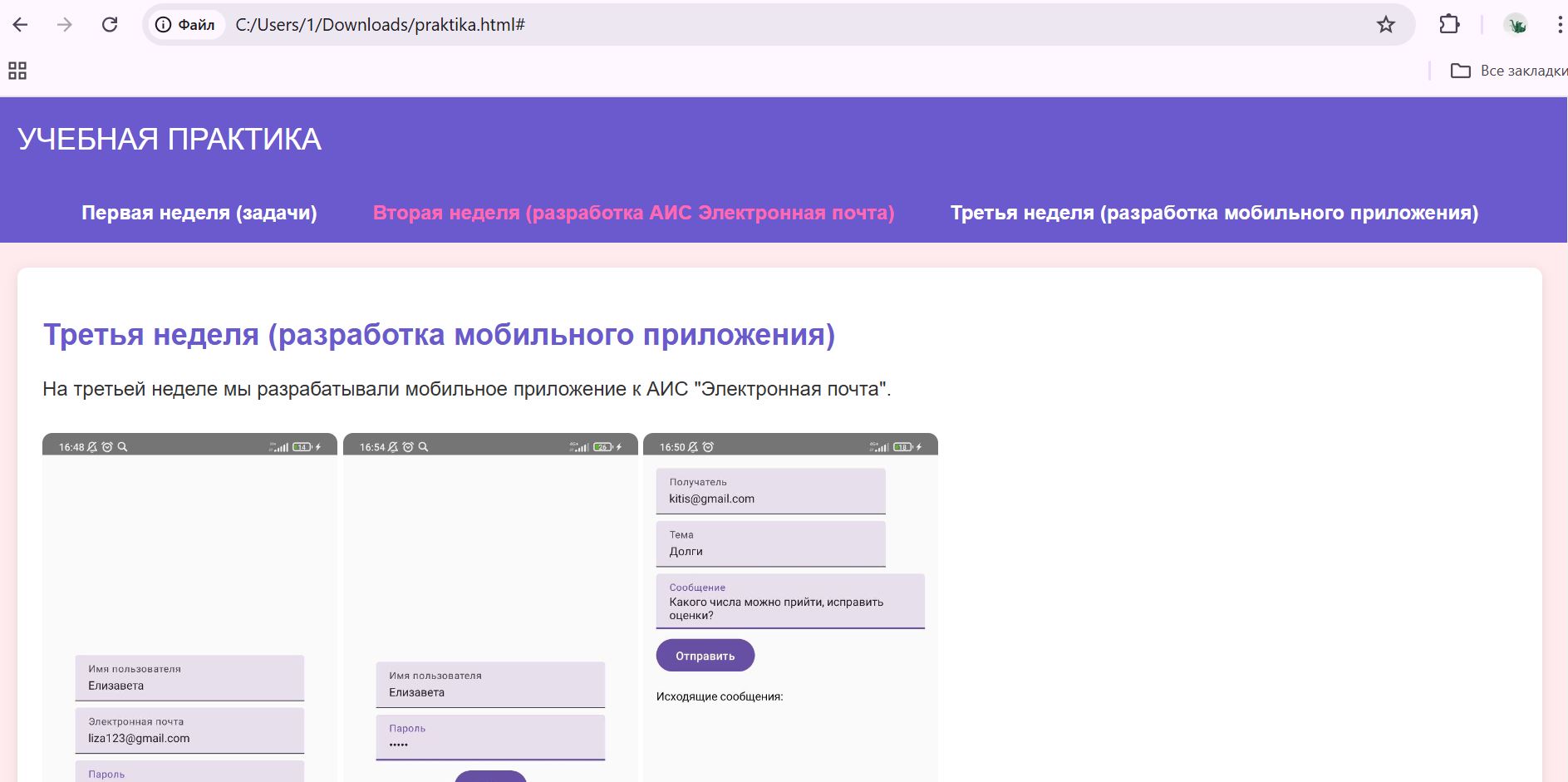


Рисунок 31 – Вкладка третья неделя

1. Руководство пользователя

Введение

Данное руководство предназначено для системных программистов и разработчиков, которые будут работать с автоматизированной информационной системой (АИС) "Электронная почта". В нем описываются этапы разработки системы, ее архитектура, оценка эффективности использования ресурсов, а также предложения по дальнейшей оптимизации.

Разработка серверной части

* Использование среды разработки IntelliJ IDEA.
* Подключение к базе данных MySQL для хранения и обработки данных.
* Реализация бизнес-логики на языке Java.

Интеграция с 1С

* Реализация интерфейсов для взаимодействия с системой 1С, что позволяет интегрировать АИС "Электронная почта" с существующими бизнес-процессами.
* Создание мобильного приложения
* Разработка мобильного приложения на платформе Android Studio.
* Использование языка Java для создания пользовательского интерфейса и взаимодействия с серверной частью.

При оценке программного продукта с точки зрения эффективности использования ресурсов следует учитывать:

* Производительность: система должна обеспечивать быструю обработку запросов и минимальное время отклика.
* Масштабируемость: возможность увеличения нагрузки на систему без значительных затрат на переработку архитектуры.
* Надежность: система должна быть устойчива к сбоям и обеспечивать сохранность данных.

Для реализации АИС "Электронная почта" был выбран паттерн проектирования MVC (Model-View-Controller). Обоснование выбора:

* Разделение ответственности: паттерн позволяет четко разделить бизнес-логику, пользовательский интерфейс и управление данными, что упрощает поддержку и развитие системы.
* Упрощение тестирования: благодаря разделению компонентов, тестирование отдельных частей системы становится более простым и эффективным.
* Гибкость: возможность легко изменять или заменять компоненты без влияния на другие части системы.

Документ разработан в соответствии со стандартом ЕСПД и содержит информацию о структуре и функциональности АИС "Электронная почта".

* Серверная часть: реализована на Java, использует MySQL для хранения данных.
* Клиентская часть: мобильное приложение на Android, использующее REST

API для взаимодействия с сервером.

1. Установить MySQL и создать базу данных.

2. Настроить серверную часть, указав параметры подключения к базе данных.

3. Установить Android Studio и импортировать проект мобильного приложения.

4. Настроить параметры подключения к серверу в мобильном приложении.

Регулярно проверять обновления для библиотек и компонентов системы. Рекомендуется проводить тестирование после каждого обновления.

1. Кэширование: внедрение механизма кэширования для уменьшения нагрузки на базу данных и ускорения обработки запросов.

2. Оптимизация запросов: анализ и оптимизация SQL-запросов для повышения производительности.

3. Использование микро сервисной архитектуры: разделение системы на независимые сервисы для улучшения масштабируемости и упрощения разработки.

4. Мониторинг и анализ производительности: внедрение инструментов мониторинга для выявления узких мест и оптимизации работы системы.

Заключение

В ходе выполнения индивидуального задания по учебной практике был достигнут значительный прогресс в освоении ключевых навыков разработки программного обеспечения. Применение языков программирования, таких как Java, а также работа с базами данных и проектирование автоматизированных информационных систем позволили глубже понять основные принципы и подходы в этой области.

Каждый этап практики способствовал не только закреплению теоретических знаний, но и развитию практических умений, необходимых для успешной работы в индустрии. Проектирование и реализация АИС, создание диаграмм для системного анализа и тестирование программных решений стали важными шагами на пути к формированию комплексного подхода к разработке программного обеспечения.

Завершение практики с разработкой мобильного приложения и созданием сайта на базе CMS подтвердило полученные знания и навыки, а также позволило оценить эффективность применяемых паттернов проектирования. Отчет по учебной практике стал итоговым документом, отражающим все проделанные работы и достигнутые результаты, и будет служить основой для дальнейшего профессионального роста и развития в сфере информационных технологий.

<https://github.com/Kolyvanova-Elizabeth/-.git> – Ссылка на GitHub

# Список использованной литературы

1. IntelliJ IDEA  
   JetBrains. (2023). IntelliJ IDEA Documentation. Retrieved from <https://www.jetbrains.com/idea/documentation/>
2. Java  
   Oracle. (2023). The Java™ Tutorials. Retrieved from <https://docs.oracle.com/javase/tutorial/>
3. Ассемблер  
   Кравченко, В. В. (2018). Ассемблер: программирование для процессоров Intel. Москва: БХВ-Петербург.
4. 1С  
   1С. (2023). Документация по 1С: Предприятие. Retrieved from <https://its.1c.ru/db/v8std/content/>
5. MySQL  
   MySQL AB. (2023). MySQL Documentation. Retrieved from <https://dev.mysql.com/doc/>
6. Android Studio  
   Google. (2023). Android Studio User Guide. Retrieved from <https://developer.android.com/studio/>
7. CMS для создания сайтов  
   Власов, А. (2020). Создание сайтов на CMS: от новичка до профессионала. Санкт-Петербург: Питер.