Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования «Магнитогорский государственный технический университет им. Г.И.Носова»

Многопрофильный колледж

Отделение № 2 Информационных технологий и транспорта

ПЦК\_\_Информатики и вычислительной техники\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Допустить к защите

Заведующий отделением

\_\_\_\_\_\_\_\_\_\_\_/\_Н.В. Сидорова

«\_\_\_\_»\_\_\_\_\_\_\_\_2025г.

**ДИПЛОМНЫЙ ПРОЕКТ**

**O.09.02.07.ДП.ПЗ**

Обучающегося\_\_\_\_\_\_\_\_Абушкевича Дениса Витальевича\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(фамилия имя отчество)

На тему\_Разработка приложения для учёта пациентов поликлиники в среде Visual Studio

(полное наименование темы)

Состав дипломной работы:

1. Пояснительная записка на \_\_\_\_ страницах
2. Графическая часть на \_\_\_\_\_\_ листах

Руководитель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_преподаватель, Тутарова В.Д.

Председатель ПЦК\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ преподаватель, Ремез Т.Б.\_\_

(подпись, дата, должность, ученая степень, звание, Ф.И.О.)

|  |  |  |  |
| --- | --- | --- | --- |
| Отметка нормоконтролера | | Обучающийся\_\_\_\_\_\_\_\_\_/ Абушкевич Д.В. |  |
| \_\_\_\_\_\_\_\_\_\_\_\_\_ /\_\_ Сидовова Н.В / | |  |
| (подпись) | (ФИО) | (подпись) |  |
| «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2025 г. |  |
|  | «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2025 г. |  |

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Магнитогорский государственный технический университет им. Г.И.Носова»

Многопрофильный колледж

Отделение № 2 Информационных

технологий и транспорта

Утверждаю:

Заведующий отделением

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Н.В. Сидорова

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2025 г.

**ЗАДАНИЕ**

**НА ДИПЛОМНЫЙ ПРОЕКТ**

Тема\_\_ Разработка приложения для учёта пациентов поликлиники в среде Visual Studio \_\_\_\_

Обучающемуся \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Абушкевичу Денису Витальевичу\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(фамилия имя отчество)

Тема утверждена приказом № 10-35/731 от 17 апреля 2025 г.

Исходные данные к проекту

* литературные источники;
* интернет источники.

Перечень вопросов, подлежащих разработке:

1 ТЕОРЕТИКО-ИНФОРМАЦИОННЫЙ АНАЛИЗ ДЕЯТЕЛЬНОСТИ ПОЛИКЛИНИКИ

1.1 Анализ предметной области

1.2 Анализ и оценка существующих программных продуктов

1.3 Постановка задачи

2 ОСОБЕННОСТИ ПОСТРОЕНИЯ ЧАСТИ «ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА»

2.1 Моделирование программного продукта

2.2 Обоснование выбора средств разработки

2.3 Разработка базы данных

2.4 Проектирование меню и интерфейса программного продукта

2.5 Спецификация программы

2.6 Описание запросов при разработке программного продукта

2.7 Проектирование отчетов

2.8 Администрирование программного продукта

3 ОСОБЕННОСТИ ПОСТРОЕНИЯ ЧАСТИ «ОПИСАНИЕ РАБОТЫ ПРОГРАММНОГО ПРОДУКТА»\_

3.1 Руководство пользователя

3.2 Руководство программиста

4 ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ

4.1 Виды возможных угроз программного продукта

4.2 Информационная безопасность программного продукта

5 УПРАВЛЕНИЕ ПРОЕКТОМ

5.1 Понятие проекта, необходимость управления проектом

5.2 Постановка цели проекта по SMART

5.3 Анализ заинтересованных сторон проекта

5.4 Устав проекта

5.5 Планирование проекта

5.6 Управление стоимостью проекта

5.7 Управление рисками проекта

Графическая часть:

* Презентация разработанная средствами PowerPoint.

Руководитель:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/\_\_\_Тутарова В.Д.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/

подпись

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_ г.

Задание получил:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/\_\_Абушкевич Д.В.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/

подпись

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_ г.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 8](#_Toc200110764)

[1 ТЕОРЕТИКО-ИНФОРМАЦИОННЫЙ АНАЛИЗ ДЕЯТЕЛЬНОСТИ ПОЛИКЛИНИКИ 9](#_Toc200110765)

[1.1 Анализ предметной области 9](#_Toc200110766)

[1.2 Анализ и оценка существующих программных продуктов 16](#_Toc200110767)

[1.3 Постановка задачи 19](#_Toc200110773)

[2 ОСОБЕННОСТИ ПОСТРОЕНИЯ ЧАСТИ «ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА» 22](#_Toc200110774)

[2.1 Моделирование программного продукта 22](#_Toc200110775)

[2.2 Обоснование выбора средств разработки 25](#_Toc200110776)

[2.3 Разработка базы данных 26](#_Toc200110777)

[2.4 Проектирование меню и интерфейса программного продукта 39](#_Toc200110778)

[2.5 Спецификация программы 42](#_Toc200110779)

[2.6 Описание запросов при разработке программного продукта 43](#_Toc200110780)

[2.7 Проектирование отчетов 44](#_Toc200110781)

[2.8 Администрирование программного продукта 44](#_Toc200110782)

[3 ОСОБЕННОСТИ ПОСТРОЕНИЯ ЧАСТИ «ОПИСАНИЕ РАБОТЫ ПРОГРАММНОГО ПРОДУКТА» 47](#_Toc200110783)

[3.1 Руководство пользователя 47](#_Toc200110784)

[3.1.1 Введение 47](#_Toc200110785)

[3.1.2 Назначение и условия применения ПО 47](#_Toc200110786)

[3.1.3 Подготовка к работе 48](#_Toc200110787)

[3.1.4 Уровень подготовки пользователя 48](#_Toc200110788)

[3.1.5 Аварийные ситуации 49](#_Toc200110789)

[3.1.6 Рекомендации по освоению 49](#_Toc200110790)

[3.2 Руководство программиста 50](#_Toc200110791)

[3.2.1 Общие сведения 50](#_Toc200110792)

[3.2.2 Обязанности и функции системного программиста 50](#_Toc200110793)

[3.2.3 Требования к квалификации и образованию системного программиста 50](#_Toc200110794)

[3.2.4 Описание профессиональных навыков и знаний, необходимых для работы 51](#_Toc200110795)

[3.2.5 Описание процедур и методов разработки программного обеспечения 51](#_Toc200110796)

[3.2.6 Инструкции по использованию специализированного программного обеспечения 51](#_Toc200110797)

[3.2.7 Информация о стандартах и методах тестирования программного обеспечения 51](#_Toc200110798)

[3.2.8 Контроль качества разработки и сопровождения программного обеспечения 51](#_Toc200110799)

[3.2.9 Рекомендации по эффективному управлению проектами   
разработки ПО 52](#_Toc200110800)

[3.2.10 Инструкции по документированию и поддержке   
программного продукта 52](#_Toc200110801)

[4 ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ 53](#_Toc200110802)

[4.1 Виды возможных угроз программного продукта 53](#_Toc200110803)

[4.2 Информационная безопасность программного продукта 54](#_Toc200110804)

[5 УПРАВЛЕНИЕ ПРОЕКТОМ 56](#_Toc200110805)

[5.1 Понятие проекта, необходимость управления проектом 56](#_Toc200110806)

[5.2 Постановка цели проекта по SMART 56](#_Toc200110807)

[5.3 Анализ заинтересованных сторон проекта 58](#_Toc200110808)

[5.4 Устав проекта 60](#_Toc200110809)

[5.5 Планирование проекта 61](#_Toc200110810)

[5.6 Управление стоимостью проекта 64](#_Toc200110811)

[5.7 Управление рисками проекта 68](#_Toc200110812)

[ЗАКЛЮЧЕНИЕ 71](#_Toc200110813)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 72](#_Toc200110814)

[ПРИЛОЖЕНИЕ А (обязательное) Листинги вёрстки приложения 79](#_Toc200110815)

[ПРИЛОЖЕНИЕ Б (обязательное) Листинги логики приложения 90](#_Toc200110816)

# **ВВЕДЕНИЕ**

Больница – это медицинское учреждение, предоставляющее широкий спектр услуг для диагностики, лечения и профилактики заболеваний. Практически 70% рабочего времени врачи тратят на бумажную работу – заполнение различной документации. Длительный и трудоемкий процесс приводит к тому, что пациентам начинает уделяться минимум внимания, что недопустимо. Только на заполнение амбулаторных карт уходит около 3 часов в день. Плюс ко всему врачи переписывают от руки бланки и ведут журнал учета пациентов в больнице или поликлинике.

Учет пациентов в больнице является ключевым фактором организации медицинского обслуживания, который влияет на эффективность работы медицинских учреждений и качество предоставляемых услуг. Систематизация информации о пациентах, их истории болезни и лечебных процессах способствует не только улучшению диагностики и лечения, но и оптимизации управленческих решений внутри больницы.

Для отслеживания расположения пациентов необходимо построить эффективную систему взаимодействия и управления.

Дипломный проект будет направлен на реализацию следующих функций:

* автоматическое формирование отчётов,
* удобное добавление пациентов и запись их на приём в регистратуре,
* удобное управление записями на приём и возможность выписывания других процедур в ходе приёма для специалистов,
* управление сотрудниками для главного врача,
* удобное занесение результатов процедур для лаборантов.

Разработка такого приложения требует хорошего анализа требований пользователей, выбора оптимальных технологий и подходов к разработке, а также обеспечения надежности, масштабируемости и безопасности системы. В процессе создания приложения необходимо учитывать различные аспекты, такие как обработка больших объемов информации, визуализация данных и обеспечение удобного пользовательского интерфейса. Кроме того, следует уделить особое внимание вопросам безопасности и конфиденциальности данных, чтобы обеспечить защиту информации о сотрудниках и пациентах.

Успешная реализация дипломного проекта будет экономить много времени врачам и позволит отслеживать всё в удобном формате, а так же это приведёт к минимизации ошибок в сравнении с бумажным заполнением всей документации.

# 1 ТЕОРЕТИКО-ИНФОРМАЦИОННЫЙ АНАЛИЗ ДЕЯТЕЛЬНОСТИ ПОЛИКЛИНИКИ

## Анализ предметной области

Сегодня государственные/муниципальные учреждения здравоохранения пользуются наибольшей популярностью у населения, в течение последнего года их посещали 59%. В частных клиниках бывали треть россиян - 31%. По возрастным группам максимум тех, кто не обращался в лечебные учреждения, приходится на респондентов старше 35 лет - 29-31%, у молодежи 18-34 лет аналогичный показатель составляет 21-26% [1].

Согласно приказу Минздрава на каждые 20-25 тыс. человек должна быть поликлиника, а на каждые 250 тыс. – консультативно-диагностический центр. Дети, пожилые люди, люди среднего возраста – все болеют, по крайней мере, респираторными заболеваниями, а чем старше, тем больше нуждаются в медицинской помощи при гипертонии или кардиологических проблемах и онкологии и т.д. Средний чек в платной поликлинике для одного посещения – 1700 руб. Обычно в него входит прием и лабораторная или инструментальная диагностика [2].

Есть несколько способов учета пациентов: выбор конкретного зависит от масштабов деятельности клиники, числа посетителей в месяц, количества сотрудников и других факторов [3].

Учет в журналах, амбулаторных картах и других бумажных носителях. Несмотря на архаичность, этот способ учета пациентов все еще востребован. Причин несколько: это бесплатно, просто, не требуется обучение персонала и какое-либо оборудование. Даже в насквозь цифровом 21 веке есть случаи, когда учет пациентов на бумажных носителях будет актуален. Он вполне подойдет микроклиникам. При таком способе нагрузка на сотрудников очень большая [3].

Достоинства учета в журналах, амбулаторных картах и других бумажных носителях [3]:

* низкая стоимость,
* привычно для персонала,
* не нужен компьютер, планшет и другое оборудование,
* можно вести учет без интернета и электричества.

Недостатки учета в журналах, амбулаторных картах и других бумажных носителях [3]:

* человеческий фактор (администратор может [потерять документ, ввести неверные данные](https://archimed.pro/blog/zabyli-poteryali-pereputali-kak-meditsinskaya-informatsionnaya-sistema-reshaet-problemu-chelovechesk/?sphrase_id=800)),
* информация существует в единственном экземпляре и доступна в одном месте,
* невозможно организовать обмен данными,
* подходит для ограниченного числа пациентов,
* высокая нагрузка на персонал (много ручной работы, писанины),
* отсутствие автоматизации. это влечет за собой увеличение штата и фонда оплаты труда: кто-то должен заниматься бумажной волокитой [3].

Учет пациентов в таблицах Excel. Excel, Google-таблицы и им подобные инструменты обеспечивают начальный уровень автоматизации работы клиники. Они серьезно сокращают количество бумажной волокиты и уменьшают нагрузку на персонал. Существует много шаблонов таблиц, разработанные предметно для медицинской сферы. Там есть все необходимые разделы для пациентов: диагноз, анамнез, результаты исследований, анализов и так далее. Так же есть интересные решения, разработанные как раз на основе таких таблиц: всевозможные дашборды, инфографики, анимации, визуализации. Специально разработанные программы берут данные из таблиц и преобразует их в графические изображения [3].

Пример данных из Excel таблицы изображен на рисунке 1 и дашборд для этих данных изображён на рисунке 2.

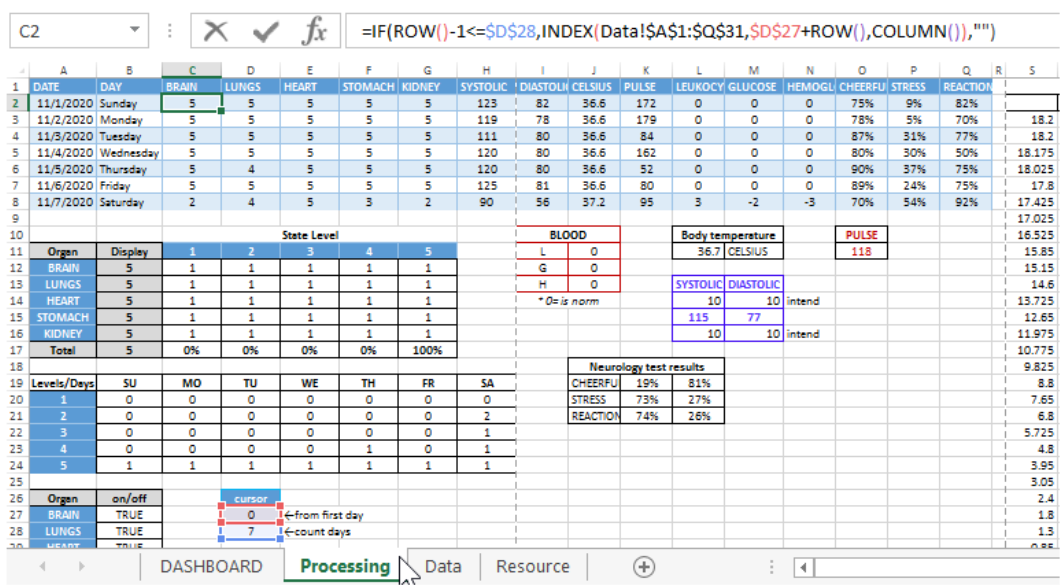


Рисунок 1 ­ Таблица Excel

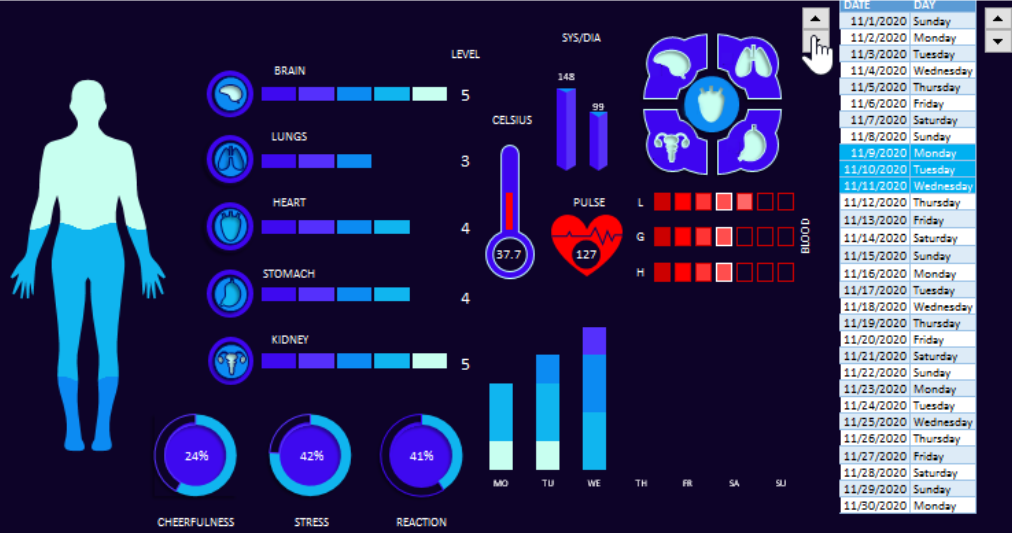


Рисунок 2 ­ Дашбоард в таблице Excel

Достоинства учета пациентов в таблицах Excel [3]:

* небольшая автоматизация: таблицы автоматически заполняют ячейки согласно прописанному алгоритму;
* меньшая нагрузка на персонал, сокращение бумажной работы;
* данными можно поделиться: есть онлайн-версии таблиц, документ можно отправить получателю в виде файла, дать доступ к нему нескольких сотрудников;
* это тоже почти бесплатно, заплатить придется только за операционную систему или платный шаблон;
* когда клиника захочет перейти на более продвинутую медицинскую информационную систему, базу пациентов можно выгрузить из excel автоматически.

Недостатки учета пациентов в таблицах Excel [3]:

* не подходит для клиник с большим числом пациентов (100 человек в месяц — уже критично: таблицы становятся громоздкими и нечитабельными);
* если по каждому пациенту много данных (результаты обследований, история болезни, история посещений) таблицы тоже неудобны;
* нельзя интегрировать с другими сервисами: системой оплаты, электронной медицинской картой, федеральными сервисами;
* сложности с соблюдением законодательства в части персональных данных и врачебной тайны;
* если база пациентов хранится на компьютере клиники, в случае поломки жесткого диска можно потерять все данные.

Медицинские информационные системы – МИС. Самое оптимальное решение для автоматизации клиники в целом и учета пациентов в частности. Медицинская информационная система — это программа, разработанная с учетом специфики отрасли [3].

Достоинства МИС [3]:

* полная автоматизация работы клиники. все модули работают совместно друг с другом: данные подтягиваются из раздела в раздел без участия персонала;
* доступ к программе и данные возможен из любого места земли, где есть интернет-соединение;
* программа легко масштабируется. на старте, пока клиника небольшая и пациентов, можно использовать базовые модуль «[врач](https://archimed.pro/blog/modul-vrach-demonstratsiya-raboty/?sphrase_id=795)». по мере роста нужно просто подключать дополнительные модули: «[касса](https://archimed.pro/blog/programma-kassy-dlya-administratorov-instruktsiya-po-primeneniyu/?sphrase_id=796)», [CRM](https://archimed.pro/blog/crm-sistema-dlya-kliniki-stomatologii-i-chastnogo-vracha-chto-takoe-zachem-nuzhna-i-kak-podklyuchit-/), «[стационар](https://archimed.pro/blog/modul-statsionar-dlya-vracha-instruktsiya-po-primeneniyu/?sphrase_id=795)» и другие.;
* легкая интеграция с любыми сторонними сервисами, в том числе, федеральными: [онлайн-запись на прием](https://archimed.pro/solutions/onlaynzapis/), [лаборатории](https://archimed.pro/solutions/laboratoriy/), [ЕГИСЗ](https://archimed.pro/solutions/egisz/), ДМС, МДЛП;
* оптимизация штата за счет снижения нагрузки на персонал;
* печать стандартных форм: амбулаторной карты 025/у (рис. 3), стоматологической карты 043/у (рис. 4) и любых других.

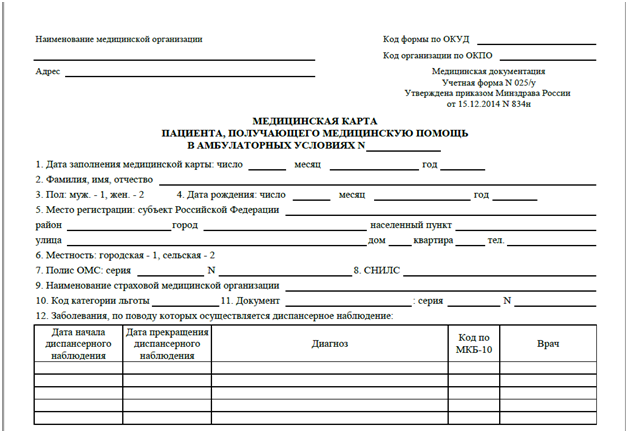


Рисунок 3 ­ Образец форма 025/у «Медицинская карта пациента, получающего медицинскую помощь в амбулаторных условиях»

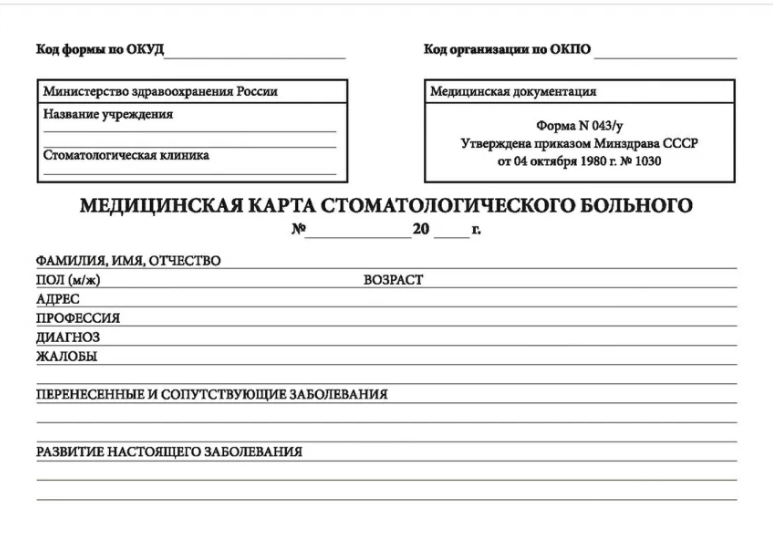


Рисунок 4 ­ Образец форма 043/у «Медицинская карта стоматологического пациента, получающего стоматологическую помощь в амбулаторных условиях»

Недостатки МИС [3]:

* это платно. Стоимость коробочных МИС составляет несколько десятков тысяч рублей (в среднем – 30-40 тысяч), а облачных – 4-5 тысяч рублей в месяц;
* требуется обучение персонала. Для сотрудников, не привыкших работать так, это может оказаться проблемой;
* нужно современное оборудование: ПК с интернетом, смартфоны, оргтехника;
* требуется время на внедрение.

Учет пациентов в больнице представляет собой систему сбора, хранения и обработки информации о пациентах, их заболеваниях, лечении и других связанных данных. Эффективный учет обеспечивает качественное медицинское обслуживание, позволяет следить за состоянием здоровья пациентов и оптимизировать рабочие процессы в медицинских учреждениях.

Основным учетным документом в поликлинике является «Медицинская карта больного», которая заводится на каждого жителя района деятельности поликлиники, что обратился за медицинской помощью. Её составной частью является листок записи заключительного диагноза — данные к нему заносятся одновременно с внесением в статистический талон заключительного диагноза. Пример заключительного диагноза изображён на рисунке 5.

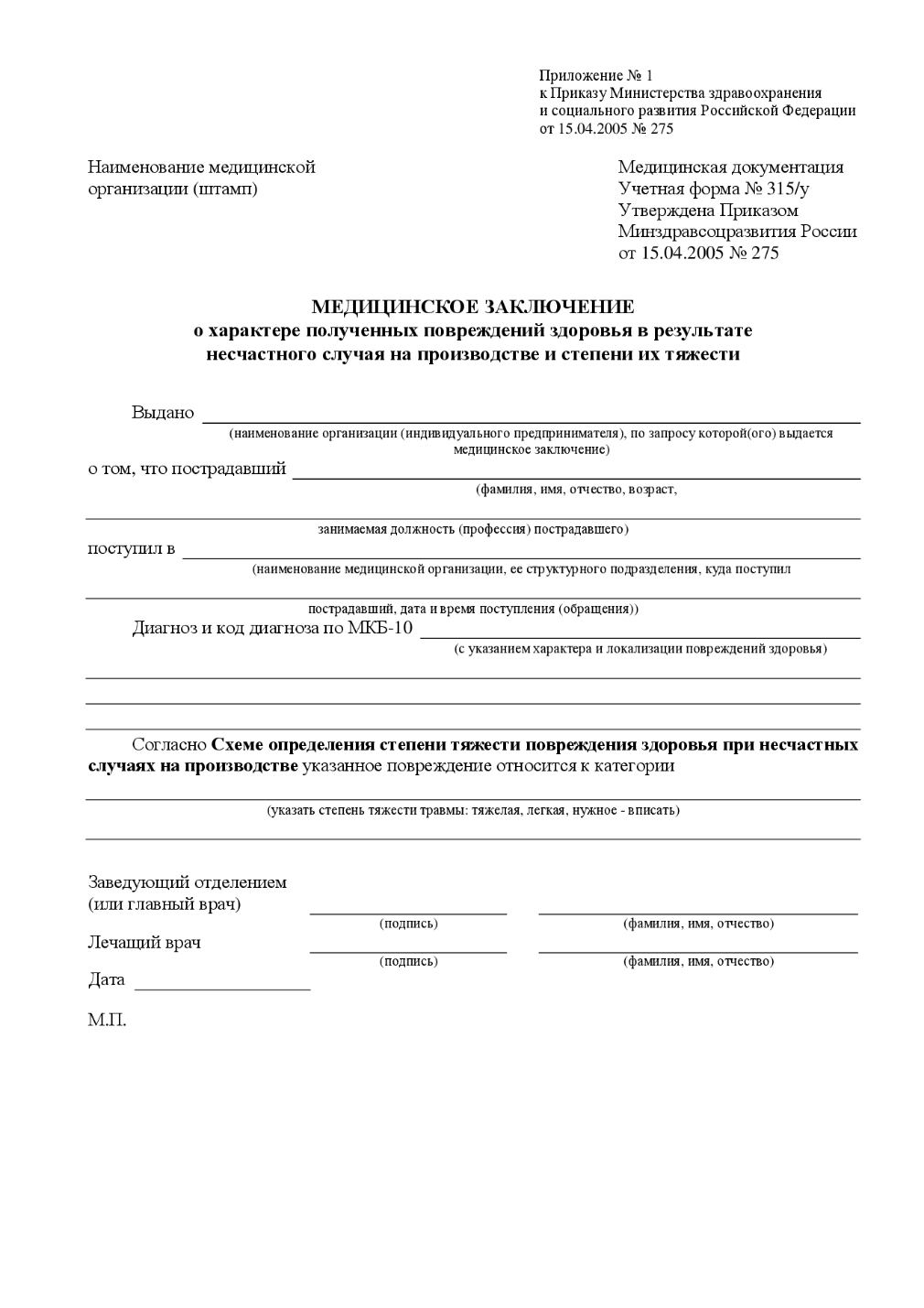


Рисунок 5 ­ Образец форма 315/у «Медицинское заключение о характере полученных повреждений здоровья в результате несчастного случая»

Данные учетных документов являются основой для анализа деятельности заведения, отдельного специалиста за определенный отчетный период. Они необходимы также для оперативного руководства работой.

Информация может записываться как на бумажных носителях, так и в компьютерной базе данных. В заведениях здравоохранения и органах управления накоплен значительный опыт автоматизированного учета информации, что позволяет расширить возможности анализа при уменьшении затрат времени на обработку полученных данных.

Отчет составляют врачи, которые работают в кабинетах учета и медицинской статистики или в информационно-аналитических отделах поликлиники.

На основе отчетных данных вычисляют и анализируют соответствующие показатели при участии руководителей заведения и соответствующих подразделов.

Разработка и анализ статистической информации позволяет врачам оценивать объем деятельности, качество и эффективность лечения, состояние здоровья населения.

Руководители заведения и подразделов имеют возможность оперативно управлять заведением, своевременно разрабатывать мероприятия по улучшению организации медицинского обеспечения населения.

В поликлинике есть следующие типы сотрудников:

* специалист регистратуры,
* врачи – специалисты,
* лаборант,
* главный врач.

Специалисты врачи делятся на более мелкие категории:

* терапевт,
* хирург,
* кардиолог,
* невролог,
* педиатр,
* офтальмолог,
* отоларинголог,
* дерматолог,
* эндокринолог,
* психиатр,
* инфекционист,
* реаниматолог,
* анестезиолог,
* и друие.

Приём и лечение пациента происходит следующим образом:

1. сначала пациент подходит в регистратуру и его данные заносят в базу данных, если их там нет. После этого его записывают на приём к специалисту.
2. после прихода пациента к специалисту специалист осматривает пациента и если диагноз ясен, выписывает лечение. Eсли диагноз не ясен, его записывают на дополнительные обследования или направляют к другому специалисту.

## 1.2 Анализ и оценка существующих программных продуктов

В сфере медицины, в связи с развитием технологий и цифровизацией, возрастает количество существующих программных продуктов. В основном, все приложения направлены на автоматизацию технологического процесса и учёта. Разработаны для операционных систем, такие как Windows и Linux.

### 1.2.1 Медицинская информационная система Renovato [4]

Медицинская информационная система Renovatio предоставляет широкий набор различных функциональных модулей, назначение которых — полноценная и эффективная автоматизация медицинского учреждения. Функции, которые выполняет эта система:

* история болезни, обследования, диагнозы;
* результаты лабораторных и диагностических исследований;
* планирование и учет лечения;
* электронные рецепты и назначения.

Преимущество этой системы заключается в очень хорошей безопасности. Сайт информационной системы представлен на рисунке 6.

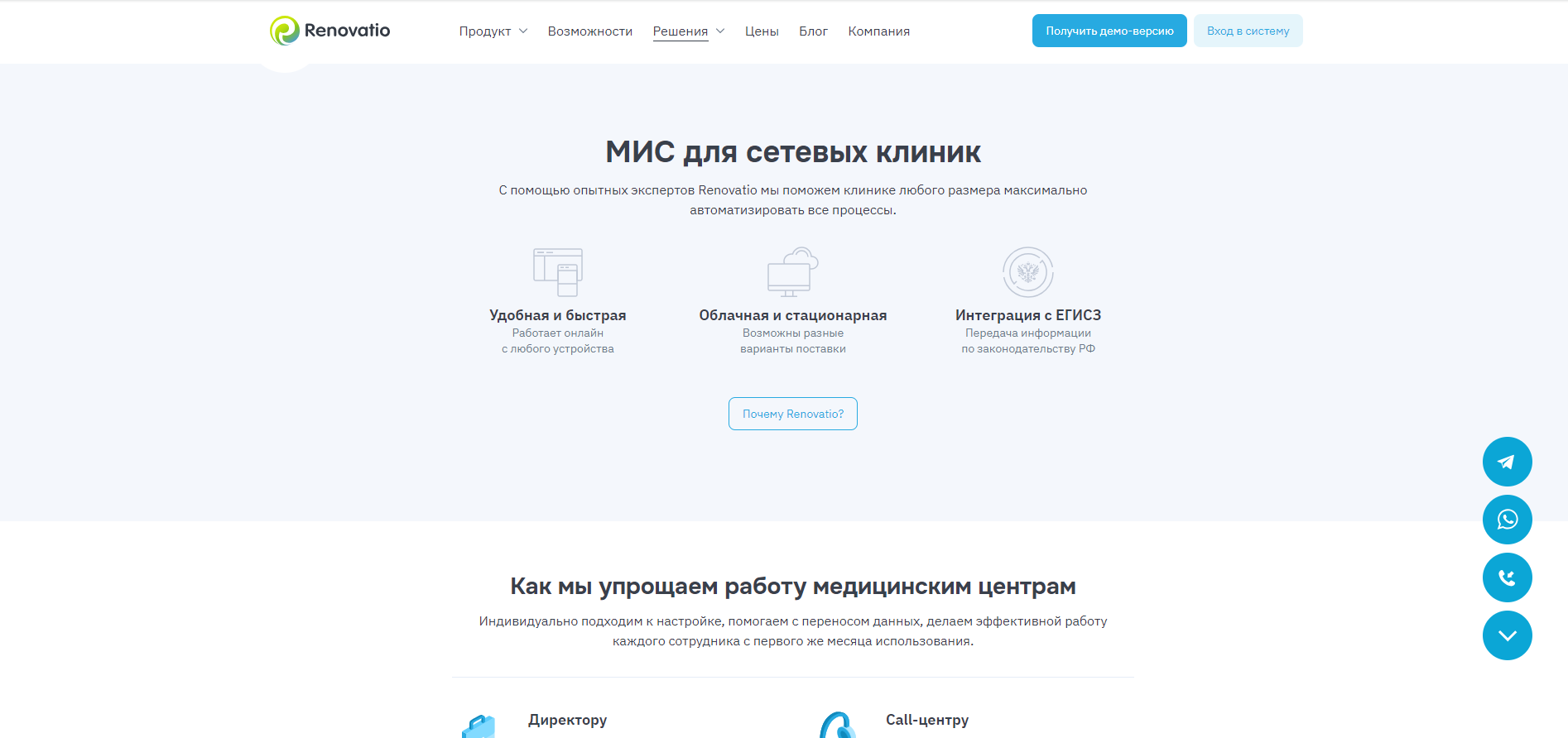
****

Рисунок 6 — Сайт информационной системы Renovato

### 1.2.2 Медицинская информационная система МедАнгел [5]

МедАнгел - это современный инструмент для эффективного цифрового управления медучреждением. Контроль работы на основании комплексного учета данных делает работу прозрачной и системной. Функции, которые выполняет эта система:

* регистрация пациентов,
* расписание и управление приёмами,
* онлайн-запись,
* базовые уведомления и отчётность.

Сайт информационной системы представлен на рисунке 7.

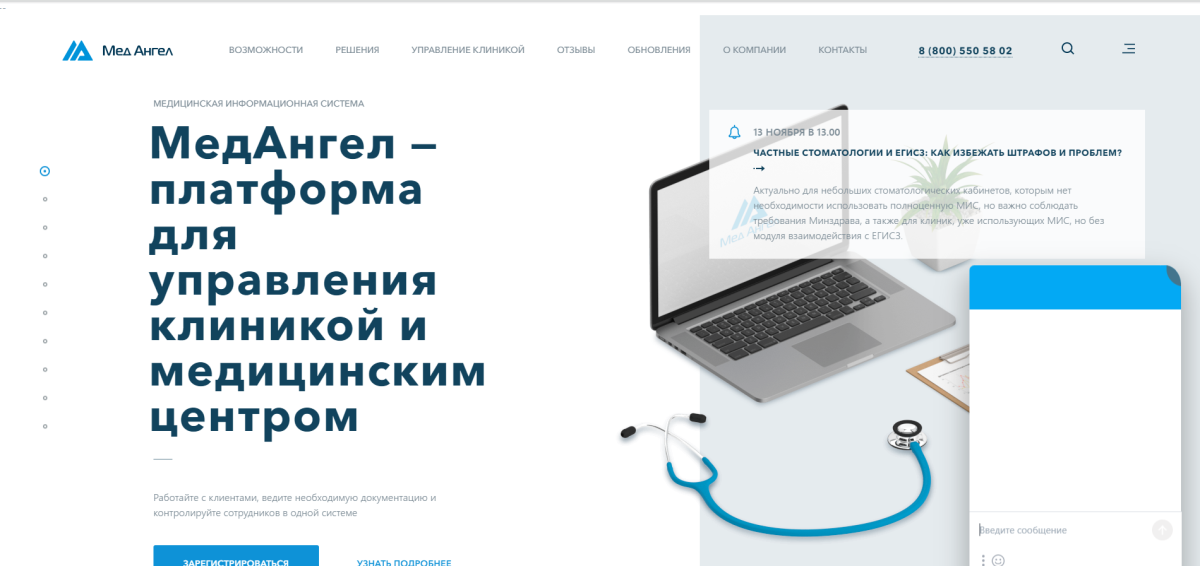


Рисунок 7 — Сайт информационной системы МедАнгел

### 1.2.3 Медицинская информационная система 1С:Медицина. Больница [6]

Комплексное решение 1С:Медицина Больница позволяет создать единое информационное пространство учета в медицинском учреждении. Программа подойдет для различных форм и организаций медицинской отрасли: областных, городских, районных больниц и поликлиники, а также диспансеров всех специализаций.

Функции, которые выполняет эта система:

* бухгалтерский учёт и финансы,
* регистрация пациентов,
* аптека, склады, закупки,
* расписание, графики смен,
* интеграция с 1C:Бухгалтерией.

Преимущество этой системы заключается в интеграции с 1C и соответственно удобного ведения бухгалтерии, отчётности. Информационная система 1С:Медицина Больница представлен на рисунке 8.

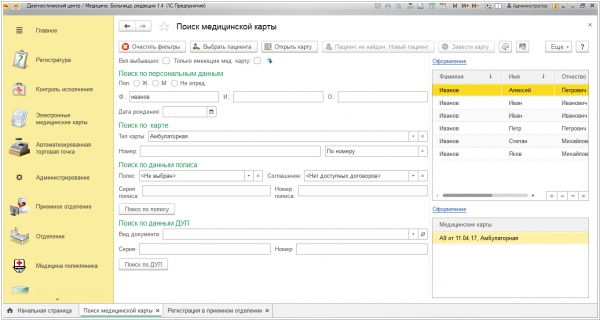


Рисунок 8 — Информационная система 1С:Медицина Больница

### 1.2.4 Медицинская информационная система med.me [7]

Med.me — облачная платформа, соединяющая пациентов, врачей, клиники, медицинские организации, страховые компании, фармкомпании и лидогенераторы.

Функции которые выполняет эта система:

* онлайн-запись через сайт и приложения,
* управление расписанием врачей,
* пациентский портал для просмотра записей,
* уведомления пациентам (sms, email).

Преимущество этой системы заключается в хорошем взаимодействии с пациентами. Сайт информационной системы представлен на рисунке 9.

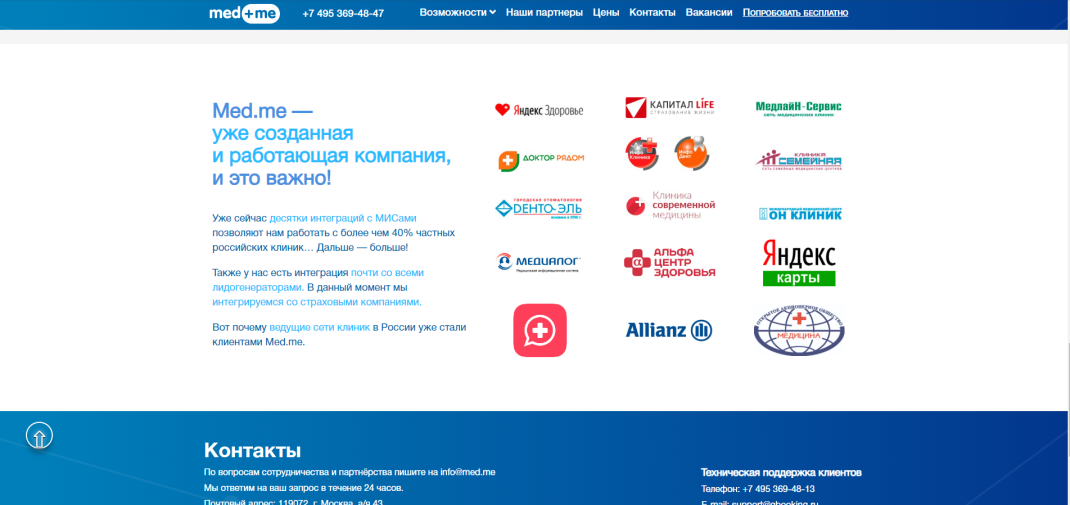


Рисунок 9 ­ Сайт информационной системы med.me

### 1.2.5 Медицинская информационная система ClinicIQ [8]

Проект ClinicIQ - новый взгляд на управление стоматологической клиникой. Это не просто программа для ведения учета клиники. Это инструмент, который поможет создать действительно современную клинику, первоклассный сервис для пациентов, увеличить доходы.

Функции, которые выполняет эта система:

* управление расписанием и приёмами,
* отчётность,
* онлайн-запись.

Сайт информационной системы представлен на рисунке 10.

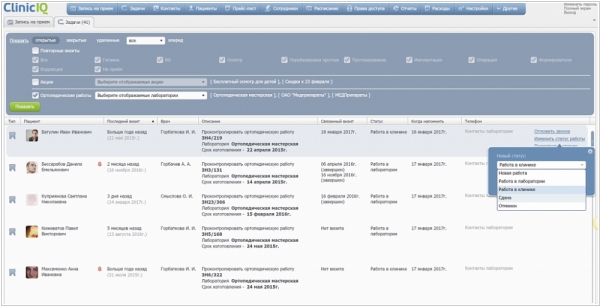


Рисунок 10 — Сайт информационной системы ClinicIQ

## 1.3 Постановка задачи

На основе анализа предметной области и аналогичных программных продуктов была поставлена цель – разработать систему учёта пациентов обращающихся в поликлинику.

Так же в системе необходимо реализовать Распределение обязанностей: Каждая роль выполняет строго определённые задачи, что обеспечивает эффективное функционирование процесса. Например, врач занимается медицинскими аспектами, а администратор — организационными. Распределение задач позволяет избежать перегрузки сотрудников и ускоряет процесс обслуживания пациентов, каждая роль требует своей экспертизы. Например, администратор не проводит осмотры, а врач не занимается регистрацией пациента. И это приводит к уменьшению рисков ошибок, таких как потеря данных или неправильный диагноз.

Медицинское учреждение, которое было взято за основу проекта - это частная клиника. Она имеет штат специалистов и занимается приемом пациентов по предварительной записи. Оплата приема осуществляется после получения амбулаторной карты, которая оформляется в регистратуре. Информация о каждом пациенте хранится в базе данных больницы, поэтому имеется возможность проследить за всеми посещениями выбранного пациента данной клиники.

Информационная система «Учёт пациентов» позволяет проследить путь пациента от регистратуры до приема у врача, установления диагноза и назначения лечения.

Система предназначена для обработки данных о врачах, пациентах, приеме пациентов и лечении, должна выдавать отчеты по запросу администрации.

Перед приемом в больницу пациент проходит регистрацию. Если пациент уже был на приеме в данной больнице, тогда проводится поиск по базе и данного пациента записывают на прием и выдают амбулаторный лист. В амбулаторном листе указывается, в какое время и на который день назначен прием, к какому специалисту (специализация, фамилия врача), в каком кабинете будет осуществляться прием, стоимость приема и пометка оплачен прием или нет. Если же пациент впервые в данной больнице, тогда ему присваивается уникальный регистрационный номер, заносятся в базу данных (фамилия, имя и отчество, адрес и дата рождения), а затем уже происходит запись на прием к конкретному врачу и на конкретное время и выдается амбулаторный лист. Пациент должен быть зарегистрирован в системе до приема в больнице.

Один пациент может быть записан на прием к нескольким врачам, каждому приему назначается уникальный номер. Каждый врач может просмотреть расписание приемов на любой из выбранных дней.

После регистрации пациента, в назначенный день и время он приходит на прием к специалисту. Врач осуществляет поиск и выбор данного пациента, выбирает прием, и исходя из выбранного приема, делает запись о диагнозе и назначении лечения данному пациенту. При повторном приеме происходит редактирование истории болезни.

Целью дипломного проекта является разработка информационной системы для учёта пациентов в больнице. Информационная система должна иметь следующие возможности для специалистов:

* просмотр графика работы сотрудников,
* просмотр списка записей к врачу,
* возможность добавления записи к врачу,
* возможность изменения и дополнения записи к врачу,
* возможность выписать направление на консультацию пациенту,
* возможность выписать направление на обследование пациенту,
* возможность выписать направление на исследование пациенту.

Для регистратуры информационная система должна выполнять следующие функции:

* просмотр графика работы сотрудников,
* просмотр списка записей к врачу,
* возможность добавления записи к врачу,
* возможность изменения и дополнения записи к врачу.

Для лаборанта информационная система должна выполнять следующие функции:

* просмотр графика работы сотрудников,
* просмотр списка исследований,
* добавление исследования пациенту,
* изменение исследований пациента.

Для главного врача информационная система должна выполнять следующие функции:

* просмотр графика работы сотрудников,
* добавление графика работы для сотрудника,
* изменение графика работы сотрудников,
* просмотр отчётов,
* добавление сотрудников,
* изменение данных сотрудников.

# 2 ОСОБЕННОСТИ ПОСТРОЕНИЯ ЧАСТИ «ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА»

## 2.1 Моделирование программного продукта

Моделирование информационной системы — это процесс создания абстрактной модели или представления информационной системы с целью анализа её структуры, функций, процессов и взаимодействий. Этот процесс включает в себя создание различных типов моделей, которые помогают специалистам в разработке, управлении и оптимизации информационных систем.

Диаграмма вариантов использования (прецедентов) в UML – диаграмма, отражающая отношения между акторами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне. Прецедент – возможность моделируемой системы (часть её функциональности), благодаря которой пользователь может получить конкретный, измеримый и нужный ему результат. Прецедент соответствует отдельному сервису системы, определяет один из вариантов её использования и описывает типичный способ взаимодействия пользователя с системой. Варианты использования обычно применяются для спецификации внешних требований к системе. Основное назначение диаграммы – описание функциональности и поведения, позволяющее заказчику, конечному пользователю и разработчику совместно обсуждать проектируемую или существующую систему. Диаграмма прецедентов для информационного продукта этого дипломного проекта представлена на рисунке 11.

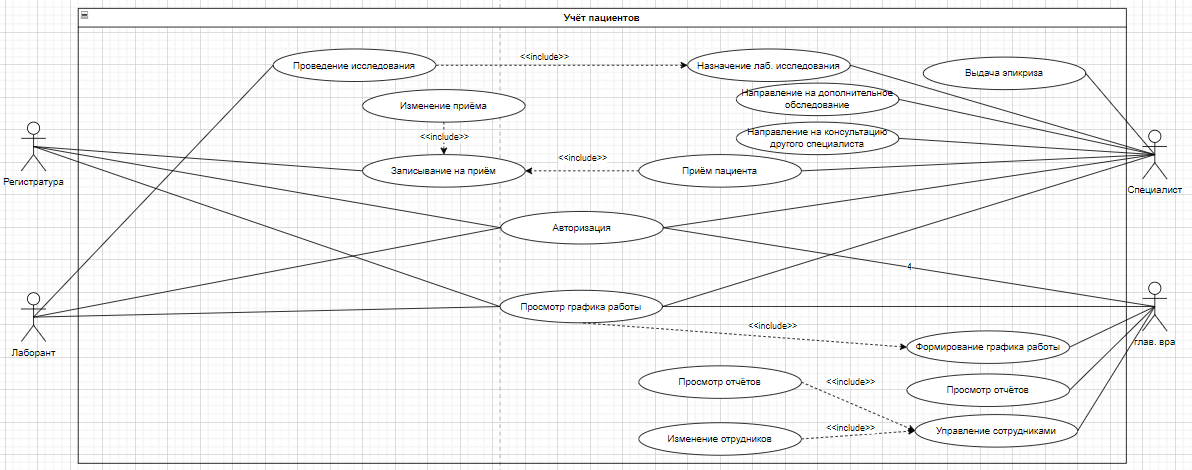


Рисунок 11 ­ Диаграмма прецедентов

Диаграмма активности UML позволяет более детально визуализировать конкретный случай использования. Это поведенческая диаграмма, которая иллюстрирует поток деятельности через систему. Диаграммы активности UML также могут быть использованы для отображения потока событий в бизнес-процессе. Они могут быть использованы для изучения бизнес-процессов с целью определения их потока и требований. Диаграмма активностей для информационного продукта этого дипломного проекта представлена на рисунке 12.

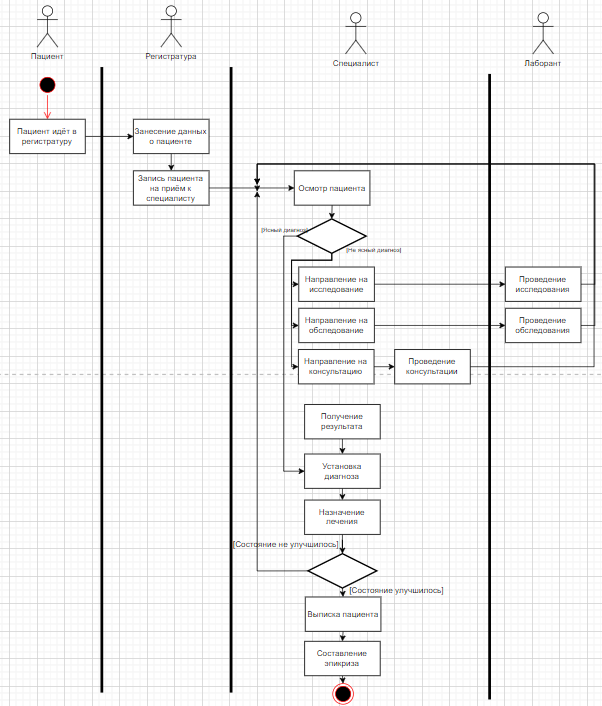


Рисунок 12 ­ Диаграмма активностей

Диаграмма последовательностей служит основным способом расшифровки последовательности действий в процессе выполнения того или иного варианта использования. Таким образом, диаграмма последовательностей всегда создается в привязке к варианту использования. Последовательная диаграмма дипломного проекта бизнес процесса «Запись пациента на приём» изображена на рисунке 13.

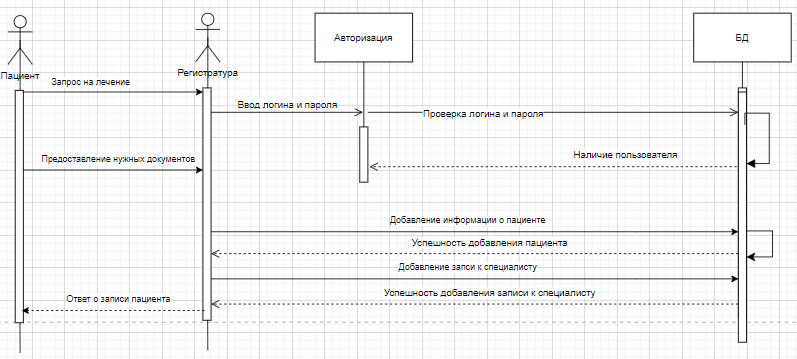


Рисунок 13 ­ Диаграмма последовательностей

## 2.2 Обоснование выбора средств разработки

Для реализации информационной системы были выбраны приложение WPF, СУБД Postgre Sql и специальная библиотека для их соединения NpgSQL. Описать их можно следующим образом:

1. СУБД PostgreSQL:

PostgreSQL является мощной и надежной реляционной базой данных, которая обладает множеством возможностей для обработки и хранения данных. Она поддерживает транзакции, многопользовательский доступ, средства обеспечения безопасности и масштабируемость. PostgreSQL также имеет открытый исходный код, обширное сообщество пользователей и хорошую документацию, что делает ее отличным выбором для проектов различного масштаба.

1. язык программирования С#, WPF:

C# является мощным и гибким языком программирования, который широко используется для разработки корпоративных приложений под платформу .NET. Он обладает богатой экосистемой инструментов и библиотек, а также хорошей производительностью и поддержкой объектно-ориентированного программирования.

1. способ связывания СУБД и языка программирования: библиотека npgsql:

Npgsql является поставщиком данных ADO.NET с открытым исходным кодом для PostgreSQL. Он позволяет программам, написанным на C# получать доступ к серверу базы данных PostgreSQL. Он реализован на 100% коде C#, бесплатен и имеет открытый исходный код. Также доступен поставщик Entity Framework Core, который предоставляет пользователям EF Core некоторые функции, уникальные для базы данных PostgreSQL.

## 2.3 Разработка базы данных

При создании базы данных наиболее важными являются задачи, связанные с созданием правильной логической структуры данных, обеспечивающей решение всего набора требуемых задач. Под правильной логической структурой в данном случае понимается структура, созданная с учетом особенностей организации хранения данных, используемых при решении требуемых задач. База данных, разработанная без учета того, как она в дальнейшем будет использоваться, оказывается, как правило, неуклюжей и неэффективной. Создание правильной логической структуры предусматривает комплексный анализ всех факторов, влияющих на формирование и обработку данных.

Цель логического проектирования – преобразование концептуальной модели на основе выбранной модели данных в логическую модель, не зависимую от особенностей используемой в дальнейшем СУБД для физической реализации базы данных. Логическая модель данных этого дипломного проекта изображена на рисунке 14.

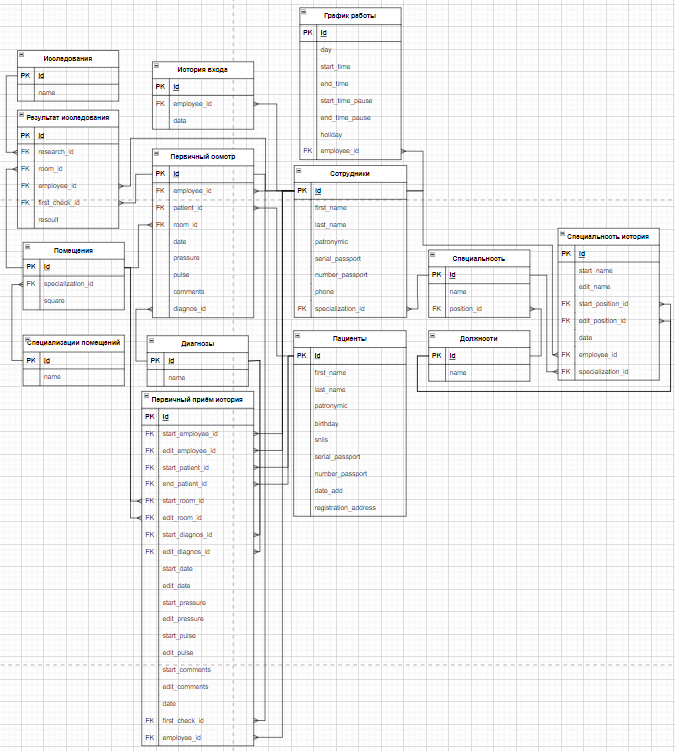


Рисунок 14 ­ Логическая модель базы данных

На физическом уровне производится выбор рациональной структуры хранения данных и методов доступа к ним, которые обеспечивает выбранная СУБД. На этом уровне решаются вопросы эффективного выполнения запросов к базе данных, для чего строятся дополнительные структуры, например индексы. В физической модели содержится информация обо всех объектах базы данных (таблицах, индексах, процедурах и др.) и используемых типах данных. Физическая модель зависит от конкретной СУБД. Одной и той же логической модели может соответствовать несколько разных физических моделей. Физическая модель базы данных этого дипломного проекта изображена на рисунке 15.

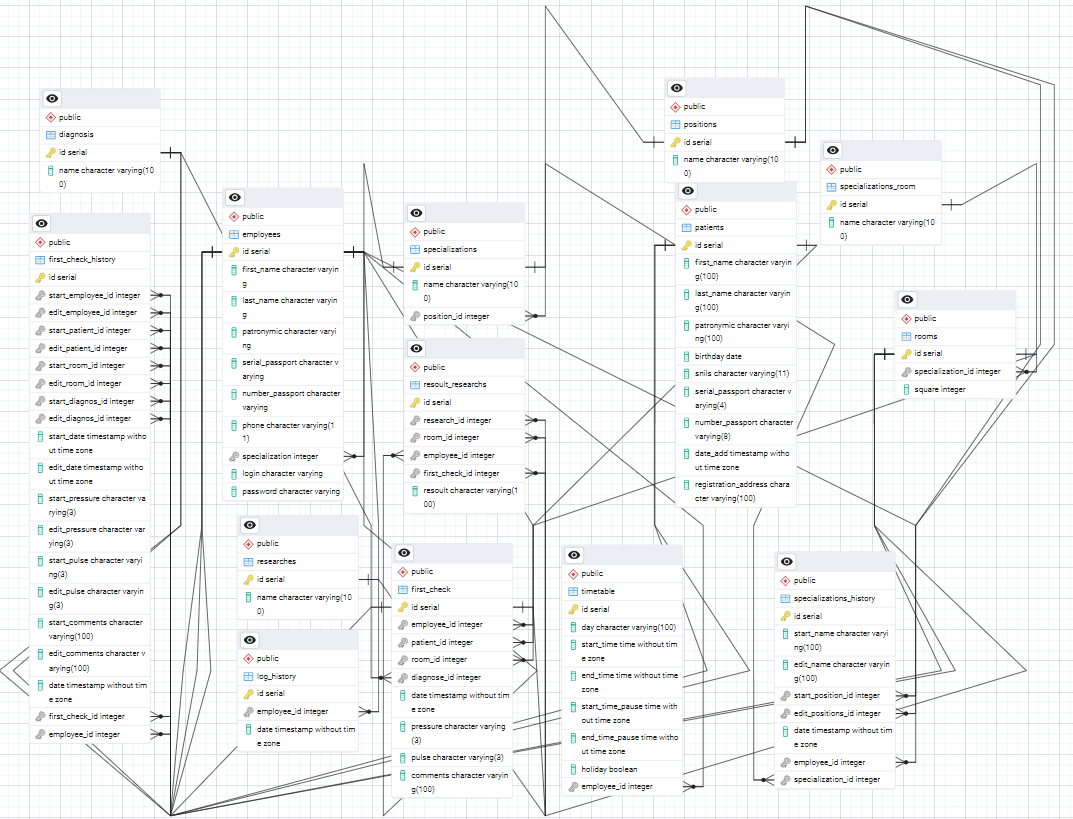


Рисунок 15 ­ Физическая модель базы данных

На этапе реализации производится создание базы данных и разработка приложений в выбранной СУБД. Например, запрос для создания таблицы employees, её характеристики представлены в таблице 1.

Create table employees (

id serial PRIMARY KEY,

first\_name character varying,

last\_name character varying,

patronymic character varying,

serial\_passport character varying,

number\_passport character varying,

phone character varying(11),

specialization integer,

login character varying,

password character varying

);

Таблица 1 ­ Характеристики таблицы employees

| Имя столбца | Тип данных | Свойства поля |
| --- | --- | --- |
| Id | Integer | Ключевое поле (первичный ключ)  Автоинкремент  Обязательное поле |
| first\_name | Varchar(100) | Обязательное поле  Максимум 100 символов |
| last\_name | Varchar(100) | Обязательное поле  Максимум 100 символов |
| patronymic | Varchar(100) | Необязательное поле  Максимум 100 символов |
| serial\_passport | Varchar(4) | Необязательное поле  Максимум 4 символа |
| number\_passport | Varchar(6) | Необязательное поле  Максимум 6 символов |
| phone | Varchar(11) | Обязательное поле  Максимум 11 символов |
| specialization | Integer | Внешний ключ таблицы specializations  Обязательное поле |
| login | Varchar(50) | Обязательное поле  Ограничение на уникальность |
| password | Varchar(100) | Обязательное поле |

Запрос для создания таблицы positions, характеристики этой таблицы представлены в таблице 2.

create table positions(

id serial primary key,

name varchar(100)

)

Таблица 2 ­ Характеристики таблицы positions

| Имя столбца | Тип данных | Свойства поля |
| --- | --- | --- |
| Id | Integer | Ключевое поле (первичный ключ)  Автоинкремент  Обязательное поле |
| Name | Varchar(100) | Обязательное поле Максимум 100 символов |

Запрос для создания таблицы specializations, характеристики этой таблицы представлены в таблице 3.

create table specializations(

id serial primary key,

name varchar(100),

position\_id int4 references(positions)

)

Таблица 3 ­ Характеристики таблицы specializations

| Имя столбца | Тип данных | Свойства поля |
| --- | --- | --- |
| id | Integer | Ключевое поле (первичный ключ)  Автоинкремент  Обязательное поле |
| name | Varchar(100) | Обязательное поле  Максимум 100 символов |
| Position\_id | Integer | Внешний ключ таблицы positions  Обязательное поле |

Запрос для создания таблицы patients, характеристики этой таблицы представлены в таблице 4.

create table patients(

id serial primary key,

first\_name varchar(100),

last\_name varchar(100),

patronymic varchar(100),

birthday date,

snils varchar(11),

serial\_passport varchar(4),

number\_passport varchar(8),

date\_add timestamp,

registration\_address varchar(100)

);

Таблица 4 ­ Характеристики таблицы patients

| Имя столбца | Тип данных | Свойства поля |
| --- | --- | --- |
| id | Integer | Ключевое поле (первичный ключ)  Автоинкремент  Обязательное поле |
| First\_name | Varchar(100) | Обязательное поле  Максимум 100 символов |
| Last\_name | Varchar(100) | Обязательное поле  Максимум 100 символов |
| patronymic | Varchar(100) | Обязательное поле  Максимум 100 символов |
| birthday | date | - |
| snils | Varchar(11) | Максимум 11 символа |
| Serial\_passport | Varchar(4) | Максимум 4 символа |
| Number\_passport | Varchar(8) | Максимум 8 симолов |
| Date\_add | timestamp | - |
| Registration\_address | Varchar(100) | Максимум 100 симолов |

Запрос для создания таблицы timetable, характеристики этой таблицы представлены в таблице 5.

create table timetable(

id serial primary key,

day varchar(100),

start\_time time,

end\_time time,

start\_time\_pause time,

end\_time\_pause time,

holiday bool,

employee\_id int4 references employees(id)

);

Таблица 5 ­ Характеристики таблицы timetable

| Имя столбца | Тип данных | Свойства поля |
| --- | --- | --- |
| id | Integer | Ключевое поле (первичный ключ)  Автоинкремент  Обязательное поле |
| day | Varchar(100) | Максимум 100 символов |
| start\_time | time | Максимум 100 символов |
| End\_time | time | Обязательное поле  Максимум 100 символов |
| Start\_time\_pause | time | - |
| End\_time\_pause | time | Максимум 11 символа |
| holiday | bool | - |
| Employee\_id | Integer | Обязательное поле  Внешний ключ таблицы employees |

Запрос для создания таблицы log\_history, характеристики этой таблицы представлены в таблице 6.

create table log\_history(

id serial primary key,

employee\_id integer,

date timestamp

)

Таблица 6 ­ Характеристики таблицы log\_history

| Имя столбца | Тип данных | Свойства поля |
| --- | --- | --- |
| id | Integer | Ключевое поле (первичный ключ)  Автоинкремент  Обязательное поле |
| Employee\_id | Integer | Обязательное поле  Внешний ключ таблицы employees |
| date | timestamp | - |

Запрос для создания таблицы specialization\_rooms, характеристики этой таблицы представлены в таблице 7.

create table specialization\_rooms(

id serial primary key,

name varchar(100)

)

Таблица 7 ­ Характеристики таблицы specializations\_room

| Имя столбца | Тип данных | Свойства поля |
| --- | --- | --- |
| id | Integer | Ключевое поле (первичный ключ)  Автоинкремент  Обязательное поле |
| name | Varchar(100) | Максимум 100 символов |

Запрос для создания таблицы rooms, характеристики этой таблицы представлены в таблице 8.

create table rooms(

id serial primary key,

specialization\_id int4 references specializations\_room(id),

square int4

)

Таблица 8 ­ Характеристики таблицы rooms

| Имя столбца | Тип данных | Свойства поля |
| --- | --- | --- |
| id | Integer | Ключевое поле (первичный ключ)  Автоинкремент  Обязательное поле |
| specialization\_id | Integer | Обязательное поле  Внешний ключ таблицы specializations\_room |
| square | integer | - |

Запрос для создания таблицы diagnosis, характеристики этой таблицы представлены в таблице 9.

create table diagnosis(

id serial primary key,

name varchar(100)

)

Таблица 9 ­ Характеристики таблицы diagnosis

| Имя столбца | Тип данных | Свойства поля |
| --- | --- | --- |
| id | Integer | Ключевое поле (первичный ключ)  Автоинкремент  Обязательное поле |
| name | Varchar(100) | Максимум 100 симолов |

Запрос для создания таблицы diagnosis, характеристики этой таблицы представлены в таблице 10.

create table first\_\_check(

id serial primary key,

employee\_id int4 references employees(id),

patient\_id int4 references patients(id),

room\_id int4 references rooms(id),

diagnose\_id references diagnosis(id)

date timestamp, pressure varchar(3),

pulse varchar(3), comments varchar(100)

);

Таблица 10 ­ Характеристики таблицы diagnosis

| Имя столбца | Тип данных | Свойства поля |
| --- | --- | --- |
| id | Integer | Ключевое поле (первичный ключ)  Автоинкремент  Обязательное поле |
| Employee\_id | Integer | Внешний ключ таблицы Employees |
| Patient\_id | Integer | Внешний ключ таблицы Patients |
| Room\_id | Integer | Внешний ключ таблицы Rooms |
| diagnose\_id | Integer | Внешний ключ таблицы Diagnosis |
| date | timestamp | - |
| pressure | Varchar(3) | Максимум 3 символа |
| pulse | Varchar(3) | Максимум 3 символа |
| comments | Varchar(100) | Максимум 100 символов |

Запрос для создания таблицы researches, характеристики этой таблицы представлены в таблице 11.

create table researches(

id serial primary key, name varchar(100)

);

Таблица 11 ­ Характеристики таблицы researches

| Имя столбца | Тип данных | Свойства поля |
| --- | --- | --- |
| id | Integer | Ключевое поле (первичный ключ)  Автоинкремент  Обязательное поле |
| name | Varchar(100) | Максимум 100 симолов |

Запрос для создания таблицы resoult\_researchs, характеристики этой таблицы представлены в таблице 12.

create table resoult\_researchs(

id serial primary key,

research\_id int4 references researches(id),

room\_id int4 references rooms(id),

employee\_id int4 references employees(id),

first\_check\_id int4 references first\_check(id),

resoult varchar(100)

);

Таблица 12 ­ Характеристики таблицы resoult\_researchs

| Имя столбца | Тип данных | Свойства поля |
| --- | --- | --- |
| id | Integer | Ключевое поле (первичный ключ)  Автоинкремент  Обязательное поле |
| Research\_id | Integer | Внешний ключ таблицы Researches |
| Room\_id | Integer | Внешний ключ таблицы Rooms |
| Employee\_id | Integer | Внешний ключ таблицы Employees |
| First\_check\_id | Integer | Внешний ключ таблицы First\_check |
| Resoult | Varchar(100) | Максимум 100 симолов |

Запрос для создания таблицы specializations\_history, характеристики этой таблицы представлены в таблице 13.

create table specializations\_history(

id serial primary key,

start\_name varchar(100),

edit\_name varchar(100),

start\_position\_id int4 references positions(id),

edit\_positions\_id int4 references positions(id),

date timestamp,

employee\_id int4 references employees(id),

specialization\_id int4 references specializations(id)

)

Таблица 13 ­ Характеристики таблицы specializations\_history

| Имя столбца | Тип данных | Свойства поля |
| --- | --- | --- |
| id | Integer | Ключевое поле (первичный ключ)  Автоинкремент Обязательное поле |
| Start\_name | Varchar(100) | Максимум 100 символов |
| Edit\_name | Varchar(100) | Максимум 100 символов |
| Start\_position\_id | Integer | Внешний ключ таблицы Positions\_id |
| Edit\_positions\_id | Integer | Внешний ключ таблицы Positions\_id |
| Date | timestamp | - |
| Employee\_id | Integer | Внешний ключ таблицы Employees |
| Specialization\_id | Integer | Внешний ключ таблицы specializations |

Запрос для создания таблицы first\_check\_history, характеристики этой таблицы представлены в таблице 14.

create table first\_check\_history(

id serial primary key,

start\_employee\_id int4 references employees(id),

edit\_employee\_id int4 references employees(id),

start\_patient\_id int4 references patients(id),

edit\_patient\_id int4 references patients(id),

start\_room\_id int4 references rooms(id),

edit\_room\_id int4 references rooms(id),

start\_diagnos\_id int4 references diagnosis(id),

edit\_diagnos\_id int4 references diagnosis(id),

start\_date timestamp,

edit\_date timestamp,

start\_pressure varchar(3),

edit\_pressure varchar(3),

start\_pulse varchar(3),

edit\_pulse varchar(3),

start\_comments varchar(100),

edit\_comments varchar(100),

date timestamp,

first\_check\_id int4 references first\_check(id),

employee\_id int4 references employees(id))

Таблица 14 ­ Характеристики таблицы first\_check\_history

| Имя столбца | Тип данных | Свойства поля |
| --- | --- | --- |
| id | Integer | Ключевое поле (первичный ключ)  Автоинкремент Обязательное поле |
| Start\_employee\_id | Integer | Внешний ключ таблицы Employees |
| Edit\_employee\_id | Integer | Внешний ключ таблицы Employees |
| Start\_patient\_id | Integer | Внешний ключ таблицы Employees |
| Edit\_patient\_id | Integer | Внешний ключ таблицы Employees |
| Start\_room\_id | Integer | Внешний ключ таблицы Employees |
| Edit\_room\_id | Integer | Внешний ключ таблицы Employees |
| Start\_diagnos\_id | Integer | Внешний ключ таблицы specializations |
| Edit\_diagnos\_id | Integer | Внешний ключ таблицы Employees |
| Start\_date | Timestamp | - |
| Edit\_date | Timestamp | - |

Продолжение таблицы 14

|  |  |  |
| --- | --- | --- |
| Имя столбца | Тип данных | Свойства поля |
| Start\_pressure | Varchar(3) | Максимум 3 символа |
| Edit\_pressure | Varchar(3) | Максимум 3 символа |
| Start\_pulse | Varchar(3) | Максимум 3 символа |
| Edit\_pulse | Varchar(3) | Максимум 3 символа |
| Start\_comments | Varchar(100) | Максимум 100 символов |
| Edit\_comments | Varchar(100) | Максимум 100 символов |
| Date | Timestamp | - |
| First\_check\_id | Integer | Внешний ключ таблицы Employees |
| Employee\_id | Integer | Внешний ключ таблицы Employees |

## 2.4 Проектирование меню и интерфейса программного продукта

Интерфейс – это граница между объектами системы, через которую они взаимодействуют. Выделяют человеко-машинный интерфейс, как способ общения человека с компьютером (формы, элементы управления, командная строка и др.). Также есть интерфейс прикладного программирования (API) – описания способов, с помощью которых одно программное приложение может взаимодействовать с другим.

Интерфейс пользователя, он же пользовательский интерфейс (UI – англ. user interface) – интерфейс, обеспечивающий передачу информации между пользователем-человеком и программно-аппаратными компонентами компьютерной системы. Основным предназначением приложения является предоставление пользователю необходимой функциональности. Wireframe — это наглядная схема, которая моделирует скелет проекта или технологии. Иногда Wireframe называют схемой страниц или экрана, на которой показано, как элементы связаны друг с другом и как они структурированы, эта диаграмма изображена на рисунке 16.

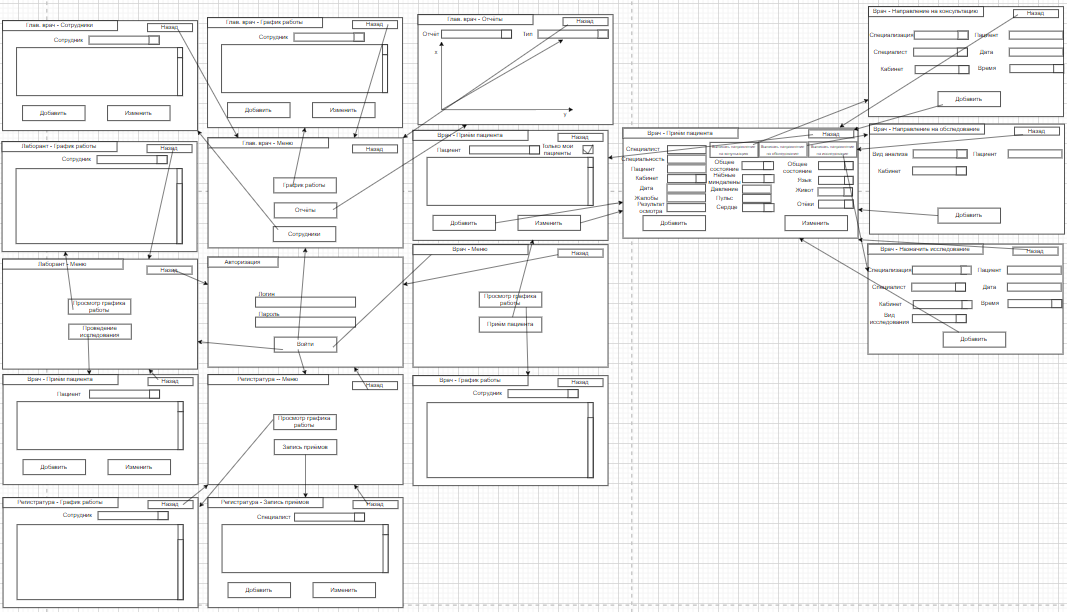


Рисунок 16 ­ Wireframe диаграмма

Пример интерфейса специалиста представлен на рисунке 17. XAML код окна для него представлен в листинге 1 приложения А, логика окна представлена в листинге 1 приложения Б

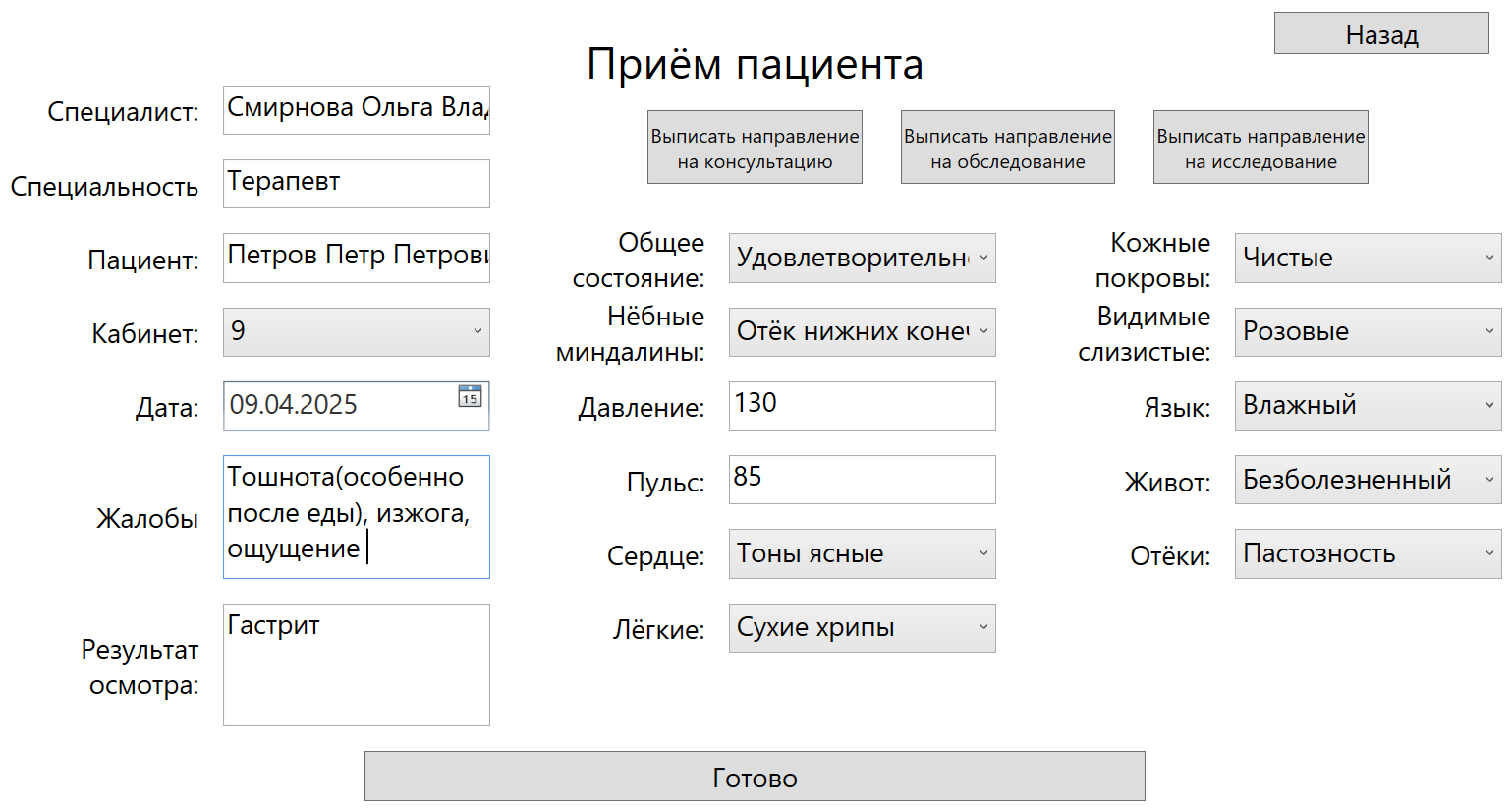


Рисунок 17 ­ Окно специалиста для приёма пациента

Ещё один пример интерфейса это окно просмотра графика работы сотрудников, его пример изображён на рисунке 18. XAML код представлен в листинге 2 приложения А, а логика в листинге 2 приложения Б.

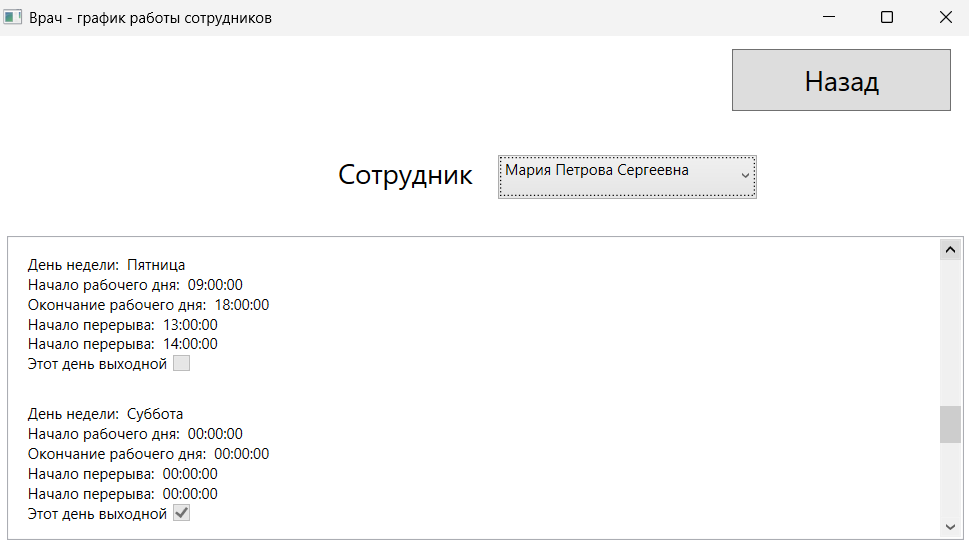


Рисунок 18 ­ Окно просмотра графика работы сотрудников

При создании интерфейса, к нему предъявляются следующие требования:

* наличие форм ввода данных,
* наличие форм вывода данных,
* удобная навигация,
* наличие меню,
* наличие всплывающих подсказок,
* защита от некорректного ввода данных,
* обработка исключительных ситуаций.

## 2.5 Спецификация программы

Спецификация программы – точная и полная формулировка задачи, содержащая информацию, необходимую для построения алгоритма (программы) решения этой задачи. В отличие от компьютерной программы спецификация обращена, прежде всего, к человеку и представляет собой описание в терминах, характерных для самой задачи, а не для ее реализации. Она служит основой дальнейшей детализации и разработки.

Каждый модуль отвечает за свои определенные действия и выполняет определенные функции. Модуль авторизации проверяет на подлинность введенные данные, а именно – логин и пароль. Если эти данные введены неверно, появляется сообщение об ошибке. Если данные успешно прошли проверку, пользователь попадает на модуль формы меню. Модуль меню представляет собой форму для просмотра других и перехода между ними, а также возврат на модуль авторизации.

## 2.6 Описание запросов при разработке программного продукта

Создание запросов в виде хранимых процедур, представлений должно сопровождаться обоснованием их целесообразности и необходимости.

Например, для работы запроса на список приёмов пациента был разработан следующий запрос: select concat(employees.last\_name, ' ',employees.first\_name,' ', employees.patronymic),

concat(patients.last\_name, ' ',patients.first\_name,' ', patients.patronymic),

first\_check.\*, specializations.name from first\_check

join employees on first\_check.employee\_id = employees.id

join patients on first\_check.patient\_id = patients.id

join specializations on employees.specialization = specializations.id

where concat(patients.last\_name, ' ',patients.first\_name,' ', patients.patronymic) = @patient

XAML код окна для вывода этого запроса представлен в листинге 1 приложения А, логика окна представлена в листинге 1 приложения Б. Окно изображено на рисунке 19.

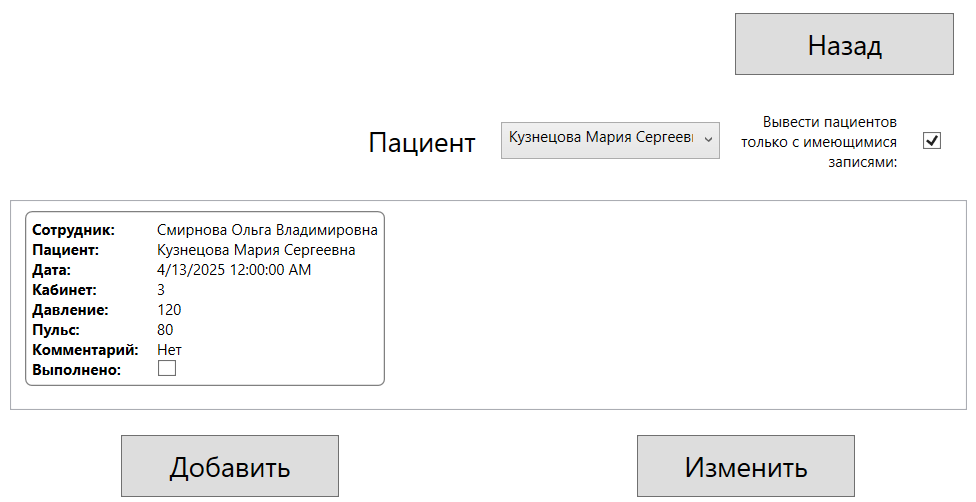


Рисунок 19 ­ Окно для вывода списка приёмов пациентов

## 2.7 Проектирование отчетов

Проектирование отчетов в приложении для автоматизации учёта пациентов является важной задачей, направленной на обеспечение возможности получения аналитической и управленческой информации, необходимой для контроля и планирования деятельности медицинского учреждения.

В рамках разработки приложения были предусмотрены следующие ключевые аспекты проектирования отчетности:

1. определение целей отчетности.

Основная цель отчётов — предоставить ответственным за это инструмент для мониторинга ключевых показателей работы учреждения, анализа данных о пациентах, оказанных услугах, загруженности специалистов, а также формирование бухгалтерских и управленческих сводок.

1. разработка структуры отчетов.

Были разработаны следующие типы отчетов:

* оперативные отчёты (сводные таблицы по количеству приёмов, услуг, посещений)
* аналитические отчёты (отчёты по доходности, статистике посещаемости, частоте использования услуг)
* отчёты по безопасности (журнал аудита входов в систему и действий пользователей).

1. реализация механизма формирования отчетов.

Использована встроенная функциональность WPF для генерации таблиц и вывода данных. Предусмотрена выгрузка отчетов в распространённые форматы (например, Excel, PDF) для дальнейшего использования.

## 2.8 Администрирование программного продукта

Администрирование программного обеспечения – это приведение программного обеспечения в соответствие с целями и задачами, для которых оно предназначено. Достигается путём управления, позволяющего минимизировать затраты времени и ресурсов, направляемых на управление системой, и в тоже время максимизировать доступность, производительность и продуктивность системы. Администрирование базы данных подразумевает под собой разграничение ролей пользователя. Это необходимо для сохранности и конфиденциальности информации. В дипломном проекте существуют следующие роли:

* регистратура,
* специалист,
* лаборант,
* главный врач.

Основной задачей регистратуры является оформление и запись пациента на приём, для этого ему доступно расписание врачей и управление записями на приём.

Основной задачей специалиста является проведение осмотра пациента и занесение данных об этом. Так же при не ясном диагнозе он может направить на консультацию к другим специалистам или на обследование/исследование лаборантам.

Основной задачей лаборанта является нужное обследование и исследование пациента для дальнейшего установления диагноза, если до этого это не удалось.

Основной задачей главного врача является управление сотрудниками, их расписанием и формирование отчётов о приёме пациентов.

Для того чтобы осуществлять все эти действия через приложение, и никто, кроме самого администратора, регистратора, врача или лаборанта не мог вносить изменения в данные, при запуске приложения пользователь попадает на форму «Авторизации» изображённом на рисунке 20. После работы с данными, у администратора, регистратора, врача и лаборанта имеется возможность выйти снова на форму «Авторизации», чтобы другой пользователь не смог изменить уже существующие или добавить новые данные.

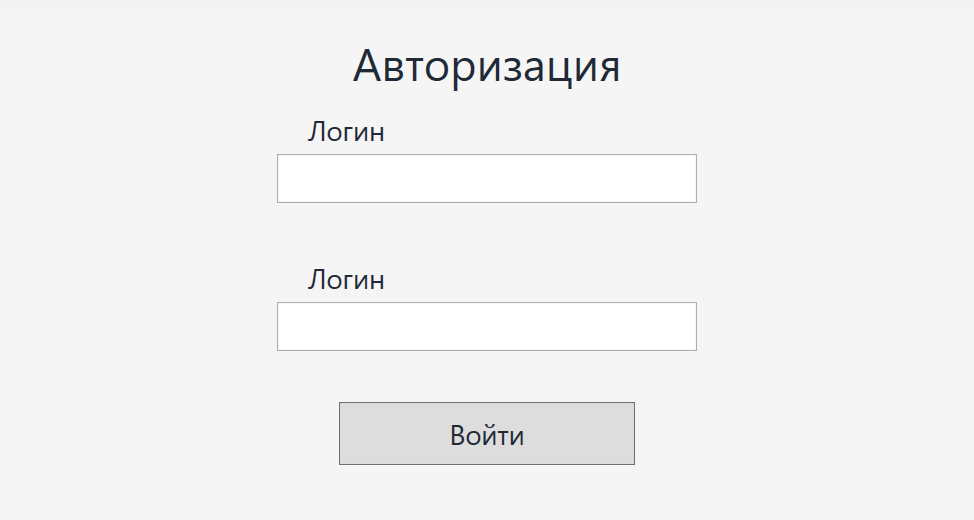


Рисунок 20 ­ Окно авторизации

На форме «Авторизации» пользователь должен ввести правильные логин и пароль, которые сверяются с наличием их в базе данных. Если эти данные совпадают, пользователь попадает в окно «Меню», откуда у него появляется возможность изменять данные. Иначе пользователь получит сообщение об ошибке и останется снова на форме авторизации.

# 3 ОСОБЕННОСТИ ПОСТРОЕНИЯ ЧАСТИ «ОПИСАНИЕ РАБОТЫ ПРОГРАММНОГО ПРОДУКТА»

## 3.1 Руководство пользователя

### 3.1.1 Введение

#### 3.1.1.1 Область применения

Данная информационная система предназначена для учёта пациентов в больницах и ведение их болезней.

#### 3.1.1.2 Краткое описание возможностей

* просмотр графика работы сотрудников,
* просмотр и управление пациентами,
* просмотр и управление приёмами пациентов,
* выписывание направлений на консультацию/исследование/обследование.

#### 3.1.1.3 Уровень подготовки пользователя

Для использования этой информационной системы достаточно базовых знаний управления компьютером.

### 3.1.2 Назначение и условия применения ПО

#### 3.1.2.1 Виды деятельности для автоматизации которых предназначено ПО

Приложение предназначено для учёта пациентов и хранение данных о них.

#### 3.1.2.2 Требования к составу и характеристикам технических средств

Минимальные технические требования(процессор Pentium G4560, оперативная память 4гб.).

#### 3.1.2.3 Требования к программной совместимости

Должна быть одна из следующих ОС(Windows 10, Windows 11).

#### 3.1.2.4 Требования к входной информации

Входной информацией являются sql запросы, которые срабатывают при нажатии на кнопки.

### 3.1.3 Подготовка к работе

#### 3.1.3.1 Состав и содержание дистрибутивного носителя данных

* исполнительный файл;
* sql backup для создания базы данных.

#### 3.1.3.2 Порядок установки ПО

1. скачать архив с сайта;
2. установить postgresql, если его нет;
3. распаковать архив
4. выполнить sql backup;

#### 3.1.3.3 Порядок проверки работоспособности установленного ПО

Запустить, последовательно проверить все функции.

### 3.1.4 Уровень подготовки пользователя

#### 3.1.4.1 Перечень всех выполняемых функций

* просмотр графика работы сотрудников,
* просмотр списка записей к врачу,
* возможность добавления записи к врачу,
* возможность изменения и дополнения записи к врачу,
* возможность выписать направление на консультацию/обследование исследование пациенту.

#### 3.1.4.2 Описание всех функций выполняемых по отдельности

* просмотр графика работы сотрудников (функция доступна в меню для всех пользователей);
* просмотр списка записей к врачу (функция доступна в меню только регистратуры и специалиста);
* возможность добавления записи к врачу (функция доступна в меню только для регистратуры);
* возможность изменения и дополнения записи к врачу (функция доступна в меню только для специалиста);
* возможность выписать направление на консультацию пациенту (функция доступна в меню только для специалиста);
* возможность выписать направление на обследование пациенту (функция доступна в меню только для специалиста);
* возможность выписать направление на исследование пациенту (функция доступна в меню только для специалиста).

### 3.1.5 Аварийные ситуации

#### 3.1.5.1 Действия в случае несоблюдения условий эксплуатаций, в том числе длительном отказе технических средств

При несоблюдении условий эксплуатаций возможны ошибки при запуске и исключения.

#### 3.1.5.2 Действия по восстановлению программ или данных при отказе магнитных носителей или при обнаружении ошибок в данных

Перезагрузить игру/компьютер, при отсутствии изменений переустановка игры.

### 3.1.6 Рекомендации по освоению

#### 3.1.6.1 Рекомендации по освоению ПО

Перед использованием рекомендуется изучить базовые знания работы с компьютером.

#### Описание контрольного примера, правила его запуска и выполнения

1. для начала работы нужно зайти в папку проекта и запустить файл “clinic.exe” (рис. 21).

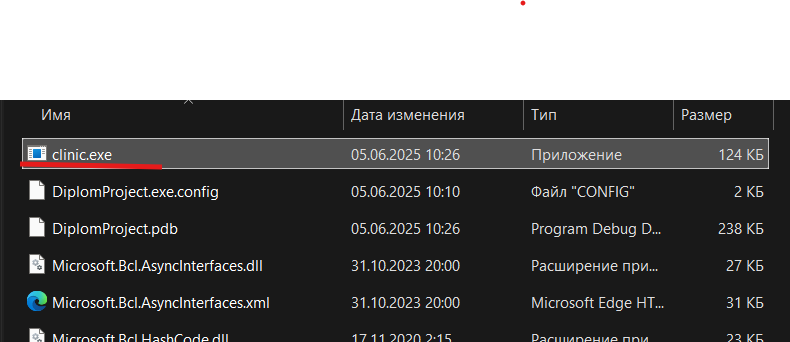


Рисунок 21 ­ Нужный файл для запуска

1. после запуска приложения нужно ввести логин и пароль сотрудника. После этого нужно нажать кнопку «Войти» (рис. 22).

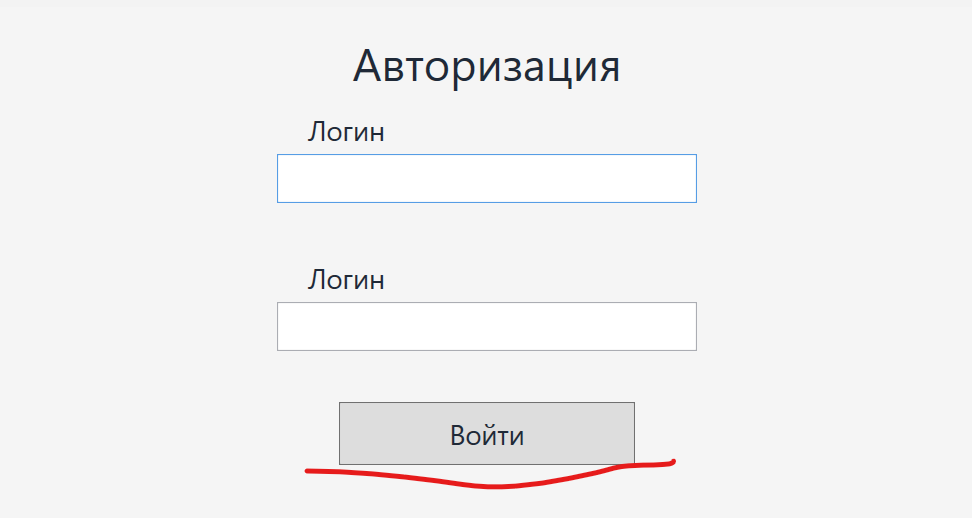


Рисунок 22 ­ Кнопка для входа в систему

1. после успешного входа закроется окно авторизации и появится меню сотрудника, соответствующего его роли.

## 3.2 Руководство программиста

### 3.2.1 Общие сведения

Название руководства: руководство системного программиста для информационной системы по учёту пациентов

Цель: Предоставление инструкций для поддержки, модификации и расширения функциональности приложения.

Область применения: Программное обеспечение для учёта пациентов.

### 3.2.2 Обязанности и функции системного программиста

* поддержка кода приложения,
* исправление ошибок в логике работы программы,
* внедрение новых функций в соответствии с требованиями пользователей.

### 3.2.3 Требования к квалификации и образованию системного программиста

* высшее техническое образование в области программирования, информационных технологий или смежных дисциплин;
* знание C#, XAML, WPF и PostgreSQL;
* навыки работы с системами контроля версий (например, Git).

### 3.2.4 Описание профессиональных навыков и знаний, необходимых для работы

* глубокие знания языка программирования C#,
* умение проектировать удобный пользовательский интерфейс с использованием WPF и XAML,
* знание принципов ООП и паттернов проектирования,
* опыт работы с обработкой исключений и логированием.

### 3.2.5 Описание процедур и методов разработки программного обеспечения

* **архитектура проекта:**
* файлы XAML используются для проектирования пользовательского интерфейса,
* логика приложения размещена в соответствующих .CS файлах.
* **добавление нового функционала:**
* определите требуемую таблицу или данные в базе,
* создайте соответствующие методы в коде, включая валидацию данных,
* обновите интерфейс, добавив новые элементы управления.

### 3.2.6 Инструкции по использованию специализированного программного обеспечения

* среда разработки: Visual Studio (2022 и выше),
* версия PostgreSQL не меньше 17.0.

### 3.2.7 Информация о стандартах и методах тестирования программного обеспечения

* для тестирования используйте встроенные инструменты Visual Studio (Test Explorer),
* проверяйте интерфейс в разных разрешениях экрана.

### 3.2.8 Контроль качества разработки и сопровождения программного обеспечения

* используйте систему контроля версий Git для отслеживания изменений,
* осуществляйте код-ревью перед добавлением новых функций,
* регулярно обновляйте документацию по проекту.

### 3.2.9 Рекомендации по эффективному управлению проектами разработки ПО

* используйте Agile или Scrum методологии для управления задачами,
* разделяйте крупные задачи на мелкие, управляемые этапы,
* устанавливайте сроки и отслеживайте прогресс в системах управления проектами (например, Jira, Trello).

### 3.2.10 Инструкции по документированию и поддержке программного продукта

* документируйте каждую добавленную функцию, описывая её назначение и метод использования,
* храните все файлы документации в одном репозитории с исходным кодом,
* предоставляйте инструкцию по установке и настройке системы для новых разработчиков.

# 4 ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ

Информационная безопасность комплексная задача, которая направлена на обеспечение безопасности и реализуется внедрением системы безопасности.

Информационная безопасность у приложений - это совокупность мер, направленных на защиту приложений от угроз, связанных с неправомерным доступом, повреждением или уничтожением данных или кода приложения. Для обеспечения информационной безопасности у приложений необходимо учитывать следующие аспекты:

## 4.1 Виды возможных угроз программного продукта

Программный продукт автоматизации учёта пациентов подвергается множеству угроз информационной безопасности. Эти угрозы могут быть классифицированы по источникам возникновения и характеру воздействия. Основные виды возможных угроз:

1. угрозы конфиденциальности данных

* несанкционированный доступ к данным,
* получение неавторизованного доступа к персональным данным пациентов, в том числе со стороны внешних злоумышленников,
* перехват данных при передаче,
* использование незащищённых каналов связи может привести к перехвату данных,
* утечка данных из-за ошибок конфигурации,
* ошибки при настройке системы или инфраструктуры (например, неправильные права доступа, открытые порты) могут привести к случайной утечке данных.

1. угрозы целостности данных

* неавторизованные изменения данных,
* внесение изменений в данные пациентов без соответствующих прав (например, фальсификация истории болезни),
* ошибка в работе программного обеспечения,
* ошибки в коде или сбои в аппаратной части могут привести к повреждению или потере данных,
* вирусные и вредоносные программы,
* заражение системы вирусами может нарушить целостность данных, повредить или уничтожить информацию.

1. угрозы доступности данных и услуг

* ddos-атаки,
* отказ оборудования,
* сбои в работе серверов, сетевого оборудования, систем хранения данных,
* программные сбои и ошибки конфигурации,
* некорректные настройки могут привести к недоступности системы для пользователей,
* заражение системы программами-вымогателями с блокировкой доступа к данным.

1. угрозы, связанные с человеческим фактором

* неверные действия сотрудников (случайное удаление или изменение данных),
* социальная инженерия,
* получение доступа к системе через обман сотрудников,

1. угрозы со стороны внешней инфраструктуры

* надёжность поставщиков облачных сервисов,
* проблемы у поставщиков облаков или дата-центров могут повлиять на работоспособность системы,
* перебои в электропитании или сетевом соединении.

## 4.2 Информационная безопасность программного продукта

Информационная безопасность программного продукта автоматизации учёта пациентов является критически важной для защиты персональных данных пациентов и обеспечения соблюдения требований законодательства и стандартов.

Для обеспечения безопасности в системе реализуются следующие меры:

1. **защита данных**

Данные пациентов хранятся в зашифрованном виде с использованием современных алгоритмов.

Система поддерживает контроль целостности данных с использованием контрольных сумм и цифровых подписей.

1. **управление доступом**

Используется система аутентификации и авторизации, включая:

* ролевую модель доступа с гибкой настройкой прав пользователей;
* логирование всех действий пользователей с возможностью последующего аудита.

1. **мониторинг и анализ безопасности**

Реализована система аудита и мониторинга, позволяющая отслеживать все действия сотрудника.

1. **защита от угроз**

* система защищена от распространённых видов атак (sql-инъекции, xss, csrf) за счёт применения современных методов разработки и фильтрации данных.
* ограничение количества одновременных сессий и времени их активности.

1. **соответствие нормативным требованиям**

Программный продукт соответствует локальным требованиям к защите информации, включая классификацию информационных систем.

1. **резервное копирование и восстановление**

* Автоматизированное создание резервных копий данных с шифрованием.
* Возможность быстрого восстановления данных в случае сбоя или утраты.

# 5 УПРАВЛЕНИЕ ПРОЕКТОМ

## 5.1 Понятие проекта, необходимость управления проектом [9, 10]

Проектный менеджмент – это процесс управления проектами, предполагающий их планирование, организацию и реализацию для достижения поставленных целей в конкретно заданных временных и бюджетных рамках.

Проектный менеджмент позволяет пройти путь от точки А к точке В по наиболее выгодному для компании маршруту. Он помогает сфокусироваться на конкретной цели, правильно распределить ресурсы и своевременно получить желаемый результат. Проектный менеджмент позволяет решать следующие задачи:

* управлять целями проекта,
* контролировать бюджет, чтобы избежать незапланированных расходов,
* определять сроки реализации проекта,
* распределять задачи и устанавливать ответственных лиц,
* предусматривать возможные риски и заранее выявлять угрозы,
* вести контроль за процессом реализации проекта,
* налаживать эффективную коммуникацию между членами команды для достижения поставленных целей.

С помощью проектного менеджмента разрабатывают четкий план действий для достижения целей, управляют коммуникацией между членами команды и отделами, а также синхронизируют их деятельность для повышения эффективности работы. Проектный менеджмент предполагает распределение зон ответственности между сотрудниками согласно их навыкам и разумное использование ресурсов компании.

Проектный менеджмент позволяет контролировать расходы, устанавливать временные рамки, формировать бюджет и предусматривать трудности, которые могут возникнуть в процессе работы.

## 5.2 Постановка цели проекта по SMART [11]

Разработка информационной системы требует четкой формулировки целей в соответствии с методологией SMART. Данный подход обеспечивает конкретность, измеримость, достижимость, релевантность и определенность во времени поставленных задач, что является критически важным для успешной реализации проекта.

Конкретность (Specific): цель проекта заключается в создании интерактивного мобильного приложения, которое позволит абитуриентам, студентам и гостям колледжа получать информационно-образовательный контент в формате дополненной реальности. Приложение должно предоставлять сведения об инфраструктуре учебного заведения, истории его развития, особенностях образовательных программ и внеучебной деятельности через визуализацию AR-объектов при сканировании специальных маркеров, расположенных на территории колледжа

Измеримость (Measurable): успешность реализации проекта будет оцениваться по следующим количественным показателям:

* полнота охвата ключевых объектов,
* скорость работы приложения (время отклика не более 1,5 секунд при стандартных условиях),
* количество поддерживаемых платформ (как минимум Windows 10 и Windows 11),
* охват функциональных возможностей (100% запланированного функционала согласно техническому заданию).

Достижимость (Achievable): поставленная цель является реалистичной при имеющихся ресурсах:

* доступ к необходимому аппаратному обеспечению для тестирования,
* использование бесплатных и открытых решений для разработки системы,
* наличие аналогичных успешных кейсов в целевой сфере.

Релевантность (Relevant): разработка приложения соответствует стратегическим задачам образовательного учреждения:

* внедрение инновационных технологий в процесс учёта,
* соответствие трендам цифровизации.

Ограниченность во времени (Time-bound): реализация проекта планируется в следующие временные рамки:

* разработка прототипа - 2 месяца,
* создание полной версии приложения - 4 месяца,
* тестирование и отладка - 1 месяц,
* внедрение - 1 месяц,
* полный цикл разработки не должен превышать 8 месяцев.

Дополнительным критерием успешности проекта станет положительная обратная связь от пользователей и отсутствие критических замечаний по работе приложения после выпуска первой стабильной версии.

Такая постановка цели по методологии SMART создает четкие ориентиры для разработки, позволяет объективно оценивать прогресс и обеспечивает достижение значимого результата, соответствующего потребностям колледжа и современным тенденциям в области образовательных технологий.

## 5.3 Анализ заинтересованных сторон проекта [12]

Составляющей частью любого проекта являются заинтересованные стороны. Данным понятием обозначают лиц и организации, которые активно вовлечены в реализуемый проект.

Помимо того, заинтересованными сторонами считают лиц, чьи интересы могут быть тем или иным образом затронуты в результате реализации проекта.

Для того чтобы зафиксировать круг этих лиц, в рамках практики управления проектами был разработан специальный документ – реестр заинтересованных сторон проекта. Этот реестр предназначен для отражения значимой информации о всех заинтересованных сторонах проекта.

Как правило, содержание реестра заинтересованных сторон проекта представлено следующими данными: фамилия, имя, отчество физического лица, или наименование юридического лица (организации); должность физического лица; адрес проживания физического лица, или юридический адрес организации; роль, которую данная заинтересованная сторона играет в этом проекте; контактная информация (номер телефона, адрес электронной почты и др.); основные требования, которые заинтересованная сторона предъявляет к этому проекту; основные ожидания заинтересованной стороны, связанные с реализацией этого проекта; уровень потенциального влияния, которое заинтересованная сторона может оказать на ход реализации проекта; этапы (фазы) жизненного цикла проекта, которые представляют для заинтересованной стороны наибольший интерес.

Реестр заинтересованных сторон представляет собой документ, который разрабатывают на стадии инициации проекта. Кроме информации о непосредственно заинтересованных сторонах, реестр включает в себя титульную информацию о проекте. Реестр заинтересованных сторон представлен в таблице 15.

Таблица15 – Реестр заинтересованных лиц

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Стейкхолдеры | Внешний/внутренний | Что ему нужно? | Что нужно нам? | Риск неосведомленности | Влияние на проект (от 0 до 5) | Заинтересованность (от -5 до +5) |
| Студент | Внешний | Успешное выполнение проекта в сроки в соответствии с требованиями | | Сдать проект в сроки в соответствии с требованиями | 5 | 5 |
| Заведующий отделением в лице Сидоровой Н.В. | Внешний | Успешное выполнение проекта в сроки в соответствии с требованиями | Утверждение темы проекта. Подпись диплома. | Срыв сроков | 4 | 4 |
| Руководитель дипломного проекта в лице Тутарова В.Д. | Внешний | Успешное выполнение проекта в сроки в соответствии с требованиями | Выдача теоретического материала | Срыв сроков | 4 | 4 |
| Пользователи | Внешний | Полностью функционирующая пропускная система для пропуска машин | Осуществление заказов, обратная связь | Продукт не будет функционировать, как должен | 1 | 2 |

## 5.4 Устав проекта

Устав – это официальный документ, в котором приводится краткое, но емкое описание проекта: его целей, порядка реализации, заинтересованных лиц. Это важнейший элемент планирования проекта, т.к. устав используют в течение всего жизненного цикла проекта [13].

Цель проекта: Создание информационной системы для учёта деятельности поликлиники, в частности учёт пациентов [13].

Cрок реализации проекта: 14.06.2025 г.

Требования:

* создание базы данных (PostgreSQL) с информацией о пациентах,
* создание приложения WPF с удобным интерфейсом,
* вход в систему должен быть разграничен правами доступа.

Результат завершения проекта:

* готовая информационная система для учёта пациентов.

Ограничения и допущения:

* поддержка только операционных систем Windows 10 и Windows 11,
* ограниченные тестовые данные.

Ключевые события:

* получение темы проекта для последующей ее реализации,
* постановка задач для успешного завершения проекта,
* определение средств разработки,
* установка средств разработки,
* проектирование систем управления базами данных,
* разработка backend части проекта,
* заполнение систем управления базами данных,
* составление руководства программиста,
* составление руководства пользователя,
* написание разделов пояснительной записки,
* тестирование функционала системы,
* проверка грамотности системы,
* исправление ошибок.

Ориентировочный бюджет проекта: 0 рублей.

Ключевые заинтересованные стороны:

* студент,
* заведующий отделением в лице Сидоровой Н.В,
* руководитель дипломного проекта в лице Тутарова В.Д.,
* пользователи.

Риски проекта:

* типовые риски предприятия, известные к моменту начала проекта – согласно карте рисков.

## 5.5 Планирование проекта

Планирование проекта – это процесс определения задач и расстановки их приоритетов, распределения затрат, объема и графика работ. Этот процесс необходим для выполнения проекта в установленный срок и в рамках бюджета. Данные установки разрабатываются чаще всего в виде дорожной карты и характеризуют будущее, а также по возможности выражаются численно набором показателей, являющихся ключевыми для определенного уровня управления [14].

На этапе планирования проекта решается ряд задач [14]:

* утверждение плана проекта,
* согласование плана основными участниками проекта,
* создание и уточнение процедур управления проектом,
* уточнение и детализация целей и задач проекта,
* уточнение объема работ и состава проекта,
* уточнение порядка взаимодействия в проектной команде, а также между внешней средой и проектной командой,
* оценка рисков и формирование плана реагирования на них,
* уточнение потребности в ресурсах, план ресурсного обеспечения проекта (или отдельных этапов проекта),
* разработка бюджета проекта и реального расписания (или отдельных его этапов).

WBS — work breakdown structure, или структурная декомпозиция, иерархическая структура работ. С помощью структурной декомпозиции большие и сложные проекты делят на маленькие простые части. Например, проект разбивают на блоки, блоки — на задачи, задачи — на подзадачи [15]. Получается иерархическая схема, в которой отражены все работы по проекту (рис. 23).

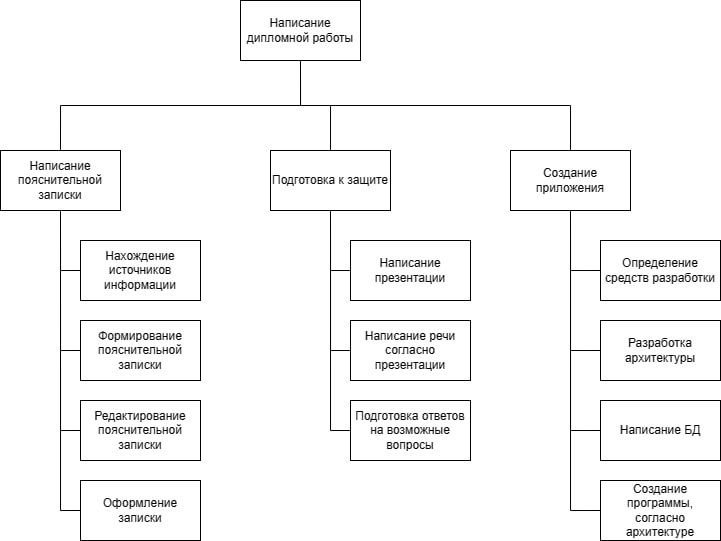


Рисунок 23 ­ WBS диаграмма

Диаграмма Ганта — это инструмент, позволяющий визуализировать и управлять проектами, структурировать их выполнение и видеть общую картину задач, как личных, так и организации [16]. Первая часть диаграммы Ганта представлена на рисунке 24, вторая часть диаграммы изображена на рисунке 25.

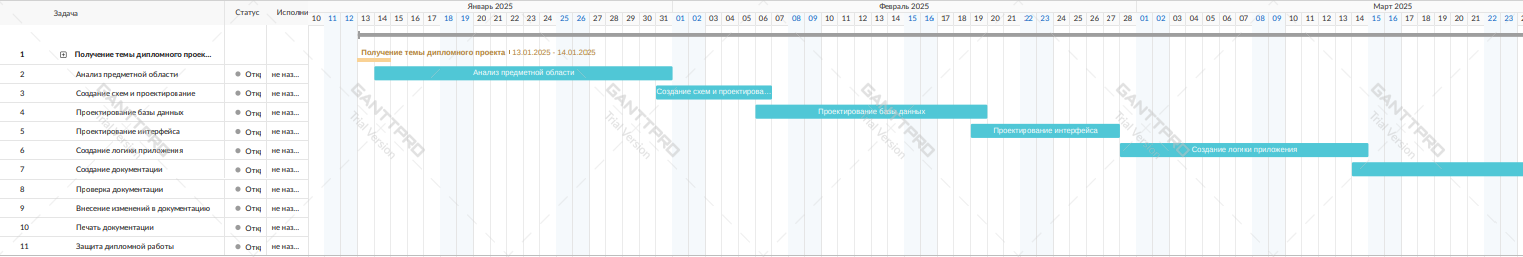


Рисунок 24 ­ Первая часть диаграммы Ганта

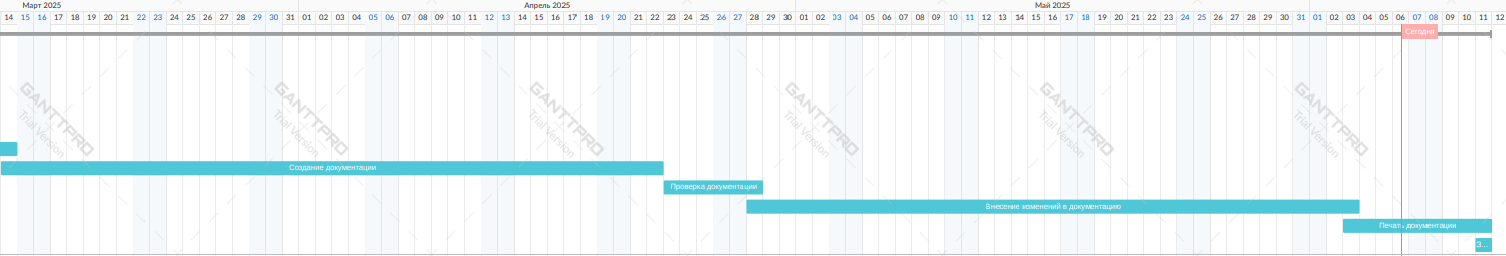


Рисунок 25 ­ Вторая часть диаграммы Ганта

## 5.6 Управление стоимостью проекта

Трудоемкость разработки программного обеспечения в чел. час определяется по формуле (1):

|  |  |
| --- | --- |
| Т = То+Ти+Та+Тп+Тотл+Тд, | (1) |

где Т – общие затраты труда, чел.час.

То – затраты труда на описание задачи

Ти – затраты на исследование предметной области

Та – затраты на разработку алгоритма решения задачи

Тп – затраты на программирование

Тотл – затраты на отладку программы

Тд – затраты на подготовку документации

Все составляющие определяются через условное число операторов Q:

|  |  |
| --- | --- |
| Q=q\*c\*(1+p), | (2) |

где q – число операторов (оператор – предписание о выполнении конкретных действий. Например, в языке программирования Паскаль содержится 11 операторов.)

c – коэффициент сложности задачи (принимается от 1,2 … 2)

p – коэффициент коррекции программы, учитывающий новизну проекта (для совершенно новой программы равен 0,1)

|  |  |
| --- | --- |
| Q = 1200\*1.5+(800\*1,3) = 2840 | (3) |

где То – принимаем в диапазоне от 30 до 40 чел.час (в зависимости от сложности продукта)

Ти определяется по формуле:

|  |  |
| --- | --- |
| Ти = (Q\*B) / (Sи\*k) | (4) |

где В – коэффициент увеличения затрат труда в следствие недостаточного описания задачи (1,2 – 1,5)

Sи – количество операторов, приходящихся на 1 чел.час ( 75-85 )

k – коэффициент квалификации работника, подробности на таблице 15

Таблица 15 - Квалификация работников

|  |  |
| --- | --- |
| Стаж | Коэффициент |
| до 2-х лет | 0,8 |
| 2- 3 года | 1,0 |
| 3-5 лет | 1,1 – 1,2 |
| 5 – 7 лет | 1,3 – 1,4 |
| свыше 7 лет | 1,5 – 1,6 |

|  |  |
| --- | --- |
| Ти = (2840\*1.4) / (80\*0.8) = 62,125 чел. Час | (5) |

Та рассчитывается по формуле:

Та = Q / (Sа\*k),

где Sа - количество операторов на разработку алгоритма решения задачи, приходящихся на 1 чел.час ( 20 – 25)

|  |  |
| --- | --- |
| Та = 2840/(22\*0,8) = 161,36 чел.час | (6) |

Затраты труда на программирование найдем по формуле:

|  |  |
| --- | --- |
| Тп = Q / (Sп\*k), | (7) |

где Sп - количество операторов на программирование, приходящихся на 1 чел.час (20 – 25)

|  |  |
| --- | --- |
| Тп = 2840 / (21\*0,8) = 169,04 чел.час | (8) |

Затраты труда на отладку программы находим по формуле:

|  |  |
| --- | --- |
| Тотл = (Q / (Sотл\*k))/20, | (9) |

где Sотл - количество операторов на отладку программы, приходящихся на 1 чел. час (4 - 5)

|  |  |
| --- | --- |
| Тотл = (2840 / (5\*0,8)) / 20 = 35,5 чел.час | (10) |

Затраты на подготовку документации по задаче определяются по формуле:

|  |  |
| --- | --- |
| Тд = Тдр + Тдо, | (11) |

где Тдр – затраты труда на подготовку материала рукописи;

Тдо – затраты труда на редактирование, печать и оформление документов.

|  |  |
| --- | --- |
| Тдр = (Q / (Sдр\*k))/10, | (12) |

где Sдр - количество операторов на подготовку материала рукописи, приходящихся на 1 чел.час (15 – 20).

|  |  |
| --- | --- |
| Тдр = (2840 / (17\*0,8)) / 10 = 20,88 чел.час | (13) |

Тдо определяется по формуле:

|  |  |
| --- | --- |
| Тдо = 0,75 \* Тдр, | (14) |
| Тдо = 0,75 \* 20,88 = 15,66 чел.час, | (14.1) |
| Тд = 20,88 + 15,66 = 36,54 чел.час, | (14.2) |
| Т = 35 + 62,125 + 161,36+ 169,04+ 35,5 + 36,54 = 499,565 чел. час. | (14.2) |

Полученное значение общей трудоемкости необходимо скорректировать с учетом уровня языка программирования:

|  |  |
| --- | --- |
| Ткор = Т\* kкор, = 0,8 | (15) |

где kкор- коэффициент учитывающий уровень языка программирования (0,8 – 1)

Часовая тарифная ставка инженера технолога определяется по формуле:

|  |  |
| --- | --- |
| Тст = МРОТ/Вф, | (16) |

где МРОТ – минимальный размер оплаты труда на 01.01.2025 г. (оклад сотрудника ), равный 22 440 руб./мес.;

Вф – фактически отработанное время (168 час. в месяц, т.е 21 смена \*8 часов).

|  |  |
| --- | --- |
| Тст = 22440 / 168 = 133,57 руб. | (17) |

Время работы персонального компьютера при создании программного продукта определяется по формуле:

|  |  |
| --- | --- |
| Фв = 1,15 \* (Тп + Тд + Тотл) \* kкор | (18) |
| Фв = 1,15 \* (169,04 + 36,54 + 35,5) \* 0,8 = 221,79 час. | (19.1) |
|  |  |

Таблица 16 - Затраты на разработку программного продукта

|  |  |  |
| --- | --- | --- |
| Наименование статьи | Формула для расчета | Сумма затрат |
| Основная ЗП с учетом коэффициента корректировки | ЗПо = (Тотл + Тп) \* kкор \*Тст + (Тдр +Та) kкор \*Тст + (Тп +Ти +То) kкор \*Тст = (35,5 + 169,04) ⋅ 0,8 ⋅ 133,57 + (36,54 + 161,36) ⋅ 0,8 ⋅ 133,57 + (169,04 + 62,125 + 35) ⋅ 0,8 ⋅ 133,57 = 71444.45 руб. | 71444,45 руб. |
| Дополнительная заработная плата | ЗПд = 0,2 \* ЗПо = 0,2 \* 71444,45 = 14288,89 руб. | 14288,89 руб. |
| Отчисления на социальные нужды | ЕСН = (ЗПо+ЗПд)\*0,3 = (71444.45 + 14288.89) \* 0,3 = 25720 руб. | 25720 руб. |
| Расходы на материалы и запчасти | Хостинг Play2go 200 рублей | 200 руб. |
| Затраты на ТО и ТР | 2 % от балансовой стоимости оборудования. Стоимость ПК 50000 руб\*0,02=1000 руб) | 1000 руб. |
| Затраты на потребляемые энергоресурсы | Зэ = 0,3 кВт\ч \* 6 руб. \* 121,9 ч. = 291,42 руб. | 219,42 руб. |
| Итого (Зи): | С=14288,89 + 25720 + 200 + 219,42 + 51836 = | 92264,31 руб. |

Стоимость услуг за разработку программного продукта должна составлять 92264,31 руб.

## 5.7 Управление рисками проекта

Управление рисками – это систематический процесс снижения неопределенности и управления вероятностью событий в проекте [17].

Цель управления рисками проекта – повышение вероятности возникновения и воздействия благоприятных событий и снижение вероятности возникновения и воздействия неблагоприятных для проекта событий [17].

Весь процесс управления рисками проекта разделяется на отдельные подпроцессы, которые возникают в каждом проекте, а некоторые из них реализуются на нескольких стадиях проекта [17].

Мероприятия реагирования на риски могут быть направлены на недопущение рисковых событий или минимизацию негативных последствий [17].

Мероприятия по недопущению наступления рисковых событий. Данные мероприятия после их разработки попадают в план проекта [17]:

* мероприятия по снижению вероятности наступления,
* мероприятия по снижению негативного влияния на риски.

Мероприятия по минимизации негативных последствий. Данные мероприятия попадают в план проекта в случае возникновения риска проекта [17].

Наиболее распространены 4 стратегии работы с рисками [17]:

* стратегия уклонения. Стратегия уклонение подразумевает выбор технологии и продуктов проектов, реализовав которые проект не столкнемся с риском. Выбор – это стратеги также подразумевает потеря уникальности продукта проекта и как результат снижение прибыли компании. Наиболее известным примером для строительных проектов является строительство типового жилья. Выбор для реализации проект строительства типового объекта позволяет снизить вероятность наступления рисковых события связанных с уникальностью. Легче отказаться от строительства сложных и уникальных конструктивов чем брать технологические риски и риски продукта на себя.
* стратегия передачи. Стратегия передачи подразумевает передачу риска другим заинтересованным сторонам проекта. К примеру, для выполнения высоко рисковой части работ приглашается подрядная организация, в контракте с которой, прописывается что она полностью берет на себя риски, связанные выполнением работ. Конечно, это повлияет на увеличение суммы контракта с подрядной организацией. Так же можно передать риск заказчику прописать в контракте с ним что компания не несет ответственность за те или иные риски. Это так же влияет на сумму контракта с заказчиком.
* стратегия снижения. Стратегия снижения подразумевает снижение вероятности и/или негативного влияние рискового события на проект. При роботе с вероятностью стараемся разработать мероприятия, которые снизят возможность наступления риска. Если снизить вероятность не удаётся разрабатываются мероприятия по снижению негативного влияния рискового события. Менеджеры проекта разрабатывают оба типа мероприятия выбирая в последствии наиболее результативные и эффективные мероприятия.
* стратегия принятия. Стратегия приятия подразумевает, что принимая факт наступления рискового события и разрабатываем мероприятия по снижению его негативного влияния. В рамках этой стратегии возможно, как резервирования времени и финансовых средств для преодоления последствий рискового события, так и разработка мероприятий, которые менеджер проекта реализует чтобы минимизировать негативный влияние рискового события.

Идея диаграммы причинно-следственных связей в форме рыбьей кости принадлежит крупнейшему эксперту по управлению качеством Каору Исикаве. Эта техника изначально применялась для анализа факторов, которые приводят к производственному браку. Но со временем её начали использовать и в других процессах, где что‑то сломалось или идёт не по плану. Метод «рыбной кости» остаётся популярным более полустолетия. И это неудивительно, учитывая, что он: даёт возможность глубже вникнуть в проблему, рассмотреть её с различных сторон и выявить порой неожиданные причинно‑следственные связи; помогает раскрыть творческий потенциал и нестандартно подойти к решению вопроса; универсален — диаграмму можно применять в разных жизненных и рабочих ситуациях, когда результаты не оправдали ожиданий [18].

Рассмотренная на рисунке 26 диаграмма Исикавы, полностью демонстрирует все факторы и риски при достижении поставленной цели. Подробное описание факторов позволит снизить влияние рисков в процессе выполнения проекта.



Рисунок 26 ­ Диаграмма Исикавы

# **ЗАКЛЮЧЕНИЕ**

Завершение дипломного проекта по разработке приложения для автоматизации учёта пациентов и его последующее внедрение, должно привести к повышению эффективности работы персоналом, а также повышению качества управления обязанностями персонала. Разработанное приложение успешно интегрировало все ключевые функции, такие как разграничение ролей, управление и просмотр таблиц. Внедрение приложения создало новые возможности для анализа и планирования мероприятий, что способствует дальнейшему развитию и модернизации процесса. По мере разработки проекта было изучено множество важных функций, например как взаимодействие базы данных PostgreSQL со средой разработки WPF.

Разработка базы данных в PostgeSQL:

* создана база данных, включающая 16 таблиц для хранения информации,
* внедрен механизм аудита для отслеживания входа в систему,
* осуществлено разграничение прав доступа к данным для различных ролей пользователей.

Создание интерфейса для ввода и обработки данных:

* разработан пользовательский интерфейс на базе WPF для ввода и обработки данных,
* обеспечена возможность добавления, редактирования и удаления для таблиц по необходимым правам доступа.

Также в процессе проектирования были разработаны ключевые UML-диаграммы, позволившие визуализировать архитектуру системы:

* диаграмма вариантов прецедентов (Use Case Diagram), отображающая все сценарии взаимодействия пользователей с системой;
* диаграмма последовательностей (Sequence Diagram), демонстрирующая временные аспекты взаимодействия компонентов;
* диаграмма активностей (Activity Diagram), отображающая динамические аспекты поведения системы.

Разработанное приложение может стать основой для внедрения в медицинские учреждения, где требуется автоматизация процессов учёта пациентов, а также может быть расширено дополнительными модулями для телемедицины, аналитики и интеграции с внешними системами. Приобретённые знания и опыт при выполнении данного дипломного проекта станут прочной базой для дальнейшей профессиональной деятельности в области разработки программного обеспечения.

# **СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ**

Интернет-ресурсы

* 1. Новостная система ВЦИОМ [Электронный ресурс]. – Режим доступа: https://wciom.ru/analytical-reviews/analiticheskii-obzor/gosudarstvennaja-medicina-v-pogone-za-kachestvom, свободный.– Загл. с экрана. Яз. рус.
  2. Новостная система РБК [Электронный ресурс]. – Режим доступа: https://plus.rbc.ru/news/5d94bdf97a8aa9726f930f10, свободный.– Загл. с экрана. Яз. рус.
  3. Медицинская система для частных медицинских центров и стоматологических клиник [Электронный ресурс]. – Режим доступа: https://archimed.pro/blog/uchet-patsientov-v-klinike-meditsinskom-tsentre-kak-organizovat-instrumenty-programmy-sovety/, свободный.– Загл. с экрана. Яз. рус.
  4. Софтлайн Решения [Электронный ресурс]. – Режим доступа: https://store.softline.ru/renovatsio/meditsinskaya-informatsionnaya-sistema-renovatio-clinica/, свободный.– Загл. с экрана. Яз. рус.
  5. МедАнгел: медицинская информационная система [Электронный ресурс]. – Режим доступа: https://medangel.angelsit.ru/, свободный.– Загл. с экрана. Яз. рус.
  6. Ресурс фирмы "1С", на котором собрана информация обо всех отраслевых и специализированных решениях [Электронный ресурс]. – Режим доступа: https://solutions.1c.ru/catalog/hospital/, свободный.– Загл. с экрана. Яз. рус.
  7. Med.me – платформа для онлайн-записи на приём к врачу [Электронный ресурс]. – Режим доступа: https://med.me/ru/, свободный.– Загл. с экрана. Яз. рус.
  8. CliniqLQ – cервис с облачным размещением для управления клиникой [Электронный ресурс]. – Режим доступа: https://www.cliniciq.ru/, свободный.– Загл. с экрана. Яз. рус.
  9. Платформа DTU ­ Что такое Project Management [Электронный ресурс]. – Режим доступа: https://www.dtu.kz/blog-post/chto-takoe-project-management/, свободный.– Загл. с экрана. Яз. рус.
  10. Платформа Яндекс Практикум ­ Управление проектами: что это такое и как работает проектный менеджмент в IT [Электронный ресурс]. – Режим доступа: https://practicum.yandex.ru/blog/chto-takoe-upravlenie-proektami/, свободный.– Загл. с экрана. Яз. рус.
  11. Платформа YAGLA ­ Цели SMART: критерии и примеры [Электронный ресурс]. – Режим доступа: https://yagla.ru/blog/marketing/celi-smart-kriterii-i-primery--2108u95038/, свободный.– Загл. с экрана. Яз. рус.
  12. Платформа ASANA ­ Что такое анализ заинтересованных лиц и почему он важен [Электронный ресурс]. – Режим доступа: https://asana.com/ru/resources/project-stakeholder, свободный.– Загл. с экрана. Яз. рус.
  13. Платформа Wrike ­ Что такое устав проекта в управлении проектами [Электронный ресурс]. – Режим доступа: https://www.wrike.com/ru/project-management-guide/faq/ustav-proekta/, свободный.– Загл. с экрана. Яз. рус.
  14. Платформа Atlassian ­ Как создавать простые и надежные планы проектов [Электронный ресурс]. – Режим доступа: https://www.atlassian.com/ru/work-management/project-management/project-planning/project-plan, свободный.– Загл. с экрана. Яз. рус.
  15. Платформа moo.team ­ WBS, или структурная декомпозиция, что это [Электронный ресурс]. – Режим доступа: https://moo.team/blog/post/wbs-ili-strukturnaya-dekompozicziya/, свободный.– Загл. с экрана. Яз. рус.
  16. Платформа Ganttpro ­ Все о Диаграмме Ганта: Как Успешно и Вовремя Завершать Проекты [Электронный ресурс]. – Режим доступа: https://blog.ganttpro.com/ru/diagramma-gantta-gantt-chart/, свободный.– Загл. с экрана. Яз. рус.
  17. Платформа E-xecutive ­ Риски в управлении проектами [Электронный ресурс]. – Режим доступа: https://www.e-xecutive.ru/wiki/index.php/Риски\_в \_управлении\_проектами, свободный.– Загл. с экрана. Яз. рус.
  18. Платформа pmclub ­ Диаграмма Исикавы: зачем применять и как построить [Электронный ресурс]. – Режим доступа: https://pmclub.pro/articles/diagramma-isikavy-zachem-primenyat-i-kak-postroit/, свободный.– Загл. с экрана. Яз. рус.

Дополнительные источники

19. **Дадаян, Э. Г.** Основы языка программирования 1С 8.3 [Электронный ресурс] : учебное пособие / Э. Г. Дадян. — Москва : Вузовский учебник: ИНФРА-М, 2018. — 132 с. - Режим доступа: https://new.znanium.com/read?id=333502 – Загл. с экрана.

20. **Соколова, В. В.** Вычислительная техника и информационные технологии. Разработка мобильных приложений [Электронный ресурс] : учебное пособие для прикладного бакалавриата / В. В. Соколова. — Москва : Издательство Юрайт, 2019. — 175 с. — (Университеты России). — ISBN 978-5-9916-6525-4. — Режим доступа: https://urait.ru/bcode/433981 – Загл. с экрана.

21. **Хорев, П. Б.** Объектно-ориентированное программирование с примерами на С# [Электронный ресурс] : учеб. пособие / П. Б. Хорев. — Москва : ФОРУМ : ИНФРА-М, 2019. — 200 с. — (Высшее образование: Бакалавриат). - Режим доступа: https://new.znanium.com/read?id=339308 – Загл. с экрана.

22. **Черпаков, И. В.** Основы программирования [Электронный ресурс] : учебник и практикум для среднего профессионального образования / И. В. Черпаков. — Москва : Издательство Юрайт, 2019. — 219 с. — (Профессиональное образование). — ISBN 978-5-9916-9984-6. — Режим доступа: https://urait.ru/bcode/436557 – Загл. с экрана.

23. **Федорова, Г. Н.** Разработка, внедрение и адаптация программного обеспечения отраслевой направленности [Электронный ресурс] : учебное пособие / Г. Н. Федорова. — М. :КУРС : ИНФРА-М, 2019. — 336 с. (Среднее Профессиональное Образование). - Режим доступа: https://new.znanium.com/read?id=330691 – Загл. с экрана.

24. **Хорев, П. Б.** Объектно-ориентированное программирование с примерами на С# [Электронный ресурс] : учебное пособие / П. Б. Хорев. — Москва : ФОРУМ : ИНФРА-М, 2019. — 200 с. — Режим доступа: https://new.znanium.com/read?id=339308 – Загл. с экрана.

25. **Баранова, Е. К.** Основы информационной безопасности [Электронный ресурс] : учебник / Е. К. Баранова, А. В. Бабаш. - Москва : РИОР: ИНФРА-М, 2019. - 202 с. - (Среднее профессиональное образование). - Режитм доступа: https://new.znanium.com/read?id=339532 – Загл. с экрана.

26. **Казарин, О. В.** Надежность и безопасность программного обеспечения [Электронный ресурс] : учебное пособие для бакалавриата и магистратуры / О. В. Казарин, И. Б. Шубинский. — Москва : Издательство Юрайт, 2019. — 342 с. — (Бакалавр и магистр. Модуль). — ISBN 978-5-534-05142-1. — Режим доступа: https://www.urait.ru/bcode/441287 – Загл. с экрана.

27. **Гагарина, Л. Г.** Разработка и эксплуатация автоматизированных информационных систем [Электронный ресурс] : учеб. пособие / Л.Г. Гагарина. — Москва : ИД «ФОРУМ» : ИНФРА-М, 2019. — 384 с. — (Среднее профессиональное образование). - Режим доступа: https://new.znanium.com/read?id=333679 – Загл. с экрана.

28. **Маркин, А. В.** Программирование на SQL в 2 ч. Часть 1 [Электронный ресурс] : учебник и практикум для вузов / А. В. Маркин. — 2-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2019. — 403 с. — (Высшее образование). — ISBN 978-5-534-12256-5. — Режим доступа: https://urait.ru/bcode/447115 – Загл. с экрана.

29. **Маркин, А. В.** Программирование на SQL в 2 ч. Часть 2 [Электронный ресурс] : учебник и практикум для вузов / А. В. Маркин. — 2-е изд., испр. и доп. — Москва : Издательство Юрайт, 2019. — 340 с. — (Высшее образование). — ISBN 978-5-534-12258-9. — Режим доступа: https://urait.ru/bcode/448191 – Загл. с экрана.

30. **Мартишин, С. А.** Базы данных. Практическое применение СУБД SQL и NoSQL-типа для проектирования информационных систем [Электронный ресурс] : учеб. пособие / С. А. Мартишин, В. Л. Симонов, М. В. Храпченко. — Москва : ИД «ФОРУМ» : ИНФРА-М, 2019. — 368 с. — (Высшее образование: Бакалавриат). - Режим доступа: https://new.znanium.com/read?id=333330 – Загл. с экрана.

.

# **ПРИЛОЖЕНИЕ А**

**(обязательное)**

**Листинги вёрстки приложения**

Листинг 1 – XAML код окна «Врач - Приём пациента»

<Window x:Class="DiplomProject.SpecialistWindows.AcceptancePatient.AcceptancePatientAddEditWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:DiplomProject.SpecialistWindows.AcceptancePatient"

mc:Ignorable="d"

Title="Врач - Приём пациента" Height="700" Width="1250" MinHeight="700" MinWidth="1250" WindowStartupLocation="CenterScreen">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="175"/>

<ColumnDefinition/>

<ColumnDefinition Width="175"/>

<ColumnDefinition/>

<ColumnDefinition Width="175"/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

</Grid.RowDefinitions>

<Button Content="Назад" Width="175" Height="35" HorizontalAlignment="Right" VerticalAlignment="Top" Margin="0,10,20,0" FontSize="22" Grid.Column="1" Grid.ColumnSpan="5" Click="Button\_Click"/>

<TextBlock Text="Приём пациента" Grid.Row="0" Grid.ColumnSpan="6" Grid.RowSpan="2" HorizontalAlignment="Center" VerticalAlignment="Center" Margin="0,0,0,20" FontSize="36"/>

<TextBlock Text="Специалист:" FontSize="22" Grid.Row="1" HorizontalAlignment="Right" VerticalAlignment="Center" Margin="10"/>

<TextBox Name="employee\_txb" FontSize="22" Focusable="False" Grid.Row="1" Grid.Column="1" Margin="10,10"/>

<TextBlock Text="Специальность" FontSize="22" Grid.Row="1" Grid.RowSpan="3" HorizontalAlignment="Right" VerticalAlignment="Center" Margin="10" TextWrapping="Wrap"/>

<TextBox Name="specialization\_txb" FontSize="22" Grid.Row="2" Grid.Column="1" Margin="10,10" Focusable="False"/>

<TextBlock Text="Пациент:" FontSize="22" Grid.Row="3" HorizontalAlignment="Right" VerticalAlignment="Center" Margin="10"/>

<TextBox Name="patient\_txb" Focusable="False" FontSize="22" Grid.Row="3" Grid.Column="1" Margin="10,10"/>

<TextBlock Text="Кабинет:" FontSize="22" Grid.Row="4" HorizontalAlignment="Right" VerticalAlignment="Center" Margin="10"/>

<ComboBox Name="room\_cmb" FontSize="22" Grid.Row="4" Grid.Column="1" Margin="10,10"/>

<TextBlock Text="Дата:" FontSize="22" Grid.Row="5" Grid.Column="0" HorizontalAlignment="Right" VerticalAlignment="Center" Margin="10"/>

<DatePicker Name="date\_dp" FontSize="22" Grid.Row="5" Grid.Column="1" Margin="10,10"/>

<TextBlock Text="Давление:" FontSize="22" Grid.Row="5" Grid.Column="2" HorizontalAlignment="Right" VerticalAlignment="Center" Margin="10"/>

<TextBox Name="pressure\_txb" FontSize="22" Grid.Row="5" Grid.Column="3" Margin="10,10"/>

<TextBlock Text="Пульс:" FontSize="22" Grid.Row="6" Grid.Column="2" HorizontalAlignment="Right" VerticalAlignment="Center" Margin="10" />

<TextBox Name="pulse\_txb" FontSize="22" Grid.Row="6" Grid.Column="3" Margin="10,10"/>

<TextBlock FontSize="22" Grid.Row="2" Grid.Column="2" Grid.RowSpan="3" HorizontalAlignment="Right" VerticalAlignment="Center" Margin="10" TextAlignment="Right" TextWrapping="Wrap">Общее состояние:</TextBlock>

<ComboBox Name="general\_status\_txb" FontSize="22" Grid.Row="3" Grid.Column="3" Margin="10,10"></ComboBox>

<TextBlock Text="Кожные покровы:" FontSize="22" Grid.Row="2" Grid.Column="4" Grid.RowSpan="3" HorizontalAlignment="Right" VerticalAlignment="Center" Margin="10" TextAlignment="Right" TextWrapping="Wrap"></TextBlock>

<ComboBox Name="skin\_status\_txb" FontSize="22" Grid.Row="3" Grid.Column="5" Margin="10,10"></ComboBox>

<TextBlock Text="Видимые слизистые:" FontSize="22" Grid.Row="3" Grid.Column="4" Grid.RowSpan="3" HorizontalAlignment="Right" VerticalAlignment="Center" Margin="10" TextAlignment="Right" TextWrapping="Wrap"></TextBlock>

<ComboBox Name="mucous\_status\_txb" FontSize="22" Grid.Row="4" Grid.Column="5" Margin="10,10"></ComboBox>

<TextBlock Text="Лёгкие:" FontSize="22" Grid.Row="7" Grid.Column="2" Grid.RowSpan="3" HorizontalAlignment="Right" VerticalAlignment="Center" Margin="10" TextAlignment="Right" TextWrapping="Wrap"></TextBlock>

<ComboBox Name="lungs\_status\_txb" FontSize="22" Grid.Row="8" Grid.Column="3" Margin="10,10"></ComboBox>

<TextBlock Text="Сердце:" FontSize="22" Grid.Row="6" Grid.Column="2" Grid.RowSpan="3" HorizontalAlignment="Right" VerticalAlignment="Center" Margin="10" TextAlignment="Right" TextWrapping="Wrap"></TextBlock>

<ComboBox Name="heart\_status\_txb" FontSize="22" Grid.Row="7" Grid.Column="3" Margin="10,10"></ComboBox>

<TextBlock Text="Язык:" FontSize="22" Grid.Row="4" Grid.Column="4" Grid.RowSpan="3" HorizontalAlignment="Right" VerticalAlignment="Center" Margin="10" TextAlignment="Right" TextWrapping="Wrap"></TextBlock>

<ComboBox Name="tongue\_status\_txb" FontSize="22" Grid.Row="5" Grid.Column="5" Margin="10,10"></ComboBox>

<TextBlock Text="Живот:" FontSize="22" Grid.Row="5" Grid.Column="4" Grid.RowSpan="3" HorizontalAlignment="Right" VerticalAlignment="Center" Margin="10" TextAlignment="Right" TextWrapping="Wrap"></TextBlock>

<ComboBox Name="stomach\_status\_txb" FontSize="22" Grid.Row="6" Grid.Column="5" Margin="10,10"></ComboBox>

<TextBlock Text="Отёки:" FontSize="22" Grid.Row="6" Grid.Column="4" Grid.RowSpan="3" HorizontalAlignment="Right" VerticalAlignment="Center" Margin="10" TextAlignment="Right" TextWrapping="Wrap"></TextBlock>

<ComboBox Name="urination\_status\_txb" FontSize="22" Grid.Row="7" Grid.Column="5" Margin="10,10"></ComboBox>

<TextBlock Text="Нёбные миндалины:" FontSize="22" Grid.Row="3" Grid.Column="2" Grid.RowSpan="3" HorizontalAlignment="Right" VerticalAlignment="Center" Margin="10" TextAlignment="Right" TextWrapping="Wrap"></TextBlock>

<ComboBox Name="palatine\_status\_txb" FontSize="22" Grid.Row="4" Grid.Column="3" Margin="10,10"></ComboBox>

<TextBlock Text="Жалобы" FontSize="22" Grid.Row="6" Grid.RowSpan="2" HorizontalAlignment="Right" VerticalAlignment="Center" Margin="10"/>

<TextBox Name="report\_txb" FontSize="22" Grid.Row="6" Grid.Column="1" Grid.RowSpan="2" Grid.ColumnSpan="1" Margin="10,10" TextWrapping="Wrap"/>

<TextBlock Text="Результат осмотра:" FontSize="22" Grid.Row="8" Grid.RowSpan="2" HorizontalAlignment="Right" VerticalAlignment="Center" Margin="10" TextWrapping="Wrap" TextAlignment="Right"/>

<TextBox Name="comment\_txb" FontSize="22" Grid.Row="8" Grid.Column="1" Grid.RowSpan="2" Grid.ColumnSpan="1" Margin="10,10" TextWrapping="Wrap"/>

<Button Grid.Column="2" Grid.Row="1" Grid.RowSpan="2" Grid.ColumnSpan="2" Margin="30,30" Width="175" Click="Button\_Click\_2"><TextBlock FontSize="16" TextWrapping="Wrap" TextAlignment="Center">Выписать направление на консультацию</TextBlock></Button>

<Button Grid.Column="2" Grid.Row="1" Grid.RowSpan="2" Grid.ColumnSpan="4" Margin="30,30" Width="175" Click="Button\_Click\_3"><TextBlock FontSize="16" TextWrapping="Wrap" TextAlignment="Center">Выписать направление на обследование</TextBlock></Button>

<Button Grid.Column="4" Grid.Row="1" Grid.RowSpan="2" Grid.ColumnSpan="2" Margin="30,30" Width="175"><TextBlock FontSize="16" TextWrapping="Wrap" TextAlignment="Center">Выписать направление на исследование</TextBlock></Button>

<Button Content="Готово" Grid.Row="10" Grid.ColumnSpan="6" FontSize="22" Margin="300,10" Click="Button\_Click\_1"/>

</Grid>

</Window>

Листинг 2 – xaml код окна «График работы сотрудников»

<Window x:Class="DiplomProject.SpecialistWindows.TimetableWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:DiplomProject.SpecialistWindows"

mc:Ignorable="d"

Title="Врач - график работы сотрудников" Height="450" Width="800" WindowStartupLocation="CenterOwner">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="75"/>

<RowDefinition Height="75"/>

<RowDefinition/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Button Content="Назад" Width="175" Height="50" HorizontalAlignment="Right" VerticalAlignment="Top" Margin="0,10,20,0" FontSize="22" Grid.Column="1" Click="Button\_Click"/>

<TextBlock Text="Специалист" Grid.Row="1" FontSize="22" HorizontalAlignment="Right" Margin="100,20,10,20"/>

<ComboBox Name="employees\_cmb" Grid.Row="1" Grid.Column="1" Margin="10,20,175,20" SelectionChanged="employees\_cmb\_SelectionChanged"/>

<ListBox Name="timetableListBox" Grid.Row="2" Grid.ColumnSpan="2" Margin="10">

<ListBox.ItemTemplate>

<DataTemplate>

<StackPanel Margin="10">

<TextBlock>

<Run Text="День недели: " />

<Run Text="{Binding Day}" />

</TextBlock>

<TextBlock>

<Run Text="Начало рабочего дня: " />

<Run Text="{Binding StartTime}" />

</TextBlock>

<TextBlock>

<Run Text="Окончание рабочего дня: " />

<Run Text="{Binding EndTime}" />

</TextBlock>

<TextBlock>

<Run Text="Начало перерыва: " />

<Run Text="{Binding StartTimePause}" />

</TextBlock>

<TextBlock>

<Run Text="Начало перерыва: " />

<Run Text="{Binding EndTimePause}" />

</TextBlock>

<WrapPanel>

<TextBlock Text="Этот день выходной "/>

<CheckBox Content="Holiday" IsChecked="{Binding Holiday}" IsEnabled="False"/>

</WrapPanel>

</StackPanel>

</DataTemplate>

</ListBox.ItemTemplate>

</ListBox>

</Grid>

</Window>

Листинг 3 – XAML код окна «Врач - Приём пациента»

<Window x:Class="DiplomProject.SpecialistWindows.AcceptancePatientWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:DiplomProject.SpecialistWindows"

mc:Ignorable="d"

Title="Врач - Приём пациента" Height="450" Width="800" WindowStartupLocation="CenterOwner">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="75"/>

<RowDefinition Height="75"/>

<RowDefinition/>

<RowDefinition Height="75"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Button Content="Назад" Width="175" Height="50" HorizontalAlignment="Right" VerticalAlignment="Top" Margin="0,10,20,0" FontSize="22" Grid.Column="3" Click="Button\_Click"/>

<TextBlock Text="Пациент" Grid.Column="1" Grid.Row="1" FontSize="22" HorizontalAlignment="Right" VerticalAlignment="Center" Margin="100,20,10,20"/>

<ComboBox Name="patients\_cmb" Grid.Row="1" Grid.Column="2" Margin="10,20" Width="175" Height="30" SelectedIndex="1" VerticalAlignment="Center" SelectionChanged="patients\_cmb\_SelectionChanged"/>

<Grid Grid.Column="3" Grid.Row="1">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="2\*"/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<TextBlock Text="Вывести пациентов только с имеющимися записями:" TextAlignment="Right" TextWrapping="Wrap" VerticalAlignment="Center"/>

<CheckBox Name="ch" Grid.Column="1" VerticalAlignment="Center" Margin="20,0,0,0" IsChecked="True" Click="ch\_Click"/>

</Grid>

<ListBox Name="firstCheckListBox" Grid.Row="2" Grid.ColumnSpan="4" Margin="10">

<ListBox.ItemTemplate>

<DataTemplate>

<Border BorderBrush="Gray" BorderThickness="1" Margin="5" Padding="5" CornerRadius="5">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="100" />

<ColumnDefinition Width="1\*" />

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition />

<RowDefinition />

<RowDefinition />

<RowDefinition />

<RowDefinition />

<RowDefinition />

<RowDefinition />

<RowDefinition />

</Grid.RowDefinitions>

<TextBlock Grid.Row="0" Grid.Column="0" Text="Сотрудник:" FontWeight="Bold"/>

<TextBlock Grid.Row="0" Grid.Column="1" Text="{Binding employeeFullName}" />

<TextBlock Grid.Row="1" Grid.Column="0" Text="Пациент:" FontWeight="Bold"/>

<TextBlock Grid.Row="1" Grid.Column="1" Text="{Binding patientFullNmae}" />

<TextBlock Grid.Row="2" Grid.Column="0" Text="Дата:" FontWeight="Bold"/>

<TextBlock Grid.Row="2" Grid.Column="1" Text="{Binding date}" />

<TextBlock Grid.Row="3" Grid.Column="0" Text="Кабинет:" FontWeight="Bold"/>

<TextBlock Grid.Row="3" Grid.Column="1" Text="{Binding roomId}" />

<TextBlock Grid.Row="4" Grid.Column="0" Text="Давление:" FontWeight="Bold"/>

<TextBlock Grid.Row="4" Grid.Column="1" Text="{Binding pressure}" />

<TextBlock Grid.Row="5" Grid.Column="0" Text="Пульс:" FontWeight="Bold"/>

<TextBlock Grid.Row="5" Grid.Column="1" Text="{Binding pulse}" />

<TextBlock Grid.Row="6" Grid.Column="0" Text="Комментарий:" FontWeight="Bold"/>

<TextBlock Grid.Row="6" Grid.Column="1" Text="{Binding comments}" TextWrapping="Wrap"/>

<TextBlock Grid.Row="7" Grid.Column="0" Text="Выполнено: " FontWeight="Bold"/>

<CheckBox Grid.Row="7" Grid.Column="1" IsChecked="{Binding complated}" />

</Grid>

</Border>

</DataTemplate>

</ListBox.ItemTemplate>

</ListBox>

<Button Content="Добавить" Width="175" Height="50" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="0,10,20,0" FontSize="22" Grid.Column="0" Grid.ColumnSpan="2" Grid.Row="3" Click="Button\_Click\_1"/>

<Button Content="Изменить" Width="175" Height="50" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="20,10,0,0" FontSize="22" Grid.Column="2" Grid.ColumnSpan="2" Grid.Row="3" Click="Button\_Click\_2"/>

</Grid>

</Window>

# **ПРИЛОЖЕНИЕ Б**

**(обязательное)**

**Листинги логики приложения**

Листинг 1 – Логика окна «Врач - Приём пациента».

public partial class AcceptancePatientAddEditWindow : Window

{

EmployeeClass employeeClass;

bool add = true;

NpgsqlConnection con = new NpgsqlConnection();

public ObservableCollection<FirstCheckClass> firstCheckItem { get; set; }

public FirstCheckClass selectedItem;

public AcceptancePatientAddEditWindow(EmployeeClass e, bool ch, FirstCheckClass select)

{

InitializeComponent();

ConnectionClass x = new ConnectionClass();

employeeClass = e;

add = ch;

selectedItem = select;

con = x.con;

refresh\_cmb();

}

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

{

AcceptancePatientWindow window = new AcceptancePatientWindow(employeeClass);

window.Show();

this.Close();

}

private void Button\_Click\_1(object sender, RoutedEventArgs e)

{

if (string.IsNullOrWhiteSpace(employee\_txb.Text) || string.IsNullOrWhiteSpace(patient\_txb.Text) || string.IsNullOrWhiteSpace(room\_cmb.Text) || date\_dp.SelectedDate == null || string.IsNullOrWhiteSpace(pressure\_txb.Text) || string.IsNullOrWhiteSpace(pulse\_txb.Text) || string.IsNullOrWhiteSpace(comment\_txb.Text))

{

MessageBox.Show("Все поля должны быть заполнены!", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);

}

else

{

try

{

con.Open();

if (add) {

using (NpgsqlCommand cmd = new NpgsqlCommand("INSERT INTO first\_check (employee\_id, patient\_id, room\_id, date, pressure, pulse, comments, report, general\_status,skin\_status,mucous\_status,lungs\_status,heart\_status, tongue\_status, stomach\_status, urination\_status,palatine\_status) VALUES ((SELECT id FROM employees WHERE first\_name = @employee\_first AND last\_name = @employee\_last AND patronymic = @employee\_patronymic), (SELECT id FROM patients WHERE first\_name = @patient\_first AND last\_name = @patient\_last AND patronymic = @patient\_patronymic),@room\_id::int4, @date::timestamp, @pressure, @pulse, @comments, @report, @general\_status,@skin\_status,@mucous\_status,@lungs\_status,@heart\_status,@tongue\_status,@stomach\_status,@urination\_status,@palatine\_status)", con))

{

string[] employee = employee\_txb.Text.Split(' ');

string[] patient = patient\_txb.Text.Split(' ');

cmd.Parameters.AddWithValue("employee\_first", employee[1]);

cmd.Parameters.AddWithValue("employee\_last", employee[0]);

cmd.Parameters.AddWithValue("employee\_patronymic", employee[2]);

cmd.Parameters.AddWithValue("patient\_first", patient[1]);

cmd.Parameters.AddWithValue("patient\_last", patient[0]);

cmd.Parameters.AddWithValue("patient\_patronymic", patient[2]);

cmd.Parameters.AddWithValue("room\_id", room\_cmb.SelectedValue);

cmd.Parameters.AddWithValue("date", date\_dp.SelectedDate.Value);

cmd.Parameters.AddWithValue("pressure", pressure\_txb.Text);

cmd.Parameters.AddWithValue("pulse", pulse\_txb.Text);

cmd.Parameters.AddWithValue("comments", comment\_txb.Text);

cmd.Parameters.AddWithValue("report", report\_txb.Text);

cmd.Parameters.AddWithValue("general\_status", general\_status\_txb.Text);

cmd.Parameters.AddWithValue("skin\_status", skin\_status\_txb.Text);

cmd.Parameters.AddWithValue("mucous\_status", mucous\_status\_txb.Text);

cmd.Parameters.AddWithValue("lungs\_status", lungs\_status\_txb.Text);

cmd.Parameters.AddWithValue("heart\_status", heart\_status\_txb.Text);

cmd.Parameters.AddWithValue("tongue\_status", tongue\_status\_txb.Text);

cmd.Parameters.AddWithValue("stomach\_status", stomach\_status\_txb.Text);

cmd.Parameters.AddWithValue("urination\_status", urination\_status\_txb.Text);

cmd.Parameters.AddWithValue("palatine\_status", palatine\_status\_txb.Text);

cmd.ExecuteNonQuery();

}

MessageBox.Show("Запись успешно добавлена");

}

else

{

using (NpgsqlCommand cmd = new NpgsqlCommand("UPDATE first\_check SET employee\_id = (SELECT id FROM employees WHERE first\_name = @employee\_first AND last\_name = @employee\_last AND patronymic = @employee\_patronymic), patient\_id = (SELECT id FROM patients WHERE first\_name = @patient\_first AND last\_name = @patient\_last AND patronymic = @patient\_patronymic),room\_id = @room\_id, date = @date, pressure = @pressure, pulse = @pulse, comments = @comments, report = @report, general\_status = @general\_status, skin\_status = @skin\_status, mucous\_status = @mucous\_status, lungs\_status = @lungs\_status, heart\_status = @heart\_status, tongue\_status = @tongue\_status, stomach\_status = @stomach\_status, urination\_status = @urination\_status, palatine\_status = @palatine\_status WHERE id = @id", con))

{

string[] employee = employee\_txb.Text.Split(' ');

string[] patient = patient\_txb.Text.Split(' ');

cmd.Parameters.AddWithValue("employee\_first", employee[1]);

cmd.Parameters.AddWithValue("employee\_last", employee[0]);

cmd.Parameters.AddWithValue("employee\_patronymic", employee[2]);

cmd.Parameters.AddWithValue("patient\_first", patient[1]);

cmd.Parameters.AddWithValue("patient\_last", patient[0]);

cmd.Parameters.AddWithValue("patient\_patronymic", patient[2]);

cmd.Parameters.AddWithValue("room\_id", room\_cmb.SelectedValue);

cmd.Parameters.AddWithValue("date", date\_dp.SelectedDate.Value);

cmd.Parameters.AddWithValue("pressure", pressure\_txb.Text);

cmd.Parameters.AddWithValue("pulse", pulse\_txb.Text);

cmd.Parameters.AddWithValue("comments", comment\_txb.Text);

cmd.Parameters.AddWithValue("id", selectedItem.id);

cmd.Parameters.AddWithValue("report", report\_txb.Text);

cmd.Parameters.AddWithValue("general\_status", general\_status\_txb.Text);

cmd.Parameters.AddWithValue("skin\_status", skin\_status\_txb.Text);

cmd.Parameters.AddWithValue("mucous\_status", mucous\_status\_txb.Text);

cmd.Parameters.AddWithValue("lungs\_status", lungs\_status\_txb.Text);

cmd.Parameters.AddWithValue("heart\_status", heart\_status\_txb.Text);

cmd.Parameters.AddWithValue("tongue\_status", tongue\_status\_txb.Text);

cmd.Parameters.AddWithValue("stomach\_status", stomach\_status\_txb.Text);

cmd.Parameters.AddWithValue("urination\_status", urination\_status\_txb.Text);

cmd.Parameters.AddWithValue("palatine\_status", palatine\_status\_txb.Text);

cmd.ExecuteNonQuery();

}

MessageBox.Show("Запись успешно изменена");

}

AcceptancePatientWindow window = new AcceptancePatientWindow(employeeClass);

window.Show();

this.Close();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

con.Close();

}

}

}

public void refresh\_cmb()

{

try

{

con.Open();

using (NpgsqlCommand cmd = new NpgsqlCommand("select concat(employees.last\_name, ' ',employees.first\_name,' ', employees.patronymic) from employees", con))

{

using (NpgsqlDataReader reader = cmd.ExecuteReader())

{

while (reader.Read())

{

employee\_txb.Text = reader.GetString(0);

}

}

}

using (NpgsqlCommand cmd = new NpgsqlCommand("select concat(patients.last\_name, ' ',patients.first\_name,' ', patients.patronymic) from patients", con))

{

using (NpgsqlDataReader reader = cmd.ExecuteReader())

{

while (reader.Read())

{

patient\_txb.Text = reader.GetString(0);

}

}

}

using (NpgsqlCommand cmd = new NpgsqlCommand("select id from rooms", con))

{

using (NpgsqlDataReader reader = cmd.ExecuteReader())

{

room\_cmb.Items.Clear();

while (reader.Read())

{

room\_cmb.Items.Add(reader.GetInt32(0));

}

}

}

urination\_status\_txb.SelectedValue = "Пастозность";

employee\_txb.Text = employeeClass.last\_name + " " + employeeClass.first\_name + " " + employeeClass.patronymic;

date\_dp.SelectedDate = DateTime.Now;

if (!add)

{

employee\_txb.Text = selectedItem.employeeFullName;

specialization\_txb.Text = selectedItem.specialization;

patient\_txb.Text = selectedItem.patientFullNmae;

date\_dp.SelectedDate = selectedItem.date;

room\_cmb.SelectedValue = selectedItem.roomId;

pressure\_txb.Text = selectedItem.pressure;

pulse\_txb.Text = selectedItem.pulse;

comment\_txb.Text = selectedItem.comments;

report\_txb.Text = selectedItem.report;

general\_status\_txb.SelectedValue = selectedItem.general\_status;

skin\_status\_txb.SelectedValue= selectedItem.skin\_status;

mucous\_status\_txb.SelectedValue = selectedItem.mucous\_status;

lungs\_status\_txb.SelectedValue = selectedItem.lungs\_status;

heart\_status\_txb.SelectedValue = selectedItem.heart\_status;

tongue\_status\_txb.SelectedValue = selectedItem.tongue\_status;

stomach\_status\_txb.SelectedValue = selectedItem.stomach\_status;

urination\_status\_txb.SelectedValue = selectedItem.urination\_status;

palatine\_status\_txb.SelectedValue = selectedItem.palatine\_status;

}

using (NpgsqlCommand cmd = new NpgsqlCommand("select specializations.name from employees join specializations on specializations.id = employees.specialization where concat(employees.last\_name, ' ',employees.first\_name,' ', employees.patronymic) = @employee", con))

{

cmd.Parameters.AddWithValue("employee", employee\_txb.Text);

using (NpgsqlDataReader reader = cmd.ExecuteReader())

{

while (reader.Read())

{

specialization\_txb.Text = reader.GetString(0);

}

}

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

con.Close();

}

List<string> general\_status = new List<string>() { "Удовлетворительное", "Средней тяжести", "Тяжёлое" };

general\_status\_txb.ItemsSource = general\_status;

List<string> palatine\_status = new List<string>() { "Нет", "Пастозностьи", "Отёк нижних конечностей", "Отёк лица" };

palatine\_status\_txb.ItemsSource = palatine\_status;

List<string> heart\_status = new List<string>() { "Тоны ясные", "Ритмичные", "Приглушены", "Аритмичны" };

heart\_status\_txb.ItemsSource = heart\_status;

List<string> lungs\_status = new List<string>() { "Дыхание везукулярное", "Жёсткое", "Ослабленное везукулярное", "Сухие хрипы", "Влажные хрипы" };

lungs\_status\_txb.ItemsSource = lungs\_status;

List<string> skin\_status = new List<string>() { "Обычной окраски", "Чистые", "Бледные", "Желтушные", "Цианотичные" };

skin\_status\_txb.ItemsSource = skin\_status;

List<string> mucous\_status = new List<string>() { "Розовые", "Бледные", "Гипермированные", "Иктеричные", "Чистые" };

mucous\_status\_txb.ItemsSource = mucous\_status;

List<string> tongue\_status = new List<string>() { "Чистый", "Влажный", "Сухой", "Обложен налётом" };

tongue\_status\_txb.ItemsSource = tongue\_status;

List<string> stomach\_status = new List<string>() { "Мягкий", "Безболезненный", "Участвует в акте дыхания", "Обложен налётом", "Не вздут" };

stomach\_status\_txb.ItemsSource = stomach\_status;

List<string> urination\_status = new List<string>() { "Нет", "Пастозность", "Отёк нижних конечностей", "Отёк лица" };

urination\_status\_txb.ItemsSource= urination\_status;

}

private void Button\_Click\_2(object sender, RoutedEventArgs e)

{

ConsulationWindow window = new ConsulationWindow(employeeClass, add, selectedItem);

window.Show();

this.Close();

}

private void Button\_Click\_3(object sender, RoutedEventArgs e)

{

AnalysisWindow window = new AnalysisWindow(employeeClass, add, selectedItem);

window.Show();

this.Close();

}

}

Листинг 2 – Логика окна «График работы сотрудников»

public partial class TimetableWindow : Window

{

public ObservableCollection<TimetableClass> timetableItem { get; set; }

List<string> employees = new List<string>();

EmployeeClass employeeClass;

ConnectionClass con;

NpgsqlConnection conn;

public TimetableWindow(EmployeeClass e)

{

InitializeComponent();

con = new ConnectionClass();

conn = con.con;

employeeClass = e;

refresh\_combobox();

}

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

{

MenuWindow window = new MenuWindow(employeeClass);

window.Show();

this.Close();

}

public void refresh\_combobox()

{

try

{

employees\_cmb.Items.Clear();

conn.Open();

using (NpgsqlCommand cmd = new NpgsqlCommand("select distinct(concat(employees.first\_name, ' ', employees.last\_name, ' ', employees.patronymic)) from timetable left join employees on employees.id = timetable.employee\_id", conn))

{

using (NpgsqlDataReader reader = cmd.ExecuteReader())

{

while (reader.Read())

{

employees.Add(reader.GetString(0));

}

}

}

employees\_cmb.ItemsSource = employees;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

conn.Close();

}

}

private void employees\_cmb\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

try

{

string selectedEmployee = employees\_cmb.SelectedValue.ToString();

conn.Open();

using (NpgsqlCommand cmd = new NpgsqlCommand($"select concat(employees.first\_name, ' ', employees.last\_name, ' ', employees.patronymic) as full\_name, timetable.\* from timetable join employees on employees.id = timetable.employee\_id WHERE CONCAT\_WS(' ', employees.first\_name, employees.last\_name, employees.patronymic) = @full\_name", conn)) ///\*WHERE CONCAT\_WS(' ', employees.first\_name, employees.last\_name, employees.patronymic) = {timetableListBox.SelectedItem}\*/

{

cmd.Parameters.AddWithValue("full\_name", selectedEmployee);

using (NpgsqlDataReader reader = cmd.ExecuteReader())

{

timetableItem = new ObservableCollection<TimetableClass>();

timetableItem.Clear();

while (reader.Read())

{

timetableItem.Add(new TimetableClass

{

FullName = reader.IsDBNull(0) ? "Неизвестно" : reader.GetString(0),

Id = reader.IsDBNull(1) ? 0 : reader.GetInt32(1),

Day = reader.IsDBNull(2) ? "Не указан" : reader.GetString(2),

StartTime = reader.IsDBNull(4) ? TimeSpan.Zero : reader.GetTimeSpan(4),

EndTime = reader.IsDBNull(5) ? TimeSpan.Zero : reader.GetTimeSpan(5),

StartTimePause = reader.IsDBNull(6) ? TimeSpan.Zero : reader.GetTimeSpan(6),

EndTimePause = reader.IsDBNull(7) ? TimeSpan.Zero : reader.GetTimeSpan(7),

Holiday = !reader.IsDBNull(8) && reader.GetBoolean(8)

});

}

}

}

timetableListBox.ItemsSource = timetableItem;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

conn.Close();

}

}

}

Листинг 3 – Логика окна «Врач - Приём пациента»

public partial class AcceptancePatientWindow : Window

{

bool add = true;

public NpgsqlConnection con;

EmployeeClass employeeClass;

public ObservableCollection<FirstCheckClass> firstCheckItem { get; set; }

public List<string> patientItem = new List<string>();

public List<string> speciallizationItem = new List<string>();

public AcceptancePatientWindow(EmployeeClass e)

{

InitializeComponent();

ConnectionClass x = new ConnectionClass();

employeeClass = e;

con = x.con;

refresh\_cmb();

patients\_cmb\_SelectionChanged(null, null);

}

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

{

MenuWindow window = new MenuWindow(null);

window.Show();

this.Close();

}

private void Button\_Click\_1(object sender, RoutedEventArgs e)

{

add = true;

AcceptancePatientAddEditWindow window = new AcceptancePatientAddEditWindow(employeeClass, add, null);

window.Show();

this.Close();

}

private void Button\_Click\_2(object sender, RoutedEventArgs e)

{

try

{

if (firstCheckListBox.SelectedItem == null)

{

MessageBox.Show("Выберите запись!");

}

else

{

FirstCheckClass selectedItem = (FirstCheckClass)firstCheckListBox.SelectedItem;

add = false;

AcceptancePatientAddEditWindow window = new AcceptancePatientAddEditWindow(employeeClass, add, selectedItem);

window.Show();

this.Close();

}

}

catch

{

MessageBox.Show("Ошибка");

}

}

private void patients\_cmb\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

if (patients\_cmb.SelectedValue != null)

{

try

{

string patient = patients\_cmb.SelectedValue.ToString();

string employee = employeeClass.last\_name + " " + employeeClass.first\_name + " " + employeeClass.patronymic;

con.Close();

con.Open();

if (ch.IsChecked == true)

{

using (NpgsqlCommand cmd = new NpgsqlCommand("select concat(employees.last\_name, ' ',employees.first\_name,' ', employees.patronymic), concat(patients.last\_name, ' ',patients.first\_name,' ', patients.patronymic) , first\_check.\*, specializations.name from first\_check join employees on first\_check.employee\_id = employees.id join patients on first\_check.patient\_id = patients.id join specializations on employees.specialization = specializations.id where concat(patients.last\_name, ' ',patients.first\_name,' ', patients.patronymic) = @patient and concat(employees.last\_name, ' ',employees.first\_name,' ', employees.patronymic) = @employee", con))

{

cmd.Parameters.AddWithValue("patient", patient);

cmd.Parameters.AddWithValue("employee", employee);

using (NpgsqlDataReader reader = cmd.ExecuteReader())

{

firstCheckItem = new ObservableCollection<FirstCheckClass>();

firstCheckItem.Clear();

while (reader.Read())

{

firstCheckItem.Add(new FirstCheckClass

{

employeeFullName = reader.GetString(0),

patientFullNmae = reader.GetString(1),

id = reader.GetInt32(2),

employeeId = reader.GetInt32(3),

patientId = reader.GetInt32(4),

roomId = reader.GetInt32(5),

date = reader.GetDateTime(6),

pressure = reader.GetString(7),

pulse = reader.GetString(8),

comments = reader.GetString(9),

report = reader.GetString(10),

general\_status = reader.GetString(11),

skin\_status = reader.GetString(12),

mucous\_status = reader.GetString(13),

lungs\_status = reader.GetString(14),

heart\_status = reader.GetString(15),

tongue\_status = reader.GetString(16),

stomach\_status = reader.GetString(17),

urination\_status = reader.GetString(18),

palatine\_status = reader.GetString(19),

complated = reader.GetBoolean(20),

specialization = reader.GetString(21)

});

}

}

}

}

else

{

using (NpgsqlCommand cmd = new NpgsqlCommand("select concat(employees.last\_name, ' ',employees.first\_name,' ', employees.patronymic), concat(patients.last\_name, ' ',patients.first\_name,' ', patients.patronymic) , first\_check.\*, specializations.name from first\_check join employees on first\_check.employee\_id = employees.id join patients on first\_check.patient\_id = patients.id join specializations on employees.specialization = specializations.id where concat(patients.last\_name, ' ',patients.first\_name,' ', patients.patronymic) = @patient", con))

{

cmd.Parameters.AddWithValue("patient", patient);

using (NpgsqlDataReader reader = cmd.ExecuteReader())

{

firstCheckItem = new ObservableCollection<FirstCheckClass>();

firstCheckItem.Clear();

while (reader.Read())

{

firstCheckItem.Add(new FirstCheckClass

{

employeeFullName = reader.GetString(0),

patientFullNmae = reader.GetString(1),

id = reader.GetInt32(2),

employeeId = reader.GetInt32(3),

patientId = reader.GetInt32(4),

roomId = reader.GetInt32(5),

date = reader.GetDateTime(6),

pressure = reader.GetString(7),

pulse = reader.GetString(8),

comments = reader.GetString(9),

report = reader.GetString(10),

general\_status = reader.GetString(11),

skin\_status = reader.GetString(12),

mucous\_status = reader.GetString(13),

lungs\_status = reader.GetString(14),

heart\_status = reader.GetString(15),

tongue\_status = reader.GetString(16),

stomach\_status = reader.GetString(17),

urination\_status = reader.GetString(18),

palatine\_status = reader.GetString(19),

complated = reader.GetBoolean(20),

specialization = reader.GetString(21)

});

}

}

}

}

firstCheckListBox.ItemsSource = firstCheckItem;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

con.Close();

}

}

}

public void refresh\_cmb()

{

patientItem.Clear();

string employee = employeeClass.last\_name + " " + employeeClass.first\_name + " " + employeeClass.patronymic;

try

{

con.Close();

con.Open();

if (ch.IsChecked == false) {

using (NpgsqlCommand cmd = new NpgsqlCommand("select concat(patients.last\_name, ' ',patients.first\_name,' ', patients.patronymic) from patients", con))

{

using (NpgsqlDataReader reader = cmd.ExecuteReader())

{

while (reader.Read())

{

patientItem.Add(reader.GetString(0));

}

}

}

}

else

{

using (NpgsqlCommand cmd = new NpgsqlCommand("select distinct(concat(patients.last\_name, ' ',patients.first\_name,' ', patients.patronymic)) from first\_check join employees on first\_check.employee\_id = employees.id join patients on first\_check.patient\_id = patients.id join specializations on employees.specialization = specializations.id where concat(employees.last\_name, ' ',employees.first\_name,' ', employees.patronymic) = @employee", con))

{

cmd.Parameters.AddWithValue("employee", employee);

using (NpgsqlDataReader reader = cmd.ExecuteReader())

{

while (reader.Read())

{

patientItem.Add(reader.GetString(0));

}

}

}

}

patients\_cmb.ItemsSource = null;

patients\_cmb.ItemsSource = patientItem;

patients\_cmb.SelectedIndex = 0;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

con.Close();

}

}

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

{

refresh\_cmb();

patients\_cmb\_SelectionChanged(null,null);

}

}