

Министерство науки и высшего образования Российской Федерации
Муромский институт (филиал)
федерального государственного бюджетного образовательного учреждения
высшего образования
«Владимирский государственный университет
Имени Александра Григорьевича и Николая Григорьевича Столетовых»
(МИВлГУ)

Факультет _____ ИТ _____

Кафедра _____ ПИИ _____

КУРСОВАЯ РАБОТА

по _____ Разработке кроссплатформерных приложений _____

Тема _____ Приложение «АИС ветеринарной клиники» _____

(оценка)

Руководитель

Кульков Я.Ю.
(фамилия, инициалы)

(подпись) (дата)

Члены комиссии

Студент ПИИ-119
(группа)

(подпись) (Ф.И.О.)

Мартынов Е. С.
(фамилия, инициалы)

(подпись) (Ф.И.О.)

(подпись) (дата)

Муром 2022

В данной курсовой работе была разработана БД и АИС Ветеринарной клиники. В ходе выполнения курсовой работы выявлены требования к программе, разработаны модели данных. На основе разработанных моделей создана БД и приложение, работающие с ней. На основе разработанных моделей реализован набор классов и разработано приложение на языке программирования java в среде разработки IntelliJ IDEA 2022. На заключительном этапе работы произведено тестирование разработанного продукта.

In this course work there was a budget database and an automated information system of the Veterinary Clinic. In the course of the course work, the requirements for the program were identified, data models were developed. On the basis of the developed models, a database and an application working with it have been created. On the basis of the developed models, a set of classes was implemented and an application was developed in the java programming language in the IntelliJ IDEA 2022 development environment. At the final stage of the product operation, the developed product was tested.

задание

Содержание

Введение	6
1. Анализ технического задания	8
1.1 Аналогии.....	8
1.2 Обоснование выбора средств реализации	10
1.3 Функциональные требования.....	11
1.4 Системные требования	12
2. Разработка алгоритмов.....	13
3. Руководство программиста.....	26
4. Руководство пользователя.....	28
5. Тестирование	32
Заключение.....	38
Список используемой литературы.....	39
Приложение 1. Скриншоты программы	40
Приложение 2. Исходный код программы	44
Приложение 2. Документация JavaDoc	45

					МИВУ 09.03.04-10.000 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата				
Разраб.		Мартынов Е.С			АИС Ветеринарной клиники		Лит.	Лист
Провер.		Кульков Я.Ю.						Листов
Реценз.								
Н. Контр.								
Утверд.								
						546		
						МИ ВлГУ ПИН-119		

Введение

Автоматизированная информационная система или АИС - это совокупность различных программно - аппаратных средств, которые предназначены для автоматизации какой - либо деятельности, связанной с передачей, хранением и обработкой различной информации.

В автоматизированных информационных системах за хранение любой информации отвечают:

1. На физическом уровне:

- а. внешние накопители;
- б. встроенные устройства памяти (RAM);
- в. массивы дисков;

2. На программном уровне:

- а. СУБД;
- б. файловая система ОС;
- в. системы хранения мультимедиа, документов и т. д.

Даже сейчас во многих ветеринарных клиниках данные хранятся в различных электронных или бумажных документах. Это затрудняет быстрый доступ к необходимой для работы врачей информации. Разработка АИС должна решить данную проблему.

Эффективное управление предприятием в современных условиях невозможно без использования компьютерных технологий. Разработка автоматизированной информационной системы (АИС) ветеринарной клиник важна, так как будет в результате выполнения данной работы будут автоматизированы процессы записи информации о проведённом лечении, просмотр информации о предыдущих болезнях и назначениях питомца.

Внедрение информационных технологий в процесс работы ветеринарных клиник предусматривает обработку и анализ больших массивов данных.

Функции, которые будут автоматизированы:

					МИВУ 09.03.04-10.000 ПЗ	Лист
						6
Изм.	Лист	№ докум.	Подпись	Дата		

– В медицине важно знать сколько осмотров провёл врач за определённый период времени. АИС будет предоставлять функцию просмотра отчётности по сотруднику;

Целью курсовой работы является проектирование и разработка автоматизированной информационной системы ветеринарной клиники.

Для достижения поставленной цели были поставлены следующие задачи:

- проанализировать предметную область;
- разработать модели данных;
- реализовать базу данных;
- разработать клиентское приложение;
- протестировать программный продукт.

					МИВУ 09.03.04-10.000 ПЗ	Лист
						7
Изм.	Лист	№ докум.	Подпись	Дата		

1 Анализ технического задания.

1.1 Аналоги

Для наиболее точного анализа предоставленного технического задания были найдены аналоги разрабатываемого программного средства.

1 VetAIS

Программа «VetAIS» — предназначена для ветеринарных учреждений: ветеринарных центров, клиник, кабинетов и других ветеринарных лечебных организаций. Данная программа позволит автоматизировать работу приема клиентов ветеринарного учреждения, а также рабочие места ветеринарных врачей, благодаря ведению всестороннего учета. Имеется предварительная запись пациентов, контроль оказанных услуг, хранение информации о пациентах и всех их посещениях, а также хранение истории болезни каждого клиента со всеми рентгеновскими снимками, УЗИ и анализами. Труд ветеринарного врача упрощается с помощью системы обследований, экспресс анализов и рекомендаций. В то же время, всегда можно посмотреть историю болезни пациента и диагноза. Система напомнит администраторам о необходимости связаться с клиентом для напоминания о записи, произведенных анализах, пошлет, при необходимости, письмо по электронной почте и/или сообщит с помощью СМС. В результате внедрения данной программы повысится производительность труда персонала, общая эффективность работы организации и как следствие, прибыль. На рисунке 1 представлен скриншот работы программы.

					МИВУ 09.03.04-10.000 ПЗ	Лист
						8
Изм.	Лист	№ докум.	Подпись	Дата		

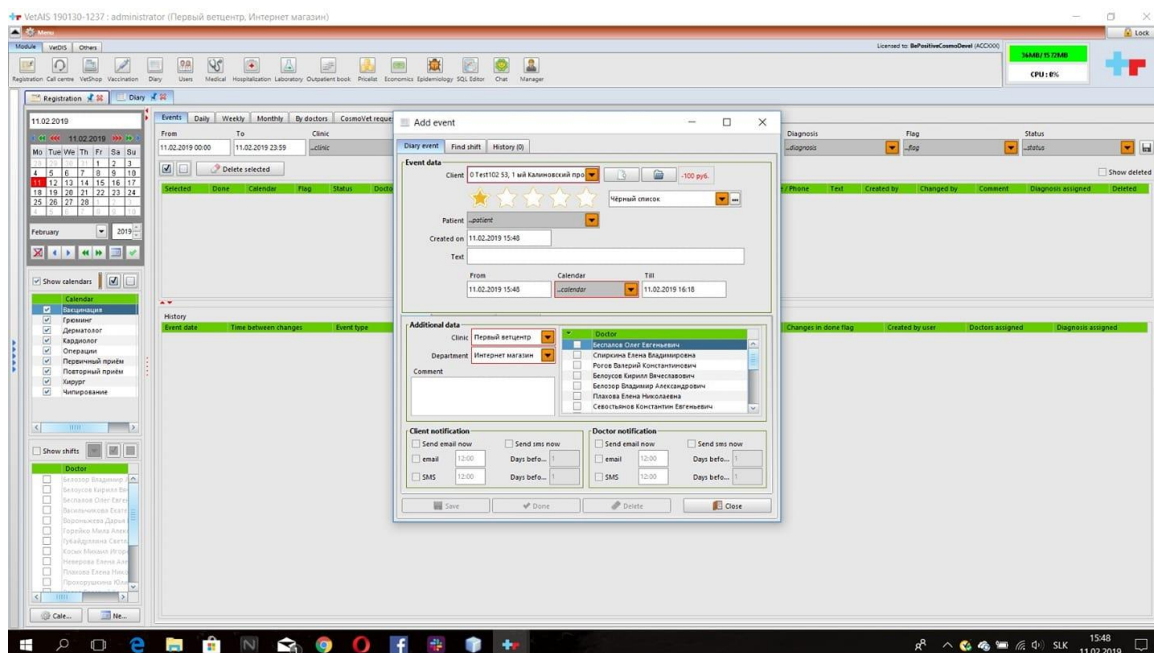


Рисунок 1 – скриншот работы программы

2 Конфигурация MedWork-Ветклиника

Данная программа является конфигурацией к автоматизированной информационной системе MedWork. Область деятельности ветеринарных клиник очень специфична, поэтому потребовалось проработать и заложить логику электронной медицинской документации, необходимой для работы специалистов клиники. Были разработаны документы, отражающие процесс лечения животного в ветеринарной клинике, такие как: форма первичного и повторного приема врача-ветеринара, дневник курации животного, позволяющий объективно следить за динамикой состояния животного, находящегося в стационаре, паспорта животного и журналы вакцинации, бланки трихографического исследования, гельминтоовоскопии, гистологического, микологического, цитологического, паразитологического исследований и д.р. Все эти документы связаны определенными программными алгоритмами, автоматизирующими и упрощающими процесс работы над каждым медицинским случаем. В системе так же созданы специфические отчеты и журналы, например журнал вакцинации, универсальный отчет по оказанным

Изм.	Лист	№ докум.	Подпись	Дата

МИВУ 09.03.04-10.000 ПЗ

Лист

9

услугам, отчет для сверки услуг, отчет по павшим животным, и т.д. Ведется интерактивный график загрузки клеточного фонда. Такая отчетность позволяет анализировать качество оказываемых услуг, ежедневно улучшая деятельность клиники. На рисунке 2 представлен скриншот работы программы.

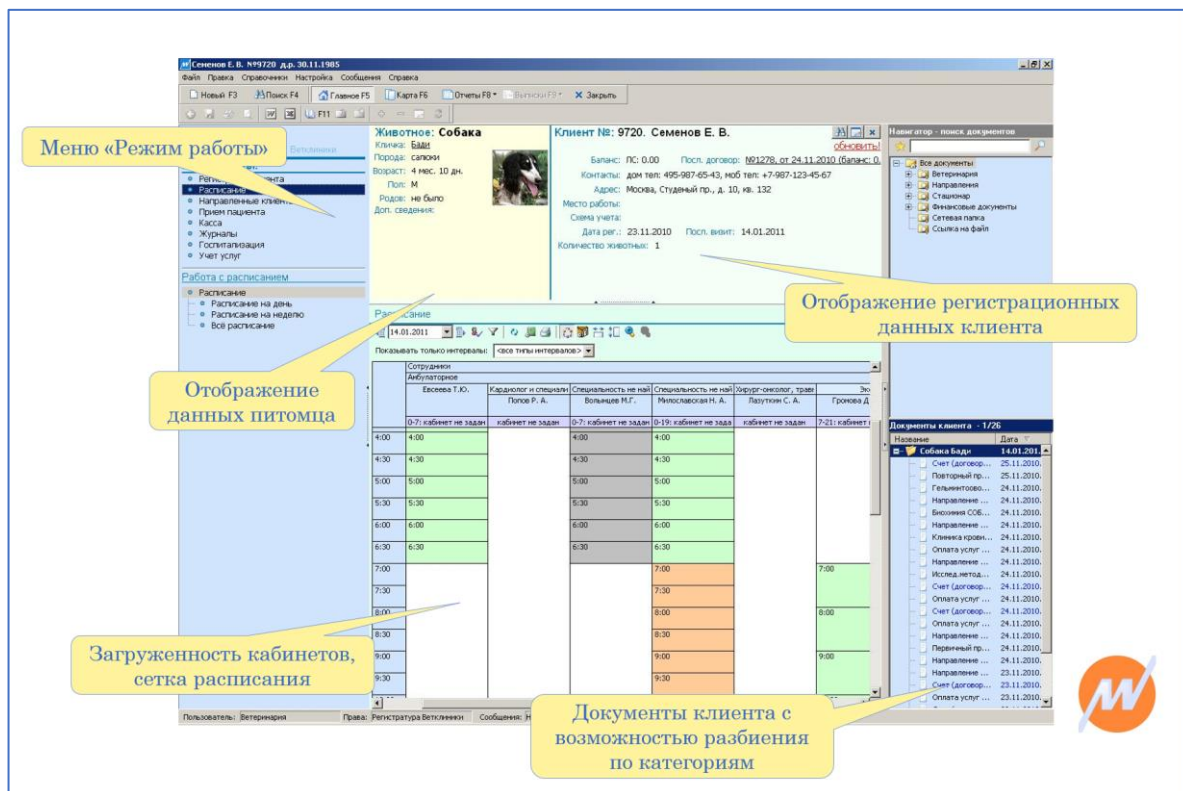


Рисунок 2 – скриншот работы программы

1.2 Обоснование выбора средств реализации

Для разработки базы данных необходимо выбрать СУБД. В настоящее время существует достаточно большое количество различных СУБД, позволяющих создавать и управлять разного рода базами данных.

Чтобы разработать БД Страхового агентства выбрана СУБД MySQL. Одной из самых популярных СУБД на сегодняшний день является MySQL, распространяемая свободно (с некоторыми ограничениями).

Изм.	Лист	№ докум.	Подпись	Дата

Помимо универсальности и распространенности СУБД MySQL обладает целым комплексом важных преимуществ перед другими системами. В частности следует отметить такие качества как:

- Простота в использовании. MySQL достаточно легко устанавливается, а наличие множества плагинов и вспомогательных приложений упрощает работу с базами данных.
- Обширный функционал. Система MySQL обладает практически всем необходимым инструментарием, который может понадобиться в реализации практически любого проекта.
- Безопасность. Система изначально создана таким образом, что множество встроенных функций безопасности в ней работают по умолчанию.
- Масштабируемость. Являясь весьма универсальной СУБД, MySQL в равной степени легко может быть использована для работы и с малыми, и с большими объемами данных.
- Скорость. Высокая производительность системы обеспечивается за счет упрощения некоторых используемых в ней стандартов.

В качестве среды для разработки прикладной программы для работы с созданной в SQL Server базой данных, была выбрана среда объектно-ориентированного программирования IntelliJ IDEA 2022, язык программирования Java. После индексирования исходного кода IntelliJ IDEA предоставляет массу возможностей для быстрой и эффективной разработки: умное автодополнение, анализ кода в реальном времени и надежные рефакторинги.

Визуальная часть приложения разработана на JavaFX. JavaFX - это платформа клиентских приложений нового поколения с открытым исходным кодом для настольных, мобильных и встраиваемых систем, построенных на Java. Это совместная работа многих частных лиц и компаний с целью создания современного, эффективного и полнофункционального инструментария для разработки расширенных клиентских приложений.

					МИВУ 09.03.04-10.000 ПЗ	Лист
						11
Изм.	Лист	№ докум.	Подпись	Дата		

Разрабатываемое программного средство должна обеспечивать получение из базы данных всей необходимой информации в полном объеме, а также возможность её редактирования и удаления. Также программа должна иметь визуальный интерфейс для работы с базой данных.

1.3 Функциональные требования

Программа должна содержать следующие функциональные возможности:

- загрузка и отображение изображений;
- добавление новой информации в БД;
- хранение информации;
- Предоставление информации на форме в табличном виде.

Согласно заданию, в программе нужно учесть следующие особенности:

- В базе данных должны содержаться информация о животных и их хозяевах.
- В базе данных должен содержаться список процедур, список работников.
- После посещения каждый врач вносит в карточку данные об осмотре, проведённом лечении.

1.4 Системные требования

- Процессор: не менее 2 ГГц или SoC
- ОЗУ: от 2 ГБ
- Место на жестком диске: от 500Мб
- Видеоадаптер: DirectX 9 или более поздняя версия с драйвером WDDM 1.0
- Экран: 1600 × 900 или более
- Мышь и клавиатура

					МИВУ 09.03.04-10.000 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		12

2. Разработка алгоритмов

Данный этап является самым важным при создании АИС. Здесь выделяются сущности, атрибуты сущностей и связи между сущностями. На основе полученной диаграммы “Сущность – связь” или логической модели строятся функциональные модели системы и диаграмма потоков данных. Для создания базы данных, нужно логическую модель представить в виде физической.

Но прежде, чем разрабатывать модели данных нужно выявить ограничения предметной области. В данной предметной области существуют следующие ограничения:

- Приём в больнице может осуществлять врач
- Во время приёма врачу может помогать другой врач или иной медицинский работник
- За один приём врач может осмотреть только одного питомца
- У питомца может быть только один хозяин
- У питомца может быть хотя бы одна фотография
- Во время приёма врач в свободной форме записывает поставленный диагноз и процедуры, которые клиент должен проводить самостоятельно
- Во время приёма могут быть проведены процедуры, оказываемые в ветеринарной клинике

2.1 Концептуальная модель

Концептуальная модель — это отражение предметной области, для которой разрабатывается база данных.

Объектами на разработанной модели являются Клиенты, животные, приёмы, процедуры и сотрудники. Сотрудник может оформить приём, в котором он указывает питомца, у которого есть хозяин, проведённые процедуры, диагноз и назначение.

					МИВУ 09.03.04-10.000 ПЗ	Лист
						13
Изм.	Лист	№ докум.	Подпись	Дата		

Далее представлена концептуальная модель (Рис. 1).

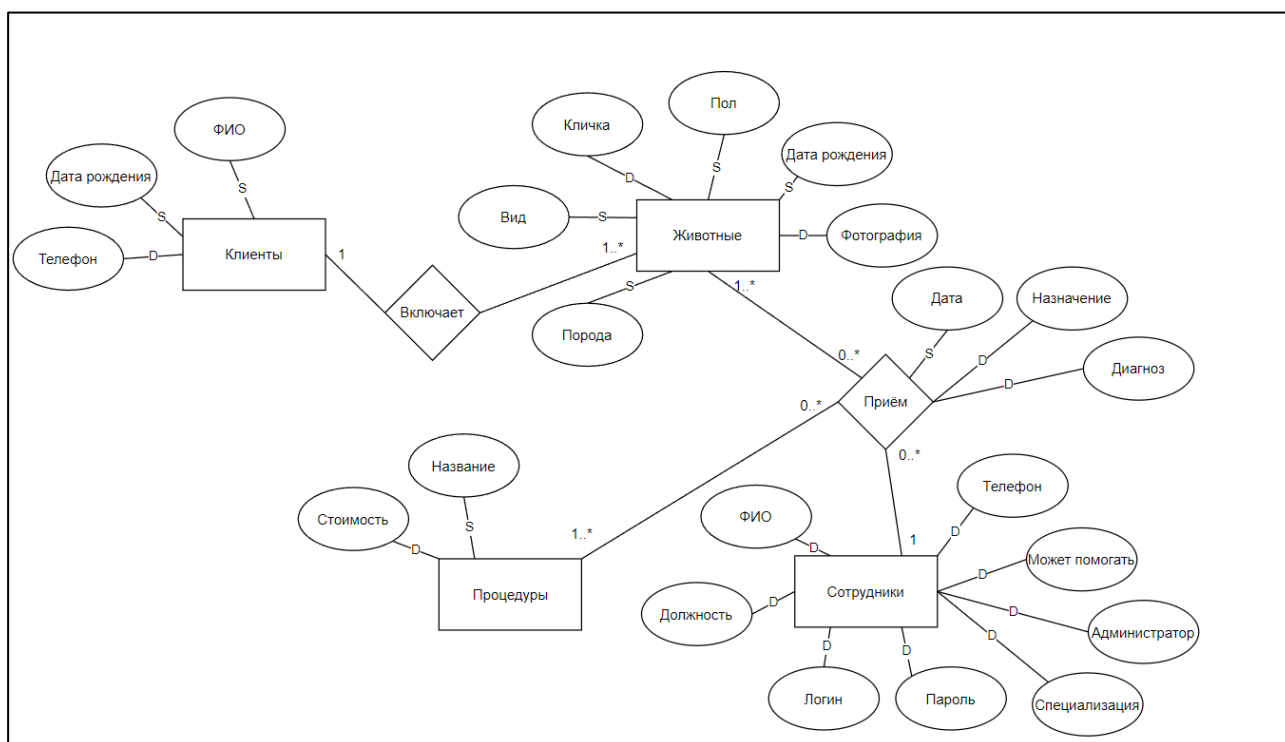


Рисунок 3 – концептуальная модель данных

2.2 Логическая модель данных

Целью построения логической модели является получение графического представления логической структуры исследуемой предметной области. Логическая модель предметной области иллюстрирует сущности, а также их взаимоотношения между собой.

Из задания понятно, что основной таблицей базы данных будет являться приём, в котором врачи будут записывать данные о животном, его хозяевах и назначенных процедурах. Соответственно необходимыми сущностями будут животные, сотрудники, как заполняющие сведения о приёме, так и обслуживающий персонал. У каждого животного есть хозяин, который привёл его на приём. Для этого проще создать отдельную сущность «Клиенты». Для

удобства работы с назначенными процедурами можно выделить их в отдельную таблицу, содержащую название процедуры и её стоимость.

Далее представлена логическая модель базы данных.

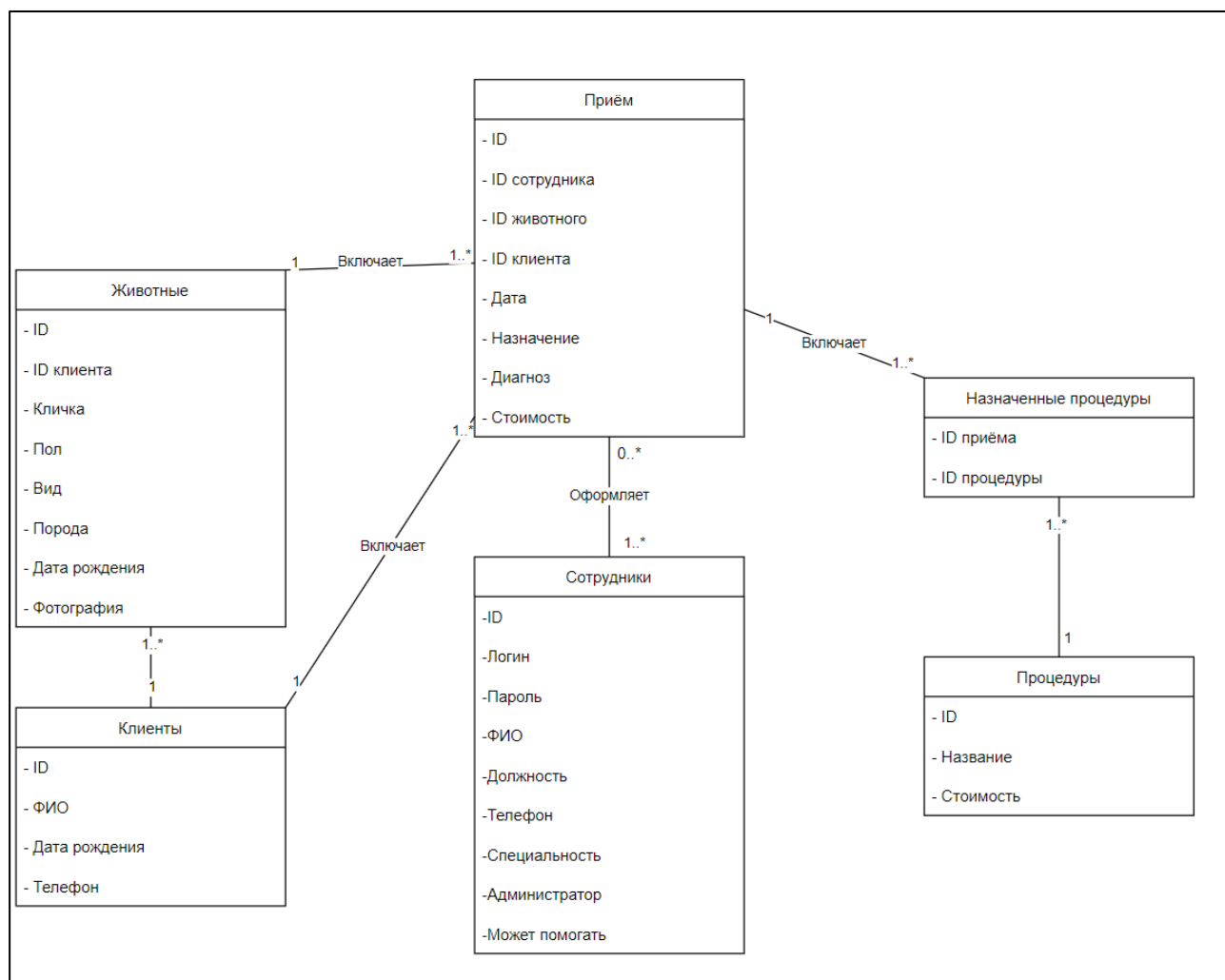


Рисунок 4 – логическая модель данных

После разработки моделей данных была создана сама база данных и соответствующие таблицы.

Так как основу программы составляет взаимодействия с базой данных, то рассмотрим его на примере некоторых методов из класса DB.

Кроме того, данные в программе между классами и методами передаются в форме моделей, соответствующих таблицам и их полям. Так, в программе существуют модели клиентов – класс Client, питомцев - Animal, приёмов - Visit, процедур Procedure, работников – Employee.

Снимки разработанного приложения приведены в приложении 2.

2.3 Авторизация

Данный метод обращается к классу базы данных, пытаясь получить сотрудника с логином соответствующим введённому. Далее идёт проверка на соответствие пароля найденного пользователя с введённым в форму. В случае если сотрудник не найден или пароли не совпадают выводится сообщение о неверных данных.

Листинг метода:

```
public void onEnter(ActionEvent actionEvent) throws SQLException {
    Employee employee = DB.getEmployee(tfLogin.getText());
    if (employee == null || employee.Password.equals(tfPassword.getText()) ==
false) {
        new Alert(Alert.AlertType.ERROR, "Пользователь с введёнными данными не
существует").show();
        return;
    }
    Parent root;
    try {
        FXMLLoader fxmllLoader = new FXMLLoader();
        fxmllLoader.setLocation(Main.class.getResource("hello-view.fxml"));
        Stage stage = new Stage();
        stage.setScene(new Scene(fxmllLoader.load(), 900, 600));
        HelloController helloController = fxmllLoader.getController();
        helloController.employee = employee;
        helloController.onCards(null);
        stage.show();
        ((Node) (actionEvent.getSource())).getScene().getWindow().hide();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

					МИВУ 09.03.04-10.000 ПЗ	Лист
						16
Изм.	Лист	№ докум.	Подпись	Дата		

2.4 Добавление данных в БД

В качестве примера добавления данных рассмотрим алгоритм приёма нового клиента.

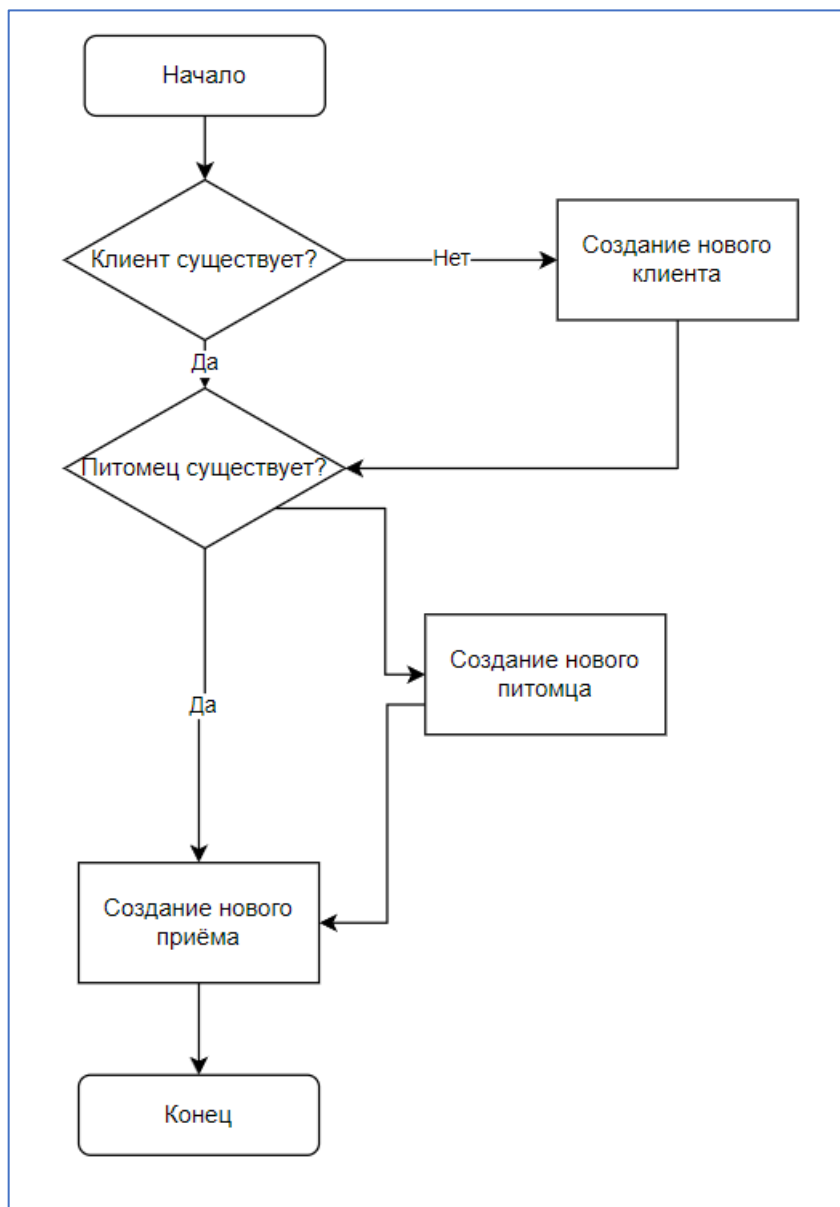


Рисунок 5 – Алгоритм записи приёма

Если в базе данных не существует клиента или у клиента не записано ни одного питомца, то требуется создание данных записей.

Само добавление записи происходит с помощью запроса, расположенного в специальном методе в классе базы данных. Ниже приведён пример добавление клиента

```
public static void addClient(Client client) {
    String sqlExpression = "Insert Into Clients (FullName, BirthDate, Phone) values(?, ?, ?)";
    PreparedStatement statement = null;
    try {
        statement = connection.prepareStatement(sqlExpression);
        statement.setString(1, client.FullName);
        statement.setString(2, String.valueOf(client.BirthDate));
        statement.setString(3, client.Phone);
        statement.execute();
    } catch (SQLException e) {
        throw new RuntimeException(e);
    }
}
```

Аналогичным образом происходит добавление питомцев, сотрудников, процедур.

2.5 Добавление записи о приёме

Поскольку таблица приёмов соединена связью «многие-ко многим» с таблицей процедур, для добавления записи необходимо выполнить несколько запросов:

```
public static void addVisit(Visit visit, List<Procedure> procedures) throws SQLException {
    String sqlExpression = "Insert Into Visits (IDAnimal, IDEmployee, IDClient, Date, Diagnosis, Assignment, IDHelperEmployee, TotalCost) values(?, ?, ?, ?, ?, ?, ?, ?)";
    PreparedStatement statement = null;
    connection.setAutoCommit(false);
    int ID = 0;
    try {
        statement = connection.prepareStatement(sqlExpression, Statement.RETURN_GENERATED_KEYS);
        statement.setInt(1, visit.IDAnimal);
        statement.setInt(2, visit.IDEmployee);
        statement.setInt(3, visit.IDClient);
        statement.setDate(4, (Date) visit.Date);
        statement.setString(5, visit.Diagnosis);
        statement.setString(6, visit.Assignment);
        statement.setInt(7, visit.IDHelperEmployee);
        statement.setInt(8, visit.TotalCost);
        statement.execute();
        ResultSet res = statement.getGeneratedKeys();
    }
```

					МИВУ 09.03.04-10.000 ПЗ	Лист
						18
Изм.	Лист	№ докум.	Подпись	Дата		

```

        if (res.next()) {
            ID = res.getInt(1);
        }

    } catch (Exception e) {
        throw new RuntimeException(e);
    }

    if (procedures.size() == 0) {
        return;
    }

    sqlExpression = "Insert Into PerformedProcedures (IDVisit, IDProcedure) values(?, (Select ID from ProceduresList
where Name = ?))";
    statement = null;
    try {
        for (Procedure procedure : procedures) {
            statement = connection.prepareStatement(sqlExpression);
            statement.setInt(1, ID);
            statement.setString(2, procedure.Name);
            statement.execute();
        }
    } catch (Exception e) {
        throw new RuntimeException(e);
    }
    connection.commit();
    connection.setAutoCommit(true);
}

```

2.6 Изменение записей

Для изменения записей в базе данных достаточно выполнить запрос Update, передающий необходимые данные. Ниже приведён пример изменения записи о клиенте

```

public static void changeClient(Client client) {
    String sqlExpression = "UPDATE Clients SET FullName = ?, BirthDate = ?, Phone = ? WHERE ID = ?";
    PreparedStatement statement = null;
    try {
        statement = connection.prepareStatement(sqlExpression);
        statement.setString(1, client.FullName);
    }
}

```

					МИВУ 09.03.04-10.000 ПЗ	Лист
						19
Изм.	Лист	№ докум.	Подпись	Дата		

```

statement.setString(2, String.valueOf(client.BirthDate));
statement.setString(3, client.Phone);
statement.execute();
} catch (SQLException e) {
    throw new RuntimeException(e);
}
}

```

2.7 Изменение записи о приёме

Из-за связи таблиц приёмов и процедур изменение таблицы приёмов требует особого, отличного от изменения других таблиц, подхода. Так, для изменения связи между таблицами сначала удаляются все процедуры, которые были в этой связи ранее и добавляется новый список процедур. Ниже приведён метод изменения записи о приёме:

```

public static void changeVisit(Visit visit, List<Procedure> procedures) throws SQLException {
    String sqlExpression = "Update Visits SET IDAnimal=?, IDEmployee=?, IDClient=?, Date=?, Diagnosis=?,
Assignment=?, IDHelperEmployee=?, TotalCost=? Where ID = ?";
    PreparedStatement statement = null;
    connection.setAutoCommit(false);
    try {
        statement = connection.prepareStatement(sqlExpression);
        statement.setInt(1, visit.IDAnimal);
        statement.setInt(2, visit.IDEmployee);
        statement.setInt(3, visit.IDClient);
        statement.setDate(4, (Date) visit.Date);
        statement.setString(5, visit.Diagnosis);
        statement.setString(6, visit.Assignment);
        statement.setInt(7, visit.IDHelperEmployee);
        statement.setInt(8, visit.TotalCost);
        statement.setInt(9, visit.ID);
        statement.execute();

        sqlExpression = "Delete FROM PerformedProcedures WHERE IDVisit = ?";
        statement = connection.prepareStatement(sqlExpression);
        statement.setInt(1, visit.ID);

        if (procedures.size() > 0) {
            sqlExpression = "Insert Into PerformedProcedures (IDVisit, IDProcedure) values(?, (Select ID from
ProceduresList where Name = ?))";
            try {

```

					МИВУ 09.03.04-10.000 ПЗ	Лист
						20
Изм.	Лист	№ докум.	Подпись	Дата		

```

        for (Procedure procedure : procedures) {
            statement = connection.prepareStatement(sqlExpression);
            statement.setInt(1, procedure.ID);
            statement.setString(2, procedure.Name);
            statement.execute();
        }
    } catch (Exception e) {
        throw new RuntimeException(e);
    }
}
} catch (Exception e) {

}

connection.commit();
connection.setAutoCommit(true);
}

```

Кроме того, так как операция затрагивает несколько таблиц и содержит в себе удаление данных, необходимо выполнять её как транзакцию, чтобы в случае ошибки на одном из этапов выполнения в базе данных не осталось повреждённых или неправильных данных.

2.8 Выборка данных из БД

В процессе работы АИС часто приходится обращаться к базе данных с целью получения той или иной сущности или списка сущностей, которые зачастую должны соответствовать определённым условиям. Для этой цели в базе данных присутствует выборка из всех таблиц как всего списка хранящихся значений, так и определённых записей, например выбора питомца по его идентификатору, работника по его логину и так далее. Ниже приведён пример получения клиента по ID:

```

public static Client getClient(int ID) throws SQLException {
    String sqlExpression = "SELECT * from Clients Where ID = ?";
    PreparedStatement statement = null;
    statement = connection.prepareStatement(sqlExpression);
    statement.setInt(1, ID);
    ResultSet result = statement.executeQuery();
}

```

					МИВУ 09.03.04-10.000 ПЗ	Лист
						21
Изм.	Лист	№ докум.	Подпись	Дата		

```

if (result.next()) {
    Client client = new Client(
        result.getString("FullName"),
        result.getString("Phone"),
        result.getDate("BirthDate")
    );
    client.ID = result.getInt("ID");
    return client;
}
return null;
}

```

Так же присутствует возможность получения всего списка клиентов:

```

public static List<Client> getClients() throws SQLException, ParseException {
    String sqlExpression = "SELECT * from Clients";
    List<Client> clients = new ArrayList<>();
    Statement statement = connection.createStatement();
    ResultSet result = statement.executeQuery(sqlExpression);
    while (result.next()) {
        Client client = new Client(
            result.getString("FullName"),
            result.getString("Phone"),
            new SimpleDateFormat("yyyy-MM-dd").parse(result.getString("BirthDate")));
        client.ID = result.getInt("ID");
        clients.add(client);
    }
    return clients;
}

```

2.9 Выборка отчётов

Одним из главных преимуществ АИС над другими средствами ведения дела является возможность быстрого получения отчётности. Ниже приведён метод, получающий из базы данных информацию о сотруднике по заданным параметрам – тип участия в приёмах, дата начала и окончания выборки.

```

public static ResultSet getEmployeeRep(int idEmployee, boolean main, boolean helper, java.util.Date minDate,
java.util.Date maxDate){
    String sqlExpression = "SELECT Visits.ID, Animals.Name as Питомец, Emp.Name as Врач, Visits.Date,
Clients.FullName, Visits.Diagnosis, Visits.Assignment, Helper.Name as Помощник, Visits.TotalCost as Стоимость " +
        "FROM Visits " +
        "JOIN Animals on Visits.IDAnimal = Animals.ID " +
        "JOIN Employees as Emp on Visits.IDEmployee = Emp.ID " +
        "JOIN Clients on Visits.IDClient = Clients.ID " +
        "LEFT JOIN Employees as Helper on Visits.IDHelperEmployee = Helper.ID " +
        "Where Visits.Date > " + minDate.toString() + " " +
        "AND Visits.Date < " + maxDate.toString() + " ";
    if (main && helper)

```

					МИВУ 09.03.04-10.000 ПЗ	Лист
						22
Изм.	Лист	№ докум.	Подпись	Дата		

```

{
    sqlExpression += " AND Visits.IDEmployee = '" + idEmployee + "' OR Visits.IDHelperEmployee = '" +
idEmployee + "'";
}
else if (main)
{
    sqlExpression += " AND Visits.IDEmployee = '" + idEmployee + "'";
}
else if (helper)
{
    sqlExpression += " AND Visits.IDHelperEmployee = '" + idEmployee + "'";
}
try {
    PreparedStatement statement = connection.prepareStatement(sqlExpression);
    return statement.executeQuery(sqlExpression);
} catch (Exception e) {
    throw new RuntimeException(e);
}
}

```

Данный метод возвращает параметр типа DataSet, который может быть записан например в таблицу на форме.

2.10 Хранение изображений

В базе данных хранятся только пути к изображениям, поэтому при добавлении питомцев фотография копируется по специальному пути, прописанному в конфигурационном файле:

```

if (photoPath != null) {
try {
    String path = new File(".").getCanonicalPath();
    newPath = "\\Photo\\" + name + date + ".png";
    Path source = Paths.get(photoPath);
    Path dst = Paths.get(path + newPath);
    Files.copy(source, dst, StandardCopyOption.REPLACE_EXISTING);

} catch (IOException e) {
    throw new RuntimeException(e);
}
}

```

И далее путь к фотографии записывается в БД

					МИВУ 09.03.04-10.000 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		23

2.11 Отображение списка карточек

На главной странице приложения находится список карточек, каждая из которых представляет собой отдельный компонент со своим контроллером.

Создание этого списка происходит следующим образом:

```
public void onCards(ActionEvent actionEvent) {
    try {
        loader = new FXMLLoader();
        loader.setLocation(Main.class.getResource("cards-view.fxml"));
        pane = loader.load();
        controller = loader.getController();
        controller.employee = employee;
        controller.initialize();
    } catch (IOException ignored) {

    }
    hbMainPanel.getChildren().clear();
    hbMainPanel.getChildren().add(pane);
}
```

hbMainPanel это панель на главной форме в которую добавляется список.

Список необходимо инициализировать:

```
public void initialize() {
    try {
        clients = DB.getClients();
        cbClients.setItems(FXCollections.observableArrayList(clients.stream().map(c -> c.FullName).toList()));
        setCards(null);
    } catch (Exception e) {
        new Alert(Alert.AlertType.ERROR, "Что-то пошло не так").show();
    }
}
```

Метод setCards выглядит следующим образом:

```
private void setCards(Client client) {
    try {
        vbCards.getChildren().clear();
        List<Animal> animals = DB.getAnimals();
        if (client != null){
            animals = animals.stream().filter(a->a.ClientID == client.ID).toList();
        }
        for (Animal animal : animals) {
            FXMLLoader loader = new FXMLLoader();
            loader.setLocation(Main.class.getResource("card-view.fxml"));
            Parent pane = loader.load();
            CardController cardController = loader.getController();
            cardController.setAnimal(animal);
            cardController.setController(this);
            cardController.setEmployee(employee);
            vbCards.getChildren().add(pane);
        }
    }
}
```

					МИВУ 09.03.04-10.000 ПЗ	Лист
						24
Изм.	Лист	№ докум.	Подпись	Дата		


```

    } catch (Exception e) {
        new Alert(Alert.AlertType.ERROR, "Что-то пошло не так").show();
    }
}

```

Для каждого питомца из списка создаётся новая карточка, в которую передаются необходимые параметры для автономной работы.

					МИВУ 09.03.04-10.000 ПЗ	Лист
						25
Изм.	Лист	№ докум.	Подпись	Дата		

3. Руководство программиста

В программе присутствует ряд моделей, отражающих соответствующие таблицы в базе данных – классы Animal (питомцы), Client (клиент), Employee (сотрудник), Procedure (процедура), Visit (приём).

Для расширения или модификации программы основной класс, информация о котором необходима, это класс связи с базой данных – DB.java. За настройки подключения в нём отвечают приватные поля – `connectionUrl`, `user`, `password`, `driver`. С помощью их изменения можно подключиться к любой существующей базе данных, однако предпочтительно использовать MySQL.

Перед использованием класса необходимо вызвать метод `start()`, который инициализирует соединение с БД. По умолчанию он вызывается в стартовом методе в классе Main. В классе находятся методы взаимодействия с каждой моделью, такие как добавление, изменение, поиск, выборка и получение отчёта (заполнение всех полей-ссылок). Отчёты можно получить через метод `public static ResultSet getTable(String name)`, передав туда название существующей таблицы.

Например для модели Client существуют методы `public static void addClient(Client client)`, `public static void changeClient(Client client)`, `public static Client getClient(int ID)`, `public static List<Client> getClients()` и `private static ResultSet getClientsRep()`. Методы для остальных моделей аналогичны.

За добавление и изменение данных в БД отвечают контроллеры, прикрепленные к соответствующим формам: за работу с питомцами отвечают `AddAnimalController` и `EditAnimalController`, за работу с клиентами `AddClientController` и `EditClientController`, за работу с сотрудниками `AddEmployeeController` и `EditEmployeeController`, за работу с процедурами `AddProcedureController` и `EditProcedureController`, за работу с приёмами `AddVisitController` и `EditVisitController`. Для изменения модели её сначала нужно передать в соответствующий контроллер при инициализации формы.

					МИВУ 09.03.04-10.000 ПЗ	Лист
						26
Изм.	Лист	№ докум.	Подпись	Дата		

При запуске приложения сначала появляется форма авторизации, за работу которой отвечает AuthorizationController.

После успешной авторизации появляется главное окно приложения, за работу которого отвечает MainController. На начальной странице расположен список карточек питомцев – за работу этого списка отвечает контроллер CardsController, а за работу конкретной карточки отвечает контроллер CardController.

Кроме того, в приложении существуют формы обозревателя БД, за работу которого отвечает класс DBViewController, для которого также используется перечисление Model, и форма с отчётами, за работу которой отвечает ReportController.

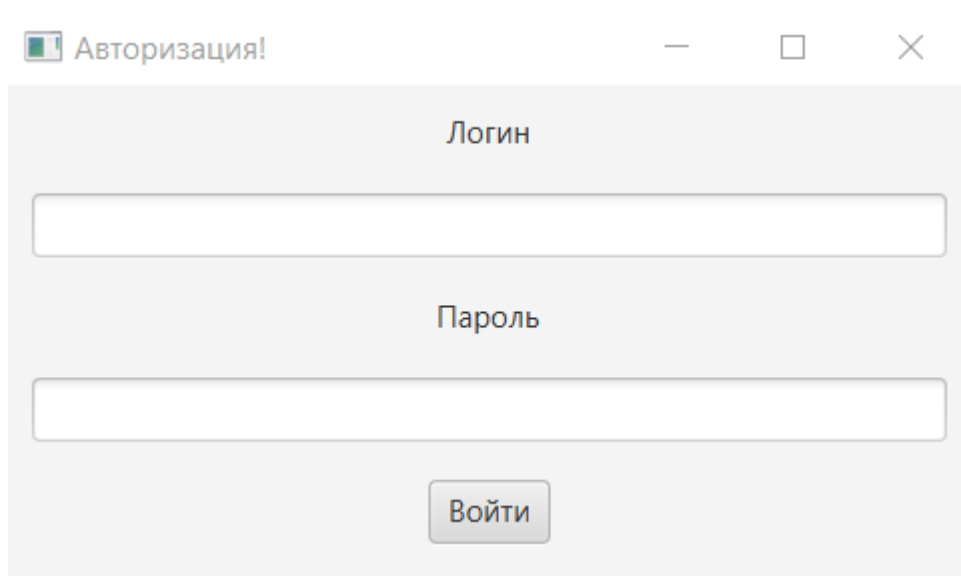
Fxml-разметка страниц расположена в файлах с названием, соответствующем контроллеру. Например, для контроллера DBViewController разметка хранится в файле dbView-view.fxml. Аналогично и для остальных контроллеров

Полную документацию по классам можно найти по ссылке, указанной в приложении 3

					МИВУ 09.03.04-10.000 ПЗ	Лист
						27
Изм.	Лист	№ докум.	Подпись	Дата		

4. Руководство пользователя

При запуске приложения пользователя встречает форма авторизации:



The image shows a screenshot of a software window titled "Авторизация!". The window has a standard Windows-style title bar with minimize, maximize, and close buttons. Inside the window, there is a light gray background. At the top, the word "Логин" is centered above a white rectangular input field. Below this, the word "Пароль" is centered above another white rectangular input field. At the bottom center, there is a button labeled "Войти".

Рисунок 6 – форма авторизации

После введения правильного логина и пароля откроется основная форма приложения:

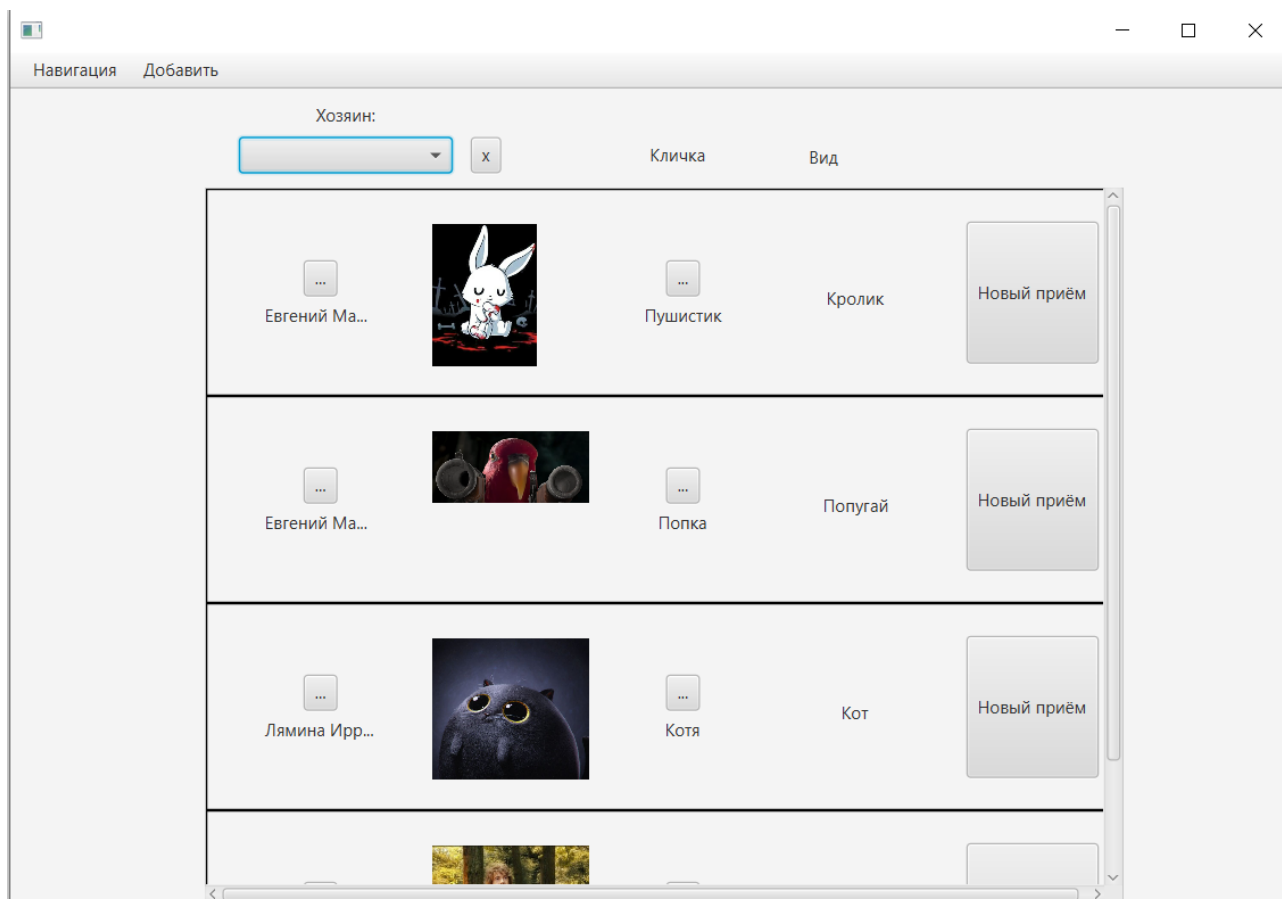


Рисунок 7 – основная форма приложения

Здесь можно добавить приём для питомца нажав на соответствующую кнопку, или открыть окно изменения питомца или клиента нажав на кнопку «...». Можно провести фильтрацию питомцев по хозяину.

По вкладке меню «Навигация» можно перемещаться по другим формам приложения: обозревателю БД и отчётам.

ID	Name	Name	Date	FullName	Diagnosis	Assignment	Name	TotalCost
5	Пушистик	Мартынов Евг...	2022-05-01 00:...	Евгений Март...	Смерть	ЫВ	Иванов Дмитр...	200
6	Воробей	Мартынов Евг...	2022-05-01 00:...	Евгений Март...	S	S	Иванов Дмитр...	200

Рисунок 9 – форма отчётов

Для формирования отчёта необходимо указать, будет ли проводиться отбор по сотрудникам (первая галочка). Если да, необходимо выбрать сотрудника и его роль в приёмах – был он главным, помощником или и тем и другим. Далее необходимо выбрать даты выборки и сформировать отчёт.

5. Тестирование

5.1 Авторизация

В случае если введён несуществующий логин или неверный пароль будет выведено сообщение об ошибке:

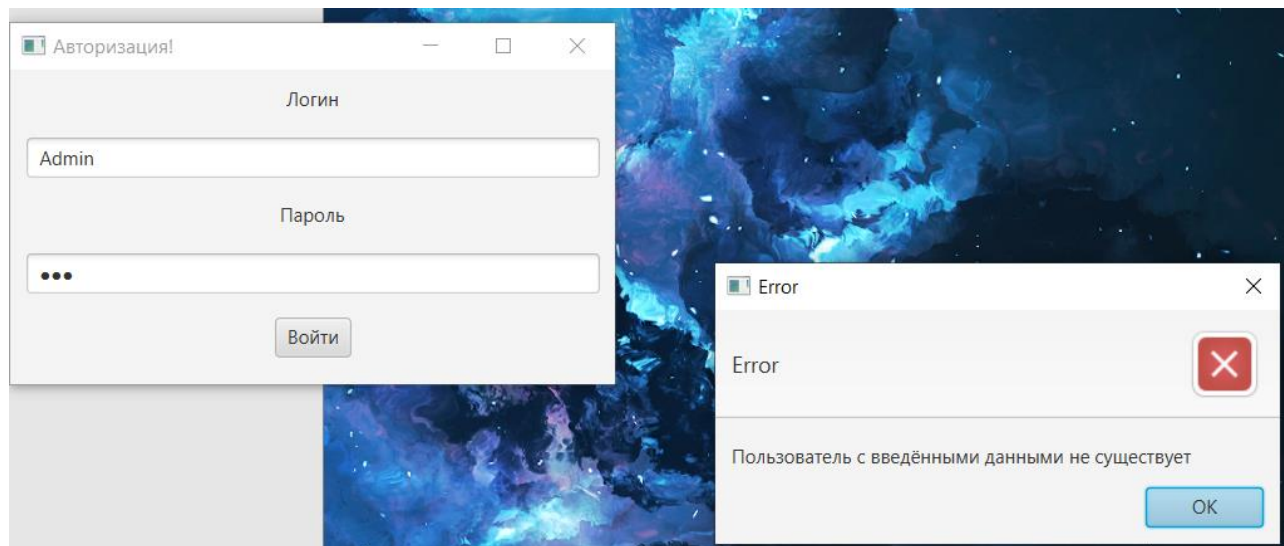


Рисунок 10 – Ошибка при авторизации

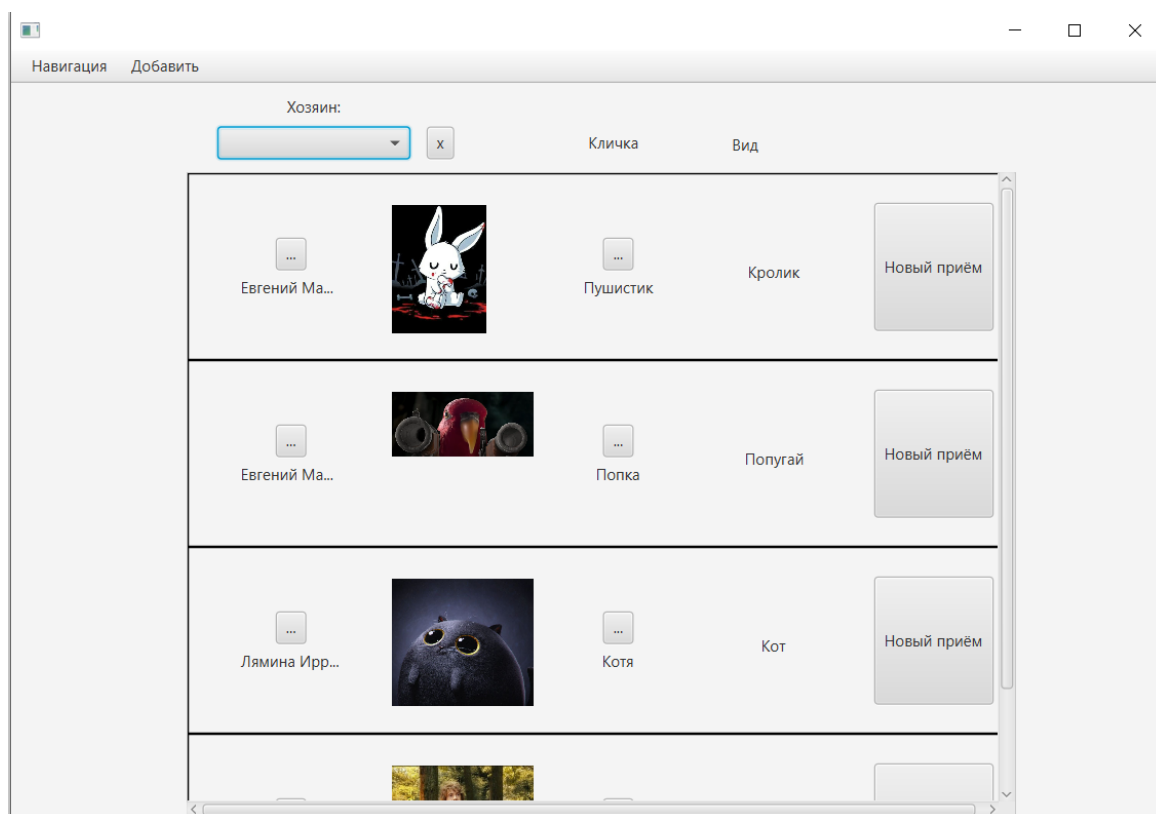


Рисунок 11 – успешное прохождение авторизации

Изм.	Лист	№ докум.	Подпись	Дата

5.2 Добавление и изменение сотрудников, питомцев, клиентов, процедур

Для сотрудников предусмотрены обязательные поля, которые должны быть у каждого работника. Это имя и должность. Если они не заполнены, программа не даст создать сотрудника и выдаст сообщение.

Полное имя*

Логин

Пароль

Телефон

Должность*

Специальность

Помощник ☒

Администратор ☐

Добавить Отмена

Error

Error

Заполните все необходимые поля

OK

Рисунок 12 – ошибка при создании сотрудника

При попытке создания сотрудника с логином, который уже существует будет выведена соответствующая ошибка

Полное имя*

Захаров Владислав Сергеевич

Логин

Admin

Пароль

•••••

Телефон

Должность*

Ветеринар

Специальность

Помощник ☐

Администратор ☐

Добавить Отмена

Error

Error

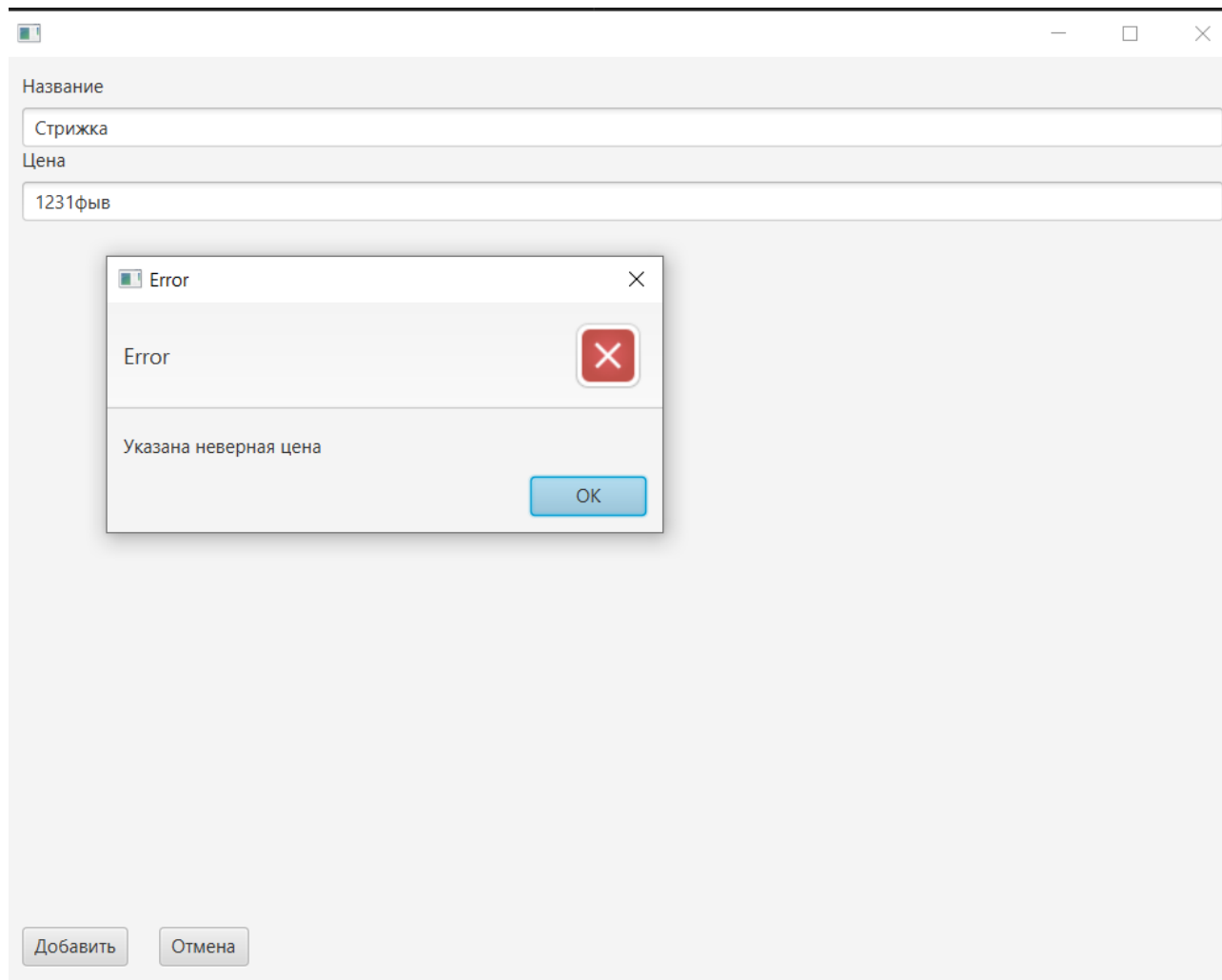
Данный логин уже занят

OK

Рисунок 13 – ошибка при создании сотрудника с уже существующим логином

При изменении существующего сотрудника проходит такая же проверка, остальные данные могут быть произвольной формы и контроль над их правильностью остаётся за пользователем.

Аналогичным образом выполнено добавление и изменение клиентов, животных, процедур. Ниже приведены скриншоты работы этих форм



Название

Стрижка

Цена

1231фыв

Error

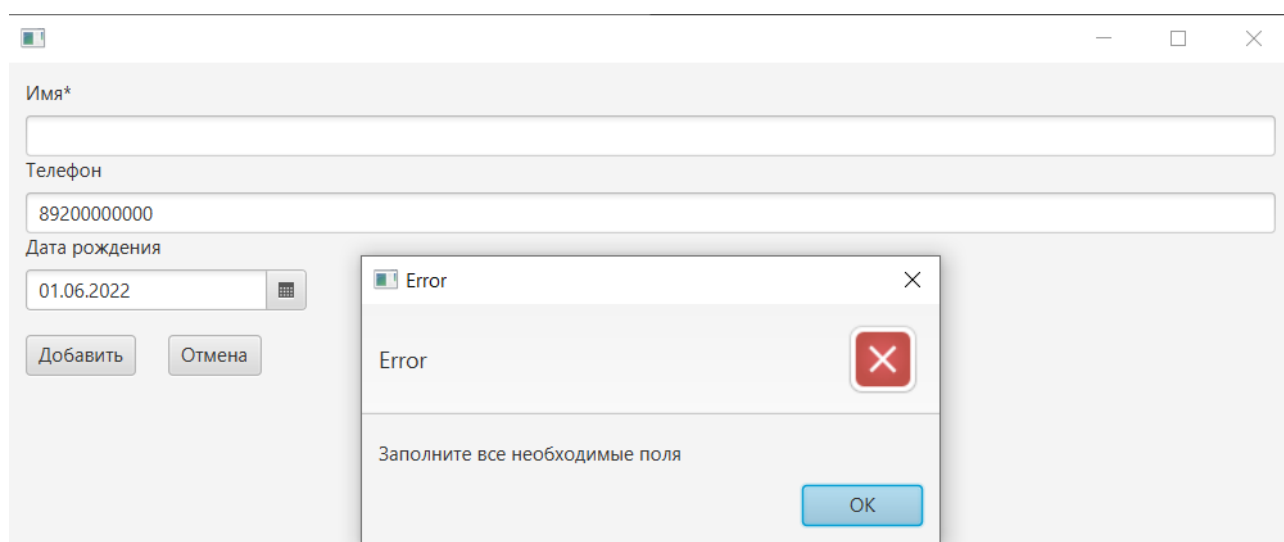
Error

Указана неверная цена

OK

Добавить Отмена

Рисунок 14 – форма добавления и изменения процедур



Имя*

Телефон

89200000000

Дата рождения

01.06.2022

Добавить Отмена

Error

Error

Заполните все необходимые поля

OK

Рисунок 15 – форма добавления и изменения клиентов

The screenshot shows a web form for adding a pet. The form includes fields for 'Хозяин*' (Owner), 'Кличка*' (Nickname) with the value 'Жека', 'Вид*' (Species) with the value 'Котик', 'Порода' (Breed), 'Дата рождения' (Date of birth) with the value '02.06.2022', and 'Пол' (Sex) with the value 'М'. There are buttons for 'Добавить' (Add), 'Отмена' (Cancel), 'Выбрать фотографию' (Select photo), and 'Очистить' (Clear). An 'Error' dialog box is overlaid on the form, displaying the message 'Заполните все необходимые поля' (Fill in all necessary fields) and an 'OK' button.

Рисунок 16 – форма добавления питомцев

5.4 Просмотр отчётов

The screenshot shows a report viewing interface. At the top, there are navigation buttons 'Навигация' and 'Добавить'. Below them are filters for 'Врач' (Doctor) and 'Помощник' (Assistant), a date range from '01.04.2022' to '01.06.2022', and a 'Сформировать' (Generate) button. The main area contains a table with the following data:

ID	Name	Name	Date	FullName	Diagnosis	Assignment	Name	TotalCost
1	Пушистик	Иванов Дмитр...	2022-05-31 00:...	Евгений Март...	Helper	Helper	Иванов Дмитр...	200
2	Пушистик	Иванов Дмитр...	2022-05-31 00:...	Евгений Март...	Helper	Helper	Иванов Дмитр...	200
4	Пушистик	Иванов Дмитр...	2022-05-31 00:...	Евгений Март...	Helper	Helper	Иванов Дмитр...	200
5	Пушистик	Мартынов Евг...	2022-05-01 00:...	Евгений Март...	Смерть	ЫВ	Иванов Дмитр...	200
6	Воробей	Мартынов Евг...	2022-05-01 00:...	Евгений Март...	S	S	Иванов Дмитр...	200

At the bottom right, it says 'Всего записей: 5' (Total records: 5).

Рисунок 18 – вкладка с отчётами

Первая означает будет ли проводиться выборка по сотрудникам, и если она установлена то необходимо выбрать сотрудника и хотя бы одну роль (Врач, помощник, или и то и другое).

The screenshot shows a software window with a table of medical records. The table has columns: ID, Name, Name, Date, FullName, Diagnosis, Assignment, Name, and TotalCost. The data rows are as follows:

ID	Name	Name	Date	FullName	Diagnosis	Assignment	Name	TotalCost
1	Пушистик	Иванов Дмитр...	2022-05-31 00:...	Евгений Март...	Helper	Helper	Иванов Дмитр...	200
2	Пушистик	Иванов Дмитр...	2022-05-31 00:...	Евгений Март...	Helper	Helper	Иванов Дмитр...	200
4	Пушистик	Иванов Дмитр...	2022-05-31 00:...	Евгений Март...	Helper	Helper	Иванов Дмитр...	200
5	Пушистик	Мартынов Евг...	2022-05-01 00:...	Евгений Март...	Смерть	ЫВ	Иванов Дмитр...	200
6	Воробей	Мартынов Евг...	2022-05-01 00:...	Евгений Март...	S	S	Иванов Дмитр...	200

An error dialog box is displayed in the center of the window. The dialog box has a title bar "Error" and a close button. The text inside the dialog box reads: "Пожалуйста, выберите тип участия (врач и/или помощник)". There is an "OK" button at the bottom right of the dialog box.

Рисунок 19 – попытка формирования отчёта с неверными параметрами

Заключение

В данной курсовой работе в соответствии с заданием была разработана АИС Ветеринарной клиники

В ходе выполнения курсовой работы были выполнены следующие задачи:

- выявлены требования к программе;
- разработаны модели данных;
- создана база данных;
- разработана программа;
- осуществлено ее тестирование.

Разработанная программа обеспечивает осуществление следующих функций:

1. Загрузка и отображение изображений;
2. Добавление новой информации в БД;
3. Хранение информации;
4. Предоставление информации на форме в табличном виде.
5. В базе данных содержится информация о животных и их хозяевах.
6. В базе данных содержится список процедур, список работников.
7. После посещения каждый врач вносит в карточку данные об осмотре, проведённом лечении.

Разработанная программа ускорила работу ветеринарной клиники, упростила ведение отчётности и работу врачей.

					МИВУ 09.03.04-10.000 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		38

Список используемой литературы

1. Блох, Дж. Java. Эффективное программирование / Дж. Блох ; перевод В. Стрельцов ; под редакцией Р. Усманов. — 2-е изд. — Саратов : Профобразование, 2019. — 310 с. — ISBN 978-5- 4488-0127-3. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: <http://www.iprbookshop.ru/89870.html>Руководство по MS SQL Server – [Электронный ресурс] // URL: <https://metanit.com/sql/sqlserver/> (Дата обращения – 17.11.2021)
2. Вязовик, Н. А. Программирование на Java : учебное пособие / Н. А. Вязовик. — 3-е изд. — Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2021. — 601 с. — ISBN 978-5-4497-0852-6. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: <http://www.iprbookshop.ru/102048.html>
3. Гуськова, О. И. Объектно ориентированное программирование в Java : учебное пособие / О. И. Гуськова. — Москва : Московский педагогический государственный университет, 2018. — 240 с. — ISBN 978-5-4263-0648-6. — Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт]. — URL: <https://www.iprbookshop.ru/97750.html>

Приложение 1 Скриншоты программы

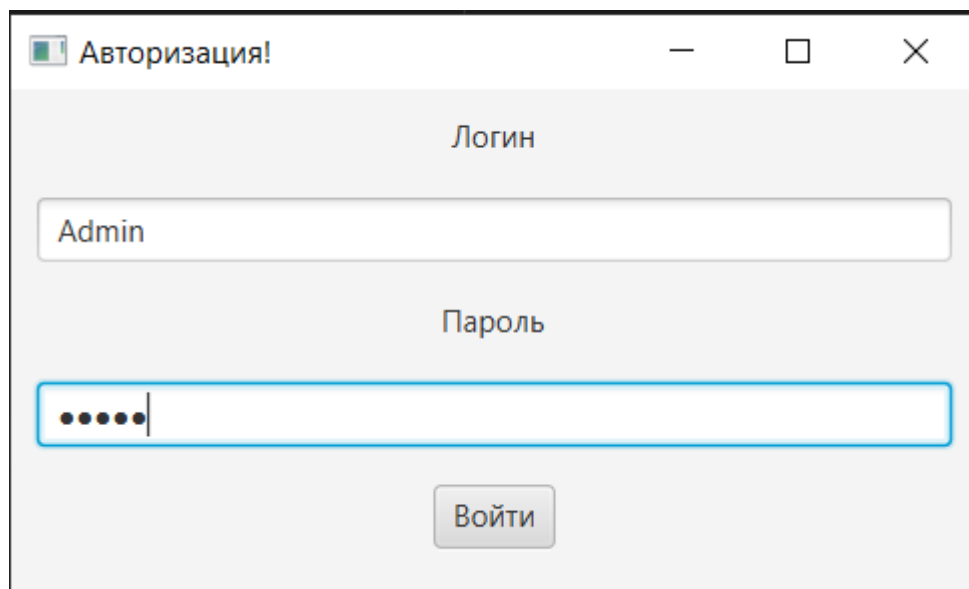


Рисунок 19 – окно авторизации

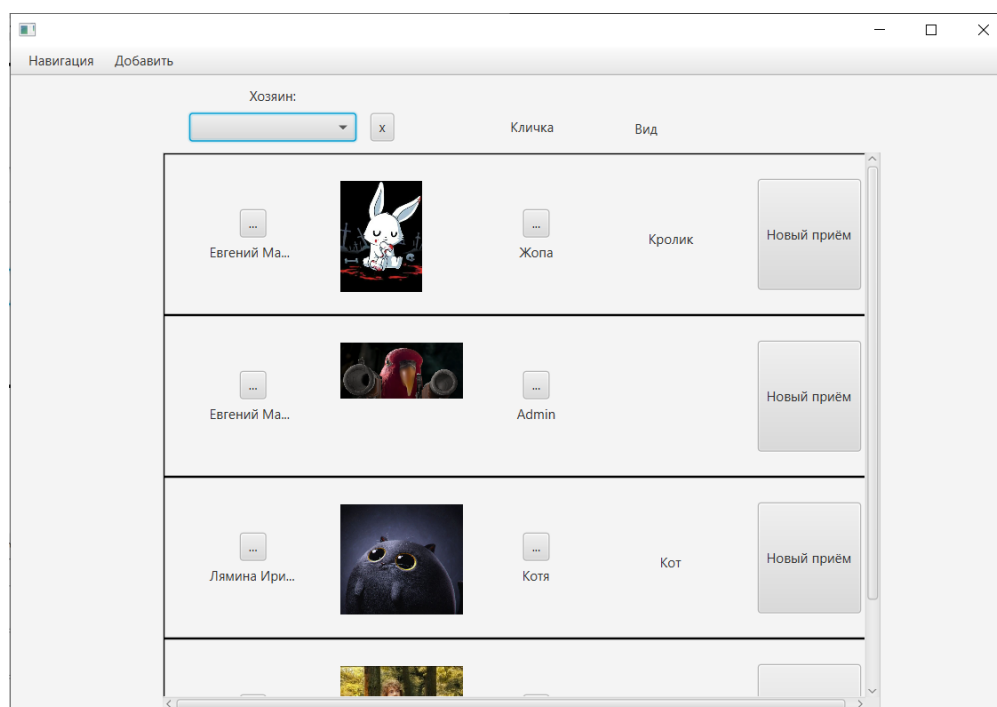



Рисунок 20 – главное окно приложения

					МИВУ 09.03.04-10.000 ПЗ	Лист
						40
Изм.	Лист	№ докум.	Подпись	Дата		



Дата

Время

Клиент

Лямина Ирина

Питомец

Котя

Помощник

Ответственный

Мартынов Евгений

Процедуры

Диагноз:

Назначение:

Проведённые процедуры:

История

Добавить

Отменить

Общая стоимость:

Рисунок 21 – Окно работы с приёмом

Изм.	Лист	№ докум.	Подпись	Дата

МИВУ 09.03.04-10.000 ПЗ

Лист
41

Название

Цена

Добавить Отмена

Рисунок 22 – окно работы с процедурой

Хозяин*

Кличка*

Вид*

Порода

Дата рождения

Пол

М

Добавить Отмена

Выбрать фотографию

Очистить

Рисунок 23 – окно работы с питомцами

Изм.	Лист	№ докум.	Подпись	Дата

Имя*

Телефон

Дата рождения

Добавить Отмена

Рисунок 24 – окно работы с клиентами

Полное имя*

Логин

Пароль

Телефон

Должность*

Специальность

Помощник ☐

Администратор ☐

Добавить Отмена

Рисунок 25 – окно работы с сотрудниками

[illegible]

Рисунок 26 – Обзорщик базы данных

[illegible]

Рисунок 27 – Отчёты

					МИВУ 09.03.04-10.000 ПЗ	Лист
						44
Изм.	Лист	№ докум.	Подпись	Дата		

							Лист
					МИВУ 09.03.04-10.000 ПЗ		45
Изм.	Лист	№ докум.	Подпись	Дата			

Приложение 2. Исходный код программы

Находится на сайте GitHub по ссылке:

<https://github.com/PrinzEugen212/RKP>

					МИВУ 09.03.04-10.000 ПЗ	Лист
						46
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение 3. Документация JavaDoc

<https://github.com/PrinzEugen212/RKP/tree/main/course/JavaDoc>

					МИВУ 09.03.04-10.000 ПЗ	Лист
						47
Изм.	Лист	№ докум.	Подпись	Дата		