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 «11» ноября 2020 г. по «01» декабря 2020 г.

Место практики ГБУ КО ПОО «КИТиС»

|  |  |
| --- | --- |
| Выполнил: | студент 3 курса,  группы ИС 18-1  Коробов Артур Эдуардович  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |
| Проверила: | Большакова-Стрекалова Анна Викторовна  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (оценка)  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) |

Калининград, 2020

Содержание

[Введение 3](#_Toc58512949)

[1 Решение задач 4](#_Toc58512950)

[Задача 1 4](#_Toc58512951)

[Задача 2 5](#_Toc58512952)

[Задача 3 9](#_Toc58512953)

[Задача 4 12](#_Toc58512954)

[Задача 5 15](#_Toc58512955)

[2 Разработка АИС 17](#_Toc58512956)

[2.1 Анализ предметной области 17](#_Toc58512957)

[2.2 Техническое задание 21](#_Toc58512958)

[2.2.1 Введение 21](#_Toc58512959)

[2.2.2 Основания для разработки 21](#_Toc58512960)

[2.2.3 Назначение разработки 21](#_Toc58512961)

[2.2.4 Требования к программе 22](#_Toc58512962)

[2.2.5 Требования к программной документации 22](#_Toc58512963)

[2.2.6 Технико-экономические показатели 23](#_Toc58512964)

[2.2.7 Стадии и этапы разработки 23](#_Toc58512965)

[2.2.8. Порядок контроля и приемки 23](#_Toc58512966)

[2.3 Разработка структурной и функциональной схемы 24](#_Toc58512967)

[2.4 Проведение тестирования 25](#_Toc58512968)

[3 Разработка мобильного приложения 29](#_Toc58512969)

[4 Сайт 31](#_Toc58512970)

[Выводы 33](#_Toc58512971)

[Список использованных источников 35](#_Toc58512972)

[Приложения 36](#_Toc58512973)

# Введение

Для успешного и эффективного функционирования практически любой системы необходимы ведение контроля, анализа и внедрение автоматизации для отдельных процессов или системы в целом.

Автоматизация позволяет повысить производительность и качество системы, оптимизировать процессы управления, снизить затраты. Комплексная автоматизированная система управления предприятием является сегодня одной из важнейших составляющих успешного развития бизнеса. Если эта система выбрана и реализована правильно, она помогает устранить многие недостатки в управлении: разобщенность управленческих и информационных технологий; несоответствие систем планирования и контроля; неэффективность управления затратами и использования финансовых ресурсов.

Одной из основных целей данной учебной практики является проектирование автоматизированной информационной системы поликлиники, которая позволила бы значительно улучшить качество обслуживания пациентов и упростить процессы внутри учреждения, а также достичь максимального результата работы поликлиники.

Для описания предметной области и проектируемой системы будем использовать диаграммы UML. Язык моделирования UML предоставляет выразительные средства для создания визуальных моделей и обладает рядом значимых преимуществ:

1. UML объектно-ориентирован, в результате чего методы описания результатов анализа и проектирования семантически близки к методам программирования на современных объектно-ориентированных языках;
2. UML позволяет описать систему практически со всех возможных точек зрения и разные аспекты поведения системы;
3. диаграммы UML сравнительно просты для чтения после быстрого ознакомления с его синтаксисом;
4. UML получил широкое распространение и динамично развивается.

Помимо разработки АИС, учебная практике предусматривает разработку мобильного приложения и сайта, а также решение задач.

# 1 Решение задач

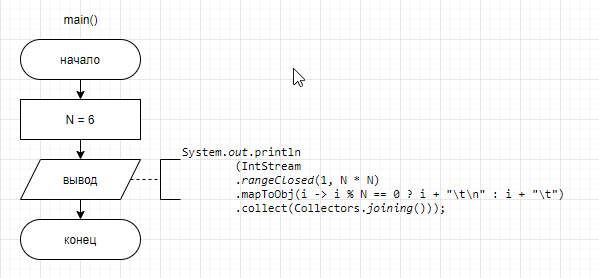
# Задача 1

Условие: вывести числа от 1 до k в виде матрицы N x N слева направо и сверху вниз.

Листинг программы:

package com.korobovartur;  
  
import java.util.stream.Collectors;  
import java.util.stream.IntStream;  
  
public class Main {  
  
 public static void main(String[] args) {  
 int N = 6;  
  
 System.*out*.println  
 (IntStream  
 .*rangeClosed*(1, N \* N)  
 .mapToObj(i -> i % N == 0 ? i + "\t\n" : i + "\t")  
 .collect(Collectors.*joining*()));  
 }  
}

Блок-схема:



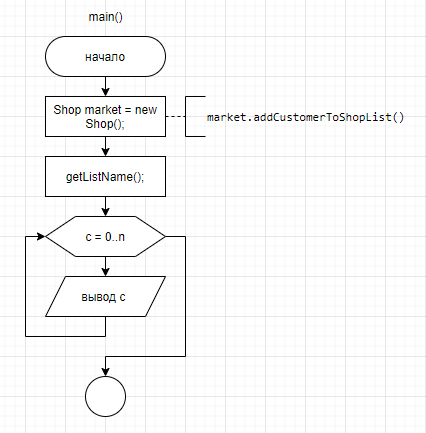
# Задача 2

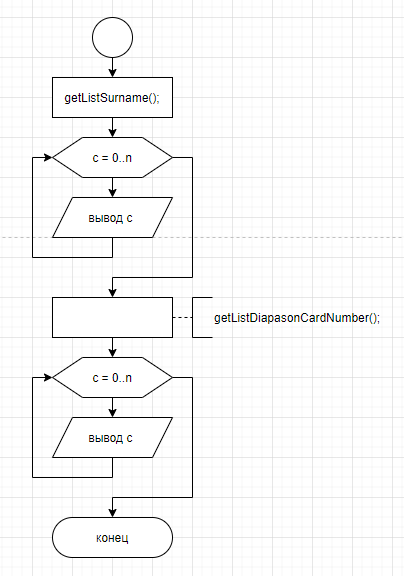
Условие: Customer имеет id, Фамилия, Имя, Отчество, Адрес, Номер кредитной карточки, Номер банковского счета. Создать массив объектов. Вывести: a) список покупателей в алфавитном порядке; b) список покупателей, у которых номер кредитной карточки находится в заданном интервале.

Листинг программы:

package com.korobovartur;  
  
import java.util.ArrayList;  
import java.util.Collections;  
import java.util.Comparator;  
import java.util.List;  
  
public class Main {  
  
 public static void main(String[] args) {  
 Shop market = new Shop("INTERSPAR");  
  
 market.addCustomerToShopList(new Customer("Коробов", "Артур", "Эдуардович", 264261, 45895674));  
 market.addCustomerToShopList(new Customer("Петренко", "Алексей", "Алексеевич", 353453, 16684336));  
 market.addCustomerToShopList(new Customer("Путин", "Артем", "Артемович", 212134, 15984467));  
 market.addCustomerToShopList(new Customer("Трамп", "Андрей", "Дмитриевич", 888888, 47853345));  
 market.addCustomerToShopList(new Customer("Белый", "Владимир", "Васильевич", 545456, 25693256));  
  
  
 System.*out*.println("\nСортировка по имени:");  
 List<Customer> listSortByName = market.getListName();  
 for (Customer c : listSortByName) {  
 System.*out*.println(c);  
 }  
  
 System.*out*.println("\nСоритровка по фамилии:");  
 List<Customer> listSortBySurname = market.getListSurname();  
 for (Customer c : listSortBySurname) {  
 System.*out*.println(c);  
 }  
  
 System.*out*.println("\nНомера карт в диапазоне от 300000 до 600000:");  
 List<Customer> diapasonCardumber = market.getListDiapasonCardNumber(300000, 600000);  
 for (Customer c : diapasonCardumber) {  
 System.*out*.println(c);  
 }  
 }  
}  
  
class Shop {  
  
 private String ShopName;  
 private ArrayList<Customer> customerList = new ArrayList<>();  
  
 Shop(String ShopName) {  
 this.ShopName = ShopName;  
 }  
  
 void addCustomerToShopList(Customer customer) {  
 customerList.add(customer);  
 }  
  
 List<Customer> getListName() {  
 List<Customer> list = new ArrayList<>(customerList);  
 Collections.*sort*(list, new Comparator<Customer>() {  
 @Override  
 public int compare(Customer o1, Customer o2) {  
 return o1.getName().compareToIgnoreCase(o2.getName());  
 }  
  
 });  
 return list;  
 }  
  
 List<Customer> getListSurname() {  
 List<Customer> list = new ArrayList<>(customerList);  
 Collections.*sort*(list, new Comparator<Customer>() {  
 @Override  
 public int compare(Customer o1, Customer o2) {  
 return o1.getSurname().compareToIgnoreCase(o2.getSurname());  
 }  
 });  
 return list;  
 }  
  
 List<Customer> getListDiapasonCardNumber(int diapasonStart, int diapasonEnd) {  
 List<Customer> list = new ArrayList<>();  
 for (Customer c : customerList) {  
 if (c.getCardNumber() >= diapasonStart && c.getCardNumber() <= diapasonEnd) {  
 list.add(c);  
 }  
 }  
 return list;  
 }  
  
}  
  
class Customer {  
  
 {  
 *idGenertor*++;  
 }  
  
 private static int *idGenertor* = 0;  
 private int id = *idGenertor*;  
 private String surname;  
 private String name;  
 private String patronymic;  
 private int CardNumber;  
 private int BankNumber;  
  
 Customer(String surname, String name, String patronymic, int CardNumber, int BankNumber) {  
 this.surname = surname;  
 this.name = name;  
 this.patronymic = patronymic;  
 this.CardNumber = CardNumber;  
 this.BankNumber = BankNumber;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public String getSurname() {  
 return surname;  
 }  
  
 public String getPatronymic() {  
 return patronymic;  
 }  
  
 public int getCardNumber() {  
 return CardNumber;  
 }  
  
 public int getBankNumber() {  
 return BankNumber;  
 }  
  
 public String toString() {  
 return String.*format*("ID: %d\t Фамилия: %s\t Имя: %s\t Отчество: %s\t Номер карты: %d\t Банковский номер: %d",  
 id, surname, name, patronymic, CardNumber, BankNumber);  
 }  
  
}

Блок-схема:





# Задача 3

Условие: создать объект класса Автомобиль, используя классы Колесо, Двигатель. Методы: ехать, заправляться, менять колесо, вывести на консоль марку автомобиля.

Листинг класса Main.java:

package com.korobovartur;  
  
import java.util.Scanner;  
  
public class Main {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 Car car = Car.*get*(scanner);  
 while (true) {  
 System.*out*.println(  
 "Выберете пункт меню:\n" +  
 "0. выход\n" +  
 "1. ехать\n" +  
 "2. заправляться\n" +  
 "3. менять колесо\n" +  
 "4. вывести информацию об авто (марку тоже)\n" +  
 ": "  
 );  
 int choice = scanner.nextInt();  
 if (choice == 0)  
 break;  
 if (choice < 1 || choice > 4) {  
 System.*out*.println("выбран неправильный пункт меню, повторите ввод.");  
 continue;  
 }  
 switch (choice) {  
 case 1:  
 car.move();  
 break;  
 case 2:  
 System.*out*.println("Введи количество топлива:");  
 double fuel = scanner.nextDouble();  
 car.add\_fuel(fuel);  
 break;  
 case 3:  
 car.change\_wheel(scanner);  
 break;  
 case 4:  
 System.*out*.println(car.toString());  
 break;  
 }  
 }  
 }  
}

Листинг класса Car.java:

package com.korobovartur;  
  
import java.util.\*;  
  
class Car {  
 private Wheel wheels[];  
 private Engine engine;  
 private String brand;  
 private double consumption;  
 private double fuel\_level;  
 public Car(String brand\_, double consumption\_,  
 Engine engine\_, Wheel wheels\_[], double fuel\_level\_) {  
 brand = brand\_;  
 consumption = consumption;  
 engine = engine\_;  
 wheels = wheels\_;  
 fuel\_level = fuel\_level\_;  
 }  
 public String toString() {  
 String string = "";  
 string += brand + engine.toString() + ";" + consumption + "L/100km { " + fuel\_level + "}; [";  
 for (int i = 0; i < 4; ++i) {  
 string += wheels[i].toString() + " ";  
 }  
 return string + "]";  
 }  
 public static Car get(Scanner scanner) {  
 Car object = new Car("", 0, new Engine(0, 0), new Wheel[4], 0);  
 System.*out*.print("Введи марку автомобиля: ");  
 object.brand = Common.*readNotEmptyString*(scanner);  
 System.*out*.print("Введи расход топлива на 100км: ");  
 object.consumption = scanner.nextDouble();  
 System.*out*.print("Введи текущий уровень топлива:");  
 object.fuel\_level = scanner.nextDouble();  
 System.*out*.println("Введи колеса:");  
 for (int i = 0; i < 4; ++i) {  
 System.*out*.println("колесо #" + i);  
 object.wheels[i] = Wheel.*get*(scanner);  
 }  
 object.engine = Engine.*get*(scanner);  
 return object;  
 }  
 public void move() {  
 if (fuel\_level <= 0) {  
 System.*out*.println("Нет бензина, машина не может двигаться");  
 return;  
 }  
 for (int i = 1; i < 4; ++i) {  
 if (wheels[i].get\_diameter() != wheels[i - 1].get\_diameter()) {  
 System.*out*.println("Колеса разных диаметров, машина не может двигаться");  
 return;  
 }  
 }  
 double distance = fuel\_level / consumption;  
 System.*out*.println("машина проехала " + distance \* 100 + "км.");  
 fuel\_level = 0;  
 }  
 public void add\_fuel(double value) {  
 fuel\_level += value;  
 }  
 public void change\_wheel(Scanner scanner) {  
 System.*out*.println("Введи номер колеса [0-3]:");  
 int number;  
 while (true) {  
 number = scanner.nextInt();  
 if (number >= 0 && number <= 3)  
 break;  
 System.*out*.println("Неправильный номер, повтори ввод");  
 }  
 wheels[number] = Wheel.*get*(scanner);  
 }  
}

Листинг класса Engine.java:

package com.korobovartur;  
  
import java.util.\*;  
  
public class Engine {  
 private int number;  
 private double power;  
 public Engine(int number\_, double power\_) {  
 number = number\_;  
 power = power\_;  
 }  
 public String toString() {  
 String string = "engine: " + number + ", power:" + power;  
 return string;  
 }  
 public static Engine get(Scanner scanner) {  
 Engine object = new Engine(0, 0);  
 System.*out*.print("Введи номер двигателя: ");  
 object.number = scanner.nextInt();  
 System.*out*.print("Введи мощность двигателя: ");  
 object.power = scanner.nextDouble();  
 return object;  
 }  
}

Листинг класса Wheel.java:

package com.korobovartur;  
  
import java.util.\*;  
  
class Wheel {  
 public enum Type { *Winter*, *Summer* }  
  
 private double diameter;  
 private String brand;  
 private Type type;  
  
 public Wheel(String brand\_, double diameter\_, Type type\_) {  
 brand = brand\_;  
 diameter = diameter\_;  
 type = type\_;  
 }  
  
 public String toString() {  
 String string = "";  
  
 string += brand + "(" + diameter + "): ";  
 if (type == Type.*Winter*)  
 string += "Winter";  
 else  
 string += "Summer";  
  
 return string;  
 }  
  
 public static Wheel get(Scanner scanner) {  
 Wheel object = new Wheel("", 0, Type.*Winter*);  
  
 System.*out*.print("Введи бренд: ");  
 object.brand = Common.*readNotEmptyString*(scanner);  
  
 System.*out*.print("Введи диаметр: ");  
 object.diameter = scanner.nextDouble();  
  
 System.*out*.print("Введи тип (1) - зимние, (2) - летние: ");  
 while (true) {  
 int value = scanner.nextInt();  
 if (value == 1) {  
 object.type = Type.*Winter*;  
 break;  
 }  
 if (value == 2) {  
 object.type = Type.*Summer*;  
 break;  
 }  
 System.*out*.print("Введено неправильное значение, повтори ввод.");  
 }  
 return object;  
 }  
  
 public double get\_diameter() {  
 return diameter;  
 }  
}

# Задача 4

Условие: создать класс Payment с внутренним классом, с помощью объектов которого можно сформировать покупку из нескольких товаров.

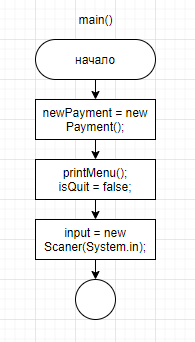
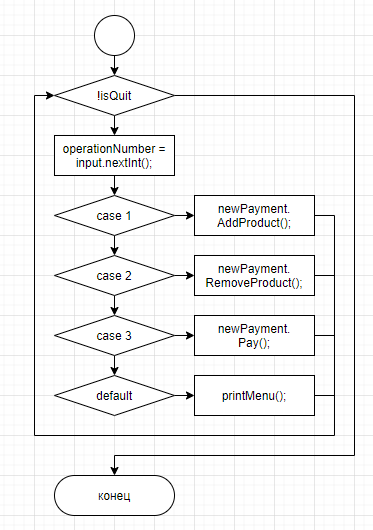
Листинг класса Main.java:

package com.korobovartur;  
  
import java.util.Scanner;  
  
public class Main {  
  
 public static void main(String[] args) {  
 Payment newPayment = new Payment();  
  
 Scanner input = new Scanner(System.*in*);  
  
 *printMenu*();  
  
 boolean isQuit = false;  
 while (!isQuit) {  
 int operationNumber = input.nextInt();  
  
 switch (operationNumber) {  
 case 1:  
 newPayment.AddProduct();  
 break;  
 case 2:  
 newPayment.RemoveProduct();  
 break;  
 case 3:  
 newPayment.Pay();  
 break;  
 default:  
 *printMenu*();  
 }  
 }  
 }  
  
 public static void printMenu() {  
 System.*out*.println("Введите номер операции:");  
 System.*out*.println("1 - Добавить продукт");  
 System.*out*.println("2 - Удалить продукт");  
 System.*out*.println("3 - Совершить платеж");  
 }  
}

Листинг класса Payment.java:

package com.korobovartur;  
  
import java.util.ArrayList;  
  
public class Payment {  
  
 private ArrayList<Product> itemsList = new ArrayList<>();  
 private int id = 0;  
  
  
 public Payment() {  
 this.id = 10000000 + (int)(Math.*random*() \* 100000000);  
  
 System.*out*.printf("Платеж %d создан! \n", this.id);  
 }  
  
  
 public void Pay() {  
 if (itemsList.size() > 0) {  
 System.*out*.println("Платеж выполняется...");  
 System.*out*.println("Платеж успешно выполнен!");  
  
 itemsList.clear();  
 } else {  
 System.*out*.println("Ваша корзина пуста! Платеж не выполнен.");  
 }  
 }  
  
  
 public void AddProduct() {  
 Product newProduct = new Product(itemsList.size() + 1);  
  
 itemsList.add(newProduct);  
  
 System.*out*.printf("Продукт %d добавлен список! \n", itemsList.get(itemsList.size() - 1).id);  
 }  
  
  
 public void RemoveProduct() {  
 if (itemsList.size() > 0) {  
 System.*out*.printf("Продукт %d удален из списка! \n", itemsList.get(0).id);  
  
 itemsList.remove(0);  
 } else {  
 System.*out*.println("Ваша корзина пуста!");  
 }  
 }  
  
  
 public class Product {  
  
 private int id;  
  
  
 public Product(int \_id) {  
 this.id = \_id;  
  
 }  
 }  
}

Блок-схема:

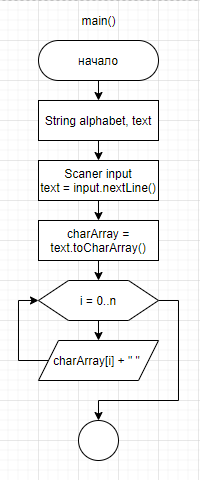
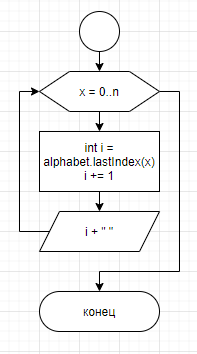
# Задача 5

Условие: в тексте каждую букву заменить ее порядковым номером в алфавите. При выводе в одной строке печатать текст с двумя пробелами между буквами, в следующей строке внизу под каждой буквой печатать ее номер.

Листинг программы:

package com.korobovartur;  
  
import java.util.Scanner;  
  
public class Main {  
  
 public static void main(String[] args) {  
 String alphabet = "абвгдеёжзийклмнопрстуфхцчшщъыьэюя"; // Алфавит  
  
 String text = "";  
 System.*out*.println("Введите текст");  
 Scanner input = new Scanner(System.*in*);  
 text = input.nextLine();  
  
 char[] charArray = text.toCharArray();  
  
 //Выводим в консоль введенные символы с двумя пробелами  
 for(int i = 0; i < text.length(); i++){  
 System.*out*.print(charArray[i] + " ");  
 }  
  
 System.*out*.println("");  
  
 //Выводим в консоль номер позиции буквы в алфавите  
 for (char x : charArray){  
 //Игнорируем символ пробела в веденном тексте  
 if(x == ' '){  
 System.*out*.print("");  
 } else {  
 int i = alphabet.lastIndexOf(x);  
 i+=1;  
 System.*out*.print(i + " ");  
 }  
 }  
 }  
}

Блок-схема:

# 2 Разработка АИС

# 2.1 Анализ предметной области

Создание современных электронных вычислительных машин позволило автоматизировать обработку данных во многих сферах человеческой деятельности. Без современных систем обработки данных трудно представить сегодня передовые производственные технологии, управление экономикой на всех ее уровнях, научные исследования, образование, издательское дело, функционирование средств массовой информации, проведение крупных спортивных состязаний. Значительно расширило сферу применения систем обработки данных появление персональных компьютеров.

Одним из наиболее распространенных классов систем обработки данных являются информационные системы. Хотя назначение таких систем понятно каждому, для серьезного обсуждения технологий современных информационных систем необходимо более четко определить, в чем заключаются их специфические особенности, чем они отличаются от других систем обработки данных, какие функции они могут выполнять, какими ресурсами они располагают.

Любой вид деятельности основывается на информации о свойствах состояния и поведения той части реального мира, с которой связана эта деятельность. Для получения такой информации во многих случаях необходимо регулярно через некоторые интервалы времени проводить натурные измерения (или наблюдения), позволяющие определять характеристики состояния сущностей реального мира и протекающих процессов, соответствующие моментам времени, когда эти измерения производятся.

В других случаях удается воспользоваться «материализованной» информацией, содержащейся в различного рода бумажных документах, отчетах или публикациях, которые также выступают как часть реальности. Требуемую информацию можно извлечь из них путем своего рода «наблюдения».

Значительно сократить объем необходимых натурных измерений позволяет компьютерное моделирование реальности. Если компьютерная модель адекватно (относительно информационных потребностей пользователей) отражает состояние и динамику реальности, то многие необходимые сведения можно получать с помощью такой модели, избегая тем самым натурных измерений, с существенно меньшими затратами времени, а возможно, и при более низкой стоимости. Именно для поддержки таких моделей служит специальный класс систем обработки данных - автоматизированные информационные системы.

Автоматизированной информационной системой называется комплекс, включающий вычислительное и коммуникационное оборудование, программное обеспечение, информационные ресурсы, а также системный персонал и обеспечивающий поддержку динамической информационной модели некоторой части реального мира для удовлетворения информационных потребностей пользователей.

Поскольку модель предметной области, поддерживаемая информационной системой, материализуется в форме организованных необходимым образом информационных ресурсов, она называется информационной моделью.

АИС не всегда функционирует самостоятельно. Она может входить в качестве компонента (подсистемы) в более сложную систему, такую, например, как система управления торговой компанией, САПР или система управления производством.

Актуальность данной работы обусловлена теоретической и практической значимостью проблем, связанных с реализацией кадровой политики любой компании в сфере учета кадров, как важнейшей гарантии укрепления законности и правопорядка.

Все это в совокупности и обусловило определение цели, задач, объекта и предмета данной работы.

Целью данной работы является создание автоматизированной информационной системы заданной предметной области - кадровый учет поликлиники. С помощью АИС поликлиники появляется возможность ведения грамотного кадрового учета.

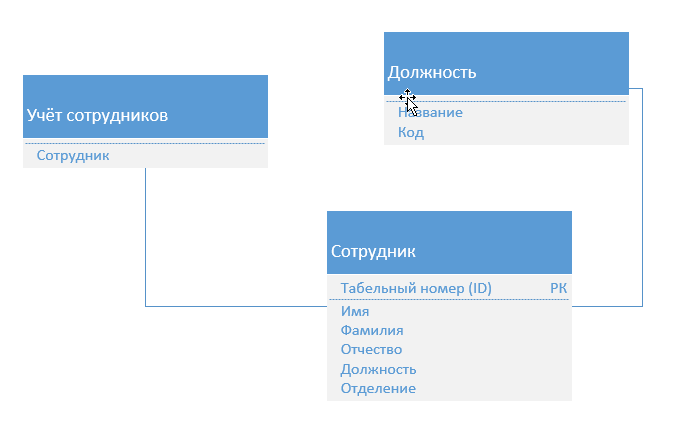


Рисунок 1 – ER диаграмма

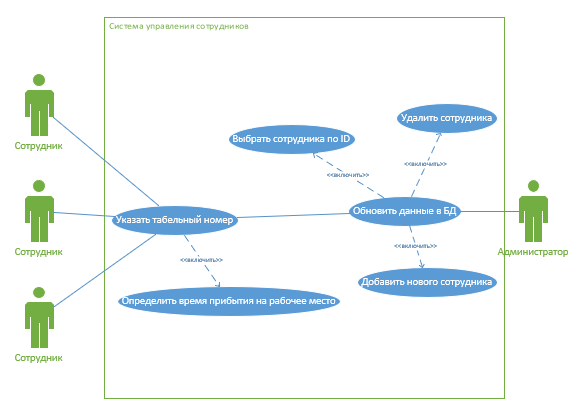


Рисунок 2 – Use-case диаграмма

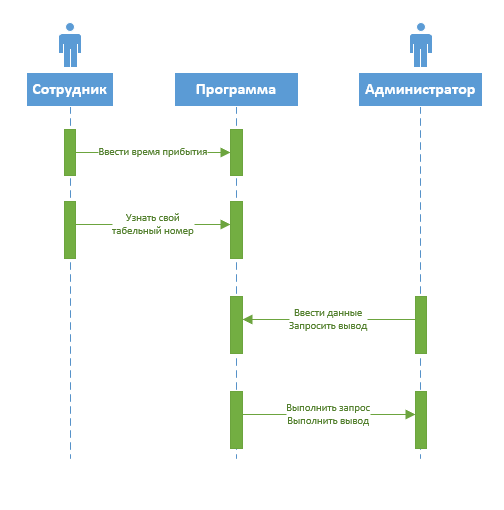


Рисунок 3 – Диаграмма последовательностей

# 2.2 Техническое задание

# 2.2.1 Введение

2.2.1.1. Наименование: «Консольное приложение для работы с БД сотрудников поликлиники».

2.2.1.2. Краткая характеристика области применения: программа предназначена для автоматизации ведения кадрового учета поликлиники. Основная цель разрабатываемой программы – облегчить ведение кадрового учета поликлиники.

# 2.2.2 Основания для разработки

2.2.2.1. Основанием для проведения разработки служит приказ №\_\_ от \_\_ декабря 2020 года. Приказ утвержден директором  ГБУ КО ПОО «КИТиС».

2.2.2.2. Наименование темы разработки – «Разработка консольного приложения работы с БД для поликлиники». Уловное обозначение темы разработки – «PolyApp».

# 2.2.3 Назначение разработки

2.2.3.1. Функциональное назначение: программа предоставляет возможность работы с базой данных сотрудников поликлиники.

2.2.3.2. Эксплуатационное назначение: программа должна эксплуатироваться в профильном подразделении на объекте заказчика. Пользователями программы должны являться сотрудники профильных подразделений объекта заказчика.

# 2.2.4 Требования к программе

2.2.4.1. Требования к функциональным характеристикам: программа должна выступать в роли клиента для связи с серверной частью (MySQL). Программа должна обеспечивать возможность работы с таблицей сотрудников (экспорт записей, добавление и(или) удаление записей, выборку записей по ключу).

2.2.4.2. Требование к надежности: программа должна выполняться без ошибок. Необходимо обеспечить проверку входных и выходных данных.

2.2.4.3. Условия эксплуатации: минимальное количество персонала, требуемого для работы программы, должно составлять не менее 2 штатных единиц – сотрудник(врач) и пользователь программы – оператор.

2.2.4.4. Требования к информационной и программной совместимости:

2.2.4.4.1. Используемый язык программирования – Java.

2.2.4.4.2. Используемая среда разработки – IntelliJ IDEA CE.

2.2.4.4.3. Целевая ОС – Windows 10.

2.2.4.4.3. Используемые библиотеки: коннектор для работы с MySQL в среде разработки IntelliJ IDEA CE.

2.2.4.5. Требования к маркировке и упаковке: программа поставляется в виде jar-файла, исполняемого файла и необходимой библиотеки. Не требует установки. Требования к маркировке и упаковке не предъявляются.

2.2.4.6. Требования к транспортированию и хранению:

2.2.4.6.1. Ссылка на GitHub проекта: https://github.com/unicorndk/e\_practice/tree/week2/PolyAppProject

# 2.2.5 Требования к программной документации

* «Программа для ведения кадрового учета поликлиники». Техническое задание;
* «Программа для ведения кадрового учета поликлиники». Руководство администратора;

# 2.2.6 Технико-экономические показатели

Использование разрабатываемого продукты сократит время, затрачиваемое на кадровый учет.

# 2.2.7 Стадии и этапы разработки

2.2.7.1. Стадии разработки:

* техническое задание;
* технический и рабочий проекты;
* внедрение.

2.2.7.1. Этапы разработки:

На стадии «Техническое задание» должен быть выполнен этап разработки, согласования и утверждения настоящего технического задания.

На стадии «Технический (и рабочий) проект» должны быть выполнены перечисленные ниже этапы работ:

* разработка программы;
* разработка программной документации;
* испытания программы.

На стадии «Внедрение» должен быть выполнен этап разработки «Подготовка и передача программы».

# 2.2.8. Порядок контроля и приемки

Производится проверка корректного выполнения программой заложенных в нее функций, т.е. осуществляется функциональное тестирование программы.

# 2.3 Разработка структурной и функциональной схемы



Рисунок 4 – Структурная схема



Рисунок 5 – Функциональная схема

# 2.4 Проведение тестирования

Тестирование программы осуществлялось непосредственно в среде разработке, а так же на различных компьютерах.

Результаты тестирования dev версии приложения:

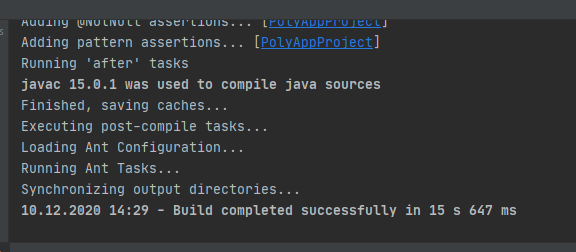


Рисунок 6 – Успешная сборка проекта

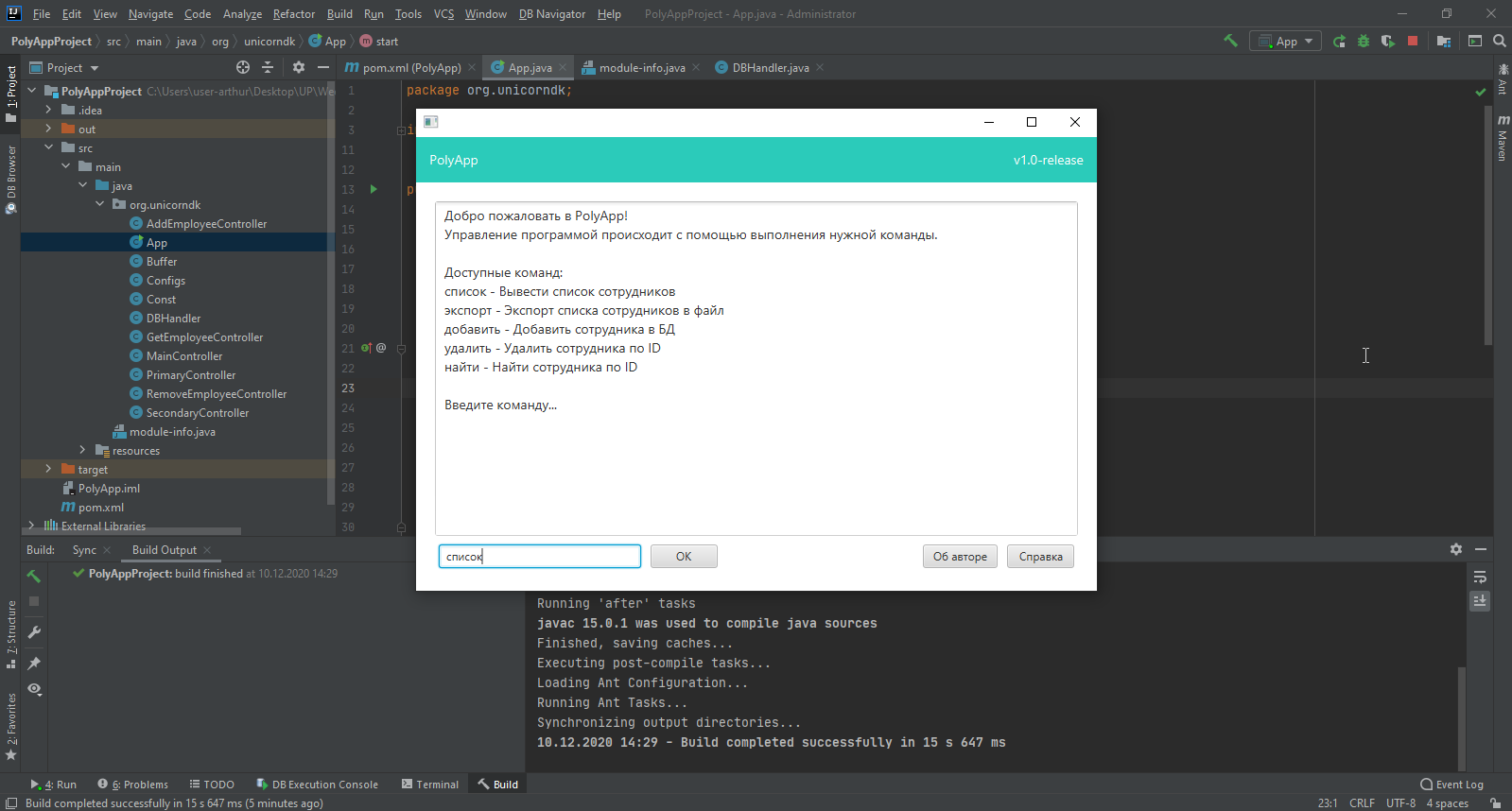


Рисунок 7 – Запуск главного окна программы (dev версия)

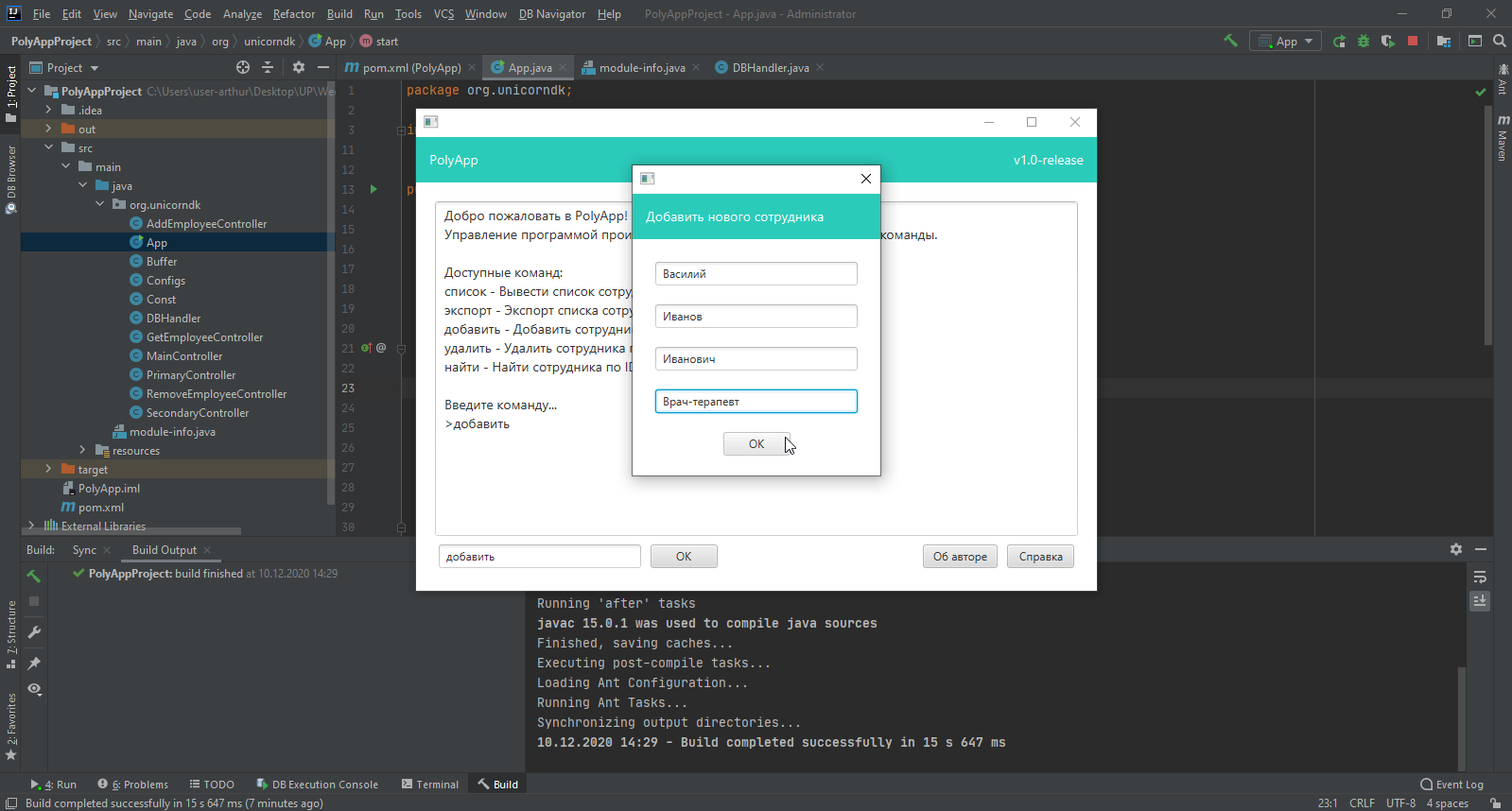


Рисунок 8 – Тестирование функциональности программы (dev версия)

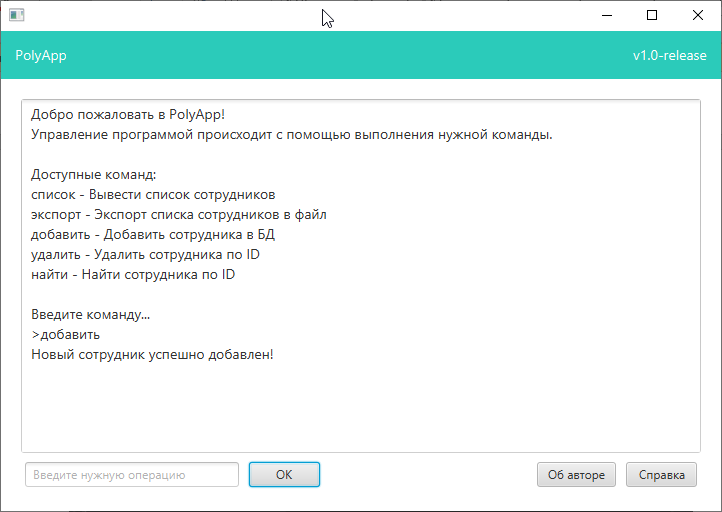


Рисунок 9 – Успешной выполнение функции (dev версия)

Тестирование release версии приложения осуществлялось на компьютере с OS Windows 10.

Папка вывода содержит .jar исполняемый файл, папку с библиотекой JavaFx, а также исполняемый .bat файл, который связывает .jar файл с JavaFx библиотекой и осуществляет запуск программы.

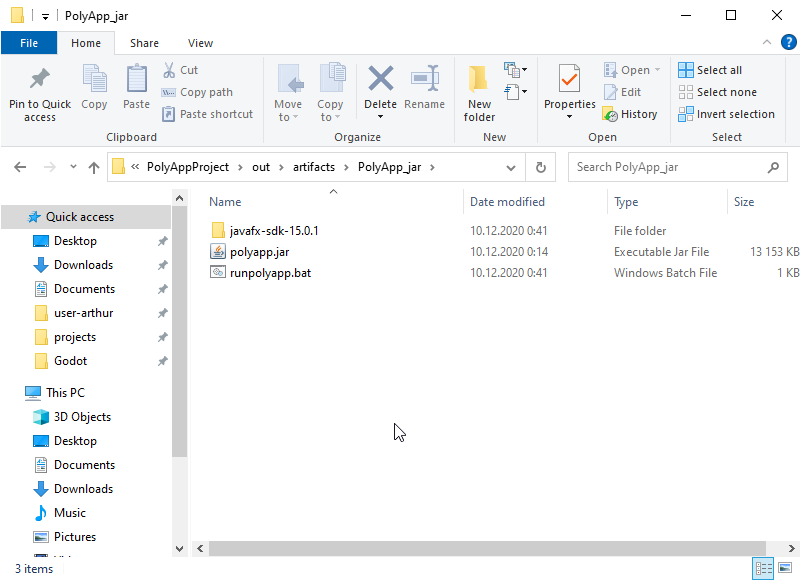


Рисунок 10 – Папка вывода

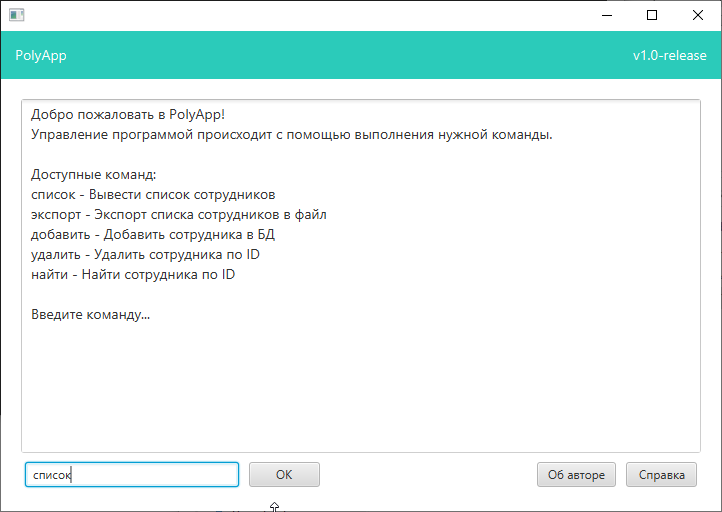


Рисунок 11 – Запуск главного окна программы (release версия)

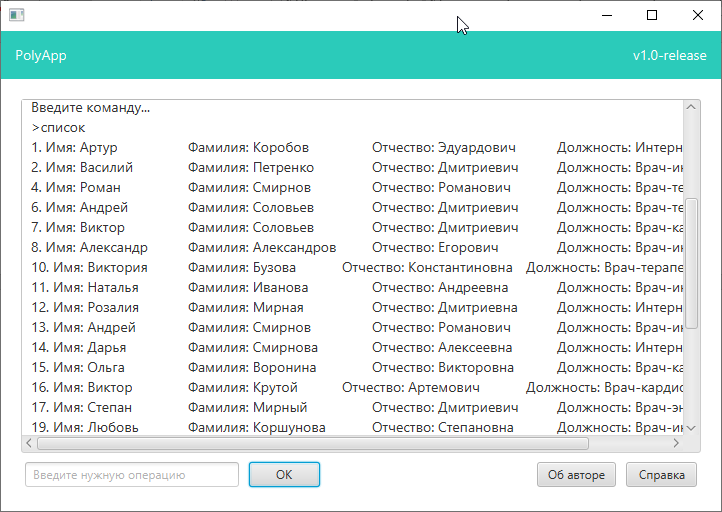


Рисунок 12 – Тестирование функциональности программы (release версия)

# 3 Разработка мобильного приложения

Разработка мобильного приложения начинается с создания идеи. Моя идея – разработка анонимного чат-приложения под на званием «Chatty».

В качестве среды разработки была выбрана Android Studio, на мой взгляд, это лучшая среда разработки под платформу Android.

Серверная часть приложения реализована с помощью Google Firebase Realtime Database. Firebase предоставляет удобный интерфейс для работы с базой данных в режиме реального времени.

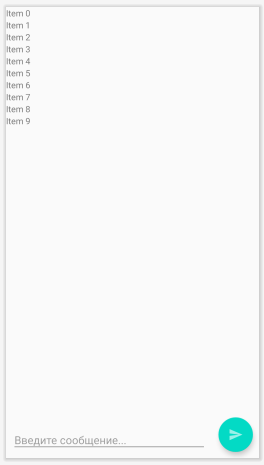


Рисунок 13 – Макет приложения в Android Studio

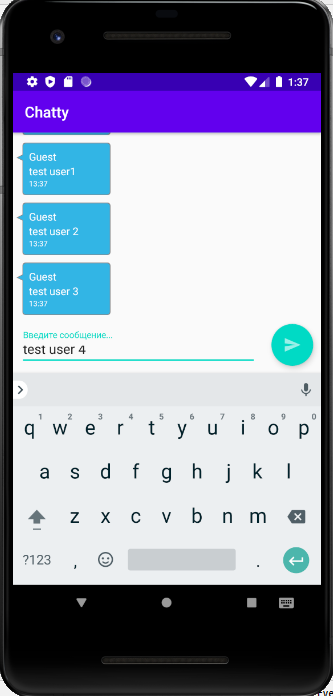
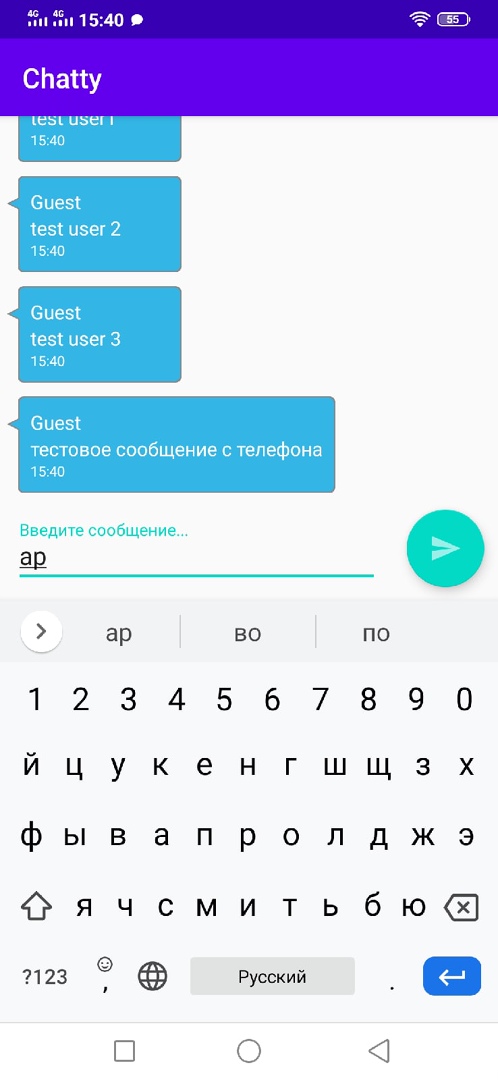
 

Рисунок 14 – Приложение запущенное на эмуляторе и на устройстве

# 4 Сайт

Разработка сайта — это процесс, состоящий из нескольких последовательных этапов.

Работу по созданию можно разделить на такие этапы:

- подготовительный;

- разработка макета;

- верстка;

- программирование;

- наполнение контентом;

- администрирование (поддержка) сайта.

Подготовительный этап начинается с определения цели разработки, создания идеи, поиска необходимых ресурсов для реализации, выбора среды разработки.

Основная цель разработки – это создание сайта-визитки для разработчика. Сайт должен быть наполнен контентом в виде информации о проектах разработчика.

Верстка сайта осуществлялась с помощью фреймворка Bootstrap версии 4. Данный фреймворк содержит готовые классы и стили, которые упрощают процесс верстки. Разработке осуществлялась в Atom IDE. Хостинг сайта осуществляется с помощью GitPages.

Также стоит отметить, что сайт является адаптивным. Его страницы по-разному отображаются на разных устройствах.

Ccылка на GitHub репозиторий: <https://github.com/unicorndk/e_practice>

Ссылка на сайт (GitHub Pages): <https://unicorndk.github.io>

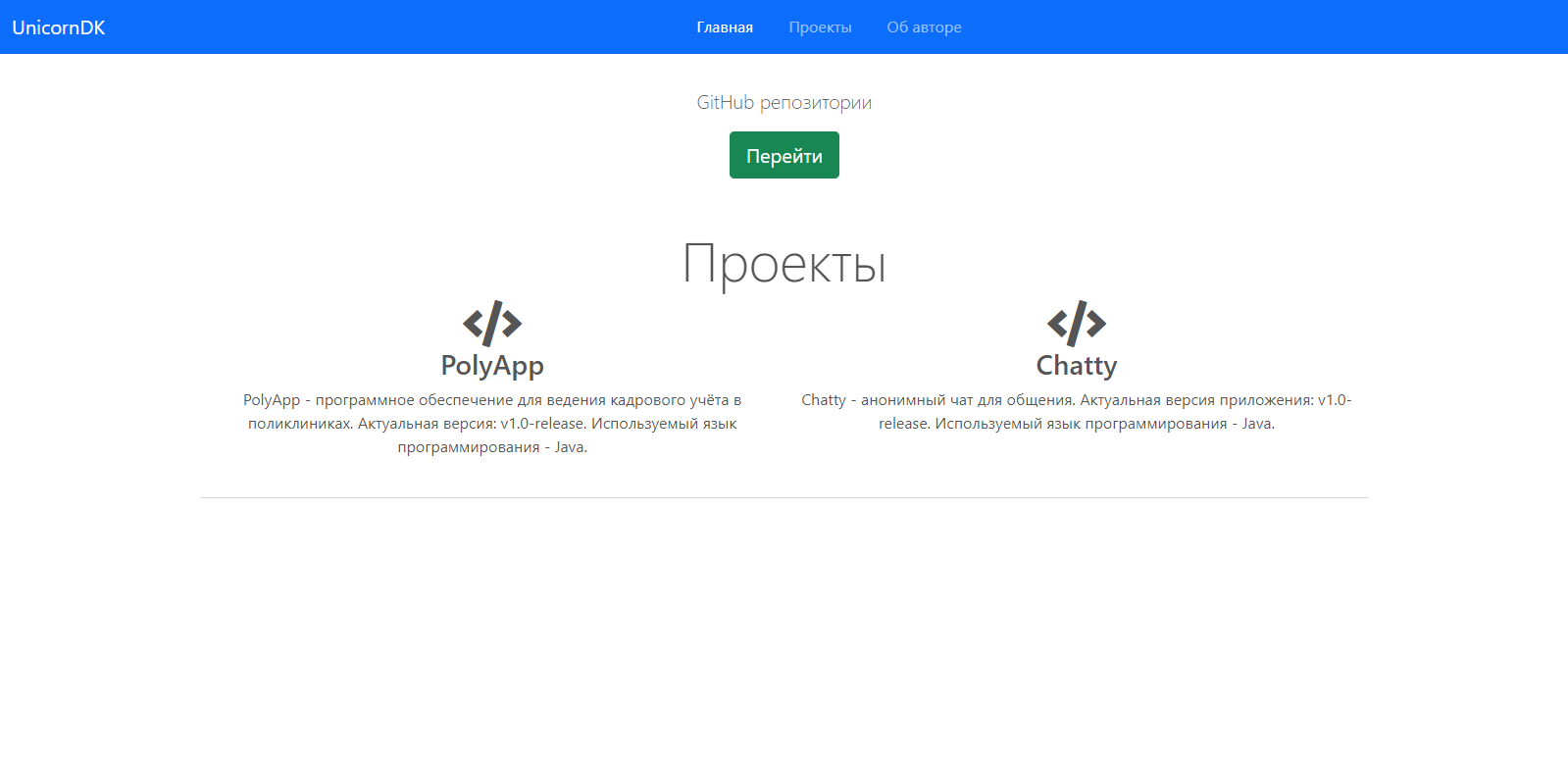


Рисунок 15 – Desktop версия сайта

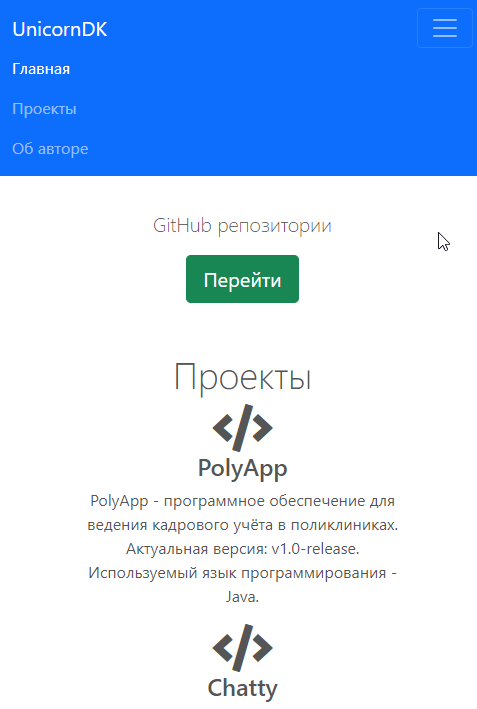


Рисунок 16 – Mobile версия сайта

# Выводы

Во время практики мною были изучены тонкости процесса разработки различных программных продуктов. Практика помогла мне закрепить навыки, полученные во время обучения. Я приобрел незаменимый опыт, который, несомненно, очень пригодится в дальнейшей профессиональной деятельности.

На этапе разработки мобильного приложения я закрепил навыки разработки Android приложений. На мой взгляд, Android разработка – одно из самых перспективных направлений в моей профессии.

В данной работе также была смоделирована информационная система кадрового учета для поликлиники, то есть была спроектирована информационная система для более удобного ведения кадрового учета.

С помощью MicrosoftVisio можно упростить работу разработчика, так как для построения диаграмм не нужно обладать глубокими знаниями в среде программирования.

Проектируемая информационная система для поликлиники содержит в себе такие процессы, как:

* добавление новых сотрудников в БД;
* поиск сотрудника в БД;
* удаление сотрудника из БД;
* вывод списка сотрудников из БД;
* экспорт списка сотрудников в файл.

Автоматизированная система кадрового учета позволит повысить скорость работы поликлиники, также сократить время обработки и анализа информации.

Таким образом, спроектирована система, решающая все выявленные на начальном этапе проблемы.

Я хотел бы дальше развиваться в области разработки программного обеспечения на языке Java. На мой взгляд это перспективное направление.

Дата сдачи отчета:\_\_\_\_\_\_\_\_\_\_ Оценка:\_\_\_\_\_

# Список использованных источников

1. В. В. Бахтизин, Л. А. Глухова. Технология разработки программного обеспечения : учеб. пособие / – Минск : БГУИР, 2018
2. К.Вигерс. Разработка требований к программному обеспечению. Издательско-торговый дом «Русская редакция», 2019
3. Гагарина Л.Г., Кокорева Е.В., Виснадул Б.Д. Технология разработки программного обеспечения: учебное пособие.- М.: ИД «Форум»: ИНФРА-М, 2019
4. А.В. Рудаков .Технология разработки программных продуктов. М.: ACADEMIA, 2016
5. Рудаков А.В., Федорова Г.Н. Технология разработки программных продуктов. Практикум: -М.:ACADEMA, 2016

# Приложения

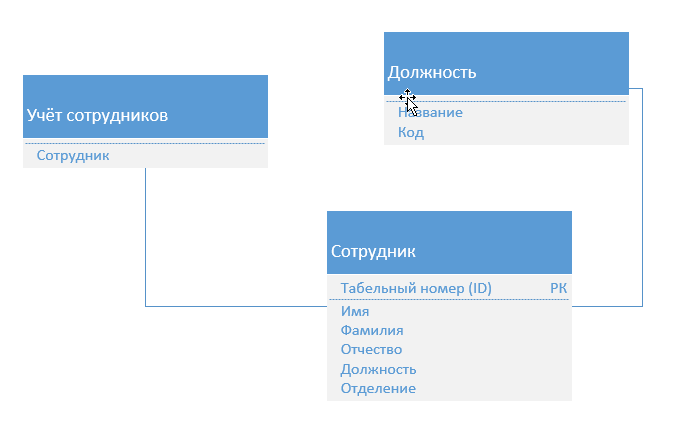


Рисунок 1 – ER диаграмма

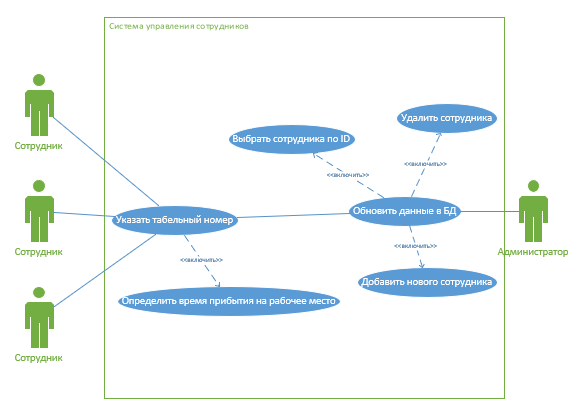


Рисунок 2 – Use-case диаграмма

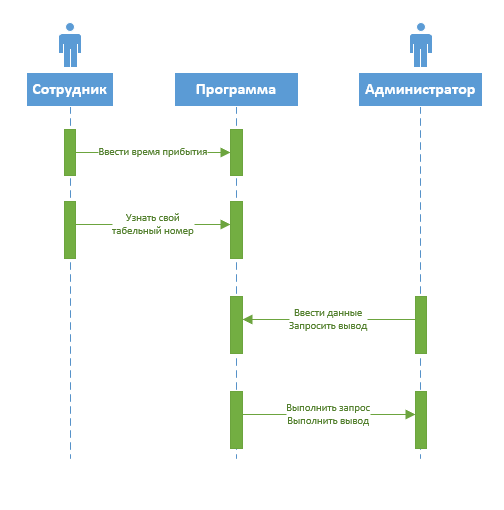


Рисунок 3 – Диаграмма последовательностей



Рисунок 4 – Структурная схема



Рисунок 5 – Функциональная схема

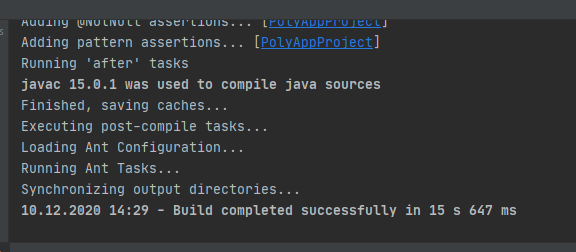


Рисунок 6 – Успешная сборка проекта

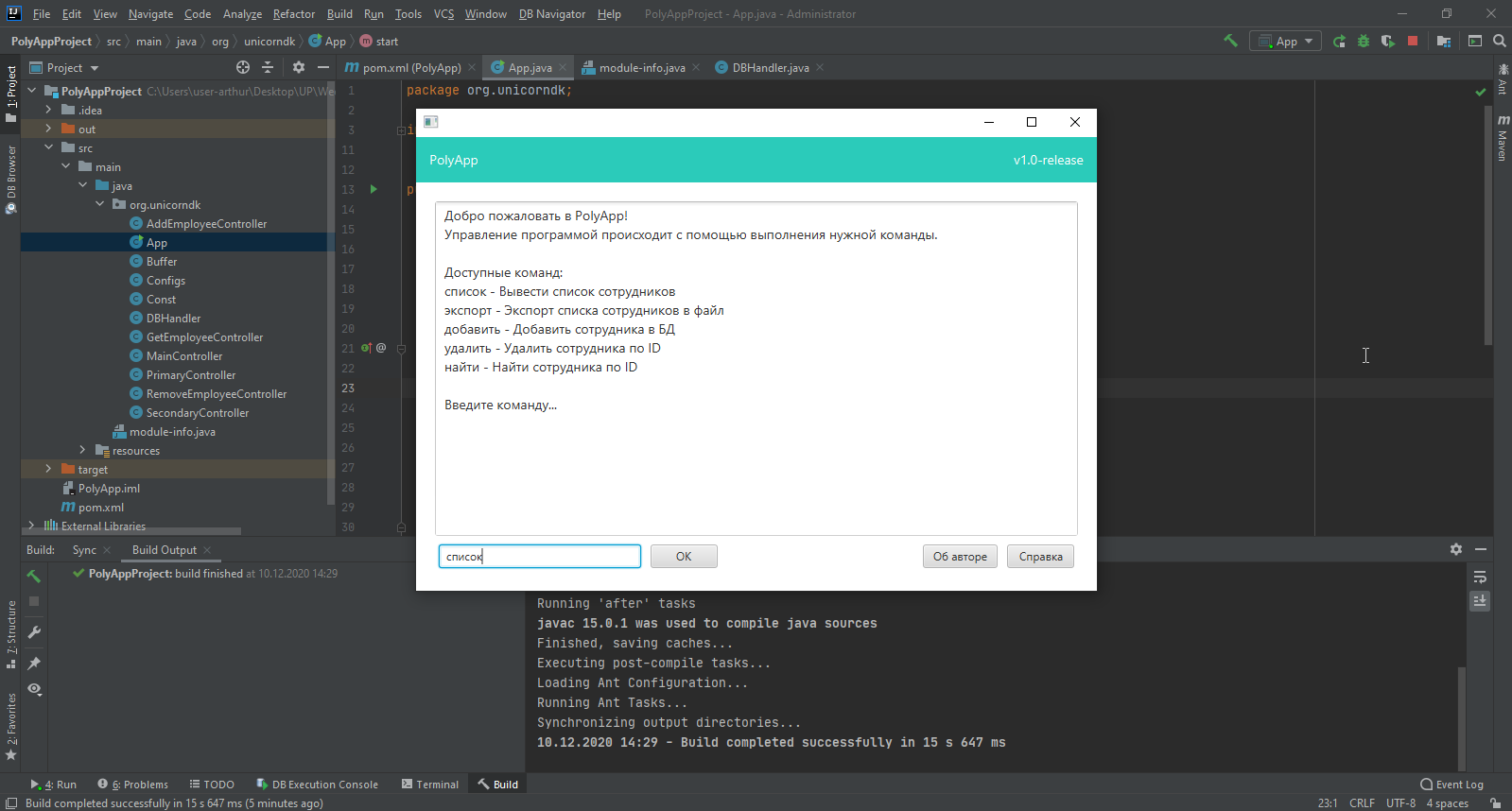


Рисунок 7 – Запуск главного окна программы (dev версия)

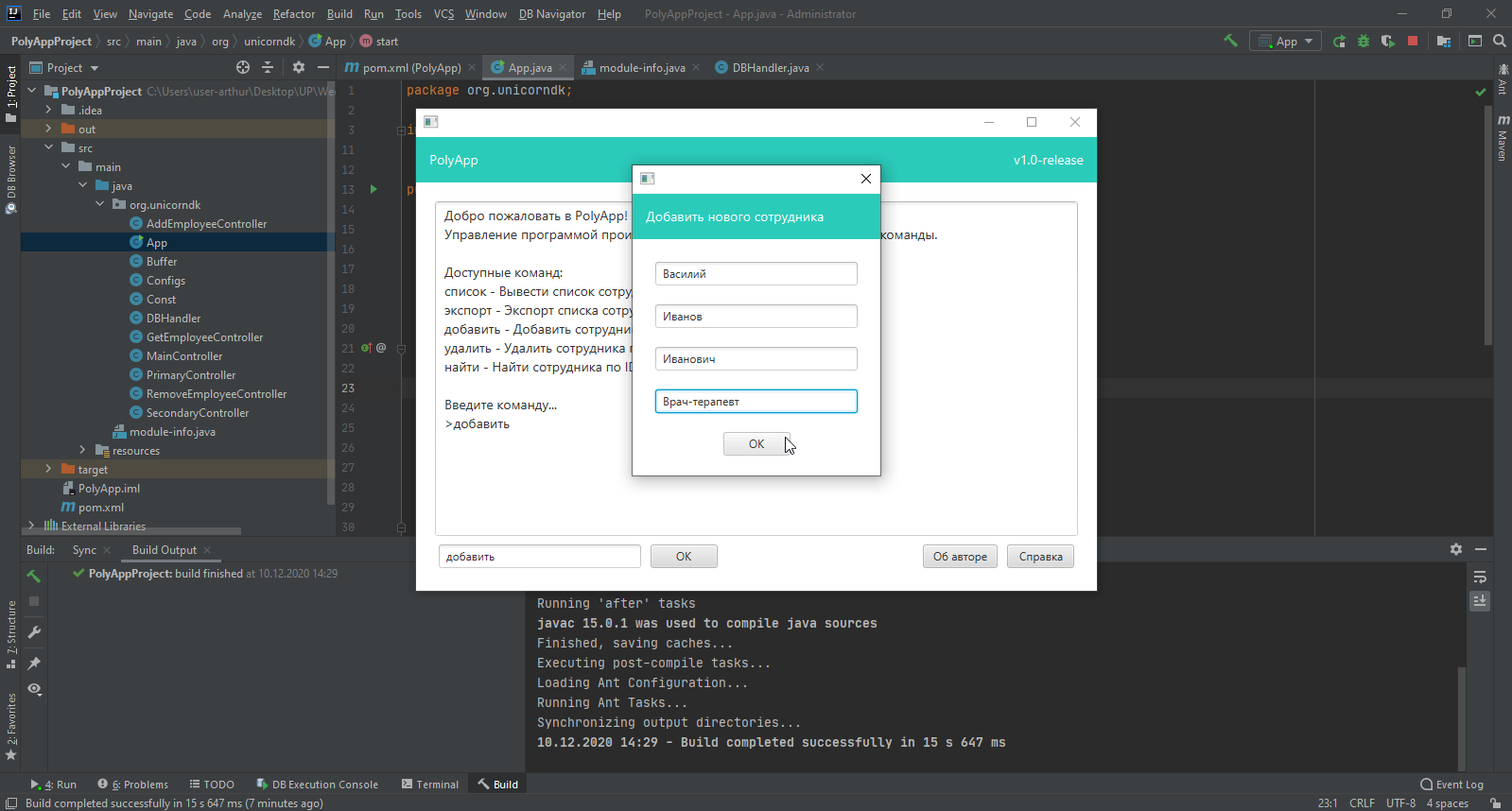


Рисунок 8 – Тестирование функциональности программы (dev версия)

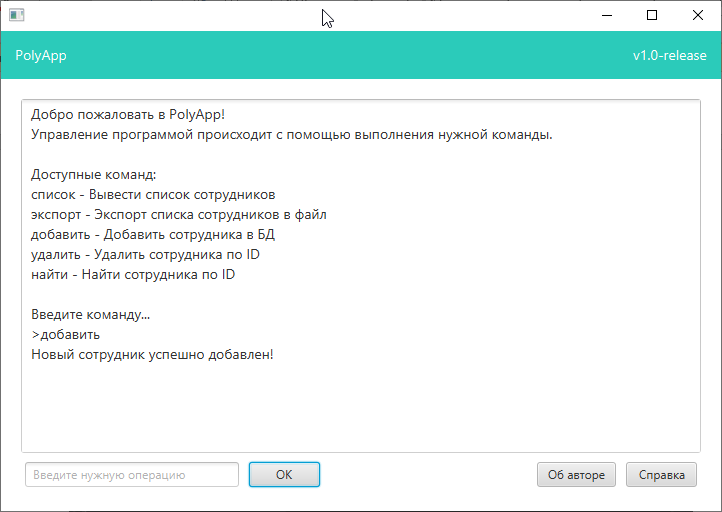


Рисунок 9 – Успешной выполнение функции (dev версия)

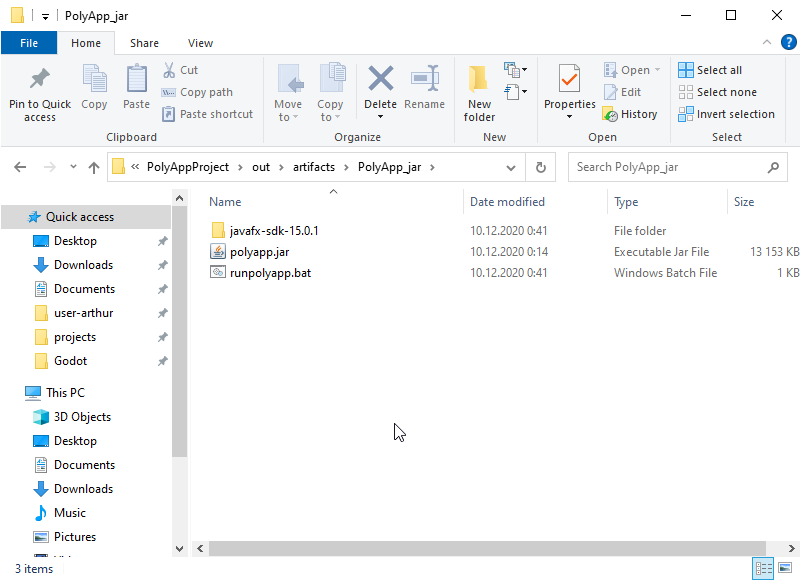


Рисунок 10 – Папка вывода

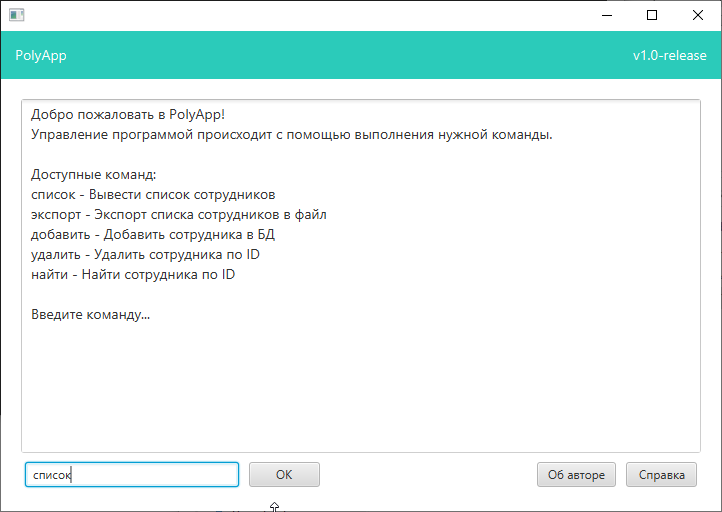


Рисунок 11 – Запуск главного окна программы (release версия)

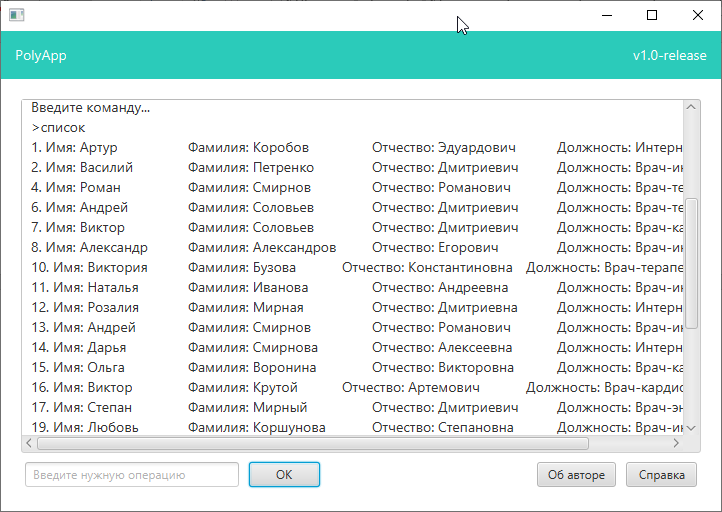


Рисунок 12 – Тестирование функциональности программы (release версия)

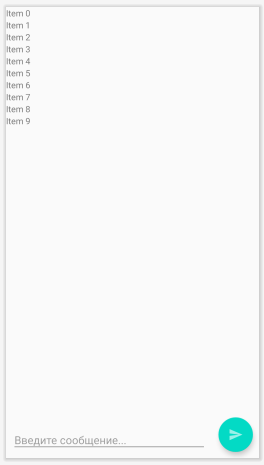


Рисунок 13 – Макет приложения в Android Studio

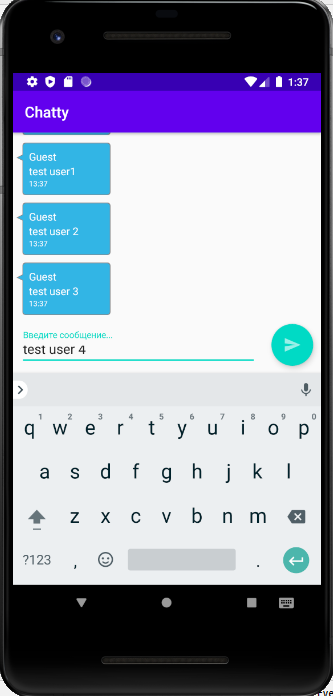
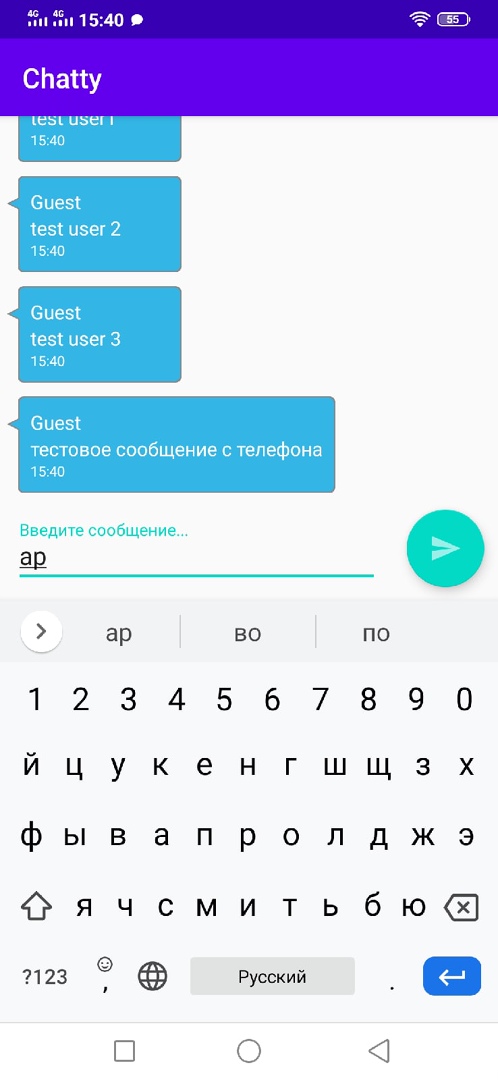
 

Рисунок 14 – Приложение запущенное на эмуляторе и на устройстве

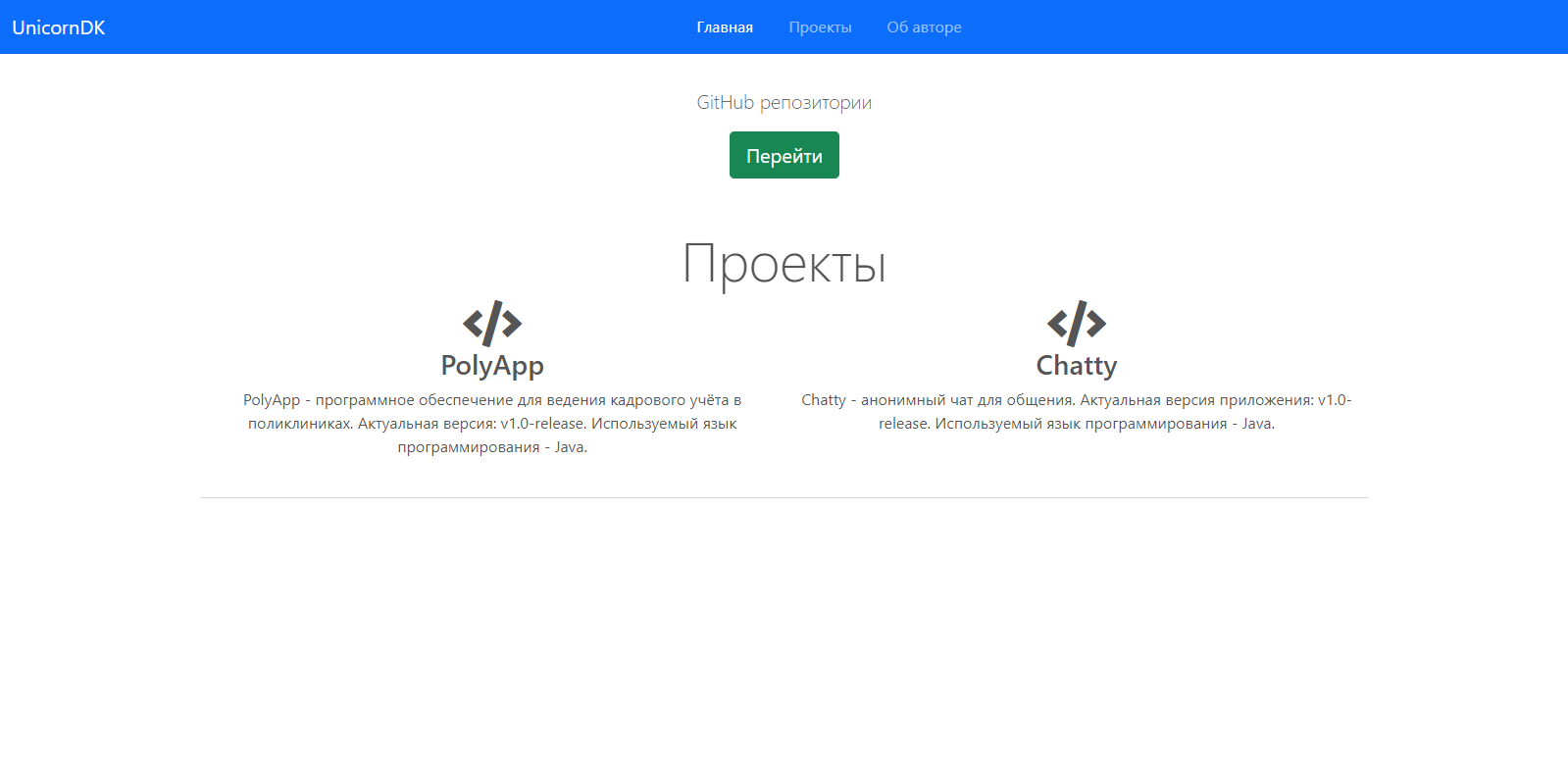


Рисунок 15 – Dekstop версия сайта

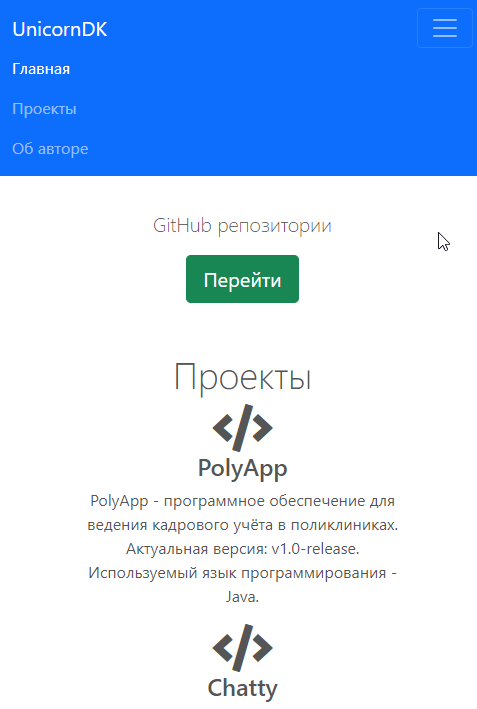


Рисунок 16 – Mobile версия сайта