МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ
им. ГУМАРБЕКА ДАУКЕЕВА»
Кафедра IT-инжиниринг

«ДОПУЩЕН К ЗАЩИТЕ»

Зав. кафедрой, PhD, доцент Досжанова А.А.

_____ «___» ___ _____ 2020 г.
(подпись)

ДИЛОМНЫЙ ПРОЕКТ

На тему: Разработка информационной системы ведения электронных медицинских карт пациентов

Специальность 5В060200 – Информатика

Выполнила Рудич Екатерина Алексеевна Группа Инф-16-2

Научный руководитель: к.т.н., доцент Тусупова Б.Б.

Консультанты:
по экономической части: к.э.н., проф. Габелашвили К.Р.

_____ «___» _____ 2020 г.
(подпись)

по безопасности жизнедеятельности: ассист. Тыщенко Е.М.

_____ «___» _____ 2020 г.
(подпись)

по программному обеспечению: ст. преп. Майкотов М.Н.

_____ «___» _____ 2020 г.
(подпись)

Нормоконтролёр: ст. преп. Абсатарова Б.Р.

_____ «___» _____ 2020 г.
(подпись)

Рецензент: _____
(учёная степень, звание, Ф.И.О.)

_____ «___» _____ 2020 г.
(подпись)

Алматы 2020

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ
им. ГУМАРБЕКА ДАУКЕЕВА»

Институт систем управления и информационных технологий

Специальность 5В060200 – Информатика

Кафедра IT-инжиниринг

ЗАДАНИЕ
на выполнение дипломного проекта

Студенту Рудич Е.А.

Тема проекта: Разработка информационной системы ведения электронных медицинских карт пациентов

Утверждена приказом по университету № _____ от «____» _____ 2020 г.
Срок сдачи законченного проекта «____» _____ 2020 г.

Исходные данные к проекту (требуемые параметры результатов исследования (проектирования) и исходные данные объекта):

- интегрированная среда разработки Microsoft Visual Studio 2017;
- утилита для СУБД – Microsoft SQL Server Management Studio 2018.

Перечень вопросов, подлежащих разработке в дипломном проекте, или краткое содержание дипломного проекта:

1) Аналитическая часть
2) Проектная часть
3) Экспериментальная часть
4) Технико-экономическое обоснование
5) Раздел безопасности жизнедеятельности

Перечень графического материала (с точным указанием обязательных чертежей): 14 таблиц, 55 иллюстраций

Основная рекомендуемая литература:

1 C# documentation // DOCS.MICROSOFT.COM: Microsoft Docs. 2020.URL: https://docs.microsoft.com/en-us/dotnet/csharp/

2 SQL Server technical documentation // DOCS.MICROSOFT.COM: Microsoft Docs. 2020.URL: https://docs.microsoft.com/en-us/sql/sql-server/?view=sql-server-ver15

3 Абдуманонов А.А., Алиев Р.Э., Карабаев М.К., Хошимов В.Г. О проектировании медицинских баз данных и информационных систем для

организации и управления лечебно-диагностических процессов // T-Comm: Телекоммуникации и транспорт. – 2016. – Том 10. – №1. – С. 45-53.

Консультация по проекту с указанием относящихся к ним разделов проекта

Раздел	Консультант	Сроки	Подпись
Экономическая часть	Габелашвили К.Р.	30.04.2020	
Безопасность жизнедеятельности	Тыщенко Е.М.	30.04.2020	
Программная часть	Майкотов М.Н.	14.05.2020	
Нормоконтроль	Абсатарова Б.Р.	18.05.2020	

ГРАФИК
подготовки дипломной работы (проекта)

Наименования разделов, перечень разрабатываемых вопросов	Сроки представления научному руководителю	Примечания
Анализ предметной области, обзор литературы, программных и инструментальных средств	13.01.2020-31.01.2020	
Проектирование информационной системы	1.02.2020-28.02.2020	
Разработка программного обеспечения	1.02.2020-28.02.2020	
Экономическая часть	29.02.2020-30.04.2020	
Раздел безопасности жизнедеятельности	29.02.2020-30.04.2020	

Дата выдачи задания «____» _____ 2020 г.

Заведующий кафедрой _____ (Досжанова А.А.)
(подпись)

Научный руководитель проекта _____ (Тусупова Б.Б.)
(подпись)

Задание принял к исполнению студент _____ (Рудич Е.А.)
(подпись)

АҢДАТПА

Дипломдық жобаның тақырыбы – «Емделушілердің электрондық медициналық карталарын жүргізудің ақпараттық жүйесін құрұ».

Жұмыс жазылу кезінде зерттеу нысаны ретінде «Uniserv Medical Center» көпсалалы клиникасы қызмет етті, ол үшін C# бағдарламалау тілі мен Microsoft SQL Server ДББЖ көмегімен десктопты қосымша жасалды.

Дипломдық жоба кіріспеден, бес негізгі тараудан – аналитикалық, жобалық және эксперименттік бөлімдер, техникалық-экономикалық негіздеме, тіршілік қауіпсіздігі – және қорытындынан тұрады.

АННОТАЦИЯ

Темой дипломного проекта является «Разработка информационной системы ведения электронных медицинских карт пациентов».

Объектом исследования при написании работы послужила многопрофильная клиника «Uniserv Medical Center», для которой было разработано десктопное приложение с использованием языка программирования C# и СУБД Microsoft SQL Server.

Дипломный проект состоит из введения, пяти основных глав – аналитическая, проектная и экспериментальная части, технико-экономическое обоснование проекта, безопасность жизнедеятельности – и заключения.

ANNOTATION

The topic of the thesis project is «Development of an information system for maintaining electronic medical records of patients».

The object of research was «Uniserv Medical Center» medical clinic, for which the desktop application using C# programming language and Microsoft SQL Server DBMS was developed.

The thesis project contains introduction, five main chapters – analytical, project and experimental parts, technical-economic study, life safety – and conclusion.

Содержание

Введение

С каждым днём информационные технологии всё больше проникают в различные области нашей жизни и помогают улучшать её качество. Благодаря компьютерам и другой запрограммированной технике человек избавляется от необходимости делать вещи, которые может сделать за него компьютер. Это экономит важнейший человеческий ресурс – время, который можно потратить на совершение других немаловажных дел, на которые современные компьютеры пока не способны. В здравоохранении время играет критическую роль, так как на кону может быть жизнь человека. Поэтому необходимость в качественном внедрении информационных технологий и цифровизации здравоохранения не оставляет никаких сомнений [1].

Медицинская карта пациента является неотъемлемой частью системы здравоохранения. Это медицинский документ, без которого невозможно представить нормальное функционирование процессов медицинской организации. Медицинская карта больного содержит подробную информацию о пациенте, позволяет контролировать состояние его здоровья, своевременно выявлять заболевания и осуществлять лечение [2]. В связи с тем, что информации, которую необходимо хранить и извлекать в нужное время, много, этот процесс занимает существенную часть рабочего времени медицинских работников. В то же время, существует необходимость в объединении всех данных о пациенте и его здоровье в единую базу данных, так как они в большинстве случаев рассредоточены по разным медицинским учреждениям, в которые пациент обращается. Решению одной из этих задач посвящён настоящий дипломный проект, в результате которого будет разработано десктопное приложение для ведения электронных медицинских карт пациентов.

Объектом для исследования послужит работа многопрофильной медицинской клиники «Uniserv Medical Center».

Основная часть дипломного проекта состоит из аналитической, проектной и экспериментальной частей.

В аналитической части содержится обоснование необходимости автоматизации объекта исследования, описание его предметной области, обзор отечественных и зарубежных аналогичных продуктов, обзор программных и инструментальных средств для разработки необходимого программного обеспечения (ПО).

В проектной части описываются процессы разработки информационной модели системы, проектирования и программная реализация проекта.

В экспериментальной части содержатся инструкции по эксплуатации разработанного программного обеспечения, описываются назначение и условия выполнения программы.

Технико-экономическое обоснование содержит оценку эффективности использования разрабатываемого программного продукта за счёт его использования.

В разделе безопасности жизнедеятельности будут даны требования к условиям труда при проектировании системы, а также рассчитано производственное освещение рабочего места при работе разрабатываемым программным продуктом, удовлетворяющее требованиям техники безопасности и охраны труда.

Для выполнения дипломного проекта поставлены следующие задачи:

- изучение и сбор литературы по соответствующей теме;
- исследование предметной области проекта;
- разработка информационной модели и проектирование базы данных;
- разработка программного обеспечения;
- анализ технико-экономического обоснования;
- оценка охраны труда проекта;
- оформление в соответствии с установленными требованиями.

Целями дипломного проекта являются:

- подготовка выпускника к самостоятельному решению конкретной научно-технической задачи, для которой требуется квалификация бакалавра;
- закрепление и расширение знаний, полученных в процессе обучения, с последующим использованием их для решения инженерно-технических задач, связанных с темой данного проекта.

1 Аналитическая часть

1.1 Обоснование необходимости автоматизации объекта исследования

На сегодняшний день благодаря тому, что информационно-коммуникационные технологии и Интернет приобретают повсеместный характер, информатизация здравоохранения является достижимой целью.

Автоматизация здравоохранения призвана решить ряд задач:

- улучшить взаимодействие медицинских работников с пациентами;
- значительно уменьшить количество бумажной работы и сэкономить рабочее время медицинских работников;
- устранить проблему неразборчивого почерка;
- повысить качество и скорость предоставляемых услуг.

Одним из способов автоматизации здравоохранения является разработка информационной системы для ведения электронных медицинских карт (ЭМК) пациентов. Использование ЭМК по сравнению с «бумажными» носителями информации предлагает следующие преимущества:

- Данные будут храниться в защищённом облачном хранилище или на сервере организации, что обеспечит надёжность хранимой информации.
- Все данные о пациенте, в том числе перенесённые заболевания, наличие аллергии, результаты анализов и обследований, сохраняются, таким образом, их доступность обеспечена в любое время.
- ЭМК содержит полную историю болезни пациента в течение всей его жизни. Благодаря этому врач сможет быстрее проанализировать текущую ситуацию, более точно поставить диагноз и, соответственно, быстрее назначить лечение.
- Хранимая в электронном виде информация позволит вести статистику организации, что поможет работать над увеличением эффективности работы.
- Кроме того, ЭМК позволит работать с большими данными (big data), благодаря чему появится возможность предсказывать вероятность возникновения заболеваний, прогнозировать их развитие и многие другие возможности [1].

ЭМК составляют основу медицинских информационных систем (МИС). В западной литературе их называют EHR (Electronic Health Record). МИС – это система автоматизации документооборота медицинских организаций, которая содержит, в первую очередь, ЭМК, истории болезни пациентов, средства взаимодействия между персоналом, финансовая и отчётная информация. Благодаря созданию МИС и их интеграции с ИС других инфраструктур станет возможным осуществление государственных программ по отслеживанию качества здравоохранения в стране. Например, использование такой МИС позволит своевременно получать информацию об оказанных населению медицинских услугах, чтобы в дальнейшем более

эффективно распределять бюджетные средства на медицину. Такое решение не может не сказаться положительно на экономике государства.

На сегодняшний день единую систему ЭМК успешно используют во многих странах: Австрии, Израиле, Испании, Сингапуре, Словакии, Финляндии, Франции, Швеции, Эстонии, Японии и частично в Великобритании [3]. Согласно данным консалтинговой фирмы Frost & Sullivan, занимающейся исследованиями, доход от цифрового рынка здравоохранения в 2021 году составит 6 млрд долларов, что почти в 10 раз больше по сравнению с 2014 годом [4]. Львиную долю рынка телемедицины занимает США. В Республике Казахстан внедрение единой системы ЭМК пока остаётся актуальной задачей.

Ещё в начале 2000-х годов в Казахстане планировали реализовать переход на Единую информационную систему здравоохранения, но он оказался неуспешным [5]. 10 января 2018 года Первый Президент Казахстана в Послании народу выразил необходимость в повышении эффективности медицинской помощи населению через «интеграцию информационных систем, … внедрение электронных паспортов здоровья и переход на «безбумажные» больницы» [6]. По словам Министра здравоохранения РК Елжана Биртанова, автоматизация деятельности позволила сэкономить на закупе бумаги, на 8% уменьшилось количество посещений медицинских организаций и на 50% – время нахождения пациентов в них до получения услуг, что в общей сложности позволило сэкономить бюджет государства на 50 млрд тенге в год [7]. В настоящее время по данным «Республиканского центра электронного здравоохранения» РК взаимодействия ИС Минздрава со сторонними МИС находятся на стадии тестирования. Согласно Дорожной карте проекта «Цифровизация системы здравоохранения», охват населения электронными паспортами здоровья составляет 60% в 2020 году, предполагается достичь 90%-го охвата к 2025 году. Охват рынка МИС-ами в 2019 году по областям показан на рисунке 1.1 [8]. Исходя из этих данных видно, что в г. Нур-Султан лишь 69,4% медицинских организаций внедрили МИС, а в г. Актобе этот показатель не достиг и половины (33,3%). Наилучшая ситуация наблюдается в Западно-Казахстанской области и в г. Шымкент.

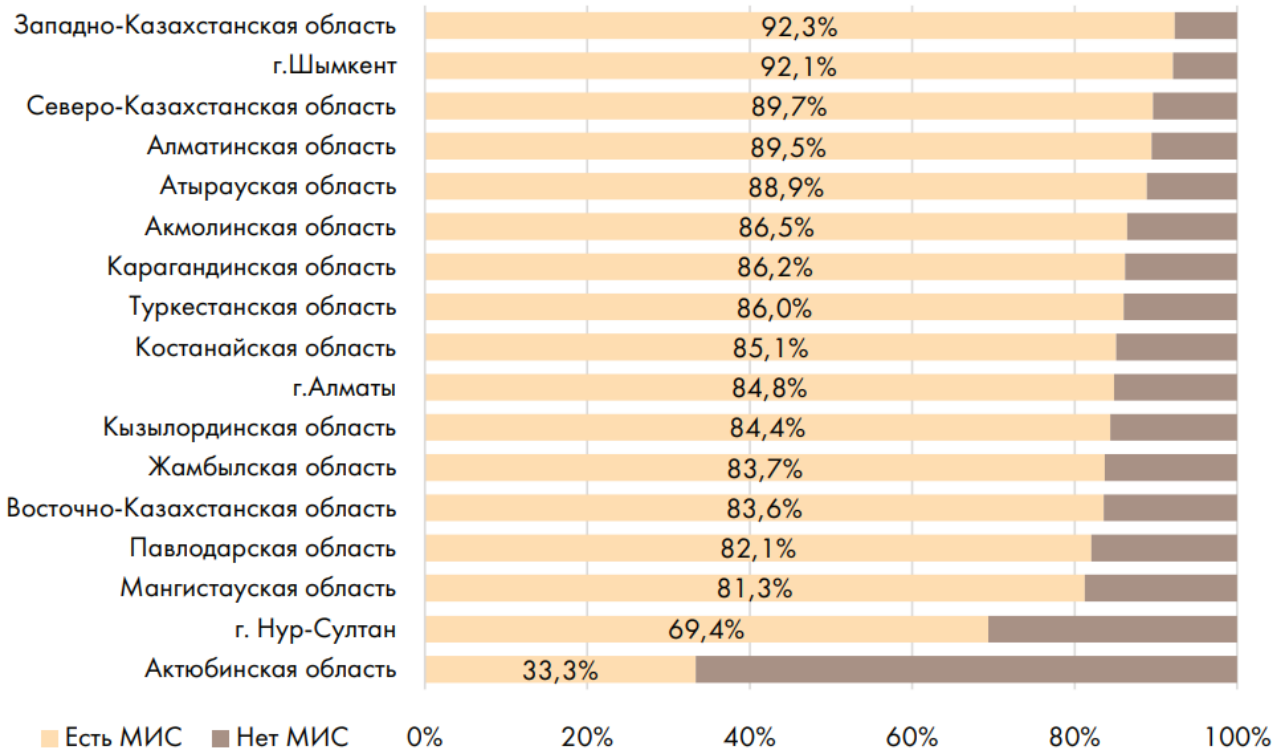


Рисунок 1.1 – Охват рынка Казахстана МИС-ами

По итогам данного раздела можно сделать следующие выводы. Благодаря внедрению ЭМК будет существенно сэкономлено рабочее время специалиста, так как значительно уменьшится количество бумажной работы за счёт автоматизации процесса обработки информации и устранения проблемы неразборчивого почерка, существенно увеличится объём перерабатываемой информации. В конечном итоге применение ЭМК приведёт к повышению уровня здравоохранения и экономики в стране. Данные на рисунке 1.1 только подтверждают актуальность разработки ЭМК и необходимость их применения.

1.2 Описание предметной области

Прежде чем приступить к проектированию информационной системы, нужно изучить предметную область, для которой она разрабатывается. Анализ предметной области повлияет на дальнейшие этапы разработки и поможет определить основные аспекты проекта, такие как: требования к разрабатываемой системе, взаимодействие с пользователем, какую информацию потребуется хранить в базе данных и т. д.

Практическим результатом данного дипломного проекта будет разработка информационной системы (ИС) ведения электронных медицинских карт пациентов в виде настольного (десктопного) приложения. Предметной областью является деятельность многопрофильной клиники «Uniserv Medical Center», для которой будет разработано приложение.

Клиника была открыта 3 января 2019 года в г. Уральске (ЗКО) и состоит из пяти отделений: консультативно-диагностическое, стоматологическое, офтальмологическое, отделение лучевой диагностики и клинико-диагностическая лаборатория.

В клинике работают сотрудники регистратуры (операторы) и врачи (в том числе главный врач клиники, его заместитель и заведующие отделениями).

Клиника предлагает широкий спектр услуг в сфере общей практики, эндокринологии, гинекологии, проктологии, гастроэнтерологии, пульмонологии-аллергологии, урологии, неврологии, кардиологии, нефрологии, онкологии-маммологии, отоларингологии, офтальмологии, стоматологии и функциональной диагностики.

Сотрудники регистратуры (операторы) осуществляют учёт картотек пациентов, а также запись пациентов на приём к желаемому врачу. Если пациент обращается впервые, оператор вводит пациента в базу данных, после чего пациента можно записать на приём. Сделать это можно как при личном обращении в клинику, так и по телефону. Консультация, обследование пациента и диагностика здоровья проводятся врачами клиники. Именно они могут фиксировать приём пациента, ставить диагноз, назначать ему последующее лечение и записывать на повторный приём и, помимо этого, вносить изменения в данные пациента.

Организационная структура сотрудников клиники «Uniserv Medical Center» по должностям показана на рисунке 1.2.

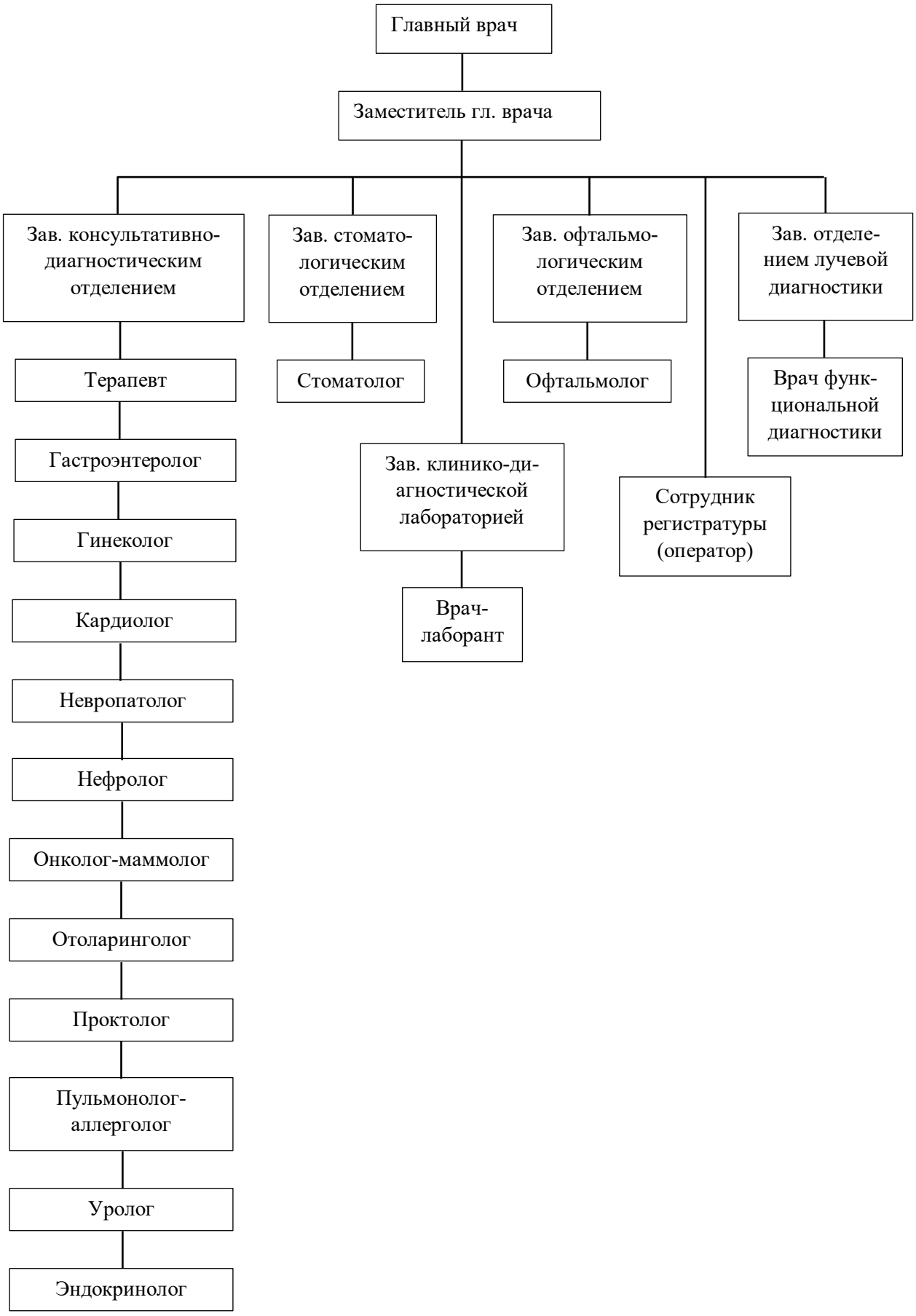


Рисунок 1.2 Организационная структура сотрудников клиники

Ключевые моменты работы клиники «Uniserv Medical Center»:

1) Оказание услуг начинается с обращения пациента в регистратуру. В медцентр приходит пациент, оператор создаёт карту пациента в базе данных и выписывает направление к требуемому врачу.

2) Во время приёма врач фиксирует данные об осмотре пациента в базу данных, распечатывает эту информацию для пациента и при необходимости вносит изменения в его картотеку. Врач также может выписать направление на повторный приём и распечатать талон пациенту.

3) Свободные для записи даты и время зависят от должности врача:

- главный врач и его заместитель могут вести приём только в понедельник с 12:00 до 17:00;
- заведующие отделений – во вторник и среду с 12:00 до 17:00;
- остальные врачи принимают с понедельника по пятницу с 8:00 до 19:00, в субботу – с 9:00 до 14:00.

4) На один приём предусмотрено до 30 минут.

5) Стоимость приёма формируется в зависимости от:

- должности ведущего приём врача. Стоимость первичного приёма:
- главного врача – 25 000 тенге;
- заместителя главного врача – 20 000 тенге;
- заведующего отделением – 15 000 тенге;
- остальных врачей – в зависимости от категории:
- вторая – 5 000 тенге за первичный приём;
- первая – 7 000 тенге;
- высшая – 10 000 тенге.
- вида приёма (первичный, повторный).

Скидка на повторный приём составляет 1 000 тенге от стоимости первичного приёма.

6) В клинике ведётся статистика посещений. Она формируется по трём критериям: выбранному врачу, специализации, за указанный месяц.

7) В медицинской картотеке хранится информация:

- о пациентах: уникальный номер пациента, его фотография, фамилия, имя, отчество, дата рождения, возраст, пол, категория гражданина, вид документа, номер документа, ИИН (при наличии), телефон, домашний адрес, группа крови, резус-фактор, рост, вес, перенесённые заболевания, аллергии (при наличии), фамилия и инициалы лечащего врача и кто внёс изменения в картотеку пациента;
- о врачах: уникальный номер врача, его фамилия, имя, отчество, должность, специализация и категория;
- об истории посещений пациента (номер приёма, фамилия и инициалы врача, проведшего осмотр, дата приёма) с возможностью просмотреть подробную информацию о посещении, заполненную врачом;
- записях пациентов на приём к врачу (уникальный номер записи, фамилия, имя и отчество пациента, фамилия и инициалы врача, вид приёма,

назначенные дата и время, стоимость приёма, кто и когда записал). Направление на приём должен быть предоставлен пациенту.

8) С помощью медицинской карты пациента врач осуществляет осмотр пациента, внося в неё необходимые данные: фамилия, имя и отчество свои, а также пациента, дата и время посещения, анамнез жизни, жалобы, данные объективного исследования, диагноз, назначенное лечение, план обследования, дату следующего приёма и стоимость приёма. Врач также должен предоставить результаты осмотра для пациента (выписку).

9) Для каждого врача и каждой специализации проводится мини-статистика – подсчёт количества посещений – для улучшения предоставляемых услуг и качества обслуживания.

1.3 Обзор отечественных и зарубежных аналогичных продуктов

Аналогов электронной медицинской карты в отдельности не существует, существуют аналоги медицинской информационной системы. Рассмотрим их.

Согласно официальным данным Минздрава РК, внедрённые медицинские ИС представлены на рисунке 1.3 [9]. Данные МИС являются отечественными разработками.

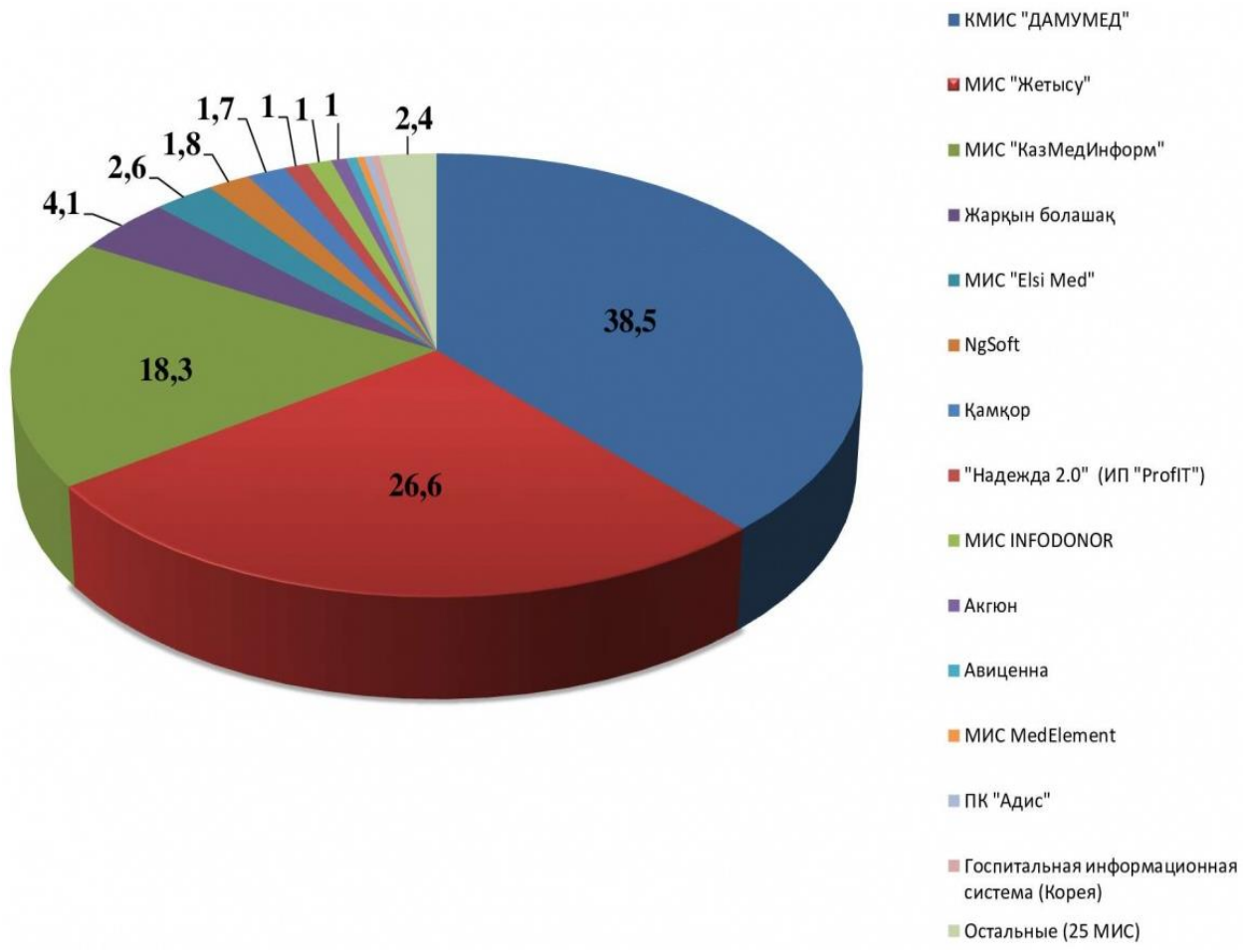


Рисунок 1.3 – Доля МИС в частных медицинских организациях РК

1.3.1 Отечественная МИС «Damumed»

Как видно из рисунка 1.3, преобладающую долю рынка МИС в Казахстане занимает «Damumed» при большом количестве конкурентов. Данная МИС разработана ТОО «Центр Информационных Технологий “Даму”» и является веб-приложением, т.е. доступ к ней осуществляется посредством веб-браузера. МИС «Damumed» в зависимости от потребностей предлагает обширный пакет возможностей: модуль «Регистратура», модуль «Кабинет врача» (позволяет просматривать данные пациента и вести приём), модуль «Профилактика» (позволяет контролировать запланированные осмотры и скрининги), модуль «Флюоротека» (осуществляет профилактику туберкулёзных заболеваний), модуль «Кабинет пациента» (позволяет записаться на приём и вызвать врача на дом), модуль «Ситуационный центр» (мониторинг информации для ведения статистики работы организации), модуль «Стационар» (помогает организовать рабочее пространство медицинского работника, отслеживать показания пациента и различные финансовые показатели). Помимо веб-приложения, «Damumed» предлагает ПО для стационарных терминалов самообслуживания для записи пациента к врачу самостоятельно [10]. Интерфейс программы представлен на рисунке 1.4.

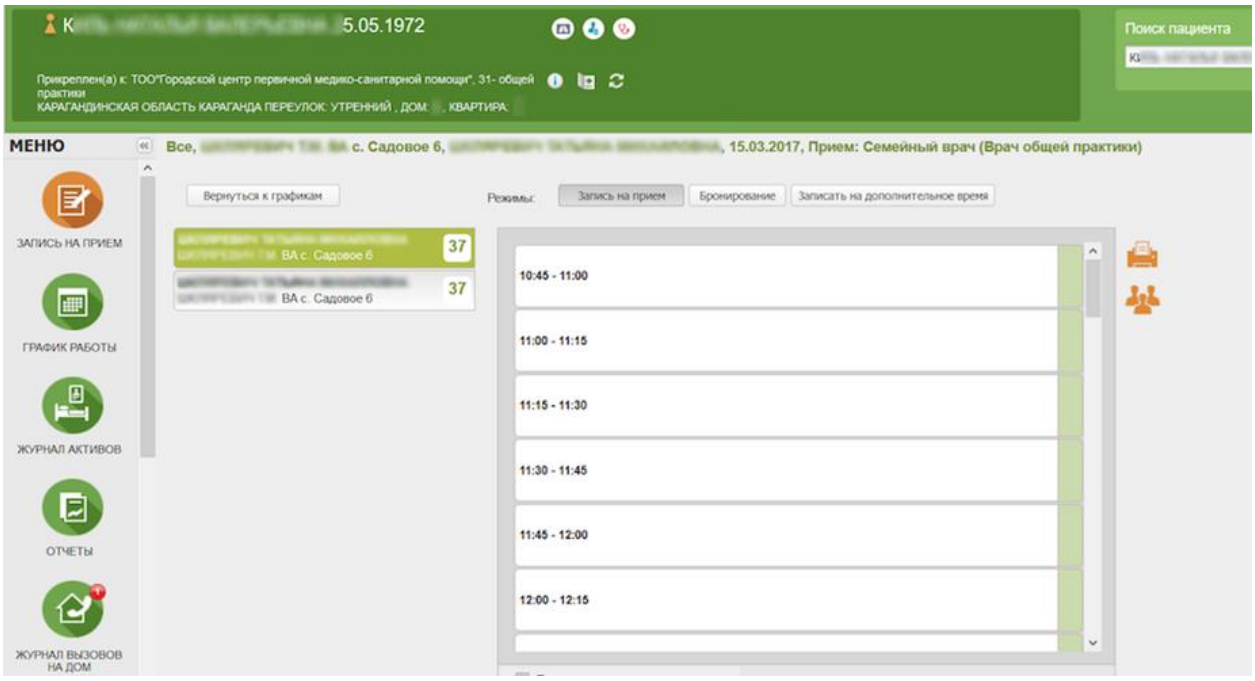


Рисунок 1.4 – Интерфейс МИС «Damumed»

Единственным минусом, пожалуй, ввиду мощного функционала и монопольного положения на рынке, является дороговизна данного программного продукта.

1.3.2 Отечественная МИС «Жетісу»

Данная МИС позволяет автоматизировать все лечебно-профилактические процессы медицинского учреждения. Включает в себя автоматизированные рабочие места: «Регистратура», «Врач», «Приёмный покой», «Медсестра», «Лаборант» и модули: «Стационар», «Поликлиника», «Скорая помощь», «Лаборатория». Выполняет следующие функции: регистрация пациентов, закрепление за специалистом и запись на приём, учёт медработников и выдачи листков нетрудоспособности. Разработчик данной МИС является официальным партнёром фирмы «1С». МИС «Жетісу» основана на клиент-серверной технологии и реализована с помощью «1С:Бухгалтерия 8 для Казахстана» и Microsoft SQL Server [11].

1.3.3 Российский аналог МИС «Medesk»

Рассмотрим зарубежный российский аналог МИС – «Medesk», который также является отечественной разработкой и представлен во многих рейтингах российских МИС. Так же, как и «Damumed», представляет собой веб-приложение и предлагает следующие возможности: функционал регистратуры, онлайн-запись к врачу, SMS-оповещение пациентов, модуль «Рабочее место врача», работу с кассовыми аппаратами, проведение удалённых консультаций посредством видеоконференций, учёт расходов и ведение статистики организации. Интерфейс показан на рисунке 1.5 [12].

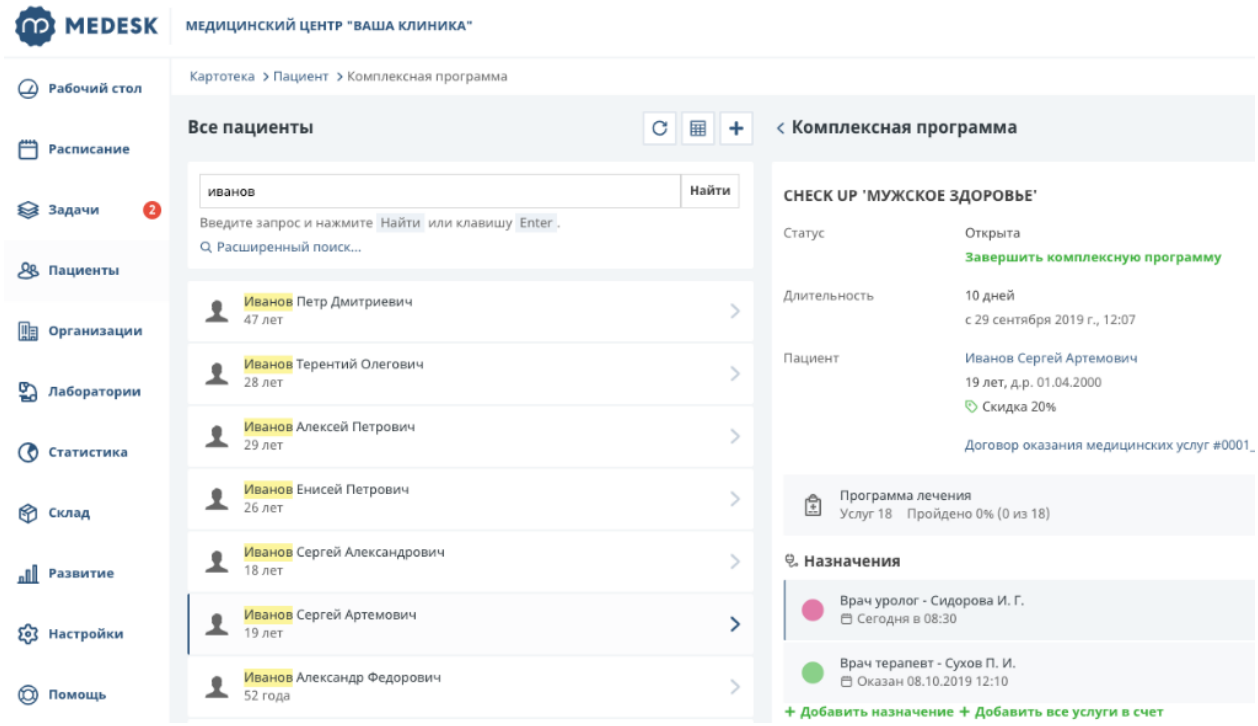


Рисунок 1.5 – Интерфейс МИС «Medesk»

К плюсам можно отнести то, что приложение подходит для клиник всех специализаций и предлагает неоплачиваемый пробный период. Так как приложение использует облачные технологии, исчезает необходимость в закупке серверов и оборудования. Круглосуточная бесплатная техподдержка от разработчиков позволяет не нанимать дополнительных IT-сотрудников для поддержки работоспособности данного ПО.

1.3.4 Западный аналог МИС «AdvancedMD»

Одним из популярных западных аналогов МИС – «AdvancedMD» HER (рисунок 1.6). По результатам многих исследований она занимает лидирующие позиции среди англоязычных EHR. «AdvancedMD» предлагает программные решения не только для медицинских организаций, но и для независимых практикующих специалистов. Разработчики данного ПО акцентируют внимание на удобном UI (User Interface, интерфейс пользователя), который призван повысить не только личную эффективность и работоспособность медработника, но и всей организации в целом. Пользователь составляет список задач, которые нужно выполнить в порядке приоритетности, отслеживает объём уже выполненной работы. Интегрированная система рабочих процессов и отслеживания пациентов во время нахождения в клинике упрощает процесс посещения для пациентов и даёт возможность для телемедицины. Также реализована система автоматического оповещения пациентов посредством SMS или электронной почты, в случае окончания срока действия страховки персонал также будет предупреждён. В остальном функционал приложения схож с функционалом ранее рассмотренных продуктов [13].

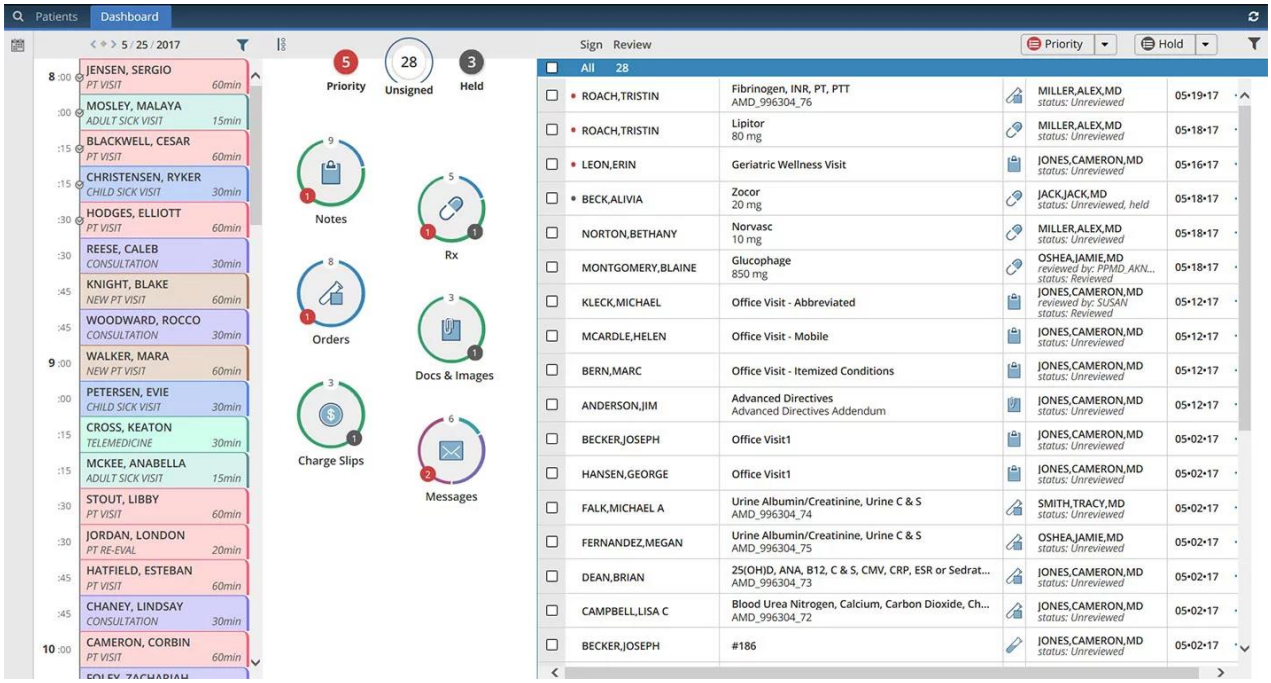


Рисунок 1.6 – Интерфейс МИС «AdvancedMD»

Учитывая, что на западном рынке очень много конкурентноспособных EHR, пользователи могут сравнить их между собой. По мнению некоторых независимых специалистов, проводящих частные практики и использующих «AdvancedMD», оно слишком замысловатое, что для них является недостатком. Специалисты по ментальному здоровью (психологи, психотерапевты и т.д.) считают, что приложение не очень для них подходит и его цена неоправданно завышена. Также есть мнение что приложение лучше всего работает в браузере Internet Explorer; это не очень удобно, учитывая, что многие пользуются браузерами, отличными от Internet Explorer, такими как Google Chrome и др. В остальном пользователи очень довольны данным продуктом и рекомендуют его своим коллегам.

В данном разделе были рассмотрены отечественные и зарубежные аналоги разрабатываемой в дипломном проекте информационной системы. В ходе проведённого анализа можно сделать вывод, что отечественные разработки по функционалу и возможностям уступают зарубежным и есть необходимость в создании конкурентноспособного отечественного продукта.

1.4 Обзор программных и инструментальных средств

В результате данной дипломной работы будет разработано настольное (десктопное) приложение для ведения электронных медицинских карт пациентов. Приложение представляет собой программное обеспечение (ПО). Для создания такого ПО тоже потребуется ПО, но уже другого назначения.

Программное обеспечение для ПК делится на три вида (как показано на рисунке 1.7).

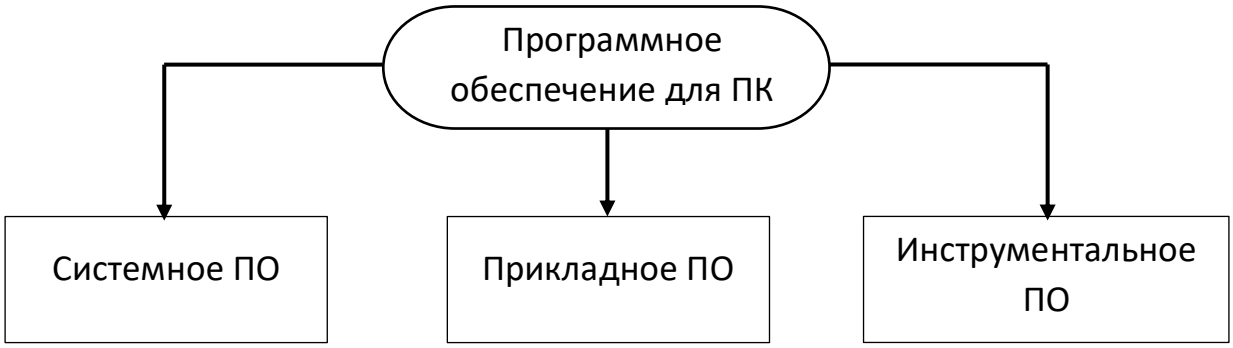


Рисунок 1.7 – Виды ПО для ПК

Системное ПО предназначено для аппаратной стороны ПК и осуществляет взаимодействие и бесперебойную работу его компонентов. Операционная система (ОС) относится к такому ПО. В дипломной работе будет использоваться ОС «Windows 10 Домашняя», новейшая версия линейки от «Microsoft Corporation».

Прикладное ПО предназначено для выполнения конкретной задачи. К нему относятся текстовые и графические редакторы, браузеры, архиваторы, сервисы видеоконференций, компьютерные игры и пр. Прикладное ПО, которое будет использоваться в дипломном проекте – программный пакет «Microsoft Office 365», браузер «Google Chrome», калькулятор, инструмент заметок «Simple Sticky Notes».

С помощью инструментального ПО можно осуществлять работу с кодом программы: создавать, изменять, отлаживать и запускать его. Таким ПО пользуются программисты-разработчики, для этого нужно знать специальный машинный код или язык программирования. Примерами инструментального ПО являются текстовые редакторы кода, компиляторы, дебаггеры и т. д. Интегрированная среда разработки сочетает в себе основные виды инструментального ПО для работы с различными языками программирования и системами управления базами данных.

С момента появления информационных технологий средства разработки прошли большой этап развития и на сегодняшний день существует большое разнообразие программных и инструментальных средств для разработки ПО, поэтому есть возможность выбора наиболее удобной и практичной технологии. Чтобы выяснить, какое инструментальное ПО следует использовать в разработке дипломного проекта, нужно провести сравнительный анализ.

Во-первых, необходимо определиться: приложение какого типа будет разрабатываться. Приложения делятся на десктопные и веб-приложениями. Десктоп-приложение – это программа, которая обрабатывается на стороне клиента и работает автономно на рабочем столе компьютера или ноутбука пользователя. Веб-приложение основано на технологии «клиент – сервер», взаимодействие которых осуществляется через браузер. Нельзя сказать, какие приложения используются чаще и больше, поскольку доля их использования примерно одинаковая. В этой связи вопрос «что лучше: десктоп или веб» вызывает многочисленные споры у разработчиков. Выбор основывается на том, какие требования предъявляются к разрабатываемому приложению. Рассмотрим основные отличительные характеристики в таблице 1.1.

Таблица 1.1 – Сравнительная характеристика десктопных и веб-приложений

Параметр	Десктопное приложение	Веб-приложение	Вывод
Доступность	Привязано к физическому местоположению	Доступно в любой точке мира, где есть Интернет	Доступность десктопного приложения ограниче-на
Сопровождение	Устанавливается каждый раз на каждое новое устройство	Устанавливается единожды	В случае с десктопными приложениями процесс установки обновлений довольно обременительный т. к. его придётся устанавливать по

			отдель-ности на каждом устрой-стве

Продолжение таблицы 1.1

Производи-тельность	Являются автономными и не зависят от Интернета	Зависит в исключительной степени от наличия Интернета и его скорости	Отсутствие Интернета или его плохое качество приводит к потере производительности веб-приложения
Расходы на интернет-трафик	Могут быть исключены	Есть всегда	Для веб-приложения расходы на Интернет будут всегда
Безопасность и надёжность	Офлайн-работа поможет избежать утечки данных	Вероятность нестабильной работы браузера и утечки информации	Надёжнее и безопаснее использовать десктопное приложение
Интерфейс	В обоих случаях современные технологи позволяют реализовать любое интересующее интерфейсное решение		
Функциональ-ность	Благодаря разработчикам функциональность обоих приложений не уступает друг другу		
Кроссплатфор-менность	Предназначено только для ПК и ноутбука	Работает в любом устройстве, где есть браузер и интернет	Веб-приложение доступно с любого устройства
Быстродейст-вие	Десктопное приложение всё же работает быстрее, чем веб-приложение		
Работа с файла-ми	Локальные файлы и специфичное оборудование (например, сканер удостоверения личности) для веб-приложений недоступны		

Определим основные критерии разрабатываемого ПО:

1) ПО будет использоваться стационарно – на рабочем месте медработника, т. е. мобильность приложения не критична.
2) Обновления ПО предполагаются в редких случаях, поэтому их установка не будет довольно проблематичной.
3) Приложение предназначено для ПК, поэтому кроссплатформенность не требуется.

Таким образом, изучив все требования к создаваемому ПО, было принято решение разрабатывать десктопное приложение.

Ниже в таблице 1.2 приведён рейтинг популярных языков программирования на май 2020 года согласно авторитетному индексу TIOBE, который основан на поисковых запросах в Google, Bing, Yahoo, Wikipedia, Amazon, YouTube и Baidu [14].

Таблица 1.2 – Индекс «TIOBE»

Май 2020 г.	Май 2019 г.	Изменение	Язык программирования	Рейтинг	Изменение

1	2	↑[1]	C	17,07%	+2,82%
2	1	↓[2]	Java	16,28%	+0,28%
3	4	↑	Python	9,12%	+1,29%
4	3	↓	C++	6,13%	-1,97%

Продолжение таблицы 1.2

5	6	↑	C#	4,29%	+0,30%
6	5	↓	Visual Basic	4,18%	-1,01%
7	7	=[3]	JavaScript	2,68%	-0,01%
8	9	↑	PHP	2,49%	-0,00%
9	8	↓	SQL	2,09%	-0,47%
10	21	↑↑[4]	R	1,85%	+0,90%
11	18	↑↑	Swift	1,79%	+0,64%
12	19	↑↑	Go	1,27%	+0,15%
13	14	↑	MATLAB	1,17%	-0,20%
14	10	↓↓[5]	Assembly language	1,12%	-0,69%
15	15	=	Ruby	1,02%	-0,32%
16	20	↑↑	PL/SQL	0,99%	-0,03%
17	16	↓	Classic Visual Basic	0,89%	-0,43%
18	13	↓↓	Perl	0,88%	-0,51%
19	28	↑↑	Scratch	0,83%	+0,32%
20	11	↓↓	Objective-C	0,80%	-0,83%

Как видно из таблицы 1.2, за последний год самым популярным языком стал C, который на протяжении долгого времени конкурировал с Java за первенство (последний раз он был на первом месте в 2015 году). По мнению CEO (генеральный директор) TIOBE, Пола Янсена, это, возможно, связано с коронавирусом. Может прозвучать глупо, но некоторые ЯП действительно извлекают выгоду из этой ситуации. Например, языки Python и R, которые активно используются в Data Science (наука о данных) для поиска лекарства от коронавируса. Языки C и C++ также пользуются спросом за счёт их использования в медицинских устройствах. Резко возросла популярность и ЯП Swift, используемого в основном для разработки приложений для устройств от Apple, а также языка Go, разработанного в компании Google в 2009 году и рассматриваемого в качестве замены языкам C и C++. С другой стороны, стоит отметить, что такие ЯП, как Assembly language (язык ассемблера), Perl и Objective-C потеряли прежнюю востребованность. JavaScript и Ruby сохранили свои позиции (7 и 15, соответственно) по сравнению с прошлым годом.

Согласно рейтингу TIOBE, в топе-7 популярных ЯП на протяжении последних двух лет сохранились C, Java, Python, C++, C#, Visual Basic и

[1] Поднялся на одну строку.
[2] Опустился на одну строку.
[3] Остался на том же уровне.
[4] Поднялся на несколько строк.
[5] Опустился на несколько строк.

JavaScript. На каждом из этих языков можно писать десктоп-приложения. Рассмотрим каждый по отдельности.

Язык C появился в 1972 году. Изначально предназначался для ОС Unix, позже был перенесён и на остальные платформы. Язык C завоевал популярность в 1980-х годах и стал одним из самых часто используемых языков. На нём написаны многие операционные системы и различное ПО, например браузеры, драйверы. Среди разработчиков десктопных приложений данный язык не очень популярен, т. к. он сложен для освоения, потому что для него существует мало обучающей литературы, и требует большого объёма кода, т. к. имеет небольшое количество стандартных библиотек, в связи с чем необходимо устанавливать специальные библиотеки. На основе языка C были созданы языки C++, C#, Java и Objective-C.

Язык Java был создан в 1995 году. Изначально был задуман для программирования бытовых устройств, затем с его помощью создали карманный ПК (из-за своей дороговизны он не стал популярным) и интерактивное телевидение (также не приобрело популярность). С середины 1990-х годов на Java начали разрабатывать клиент-серверное ПО. Java предлагает широкие возможности для разработки и стал является одним из самых востребованных языков программирования, на нём написаны многие виды ПО: многие приложения для авиаслужб, авиакомпаний, банков, CRM-системы, IDE (среда разработки) и редакторы кода, Android-приложения. Если требуется разработать кроссплатформенное ПО, то Java станет хорошим вариантом. Java использовали такие компании-гиганты, как Ebay, Amazon, Linkedin, Google, Twitter, Facebook, он зарекомендовал себя как быстрый и надёжный язык программирования с высоким уровнем защиты.

JavaScript возник в 1995 году. Он кроссплатформенный, для изучения достаточно прост, испытал влияние языков C и Java. Очень популярен, т. к. практически всегда используется в веб-разработке, браузерных ОС, разработке браузерных приложений. На нём также можно писать серверные, мобильные приложения, прикладное ПО (например, на JavaScript частично написан браузер Mozilla FireFox). С некоторого времени стало возможным создавать и десктопные приложения с помощью JavaScript. Для этого существует несколько фреймворков, таких как Electron, Proton Native, Meteor и др.

Python существует с 1991 года. Его синтаксис довольно прост, а код легко читаем, поэтому язык обладает низким порогом вхождения. Python используется для написания прикладного ПО и в веб-разработке, обладает высокой производительностью и большим количеством встроенных библиотек. Последние пару лет активно используется в Data Science (наука о данных) и Machine learning (машинное обучение), поэтому имеет большой спрос и становится всё более популярным. Однако, если рассматривать обычные программы, то скорость их работы не такая высокая, особенно если сравнивать Python с C или C++.

Язык C++ был разработан в далёком 1983 году, предшественник языка C. Многим известен своим сложным синтаксисом, что представляет определённые трудности для новичков при его изучении. У C++ существует огромное количество библиотек, на нём можно писать всё что угодно (например, браузер, сервер, игры) и для чего угодно (язык кроссплатформенный). При выборе правильного подхода C++ позволяет создавать приложения любой сложности, максимально производительные, быстрые и мощные. Разработка интерфейса может доставить определённые трудности, т.к. необходимые фреймворки сложны для освоения. В основном C++ становится выбором опытных разработчиков при написании программ, требовательных к производительности. Однако, для того чтобы написать хороший код на C++, нужно приложить немало усилий, чтобы основательно его выучить.

Microsoft Visual Basic появился одновременно с первой успешной версией ОС Windows в 1991 году, а в 2001 году он эволюционировал в Visual Basic .NET (VB.NET). Синтаксис очень похож на обычный английский язык, изучать его довольно легко, поэтому он пользуется спросом у начинающих программистов. На VB.NET очень удобно разрабатывать десктопные приложения. Но сегодня его эра подходит к концу, опытными программистами он практически не используется. В марте этого года Microsoft объявила, что больше не будет развивать этот язык и добавлять в него новые функции. VB.NET был вытеснен языком C#, который очень похож на него и разработан также Microsoft. C# обладает бóльшими возможностями и, в отличие от VB.NET, обновляется и поддерживается по настоящее время.

Первая версия C# вышла в 2002 году вместе с выходом Microsoft Visual Studio .NET [15]. Он был разработан специально для платформы .NET Framework. О языке C# можно сказать, что «с синтаксической точки зрения он является таким же чистым, как Java, столь же простым, как Visual Basic, и таким же гибким и мощным, как C++» [16]. Для его изучения существует много литературы и обучающих видео; он достаточно лёгок для освоения, написанный на C# код понятен и лаконичен. Плюс ко всему разработчиками создана подробная документация, а также большое количество фреймворков и библиотек [17]. Многие программисты находят C# идеальным и, пожалуй, лучшим для разработки приложений под Windows. Однако, на нём можно писать и для Mac, Linux, iOS и Android. Помимо этого, он используется для создания веб- и мобильных приложений, веб-сервисов, игровых программ.

Рассмотрев самые популярные на сегодняшний день языки программирования, было решено разрабатывать ИС для дипломного проекта на языке C#. Выбор обусловлен большим количеством доступной литературы, обучающих материалов, простотой и надёжностью языка.

Как уже было сказано, язык был разработан специально для платформы .NET Framework, соответственно и разработка будет вестись на нём. Стоит отметить, что помимо C# данный фреймворк предлагает возможности разработки на Visual Basic, JScript, C++, F#, J# и имеет разные готовые API-

интерфейсы: ASP.NET, WPF, ADO.NET и др. Для разработки десктопных приложений используется WPF – Windows Presentation Foundation. Наиболее подходящей интегрированной средой разработки (IDE) является Microsoft Visual Studio, т. к. и сам язык C#, и платформа .NET разработаны Microsoft. Она официальная и, к тому же, бесплатная (версия Community Edition). Среда разработки упрощает процесс разработки приложения, предлагает удобный редактор кода (который содержит подсказки и автоматические отступы, конструктор (показывающий как будет выглядеть приложение), а также позволяет отлаживать и опубликовывать приложения [18].

Таким образом, разработка десктопного приложения для настоящего дипломного проекта будет вестись на языке C#, платформе .NET Framework в среде разработки Microsoft Visual Studio 2017. Интерфейс программы показан на рисунке 1.8.

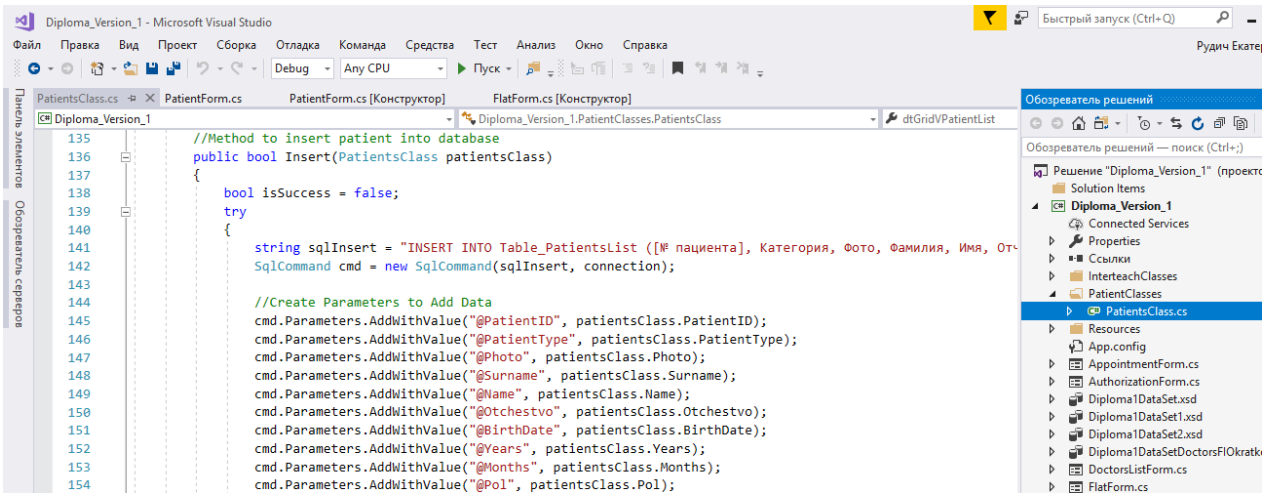


Рисунок 1.8 – Интерфейс Microsoft Visual Studio 2017

2 Проектная часть

2.1 Разработка информационной модели проектируемой системы

Для разработки ИС ведения электронных медицинских карт пациентов нужно определить процессы, которые будут автоматизированы.

Диаграмма процессов клиники показана на рисунке 2.1. Главной задачей клиники является предоставление услуги пациенту. Входными данными будет «Пациент». Выходными данными будут «Карта в регистратуру» и «Пациент, получивший услугу». Процесс предоставления услуги будет выполняться оператором или врачом, которые должны следовать распорядку клиники.

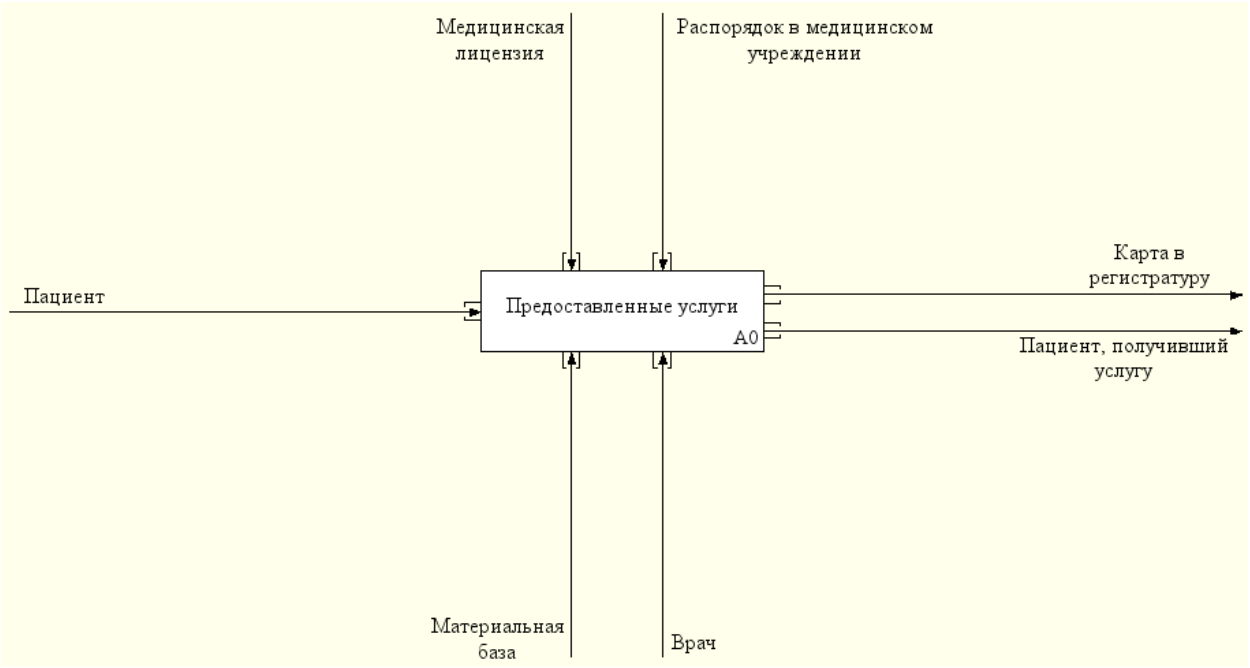


Рисунок 2.1 – Диаграмма процессов клиники

На рисунке 2.2 представлена декомпозиция диаграммы процессов клиники. Первым процессом является «Регистрация», входными данными будет «Пациент», выходными – «Медицинская карта». В регистрации пациента участвует оператор.

Вторым процессом является «Осмотр», входные данные – «Пациент после регистрации», выходные – «Карта с записями осмотра» и «Направление на повторный осмотр». Осмотр проводится врачом.

Третьим процессом является «Лечение», входными данными является «Пациент после назначенного лечения», выходными – пациент, который получил услугу. Если пациент не вылечился, то он отправляется на повторное посещение и обследование.

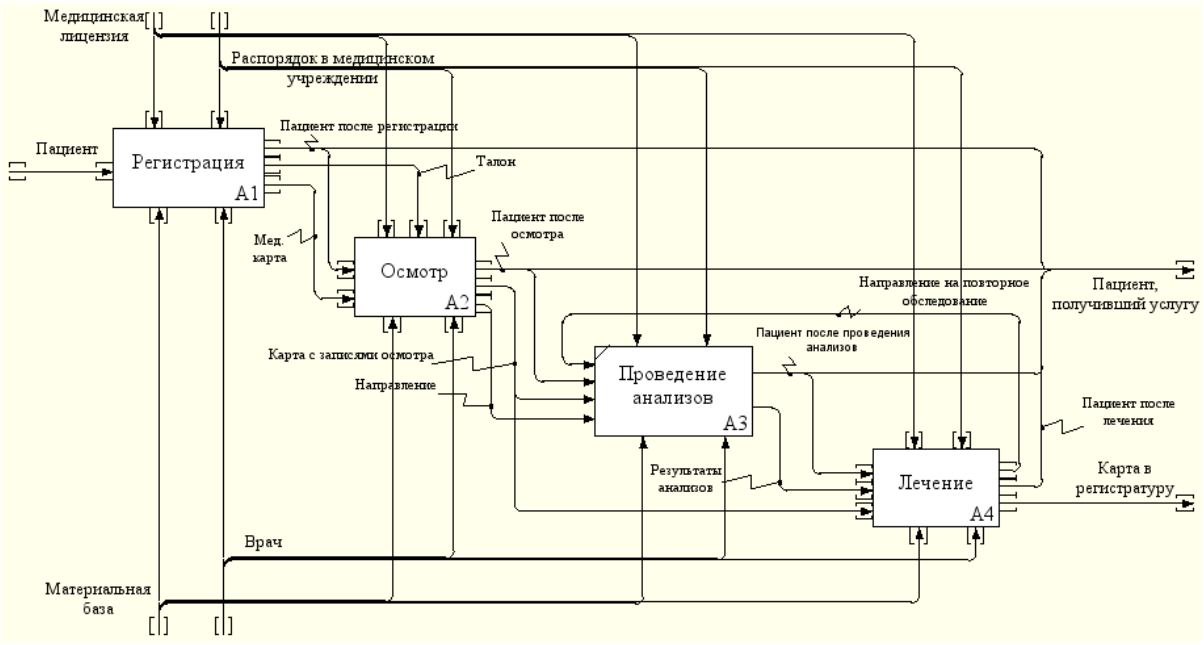


Рисунок 2.2 – Декомпозиция процессов клиники

На рисунке 2.3 представлена декомпозиция процесса «Регистрация». Первым процессом будет «Предварительная запись», входные данные – «Пациент», выходные – «Начальные данные о больном».

Второй процесс – «Оформление направления», входные данные – «Пациент после предварительной записи», на выходе – «Направление на приём».

Третий процесс – «Оформление медицинской карты». Входными данными будет «Пациент после получения направления», выходными – «Карта». В этих процессах принимает участие оператор.

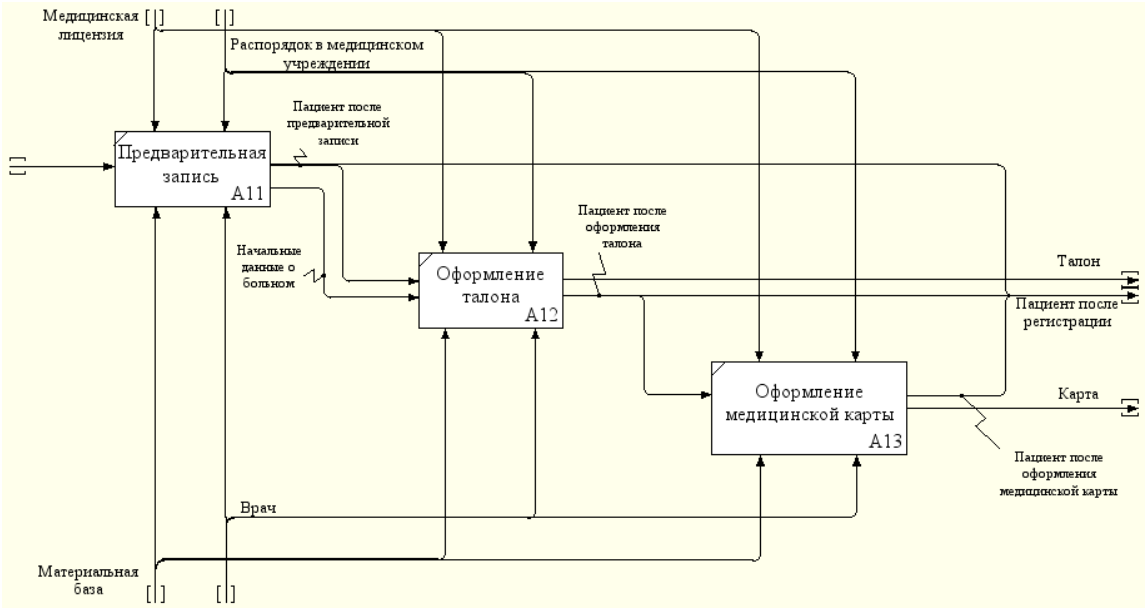


Рисунок 2.3 – Декомпозиция процесса «Регистрация»

На приёме врач осматривает пациента, вносит информацию о его состоянии в карту и составляет план обследования и лечения. На рисунке 2.4 представлена декомпозиция процесса «Осмотр». Первым процессом является «Беседа с пациентом», входными данными будет «Пациент после регистрации», выходными – «Данные о больном».

Вторым процессом является «Осмотр пациента», входными данными являются «Данные о состоянии пациента», на выходе – «Диагноз и назначенное лечение».

Четвёртым процессом является «Заполнение медицинской карты». Входные данные – «Медицинская карта» и «Результаты осмотра», выходные – мы получаем «Результаты осмотра».

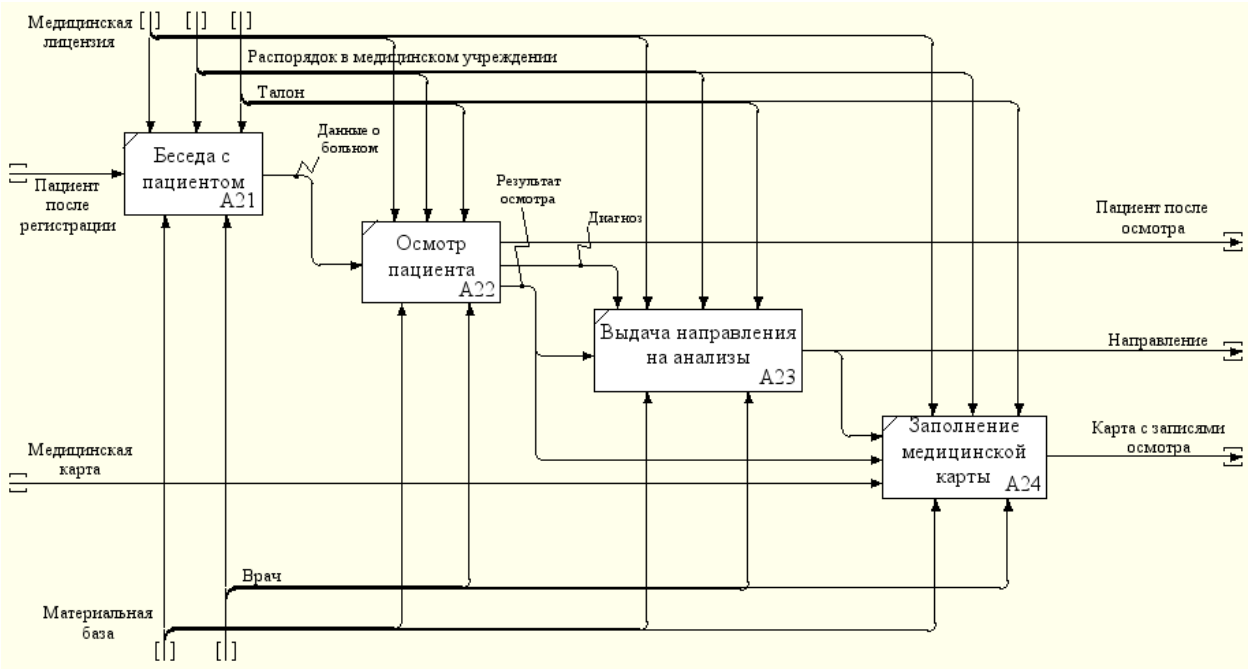


Рисунок 2.4 – Декомпозиция процесса «Осмотр»

На рисунке 2.5 представлена декомпозиция процесса «Лечение». Первым процессом будет «Назначение лечения», входными данными – «Пациент после предварительного осмотра», выходными – «Запись в карте о жалобах».

Второй процесс – «Заполнение медицинской карты», входные данные – «Карта с записями осмотра», выходные – «Запись в карте пациента о результатах лечения» или «Направление на повторное обследование» в случае, если пациент не вылечился.

Третьим процессом является «Выписка». Входными данными является «Карта пациента», а на выходе мы получаем карту в регистратуру. В этих процессах участвует врач.

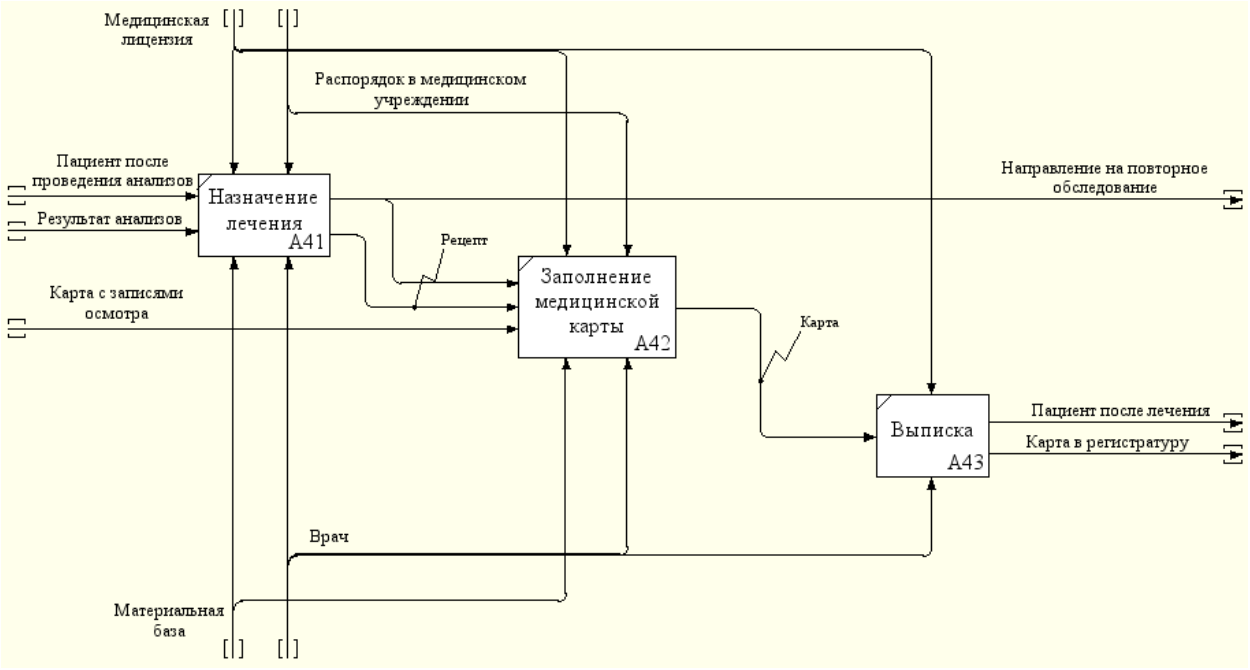


Рисунок 2.5 – Декомпозиция процесса «Лечение»

2.2 Проектирование объекта исследования

Данный раздел содержит описание проектирования модели разрабатываемой ИС, а также описание таблиц и сущностей, которые должны присутствовать в базе данных для корректной работы приложения. База данных должна содержать в себе чёткие, упорядоченные таблицы и данные, которые связаны между собой.

2.2.1 Функциональная структура приложения

Функциональная структура приложения предназначена для того, чтобы наглядно показать, какими возможностями обладает ПО. Глядя на такую структуру, сразу становится ясным, стоит ли использовать это приложение и приведет ли оно к желаемому результату. Функциональная структура приложения представлена на рисунке 2.6.

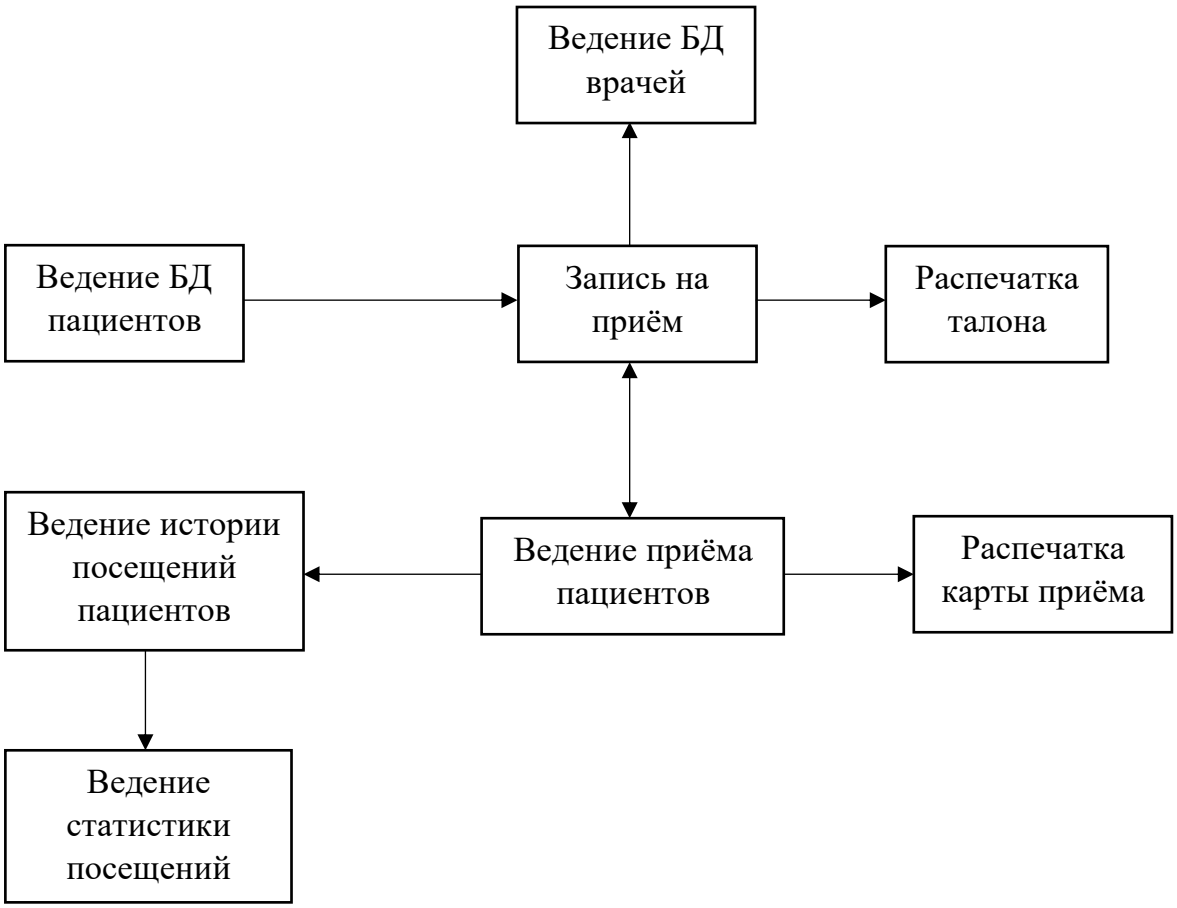


Рисунок 2.6 – Функциональная структура приложения

2.2.2 Выбор системы управления базами данных

Разрабатываемая в дипломном проекте информационная система основана на использовании баз данных (БД). Без них невозможно представить эффективную работу какого-либо учреждения. Поэтому всестороннее овладение системами управления базами данных (СУБД) и их надлежащее применение представляют острую необходимость в проектировании информационных систем. Данный раздел посвящён выбору программных средств проектирования, так как от этого, а также от правильного составления схемы базы данных и запросов к данным напрямую зависят эффективность и производительность конечной ИС.

Система управления базами данных – это комплекс языковых и программных средств, позволяющий нескольким пользователям создавать, вести и совместно использовать базы данных [19].

Современные СУБД в зависимости от используемой модели данных бывают двух видов: SQL и NoSQL, реляционные и нереляционные. Они различаются применяемыми в них принципами, типами поддерживаемых данных и способами хранения этих данных [20].

Реляционные базы данных отличает хранение данных, представляющих конкретные объекты, в структурированном виде. Например, данные о врачах организации, информация в талоне на приём к врачу, описание содержимого корзины в интернет-магазине, хранящиеся в виде таблиц и сгруппированные по типу конкретного объекта. Такими БД удобно пользоваться в режиме реального времени, помимо этого они обеспечивают целостность и структурированность информации.

Нереляционные базы данных устроены по-другому. В них информация может храниться не в виде взаимосвязанных таблиц, а в виде целостной самостоятельной сущности. В основном такие БД удобны для большого объёма данных, быстродейственны и их легче масштабировать [21].

Предпочтение той или иной системе управления определяется особенностями разрабатываемого продукта. В некоторых проектах удобно использование и SQL, и NoSQL баз данных.

В разработке приложения для данного дипломного проекта можно ограничиться использованием только реляционной базы данных, так как все данные, которые будут задействованы в приложении, можно представить в виде взаимосвязанных таблиц. К тому же, после внедрения ПО, в структуру проектируемой БД изменений вносить не планируется (тем не менее, если потребуется, данный процесс не представит трудностей, так как данные структурированы и подвержены «безболезненному» изменению).

На данный момент большой выбор реляционных СУБД позволяет выбрать наиболее подходящую. К самым распространённым относятся Microsoft SQL Server, Oracle, PostgreSQL, MySQL, SQLite, Microsoft Access, FoxPro и др.

По способу доступа к базам данных СУБД делятся на:

- клиент-серверные;
- файл-серверные;
- встраиваемые [22].

В клиент-серверных СУБД (Microsoft SQL Server, Oracle, PostgreSQL, MySQL) данные хранятся и обрабатываются в определённом месте – на сервере, куда приложения-клиенты посылают запросы и получают ответы (без доступа к файлам данных).

В файл-серверных СУБД (Microsoft Access, FoxPro) приложения имеют доступ ко всем файлам БД, которые хранятся в файловом хранилище, и самостоятельно обрабатывают информацию. Сейчас файл-серверные СУБД практически не используются, так как уступают клиент-серверным во многих преимуществах.

Встраиваемые СУБД (SQLite) входят в состав готового ПО без необходимости дополнительной установки и позволяют локально хранить данные без возможности общего использования в сети. Такая встраиваемая СУБД как SQLite широко применяется во многих приложениях для ОС Android.

Для приложения настоящего дипломного проекта наиболее удобным будет использование клиент-серверной СУБД, так как хранимыми в ней данными будут пользоваться и их изменять одновременно несколько пользователей.

В связи с тем, что (как было решено ранее) приложение будет разрабатываться в интегрированной среде Microsoft Visual Studio, использование СУБД Microsoft SQL Server является самым оптимальным программным решением. Оба продукта представлены одним разработчиком – Microsoft Corporation, что обусловливает их безоговорочную совместимость. Кроме того, Microsoft SQL Server – это одна из самых популярных СУБД на сегодняшний день, которая заслужила симпатии программистов по всему миру. СУБД обладает полным комплексом инструментальных средств, позволяющих проектировать комплексные ИС. Для неё создана подробная документация от разработчиков [23]. Используемый язык запросов – Transact-SQL, также поддерживается язык XML и протокол HTTP. Платформа анализа данных MS SQL Server интегрирована с MS Office и позволяет открыть доступ к необходимой информации с помощью интерфейса MS Word и MS Excel [24]. Помимо этого, здесь реализован ряд важных современных функций для работы с базами данных: внешние ключи, сложные запросы, триггеры и т. п.

Для удобства работы с базой данных будет использована утилита Microsoft SQL Server Management Studio, которая предлагает текстовый редактор с цветной подсветкой, возможность представления данных в виде текста, таблиц, диаграмм [25]. Интерфейс программы представлен на рисунке 2.7

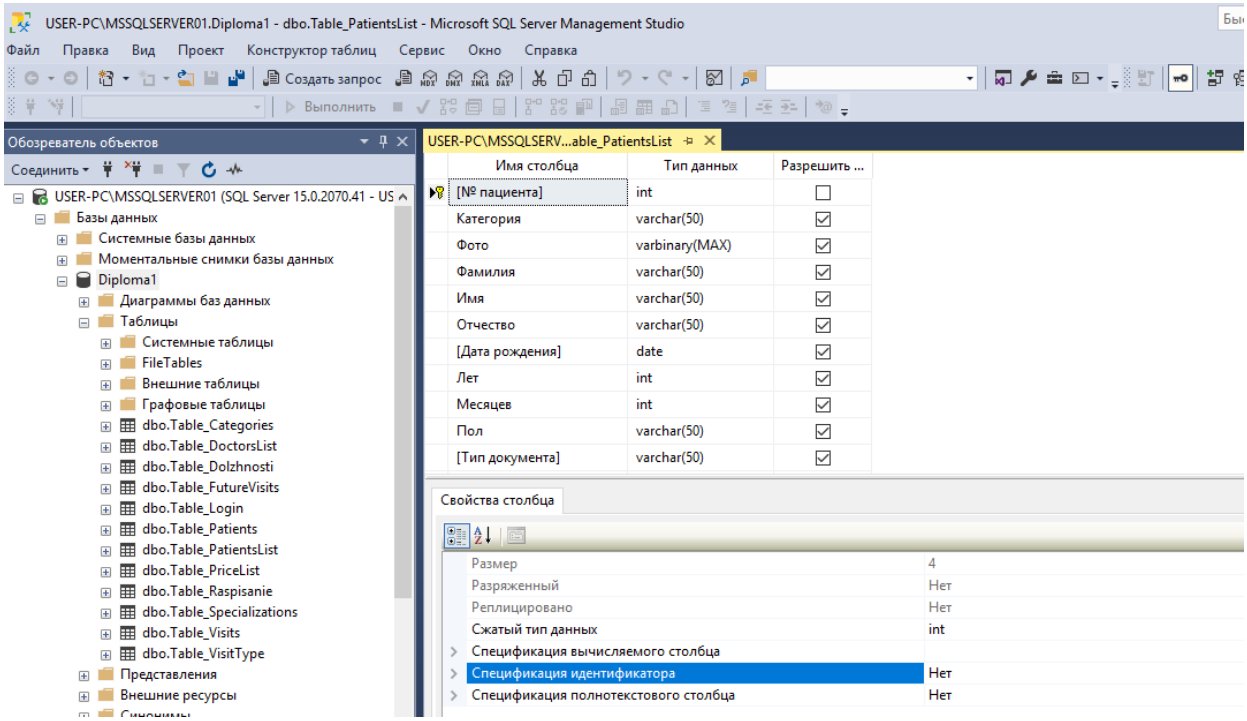


Рисунок 2.7 – Интерфейс Microsoft SQL Server Management Studio 2018

2.2.3 Проектирование базы данных

В теории баз данных существуют такие понятия, как сущность, атрибут, запись, ключ (первичный, вторичный), отношение.

Сущность представляет собой объект в БД, в котором хранятся данные.

Атрибут – свойство, которое описывает сущность. Им является столбец.

Запись – строка таблицы.

Первичный ключ – набор атрибутов, однозначно определяющий запись.

Внешний ключ – набор атрибутов, ссылающихся на ключ другой сущности [26].

В данном разделе будут представлены таблицы данных, используемые в приложении, с описанием типов данных атрибутов сущностей, указанием первичных ключей (первая строка таблицы).

База данных разрабатываемого приложения содержит данные пользователей программы, пациентов и врачей клиники, данные о посещениях пациентов и записях к врачу. Всего БД приложения содержит 11 таблиц.

Типы данных, которые будут использоваться в БД дипломного проекта:

- int – целое число;
- varchar – строка;
- nvarchar – строка в кодировке Unicode;
- varbinary – бинарные данные;
- date – дата;
- datetime – дата и время;
- money – денежная величина.

Для входа в приложение требуется авторизация пользователя. Для этого пользователю необходимо иметь собственные логин и пароль, заданные администратором БД (IT-специалистом клиники). В качестве логина выступает набор символов на латинице, состоящий из имени и фамилии сотрудника, разделённых точкой. Пароль – какой-либо случайный набор символов.

Существует 2 типа пользователей: оператор и врач. Оператор может внести пациента в базу данных, при этом и оператор, и врач могут изменять его данные или удалить его из БД. Оператор может записать пациента на приём к любому врачу, в то время как сам врач может записать пациента только к себе. В данном случае сущностью выступает Пользователь, атрибутами сущности – ID пользователя, Логин, Пароль, Имя, ФИО, Специализация, тип данных всех атрибутов – varchar (символьные данные). Таблица Пользователи показана на рисунке 2.8.

Table_Login

	Имя столбца	Тип данных
🔑	LoginID	int
	UserName	varchar(50)
	Password	varchar(50)
	FirstName	varchar(50)
	FIO	varchar(150)
	Specialization	varchar(50)

Рисунок 2.8 – Таблица Пользователи

После входа в приложение пользователь может просмотреть данные о врачах. Структура таблицы Врачи представлена на рисунке 2.9.

Table_DoctorsList

	Имя столбца	Тип данных
🔑	ВрачID	int
	ФИО	varchar(200)
	ФИОкратко	varchar(50)
	ДолжностьID	int
	СпециализацияID	int
	КатегорияID	int

Рисунок 2.9 – Таблица Врачи

Таблица Врачи связана с таблицами Должность врача, Категория врача и Специализация врача. Данные таблицы показаны на рисунках 2.10, 2.11 и 2.12, соответственно.

Table_Dolzhnosti

	Имя столбца	Тип данных
🔑	ДолжностьID	int
	Название	varchar(50)

Рисунок 2.10 – Таблица Должность врача

Table_Categories

	Имя столбца	Тип данных
🔑	КатегорияID	int
	Название	varchar(50)

Рисунок 2.11 – Таблица Категория врача

Table_Specializations

	Имя столбца	Тип данных
🔑	СпециализацияID	int
	Название	varchar(150)

Рисунок 2.12 – Таблица Специализация врача

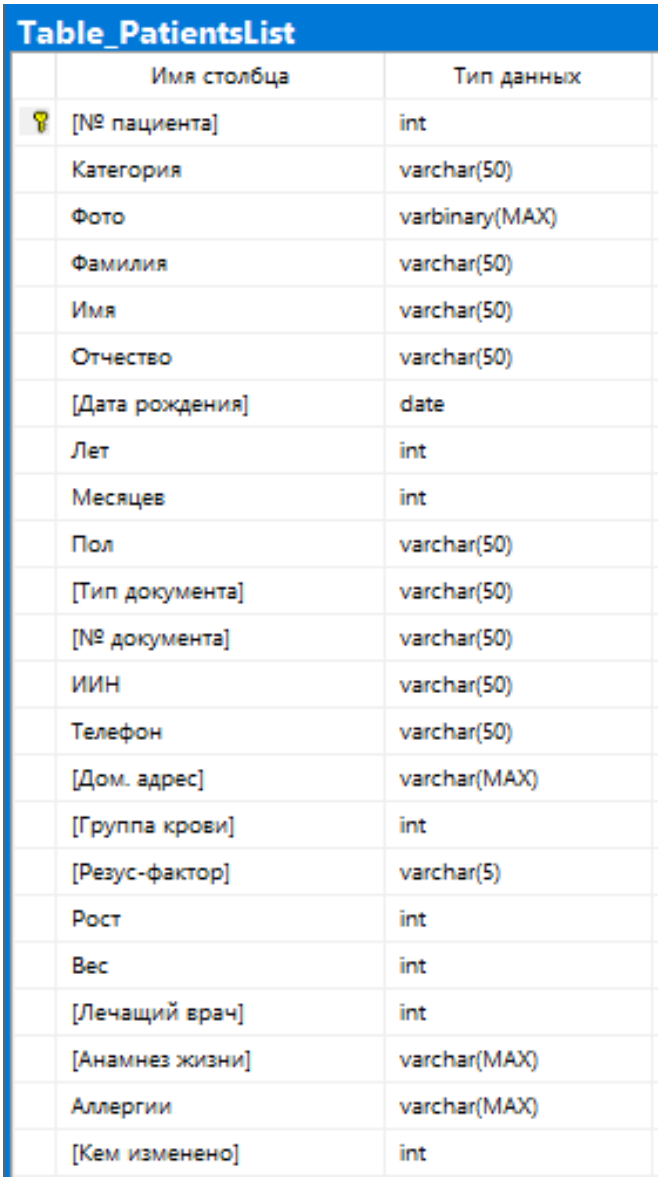
В таблице Прайслист (рисунок 2.13) показаны цены за один приём каждого специалиста. Цена формируется в зависимости от занимаемой должности врача (главный врач, заместитель главного врача, заведующий отделением) и его категории (высшая, первая, вторая).

Table_PriceList

	Имя столбца	Тип данных
🔑	ЦенаID	int
	ДолжностьID	int
	КатегорияID	int
	Цена	money

Рисунок 2.13 – Таблица Прайслист

Пользователь может добавить пациента в базу данных, изменить данные о нём или же удалить пациента из БД. При этом информация о том, кто внёс пациента в БД, фиксируются. Таблица Пациенты представлена на рисунке 2.14.



Table_PatientsList

	Имя столбца	Тип данных
🔑	[№ пациента]	int
	Категория	varchar(50)
	Фото	varbinary(MAX)
	Фамилия	varchar(50)
	Имя	varchar(50)
	Отчество	varchar(50)
	[Дата рождения]	date
	Лет	int
	Месяцев	int
	Пол	varchar(50)
	[Тип документа]	varchar(50)
	[№ документа]	varchar(50)
	ИИН	varchar(50)
	Телефон	varchar(50)
	[Дом. адрес]	varchar(MAX)
	[Группа крови]	int
	[Резус-фактор]	varchar(5)
	Рост	int
	Вес	int
	[Лечащий врач]	int
	[Анамнез жизни]	varchar(MAX)
	Аллергии	varchar(MAX)
	[Кем изменено]	int

Рисунок 2.14 – Таблица Пациенты

Таблица Посещения (рисунок 2.15) хранит данные о посещениях пациента. Здесь помимо введённых врачом данных о состоянии пациента и назначенном лечении фиксируются дата приёма, какой врач принял пациента и дата следующего осмотра (по умолчанию задаётся через неделю от текущей даты, но может быть изменена по необходимости).

Table_Visits

	Имя столбца	Тип данных
🔑	ПриемID	int
	LoginID	int
	ВрачID	int
	ПациентID	int
	ДатаПриема	datetime
	Жалобы	nvarchar(150)
	ДанныеОИ	nvarchar(150)
	Диагноз	nvarchar(50)
	Лечение	nvarchar(150)
	ПланОбследования	nvarchar(150)
	СледующийПрием	datetime
	СтоимостьПриема	money

Рисунок 2.15 – Таблица Посещения

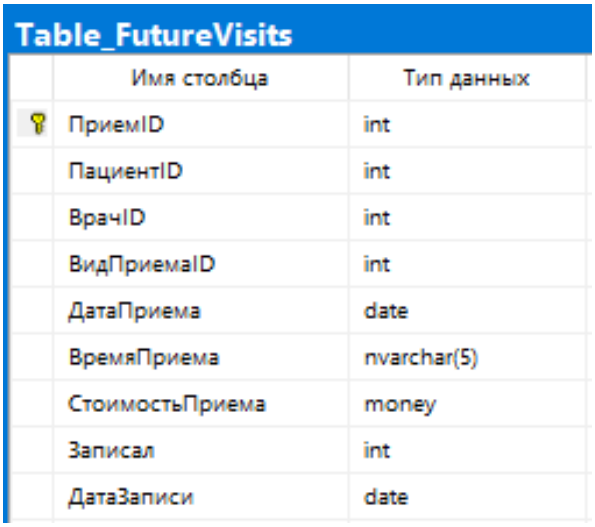
Таблица Расписание (рисунок 2.16) хранит информацию о том, в какие дни и в какое время принимает врач в зависимости от его должности. Главный врач и зам. глав. врача принимают только по понедельникам с 12:00 до 17:00, заведующие отделений – во вторник и среду так же с 12:00 до 17:00, остальные врачи принимают с понедельника по пятницу с 8:00 до 19:00, в субботу – с 9:00 до 14:00.

Table_Raspisanie

	Имя столбца	Тип данных
🔑	РасписаниеID	int
	ДолжностьID	int
	ДеньНедели	nvarchar(10)
	ВремяПриема	nvarchar(50)

Рисунок 2.16 – Таблица Расписание

Для реализации записи на приём существует таблица Будущие визиты (рисунок 2.17). В ней фиксируется пациент, которого необходимо записать на приём, врач, к которому записывается пациент, дата и время приёма, кто и когда (дата) записал пациента на приём, а также вид приёма (первичный или повторный) и его стоимость.



Table_FutureVisits

	Имя столбца	Тип данных
🔑	ПриемID	int
	ПациентID	int
	ВрачID	int
	ВидПриемаID	int
	ДатаПриема	date
	ВремяПриема	nvarchar(5)
	СтоимостьПриема	money
	Записал	int
	ДатаЗаписи	date

Рисунок 2.17 – Таблица Будущие визиты

Отношение в БД – это когда одна сущность ссылается на первичный ключ другой сущности. Существует три типа отношений:

- один-к-одному – каждой записи первой сущности соответствует только одна запись из второй сущности (например, у каждого пациента есть только один уникальный номер в клинике);

- один-ко-многим – каждой записи первой сущности могут соответствовать несколько записей из второй сущности (например, у одного врача может быть несколько пациентов);

- многие-ко-многим – каждой записи первой сущности могут соответствовать несколько записей из второй сущности и каждой записи второй сущности могут соответствовать несколько записей из первой сущности (например, пациент может прийти на приём к разным врачам, врач может принять разных пациентов).

Отношения между таблицами базы данных дипломного проекта показаны на рисунке 2.18.

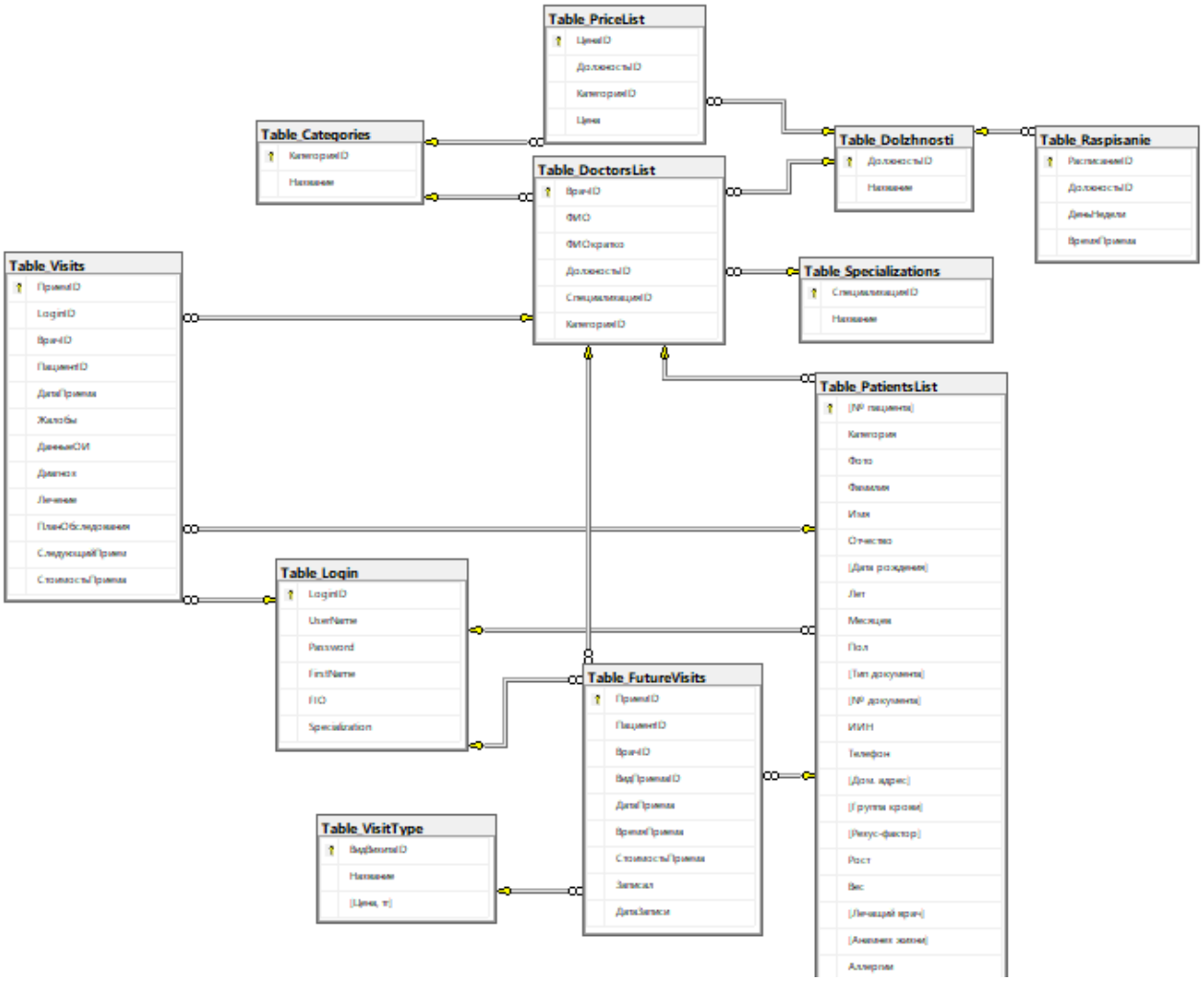


Рисунок 2.18 – Отношения между таблицами БД

2.3 Программная реализация проекта

В данной главе будет описан процесс разработки ПО. В первую очередь необходимо разработать, или, вернее, спроектировать, серверную часть проекта. Данный процесс начинается с создания базы данных Diploma1 в утилите Microsoft SQL Server Management Studio, как показано на рисунке 2.19.

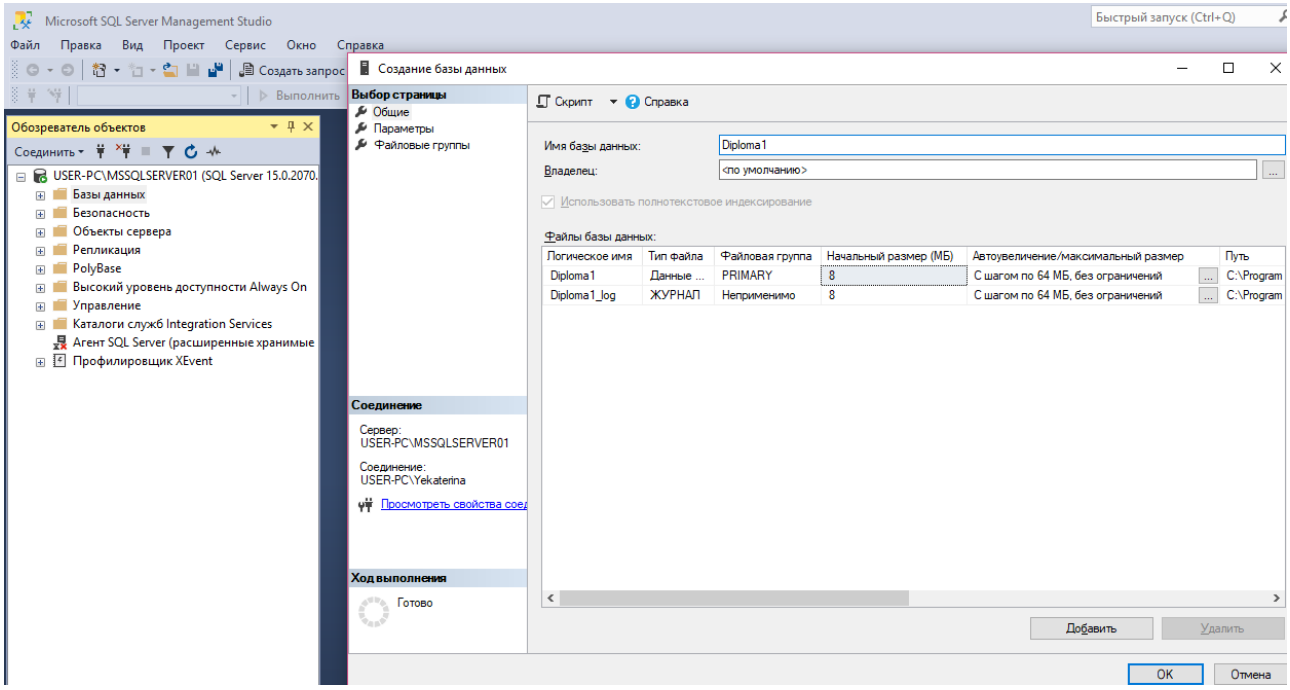


Рисунок 2.19 – Создание БД «Diploma1» в MS SQL Server Management Studio

После создания базы данных необходимо сформировать в ней таблицы и отношения (рисунок 2.20). На этом разработка серверной части завершена.

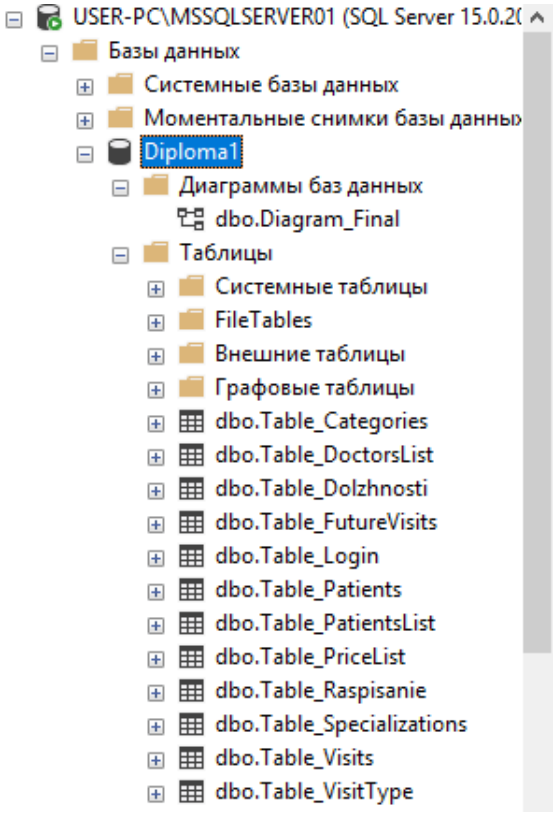


Рисунок 2.20 – Таблицы и диаграммы базы данных «Diploma1»

Далее необходимо приступить непосредственно к разработке клиентской части программы. Для этого необходимо создать проект – Приложение Windows Forms – в Microsoft Visual Studio (рисунок 2.21), которое будет включать в себя все необходимые средства для разработки.

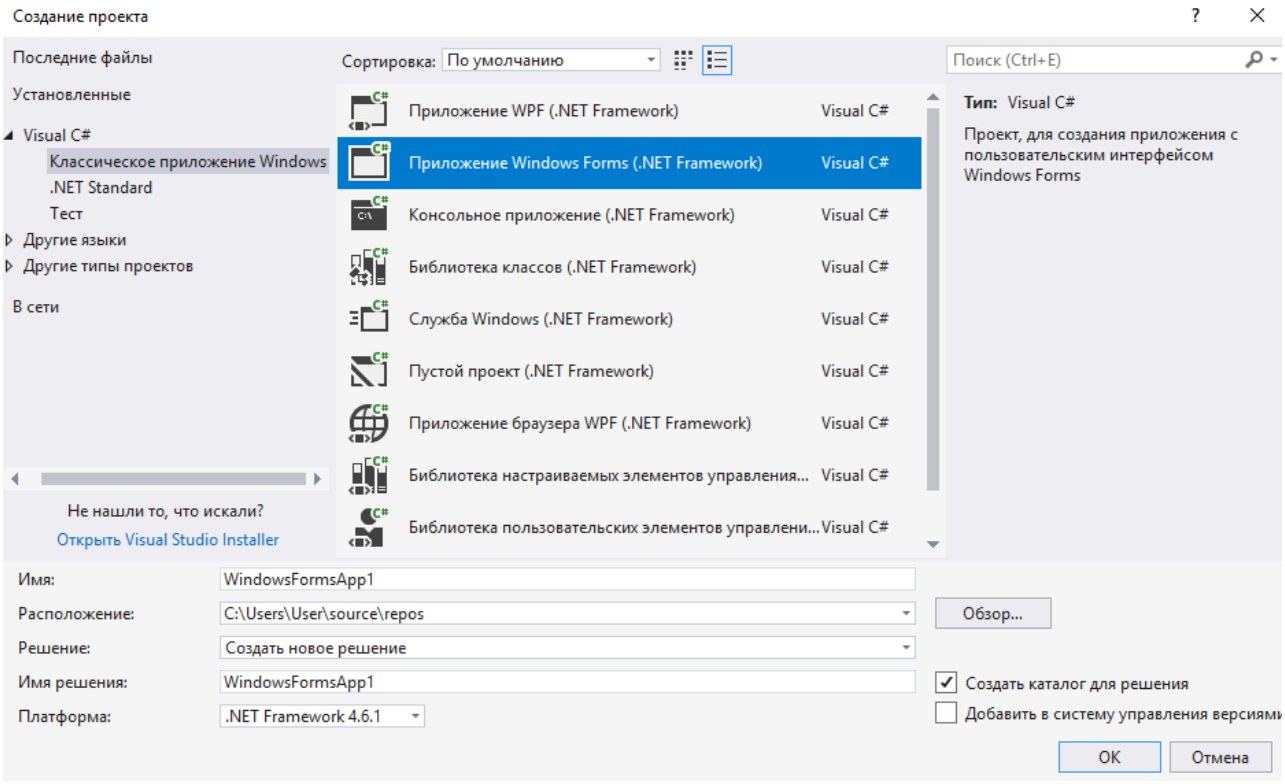


Рисунок 2.21 – Создание Приложения Windows Forms

Затем необходимо подключить приложение к серверу БД, т.е. создать строку подключения к БД (рисунок 2.22) в программном файле App.config.

```
connectionString="Data Source=172.20.10.2,1433;Initial Catalog=Diploma1;User ID=UniservMC1;Password=sa*QWf70;"
```

Рисунок 2.22 – Строка подключения к БД

После этого можно приступить к созданию и разработке дизайна форм приложения. Все используемые в приложении формы представлены на рисунке 2.23.

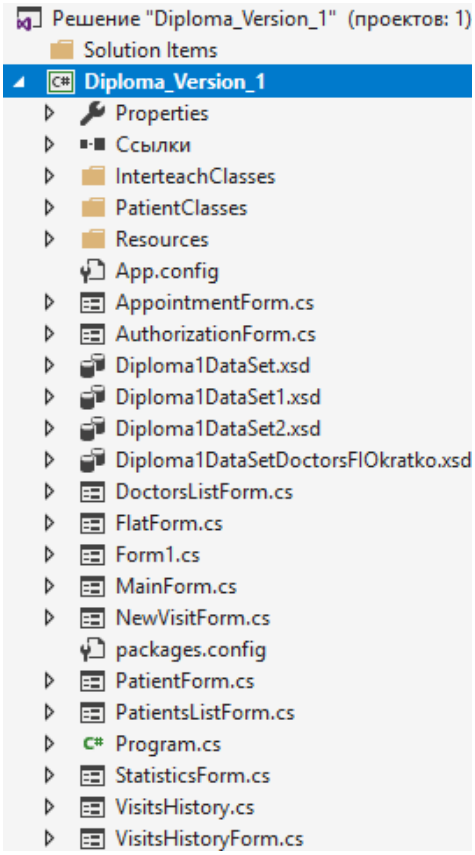


Рисунок 2.23 – Формы приложения

Для дизайна кнопок основной формы (рисунок 3.2) используется библиотека иконок FontAwesome.Sharp, которая является бесплатным open-source проектом. Для добавления этой библиотеки необходимо левой кнопкой мыши нажать на Ссылки в Обозревателе решений и выбрать Управление пакетами NuGet… NuGet – система управления пакетами для платформ разработки Microsoft. Найдём нужный пакет и установим его, как показано на рисунке 2.24.

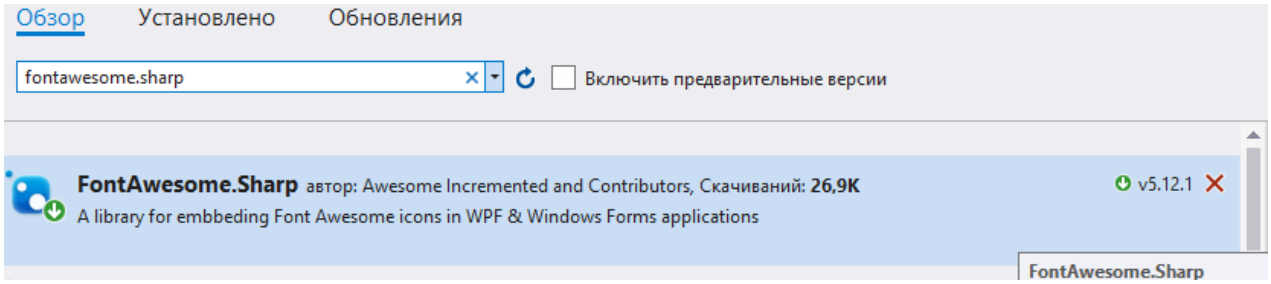


Рисунок 2.24 – Установка пакета иконок FontAwesome.Sharp

С созданием форм приступаем непосредственно к программированию. Для ввода, удаления и изменения данных пациента в БД был создан класс PatientsClass. Фрагмент кода этого класса с SQL-запросом показан на рисунке 2.25.

```
public bool Insert(PatientsClass patientsClass)
{
    bool isSuccess = false;
    try
    {
        string sqlInsert = "INSERT INTO Table_PatientsList ([№ пациента], Категория, Фото,
        SqlCommand cmd = new SqlCommand(sqlInsert, connection);

        //Create Parameters to Add Data
        cmd.Parameters.AddWithValue("@PatientID", patientsClass.PatientID);
        cmd.Parameters.AddWithValue("@PatientType", patientsClass.PatientType);
        cmd.Parameters.AddWithValue("@Photo", patientsClass.Photo);
        cmd.Parameters.AddWithValue("@Surname", patientsClass.Surname);
        cmd.Parameters.AddWithValue("@Name", patientsClass.Name);
        cmd.Parameters.AddWithValue("@Otchestvo", patientsClass.Otchestvo);
        cmd.Parameters.AddWithValue("@BirthDate", patientsClass.BirthDate);
        cmd.Parameters.AddWithValue("@Years", patientsClass.Years);
        cmd.Parameters.AddWithValue("@Months", patientsClass.Months);
        cmd.Parameters.AddWithValue("@Pol", patientsClass.Pol);
        cmd.Parameters.AddWithValue("@DocumentType", patientsClass.DocumentType);
```

Рисунок 2.25 – Фрагмент кода класса PatientsClass

2.3.1 Форма Авторизация (AuthorizationForm)

Для авторизации и входа в приложение необходимо проверить, совпадают ли данные, которые ввёл пользователь, с данными из БД. Для этого необходимо открыть подключение к БД и выполнить SQL-запрос, показанный на рисунке 2.26. При успешной авторизации открывается главное окно приложения, а введённый логин пользователя в неё передаётся.

```
SqlConnection connection = new SqlConnection(myConnectionString);
string sql = "SELECT * FROM Table_Login WHERE UserName = '"+txtBoxUserName.Text.Trim()+"' AND Password = '"+txtBoxPassword.Text.Trim() + "
SqlDataAdapter sqlDataAdapter = new SqlDataAdapter(sql, connection);
DataTable dataTable = new DataTable();
sqlDataAdapter.Fill(dataTable);
if (dataTable.Rows.Count == 1)
{
    FlatForm flatForm = new FlatForm(this.txtBoxUserName.Text.ToString());
    this.Hide();
    flatForm.Show();
}
else
{
    MessageBox.Show("Введён неверный логин или пароль. Попробуйте ещё раз");
}
```

Рисунок 2.26 – Фрагмент кода для авторизации в приложении

2.3.2 Форма Главное меню (FlatForm)

При открытии формы FlatForm из БД достаётся данные о пользователе (имя и специализация), которые отображаются на форме во время пользования программой. Также реализовано открытие дочерних форм внутри основной формы-контейнера, а также изменение цвета иконок и изменение их

расположения относительно текста на кнопках. Фрагмент кода показан на рисунке 2.27.

```
public void OpenChildForm(Form childForm)
{
    if (currentChildForm != null)
    {
        currentChildForm.Close();
    }
    currentChildForm = childForm;
    currentChildForm.TopLevel = false;
    childForm.FormBorderStyle = FormBorderStyle.None;
    childForm.Dock = DockStyle.Fill;
    panelDesktop.Controls.Add(childForm);
    panelDesktop.Tag = childForm;
    childForm.BringToFront();
    childForm.Show();
    labelTitleChildForm.Text = childForm.Text;
}
private void iconButtonPatients_Click(object sender, EventArgs e)
{
    ActivateButton(sender, RGBcolors.color1);
    OpenChildForm(new PatientForm(userName));
}
```

Рисунок 2.27 – Фрагмент кода для открытия дочерней формы внутри формы-контейнера

2.3.3 Форма Сменить пароль (ChangePasswordForm)

В данной форме реализована возможность смены пароля от аккаунта пользователя. Открыть эту форму можно из форм Авторизация или Главное меню. Фрагмент кода смены пароля показан на рисунке 2.28.

```
string sqlUpdate = "UPDATE Table_Login SET Password = '" + txtBoxNewPassword.Text.Trim() + "' WHERE [UserN
SqlCommand cmd = new SqlCommand(sqlUpdate, connection);
cmd.Parameters.AddWithValue("@Password", txtBoxNewPassword.Text.Trim());
connection.Open();
int rows = cmd.ExecuteNonQuery();
if (txtBoxOldPassword.Text == txtBoxNewPassword.Text)
{
    MessageBox.Show("Введённые пароли совпадают. Придумайте новый");
}
//If the query runs successfully then the value of rows is greater than 0 else its value is 0
else if (rows > 0)
{
    MessageBox.Show("Пароль успешно изменён!");
    FlatForm flatForm = new FlatForm(this.txtBoxUserName.Text.ToString());
    this.Hide();
    flatForm.Show();
}
else
{
    MessageBox.Show("Введён неверный логин или пароль. Попробуйте ещё раз");
}
connection.Close();
```

Рисунок 2.28 – SQL-запрос UPDATE формы ChangePasswordForm

2.3.4 Форма Пациенты (PatientForm)

В форме PatientForm выполняются ввод, удаление или изменение данных о пациенте в БД с помощью созданного класса PatientsClass, а также выгружаются данные уже внесённых в БД пациентов. Фрагмент кода с SQL-запросом SELECT показан на рисунке 2.29.

```
//Открываем БД и достаём оттуда значения по patientID
string sqlPatient = "SELECT Категория, Фото, Фамилия, Имя, Отчество, [Дата рождения], Лет, Месяцев, Пол, [Тип докум
SqlCommand cmdPatient = new SqlCommand(sqlPatient, connection);
//string sqlDoctor = "SELECT FIOкратко, Specialization FROM Table_Login WHERE UserName = '" + userName + "'";
//SqlCommand cmdDoctor = new SqlCommand(sqlDoctor, connection);
connection.Open();
string patientType = "";
Image patientPhoto = null;
string patientSurname = "";
string patientName = "";
string patientOtchestvo = "";
DateTime patientBirthDate = DateTime.Now;
int years = 0;
int months = 0;
string gender = "";
```

Рисунок 2.29 – Фрагмент кода с SQL-запросом SELECT формы PatientForm

При выделении строки таблицы номер пациента записывается в глобальную переменную patientIDValue при помощи созданного в программном файле Program.cs класса GetPatientId (рисунок 2.30), для того чтобы к нему можно было обращаться из другой формы.

```
static class GetPatientID
{
    public static int patientIDValue { get; set; }
}
```

Рисунок 2.30 – Класс GetPatientId

2.3.5 Форма История посещений (VisitsHistory)

Форма VisitsHistory содержит таблицу с данными о предыдущих посещениях пациента (дата, врач) с возможностью просмотреть данные каждого зафиксированного осмотра. Метод для заполнения таблицы необходимыми данными показан на рисунке 2.31.

```
        private void FilldtGrdViewVisitsHistoryList()
        {
            //DataGrid заполняется данными о приемах
            dtGrdViewVisitsHistoryList.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;
            DataTable dataTable = new DataTable();
            string sqlSelect = @"SELECT
        [ПриемID],
        [Table_DoctorsList].[ФИОкратко] AS Врач,
        ДатаПриема
    FROM [Diploma1].[dbo].[Table_Visits]
    INNER JOIN [Table_DoctorsList] ON [Table_Visits].ВрачID=Table_DoctorsList.ВрачID WHERE [ПациентID]= "+ patientID+";";
            SqlCommand cmd = new SqlCommand(sqlSelect, connection);
            SqlDataAdapter adapter = new SqlDataAdapter(cmd);
            connection.Open();
            adapter.Fill(dataTable);
            connection.Close();
            dtGrdViewVisitsHistoryList.DataSource = dataTable;
            addButton();
        }
```

Рисунок 2.31 – Метод для заполнения таблицы с историей посещений

2.3.6 Форма Врачи (DoctorsListForm)

Форма DoctorsListForm содержит список врачей с возможностью поиска по выбранному критерию. Фрагмент кода для поиска врача представлен на рисунке 2.32. При выборе строки с нужным врачом его идентификатор записывается в глобальную переменную (doctorIDValue) с помощью класса GetDoctorId, чтобы в последующем была возможность записаться к нему на приём.

```
private void txtBoxFindPatient_TextChanged(object sender, EventArgs e)
{
    //Get the value from textbox
    string keyword = txtBoxFindPatient.Text;

    if (cmbBoxFindPatientBy.Text == "фамилии")
    {
        SqlDataAdapter sqlDataAdapter = new SqlDataAdapter("SELECT ВрачID AS [ID врача], ФИО AS Врач, Table_Dolzhnosti.Название AS Должно
        DataTable dt = new DataTable();
        sqlDataAdapter.Fill(dt);
        dtGrdViewDoctorsList.DataSource = dt;
    }
```

Рисунок 2.32 – Фрагмент кода для поиска врача

2.3.7 Форма Приём пациента (NewVisitForm)

В данной форме данные о приёме записываются в БД врачом. SQL-запрос для записи данных в БД показан на рисунке 2.33.

```
private void btnUpdatePatient_Click(object sender, EventArgs e)//добавление приема в таблицу
{
    if (patientSurname == "")
    {
        MessageBox.Show("Для сохранения приема необходимо выбрать пациента");
        return;
    }
    VisitDate = DateTime.Now.ToString();
    string sqlVisit = "insert into Diploma1.dbo.Table_Visits (LoginID,ПациентID,ВрачID,ДатаПриема,Жалобы, ,
    SqlCommand cmdVisit = new SqlCommand(sqlVisit, connection);
    connection.Open();
    cmdVisit.ExecuteNonQuery();
    connection.Close();

    string sqlCheck = "select * from Diploma1.dbo.Table_Visits where  ДатаПриема='" + VisitDate + "';";
    SqlCommand cmdCheck = new SqlCommand(sqlCheck, connection);
    connection.Open();
    using (SqlDataReader reader = cmdCheck.ExecuteReader())
    {
        while (reader.Read())
        {

            if (reader.HasRows == true)
            {
                MessageBox.Show("Данные о приеме добавлены!");
                visitAdded = true;
            }
        }
    }
```

Рисунок 2.33 – SQL-запрос для записи данных о приёме в БД

2.3.8 Форма Запись на приём (AppointmentForm)

Данная форма позволяет осуществить запись пациента на свободное время к выбранному врачу. Для этого используется очень много функций, таких как CheckWorkHours() – для проверки работы врача в выбранный день, CheckTime() – для проверки в какое время работает врач, GeneratTimeForBox() – добавления доступных часов в combobox, DateTimeActions() – отображения работает ли врач в выбранный день, FillDayOfWeekCodeDictionary () – для заполнения словаря, а также класс Dictionary – словарь для хранения пар "Дата приема – время приема" и хранения дней недели и их номеров. Фрагмент кода для метода CheckWorkHours() показан на рисунке 2.34.

```
//Проверка, работает ли врач в выбраный день
private bool CheckWorkHours(int ChDate)
{
    List<int> newDatesRules = new List<int>();
    if (doctorWorkDays == "")
    {
        VisitDateInfo.Text = "Не выбран врач";
        visitDatePicker.Enabled = false;
        return false;
    }
    else
        visitDatePicker.Enabled = true;

    if (!doctorWorkDays.Contains("-") && !doctorWorkDays.Contains("/"))
    {
        newDatesRules.Add(Convert.ToInt32(doctorWorkDays));
        if (ChDate == newDatesRules[0])
            return true;
    }

    if (doctorWorkDays.Contains("-"))
    {
        newDatesRules.Add(Convert.ToInt32(doctorWorkDays.Split('-')[0]));
        newDatesRules.Add(Convert.ToInt32(doctorWorkDays.Split('-')[1].Split('/')[0]));

        if (ChDate >= newDatesRules[0] && ChDate <= newDatesRules[1])
            return true;

    }
```

Рисунок 2.34 – Фрагмент кода для поиска врача

2.3.9 Форма Статистика (StatisticsForm)

Для подсчёта количества посещений по врачу, специализации и месяцу создано три метода: ShowDoctorStats(), ShowDoctorStats() и ShowMonthStats(), соответственно. Метод ShowDoctorStats() продемонстрирован на рисунке 2.35.

```
//метод  нахождения статистики для выбранного врача
private void ShowDoctorStats()
{

    string sqlShowDoctorStats = @"SELECT count(*)  Количество
    from Table_Visits
    INNER JOIN Table_DoctorsList ON Table_Visits.ВрачID = Table_DoctorsList.ВрачID where ФИОкратко ='" + cmbBoxFindStatsByOptions.SelectedIte
    SqlCommand cmdShowDoctorStats = new SqlCommand(sqlShowDoctorStats, connection);
    connection.Open();
    using (SqlDataReader reader = cmdShowDoctorStats.ExecuteReader())
    {
        while (reader.Read())
        {
            number = Convert.ToInt32(reader["Количество"]);
        }
    }
    connection.Close();
    showStatsLabel.Text = "У врача " + cmbBoxFindStatsByOptions.SelectedItem.ToString() + " " + number + " "+ending(number);

}
```

Рисунок 2.35 – Метод ShowDoctorStats()

3 Экспериментальная часть

3.1 Назначение и условия выполнения программы

Потенциальными пользователями программы являются сотрудники регистратуры (операторы) и врачи клиники «Uniserv Medical Center». Сервер базы данных программы находится на серверном ПК организации.

Программа предназначена для ОС Windows.

3.2 Руководство к эксплуатации программы

В данном разделе будут даны инструкции для использования программы.

После запуска программы открывается окно авторизации (рисунок 3.1). Логин и пароль выдаются пользователю IT-специалистом организации, который управляет ими на сервере программы. Если логин или пароль введены неверно, появится окошко с предупреждением.

Рисунок 3.1 – Окно авторизации

При желании пользователь может сменить пароль от своего аккаунта, нажав соответствующую кнопку на форме Авторизация или на Главной форме. Окно смены пароля представлено на рисунке 3.2.

Рисунок 3.2 – Окно смены пароля

При успешной авторизации (верно введённых логине и пароле) открывается Главная форма приложения, на которой отображаются: логотип клиники, текущие время и дата, кнопки для открытия других форм, выхода из приложения, аккаунта и смены пароля, а также имя и специализация пользователя, вошедшего в программу (рисунок 3.3).

Для пользователя «оператор» отсутствует кнопка «Приём пациента», позволяющая врачам фиксировать осмотр в истории болезни пациента.

Рисунок 3.3 – Главная форма

В форме Пациенты (рисунок 3.4) уже добавленные в БД пациенты отображаются в таблице. Можно осуществить поиск пациента по определённому критерию, добавить или изменить данные пациента. Если пользователь является оператором, он может внести карту пациента в базу данных, заполнив все поля формы и нажав кнопку «Добавить пациента». С помощью кнопки «Обновить информацию о пациенте» оператор или врач могут внести изменения в его карту. Кнопка «Очистить поля» позволяет стереть заполненные поля формы. При выделении строки, соответствующей нужному пациенту, карта заполняется его данными, которые при необходимости можно изменить. Номер пациента является уникальным для каждого пациента, поэтому добавить пациента с уже существующим номером не получится: программа выдаст диалоговое окно с предупреждением. Возраст пациента в месяцах и годах высчитывается автоматически на основе введённой даты рождения.

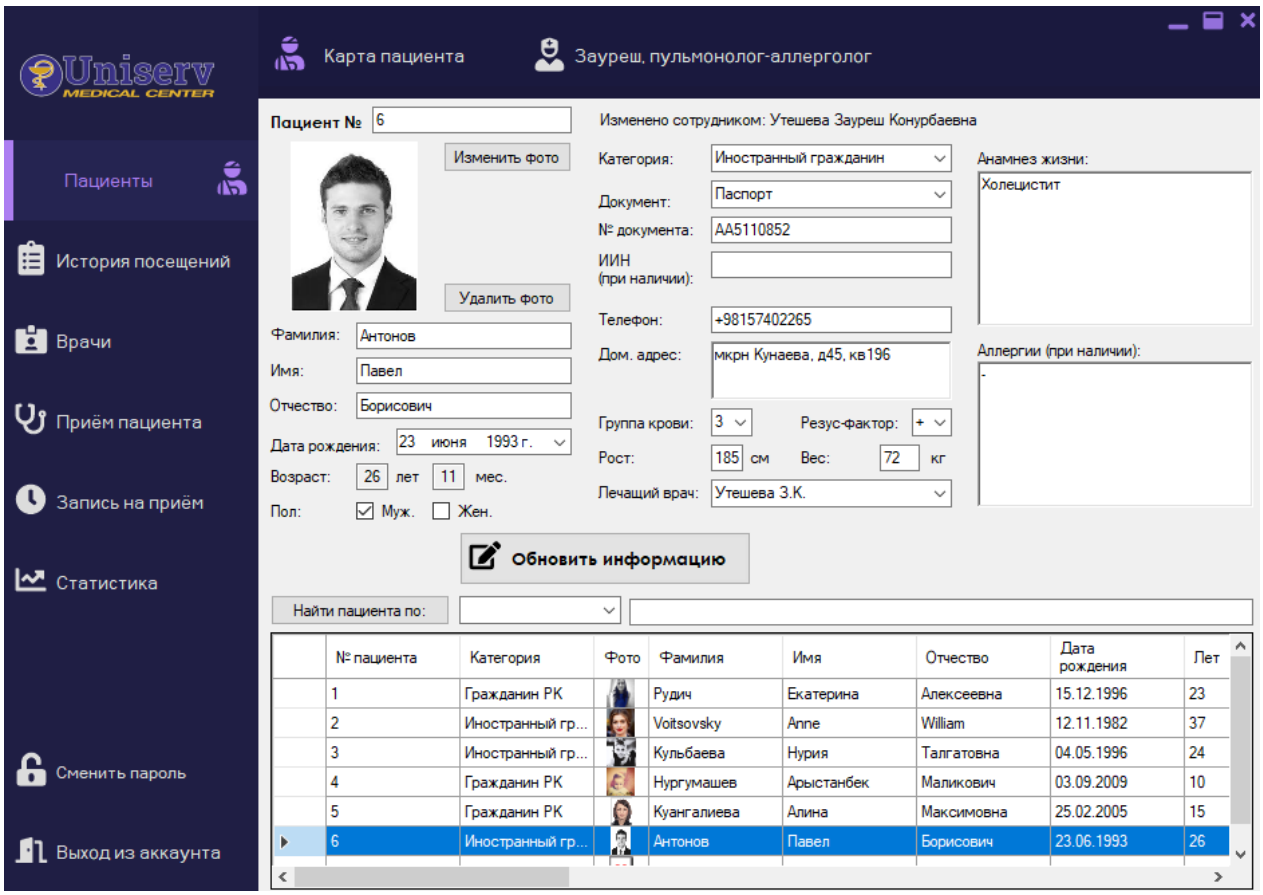


Рисунок 3.4 – Картотека пациентов

Форма История посещений (рисунок 3.5) позволяет просмотреть информацию о предыдущих посещениях пациента (дата, врач). В целях безопасности данных удалять или изменять данные о предыдущих осмотрах нельзя.

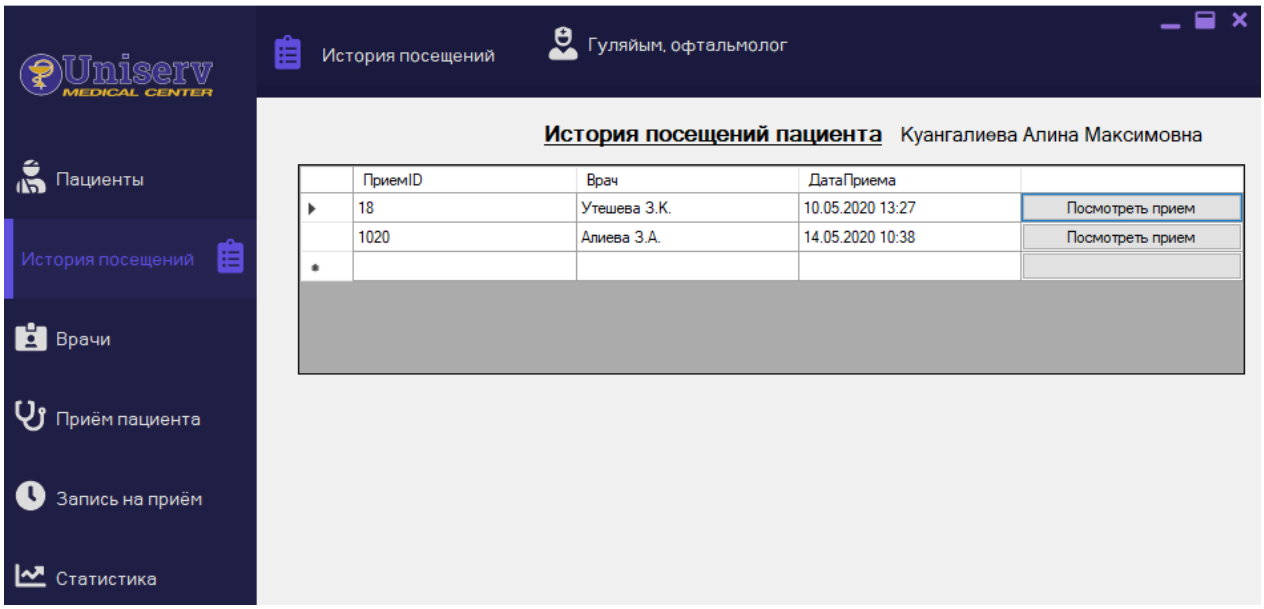


Рисунок 3.5 – Окно истории посещений пациента

При необходимости с помощью кнопки Посмотреть приём можно просмотреть данные предыдущего осмотра (рисунок 3.6), а также распечатать выписку. Вносить изменения нельзя.

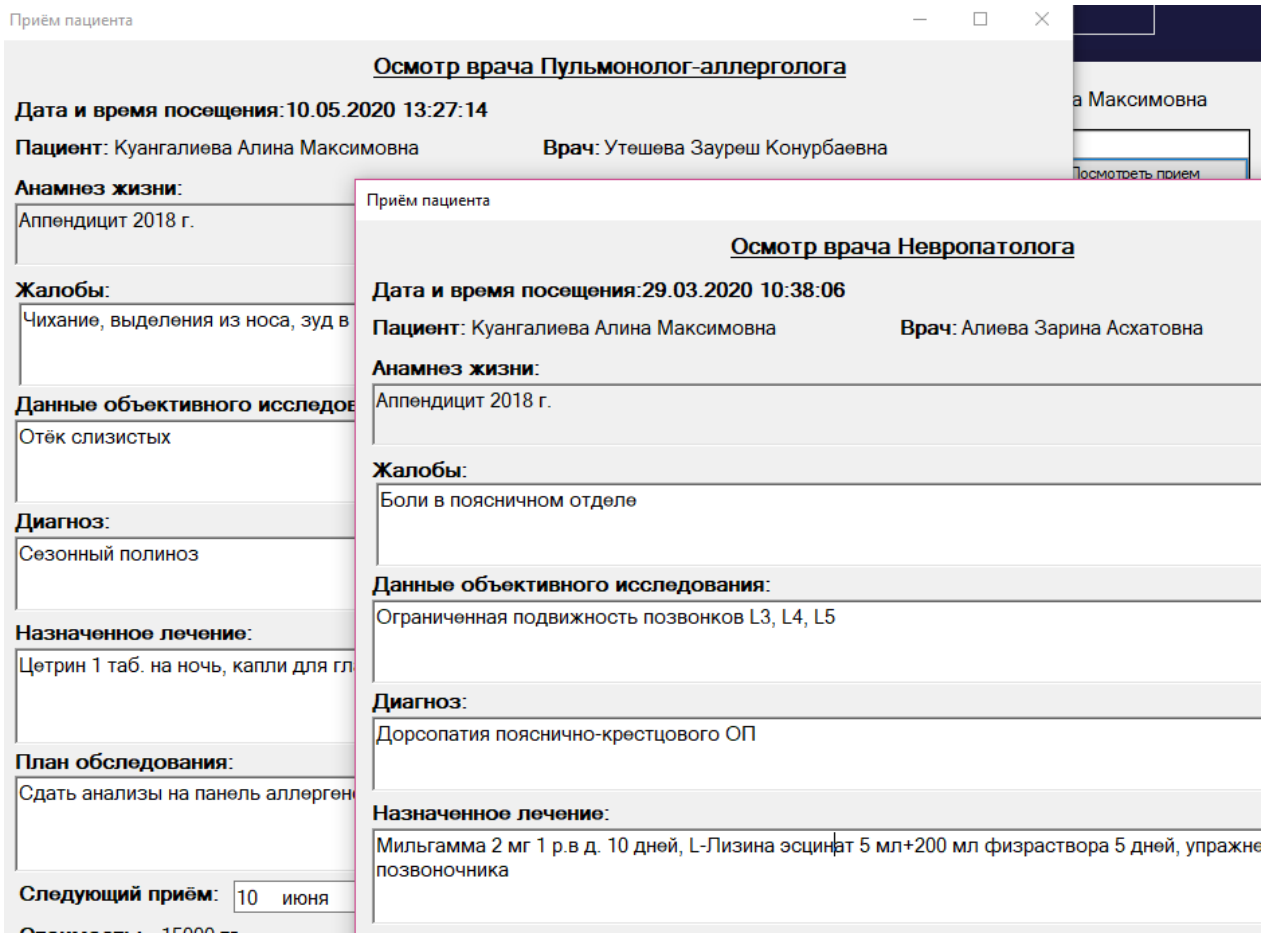


Рисунок 3.6 – Просмотр предыдущих приёмов пациента

Во вкладке Приём пациента (рисунок 3.7) врач фиксирует данные об осмотре пациента и распечатывает выписку. Автоматически отображается стоимость приёма врача и дата следующего приёма (через неделю от текущей даты). Дату при необходимости можно изменить.

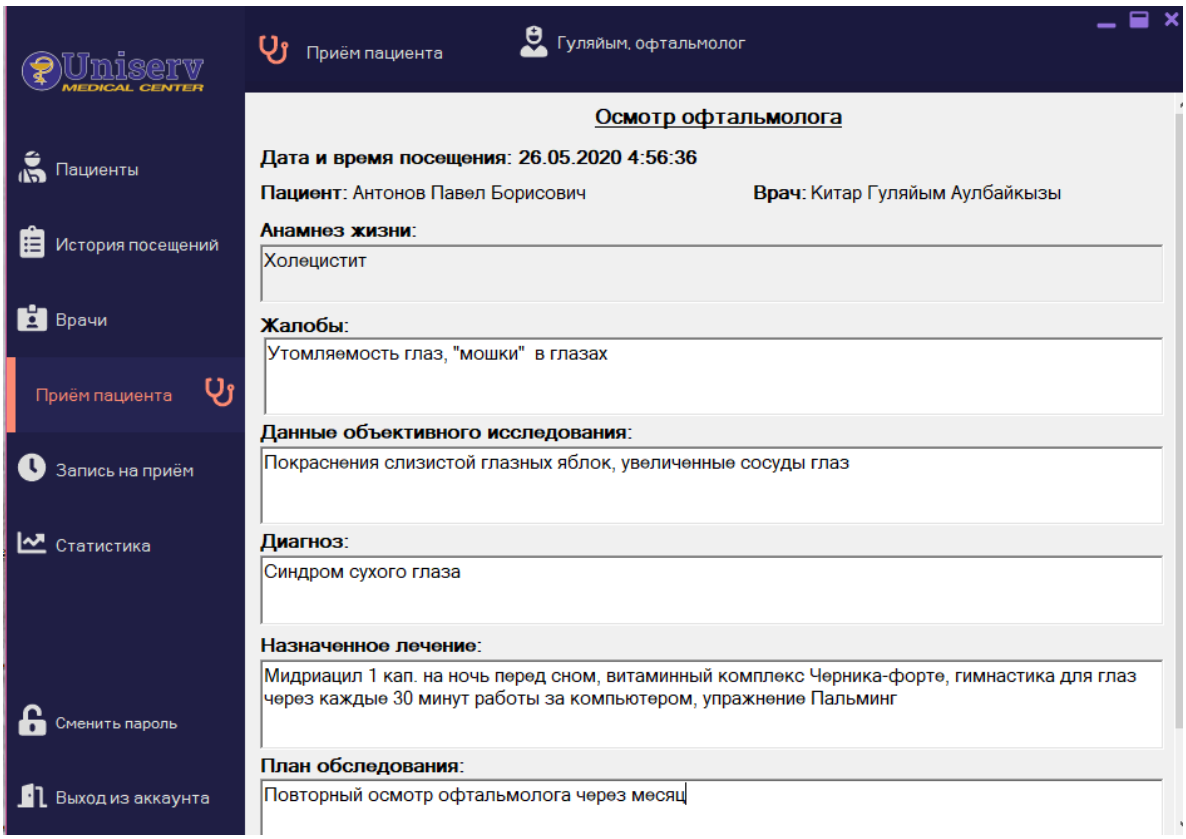
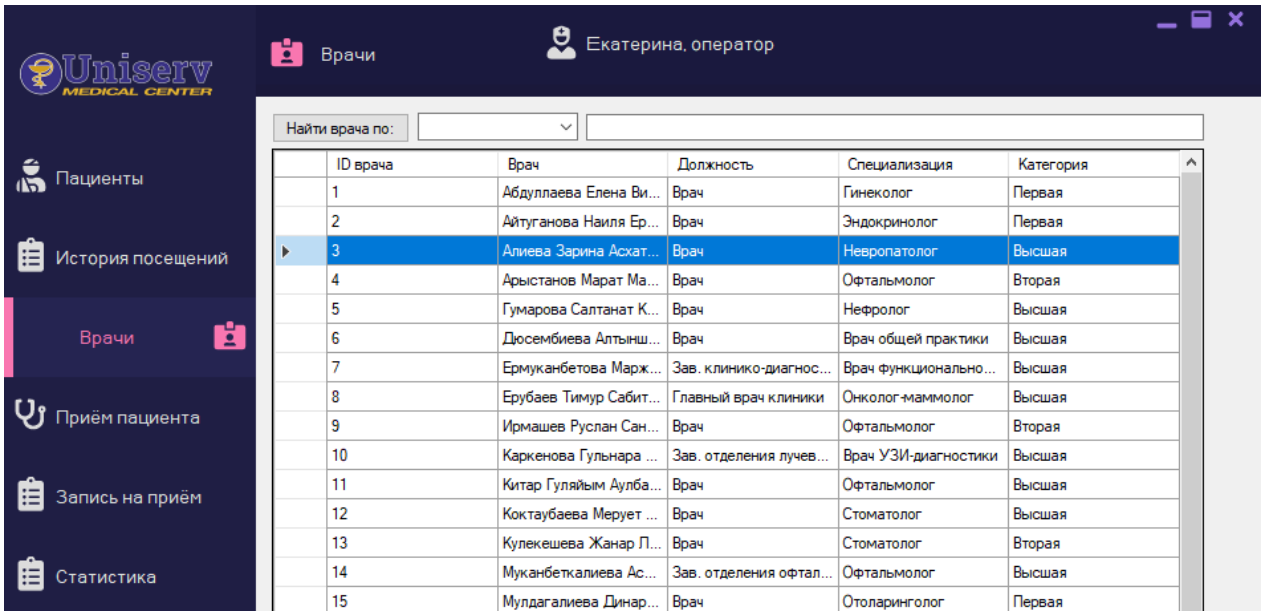


Рисунок 3.7 – Внесение информации об осмотре пациента

Форма Врачи (рисунок 3.8) отображает список врачей, работающих в клинике, а также данные о занимаемой должности, специализации врача и его категории. При необходимости можно осуществить поиск врача по выбранному параметру (фамилии, должности, специализации и категории).



ID врача	Врач	Должность	Специализация	Категория
1	Абдуллаева Елена Ви...	Врач	Гинеколог	Первая
2	Айтуганова Наиля Ер...	Врач	Эндокринолог	Первая
3	Алиева Зарина Асхат...	Врач	Невропатолог	Высшая
4	Арыстанов Марат Ма...	Врач	Офтальмолог	Вторая
5	Гумарова Салтанат К...	Врач	Нефролог	Высшая
6	Дюсембиева Алтынш...	Врач	Врач общей практики	Высшая
7	Ермуканбетова Марж...	Зав. клинико-диагнос...	Врач функционально...	Высшая
8	Ерубаев Тимур Сабит...	Главный врач клиники	Онколог-маммолог	Высшая
9	Ирмашев Руслан Сан...	Врач	Офтальмолог	Вторая
10	Каркенова Гульнара ...	Зав. отделения лучев...	Врач УЗИ-диагностики	Высшая
11	Китар Гуляйым Аулба...	Врач	Офтальмолог	Высшая
12	Коктаубаева Мерует ...	Врач	Стоматолог	Высшая
13	Кулекешева Жанар Л...	Врач	Стоматолог	Вторая
14	Муканбеткалиева Ас...	Зав. отделения офтал...	Офтальмолог	Высшая
15	Мулдагалиева Динар...	Врач	Отоларинголог	Первая

Рисунок 3.8 – Список врачей клиники

Запись на приём осуществляется в соответствующей форме (рисунок 3.9). Записать пациента на приём к любому врачу может только сотрудник регистратуры. Врач же может записать пациента на повторный приём только к себе. Можно также удалить запись, если, например, пациент не сможет прийти и отменит приём. Оператору чтобы записать пациента на приём, необходимо в форме Врачи выбрать нужного врача правой кнопкой мыши и перейти в форму Запись на приём. Выбрав вид приёма (первичный, повторный) и время приёма из доступных вариантов, необходимо сохранить запись и распечатать направление для пациента.

Свободные для записи даты и время зависят от должности врача:

- главный врач и его заместитель могут вести приём только в понедельник с 12:00 до 17:00;
- заведующие отделений – во вторник и среду с 12:00 до 17:00;
- остальные врачи принимают с понедельника по пятницу с 8:00 до 19:00, в субботу – с 9:00 до 14:00.

Приёмы ведутся с интервалами в 30 минут.

Стоимость приёма формируется в зависимости от:

- должности врача. Стоимость первичного приёма:
- главного врача – 25 000 тенге;
- заместителя главного врача – 20 000 тенге;
- заведующего отделением – 15 000 тенге;
- остальных врачей – в зависимости от категории:
- вторая – 5 000 тенге за первичный приём;
- первая – 7 000 тенге;
- высшая – 10 000 тенге.
- вида приёма (первичный, повторный).

Скидка на повторный приём составляет 1 000 тенге от стоимости первичного приёма.

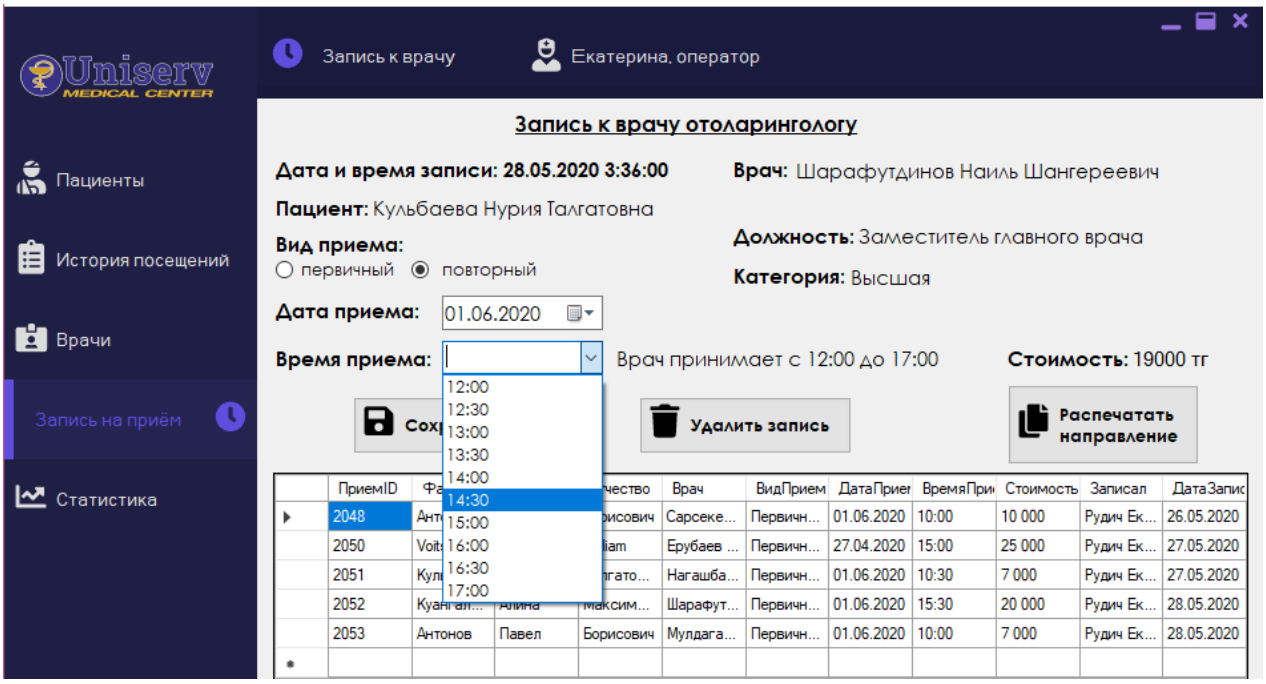


Рисунок 3.9 – Запись пациента на приём

В форме Статистика можно просмотреть количество посещений по выбранному параметру (врач, специализация, месяц) (рисунок 3.10).

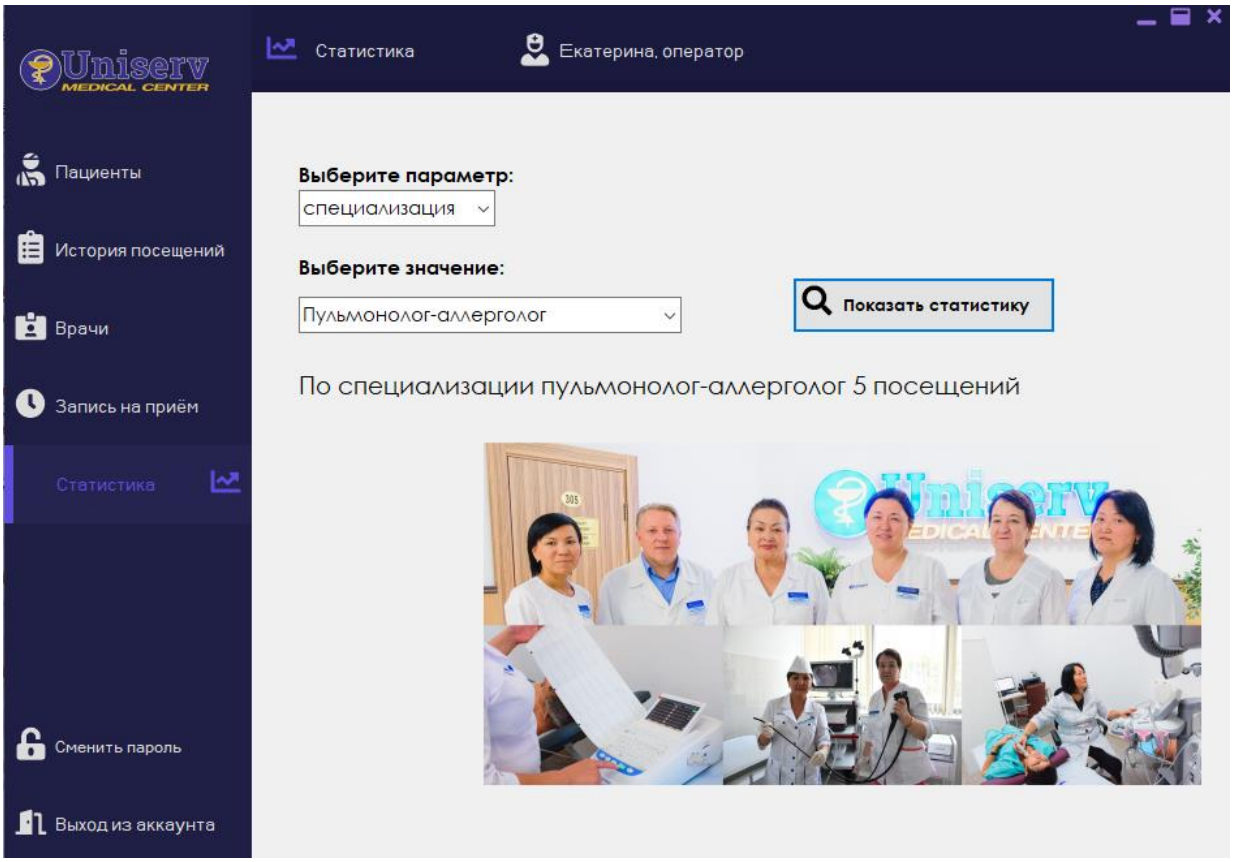


Рисунок 3.10 – Статистика посещений по выбранному критерию

4 Технико-экономическое обоснование проекта

4.1 Цели и задачи, решаемые в экономической части

В результате выполнения данной дипломной работы будет разработана информационная система (десктоп-приложение) для ведения электронных медицинских карт пациентов.

Для разработки данного приложения потребуется участие группы специалистов, состоящей из руководителя проекта и программиста-разработчика. Руководитель проекта несёт ответственность за распределение этапов работы и соблюдение графиков исполнения. Обязанностями программиста-разработчика являются поиск наиболее рационального решения, разработка и тестирование программного продукта.

Технико-экономическое обоснование дипломного проекта содержит следующие пункты:

- определение объёма и трудоёмкости разработки программного обеспечения;
- расчёт затрат на разработку ПО;
- смета затрат на разработку ПО;
- расчёт возможной цены разрабатываемого ПО;
- расчёт эксплуатационных затрат при использовании ПО;
- расчёт результатов от создания и использования ПО;
- расчёт основных показателей экономической эффективности и срока окупаемости проекта;
- выводы технико-экономической части [27].

4.2 Определение объёма и трудоёмкости разработки программного обеспечения

Основой для расчёта сметы затрат на разработку программного продукта являются объём и трудоёмкость процесса разработки. Для их определения процесс разработки был разделён на несколько этапов, которые представлены в таблице 4.1.

Таблица 4.1 – Этапы разработки ПО и оценка их трудоёмкости

№ этапа п/п	Описание работы	Трудоёмкость, чел.×ч.
1	Постановка задачи проекта	8
2	Разработка и утверждение технического задания на разработку ПО	16
3	Поиск и изучение сопутствующей литературы и подобных программ	24
4	Разработка клиентской части приложения	96
5	Разработка серверной части приложения	40

Продолжение таблицы 4.1

6	Тестирование, отладка и устранение неполадок программы	24
7	Внедрение готового программного продукта	16
Итого, трудоёмкость разработки ПО		224

Продолжительность рабочего дня равна восьми часам. Таким образом, для создания разрабатываемого программного обеспечения потребуется 224 /8 = 28 рабочих дней.

4. 3 Расчёт затрат на разработку программного продукта

Для нахождения затрат на разработку ПП необходимо вычислить следующие статьи расходов:

- расчёт затрат на разработку ПО;
- затраты на материалы и специальные программные средства, необходимые для разработки ПО;
- затраты на электроэнергию;
- затраты на оплату труда специалистов;
- затраты на социальный налог;
- амортизация основных фондов.

4.3.1 Расчёт затрат на материалы и специальные программные средства

Затраты на материалы для разработки ПО представлены в таблице 4.2.

Таблица 4.2 – Затраты на материалы

Наименование материального ресурса	Марка	Единица измерения	Коли-чество	Цена за единицу, тг	Общая сумма, тг
Блокнот	«Альт»	шт.	1	200	200
Тетрадь (36 листов)	«Маяк Канц»	шт.	1	120	120
Ручка	«Obama Marble»	шт.	2	100	200
Бумага офисная (А4)	«SvetoCopy»	упак.	1	2 000	2 000
Итого					2 520

При разработке будет использоваться ноутбук Dell Vostro 3568 210-AJIE, мощности которого достаточно для выполнения поставленной задачи. Ноутбук содержит установленную операционную систему Windows 10; программное обеспечение, необходимое для разработки ПО (Microsoft Visual Studio 2017, Microsoft SQL Manager Studio 2018), устанавливается бесплатно, поэтому производить дополнительные затраты на ОС и ПО не требуется. Затраты на специальные программные средства представлены в таблице 4.3.

Таблица 4.3 – Затраты на специальные программные средства

Наименование программного средства	Модель/ Название	Единица измерения	Количество	Цена за единицу, тг	Общая сумма, тг
Ноутбук	DELL Vostro 3568 210-AJIE	шт.	1	280 990	280 990
Мышь компьютерная	Delux DLM-125OGB	шт.	1	3 590	3 590
Батареи 1,5 V	«Сони»	шт.	2	80	160
Принтер	Samsung Xpress M2020W	шт.	1	45 800	45 800
Картридж для принтера	–	шт.	1	2 000	2 000
Интернет	Altel 4G	месяц	1	2 170	2 170
Операционная система	Windows 10	шт.	1	–	–
Среда разработки ПО	Microsoft Visual Studio 2017	шт.	1	–	–
Утилита для СУБД	MS SQL Manager Studio 2019	шт.	1	–	–
Итого					334 210

Общая сумма затрат на материалы и специальные программные средства ($З_м$) определяется по формуле:

$$З_м = \sum_{i=1}^{n} P_i * Ц_i, \quad (4.1)$$

где P_i – расход i-го вида материального ресурса, натуральные единицы;
$Ц_i$ – цена за единицу i-го вида материального ресурса, тг;
i – вид материального ресурса;
n – количество видов материальных ресурсов.

$$З_м = (2\,520 + 334\,210)\ \text{тг} = 336\,730\ \text{тг}$$

Таким образом, для разработки ПО понадобится материалов и специальных программных средств на сумму 336 730 тенге.

4.3.2 Расчёт затрат на электроэнергию

Так как для разработки ПО также потребуется электроэнергия, необходимо произвести расчёт затрат на электроэнергию, которая будет потрачена в течение всего времени разработки ПО (224 часа). Так как принтер не будет использоваться постоянно, для него расчёт будет произведён для периода в 24 часа.

Тариф на электоэнерию для юридических лиц города Алматы с 1 января 2020 года составляет 19,17 тенге за 1 кВт×ч с учётом НДС (согласно данным, опубликованным на официальном сайте ТОО «АлматыЭнергоСбыт»). Общая сумма затрат на электроэнергию ($З_Э$) рассчитывается по формуле:

$$З_Э = \sum_{i=1}^{n} M_i \times K_i \times T_i \times Ц\,, \qquad (4.2)$$

где M_i – паспортная мощность i-го электрооборудования, кВт;
K_i – коэффициент использования мощности i-го электрооборудования (K_i = 0,7..0,9);
T_i – время работы i-го оборудования за весь период разработки ПО, ч.;
Ц – цена электроэнергии (тариф), тг/кВт×ч.;
i – вид электрооборудования;
n – количество электрооборудования.
Результаты расчётов представлены в таблице 4.4.

Таблица 4.4 – Затраты на электроэнергию

Наименование оборудования	Паспортная мощность, кВт	Коэффиц-т использ-ния мощности	Время работы оборудования, ч	Цена электроэнергии, тг/кВт·ч	Сумма, тг
Ноутбук	0,7	0,9	224	19,17	2 705
Принтер	0,5	0,9	24	19,17	207
Освещение	0,3	0,7	224	19,17	902
Итого					3814

Согласно выполненным расчётам, затраты на электроэнергию составляют 3814 тенге.

4.3.3 Расчёт затрат на оплату труда специалистов

Фонд оплаты труда специалистов ($З_{ФОТ}$) можно вычислить по формуле:

$$З_{ФОТ} = З_{тр} + З_{доп.}, \quad (4.3)$$

где $З_{тр}$ – основная заработная плата специалистов, тг;
$З_{доп.}$ – дополнительная заработная плата специалистов, тг.
Основная заработная плата рассчитывается по следующей формуле:

$$З_{тр} = \sum_{i=1}^{n} ЧС_i * T_i \quad (4.4)$$

где $ЧС_i$ – часовая ставка i-го работника, тг;
T_i – трудоёмкость разработки ПО, чел.×ч.;
i – категория сотрудника;
n – количество сотрудников.
Дополнительная заработная плата составляет 10% от основной зар. платы и находится по формуле:

$$З_{доп.} = З_{тр} * 0{,}1. \quad (4.5)$$

Часовая ставка сотрудника будет рассчитана по формуле:

$$ЧС_i = \frac{ЗП_i}{ФРВ_i} \quad (4.6)$$

где $ЗП_i$ – месячная заработная плата i-го сотрудника, тг;
$ФРВ_i$ – месячный фонд рабочего времени i-го сотрудника, час.
Месячный фонд рабочего времени сотрудника определяется по формуле:

$$Ч_м = N_м * Ч_{рд}, \quad (4.7)$$

где $Ч_м$ – количество рабочих часов сотрудника за месяц;
$N_м$– количество рабочих дней за месяц;
$Ч_{рд}$ – количество рабочих часов в день.
Вычислим месячный фонд каждого сотрудника по формуле (4.7):

$$Ч_м = 28×8 = 224 \text{ ч.}$$

В разработке ПО будут задействованы два сотрудника: руководитель проекта и программист-разработчик. Средняя заработная плата руководителя проекта в Казахстане в 2020 году составляет 258 000 тенге, а программиста-

разработчика – 236 000 тенге. Вычислим часовую ставку каждого сотрудника по формуле (4.6):

$$ЧС_{рук.проекта} = \frac{258\,000}{224} = 1151{,}79 \text{ тг/ч}$$

$$ЧС_{прогр.-разр.} = \frac{236\,000}{224} = 1053{,}57 \text{ тг/ч}$$

Чтобы определить трудоёмкость разработки для каждого сотрудника, воспользуемся данными из таблицы 4.1. От руководителя проекта потребуется участие в постановке задачи проекта, разработке и утверждении технического задания, разработке клиентской части приложения, внедрении готового программного продукта:

$$Т_{рук.проекта} = 8 + 16 + 96 + 16 = 136 \text{ чел.}\times \text{ ч}$$

Таким образом, трудоёмкость руководителя проекта составит 136 чел.×ч.

Программист-разработчик будет занят разработкой и утверждением технического задания, поиском и изучением сопутствующей литературы и подобных программ, разработкой клиентской и серверной частей приложения, тестированием, отладкой и устранением неполадок программы и, наконец, внедрением готового программного продукта:

$$Т_{прогр.-разр.} = 16 + 24 + 96 + 40 + 24 + 16 = 216 \text{ чел.}\times \text{ ч}$$

Трудоёмкость программиста-разработчика составит 216×часов.

Согласно формуле (4.4), основная зар. плата специалистов составит:

- для руководителя проекта:

$$З_{тр_{рук.проекта}} = 1151{,}79 \text{ тг/ч} * 136 \text{ чел.}\times \text{ ч} = 156\,643 \text{ тг}$$

- для программиста-разработчика:

$$З_{тр_{прогр.-разр.}} = 1053{,}57 \text{ тг/ч} * 216 \text{ чел.}\times \text{ ч} = 227\,571 \text{ тг}$$

Общая сумма основной заработной платы труда специалистов равна:

$$З_{тр} = З_{тр_{рук.проекта}} + З_{тр_{прогр.-разр.}} = 156\,643 + 227\,571 = 384\,214 \text{ тг}$$

Затраты на основную заработную плату труда специалистов показаны в таблице 4.5.

Таблица 4.5 – Затраты на основную зар. плату труда специалистов

Специалист	Трудоёмкость разработки ПО, чел.×ч.	Часовая ставка специалиста, тг/ч	Сумма осн. з/п, тг
Руководитель проекта	136	1151,79	156 643
Программист-разработчик	216	1053,57	227 571
Итого			384 214

Найдём дополнительную заработную плату по формуле (4.5):

$$З_{доп.} = 384\ 214 * 0{,}1 = 38\ 421 \text{ тг.}$$

Таким образом, фонд оплаты труда специалистов согласно формуле (4.3) составит:

$$З_{ФОТ} = 384\ 214 + 38\ 421 = 422\ 635 \text{ тг.}$$

4.3.4 Расчёт затрат на налоги

Для подсчёта налогов, уплаченных юридическим лицом, необходимо вычислить размер обязательных пенсионных взносов (ОПВ), социальные отчисления (СО), отчисления на обязательное социальное медицинское страхование (ОСМС) и социальный налог (СН). Расчёты представлены в таблице 4.6. Итоговые налоговые отчисления составляют 10,46% от фонда оплаты труда (согласно Налоговому Кодексу РК).

Пенсионные отчисления составляют 10% от фонда оплаты труда. Таким образом,

$$ОПВ = З_{ФОТ} * 0,1 = 422\ 635 * 0{,}1 = 42\ 263{,}5 \text{ тг.}$$

Таблица 4.6. Расчёт затрат на налоги

Налог	Фонд оплаты труда ($З_{ФОТ}$), тг	Формула для вычисления	Сумма, тг
СО	422 635	($З_{ФОТ}$ – ОПВ)*3,5%	13 313
ОСМС		$З_{ФОТ}$*2%	8 452,7
СН		($З_{ФОТ}$ – ОПВ – ОСМС)*9,5% – СО	22 019,29
Итого			**44 208**

4.3.5 Амортизация основных фондов (ОФ)

Общая сумма амортизационных отчислений определяется по формуле:

$$З_А = \sum_{i=0}^{n} \frac{Ф_i * Н_{A_i} * N_i}{100 * 12 * n}, \quad (4.10)$$

где $Ф_i$ – стоимость i-го ОФ, тг;

$Н_{A_i}$ – годовая норма амортизации i-го ОФ, %;

N_i – время работы i-го ОФ за весь период разработки ПО, дней;

n – количество рабочих дней в месяце;

i – вид ОФ.

Основными фондами в данном дипломном проекте выступают ноутбук и принтер.

Годовая норма амортизации ОФ вычисляется по формуле:

$$H_{Ai} = \frac{100}{T_{Ni}}, \quad (4.11)$$

где $Т_{Ni}$– возможный срок использования i-го ОФ, год;

Норма амортизации для ноутбука составит:

$$Н_{А_{ноут.}} = \frac{100}{4} = 25\%.$$

Норма амортизации для принтера составит:

$$Н_{А_{прин.}} = \frac{100}{5} = 20\%.$$

Ноутбук за весь период разработки ПО понадобится для поиска и изучения сопутствующей литературы и подобных программ, разработки клиентской и серверной частей приложения, тестирования, отладки и устранения неполадок программы. Согласно таблице 4.1, время работы в днях составит:

$$N_{ноут.} = \frac{(24 + 96 + 40 + 24)\ ч}{8\ ч} = \frac{184}{8} = 23\ дня.$$

Принтер, как было упомянуто ранее, будет задействован в работе на 24 часа ($N_{прин.} = 1$ день).

Количество рабочих дней в месяце примем равным 21 (n=21).

Найдём общую сумму амортизационных отчислений по формуле (4.10):

$$З_А = \frac{280\,990 * 25 * 23}{100 * 12 * 21} + \frac{45\,800 * 20 * 1}{100 * 12 * 21} = (6411{,}5 + 36{,}3)\ \text{тг} = 6448\ \text{тг}.$$

Результаты расчётов показаны в таблице 4.7.

Таблица 4.7 – Амортизация основных фондов

Наименование оборудования	Стоимость оборудования, тг	Годовая норма амортизации, %	Время работы оборудования, дни	Сумма, тг
Ноутбук	280 990	25	23	6411,5
Принтер	45 800	20	1	36,3
Итого				6448

4.4 Смета затрат на разработку ПО

На основе произведённых расчётов оформляется смета затрат на разработку ПО, которая представлена в таблице 4.8. Соответствующая диаграмма продемонстрирована на рисунке 4.1.

Таблица 4.8 – Смета затрат на разработку ПО

Наименование статей затрат	Сумма, тг	% от общей суммы
Затраты на материалы	2 520	0,31
Затраты на специальные программные средства	334 210	41,07
Затраты на электроэнергию	3 814	0,47
Затраты на оплату труда специалистов	422 635	51,93
Затраты на социальный налог	44 208	5,43
Амортизация основных фондов	6 448	0,79
Итого	813 835	100

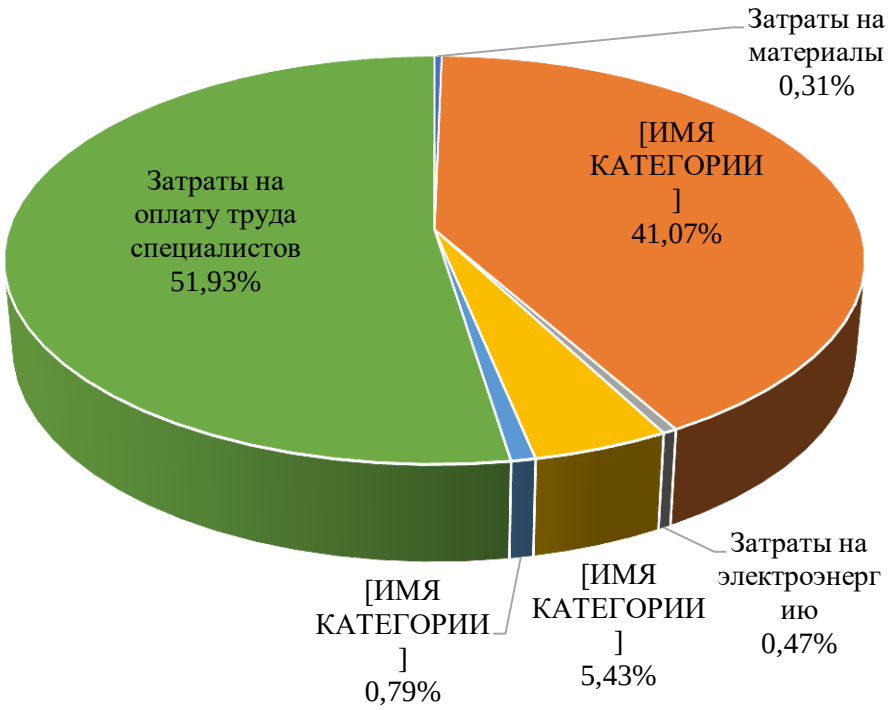


Рисунок 4.1 – Смета затрат на разработку ПО

4.5 Расчёт ориентировочной цены ПО

Стоимость программного обеспечения ($Ц_Д$) формируется в зависимости от затрат на его разработку и уровня рентабельности ПО и вычисляется по формуле:

$$Ц_Д = З_{нир}\left(1 + \frac{P}{100}\right), \quad (4.12)$$

где $З_{нир}$ – затраты на разработку программного обеспечения, тг;
Р – средний уровень рентабельности ПО, (%) (принят равным 25%).

$$Ц_Д = 805\,762\left(1 + \frac{25}{100}\right) = 1\,007\,202{,}5 \text{ тг}$$

Согласно Налоговому Кодексу РК, оплата налога на добавленную стоимость (НДС) является обязательной. Ставка НДС в 2020 году составляет 12% для ПО. Определить реализацию ПО с учётом НДС можно по формуле:

$$Ц_р = Ц_д(1 + НДС), \quad (4.13)$$

$$Ц_р = 1\,007\,202{,}5(1 + 0{,}12) = 1\,128\,066{,}8 \text{ тг.}$$

Получившуюся суму можно округлить до 1 128 067 тенге.

4.6 Расчёт эксплуатационных затрат при использовании ПО

Годовые эксплуатационные текущие затраты в условиях функционирования ПО (C) рассчитываются по формуле:

$$\text{С} = \text{М} + \text{ЗП} + \text{А} + \text{ОТ} + \text{НР} = \text{Ц}_{\text{д}}(1 + \text{НДС}), \quad (4.14)$$

где М – годовые материальные затраты на сопровождение программного продукта, тг;

ЗП – годовые затраты на оплату труда специалиста (программиста-разработчика), тг;

А – затраты на амортизацию, тг;

ОТ – отчисления на социальный налог, тг;

НР – накладные расходы, тг.

Затраты на материалы в условиях функционирования ПО не ожидается, соответственно $\text{М} = 0$.

Годовые затраты на заработную плату сотрудников определяются по формуле:

$$\text{ЗП} = \frac{\text{О}_{\text{с}} \cdot \text{Ч}_{\text{с}} \cdot 12}{\Phi_{\text{р.в.}}} \cdot t_{\text{общ}} \cdot 12 \cdot \left(1 + \text{К}_{\text{Д}}\right), \quad (4.15)$$

где O_c – оклад специалиста, тг/мес.;

Ч_c – численность специалистов, участвующих в процессе, чел.;

$\Phi_{\text{р.в.}}$ – годовой фонд рабочего времени, час;

$t_{\text{общ}}$ – трудоёмкость решения задач в условиях функционирования ПО в месяц, час;

$K_{\text{д}}$ – коэффициент дополнительной заработной платы.

Трудоёмкость решения задач вычисляется следующим образом:

$$t_{\text{общ}} = \sum_{k=1}^{n} t_k \cdot K_k, \quad (4.16)$$

где t_{k} – затраты времени на решение k-й задачи, час;

K_{k} – количество решаемых k-х задач в месяц, ед.

В ходе эксплуатации программы предполагается возникновение серверных ошибок. На поиск и исправление одной ошибки в среднем уходит около полутора часа. В течение месяца эксплуатации ПО может возникать до 5 ошибок. Помимо этого, ожидается внесение изменений в базу данных, а также доработка компонентов по требованию заказчика. Всего компонентов в системе 4, на доработку одного уходит в среднем около 6 часов.

$$t_{\text{общ}} = \ 1{,}5 \ \cdot 5 + 6 \ \cdot 4 = 7{,}5 + 24 = 31{,}5 \text{ часов}.$$

Годовой фонд рабочего времени равен 1632 часам (разработка клиентской и серверной частей из таблицы 4.1). Таким образом,

$$ЗП = \frac{240\,000 \cdot 1 \cdot 12}{1632} \cdot 31{,}5 \cdot 12 \cdot (1 + 0{,}1) = 733\,765 \text{ тенге.}$$

Амортизационные отчисления выражаются в процентах к балансовой стоимости оборудования и вычисляются по формуле:

$$А = \sum_{i=1}^{n} С_{обор.i} \cdot Н_{а_i}, \qquad (4.17)$$

где $С_{обор.i}$ – первоначальная стоимость оборудования;

$Н_{а_i}$ – норма амортизации i-го оборудования;

i – вид оборудования;

n – количество видов оборудования.

Затраты на амортизацию составят:

$$А = 280\,990 * 0{,}25 + 45\,800 * 0{,}2 = 70\,247{,}5 + 9\,160 = 79\,408 \text{ тг.}$$

Расчёты отчислений на социальный налог и накладные расходы производятся так же, как и при разработке ПО.

$$ОТ = ЗП * 10{,}46\% = 733\,765 * 0{,}1046 = 76\,752 \text{ тенге.}$$

$$НР = ЗП \cdot \frac{Н_{нр}}{100} = 733\,765 * 0{,}7 = 513\,636 \text{ тенге.}$$

Эксплуатационные затраты показаны в таблице 4.9.

Таблица 4.9 – Эксплуатационные затраты

Вид затраты	Сумма, тг	В процентах от общей суммы, %
Материальные затраты (М)	0	0
Заработная плата (ЗП)	733 765	52,28
Амортизационные отчисления (А)	79 408	5,66
Социальные отчисления и налоги (ОТ)	76 752	5,47
Накладные расходы (НР)	513 636	36,6
Итого	1 403 561	100

Годовые эксплуатационные затраты в условиях функционирования ПО составят 1 403 561 тенге.

4.7 Расчёт результатов от создания и использования ПО

Разрабатываемое ПО предназначено для хранения систематизированной информации о пациентах медицинского учреждения в цифровом виде, что обеспечивает экономию рабочего времени специалиста за счёт уменьшения количества бумажной работы и автоматизированном поиске информации из электронной базы данных.

Для анализа экономии в результате использования разрабатываемого ПО необходимо сравнить эксплуатационные расходы с внедрением ПО и без.

Статьи затрат при использовании ПО включают в себя:

- заработная плата специалиста, который осуществляет поддержку и сопровождение системы;
- износ оборудования;
- накладные расходы.

Использование ПО не предполагает материальных расходов.

В организации имеются два администратора, заработная плата которых 130 000 тенге в месяц. Основной задачей каждого администратора является ведение бумажной картотеки пациентов учреждения, осуществление записи пациентов на приём.

Годовые затраты на оплату труда двух администраторов рассчитываются по формуле (4.15):

$$ЗП = \frac{150\,000 \cdot 2 \cdot 12}{1200} \cdot 24 \cdot 12 \cdot (1 + 0{,}1) = 823\,680 \text{ тенге.}$$

Социальные отчисления и налоги:

$$ОТ = 823\,680 * 10{,}46\% = 86\,157 \text{ тенге.}$$

Для ведения записей администратору потребуются бумажная картотека, сделанная на заказ в типографии (1300 тг за экземпляр, около 50 экземпляров в месяц, итого ~65 000 тенге), канцелярных принадлежностей на сумму 10 000 тенге, принтер для ксерокопирования (73 000 тенге). Итого затрат на материалы и оборудование – 148 000 тенге.

Затраты на амортизацию составят:

$$А = 73\,000 * 0{,}25 = 18\,250 \text{ тенге.}$$

На заправку картриджа в месяц уходит порядка 20 000 тенге.

Накладные расходы составят:

$$НР = ЗП \cdot \frac{Н_{нр}}{100} = 823\,680 * 0{,}7 = 576\,576 \text{ тенге.}$$

Сравним следующие статьи затрат без и с применением разрабатываемого ПО: заработная плата специалистов, расход на материалы и оборудование, износ оборудования и накладные расходы в таблице 4.10:

Таблица 4.10 – Сравнение статей затрат с использованием ПО и без

Статья затрат	Без использования ПО, тг	С использованием ПО, тг
Материальные затраты (М)	148 000	0
Заработная плата (ЗП)	823 680	733 765
Амортизационные отчисления (А)	18 250	79 408
Социальные отчисления и налоги (ОТ)	86 157	76 752
Накладные расходы (НР)	576 576	513 636
Итого	1 652 663	1 403 561

Вычислим ожидаемую условно-годовую экономию по формуле:

$$Э_{уг} = С_1 - С_2 + \sum Э_i, \quad (4.18)$$

где $Э_{уг}$ – величина экономии, тг;

$С_1$ и $С_2$ – показатели затрат по базовому и внедряемому вариантам, тг:

$\Sigma Эi$ – ожидаемый дополнительный эффект от различных факторов, тг.

Таким образом, ожидаемая условно-годовая экономия составит 249 102 тенге.

4.8 Расчёт основных показателей экономической эффективности

Эффективность разрабатываемого ПО формируется за счёт экономии в сравнении с предыдущим периодом работы без его использования.

Для расчёта величины ожидаемого годового экономического эффекта от внедрения ПО используется формула:

$$Э_г = Э_{уг} - К * Е_Н, \quad (4.19)$$

где $Э_г$ – ожидаемый годовой экономический эффект, тг:

$Э_{уг}$ – ожидаемая условно-годовая экономия, тг;

К – капитальные вложения, тг;

$Е_Н$ – нормативный коэффициент экономической эффективности капитальных вложений.

Нормативный коэффициент экономической эффективности капитальных вложений вычисляется по формуле:

$$E_Н = \frac{1}{T_Н}, \quad (4.20)$$

где $T_Н$ – нормативный срок окупаемости капитальных вложений, лет. Для ПО примем равным четырём годам ($T_Н = 4$).

Следовательно,

$$E_Н = \frac{1}{4} = 0{,}25.$$

Ожидаемый годовой экономический эффект согласно формуле (4.19) равен:

$$Э_{уг} = 1\,128\,067 - 249\,102 * 0{,}25 = \ 1\,065\,792 \text{ тенге.}$$

Расчетный коэффициент экономической эффективности капитальных вложений находится по формуле:

$$E_р = \frac{Э_г}{К}, \quad (4.21)$$

где $E_р$ – расчётный коэффициент экономической эффективности капитальных вложений;
$Э_г$ – ожидаемая годовая экономия, тг;
К – капитальные вложения на создание системы, тг.

$$E_р = \frac{1\,128\,067}{249\,102} = 4{,}53.$$

Расчетный срок окупаемости капитальных вложений составляет:

$$T_Р = \frac{1}{E_Р} = \frac{1}{4{,}53} = 0{,}22 \text{ года} \approx 2{,}6 \text{ месяцев.}$$

Основные показатели экономической эффективности проекта представлены в таблице 4.11:

Таблица 4.11 – Основные показатели экономической эффективности

Показатель	Значение
Ожидаемый годовой экономический эффект, тг	1 065 792
Коэффициент экономической эффективности капитальных вложений	4,53
Срок окупаемости капитальных вложений, месяцев	2,6

4.9 Выводы по технико-экономическому обоснованию

В данной главе были проанализированы и вычислены затраты на разработку программного обеспечения. Разрабатываемое ПО обеспечит экономию рабочего времени специалиста за счёт уменьшения количества бумажной работы и автоматизации процесса поиска, а также снижение расходов на материалы, так как вся информация будет в электронном виде.

Большую часть затрат на разработку ПО составят затраты на оплату труда специалистов (52% от общих расходов).

Согласно произведённым расчётам, цена реализации разрабатываемого программного продукта с учётом НДС составит 1 128 067 тенге, себестоимость ПО (смета затрат) – 805 762 тенге. Соответственно, ожидаемая прибыль от реализации ПО составит 322 305 тенге.

Ожидаемый годовой экономический эффект составит 1 065 792 тенге. Приложение окупится в первые 2,6 месяцев его использования.

5 Безопасность жизнедеятельности

Дипломный проект посвящён разработке информационной системы ведения электронных медицинских карт пациентов. Условием для работы с проектируемой системой является соблюдение условий труда, установленных Санитарными правилами [28]. В данном разделе дипломного проекта будет дана санитарно-гигиеническая оценка условий труда, а также рассчитано производственное освещение рабочего места, удовлетворяющее требованиям техники безопасности и охраны труда при работе с проектируемой системой.

Помещением, где будет использоваться программа, является кабинет медицинского учреждения, расположенный в городе Уральске. Площадь такого помещения составляет 18 м^2 (6 метров в длину и 3 метра в ширину). Потребуется одно рабочее место – стол с компьютером (ПК).

5.1 Расчёт санитарно-гигиенической оценки условий труда

Санитарно-гигиеническая оценка условий труда будет дана на основе требований к размещению ПК, параметров микроклимата, уровня звукового давления в октавных полосах частот и уровня звука, создаваемого компьютером, виброускорения, значения уровней неионизирующих электромагнитных излучений, уровня концентраций аэроионов и коэффициента униполярности.

Согласно требованиям к размещению и эксплуатации компьютера, площадь на одно рабочее место пользователя ПК составляет не менее 4 м^2, расстояние между стеной с оконными проемами и столами – 0,5 метров, стеной и столами – 0,4 м при периметральной расстановке. План размещения рабочего места, соответствующий требованиям, представлен на рисунке 5.1.

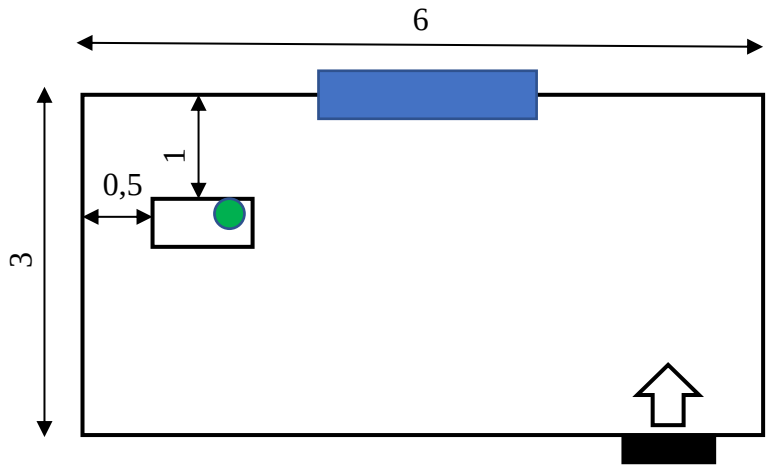


Рисунок 5.1 – План размещения рабочего места пользователя ПК

Должны быть соблюдены следующие требования к микроклимату в помещении:

- помещение должно быть оборудовано системами отопления, вентиляцией, кондиционерами;
- перед началом работы и через каждый час работы за компьютером (ноутбуком) должно осуществляться сквозное проветривание.

Следующие измерения проводятся на рабочем месте пользователя: уровень звукового давления и звука, создаваемого компьютером (ноутбуком), виброускорение, уровень неионизирующих электромагнитных излучений и уровень концентрации аэроионов. Допустимые значения для этих параметров представлены в таблице 5.1 (выделены «полужирным» шрифтом).

Таблица 5.1 – Допустимые значения санитарно-гигиенических параметров

<table>
<tr><td>Наименование параметра</td><td colspan="10">Допустимое значение</td></tr>
<tr><td rowspan="2">Уровень звукового давления в октавных полосах частот в зависимости от среднегеометрических частот</td><td>Среднегеометрическая частота, Гц</td><td>31,5</td><td>63</td><td>125</td><td>250</td><td>500</td><td>1000</td><td>2000</td><td>4000</td><td>8000</td></tr>
<tr><td>Уровень звукового давления, дБ</td><td>86</td><td>71</td><td>61</td><td>54</td><td>49</td><td>45</td><td>42</td><td>40</td><td>38</td></tr>
<tr><td>Уровень звука, создаваемого компьютером (ноутбуком), дБА</td><td colspan="10">50</td></tr>
<tr><td rowspan="2">Виброускорение</td><td colspan="6">Корректированное значение ($мс^{-2}$)*10^{-2}</td><td colspan="4">1,0</td></tr>
<tr><td colspan="6">Корректированный уровень, дБ</td><td colspan="4">80</td></tr>
<tr><td rowspan="3">Уровень неионизирующих электромагнитных излучений (для монитора, клавиатуры, мыши)</td><td colspan="6">Напряженность электро-статического поля, кВ/м</td><td colspan="4">20</td></tr>
<tr><td colspan="6">Плотность магнитного потока вокруг ПК, нТл
в диапазоне частот 5 -2000 Гц:
в диапазоне частот 2-400 кГц:</td><td colspan="4">250
25</td></tr>
<tr><td colspan="6">Поверхностный электростатический потенциал от монитора, В</td><td colspan="4">500</td></tr>
<tr><td rowspan="3">Уровень концентрации аэроионов, ион/$см^3$</td><td>Значение</td><td colspan="3">Положительной полярности</td><td colspan="3">Отрицательной полярности</td><td colspan="2">Коэффициент униполярности, У</td></tr>
<tr><td>Оптимальное</td><td colspan="3">1 500 ≤ ρ^+ < 3 000</td><td colspan="3">3 000 < ρ^- ≤ 5 000</td><td colspan="2" rowspan="2">0,4 < У < 1,0</td></tr>
<tr><td>Максимально допустимое</td><td colspan="3">ρ^+ < 50 000</td><td colspan="3">ρ^- ≤ 50 000</td></tr>
</table>

Коэффициент униполярности вычисляется по формуле:

$$У = \frac{\rho^+}{\rho^-}, \quad (5.1)$$

где ρ^+ – количество ионов положительной полярности в одном кубическом

сантиметре воздуха, ион/см^3;

ρ^- – количество ионов отрицательной полярности в одном кубическом

сантиметре воздуха, ион/см^3.

Допустим, количество ионов положительной полярности в одном кубическом сантиметре воздуха составляет 2208 ион/см^3, а отрицательной полярности – 4395 ион/см^3. Коэффициент униполярности равен:

$$У = \frac{2208 \text{ ион/см}^3}{4395 \text{ ион/см}^3} = 0{,}502,$$

что соответствует оптимальному значению.

5.2 Расчёт производственного освещения рабочего места

Расчёты в данном разделе будут выполнены в соответствии с требованиями СНиП РК 2.04-05-2002 [29].

Так как в кабинете медицинского учреждения будут постоянно пребывать люди, поэтому там должно быть естественное освещение.

В кабинете естественное освещение будет боковым. Для его проектирования нужно вычислить площадь световых проёмов по формуле:

$$100\frac{S_0}{S_n} = \frac{e_N K_з \eta_0}{\tau_0 r_1} K_{зд}, \quad (5.2)$$

где S_0 – площадь световых проемов при боковом освещении, м^2;

S_n – площадь пола помещения, м^2;

e_N – нормируемое значение КЕО;

$K_з$ – коэффициент запаса;

η_0 – световая характеристика окон;

τ_0 – общий коэффициент светопропускания, определяемый по формуле:

$$\tau_0 = \tau_1\tau_2\tau_3\tau_4\tau_5, \quad (5.3)$$

где τ_1 – коэффициент светопропускания материала,

τ_2 – коэффициент, учитывающий потери света в переплетах светопроёма;

τ_3 – коэффициент, учитывающий потери света в несущих конструкциях;

τ_4 – коэффициент, учитывающий потери света в солнцезащитных устройствах;

τ_5 – коэффициент, учитывающий потери света в защитной сетке, устанавливаемой под фонарями, принимают равным 0,9;

r_1 – коэффициент, учитывающий повышение КЕО при боковом освещении, благодаря свету, отраженному от поверхности помещения и подстилающего слоя, примыкающего к зданию;

$K_{зд}$ – коэффициент, учитывающий затемнение окон противостоящими зданиями.

Нормированное значение КЕО e_N для зданий, располагаемых в различных районах, определяется по формуле:

$$e_N = e_н m_N, \quad (5.4)$$

где N – номер группы обеспеченности естественным светом;

$e_н$ – значения КЕО;

m_N – коэффициент светового климата.

Город Уральск относится ко второй группе административных районов, поэтому N=2. Световой проём ориентирован на юго-запад (коэффициент светового климата m_2 равен 0,8). Так как характеристика зрительной работы предполагает наивысшую точность, $e_н$=2. Вычислим нормированное значение КЕО:

$$e_N = 2 * 0{,}8 = 1{,}6.$$

Светопропускающий материал – стекло оконное листовое тройное, коэффициент светопропускания τ_1 = 0,75.

Переплёт для окон деревянный спаренный, соответственно коэффициент, учитывающий потери света в переплётах τ_2, равен 0,7;

Коэффициент, учитывающий потери света в несущих конструкциях, при боковом освещении равен τ_3 = 1.

Так как используются убирающиеся регулируемые жалюзи и шторы, коэффициент, учитывающий потери света в солнцезащитных устройствах, равен τ_4= 1;

Коэффициент, учитывающий потери света в защитной сетке, устанавливаемой под фонарями, принимают равным τ_5 = 0,9.

Вычислим общий коэффициент светопропускания по формуле (5.3):

$$\tau_0 = 0{,}75 * 0{,}7 * 1 * 1 * 0{,}9 = 0{,}4725.$$

Площадь пола помещения равна S_n = 18 м^2.

Для кабинета мед. учреждения при расположении светопропускающего материала вертикально коэффициент запаса равен $K_з$ = 1,2.

Отношение длины помещения к его глубине (6/3) равно двум, отношение глубины помещения к его высоте от уровня условной рабочей поверхности до верха окна (3/1,95) равно 1,54, соответственно, световая характеристика окон равна η_0 = 9.

Отношение расстояния расчётной точки от наружной стены к глубине помещения (рисунок 5.1) равно 1. Таким образом, коэффициент, учитывающий повышение КЕО при боковом освещении, равен r_1 = 1,8;

Отношение расстояния между рассматриваемым и противостоящим зданием Р к высоте расположения карниза противостоящего здания над подоконником рассматриваемого окна $Н_{зд}$ (150/10) равно 15, поэтому коэффициент, учитывающий затемнение окон противостоящими зданиями, равен $K_{зд}$ = 1.

Выразим площадь световых проемов для проектируемого помещения из формулы (5.2) и найдём её:

$$S_0 = \frac{S_n e_N K_з \eta_0 K_{зд}}{100 \tau_0 r_1} = \frac{18 \text{ м}^2 \cdot 1{,}6 \cdot 1{,}2 \cdot 9 \cdot 1}{100 \cdot 0{,}4725 \cdot 1{,}8} = 3{,}657 \text{ м}^2 \approx 3{,}66 \text{ м}^2.$$

5.3 Выводы раздела безопасности жизнедеятельности

В данном разделе дипломного проекта была дана санитарно-гигиеническая оценка условий труда при проектировании системы, а также произведён расчёт производственного освещения рабочего места. Было вычислено, что площадь световых проемов для проектируемого помещения (кабинета медучреждения в г. Уральске) должна составлять 3,66 м^2.

Заключение

Благодаря стремительной информатизации общества и доступности сети Интернет информатизация здравоохранения является вполне выполнимой задачей. Электронные медицинские карты обладают рядом существенных преимуществ как для врачей, так и для пациентов, и автоматизация их ведения представляет большую практическую ценность: уменьшается бюрократия, сокращается и время для сбора информации, соответственно пациент быстрее получает лечение. Кроме того, повышается общий уровень и качество предоставления медицинских услуг.

Для решения данной задачи в результате настоящей дипломной работы было разработано десктопное приложение для ведения ЭМК пациентов медицинской клиники «Uniserv Medical Center».

В результате выполнения дипломной работы было выяснено, что разрабатываемое ПО обеспечит экономию рабочего времени специалиста за счёт уменьшения количества бумажной работы и автоматизации процесса поиска, а также снижение расходов на материалы, так как вся информация будет в электронном виде.

Процесс подготовки к разработке ПО и непосредственно самого проектирования был описан в аналитической и проектной частях. В экспериментальной части были даны инструкции к эксплуатации программы.

В экономической части проекта было выяснено, что ожидаемая прибыль от реализации ПО составит 322 305 тенге. Ожидаемый годовой экономический эффект составит 1 065 792 тенге. Приложение окупится в первые 2,6 месяцев его использования.

В разделе безопасность жизнедеятельности была дана санитарно-гигиеническая оценка условий труда во время разработки ПО, а также было вычислено, что площадь световых проемов для помещения, в котором будет использоваться программный продукт, должна составлять 3,66 $м^2$, чтобы удовлетворить требования техники безопасности и охраны труда.

В ходе выполнения дипломного проекта были закреплены и расширены знания, полученные в процессе обучения, выработалась возможность последующего использования данных навыков для решения инженерно-технических задач, связанных с темой дипломного проекта.

Список использованной литературы

1 Абдуманонов А.А., Алиев Р.Э., Карабаев М.К., Хошимов В.Г. О проектировании медицинских баз данных и информационных систем для организации и управления лечебно-диагностических процессов // T-Comm: Телекоммуникации и транспорт. – 2016. – Том 10. – №1. – С. 45-53.

2 Путило Н.В. Информационные технологии в сфере охраны здоровья: Научно-практический комментарий к федеральному закону от 29 июля 2017 г. № 242-ФЗ. – М.: Проспект, 2018. – 82 с.

3 Кухтичев А.А. Электронная медицинская карта как основа сервисов цифровой медицины информационной системы «ЦифроМед» // Вестник НГУ. Серия: Информационные технологии. – 2016. – Т. 14, №1. – С. 61-75.

4 Belcher, K. From $600 M to $6 Billion, Artificial Intelligence Systems Poised for Dramatic Market Expansion in Healthcare // WW2.FROST.COM: Frost & Sullivan. 2016. URL: https://ww2.frost.com/news/press-releases/600-m-6-billion-artificial-intelligence-systems-poised-dramatic-market-expansion-healthcare/ (дата обращения: 10.02.2020).

5 РГП на ПХВ «Республиканский центр здравоохранения» МЗ РК. Внедрение процедуры сертификации на соответствие стандартам электронного здравоохранения. Нур-Султан: rcrz.kz, 2018. – 5 с.

6 Цифровизация здравоохранения // M.EGOV.KZ: Электронное правительство Республики Казахстан. 2020. URL: http://m.egov.kz/cms/ru/healthcare?mobile=yes (дата обращения: 20.02.2020).

7 Биртанов рассказал о преимуществах и минусах цифровизации // PROFIT.KZ: PROFIT Online. 2019. URL: https://profit.kz/news/53519/Birtanov-rasskazal-o-preimuschestvah-i-minusah-cifrovizacii/ (дата обращения: 9.02.2020).

8 Каирленов М. Рынок МИС-ов 2020: ПАКС-ы, ЛИС-ы и цифровое рабство // ULAGAT-M.KZ: Деловой медицинский журнал «Улагатты медицина». 2020. URL: https://ulagat-m.kz/special-projects/rynok-mis-ov-2020-paks-y-lis-y-i-tsifrovoe-rabstvo.html (дата обращения: 20.02.2020).

9 Facebook.com // [Персональная страница Абишева О.] / 2018. URL: https://web.facebook.com/photo.php?fbid=10160469910065304&set=a.312044070303.328851.724200303&type=3&_rdc=1&_rdr (дата обращения: 20.02.2020).

10 Делаем медицину ближе и доступнее // CIT-DAMU.KZ: ТОО "Центр Информационных Технологий ДАМУ". 2020. URL: http://www.cit-damu.kz/ (дата обращения: 20.02.2020).

11 О компании // E-MIS.KZ: Медицинская информационная система «Жетісу». 2017. URL: http://e-mis.kz/ (дата обращения: 20.02.2020).

12 Медицинская информационная система Medesk // MEDESK.NET: Medesk. 2020. URL: https://www.medesk.net/ru/ (дата обращения: 20.02.2020).

13 We’re here to help // ADVANCEDMD.COM: AdvancedMD, Inc. 2020. URL: https://www.advancedmd.com/ (дата обращения: 20.02.2020).

14 TIOBE Index for May 2020 // TIOBE.COM: TIOBE Software BV. 2020. URL: https://www.tiobe.com/tiobe-index/ (дата обращения: 10.05.2020).

15 Фленов М.Е. Библия C#. – 3-е изд., перераб. и доп. – СПб.: БХВ-Петербург, 2016. – 544 с.

16 Пахомов Б.И. C# для начинающих. – СПб.: БХВ-Петербург, 2014. – 432 с.

17 C# documentation // DOCS.MICROSOFT.COM: Microsoft Docs. 2020. URL: https://docs.microsoft.com/en-us/dotnet/csharp/ (дата обращения:15.02.2020).

18 Викулина Д.А., Макаров С.Н., Кунгурцева К.В., Гаранина Е.А. Современные технологии создания desktop-приложений // Наука и современность. – 2012. – С. 180-186.

19 Хомоненко А. Д., Цыганков В. М., Мальцев М. Г. Базы данных: Учебник для высших учебных заведений. – 6-е изд., доп. – СПб.: Корона-Принт, 2009. – 736 с.

20 SQL или NoSQL – вот в чём вопрос // HABR.COM: Хабр. 2017. URL: https://habr.com/ru/company/ruvds/blog/324936/ (дата обращения: 5.03.2020).

21 Что такое базы данных NoSQL? // AWS.AMAZON.COM: Amazon Web Services. 2020. URL: https://aws.amazon.com/ru/nosql/ (дата обращения: 10.03.2020).

22 Пивоваров С. Обзор систем управления базами данных (СУБД) для систем контроля и управления доступом (СКУД) // PARSEC.RU: Parsec. 2020. URL: https://www.parsec.ru/articles/obzor-sistem-upravleniya-bazami-dannykh-subd-dlya-sistem-kontrolya-i-upravleniya-dostupom-skud/ (дата обращения: 10.03.2020).

23 SQL Server technical documentation // DOCS.MICROSOFT.COM: Microsoft Docs. 2020. URL: https://docs.microsoft.com/en-us/sql/sql-server/?view=sql-server-ver15 (дата обращения: 5.03.2020).

24 Базы данных: Учебно-методическое пособие по дисциплине «Базы данных» для направления подготовки 38.03.05 «Бизнес-информатика» (бакалавриат) / Р. А. Жуков. – М: Директ-Медиа, 2019. – 177 с.

25 Поелуева Е.С., Козюкова Е.С., Ветчинкин Д.А. Обзор средств разработки информационных систем // Apriori. Серия: Естественные и технические науки. – 2015. - №6. – С. 1-6.

26 Введение в проектирование баз данных // ANALYTICS.INFOZONE.PRO: База знаний по бизнес-анализу. 2015. URL: https://analytics.infozone.pro/vvedenie-v-proektirovanie-baz-dannih/ (дата обращения: 19.02.2020).

27 Методические указания по выполнению экономической части выпускной работы для студентов специальностей 5В070400 – Вычислительная техника и программное обеспечение, 5В070300 – Информационные системы, 5В060200 - Информатика / Г.Ш. Боканова. – Алматы: АУЭС, 2020 – 35 с.

28 Об утверждении Санитарных правил «Санитарно-эпидемиологические требования к условиям работы с источниками физических факторов (компьютеры и видеотерминалы), оказывающих

воздействие на человека»: Приказ министра национальной экономики Республики Казахстан от 21.01.2015 № 38 // Информационно-правовая система нормативных правовых актов РК «Әділет». – 2015.

29 СНиП РК 2.04.-05.2002 Естественное и искусственное освещение. Государственные нормативы в области архитектуры, градостроительства и строительства.

Приложение А
Техническое задание

Название программы: Электронная картотека «Uniserv MC»

Краткое описание: Программа для ведения электронных медицинских карт пациентов, историй их болезни и записи пациентов на приём

Постановка задачи:

1) Реализовать возможность создания логинов и паролей для пользователей программы IT-специалисту клиники.

2) Создать формы Авторизация, Главное окно, Пациенты, История посещений, Врачи, Приём пациента, Запись на приём, Статистика, Выход из аккаунта

3) В форме «Авторизация» реализовать:

- вход в приложение в качестве оператора (сотрудник регистратуры) или врача;
- возможность смены пароля аккаунта для всех пользователей.

4) В форме «Главное окно» реализовать отображение:

- имени и специализации авторизованного пользователя;
- логотипа компании;
- часов;
- текущей даты.

5) В форме «Пациенты» реализовать:

- добавление, удаление, изменение информации о пациентах;
- запрет на добавление пациента в базу данных пользователям-врачам;
- вывод списка пациентов таблицей с возможностью поиска.

6) В форме «История посещений» реализовать:

- вывод списка посещений пациентов таблицей с возможностью просмотреть данные выбранного приёма в отдельно открывающейся форме

7) В форме «Врачи» реализовать:

- вывод списка врачей таблицей с возможностью поиска.

8) В форме «Приём пациента» реализовать:

- запрет доступа к форме для пользователей-операторов;
- вывод текущих даты и времени;
- вывод ФИО, специализации авторизованного врача;
- вывод ФИО выбранного пациента;
- вывод Анамнеза жизни из картотеки пациента;
- вывод стоимости приёма пациента;
- вывод даты следующего осмотра автоматически через неделю с возможностью смены на любой другой день;
- ввод данных об осмотре пациента;
- сохранение выведенных данных осмотра в БД;
- распечатку данных осмотра.

9) В форме «Запись на приём» реализовать:

- возможность записать пациента к любому врачу для пользователей-операторов;
- возможность записать пациента только к авторизованному врачу для пользователей-врачей;
- вывод ФИО, специализации, должности и категории авторизованного или выбранного врача;
- вывод ФИО выбранного пациента;
- вывод вида приёма;
- вывод текущей даты приёма с возможностью изменить на запланированную, свободную дату;
- выбор запланированного, свободного времени приёма;
- вывод стоимости приёма;
- сохранение записи в БД;
- удаление записи из БД;
- распечатку квитанции на приём.

10) В форме «Статистика» реализовать:

- вывод количества посещений по выбранному врачу, специализации, месяцу.

Приложение Б
Листинг программы

```
//Код класса PatientClasses для добавления, удаления и изменения данных о пациенте в БД
using System;
using System.Configuration;
using System.Data;
using System.Data.SqlClient;
using System.Drawing.Imaging;
using System.IO;
using System.Windows.Forms;

namespace Diploma_Version_1.PatientClasses
{
   class PatientsClass
   {
      //Getter'ы и Setter'ы. Передают значения переменных
      public DataGridView dtGridVPatientList { get; set; }
      public int PatientID { get; set; }
      public string PatientType { get; set; }
      public byte[] Photo { get; set; }
      public string FileName { get; set; }
      public string Surname { get; set; }
      public string Name { get; set; }
      public string Otchestvo { get; set; }
      public DateTime BirthDate { get; set; }
      public int Years { get; set; }
      public int Months { get; set; }
      public string Pol { get; set; }
      public string DocumentType { get; set; }
      public string DocumentNo { get; set; }
      public string IIN { get; set; }
      public string Telefon { get; set; }
      public string DomAdres { get; set; }
      public int GruppaKrovi { get; set; }
      public string Rezus { get; set; }
      public int Rost { get; set; }
      public int Ves { get; set; }
      public int LechVrach { get; set; }
      public string Zabolevaniya { get; set; }
      public string Allergies { get; set; }
      public int ModifiedBy { get; set; }
      public object SqlConection { get; private set; }

      //Строка подключения к БД
      static string myConnectionString =
ConfigurationManager.ConnectionStrings["connectionString"].ConnectionString;
      SqlConnection connection = new SqlConnection(myConnectionString);

      //Метод для выбора необходимых данных из БД
      public DataTable Select()
```

```
{
        //Connect database
        DataTable dataTable = new DataTable();
        try
        {
           //SQL-запрос
           string testSqlSelect = "SELECT [№ пациента], Категория, Фото, Фамилия, Имя,
Отчество, [Дата рождения], Лет, Месяцев, Пол, [Тип документа], [№ документа], ИИН,
Телефон, [Дом. адрес], [Группа крови], [Резус-фактор], Рост, Вес,
Table_DoctorsList.ФИОкратко AS [Лечащий врач], [Анамнез жизни], Аллергии,
Table_Login.FIO AS [Кем изменено] FROM Table_PatientsList INNER JOIN
Table_DoctorsList ON Table_PatientsList.[Лечащий врач] = Table_DoctorsList.[ВрачID]
INNER JOIN Table_Login ON Table_PatientsList.[Кем изменено] = Table_Login.LoginID";
           SqlCommand cmd = new SqlCommand(testSqlSelect, connection);
           SqlDataAdapter adapter = new SqlDataAdapter(cmd);
           connection.Open();
           adapter.Fill(dataTable);
        }
        catch (Exception ex)
        {

        }
        finally
        {
           connection.Close();
        }
        return dataTable;
     }

     //Метод для добавления пациента в БД
     public bool Insert(PatientsClass patientsClass)
     {
        bool isSuccess = false;
        try
        {
           string sqlInsert = "INSERT INTO Table_PatientsList ([№ пациента], Категория,
Фото, Фамилия, Имя, Отчество, [Дата рождения], Лет, Месяцев, Пол, [Тип документа], [№
документа], ИИН, Телефон, [Дом. адрес], [Группа крови], [Резус-фактор], Рост, Вес,
[Лечащий врач], [Анамнез жизни], Аллергии, [Кем изменено]) VALUES (@PatientID,
@PatientType, @Photo, @Surname, @Name, @Otchestvo, @BirthDate, @Years, @Months,
@Pol, @DocumentType, @DocumentNo, @IIN, @Telefon, @DomAdres, @GruppaKrovi,
@Rezus, @Rost, @Ves, @LechVrach, @Zabolevaniya, @Allergies, @ModifiedBy)";
           SqlCommand cmd = new SqlCommand(sqlInsert, connection);

           //Использование свойства Parameters для добавления значений переменных в БД
           cmd.Parameters.AddWithValue("@PatientID", patientsClass.PatientID);
           cmd.Parameters.AddWithValue("@PatientType", patientsClass.PatientType);
           cmd.Parameters.AddWithValue("@Photo", patientsClass.Photo);
           cmd.Parameters.AddWithValue("@Surname", patientsClass.Surname);
```

```
            cmd.Parameters.AddWithValue("@Name", patientsClass.Name);
            cmd.Parameters.AddWithValue("@Otchestvo", patientsClass.Otchestvo);
            cmd.Parameters.AddWithValue("@BirthDate", patientsClass.BirthDate);
            cmd.Parameters.AddWithValue("@Years", patientsClass.Years);
            cmd.Parameters.AddWithValue("@Months", patientsClass.Months);
            cmd.Parameters.AddWithValue("@Pol", patientsClass.Pol);
            cmd.Parameters.AddWithValue("@DocumentType", patientsClass.DocumentType);
            cmd.Parameters.AddWithValue("@DocumentNo", patientsClass.DocumentNo);
            cmd.Parameters.AddWithValue("@IIN", patientsClass.IIN);
            cmd.Parameters.AddWithValue("@Telefon", patientsClass.Telefon);
            cmd.Parameters.AddWithValue("@DomAdres", patientsClass.DomAdres);
            cmd.Parameters.AddWithValue("@GruppaKrovi", patientsClass.GruppaKrovi);
            cmd.Parameters.AddWithValue("@Rezus", patientsClass.Rezus);
            cmd.Parameters.AddWithValue("@Rost", patientsClass.Rost);
            cmd.Parameters.AddWithValue("@Ves", patientsClass.Ves);
            cmd.Parameters.AddWithValue("@LechVrach", patientsClass.LechVrach);
            cmd.Parameters.AddWithValue("@Zabolevaniya", patientsClass.Zabolevaniya);
            cmd.Parameters.AddWithValue("@Allergies", patientsClass.Allergies);
            cmd.Parameters.AddWithValue("@ModifiedBy", patientsClass.ModifiedBy);

            connection.Open();
            int rows = cmd.ExecuteNonQuery();
            if (rows > 0)
            {
                isSuccess = true;
            }
            else
            {
                isSuccess = false;
            }
        }
        catch (Exception ex)
        {

        }
        finally
        {
            connection.Close();
        }
        return isSuccess;
    }

    // Метод для удаления пациента из БД
    public bool Delete(PatientsClass patientsClass)
    {
        //Задание значения return по умолчанию равным false
        bool isSuccess = false;
        try
        {
```

```
            //SQL-запрос
            string sqlDelete = "DELETE FROM Table_PatientsList WHERE [№
Пациента]=@PatientID";
            SqlCommand cmd = new SqlCommand(sqlDelete, connection);
            cmd.Parameters.AddWithValue("PatientID", patientsClass.PatientID);
            connection.Open();
            int rows = cmd.ExecuteNonQuery();
            //Запрос выполнен успешно, если количество строк больше нуля, иначе
неуспешно
            if (rows > 0)
            {
               isSuccess = true;
            }
            else
            {
               isSuccess = false;
            }
         }
         catch (Exception ex)
         {

         }
         finally
         {
            connection.Close();
         }
         return isSuccess;
      }

      // Метод для обновления данных пациента в БД
      public bool Update(PatientsClass patientsClass)
      {
         bool isSuccess = false;
         try
         {
            //SQL-запрос
            string sqlUpdate = "UPDATE Table_PatientsList SET [№ пациента]=@PatientID,
Категория=@PatientType, Фото=@Photo, Фамилия=@Surname, Имя=@Name,
Отчество=@Otchestvo, [Дата рождения]=@BirthDate, Лет=@Years, Месяцев=@Months,
Пол=@Pol, [Тип документа]=@DocumentType, [№ документа]=@DocumentNo, ИИН=@IIN,
Телефон=@Telefon, [Дом. адрес]=@DomAdres, [Группа крови]=@GruppaKrovi, [Резус-
фактор]=@Rezus, Рост=@Rost, Вес=@Ves, [Лечащий врач]=@LechVrach, [Анамнез
жизни]=@Zabolevaniya, Аллергии=@Allergies WHERE [№ пациента]=@PatientID";
            SqlCommand cmd = new SqlCommand(sqlUpdate, connection);
            cmd.Parameters.AddWithValue("@PatientID", patientsClass.PatientID);
            cmd.Parameters.AddWithValue("@PatientType", patientsClass.PatientType);
            cmd.Parameters.AddWithValue("@Photo", patientsClass.Photo);
            cmd.Parameters.AddWithValue("@Surname", patientsClass.Surname);
            cmd.Parameters.AddWithValue("@Name", patientsClass.Name);
```

```
            cmd.Parameters.AddWithValue("@Otchestvo", patientsClass.Otchestvo);
```

Продолжение приложения Б

```
            cmd.Parameters.AddWithValue("@BirthDate", patientsClass.BirthDate);
            cmd.Parameters.AddWithValue("@Years", patientsClass.Years);
            cmd.Parameters.AddWithValue("@Months", patientsClass.Months);
            cmd.Parameters.AddWithValue("@Pol", patientsClass.Pol);
            cmd.Parameters.AddWithValue("@DocumentType", patientsClass.DocumentType);
            cmd.Parameters.AddWithValue("@DocumentNo", patientsClass.DocumentNo);
            cmd.Parameters.AddWithValue("@IIN", patientsClass.IIN);
            cmd.Parameters.AddWithValue("@Telefon", patientsClass.Telefon);
            cmd.Parameters.AddWithValue("@DomAdres", patientsClass.DomAdres);
            cmd.Parameters.AddWithValue("@GruppaKrovi", patientsClass.GruppaKrovi);
            cmd.Parameters.AddWithValue("@Rezus", patientsClass.Rezus);
            cmd.Parameters.AddWithValue("@Rost", patientsClass.Rost);
            cmd.Parameters.AddWithValue("@Ves", patientsClass.Ves);
            cmd.Parameters.AddWithValue("@LechVrach", patientsClass.LechVrach);
            cmd.Parameters.AddWithValue("@Zabolevaniya", patientsClass.Zabolevaniya);
            cmd.Parameters.AddWithValue("@Allergies", patientsClass.Allergies);

            connection.Open();

            int rows = cmd.ExecuteNonQuery();
            // Запрос выполнен успешно, если количество строк больше нуля, иначе
неуспешно
            if (rows > 0)
            {
                isSuccess = true;
            }
            else
            {
                isSuccess = false;
            }
        }
        catch (Exception ex)
        {

        }
        finally
        {
            connection.Close();
        }
        return isSuccess;
    }
  }
}

//Код формы PatientForm.cs
using Diploma_Version_1.PatientClasses;
using System;
using System.Configuration;
```

```
using System.Data;
using System.Data.SqlClient;
```

Продолжение приложения Б

```
using System.Drawing;
using System.Windows.Forms;

namespace Diploma_Version_1
{
   public partial class PatientForm : Form
   {
      static string myConnectionString =
ConfigurationManager.ConnectionStrings["connectionString"].ConnectionString;
      SqlConnection connection = new SqlConnection(myConnectionString);
      string imagePath = null;

      public PatientForm(string userName)
      {
         InitializeComponent();
         this.userName = userName;
      }

      string userName;
      int userID = 0;
      string FIO = "";
      string dolzhnost = "";
      int lechVrachID = 0;
      int patientID = 0;
      PatientsClass patientsClass = new PatientsClass();

      private void PatientForm_Load(object sender, EventArgs e)
      {
         if (GetPatientID.patientIDValue != 0)
         {
            btnAddPatient.Visible = false;
            btnUpdatePatient.Visible = true;
            btnDeletePatient.Visible = true;
         }
         // TODO: данная строка кода позволяет загрузить данные в таблицу
"diploma1DataSetDoctorsFIOkratko.Table_DoctorsList". При необходимости она может быть
перемещена или удалена.

this.table_DoctorsListTableAdapter1.Fill(this.diploma1DataSetDoctorsFIOkratko.Table_Doctors
List);
         // TODO: данная строка кода позволяет загрузить данные в таблицу
"diploma1DataSet1.Table_DoctorsList". При необходимости она может быть перемещена или
удалена.
         this.table_DoctorsListTableAdapter.Fill(this.diploma1DataSet1.Table_DoctorsList);
         // TODO: данная строка кода позволяет загрузить данные в таблицу
"diploma1DataSet.Table_PatientsList". При необходимости она может быть перемещена или
удалена.
```

```
        patientsClass.dtGridVPatientList = dtGridVPatientList;
        this.table_PatientsListTableAdapter.Fill(this.diploma1DataSet.Table_PatientsList);
```

Продолжение приложения Б

```
        DataTable dataTable = patientsClass.Select();
        dtGridVPatientList.DataSource = dataTable;
        dtGridVPatientList.Columns[2].AutoSizeMode =
DataGridViewAutoSizeColumnMode.ColumnHeader;
        for (int i = 0; i < dtGridVPatientList.Columns.Count; i++)
        {
            if (dtGridVPatientList.Columns[i] is DataGridViewImageColumn)
            {
                ((DataGridViewImageColumn)dtGridVPatientList.Columns[i]).ImageLayout =
DataGridViewImageCellLayout.Zoom;
            }
        }
        patientID = GetPatientID.patientIDValue;
        if (patientID != 0)
        {
            //Открываем БД и достаём оттуда значения по patientID
            string sqlPatient = "SELECT Категория, Фото, Фамилия, Имя, Отчество, [Дата
рождения], Лет, Месяцев, Пол, [Тип документа], [№ документа], ИИН, Телефон, [Дом.
адрес], [Группа крови], [Резус-фактор], Рост, Вес, Table_DoctorsList.ФИОкратко, [Анамнез
жизни], Аллергии, Table_Login.FIO FROM Table_PatientsList INNER JOIN
Table_DoctorsList ON Table_PatientsList.[Лечащий врач] = Table_DoctorsList.[ВрачID]
INNER JOIN Table_Login ON Table_PatientsList.[Кем изменено] = Table_Login.LoginID
WHERE [№ пациента] = '" + patientID + "'";
            SqlCommand cmdPatient = new SqlCommand(sqlPatient, connection);
            connection.Open();
            string patientType = "";
            Image patientPhoto = null;
            string patientSurname = "";
            string patientName = "";
            string patientOtchestvo = "";
            DateTime patientBirthDate = DateTime.Now;
            int years = 0;
            int months = 0;
            string gender = "";
            string documentType = "";
            string documentNo = "";
            string IIN = "";
            string telefon = "";
            string domAdres = "";
            int gruppaKrovi = 0;
            string rezusFactor = "";
            int rost = 0;
            int ves = 0;
            int lechVrachID = 0;
            string lechVrachKratko = "";
            string anamnez = "";
            string allergies = "";
```

```
int modifiedByID = 0;
string modifiedBy = "";
```

Продолжение приложения Б

```
using (SqlDataReader reader = cmdPatient.ExecuteReader())
{
   while (reader.Read())
   {
      patientType = reader["Категория"].ToString();
      ImageConverter imgConverter = new ImageConverter();
      try
      {
         patientPhoto = (Image)imgConverter.ConvertFrom(reader["Фото"]);
      }
      catch (Exception ex)
      {

      }
      patientSurname = reader["Фамилия"].ToString();
      patientName = reader["Имя"].ToString();
      patientOtchestvo = reader["Отчество"].ToString();
      patientBirthDate = Convert.ToDateTime(reader["Дата рождения"]);
      years = Convert.ToInt32(reader["Лет"]);
      months = Convert.ToInt32(reader["Месяцев"]);
      gender = reader["Пол"].ToString();
      documentType = reader["Тип документа"].ToString();
      documentNo = reader["№ документа"].ToString();
      IIN = reader["ИИН"].ToString();
      telefon = reader["Телефон"].ToString();
      domAdres = reader["Дом. адрес"].ToString();
      gruppaKrovi = Convert.ToInt32(reader["Группа крови"]);
      rezusFactor = reader["Резус-фактор"].ToString();
      rost = Convert.ToInt32(reader["Рост"]);
      ves = Convert.ToInt32(reader["Вес"]);
      lechVrachKratko = reader["ФИОкратко"].ToString();
      anamnez = reader["Анамнез жизни"].ToString();
      allergies = reader["Аллергии"].ToString();
      modifiedBy = reader["FIO"].ToString();
   }
}
connection.Close();

txtBoxPatientNo.Text = patientID.ToString();
cmbBoxPatientType.Text = patientType;
pictureBoxPhoto.Image = patientPhoto;
txtBoxSurname.Text = patientSurname;
txtBoxName.Text = patientName;
txtBoxOtchestvo.Text = patientOtchestvo;
dtTmPckrBirthDate.Value = patientBirthDate;
txtBoxYears.Text = years.ToString();
txtBoxMonths.Text = months.ToString();
```

```
            if (gender == "Муж.")
            {
```

Продолжение приложения Б

```
               checkBoxMale.CheckState = CheckState.Checked;
            }
            else
            {
               checkBoxFemale.CheckState = CheckState.Checked;
            }
            cmbBoxDocumentType.Text = documentType;
            txtBoxDocumentNo.Text = documentNo;
            txtBoxIIN.Text = IIN;
            txtBoxTelefon.Text = telefon;
            richTextBoxAdres.Text = domAdres;
            cmbBoxGruppaKrovi.Text = gruppaKrovi.ToString();
            cmbBoxRezus.Text = rezusFactor;
            txtBoxRost.Text = rost.ToString();
            txtBoxVes.Text = ves.ToString();
            cmbBoxLechVrach.Text = lechVrachKratko;
            richTextBoxPerenZabolevaniya.Text = anamnez;
            richTextBoxAllergies.Text = allergies;
            labelModifiedBy.Text = "Изменено сотрудником: " + modifiedBy;
         }
      }
      private void buttonChangeFhoto_Click(object sender, EventArgs e)
      {
         using (OpenFileDialog openFileDialog = new OpenFileDialog())
         {
            openFileDialog.Filter = "*.JPG; *.JPEG; *.PNG; *.GIF; *.BMP; *.ICO|*.JPG; *.JPEG;
*.PNG; *.GIF; *.BMP; *.ICO|All files (*.*)|*.*";
            if (openFileDialog.ShowDialog() == DialogResult.OK)
            {
               try
               {
                  pictureBoxPhoto.Image = Image.FromFile(openFileDialog.FileName);
                  imagePath = openFileDialog.FileName;
                  patientsClass.FileName = imagePath;
                  pictureBoxPhoto.Invalidate();
               }
               catch
               {
                  DialogResult dResult = MessageBox.Show("Невозможно открыть выбранный
файл", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
               }
            }
         }
      }
      private void txtBoxFindPatient_TextChanged_2(object sender, EventArgs e)
      {
         //Get the value from textbox
```

```
        string keyword = txtBoxFindPatient.Text;
        if (cmbBoxFindPatientBy.Text == "№ пациента")
```

Продолжение приложения Б

```
        {
            SqlDataAdapter sqlDataAdapter = new SqlDataAdapter("SELECT * FROM
Table_PatientsList WHERE [№ пациента] LIKE '%" + keyword + "%'", connection);
            DataTable dt = new DataTable();
            sqlDataAdapter.Fill(dt);
            dtGridVPatientList.DataSource = dt;

        }
        else if (cmbBoxFindPatientBy.Text == "фамилии")
        {
            SqlDataAdapter sqlDataAdapter = new SqlDataAdapter("SELECT * FROM
Table_PatientsList WHERE Фамилия LIKE '%" + keyword + "%'", connection);
            DataTable dt = new DataTable();
            sqlDataAdapter.Fill(dt);
           dtGridVPatientList.DataSource = dt;

        }
        else if (cmbBoxFindPatientBy.Text == "году рождения")
        {
            SqlDataAdapter sqlDataAdapter = new SqlDataAdapter("SELECT * FROM
Table_PatientsList WHERE [Дата рождения] LIKE '%" + keyword + "%'", connection);
            DataTable dt = new DataTable();
            sqlDataAdapter.Fill(dt);
            dtGridVPatientList.DataSource = dt;

        }
        else if (cmbBoxFindPatientBy.Text == "возрасту (полных лет)")
        {
            SqlDataAdapter sqlDataAdapter = new SqlDataAdapter("SELECT * FROM
Table_PatientsList WHERE Лет LIKE '%" + keyword + "%'", connection);
            DataTable dt = new DataTable();
            sqlDataAdapter.Fill(dt);
            dtGridVPatientList.DataSource = dt;

        }
        else if(cmbBoxFindPatientBy.Text == "полу")
        {
            SqlDataAdapter sqlDataAdapter = new SqlDataAdapter("SELECT * FROM
Table_PatientsList WHERE Пол LIKE '%" + keyword + "%'", connection);
            DataTable dt = new DataTable();
            sqlDataAdapter.Fill(dt);
            dtGridVPatientList.DataSource = dt;

        }
        else if (cmbBoxFindPatientBy.Text == "лечащему врачу")
        {
```

```
            SqlDataAdapter sqlDataAdapter = new SqlDataAdapter("SELECT * FROM
Table_PatientsList WHERE [Лечащий врач] LIKE '%" + keyword + "%'", connection);
            DataTable dt = new DataTable();
```

Продолжение приложения Б

```
            sqlDataAdapter.Fill(dt);
            dtGridVPatientList.DataSource = dt;

        }
        else if (cmbBoxFindPatientBy.Text == "ИИН")
        {
            SqlDataAdapter sqlDataAdapter = new SqlDataAdapter("SELECT * FROM
Table_PatientsList WHERE ИИН LIKE '%" + keyword + "%'", connection);
            DataTable dt = new DataTable();
            sqlDataAdapter.Fill(dt);
            dtGridVPatientList.DataSource = dt;

        }
        else if (cmbBoxFindPatientBy.Text == "группе крови")
        {
            SqlDataAdapter sqlDataAdapter = new SqlDataAdapter("SELECT * FROM
Table_PatientsList WHERE [Группа крови] LIKE '%" + keyword + "%'", connection);
            DataTable dt = new DataTable();
            sqlDataAdapter.Fill(dt);
            dtGridVPatientList.DataSource = dt;

        }
        else if (cmbBoxFindPatientBy.Text == "резус-фактору")
        {
            SqlDataAdapter sqlDataAdapter = new SqlDataAdapter("SELECT * FROM
Table_PatientsList WHERE [Резус-фактор] LIKE '%" + keyword + "%'", connection);
            DataTable dt = new DataTable();
            sqlDataAdapter.Fill(dt);
            dtGridVPatientList.DataSource = dt;

        }
        else if (cmbBoxFindPatientBy.Text == "заболеваниям")
        {
            SqlDataAdapter sqlDataAdapter = new SqlDataAdapter("SELECT * FROM
Table_PatientsList WHERE [Анамнез жизни] LIKE '%" + keyword + "%'", connection);
            DataTable dt = new DataTable();
            sqlDataAdapter.Fill(dt);
            dtGridVPatientList.DataSource = dt;
        }
    }

    private void btnFindPatient_Click(object sender, EventArgs e)
    {
        Form1 form1 = new Form1();
        form1.Show();
    }
```

```
private void GetInfoFromDB()
{
```

Продолжение приложения Б

```
    string sqlTableLogin = "SELECT LoginID, FIO, Specialization FROM Table_Login
WHERE UserName = '" + userName + "'";
    SqlCommand cmd1 = new SqlCommand(sqlTableLogin, connection);
    string FIOkratko = cmbBoxLechVrach.Text;
    string sqlTableDoctorsList = "SELECT ВрачID FROM Table_DoctorsList WHERE
ФИОкратко = '" + FIOkratko + "'";
    SqlCommand cmd2 = new SqlCommand(sqlTableDoctorsList, connection);
    connection.Open();
    using (SqlDataReader reader1 = cmd1.ExecuteReader())
    {
      while (reader1.Read())
      {
        userID = Convert.ToInt32(reader1["LoginID"]);
        FIO = reader1["FIO"].ToString();
        dolzhnost = reader1["Specialization"].ToString();
      }
    }
    using (SqlDataReader reader2 = cmd2.ExecuteReader())
    {
      while (reader2.Read())
      {
        lechVrachID = Convert.ToInt32(reader2["ВрачID"]);
      }
    }
    connection.Close();
  }

  private void btnAddPatient_Click(object sender, EventArgs e)
  {
    //Get the values from input fields and SEND them to PatiensClass
    GetInfoFromDB();
    if (dolzhnost != "оператор" && dolzhnost != "администратор")
    {
      MessageBox.Show("Для добавления пациента нужно обратиться в регистратуру");
    }
    else
    {
      try
      {
        patientsClass.PatientID = Convert.ToInt32(txtBoxPatientNo.Text);
      }
      catch (Exception ex)
      {
        MessageBox.Show("Введите № пациента");
        return;
      }
```

```
            patientsClass.PatientType = cmbBoxPatientType.Text;
            Image photo = pictureBoxPhoto.Image;
            byte[] arr;
```

Продолжение приложения Б

```
            ImageConverter imgConverter = new ImageConverter();
            arr = (byte[])imgConverter.ConvertTo(photo, typeof(byte[]));
            patientsClass.Photo = arr;

            patientsClass.Surname = txtBoxSurname.Text;
            patientsClass.Name = txtBoxName.Text;
            patientsClass.Otchestvo = txtBoxOtchestvo.Text;
            patientsClass.BirthDate = dtTmPckrBirthDate.Value;
            DateTime currentDate = DateTime.Now;
            DateTime BirthDate = dtTmPckrBirthDate.Value;
            int countYears = currentDate.Year - BirthDate.Year;
            int countMonths = currentDate.Month - BirthDate.Month;
            int countDays = currentDate.Day - BirthDate.Day;
            int Years = 0;
            int Months = 0;

            //Вычисление возраста пациента в годах
            if ((countYears == 0) || (countYears == 1 && countMonths < 0) || (countYears == 1 &&
countMonths == 0 && countDays < 0))
            {
               Years = 0;
            }
            else if (countYears == 1 && countMonths == 0 && countDays == 0)
            {
               Years = 1;
            }
            else
            {
               Years = countYears;
            }

            // Вычисление возраста пациента в месяцах
            if (countMonths > 0 && countDays >= 0)
            {
               Months = countMonths;
            }
            else if (countMonths > 0 && countDays < 0)
            {
               Months = countMonths - 1;
            }
            else if (countMonths < 0 && countDays >= 0)
            {
               Months = 12 + countMonths;
            }
            else if (countMonths < 0 && countDays < 0)
            {
```

```
    Months = 11 + countMonths;
}
else if (countMonths == 0 && countDays < 0)
```

Продолжение приложения Б

```
{
    Months = 11;
}
else
{
    Months = 0;
}
patientsClass.Years = Years;
patientsClass.Months = Months;

if (checkBoxMale.Checked)
{
    patientsClass.Pol = checkBoxMale.Text;
}
else
{
    patientsClass.Pol = checkBoxFemale.Text;
}
patientsClass.DocumentType = cmbBoxDocumentType.Text;
patientsClass.DocumentNo = txtBoxDocumentNo.Text;
patientsClass.IIN = txtBoxIIN.Text;
patientsClass.Telefon = txtBoxTelefon.Text;
patientsClass.DomAdres = richTextBoxAdres.Text;
try
{
    patientsClass.GruppaKrovi = Convert.ToInt32(cmbBoxGruppaKrovi.Text);
}
catch (Exception ex)
{
    MessageBox.Show("Введите группу крови пациента");
    return;
}
patientsClass.Rezus = cmbBoxRezus.Text;
try
{
    patientsClass.Rost = Convert.ToInt32(txtBoxRost.Text);
}
catch (Exception ex)
{
    MessageBox.Show("Введите рост пациента");
    return;
}
try
{
    patientsClass.Ves = Convert.ToInt32(txtBoxVes.Text);
}
```

```
            catch (Exception ex)
            {
                MessageBox.Show("Введите вес пациента");
```

Продолжение приложения Б

```
                return;
            }
            patientsClass.Zabolevaniya = richTextBoxPerenZabolevaniya.Text;
            patientsClass.Allergies = richTextBoxAllergies.Text;
            patientsClass.ModifiedBy = userID;
            try
            {
                patientsClass.LechVrach = lechVrachID;
            }
            catch (Exception ex)
            {
                MessageBox.Show("Введите лечащего врача пациента");
                return;
            }

            //Inserting data into database
            bool success = patientsClass.Insert(patientsClass);
            if (success == true)
            {
                btnAddPatient.Visible = false;
                MessageBox.Show("Новый пациент успешно внесён в базу данных");
                //Load data on datadridview
                DataTable dataTable = patientsClass.Select();
                dtGridVPatientList.DataSource = dataTable;
                txtBoxYears.Text = Years.ToString();
                txtBoxMonths.Text = Months.ToString();
                btnAddPatient.Visible = false;
                btnUpdatePatient.Visible = true;
                btnDeletePatient.Visible = true;
            }
            else
            {
                MessageBox.Show("Пациент с таким номером уже существует. Попробуйте ещё
раз");
            }
        }
    }
    private void checkBoxMale_CheckedChanged(object sender, EventArgs e)
    {
        if (checkBoxMale.Checked)
        {
            checkBoxFemale.CheckState = CheckState.Unchecked;
        }
    }

    private void checkBoxFemale_CheckedChanged(object sender, EventArgs e)
```

```
{
   if (checkBoxFemale.Checked)
   {
```



```
      checkBoxMale.CheckState = CheckState.Unchecked;
   }
}
private void btnUpdatePatient_Click(object sender, EventArgs e)
{
   GetInfoFromDB();
   //Get the values from input fields and SEND them to PatiensClass
   patientsClass.PatientID = Convert.ToInt32(txtBoxPatientNo.Text);
   patientsClass.PatientType = cmbBoxPatientType.Text;

   Image photo = pictureBoxPhoto.Image;
   byte[] arr;
   ImageConverter imgConverter = new ImageConverter();
   arr = (byte[])imgConverter.ConvertTo(photo, typeof(byte[]));
   patientsClass.Photo = arr;

   patientsClass.Surname = txtBoxSurname.Text;
   patientsClass.Name = txtBoxName.Text;
   patientsClass.Otchestvo = txtBoxOtchestvo.Text;
   patientsClass.BirthDate = dtTmPckrBirthDate.Value;
   patientsClass.Years = Years;
   patientsClass.Months = Months;

   if (checkBoxMale.Checked)
   {
      patientsClass.Pol = checkBoxMale.Text;
   }
   else
   {
      patientsClass.Pol = checkBoxFemale.Text;
   }
   patientsClass.DocumentType = cmbBoxDocumentType.Text;
   patientsClass.DocumentNo = txtBoxDocumentNo.Text;
   patientsClass.IIN = txtBoxIIN.Text;
   patientsClass.Telefon = txtBoxTelefon.Text;
   patientsClass.DomAdres = richTextBoxAdres.Text;
   patientsClass.GruppaKrovi = Convert.ToInt32(cmbBoxGruppaKrovi.Text);
   patientsClass.Rezus = cmbBoxRezus.Text;
   patientsClass.Rost = Convert.ToInt32(txtBoxRost.Text);
   patientsClass.Ves = Convert.ToInt32(txtBoxVes.Text);
   patientsClass.LechVrach = lechVrachID;
   patientsClass.Zabolevaniya = richTextBoxPerenZabolevaniya.Text;
   patientsClass.Allergies = richTextBoxAllergies.Text;

   //Update data in database
   bool success = patientsClass.Update(patientsClass);
```

```
    if (success == true)
    {
        //Updated successfully
```

Продолжение приложения Б

```
        MessageBox.Show("Данные пациента успешно обновлены");
        //Load data on datagridview
        DataTable dt = patientsClass.Select();
        dtGridVPatientList.DataSource = dt;
    }
    else
    {
        MessageBox.Show("Не удалось обновить данные пациента. Попробуйте ещё раз.");
    }
}

private void btnDeletePatient_Click(object sender, EventArgs e)
{
    //Get PatientID from textbox
    patientsClass.PatientID = Convert.ToInt32(txtBoxPatientNo.Text);
    bool success = patientsClass.Delete(patientsClass);
    if (success == true)
    {
        MessageBox.Show("Пациент успешно удалён из базы данных");
        //Refresh dataGridView
        //Load data on datagridview
        DataTable dt = patientsClass.Select();
        dtGridVPatientList.DataSource = dt;
        //Clear input fields
        Clear();
        btnAddPatient.Visible = true;
        btnUpdatePatient.Visible = false;
        btnDeletePatient.Visible = false;
    }
    else
    {
        MessageBox.Show("Не удалось удалить пациента. Попробуйте ещё раз");
    }
}

public void Clear()
{
    labelModifiedBy.Text = "";
    txtBoxPatientNo.Text = "";
    cmbBoxPatientType.Text = "";
    pictureBoxPhoto.Image = Properties.Resources.Photo;
    txtBoxSurname.Text = "";
    txtBoxName.Text = "";
    txtBoxOtchestvo.Text = "";
    dtTmPckrBirthDate.Value = DateTime.Now;
    txtBoxYears.Text = "";
```

```
            txtBoxMonths.Text = "";
            checkBoxFemale.CheckState = CheckState.Unchecked;
            checkBoxMale.CheckState = CheckState.Unchecked;
```

Продолжение приложения Б

```
            cmbBoxDocumentType.Text = "";
            txtBoxDocumentNo.Text = "";
            txtBoxIIN.Text = "";
            txtBoxTelefon.Text = "";
            richTextBoxAdres.Text = "";
            cmbBoxGruppaKrovi.Text = "";
            cmbBoxRezus.Text = "";
            txtBoxRost.Text = "";
            txtBoxVes.Text = "";
            cmbBoxLechVrach.Text = "";
            richTextBoxPerenZabolevaniya.Text = "";
            richTextBoxAllergies.Text = "";
        }

        private void btnClearFields_Click(object sender, EventArgs e)
        {
            Clear();
            btnAddPatient.Visible = true;
            btnUpdatePatient.Visible = false;
            btnDeletePatient.Visible = false;
            GetPatientID.patientIDValue = 0;
        }
        private void dtGridVPatientList_RowHeaderMouseClick(object sender,
DataGridViewCellMouseEventArgs e)
        {
            //Identify the row in which mouse is clicked
            int rowIndex = e.RowIndex;

            //Get data from DataGridView and load it to textboxes
            txtBoxPatientNo.Text = dtGridVPatientList.Rows[rowIndex].Cells[0].Value.ToString();
            cmbBoxPatientType.Text =
dtGridVPatientList.Rows[rowIndex].Cells[1].Value.ToString();
            ImageConverter imgConverter = new ImageConverter();
            try
            {
                Image photo =
(Image)imgConverter.ConvertFrom(dtGridVPatientList.Rows[rowIndex].Cells[2].Value);
                pictureBoxPhoto.Image = photo;
            }
            catch (Exception ex)
            {

            }
            txtBoxSurname.Text = dtGridVPatientList.Rows[rowIndex].Cells[3].Value.ToString();
            txtBoxName.Text = dtGridVPatientList.Rows[rowIndex].Cells[4].Value.ToString();
            txtBoxOtchestvo.Text = dtGridVPatientList.Rows[rowIndex].Cells[5].Value.ToString();
```

```
        dtTmPckrBirthDate.Value =
Convert.ToDateTime(dtGridVPatientList.Rows[rowIndex].Cells[6].Value);
        txtBoxYears.Text = dtGridVPatientList.Rows[rowIndex].Cells[7].Value.ToString();
```

Продолжение приложения Б

```
        txtBoxMonths.Text = dtGridVPatientList.Rows[rowIndex].Cells[8].Value.ToString();
        if (dtGridVPatientList.Rows[rowIndex].Cells[9].Value.ToString() == "Муж.")
        {
           checkBoxMale.CheckState = CheckState.Checked;
        }
        else
        {
           checkBoxFemale.CheckState = CheckState.Checked;
        }
        cmbBoxDocumentType.Text =
dtGridVPatientList.Rows[rowIndex].Cells[10].Value.ToString();
        txtBoxDocumentNo.Text =
dtGridVPatientList.Rows[rowIndex].Cells[11].Value.ToString();
        txtBoxIIN.Text = dtGridVPatientList.Rows[rowIndex].Cells[12].Value.ToString();
        txtBoxTelefon.Text = dtGridVPatientList.Rows[rowIndex].Cells[13].Value.ToString();
        richTextBoxAdres.Text =
dtGridVPatientList.Rows[rowIndex].Cells[14].Value.ToString();
        cmbBoxGruppaKrovi.Text =
dtGridVPatientList.Rows[rowIndex].Cells[15].Value.ToString();
        cmbBoxRezus.Text = dtGridVPatientList.Rows[rowIndex].Cells[16].Value.ToString();
        txtBoxRost.Text = dtGridVPatientList.Rows[rowIndex].Cells[17].Value.ToString();
        txtBoxVes.Text = dtGridVPatientList.Rows[rowIndex].Cells[18].Value.ToString();
        cmbBoxLechVrach.Text =
dtGridVPatientList.Rows[rowIndex].Cells[19].Value.ToString();
        richTextBoxPerenZabolevaniya.Text =
dtGridVPatientList.Rows[rowIndex].Cells[20].Value.ToString();
        richTextBoxAllergies.Text =
dtGridVPatientList.Rows[rowIndex].Cells[21].Value.ToString();
        labelModifiedBy.Text = "Изменено сотрудником: " +
dtGridVPatientList.Rows[rowIndex].Cells[22].Value.ToString();

        btnUpdatePatient.Visible = true;
        btnDeletePatient.Visible = true;
        btnAddPatient.Visible = false;

        GetInfoFromDB();
        int patientID =
Convert.ToInt32(this.dtGridVPatientList.Rows[rowIndex].Cells[0].Value.ToString());
        GetPatientID.patientIDValue = patientID;
    }

    private void buttonDeletePhoto_Click(object sender, EventArgs e)
    {
        pictureBoxPhoto.Image = Properties.Resources.Photo;
    }
```

```
        private void buttonExit_Click(object sender, EventArgs e)
        {
            Application.Exit();
```

Продолжение приложения Б

```
        }

        private void buttonBack_Click(object sender, EventArgs e)
        {
            MainForm mainForm = new MainForm(userName);
            this.Hide();
            mainForm.Show();
        }
    }
}
```

Приложение В
Акт внедрения

ТОВАРИЩЕСТВО С ОГРАНИЧЕННОЙ ОТВЕТСТВЕННОСТЬЮ

БИН 180 740 022 249
KZ199470398991392266 в
АО «ДБ» АЛЬФА-БАНК»
БИК ALFAKZKA

Uniserv
MEDICAL CENTER

090000, Республика
Казахстан, Западно-
Казахстанская область, г.
Уральск, ул. Шолохова, 36

АКТ внедрения программы «Электронная картотека “Uniserv MC”»

Настоящий Акт свидетельствует, что «Электронная картотека “Uniserv MC”», разработанная Рудич Екатериной Алексеевной, внедрена в ТОО «Uniserv Medical Center».

Процесс внедрения проходил с 11 по 15 мая 2020 г.

Заявленные характеристики системы предполагают наличие следующих основных возможностей:

- ведение электронной картотеки пациентов;
- запись пациентов на приём;
- ведение статистики посещений.

В ходе эксплуатации электронной картотеки было подтверждено, что она обладает всеми заявленными возможностями и позволяет активно ей пользоваться.

На момент подписания настоящего Акта база данных установлена на сервере компании.

Генеральный директор
ТОО «Uniserv Medical Center» **Булкашева С.Б.**

20.05.2020 г.

ISO 9001 Management system certified | ISO 14001 Management system certified | OHSAS 18001 Management system certified | SA 8000 Management system certified