**СОДЕРЖАНИЕ**

Введение 5

1 Технический проект 8

1.1 Анализ предметной области 8

1.2 Постановка задачи 10

1.3 Проектирование функциональности программы 13

1.4 Проектирование структуры проекта 14

1.5 Проектирование базы данных 16

1.6 Выбор средств разработки 17

1.7 Требования к программному обеспечению и техническим средствам 21

2 Рабочий проект 23

2.1 Физическая модель данных 23

2.2 Функциональное взамодействие модулей программы 26

2.3 Входные и выходные данные 30

2.4 Установка и настройка программы 32

2.5 Работа с программой 33

2.5.1 Desktop-приложение «Справочник Заболеваний» 33

2.5.2 Android – приложение «Справочник Заболеваний» 37

Заключение 44

Список использованных источников 47

Приложение А (обязательное) Диаграмма Use Case для проектируемого

портала 48

Приложение Б (обязательное) Схема проектируемой структуры клиентских

Desktop и Android приложений 49

Приложение В (обязательное) Логическая модель базы данных 50

Приложение Г (справочное) Листинг модуля DiseaseController.java 51

Приложение Д (справочное) Содержание документации API 54

Приложение Е (справочное) Листинг модуля HttpHelper.cs 55

Приложение Ж (обязательное) Схема функуционального взаимодействия

модулей desktop-приложения 58

Приложение И (обязательное) Схема функуционального взаимодействия

модулей android-приложения 59

**ВВЕДЕНИЕ**

Темой курсового проекта является разработка информационного портала «Справочник Заболеваний».

Актуальность проекта заключается в предоставлении общедоступной информации о заболеваниях людям, которые обращают внимание на свое здоровье и заинтересованы в ее получении, а также проявления его вклада в общественное здоровье. Предоставление доступной и надежной информации о заболеваниях способствует повышению осведомленности людей о своем здоровье и позволяет им принимать информированные решения о профилактике, лечении и соблюдении рекомендаций врачей. Это в свою очередь способствует снижению распространения заболеваний и улучшению общественного здоровья.

Целью данного курсового проекта является разработка портала, который станет удобным источником информации о различных заболеваниях для различных платформ. Пользователи смогут использовать портал для получения информации о здоровье, а также для получения рекомендаций по профилактике и лечению различных заболеваний.

Написание данного курсового проекта является целесообразным по причине:

* доступности просмотра материала;
* удобного формата для ознакомления;
* предоставления доступа к обширной базе данных (БД), содержащей информацию о причинах, симптомах, методах диагностики и лечения разного рода заболеваний;
* минимальной конкуретности на рынке программных изделий.

Объект исследования – заболевания человека.

Предмет исследования – причины, симптомы, методы диагностики и лечения разного рода заболеваний, а также средства реализации информационного портала.

Для достижения поставленной цели будут рассмотрены и решены следующие задачи:

* изучены особенности функционирования информационных порталов;
* разработана структура и функциональность портала;
* описаны входные и выходные данные;
* произведен выбор средств реализации;
* описаны минимальные требования к программно-техническим средствам;
* разработан программный код приложений;
* создано руководство пользователя;
* разработана инсталляция.

Основными пользователями данной программы будут люди желающие ознакомиться с описанием заболеваний, их симптомами, методами диагностики и лечения.

Портал способствует образованию, информированности и принятию осознанных решений пользователем относительно своего здоровья. Он помогает предупредить заболевания, рано их обнаружить, облегчает поддержку медицинских решений и снижает нагрузку на медицинский персонал.

Информационный портал «Справочник Заболеваний» влияет на образование и информированность в области медицины, а также на медицинские исследования и общественное здоровье. Его предоставление достоверной и актуальной информации о заболеваниях способствует повышению качества медицинской помощи и улучшению здоровья населения.

Теоретическими основами и методами решения поставленных задач будут методологии проектирования структуры БД на основе собранной и систематизированной информации о различных заболеваниях, включающей причины, симптомы, методы диагностики и лечения, а также базовой функциональности приложений, особенности реализации программного кода с помощью конкретного языка программирования, принципы и алгоритмы применения инструментальных средств на различных этапах разработки программного продукта.

**1 ТЕХНИЧЕСКИЙ ПРОЕКТ**

**1.1 Анализ предметной области**

Заболевания – это состояния, при которых нарушается нормальное функционирование организма человека. Они могут быть вызваны различными факторами, включая инфекции, генетические нарушения, вредные привычки, плохое питание, стресс и другие.

Информационный портал «Справочник Заболеваний» является важным информационным ресурсом, который предоставляет пользователю информацию о различных заболеваниях, их симптомах, причинах, методах лечения и профилактике.

Портал обладает рядом преимуществ, которые делают его ценным ресурсом для пользователей.

Некоторые из этих преимуществ включают:

* доступность и удобство использования – портал доступен для пользователей в любое время и на любой платформе. Он имеет удобный интерфейс и простую навигацию, что делает поиск информации о заболеваниях быстрым и удобным;
* обширная информация – портал содержит обширную БД о различных заболеваниях, их симптомах, причинах, методах диагностики и лечения. Это позволяет пользователям получить полное представление о конкретном заболевании, его последствиях и возможностях лечения;
* поиск по симптомам – портал позволяет пользователям осуществлять поиск заболеваний их симптомам. Это позволяет пользователям сравнить свои симптомы с описанием заболеваний и получить предварительное представление о возможной причине своего состояния;
* широкая аудитория – портал может быть полезен для широкого круга пользователей, включая пациентов, медицинский персонал, студентов и просто людей, интересующихся здоровьем. Это делает его полезным и информативным ресурсом для различных категорий пользователей;
* поиск информации – портал предоставляет функцию поиска, которая позволяет пользователям быстро найти информацию о конкретном заболевании или его симптомах. Это экономит время и упрощает процесс получения нужной информации.

Портал предназначен для выполнения ряда актуальных и значимых целей, таких как:

* образование и информированность – портал предоставляет пользователю доступ к достоверной и проверенной информации о различных заболеваниях, их симптомах, причинах, методах диагностики и лечения. Это помогает пользователям расширить свои знания о медицине и здоровье и стать более информированными в принятии решений относительно своего здоровья;
* предупреждение заболеваний и раннее обнаружение – портал предоставляет пользователям информацию о симптомах различных заболеваний. Это помогает им распознать потенциальные признаки заболевания и принять меры к его раннему выявлению и лечению. Таким образом, портал способствует предупреждению и своевременному лечению заболеваний;
* поддержка принятия решений – портал предоставляет информацию о различных методах диагностики и лечения заболеваний. Это помогает пользователям ознакомиться с возможными вариантами и принять осознанное решение относительно своего лечения. Портал может также предоставлять информацию о побочных эффектах и последствиях лечения, что помогает пользователям принять взвешенное решение;
* сокращение нагрузки на медицинский персонал – предоставление пользователю доступа к информации о заболеваниях и методах лечения может помочь снизить нагрузку на медицинский персонал. Пользователи могут получить необходимую информацию о заболевании и принять решение о посещении врача без необходимости немедленного обращения к специалисту.

**1.2 Постановка задачи**

Необходимо разработать Desktop-приложение и мобильное приложение для операционной системы (ОС) Android для поиска и получения информации о заболеваниях, а также разработать Web-приложение для управления данными портала.

Портал должен использовать информацию, хранимую в БД. Данные по заболеваниям должны быть приведены к третьей нормальной форме (3НФ).

Информационный портал «Справочник Заболеваний» необходимо разработать с использованием трехуровневой архитектуры. Для бизнес-логики необходимо будет разработать API.

Портал должен обеспечивать следующие возможности:

* авторизацию пользователей, имеющих доступ к управлению данными на портале;
* изменение данных (доступно только для авторизованных пользователей);
* добавление данных (доступно только для авторизованных пользователей);
* удаление данных (доступно только для авторизованных пользователей);
* получение информации;
* поиск информации;
* фильтрация информации по запросу пользователя;
* обеспечение безопасности данных БД от несанкицонированного управления.

Программный продукт должен включать в себя:

* клиентское приложения для Desktop;
* клиентское приложение для Android;
* API для бизнес-логики;
* Web-приложение для управления данными портала.

План реализации программного продукта включает:

* исследование требований и проектирование архитектуры системы, включая структуру БД, API и интерфейсы приложений;
* создание API для доступа к данным, обеспечения безопасности и аутентификации;
* реализация веб-приложения для администрирования данных в БД;
* cоздание Android-приложения для мобильных устройств;
* разработка Desktop-приложения для персональных компьютеров;
* развертывание системы, обеспечение ее работоспособности.

Требования по входным данным:

* запросы пользователей – данные сформулированные пользователями для поиска информации о заболеваниях. Запросы могут содержать ключевые слова, например, названия заболеваний;
* данные администраторов – ключает информацию, необходимую для управления данными в БД. Это могут быть данные о новых заболеваниях, обновлениях или изменениях в существующей информации;
* данные аутентификации и учетной записи – включают информацию, предоставляемую администраторами при регистрации и входе на портал. Эти данные включают идентификационную информацию, такую как имя, электронная почта и пароль.

Требования по выходным данным:

* информация о заболеваниях – выходные данные представляют собой информацию о заболеваниях, включая описание, симптомы, методы лечения и профилактику. Эта информация предоставляется пользователям в ответ на их запросы;
* результаты поиска – включают результаты поиска, которые соответствуют запросам пользователей. Эти данные включают в себя список заболеваний, которые соответствуют поисковому запросу;
* интерфейс администратора – выходные данные для администраторов включают интерфейс управления данными, который позволяет им добавлять, редактировать и удалять информацию о заболеваниях;
* данные ответов на запросы от API – предоставляют внешним приложениям доступ к информации о заболеваниях. Эти данные должны быть в формате JSON и могут содержать запрошенную информацию;
* данные учетных записей – включают информацию о профилях администраторов и их настройках;
* сообщения и уведомления – включают сообщения и уведомления для пользователей, например, о подтверждении учетной записи, смене пароля или уведомлениях об изменениях на портале, ошибках для внешних приложений.

Данные запросов должны поступать на API и отправляться внешним приложениями в формате JSON.

Данные должны храниться и передаваться в безопасной и защищенной форме. Доступ к данным администраторов должен быть строго контролируемым через аутентификацию. Данные администраторов должны храниться в зашифрованной форме, и должны быть приняты меры для обеспечения их конфиденциальности для предупреждения несанкиционированного доступа к получению прав администратора.

**1.3 Проектирование функциональности программы**

Для описания взаимодействия пользователей и администраторов с информационным порталом «Справочник Заболеваний» воспользуемся методологией UML.

Рассмотрим на диаграмме вариантов использование взаимодействие пользователей с разрабатываемым порталом. Для этого построим диаграмму Use Case в которой имеются:

* актеры («Пользователь», «Администратор», «API», «DB»);
* прецеденты («Авторизация», «Добавление информации», «Удаление информации», «Обновление информации», «Поиск по заболеванию», «Просмотр информации о заболевании», «Обработка запроса API», «Обработка запроса БД»).

Диаграмма вариантов использования представлена в приложении А.

Рассмотрим ключевые моменты диаграммы:

* Пользователь может выполнять прецеденты «Поиск по заболеваниям» и «Просмотр информации о заболевании»;
* «Администратор» может выполнять прецеденты «Авторизация», «Добавление записи о заболевании», «Удаление записи о заболевании» и «Обновление записи о заболевании»;
* API обрабатывает запросы от актеров и взаимодействует с БД;
* DB обрабатывает запросы от API и доступ к данным.

При рассмотрении можно увидеть, что актер «Администратор» расширяет актера «Пользователя» и поэтому также имеет доступ к прецедентам «Поиск по заболеванию», «Просмотр информации о заболевании».

Прецеденты такие как: «Добавление информации», «Удаление информации» и «Обновление информации» выполняются после выполненного прецедента «Авторизация» для обеспечения санкционированного доступа к управлению данными о заболевании.

Прецедент «Обработка запроса API» проверяет роль пользователя при, а затем выполняет запрос. «Обработка запроса API» включает в себя прецедент «Обработка запроса БД», который выполняется после условий для выполнения прецедента «Обработка запроса API», и получает от прецедента «Обработка запроса API» необходимый запрос для получения данных из БД.

**1.4 Проектирование структуры проекта**

С целью удостоверения предъявляемых к программе требований, приведенных в разделах 1.1-1.2, при реализации портала будет использоваться трехзвенная архитектура. Данная архитектура представляет собой организацию приложения на трех уровнях, каждый из которых выполняет определенные функции:

* представление – уровень отвечает за отображение пользовательского интерфейса и взаимодействие с пользователем. Он включает в себя компоненты, отвечающие за визуализацию данных и обработку пользовательских действий [10]. В этот уровень будут входить: Desktop, Android и Web-приложения;
* логика приложения – на этом уровне происходит обработка бизнес-логики, связанной с функциональностью портала. Здесь будут располагаться компоненты, отвечающие за обработку запросов пользователей, проверку прав доступа, обработку данных и взаимодействие с БД [10]. На этом уровне архитектуры будет располагаться модуль поиска информации о заболеваниях, модуль аутентификации пользователей, модуль обработки форм обратной связи и так далее;
* слой данных – уровень отвечает за хранение и доступ к данным. Здесь находится БД, содержащая информацию о заболеваниях, и компоненты, реализующие операции чтения и записи данных [10]. На портале этот слой будет представлять СУБД и компоненты, работающие с ней для получения и записи данных.

Структура клиентских Desktop и Android приложений будет включать в себя следующие окна:

* загрузка (запуск перед отображением главное окна);
* главное (для просмотра списка о заболеваниях и их спецификациях);
* заболевания (просмотр подробной информации о выбранном из списка заболевании на главном окне);
* разделы (просмотр списка разделов заболеваний, систематизированных по органам);
* список заболеваний с фильтрацией по разделу (просмотр списка заболеваний принадлежащих выбранному разделу в окне разделов);
* настройки (настройка для удобства просмотра информации о заболеваниях, например, увеличение шрифта, его изменение, смена темной/светлой темы приложения);
* справка (справочный материал для пользователя о приложении).

В приложениях должна быть предусмотрена реализация меню, которое позволит переходить между разными окнами или разделами для легкого доступа и удобной навигации для пользователя. Для выполнения требования по получению пользователями информации, путем поиска по названию заболевания, необходимо реализовать поисковую систему в приложениях.

Схема проектируемой структуры приложений представлена в приложении Б.

Разрабатываемое Web-приложение будет состоять из следующих составляющих:

* аутентификация и авторизация (форма входа для аутентификации администраторов с разделением прав доступа);
* панель управления (центральная страница, где администраторы могут видеть общий обзор и доступ к основным функциям);
* просмотр и удаление имеющихся заболеваний (страница со списком заболеваний с возможностью удаления заболевания);
* управление данными о заболеваниях (форма для добавления, редактирования информации о заболеваниях, включая название, описание, симптомы, лечение и так далее);
* управление изображениями и медиа-контентом (дополнительная панель формы добавления, изменения заболевания для загрузки изображений, связанных с заболеваниями);
* управление настройками (настройка параметров приложения, таких как темы, шрифты и другие настройки).

Тестирование программ планируется проводить путём выбора каждого пункта меню и проверки корректности получаемого отклика от программы. При тестировании будет производиться эмуляция различных вариантов пользовательских действий.

**1.5 Проектирование базы данных**

Для хранения данных о заболевании будет разработана БД. Она будет состоять из следующих таблиц:

* Заболевание (id, classification, danger, definition, image, name, therapy, category\_id, doctor\_speciality\_id);
* Категория (id, value);
* Специальность доктора (id, value);
* Причина (id, value);
* Причина Заболевания (disease\_id, reasons\_id);
* Группа риска (id, value);
* Группа риска Заболевания (disease\_id, risk\_group\_id);
* Симптом (id, value);
* Симптом Заболевания (disease\_id, symptoms\_id);
* Диагностика (id, value);
* Диагностика Заболевания (disease\_id, diagnostic\_id);
* Профилактика (id, value);
* Профилактика Заболевания (disease\_id, preventions\_id).

Логическая модель БД представлена в приложении В.

**1.6 Выбор средств разработки**

Из множества различных языков программирования для реализации курсового проекта были выбраны C#, Kotlin, Java и интегрированные среды разработки Rider 2023, Android Studio 2022, IntelliJ IDEA 2023.1.1.

C# – это современный, объектно-ориентированный язык программирования, разработанный корпорацией Microsoft. Язык программирования C# поддерживает парадигму объектно-ориентированного программирования (ООП), что позволяет разрабатывать модульные и структурированные приложения. C# является строго типизированным языком, что означает, что все переменные должны быть объявлены с определенным типом данных. Начиная с .NET Core (позднее .NET 5 и .NET 6), стал более мультиплатформенным и поддерживает разработку приложений на различных операционных системах, включая Windows, Linux и macOS. Он взаимодействует с платформой .NET, предоставляя доступ к множеству библиотек и инструментов, что упрощает разработку приложений. C# постоянно обновляется, добавляя новые синтаксические особенности и возможности, такие как асинхронное программирование, лямбда-выражения и так далее [6].

Rider – это кросс-платформенная IDE для .NET-разработчиков, основанная на платформе IntelliJ и ReSharper. Rider поддерживает не только C#, но и другие языки, такие как F#, VB.NET, и даже JavaScript, TypeScript, HTML, CSS, и SQL. Это делает его мощным инструментом для разработки полноценных веб-приложений и многих других типов приложений. IDE тесно интегрирована с .NET и ASP.NET, обеспечивая удобное создание веб-приложений и служб. Среда разработки поддерживает автоформатирование кода и множество возможностей для рефакторинга, что упрощает поддержку и улучшение кода. Rider также предлагает разнообразные инструменты для улучшения кода, включая автоматическое создание свойств, методов и многие другие функции [6].

Язык программирования C# и среда разработки Rider были выбраны для разработки Desktop-приложения.

Java – это мощный и универсальный язык программирования, который известен своей платформой Java Virtual Machine (JVM) и широким применением в различных областях разработки.Одним из главных преимуществ Java является возможность запускать программы на любой платформе, поддерживающей JVM. Это делает Java кросс-платформенным языком и позволяет создавать приложения, которые работают на разных операционных системах без изменений в исходном коде. Данный язык программирования предоставляет встроенную поддержку для многопоточности, что позволяет создавать многозадачные приложения и эффективно использовать многоядерные процессоры. Java автоматически управляет памятью, включая сборку мусора. Это освобождает разработчиков от необходимости явно выделять и освобождать память, что снижает вероятность утечек памяти и делает приложения более надежными [4].

Spring Boot – это популярный фреймворк для разработки приложений на языке Java. Он обеспечивает простоту и удобство создания высокопроизводительных и надежных веб-приложений. Включает в себя встроенные серверы приложений, такие как Tomcat, Jetty и Undertow. Это позволяет развертывать приложения без необходимости настройки отдельных серверов. Spring Boot имеет обширную экосистему, включая множество расширений и сторонних библиотек, что делает его мощным инструментом для разработки разнообразных приложений. Фреймворк предоставляет хорошую поддержку для создания RESTful API, что делает его идеальным выбором для разработки микросервисов [2].

IntelliJ IDEA – это одна из самых популярных интегрированных сред разработки (IDE) для языка программирования Java. Разработана компанией JetBrains и предоставляет разработчикам широкий набор инструментов и функций для эффективной работы. Предлагает автодополнение кода, что ускоряет процесс написания и уменьшает количество ошибок. Она также учитывает контекст и структуру вашего проекта, предлагая наиболее подходящие варианты. IntelliJ IDEA поддерживает разработку не только на Java, но и на множестве других языков, включая Kotlin, Scala, Groovy, JavaScript, и многие другие. Она также интегрируется с различными фреймворками и технологиями. IDEA интегрируется с системами контроля версий, такими как Git, что облегчает работу в команде и управление исходным кодом проекта [4].

Язык программирования Java с фреймворком Spring Boot и средой разработки IntelliJ IDEA выбраны для разработки API.

Kotlin – это современный статически типизированный язык программирования, разработанный компанией JetBrains. Он был создан с учетом многолетнего опыта в разработке программного обеспечения и призван упростить и улучшить процесс разработки. Данный язык предотвращает многие ошибки, связанные с нулевыми ссылками, благодаря использованию Nullable и Non-Nullable типов данных, а также предоставляет возможность создавать код, который может быть выполнен на разных платформах, включая JVM, Android, JavaScript и Native [4].

Android Studio – это интегрированная среда разработки (IDE), разработанная компанией JetBrains и предназначенная специально для создания приложений для платформы Android. Это официальная IDE, разработанная Google для создания Android-приложений. Она предоставляет доступ к последним Android SDK и инструментам для разработки. У нее интуитивно понятный и легкий в использовании интерфейс, который ускоряет процесс разработки приложений. Включает в себя компоненты, такие как редактор макетов, анализаторы производительности и отладчик. IDE обеспечивает поддержку для множества Android-устройств, что позволяет разработчикам адаптировать свои приложения для различных экранов и разрешений [4].

Kotlin и Android Studio были выбраны для разработки клиентского Android-приложения.

PostgreSQL – это мощная и открытая реляционная система управления БД (RDBMS). Она распространяется под открытой лицензией (PostgreSQL License), что означает, что можно свободно использовать, модифицировать и распространять ее без ограничений. Это делает ее доступной для широкого круга пользователей [1].

Для создания инсталляции Desktop-приложения был выбран Smart InstallMaker. Это условно-бесплатная утилита с закрытым исходным кодом, которая представляет пользователям мощный и простой в использовании инструмент для создания высокоэффективных инсталляторов в 32-битных ОС Microsoft Windows.

Smart InstallMaker обладает богатыми функциональными возможностями для создания надёжных и красивых инсталляционных пакетов. Утилита предоставляет мощный алгоритм Cabinet для сжатия и создания компактных инсталляторов с маленьким размером, поддерживает создание многоязычного инсталлятора (до 25 языков), а также предоставляет средства для управления внешним видом программы-установки. Среди прочих возможностей Smart InstallMaker, можно выделить создание деинсталлятора, ярлыков в указанных директориях системы, регистрацию новых шрифтов или расширений в операционной системе, поддержку ActiveX и переменных, проверку установленного .NET Framework, разделение инсталлятора на установочные диски, детальную настройку информации об установочном файле, системных требований (например, запретить установку программы в Windows 95), а также его внешнего вида.

**1.7 Требования к программному обеспечению и техническим средствам**

Для корректной работы Desktop-приложения необходимо, чтобы компьютер удовлетворял следующим минимальным требованиям:

* процессор Intel Core i3 4100f;
* объем свободной оперативной памяти не менее 4 Гб;
* видеоадаптер Intel HD Graphics 405;
* 500 Мб свободного места на жестком диске;
* стандартная клавиатура с 101/109 клавишами;
* манипулятор типа «мышь».

Для корректного функционирования системы на компьютере должна быть установлена ОС Windows 10/11, СУБД PostgreSQL.

Для корректной работы Android-приложения необходимо, чтобы смартфон удовлетворял следующим минимальным требованиям:

* Android версии 11 и старше;
* оперативная память не менее 4 Гб;
* сенсорный экран.

Для корректной работы Web-приложения необходимы следующие требования:

* процессор:
* минимальные требования – x86- или x64-разрядный двухъядерный процессор с тактовой частотой 1,9 ГГц, с набором инструкций SSE2;
* рекомендуемые требования – x64-разрядный двухъядерный процессор с тактовой частотой 3,3 ГГц и выше, с набором инструкций SSE2;
* память:
* минимальные требования – ОЗУ объемом 2 Гб;
* рекомендуемые требования – ОЗУ объемом 4 Гб и более;
* дисплей:
* минимальные требования – Super VGA с разрешением 1024 x 768;
* рекомендуемые требования – Super VGA с разрешением 1024 x 768.
* требования к сети:
* пропускная способность более 50 КБ/с (400 КБ/с);
* задержка меннее 150 мс.
* поддержка системой веб-браузера Microsoft Edge или Mozilla Firefox, Google Chrome.

**2 РАБОЧИЙ ПРОЕКТ**

**2.1 Физическая модель данных**

Физическая модель данных описывает структуру и организацию данных в информационной системе. Она определяет, какие элементы данных будут храниться, как они связаны между собой и как они будут доступны и обрабатываться. Физическая модель БД также определяет способ доступа к данным и методы их обновления и удаления. Она является важной частью проектирования БД и помогает обеспечить эффективное хранение и доступ к данным.

В случае реляционной СУБД, физическая модель данных описывает таблицы, поля, связи и другие элементы, которые составляют БД. Таблицы представляют собой наборы записей, которые содержат данные об определенном элементе или сущности. Поля представляют собой отдельные атрибуты или свойства записей в таблице. Связи позволяют связывать данные из разных таблиц и обеспечивать целостность данных.

На основе представленной логической модели была создана физическая модель БД, состоящая из следующих таблиц:

* disease (таблица 1);
* doctor\_speciality (таблица 2);
* category (таблица 3);
* risk\_item (таблица 4);
* symptom (таблица 5);
* reason (таблица 6);
* prevention\_item (таблица 7);
* diagnostic\_item (таблица 8);
* disease\_preventions (таблица 9);
* disease\_diagnostic (таблица 10);
* disease\_risk\_group (таблица 11);
* disease\_symptoms (таблица 12);
* disease\_reasons (таблица 13).

Таблица 1 – Структура таблицы «disease» (Заболевание)

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Подпись поля | Тип данных | Ключ |
| id | Код заболевания | integer | PK |
| classification | Классификация | varchar(32600) | - |
| danger | Опасность | varchar(32600) | - |
| definition | Определение | varchar(32600) | - |
| image | Картинка | bytea | - |
| name | Наименование | varchar(255) | - |
| doctor\_speciality\_id | Код специализации | integer | FK |
| therapy | Терапия | varchar(32600) | - |
| category\_id | Код категории | integer | FK |

Таблица 2 – Структура таблицы «doctor\_speciality» (Специализация врача)

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Подпись поля | Тип данных | Ключ |
| id | Код специалиста | integer | PK |
| value | Наименование | varchar(255) | FK |

Таблица 3 Структура таблицы «category» (Категория)

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Подпись поля | Тип данных | Ключ |
| id | Код категории | integer | PK |
| value | Наименование | varchar(255) | FK |

Таблица 4 – Структура таблицы «risk\_item» (Риск)

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Подпись поля | Тип данных | Ключ |
| id | Код риска | integer | PK |
| value | Наименование | varchar(255) | FK |

Таблица 5 – Структура таблицы «symptom» (Симптом)

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Подпись поля | Тип данных | Ключ |
| id | Код симптома | integer | PK |
| value | Наименование | varchar(255) | FK |

Таблица 6 – Структура таблицы «reason» (Причина)

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Подпись поля | Тип данных | Ключ |
| id | Код причины | integer | PK |
| value | Наименование | varchar(255) | FK |

Таблица 7 – Структура таблицы «prevention\_item» (Профилактика)

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Подпись поля | Тип данных | Ключ |
| id | Код | integer | PK |
| value | Наименование | varchar(255) | FK |

Таблица 8 – Структура таблицы «diagnostic\_item» (Диагностика)

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Подпись поля | Тип данных | Ключ |
| id | Код | integer | PK |
| value | Наименование | varchar(255) | FK |

Таблица 9 – Структура таблицы «disease\_preventions» (Профилактика заболевания)

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Подпись поля | Тип данных | Ключ |
| disease\_id | Код заболевания | integer | FK |
| preventions\_id | Код профилактики | integer | FK |

Таблица 10 – Структура таблицы «disease\_diagnostic» (Диагностика заболевания)

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Подпись поля | Тип данных | Ключ |
| disease\_id | Код заболевания | integer | FK |
| diagnostic\_id | Код диагностики | integer | FK |

Таблица 11 – Структура таблицы «disease\_risk\_group» (Группы риска заболевания)

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Подпись поля | Тип данных | Ключ |
| disease\_id | Код заболевания | integer | FK |
| risk\_group\_id | Код риска | integer | FK |

Таблица 12 – Структура таблицы «disease\_symptoms» (Симптомы заболевания)

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Подпись поля | Тип данных | Ключ |
| disease\_id | Код заболевания | integer | FK |
| symptoms\_id | Код симптома | integer | FK |

Таблица 13 – Структура таблицы «disease\_reasons» (Причины заболевания)

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Подпись поля | Тип данных | Ключ |
| disease\_id | Код заболевания | integer | FK |
| reasons\_id | Код причины | integer | FK |

**2.2 Функциональное взаимодействие модулей программы**

На основе технического проекта на языке Java с использованием фремворка Spring был разработан REST API, который будет предоставлять различные операции для взаимодействия с БД.

В приложении REST API используются следующие системные модули:

* Spring Data – предоставляет различные инструменты для работы с БД. Он позволяет упростить работу с БД и облегчить разработку приложений, работающих с данными;
* Spring Security – предоставляет инструменты для обеспечения безопасности приложений. Он позволяет защитить приложения от несанкционированного доступа и других угроз безопасности;
* Spring Boot – предоставляет инструменты для быстрого и простого создания приложений. Он позволяет упростить процесс разработки приложений и сократить время, необходимое для создания работающего приложения [8].

Также в приложении REST API были созданы следующие пользовательские модули:

* DiseaseController – обрабатывает запросы к API, связанным с заболеваниями, и предоставляет возможности для получения списка заболеваний, получения информации о конкретном заболевании, создания нового заболевания и так далее (приложение Г);
* CategoryController – обрабатывает запросы к API, связанным с отделением заболеваний, и предоставляет возможности для получения списка отделений.

При разработке REST API был использован подход Code First к работе с БД. Для этого были написаны классы сущностей на языке Java, которые отражают структуру таблиц в БД. Генерация таблиц БД на основе классов представлена ORM-фреймворком Hibernate и JPA.

Список операций, предоставленных через API, включает в себя:

* получение списка всех заболеваний;
* получение списка разделов заболеваний;
* получение подробной информации о конкретном заболевании;
* добавление нового заболевания;
* изменение информации о существующем заболевании;
* удаление заболевания.

Для разработки документации к API использован набор инструментов Swagger. Краткое содержание документации представлено в приложении Д.

Также были разработаны desktop и android приложения «Справочник Заболеваний».

Для создания desktop-приложения была использована среда программирования Rider 2023, язык программирования C# на платформе .NET Core 6.0. Для создания графического интерфейса использовалась технология WPF.

В desktop-приложении используются следующие языковые модули:

* System – содержит фундаментальные и базовые классы, определяющие часто используемые типы значений и ссылочных данных, события и обработчики событий, интерфейсы, атрибуты и исключения обработки;
* System.Windows – содержит типы, используемые в приложениях;
* Windows Presentation Foundation (WPF), включая клиенты анимации, элементы управления пользовательского интерфейса, привязку данных и преобразование типов;
* System.Net.Http – предоставляет класс для отправки HTTP-запросов и получения HTTP-ответов от ресурса, определяемого URI;
* System.Threading – предоставляет классы и интерфейсы для многопоточного программирования. Помимо классов синхронизации работы потоков и доступа к данным (Mutex, Monitor, Interlocked, AutoResetEvent и так далее), это пространство имен содержит класс ThreadPool, позволяющий использовать пул предоставляемых системой потоков, и класс Timer, выполняющий методы обратного вызова в потоках пула потоков;
* System.Collections.Generic – содержит интерфейсы и классы, определяющие универсальные коллекции, которые позволяют пользователям создавать строго типизированные коллекции, обеспечивающие повышенную производительность и безопасность типов по сравнению с неуниверсальными строго типизированными коллекциями [6].

Также были созданы следующие пользовательские модули:

* AboutAppPage.xaml.cs – реализует окно с информацией о приложении;
* DepartmentPage.xaml.cs – реализует окно со списком отделений;
* DiseasePage.xaml.cs – реализует окно заболевания;
* DiseaseListPage.xaml.cs – реализует окно со списком заболеваний;
* HttpHelper.cs – реализует возможность отправки HTTP запросов на API (приложение Е);
* DepartmentsViewModel.cs – реализует свойства и команды, к которым представление может привязать модель списка отделений, и уведомляет представление об изменениях состояния с помощью событий уведомления об изменениях;
* DiseasesViewModel.cs – реализует свойства и команды, к которым представление может привязать модель списка заболеваний, и уведомляет представление об изменениях состояния с помощью событий уведомления об изменениях.

Схема функционального взаимодействия модулей программы представлена в приложении Ж.

Для создания android-приложения была использована среда программирования Android Studio 2023 и язык программирования Kotlin.

В android-приложении используются следующие языковые модули:

* androidx.fragment.app.Fragment – содержит методы для работы с фрагментами;
* android.view – представляет базовый строительный блок для создания интерактивных компонентов пользовательского интерфейса (кнопок, текстовых полей и так далее);
* android.recyclerview – содержит набор классов и интерфейсов, предоставляемых Android SDK для создания пользовательских интерфейсов с использованием RecyclerView;
* kotlinx.coroutines – библиотека, предоставляемая Kotlin для работы с параллельными и асинхронными операциями в приложении для Android [4].

Также были созданы следующие пользовательские модули:

* DiseaseAdapter.kt – реализует паттерн Адаптер, позволяющий объектам работать вместе с несовместимыми интерфейсами;
* DiseaseClient.kt – реализует возможность отправки HTTP запросов на API;
* AboutAppFragment.kt – реализует фрагмент для отображения информации о программе;
* CategoriesFragment.kt – реализует фрагмент для отображения списка категорий;
* DiseaseFragment.kt – реализует фрагмент для отображения подробной информации о заболевании;
* DiseaseListFragment.kt – реализует фрагмент для отображения списка заболеваний;
* OptionsFragment.kt – реализует фрагмент для отображения настроек приложения;
* SplashScreenActivity.kt – реализует заставку приложения.

Схема функционального взаимодействия модулей android-приложения представлена в приложении И.

**2.3** **Входные и выходные данные**

В модуле DepartmentsViewModel.cs и CategoryAdapter.kt входные данные представляют собой информацию, полученную из тела HTTP-запроса к API в формате JSON. После получения ответа, содержимое тела запроса десериализуется в список объектов класса Category.

В модуле DepartmentsViewModel.cs и CategoryAdapter.kt выходные данные представляют информацию о списке отделений и фильтрацию заболеваний по отделению.

В модуле DiseasesViewModel.cs и DiseasesAdapter.kt входные данные представляют собой информацию, полученную из тела HTTP-запроса к API в формате JSON. После получения ответа, содержимое тела запроса десериализуется в список объектов класса Disease.

В модуле DiseasesViewModel.cs и DiseasesAdapter.kt выходные данные представляют список заболеваний и возможность получения подробной информации по выбранному заболеванию из списка.

В модуле DiseaseViewModel.cs и DiseaseFragment.kt входные данные представляют собой информацию, полученную из тела HTTP-запроса к API в формате JSON. После получения ответа, содержимое тела запроса десериализуется объект класса Disease.

В модуле DiseaseViewModel.cs и DiseaseFragment.kt выходные данные представляют информацию о заболевании включающую в себя:

* наименование;
* определение;
* список причин;
* список симптомов;
* раздел терапии;
* раздел классификации;
* раздел опасность;
* список группы риска;
* список диагностики;
* список профилактики;
* специализация врача;
* отделение, к которому относится заболевание;
* иллюстрация.

**2.4 Установка и настройка программы**

Desktop-приложение устанавливается на компьютер при запуске файла «Setup.exe». При установке программы происходит создание папки «Disease Directory» в папке программ системы и осуществляется копирование файлов необходимых для запуска и корректной работы программы.

Для корректной работы приложения на компьютере также должен быть установлен PostgreSQL и запущен сервер API.

Программу можно удалить, выбрав пункт меню «Деинсталлировать» в меню Пуск/Программы или запустив файл uninstall.exe в папке программы.

Для стабильной работы desktop-приложения необходимо:

* процессор Intel Core i3 4100f;
* объем свободной оперативной памяти не менее 4 Гб;
* видеоадаптер Intel HD Graphics 405;
* 500 Мб свободного места на жестком диске;
* стандартная клавиатура с 101/109 клавишами;
* манипулятор типа «мышь».

Для создания инсталляции была использована программа Smart Install Maker 5.04. В ходе инсталляции пользователю показываются следующие окна:

* приветствие – в нем предлагается закрыть все запущенные приложения;
* краткая информация об устанавливаемом приложении – назначение, минимальные требования к программному обеспечению и техническим средствам;
* выбор папки установки;
* создание дополнительных ярлык – позволяет создать ярлык на рабочем столе или отказаться от создания ярлыков;
* ход выполнения процесса установки;
* завершение установки – позволяет выбрать файлы, которые нужно запустить после установки.

Для установки мобильной версии нужно разрешить в настройках устройства установку неизвестных приложений. Затем перейти в мобильный файловый менеджер и нажать на APK-файл для скачивания. Для установки потребуется разрешение, нужно нажать на кнопку «Установить».

**2.5 Работа с программой**

**2.5.1 Desktop-приложение «Справочник Заболеваний»**

Для запуска программы нужно нажать на иконку с названием «Disease Direction.exe», двойным кликом. После запуска программы открывается главное окно с активной страницей «Список заболеваний» (рисунок 1).

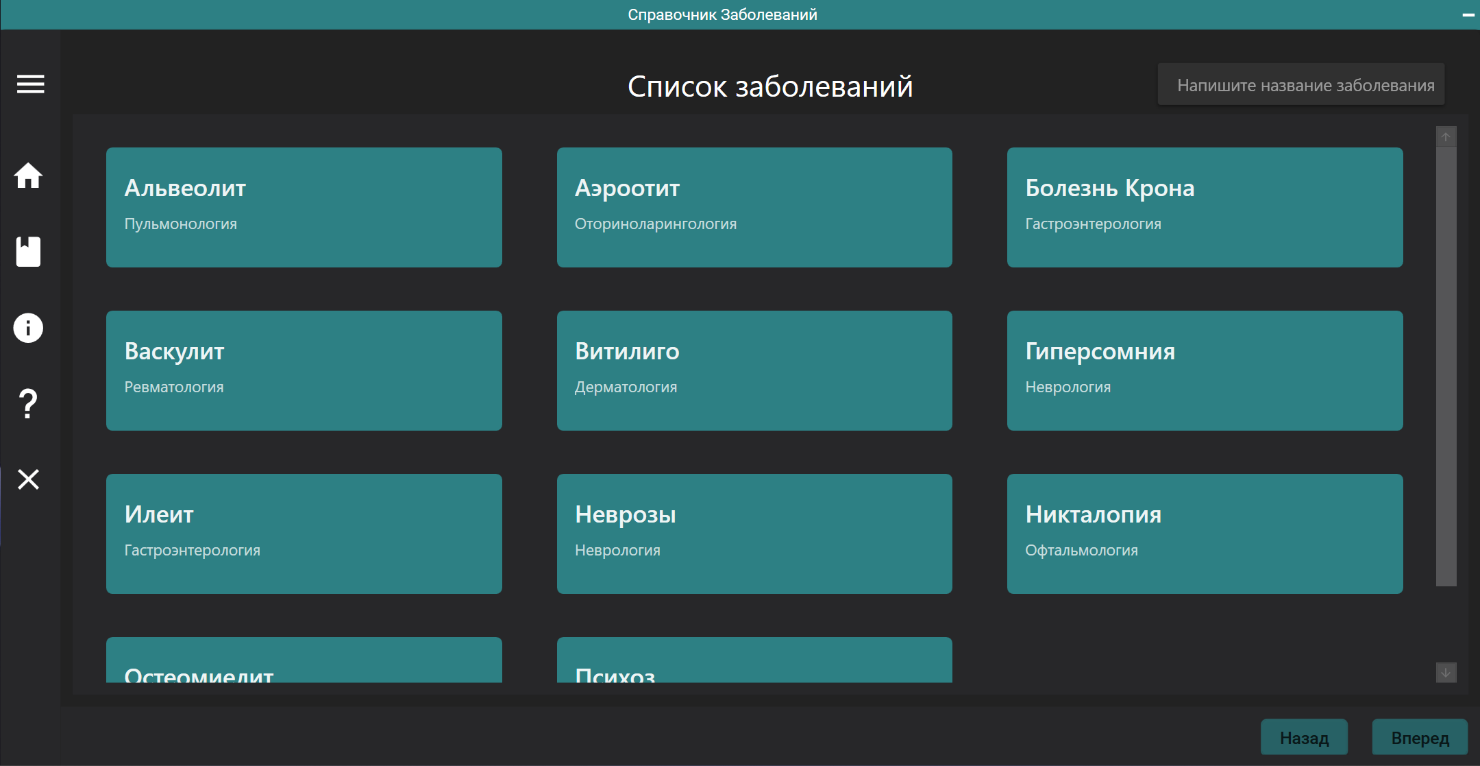


Рисунок 1 – Список заболеваний

На странице со списком заболеваний можно воспользоваться поисковой строкой, расположенной в правом верхнем углу страницы, которая позволяет найти необходимое заболевание из списка по его названию. Для этого необходимо ввести в поле поиска наименование заболевания. Поиск является индексированным и позволяет пользователю находить информацию, начиная с ввода первой буквы или нескольких первых букв ключевого слова (рисунок 2). Поиск предусматривает возможность поиска как в нижнем так и в верхнем регистре.

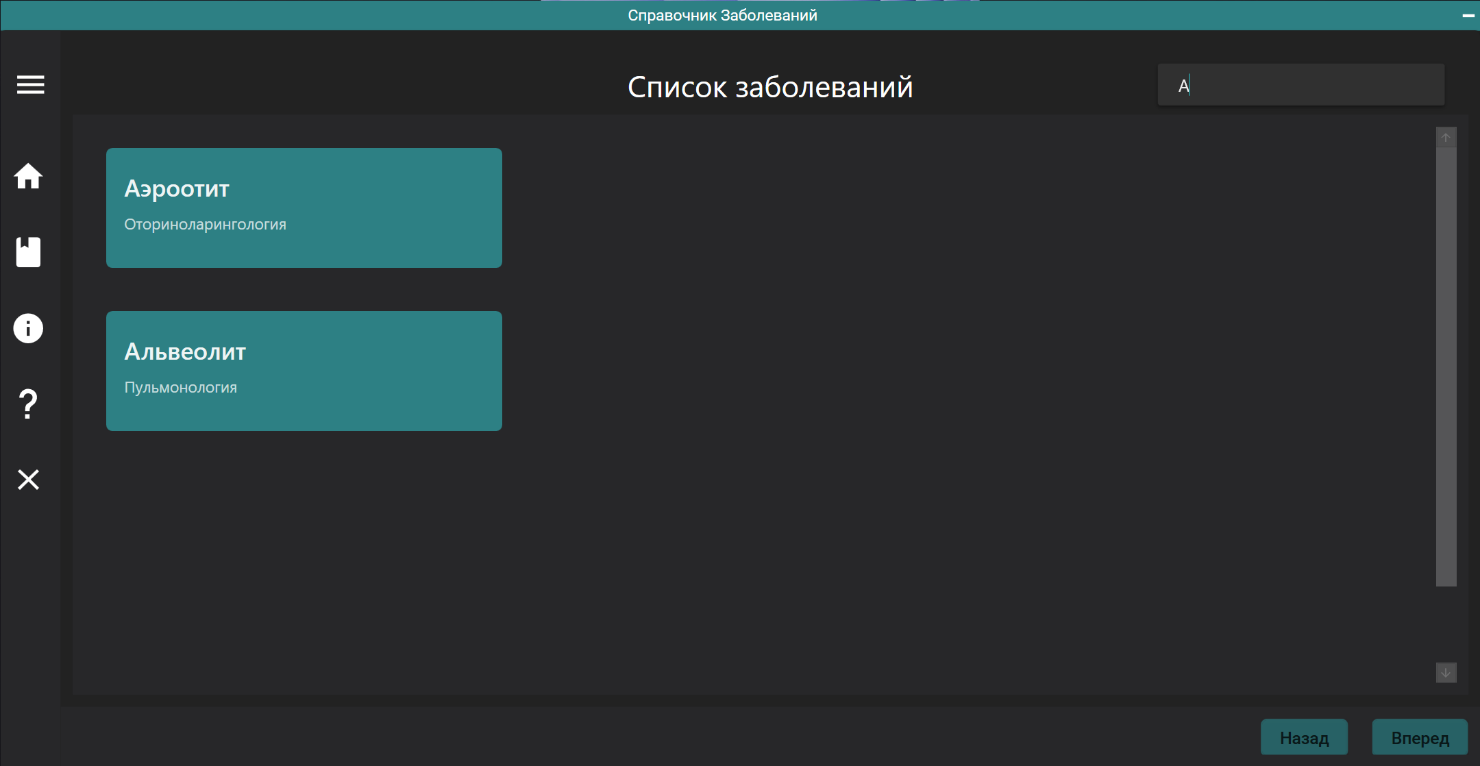


Рисунок 2 – Работа поисковой строки

При выборе заболевания из списка откроется страница с подробной информацией о нем (рисунок 3).

Для навигации в «Справочнике Заболеваний» используется навигационное меню, расположенное в левой части окна, а также кнопки навигации «Вперед» и «Назад» для перемещения между открытыми ранее страницами.

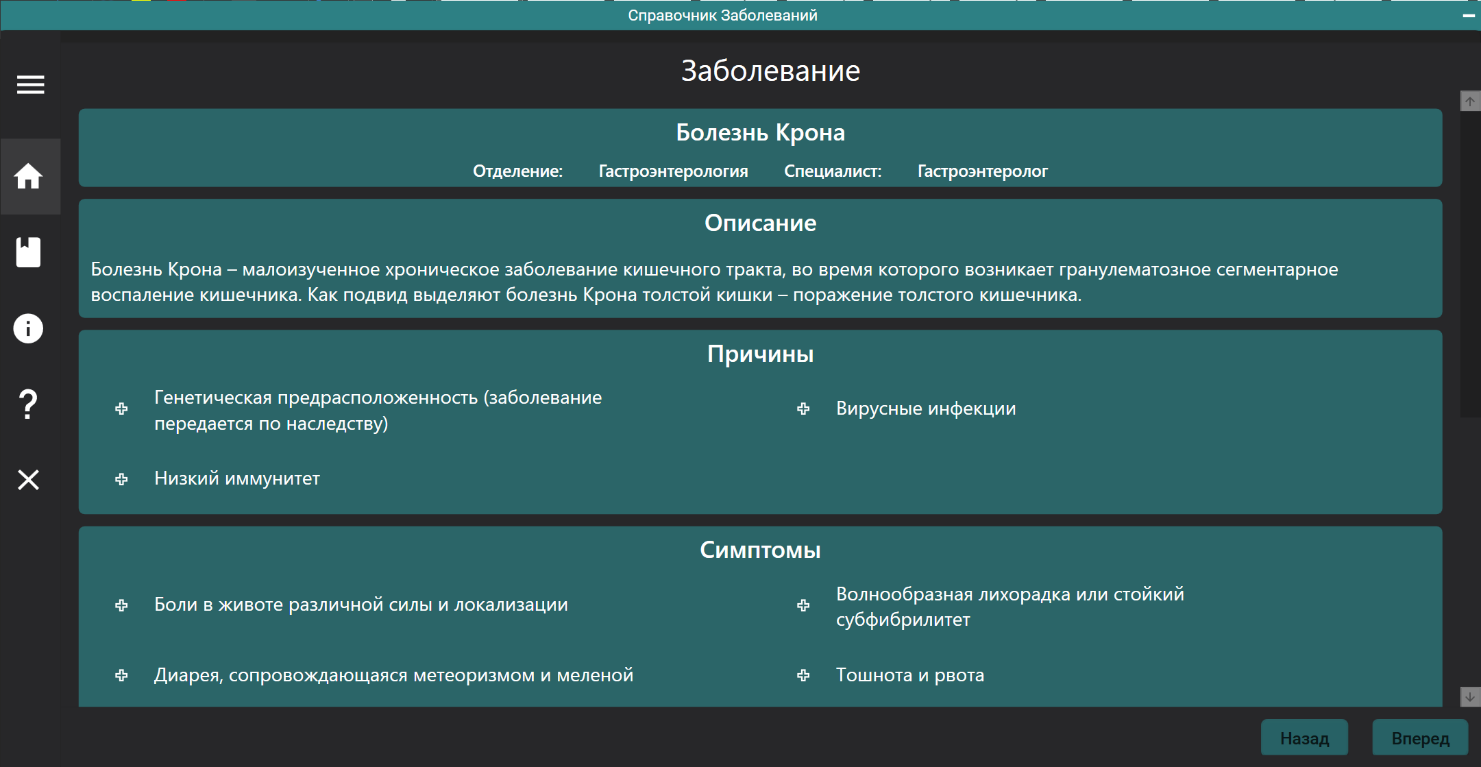


Рисунок 3 – Страница заболевания «Болезнь Крона»

Навигационное меню содержит:

* «Главное меню» (переход на окно «Список заболеваний»);
* «Отделение» (переход на страницу со списком отделений);
* «О программе» (переход на страницу «О программе»);
* «Справка»;
* «Закрыть приложение».

Страница отделений предназначена для отображения информации о различных медицинских отделениях, к которым относятся определенные заболевания. Для каждого отделения отображается список заболеваний,  
которые относятся к этому отделению. Это позволяет пользователям быстро и легко найти информацию о заболеваниях, связанных по определенному отделению.

Ниже представлена страница «Отделения» с открытым навигационным меню (рисунок 4).

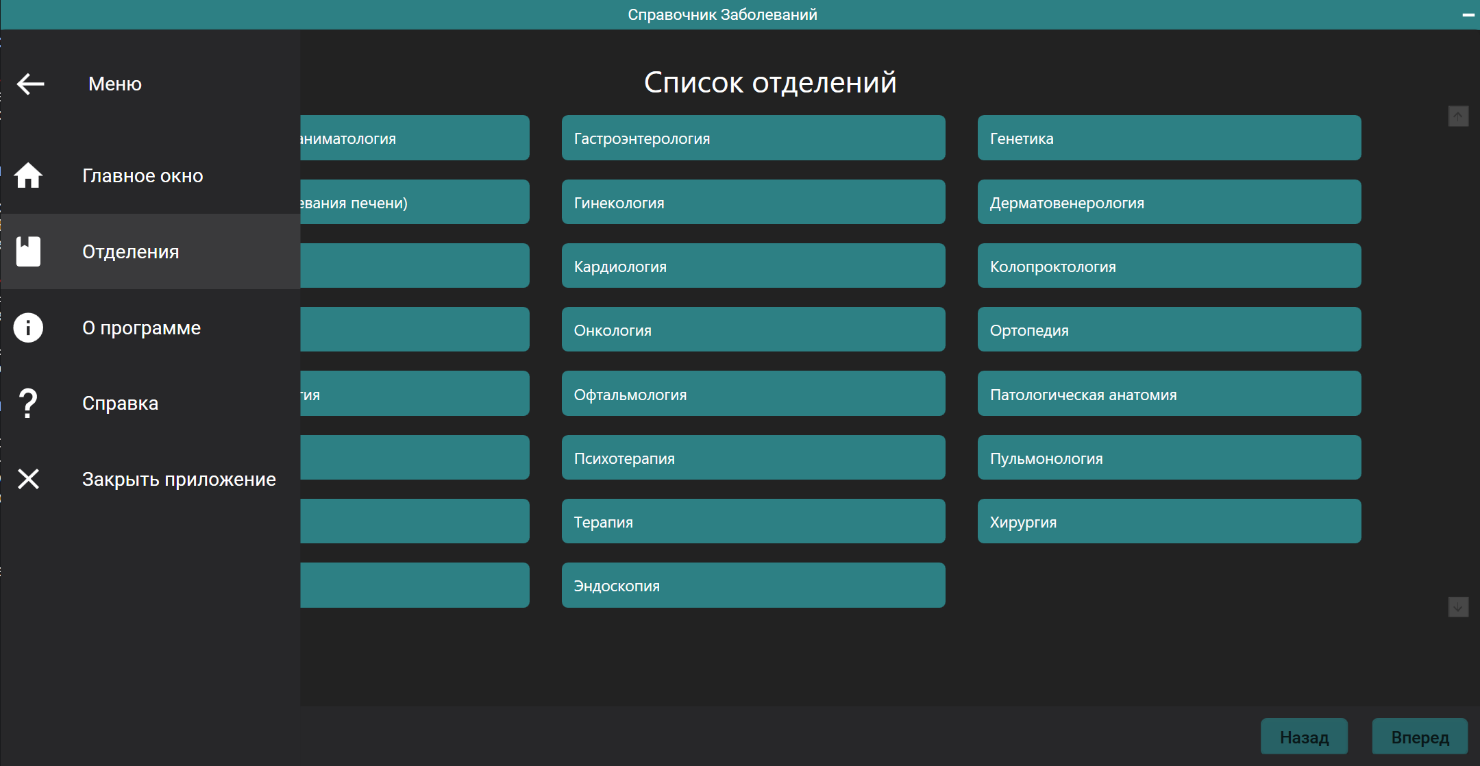


Рисунок 4 – Навигационное меню

При выборе в навигационном меню пункта «Справка» откроется окно справки, в котором будет информация о работе с программой (рисунок 5).

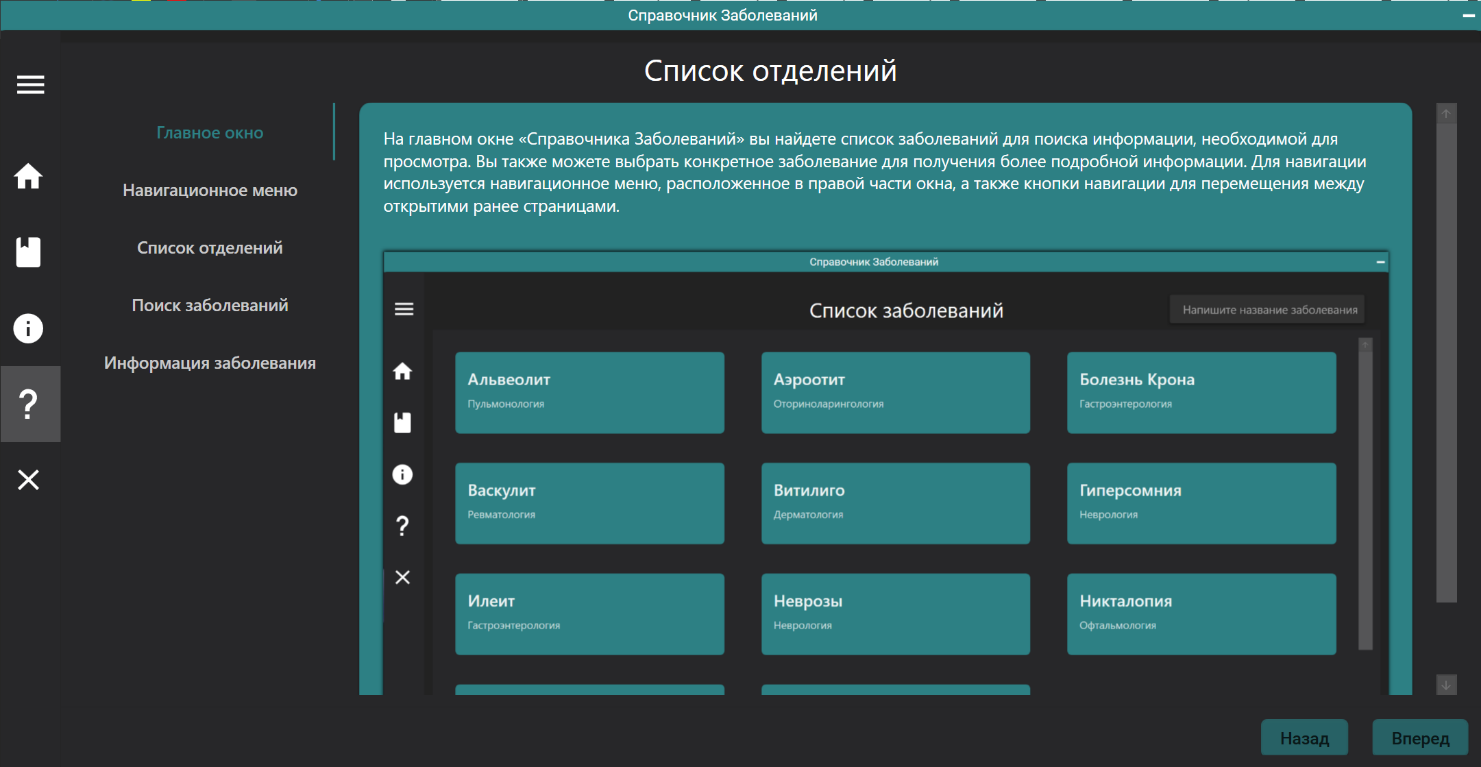


Рисунок 5 – Страница «Справка»

**2.5.2 Android – приложение «Справочник Заболеваний»**

Для запуска программы нужно нажать на иконку с названием «Disease Direction». После запуска программы открывается главное окно с активной страницей «Список заболеваний» (рисунок 6).

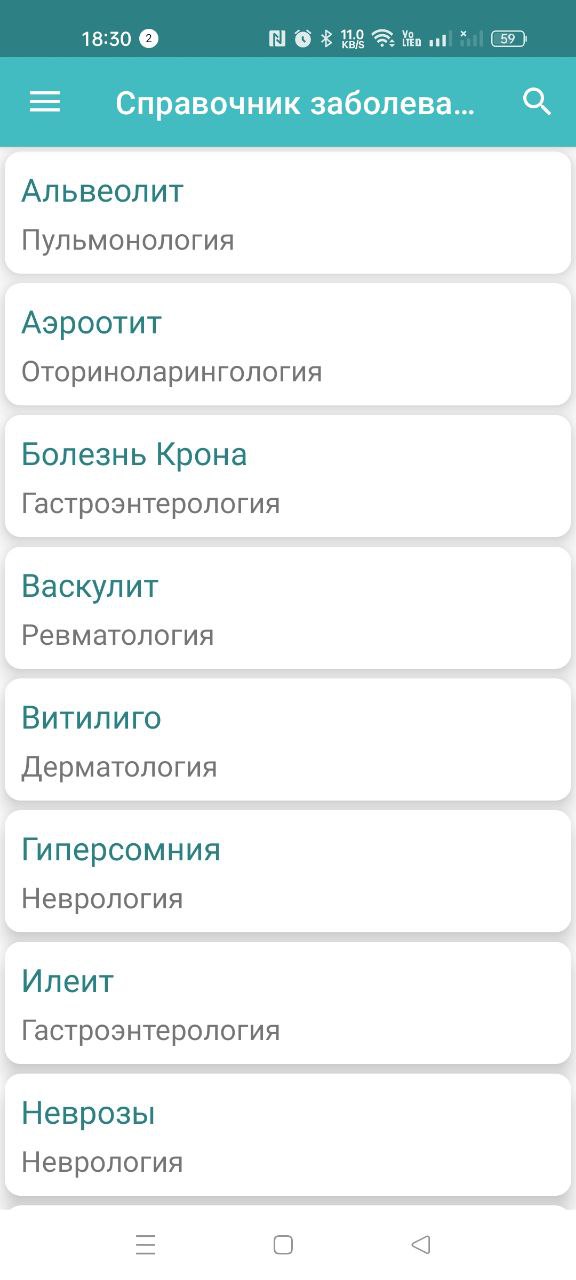


Рисунок 6 – Список заболеваний

На главном окне «Справочника Заболеваний» располагается список заболеваний для поиска информации. Из списка предусматривается выбор заболевания для получения более подробной информации.

На странице со списком заболеваний можно воспользоваться поисковой строкой, которая позволяет найти необходимое заболевание из списка по его названию. Для этого необходимо ввести в поле поиска наименование заболевания (рисунок 7).

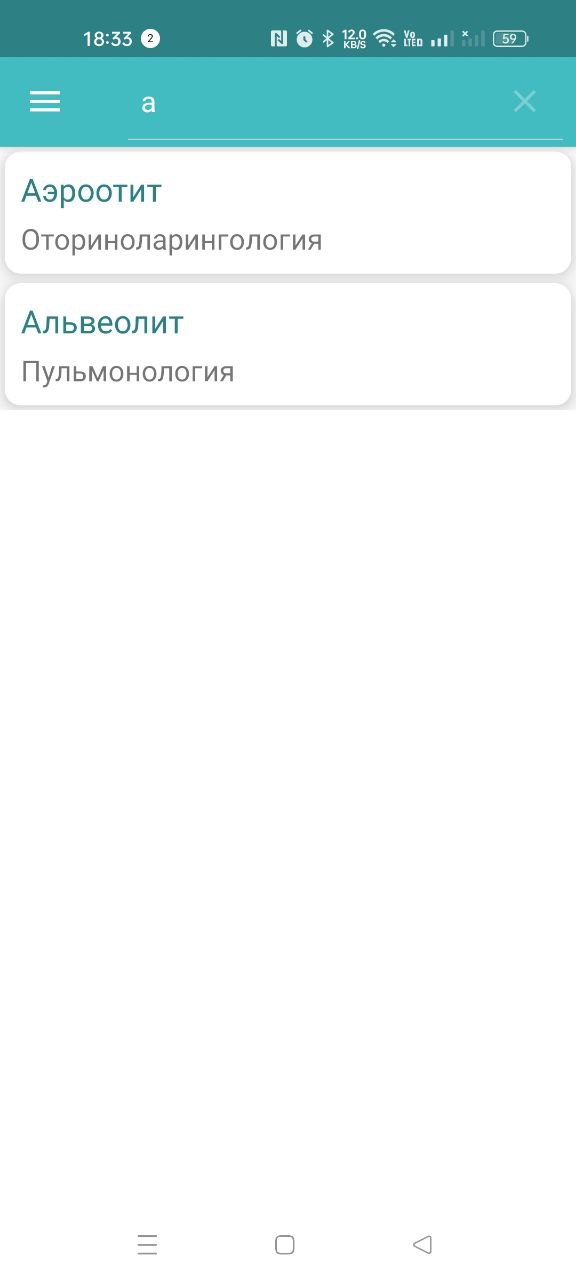


Рисунок 7 – Работа поиска по названию заболевания

Для навигации в «Справочнике Заболеваний» используется навигационное меню, расположенное в левой части окна, а также кнопки навигации для перемещения между открытими ранее страницами (рисунок 8). Для отображения меню нажмите на иконку навигационного меню в левом верхнем углу. После нажатия на иконку на экране будут доступны элементы меню для выбора и перехода на выбранный фрагмент. Чтобы перейти на необходимый фрагмент, нажмите на элемент из списка представленных в навигационном меню.

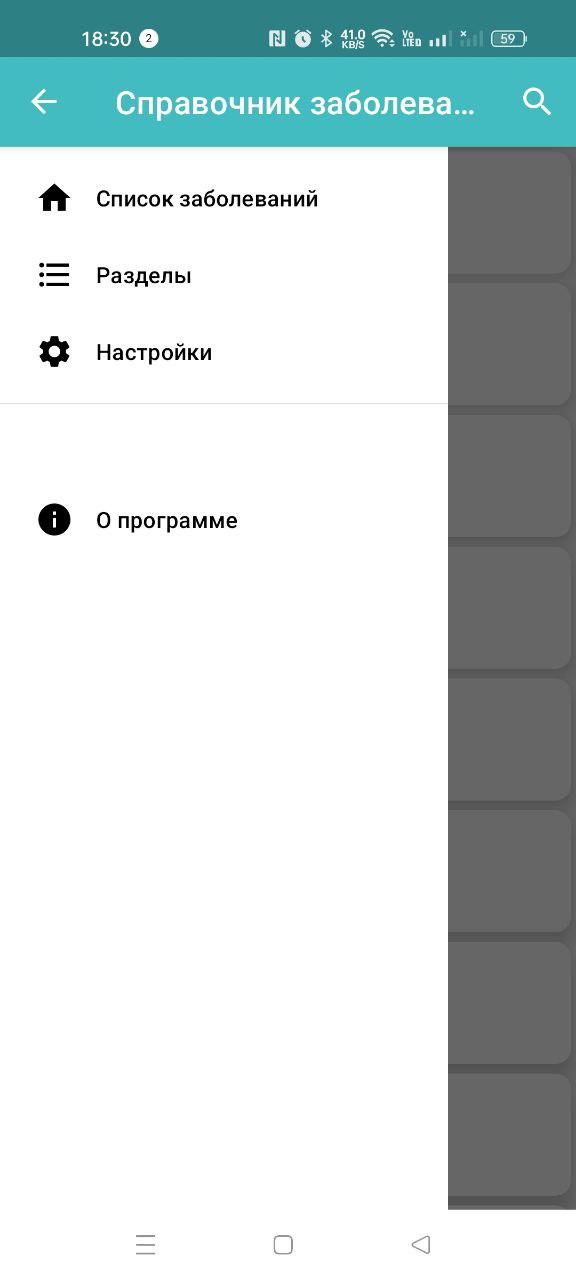


Рисунок 8 – Навигационное меню

Навигационное меню содержит:

* «Список Заболеваний» (переход на окно «Список заболеваний»);
* «Разделы» (переход на страницу со списком отделений);
* «Настройки» (переход на страницу с настройками);
* «О программе» (переход на страницу «О программе»).

Фрагмент с разделами предназначен для отображения информации о различных медицинских отделениях, к которым относятся определенные заболевания (рисунок 9). Для отображения раздела необходимо перейти к навигационному меню, нажав на иконку меню в левом углу, и нажать на элемент «Разделы». Для каждого отделения отображается список заболеваний, которые относятся к этому отделению. Это позволяет пользователям быстро и легко найти информацию о заболеваниях, связанных отделением. Чтобы отобразить список заболеваний по разделу, нажмите на элемент раздела из списка разделов. После выбора раздела будут отображены заболевания, относящиеся к выбранному разделу.

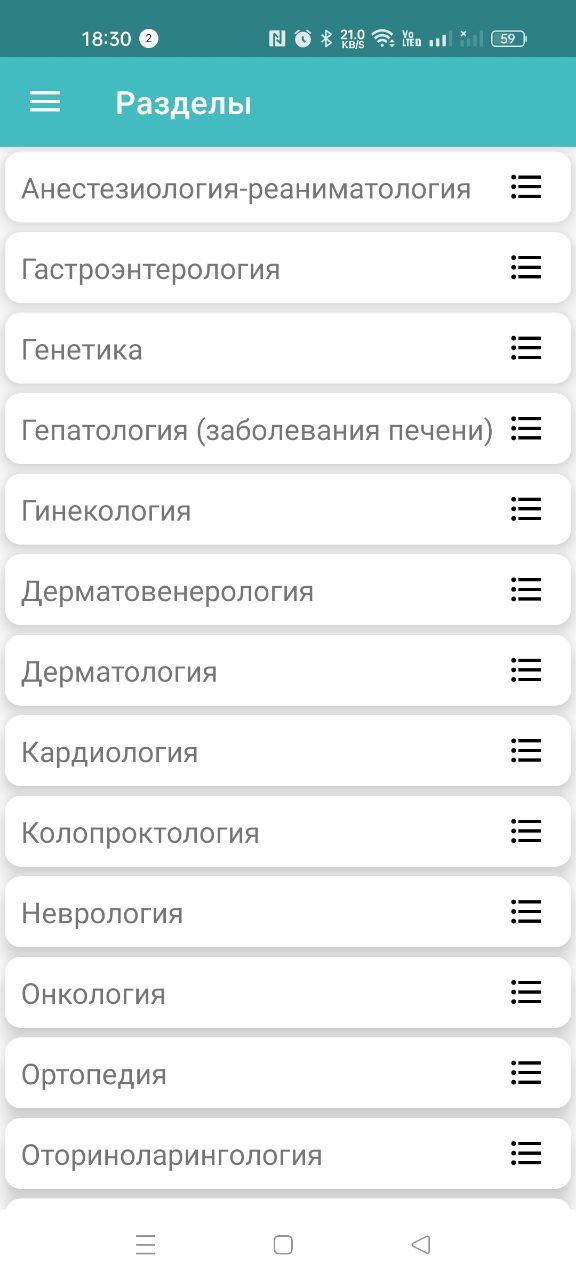
.

Рисунок 9 – Страница отделений

Для Android версии приложения доступна возможность настроить тему приложения, шрифт и его размер. Для этого необходимо перейти на вкладку «Настройки» и воспользоваться настройками приложения (рисунок 10).

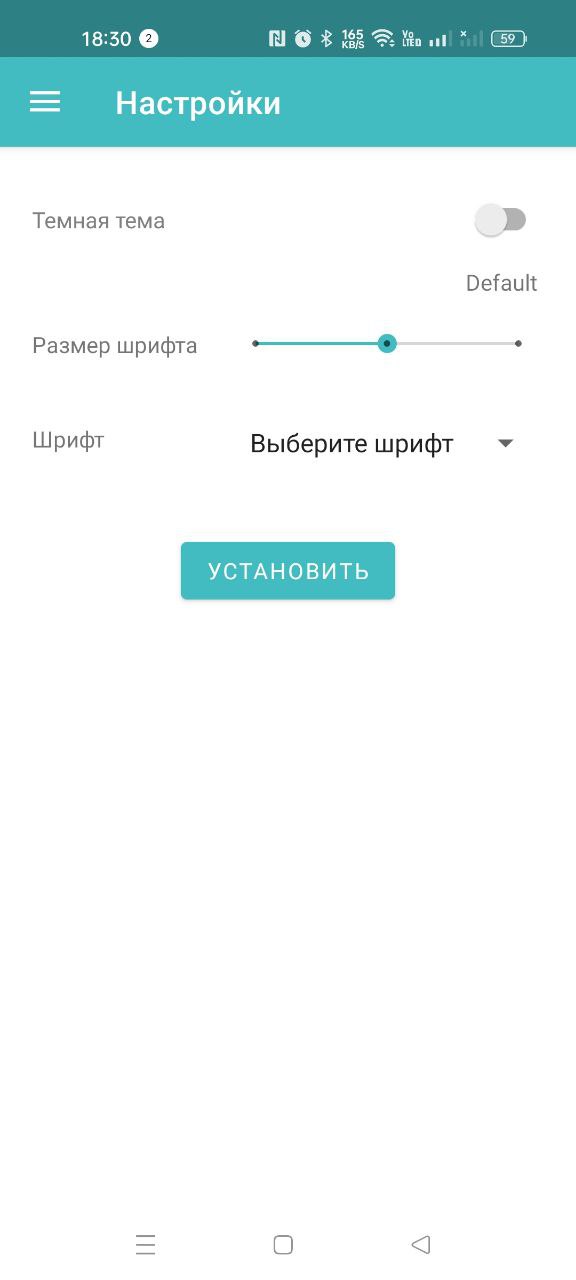


Рисунок 10 – Настройки приложения «Справочник Заболеваний»

Подробная информация о заболевании находится на фрагменте «Заболевание». Выше были представлены несколько способов получения искомого заболевания, а именно:

* получение путем выбора из списка во вкладке «Список Заболеваний»;
* получение в резутальте фильтрации по разделам во вкладке «Разделы»;
* получение при вводе в поисковой строке названия заболевания.

Фрагмент одного из заболеваний представлен на рисунке 11.

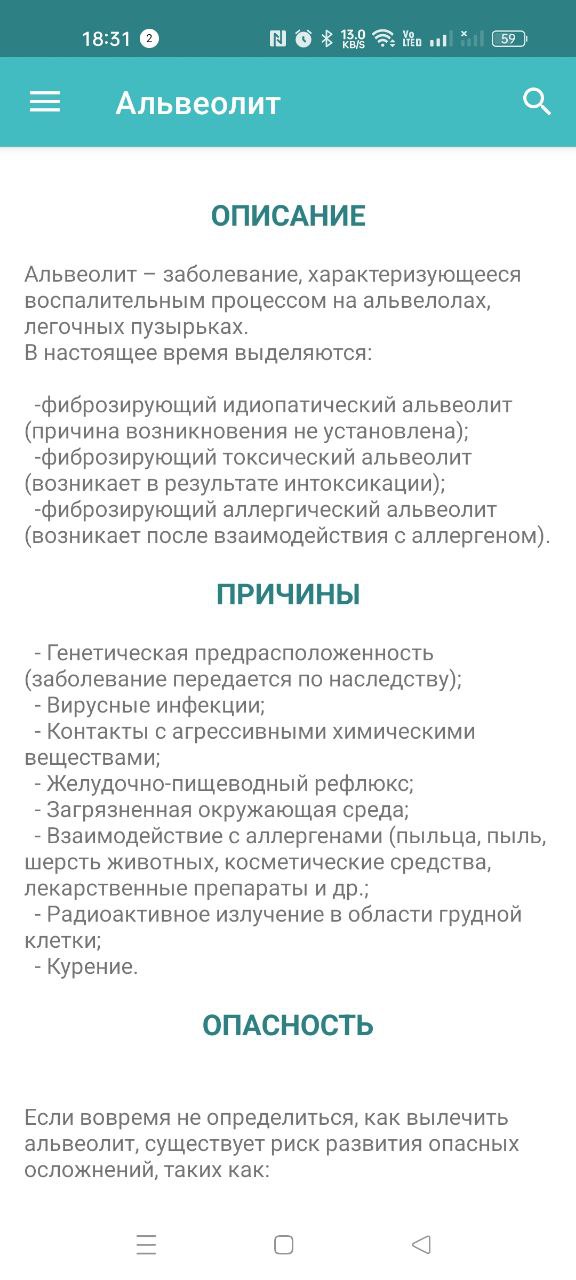


Рисунок 11 – Фрагмент заболевания

Фрагмент «Заболевание» содержит следующую информацию:

* название заболевания;
* описание и характеристики заболевания;
* специализация врача, который занимается лечением данного заболевания;
* отделение, в котором проводится лечение;
* причины возникновения заболевания;
* симптомы и признаки заболевания;
* классификация и стадии развития заболевания;
* диагностика и методы определения заболевания;
* опасности и возможные осложнения заболевания;
* профилактика и меры предосторожности;
* терапия и методы лечения заболевания;
* группа риска и кто может быть подвержен заболеванию;
* иллюстрация или фотография, связанная с заболеванием.

**ЗАКЛЮЧЕНИЕ**

В ходе выполнения курсового проекта была разработана БД для хранения информации, API для получения инфомарции с БД, desktop и android приложения для получения данных с БД через REST API. Также было разработано документальное сопровождение программы, включающее в себя технический и рабочий проекты. В ходе выполнения курсового проекта была достигнута поставленная цель и решены следующие задачи:

* выполнен анализ предметной области и спроектированы концептуальная и логическая модели БД для хранения необходимой информации;
* спроектирована функциональность и структура приложения, работающего с созданной БД;
* произведен выбор средств реализации БД, приложения, справочной системы и инсталляции;
* определены минимальные требования к техническим и программным средствам;
* создана физическая модель БД с помощью СУБД PostgreSQL и генерации таблиц БД на основе классов в API;
* с помощью средств интегрированной среды разработки IntelliJ IDEA и языка программирования Java была разработа REST API, позволяющая получить доступ к данным БД;
* с помощью средств интегрированной среды разработки Rider 2023 и языка программирования С# были разработаны модули, обеспечивающие выполнение всех необходимых функций;
* с помощью средств интегрированной среды разработки Rider 2023 и языка программирования С# был разработан интерфейс приложения, позволяющий получить доступ ко всем необходимым функциям;
* с помощью средств интегрированной среды разработки Android Studio 2023 и языка программирования Kotlin были разработаны модули, обеспечивающие выполнение всех необходимых функций;
* с помощью средств интегрированной среды разработки Android Studio 2023 и языка программирования Kotlin был разработан интерфейс приложения, позволяющий получить доступ ко всем необходимым функциям;
* создана инсталляция с помощью средств InstallMaker 5.04.

Полученные в результате анализа предметной области данные были распределены по таблицам, которые логически были связаны между собой. Основываясь на логической связи таблиц, строились запросы к БД.

Разработанная БД находится в 3НФ, что позволяет быстро и эффективно работать с информацией данной предметной области.

Для доступа к БД использовался REST API, написанный с использованием фремворка Spring на языке Java, который предоставляет различные операции для взаимодействия с БД.

Удобный интерфейс программ, с одной стороны, позволяет легко ориентироваться в приложении, не требуя от пользователя каких-либо специальных навыков работы, с другой стороны – предоставляет пользователю оперативную информацию.

Разработанные приложения обеспечивает следующие возможности:

* доступ к окнам и функциям системы через основное меню;
* получения списка заболеваний;
* поиск заболевания;
* фильтрацию заболеваний по отделениям;
* получение подробной информации о заболевании;
* работу с информационной БД.

Основные достоинства разработанного приложения:

* низкая требовательность к ресурсам системы;
* простота установки и настройки;
* удобство эксплуатации;
* возможность оперативного получения необходимых информации о заболевания.

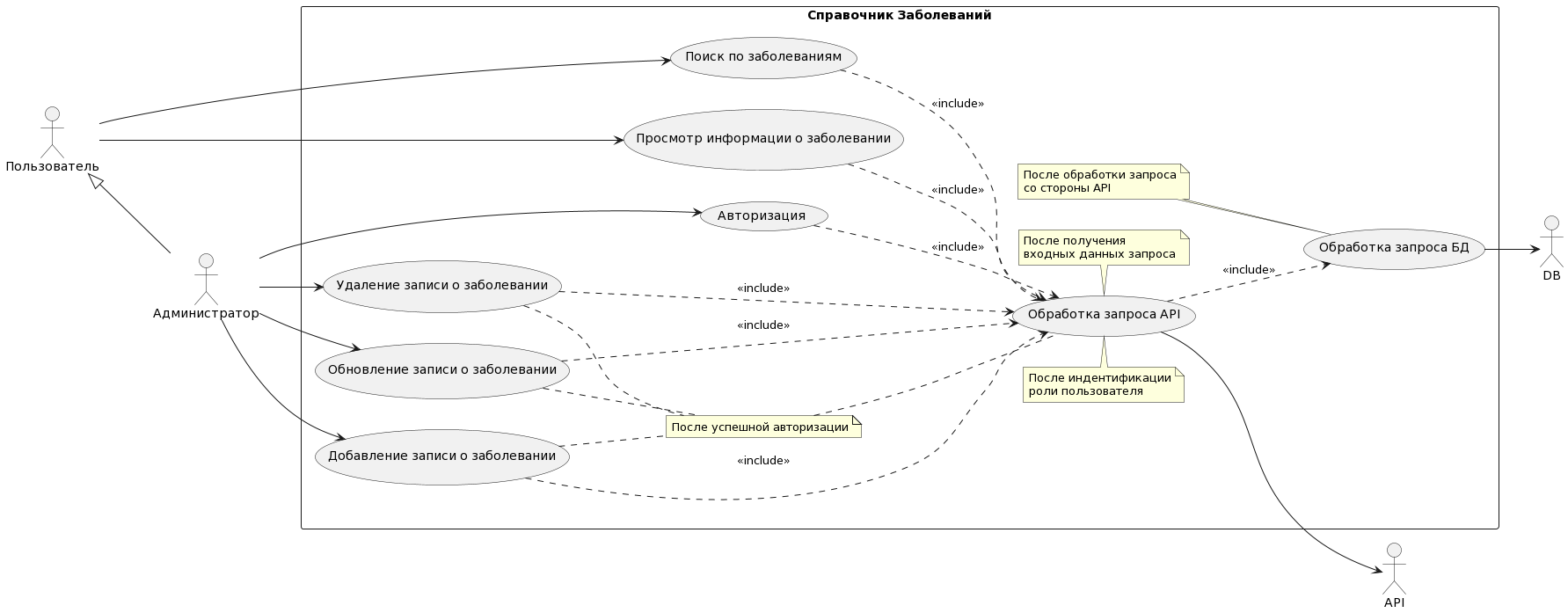
К недостаткам данного приложения можно отнести отсутствие графического приложения для роли администратора для добавления, изменения и удаления данных с БД. В будущем это планируется исправить.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Джуба, С. Изучаем PostgreSQL 10 / С. Джуба, А. Волков. – Москва: ДМК Пресс, 2019. – 129 c.
2. Крейг, У. Spring в действии / У. Крейг. – Москва: ДМК Пресс, 2022. – 196c.
3. Изучение библиотеки Material Design // Github.com: [сайт]. – URL: https://github.com/MaterialDesignInXAML/MaterialDesignInXamlToolkit (дата обращения: 18.12.2023).
4. Изучение документации AndroidX // Android.com: [сайт]. – URL: https:// developer.android.com/reference/kotlin/androidx/classes (дата обращения: 10.12.2023).
5. Изучение документации Hibernate для Spring Framework // Spring.io: [сайт]. – URL: https://docs.spring.io/spring-framework/reference/data-access/orm/ hibernate.html (дата обращения: 01.12.2023).
6. Изучение документации WPF элементов // Microsoft.com: [сайт]. – URL: https://learn.microsoft.com/en-us/dotnet/desktop/wpf/controls/wpf-content-model? view=netframeworkdesktop-4.8 (дата обращения: 09.12.2023).
7. Изучение обработки исключений в Spring для REST API элементов // Baeldung.com: [сайт]. – URL: <https://www.baeldung.com/exception-handling-for-rest-with-spring> (дата обращения: 05.12.2023).
8. Подключение OpenAPI 3 // Baeldung.com: [сайт]. – URL: https://www. baeldung.com/spring-rest-openapi-documentation (дата обращения: 15.12.2023).
9. Руководство по Entity // Metanit.com: [сайт]. – URL: https://metanit.com/ sharp/entityframework/ (дата обращения: 15.11.2023).
10. Трехзвенная архитектура взаимодействия // JavaRush.com: [сайт]. – URL: <https://javarush.com/quests/lectures/questservlets.level14.lecture01> (дата обращения: 01.12.2023).
11. Энциклопедия заболеваний // Probolezny.com: [сайт]. – URL: <https://probolezny.ru/> (дата обращения: 01.12.2023)

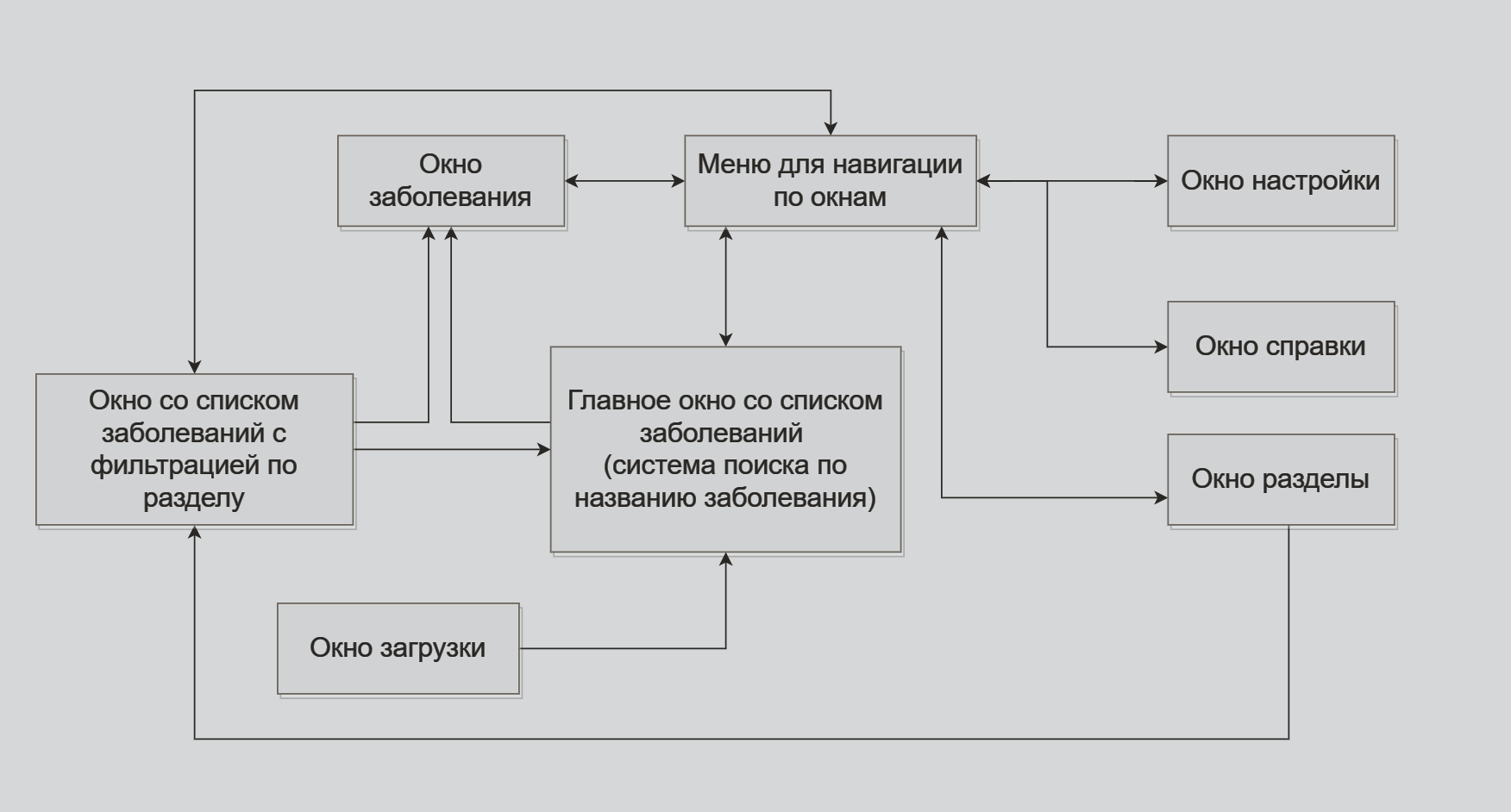
**ПРИЛОЖЕНИЕ А  
(обязательное)**

**Диаграмма Use Case для проектируемого портала**



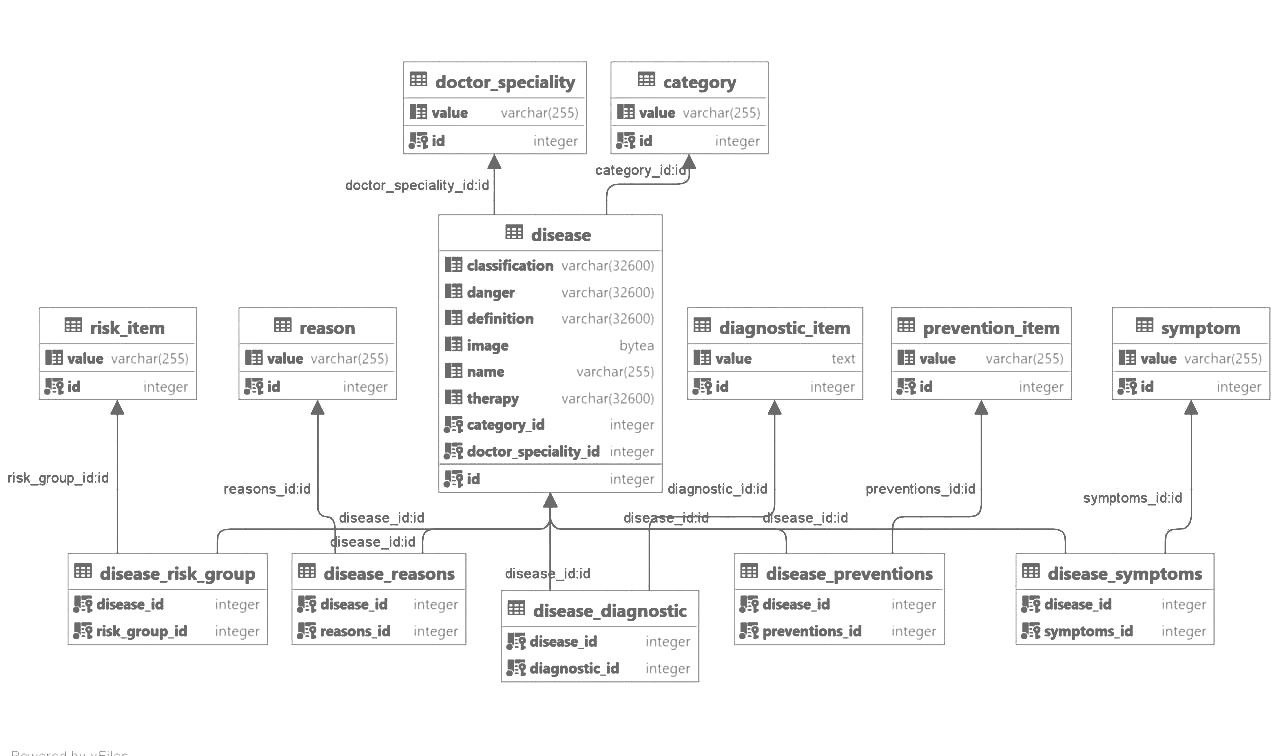
**ПРИЛОЖЕНИЕ Б  
(обязательное)**

**Схема проектируемой структуры клиентских Desktop и Android приложений**



**ПРИЛОЖЕНИЕ В  
(обязательное)**

**Логическая модель базы данных**



**ПРИЛОЖЕНИЕ Г  
(справочное)**

**Листинг модуля DiseaseController.java**

@RestController

@RequiredArgsConstructor

public class DiseaseController {

private final DiseaseService diseaseService;

@Operation(summary = "Получение списка заболеваний")

@GetMapping("/diseases")

public Iterable<Disease> getAll() {

return diseaseService.getAll();

}

@Operation(summary = "Получение заболевания по id")

@GetMapping("/diseases/{id}")

public ResponseEntity<Disease> getById(@PathVariable int id) {

Disease disease = diseaseService.getById(id);

return new ResponseEntity<>(disease, HttpStatus.OK);

}

@Operation(summary = "Получение заболевания по названию")

@GetMapping("/diseases/name/{name}")

@SneakyThrows

public ResponseEntity<Iterable<Disease>> getByNameContaining(@PathVariable String name) {

return new ResponseEntity<>(diseaseService.getByNameContainingIgnoreCase(name), HttpStatus.OK);

}

@Operation(summary = "Получение заболевания начинающегося на входящий параметр")

@GetMapping("/diseases/name/start/{name}")

@SneakyThrows

public ResponseEntity<Iterable<Disease>> getByNameStartWith(@PathVariable String name) {

return new ResponseEntity<>(diseaseService.getByNameStartWith(name), HttpStatus.OK);

}

@Operation(summary = "Создание заболевания")

@PostMapping("/diseases")

@SneakyThrows

public ResponseEntity<?> create(@RequestBody DiseaseRequest diseaseRequest) {

diseaseService.create(diseaseRequest);

return new ResponseEntity<>(HttpStatus.CREATED);

}

@Operation(summary = "Изменение заболевания")

@PutMapping("/diseases/{id}")

@SneakyThrows

public ResponseEntity<?> update(@PathVariable int id, @RequestBody Disease disease) {

diseaseService.update(id, disease);

return new ResponseEntity<>(HttpStatus.OK);

}

@Operation(summary = "Удаление заболевания")

@DeleteMapping("/diseases/{id}")

@SneakyThrows

public ResponseEntity<?> delete(@PathVariable int id) {

diseaseService.delete(id);

return new ResponseEntity<>(HttpStatus.OK);

}

@Operation(summary = "Добавление иллюстрации к заболеванию")

@PostMapping("/diseases/upload-image/{id}")

@SneakyThrows

public ResponseEntity<?> uploadImage(

@PathVariable int id,

@RequestBody MultipartFile file

){

diseaseService.uploadImage(id, file);

return new ResponseEntity<>(HttpStatus.CREATED);

}

}

**ПРИЛОЖЕНИЕ Д  
(справочное)**

**Содержание документации API**



**ПРИЛОЖЕНИЕ Е  
(справочное)**

**Листинг модуля HttpHelper.cs**

public static class HttpHelper{

private const string Url = "http://localhost:8080/";

private const string UrlGetDiseases = Url + "diseases";

private const string UrlGetCategories = Url + "categories";

private const string UrlGetDiseasesStartWith = Url + "diseases/name/start/";

private const string UrlGetDiseasesName = Url + "diseases/name/";

private static readonly HttpClient s\_client;

static HttpHelper()

{

s\_client = new HttpClient();

}

public static async Task<IEnumerable<Disease>?> GetDiseasesAsync()

{

Try

{

var response = await s\_client.GetAsync(UrlGetDiseases);

return JsonConvert.DeserializeObject<IEnumerable<Disease>>(await response.Content.ReadAsStringAsync());

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

return null;

}

}

public static async Task<IEnumerable<Disease>?> GetDiseasesStartNameAsync(string startName)

{

Try

{

var response = await s\_client.GetAsync(UrlGetDiseasesStartWith + startName);

return JsonConvert.DeserializeObject<IEnumerable<Disease>>(await response.Content.ReadAsStringAsync());

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);  
 return null;

}

}

public static async Task<IEnumerable<Disease>?> GetDiseasesNameAsync(string name)

{

Try

{

var response = await s\_client.GetAsync(UrlGetDiseasesName + name);

return JsonConvert.DeserializeObject<IEnumerable<Disease>>(await response.Content.ReadAsStringAsync());

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);  
 return null;

}

}

public static async Task<IEnumerable<Category>?> GetCategoriesAsync()

{

Try

{

var response = await s\_client.GetAsync(UrlGetCategories);

return JsonConvert.DeserializeObject<IEnumerable<Category>>(await response.Content.ReadAsStringAsync());

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

return null;

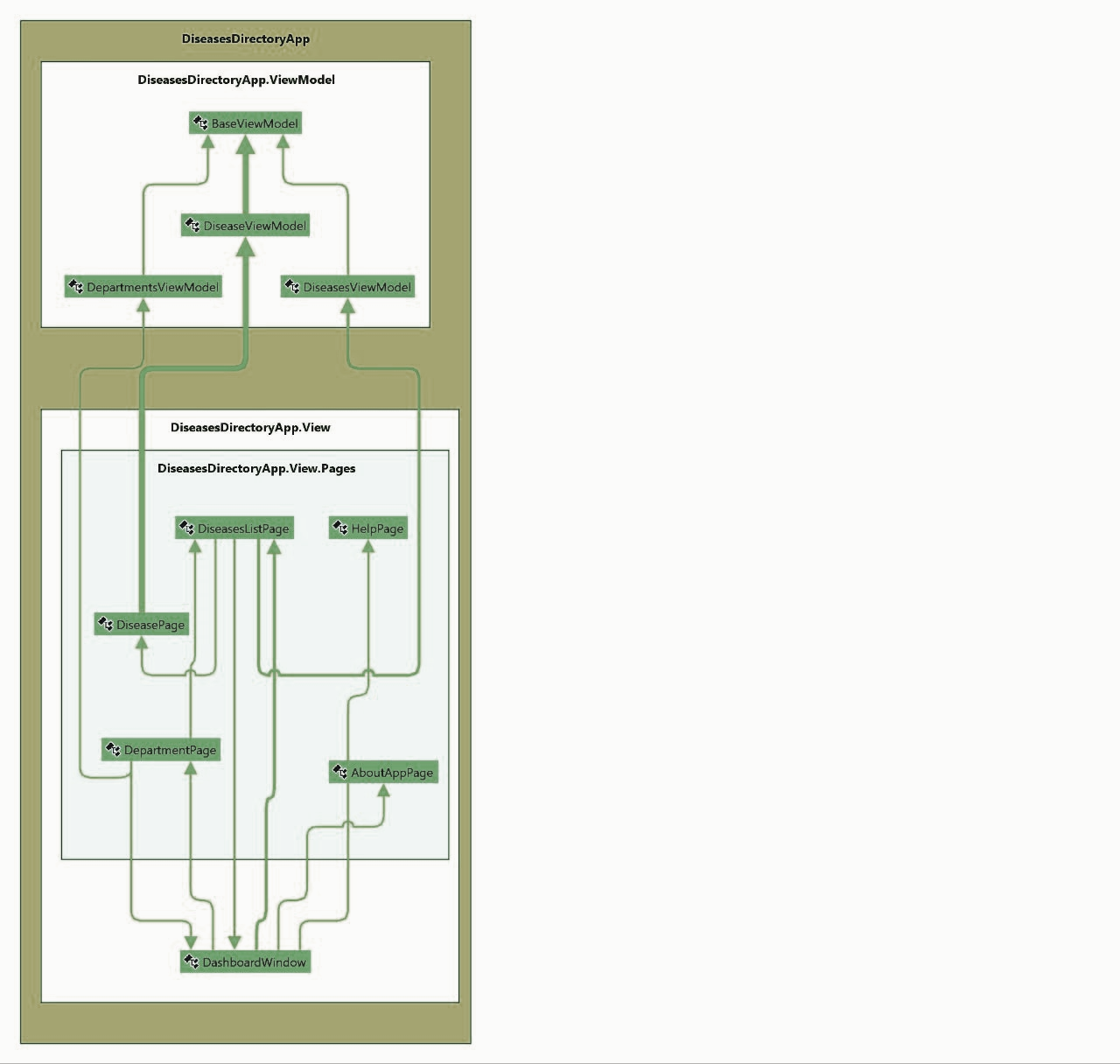
}

}

}

**ПРИЛОЖЕНИЕ Ж  
(обязательное)**

**Схема функционального взаимодействия модулей   
desktop-приложения**



**ПРИЛОЖЕНИЕ И  
(обязательное)**

**Схема функционального взаимодействия модулей android-приложения**

