

## ОГЛАВЛЕНИЕ

.....	2
ВВЕДЕНИЕ.....	3
I. СИСТЕМНЫЙ АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ.....	5
1.1 Состав, параметры объектов (сущностей) и ограничения предметной области .....	5
1.2 Состав групп пользователей системы.....	7
1.3 Обязанности и права доступа групп пользователей к информации в базе данных. ....	7
1.4 Формирование отчётности .....	8
II. ИНФОЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ .....	8
III. КОНЦЕПТУАЛЬНОЕ ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ .....	11
3.1 Преобразование сущностей в отношения (связи).....	11
3.2 Определение свойств атрибутов отношений (столбцов таблиц) .....	13
3.3 Определение внешних ключей .....	15
3.5 Концептуальная схема БД.....	16
IV. РЕАЛИЗАЦИЯ БАЗЫ ДАННЫХ .....	16
V. ЗАПРОСЫ К БАЗЕ ДАННЫХ.....	23
VI. РУКОВОДСТВО ПРОГРАММИСТА .....	24
VII. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ.....	26
ЗАКЛЮЧЕНИЕ .....	32
СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ.....	33

## **ВВЕДЕНИЕ**

### **Актуальность**

Банковское приложение является неотъемлемой частью современных финансовых технологий. В эпоху цифровизации и мобильных устройств, банки сталкиваются с растущим спросом на удобные и безопасные способы управления финансами. Разрабатываемая в данной работе система отражает работу небольшого банковского приложения, а также может помочь в автоматизации некоторых бизнес-процессов банка

### **Цели и задачи курсовой работы:**

Целью курсовой работы является приобретение студентами навыков: применения теоретических знаний и практических умений для самостоятельного анализа и формулирования задачи повышения эффективности деятельности организации или предприятия; разработки информационного и программного обеспечения автоматизации деятельности специалистов организации или предприятия; грамотного и последовательного изложения материала проведенных исследований.

Задачи курсовой работы состоят в овладении практическими навыками выполнения следующих этапов проектирования и реализации проекта:

1. Системный анализ и словесное описание информационных объектов предметной области.
2. Проектирование инфологической модели предметной области - формализованное описание объектов предметной области в терминах некоторой семантической модели, например, в терминах ER-модели.
3. Концептуальное проектирование базы данных в рамках реляционной модели данных.
4. Реализация проекта и создание клиентского приложения для работы с базой данных.

### **Требования к приложению:**

Приложение базы данных должно:

- обеспечивать авторизацию и разделение прав выделенных пользователей;
- обеспечивать выполнение всех задач пользователей, описанных в системном анализе предметной области;
- позволять вводить новые записи в любую таблицу базы данных, редактировать и удалять записи таблиц с обеспечением целостности данных;
- позволять выполнять поиск и/или фильтрацию данных таблиц;
- позволять выполнять некоторые запросы (не менее 5 запросов);
- иметь дружественный, интуитивно- и профессионально-понятный интерфейс.

### **Краткое содержание разделов курсовой:**

*Раздел «1. Системный анализ предметной области»* содержит словесное описание предметной области.

*Раздел «2. Инфологическое проектирование предметной области»* содержит результаты проектирования по методике, изученной на практическом занятии 1: сущности, атрибуты каждой сущности, ограничения на информацию каждой сущности, связи между сущностями.

*Раздел «3. Концептуальное проектирование базы данных»* содержит результаты проектирования по методике, изученной на практическом занятии 2.

*Раздел «4. Реализация базы данных»* содержит обоснование выбора СУБД для реализации базы данных, а также команды SQL для создания базы данных и всех ее объектов – таблиц, индексов, представлений, триггеров и т.д.

*Раздел «5. Запросы к базе данных»* должен содержать словесную формулировку каждого запроса на выборку данных из таблиц, соответствующую команду SQL и результаты ее выполнения.

*Раздел «6. Руководство программиста»* содержит список программных и инструментальных средств, выбранных для реализации приложения, а также описание архитектуры и основных модулей разработанного приложения.

*Раздел «7. Руководство пользователя»* содержит описание работы пользователя с разработанным приложением, включая скриншоты форм.

*Раздел «Заключение»* содержит основные выводы о проделанной работе и достигнутым результатам.

*Раздел «Список литературы»* содержит библиографические записи, оформленные в соответствии с требованиями ГОСТ 7.1-2003.

## **I. СИСТЕМНЫЙ АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ**

### **1.1 Состав, параметры объектов (сущностей) и ограничения предметной области**

Стандартное банковское приложение должно вести учёт счетов пользователей (кредитных, дебетовых), переводов, счетов на оплату, а также встреч с представителями банка

Каждый пользователь имеет:

- имя;
- фамилия;
- отчество;
- номер телефона;
- электронную почту;
- логин;
- пароль;
- один или несколько счетов;
- встречи с представителями банка;
- счета на оплату.

У каждого пользователь может быть несколько счетов в банке кредитных и дебетовых. Каждый счёт имеет:

- идентификационный номер;
- дата открытия;
- баланс;

- тип счёта (кредитный / дебетовый);
- дополнительная информация;
- транзакции;
- счета на оплату.

Каждый счёт может иметь несколько счетов на оплату и транзакций  
Следующая информация хранится о транзакций:

- дата перевода;
- дополнительная информация;
- сумма перевода;
- номер счёта отправителя;
- номер счёта получателя.

Счёт на оплату хранит в себе следующую информацию:

- дата создания;
- описание услуги;
- номер счёта для оплаты;
- статус (оплачено / не оплачено);
- дополнительная информация

Также отдельно у администратор системы (пользователь с особым уровнем доступа) может назначить встречу с представителями банка. Каждая встреча хранит в себе следующую информацию:

- дата встречи;
- локация;
- описание встречи;
- подтверждение встречи.

Клиент и администратор имеют аккаунты с соответствующими правами доступа, описанными в пункте 1.3.

Дополнительно вводятся следующие ограничения:

1. Поля дополнительной информации могут быть не заполнены.
2. У клиентов может не быть фамилии.

3. Все даты заносятся в формате YYYY-MM-DD

## **1.2 Состав групп пользователей системы**

С информацией работают следующие группы пользователей:

— администратор;

— клиент.

## **1.3 Обязанности и права доступа групп пользователей к информации в базе данных.**

Администратор выполняет следующие действия:

1. Просмотр, редактирование и удаление информации о всех пользователях системы, а также добавление новых пользователей с разными ролями.
2. Просмотр, редактирование и удаление информации о всех встречах в системе, а также добавление новых.
3. Просмотр, редактирование и удаление информации о всех банковских счетах, а также добавление новых.
4. Просмотр, редактирование и удаление информации о всех типах банковских счетов, а также добавление новых.
5. Просмотр, редактирование и удаление информации о всех типах транзакций, а также добавление новых.
6. Просмотр, редактирование и удаление информации о всех типах счетов на оплату, а также добавление новых.

Пользователь может:

1. Просмотреть информацию о транзакциях.
2. Просмотреть описание каждого своего счёта
3. Просмотреть и отредактировать свой профиль
4. Просмотреть информацию о счетах на оплату, а также выставить счёт другому пользователю
5. Перевести сумму со своего счёта на любой счёт пользователя системы

## 1.4 Формирование отчётности

Система может выводить следующие отчёты:

1. Вывод всех пользователей системы.
2. Вывод всех назначенных встреч в системе
3. Вывод всех банковских счетов
4. Вывод всех типов банковских счетов
5. Вывод всех транзакций в системе

## II. ИНФОЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

Результаты инфологического проектирования предметной области представлены в таблицах 2.1 – 2.3 и на рисунке 2.1.

**Таблица 2.1 - Перечень атрибутов**

Обозначение атрибута	Атрибут	Примечание
X1	ID счёта	
X2	Идентификационный номер	Уникальное
X3	Дата открытия счёта	YYYY-MM-DD
X4	Дополнительная информация	Может не быть
X5	Баланс	
X6	Тип счёта	
X7	ID пользователя-владельца	
X8	ID типа счета	
X9	Наименование типа счёта	
X10	ID встречи	
X11	Дата встречи	YYYY-MM-DD
X12	Локация встречи	
X13	Описание встречи	
X14	Подтверждение встречи	

X15	ID пользователя для встречи	
X16	ID счета на оплату	
X17	Дата создания счета	YYYY-MM-DD
X18	Описание услуги	
X19	Детали счёта на оплату	Может не быть
X20	Стоимость	
X21	Статус счёта	
X22	ID банковского счёта для оплаты	
X23	ID пользователя-отправителя	
X24	ID пользователя-получателя	
X25	ID транзакции	
X26	Дата перевода	YYYY-MM-DD
X27	Дополнительная информация	Может не быть
X28	ID пользователя-отправителя	
X29	ID пользователя-получателя	
X30	ID пользователя	
X31	Электронная почта	Уникальное
X32	Имя	
X33	Фамилия	Может не быть
X34	Отчество	
X35	Логин пользователя	Уникальное
X36	Пароль пользователя	
X37	Роль пользователя	
X38	Номер телефона	
X39	Сумма перевода	



**Таблица 2.2 - Определение сущностей**

Обозначение сущности	Название	Состав атрибутов	Первичный ключ
Y1	Банковский счёт	X1-X7	X1
Y2	Тип счёта	X8-X9	X8
Y3	Встреча	X10-X15	X10
Y4	Счет на оплату	X16-X24	X16
Y5	Транзакция	X25-X29, X39	X25
Y6	Пользователь	X30-X38	X30

**Таблица 2.3 - Определение связи между сущностями**

Сущность 1	Сущность 2	Тип связи	Обязательность
Y1	Y2	M:1	1-0
Y1	Y4	1:M	0-1
Y1	Y5	1:M	0-1
Y1	Y6	M:1	1-1
Y6	Y3	1:M	0-1
Y6	Y4	1:M	0-1

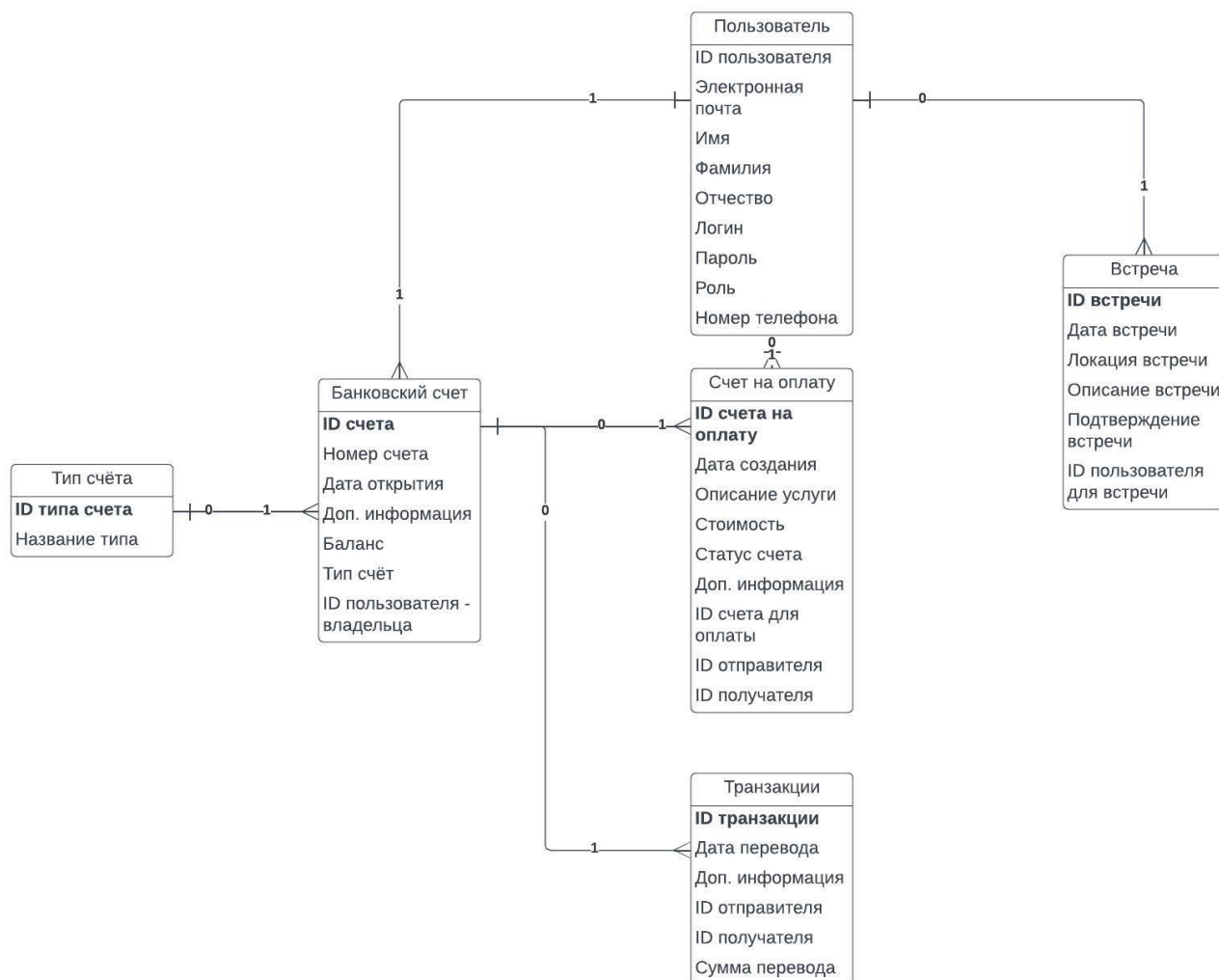
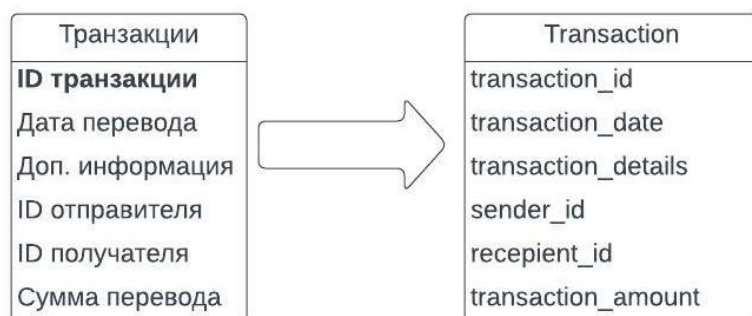


Рисунок 2.1 – Инфологическая схема

### III. КОНЦЕПТУАЛЬНОЕ ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ

#### 3.1 Преобразование сущностей в отношения (связи)

Результат преобразования сущностей в отношения представлен на рис. 3.1.



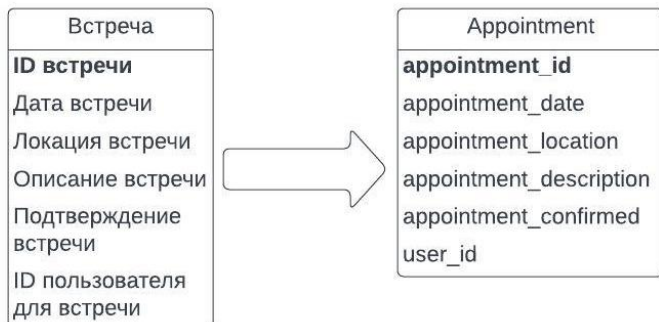
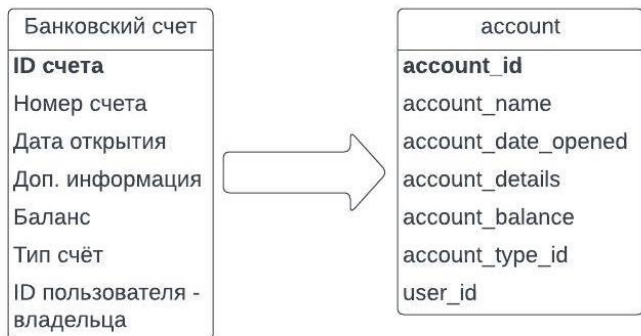
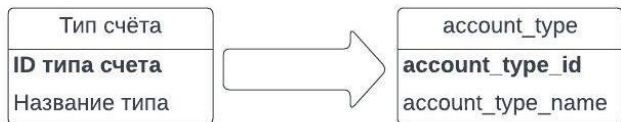
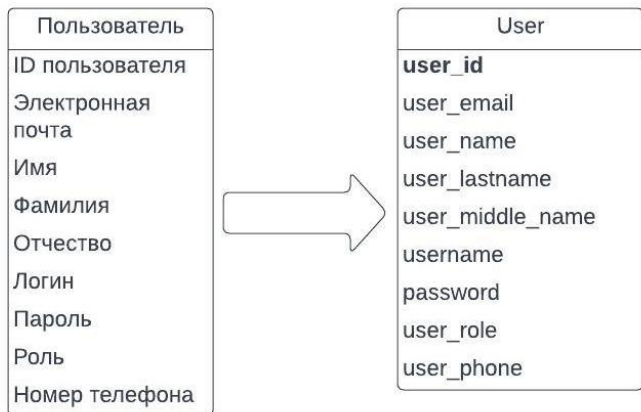


Рис. 3.1 – Преобразование сущностей в отношения

### 3.2 Определение свойств атрибутов отношений (столбцов таблиц)

Первичные ключи всех отношений, кроме account, по умолчанию имеют значение nextval('генератор последовательности '::regclass). Свойства атрибутов отношений представлены в таблицах 3.2.1-3.2.10.

**Таблица 3.2.1 - Свойства атрибутов отношения «account\_type»**

Атрибут	Тип (размер)	Уникальность	Null	Ограничения
id	integer	PK	Not null	
name	text		Not null	Unique

**Таблица 3.2.2 - Свойства атрибутов отношения «account»**

Атрибут	Тип (размер)	Уникальность	Null	Ограничения
id	integer	PK	Not null	
name	text		Not null	Unique
date_opened	date		Not null	
details	text			
balance	numeric(38,2)		Not null	
account_type_id	integer	FK	Not null	
user_id	integer	FK	Not null	

**Таблица 3.2.3 - Свойства атрибутов отношения «appointment»**

Атрибут	Тип (размер)	Уникальность	Null	Ограничения
id	integer	PK	Not null	
confirmed	boolean		Not null	
location	text		Not null	
description	text		Not null	
date	date		Not null	
user_id	integer	FK	Not null	

**Таблица 3.2.4 - Свойства атрибутов отношения таблицы «transaction»**

Атрибут	Тип (размер)	Уникальность	Null	Ограничения
id	integer	PK	Not null	
details	text			
date	date		Not null	
transfer_amount	numeric(38,2)		Not null	
recipient_id	integer	FK	Not null	
sender_id	integer	FK	Not null	

**Таблица 3.2.5 - Свойства атрибутов отношения таблицы «user»**

Атрибут	Тип (размер)	Уникальность	Null	Ограничения
id	integer	PK	Not null	
email	text		Not null	Unique
name	text		Not null	
lastname	text		Not null	
middle_name	text			
username	text		Not null	Unique
password	text		Not null	
role	text		Not null	
phone	text		Not null	

**Таблица 3.2.6 - Свойства атрибутов отношения таблицы «bill»**

Атрибут	Тип (размер)	Уникальность	Null	Ограничения
id	integer	PK	Not null	
date_created	date		Not null	Unique
details	numeric(38,2)		Not null	
price	text		Not null	
service	text			
status	text		Not null	Unique
credit_account_id	integer	FK	Not null	
recipient_id	integer	FK	Not null	
sender_id	integer	FK	Not null	

### 3.3 Определение внешних ключей

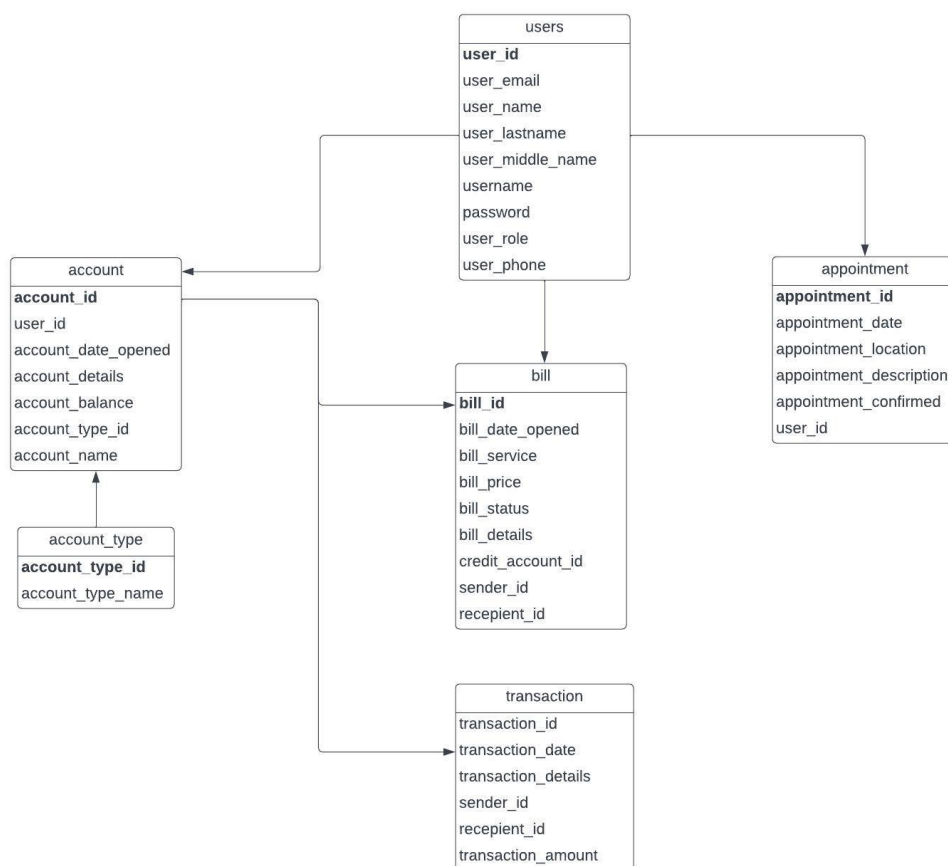
Перечень внешних ключей представлен в таблице 3.3.1.

**Таблица 3.3.1 – Перечень внешних ключей**

Дочернее отношение	Внешний ключ	Тип (размер)	Null	Ссылка
account	account_type_id	integer	Not null	account_type
account, appointment	user_id	integer	Not null	user
bill	credit_account_id	integer	Not null	account
transaction, bill	recipient_id	integer	Not null	user
transaction, bill	sender_id	integer	Not null	user

### 3.5 Концептуальная схема БД

Концептуальная модель реализованной базы данных представлена на рис. 3.5.1.



## IV. РЕАЛИЗАЦИЯ БАЗЫ ДАННЫХ

В данной работе используется PostgreSQL. Выбор был обусловлен бесплатностью, поддержкой БД неограниченного размера, мощными и надёжными механизмами транзакций и репликации, а также огромным количеством документации.

Для создания базы данных используется Java Persistence API и её реализация Hibernate. Описание атрибутов и таблиц происходит в классах сущностей.

Формирование таблиц базы данных происходит путём создания классов POJO и специальных аннотаций.

@Entity – даёт программе понять, что должны формироваться сущности.

@Id – отвечает за первичный ключ.

@Column – определяет имя, уникальность, обязательность и тип атрибута.

@ManyToOne – формирует отношение «многие к одному»

@OneToMany – формирует отношение «один ко многим»

@JoinColumn – определяет внешний ключ.

@JoinTable – создаёт таблицу для связи «многие ко многим» и определяет её поля.

@Getter, @Setter, @NoArgsConstructor автоматически создают геттеры, сеттеры, конструктор без параметров.

В свойства @OneToOne @ManyToOne cascade = CascadeType.REMOVE означает каскадное удаление записей.

Программная реализация показана на рисунках 4.1 – 4.6.

На рисунке 4.11 указано исходное состояние базы данных после создания сущностей.

На рисунке 4.12 показаны исходные данные базы данных.

На рисунках 4.13 – 4.23 показано состояние таблиц базы данных после загрузки исходных данных.

```
@Data
@Entity
public class Account {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;

    @Column(columnDefinition="TEXT", nullable = false, unique = true)
    private String name;

    @Column(nullable = false)
    private Date dateOpened;

    @Column(columnDefinition="TEXT", nullable = true)
    private String accountDetails;

    @Column(nullable = false)
    private BigDecimal accountBalance;

    @ManyToOne
    @JoinColumn(name = "account_type_id", nullable = false)
    private AccountType accountType;

    @ManyToOne
    @JoinColumn(name = "user_id", nullable = false)
    private User user;

    @OneToMany(mappedBy = "recipient", cascade = CascadeType.ALL)
    private List<Transaction> recipientList;

    @OneToMany(mappedBy = "sender", cascade = CascadeType.ALL)
    private List<Transaction> senderList;

    @OneToMany(mappedBy = "account")
```



```
private List<Bill> bills;
```

Рисунок 4.1. Программная реализация таблицы «account»

```
@Data
@Entity
public class AccountType {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;

    @Column(columnDefinition="TEXT", nullable = false, unique = true)
    private String name;

    @OneToMany(mappedBy = "accountType")
    private List<Account> accounts;
}
```

Рисунок 4.2. Программная реализация таблицы «accout\_type»

```
@Data
@Entity
public class Appointment {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;

    @Column(nullable = false)
    private Date date;

    @Column(columnDefinition="TEXT", nullable = false)
    private String location;

    @Column(columnDefinition="TEXT", nullable = false)
    private String description;

    @Column(nullable = false)
    private Boolean confirmed;

    @ManyToOne
    @JoinColumn(name = "user_id", nullable = false)
    private User user;
}
```

Рисунок 4.3. Программная реализация таблицы «appointment»

```
@Data
@Entity
public class Bill {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;

    @Column(nullable = false)
    private Date dateCreated;

    @Column(columnDefinition="TEXT", nullable = false)
    private String service;

    @Column(columnDefinition="TEXT", nullable = true)
    private String details;

    @Column(nullable = false)
    private BigDecimal price;

    @Column(columnDefinition="TEXT", nullable = false)
    private String status;

    @ManyToOne
    @JoinColumn(name = "credit_account_id", nullable = false)
    private Account account;

    @ManyToOne
    @JoinColumn(name = "creator_id", nullable = false)
```

```

    private User creator;

    @ManyToOne
    @JoinColumn(name = "recipient_id", nullable = false)
    private User recipient;
}

```

Рисунок 4.4. Программная реализация таблицы «bill»

```

@Data
@NoArgsConstructor
@Entity
public class Transaction {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;

    @Column(nullable = false)
    private Date date;

    @Column(columnDefinition="TEXT", nullable = true)
    private String details;

    @Column(nullable = false)
    private BigDecimal transferAmount;

    @ManyToOne
    @JoinColumn(name = "recipient_id", nullable = false, columnDefinition = "Integer Check (recipient_id != sender_id)")
    private Account recipient;

    @ManyToOne
    @JoinColumn(name = "sender_id", nullable = false)
    private Account sender;
}

```

Рисунок 4.5. Программная реализация таблицы «transaction»

```

@Getter
@Setter
@NoArgsConstructor
@Entity
@Table(name = "users")
public class User implements UserDetails {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;

    @Column(columnDefinition="TEXT", nullable = false, unique = true)
    private String email;

    @Column(columnDefinition="TEXT", nullable = false)
    private String firstname;

    @Column(columnDefinition="TEXT", nullable = false)
    private String lastname;

    @Column(columnDefinition="TEXT", nullable = true)
    private String middleName;

    @Column(columnDefinition="TEXT", nullable = false)
    private String phone;

    @Column(columnDefinition="TEXT", nullable = false, unique = true)
    private String username;

    @Column(columnDefinition="TEXT", nullable = false)
    private String password;

    @Column(columnDefinition="TEXT", nullable = false)
    private String role;

    @OneToMany(mappedBy = "user", cascade = CascadeType.ALL, fetch = FetchType.LAZY)
}

```

```

private List<Appointment> appointmentList;

@OneToMany(mappedBy = "user", cascade = CascadeType.ALL, fetch = FetchType.LAZY)
private List<Account> accountList;

@OneToMany(mappedBy = "creator", cascade = CascadeType.ALL, fetch = FetchType.LAZY)
private List<Bill> creatorsBills;

@OneToMany(mappedBy = "recipient", cascade = CascadeType.ALL, fetch = FetchType.LAZY)
private List<Bill> recipientsBills;

```

Рисунок 4.6. Программная реализация таблицы «users»

Список отношений			
Схема	Имя	Тип	Владелец
public	account	таблица	postgres
public	account_type	таблица	postgres
public	appointment	таблица	postgres
public	bill	таблица	postgres
public	transaction	таблица	postgres
public	users	таблица	postgres

Рисунок 4.7.Состояние базы данных

```

INSERT INTO users (email, firstname, lastname, middle_name, password, phone, role, username)
VALUES ('inna9544@yandex.ru', 'Инна', 'Смолянинова', 'Порфирьевна', 'qwerty', '9178272152',
'ROLE_USER', 'inna9544');
INSERT INTO users (email, firstname, lastname, middle_name, password, phone, role, username)
VALUES ('evgeniy1991@hotmail.com', 'Евгений', 'Куаньшбаев', 'Никитьевич', 'qwerty',
'9427135068', 'ROLE_USER', 'evgeniy1991');
INSERT INTO users (email, firstname, lastname, middle_name, password, phone, role, username)
VALUES ('viktor.priluckiy@ya.ru', 'Виктор', 'Прилуцкий', 'Феокистович', 'qwerty',
'9993813999', 'ROLE_USER', 'priluckiy');
INSERT INTO users (email, firstname, lastname, middle_name, password, phone, role, username)
VALUES ('aleksandr1995@outlook.com', 'Александр', 'Косма', 'Иванович', 'qwerty', '9548812811',
'ROLE_USER', 'aleksandr1995');
INSERT INTO users (email, firstname, lastname, middle_name, password, phone, role, username)
VALUES ('anastasiya09031982@rambler.ru', 'Анастасия', 'Рабиновича', 'Ираклиевна', 'qwerty',
'9427135068', 'ROLE_ADMIN', 'anastasiya09031982');

INSERT INTO account_type (name) VALUES ('Дебитовый');
INSERT INTO account_type (name) VALUES ('Кредитный');

INSERT INTO appointment (confirmed, date, description, location, user_id) VALUES (true, '2022-
12-12', 'Принести с собой паспорт', 'Пушкина 21', 1);
INSERT INTO appointment (confirmed, date, description, location, user_id) VALUES (true, '2022-
12-13', 'Принести с собой паспорт', 'Пушкина 21', 2);
INSERT INTO appointment (confirmed, date, description, location, user_id) VALUES (false,
'2022-12-14', 'Принести с собой паспорт', 'Пушкина 21', 3);
INSERT INTO appointment (confirmed, date, description, location, user_id) VALUES (true, '2022-
12-15', 'Принести с собой паспорт', 'Пушкина 21', 4);
INSERT INTO appointment (confirmed, date, description, location, user_id) VALUES (false,
'2022-12-16', 'Принести с собой паспорт', 'Пушкина 21', 5);

INSERT INTO account (account_balance, account_details, date_opened, name, account_type_id,
user_id) VALUES (10, '', '2022-12-12', '4000 0012 3456 7899', 1, 1);
INSERT INTO account (account_balance, account_details, date_opened, name, account_type_id,
user_id) VALUES (0, '', '2022-12-12', '4000 0012 3456 7900', 2, 1);
INSERT INTO account (account_balance, account_details, date_opened, name, account_type_id,

```

```

user_id) VALUES (0, '', '2022-12-12', '4000 0012 3456 7901', 1, 2);
INSERT INTO account (account_balance, account_details, date_opened, name, account_type_id,
user_id) VALUES (0, '', '2022-12-12', '4000 0012 3456 7902', 2, 3);
INSERT INTO account (account_balance, account_details, date_opened, name, account_type_id,
user_id) VALUES (0, '', '2022-12-12', '4000 0012 3456 7903', 1, 4);
INSERT INTO account (account_balance, account_details, date_opened, name, account_type_id,
user_id) VALUES (0, '', '2022-12-12', '4000 0012 3456 7904', 2, 4);
INSERT INTO account (account_balance, account_details, date_opened, name, account_type_id,
user_id) VALUES (0, '', '2022-12-12', '4000 0012 3456 7905', 2, 5);

INSERT INTO bill (date_created, details, price, service, status, credit_account_id,
creator_id, recipient_id) VALUES ('2022-12-12', '', 1000, 'Подключение ТВ', 'Не оплачено', 3,
2, 1);

```

Рисунок 4.8. Файл для загрузки изначальных данных

```
select * from account;
```

id	account_balance	account_details	date_opened	name	account_type_id	user_id
1	10.00		2022-12-12	4000 0012 3456 7899	1	1
2	0.00		2022-12-12	4000 0012 3456 7900	2	1
3	0.00		2022-12-12	4000 0012 3456 7901	1	2
4	0.00		2022-12-12	4000 0012 3456 7902	2	3
5	0.00		2022-12-12	4000 0012 3456 7903	1	4
6	0.00		2022-12-12	4000 0012 3456 7904	2	4
7	0.00		2022-12-12	4000 0012 3456 7905	2	5
VeselovNF   ec1e061c5   ROLE_STUDENT						

Рисунок 4.9. Состояние таблицы «account» после загрузки данных

```
select * from account_type;
```

id	name
1	Дебитовый
2	Кредитный6

Рисунок 4.10. Состояние таблицы «account\_type» после загрузки данных

```
select * from appointment;
```

id	confirmed	date	description	location	user_id
----	-----------	------	-------------	----------	---------

id	confirmed	date	description	location	user_id
----	-----------	------	-------------	----------	---------

1	t	2022-12-12	Принести с собой паспорт	Пушкина 21	1
---	---	------------	--------------------------	------------	---

2	t	2022-12-13	Принести с собой паспорт	Пушкина 21	2
---	---	------------	--------------------------	------------	---

3	f	2022-12-14	Принести с собой паспорт	Пушкина 21	3
---	---	------------	--------------------------	------------	---

4	t	2022-12-15	Принести с собой паспорт	Пушкина 21	4
---	---	------------	--------------------------	------------	---

5	f	2022-12-16	Принести с собой паспорт	Пушкина 21	5
---	---	------------	--------------------------	------------	---

Рисунок 4.11. Состояние таблицы «appointment» после загрузки данных

```
select * from bill;
```

id	date_created	details	price	service	status	credit_account_id	creator_id	recipient_id
----	--------------	---------	-------	---------	--------	-------------------	------------	--------------

id	date_created	details	price	service	status	credit_account_id	creator_id	recipient_id
----	--------------	---------	-------	---------	--------	-------------------	------------	--------------

1	2022-12-12		1000.00	Подключение ТВ	Не оплачено	3	2	1
---	------------	--	---------	----------------	-------------	---	---	---

Рисунок 4.12. Состояние таблицы «bill» после загрузки данных

```
select * from users;
```

id	email	firstname	lastname	middle_name	password	phone
role	username					

id	email	firstname	lastname	middle_name	password	phone
role	username					

1	inna9544@yandex.ru	Инна	Смолянинова	Порфнрьевна	qwerty	9178272152
ROLE_USER	inna9544					

2	evgeniy1991@hotmail.com	Евгений	Куанышбаев	Никитьевич	qwerty	9427135068
ROLE_USER	evgeniy1991					

3	viktor.priluckiy@ya.ru	Виктор	Прилуцкий	Феоктистович	qwerty	9993813999
ROLE_USER	priluckiy					

4	aleksandr1995@outlook.com	Александр	Косма	Иванович	qwerty	9548812811
ROLE_USER	aleksandr1995					

5	anastasiya09031982@rambler.ru	Анастасия	Рабиновича	Ираклиевна	qwerty	9427135068
ROLE_ADMIN	anastasiya09031982					

Рисунок 4.13. Состояние таблицы «users» после загрузки данных

## V. ЗАПРОСЫ К БАЗЕ ДАННЫХ

Основные запросы формируются фреймворком. Для создания репозитория конкретной сущности нужно создать новый класс и унаследовать его от `CrudRepository<T, ID>`. При необходимости можно создать дополнительные запросы прямо из имени методов. Для этого используется механизм префиксов `find...By`, `read...By`, `query...By`, `count...By`, и `get...By`, далее от префикса метода начинается разбор остальной части. Вводное предложение может содержать дополнительные выражения, например, `Distinct`. Далее первый `By` действует как разделитель, чтобы указать начало фактических критериев. Можно определить условия для свойств сущностей и объединить их с помощью `And` и `Or`. Интерфейс стандартной реализации показан на рисунке 5.1.

```
public interface CrudRepository<T, ID> extends Repository<T, ID> {
    <S extends T> S save(S entity); // Сохраняет экземпляр сущности
    <S extends T> Iterable<S> saveAll(Iterable<S> entities); // Сохраняет коллекцию экземпляров
    Optional<T> findById(ID id); // Возвращает экземпляр по ключу
    boolean existsById(ID id); // Проверяет на существование экземпляра
    Iterable<T> findAll(); // Возвращает коллекцию экземпляров
    Iterable<T> findAllById(Iterable<ID> ids); // Возвращает коллекцию экземпляров с заданным
ключом
    long count(); // Возвращает количество записей
    void deleteById(ID id); // Удаляет по ключу
    void delete(T entity); // Удаляет переданный экземпляр
    void deleteAllById(Iterable<? extends ID> ids); // Удаляет все экземпляры с заданным ключом
    void deleteAll(Iterable<? extends T> entities); // Удаляет коллекцию экземпляров
    void deleteAll(); // Удаляет все записи в таблице
}
```

Рисунок 5.1. Интерфейс `CrudRepository`

## VI. РУКОВОДСТВО ПРОГРАММИСТА

Приложение написано на Java версии 17. Для реализации веб-приложения использованы следующие библиотеки и фреймворки:

- Spring Boot 2.7.5.
- Spring Security
- Spring MVC
- Lombok
- Thymeleaf
- Bootstrap 5
- Simple-DataTables

Приложение состоит из следующих уровней:

- Уровень представления. Представляет контент конечному пользователю через графический интерфейс. Веб-страницы отправляются клиенту по запросу через контроллеры. Клиентом является веб-браузер.
- Уровень бизнес-логики. Управляет обработкой и валидацией данных. Состоит из сервисных классов. Может обращаться к уровню данных.
- Уровень данных. Организует хранение и извлечение данных из базы данных. Определяет реализацию основных CRUD операций. Представлен в виде репозитория.

Приложение состоит из следующих пакетов:

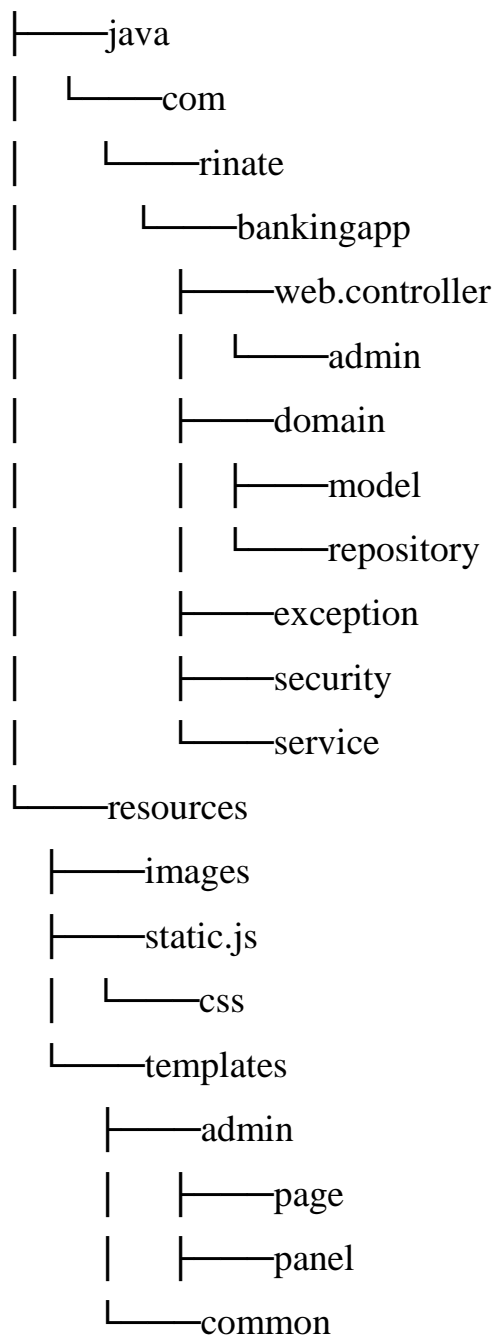
- controller – содержит классы контроллеров.
- domain.model – содержит POJO (простые java объекты) с аннотациями для создания сущностей в базе данных.
- domain.repository – содержит репозитории для работы с базой данных.
- security – содержит классы для настройки Spring Security и создания собственной системы авторизации.
- web.controller – содержит классы для взаимодействия с уровнем бизнес логики и уровнем представления

Помимо пакетов приложение содержит статические html страницы и их стили, а также js скрипты для придания интерактивности в папке static.

В папке templates лежат шаблоны, обрабатываемые thymeleaf и выдаваемые клиенту.

В файле application.properties содержатся настройки для подключения к базе данных, а также дополнительные настройки, необходимые для корректной работы приложения.

Структура проекта:





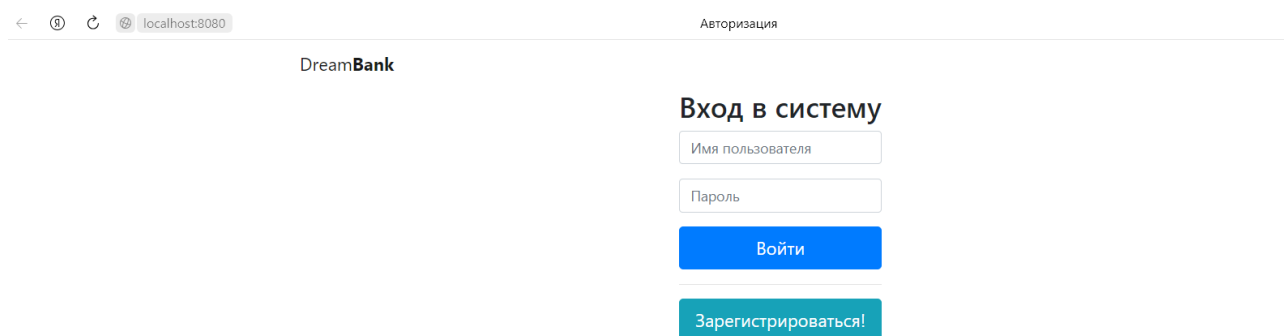
## VII. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

С целью приобретения навыков разработки информационного и программного обеспечения автоматизации деятельности специалистов организации или предприятия в рамках данного проекта было создано клиентское приложение для работы с базой данных "Банковское приложение".

Согласно техническому заданию, база данных позволяет вводить новые записи в любую таблицу базы данных, редактировать и удалять записи таблиц с обеспечением целостности данных; позволяет выполнять поиск и фильтрацию данных таблиц. Для облегчения взаимодействия пользователей с базой данных, программа также имеет дружелюбный интерфейс.

База данных содержит информацию о учениках, преподавателях, их аккаунтов, занятиях, их дней недели и тип, курсах, их категориях, уровнях и языках. Каждая из этих категорий имеют свою таблицу. Вы можете добавлять в каждую таблицу новые записи, редактировать и удалять старые при условии обеспечения целостности данных. Все действия, совершаемые в приложении, совершаются и над таблицами в самой базе данных.

Для получения доступа к функционалу, нужно перейти на сайт, где развернуто веб-приложение в персонального компьютера или другого устройства, дизайн адаптивный. В случае отсутствия авторизации будет автоматическая переадресация на страницу входа. (Рис. 7.1).



← ⓘ ↻ 🌐 localhost:8080

Авторизация

DreamBank

**Вход в систему**

Имя пользователя

Пароль

Войти

Зарегистрироваться!

Рисунок 7.1. Страница входа в систему

В зависимости от роли будут выводиться разные страницы (рис. 7.2 – 7.4).

DreamBank Пользователи Встречи Банковские счета Типы счётов Транзакции Счета Выйти

Добро пожаловать в админ панель. Переход по таблицам осуществляется через верхнее меню.

Рисунок 7.2. Интерфейс администратора

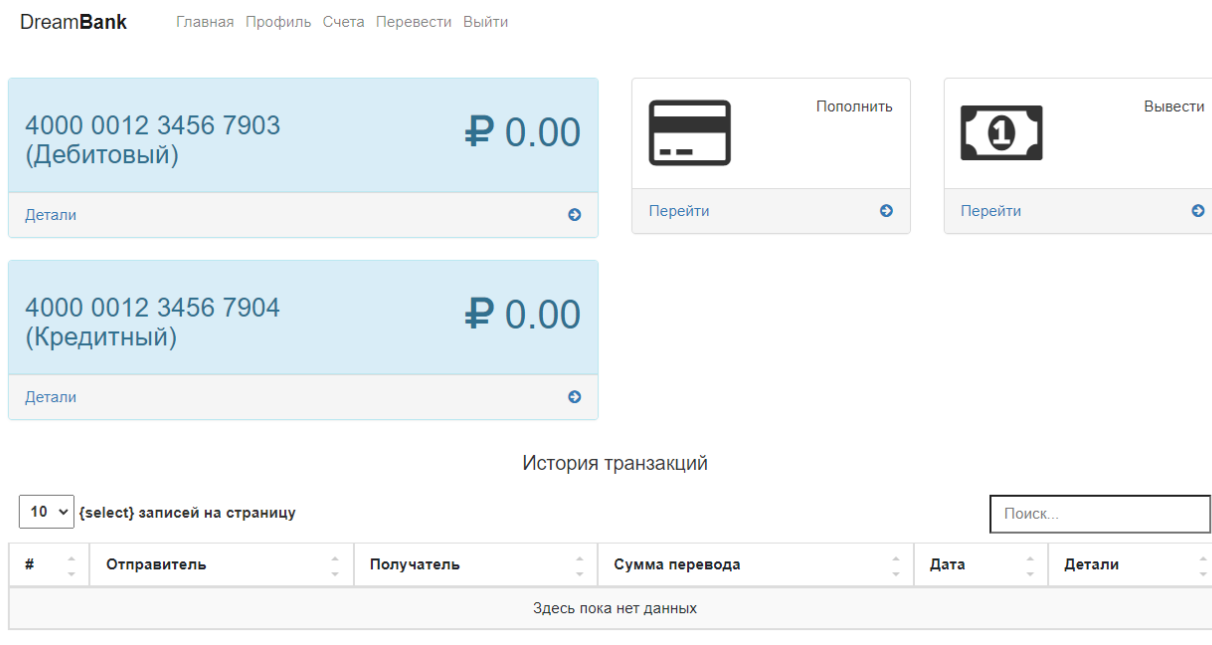


Рисунок 7.3. Интерфейс клиента

Рассмотрим возможности администратора. Администратор может удалять, изменять и редактировать любые записи в любых таблицах. Для поиска по таблице достаточно ввести в поле поиска ключевое слово и записи сами отсортируются (рис. 7.5 – 7.6).

## Пользователи

10 {select} записей на страницу

Поиск...

#	ФИО	Почта	Телефон	Логин	Пароль	Роль	Действия
1	Инна Смольянинова Порфнрьевна	inna9544@yandex.ru	9178272152	inna9544	qwerty	Пользователь	✎ □
2	Евгений Куанышбаев Никитьевич	evgeniy1991@hotmail.com	9427135068	evgeniy1991	qwerty	Пользователь	✎ □
3	Виктор Прилуцкий Феокистович	viktor.priluckiy@ya.ru	9993813999	priluckiy	qwerty	Пользователь	✎ □
4	Александр Косма Иванович	aleksandr1995@outlook.com	9548812811	aleksandr1995	qwerty	Пользователь	✎ □
5	Анастасия Рабиновича Ираклиевна	anastasiya09031982@rambler.ru	9427135068	anastasiya09031982	qwerty	Администратор	✎ □

Рисунок 7.5. Таблица «Пользователи»

## Пользователи

10 {select} записей на страницу

Евген

#	ФИО	Почта	Телефон	Логин	Пароль	Роль	Действия
2	Евгений Куанышбаев Никитьевич	evgeniy1991@hotmail.com	9427135068	evgeniy1991	qwerty	Пользователь	✎ □

Показаны записи 1 - 1 из 1

## Добавление записи

Фамилия

Фамилия

Имя

Имя

Отчество

Отчество

Почта

Почта

Телефон

Всё после +7

Логин

Рисунок 7.6. Поиск по таблице

Также можно добавить записи, заполнив форму снизу (рис. 7.7).

### Добавление записи

Фамилия

Фамилия

Имя

Имя

Отчество

Отчество

Почта

Почта

Телефон

Всё после +7

Логин

Логин

Пароль

Пароль

Роль

Пользователь

Сохранить

Рисунок 7.7. Добавление нового пользователя

В случае ошибок при заполнении на экране появится уведомляющее сообщение (рис. 7.8).

### Добавление записи

Фамилия

Фамилия

Имя

Имя

Отчество

Отчество

Почта

Почта

Телефон

Всё после +7

Логин

Логин

Пароль

Пароль

Роль

Пользователь

Сохранить

Вы пропустили это поле.

## Рисунок 7.8. Уведомление об ошибке

Большая часть страниц администратора построена похожим образом

Рассмотрим возможности клиента. У него имеется только права на просмотр счетов, с которыми он связан, на добавление нового счёта на оплату, а также на возможность делать переводы(рис. 7.9 – 7.10).

DreamBank Главная Профиль Счета Перевести Выйти

Добавление записи

**Дата**

дд. мм. гггг

**Счёт оплаты**

4000 0012 3456 7899

**Получатель**

inna9544

**Услуга**

Услуга

**Стоимость**

Стоимость

**Статус**

Оплачено

**Детали**

Детали

Счета

10  {select} записей на страницу

## Рисунок 7.9. Список счетов об оплате

1. Укажите счёт, с которого вы хотите перевести средства:

4000 0012 3456 7903

2. Укажите счёт, на который вы хотите перевести средства:

4000 0012 3456 7899

3. Укажите сумму, которую вы хотите перевести:

Сумма

0

Перевести

Рисунок 7.10. Перевод со счета клиента на любой другой счет

## **ЗАКЛЮЧЕНИЕ**

В результате выполнения данной курсовой работы была создана программа для работы с базой данных. Были приобретены навыки применения теоретических знаний и практических умений для самостоятельного анализа и формулирования задачи повышения эффективности деятельности организации или предприятия; разработки информационного и программного обеспечения автоматизации деятельности специалистов организации или предприятия; грамотного и последовательного изложения материала проведенных исследований. а также были выработаны новые навыки владения языком программирования Java.

На основе исходных данных выполнены все требования к реализации программы, получен ожидаемый результат.

## СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Карнелл, Д. Микросервисы Spring / Д. Карнелл, И. У. Санчес ; перевод с английского А. Н. Киселева. — Москва : ДМК Пресс, 2022. — 490 с. — ISBN 978-5-97060-971-2. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/241172> (дата обращения: 14.10.2022).
2. Бауэр, К. Java Persistence API и Hibernate / К. Бауэр, Г. Кинг, Г. Грегори ; под редакцией А. Н. Киселева ; перевод с английского Д. А. Зинкевич. — Москва : ДМК Пресс, 2017. — 632 с. — ISBN 978-5-97060-180-8. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/111435> (дата обращения: 14.10.2022).