# ОГЛАВЛЕНИЕ

[ОГЛАВЛЕНИЕ 2](#_Toc133413095)

[ВВЕДЕНИЕ 3](#_Toc133413096)

[ГЛАВА 1. ВЫБОР ИНСТРУМЕНТАРИЯ 4](#_Toc133413097)

[1.1 Платформа .NET 4](#_Toc133413098)

[1.2 Язык программирования С# 5](#_Toc133413099)

[1.3 Windows Presentation Foundation (WPF) 7](#_Toc133413100)

[1.4 СУБД SQL server 8](#_Toc133413101)

[1.5 Microsoft SQL Server Management Studio 11](#_Toc133413102)

[1.6 Entity Framework 13](#_Toc133413103)

[ГЛАВА 2. ПРОЕТИРОВАНИЕ БАЗЫ ДАННЫХ 15](#_Toc133413104)

[2.1 Разработка диаграммы ERD 15](#_Toc133413105)

[2.2 Разработка базы данных 17](#_Toc133413106)

[ГЛАВА 3. РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ 21](#_Toc133413107)

[3.1 Разработка интерфейса информационной системы. 21](#_Toc133413108)

[3.2 Программирование информационной системы 29](#_Toc133413109)

[ЗАКЛЮЧЕНИЕ 38](#_Toc133413110)

[СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 39](#_Toc133413111)

# ВВЕДЕНИЕ

Аптека - особая специализированная организация системы здравоохранения, занимающаяся изготовлением, фасовкой, анализом и продажей лекарственных средств.

Аптеку традиционно рассматривают как учреждение здравоохранения, а её деятельность формулируют как «оказание фармацевтической помощи населению». Фармацевтическая помощь включает в себя процедуру консультирования врача и пациента с целью определения наиболее эффективного, безопасного и экономически оправданного курса лечения.

Постоянный рост объемов информации, многообразие и противоречивость поступающих предложений, жесткая конкуренция на фармацевтическом рынке требуют от современных аптечных предприятий работать оперативно и эффективно.

Основная идея автоматизации аптек состоит в том, чтобы труд сотрудников не был монотонным и изнурительным, и у них не было просчетов, чтобы пациенты всегда имели нужное лекарство и не толпились в очередях. Хранение в бумажной форме колоссальных объемов информации ведет к различным ошибкам. Применение новейших технологий позволяет свести к минимуму риск их совершения, полностью систематизировать и автоматизировать работу аптечного предприятия, сделать ее максимально удобной и эффективной.

**Объектом** исследования является информационно-справочная система «Аптека».

**Предмет** данной курсовой работы - автоматизация аптечного бизнеса.

**Целью** работы является создание информационно-справочной системы «Аптека».

Для достижения цели проекта нужно выполнить следующие **задачи**:

• на основании анализа технологического задания выбрать инструментарий для разработки информационной системы;

• в соответствии с техническим заданием выполнить проектирование базы данных в выбранной среде программирования;

• выполнить разработку информационной системы.

# ГЛАВА 1. ВЫБОР ИНСТРУМЕНТАРИЯ

# Платформа .NET

Платформа .NET Framework — это технология, которая поддерживает создание и выполнение веб-служб и приложений Windows.

Платформа .NET Framework состоит из общеязыковой среды выполнения (среды CLR) и библиотеки классов .NET Framework. Основой платформы .NET Framework является среда CLR. Среду выполнения можно считать агентом, который управляет кодом во время выполнения и предоставляет основные службы, такие как управление памятью, управление потоками и удаленное взаимодействие. При этом средой накладываются условия строгой типизации и другие виды проверки точности кода, обеспечивающие безопасность и надежность. Фактически основной задачей среды выполнения является управление кодом. Код, который обращается к среде выполнения, называют управляемым кодом, а код, который не обращается к среде выполнения, называют неуправляемым кодом. Библиотека классов является комплексной объектно-ориентированной коллекцией повторно используемых типов, которые применяются для разработки приложений — начиная с обычных приложений, запускаемых из командной строки, и приложений с графическим интерфейсом (GUI) и заканчивая приложениями, использующими последние технологические возможности ASP.NET, такие как веб-формы и веб-службы XML.

Платформа .NET Framework может размещаться неуправляемыми компонентами, которые загружают среду CLR в собственные процессы и запускают выполнение управляемого кода, создавая таким образом программную среду, позволяющую использовать средства как управляемого, так и неуправляемого выполнения. Платформа .NET Framework не только предоставляет несколько базовых сред выполнения, но также поддерживает разработку базовых сред выполнения независимыми производителями.

Например, ASP.NET размещает среду выполнения и обеспечивает масштабируемую среду для управляемого кода на стороне сервера. ASP.NET работает непосредственно со средой выполнения, чтобы обеспечить выполнение приложений ASP.NET и веб-служб XML, обсуждаемых ниже в этой статье.

Обозреватель Internet Explorer может служить примером неуправляемого приложения, размещающего среду выполнения. Размещение среды выполнения в обозревателе Internet Explorer позволяет внедрять управляемые компоненты или элементы управления Windows Forms в HTML-документы. Такое размещение среды позволяет выполнять управляемый мобильный код и пользоваться его существенными преимуществами, в частности выполнением в условиях неполного доверия и изолированным хранением файлов.

# Язык программирования С#

C# — современный объектно-ориентированный язык программирования. C# позволяет разработчикам создавать разные типы безопасных и надежных приложений, выполняющихся в .NET.

C# предоставляет языковые конструкции для непосредственной поддержки такой концепции работы. Благодаря этому C# подходит для создания и применения программных компонентов. С момента создания язык C# обогатился функциями для поддержки новых рабочих нагрузок и современными рекомендациями по разработке ПО.

C# относится к семье языков с [C-подобным синтаксисом](https://ru.wikipedia.org/wiki/C-%D0%BF%D0%BE%D0%B4%D0%BE%D0%B1%D0%BD%D1%8B%D0%B9_%D1%81%D0%B8%D0%BD%D1%82%D0%B0%D0%BA%D1%81%D0%B8%D1%81), из них его синтаксис наиболее близок к [C++](https://ru.wikipedia.org/wiki/C%2B%2B) и [Java](https://ru.wikipedia.org/wiki/Java). Язык имеет [статическую типизацию](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F), поддерживает [полиморфизм](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%B8%D0%BC%D0%BE%D1%80%D1%84%D0%B8%D0%B7%D0%BC_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)), [перегрузку операторов](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%B3%D1%80%D1%83%D0%B7%D0%BA%D0%B0_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BE%D0%B2) (в том числе операторов явного и неявного приведения типа), [делегаты](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%BB%D0%B5%D0%B3%D0%B0%D1%82_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), атрибуты, [события](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D0%B1%D1%8B%D1%82%D0%B8%D0%B9%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), [переменные](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), [свойства](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B9%D1%81%D1%82%D0%B2%D0%BE_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), [обобщённые](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D0%BE%D0%B1%D1%89%D1%91%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) типы и методы, [итераторы](https://ru.wikipedia.org/wiki/%D0%98%D1%82%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80), [анонимные функции](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%BE%D0%BD%D0%B8%D0%BC%D0%BD%D0%B0%D1%8F_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F) с поддержкой [замыканий](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%BC%D1%8B%D0%BA%D0%B0%D0%BD%D0%B8%D0%B5_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), [LINQ](https://ru.wikipedia.org/wiki/Language_Integrated_Query), [исключения](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D0%B8%D1%81%D0%BA%D0%BB%D1%8E%D1%87%D0%B5%D0%BD%D0%B8%D0%B9), [комментарии](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BC%D0%B5%D0%BD%D1%82%D0%B0%D1%80%D0%B8%D0%B8_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) в формате [XML](https://ru.wikipedia.org/wiki/XML).

Переняв многое от своих предшественников — языков [C++](https://ru.wikipedia.org/wiki/C%2B%2B), [Delphi](https://ru.wikipedia.org/wiki/Delphi_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)), [Модула](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D1%83%D0%BB%D0%B0-2), [Smalltalk](https://ru.wikipedia.org/wiki/Smalltalk" \o "Smalltalk) и, в особенности, [Java](https://ru.wikipedia.org/wiki/Java) — С#, опираясь на практику их использования, исключает некоторые модели, зарекомендовавшие себя как проблематичные при разработке программных систем, например, C# в отличие от C++ не поддерживает [множественное наследование](https://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BD%D0%B0%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) классов (между тем допускается [множественная реализация интерфейсов](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_(%D0%BE%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)#%D0%9C%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BD%D0%B0%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%B8_%D1%80%D0%B5%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81%D0%BE%D0%B2)).

C# уже не молодой язык и как и вся платформа .NET уже прошел большой путь. Первая версия языка вышла вместе с релизом Microsoft Visual Studio .NET в феврале 2002 года. Текущей версией языка является версия C# 11, которая вышла 8 ноября 2022 года вместе с релизом .NET 7.

C# является языком с Си-подобным синтаксисом и близок в этом отношении к C++ и Java. Поэтому, если вы знакомы с одним из этих языков, то овладеть C# будет легче.

C# является объектно-ориентированным и в этом плане много перенял у Java и С++. Например, C# поддерживает полиморфизм, наследование, перегрузку операторов, статическую типизацию. Объектно-ориентированный подход позволяет решить задачи по построению крупных, но в тоже время гибких, масштабируемых и расширяемых приложений. И C# продолжает активно развиваться, и с каждой новой версией появляется все больше интересных функциональностей.

Когда говорят C#, нередко имеют в виду технологии платформы .NET (Windows Forms, WPF, ASP.NET, Xamarin). И, наоборот, когда говорят .NET, нередко имеют в виду C#. Однако, хотя эти понятия связаны, отождествлять их неверно. Язык C# был создан специально для работы с фреймворком .NET, однако само понятие .NET несколько шире.

Как-то Билл Гейтс сказал, что платформа .NET - это лучшее, что создала компания Microsoft. Возможно, он был прав. Фреймворк .NET представляет мощную платформу для создания приложений. Можно выделить следующие ее основные черты:

Поддержка нескольких языков. Основой платформы является общеязыковая среда исполнения Common Language Runtime (CLR), благодаря чему .NET поддерживает несколько языков: наряду с C# это также VB.NET, C++, F#, а также различные диалекты других языков, привязанные к .NET, например, Delphi.NET. При компиляции код на любом из этих языков компилируется в сборку на общем языке CIL (Common Intermediate Language) - своего рода ассемблер платформы .NET. Поэтому при определенных условиях мы можем сделать отдельные модули одного приложения на отдельных языках.

Кроссплатформенность. .NET является переносимой платформой (с некоторыми ограничениями). Например, последняя версия платформы на данный момент - .NET 7 поддерживается на большинстве современных ОС Windows, MacOS, Linux. Используя различные технологии на платформе .NET, можно разрабатывать приложения на языке C# для самых разных платформ - Windows, MacOS, Linux, Android, iOS, Tizen.

Мощная библиотека классов. .NET представляет единую для всех поддерживаемых языков библиотеку классов. И какое бы приложение мы не собирались писать на C# - текстовый редактор, чат или сложный веб-сайт - так или иначе мы задействуем библиотеку классов .NET.

Разнообразие технологий. Общеязыковая среда исполнения CLR и базовая библиотека классов являются основой для целого стека технологий, которые разработчики могут задействовать при построении тех или иных приложений. Например, для работы с базами данных в этом стеке технологий предназначена технология ADO.NET и Entity Framework Core. Для построения графических приложений с богатым насыщенным интерфейсом - технология WPF и WinUI, для создания более простых графических приложений - Windows Forms. Для разработки кроссплатформенных мобильных и десктопных приложений - Xamarin/MAUI. Для создания веб-сайтов и веб-приложений - ASP.NET и т.д.

К этому стоит добавить активной развивающийся и набирающий популяность Blazor - фреймворк, который работает поверх .NET и который позволяет создавать веб-приложения как на стороне сервера, так и на стороне клиента. А в будущем будет поддерживать создание мобильных приложений и, возможно, десктоп-приложений.

Производительность. Согласно ряду тестов веб-приложения на .NET 7 в ряде категорий сильно опережают веб-приложения, построенные с помощью других технологий. Приложения на .NET 7 в принципе отличаются высокой производительностью.

Также еще следует отметить такую особенность языка C# и фреймворка .NET, как автоматическая сборка мусора. А это значит, что нам в большинстве случаев не придется, в отличие от С++, заботиться об освобождении памяти. Вышеупомянутая общеязыковая среда CLR сама вызовет сборщик мусора и очистит память.

# Windows Presentation Foundation (WPF)

Технология WPF (Windows Presentation Foundation) является часть экосистемы платформы .NET и представляет собой подсистему для построения графических интерфейсов.

Если при создании традиционных приложений на основе WinForms за отрисовку элементов управления и графики отвечали такие части ОС Windows, как User32 и GDI+, то приложения WPF основаны на DirectX. В этом состоит ключевая особенность рендеринга графики в WPF: используя WPF, значительная часть работы по отрисовке графики, как простейших кнопочек, так и сложных 3D-моделей, ложиться на графический процессор на видеокарте, что также позволяет воспользоваться аппаратным ускорением графики.

Одной из важных особенностей является использование языка декларативной разметки интерфейса XAML, основанного на XML: вы можете создавать насыщенный графический интерфейс, используя или декларативное объявление интерфейса, или код на управляемых языках C#, VB.NET и F#, либо совмещать и то, и другое.

Первая версия - WPF 3.0 вышла вместе с .NET Framework 3.0 и операционной системой Windows Vista в 2006 году. И с тех пор платформа WPF является частью экосистемы .NET и развивается вместе с фреймворком .NET. Например, на сегодняшний день последней версией фреймворка .NET является .NET 7, и WPF полностью поддерживается этой версией фреймворка.

В основе WPF лежит векторная система визуализации, не зависящая от разрешения устройства вывода и созданная с учётом возможностей современного графического оборудования. WPF предоставляет средства для создания визуального интерфейса, включая язык XAML (Extensible Application Markup Language), элементы управления, привязку данных, макеты, двухмерную и трёхмерную графику, анимацию, стили, шаблоны, документы, текст, мультимедиа и оформление.

Графической технологией, лежащей в основе WPF, является DirectX, в отличие от Windows Forms, где используется GDI/GDI+. Производительность WPF выше, чем у GDI+ за счёт использования аппаратного ускорения графики через DirectX.

Также существует урезанная версия CLR, называющаяся WPF/E, она же известна как Silverlight.

Использование разметки XAML

XAML представляет собой XML, в котором фактически реализованы классы .NET Framework. Также реализована модель разделения кода и дизайна, позволяющая кооперироваться программисту и дизайнеру. Кроме того, есть встроенная поддержка стилей элементов, а сами элементы легко разделить на элементы управления второго уровня, которые, в свою очередь, разделяются до уровня векторных фигур и свойств/действий. Это позволяет легко задать стиль для любого элемента, например, Button (кнопка).

Средства разработки

Для работы с WPF требуется любой .NET-совместимый язык. В этот список входит множество языков: C#, VB, C++, Ruby, Python, Delphi (Prism), Lua и многие другие. Для полноценной работы может быть использована как Visual Studio, так и Expression Blend. Первая ориентирована на программирование, а вторая — на дизайн и позволяет делать многие вещи, не прибегая к ручному редактированию XAML. Примеры этому — анимация, стилизация, состояния, создание элементов управления и так далее.

# СУБД SQL server

QL Server является одной из наиболее популярных систем управления базами данных (СУБД) в мире. Данная СУБД подходит для самых различных проектов: от небольших приложений до больших высоконагруженных проектов.

SQL Server был создан компанией Microsoft. Первая версия вышла в 1987 году. А текущей версией является версия 2022, которая вышла в ноябре 2022 году и которая будет использоваться в текущем руководстве.

SQL Server долгое время был исключительно системой управления базами данных для Windows, однако начиная с версии 16 эта система доступна и на Linux.

SQL Server характеризуется такими особенностями как:

* Производительность. SQL Server работает очень быстро.
* Надежность и безопасность. SQL Server предоставляет шифрование данных.
* Простота. С данной СУБД относительно легко работать и вести администрирование.
* Центральным аспектом в MS SQL Server, как и в любой СУБД, является база данных.

База данных представляет хранилище данных, организованных определенным способом. Нередко физически база данных представляет файл на жестком диске, хотя такое соответствие необязательно. Для хранения и администрирования баз данных применяются системы управления базами данных (database management system) или СУБД (DBMS). И как раз MS SQL Server является одной из такой СУБД.

Для организации баз данных MS SQL Server использует реляционную модель. Эта модель баз данных была разработана еще в 1970 году Эдгаром Коддом. А на сегодняшний день она фактически является стандартом для организации баз данных.

Реляционная модель предполагает хранение данных в виде таблиц, каждая из которых состоит из строк и столбцов. Каждая строка хранит отдельный объект, а в столбцах размещаются атрибуты этого объекта.

Для идентификации каждой строки в рамках таблицы применяется первичный ключ (primary key). В качестве первичного ключа может выступать один или несколько столбцов. Используя первичный ключ, мы можем ссылаться на определенную строку в таблице. Соответственно две строки не могут иметь один и тот же первичный ключ.

Через ключи одна таблица может быть связана с другой, то есть между двумя таблицами могут быть организованы связи. А сама таблица может быть представлена в виде отношения ("relation").

Для взаимодействия с базой данных применяется язык SQL (Structured Query Language). Клиент (например, внешняя программа) отправляет запрос на языке SQL посредством специального API. СУБД должным образом интерпретирует и выполняет запрос, а затем посылает клиенту результат выполнения.

Изначально язык SQL был разработан в компании IBM для системы баз данных, которая называлась System/R. При этом сам язык назывался SEQUEL (Structured English Query Language). Хотя в итоге ни база данных, ни сам язык не были впоследствии официально опубликованы, по традиции сам термин SQL нередко произносят как "сиквел".

В 1979 году компания Relational Software Inc. разработала первую систему управления баз данных, которая называлась Oracle и которая использовала язык SQL. В связи с успехом данного продукта компания была переименована в Oracle.

Впоследствии стали появляться другие системы баз данных, которые использовали SQL. В итоге в 1989 году Американский Национальный Институт Стандартов (ANSI) кодифицировал язык и опубликовал его первый стандарт. После этого стандарт периодически обновлялся и дополнялся. Последнее его обновление состоялось в 2011 году. Но несмотря на наличие стандарта нередко производители СУБД используют свои собственные реализации языка SQL, которые немного отличаются друг от друга.

Выделяются две разновидности языка SQL: PL-SQL и T-SQL. PL-SQL используется в таких СУБД как Oracle и MySQL. T-SQL (Transact-SQL) применяется в SQL Server. Собственно поэтому в рамках текущего руководства будет рассматриваться именно T-SQL.

В зависимости от задачи, которую выполняет команда T-SQL, он может принадлежать к одному из следующих типов:

DDL (Data Definition Language / Язык определения данных). К этому типу относятся различные команды, которые создают базу данных, таблицы, индексы, хранимые процедуры и т.д. В общем определяют данные.

В частности, к этому типу мы можем отнести следующие команды:

* CREATE: создает объекты базы данных (саму базу даных, таблицы, индексы и т.д.)
* ALTER: изменяет объекты базы данных
* DROP: удаляет объекты базы данных
* TRUNCATE: удаляет все данные из таблиц
* DML (Data Manipulation Language / Язык манипуляции данными). К этому типу относят команды на выбору данных, их обновление, добавление, удаление - в общем все те команды, с помощью которыми мы можем управлять данными.

К этому типу относятся следующие команды:

* SELECT: извлекает данные из БД
* UPDATE: обновляет данные
* INSERT: добавляет новые данные
* DELETE: удаляет данные
* DCL (Data Control Language / Язык управления доступа к данным). К этому типу относят команды, которые управляют правами по доступу к данным. В частности, это следующие команды:
* GRANT: предоставляет права для доступа к данным
* REVOKE: отзывает права на доступ к данным

# Microsoft SQL Server Management Studio

SQL Server Management Studio (SSMS) — утилита из [Microsoft SQL Server](https://ru.wikipedia.org/wiki/Microsoft_SQL_Server) 2005 и более поздних версий для конфигурирования, управления и [администрирования](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%BD%D0%BE%D0%B5_%D0%B0%D0%B4%D0%BC%D0%B8%D0%BD%D0%B8%D1%81%D1%82%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) всех компонентов Microsoft SQL Server. Утилита включает [скриптовый](https://ru.wikipedia.org/wiki/%D0%A1%D0%BA%D1%80%D0%B8%D0%BF%D1%82) редактор и графическую программу, которая работает с объектами и настройками сервера.

Среда SSMS предоставляет единую комплексную служебную программу, которая сочетает в себе обширную группу графических инструментов с рядом многофункциональных редакторов скриптов для доступа к SQL Server для разработчиков и администраторов баз данных всех профессиональных уровней.

Главным инструментом SQL Server Management Studio является Object Explorer, который позволяет пользователю просматривать, извлекать объекты сервера, а также полностью ими управлять.

Также есть SQL Server Management Studio Express для Express версии сервера, которая является бесплатной. Однако в ней нет поддержки ряда компонентов (Analysis Services, Integration Services, Notification Services, Reporting Services) и SQL Server 2005 Mobile Edition.

Сервер баз данных Microsoft SQL Server в качестве языка запросов использует версию языка SQL, получившую название Transact-SQL (сокращённо T-SQL). Язык T-SQL является реализацией SQL-92 (стандарт ISO для языка SQL) с множественными расширениями. T-SQL позволяет использовать дополнительный синтаксис для хранимых процедур и обеспечивает поддержку транзакций (взаимодействие базы данных с управляющим приложением).

При взаимодействии с сетью Microsoft SQL Server и Sybase ASE используют протокол уровня приложения под названием Tabular Data Stream (TDS, протокол передачи табличных данных). Протокол TDS также был реализован в проекте FreeTDS с целью обеспечить различным приложениям возможность взаимодействия с базами данных Microsoft SQL Server и Sybase.

Для обеспечения доступа к данным Microsoft SQL Server поддерживает Open Database Connectivity (ODBC) — интерфейс взаимодействия приложений с СУБД. Версия SQL Server 2005 обеспечивает возможность подключения пользователей через веб-сервисы, использующие протокол SOAP. Это позволяет клиентским программам, не предназначенным для Windows, кроссплатформенно соединяться с SQL Server. Компания Microsoft также выпустила сертифицированный драйвер JDBC, позволяющий приложениям под управлением Java (таким как BEA и IBM WebSphere) соединяться с Microsoft SQL Server 2000 и 2005.

Также SQL Server поддерживает зеркалирование и кластеризацию баз данных. Кластер сервера SQL — это совокупность одинаково конфигурированных серверов; такая схема помогает распределить рабочую нагрузку между несколькими серверами. Все сервера имеют одно виртуальное имя, и данные распределяются по IP-адресам машин кластера в течение рабочего цикла. Также в случае отказа или сбоя на одном из серверов кластера доступен автоматический перенос нагрузки на другой сервер.

SQL Server поддерживает избыточное дублирование данных по трем сценариям:

* Снимок: производится «снимок» базы данных, который сервер отправляет получателям.
* История изменений: все изменения базы данных непрерывно передаются пользователям.
* Синхронизация с другими серверами: Базы данных нескольких серверов синхронизируются между собой. Изменения всех баз данных происходят независимо друг от друга на каждом сервере, а при синхронизации происходит сверка данных. Данный тип дублирования предусматривает возможность разрешения противоречий между БД.

В SQL Server 2005 встроена поддержка .NET Framework. Благодаря этому хранимые процедуры БД могут быть написаны на любом языке платформы .NET, используя полный набор библиотек, доступных для .NET Framework, включая Common Type System (система обращения с типами данных в Microsoft .NET Framework). Однако, в отличие от других процессов, .NET Framework, будучи базисной системой для SQL Server 2005, выделяет дополнительную память и выстраивает средства управления SQL Server вместо того, чтобы использовать встроенные средства Windows. Это повышает производительность в сравнении с общими алгоритмами Windows, так как алгоритмы распределения ресурсов специально настроены для использования в структурах SQL Server.

Microsoft и другие компании производят большое число программных средств разработки, позволяющих разрабатывать бизнес-приложения с использованием баз данных Microsoft SQL Server. Microsoft SQL Server 2005 включает в себя также Common Language Runtime (CLR) Microsoft .NET, позволяющий реализовывать хранимые процедуры и различные функции приложениям, разработанным на языках платформы .NET (например, VB.NET или C#). Предыдущие версии средств разработки Microsoft использовали только API для получения функционального доступа к Microsoft SQL Server.

# Entity Framework

Entity Framework (EF) — объектно-ориентированная технология доступа к данным, является [object-relational mapping](https://ru.wikipedia.org/wiki/ORM) (ORM) решением для [.NET Framework](https://ru.wikipedia.org/wiki/.NET_Framework) от [Microsoft](https://ru.wikipedia.org/wiki/Microsoft). Предоставляет возможность взаимодействия с объектами как посредством [LINQ](https://ru.wikipedia.org/wiki/LINQ) в виде LINQ to Entities, так и с использованием Entity SQL. Для облегчения построения web-решений используется как [ADO.NET Data Services](https://ru.wikipedia.org/wiki/ADO.NET_Data_Services) (*[Astoria](https://ru.wikipedia.org/wiki/ADO.NET_Data_Services" \o "ADO.NET Data Services)*), так и связка из [Windows Communication Foundation](https://ru.wikipedia.org/wiki/Windows_Communication_Foundation) и [Windows Presentation Foundation](https://ru.wikipedia.org/wiki/Windows_Presentation_Foundation), позволяющая строить многоуровневые приложения, реализуя один из шаблонов проектирования [MVC](https://ru.wikipedia.org/wiki/Model-View-Controller), [MVP](https://ru.wikipedia.org/wiki/Model-View-Presenter) или [MVVM](https://ru.wikipedia.org/wiki/Model-View-ViewModel).

Entity Framework  [— это платформа ORM](https://en.wikipedia.org/wiki/Object-relational_mapping) с открытым исходным кодом для приложений .NET, поддерживаемая Microsoft. Это позволяет разработчикам работать с данными, используя объекты классов, специфичных для предметной области, не сосредотачиваясь на базовых таблицах и столбцах базы данных, в которых хранятся эти данные. С помощью Entity Framework разработчики могут работать с данными на более высоком уровне абстракции, а также создавать и поддерживать приложения, ориентированные на данные, с меньшим объемом кода по сравнению с традиционными приложениями.

Entity Framework использует сведения в файлах модели и сопоставления для преобразования запросов объектов в типы сущностей, представленные в концептуальной модели, в запросы, относящиеся к источникам данных. Результаты запроса собиваются на объекты, которыми управляет Entity Framework. Entity Framework предоставляет следующие способы запроса концептуальной модели и возврата объектов.

LINQ to Entities. Обеспечивает поддержку запросов LINQ для выполнения запросов к типам сущности, которые определены в концептуальной модели. дополнительные сведения см. в разделе LINQ to Entities.

Entity SQL. независимый от хранилища диалект SQL, который работает непосредственно с сущностями в концептуальной модели и поддерживает EDM концепции. Entity SQL используется как с запросами объектов, так и с запросами, которые выполняются с помощью поставщика EntityClient. дополнительные сведения см. в разделе Entity SQL обзор.

Entity Framework включает поставщик данных EntityClient. Этот поставщик управляет соединениями, преобразует запросы сущностей в запросы, зависящие от источника данных, и возвращает модуль чтения данных, который Entity Framework использует для материализации данных сущности в объекты. если материализация объекта не требуется, поставщик EntityClient можно также использовать как стандартный ADO.NET поставщик данных, позволяя приложениям выполнять запросы Entity SQL и использовать возвращенный модуль чтения данных только для чтения.

# ГЛАВА 2. ПРОЕТИРОВАНИЕ БАЗЫ ДАННЫХ

# 2.1 Разработка диаграммы ERD

Диаграмма ERD (Entity-Relationship Diagram) - это разновидность блок-схемы, где показано, как разные «сущности» (люди, объекты, концепции и так далее) связаны между собой внутри системы. ER-диаграммы чаще всего применяются для проектирования и отладки реляционных баз данных в сфере образования, исследования и разработки программного обеспечения и информационных систем для бизнеса. ER-диаграммы (или ER-модели) полагаются на стандартный набор символов, включая прямоугольники, ромбы, овалы и соединительные линии, для отображения сущностей, их атрибутов и связей. Эти диаграммы устроены по тому же принципу, что и грамматические структуры: сущности выполняют роль существительных, а связи — глаголов.

Базовыми понятиями ER-модели данных (ER — Entity— Relati-onship) являются сущность, атрибут и связь. Сущность — это класс однотипных реальных или абстрактных объектов (людей, событий, состояний, предметов и т.п.), информация о которых имеет существенное значение для рассматриваемой предметной области. Структурой данных называют совокупность правил и ограничений, которые отражают связи, существующие между отдельными частями (элементами) данных.

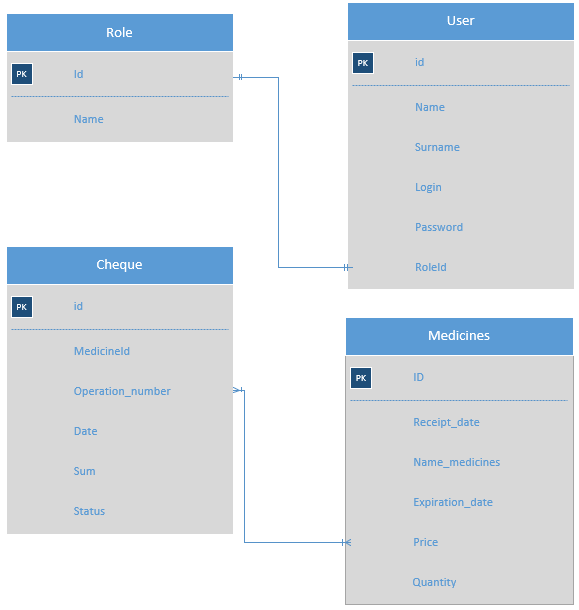
Атрибут — любая характеристика сущности, значимая для рассматриваемой предметной области и предназначенная для квалификации, идентификации, классификации, количественной характеристики или выражения состояния сущности. Атрибут, таким образом, представляет собой некоторый тип характеристик или свойств, ассоциированных с множеством реальных или абстрактных объектов. Экземпляр атрибута — определенная характеристика конкретного экземпляра сущности.

Атрибуты делятся на ключевые, т. е. входящие в состав уникального идентификатора ключа, и описательные — прочие.

Первичный ключ — это атрибут или совокупность атрибутов и (или) связей, предназначенная для уникальной идентификации каждого экземпляра сущности (совокупность признаков, позволяющих идентифицировать объект). Ключевые атрибуты помещают в начало списка и помечают символом «#».

Описательные атрибуты могут быть обязательными или необязательными. Обязательные атрибуты для каждой сущности всегда имеют конкретное значение, необязательные могут быть не определены. Обязательные и необязательные описательные атрибуты помечают символами «\*» и «°» соответственно.

Связь — это отношение одной сущности к другой или к самой себе. Каждая связь может иметь одну из двух модальностей связей. Если любой экземпляр одной сущности связан хотя бы с одним экземпляром другой сущности, то связь является обязательной. Необязательная связь представляет собой условное отношение между сущностями. Связь может иметь разную модальность с разных концов. Каждая сущность может обладать любым количеством связей с другими сущностями модели. Различают три типа отношений «один-к-одному»; «один-ко-многим»; «многие-ко-многим».

Рисунок 2.1 Диаграмма ERD

Основными сущностями для решения указанной задачи являются: Чек (Cheque), пользователь (User), роль (Role), медикаменты/лекарства (Medicines).

Проанализируем атрибуты сущностей: в чеке содержится номер операции, дата, сумма, статус. В медикаментах находится дата изготовления, название лекарства, срок годности, цена, количество. Пользователь имеет имя, фамилию, логин, пароль. В роли вписывается имя, где в дальнейшем будет разделение ролей на провизора и фармацевта.

Провизоры могут получают данные из системы и выводить их в виде отчёта. Фармацевты могут только просматривать данные.

# 2.2 Разработка базы данных

Подключение к sql-серверу через «microsoft sql server management studio express»

Чтобы получить доступ к базе данных Microsoft SQL Server, возможно использовать Microsoft SQL Server Management Studio Express. Для разработки базы данных в данном случае используется ранее разработанная ER-диаграмма.

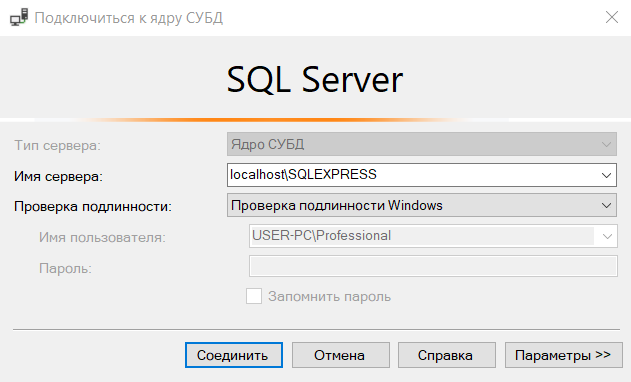


Рисунок 2.2.1. Окно подключения к локальному серверу

При нажатии кнопки “Соединить” программа подключается к локальному серверу.

После подключения к локальному серверу станет доступна меню “Обозреватель объектов”. Далее следует нажать ПКМ на «Базы данных» и «Создать базу данных».

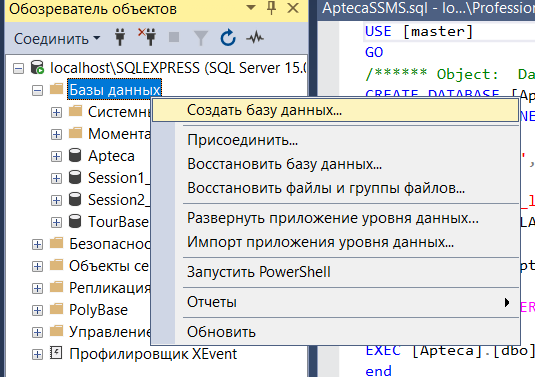


Рисунок 2.2.2. Создание базы данных

Нам выводиться окно, в котором мы даём название для базы данных (в нашем случае «Apteca») и нажимаем на кнопку OK.

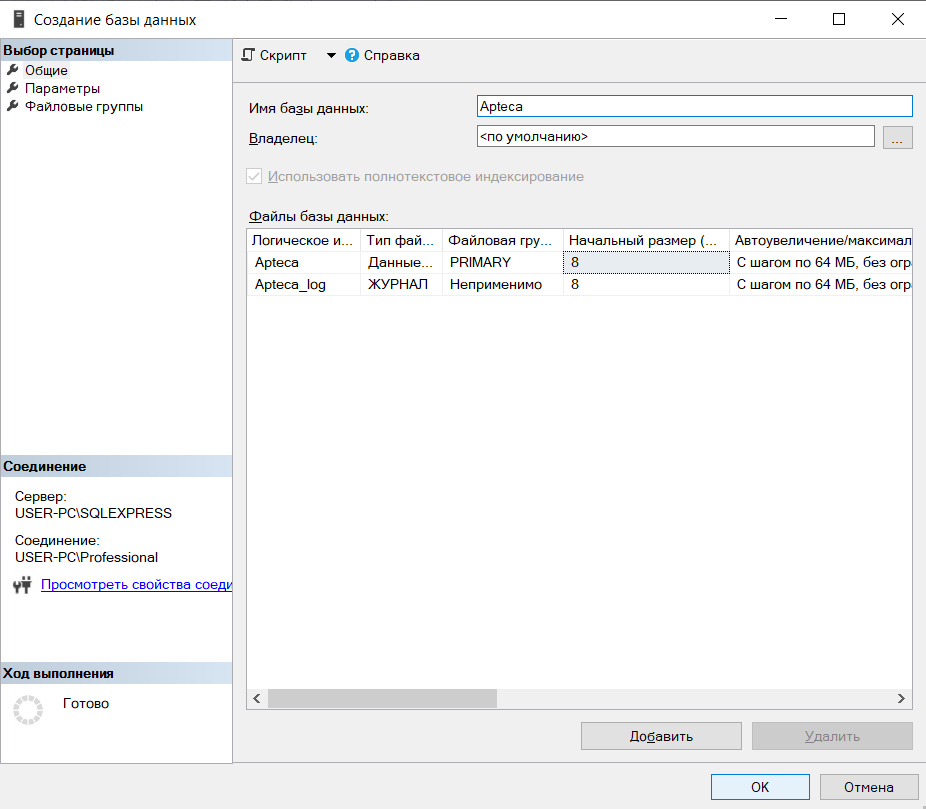


Рисунок 2.2.3. Название для БД

В Обозревателе объектов появится созданная база данных (Apteca), развернём её, нажмём ПКМ по «Диаграммы баз данных» и нажмём на «Создать диаграмму базы данных».

После чего нам выведется поле для создания диаграммы базы данных, на котором мы можем создавать таблицы с название столбцов и их типом данных.

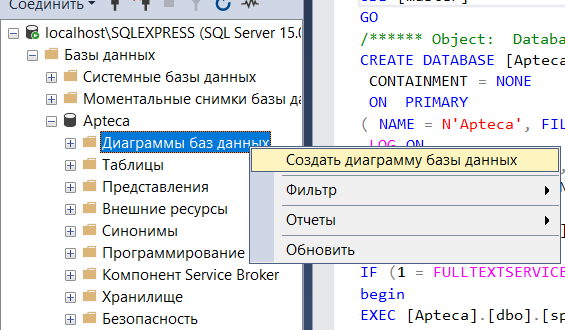


Рисунок 2.2.4. Создание диаграммы в БД

Для создания таблицы данных необходимо нажать ПКМ в любом месте диаграммы,

выбираем «Новая таблица» и даём название для нее. Для информационной системы необходимо создать четыре таблицы: Чек (Cheque), пользователь (User), роль (Role), медикаменты/лекарства (Medicines). После чего заполним таблицы именами столбцов из ERD диаграммы и типами данных.

Далее создаём связи как в ERD сущностях и после сохраняем таблицы нажав сочетание клавиш «Ctrl + S» и в высветившимся окне пишем название нашей диаграммы.

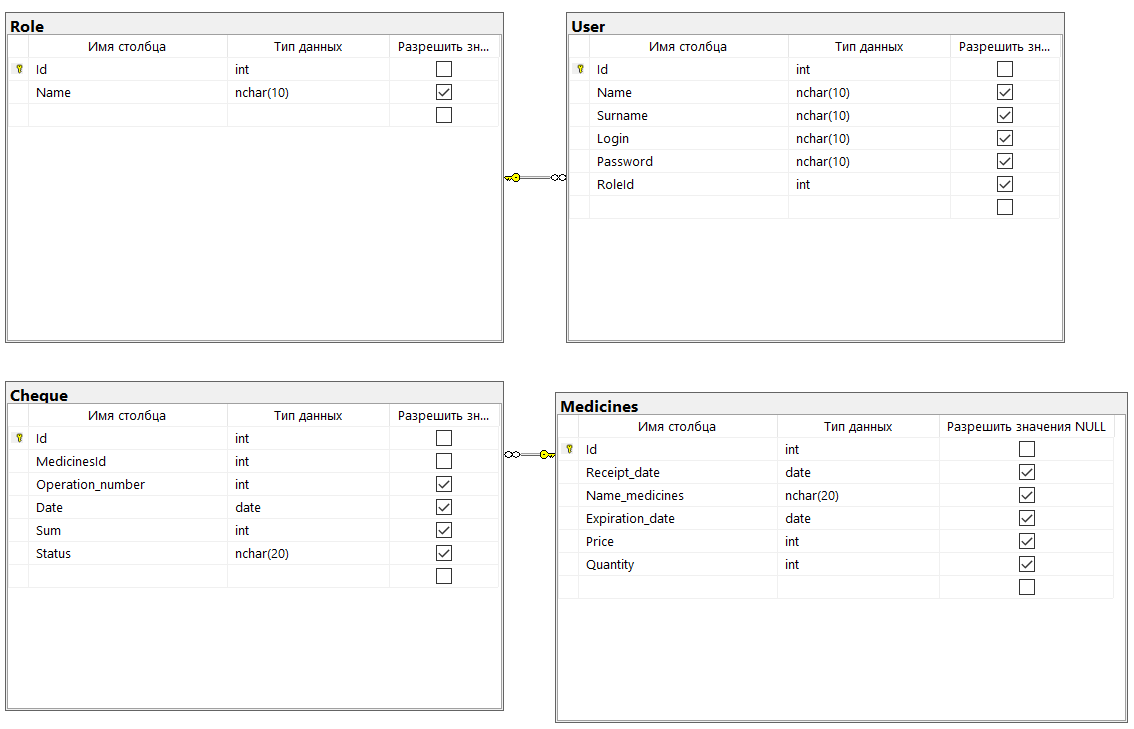


Рисунок 2.2.5. Диаграмма базы данных для ИС “Аптека”

Выбираем в папке «Таблицы» любую из сделанных нами таблиц и нажимаем по ней ПКМ и «Изменить первые 200 строк».

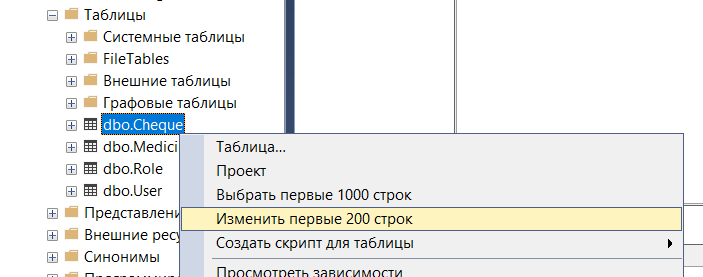


Рисунок 2.2.6. Изменение строк в таблицах

Далее заполняем таблицы информацией, которая будет отображаться в программе.

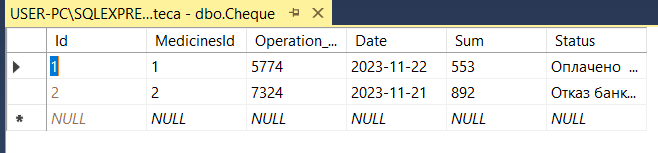


Рисунок 2.2.7. Заполнение таблиц

# ГЛАВА 3. РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ

# 3.1 Разработка интерфейса информационной системы.

Автоматизированная информационная система «Аптека» должна обеспечивать выполнение функций:

* вывод списка медикаментов, их количество, цена, дата изготовления и срока годности;
* вывод списка продаж, где находится статус оплаты, номер операции, стоимость и дата.

В программе необходимо предусмотреть возможность корректировки настроек системы; резервное сохранение данных; возможность изменения пароля входа в систему; наличие встроенной справочной системы; быстрый поиск необходимых данных; разграничение пользовательских прав и т.д.

Разработанное программное обеспечение имеет:

* парольную защиту при запуске программы;
* возможность резервного копирования информационной базы;
* разграничение пользовательских прав.

Все экранные формы пользовательского интерфейса имеют заголовок с логотипом.

Шрифт: Arial

В качестве основного фона используется белый цвет; в качестве дополнительного другие приятные цвета.

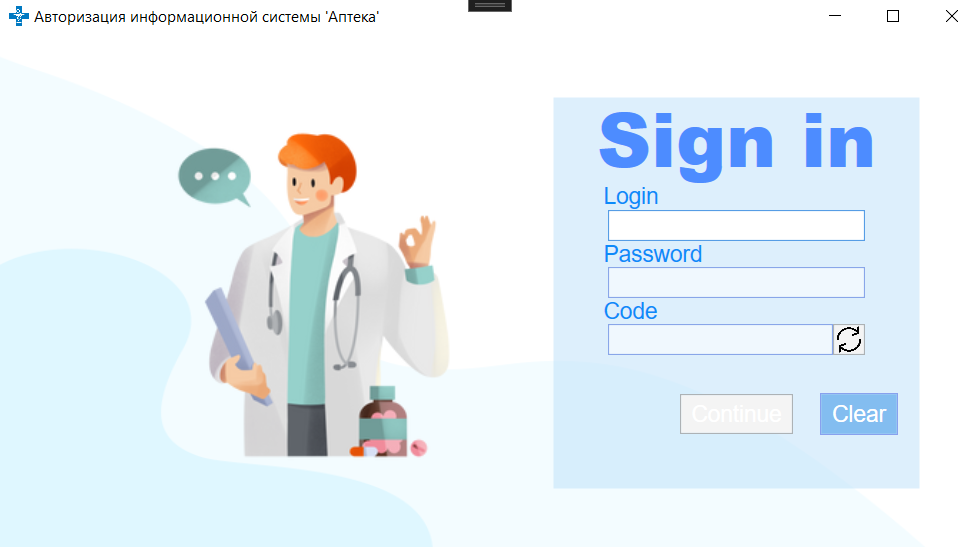


Рисунок 3.1.1. Авторизация

Разрабатываемая система имеет модульную структуру, доступ к модулям будет зависеть от роли пользователя. Доступ возможен только для авторизованных пользователей. В связи с этим при запуске системы первым окном будет окно входа.

У каждого сотрудника есть логин, пароль и сгенерированный код. После ввода логина нужно нажать кнопку “Enter” и если логин есть в базе данных – то будет открыт доступ к строке ввода пароля. Если логин будет введен неверно, то выведется маленькое окошко с предупреждением о неправильно введенном логине.

После ввода пароля, открывается поле с кодом, кнопка обновить и кнопка войти. Код случайно генерируется и состоит из 4 цифр. При неправильном вводе пароля и кода выводится специальное сообщение с ошибкой, которую совершил пользователь.

Окно авторизации

В данном окне имеется 3 кнопки. Кнопка «Continue» (Войти) отвечает за вход, если все данные верны – открывается главное меню программы. Кнопка «Обновить» (в виде стрелок) отвечает за генерацию нового кода, если код выданный ранее слишком сложный, либо же пользователь не запомнил. Кнопка «Clear» (Отмена/Очистить) отвечает за возможность очистки всех полей ввода.

**Код с разметкой окна авторизации:**

Код для названия открывающегося окна, его размера, а так же для иконки/логотипа:

Title="Авторизация информационной системы 'Аптека'" Height="450" Width="800" Icon="/Resources/icon.ico">

<Grid>

Код для добавления картинки на основной фон:

<Grid.Background>

<ImageBrush TileMode="None" Stretch="Fill" ImageSource="/fon.png"/>

</Grid.Background>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<Grid Grid.Column="1" Margin="50">

<Grid.Background>

<SolidColorBrush Color="#FFBBE0F9" Opacity="0.5"/>

</Grid.Background>

<Grid.RowDefinitions>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

</Grid.RowDefinitions>

Код для надписи "Sign in":

<TextBlock Text="Sign in" Grid.Row="0" VerticalAlignment="Bottom" HorizontalAlignment="Center" FontSize="60" Foreground="#FF4D8CFF" FontFamily="Arial Black"></TextBlock>

Код для надписи «Логин»:

<TextBlock Text="Login" Margin="40 0 0 0" Grid.Row="1" VerticalAlignment="Center" HorizontalAlignment="Left" FontSize="18" Foreground="#FF0C8AF9" FontFamily="Arial"></TextBlock>

Код для поля «Логин»:

<TextBox KeyUp="Log\_KeyUp" Grid.Row="2" VerticalAlignment="Center" HorizontalAlignment="Center" Width="205" FontSize="18" Height="25" BorderBrush="#FF486CDA" x:Name="Log" Foreground="#FF486CDA"></TextBox>

Код для надписи «Пароль»:

<TextBlock Text="Password" Margin="40 0 0 0" Grid.Row="3" VerticalAlignment="Center" HorizontalAlignment="Left" FontSize="18" Foreground="#FF0C8AF9" FontFamily="Arial"></TextBlock>

Код для поля «Пароль»

<PasswordBox KeyUp="Pass\_KeyUp" Grid.Row="4" VerticalAlignment="Center" HorizontalAlignment="Center" Width="205" FontSize="18" Height="25" Foreground="#FF486CDA" BorderBrush="#FF486CDA" x:Name="Pass"></PasswordBox>

Код для надписи «Код»:

<TextBlock Text="Code" Margin="40 0 0 0" Grid.Row="5" VerticalAlignment="Center" HorizontalAlignment="Left" FontSize="18" Foreground="#FF0C8AF9" FontFamily="Arial"></TextBlock>

<DockPanel Grid.Row="6" VerticalAlignment="Center" HorizontalAlignment="Center">

Код для поля «Код»:

<TextBox Width="180" FontSize="18" Height="25" Foreground="#FF486CDA" BorderBrush="#FF486CDA" x:Name="Code"></TextBox>

Код для кнопки «Обновить»:

<Button x:Name="Refresh" Click="Refresh\_Click" Height="25" Width="25" BorderBrush="#FF486CDA">

Код картинки стрелок для кнопки «Обновить»:

<Image Source="/Resources/strelka.png" OpacityMask="#FF83BDF0"/>

</Button>

</DockPanel>

Код для кнопки «Продолжить»:

<Button x:Name="Login" Click="Login\_Click" VerticalAlignment="Top" HorizontalAlignment="Center" Grid.Row="7" Content="Continue" Height="32" Width="90" FontSize="18" Margin="0,31,0,0" BorderBrush="#FF88A1EE" Background="#FF83BDF0" Foreground="White" FontFamily="Arial"></Button>

</Grid>

Код картинки доктора слева:

<Image Source="/Resources/login1.png" Grid.Column="0" Height="300" Margin="100 0 0 0"/>

Код кнопки «Отмена/Сброс»:

<Button x:Name="ClearBut" Content="Clear" Background="#FF83BDF0" BorderBrush="#FF88A1EE" Foreground="White" Margin="263,286,67,0" VerticalAlignment="Top" Height="34" Click="ClearBut\_Click" Grid.Column="1" FontSize="18" FontFamily="Arial"/>

</Grid>

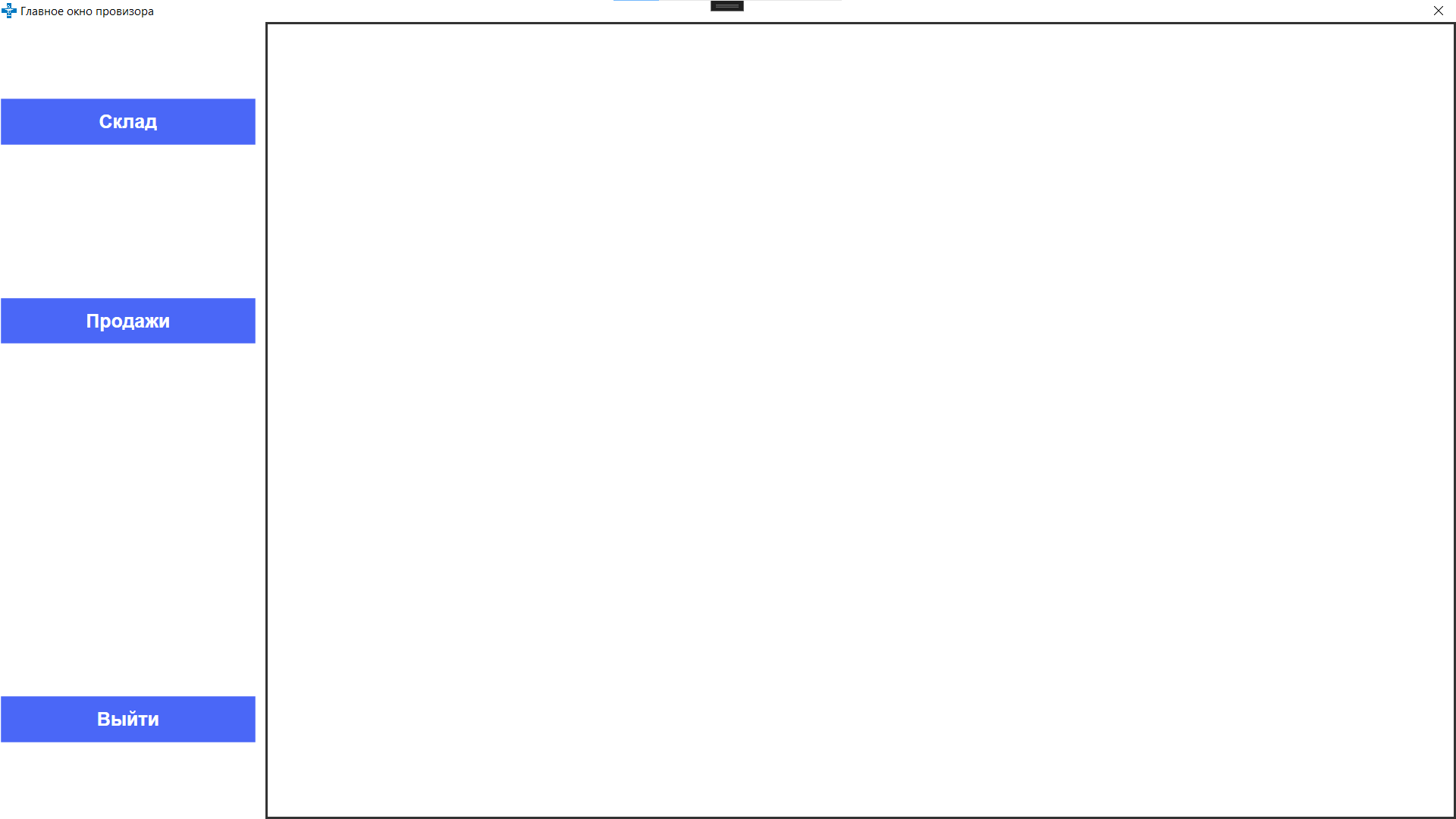


Рисунок 3.1.2. Главное окно провизора

В качестве основного фона используется белый цвет #FFFFFF; для акцентирования внимания пользователя на целевое действие интерфейса используется цвет #4A67F7.

Данное окно появляется после входа. Вход осуществлен провизором, данное окно аналогично у фармацевта.

В окне есть возможность перейти на вкладки «Склад» и «продажи». А так же имеется кнопка «Выйти», которая осуществляет выход из системы.

При нажатии на кнопку «Склад», открывается таблица с данными о лекарствах.

При нажатии на кнопку «Продажи», открывается таблица с данными об операциях покупки.

**Код разметки главного окна:**

Код для названия открывающегося окна, его размера, а так же для иконки/логотипа:

Title="Главное окно провизора" Height="450" Width="800" ResizeMode="NoResize" WindowState="Maximized" Icon="/Resources/icon.ico">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="auto"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<Grid Grid.Row="0">

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

Код для кнопки «Склад»:

<Button Name="Store" Content="Склад" Click="Store\_Click" Grid.Row="0" Grid.Column="0" Margin="0 0 10 0" Height="50" Width="270" FontSize="20" FontWeight="DemiBold" VerticalAlignment="Center" HorizontalAlignment="Left" BorderBrush="#FFFFFCFC" Background="#FF4A67F7" Foreground="White" FontFamily="Arial"/>

Код для кнопки «Продажи»:

<Button Name="Sales" Content="Продажи" Click="Sales\_Click" Grid.Row="1" Margin="0 0 10 0" Grid.Column="0" Height="50" Width="270" FontSize="20" FontWeight="DemiBold" VerticalAlignment="Center" HorizontalAlignment="Left" Background="#FF4A67F7" BorderBrush="White" Foreground="White" FontFamily="Arial"/>

Код для кнопки «Выход»:

<Button Name="Exit" Content="Выйти" Click="Exit\_Click" Grid.Row="3" Grid.Column="0" Margin="0 0 10 0" Height="50" Width="270" FontSize="20" FontWeight="DemiBold" VerticalAlignment="Center" HorizontalAlignment="Left" Background="#FF4A67F7" Foreground="White" BorderBrush="White" FontFamily="Arial"/>

</Grid>

<Frame x:Name="MainFrame" Background="White" Opacity="0.8" BorderBrush="Black" BorderThickness="2.5" NavigationUIVisibility="Hidden" Grid.Column="1" Navigated="MainFrame\_Navigated"/>

</Grid>

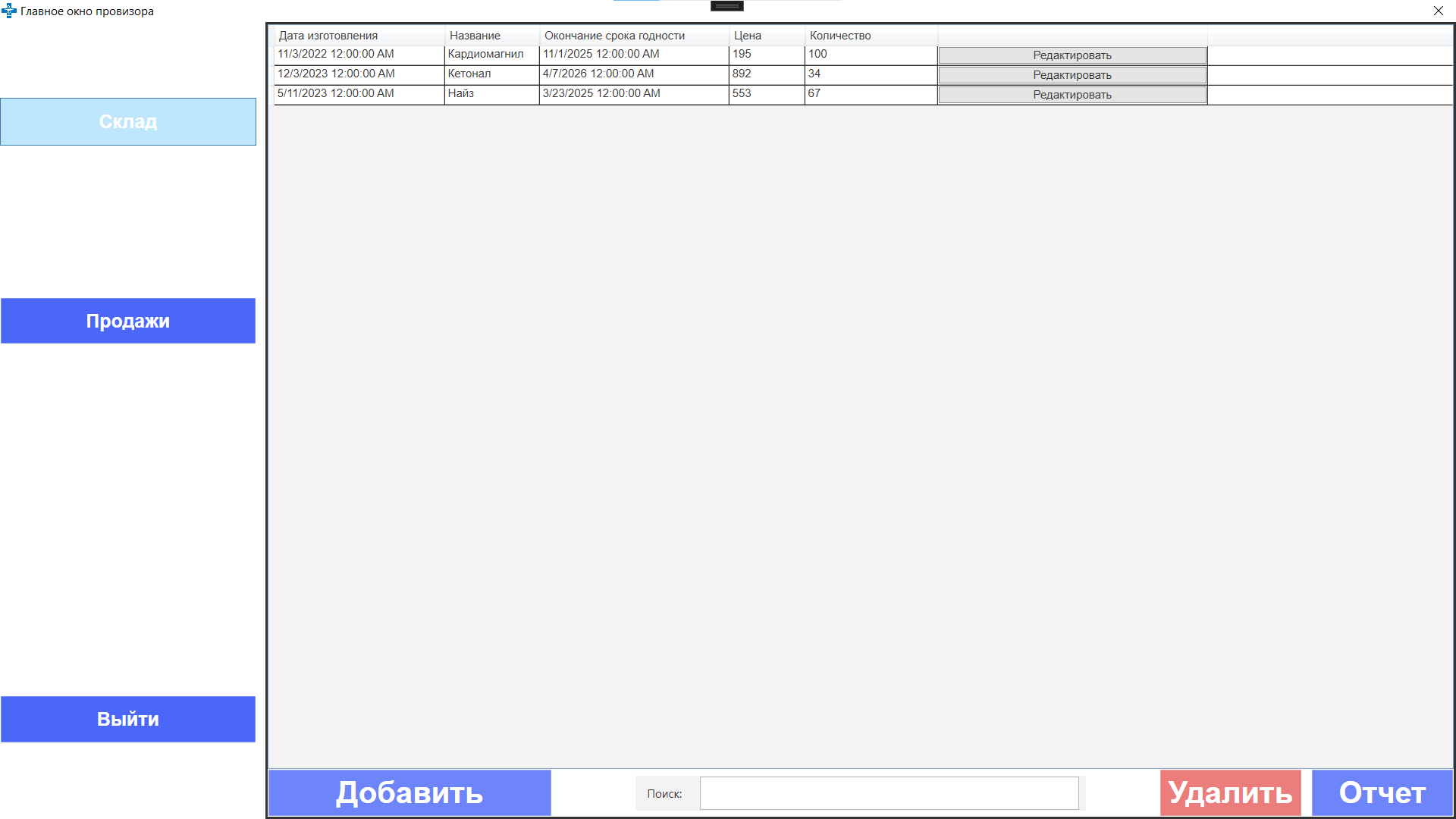


Рисунок 3.1.3. Склад

Данная страница показывает список лекарств, которые на данный момент имеются на складе.

На странице возможно редактировать данные о лекарствах.

Также присутствует возможность добавить медикамент путем нажатия на кнопку “Добавить” и заполнением специального поля (Рисунок 3.1.4). Кнопка “Удалить” отвечает за возможность удаления лекарства из таблицы. Кнопка “Отчет” позволят экспортировать данные в таблицу EXCEL.

Страница «Продажи» реализована идентичным образом.

**Код с разметкой страницы с лекарствами:**

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

<RowDefinition Height="50"/>

</Grid.RowDefinitions>

<DataGrid Name="DGridApteca" AutoGenerateColumns="False" IsReadOnly="True" FontFamily="Arial">

<DataGrid.Columns>

Код, который выводит данные из базы данных:

<DataGridTextColumn Header="Дата изготовления" Binding="{Binding Receipt\_date}" Width="180"/>

<DataGridTextColumn Header="Название" Binding="{Binding Name\_medicines}" Width="100"/>

<DataGridTextColumn Header="Окончание срока годности" Binding="{Binding Expiration\_date}" Width="200"/>

<DataGridTextColumn Header="Цена" Binding="{Binding Price}" Width="80"/>

<DataGridTextColumn Header="Количество" Binding="{Binding Quantity}" Width="140"/>

<DataGridTemplateColumn Width="285">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

Код для кнопки «Редактировать»:

<Button Content="Редактировать" Click="BtnEdit\_Click" Name="BtnEdit"/>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

</DataGrid.Columns>

</DataGrid>

<StackPanel Orientation="Horizontal" Grid.Row="1" HorizontalAlignment="Center" VerticalAlignment="Center">

<Menu>

Код для поиска:

<Label Content="Поиск:"/>

<TextBox Width="400" Height="35" Name="TBoxSearch1" TextChanged="TBoxSearch1\_TextChanged" FontSize="22" FontWeight="DemiBold"/>

</Menu>

</StackPanel>

Код для кнопки «Добавить»:

<Button Name="BtnAdd" Click="BtnAdd\_Click" Content="Добавить" Grid.Row="1" HorizontalAlignment="Left" Width="300" BorderBrush="White" Foreground="White" FontSize="32" FontWeight="DemiBold" Background="#FF4A67F7" FontFamily="Arial"/>

<DockPanel Grid.Row="1" HorizontalAlignment="Right">

Код для кнопки «Удалить»:

<Button Name="BtnDel" Click="BtnDel\_Click" Content="Удалить" Width="150" Margin="0 0 10 0" BorderBrush="White" Foreground="White" FontSize="32" FontWeight="DemiBold" Background="#FFE65C5C" FontFamily="Arial"/>

Код для кнопки «Отчет»:

<Button Name="BtnForm" Click="BtnForm\_Click" Content="Отчет" Width="150" BorderBrush="White" Foreground="White" FontSize="32" FontWeight="DemiBold" Background="#FF4A67F7" FontFamily="Arial"/>

</DockPanel>

</Grid>

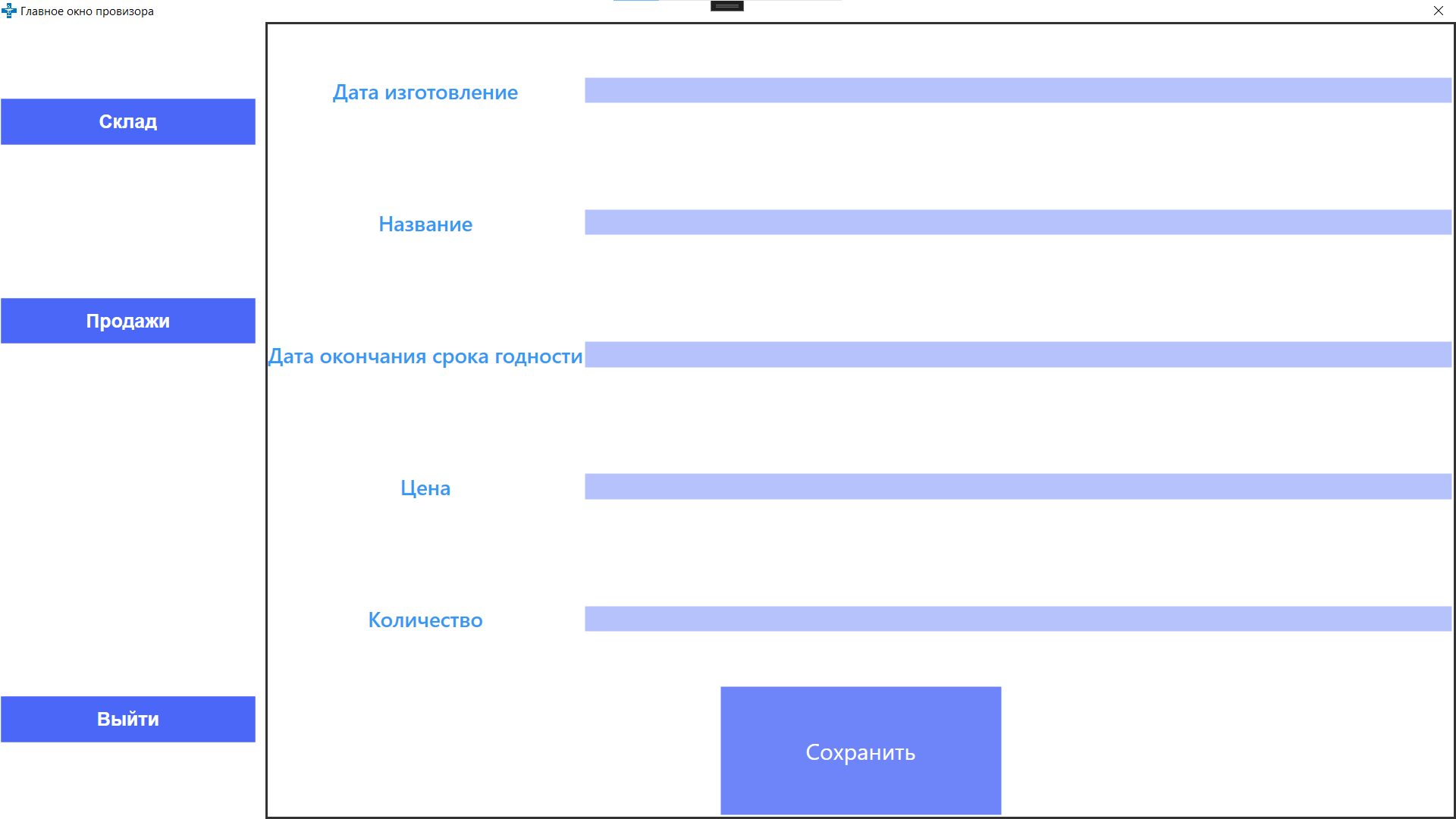


Рисунок 3.1.4. Добавление в таблицу склада

Данное поле используется при добавлении лекарств. В данном поле необходимо ввести дату изготовления лекарства, название, дату окончания срока годности, цену и количество, а так же кнопка «Сохранить», которая сохраняет все введенные данные в таблицу склада.

Страница «Продажи» реализована идентичным образом.

**Код для добавления лекарств:**

<Grid Background="White">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="Auto"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

Код для поля «Дата изготовления»:

<TextBlock Text="Дата изготовление" Grid.Row="0" Foreground="#FF0080F1" HorizontalAlignment="Center" VerticalAlignment="Center" FontSize="22" FontWeight="DemiBold"/>

Код для поля «Название»

<TextBlock Text="Название" Grid.Row="1" Foreground="#FF0080F1" HorizontalAlignment="Center" VerticalAlignment="Center" FontSize="22" FontWeight="DemiBold"/>

Код для поля «Дата окончания срока годности»:

<TextBlock Text="Дата окончания срока годности" Grid.Row="2" Foreground="#FF0080F1" HorizontalAlignment="Center" VerticalAlignment="Center" FontSize="22" FontWeight="DemiBold"/>

Код для поля «Цена»:

<TextBlock Text="Цена" Grid.Row="3" Foreground="#FF0080F1" HorizontalAlignment="Center" VerticalAlignment="Center" FontSize="22" FontWeight="DemiBold"/>

Код для поля «Количество»

<TextBlock Text="Количество" Grid.Row="4" Foreground="#FF0080F1" HorizontalAlignment="Center" VerticalAlignment="Center" FontSize="22" FontWeight="DemiBold"/>

Код для поля, которое выводит из базы данных данные из таблицы «Дата изготовления»:

<TextBox Text="{Binding Receipt\_date}" VerticalAlignment="Center" FontSize="20" MaxLength="100" Grid.Column="1" Grid.Row="0" BorderBrush="White" BorderThickness="2" Foreground="White">

<TextBox.Background>

<SolidColorBrush Color="#FF4A67F7" Opacity="0.5"/>

</TextBox.Background>

</TextBox>

Код для поля, которое выводит из базы данных данные из таблицы «Название»:

<TextBox Text="{Binding Name\_medicines}" VerticalAlignment="Center" FontSize="20" MaxLength="20" Grid.Column="1" Grid.Row="1" BorderBrush="White" BorderThickness="2" Foreground="White">

<TextBox.Background>

<SolidColorBrush Color="#FF4A67F7" Opacity="0.5"/>

</TextBox.Background>

</TextBox>

Код для поля, которое выводит из базы данных данные из таблицы «Дата окончания срока годности»:

<TextBox Text="{Binding Expiration\_date}" VerticalAlignment="Center" FontSize="20" MaxLength="15" Grid.Column="1" Grid.Row="2" BorderBrush="White" BorderThickness="2" Foreground="White">

<TextBox.Background>

<SolidColorBrush Color="#FF4A67F7" Opacity="0.5"/>

</TextBox.Background>

</TextBox>

Код для поля, которое выводит из базы данных данные из таблицы «Цена»:

<TextBox Text="{Binding Price}" VerticalAlignment="Center" PreviewTextInput="TextBox\_PreviewTextInput" FontSize="20" Grid.Column="1" Grid.Row="3" BorderBrush="White" BorderThickness="2" Foreground="White">

<TextBox.Background>

<SolidColorBrush Color="#FF4A67F7" Opacity="0.5"/>

</TextBox.Background>

</TextBox>

Код для поля, которое выводит из базы данных данные из таблицы «Количество»:

<TextBox Text="{Binding Quantity}" VerticalAlignment="Center" PreviewTextInput="TextBox\_PreviewTextInput" FontSize="20" Grid.Column="1" Grid.Row="4" BorderBrush="White" BorderThickness="2" Foreground="White">

<TextBox.Background>

<SolidColorBrush Color="#FF4A67F7" Opacity="0.5"/>

</TextBox.Background>

</TextBox>

Код для кнопки «Сохранить»:

<Button Name="BtnSaveAndExit" Click="BtnSaveAndExit\_Click" Grid.ColumnSpan="2" Grid.Row="8" Width="300" Content="Сохранить" Background="#FF4A67F7" BorderBrush="White" BorderThickness="2" Foreground="White" FontSize="24" />

</Grid>

</Page>

# 3.2 Программирование информационной системы

**Код окна авторизации:**

Код для фокусировки на поле логина:

public MainWindow()

{

InitializeComponent();

Pass.IsEnabled = false;

Code.IsEnabled = false;

Refresh.IsEnabled = false;

Login.IsEnabled = false;

Log.Focus();

}

Код для ввода логина:

private void Log\_KeyUp(object sender, KeyEventArgs e)

{

if (e.Key == Key.Enter)

{

using (var db = new AptecaEntities())

{

var number = db.Users.AsNoTracking().FirstOrDefault(m => m.Login ==

Log.Text.Trim());

if (number == null)

{

Появление окна предупреждения при неверном вводе логина:

MessageBox.Show("Такого логина не существует!");

}

else

{

Pass.IsEnabled = true;

Log.IsEnabled = false;

Pass.Focus();

}

}

}

}

Код для ввода пароля:

private void Pass\_KeyUp(object sender, KeyEventArgs e)

{

if (e.Key == Key.Enter)

{

using (var db = new AptecaEntities())

{

var password = db.Users.AsNoTracking().FirstOrDefault(m => m.Login ==

Log.Text.Trim() & (m.Password == Pass.Password));

if (password == null)

{

Появление окна предупреждения при неверном вводе пароля:

MessageBox.Show("Такого пароля не существует, попробуйте снова");

}

else

{

Pass.IsEnabled = false;

gencode();

Code.Focus();

}

}

}

}

DispatcherTimer timer = new DispatcherTimer();

string code;

private void gencode()

{

Код, отвечающий за генерацию кода. Он предназначен для генерации случайного набора цифр. Также данный блок кода блокирует некоторые поля для ввода и открывает нужные поля, кнопки для ввода:

code = null;

Random random = new Random();

string[] massiveCharacters = new string[] { "1", "2", "3", "4", "5", "6", "7", "8" };

for (int i = 0; i < 4; i++)

code += massiveCharacters[random.Next(0, massiveCharacters.Length)];

if (MessageBox.Show(code.ToString(), "Code", MessageBoxButton.OK,

MessageBoxImage.Warning) == MessageBoxResult.OK)

{

Таймер на 10 сек:

timer.Interval = TimeSpan.FromSeconds(10);

timer.Tick += Timer\_Tick;

timer.Start();

Code.IsEnabled = true;

Login.IsEnabled = true;

Refresh.IsEnabled = true;

}

}

void Timer\_Tick(object sender, EventArgs e)

{

code = null;

Окно предупреждения, если не успели ввести код за 10 сек:

MessageBox.Show("Код сброшен. Повторите попытку");

timer.Stop();

}

При нажатии на кнопку «Обновить», код повторно высылается:

private void Refresh\_Click(object sender, RoutedEventArgs e)

{

timer.Stop();

gencode();

Code.Focus();

}

private void Login\_Click(object sender, RoutedEventArgs e)

{

if (code == Code.Text)

{

timer.Stop();

using (var db = new AptecaEntities())

{

var user = db.Users.AsNoTracking().FirstOrDefault(u => u.Login == Log.Text && u.Password == Pass.Password);

if (user.RoleId == 1)

{

AdminWindow kurs = new AdminWindow();

kurs.Show();

Application.Current.MainWindow.Close();

}

else

{

UserWindow kurs2 = new UserWindow();

kurs2.Show();

Application.Current.MainWindow.Close();

}

}

}

else

{

Окно предупреждения, если ввели неверно код:

timer.Stop();

MessageBox.Show("Код неверный");

}

}

Кнопка «Отчистить», отчищающая все поля ввода:

private void ClearBut\_Click(object sender, RoutedEventArgs e)

{

Log.Clear();

Pass.Clear();

Code.Clear();

Log.IsEnabled = true;

Pass.IsEnabled = false;

Code.IsEnabled = false;

Login.IsEnabled = false;

Refresh.IsEnabled = false;

Log.Focus();

}

}

}

**Код главного окна при входе провизора в систему:**

public AdminWindow()

{

InitializeComponent();

Manager.MainFrame = MainFrame;

}

Код для кнопки «Выйти»:

private void Exit\_Click(object sender, RoutedEventArgs e)

{

MainWindow mw1 = new MainWindow();

mw1.Show();

this.Close();

}

Код для кнопки «Продажи»:

private void Sales\_Click(object sender, RoutedEventArgs e)

{

MainFrame.Navigate(new SalesEdit());

}

Код для кнопки «Склад»:

private void Store\_Click(object sender, RoutedEventArgs e)

{

MainFrame.Navigate(new StoreAdmin());

}

private void MainFrame\_Navigated(object sender, System.Windows.Navigation.NavigationEventArgs e)

{

}

private void Button\_Click(object sender, RoutedEventArgs e)

{

}

}

}

**Код страницы «Склад»:**

public StoreAdmin()

{

Код, связанный с таблицей базы данных, отвечающей за склад:

InitializeComponent();

DGridApteca.ItemsSource = AptecaEntities.GetContext().Medicines.ToList();

UpdateClients();

}

private void UpdateClients()

{

Код, обновляющий таблицу, чтобы изменения были видны:

Данный код создан только для того, чтобы после внесения изменений данных в таблицу – она обновлялась и моментально показывала любые изменения.

var currentMedicines = AptecaEntities.GetContext().Medicines.ToList();

currentMedicines = currentMedicines.Where(p => p.Name\_medicines.ToLower().Contains(TBoxSearch1.Text.ToLower())).ToList();

DGridApteca.ItemsSource = currentMedicines.OrderBy(p => p.Name\_medicines).ToList();

}

private void TBoxSearch1\_TextChanged(object sender, TextChangedEventArgs e)

{

UpdateClients();

}

Код, отвечающий за кнопку «Выйти»:

private void BtnEdit\_Click(object sender, RoutedEventArgs e)

{

Manager.MainFrame.Navigate(new AddAddit((sender as Button).DataContext as Medicine));

}

Код отвечающий за кнопку "Добавить"

Данный код открывает страницу с добавлением лекарств в таблицу:

private void BtnAdd\_Click(object sender, RoutedEventArgs e)

{

Manager.MainFrame.Navigate(new AddAddit(null));

}

Код, отвечающий за кнопку "Удалить":

При нажатии на кнопку “Удалить” будет удален элемент(-ы) из таблицы.

private void BtnDel\_Click(object sender, RoutedEventArgs e)

{

Окно предупреждения при удалении строчки таблицы:

var MedicineForRemoving = DGridApteca.SelectedItems.Cast<Medicine>().ToList();

if (MessageBox.Show($"Вы действительно хотите удалить {MedicineForRemoving.Count()} ?", "Внимание", MessageBoxButton.YesNo, MessageBoxImage.Question) == MessageBoxResult.Yes)

{

try

{

Окно подтверждения удаления строчки таблицы:

AptecaEntities.GetContext().Medicines.RemoveRange(MedicineForRemoving);

AptecaEntities.GetContext().SaveChanges();

UpdateClients();

MessageBox.Show("Удалено!");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, ToString());

DGridApteca.ItemsSource = AptecaEntities.GetContext().Medicines.ToList();

}

}

}

Код для кнопки «Отчеты», перемещающий нас в Exel:

private void BtnForm\_Click(object sender, RoutedEventArgs e)

{

this.DGridApteca.SelectAllCells();

this.DGridApteca.ClipboardCopyMode = DataGridClipboardCopyMode.IncludeHeader;

ApplicationCommands.Copy.Execute(null, this.DGridApteca);

this.DGridApteca.UnselectAllCells();

String result = Clipboard.GetData(DataFormats.CommaSeparatedValue).ToString();

try

{

//StreamWriter sw = new StreamWriter("Medicines.csv");

StreamWriter sw = new StreamWriter(new FileStream("Medicines.csv", FileMode.OpenOrCreate), Encoding.UTF32);

sw.WriteLine(result);

sw.Close();

Process.Start("Medicines.csv");

}

catch (Exception ex)

{ }

}

}

}

**Код страницы с добавлением лекарств:**

public partial class AddAddit : Page

{

Код, связанный с таблицей базы данных, отвечающей за склад:

private Medicine \_currentMedicine = new Medicine();

public AddAddit(Medicine selectedMedicine)

{

InitializeComponent();

if (selectedMedicine != null)

\_currentMedicine = selectedMedicine;

DataContext = \_currentMedicine;

}

Кнопка сохранения внесенных данных:

private void BtnSaveAndExit\_Click(object sender, RoutedEventArgs e)

{

{

StringBuilder errors = new StringBuilder();

if (\_currentMedicine.Id == 0)

AptecaEntities.GetContext().Medicines.Add(\_currentMedicine);

try

{

Код с окошком об успешном сохранении внесенной информации

AptecaEntities.GetContext().SaveChanges();

MessageBox.Show("Информация сохранена!");

Manager.MainFrame.GoBack();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message.ToString());

}

}

}

Код для строк, куда вводятся данные:

private void TextBox\_PreviewTextInput(object sender, TextCompositionEventArgs e)

{

}

}

}

**Код страницы с добавлением в таблицу «Продажи»:**

public partial class SellsAddAddit : Page

{

Код, связанный с таблицей базы данных, отвечающей за продажи:

private Cheque \_currentCheque = new Cheque();

public SellsAddAddit(Cheque selectedCheque)

{

InitializeComponent();

if (selectedCheque != null)

\_currentCheque = selectedCheque;

DataContext = \_currentCheque;

}

Кнопка сохранения внесенных данных:

private void BtnSaveAndExit\_Click(object sender, RoutedEventArgs e)

{

{

StringBuilder errors = new StringBuilder();

if (\_currentCheque.Id == 0)

AptecaEntities.GetContext().Cheques.Add(\_currentCheque);

try

{

AptecaEntities.GetContext().SaveChanges();

MessageBox.Show("Информация сохранена!");

Manager.MainFrame.GoBack();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message.ToString());

}

}

}

}

# ЗАКЛЮЧЕНИЕ

Информационную систему удобно использовать в различных организациях, и аптека не исключение. Для аптеки как торгового учреждения характерна высокая степень отчетности. Учитывая значительный объем и темпы товарооборота, специфические требования к его хранению и реализации, здесь очень важно наладить быстрый и качественный сбор, обработку и анализ информации, относящийся к коммерческой деятельности аптеки. Создание базы данных в аптеке позволит собрать все данные в одном месте и организовать доступ к ним всем сотрудникам, позволит вести четкий структурированный учет всей информации, предоставит быстрый доступ к ней, поможет автоматизировать стандартные процессы и в результате всего оптимизирует работу всей аптеки. Внедрение компьютерных систем в аптеках стало необходимым и в связи с все возрастающим потоком информации, в котором все сотрудники просто обязаны ориентироваться для того, чтобы качественно выполнять свои обязанности. А делать это становится все труднее, учитывая постоянно расширяющийся и обновляющийся ассортимент лекарственных средств, не говоря уже о таком факторе, как наличие на отечественном фармрынке огромного количества фирм-посредников, эффективное взаимодействие с которыми невозможно осуществлять без применения компьютерных технологий.

В процессе выполнения курсового проекта был произведен анализ предметной области с выделением аналогов разрабатываемого программного продукта. Благодаря этому была сформирована структура разрабатываемого приложения и его основные функции.

# СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. WPF: Windows Presentation Foundation в . NET 4.5. Мэтью Макдональд
2. Язык программирования C# 7 и платформы .NET и .NET Core Эндрю Троелсен

**Интернет-источники:**

1. Википедия (Диаграмма вариантов использования): [Электронный ресурс] - https://ru.wikipedia.org/wiki/Диаграмма\_варианотов\_использования.
2. 2. Википедия (Диаграмма деятельности): [Электронный ресурс] - https://ru.wikipedia.org/wiki/Диаграмма\_деятельности.
3. 3. Википедия (Диаграмма последовательностей): [Электронный ресурс] - https://ru.wikipedia.org/wiki/Диаграмма\_последовательностей.
4. 4. Википедия (Диаграмма состояний): [Электронный ресурс] - https://ru.wikipedia.org/wiki/Диаграмма\_состояний.
5. 5. Википедия (Диаграмма классов): [Электронный ресурс] - https://ru.wikipedia.org/wiki/Диаграмма\_классов.
6. 6. Википедия (Диаграмма компонентов): [Электронный ресурс] - https://ru.wikipedia.org/wiki/Диаграмма\_компонентов.
7. 7. Википедия (диаграмма развертывания): [Электронный ресурс] - https://ru.wikipedia.org/wiki/Диаграмма\_развертывания.
8. 8. Википедия (Диаграмма ER): [Электронный ресурс] - https://ru.wikipedia.org/wiki/Диаграмма\_ER.
9. 9. Википедия (Библиотека): [Электронный ресурс] - https://ru.wikipedia.org/wiki/Библиотека.
10. 10. Википедия (Информационная система): [Электронный ресурс] - https://ru.wikipedia.org/wiki/Информационная\_система.