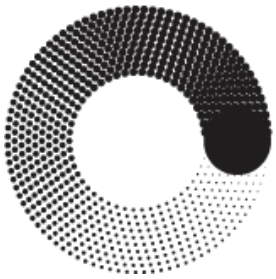
МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»



МОСКОВСКИЙ ПОЛИТЕХ

КУРСОВОЙ ПРОЕКТ

Учебная дисциплина *Основы инженерного проктирования*

Тема *Разработка десктопного приложения под Windows для контроля реабилитации пациентов*

Направление подготовки (специальность) *09.03.03 Прикладная информатика (профиль "Корпоративные информационные системы")*

Курс *1* Семестр *2* Группа *191-361*

Студент *Каменских С.С.* _____
Подпись

Оценка _____

Дата _____

Преподаватель *к.т.н. Логачёв М.С.* _____
Подпись

РОП *к.т.н. Логачёв М.С.* _____
Подпись

Москва
2020

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

СОГЛАСОВАНО

«*09*» *марта* 202*0* г.
Руководитель образовательной программы *Корпоративные информационные системы*

к.т.н. Логачёв М.С.

УТВЕРЖДАЮ

«*09*» *марта* 202*0* г.
Заведующий кафедрой *Инфокогнитивных технологий*

к.т.н. Филиппович А.Ю.

ЗАДАНИЕ

на курсовой проект

по дисциплине *Основы инженерного проектирования*

студенту *1-го курса* группы 191-361

Каменских Светлане Сергеевне

на тему: *Разработка десктопного приложения под Windows для контроля реабилитации пациентов*

Направление подготовки (специальность): *09.03.03 Прикладная информатика (профиль "Корпоративные информационные системы")*

Дата выдачи *12 марта 2020 г.* Срок окончания *30 июня 2020 г.*

ИСХОДНЫЕ ДАННЫЕ

СУБД: postgreSQL, язык программирования: java,

модели согласно структуре отчета, количество таблиц: 7

тестовые данные: от 70 записей, количество категорий пользователей: 3,

количество строк кода: более 400, наличие комментариев,

проблемная область: поликлиника

СТРУКТУРА ОТЧЁТА

Титульный лист	*1 лист*
Индивидуальное задание (*прим*. печатается с двух сторон одного листа)	*1 лист*
СОДЕРЖАНИЕ	*1 лист*
ВВЕДЕНИЕ (актуальность, цель, задачи, объект и предмет исследования, реферат)	*1–2 листа*
АНАЛИТИЧЕСКАЯ ЧАСТЬ	
1.1 Постановка проблемы	*1 лист*
1.2 Анализ программного обеспечения (*прим*. используемого в проблемной области, не менее 5)	*2–3 листа*
1.3 Формальные модели проблемной области (*прим*. с построением функциональной диаграммы процесса и ее декомпозиции)	*2–3 листа*
1.4 План реализации проекта (*прим*. с построением диаграммы Гантта)	*1 лист*
1.5 Выводы по аналитической части (*прим*. краткое описание полученных результатов в теоретической части)	*1 лист*
ПРОЕКТНАЯ ЧАСТЬ	
2.1 Модель данных (*прим*. с построением ER-диаграммы)	*2–3 листа*
2.2 Руководство пользователя	*9–14 листов*
2.3 Особенности функционирования приложения (*прим*. с построением диаграммы последовательности)	*2–3 листа*
2.4 Развитие проекта (*прим*. с построением диаграммы прецедентов)	*2–3 листа*
2.5 Выводы по проектной части (*прим*. краткое описание полученных результатов в проектной части)	*1 лист*
ЗАКЛЮЧЕНИЕ	*1 лист*
СПИСОК ЛИТЕРАТУРЫ (*прим*. не менее 15 источников, в т.ч. книжные непериодические издания, издание от 2014 года)	*2–3 листа*
ПРИЛОЖЕНИЕ (*прим*. структурированный код программы)	*10–15 листов*
Общий объем содержания отчета	*40–57 листов*
Уникальность отчета (в %)	*от 70*

Преподаватель _____ *к.т.н. Логачёв М.С.*

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Актуальность данной работы обусловлена тем, что в настоящее время мало внимания выделено для пациентов, которых только выписали из поликлиники. Ведь зачастую им требуется соблюдение дополнительных мер для полного выздоровления. В данной курсовой работе рассматривается процесс автоматизации системы поддержки пациентов вовремя и после болезни.

Внедрение программного продукта позволит отойти от выписывания рекомендаций на бумаге и рекомендаций на словах. Врачи смогут осуществлять наблюдение за показателями пациентов, что ускорит процесс реабилитации пациентов.

Объектом исследования являются государственные поликлиники Москвы.

Предметом исследования является программа для контроля реабилитации пациентов.

Целью курсовой работы является разработка приложения под платформу Windows, которое позволит врачам составлять индивидуальные программы питания, программы тренировок и следить за показателями, пациентам предоставляет программу питания, программу тренировок, а также возможность контролировать и держать все свои данные в одном месте.

В соответствии с поставленной целью в курсовой работе решаются следующие задачи:

1. Выявить проблему в предметной области и предложить способ её решения.
2. Проанализировать существующие программные обеспечения, предназначенные для решения выявленной проблемы.
3. Разработать приложение для решения выявленной проблемы.
4. Составить руководство пользователя по разработанному приложению.
5. Описать особенности функционирования приложения.
6. Сформировать план дальнейшего развития проекта.

Для оформления пояснительной записки использован ГОСТ 7.32-2017.

Работа изложена на 34 страницах и состоит из введения, двух разделов, заключения и списка литературы. В работе представлено 36 рисунков, 2 таблицы. Список литературы включает 15 источников. Работа содержит приложение, изложенное на 26 страницах.

АНАЛИТИЧЕСКАЯ ЧАСТЬ

1.1 Постановка проблемы

Исходя из актуальности данной работы, была выделена следующая проблема. Системы поддержки пациентов вовремя и после болезни является неудобной. Как правило, пациентам после болезни необходимо в течение некоторого времени соблюдать определённый режим, рекомендации, которые врачи не всегда оглашают или записывают на бумажках, которые легко потерять и нелегко прочитать. Для решения данной проблемы можно создать приложение, в котором каждому пациенту будут выписываться рекомендации, для врача это займет мало времени, а пациентам будет в разы легче получить информацию о рекомендациях.

1.2 Анализ программного обеспечения

Чтобы оценить недостатки и преимущества конкурентов на рынке в сфере медицины, необходимо произвести анализ программного обеспечения (далее ПО). Были отобраны следующие программные продукты:

1. «Реабилитация пациентов». Приложение для реабилитации пациентов кардиологического профиля. В качестве основных функций выделается: запись показателей, лечебная гимнастика, диета психология. По функциональным возможностям данное приложение практически не уступает разработанному в данной курсовой работе проекту. Отличительной чертой является в приложение наличие вкладки «Психология». В приложение не рассмотрены подробно показатели и нет возможности их удалить.

2. «Мои пациенты». Приложение, созданное специально для врачей. С помощью него можно курировать больных, записывать их данные и анализы. Функционал данного приложения гораздо меньше возможностей приложения, разрабатываемого в данной курсовой работе, также приложение направлено на аудиторию врачей. В нем присутствуют данные по пациенту, его показатели, диагноз.

3. «1С:Реабилитация». Приложение для реабилитации пациентов, перенесших инсульт. Данное приложение является сильным конкурентом, так

как дизайн приложения является достаточно оригинальным. Функционал шире, нежели возможности приложения, разрабатываемого в данной курсовой работе. Как минус можно отметить не интуитивность управления.

4. «Rehab My Patient – for Therapists». Приложение выдает необходимые упражнения для пациента, которые ему назначил терапевт. Из минусов можно отметить, то, что функционал направлен только на предоставление упражнений, которые представлены в удобной форме и подробно расписаны. Также данное приложение не русифицировано.

5. «USU». Приложение является полноценной системой управления медицинским учреждением. Данное приложение и разрабатываемое приложение имеют разные цели и направлены на разную категорию пользователей. Из этого приложения можно подчерпнуть организованность системы, четко структурированность данных, визуальное оформление.

Таблица 1.1 — Сравнение программных продуктов

Возможности	**«Реабилитация пациентов»**	**«Мои пациенты»**	**«1С:Реабилитация»**	**«Rehab My Patient – for Therapists»**	**«USU»**
Наличие мобильного приложения	Есть	Есть	Есть	Есть	Отсутствует
Целевая аудитория	Пациенты кардиологического профиля	Медицинские работники	Пациенты после инфаркта, медицинские работники	Пациенты, медицинские работники	Медицинские работники
Выбор языка	Отсутствует	Отсутствует	Отсутствует	Только английский	Есть
Цена	Бесплатно	Бесплатно	Бесплатно	Бесплатно	Бесплатно
Возможность составления персональной программы для пациентов	Отсутствует	Отсутствует	Есть	Есть	Есть
Возможность мониторинга показателей здоровья	Есть	Есть	Есть	Отсутствует	Есть

По итогу отбора программных продуктов была составлена таблица (Таблица 1.1).

Таким образом, исходя из сравнения программных продуктов по возможностям, можно сделать вывод о высокой конкурентоспособности приложения «1С:Реабилитация» на рынке, при разработке приложения были учтены его преимущества и недостатки. Так как у многих программных продуктов схожие возможности, но ни один продукт не совмещает все функции сразу [1].

1.3 Формальные модели проблемной области

В качестве формальной модели была построена функциональная модель для процесса «Назначить лечение пациенту» [10]. Для создания модели использовалась методология IDEF0. В составе IDEF0-модели была построена графическая диаграмма (Рисунок 1.1) с декомпозицией до первого уровня (Рисунок 1.2). Для построения диаграмм использовалось программное обеспечение Ramus.

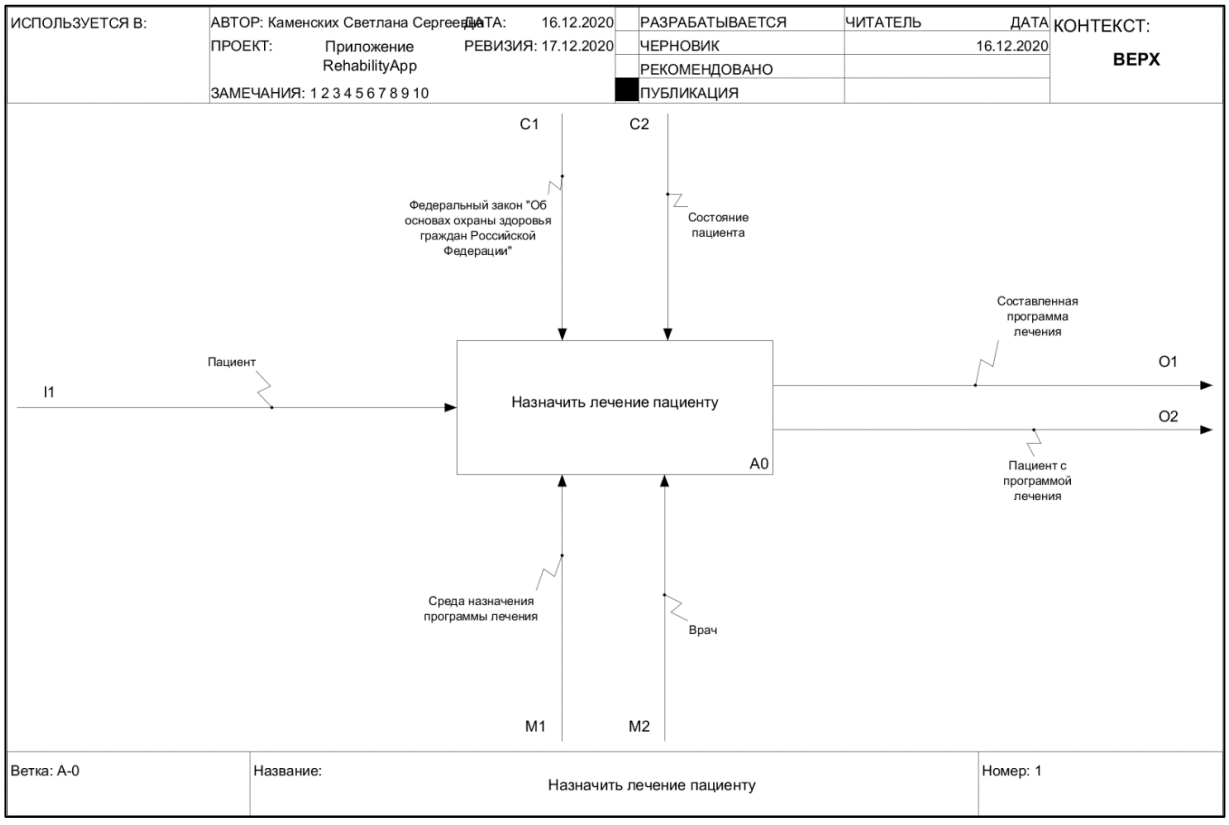


Рисунок 1.1 — Контекстная диаграмма верхнего уровня процесса «Назначить лечение пациенту»

Управлением являются: Федеральный закон "Об основах охраны здоровья граждан Российской Федерации", состояние пациента.

Механизмами осуществления процесса являются: врач, среда назначения программы лечения.

Входными данными является пациент. Результатом работы системы будут составленная программа лечения, пациент с программой лечения.

После описания контекстной диаграммы проводится функциональная декомпозиция – система разбивается на подсистемы, и каждая подсистема описывается отдельно [15]. Затем каждая подсистема, при необходимости, разбивается на более мелкие и так далее до достижения нужной степени подробности [14].

Контекстная диаграмма была разбита на 3 блока (Рисунок 1.2):

1. Зарегистрировать пациента;
2. Принять пациента;
3. Назначить программу лечения.

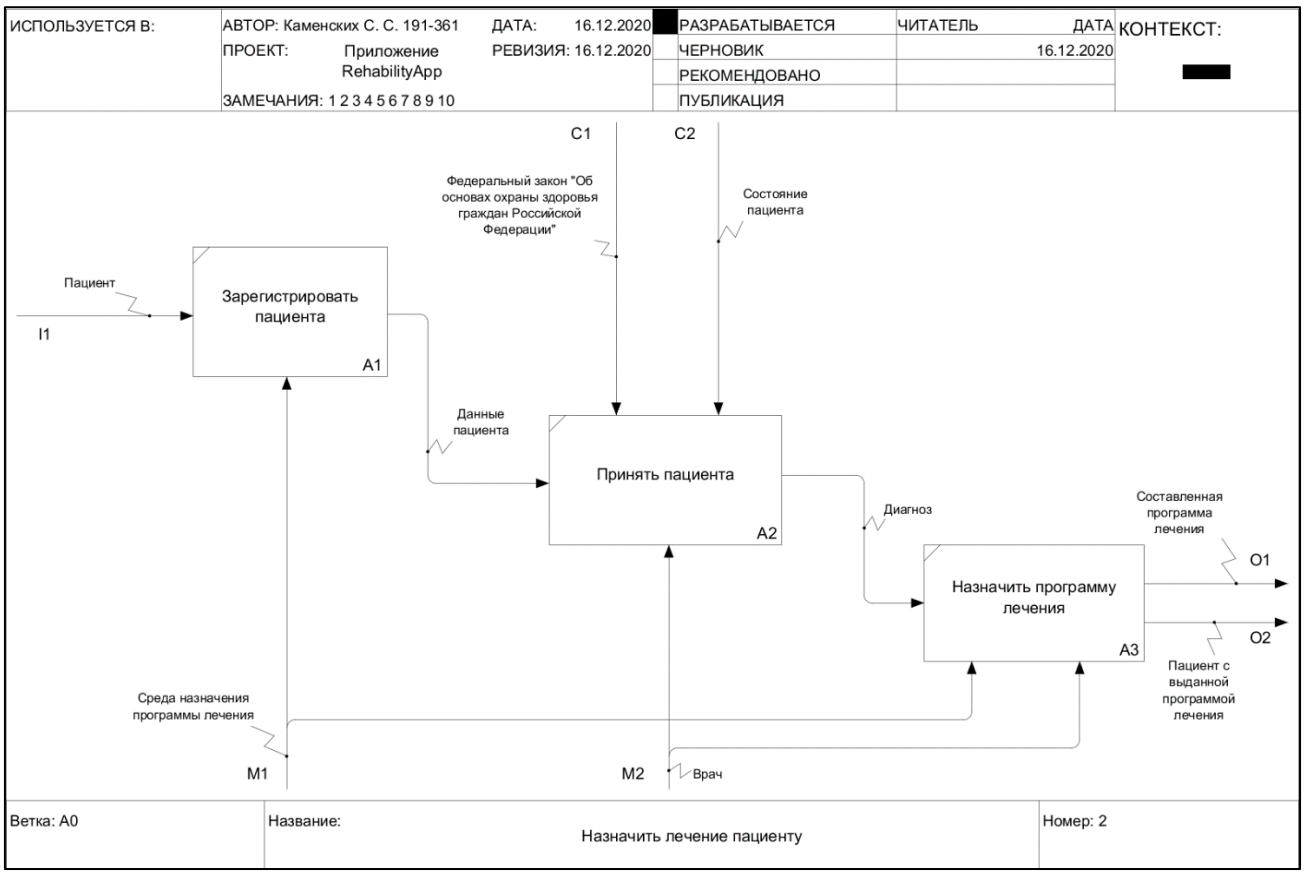


Рисунок 1.2 — Дочерняя диаграмма узла A-0

1.4 План реализации проекта

С помощью диаграммы Ганта необходимо разбить проект на задачи для наглядности полного объема работы.

Диаграмма Ганта представлена на Рисунке 1.3.

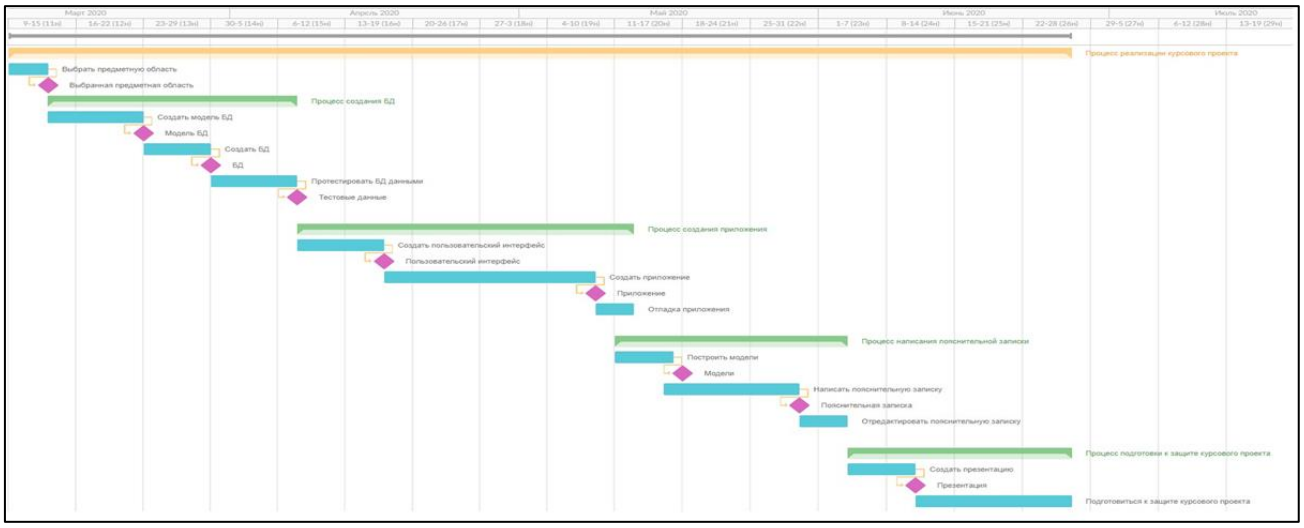


Рисунок 1.3 — Диаграмма Ганта процесса реализации курсового проекта

Данная диаграмма была построена при помощи онлайн-сервиса GanttPro.

1.5 Выводы по аналитической части

Таким образом, в аналитической части удалось решить следующие задачи: выявить проблемы, произвести анализ ПО, спроектировать формальные модели предметной области, составить план реализации проекта.

ПРОЕКТНАЯ ЧАСТЬ

2.1 Модель данных

Общим способом представления логической модели БД является построение ER–диаграмм. С помощью неё можно выделить ключевые сущности, также обозначить связи, которые могут устанавливаться между этими сущностями. В данной модели сущность определяется как дискретный объект, для которого сохраняются элементы данных, а связь описывает отношение между двумя объектами. Построенная логическая модель базы данных представлена на Рисунке 2.1 [6] [12].

В каждой таблице присутствуют ограничения, которые также показаны на ER-диаграмме (Рисунок 2.1).

Ограничения целостности данных за исключением «NOT NULL» и «(FK)» были вынесены в Таблицу 2.1 [7].

Тестовые данные представлены на рисунках Б.1–Б.7 в приложения Б.

В качестве нотации для построения ER-диаграммы была использована IDEF1X. Для построения диаграмм был использован программный продукт DBeaver.

Для создания базы данных была выбрана СУБД PostgreSQL [5] [11].

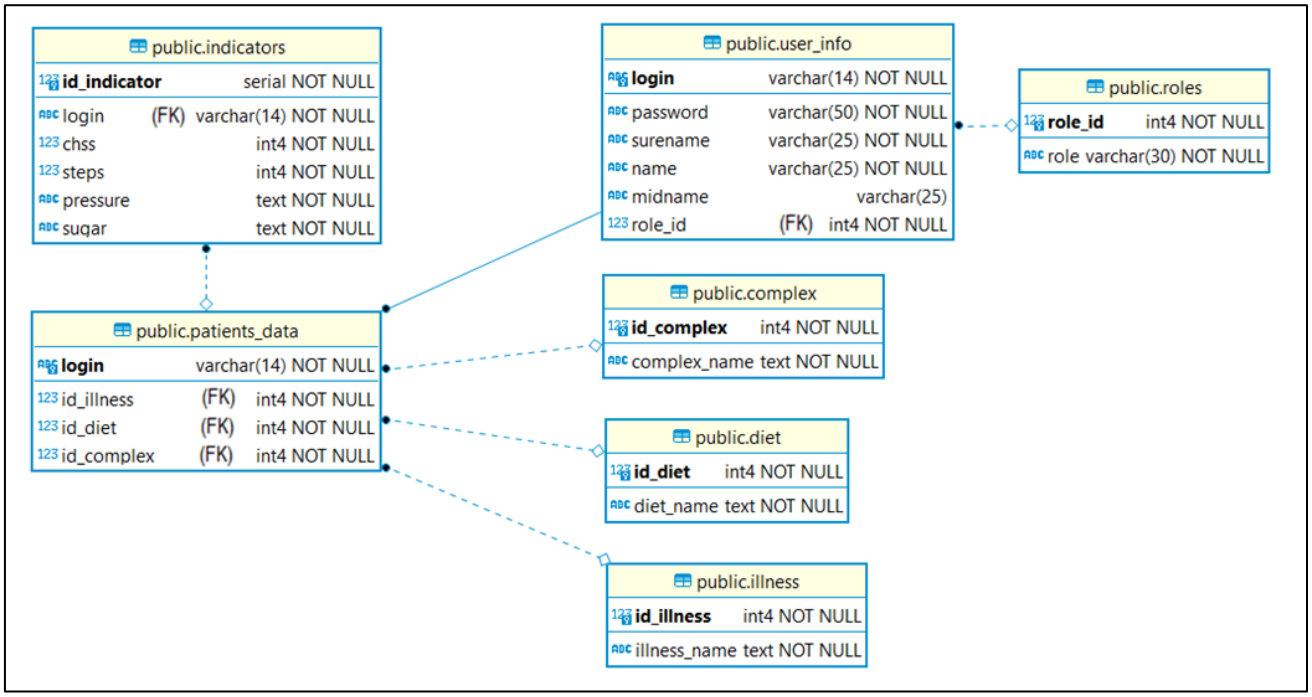


Рисунок 2.1 — ER-диаграмма схемы базы данных

Таблица 2.1 — Прочие ограничения целостности базы данных

Отношение	Ограничение
Indicator	CHECK(chss>0)
Indicator	CHECK(steps>=0)

2.2 Руководство пользователя

Введение

Далее приведено описание пользовательских операций для выполнения каждой из задач.

Область применения

Данное настольное приложение применяется для контроля показателей пациентов и формированию соответственных рекомендаций.

Уровень подготовки пользователя

Для работы с данным настольным приложением требуется минимальный опыт работы за компьютером. Оно не перегружено лишним функционалом, отвечает на каждую операцию во время работы с базой данных текстовыми сообщениями через пользовательский интерфейс.

Назначение и условия применения

Данное приложение предназначено для автоматизации процесса информирования клиентов об актуальной для них информации.

Приложение работает на стабильных версиях операционных систем Windows с версии 7 и выше. Проверка работоспособность на более ранних версиях не проводилась.

Требуется подключение к базе данных, структура которой описана в разделе на Рисунке 2.1.

Описание операций

Категория “Пациент”:

Регистрация нового пользователя

В открывшемся окне необходимо нажимать левой кнопкой мыши кнопку «Регистрация», расположенную внизу сцены по центру (Рисунок 2.2).

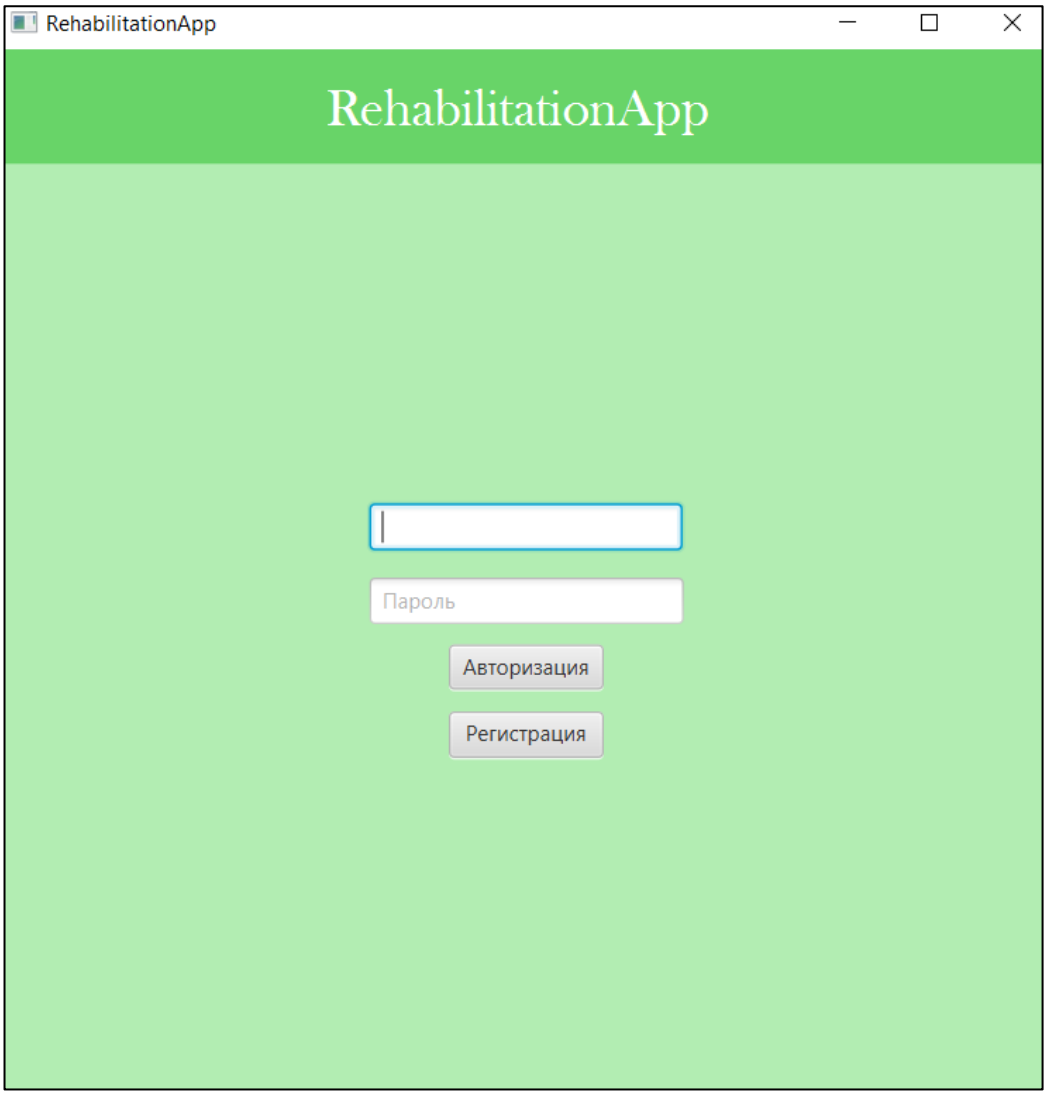


Рисунок 2.2 — Окно авторизации

В новом окне находится форма регистрации, которую необходимо заполнить корректными данными, соблюдая описанные требования. Необходимо заполнить данными следующие поля: фамилия, имя, отчество, полис, пароль, подтверждение пароля. Для этого необходимо нажать левой кнопкой мыши по полю и ввести корректные данные. Если отчество отсутствует, то его заполнять не надо. Пример заполненной формы на Рисунке 2.3.

Если все требования к вводимым данным соблюдены, выведется диалоговое окно с сообщением о том, что регистрация прошла успешно (Рисунок 2.4). Если поля заполнены некорректно, то программа выдаст диалоговое окно с указанием на некорректность введенных данных (Рисунок 2.5).

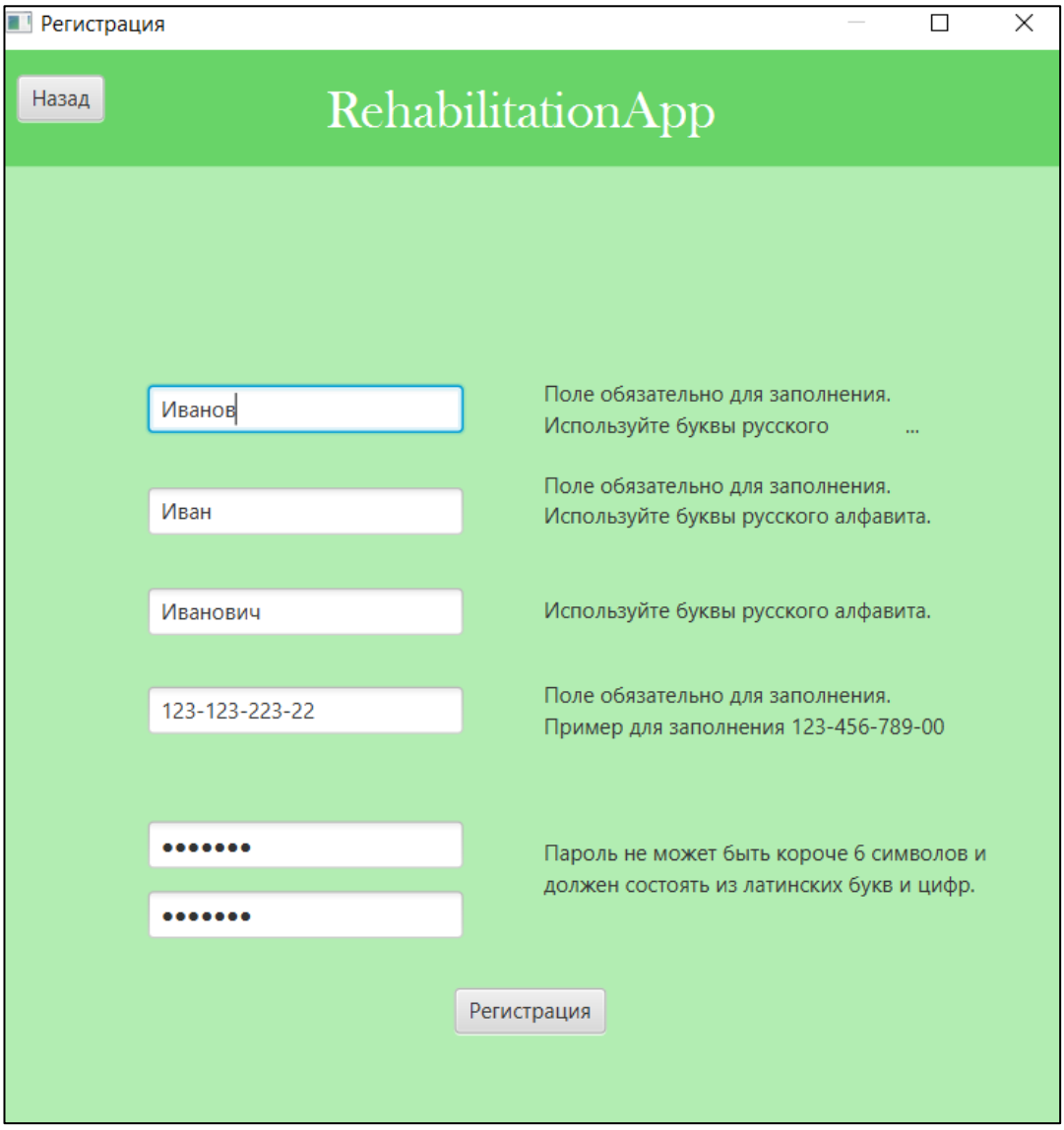


Рисунок 2.3 — Окно регистрации с заполненной формой

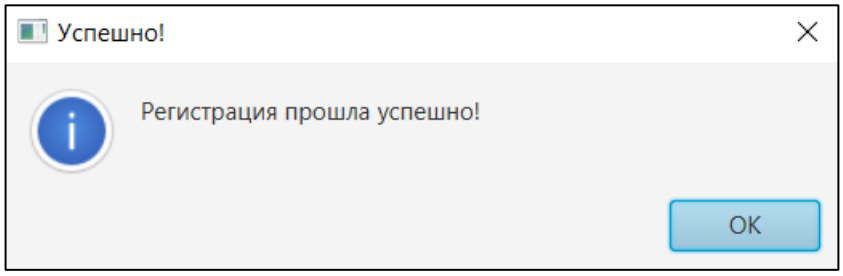


Рисунок 2.4 — Диалоговое окно с сообщением о пройденной регистрации

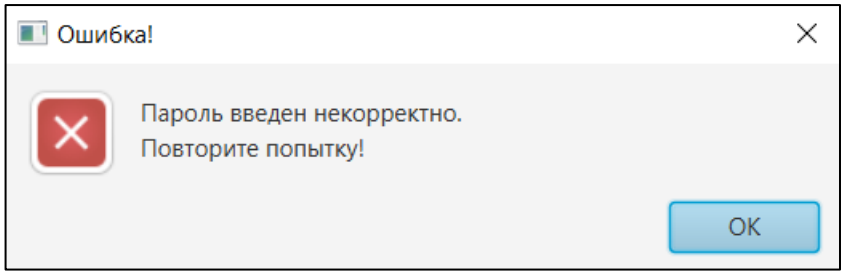


Рисунок 2.5 — Диалоговое окно с сообщением о некорректно введенных данных

Авторизация

В открывшемся окне необходимо заполнить поле «Логин или полис», где необходимо указать номер полиса для пациента (Рисунок 2.6) или логин для врача и администратора (Рисунок 2.7), и поле «Пароль». Для этого необходимо нажать левой кнопкой мыши по полю и ввести корректные данные. Далее нажать левой кнопкой мыши кнопку «Авторизация».

Если все поля заполнены корректными данными, то выведется диалоговое окно с сообщением о том, что авторизация прошла успешно (Рисунок 2.8). Если поля заполнены некорректно, то программа выдаст диалоговое окно с указанием на некорректность введенных данных (Рисунок 2.9).

После успешной авторизации откроется окно «Меню» пациента (Рисунок 2.10), «Меню» врача (Рисунок 2.11) или «Меню» администратора (Рисунок 2.12) в соответствие с ролью пользователя.

Рисунок 2.6 — Окно авторизации с заполненной формой для пациента

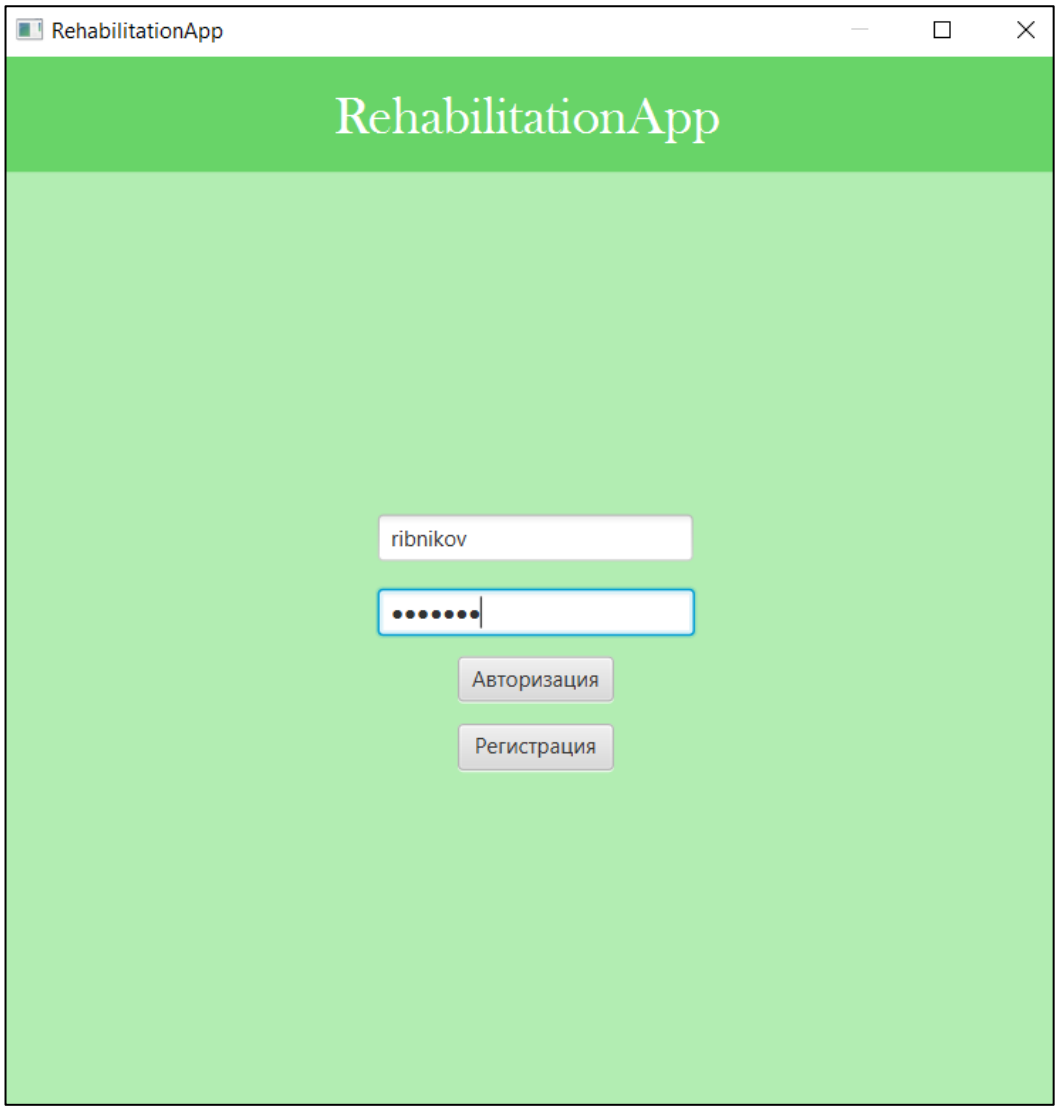


Рисунок 2.7 — Окно авторизации с заполненной формой для врача или администратора

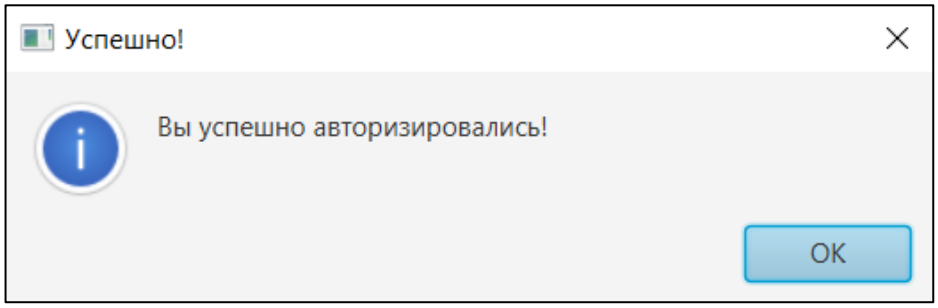


Рисунок 2.8 — Диалоговое окно с сообщением об успешной авторизации

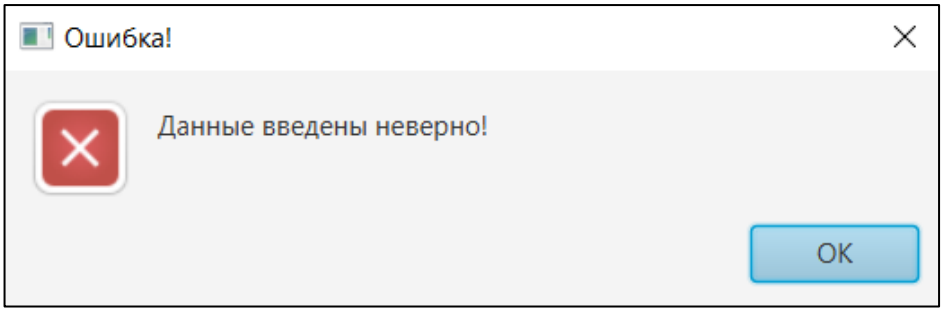


Рисунок 2.9 — Диалоговое окно с сообщением о некорректно введенных данных

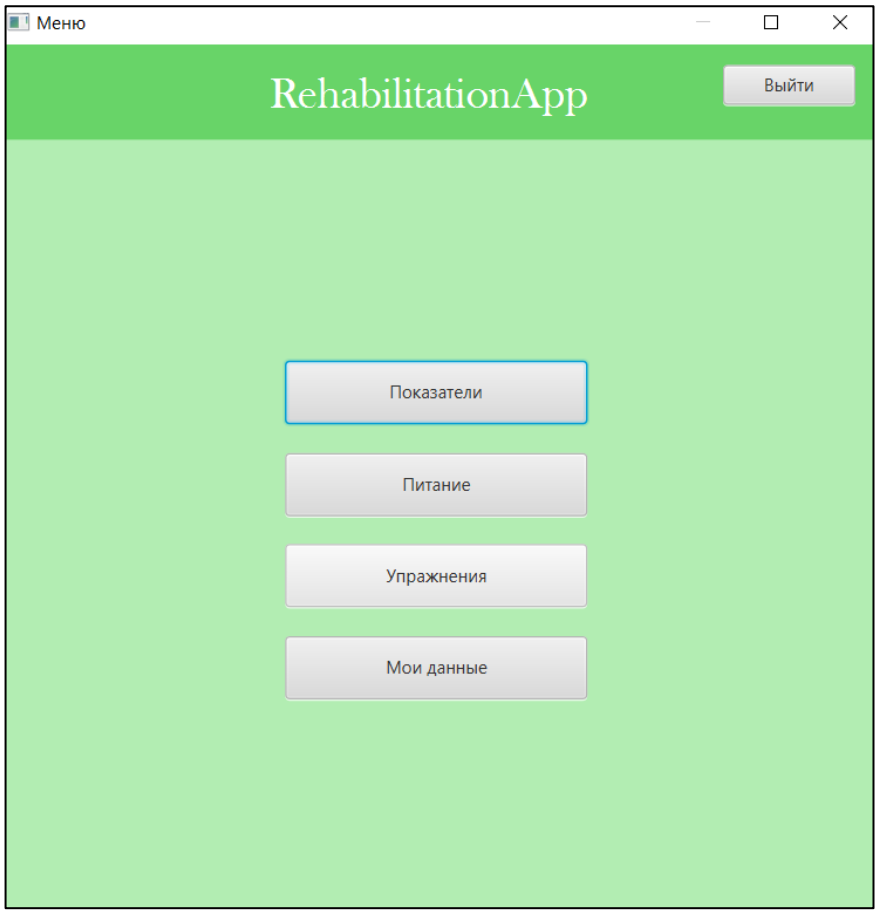


Рисунок 2.10 — Окно «Меню» для пациента

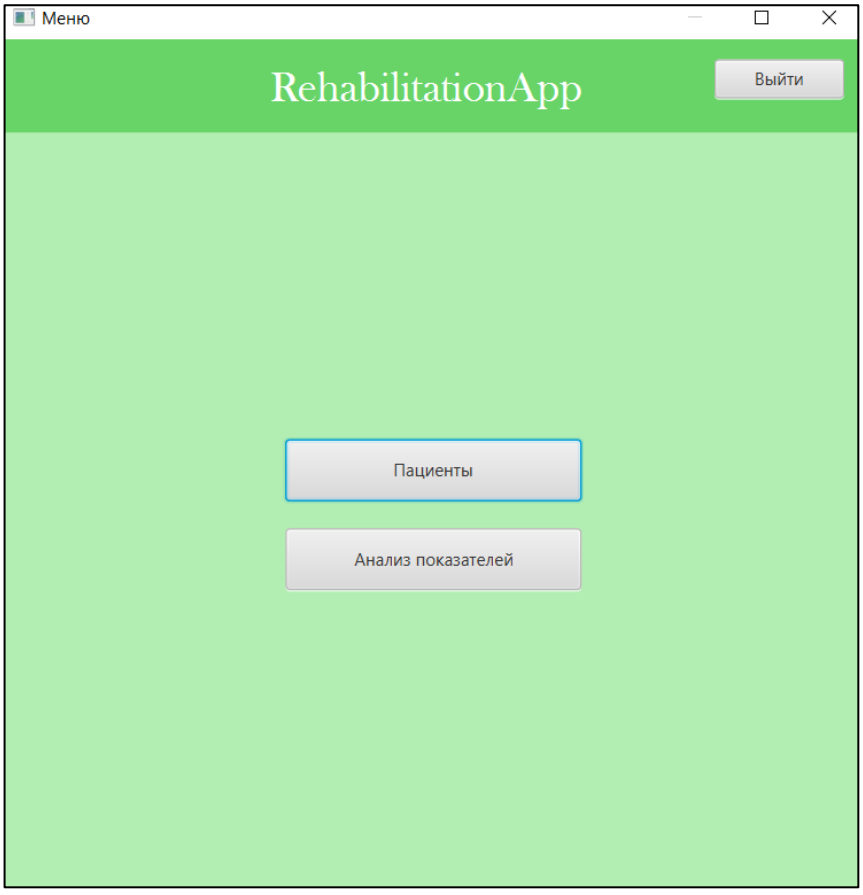


Рисунок 2.11 — Окно «Меню» для врача

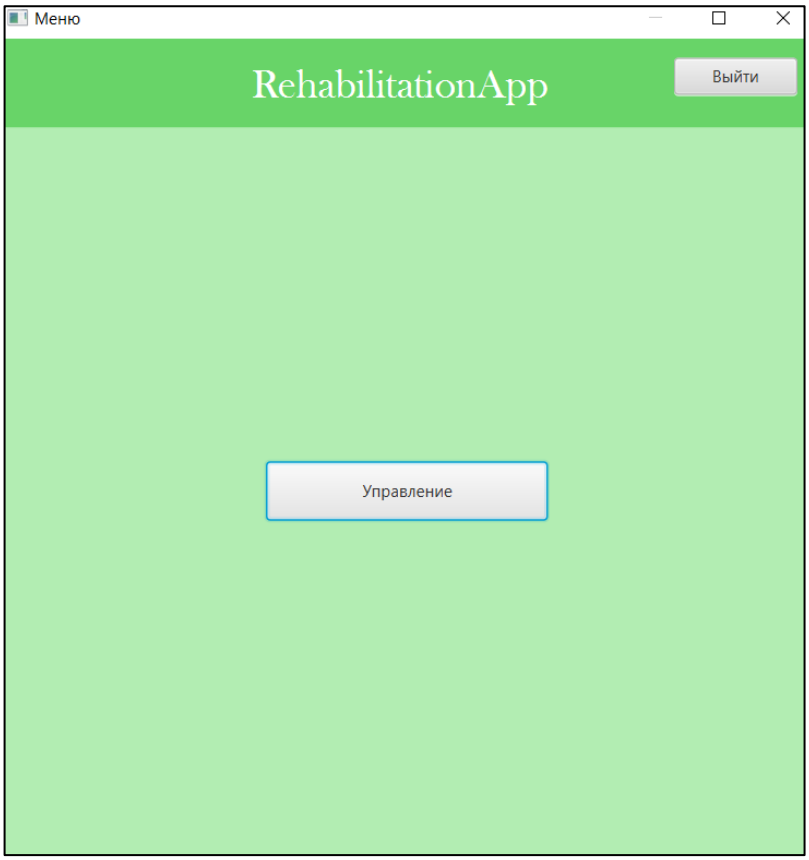


Рисунок 2.12 — Окно «Меню» для администратора

Внесение показателей здоровья

Необходимо пройти процесс авторизации, который описан выше. В открывшемся окне «Меню» нажимаем левой кнопкой мыши кнопку «Показатели».

В новом окне находится форма для внесения показателей и таблица всех внесенных пользователем показателей. Форму необходимо заполнить, соблюдая описанные требования. Пример заполненной формы на рисунке 2.14.

Далее необходимо нажать кнопку «Добавить». Выведется сообщение об успешном добавлении данных (Рисунок 2.13). Внесенные данные появятся в таблице справа (Рисунок 2.15).

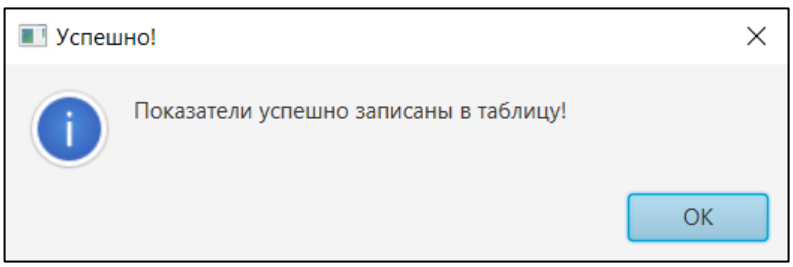


Рисунок 2.13 — Диалоговое окно с сообщением об успешном добавлении данных

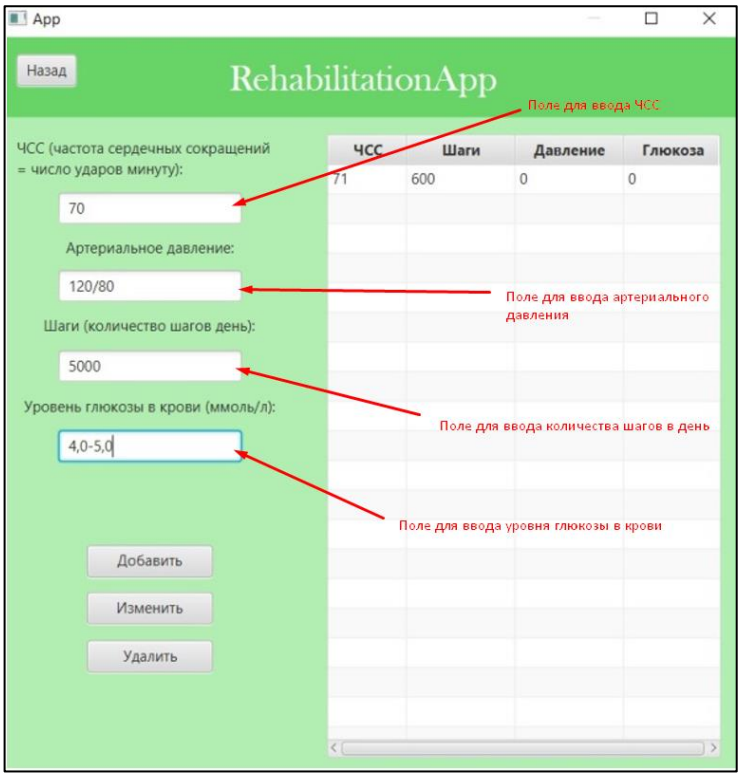


Рисунок 2.14 — Окно «Показатели» с заполненной формой

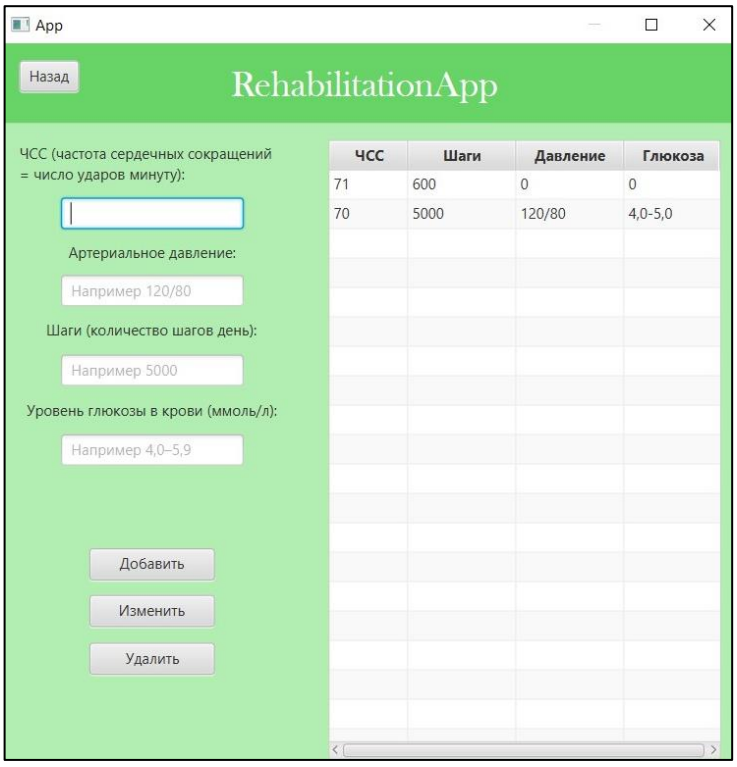


Рисунок 2.15 — Окно «Показатели» после добавления данных

Просмотр комплекса упражнений

Необходимо пройти процесс авторизации, который описан в задачах выше. В открывшемся окне «Меню» необходимо нажать левой кнопкой мыши кнопку «Упражнения».

Появится новое окно со всеми рекомендациями по соблюдению необходимых мер (Рисунок 2.16). В противном случае появится диалоговое окно с сообщением о том, что Ваш лечащий врач не назначил вам комплекс упражнений (Рисунок 2.17).

Рисунок 2.16 — Окно «Упражнения»

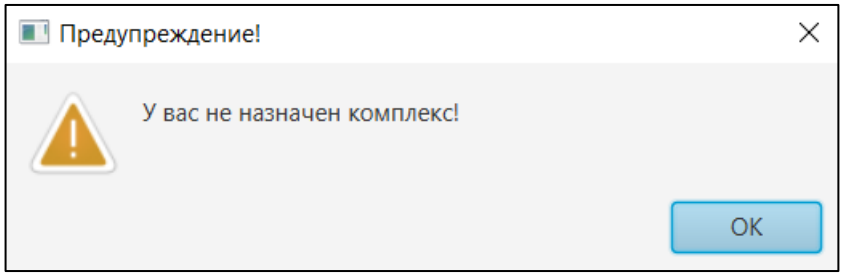


Рисунок 2.17 — Диалоговое окно с сообщением

Просмотр назначенной диеты

Необходимо пройти процесс авторизации, который описан в задачах выше. В открывшемся окне «Меню» необходимо нажать левой кнопкой мыши кнопку «Питание».

Появится новое окно со всеми рекомендациями по соблюдению питания (Рисунок 2.18). В противном случае появится диалоговое окно с сообщением о том, что Ваш лечащий врач не назначил вам программы питания (Рисунок 2.19).

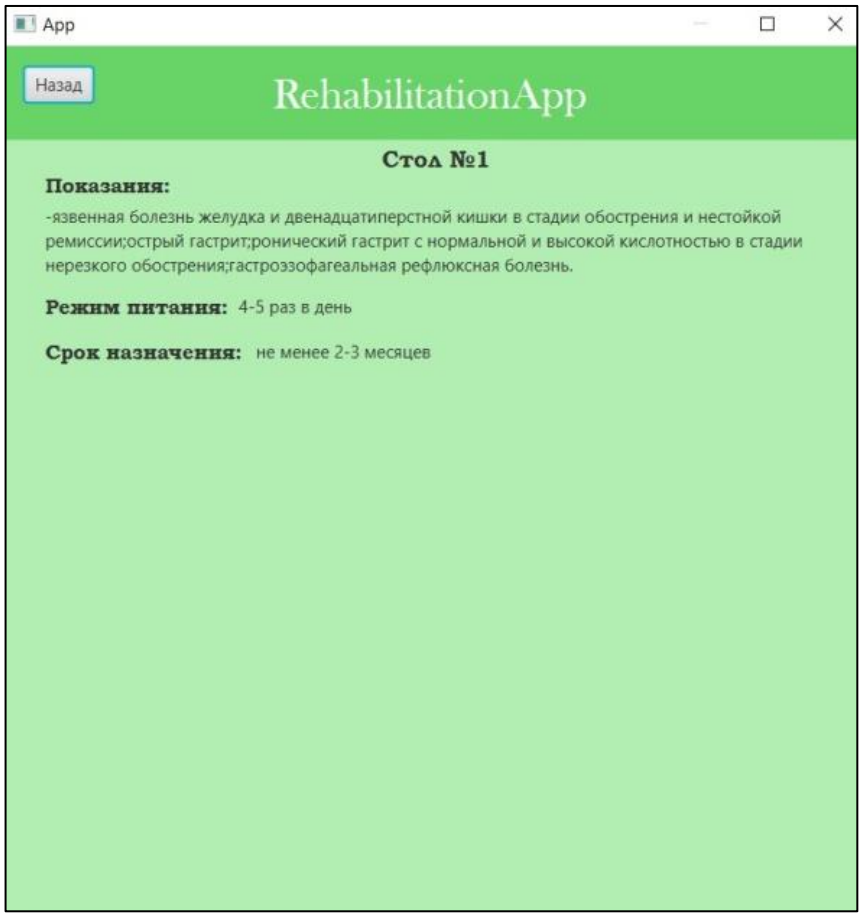


Рисунок 2.18 — Окно «Питание»

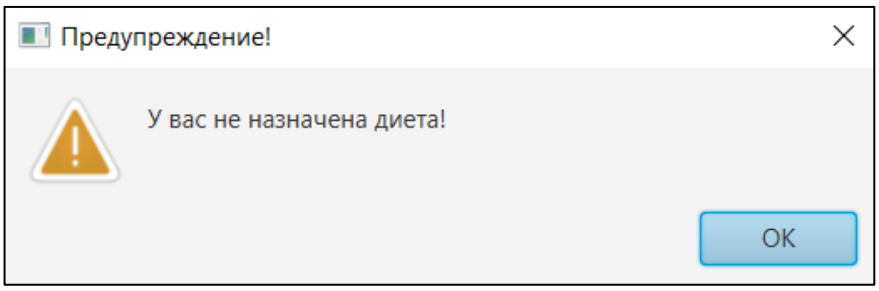


Рисунок 2.19 — Диалоговое окно с сообщением

Изменение личных данных

Необходимо пройти процесс авторизации, который описан в задачах выше. В открывшемся окне «Меню» необходимо нажать левой кнопкой мыши кнопку «Мои данные». В окне появится пустая форма и две кнопки (Рисунок 2.20).

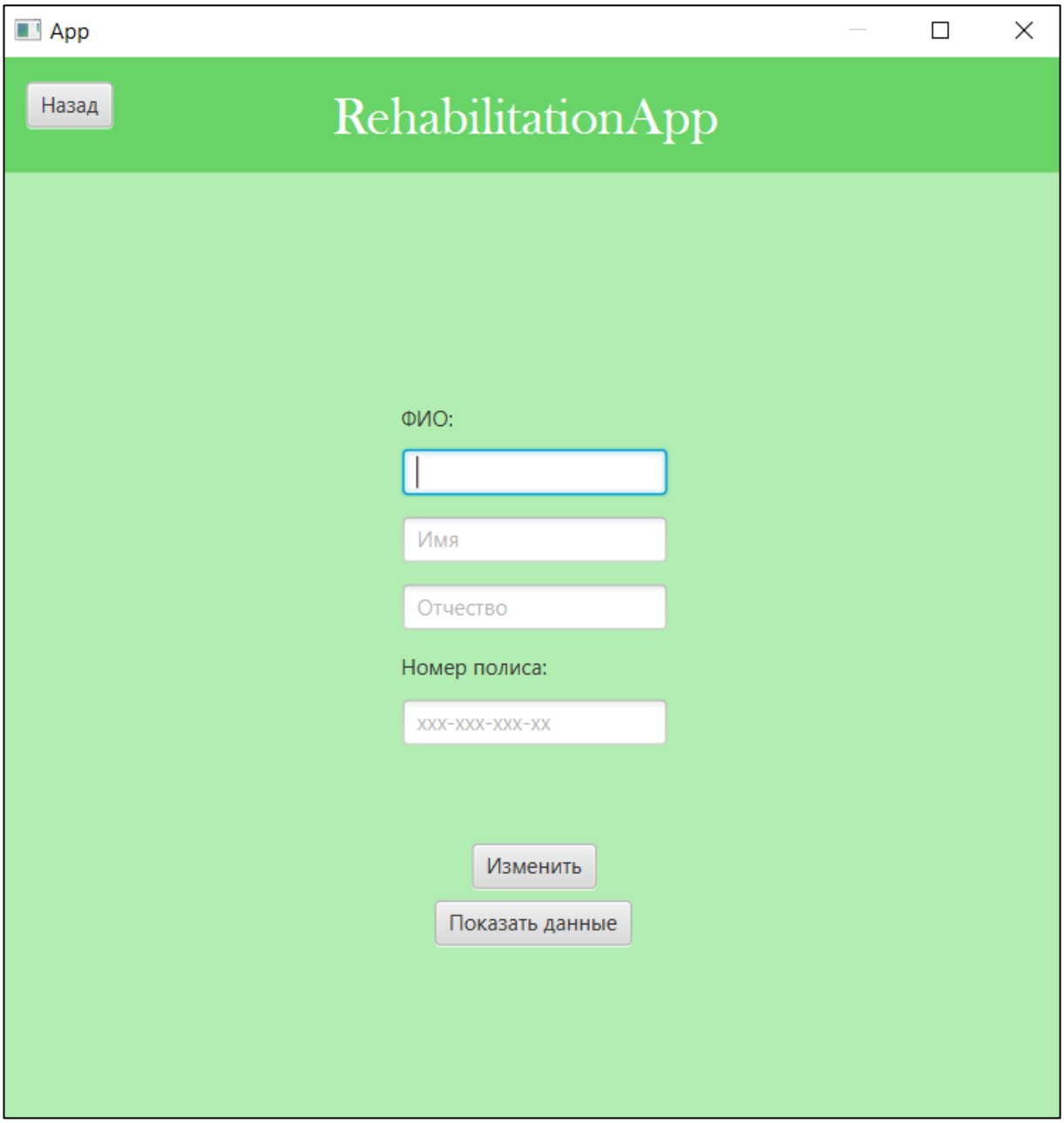


Рисунок 2.20 — Окно «Мои данные»

Для того чтобы посмотреть свои данные необходимо левой кнопкой мыли нажать на кнопку «Показать данные» (Рисунок 2.21).

Чтобы изменить данные необходимо нажать левой кнопкой мыши по полю, которое необходимо изменить и ввести корректные данные. Затем левой кнопкой мыши нажать кнопку «Изменить».

Появится диалоговое окно с подтверждением, для подтверждения действий необходимо нажать кнопку «ОК», для отмены действий нажать кнопку «Cancel» (Рисунок 2.22). Далее появится сообщение об успешном изменении данных (Рисунок 2.23).

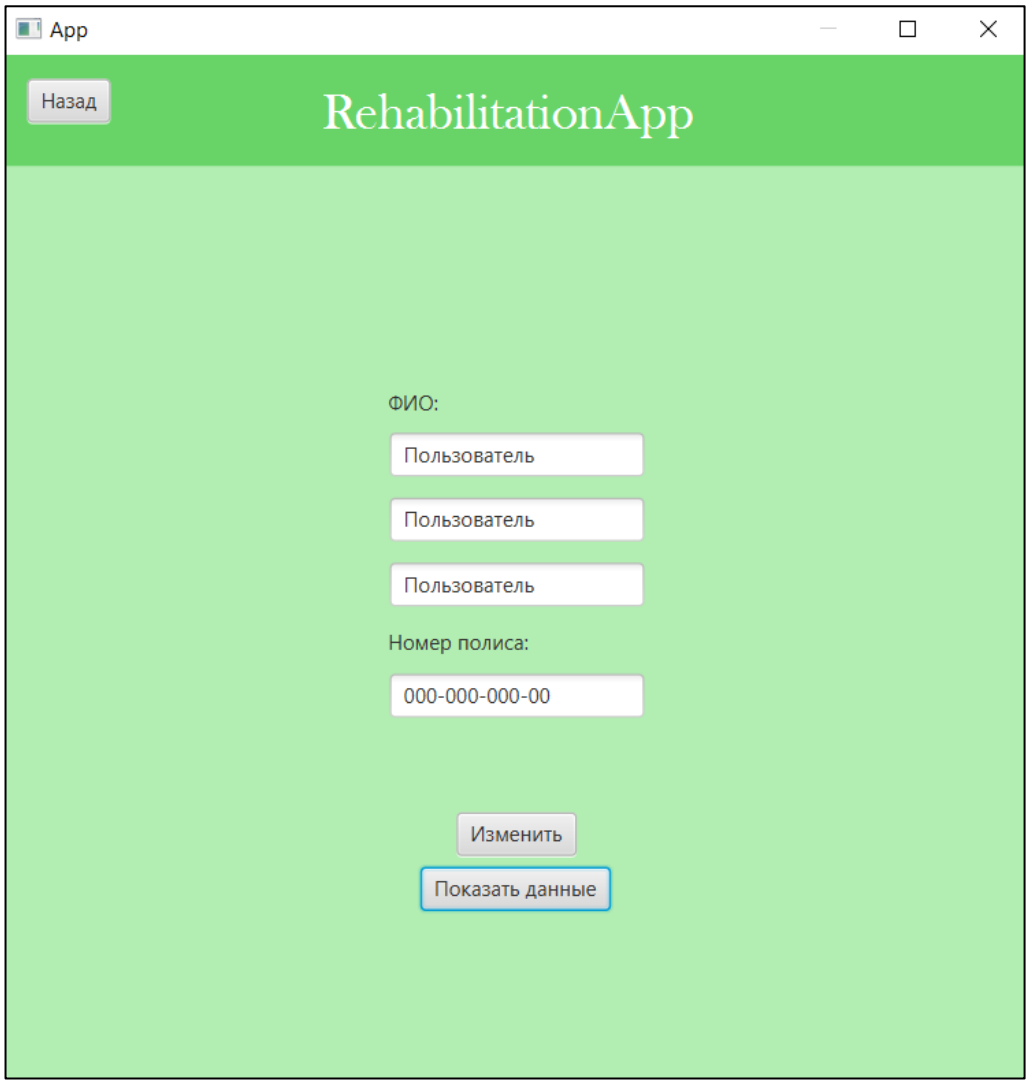


Рисунок 2.21 — Окно «Мои данные» после нажатия кнопки «Показать данные»

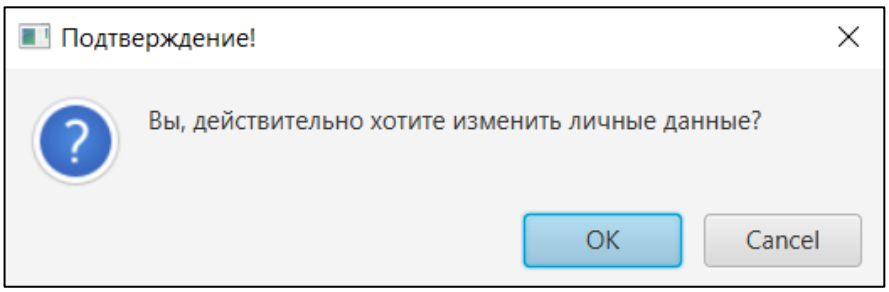


Рисунок 2.22 — Диалоговое окно с подтверждением действий

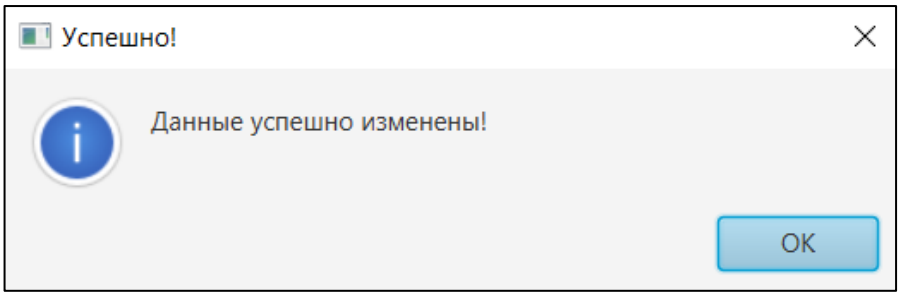


Рисунок 2.23 — Диалоговое окно с сообщением об успешном изменении данных

Категория “Врач”:

Просмотр показателей пациентов

Необходимо пройти процесс авторизации, который описан в задачах выше. В открывшемся окне «Меню» необходимо нажать левой кнопкой мыши кнопку «Анализ показателей».

Появится новое окно с таблицей, где будет представлена вся информация по показателям пациентов (Рисунок 2.24).

Также в поле ввода, расположенное над таблицей, можно ввести фамилию, имя и отчество пациента для его поиска среди остальных пациентов.

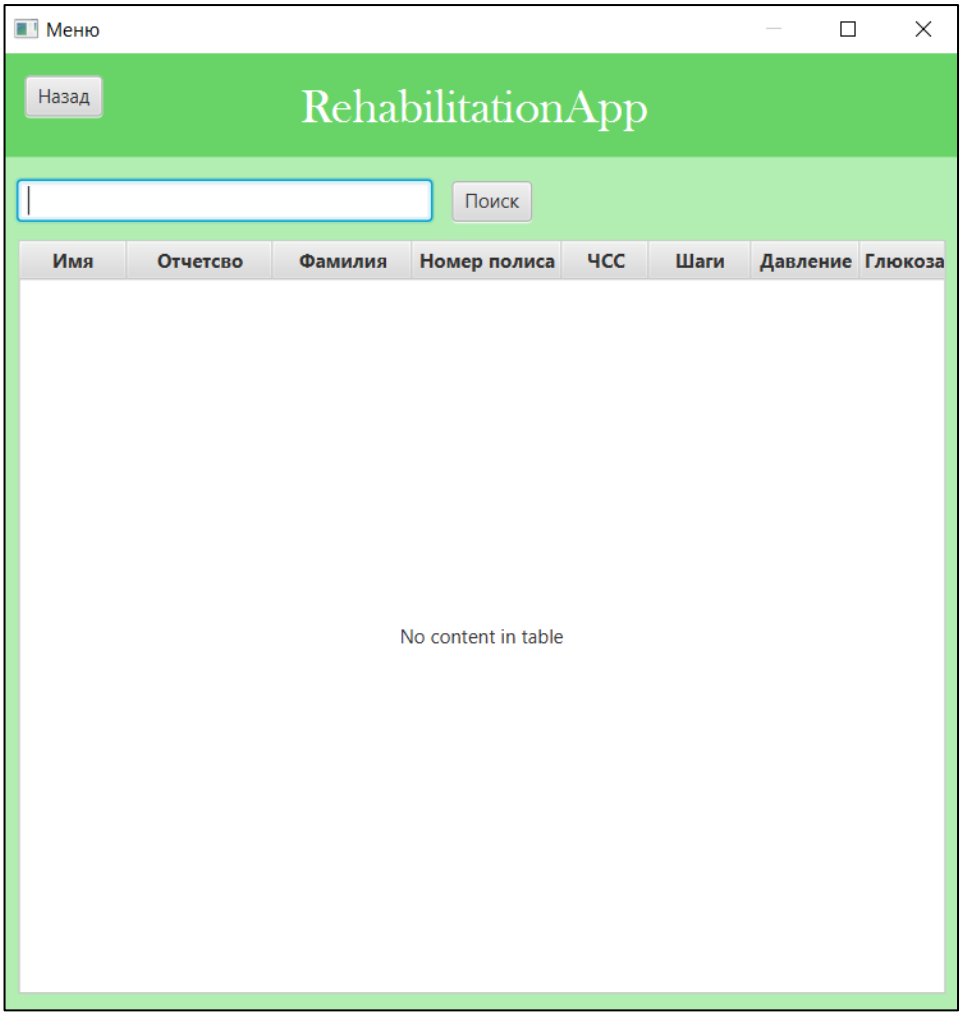


Рисунок 2.24 — Окно «Анализ показателей»

Просмотр и изменение данных пациентов

Необходимо пройти процесс авторизации, который описан в задачах выше. В открывшемся окне «Меню» необходимо нажать левой кнопкой мыши кнопку «Пациенты».

Появится новое окно с таблицей, где будет представлена вся информация о пациентах (Рисунок 2.25). Также в поле ввода можно ввести номер полиса пациента для его поиска среди остальных пациентов.

В этом же окне реализована функция назначения программы реабилитации. В форме, расположенной внизу окна необходимо ввести номер полиса пациента, которому назначается программа. Затем справа от формы нажать на список заболеваний «Заболевание», выбрать необходимое поле и нажать на него. Проделать те же самые действия с полями «Диета» и «Комплекс». Нажмите кнопку «Добавить». Изменения сразу отобразятся в таблице.

Для поиска конкретного пациента в базе реализован динамический фильтр, находящийся над таблицей. Чтобы найти пациента введите в соответствующие поле полис пациента и нажмите на кнопку «Поиск».

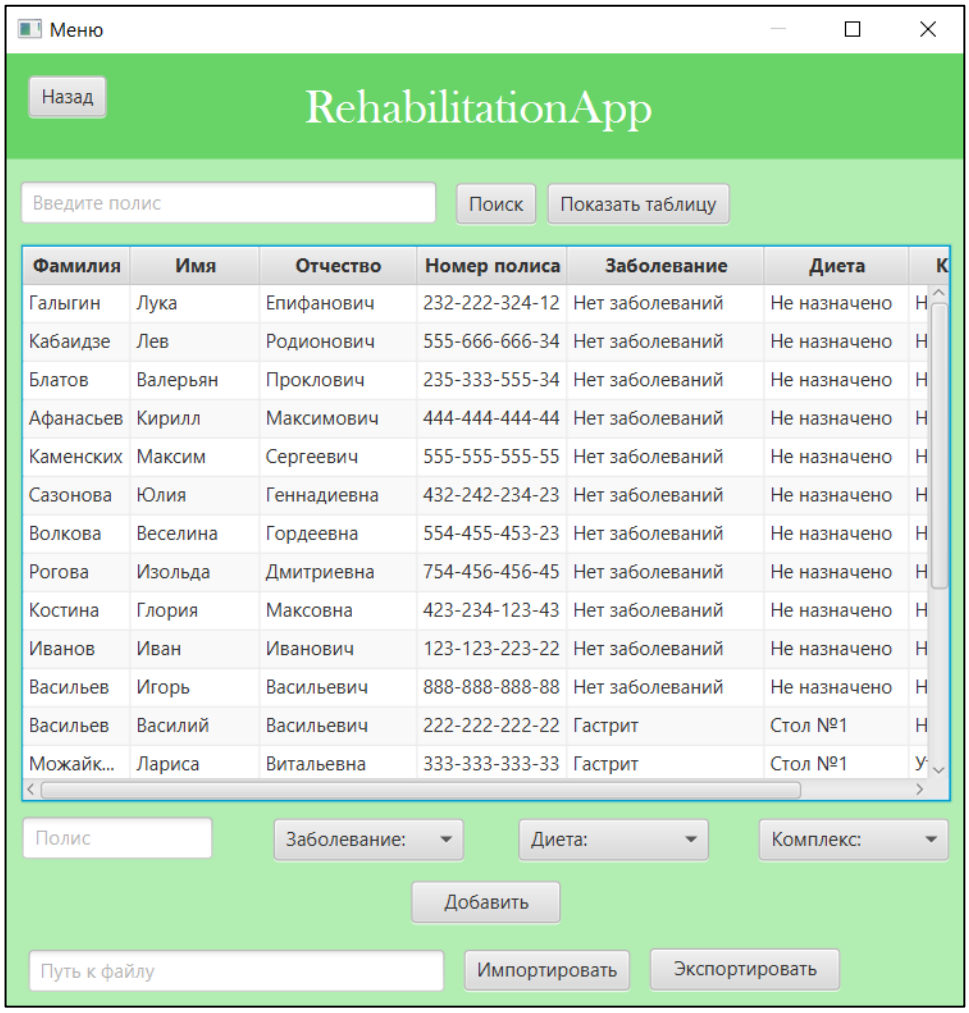


Рисунок 2.25 — Окно «Пациенты»

Импорт и экспорт данных

Необходимо пройти процесс авторизации, который описан в задачах выше. В открывшемся окне «Меню» необходимо нажать левой кнопкой мыши кнопку «Пациенты».

Внизу окна расположено поле ввода и две кнопки «Импортировать», «Экспортировать». При нажатии левой кнопкой мыши на кнопку «Импортировать», появляется окно с выбором csv-файла, необходимо выбрать csv-файл и нажать левой кнопкой мыши кнопку «Открыть». Все данные из файла будут выгружены в таблицу выше.

При нажатии левой кнопкой мыши на кнопку «Экспортировать», появится диалоговое окно с подтверждением действий, необходимо нажать «ОК» для экспортирования данных и «Cancel» для отмены операции экспортирования (Рисунок 2.26).

После нажатия «ОК» на компьютере создастся csv-файл с названием file.csv. Если файл с таким именем уже существует, то появится окно с соответствующим сообщением (Рисунок 2.27). При успешном экспорте данных выведется соответствующее сообщение (Рисунок 2.28).

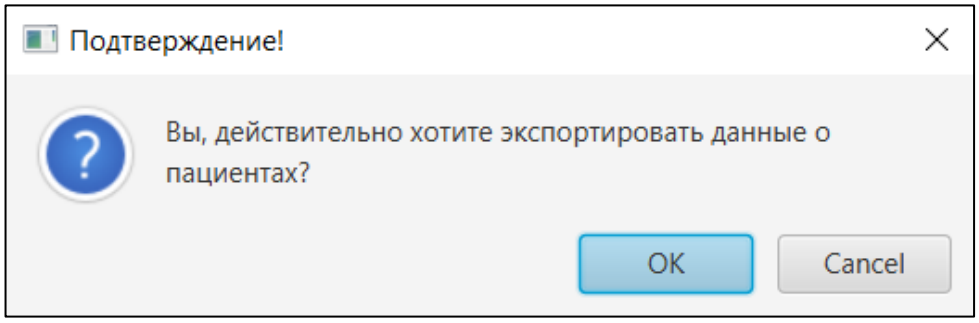


Рисунок 2.26 — Диалоговое окно с подтверждением действий

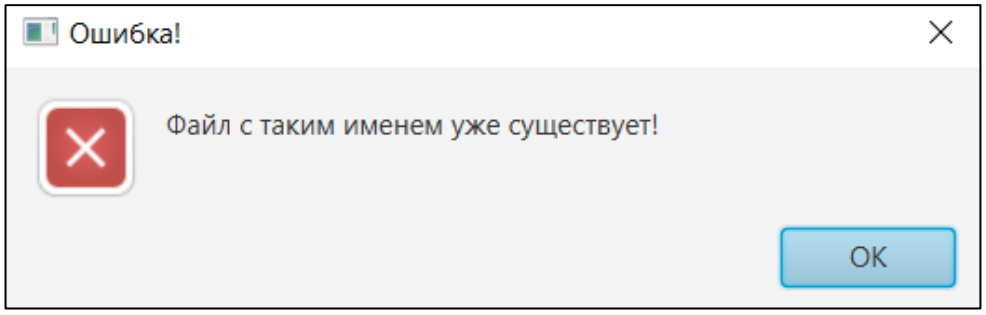


Рисунок 2.27 — Диалоговое окно с сообщением об ошибке

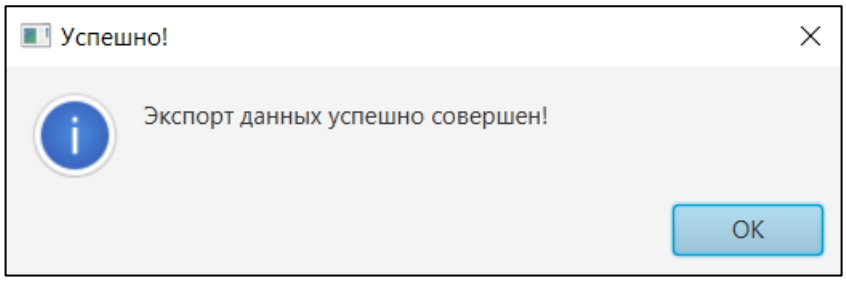


Рисунок 2.28 — Диалоговое окно с сообщением об успешном экспорте данных

Категория “Администратор”:

Просмотр данных о пользователях для администратора

Необходимо пройти процесс авторизации, который подробно описан в задачах выше. В открывшемся окне «Меню» нажимаем кнопку «Управление».

Появится новое окно с таблицей, где будет представлена вся информация о пользователях (Рисунок 2.29).

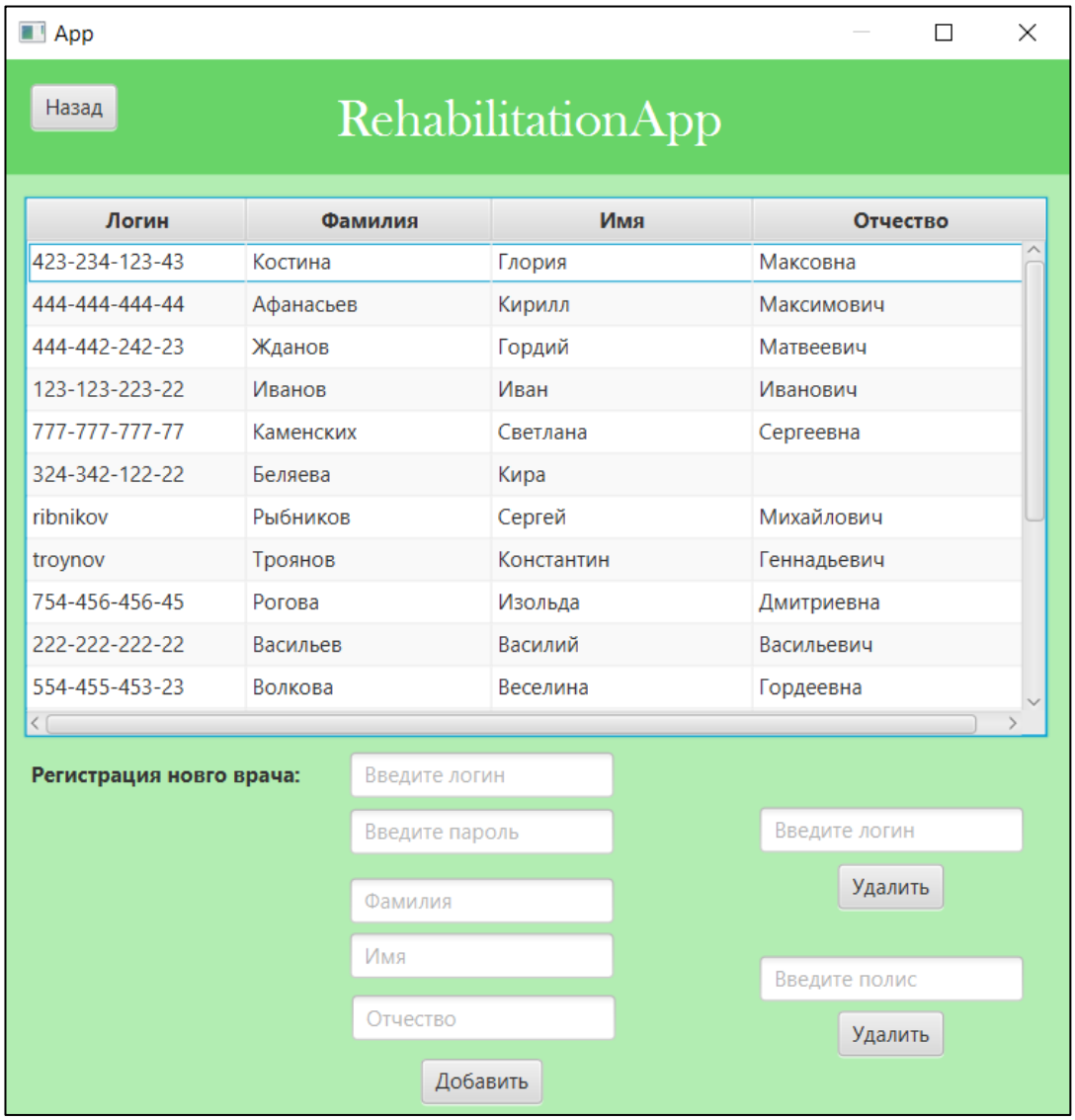


Рисунок 2.29 — Окно «Управление»

В этом же окне реализованы несколько функций: добавление нового врача, удаление пациента, удаление врача.

В центре расположена форма для добавления нового врача. Необходимо в текстовые поля ввести: логин, пароль, фамилию, имя и отчество. Затем нажмите кнопку «Добавить». Изменения сразу отобразятся в таблице.

Справа внизу расположены формы для удаления врача и пациента. Чтобы удалить врача, необходимо в поле с надписью «Введите логин» ввести логин врача, которого необходимо удалить. Затем необходимо нажать левой кнопкой мыши по кнопке «Удалить», которая находится под данным текстовым полем. Врач удален из системы.

Чтобы удалить пациента, необходимо в поле с надписью «Введите полис» ввести полис пациента, которого необходимо удалить. Затем необходимо нажать кнопку, удалить, которая находится под данным текстовым полем. Пациент удален из системы.

При успешном удалении выведется соответствующее сообщение (Рисунок 2.30).

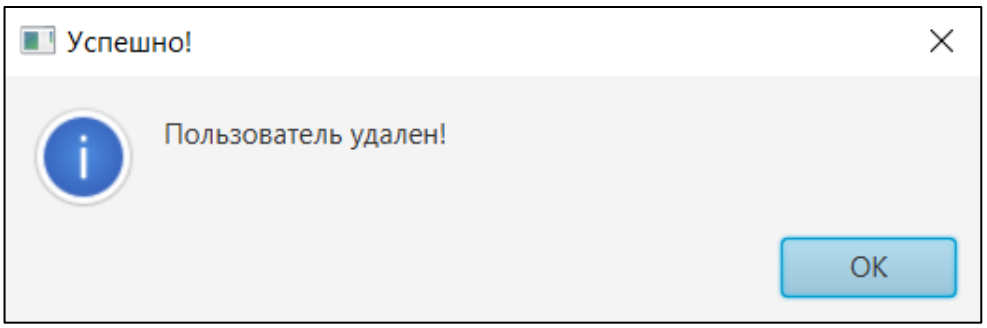


Рисунок 2.30 — Диалоговое окно с сообщением об успешном удалении пользователя

2.3 Особенности функционирования приложения

На рисунке 2.31 представлена диаграмма последовательности сценария авторизации пользователя в нотации UML 2.0 [4]. На диаграмме последовательности показаны взаимодействия объектов, упорядоченные по времени их проявления [2].

В приложении А находятся фрагменты кода, по которому описывался данный сценарий.

Реализована проверка целостности данных – помимо условий целостности данных в базе данных в приложении так же проверяются все данные, вводимые пользователем на корректность [9].

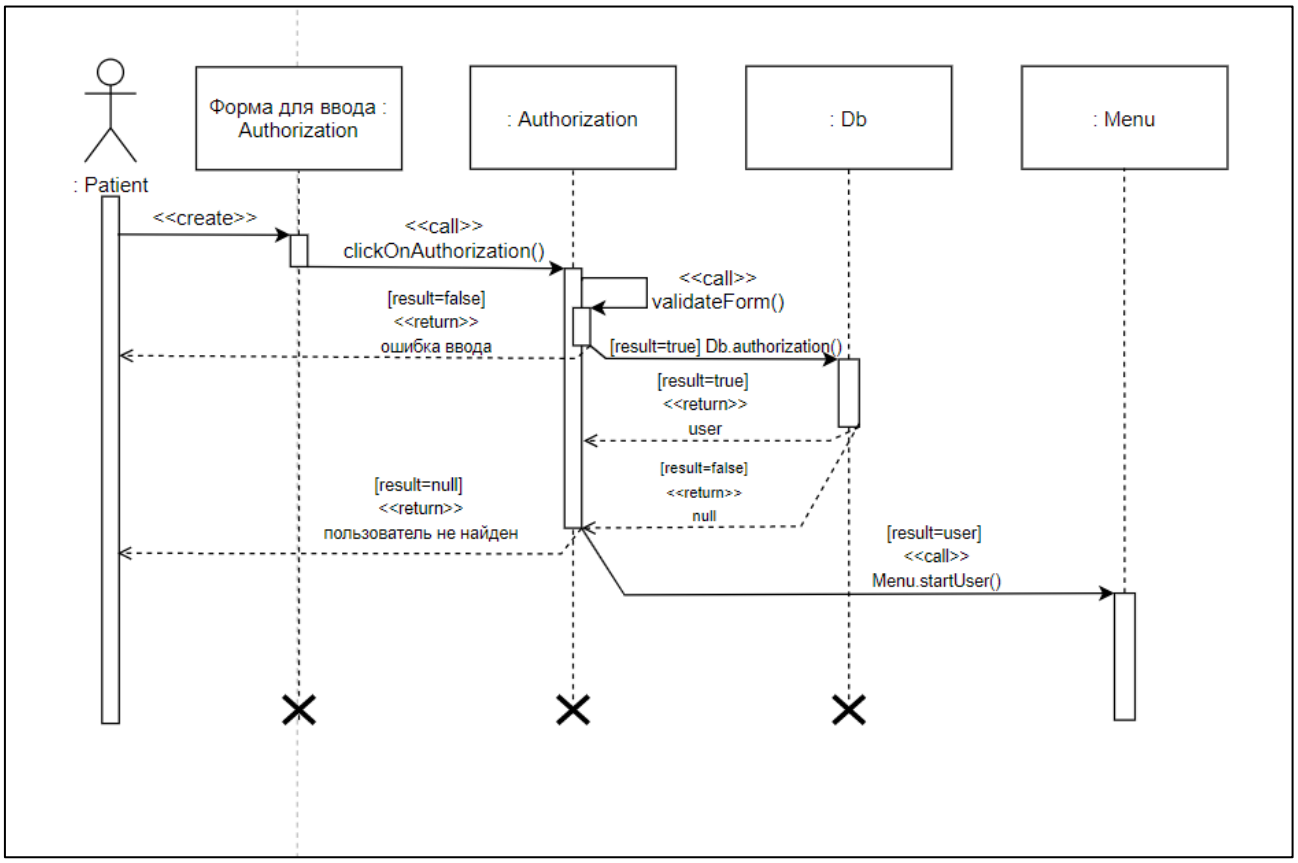


Рисунок 2.31 — Диаграмма последовательности сценария «Авторизация пациента»

2.4 Развитие проекта

Функциональные возможности системы на данном этапе представлены на диаграмме прецедентов в нотации UML 2.0 (Рисунок 2.32) [8].

Диаграмма прецедентов использования показывает отношения между актерами и прецедентами, описывая данную систему на концептуальном уровне. В данной системе существуют четыре актера: пользователь, авторизированный пациент, авторизированный врач, авторизированный администратор [3].

Функциональные возможности системы в будущем представлены на диаграмме прецедентов в нотации UML 2.0 (Рисунок 2.33).

План дальнейшего развития проекта включает в себя разработку новых функций, а так же улучшений существующих.

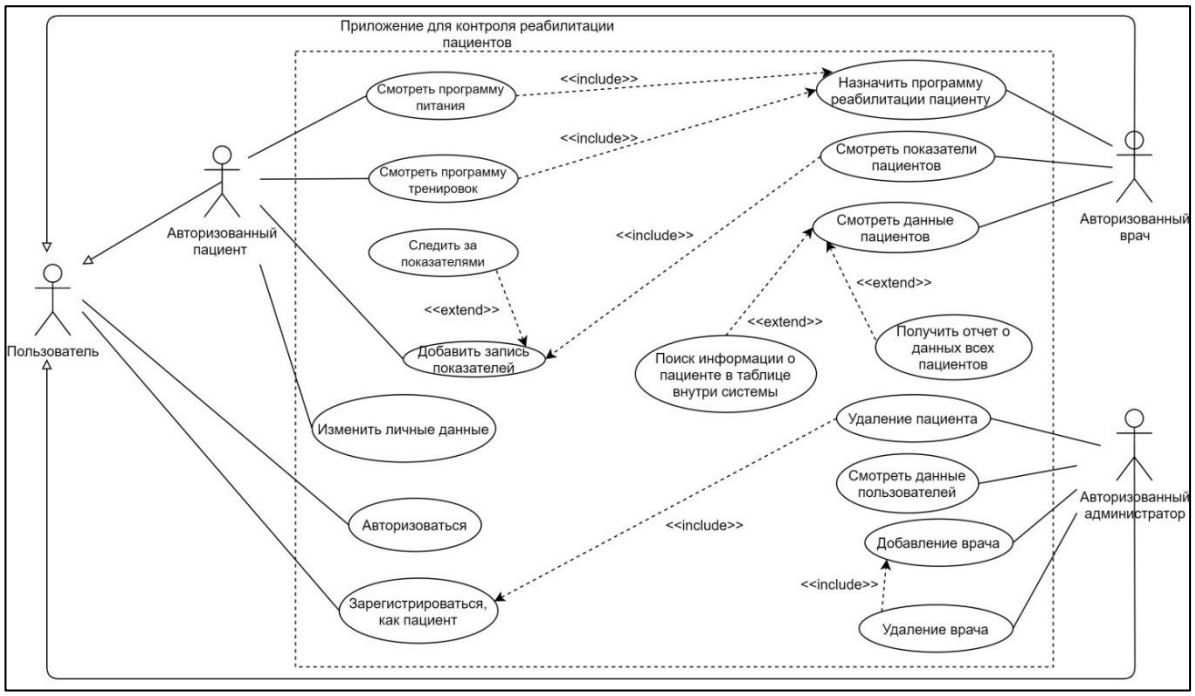


Рисунок 2.32 — Диаграмма вариантов использования системы на данном этапе

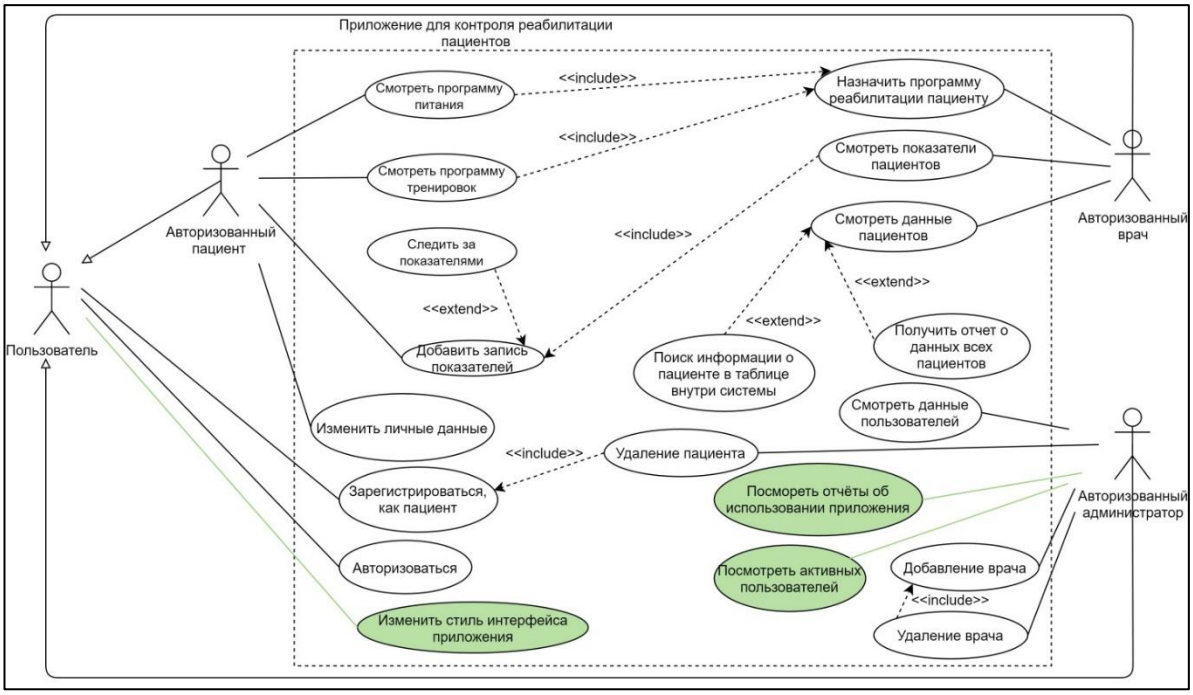


Рисунок 2.33 — Диаграмма вариантов использования системы в будущем

2.5 Выводы по проектной части

В данной части были построены следующие диаграммы: ER-диаграмма, диаграмма прецедентов, диаграмма последовательности. Было написано подробное руководство пользователя, где были рассмотрены все реализованные функции приложения, а также те, что будут разрабатываться в качестве перспективы на будущее.

ЗАКЛЮЧЕНИЕ

Анализируя тему данной курсовой работы, можно подвести общий итог проделанной работы. В аналитической части были описаны: постановка проблемы, анализ программного обеспечения, формальные модели предметной области, план реализации проекта. В практической части были описаны: модель данных, руководство пользователя, особенности функционирования приложения, развитие проекта. В результате чего, было определено, что конкурентоспособность в данной сфере на рынке достаточна высока. Это обусловлено актуальностью проблемы.

В рамках проделанной работы были изучены особенности создания программных продуктов и разработано приложение, для поликлиник позволяющее врачам осуществлять контроль за показателями пациентов, а пациентам в свою очередь позволяет вести учет своих показателей.

СПИСОК ЛИТЕРАТУРЫ

1. Блинов, И.Н., Романчик, В. С., Методы программирования: уч: мет. пособие / И. Н. Блинов, В. С. Романчик. - Минск: издательство «Четыре четверти», 2013. - 896 с.

2. Диаграмма последовательности (sequence diagram) [Электронный ресурс]. – Режим доступа: http://khpi-iip.mipk.kharkiv.edu/library/case/leon/gl8/gl8.html. — (Дата обращения: 22.05.2020).

3. Диаграмма прецедентов (вариантов использования) UML [Электронный ресурс]. – Режим доступа: https://planerka.info/item/diagramma-precedentov-variantov-ispolzovaniya-uml/. – (Дата обращения: 25.05.2020).

4. Диаграммы последовательности [Электронный ресурс]. – Режим доступа: https://wm-help.net/lib/b/book/4205506485/120. — (Дата обращения: 03.06.2020).

5. Документация к PostgreSQL 9.6.18 / Перевод на русский язык, 2015-2020 гг.: Компания «Постгрес Профессиональный», 2015. – 2468 с.

6. Е.А. Роганов. Основы информатики и программирования: Учебное пособие — М.: МГИУ, 2001. — 315 с.

7. Косяков А., Свит У. и др. Системная инженерия. Принципы и практика. Пер. с англ. под ред. В. К. Батоврина. - М.: ДМК Пресс, 2014. - 624 с.: ил.

8. Леоненков А. В. Самоучитель UML 2. - СПб.: БХВ-Петербург, 2007. - 576 с.: ил.

9. Макаров Е.М. Элементы двумерной графики в java: Учебно-методическое пособие. — Нижний Новгород: Нижегородский госуниверситет, 2017. – 56 с.

10. Методология функционального моделирования IDEF0/ — ИПК Издательство стандартов, 2000 — 75 с.

11. Моргунов, Е. П., PostgreSQL. Основы языка SQL: учеб, пособие / Е. П. Моргунов; под ред. Е. В. Рогова, П. В. Лузанова. — СПб.: БХВ-Петербург, 2018. — 336 с.: ил.

12. Основы технологий баз данных: учеб. пособие / Б. А. Новиков, Е. А. Горшкова; под ред. Е. В. Рогова. — М.: ДМК Пресс, 2019. — 240 с.

13. Роганов Е.А., Информатика: методические указания по выполнению курсовой работы для студентов направления «Прикладная математика и информатика» и специальности «Математическое обеспечение и администрирование информационных систем». — М.: МГИУ, 2007. — 32 с.

14. Функциональная диаграмма [Электронный ресурс]. – Режим доступа: https://studref.com/311808/informatika/funktsionalnye_diagrammy. — (Дата обращения: 26.05.2020).

15. Функциональное моделирование на базе стандарта IDEF0: метод. указания / сост. Д.Ю. Киселев, Ю.В. Киселев, А.В. Вавилин. – Самара: Изд-во СГАУ, 2014. – 20 с.

ПРИЛОЖЕНИЕ

Приложение А. Фрагменты кода программного продукта

Листинг А.1 – код класса Control – класс-контроллер для control.fxml.

```
package admin;
import db.Db;
import javafx.collections.*;
import javafx.event.ActionEvent;
import javafx.fxml.*;
import javafx.scene.*;
import javafx.scene.control.*;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.stage.*;
import sample.*;
import sample.Menu;
public class Control {
    @FXML
    private Button back;
    @FXML
    private TableView tableView;
    @FXML
    private TableColumn<UserLine, String> loginT, surnameT, nameT, midnameT;
    @FXML
    private TextField login_for_delete, polis_for_delete, login, password, surname, name, midname;

    public static ObservableList<UserLine> usersData = FXCollections.observableArrayList();

    @FXML
    public void initialize() throws Exception {// инициализирует форму данными
        usersData.clear();
        Db.tableFromAdmin();
        // устанавливает тип и значение которое должно хранится в колонке
        loginT.setCellValueFactory(new PropertyValueFactory<UserLine, String>("login"));
        surnameT.setCellValueFactory(new PropertyValueFactory<UserLine, String>("surename"));
        nameT.setCellValueFactory(new PropertyValueFactory<UserLine, String>("name"));
        midnameT.setCellValueFactory(new PropertyValueFactory<UserLine, String>("midname"));
        // заполняет таблицу данными
        tableView.setItems(usersData);
    }

    @FXML
    public void back(ActionEvent event) throws Exception {
        Stage stage = (Stage) back.getScene().getWindow();
        stage.close();
        Menu.startAdmin();
    }

    public static void start() throws Exception {
        Stage stage = new Stage();
        FXMLLoader fxml = new FXMLLoader(Menu.class.getResource("../admin/control.fxml"));
        Parent root = (Parent) fxml.load();
```

```
        stage.initModality(Modality.APPLICATION_MODAL);
        stage.setTitle("App");
        stage.setScene(new Scene(root, 600, 600));
        stage.show();
    }

    @FXML
    public void add(ActionEvent event) throws Exception {// добавляет врача
        String login_ = login.getText();
        String password_ = password.getText();
        String surname_ = surname.getText();
        String name_ = name.getText();
        String midname_ = midname.getText();
        if (login_ != null && password_ != null && surname_ != null && name_ != null &&
midname_ != null) {
            if (Registration.alertConfirmation("Вы уверены, что хотите добавить нового врача?")) {
                Db.addDoctor(login_, password_, surname_, name_, midname_);
            }
        } else{
            Registration.alertError("Заполните все поля корректными данными!");
        }
        login.clear();
        password.clear();
        surname.clear();
        name.clear();
        midname.clear();
        initialize();
    }
    @FXML
    public void delete(ActionEvent event) throws Exception {// удаляет доктора
        if (Registration.alertConfirmation("Вы уверены, что хотите удалить пользователя?")) {
            System.out.println("Ok");
            String login = login_for_delete.getText();
            Db.deleteDoctor(login);
            login_for_delete.clear();
            initialize();
        } else
            System.out.println("Cancel");
    }
    @FXML
    public void deleteOfPolis(ActionEvent event) throws Exception {// удаляет пацента
        if (Registration.alertConfirmation("Вы уверены, что хотите удалить пользователя?")) {
            System.out.println("Ok");
            String login = polis_for_delete.getText();
            Db.deletePatient(login);
            polis_for_delete.clear();
            initialize();
        } else
            System.out.println("Cancel");

    }
}
```

Листинг А.2 – код класса Db – обеспечивает подключение к БД и содержит все запросы необходимые для программы.

```
package db;
import admin.Control;
import doctor.Patients;
import sample.Registration;
import sample.UserLine;
import user.Indicator;
import java.sql.*;
public class Db extends Configs {
   private static Connection dbConnection;

   public static Connection getDbConnection() throws ClassNotFoundException, SQLException {
// подключение к бд
      Class.forName("org.postgresql.Driver");
      String connection = "jdbc:postgresql://" + dbHost + ":" + dbPort + "/" + dbName;
      dbConnection = DriverManager.getConnection(connection, dbUser, dbPass);
      System.out.println("\nПодключение установлено!\n");
      System.out.println(2);
      return dbConnection;
   }
   public static boolean checkPolis(String polis) throws SQLException, ClassNotFoundException {
// проверка уникальности полиса при регистрации
      ResultSet resSet = null;
      String insert = "SELECT * FROM user_info WHERE login  = "' + polis + '"";
      Statement st = dbConnection.createStatement();
      resSet = st.executeQuery(insert);
      if (resSet.next()) {
         System.out.println("Пользователь с таким полисом уже существует!\n");
         return false;
      } else {
         return true;
      }
   }
   public static String authorization(String polis, String password) throws SQLException,
ClassNotFoundException {
      // авторизация пользователей
      ResultSet resUs = null;
      String searchUser = "SELECT * FROM user_info where login = "' + polis + "' and
password="' + password + '"";
      Statement stU = getDbConnection().createStatement();
      resUs = stU.executeQuery(searchUser);
      if (resUs.next()) {
         if (resUs.getInt(6) == 1) {
            Const.TABLE = Const.USER_TABLE;
            System.out.println("Пользователь найден!\n");
            Const.POLIS = resUs.getString(1);
            System.out.println("ID_LOGIN - " + Const.POLIS);
            Const.STATUS = "user";
            Registration.alertInformation("Вы успешно авторизировались!");
            return "user";
```

```
            } else if (resUs.getInt(6) == 2) {
                Const.TABLE = Const.USER_TABLE;
                System.out.println("Доктор найден!\n");
                Const.STATUS = "doctor";
                Registration.alertInformation("Вы успешно авторизировались!");
                return "doctor";
            } else if (resUs.getInt(6) == 3) {
                Const.TABLE = Const.USER_TABLE;
                System.out.println("Администратор найден!\n");
                Const.STATUS = "admin";
                Registration.alertInformation("Вы успешно авторизировались!");
                return "admin";
            } else {
                return "null";
            }


        } else {
            return "null";
        }
    }
    public static void addUser(String polis, String password, String surename, String name, String
midname)
        // добавление пользователя в бд
            throws SQLException, ClassNotFoundException {
        String insert = "INSERT INTO user_info(login,password,surename,name,midname,role_id)
VALUES ('"
                + polis + "','" + password + "','" + surename + "','" + name + "','" + midname + "',1);
INSERT INTO " +
                "patients_data VALUES('" + polis + "',1,1,1)";
        Statement st = getDbConnection().createStatement();
        st.executeUpdate(insert);
        System.out.println("Пользователь внесен в базу данных!\n");
    }
    public static int dietDefinition() throws SQLException, ClassNotFoundException {
        // поиск комплекса питаниия пользователя
        int id_diet = 0;
        ResultSet resUs = null;
        String searchUser = "SELECT * FROM patients_data\nWHERE login ='" + Const.POLIS +
"'";
        Statement stU = dbConnection.createStatement();
        resUs = stU.executeQuery(searchUser);
        while (resUs.next()) {
            id_diet = resUs.getInt(3);
            System.out.printf("ID_DIET - %d \n", id_diet);
        }
        return id_diet;
    }
    public static int trainingDefinition() throws SQLException, ClassNotFoundException {
        // поиск комплекса упражнений пользователя
        int id_complex = 0;
        ResultSet resUs = null;
```

```
        String searchUser = "SELECT * FROM patients_data\nWHERE login ='" + Const.POLIS +
"'";
        Statement stU = dbConnection.createStatement();
        resUs = stU.executeQuery(searchUser);
        while (resUs.next()) {
            id_complex = resUs.getInt(4);
            System.out.printf("ID_COMPLEX - %d \n", id_complex);
        }
        return id_complex;
    }
    public static void addNoteI(int chss, int steps, String pressure, String sugar)
            throws SQLException, ClassNotFoundException {// добавление записей в Indicator
        String insert = "INSERT INTO indicators(login, chss, steps, pressure, sugar)  VALUES ('" +
Const.POLIS + "','" +
                chss + "','" + steps + "','" + pressure + "','" + sugar + "')";
        Statement st = dbConnection.createStatement();
        st.executeUpdate(insert);
        Registration.alertInformation("Показатели успешно записаны в таблицу!");
        System.out.println("Показатели внесены в базу данных!\n");
    }
    public static void addDoctor(String login, String password, String surename, String name, String
midname) throws SQLException, ClassNotFoundException {
        String insert = "INSERT INTO user_info " +
                "VALUES ('" + login + "','" + password + "','" + surename + "','" + name + "','" +
midname + "',2)";
        Statement st = dbConnection.createStatement();
        st.executeUpdate(insert);
        System.out.println(login + " добавлен в базу данных!\n");
    }
    public static void tableFromAdmin() throws SQLException, ClassNotFoundException {
        // вывод таблицы Пользователи
        ResultSet resUs = null;
        String union = "SELECT
user_info.login,user_info.name,user_info.midname,user_info.surename FROM user_info WHERE
role_id=1 OR role_id=2";
        Statement stU = dbConnection.createStatement();
        resUs = stU.executeQuery(union);
        while (resUs.next()) {
            String login = resUs.getString(1);
            String name = resUs.getString(2);
            String midname = resUs.getString(3);
            String surename = resUs.getString(4);
            Control.usersData.add(new UserLine(login, surename, name, midname, ""));
            System.out.printf("%s %s %s %s %s\n", login, surename, name, midname, "");
        }
    }
    public static void tableFromDoctor(String polis) throws SQLException,
ClassNotFoundException {
        // вывод таблицы Пациенты
        ResultSet resUs = null;
        String searchUser;
        if (polis.equals("")) {
```

```
            searchUser = "select
t.surename,t.name,t.midname,u.login,i.illness_name,y.diet_name,e.complex_name from
patients_data u\n" +
                    "join user_info t on t.login=u.login\n" +
                    "inner join illness i on u.id_illness=i.id_illness\n" +
                    "inner join diet y on u.id_diet=y.id_diet\n" +
                    "inner join complex e on u.id_complex=e.id_complex";
        } else {
            searchUser = "select
t.surename,t.name,t.midname,u.login,i.illness_name,y.diet_name,e.complex_name from
patients_data u\n" +
                    "join user_info t on t.login=u.login\n" +
                    "inner join illness i on u.id_illness=i.id_illness\n" +
                    "inner join diet y on u.id_diet=y.id_diet\n" +
                    "inner join complex e on u.id_complex=e.id_complex " +
                    "where u.login='" + polis + "'";
        }
        Statement stU = dbConnection.createStatement();
        resUs = stU.executeQuery(searchUser);
        while (resUs.next()) {
            Const.POLIS = resUs.getString(4);
            Const.SURNAME = resUs.getString(1);
            Const.NAME = resUs.getString(2);
            Const.MIDNAME = resUs.getString(3);
            Const.ILLNESS = resUs.getString(5);
            Const.DIET = resUs.getString(6);
            Const.COMPLEX = resUs.getString(7);
            Patients.patientData.add(new UserLine(Const.SURNAME, Const.NAME, Const.MIDNAME,
Const.POLIS, Const.ILLNESS, Const.DIET, Const.COMPLEX));
            System.out.printf("%s %s %s %s %s %s %s\n", Const.SURNAME, Const.NAME,
Const.MIDNAME, Const.POLIS, Const.ILLNESS, Const.DIET, Const.COMPLEX);
        }
    }
    public static void tableFromUser() throws SQLException, ClassNotFoundException {// вывод
таблицы Показатели
        ResultSet resUs = null;
        String searchUser = "SELECT * FROM indicators\nWHERE login='" + Const.POLIS + "'";
        Statement stU = dbConnection.createStatement();
        resUs = stU.executeQuery(searchUser);
        while (resUs.next()) {
            Const.Fi = resUs.getInt(2);
            Const.Se = resUs.getInt(3);
            Const.Tr = resUs.getString(4);
            Const.Fo = resUs.getString(5);
            Indicator.indicatorData.add(new UserLine(Const.Fi, Const.Se, Const.Tr, Const.Fo));
            System.out.printf("%d %d %s %s  \n", Const.Fi, Const.Se, Const.Tr, Const.Fo);
        }
    }
    public static void searchByPolis(String polis) throws SQLException, ClassNotFoundException {
        String search = "SELECT * FROM user_info WHERE login='" + polis + "'";
        Statement st = dbConnection.createStatement();
        if (st.executeQuery(search).next()) {
```

```
            Patients.patientData.clear();
            Db.tableFromDoctor(polis);
            Registration.alertInformation("Пациент с полисом " + polis + " найден!");
        } else Registration.alertError("Пациент с полисом " + polis + " не найден!");

    }
    public static void deleteDoctor(String login) throws SQLException, ClassNotFoundException {//
удаление врача из БД
        String delete = "DELETE FROM user_info WHERE login='" + login + "' and role_id=2";
        Statement st = dbConnection.createStatement();
        st.executeUpdate(delete);
        Registration.alertInformation("Пользователь удален!");
        System.out.println(login + " удален из базы данных!");
    }
    public static void deletePatient(String login) throws SQLException, ClassNotFoundException {//
удаление пациента из БД
        String delete = "DELETE FROM patients_data WHERE login='" + login + "';" +
                "DELETE FROM user_info WHERE login='" + login + "' and role_id=1";
        Statement st = dbConnection.createStatement();
        st.execute(delete);
        Registration.alertInformation("Пользователь удален!");
        System.out.println(login + " удален из базы данных!");
    }
    public static String[] getICD(String table, String name_column) throws SQLException,
ClassNotFoundException {
        // получение даных из БД для заполнения comboBox
        int count = count(table);
        String[] array = new String[count];
        String search = "SELECT " + name_column + " FROM " + table;
        Statement statement = dbConnection.createStatement();
        ResultSet rs = statement.executeQuery(search);
        for (int i = 0; i < count; i++) {
            if (rs.next()) {
                array[i] = rs.getString(1);// добавление значений в списо
            }
        }
        return array;
    }
    public static Integer count(String table) throws SQLException, ClassNotFoundException {//
подсчет кол-во строк в таблице
        int count = 1;
        String search = "SELECT count(*) FROM " + table;
        Statement st = dbConnection.createStatement();
        ResultSet rs = st.executeQuery(search);
        if (rs.next()) {
            count = rs.getInt(1);// кол-во строк
        }
        return count;
    }
    public static void changeParametrs(String num_polis, String illness, String complex, String diet)
throws SQLException, ClassNotFoundException {
        // изменение назначений для пациентов
```

```
        String i = getId(illness, "illness", "illness_name");
        String c = getId(complex, "complex", "complex_name");
        String d = getId(diet, "diet", "diet_name");
        String update = "UPDATE patients_data SET id_illness=" + i + ",id_diet=" + d +
",id_complex=" + c + " WHERE login='" + num_polis + "'";
        Statement st = dbConnection.createStatement();
        st.executeUpdate(update);
        System.out.println("Данные пользователя " + num_polis + " изменены!");
    }
    public static String getId(String name, String name_table, String name_column) throws
SQLException, ClassNotFoundException {
        String select = "SELECT * FROM " + name_table + " WHERE " + name_column + "='" +
name + "'";
        Statement st = dbConnection.createStatement();
        st.executeQuery(select);
        ResultSet rs = st.executeQuery(select);
        String id = null;
        if (rs.next()) {
            id = rs.getString(1);// кол-во строк
        }
        return id;
    }
    public static String[] getDataUser() throws SQLException {
        String s = "", n = "", m = "", p = "";
        ResultSet resUs = null;
        String searchUser = "SELECT * FROM user_info\nWHERE login ='" + Const.POLIS + "'";
        Statement stU = dbConnection.createStatement();
        resUs = stU.executeQuery(searchUser);
        while (resUs.next()) {
            s = resUs.getString(3);
            n = resUs.getString(4);
            m = resUs.getString(5);
            p = resUs.getString(1);
        }
        return new String[]{s, n, m, p};
    }
    public static void changeData(String polis, String surname, String name, String midname)
            throws SQLException {
        String update = "UPDATE user_info SET login='" + polis + "',surename='" + surname +
"',name='" + name + "',midname='" + midname + "' WHERE login='" + polis + "'";
        Statement st = dbConnection.createStatement();
        st.executeUpdate(update);
    }
}
```

Листинг А.3 – код класса Analisics – класс-контроллер для analysics.fxml.

```
package doctor;
import db.Db;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
```

```
import javafx.scene.*;
import javafx.stage.Modality;
import javafx.stage.Stage;
import sample.Menu;
import sample.UserLine;

public class Analysics {
    @FXML
    private TableView tableView;
    @FXML
    private TableColumn<UserLine, String> name, midname, surename, polis,
chss,steps,pressure,sugar;

    public static ObservableList<UserLine> patientData = FXCollections.observableArrayList();
    @FXML
    public void initialize() throws Exception { // инициализирует форму данными
        patientData.clear();
        //Db.tableUserAndParametr();
        // устанавливает тип и значение которое должно хранится в колонке
        // заполняет таблицу данными
        tableView.setItems(patientData);
    }
    public static void start() throws Exception {
        Stage stage = new Stage();
        FXMLLoader fxml = new FXMLLoader(Menu.class.getResource("../doctor/analysics.fxml"));
        Parent root = (Parent) fxml.load();
        stage.initModality(Modality.APPLICATION_MODAL);
        stage.setTitle("Меню");
        stage.setScene(new Scene(root, 600, 600));
        stage.show();
    }
    @FXML
    private Button back;

    @FXML
    public void back(ActionEvent event) throws Exception {
        Stage stage = (Stage) back.getScene().getWindow();
        stage.close();
        Menu.startDoctor();
    }
}
```

Листинг А.4 – код класса Patients – класс-контроллер для patients.fxml.

```
package doctor;
import db.Db;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.*;
import javafx.stage.*;
import sample.Authorization;
```

```
import sample.Menu;
import sample.Registration;
import sample.UserLine;

import java.io.*;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;
import java.util.Optional;

public class Patients {
    private FileChooser fileChooser = new FileChooser();
    private FileChooser.ExtensionFilter extensionFilter = new FileChooser.ExtensionFilter("csv-
файл", "*.csv");
    private File csvFile;
    @FXML
    private Button importing, export;
    @FXML
    private TextField path;
    @FXML
    private Button back, add, search, viewTable;
    @FXML
    private TextField polisT, searchFullName;
    @FXML
    private ComboBox illnessT, dietT, complexT;
    @FXML
    private TableView tableView;
    @FXML
    private TableColumn<UserLine, String> name, midname, surename, polis, illness, diet, complex;
    String p, i, d, c;
    public static ButtonType yes = new ButtonType("Да");
    public static ButtonType cancel = new ButtonType("Отмена");
    public static ObservableList<UserLine> patientData = FXCollections.observableArrayList();
    public static Optional<ButtonType> option;

    @FXML
    public void initialize() throws Exception { // инициализирует форму данным
        fileChooser.getExtensionFilters().add(extensionFilter);// для импорта,экспорта
        patientData.clear();
        Db.tableFromDoctor("");
        // устанавливает тип и значение которое должно хранится в колонке
        surename.setCellValueFactory(new PropertyValueFactory<UserLine, String>("surename"));
        name.setCellValueFactory(new PropertyValueFactory<UserLine, String>("name"));
        midname.setCellValueFactory(new PropertyValueFactory<UserLine, String>("midname"));
        polis.setCellValueFactory(new PropertyValueFactory<UserLine, String>("polis"));
        illness.setCellValueFactory(new PropertyValueFactory<UserLine, String>("illness"));
        diet.setCellValueFactory(new PropertyValueFactory<UserLine, String>("diet"));
        complex.setCellValueFactory(new PropertyValueFactory<UserLine, String>("complex"));
        // заполняет таблицу данными
        tableView.setItems(patientData);
    }
```

```
    public List<UserLine> Patients = new ArrayList<>();

    @FXML
    public void importing() {
        csvFile = fileChooser.showOpenDialog(Authorization.getMainStage());
        if (csvFile != null) {
            System.out.println(csvFile.getName());
            path.setText(csvFile.getAbsolutePath());

            try {
                if (Registration.alertConfirmation("Вы, действительно хотите импортировать данные
о пациентах?")) {
                    BufferedReader reader = new BufferedReader(new InputStreamReader(new
FileInputStream(csvFile), "cp1251"));
                    String line;
                    UserLine patient_;
                    patientData.clear();
                    while ((line = reader.readLine()) != null) {
                        String[] cells = line.split(";", -1);
                        patient_ = new UserLine();
                        patient_.setSurename(cells[0]);
                        patient_.setName(cells[1]);
                        patient_.setMidname(cells[2]);
                        patient_.setPolis(cells[3]);
                        patient_.setIllness(cells[4]);
                        patient_.setDiet(cells[5]);
                        patient_.setComplex(cells[6]);
                        Patients.add(patient_);
                    }
                    for (UserLine pat : Patients) {
                        patientData.add(new UserLine(pat.getSurename(), pat.getName(),
pat.getMidname(), pat.getPolis(), pat.getIllness(), pat.getDiet()
                                , pat.getComplex()));
                    }
                    tableView.setItems(patientData);
                    Registration.alertInformation("Данные о пациентах успешно импортировались!");
                }
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
    @FXML
    public void export() throws IOException {
        if (Registration.alertConfirmation("Вы, действительно хотите экспортировать данные о
пациентах?")) {
            String relativePath = "D:\\" + "file.csv";
            File file = new File(relativePath);
            if (file.createNewFile()) {
                System.out.println(relativePath + " файл создан в корневой директории проекта");
                BufferedWriter bufferedWriter = new BufferedWriter(new OutputStreamWriter(new
FileOutputStream(file), "cp1251"));
```

```
            for (UserLine pat : patientData) {
                bufferedWriter.write(pat.getSurename() + ";" + pat.getName() + ";" +
pat.getMidname() + ";" + pat.getPolis() + ";" + pat.getIllness() + ";" + pat.getDiet() + ";" +
                        pat.getComplex() + "\n");
            }

            bufferedWriter.close();
            Registration.alertInformation("Экспорт данных успешно совершен!");

        } else {
            System.out.println("Файл " + relativePath + " уже существует в директории проекта");
            Registration.alertError("Файл с таким именем уже существует!");
        }


    } else System.out.println("Cancelled!");

  }
  @FXML
  public void add(ActionEvent event) throws Exception { // изменяет болезнь|питание|комплекс
упражнения пациента
    if (Registration.alertConfirmation("Вы, уверены, что хотите изменить программу
реабилитации?")) {
        System.out.println("Ok!");
        i = "" + illnessT.getValue();// преобразовывает из объекта в строку
        c = "" + complexT.getValue();// преобразовывает из объекта в строку
        d = "" + dietT.getValue();// преобразовывает из объекта в строку
        p = polisT.getText();
        Db.changeParametrs(p, i, c, d);
        polisT.clear();
        illnessT.setItems(null);
        complexT.setItems(null);
        dietT.setItems(null);
        initialize();
    } else System.out.println("Cancelled!");
  }
  @FXML
  public void showI() throws SQLException, ClassNotFoundException {
    String[] array = Db.getICD("illness", "illness_name");// список из бд
    ObservableList<String> list = FXCollections.observableArrayList();
    for (int i = 0; i < array.length; i++) {
      list.add(array[i]);
    }
    illnessT.setItems(list);
  }
 @FXML
  public void showC() throws SQLException, ClassNotFoundException {
    String[] array = Db.getICD("complex", "complex_name");// список из бд
    ObservableList<String> list = FXCollections.observableArrayList();
    for (int i = 0; i < array.length; i++) {
      list.add(array[i]);
    }
```

```
        complexT.setItems(list);
    }
    @FXML
    public void showD() throws SQLException, ClassNotFoundException {
        String[] array = Db.getICD("diet", "diet_name");// список из бд
        ObservableList<String> list = FXCollections.observableArrayList();
        for (int i = 0; i < array.length; i++) {
            list.add(array[i]);
        }
        dietT.setItems(list);
    }
    @FXML
    public void search(ActionEvent event) throws Exception { // поиск пациента по номеру полиса
        p = searchFullName.getText();
        System.out.println(p);
        Db.searchByPolis(p);
        tableView.setItems(patientData);
    }
    @FXML
    public void viewTable(ActionEvent event) throws Exception {
        searchFullName.clear();
        initialize();
    }

    /**********/
    public static void start() throws Exception {
        Stage stage = new Stage();
        FXMLLoader fxml = new FXMLLoader(Menu.class.getResource("../doctor/patients.fxml"));
        Parent root = (Parent) fxml.load();
        stage.initModality(Modality.APPLICATION_MODAL);
        stage.setTitle("Меню");
        stage.setScene(new Scene(root, 600, 600));
        stage.show();
    }

    @FXML
    public void back(ActionEvent event) throws Exception {
        Stage stage = (Stage) back.getScene().getWindow();
        stage.close();
        Menu.startDoctor();
    }
}
```

Листинг А.5 – код класса Authorization – класс запускающий приложение, класс-контроллер для authorization.fxml.

```
package sample;

import db.Db;
import javafx.application.Application;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
```

```
import javafx.scene.*;
import javafx.scene.control.*;
import javafx.stage.*;

public class Authorization extends Application {

    @FXML
    private Button registration = new Button();
    @FXML
    private Button authorization = new Button();
    @FXML
    private TextField pol = new TextField();
    @FXML
    private PasswordField pass = new PasswordField();
    public String polis;
    public String password;
    private static Stage stage;
    @Override
    public void start(Stage primaryStage) throws Exception {
        Parent root = FXMLLoader.load(getClass().getResource("authorization.fxml"));
        primaryStage.setTitle("RehabilitationApp");
        primaryStage.setScene(new Scene(root, 600, 600));
        primaryStage.show();
    }
    public static Stage getMainStage() {
        return stage;
    }
    public static void startFromEtc() throws Exception {
        stage = new Stage();
        FXMLLoader fxml = new
FXMLLoader(Registration.class.getResource("authorization.fxml"));
        Parent root = (Parent) fxml.load();
        stage.initModality(Modality.APPLICATION_MODAL);
        stage.setTitle("RehabilitationApp");
        stage.setScene(new Scene(root, 600, 600));
        stage.show();
    }

    @FXML
    private void clickOnAuthorization(ActionEvent event) throws Exception { // обрабатывает
нажатие кнопки авторизация
        Db.getDbConnection();
        polis = pol.getText().trim();
        password = pass.getText().trim();
        System.out.println(3);

        if (validateForm(polis, password)) {
            String res_authorization=Db.authorization(polis, password);
            if (res_authorization.equals("user")) {
                Stage stage = (Stage) authorization.getScene().getWindow();
                stage.close();
                Menu.startUser();
```

```
            } else if (res_authorization.equals("doctor")) {
                System.out.println(1);
                Stage stage = (Stage) authorization.getScene().getWindow();
                stage.close();
                Menu.startDoctor();
            } else if (res_authorization.equals("admin")) {
                Stage stage = (Stage) authorization.getScene().getWindow();
                stage.close();
                Menu.startAdmin();
            } else {
                Registration.alertError("Данные введены неверно!");
                System.out.println("Пользователь не найден!");
            }
        }
    }
    @FXML
    private void clickOnRegistration(ActionEvent event) throws Exception { // обрабатывает
нажатие кнопки регистрация
        Stage stage = (Stage) registration.getScene().getWindow();
        stage.close();
        Registration.start();
    }
    public boolean validateForm(String polis, String password) throws Exception { // проверка
введенных данных
        if (!polis.equals("") && !password.equals("")) {
            return true;
        } else {
            Registration.alertError("Вы ввели неверный логин или пароль.\nПовторите попытку!");
            return false;
        }
    }
}
```

Листинг А.6 – код класса Registration – класс-контроллер для registration.fxml.

```
package sample;

import db.Db;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.*;
import javafx.scene.control.*;
import javafx.stage.*;

import java.util.Optional;

public class Registration {
    @FXML
    public TextField surename, namereg, middle_name_reg, polis_reg;
    @FXML
    private PasswordField password_reg, password_confirm;
```

```
    @FXML
    private Button reg, back;
    private String name, suname, midname, polis, password, password_conf;

    public static void start() throws Exception {
        Stage stage = new Stage();
        FXMLLoader fxml = new FXMLLoader(Registration.class.getResource("registration.fxml"));
        Parent root = (Parent) fxml.load();
        stage.initModality(Modality.APPLICATION_MODAL);
        stage.setTitle("Регистрация");
        stage.setScene(new Scene(root, 600, 600));
        stage.show();
    }

    @FXML
    public void registration(ActionEvent event) throws Exception {
        polis = polis_reg.getText();
        password = password_reg.getText();
        password_conf = password_confirm.getText();
        name = namereg.getText();
        suname = surename.getText();
        midname = middle_name_reg.getText();
        validateForm(polis, password, suname, name, midname, password_conf);
    }

    private void validateForm(String polis, String password, String suname, String name, String
midname, String password_conf) throws Exception {
        String regex_password = "(?=.*[0-9])(?=.*[a-z])[0-9a-zA-Z!@#$%^&*]{6,}";
        String regex_name = "[а-яА-Я]+";
        String regex_polis = "[0-9]{3}\\-[0-9]{3}\\-[0-9]{3}-[0-9]{2}";
        if ((name.matches(regex_name) && suname.matches(regex_name)) ||
(name.matches(regex_name) && suname.matches(regex_name) &&
midname.matches(regex_name))) {
            if (polis.matches(regex_polis) && Db.checkPolis(polis)) {
                if (password.matches(regex_password)) {
                    if (password.equals(password_conf)) {
                        Db.addUser(polis, password, suname, name, midname);
                        alertInformation("Регистрация прошла успешно!");
                        Stage stage = (Stage) reg.getScene().getWindow();
                        stage.close();
                        Authorization.startFromEtc();
                    } else {
                        alertError("Пароли не совпадают.\nПовторите попытку!");
                    }
                } else {
                    alertError("Пароль введен некорректно.\nПовторите попытку!");
                }
            } else {
                alertError("Введен некорректный полис.\nПовторите попытку!");
            }
        } else {
            alertError("Введены некорректные данные ФИО.\nПовторите попытку!");
```

```
        }
    }

    public static void alertInformation(String message) {//
        Alert alert = new Alert(Alert.AlertType.INFORMATION);
        alert.setTitle("Успешно!");
        alert.setContentText(message);
        alert.setHeaderText(null);
        alert.showAndWait();
    }

    public static void alertError(String message) {//
        Alert alert = new Alert(Alert.AlertType.ERROR);
        alert.setTitle("Ошибка!");
        alert.setContentText(message);
        alert.setHeaderText(null);
        alert.showAndWait();
    }

    public static void alertWarning(String message) {//
        Alert alert = new Alert(Alert.AlertType.WARNING);
        alert.setTitle("Предупреждение!");
        alert.setContentText(message);
        alert.setHeaderText(null);
        alert.showAndWait();
    }

    public static boolean alertConfirmation(String message) {//
        Alert alert = new Alert(Alert.AlertType.CONFIRMATION);
        alert.setTitle("Подтверждение!");
        alert.setContentText(message);
        alert.setHeaderText(null);
        Optional<ButtonType> option = alert.showAndWait();
        if (option.get() == ButtonType.OK) {
            return true;
        } else {
            return false;
        }
    }

    @FXML
    public void back(ActionEvent event) throws Exception {
        Stage stage = (Stage) back.getScene().getWindow();
        stage.close();
        Authorization.startFromEtc();
    }
}
```

Листинг А.7 – код класса Menu – класс-контроллер для menu.fxml, munu_doctor.fxml, menu_admin.fxml, отвечает за переход на страницы исходя из роли авторизированного пользователя.

```
package sample;

import admin.Control;
import db.Db;
import doctor.*;
import javafx.event.ActionEvent;
import javafx.fxml.*;
import javafx.scene.*;
import javafx.scene.control.Button;
import javafx.stage.*;
import user.*;
import java.io.IOException;

public class Menu {
    @FXML
    private Button
indicator,diet,training,information,exit,patients,indicator_analysis,patients_admin,doctors,control;

    public static void startUser() throws Exception {
        String name = "../user/menu.fxml";
        start(name);
    }

    public static void startDoctor() throws Exception {
        String name = "../doctor/menu_doctor.fxml";
        start(name);
    }
    public static void startAdmin() throws Exception {
        String name = "../admin/menu_admin.fxml";
        start(name);
    }
    public static void start(String name) throws IOException {
        Stage stage = new Stage();
        FXMLLoader fxml = new FXMLLoader(Menu.class.getResource(name));
        Parent root = (Parent) fxml.load();
        stage.initModality(Modality.APPLICATION_MODAL);
        stage.setTitle("Меню");
        stage.setScene(new Scene(root, 600, 600));
        stage.show();
    }
    @FXML
    public void exit(ActionEvent event) throws Exception {
        Stage stage = (Stage) exit.getScene().getWindow();
        stage.close();
        Authorization.startFromEtc();
    }

    /**** МЕНЮ ПАЦИЕНТА ****/
    @FXML
    public void indicator(ActionEvent event) throws Exception {
        Stage stage = (Stage) indicator.getScene().getWindow();
        stage.close();
```

```
    Indicator.start();
}

@FXML
public void diet(ActionEvent event) throws Exception {
    Stage stage = (Stage) diet.getScene().getWindow();
    stage.close();
    switch (Db.dietDefinition()) {
        case 2:
            Diet.start1();
            break;
        case 12:
            Diet.start12();
            break;
        case 15:
            Diet.start15();
            break;
        default:
            Registration.alertWarning("У вас не назначена диета!");
            Menu.startUser();
            break;
    }
}

@FXML
public void training(ActionEvent event) throws Exception {
    Stage stage = (Stage) exit.getScene().getWindow();
    stage.close();
    switch (Db.trainingDefinition()) {
        case 2:
            Training.start1();
            break;
        default:
            Registration.alertWarning("У вас не назначен комплекс!");
            Menu.startUser();
            break;
    }
}

@FXML
public void information(ActionEvent event) throws Exception {
    Stage stage = (Stage) exit.getScene().getWindow();
    stage.close();
    Information.start();
}

/**** МЕНЮ ДОКТОРА ****/
@FXML
public void patients(ActionEvent event) throws Exception {
    Stage stage = (Stage) exit.getScene().getWindow();
    stage.close();
    Patients.start();
```

```
    }

    @FXML
    public void indicator_analysis(ActionEvent event) throws Exception {
        Stage stage = (Stage) exit.getScene().getWindow();
        stage.close();
        Analysics.start();
    }

    /**** МЕНЮ АДМИНА ****/
    @FXML
    public void control(ActionEvent event) throws Exception {
        Stage stage = (Stage) control.getScene().getWindow();
        stage.close();
        Control.start();
    }
}
```

Листинг А.8 – код класса Diet – класс-контроллер для diet1.fxml, diet12.fxml, diet15.fxml.

```
package user;

import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.stage.Modality;
import javafx.stage.Stage;
import sample.Menu;


public class Diet {
    @FXML
    private Button back;
    @FXML
    public void back(ActionEvent event) throws Exception {
        Stage stage = (Stage) back.getScene().getWindow();
        stage.close();
        Menu.startUser();
    }
    public static void start(String name) throws Exception {
        Stage stage = new Stage();
        FXMLLoader fxml = new FXMLLoader(Menu.class.getResource(name));
        Parent root = (Parent) fxml.load();
        stage.initModality(Modality.APPLICATION_MODAL);
        stage.setTitle("App");
        stage.setScene(new Scene(root,600,600));
        stage.show();
    }
    public static void start1() throws Exception { // запускает окно для пациента с id_diet = 2
```

```
        String name = "../user/diet1.fxml";
        start(name);
    }
    public static void start12() throws Exception { // запускает окно для пациента с id_diet = 13
        String name = "../user/diet12.fxml";
        start(name);
    }
    public static void start15() throws Exception { // запускает окно для пациента с id_diet = 16
        String name = "../user/diet15.fxml";
        start(name);
    }
}
```

Листинг А.9 – код класса Indicator – класс-контроллер для indicator.fxml.

```
package user;
import db.Db;
import javafx.collections.*;
import javafx.event.ActionEvent;
import javafx.fxml.*;
import javafx.scene.*;
import javafx.scene.control.*;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.stage.*;
import sample.*;
import sample.Menu;

public class Indicator {
    @FXML
    private Button back, add;
    @FXML
    private TextField chssT, stepsT, pressureT, sugarT;
    @FXML
    private TableView tableView;
    @FXML
    private TableColumn<UserLine, Integer> chss, steps;
    @FXML
    private TableColumn<UserLine, String> pressure, sugar;
    public static ObservableList<UserLine> indicatorData = FXCollections.observableArrayList();
    @FXML
    public void initialize() throws Exception {// инициализирует форму данными
        indicatorData.clear();
        Db.tableFromUser();
        // устанавливает тип и значение которое должно хранится в колонке
        chss.setCellValueFactory(new PropertyValueFactory<UserLine, Integer>("chss"));
        steps.setCellValueFactory(new PropertyValueFactory<UserLine, Integer>("steps"));
        pressure.setCellValueFactory(new PropertyValueFactory<UserLine, String>("pressure"));
        sugar.setCellValueFactory(new PropertyValueFactory<UserLine, String>("sugar"));
        // заполняет таблицу данными
        tableView.setItems(indicatorData);
    }
    @FXML
    private void add(ActionEvent event) throws Exception { // добавление введенных данных из
полей (показатели здоровья) в бд
```

```
        try {
            int c = Integer.parseInt(chssT.getText());
            int st = Integer.parseInt(stepsT.getText());
            String p = pressureT.getText();
            String su = sugarT.getText();
            Db.addNoteI(c, st, p, su);
            chssT.clear();
            stepsT.clear();
            pressureT.clear();
            sugarT.clear();
            initialize();
        } catch (Exception exception) {
            exception.getMessage();
        }
    }
    @FXML
    public void back(ActionEvent event) throws Exception {
        Stage stage = (Stage) back.getScene().getWindow();
        stage.close();
        Menu.startUser();
    }

    public static void start() throws Exception {
        Stage stage = new Stage();
        FXMLLoader fxml = new
FXMLLoader(Indicator.class.getResource("../user/indicator.fxml"));
        Parent root = (Parent) fxml.load();
        stage.initModality(Modality.APPLICATION_MODAL);
        stage.setTitle("App");
        stage.setScene(new Scene(root, 600, 600));
        stage.show();
    }
}
```

Листинг А.10 – код класса Information – класс-контроллер для information.fxml.

```
package user;

import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.stage.Modality;
import javafx.stage.Stage;
import sample.Menu;


public class Information {
    @FXML
    private Button back;
```

```
    @FXML
    public void back(ActionEvent event) throws Exception {
        Stage stage = (Stage) back.getScene().getWindow();
        stage.close();
        Menu.startUser();
    }
    public static void start() throws Exception {
        Stage stage = new Stage();
        FXMLLoader fxml = new FXMLLoader(Information.class.getResource("information.fxml"));
        Parent root = (Parent) fxml.load();
        stage.initModality(Modality.APPLICATION_MODAL);
        stage.setTitle("App");
        stage.setScene(new Scene(root,600,600));
        stage.show();
    }
}
```

Листинг А.11 – код класса Training – класс-контроллер для training.fxml.

```
package user;

import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.stage.Modality;
import javafx.stage.Stage;
import sample.Menu;

public class Training {
    @FXML
    private Button back;
    @FXML
    public void back(ActionEvent event) throws Exception {
        Stage stage = (Stage) back.getScene().getWindow();
        stage.close();
        Menu.startUser();
    }
    public static void start1() throws Exception {
        Stage stage = new Stage();
        FXMLLoader fxml = new FXMLLoader(Menu.class.getResource("../user/training1.fxml"));
        Parent root = (Parent) fxml.load();
        stage.initModality(Modality.APPLICATION_MODAL);
        stage.setTitle("App");
        stage.setScene(new Scene(root,600,600));
        stage.show();
    }
}
```

Приложение Б. Данные из базы данных

login	id_illness	id_diet	id_complex
111-111-111-11	6	1	1
123-123-223-22	1	1	1
232-222-324-12	1	1	1
235-333-555-34	1	1	1
888-888-888-88	1	1	1
777-777-777-77	6	1	1
754-456-456-45	1	1	1
555-666-666-34	1	1	1
555-555-555-55	1	1	1
554-455-453-23	1	1	1
324-566-675-56	7	2	2
333-333-333-33	2	2	2
423-234-123-43	1	1	1
432-242-234-23	1	1	1
444-442-242-23	3	2	2
231-123-234-34	4	7	2
000-000-000-00	2	2	2
444-444-444-44	5	1	1

(18 строк)

Рисунок Б.1 – Таблица «patients_data»

login	chss	steps	pressure	sugar	id_indicator
000-000-000-00	67	5800	120/80	4,0-5,0	1
000-000-000-00	68	3400	120/80	4,0-4,6	2
000-000-000-00	70	12000	120/80	5,0-5,4	3

(3 строки)

Рисунок Б.2 – Таблица «indicators»

id_illness	illness_name
1	Нет заболеваний
2	Гастрит
3	Пневмония
4	Крапивница
5	ОРВИ
6	Аллергия
7	Бронхиальная астма
8	Корь
9	Отит
10	Плоскостопие
11	Сахарный диабет
12	Конъюктивит

(12 строк)

Рисунок Б.3 – Таблица «illness»

id_complex	complex_name
1	Не назначено
2	Утрення зарядка
3	ЛФК для опорно-двигательного аппарата
4	ЛФК при ОРВИ
5	Коррекция осанки

(5 строк)

Рисунок Б.4 – Таблица «complex»

id_diet	diet_name
1	Не назначено
2	Стол №1
3	Стол №2
4	Стол №3
5	Стол №4
6	Стол №5
7	Стол №6

(7 строк)

Рисунок Б.5 – Таблица «diet»

role_id	role
1	Пациент
2	Врач
3	Администратор

(3 строки)

Рисунок Б.6 – Таблица «roles»

login	password	surename	name	midname	role_id
111-111-111-11	111111q	Иванов	Иван	Иванович	1
123-123-223-22	123123t	Иванов	Иван	Иванович	1
232-222-324-12	er45er	Галыгин	Лука	Епифанович	1
235-333-555-34	rfver54	Блатов	Валерьян	Проклович	1
admin	admin	Администратор	Администратор	Администратор	3
troynov	23nkn2311	Троянов	Константин	Геннадьевич	2
synov	qaaq34	Сьянов	Тимур	Мечиславович	2
sapaleva	evsw34frdw4	Сапалёва	Роза	Федотовна	2
ribnikov	111111a	Рыбников	Сергей	Михайлович	2
puskarev	bfd436hy	Пушкарёв	Эммануил	Пахомович	2
martynov	1234qwerty	Мартынов	Владислав	Юрьевич	2
lubimova	dwe83nm	Любимова	Лариса	Сергеевна	2
levanko	23fd23	Леванько	Ирина	Михайловна	2
korotkov	lmjkocsd45	Коротков	Платон	Алексеевич	2
doc	doc	Пивоваров	Даниил	Ильич	2
dobronravov	fcsdc43	Добронравов	Харитон	Андреевич	2
burda	cvb4bnv	Бурда	Анисья	Александровна	2
888-888-888-88	666666q	Васильев	Игорь	Васильевич	1
777-777-777-77	666666q	Каменских	Светлана	Сергеевна	1
754-456-456-45	ve2vcQ22	Рогова	Изольда	Дмитриевна	1
555-666-666-34	9jbg56b	Кабаидзе	Лев	Родионович	1
555-555-555-55	1705qwe	Каменских	Максим	Сергеевич	1
554-455-453-23	vfervve03	Волкова	Веселина	Гордеевна	1
444-444-444-44	143143k	Афанасьев	Кирилл	Максимович	1
324-566-675-56	fvba34	Ростова	Ариадна	Михеевна	1
333-333-333-33	333333r	Можайкина	Лариса	Витальевна	1
423-234-123-43	vwa43tvwe	Костина	Глория	Максовна	1
432-242-234-23	xcv34343n	Сазонова	Юлия	Геннадиевна	1
444-442-242-23	Rt4356sc	Жданов	Гордий	Матвеевич	1
231-123-234-34	4cev67mve	Минеева	Эльвира	Георгиевна	1
romanov	wr43dd	Романов	Влад	Геннадьевич	2
000-000-000-00	user00	Пользователь	Пользователь	Пользователь	1

(32 строки)

Рисунок Б.7 – Таблица «user_info»