

**МИНОБРНАУКИ РОССИИ  
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ  
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ  
«ЛЭТИ» ИМ. В.И.УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра ВТ**

**КУРСОВОЙ ПРОЕКТ  
по дисциплине «Объектно-ориентированное  
программирование»  
Тема: Разработка программного комплекса на языке Java**

Студент гр. 8307

\_\_\_\_\_

Репин С.А.

Преподаватель

\_\_\_\_\_

Гречухин М.Н.

Санкт-Петербург  
2020

## ЗАДАНИЕ НА КУРСОВОЙ ПРОЕКТ

Студент Репин С.А.  
Группа 8307

Тема проекта: Разработка программного комплекса на языке Java

Исходные данные:

Разработать ПК для администратора аптеки. В ПК должны храниться сведения о болезнях и лекарствах. Администратор аптеки может добавлять, изменять и удалять эти сведения. Ему может потребоваться следующая информация:

- какие лекарства применяются для лечения указанной болезни;
- имеется ли лекарство в аптеке и в каком количестве;
- какие лекарства и в каком количестве проданы за указанный период времени;
- на какую сумму проданы лекарства за месяц.

Содержание пояснительной записки:

«Техническое задание», «Описание процесса проектирования ПК», «Руководство оператора», «Исходные тексты ПК», «Заключение», «Список использованных источников».

Предполагаемый объем пояснительной записки:  
Не менее 20 страниц.

Дата выдачи задания:	21.09.2020
Дата сдачи курсового проекта:	27.12.2020
Дата защиты курсового проекта:	.12.2020

Студент гр. 8307

\_\_\_\_\_

Репин С.А.

Преподаватель

\_\_\_\_\_

Гречухин М.Н.

## **АННОТАЦИЯ**

Целью курсового проектирования является закрепление и углубление теоретических знаний, приобретение практических навыков по проектированию и разработке программного обеспечения на объектно-ориентированном языке.

В задачи курсового проектирования входят:

- изучение особенностей конкретной предметной области, относящейся к заданию на курсовой проект, и разработка технического задания на программный комплекс (ПК);
- объектно-ориентированное проектирование ПК с использованием языка UML;
- разработка ПК на объектно-ориентированном языке;
- написание программной документации

## **SUMMARY**

The purpose of course design is to consolidate and deepen theoretical knowledge, acquisition of practical skills on the design and development of software for object oriented language.

The tasks of course design include:

- studying the features of a specific subject area, related to the assignment for the course project, and development technical specifications for the software package (PC);
- object-oriented PC design using language UML;
- PC development in object-oriented language;
- writing programming documentation

## СОДЕРЖАНИЕ

1. Техническое задание	5
1.1. Введение . . . . .	5
1.2. Основания для разработки . . . . .	5
1.3. Назначение разработки . . . . .	5
1.4. Требования к программе . . . . .	5
1.4.1. Требования к функциональным характеристикам . . .	5
1.4.2. Требования к надежности . . . . .	6
1.4.3. Условия эксплуатации . . . . .	7
1.4.4. Требования к составу и параметрам технических средств	7
1.4.5. Требования к информационной и программной совме- стимости . . . . .	7
1.5. Требования к программной документации . . . . .	8
1.6. Стадии и этапы разработки . . . . .	8
1.7. Порядок контроля и приемки . . . . .	9
2. Описание процесса проектирования ПК	10
2.1. Описание вариантов использования ПК . . . . .	10
2.2. Создание прототипа интерфейса пользователя . . . . .	13
2.3. Разработка объектной модели . . . . .	13
2.4. Построение диаграммы программных классов . . . . .	15
2.5. Описание поведения ПК . . . . .	16
2.6. Построение диаграммы действий . . . . .	16
3. Руководство оператора	20
3.1. Назначение программы . . . . .	20
3.2. Условия выполнения программы . . . . .	20
3.3. Описание задачи . . . . .	21
3.4. Входные и выходные данные . . . . .	21
3.5. Выполнение программы . . . . .	21
3.5.1. Запуск . . . . .	21
3.5.2. Выполнение программы . . . . .	22
3.5.3. Завершение программы . . . . .	27
3.6. Проверка программы . . . . .	27
3.7. Сообщения оператору . . . . .	28
4. Исходные тексты ПК	31
Заключение	32
Список использованных источников	33

# **1. ТЕХНИЧЕСКОЕ ЗАДАНИЕ**

## **1.1. Введение**

Наименование программы — «Pharmacy Console».

Программный комплекс предназначен для цифровизации работы аптеки. Он облегчает администратору (кассиру-консультанту) аптеки выполнение своих обязанностей, повышая его эффективность.

## **1.2. Основания для разработки**

Основанием для разработки является выполнение курсового проектирования как результата прохождения курса «Объектно-ориентированное программирование». Тема работы — «Разработка ПК для администратора аптеки», шифр темы — 16.

## **1.3. Назначение разработки**

Программа предоставляет возможности для учета товаров аптеки, ассистирования при выборе лекарств для покупки, ведения бухгалтерского учета заказов. Основным пользователем выступает администратор (кассир-консультант) аптеки, эксплуатация программы предполагает использование штатных ЭВМ организации.

## **1.4. Требования к программе**

### **1.4.1. Требования к функциональным характеристикам**

Программа представляет собой графическое приложение, хранящее сведения о болезнях и лекарствах. Пользователь программы, взаимодействуя элементами управления, может добавлять, изменять и удалять эти сведения.

Кроме того, пользователь имеет возможность легко получать следующую информацию:

- какие лекарства применяются для лечения указанного заболевания;
- имеется ли лекарство в аптеке и в каком количестве;
- какие лекарства и в каком количестве проданы за указанный период;
- на какую сумму проданы лекарства за месяц.

Входными данными выступают данные, вводимые пользователем с клавиатуры, а также с помощью элементов управления. Выходные данные — таблицы, списки, всплывающие окна и изменения базы данных приложения.

Данные должны надежно храниться в СУБД. Должна производиться валидация входных данных (имена должны быть непустыми, количественные характеристики должны быть натуральными числами и т.д.).

Программа должна работать безотказно и не допускать падений.

С учетом развития электронных устройств особых требований ко времени отклика программы не ставится, она лишь не должна иметь заметных для обычного пользователя лагов интерфейса. Время запуска приложения — не более 5 секунд.

#### **1.4.2. Требования к надежности**

Надежное (устойчивое) функционирование программы обеспечивается выполнением пользователем следующего набора мероприятий:

- организацией бесперебойного питания технических средств;
- соблюдением условий эксплуатации;
- выполнением требований и рекомендаций государственных актов в области использования и обслуживания программных комплексов;
- своевременным обновлением программного (в том числе, данной программы) и аппаратного обеспечения;

### **1.4.3. Условия эксплуатации**

Программа запускается на компьютерах аптеки для каждого администратора. Перед запуском программы необходимо запустить сервер СУБД. Должна обеспечиваться надежная и бесперебойная работа сервера.

Непосредственные пользователи программы должны владеть основами компьютерной грамотности и знать английский язык на уровне чтения. Для установки, первичной настройки и сервисного обслуживания может потребоваться системный администратор, обладающий навыками установки программ в текущую операционную систему и развертывания СУБД.

### **1.4.4. Требования к составу и параметрам технических средств**

Минимальный состав аппаратных средств:

- процессор архитектуры x86-64 с тактовой частотой, 2.5 ГГц;
- оперативная память объемом, 2 Гб;
- жесткий диск объемом 64 Гб;
- графический адаптер;
- монитор с разрешением экрана 1024 x 768 пикселей.

### **1.4.5. Требования к информационной и программной совместимости**

- Программа работает под операционными системами Windows и GNU/Linux, удовлетворяющими требованиям JDK 14.
- Требуются установленные программы: JDK 14 и PostgreSQL 12.

Исходный код должен быть написан на языке Java 14, собираться средствами Maven, вести журналирование своей работы.

## **1.5. Требования к программной документации**

Предварительный состав программной документации:

- техническое задание;
- документация к исходному коду в формате HTML;
- руководство оператора;
- пояснительная записка.

## **1.6. Стадии и этапы разработки**

Выделяются следующие стадии и их этапы:

1. техническое задание:

- разработка ТЗ;
- согласование ТЗ;
- утверждение ТЗ;

2. технический проект:

- разработка программы;
- разработка документации;
- тестирование программы и документации;

3. внедрение:

- подготовка к передаче;
- передача программы.

На этапе разработки технического задания можно выделить такие работы:

- постановка задачи;



- определение и уточнение требований к техническим средствам;
- определение требований к программе;
- определение стадий, этапов и сроков разработки программы и документации на нее;
- согласование и утверждение технического задания.

На этапа разработки программного продукта должны быть выполнены работы по написанию кода программы, модульных тестов к ней и ее отладка.

На этапе испытаний программы должны быть выполнены такие работы, как:

- разработка, согласование и утверждение порядка и методики испытаний;
- проведение приемо-сдаточных испытаний;
- корректировка программы и программной документации по результатам испытаний.

Срок выполнения работы: 21.09.2020-29.12.2020.

Исполнитель: Репин Степан Александрович.

### **1.7. Порядок контроля и приемки**

Должно быть проведено ручное тестирование программного комплекса при различных условиях на соответствие настоящему техническому заданию, проверка исходного кода на соответствие документации и применения модульных тестов как спецификации исходного кода.

## 2. ОПИСАНИЕ ПРОЦЕССА ПРОЕКТИРОВАНИЯ ПК

### 2.1. Описание вариантов использования ПК

Проектирование программного комплекса начинается с прояснения функциональных требований из технического задания. В итоге появляется развернутый и детализированный список требований, характеризующих процессы в текущей предметной области. Распространенными способами такого описания являются UML-диаграммы прецедентов (use case) и пользовательские истории (user story).

UML-диаграмма прецедентов состоит из набора сценариев взаимодействия программы с внешней средой в соответствии с требованиями. Прецеденты описываются от лица некоторых внешних сущностей, называемых актерами. Актеры взаимодействуют с системой, меняют ее для достижения своих целей. Между актерами и прецедентами устанавливаются направленные соединения — связи коммуникации, которые также снабжаются информацией об участвующих данных и кратности соединения.

Прецеденты могут образовывать структуру с помощью вида связи, который называется связью использования. Таким образом, появляются группы прецедентов, исполняемых вместе. Для демонстрации родственных отношений между прецедентами используется связь расширения.

Другим похожим способом описания вариантов использования приложений являются пользовательские истории, получившие широкое распространение как составные части многих гибких методологий разработки (SCRUM, экстремальное программирование и другие). Основная суть заключается в выделении сущности пользователя и написании от его лица особых фраз по специальному шаблону, которые показывают сценарии работы системы и их применение в контексте бизнес задач. Воспользуемся следующим шаблоном:

*Я как <роль> хочу <возможность>, чтобы <цель>.*

Главным отличием от use case является то, что пользовательские истории намного ближе к обычному клиенту, не обладающему профессиональными навыками разработки ПО, и формулируются на повседневном языке, отчего, однако, не оказываются столь подробными как UML-диаграммы прецедентов.

В Pharmacy Console присутствуют такие акторы, как администратор аптеки, бухгалтер аптеки и база данных. Напишем сначала пользовательские истории для администратора и бухгалтера, а затем построим диаграмму прецедентов.

Пользовательские истории:

- я как администратор аптеки хочу иметь возможность добавлять сведения о лекарствах, чтобы вести их учет;
- я как администратор аптеки хочу иметь возможность добавлять к лекарствам список болезней, которые те лечат, чтобы консультировать клиентов аптеки;
- я как администратор аптеки хочу иметь возможность удалять лекарства и болезни (если нет связи с лекарствами), чтобы легко обновлять ассортимент аптеки;
- я как администратор аптеки хочу иметь возможность редактировать информацию о лекарствах и болезнях, чтобы использовать это при обновлении ассортимента и новых поставках;
- я как администратор аптеки хочу иметь возможность видеть списки всех лекарств и болезней в виде таблицы, чтобы легко находить о них информацию;
- я как администратор аптеки хочу иметь возможность фильтровать отображаемый список лекарств по болезням, которые те лечат, чтобы легко консультировать клиентов;
- я как администратор аптеки хочу иметь возможность создавать новый заказы, указывая число покупаемых товаров, с автоматическим подсчетом суммы заказов;
- я как администратор аптеки хочу видеть список всех заказов (с возможностью указания периода времени) и общую сумму этих заказов.
- я как бухгалтер аптеки, хочу иметь возможность получать отчет по совершенным заказам в определенный период в формате PDF, чтобы совершать бухгалтерский учет;

Диаграмма прецедентов изображена на рис. 2.1.

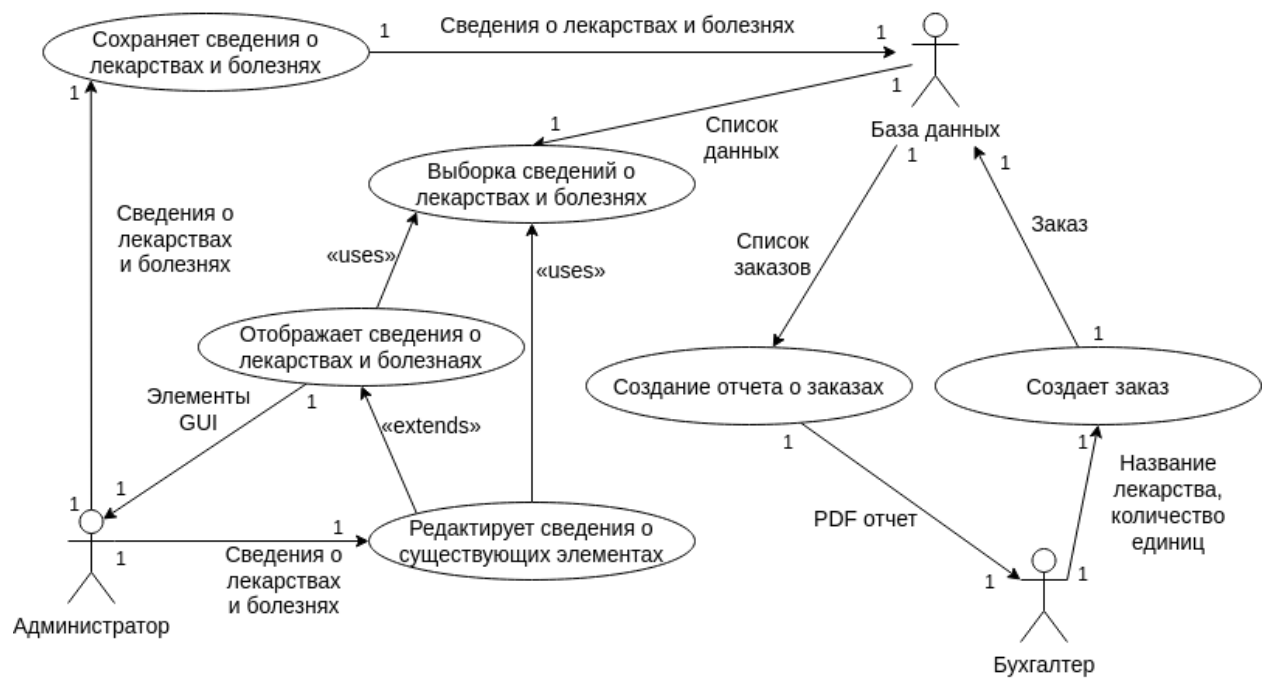


Рис. 2.1 Диаграмма прецедентов программы

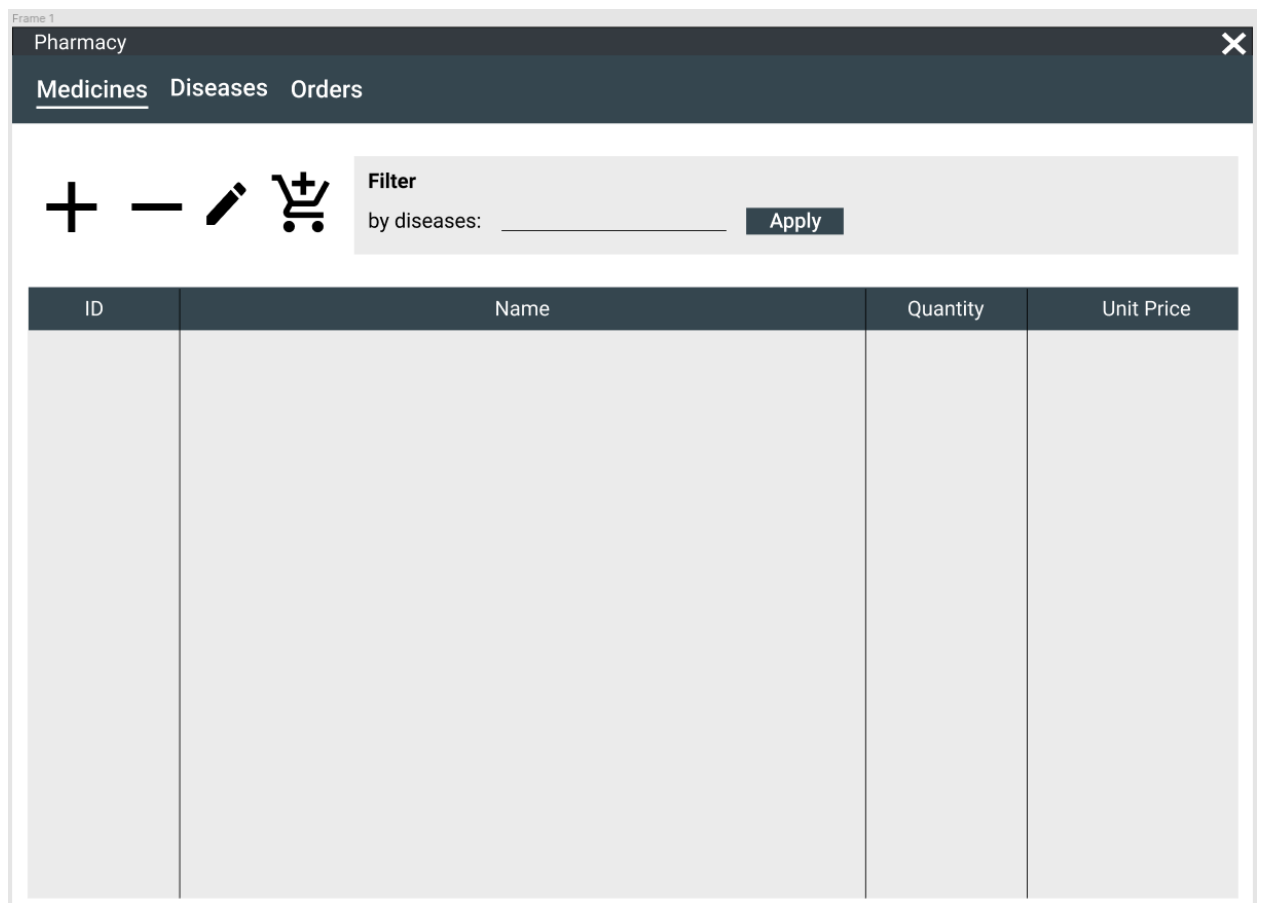


Рис. 2.2 Прототип дизайна приложения

## **2.2. Создание прототипа интерфейса пользователя**

На основе проработанные прецеденты теперь можно описать процессы в терминах пользовательских интерфейсов, детализировав тем самым реализацию. Прецеденты удобно разбить на отдельные экранные формы и определить содержимое графических представлений, способы их взаимодействия. То есть, для каждой формы можно привести набор графических элементов, действия пользователя и отклики системы. Полученные перечни удобно свести в таблицы экранных форм (табл. 1.1, 1.2, 1.3). Здесь не приводятся таблицы для окон «Добавление/редактирование заболеваний», «Создание заказа» и вкладок «Болезни», «Заказы», так как они тривиальны и выполняются аналогично.

Кроме того, общий дизайн всей программы был вначале прототипирован с помощью сервиса Figma (рис. 2.2).

## **2.3. Разработка объектной модели**

На этом этапе можно перейти к проектированию объектной модели, отображающую основные понятия предметной области и их взаимосвязь как совокупность типов объектов (далее классов, но следует отличать их от классов в языках программирования). Модель строится на основе анализа предметной области задачи.

В UML объектная модель представляется в виде диаграммы классов (class diagram), которая состоит из собственно классов, содержащих атрибуты (свойства сущности) и операции (поведения сущностей). Между собой классы выстраивают ассоциации, показывающие семантический смысл отношения двух сущностей. Ассоциации могут иметь кратность. Особый интерес представляют ассоциации, которые отражают структурные отношения («содержит», «включает», «хранит», и т.д.).

С точки зрения предметной области легко видеть, что Pharmacy Console имеет сущности «Лекарство», «Заболевание» и «Заказ лекарства». На рис. 2.3 отображены данные классы с атрибутами и операциями. Детальное описание операций в данном случае не требуется.

<b>Элементы управления</b>	<b>Действия пользователя</b>	<b>Отклик системы</b>
Таблица с полями: ID, название, количество, цена ед.	Нажать кнопку «Добавить».	Открыть окно «Добавление/изменение лекарства».
Кнопки «Добавить», «Удалить», «Изменить», «Создать заказ».  Переключатель (флаг) фильтра.  Поле ввода текста имени заболевания.	Нажать кнопку «Удалить».	Удалить выбранное в таблицу лекарство.
	Нажать кнопку «Изменить».	Открыть окно «Добавление/изменение лекарства» (с заполненными полями).
	Нажать кнопку «Создать заказ».	Открыть окно «Создание заказа».
	Переключить флаг.	Применить или отключить фильтр на основе введенного текста.

Таблица 1.1 Таблица экранной формы для вкладки «Лекарства»

<b>Элементы управления</b>	<b>Действия пользователя</b>	<b>Отклик системы</b>
Таблица с полями: ID, название лекарства, количество, общая сумма, дата заказа.	Нажать на кнопку «Создать отчет».	Открыть диалог выбора файла для сохранения отчета и создать отчет.
Кнопка «Создать отчет».	Переключить флаг.	Применить или отключить фильтр на основе введенных дат.
Переключатель (флаг) фильтра.		
Два поля ввода даты.		

Таблица 1.2 Таблица экранной формы для вкладки «Заказы»

Элементы управления	Действия пользователя	Отклик системы
Два текстовых поля для имени и болезней	Нажать на кнопку «ОК»	Добавление/ редактирование лекарства
Два текстовых поля для цены и количества	Нажать на кнопку «Отмена»	Отмена

Таблица 1.2 Таблица экранной формы  
«Добавление/редактирование лекарства»



Рис. 2.3 Диаграмма сущностей

## 2.4. Построение диаграммы программных классов

Получив объектную модель программы, можно перейти к спецификации программной реализации этой модели на основе программных классов и интерфейсов. Диаграмма классов является расширением диаграммы сущностей и добавляет в нее новые смыслы, важные при реализации структуры на объектно-ориентированном языке программирования: новые атрибуты, операции, связи, конкретизируется тип связи (ассоциация, агрегация, наследование).

На рис. 2.4 изображена UML-диаграмма классов. Заметим, что тут появились некоторые новые классы: DAO, сервисы и контроллеры. На данном рисунке также приведен лишь фрагмент полной диаграммы классов из-за большой сложности последней.

## **2.5. Описание поведения ПК**

Перед тем, как переходить к непосредственной реализации кода, необходимо еще определить поведение программы без конкретизации механизмов их реализации. В этом качестве используется диаграмма последовательностей (sequence diagram).

Диаграмма показывает возникающие в ходе выполнения сценария прецедента упорядоченные события. Для построения диаграммы воспользуемся таким алгоритмом:

1. Идентифицировать участников начальной стадии прецедента. Изобразить их в одну линию на диаграмме.
2. Выбрать операции, используемые в сценарии.
3. Изобразить операции на диаграмме, рисуя их от объекта к объекту.

Операции могут быть, в том числе, условными и циклическими. На диаграммах последовательностей также отображается время жизни объектов.

Полная диаграмма последовательностей для Pharmacy Console занимает очень много места, поэтому приведем лишь пример диаграммы последовательностей для «Удаление лекарства» и «Создание отчета» (см. рис. 2.5 и 2.6).

## **2.6. Построение диаграммы действий**

Детали реализации сложных действия удобно представлять по средствам диаграммы действий (activity diagram), представляющую собой направленный граф, вершинами которого являются действия, а ребрами — переходы между действиями. Видно сходство с блок-схемами алгоритмов, но отличительная черта здесь — это возможность представления параллельных алгоритмов. Для действий часто указывают объекты, с которыми они связаны.



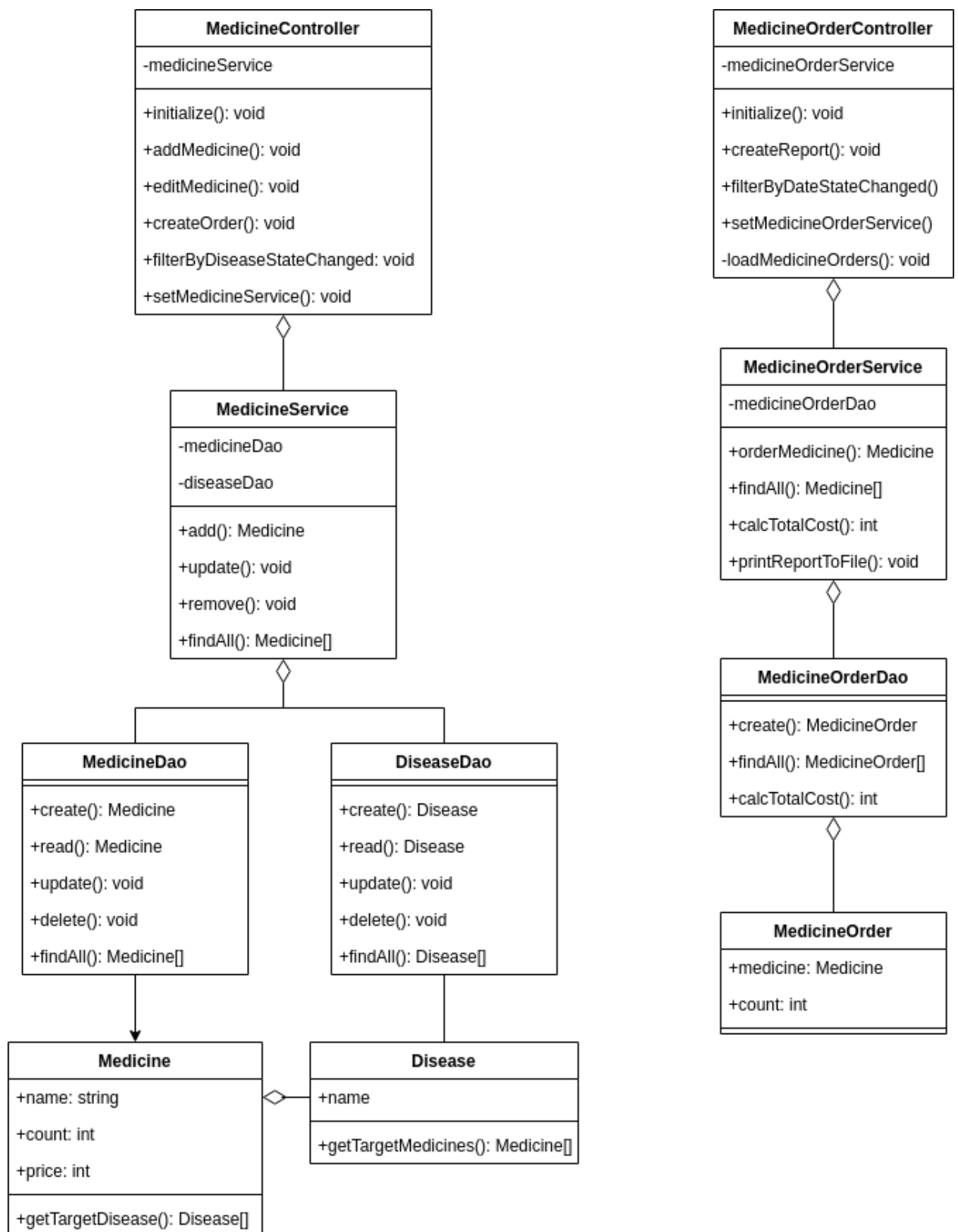


Рис. 2.4 Диаграмма классов

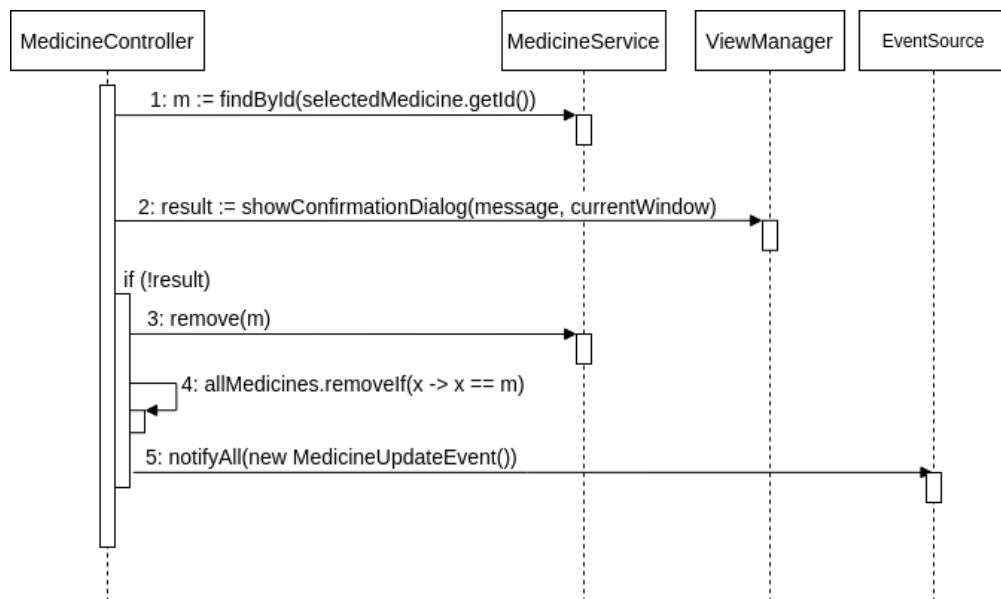


Рис. 2.5 Диаграмма последовательностей для метода *MedicineController.removeMedicineAction()*

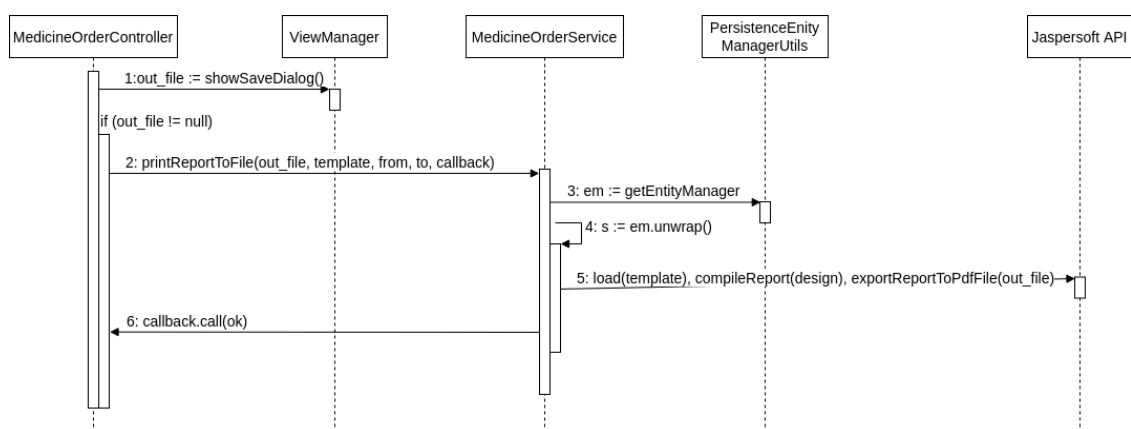


Рис. 2.6 Диаграмма последовательностей для метода *MedicineOrderController.createReport()*

На рисунке 2.7 изображена диаграмма действий для операций «Создание лекарства».

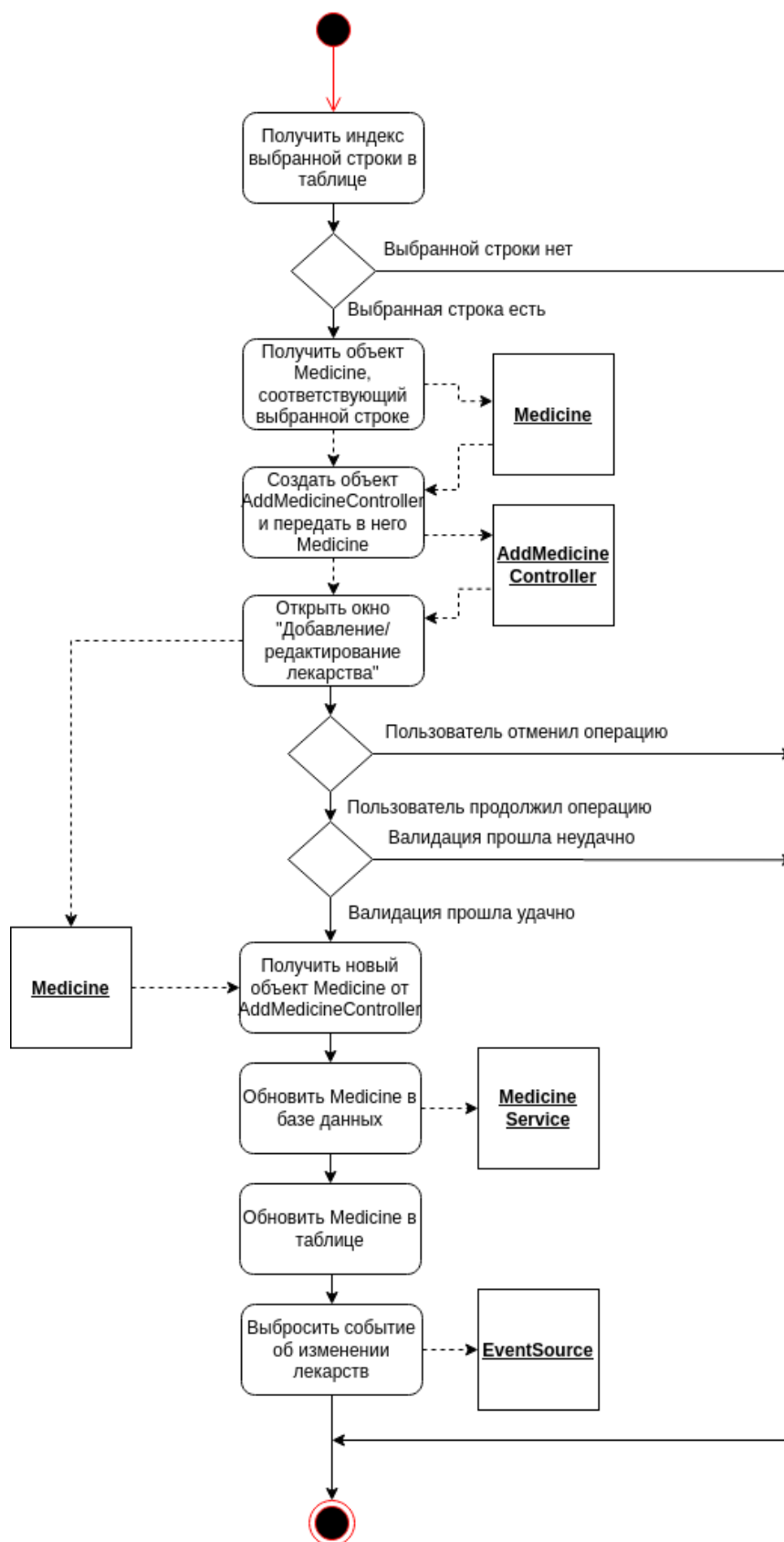


Рис. 2.7 Диаграмма действий

### **3. РУКОВОДСТВО ОПЕРАТОРА**

#### **3.1. Назначение программы**

Программный комплекс Pharmacy Console предназначен для повышения эффективности и устойчивости работы аптеки с помощью создания единого интерфейса для выполнения основных задач, возникающих перед администратором аптеки.

#### **3.2. Условия выполнения программы**

Минимальный состав аппаратных средств:

- процессор архитектуры x86-64 с тактовой частотой 2.5 ГГц;
- оперативная память объемом 2 Гб;
- жесткий диск объемом 64 Гб;
- графический адаптер;
- монитор с разрешением экрана 1024x768 пикселей.

Необходимый набор программных средств:

- ОС Windows (7, 8.1, 10) или GNU/Linux (ядро старше 3.2);
- JDK 14;
- PostgreSQL 12 сервер.

Конечный пользователь программы (оператор) должен обладать практическими навыками работы с графическим пользовательским интерфейсом операционной системы, должен быть аттестован минимум на II квалификационную группу по электробезопасности, иметь квалификацию «Пользователь ЭВМ», владеть английским языком на уровне чтения простых фраз.

### **3.3. Описание задачи**

Программный комплекс хранит сведения о болезнях и лекарствах. Оператор может добавлять, изменять и удалять эти сведения. Ему доступны отдельно следующие функции:

- получение списка лекарств для лечения указанной болезни;
- проверка наличия лекарства в аптеке;
- получения количества доступный единиц лекарства;
- получение информации о количестве и общей сумме проданных лекарств за указанный период времени.

Для решения задачи используется разработанное приложение, предоставляющее возможность выполнять указанные выше операции над сведениями о болезнях, лекарствах и заказах лекарств, которые располагаются в базе данных, через взаимодействие с графическим интерфейсом.

### **3.4. Входные и выходные данные**

Входными данными программы является набор действий оператора с элементами графического интерфейса (нажатия на кнопки, ввод текста в текстовые поля, включение/выключение флагов).

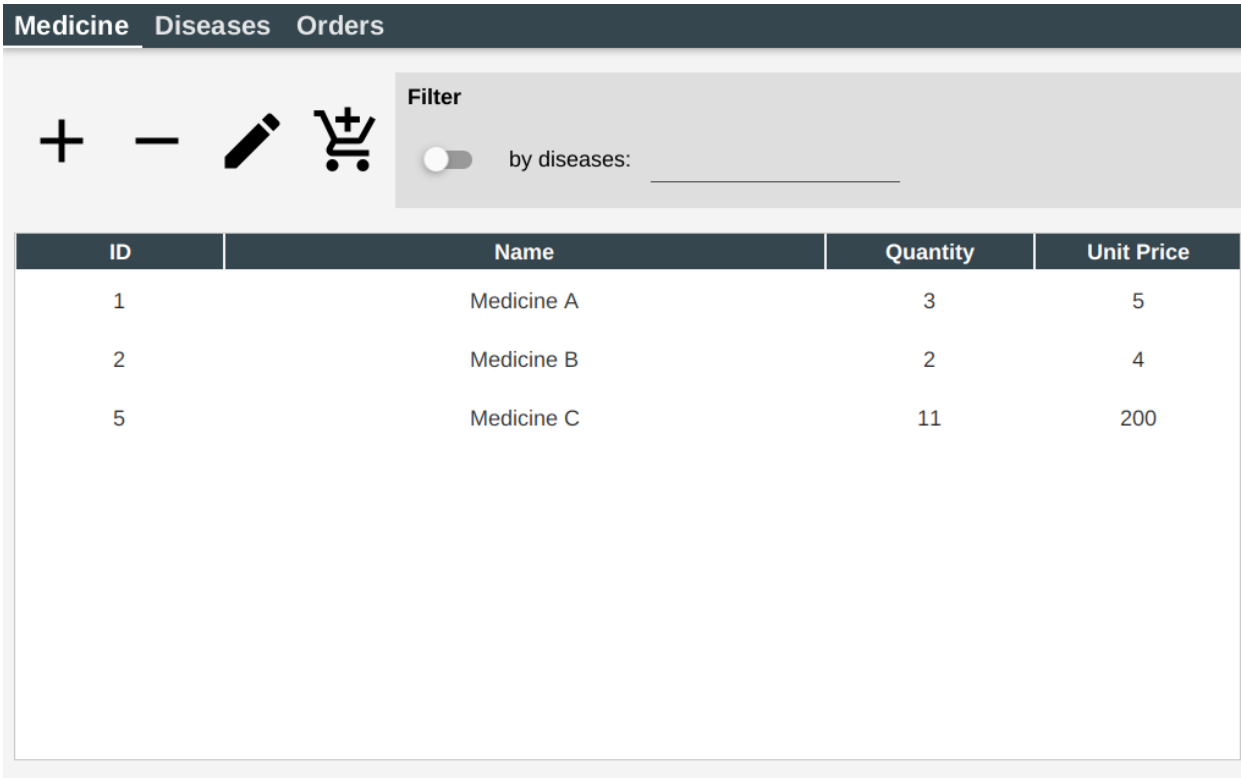
Выходные данные — изменения базы данных приложения, с последующим обновлением отображаемой интерфейсом информации (добавление или удаление строк таблицы, показ диалоговых окон). При печати отчета выходными данными также выступает результирующий PDF-файл отчета.

### **3.5. Выполнение программы**

#### **3.5.1. Запуск**

Запуск программы осуществляется с помощью выполнения JAR-файл. Перед его запуском необходимо удостовериться в том, что сервер БД запущен.

При первом запуске программы появится изображенное ниже окно.



ID	Name	Quantity	Unit Price
1	Medicine A	3	5
2	Medicine B	2	4
5	Medicine C	11	200

Рис. 3.1 Пример главного окна с открытой вкладкой «Лекарства»

### 3.5.2. Выполнение программы

Рассмотрим основные функции программы и их исполнение с помощью элементов управления.

- Переход к списку лекарств

Чтобы перейти к списку лекарств (и в дальнейшем работать с ним) необходимо нажать на кнопку «Medicine», что переключит текущую вкладку на вкладку «Лекарства».

- Добавление нового лекарства

Нажмите кнопку с изображением «+», откроется окно «Добавление/редактирование лекарства», введите в него необходимые сведения (имя не должно быть пустой строкой, болезни можно перечислять через «;») и нажмите кнопку «ОК». Для отмены нажмите кнопку «Cancel». Новое лекарство будет добавлено в таблицу.

– Удаление лекарства

Выделите в таблице строчку, соответствующую необходимому лекарству. Нажмите кнопку с изображением «-». Подтвердите свое действие. Лекарство исчезнет из таблицы.

– Редактирование лекарства

Выделите в таблице строчку, соответствующую необходимому лекарству. Нажмите кнопку с изображением карандаша, откроется окно «Добавление/редактирование лекарства», измените в нем необходимые сведения (на них накладываются такие же требования, как и в функции «Добавление нового лекарства») и нажмите кнопку «ОК». Для отмены нажмите кнопку «Cancel». Сведения о лекарстве будут обновлены в таблице.

– Создание заказа

Выделите в таблице строчку, соответствующую необходимому лекарству. Нажмите кнопку с изображением точки и знаком «+». Откроется окно «Добавление заказа», введите в нем необходимое число единиц заказываемого лекарства. Нажмите кнопку «ОК». Для отмены нажмите кнопку «Cancel». В результате количество доступных единиц лекарства в таблице лекарств уменьшится, а во вкладке «Заказы» в таблице заказов будет добавлен новый заказ, информация об общей сумме заказов обновится.

– Выборка лекарств для лечения указанной болезни

Введите название интересующей болезни в текстовое поле рядом с флагом (переключателем) на панели "Фильтр". Активируйте переключатель. В таблице останутся только те лекарства, которые применяются для лечения указанной болезни. Чтобы вернуть общий список лекарств, деактивируйте переключатель.

– Переход к списку болезней

Чтобы перейти к списку болезней (и в дальнейшем работать с ним) необходимо нажать на кнопку «Disease», что переключит текущую вкладку на вкладку «Болезни».

– Добавление новой болезни

Нажмите кнопку с изображением «+», откроется окно «Добавление/редактирование болезни», введите в него необходимые сведения (имя не должно быть пустой строкой и не должно содержать символ «;») и нажмите кнопку «ОК». Для отмены нажмите кнопку «Cancel». Новая болезнь будет добавлена в таблицу.

– Удаление болезни

Выделите в таблице строчку, соответствующую необходимой болезни. Нажмите кнопку с изображением «-». Подтвердите свое действие. Болезнь исчезнет из таблицы. Указанное действие можно выполнить только если нет лекарств, ссылающихся на данную болезнь.

– Редактирование болезни

Выделите в таблице строчку, соответствующую необходимой болезни. Нажмите кнопку с изображением карандаша, откроется окно «Добавление/редактирование болезни», измените в нем необходимые сведения (на них накладываются такие же требования, как и в функции «Добавление новой болезни») и нажмите кнопку «ОК». Для отмены нажмите кнопку «Cancel». Сведения о болезни будут обновлены в таблице.

– Переход к списку заказов

Чтобы перейти к списку болезней (и в дальнейшем работать с ним) необходимо нажать на кнопку «Medicine Order», что переключит текущую вкладку на вкладку «Заказы».

– Выборка заказов по дате

На панели «Фильтр» выберете поочередно даты начала периода и его конца (нажмите на кнопки с изображением календаря рядом с полем для ввода, откроется всплывающее окно выбора даты). Активируйте переключатель. Список заказов в таблице обновится, как и общая сумма заказов.

– Печать отчета о заказах

Нажмите кнопку с изображением принтера (она единственная на форме). Откроется диалог выбора файла для сохранения. Выберете нужный файл и нажмите «ОК» (или «Cancel», чтобы отменить печать). Отобразится элемент управления Спиннер, показывающий факт выполнения операции (она может продлиться некоторое время). В выбранном файле появится отчет.



Скриншоты выполнения некоторых функций представлены ниже. Работа с остальными функциями происходит аналогично.

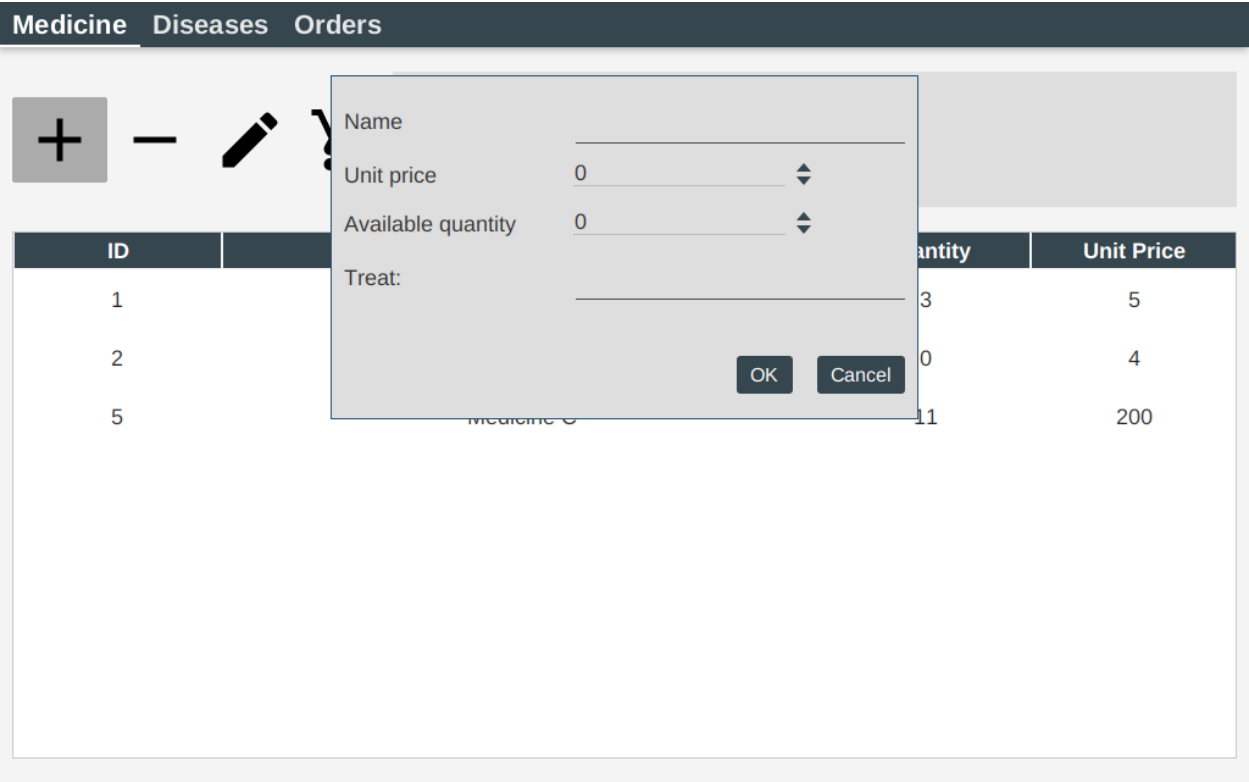


Рис. 3.2 Добавление/редактирование лекарства

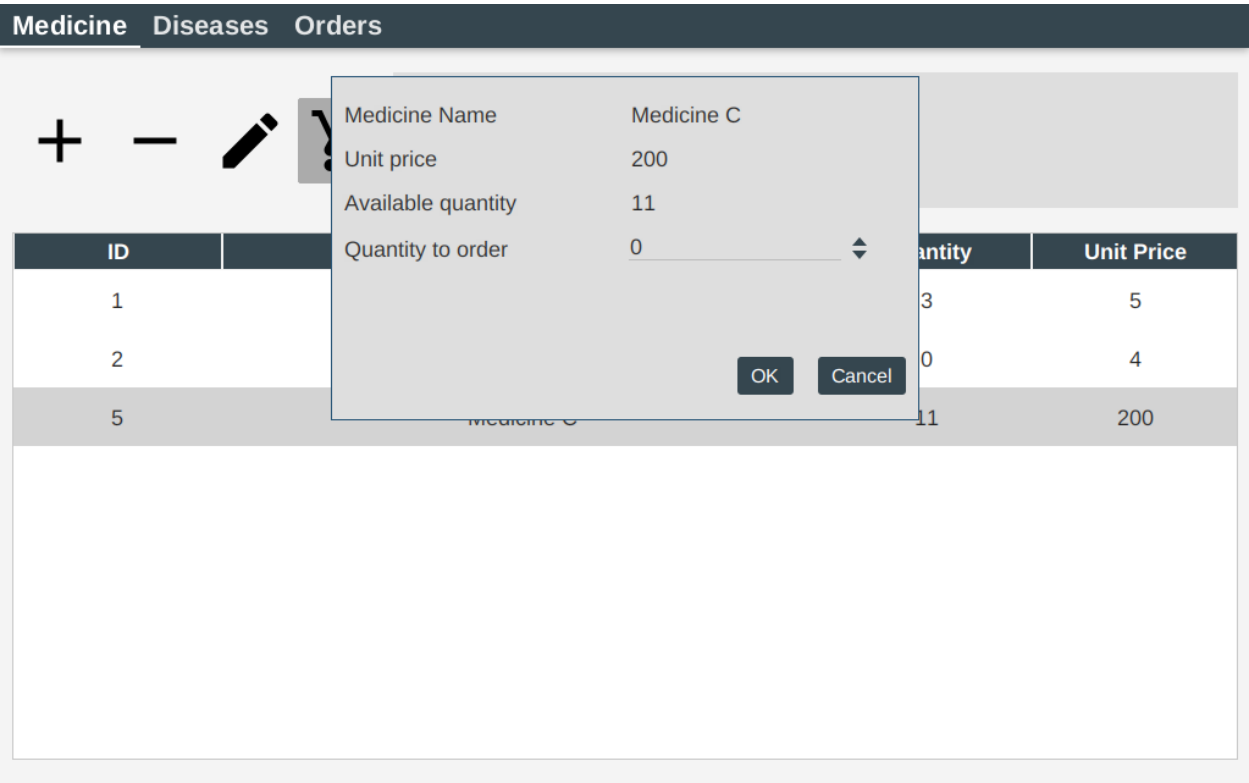


Рис. 3.3 Добавление нового заказа

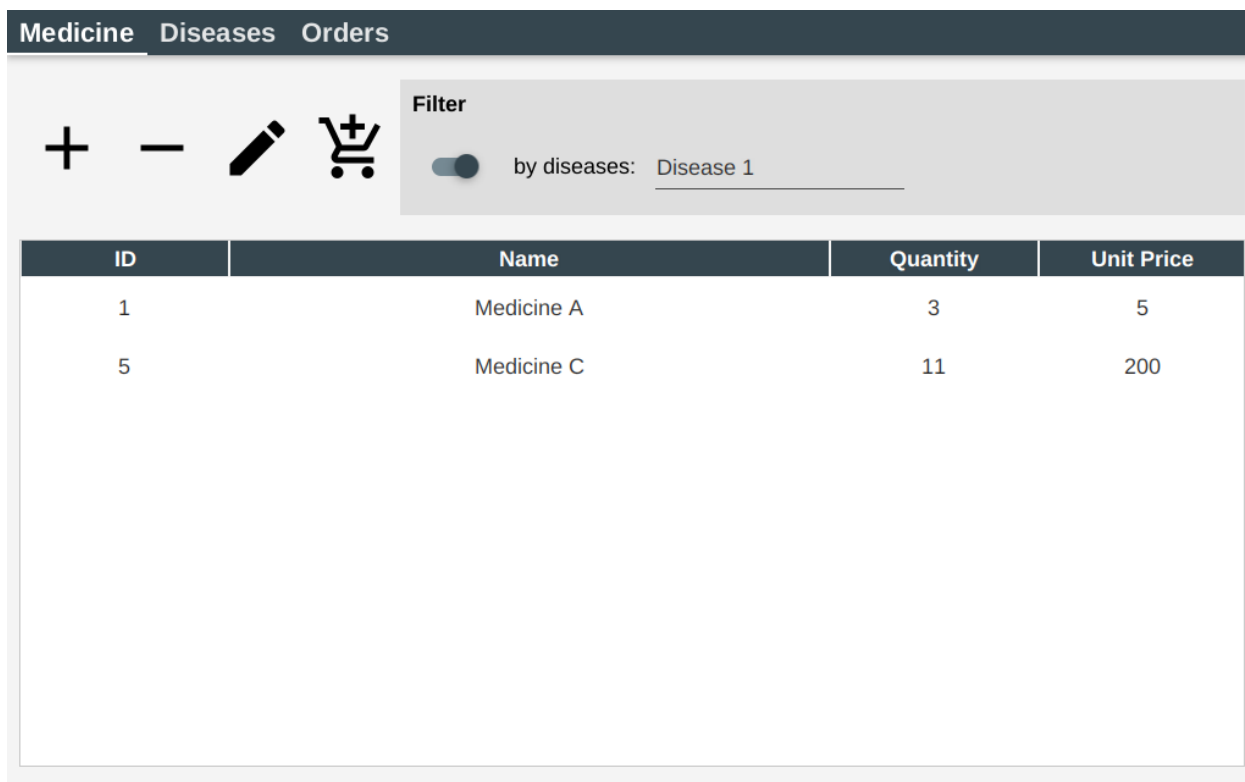


Рис. 3.4 Фильтрация лекарств по болезни

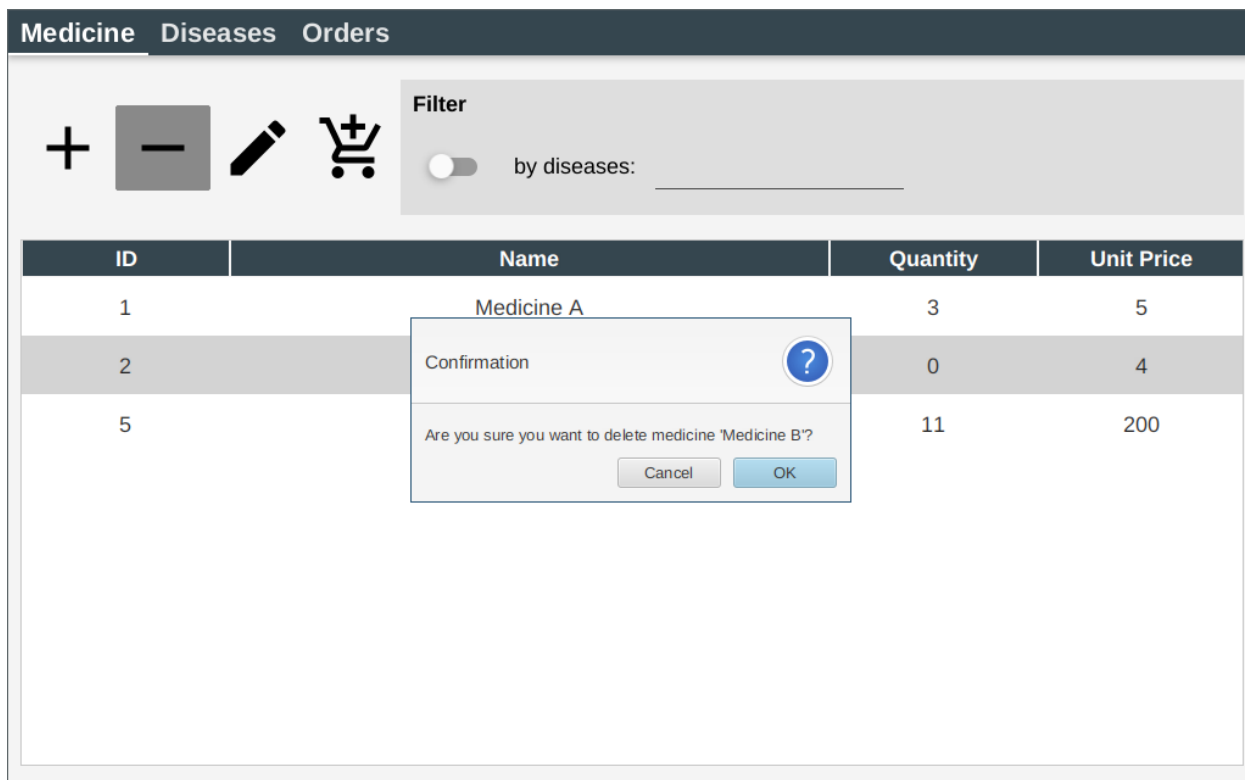


Рис. 3.5 Окно ожидания подтверждения об удалении лекарства

### 3.5.3. Завершение программы

Для завершения программы необходимо закрыть приложение штатными средствами ОС.

### 3.6. Проверка программы

В целях проверки программы следует выполнить следующие контрольные примеры:

1. Добавить лекарство с именем «М», ссылающееся на болезни «D1» и «D2» разделе «Входные и выходные данные» должны быть указаны сведения о входных и выходных д

*Ожидаемое поведение:* В таблице с лекарствами появится новое лекарство, а в таблице с болезнями появятся две новые болезни.

2. Выполнить действия аналогичные 1, но сперва добавить болезни (лекарство «M2», болезни «D3» и «D4»)

*Ожидаемое поведение:* Аналогично 1, но новые лекарства не создаются.

3. Отредактировать численные поля существующего лекарства, изменить название болезни

*Ожидаемое поведение:* Численные поля изменяться в таблице, добавится новая болезнь, старая останется без изменений.

4. Произвести фильтрацию лекарств по имени существующей и несуществующей болезни. Отменить фильтрацию

*Ожидаемое поведение:* При фильтрации по существующей болезни отобразятся только те лекарства, которые ссылаются на указанную болезнь. При фильтрации по несуществующей болезни таблица окажется пустой. При отмене фильтрации таблица вернется в исходное состояние.

5. Отредактировать название болезни, на которую ссылается лекарство

*Ожидаемое поведение:* В таблице с болезнями и в свойствах лекарства изменится имя данной болезни.

6. Создать заказ

*Ожидаемое поведение:* В таблице с заказами появится заказ с указанными параметрами (проверить соответствие названия лекарства, цены за штуку и общего количества единиц в заказе). В таблице лекарств для заказанного лекарства будет уменьшено значение поля доступного количества единиц.

#### 7. Произвести фильтрацию заказов

*Ожидаемое поведение:* в таблицу заказов отобразятся только заказы, совершенные в указанный период (смотреть по столбцу «Order Date»).

#### 8. Удалить лекарство, для которой был создан заказ

*Ожидаемое поведение:* В поле «Medicine» заказа, ссылающегося на удаленное лекарство, таблицы заказов появится надпись «N/A» вместо имени лекарства.

#### 9. Отфильтровать заказы и создать отчет

*Ожидаемое поведение:* в отчете окажутся только заказы за указанный период.

Каждый контрольный пример следует выполнять после перезагрузки программы, а ожидаемое поведение сверять до перезагрузки и сразу же после.

### 3.7. Сообщения оператору

В соответствии с идеологией Unix-way программа не выводит сообщения при удачном совершении действия (предполагается, что оператор понимает, что он делает). В иных случаях, а также в случаях, когда действия оператора меняют содержимое формы, отображаются такие сообщения:

- Окно «Ошибка подключения к БД»

Возникает, если при запуске программы не удастся установить сообщение с сервером БД.

Вероятнее всего, не запущен сервер базы данных. Необходимо запустить сервер и затем заново запустить программу.

- Окно «Возникла ошибка»

Отображается диалоговое окно ошибки, с описанием ошибки (также, если возможно приводится стек вызовов).

Если программа остается работоспособной (не завершается после нажатия на «ОК»), то от оператора требуется только повторить при необходимости выполненные последний раз действия с программой. В ином случае, потребуются повторный запуск программы.

– Окно «Подтверждение удаления»

Перед удалением лекарства или болезни у пользователя потребуют подтверждения указанного действия (нажать «ОК», для совершения удаления, иначе – «Cancel»).

– Окно «Некорректные вводимые данные»

При добавлении или редактировании лекарства (болезни) производится валидация вводимых данных. В случае нарушения соглашения о формате вводимых данных оператору будет отображено соответствующее сообщение. От него требуется перепроверить введенные данные.

Примеры сообщений приведены на рисунках ниже.

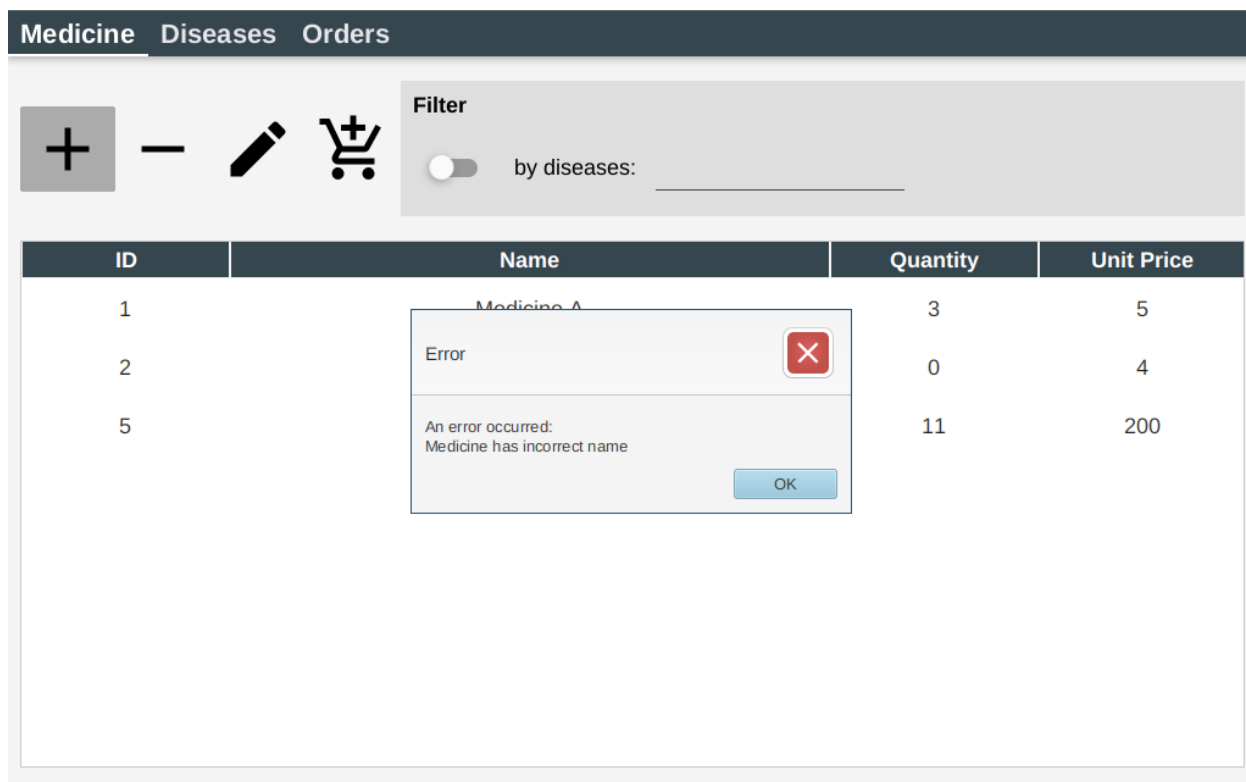


Рис. 3.6 Сообщение об ошибке при попытке добавления лекарства с пустым именем

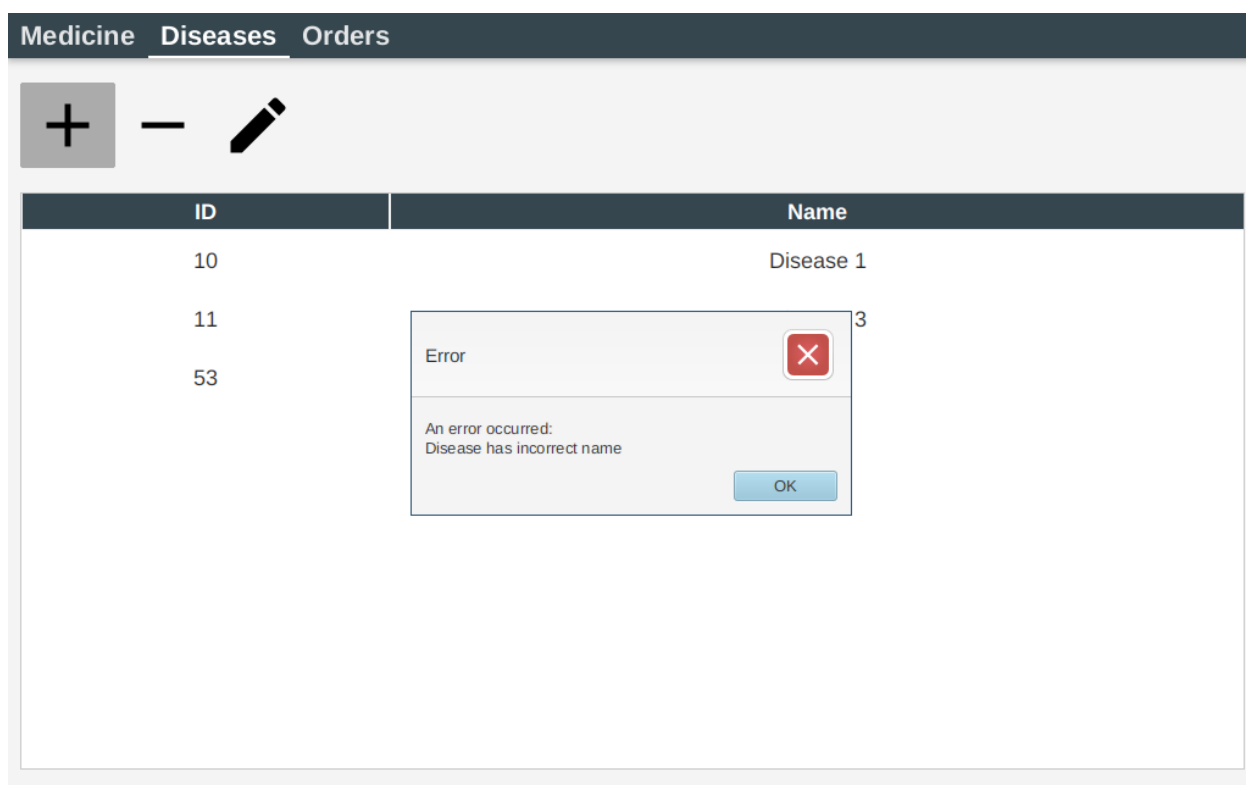


Рис. 3.7 Сообщение об ошибке при попытке добавления болезни с пустым именем

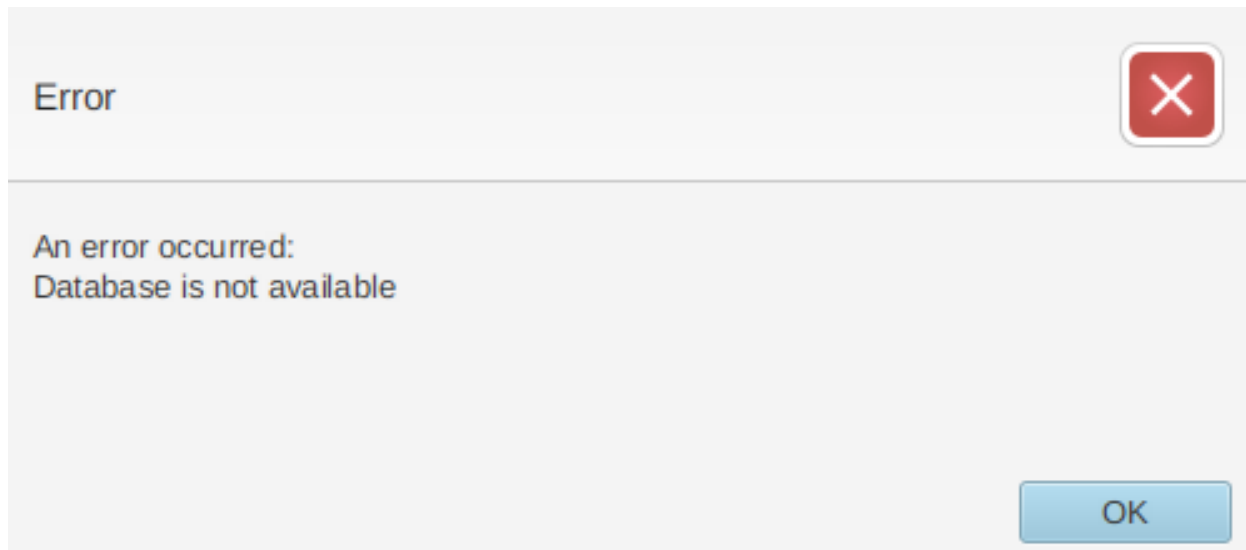


Рис. 3.8 Сообщение об ошибке при неудачной попытке соединения с БД

#### 4. ИСХОДНЫЕ ТЕКСТЫ ПК

Исходный код программного комплекса и документация исходного кода (сгенерированная с помощью Javadoc) доступны в открытом доступе по следующим адресам соответственно:

- [https://github.com/stnrepin/oop\\_labs/](https://github.com/stnrepin/oop_labs/)
- [https://stnrepin.github.io/oop\\_labs/](https://stnrepin.github.io/oop_labs/)

## ЗАКЛЮЧЕНИЕ

В ходе выполнения курсового проекта были достигнуты все поставленные цели: разработано техническое задание на ПК, проведено проектирование с применением UML, создан ПК на ОО языке, написана программная документация различных видов.

Были получены ценнейшие знания о подходах и методах проектирования сложных программных проектов, используя при этом современные и устойчивые технологии. Так, в данной курсовой работе используются государственные стандарты, методология разработки waterflow и различные виды UML-диаграмм. В совокупности все это дает возможность создания приложений, полностью удовлетворяющих потребности клиента, и делает процесс разработки абсолютно прозрачным как для программиста, так и для клиента.

При непосредственной разработке программы было глубоко изучено практическое применение объектно-ориентированного языка Java последних стандартов вместе с фундаментальными фреймворками и библиотеками языка: JavaFX, Hibernate, JPA, Jaspersoft, JUnit, Log4j и других. Знание и понимание этих технологий является неотъемлемым требованием текущего рынка труда, предъявляемым Java-программистам.

Разработка документации также познакомила с существующими в этой области подходами. Были изучены требования соответствующих государственных стандартов вкупе со средствами, предоставляемыми экосистемой Java (например, Javadoc и юнит-тестирование).



## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Объектно-ориентированное программирование: Методические указания к курсовому проектированию по дисциплине «Объектно-ориентированное программирование» / Сост. Г. В. Разумовский, А. Ф. Казак, С. А. Романенко, Д. В. Тихонов. СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2006, 32 с
2. Объектно-ориентированное программирование на языке Java: Методические указания к лабораторным работам /Сост.: С. А.Беляев, М. Г. Павловский, Г. В. Разумовский;Под общ. ред. Г. В. Разумовского. СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2013. 63с.
3. ГОСТ 24.701-86. Единая система стандартов автоматизированных систем управления. Надежность автоматизированных систем управления. Основные положения. М.: Издательство стандартов, 1987. — 17 с.
4. Пример технического задания по ГОСТ 19.201-78 [Электронный ресурс]. URL: <https://pro-prof.com/forums/topic/пример-технического-задания-по-гост-19-201-78/>
5. Официальная документация к Java Platform [Электронный ресурс] // Oracle. Java Documentation. URL: <https://docs.oracle.com/javase/8/javase-clienttechnologies>.
6. Официальная документация к Maven [Электронный ресурс] // Apache Maven Project. URL: <https://maven.apache.org/guides/index.html>
7. Ваше первое приложение на Hibernate [Электронный ресурс]. URL: <https://javarush.ru/groups/posts/hibernate-java>
8. Jeff Langr. Modern C++ Programming with Test-Driven Development: Code Better, Sleep Better. - New York: Pragmatic Bookshelf, 2013. - 368 p.