

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

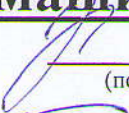
«Вычислительная техника»

Направление 09.04.01 «Информатика и вычислительная техника»
подготовки:
Магистерская «Вычислительные машины, комплексы, системы и
программа: сети»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
МАГИСТРА

на тему

Информационно-диагностическая система с
человеко-машинным интерфейсом

Студент  16.06.2020 г. Жуков Владислав Игоревич
(подпись, дата) (ФИО полностью)

Руководитель  16.06.2020 г. Зинкин С. А.
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр  Кучин А. В.
(подпись, дата) (фамилия, инициалы)

Рецензент  16.06.2020 г. Севастьянов А. В.
(подпись, дата) (фамилия, инициалы)

и. о. специалиста ООО «Бизнес-процессы»
(должность, место работы)

Работа допущена к защите (протокол зас. кафедры от 03.06.2020 г. № 12)

Заведующий кафедрой  Митрохин М. А.
(подпись) (фамилия, инициалы)

Работа защищена с отметкой  (протокол заседания ГЭК от
16.06.20 № 12)

Секретарь ГЭК  Гурин Е. И.
(подпись) (фамилия, инициалы)

Пенза 2020

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
 М. А. Митрохин
«30» сентября 2019 г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

1. Студент Жуков Владислав Игоревич гр. 18ВВм1 факультета вычислительной техники
2. Тема работы «Информационно-диагностическая система с человеко-машинным интерфейсом».
Тема утверждена приказом ПГУ № 1039/о от « 30 » сентября 2019 г.
3. Руководитель работы д. т. н., профессор каф. ВТ ПГУ Зинкин С.А.
4. Задание на работу (назначение разработки, исходные данные и т.п.)
Назначение разработки: программа, позволяющая создавать сохранять и открывать базы данных состояния пациентов.
5. Перечень подлежащих разработке вопросов.
Необходимо проанализировать ситуацию на рынке приложений для разных сфер деятельности, изучить вопрос разработки распределенной системы, рассмотреть технологии разработки и отладки программ, изучить психологию труда врача и пациента.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Анализ предметной области	17	22.03.2020	
2.	Изучение методов диспетчеризации очередей	15	06.04.2020	
3.	Разработка модели измерителя битовой скорости	15	19.04.2020	
4.	Разработка модели диспетчеризации очередей на основе измерения мгновенной битовой скорости трафиков различных классов QoS	15	07.05.2020	
5.	Отладка и тестирование	15	07.05.2020	
6.	Составление выводов	11	08.05.2020	
7.	Оформление выпускной работы	12	16.06.2020	

Дата выдачи задания «02» сентября 2019 г.

Руководитель ВКР

(подпись, дата)

Зинкин С. А.

2.09.19

Задание к исполнению принял студент

(подпись, дата)

Жуков В. И.

Работу к защите допустить
Декан факультета

(подпись, дата)

Фионова Л. Р.

РЕФЕРАТ

Пояснительная записка содержит 125 листов формата А4, 22 рисунка, 9 источников, 2 приложения.

Информационно-диагностическая система с человеко-машинным интерфейсом.

C#, MICROSOFT SQL SERVER, БАЗА ДАННЫХ, МЕДИЦИНА, .NET FRAMEWORK

Объектами исследования являются: технология проектирования и отладки программы; централизованная база данных.

Цель работы – разработка сетевой клиент-серверной системы, предназначенной для ведения электронной истории болезни.

В результате проделанной работы был создан проект, который удовлетворяет поставленной задаче.

В пояснительной записке приведен обзор методов построения систем телемедицины, обзор технологий разработки и отладки приложений.

СОДЕРЖАНИЕ

Перечень сокращений и обозначений	5
Введение	6
1 АНАЛИЗ ЗАДАЧИ И ОБОСНОВАНИЕ СПОСОБА ЕЕ РЕШЕНИЯ	7
1.1 Анализ предметной области	7
1.2 История и обзор аналогичных систем	9
1.2.1 Зарубежный опыт	9
1.2.2 Концепт телемедицины	12
1.2.3 Преимущества телемедицины для пациентов и медицинских работников	15
1.2.4 Примеры проектов	17
1.2.5 Международные информационные сети	22
1.3 Анализ и обоснование выбранной платформы	23
1.3.1 Выбор архитектуры системы	23
1.3.2 Выбор аппаратно-программной платформы	25
1.3.3 Выбор технологий и средств разработки базы данных	28
Вывод	28
2 Разработка распределенной системы	30
2.1 Принципы CASE-технологии	30
2.2 Анализ вариантов использования системы	30
2.3 Определение основных состояний системы	32
2.3.1 Начальное состояние системы	32
2.3.2 Определение состояний администраторского и пользовательского модулей	33
2.4 Определение динамического взаимодействия компонент	34
2.4.1 Запрос таблицы для просмотра и модификации	35
2.5 Описания функций передачи, обработки и доступа к данным	36
2.6 Разработка подсистемы взаимодействия с клиентом	38
2.6.1 Функции подсистемы взаимодействия с клиентом	38

2.6.2 Раздел «База пациентов»	39
2.6.3 Модуль «Пациент».....	40
2.6.4 Форма для печати	50
2.6.5 Экспорт в Excel.....	51
3 Технология разработки и отладки программы.....	53
Введение.....	53
3.1 Технология разработки программы.	53
3.2 Технология отладки программы.....	56
3.3 Инструментальные средства разработки и отладки.	57
3.4 База данных.....	64
3.5 Преимущества Microsoft SQL Server	65
Вывод.....	66
4 Психология труда врача и пациента при применении в работе Электронной карты.....	67
Введение.....	67
4.1 Средства труда или средства производства	69
4.2 Методы психологии труда	72
4.3 Задачи психологии труда	75
Вывод.....	77
Заключение	79
Список использованных источников	80
Приложение А Листинг программы.....	81
Приложение В Презентация.....	122

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

В настоящей работе применяют следующие сокращения и обозначения:

Бекан – Резервная копия.

НСД – Несанкционированный доступ.

Поток – «упрощенный процесс», запуск и выполнение которого осуществляются значительно быстрее по сравнению с обычным системным процессом, запускаемым при выполнении операции ветвления или разделения.

UML (Unified Modeling Language) – унифицированный язык моделирования.

ВВЕДЕНИЕ

В настоящее время в связи с участвовавшими эпидемиями и общим уровнем медицины в стране существует множество веских причин перевода существующей структуры работы медицинских учреждений на компьютерную основу, где под общим уровнем понимается не уровень медицины в крупных городах, а в отдаленных населенных пунктах. В настоящее время с развитием технологий и появления новых методов сбора, хранения и обработки информации имеет место быть переход сферы медицины на электронные носители информации.

Несомненно, стоит понимать что в связи со сложившейся ситуацией, когда большая часть медицинских услуг постепенно начинает переходить к частным собственникам, ведение ведомостей, а в общем, и централизация всего процесса оказания медицинских услуг является трудновыполнимой задачей, ведь каждому отдельному специалисту необходимы свои собственные инструменты для работы, подходящие в основном узкому кругу лиц.

И даже если сделать эту систему распределенной, то есть иметь возможность контролировать весь процесс оказания медицинских услуг, эту систему нужно разработать, а на это уйдёт не один год работы достаточно крупной компании, а также не один год консультации со специалистами, ведь конечный продукт будет отвечать за человеческие жизни. Да и по завершении этого проекта его нужно будет поддерживать, постоянно обновлять, то есть заниматься его сопровождением, а это так же довольно затратное мероприятие и займет очень много времени.

Поэтому в своем дипломном проекте я подробно разобрал, что необходимо знать для создания системы ведения электронной истории болезни, расписал некоторые технологии для разработки.

1 АНАЛИЗ ЗАДАЧИ И ОБОСНОВАНИЕ СПОСОБА ЕЕ РЕШЕНИЯ

1.1 Анализ предметной области

Если трезво оценить ситуацию в современном мире, и обратить внимание на последние события, становится понятно, что сейчас люди очень бережно относятся к своему здоровью, а государство старается всячески помочь и облегчить получение медицинских услуг простому населению. Для этого применяется новейшее оборудование, привлекается все большее количество специалистов, в быстром темпе строятся медицинские учреждения. И в каждом таком новом центре пользуются персональными компьютерами или терминалами для составления карт пациентов.

Электронная история болезни — это совокупная информация о пациенте, которая составляется и хранится в автоматизированной информационной базе данных медицинского учреждения и их сети.

Преимущества электронной больничной карты по сравнению с бумажной очевидны: множество специалистов имеют к ней доступ, при правильной организации системы и использовании бекапов данные, хранящиеся на электронных носителях являются более устойчивыми к ситуациям с потерей информации.

Электронная медицинская карта позволяет практически полностью отказаться от бумажных носителей информации и переходить исключительно на электронные документы, сохраняя лишь небольшую часть документов, которые являются обязательными из-за правовых и финансовых ограничений, которые вскоре могут также перейти в электронный формат.

Реализация права человека на получение квалифицированной медицинской помощи в любом месте и в любое время является главной задачей системы.

В систему входят:

- медицинские организации с их профессиональными и информационными, образовательными ресурсами, медицинскими диагностическими приборами, базами данных, а также пользователями системы,
- технические средства доступа к телекоммуникационным сетям,
- каналы связи и сетевые средства,
- датчики и другие преобразователи медицинской информации в цифровые электрические сигналы для передачи по каналам связи.

Задачи электронной медицинской карты:

- Профилактическое обслуживание населения.
- Предоставляет лечащему врачу возможность быстро и удобно вносить информацию о пациенте.
- Обеспечивает безопасность доступа к ЕМС с учетом прав доступа пользователей к медицинской информации, утвержденных медицинским учреждением.
- Позволяет просматривать ЭМС пациента и быстро находить необходимую информацию в больших объемах медицинской документации.
- Позволяет создавать на основе ЕМС различные выписки, сертификаты, эпикризы, распечатывать их и хранить копии этих документов.
- Обеспечивает возможность визуального просмотра медицинских данных пациента: диагнозы, список рецептов, составление различных графиков и т.п.
- Упрощение и ускорение процесса диагностики, сбора данных пациента, назначения лечения.
- Просмотр и редактирование карт пациентов с любого компьютера или телефона, в любой точке мира.

- Учет всех жалоб, симптомов, анализов и выполненных процедур. Запись результатов.
- Повышение уровня обслуживания.
- Исследования, результаты тестов и другая медицинская информация, в результате которой были созданы различные профессии, специально предназначенные для врачей различных специальностей.
- Модуль ЕМС / электронная история болезни поставляется с готовыми формами ввода, разработанными совместно с врачом и отлаженными в течение многих лет использования системы в медицинских учреждениях.

Система предоставляет инструменты, предназначенные для ускорения сбора текстовой информации.

- Режим автозаполнения текста позволяет использовать функцию адресной строки популярных браузеров: автоматически добавлять слова из каталога при вводе текста.

- Назначение лекарств по рецепту может производиться по шаблонам, в которые пользователь вводит только необходимые параметры: например, дозировку и частоту приема.

- Инструмент замещения макросов позволяет копировать данные из предыдущих записей истории болезни, а также облегчает ввод той же информации (протоколов операций, медосмотров и т.п.).

1.2 История и обзор аналогичных систем

1.2.1 Зарубежный опыт

Задачи исследований и разработок в области электронных карт пациентов состоят в том, чтобы объединить информационные и телекоммуникационные технологии таким образом, чтобы службы

здравоохранения могли систематически использовать медицинские ресурсы за пределами местной организации.

За рубежом проводится большой объем исследований в области электронных карт для медицинских учреждений, также называемых телемедициной, и все это финансируется непосредственно правительством США. Более того, он финансируется из бюджета Министерства здравоохранения и Министерства обороны. В этом случае наиболее интенсивные результаты исследований в области телемедицины внедряются в подразделения Министерства обороны США.

Значительные результаты по телемедицине были получены по европейским научным программам RACE, ESPRIT, DELTA и др.

В 1970 г. в США были запущены исследовательские программы по медицине, в том числе программы, пригодные для использования в сельской местности и местах, где возникали сложности с использованием высокоскоростных сетей.

При этом была продемонстрирована эффективность применения телемедицины при решении медицинских проблем в экстремальных ситуациях, координации методов лечения и др.

По данным Европейской комиссии, телемедицину можно определить как предоставление медицинских услуг посредством использования технологий, позволяющих получать данные врачу от пациента, который находится на удалении от специалиста, с использованием технологий связи.

Среди научных проектов последних лет можно отметить:

Проект по отслеживанию состояния солдата используя GPS.

Проект создания сети телемедицины в условиях подавления сигнала.

Проверка системы для передачи в реальном времени жизненно важных данных от пациентов скорой помощи в Центральную травматологическую больницу в Мэриленде по цифровым каналам системы сотовой связи.

Проект обеспечения интерактивных видеоконференций и электронной передачи изображений между медицинскими учреждениями ВМС США.

Консультации по дерматологии, неврологии, ортопедии, кардиологии, офтальмологии и др.

Проект сотрудничества сети телемедицины ВМС США с образовательной сетью учебных заведений и проектом по диагностической радиологии, хранению и передаче изображений.

Проект использования ресурсов спутниковой сети ИМАРСАТ для предоставления услуг подразделениям морской пехоты в боевой группе.

Проект "Deer Freeze", предполагающий осуществление услуг телемедицины для Антарктики в случаях проведения рискованных операций.

Проект по созданию медицинской диагностической системы на основе цифровой записи и обработки рентгенологических изображений, обеспечивающий высокоскоростной обмен данными по сетям Ethernet.

Проект по объединению телекоммуникационных и информационных ресурсов тихоокеанского региона в интересах телемедицины.

Проект по разработке и внедрению телемедицинского терминального оборудования для небольших медицинских отделений.

Проект по созданию сети телемедицины для армии США на базе медицинских центров в Каролине, Техасе и др.

Проект Tele-Dental Care для североамериканского континента.

Проект организации доступа в Интернет к графическим файлам по медицинской тематике с целью консультирования и обучения.

Проект телемедицинских консультаций в глобальном масштабе с использованием ресурсов спутниковой системы Inmarsat, телефонных сетей общего пользования и т. Д. Услуги телемедицины были предоставлены Соединенными Штатами в Хорватии, Сомали, Македонии, Кувейте, Таити и других.

В Европе развитие информационных технологий используется для реализации задач телемедицины. Большая часть лабораторного и радиологического оборудования компьютеризирована, представлена концепция интеграции информационных систем, что позволяет повысить

эффективность диагностики и лечения. Источниками данных для телерадиологических систем являются компьютерная томография, ультразвук, магнитно-резонансная томография, цифровая флюорография и компьютерная радиография. Государственные программы телемедицины охватывают области создания базы данных (мультимедийное досье пациента NUCLEUS, европейская информационная система доноров костного мозга EMDIS, европейская модель лечения EPIC, база знаний FEST для европейских служб телемедицины), телекоммуникационная инфраструктура (ISAAC-интегрированная телекоммуникационная система, SHINE -стратегическая информационная сеть для здравоохранения в Европе), значимые программы для обслуживания определенных групп населения (пожилые люди, инвалиды и т. д.) или отдельные ситуационные задачи (стихийные бедствия и т. д.).

С 1988 года в Европе началась работа по программе AIM по внедрению современной информатики в медицине. ATL представила экспертную систему HDI 5000, которая обеспечивает обработку изображений со скоростью 14 миллиардов операций в секунду с динамическим диапазоном 150 дБ, что позволяет наблюдать четкое цветное изображение кровотока. Система оснащена интеллектуальной адаптивной системой настройки, системой оптимизации параметров, обеспечивающей трехмерное изображение судов.

В России существуют экспертные системы для ортопедии. Их успех определяется степенью четкого понимания реальных медико-технических процессов. Система совмещена с мультимедийными устройствами.

Введение механизма для корректировки «весов» знаков, введения коэффициентов «достоверности» и т. Д. Такие системы предлагают варианты решения: «мягкое решение», «сложный выбор» и т.д.

1.2.2 Концепт телемедицины

В целом, телемедицина допускает медицинские действия и связанные с ними медицинские процедуры на расстоянии, хотя она не отменяет

необходимости личного контакта и не намерена заменить так называемые стандартные лекарства. Он охватывает очень разные реальности.

Иногда между участниками происходит взаимодействие в реальном времени, а в другое время между каждым вмешательством возникает промежуток времени. Для связи на расстоянии могут использоваться разные устройства, такие как устройства для видеоконференций, электронная почта, веб-камеры, КПК или смартфоны. Канал, используемый для обеспечения связи, также может различаться (широкополосный, сетевой, беспроводной). Можно называть телемедицину широкой концепцией, охватывающей все виды дистанционной медицины. Тем не менее, были разработаны более подробные концепции для определения конкретных реалий, а именно кибермедицина (практика медицинских действий с использованием Интернета) и электронное здравоохранение (обмен медицинской информацией посредством новых коммуникационных технологий). В целом, телемедицина может принимать различные формы, каждая из которых имеет свои особенности.

Телемониторинг относится к контролю жизненно важных функций на расстоянии через системы, установленные вдали от врача и даже переносимые пациентом, которые посылают сигналы тревоги в центр дистанционного управления. Эта возможность не является незначительной с точки зрения затрат на здравоохранение, поскольку она позволяет дешевое предоставление медицинской помощи без институционализации.

Телеобразование означает предоставление общей и технической информации, доступной для широких слоев населения или других специалистов в области здравоохранения, соответственно. К этой категории относятся так называемые «виртуальные больницы», поскольку они представляют собой форму передачи информации (правила процедуры, профилактические меры, клинические случаи) между медицинскими работниками через веб-сайты; однако, несмотря на его потенциал, он остается формой распространения знаний, редко используемой во многих европейских

странах. Более распространенными являются веб-сайты, предназначенные для предоставления широкой общественности информации о здравоохранении, причем некоторые из них носят технический и специфический характер, которые напоминают рекомендации, которые доктор дает своему пациенту; тем не менее, эти «мнения» даны в целом, независимо от особенностей клинической ситуации пациента.

Телеинтервенция относится к хирургическим вмешательствам, применяемым на расстоянии, прибегая к механике, такой как роботизированные и компьютеризированные машины, которые позволяют врачу вмешиваться в тело пациента, фактически не касаясь его.

Процесс телеконсультации аналогичен традиционному медицинскому консультированию, с той разницей, что врач и пациент физически разделены и общаются на расстоянии, устанавливая разговор в режиме реального времени с помощью видеоконференции, телефона или чата. Этот метод может быть применен практически ко всем медицинским специальностям, даже тем, которые требуют проведения реального физического обследования, если пациента сопровождает другой медицинский работник, способный выполнить обследование и сообщить результаты врачу, кто позаботится о дистанционной консультации. В некоторых других случаях можно преодолеть физическое расстояние, используя определенную технологию, например, позволяя стетоскопам удаленно прослушивать сердцебиение.

Наконец, другой вариант осуществления телемедицины, который также может называться телеконсультацией, относится к контактам между двумя или более медицинскими работниками в отношении определенной медицинской проблемы. Хорошими примерами этого метода консультаций среди медицинских работников являются механизмы передачи ультразвука и другие подобные устройства передачи, которые используются для работы между больницами или между больницами и машинами скорой помощи.

1.2.3 Преимущества телемедицины для пациентов и медицинских работников

Телемедицина имеет ряд преимуществ, как для врачей и медицинских учреждений, так и для пациентов. Таким образом, он играет важную роль в развитии медицины и оказании медицинской помощи в Европе.

По сути, телемедицина позволяет осуществлять дистанционное групповое сотрудничество между различными медицинскими работниками из разных мест, иногда даже из разных стран. Практикующие могут общаться с удаленными коллегами, тем самым улучшая качество предоставляемых услуг. Непрерывный поток общения между медицинскими работниками мотивируется растущей сложностью медицины, что вынуждает врачей консультироваться с более опытными коллегами или экспертами в определенной области или просто запрашивать второе мнение. Все эти виды поведения считаются хорошей медицинской практикой (при условии, что они не задерживают срочную процедуру и не представляют собой защитную медицину), и фактически отсутствие таких контактов может привести к ответственности врача за его незаинтересованность и невнимательность. С другой стороны, пациенты, которые в настоящее время лучше осведомлены о своих правах и более внимательны к риску медицинских неполадок, также обращаются за вторым мнением, иногда от врачей, с которыми они никогда не встречались, просто используя телефон, электронную почту, почта или даже сайт.

В перспективе медицинских учреждений телемедицина открывает возможность расширить спектр доступных медицинских услуг, тем самым увеличивая шансы на получение прибыли.

Кроме того, это может быть довольно удобно для самих пациентов. Телемедицина облегчает прямой доступ пациента к удаленному врачу, не требуя перемещения любого из участников, предоставляя доступ к некоторым видам медицинской помощи, которые в противном случае были бы

недоступны. По сути, телемедицина может оказаться важным инструментом в тех случаях, когда географическое расстояние затрудняет доступ к здравоохранению, особенно специализированному, как это имеет место во многих сельских районах или в развивающихся странах, в которых отсутствует инфраструктура здравоохранения. Медицинское лечение в этих сценариях потребует длительных поездок - не всегда возможно и, как правило, довольно дорого, отнимает много времени и сил - как для врачей, так и для пациентов.

Таким образом, в большинстве случаев пациенты просто лишены доступа к надлежащему медицинскому обслуживанию. Телемедицина также может быть потенциально очень полезным инструментом в профилактической медицине, предоставляя пациентам полезную информацию о состоянии их здоровья. Это может быть особенно актуально для борьбы с хроническими заболеваниями (диабет и сердечные заболевания) - растущей проблемой общественного здравоохранения, обусловленной увеличением продолжительности жизни и ростом численности пожилого населения, хотя эти заболевания также растут в молодом населении - без сохранения пациент в стационаре (телемониторинг).

Мало того, что он более удобен для пациента, который может поддерживать свою обычную жизнь как можно более нормальным, так как это значительно сокращает расходы, связанные с госпитализацией, и, в конечном итоге, общие расходы на здравоохранение, которые являются актуальной проблемой в период экономических кризисов. С другой стороны, соблазнительно для тех пациентов, которые могут чувствовать себя более комфортно, чтобы задавать вопросы о деликатных проблемах (таких как зависимость или венерические заболевания) удаленному врачу. Телемедицина также имеет большой потенциал для обеспечения безопасности пациента, поскольку позволяет избежать некоторых человеческих ошибок. Давайте возьмем в качестве примера, что происходит с электронным рецептом и как он

предотвращает ошибки, возникающие из-за трудностей с чтением каллиграфии доктора.

Улучшение оказания медицинской помощи является еще одним важным фактором. Эта цель достижима несколькими способами: вовлечение пациентов в решение их собственных проблем здравоохранения, что является механизмом для расширения возможностей пациента; расширение санитарного просвещения пациентов и технического образования медицинских работников за счет использования новых коммуникационных технологий; и легкость получения второго мнения от специалиста, особенно важного для пациентов в сельских районах или для тех, кто страдает от болезней сирот.

Несмотря на все эти преимущества, существуют трудности, которые включают в себя огромные затраты на внедрение службы телемедицины, препятствия из-за функциональной совместимости технических инфраструктур в медучреждениях, озабоченность по поводу конфиденциальности и конфиденциальности данных здравоохранения и другие проблемы.

1.2.4 Примеры проектов

В мире осуществляется более 190 проектов в области телемедицины и МИС, большинство из которых находятся в США, Европе, Австралии. Вы можете заметить, что с учетом огромной территории России, неравномерности ее расселения внедрение телемедицины в России во много раз важнее, чем в Западной Европе или США. В России есть все необходимые компоненты, чтобы в ближайшие 5-10 лет телемедицина стала одним из рутинных элементов оказания медицинской помощи населению и повышения квалификации медицинских работников.

Основная цель проектов - обеспечить доступность услуг телемедицины. С этой целью в рамках проектов проводятся технико-

экономические исследования, решаются проблемы стандартизации и взаимодействия сетей телемедицины с целью повышения эффективности использования всех телекоммуникационных ресурсов, обеспечения конкуренции между владельцами сетей. Важным фактором является координация развития телемедицинских услуг, созданных для отдельных независимых проектов.

В бывшем СССР идеи телемедицины были впервые сформулированы и апробированы в связи с реализацией программ освоения космоса. В то же время максимальная диагностическая эффективность была достигнута при ограничениях на объем бортового оборудования, пропускную способность каналов связи и т.д. В настоящее время в России продолжается работа по космической медицине, в том числе совместная работа с заинтересованными зарубежными странами.

В России реализуется ряд независимых телемедицинских программ, финансируемых за счет грантов российских и зарубежных организаций, из бюджетных средств через соответствующие научные или медицинские программы, из коммерческих источников. Успешно работает ряд коммерческих компаний, предоставляющих медицинские услуги, связанные с диагностикой и консультированием, которые можно рассматривать как готовые элементы телемедицинских услуг.

Первое широкомасштабное использование телемедицины в России было связано с землетрясением в Спитаке и катастрофой в Уфе.

Один из первых экспериментов по дистанционной диагностике в России был проведен институтом. А.В. Вишневого, в котором компьютер выполнял функции экспертной системы, получал информацию о пациентах из отдаленных регионов по низкоскоростным каналам связи, обрабатывал, выдавал наиболее вероятные диагнозы, которые были указаны в качестве дополнительной информации. В 70-х годах в СССР данные кардиограммы передавались по линиям связи, которые использовались для диагностики

отдаленного пациента. Вообще, создание диагностических центров связано с именем академика Е. Чазова.

В настоящее время во врачебной практике часто применяются сеансы видеоконференцсвязи для консультирования и диагностики. Такие услуги развиты в Центре сердечно-сосудистой хирургии им. А.Н. Бакулева, в Медицинской академии им. И.М. Сеченова, Институте педиатрии и детской хирургии, Российской военно-медицинской академии и др.

В Алтайском онкоцентре осуществляются эксперименты по предоставлению телемедицинских услуг. Аналогичные программы реализуются в Центре космической медицины.

МГУ активно работает в области международной телемедицины, в частности, с Тафтским университетом (Бостон), с институтом Де Бейки, с медицинским центром НАСА.

При поддержке РФФИ в рамках гранта 97-07-90329 РосНИИРОС, Российский научный центр "Курчатовский институт" и МГУ принимали участие в создании опытного района международной подсистемы телемедицины. Основной целью работы являлось создание опытного района подсистемы телемедицины для экспериментальной проверки эффективности ее практического применения с учетом природных и экономических особенностей России.

В рамках гранта получены результаты, подтверждающие возможность и эффективность использования предложенной системы, в том числе на международном уровне во время участия в сентябре 1997 г. в выставке "Интерактивный Телеком-97", проведенной Международным союзом электросвязи. В экспериментальной проверке в опытный район сети телемедицины были объединены Московский государственный университет, Российский кардиологический научно-производственный комплекс, больница ЗИЛ г. Москвы, компания ТАНА (радиологические исследования и производство диагностического оборудования), 1 Медицинский институт, Сыктывкарский кардиоцентр, Российский научный центр "Курчатовский

институт", Тафтский университет (Бостон), выставочный стенд России на выставке в Женеве. Соединения по Москве обеспечивались по арендованным каналам компании "Голден Лайн" и Российского НИИ развития общественных сетей, соединение с Женевой обеспечивалось по каналам Ростелеком, соединения с Сыктывкаром и Бостоном обеспечивались через спутниковый телепорт "Курчатовский институт".

Для организации спутниковых каналов связи и работы в сети использовалось оборудование как российских компаний (НПО "Исток", НИИ "Радио", Радиотехнический институт им. Минца и др.), так и зарубежных компаний, в том числе EFDData, CISCO, Motorola, Silicon Graphics, PictureTel, PairGain и др.

Самым надежным способом подключения абонентов сети телемедицины было использование выделенных каналов связи. Однако из-за их высокой стоимости использовались методы коммутации пакетов (протоколы Frame Relay) или коммутация каналов связи. В этом случае возникали ситуации, когда соединение с абонентом осуществлялось через абонентские линии связи московской городской сети. Как правило, это кабельные линии связи, которые не позволяют обеспечить высокоскоростной доступ. Для доставки информационных потоков E1 по абонентским линиям связи было протестировано оборудование от NewBridge, Shmidt, Siemens, PairGain. Наиболее стабильная и качественная связь на скоростях E1 была обеспечена оборудованием PairGain.

Следует отметить, что для обеспечения самообеспеченности сети телемедицины в условиях значительного, но несостоятельного спроса целесообразно стремиться к реализации принципов интеграции услуг пользователей сети, позволяющих предоставлять услуги сети с услугой. интеграция, включая доступ к компьютерным сетям и сетям общего пользования, в том числе к сетям дистанционного обучения, биржевым, торговым и другим коммерческим сетям. При использовании телемедицинского терминала в довольно крупных населенных пунктах такие

услуги окупают 50% стоимости технологических нужд (аренда каналов связи, доступ к коммутационному центру и т.д.).

Сеть телемедицины рассматривалась как двухуровневая иерархическая сеть, состоящая из 1) каналов связи и сетей, работающих с протоколами TCP / IP, и 2) каналов и сетей, которые предоставляют услуги ISDN. Подсеть первого уровня является элементом общедоступных компьютерных сетей и предназначена для организации обучения, доступа к базам данных и других неоперативных задач. Подсеть второго уровня является профессиональной, она используется для получения высококачественных материалов, принятия ответственных решений и проведения ответственных мероприятий в режиме реального времени. Сети этого уровня предоставляют телевидение, видеоконференции, высокоскоростные услуги передачи данных и т.д.

В рамках гранта было показано, что для внедрения телемедицины на практике необходимо тщательно изучить медицинские особенности регионов, региональную телекоммуникационную инфраструктуру, уровень профессиональной подготовки врачей к использованию технических средств информатики и необходимо изучение этических, социальных, стоимостных, правовых и других проблем. Для выявления указанных проблем используются демонстрационные проекты, обобщение результатов которых позволяет оценить экономическую эффективность, разработать технологии, разработать предложения по стандартам, принять решение о лицензировании и т.д. По сути, схема здравоохранения — это иерархическая структура, соответствующая процедуре постановки диагноза и поиска правильного лечения. Реализация этой процедуры в рамках системы телемедицины является наиболее важной задачей, которая помогает сократить время и затраты, чтобы определить правильный способ лечения и снизить риск для человека. Процедура диспетчеризации запросов осуществляется в диспетчерских центрах или «Call Center», которые определяют источник

информации для диагностики и лечения и отправляют туда запросы, обеспечивая связь по каналам связи.

В России широкомасштабный проект телемедицины впервые в мире был реализован в 1978-1984 гг. созданием "Системы дистанционной диагностики неотложных состояний" при "остром животе", включая заболевания хирургического, гинекологического и урологического профиля у взрослых и детей, нарушениях коронарного кровообращения, нарушениях мозгового кровообращения, тромбозах магистральных артерий, травме черепа.

На сегодняшний день консультативно-диагностические центры развернуты на базе шести областных, областных или городских детских больниц, которые имеют прямую телефонную связь с круглосуточным консультативно-техническим центром, который, в свою очередь, имеет прямой доступ к отделениям детской хирургии, педиатрии, детская анестезиология и реаниматология Российского государственного медицинского университета, Клиника хирургии Института педиатрии и детской хирургии Министерства здравоохранения Российской Федерации, лаборатория и клиника Республиканской детской клинической больницы и Института педиатрии РАМН. Это дает возможность квалифицированным консультациям ведущих специалистов практически по всем педиатрическим вопросам. Создана технология передачи текстовой информации о пациенте, рентгенограмм, включая компьютерные томограммы, любые данные на бумаге, микроскопические изображения цитологических и, в последнее время, гистологических препаратов.

1.2.5 Международные информационные сети

Компьютерные локальные информационные системы объединяются с помощью сети Интернет. Первая распределенная информационная система была создана по токсикологии в 1953 г. в Чикаго. В настоящее время

существуют общедоступные базы данных POISINDEX, TOXNET, MEDLINE, EUROTNET и др. с подробнейшей сопроводительной информацией по применению. Особенно Интернет полезен для пациентов, которые могут, практически, обратиться ко всему мировому сообществу за помощью. Статистика показывает, что большинство пациентов обращаются по вопросам онкологии.

1.3 Анализ и обоснование выбранной платформы

1.3.1 Выбор архитектуры системы

Информационная система должна быть сетевой и многопользовательской. База данных системы будет располагаться на сервере и доступ к ней должен быть получен с нескольких компьютеров. Чтобы выполнить данные требования, будет применяться технология «клиент/сервер».

На рисунке 1.1 показана архитектура типа «клиент/сервер».

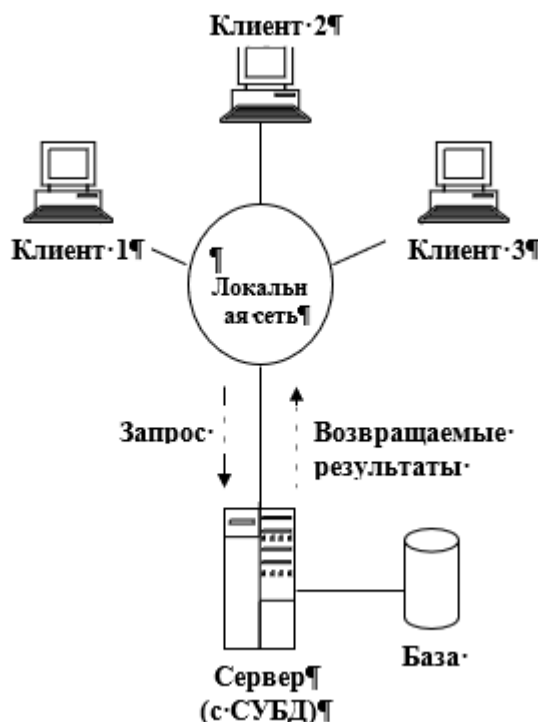


Рисунок 1.1 – Общая схема построения систем с архитектурой «клиент-сервер»

«Клиент/сервер» означает такой способ взаимодействия программных компонентов, при котором они образуют единую систему. Как видно из самого названия, существует некий клиентский процесс, требующий определённых ресурсов, а также серверный процесс, который эти ресурсы предоставляет. При этом совсем необязательно, чтобы они находились на одном и том же компьютере. На практике принято размещать сервер на одном узле локальной сети, а клиенты – на других узлах. Клиент управляет пользовательским интерфейсом и логикой приложения, действуя как сложная рабочая станция, на которой выполняются приложения баз данных. Клиент принимает от пользователя запрос, проверяет синтаксис и генерирует запрос к базе данных на языке SQL, который соответствует логике приложения. Затем он передаёт сообщение серверу, ожидает поступления ответа и форматирует полученные данные для представления их пользователю. Сервер принимает и обрабатывает запросы к базе данных, а затем передаёт полученные результаты обратно клиенту. Такая обработка включает проверку полномочий клиента, обеспечение требований целостности, поддержку системного каталога, а также выполнение запроса и обновление данных. Помимо этого, поддерживается управление параллельностью и восстановлением.

Этот тип архитектуры обладает приведёнными ниже преимуществами:

- 1) обеспечивается более широкий доступ к существующим базам данных;
- 2) повышается общая производительность системы: поскольку клиенты и сервер находятся на разных компьютерах, их процессоры способны выполнять приложения параллельно, при этом настройка производительности компьютера с сервером упрощается, если на нём выполняется только работа с базой данных;
- 3) стоимость аппаратного обеспечения снижается: достаточно мощный компьютер с большим устройством хранения нужен только серверу – для хранения и управления базой данных;

4) сокращаются коммуникационные расходы: приложения выполняют часть операций на клиентских компьютерах и посылают через сеть только запросы к базе данных, что позволяет существенно сократить объём пересылаемых по сети данных;

5) повышается уровень непротиворечивости данных: сервер может самостоятельно управлять проверкой целостности данных, поскольку все ограничения определяются и проверяются только в одном месте, при этом каждому приложению не придётся выполнять собственную проверку;

6) эта архитектура весьма естественно отображается на архитектуру открытых систем.

1.3.2 Выбор аппаратно-программной платформы

Для разработки программного обеспечения необходимо определиться с СУБД и языком программирования, которые будут использоваться.

Для хранения статей энциклопедии и других необходимых данных (словари стоп-слов, вопросительных слов и т.д.) была выбрана СУБД Mssql.

Microsoft SQL Server – это система управления реляционными базами данных, разработанная Microsoft. Как сервер базы данных, это программный продукт с основной функцией хранения и извлечения данных в соответствии с запросами других программных приложений, который может работать либо на том же компьютере, либо на другом компьютере в сети (включая Интернет). Microsoft продает по меньшей мере дюжину различных выпусков Microsoft SQL Server, предназначенных для разных аудиторий и для рабочих нагрузок, начиная от небольших приложений на одной машине до крупных приложений с выходом в Интернет со многими одновременными пользователями.

Для разработки программной части системы был выбран язык C#.

C# — объектно-ориентированный язык программирования.

C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык является универсальным языком программирования с множеством парадигм, охватывающим строгую

типизацию, лексическую область, императив, декларативный, функциональный, универсальный, объектно-ориентированный (на основе классов) и компонентно-ориентированный дисциплины программирования. Он был разработан Microsoft примерно в 2000 году в рамках инициативы .NET, а затем утвержден в качестве международного стандарта Ecma (ECMA-334) в 2002 году и ISO (ISO / IEC 23270) в 2003 году. Mono — это название свободного и открытого исходного проекта для разработки компилятора и среды выполнения для языка. C# является одним из языков программирования, разработанных для Common Language Infrastructure (CLI).

C# был разработан Андерсом Хейлсбергом, а его командой разработчиков в настоящее время руководит Мадс Торгерсен..

Переняв многое от своих предшественников — языков C++, Java — C#, взял лучшее от всех языков и по большей части не поддерживает функции языка, которые вызывают проблемы и ошибки при компиляции.

По своему дизайну C# является языком программирования, который наиболее непосредственно отражает основную инфраструктуру общего языка (CLI). Большинство его внутренних типов соответствуют типам значений, реализованным средой CLI. Тем не менее, спецификация языка не содержит требований к генерации кода компилятора, то есть не указывает, что компилятор C# должен ориентироваться на среду общего языка, генерировать общий промежуточный язык (CIL) или генерировать любой другой конкретный формат. Теоретически, компилятор C # может генерировать машинный код как традиционные компиляторы C ++ или Fortran.

Синтаксические конструкции C# унаследованы не только от C++, но и от Visual Basic. Например, в C#, как и в Visual Basic, используются свойства классов. Как C++, C# позволяет производить перегрузку операторов для созданных вами типов (Java не поддерживает ни ту, ни другую возможность). C# — это фактически гибрид разных языков. При этом C# синтаксически не менее (если не более) чист, чем Java, так же прост, как Visual Basic, и обладает

практически той же мощностью и гибкостью, что и C++. Подводя итоги, еще раз выделим основные особенности C#.

Метод в C # — это член класса, который может быть вызван как функция (последовательность инструкций), а не просто возможность удержания значения свойства класса. Как и в других синтаксически похожих языках, таких как C ++ и ANSI C, сигнатура метода представляет собой объявление, содержащее по порядку: любые необязательные ключевые слова модификатора области, явную спецификацию его возвращаемого типа, имя метода и, наконец, заключенная в скобки последовательность разделенных запятыми спецификаций параметров, каждая из которых состоит из типа параметра, его формального имени и, необязательно, значения по умолчанию, которое будет использоваться всякий раз, когда нет предоставлен. Некоторые конкретные виды методов, такие как те, которые просто получают или устанавливают свойство класса по возвращаемому значению или присваиванию, не требуют полной подписи, но в общем случае определение класса включает в себя полное объявление подписи его методов.

Обеспечивается полная поддержка использования программных интерфейсов. Однако, в отличие от классического COM, использование интерфейсов - не единственный способ работы с типами с использованием различных двоичных модулей. .NET позволяет передавать объекты (в виде ссылок или значений) через границы программных модулей.

Также предоставляется полная поддержка аспектно-ориентированных программных технологий (таких как атрибуты). Это позволяет назначать характеристики типам (которые во многом напоминают COM IDL) для описания поведения этой сущности в будущем.

Возможно, самое важное, что можно сказать о языке C #, — это то, что он генерирует код, предназначенный для запуска только во время выполнения .NET. Например, вы не можете использовать C # для создания классического COM-сервера. Согласно терминологии Microsoft, код, предназначенный для

работы в среде выполнения .NET, представляет собой управляемый код. Бинарный файл, который содержит управляемый файл, называется сборкой.

1.3.3 Выбор технологий и средств разработки базы данных

База данных будет разработана на SQL. Выбор обусловлен тем, что этот язык является единственным стандартным языком для работы с реляционными базами данных, который получил довольно широкое распространение. SQL находится в свободном доступе, и из-за этого используется большинством СУБД.

При разработке базы данных в качестве СУБД будет использоваться среда MS SQL Server 2005. Был выбран MS SQL Server, поэтому данная система является самой распространенной СУБД, которая, кроме того, имеет преимущества перед другими. Инструменты MS SQL Server позволяют создавать реляционную базу данных без больших затрат времени и имеют средства для взаимодействия с ней через архитектуру клиент-сервер с использованием языка SQL.

Для доступа к данным будет использоваться программный интерфейс ODBC, который осуществляет доступ к базам данных и имеет интерфейс, использование которого позволяет работать с несколькими источниками информации. Это осуществляется благодаря тому, что разработчики источников информации сами занимаются написанием драйверов к своим программам, чтобы интерфейс ODBC мог корректно работать с данными.

Вывод

В современном быстро развивающемся обществе информационные технологии играют основополагающую роль в жизни всех людей. И это относится ко всем сферам жизни людей, в том числе и к медицине.

Во многих странах, где интенсивно вводится электронный документооборот, вводится такое понятие, как телемедицина. Это понятие охватывает большое количество технологий, предназначенных для более эффективной и комфортной работы с данными пациентов

Телемедициной интересуются не только министерства здравоохранения различных стран, но и другие государственные органы, что способствует большим потокам финансирования проектов в этой сфере научных разработок.

Электронная медицинская карта – это документ, который отражает множество показателей пациента, доступный множеству специалистов, поэтому облегчает работу сотрудников медицинских учреждений в постановке правильного диагноза больному и назначении правильного курса лечения.

2 РАЗРАБОТКА РАСПРЕДЕЛЕННОЙ СИСТЕМЫ

2.1 Принципы CASE-технологии

Анализ программной системы с использованием CASE-технологии позволяет значительно повысить качество создаваемых программ за счет тщательной разработки алгоритмов, за счет анализа требований, предъявляемых пользователями. CASE-технология позволяет документировать взаимодействия разрабатываемой системы с пользователями и внешними системами, позволяет отразить структуру разрабатываемой системы и динамику ее поведения.

2.2 Анализ вариантов использования системы

Распределенная система ведения истории болезни используется пользователем для управления сетевой базой данных. Управление базой данных включает в себя выполнение следующих операций:

- включение и модификация записей в таблицы;
- выборка записей из таблиц;
- удаление записей из таблиц.

Построение системы подразумевает присутствие стационарного сервера, который управляет деятельностью всей системы. Для пользователя желательно наличие удобного и простого графического интерфейса.

Ниже приведена UML-диаграмма, отображающая взаимодействие основных действующих субъектов операционной среды.

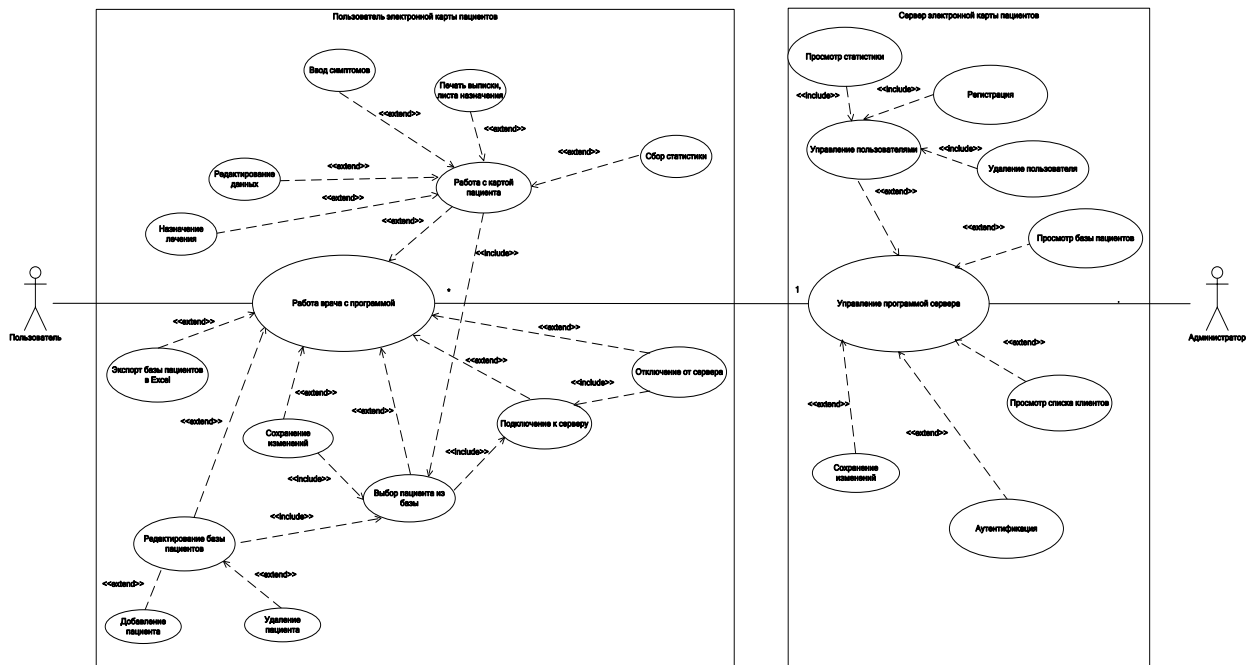


Рисунок 2.1 - UML-диаграмма вариантов использования

Графический интерфейс по запросам пользователя (посредством обработки событий от графических объектов) формирует для главной функции данные, описывающие действия, которые необходимо выполнить. Это может быть непосредственное выполнение SQL-запроса к базе данных, просмотр конкретной таблицы или ее редактирование. В соответствии с полученными данными главная функция принимает решение о том, какие действия необходимо выполнить и формирует структуру данных, в которой содержится имя узла, на который необходимо переместиться, и имя таблицы базы данных, с которой необходимо работать. Эта структура данных передается функции подключения к БД.

Основные этапы работы с базой данных посредством сервера:

- получение кода таблицы;
- передача модифицированных переменных серверу;
- непосредственное выполнение SQL-запроса к таблице.

Сервер, получив все необходимые данные от клиента, преобразует их в соответствующий SQL-запрос и передает его БД Microsoft SQL, который осуществляет непосредственную модификацию таблицы с данными.

Пример такого вывода данных показан на рисунке 2.2.

База пациентов				
База		Пациенты		
Добавить		Редактировать	Удалить	Печать
Фамилия	Имя	Отчество	Дата рождения	Возраст
Иванов	Иван	Иванович	1 января 1990 г.	22 года
Петров	Петр	Петрович	10 ноября 1989 г.	22 года
Сидоров	Иван	Сергеевич	4 октября 1989 г.	22 года

Рисунок 2.2 - Просмотр таблицы База пациентов

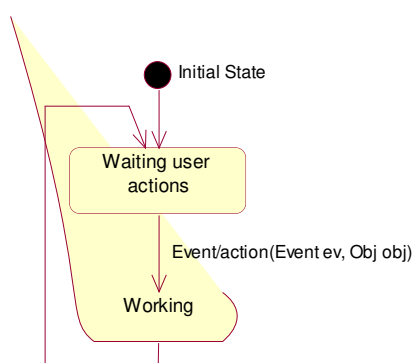
2.3 Определение основных состояний системы

2.3.1 Начальное состояние системы

Начальное состояние системы – инициализация ее компонент: запуск серверной части программы и ожидание подключения клиентов. Из этого состояния система переходит в состояние ожидания действий клиента.

Как только возникло какое-либо событие, порожденное пользователем, система переходит в активное состояние.

При возникновении какого-либо из перечисленных выше событий система переходит в состояние анализа переданного запроса. Определив тип события система переходит в состояние обработки. Обработав возникшее событие, система формирует и передает результаты анализа и обработки запроса в виде структурированного ответа клиенту и возвращается в начальное состояние ожидания реакции пользователя. UML-диаграмма, описывающая этот процесс приведена на рисунке 2.3.



2.3.2 Определение состояний администраторского и пользовательского модулей

Главный модуль после своего запуска создает объект графического окна и переходит в цикл ожидания сообщений от него. Как только такое сообщение пришло, главный модуль переходит в состояние предварительной обработки полученных от графического окна данных. В этом состоянии им определяется характер действий, которые необходимо выполнить. В первую очередь проверяется, являются ли полученные данные непосредственным SQL-запросом к базе данных. В этом случае производится предварительный анализ SQL-запроса с целью выявления таблиц, с которыми необходимо работать.

При получении результатов запроса к базе главный модуль анализирует полученную информацию и приводит ее к нужному формату для представления в графическом окне пользователю. Затем передает ее в виде результирующей таблицы объекту главного окна интерфейса пользователя. После этого главный модуль переходит в состояние ожидания сообщений от главного окна графического интерфейса пользователя. На рисунке 2.4 показаны основные состояния главного модуля.

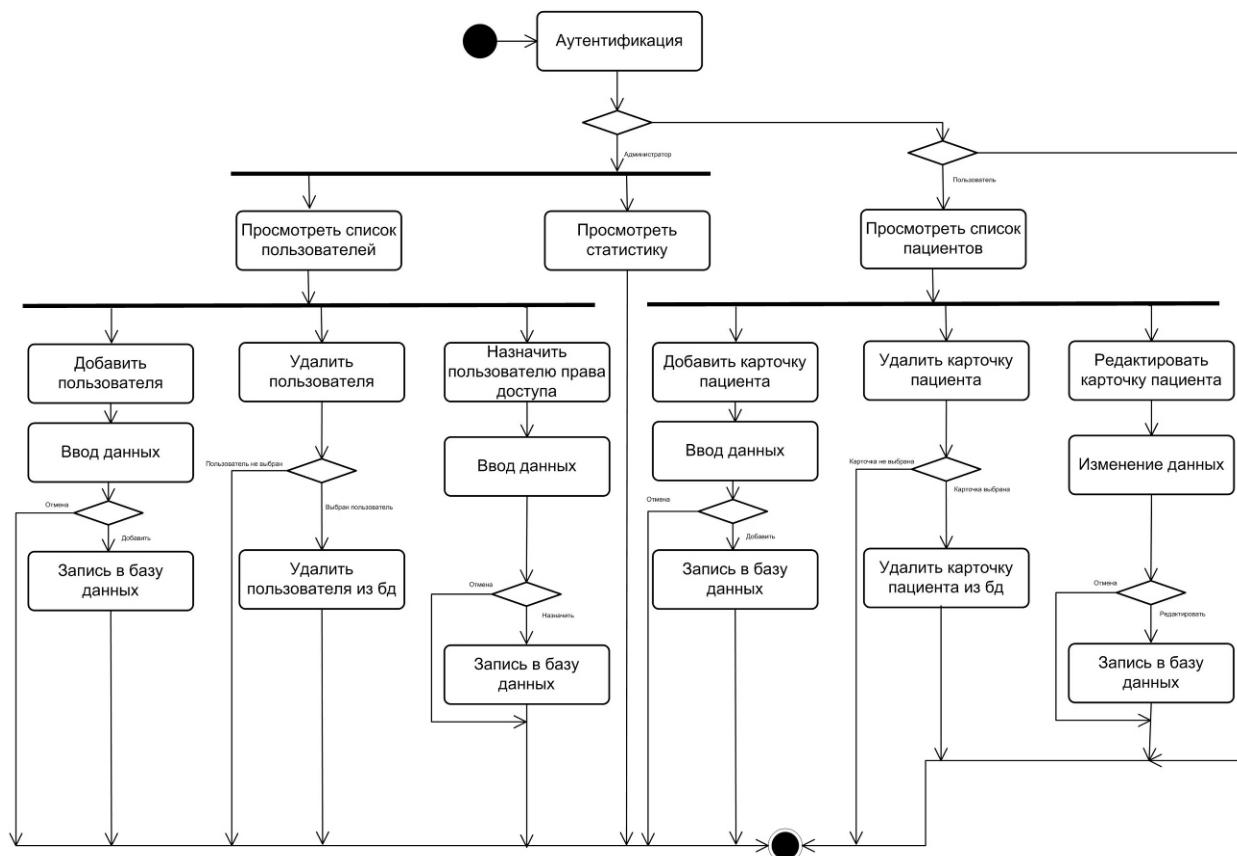


Рисунок 2.4 - Основные состояния администраторского и клиентского модулей

UML-диаграмма деятельности администраторского и клиентского модулей приведена на листе ПГУ 1.2201.20.007 Д17.

2.4 Определение динамического взаимодействия компонент

Динамическое взаимодействие компонент необходимо определить для следующих основных случаев:

- запрос таблицы для просмотра;
- запрос таблицы для модификации;
- SQL-запрос;
- конфигурирование системы.

2.4.1 Запрос таблицы для просмотра и модификации

В этом случае клиенту необходимо подключиться к серверу базы данных, который выполнит соответствующий SQL-запрос. На рисунке 2.5 показана последовательность действий для осуществления этой задачи.

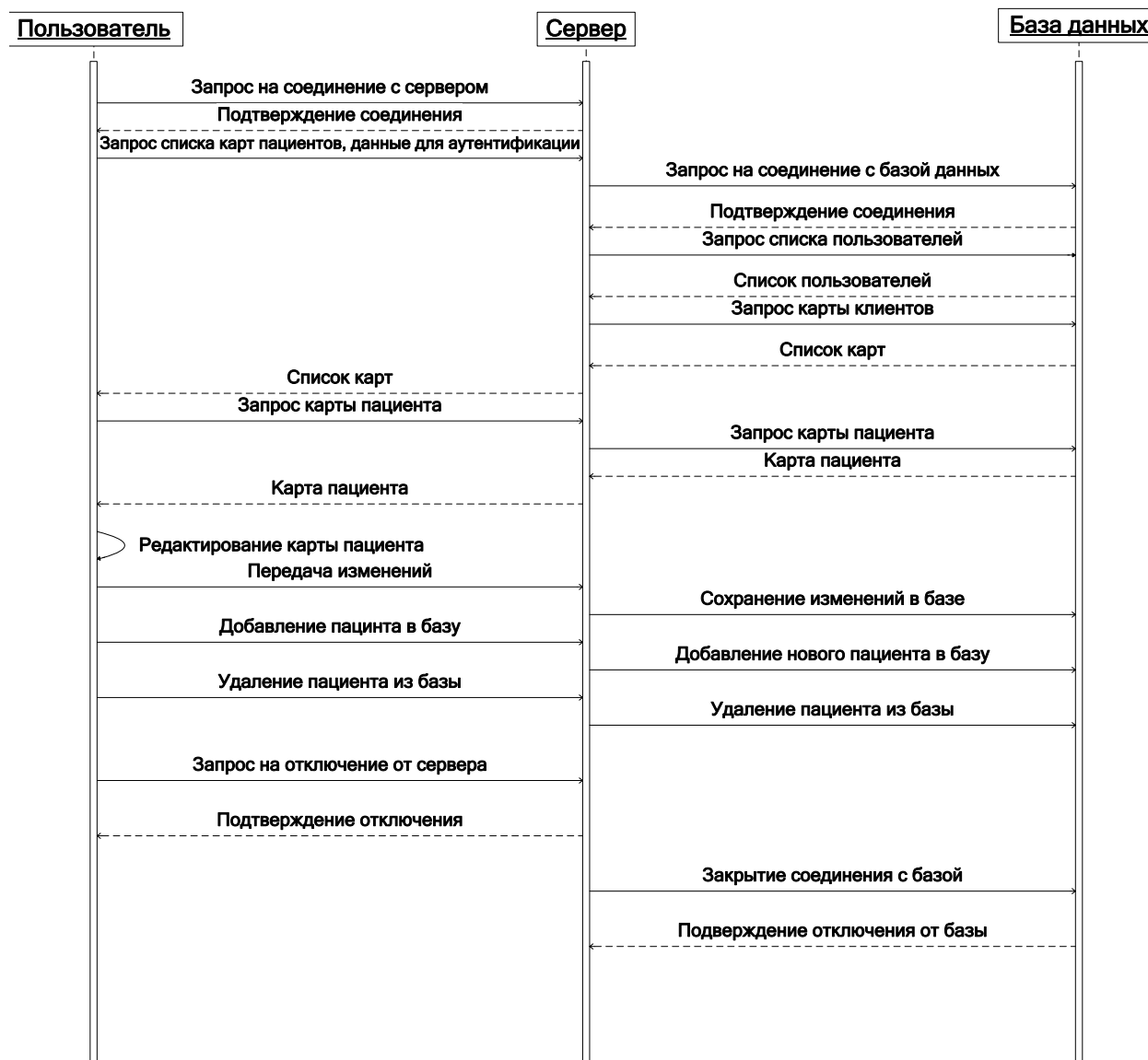


Рисунок 2.5 UML-диаграмма последовательностей

Пользователь посредством форм передает клиентскому приложению, а то, в свою очередь, серверу. Запрос обрабатывается по соответствующему алгоритму, и если это необходимо – формируется SQL-запрос.

Сервер имеет прямой доступ к серверу баз данных (аутентификация производится либо стандартными средствами Windows, либо с помощью логина и пароля). После формирования SQL-команды – Серверное

приложение через драйвер ODBC производит подключение к базе данных и выполняет запрос. Результатом запроса будет являться массив данных, который приложению необходимо будет корректно обработать.

2.5 Описания функций передачи, обработки и доступа к данным

Серверная часть разрабатываемого приложения является клиентом базы данных. Основные функции доступа к данным представлены в сборке System.Data.SqlClient.

Рассмотрим основные классы и методы для работы с базой данных:

System.Data.SqlClient.SqlConnection – класс, который представляет собой открытое подключение к базе данных. Свойства соединения описываются при создании класса как параметр `ConnectionString`. Изменяя данный параметр можно изменять свойства соединения с базой данных: включить/выключить асинхронную обработку данных, шифрование трафика при запросах, начальную базу данных на сервере, имя пользователя и пароль, а также расположение самой базы – на SQL сервере или в локальном файле.

Класс обеспечивает следующие основные методы:

`BeginTransaction()` - Выполнить транзакцию;

`ChangeDatabase(System.String)` - Изменить рабочую базу данных;

`ChangePassword(System.String, System.String)` - Изменить пароль;

`Open()` - Открыть соединение;

`Close()` - Закрыть соединение;

`GetSchema()` - Получить схему данных;

`RetrieveStatistics()` - Запросить статистику;

`ResetStatistics()` - Очистить статистику;

`CreateCommand()` - Создать команду.

`System.Data.SqlClient.SqlConnectionStringBuilder` – класс, который упрощает создание конфигурационной строки соединения для создания класса `SqlConnection`. Имеет изменяемые свойства, которые описаны в описании класса `SqlConnection`.

`System.Data.SqlClient.SqlCommand` – класс, который представляет собой транзакционную команду или хранимую процедуру для выполнения на стороне SQL сервера. Данный класс позволяет выполнить любую команду, написанную на языке SQL и попытаться отменить её, если выполнение затянулось. Также класс позволяет порождать класс `SqlDataReader`.

`System.Data.SqlClient.SqlDataReader` – класс, который позволяет выполнять последовательное потоковое чтение из таблицы. Очень удобен для вывода большого количества записей. Предоставляет набор методов для типового обращения к любому полю в записи.

`System.Data.SqlClient.SqlDataAdapter` – класс, который представляет собой совокупность соединения к базе данных и команд обработки данных. Представляет собой интерфейсный слой между концепциями `System.Data.DataSet` и базой данных SQL. Данный класс имеет следующие основные свойства:

`SelectCommand` – запрос SQL, которая будет передаваться на сервер при попытке чтения из таблицы.

`InsertCommand` – запрос SQL, которая будет передаваться на сервер при попытке добавления записи в таблицу.

`DeleteCommand` – запрос SQL, которая будет передаваться на сервер при попытке удаления из таблицы.

`UpdateCommand` – запрос SQL, которая будет передаваться на сервер при попытке обновления записи в таблице.

`System.Data.DataSet` – Класс представляет собой данные базы данных, кэшированные в памяти. Класс имеет следующие методы:

AcceptChanges() – Применить изменения. Метод синхронизирует содержимое КЭШа и базы данных.

Clear() – метод очищает набор данных.

GetChanges() – метод создает новый набор данных, который включает в себя только изменения между содержимым БД и кэша.

HasChanges() – метод проверки необходимости синхронизации КЭШа и БД.

System.Data.DataTable – класс, который представляет собой одну таблицу в КЭШе.

System.Data.DataRow – класс, который представляет собой одну запись в таблице.

2.6 Разработка подсистемы взаимодействия с клиентом

2.6.1 Функции подсистемы взаимодействия с клиентом

- Создание карточек пациентов;
- Редактирование карточек пациентов
- Удаление и компоновка карточек пациента;
- Вывод на печать выписки из карты пациента;
- Экспорт в Excel;
- Сбор статистики о пациенте

Система взаимодействия в разработанном приложении представлена несколькими разделами «База пациентов», «Пациент», «Форма для печати», «Экспорт в Excel».

Основное окно программного комплекса представлено на рисунке 2.6.

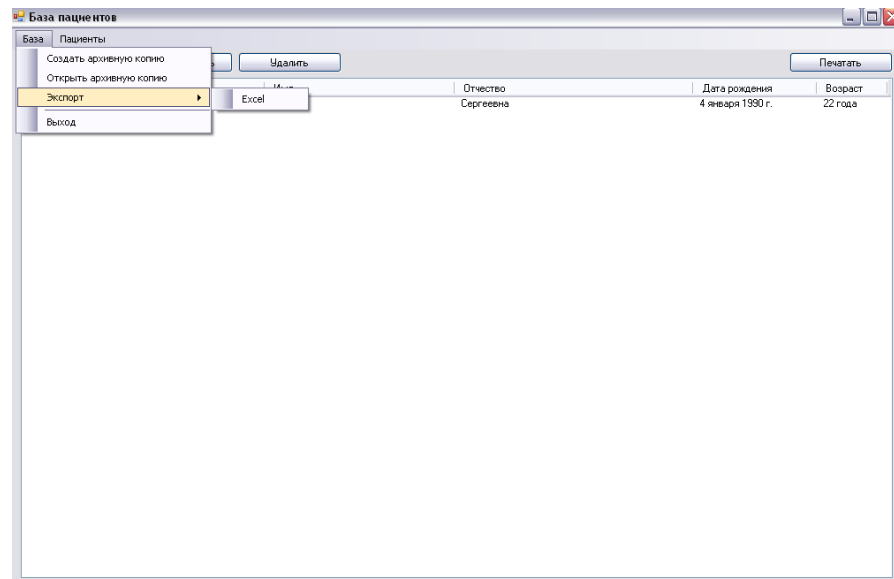


Рисунок 2.6 – Основное окно при запуске программы

Рассмотрим более подробно раздел «База пациентов»

2.6.2 Раздел «База пациентов»

При помощи данного модуля системы, пользователь с соответствующими правами доступа может редактировать базу пациентов, а именно добавлять нового пациента, удалять существующего, редактировать существующего. Для того чтобы добавить нового пациента необходимо нажать кнопку «Добавить», после чего откроется модуль «Пациент», в котором необходимо заполнить соответствующие поля. Для того чтобы удалить существующего пациента, необходимо выбрать его из списка пациентов, и нажать на кнопку «удалить». Для того чтобы редактировать существующего пациента необходимо также выбрать его из списка пациентов и нажать на кнопку «Редактировать». Интерфейс раздела «База пациентов» представлен на рисунке 2.7.

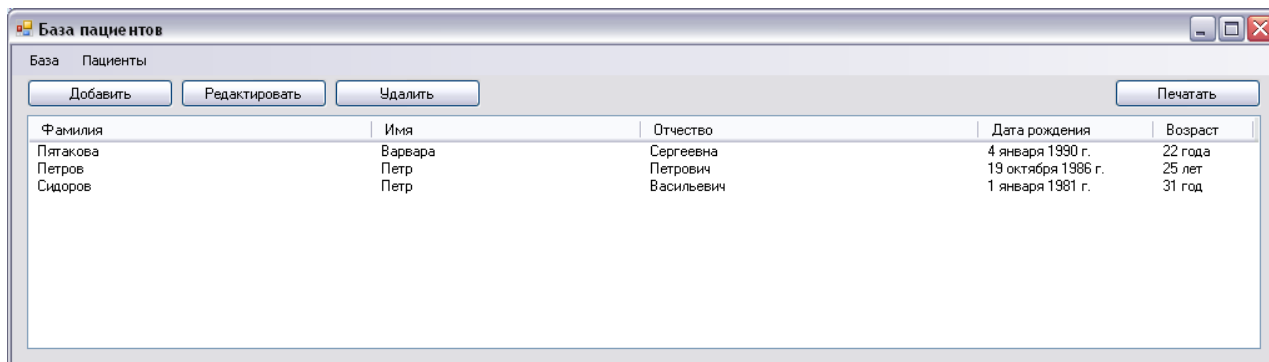


Рисунок 2.7 – Модуль «База пациентов»

2.6.3 Модуль «Пациент»

Модуль «Пациент» содержит следующие разделы:

- Общая информация;
- Жалобы;
- Анамнез;
- Осмотр;
- Результат обследования;
- Заключение;
- Диагноз;
- Лечение;
- Дополнительно

Раздел «Общая информация» предоставляет возможность заполнить основную информацию о пациенте, а именно фамилию, имя, отчество, дату рождения, адрес, телефон, специальность.

Интерфейс модуля «Пациент» раздел «Общая информация» представлен на рисунке 2.8

Пациент

Общая информация | Жалобы | Анамнез | Осмотр | Результат обследования | Заключение | Диагноз | Лечение | Дополнительно

Фамилия: Петров

Имя: Петр

Отчество: Петрович

Дата рождения: 19 октября 1986 г.

Адрес: Пенза, Пушкина 17-60

Телефон: дом. 541109, сот. 89603292416

Специальность:

Печатать | Применить | Отмена

Рисунок 2.8 – Модуль «Пациент», раздел «Общая информация»

Раздел «Жалобы» в модуле «Пациент» позволяет заполнить информацию о жалобах пациента, особенностях боли, локализации боли, интенсивности проявлений и другие показатели, необходимые при работе врача. Интерфейс данного раздела представлен на рисунке 2.9.

Пациент

Общая информация | **Жалобы** | Анамнез | Осмотр | Результат обследования | Заключение | Диагноз | Лечение | Дополнительно

Жалобы

Учащение мочеиспускания ☒ Ургентность ☐ Затруднение при мочеиспускании ☐ Чувство неполного опорожнения ☒ Необходимость натуживания ☐ Прерывистость струй мочи ☐ Ноктурия ☐

Боль

Локализация боли

в мочевом пузыре ☐ во влагалище ☒ тазовая боль ☒
 в уретре ☒ в мошонке ☐
 в области вульвы ☐ в промежности ☐

Интенсивность ☐ Слабая ☐ Умеренная ☒ Сильная ☐ Нестерпимая

Боль связана ☐ С наполнением мочевого пузыря ☐ С мочеиспусканием ☒ С движением ☐ С половым актом ☐

Утечка мочи ☐ При позыве ☐ При кашле ☐ При чихании ☐ Затрудняюсь ответить ☒ Ночью во сне ☐ Нет

Печатать Применить Отмена

Рисунок 2.9 – Модуль «Пациент», раздел «Жалобы»

Раздел «Анамнез» в модуле «Пациент» позволяет заполнить историю болезни пациента: длительности существования жалоб, родах, беременности, менопаузе, эндометриозе, миоме матки, болезнях нервной системы, инсультах, препаратах, которые принимал больной, операциях, травмах. Интерфейс данного раздела представлен на рисунке 2.10.

Пациент

Общая информация | Жалобы | **Анамнез** | Осмотр | Результат обследования | Заключение | Диагноз | Лечение | Дополнительно

Анамнез

Длительность существования жалоб

Роды

Беременности

Менопауза лет

Эндометриоз

Миома матки

Обострения цистита

Болезни нервной системы

☐ Нет
☐ Рассеянный склероз
☐ Болезнь Паркинсона
☐ Осложнения остеохондроза позвоночника
☐ Грыжа диска

Инсульт

☐ Нет
☐ Мозговой
☐ Спинальный

Какие препараты принимает в настоящий момент

Операции

Травмы

Печатать Применить Отмена

Рисунок 2.10 – Модуль «Пациент», раздел «Анамнез»

Раздел «Осмотр» в модуле «Пациент» содержит информацию о результате последнего осмотра больного. Интерфейс данного раздела представлен на рисунке 2.11.

Пациент

Общая информация Жалобы Анамнез **Осмотр** Результат обследования Заключение Диагноз Лечение Дополнительно

Наружное отверстие уретры: Без особенностей

Пролапс влагалища: апикального сегмента

Пролапс тазовых органов: I

☐ Кашлевая проба

☒ Проба Вальсавы

☐ Внеуретральная утечка

Печатать Применить Отмена

Рисунок 2.11 – Модуль «Пациент», раздел «Осмотр»

Раздел «Результат обследования» в модуле «Пациент» содержит информацию о результате последнего обследования больного. Этот раздел разделён на три группы: дневник мочеиспусканий, цистометрия наполнения, микционная цистометрия. Интерфейс данного раздела представлен на рисунке 2.12.

Пациент

Общая информация Жалобы Анамнез Осмотр **Результат обследования** Заключение Диагноз Лечение Дополнительно

Дневник мочеиспусканий

Дневной ml

Ночной ml

Суточный ml

Количество мочеиспусканий

Ноктурия

Средний объем ml

Цистометрия наполнения

Первое ощущение ml инфузии

Первый позыв ml P det H2O

Второй позыв ml P det H2O

Третий позыв ml P det H2O

Максимальная цистометрическая емкость ml

При исследовании порога абд. давления подтекание мочи на см. H2O

Подъемы детрузорного давления

Микционная цистометрия

Qmax мл/с при P det H2O

Vмоч ml

Остаточной мочи ml

Печатать Применить Отмена

Рисунок 2.12 – Модуль «Пациент», раздел «Результат обследования»

Раздел «Заключение» в модуле «Пациент» позволяет врачу заполнить заключение о состоянии пациента. Интерфейс данного раздела представлен на рисунке 2.13

Пациент

Общая информация | Жалобы | Анамнез | Осмотр | Результат обследования | **Заключение** | Диагноз | Лечение | Дополнительно

Заключение

Чувствительность мочевого пузыря: **ПОВЫШЕННАЯ** ▼

Функция детрузора при наполнении:

- ☐ нормальная
- ☐ фазная детрузорная гиперактивность
- ☐ терминальная детрузорная гиперактивность
- ☐ недержание вследствие детрузорной гиперактивности

Комплаентность:

При исследовании порога абдоминального давления подтекания мочи: ▼ 0 см. H₂O

Максимальная цистометрическая емкость: 0 ml

Сократительная способность детрузора: ▼

Инфравезикальная обструкция: ▼

☐ Уродинамическое стрессовое недержание мочи

Цитоскопия:

Рисунок 2.13 – Модуль «Пациент», раздел «Заклучение»

Раздел «Диагноз» позволяет врачу внести в базу диагноз пациента. Интерфейс раздела представлен на рисунке 2.14.

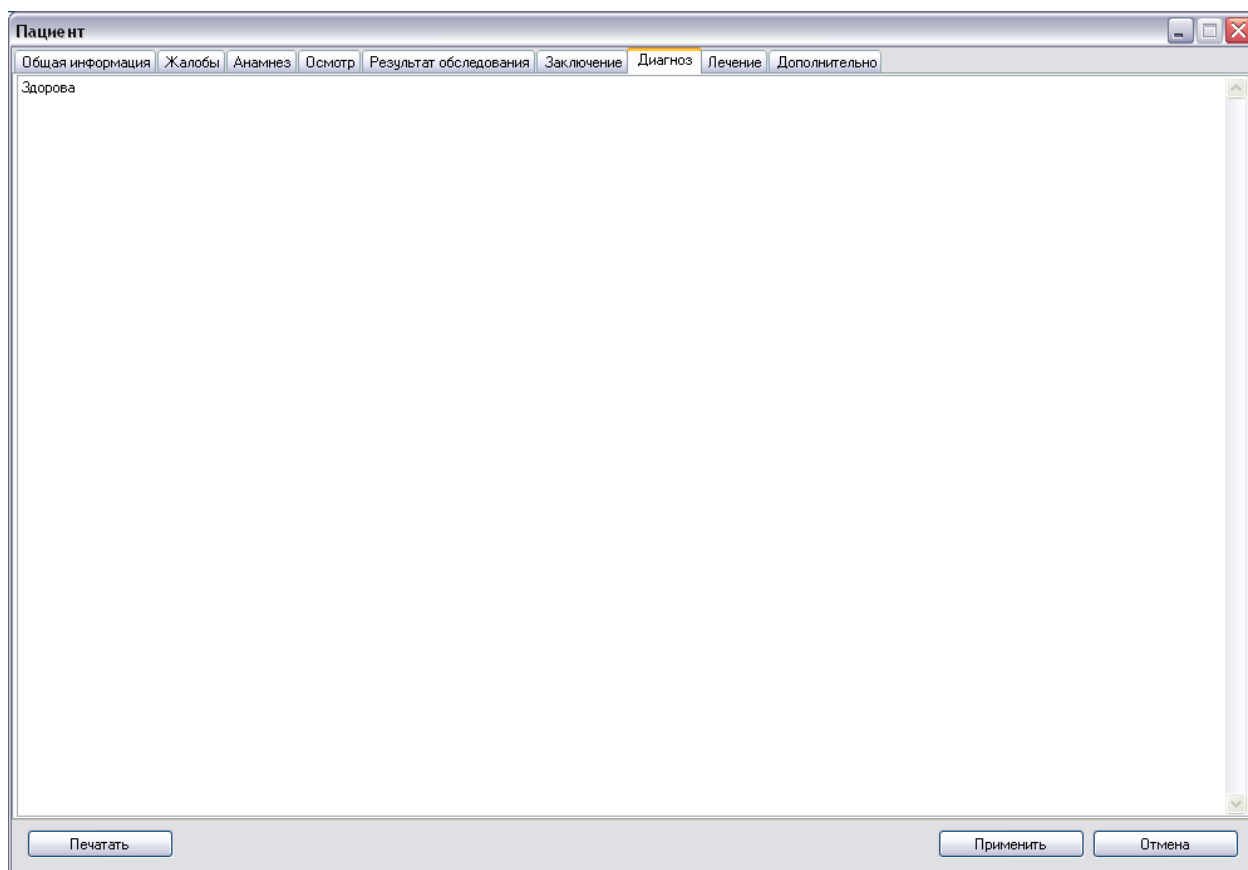


Рисунок 2.14 – Модуль «Пациент», раздел «Диагноз»

Раздел «Лечение» содержит четыре подраздела в зависимости от вида лечения, назначенного врачом: «Консервативное», «Физиотерапевтическое», «Оперативное», «Другие». Интерфейс данных разделов можно увидеть на рисунках 2.15, 2.16, 2.17, 2.18 соответственно.

Пациент

Общая информация | Жалобы | Анамнез | Осмотр | Результат обследования | Заключение | Диагноз | **Лечение** | Дополнительно

Консервативное | Физиотерапевтическое | Оперативное | Другие

- ☐ Диета (исключение алкоголя, острых соусов, приправ, копченостей, консервов, газировки, citrusовых, кофе, шоколада).
- ☐ Мочеиспускание с брюшным напряжением.
- ☐ Упражнения для укрепления мышц тазового дна.
- ☐ Ношение вагинальных шариков по 30 минут, 2 раза в неделю регулярно.
- ☐ Калимин 60 ½ тб. * 2 раза в день 2-3 мес.
- ☐ Нейромидин 20 мг. * 2 раза в день 2 мес.
- ☐ АТФ 1% 1 мл. * 1 р/день в/м 4 недели.
- ☐ Мексидол 2,0 * 2 р/день 7 дней, далее 1 таб. 125 мг. 1 р/день 1 мес.
- ☐ Везикар 1 тб. (5 мг.) в день 3 мес.
- ☐ Пикамилон 20 мг. * 3 р/день 3 мес.
- ☐ Артезин 2 мг. (1/2 тб.) на ночь 3 мес. (обратить внимание - снижает АД).
- ☐ Омник 0,4 мг. на ночь 3 мес. (обратить внимание - снижает АД).
- ☐ Тералиджен 5 - 10 мг. (1-2 тб.) на ночь 1 мес.
- ☐ Атаракс 50 мг (2 тб.) в день (12,5 мг утром, 12,5 мг днем и 25 мг на ночь).
- ☐ Пациент нуждается в консультации специалиста бюро МСЗ для решения вопроса о возможности обеспечения пациента бесплатными катетерами № 4 в сутки.
- ☐ Пациент нуждается в консультации специалиста бюро МСЗ для решения вопроса о возможности обеспечения пациента бесплатными памперсами № 3 в сутки.
- ☐ Лечение препаратом «Уро-гил», 1 инстилляция в неделю №:
- ☐ Баклофен.

ПечатаТЬ | Применить | Отмена

Рисунок 2.15 – Модуль «Пациент», раздел «Лечение», подраздел «Консервативное»

Пациент

Общая информация | Жалобы | Анамнез | Осмотр | Результат обследования | Заключение | Диагноз | **Лечение** | Дополнительно

Консервативное | **Физиотерапевтическое** | Оперативное | Другие

- ☐ Тиббиальная нейромодуляция на аппарате Биобраво 1 сеанс в неделю № 8-10.
- ☐ Экстрокорпоральная электромагнитная стимуляция мышц тазового дна на аппарате Неотонус 2 раза в неделю №8-10.
- ☐ Лечение на аппарате «Андрогин»

ПечатаТЬ | Применить | Отмена

Рисунок 2.16 – Модуль «Пациент», раздел «Лечение», подраздел «Физиотерапевтическое»

Пациент

Общая информация | Жалобы | Анамнез | Осмотр | Результат обследования | Заключение | Диагноз | **Лечение** | Дополнительно

Консервативное | Физиотерапевтическое | **Оперативное** | Другие

- ☐ Инъекции ботулотоксина в сфинктер.
- ☐ Инъекции ботулотоксина в детрузор.
- ☐ Впоризация лейкоплакии.
- ☐ Введение формообразующего геля.
- ☐ Операция Берча
- ☐ Установка слинга
- ☐ Уретротомия
- ☐ TUR простаты

Печать Применить Отмена

Рисунок 2.17 – Модуль «Пациент», раздел «Лечение», подраздел «Оперативное»

Пациент

Общая информация | Жалобы | Анамнез | Осмотр | Результат обследования | Заключение | Диагноз | **Лечение** | Дополнительно

Консервативное | Физиотерапевтическое | Оперативное | **Другие**

- ☐ Периодическая интермиттирующая катетеризация
- ☐ Регулярное ведение дневника мочеиспусканий по 48 - 72 ч.
- ☐ Обследование и лечение у невропатолога!
- ☐ Лечение у гинеколога.

Печать Применить Отмена

Рисунок 2.18 – Модуль «Пациент», раздел «Лечение», подраздел «Другие»

Раздел «Дополнительно» необходим для того, чтобы врач имел возможность делать пометки при приёме пациента. Интерфейс представлен на рисунке 2.19

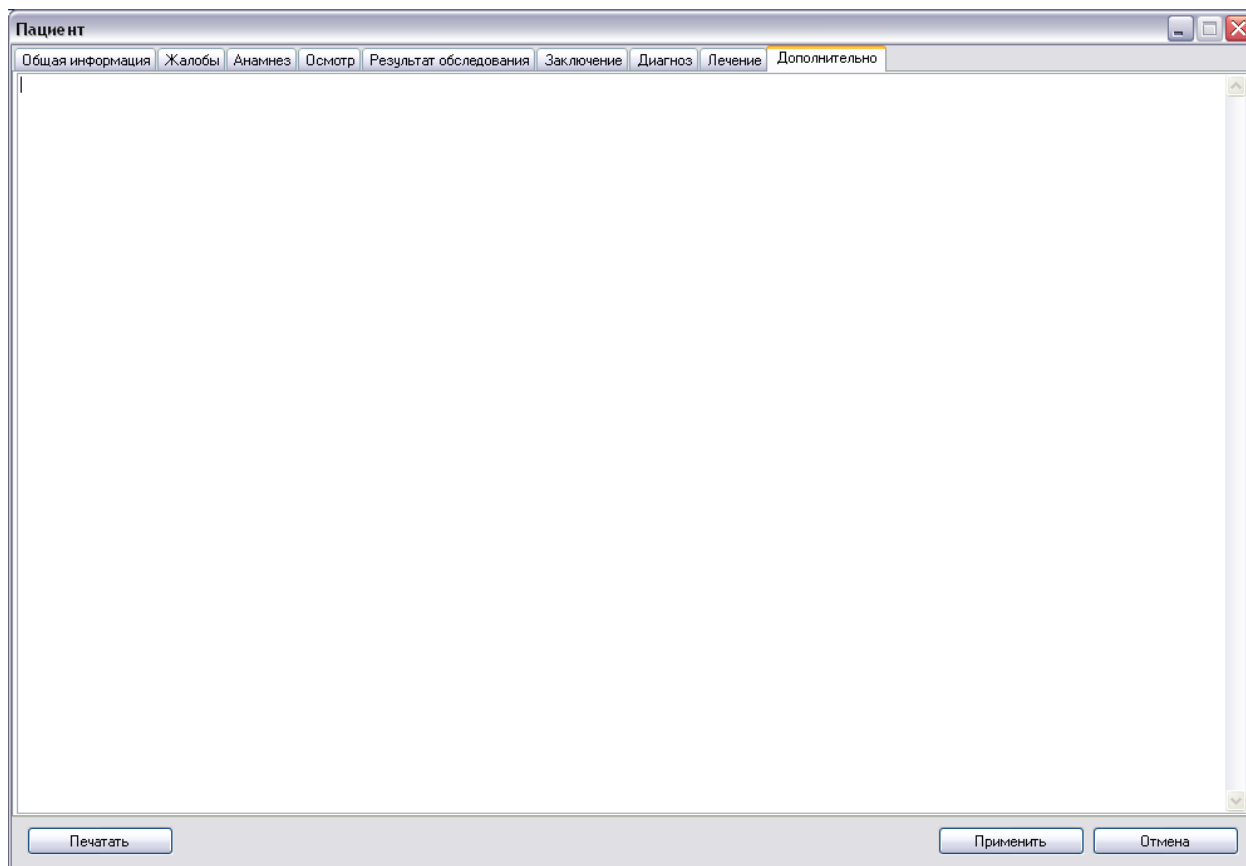


Рисунок 2.19 – Модуль «Пациент», раздел «Дополнительно»

2.6.4 Форма для печати

В клиентской части программного обеспечения реализована возможность вывода на печать выписки о пациенте. Выписка о пациенте включает следующие разделы:

- Общая информация о пациенте: фамилия, имя, отчество, возраст;
- Заключение;
- Диагноз;
- Назначенное лечение.

Интерфейс формы для печати приведён на рисунке 2.20.

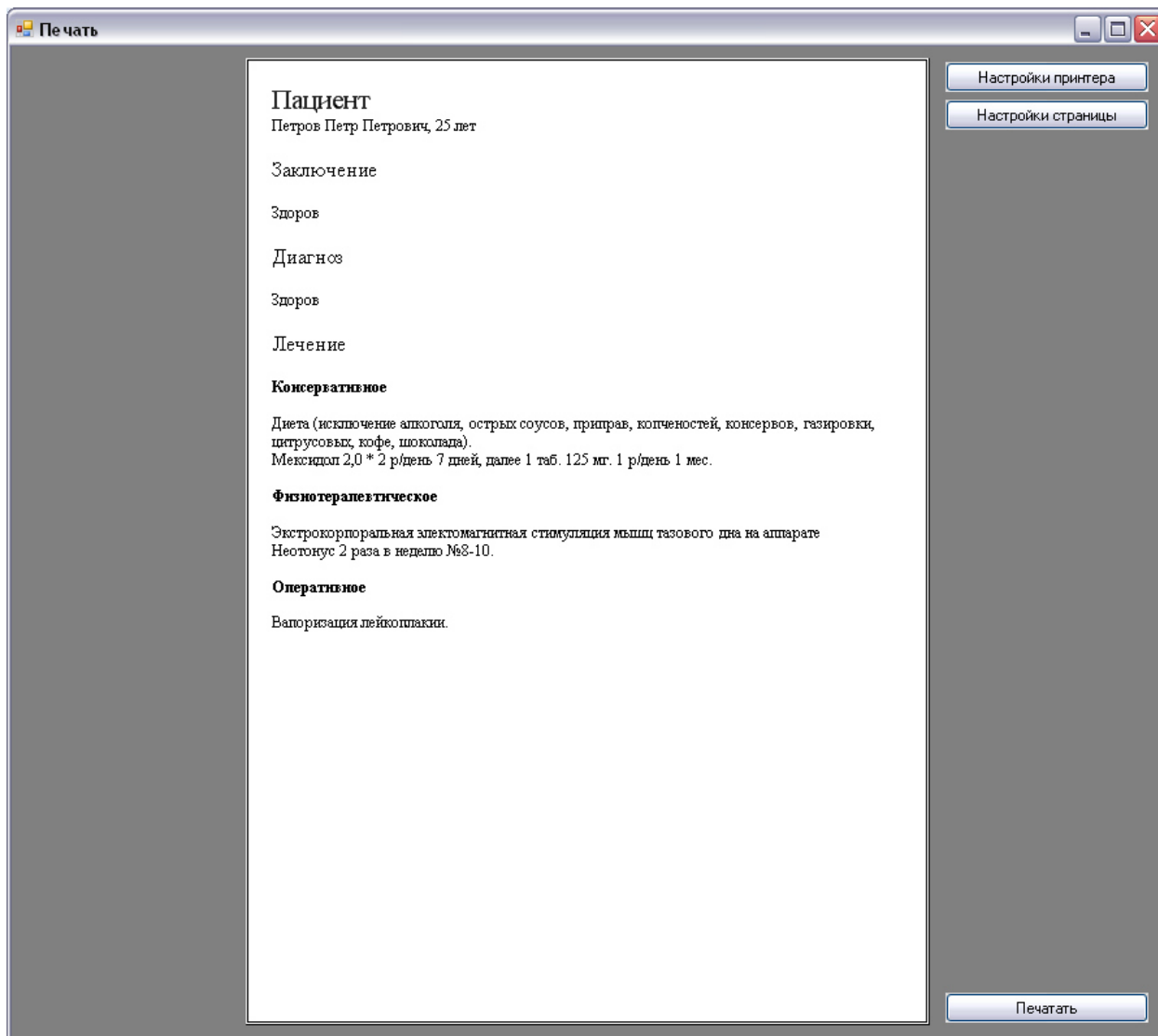


Рисунок 2.20 – Модуль «Пациент», раздел «Дополнительно»

2.6.5 Экспорт в Excel

Функция экспорта данных базы в документ формата Excel необходим для более удобной обработки данных и сбора статистики: применение произвольных фильтров, сортировок, построение графиков. Результат экспорта выглядит следующим образом:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Фамилия	Имя	Отчество	Дата рождения	Адрес	Телефон	Телефон сотовый	Локализация боли	Локализация	Локализация	Локализация	Локализация	Локализация	Локализация	Локализация	Локализация
2	Патахова	Варвара	Сергеевна	04.01.1990	Пенза, Пу	541109	89603292416	локализация боли	ИСТИНА	ИСТИНА	ИСТИНА	ИСТИНА	ИСТИНА	ИСТИНА	ИСТИНА	ИСТИНА
3	Петров	Петр	Петрович	05.01.1990	Пенза, Пу	541109	89603292416		ЛОЖЬ	ИСТИНА	ЛОЖЬ	ЛОЖЬ	ЛОЖЬ	ЛОЖЬ	ЛОЖЬ	ЛОЖЬ
4																
5																
6																
7																
8																

Рисунок 2.21 – Модуль «Пациент», раздел «Дополнительно»

При каждом запуске программы, серверная часть приложения анализирует время последнего экспорта базы в Excel. Если это время превышает двадцать четыре часа, то программа принудительно обновляет результат экспорта.

3 ТЕХНОЛОГИЯ РАЗРАБОТКИ И ОТЛАДКИ ПРОГРАММЫ

Введение

На сегодняшний момент вычислительные системы используются практически во всех сферах деятельности человека. Однако необходимо понимать, что разработка приложения это не просто постановка задачи и её выполнение. Существует большое количество математических методов, описывающих решение существующих проблем, связанных с различными типами задач, которые решаются в разных отраслях жизнедеятельности человека. Поэтому программисту, как специалисту, необходимо учитывать все нюансы, возникающие в процессе разработки, и уметь анализировать предметную область, с которой он работает.

3.1 Технология разработки программы.

Разработка программного обеспечения — это процесс разработки, определения, проектирования, программирования, документирования, тестирования и исправления ошибок, связанных с созданием и обслуживанием приложений, сред или других компонентов программного обеспечения. Разработка программного обеспечения — это процесс написания и поддержки исходного кода, но в более широком смысле он включает в себя все, что задействовано от концепции желаемого программного обеспечения до конечного проявления программного обеспечения, иногда в плановом и структурированном процессе. Следовательно, разработка программного обеспечения может включать исследования, новые разработки, создание прототипов, модификацию, повторное использование, реинжиниринг, обслуживание или любые другие действия, которые приводят к программным продуктам.

Программное обеспечение может быть разработано для различных целей, три наиболее распространенных из которых предназначены для удовлетворения конкретных потребностей конкретного клиента / бизнеса (в случае с заказным программным обеспечением), для удовлетворения предполагаемой потребности некоторого набора потенциальных пользователей (в случае с коммерческими и программное обеспечение с открытым исходным кодом), или для личного использования (например, ученый может написать программное обеспечение для автоматизации повседневных задач). Разработка встроенного программного обеспечения, то есть разработка встроенного программного обеспечения, например, используемого для управления потребительскими продуктами, требует интеграции процесса разработки с разработкой управляемого физического продукта. Системное программное обеспечение лежит в основе приложений и самого процесса программирования и часто разрабатывается отдельно.

Основным способом разработки сложных проектов является разбиение его на множество более мелких модулей. Каждый из модулей может выполняться программистами индивидуально. Чтобы объединить модули в единый проект, необходимо разработать правила для вызова модулей и правила для вызова функций и передачи параметров между отдельными модулями. Эти правила формируют внутрипрограммный интерфейс.

Устройство внутрипрограммного интерфейса определяется на начальном этапе разработки программы.

Разработка интерфейса включает:

- анализ обрабатываемой информации и выбор логической организации данных;
- выбор абстрактных структур данных для представления информации в соответствии с используемым языком программирования равной логической организацией данных;
- выбор физической организации данных;

– выбор структуры и способов взаимодействия отдельных программных модулей.

Эти этапы часто называются жизненным циклом разработки программного обеспечения. Различные подходы к разработке программного обеспечения могут выполнять эти этапы в разных порядках или посвящать больше или меньше времени различным этапам. Уровень детализации документации, создаваемой на каждом этапе разработки программного обеспечения, также может варьироваться. Эти этапы также могут выполняться по очереди (подход на основе «водопада»), или они могут повторяться в течение различных циклов или итераций (более «экстремальный» подход).

Более экстремальный подход обычно включает меньше времени, затрачиваемого на планирование и документирование, и больше времени, затрачиваемого на кодирование и разработку автоматизированных тестов.

Более «экстремальные» подходы также способствуют непрерывному тестированию на протяжении всего жизненного цикла разработки, а также постоянному наличию работающего (или безошибочного) продукта. Более структурированные или основанные на «водопаде» подходы пытаются оценить большинство рисков и разработать подробный план для программного обеспечения до начала реализации (кодирования), а также избежать значительных изменений в конструкции и перекодирования на более поздних этапах планирования жизненного цикла разработки программного обеспечения. После определения структуры программы и основных типов передаваемых данных определяются связи между модулями, решается вопрос об их взаимодействии.

От правильности выбора функций и их распределения зависит эффективность и правильность разработки сложных проектов. Наиболее распространенным подходом к проектированию является программирование сверху вниз.

Этот подход постоянно используется в разработке проектов. Нисходящее программирование основывается на последовательном разбиении сложного проекта на модули, которые можно решать независимо от других модулей, причем если модуль выходит объемным, его так же можно разбивать на отдельные куски программного кода. В итоге, получается проект с древовидной структурой, окончание каждой ветви которого является программный модуль.

На каждом этапе выполняется последовательное уточнение функций, реализованных модулями. Эта детализация выполняется до тех пор, пока не будет достигнуто элементарное представление операции при реализации каждой функции.

Нисходящее программирование позволяет реализовывать сложные проекты, разбивая его на модули. При этом создание и отладка каждого модуля по отдельности является простой задачей, которую можно доверить программистам с малым опытом работы.

3.2 Технология отладки программы.

Отладка — это процесс поиска и устранения дефектов или проблем в компьютерной программе, которые мешают правильной работе программного обеспечения или системы.

Тактика отладки может включать интерактивную отладку, анализ потока управления, модульное тестирование, интеграционное тестирование, анализ файла журнала, мониторинг на уровне приложения или системы, дампы памяти и профилирование.

Процесс отладки включает в себя:

- планирование отладки;
- получение результатов исполнения программы;
- анализ результатов исполнения, обнаружение и локализация ошибок;

- устранение ошибок: корректировка исходного текста программы.

Обычно первый шаг в отладке - попытка воспроизвести проблему. Это может быть нетривиальной задачей, например, с параллельными процессами. Кроме того, конкретная пользовательская среда и история использования могут затруднить воспроизведение проблемы.

После того, как ошибка воспроизведена, ввод программы, возможно, должен быть упрощен для облегчения отладки. Например, ошибка в компиляторе может привести к сбою при анализе большого исходного файла. Однако после упрощения контрольного примера только несколько строк из исходного файла могут оказаться достаточными для воспроизведения того же сбоя. Такое упрощение можно сделать вручную, используя подход «разделяй и властвуй». Программист попытается удалить некоторые части исходного теста и проверить, существует ли проблема. При отладке проблемы в графическом интерфейсе программист может попытаться пропустить какое-либо взаимодействие с пользователем из исходного описания проблемы и проверить, достаточно ли оставшихся действий для появления ошибок.

После того, как контрольный пример достаточно упрощен, программист может использовать инструмент отладчика для проверки состояния программы (значения переменных плюс стек вызовов) и отследить происхождение проблемы. В качестве альтернативы можно использовать трассировку.

3.3 Инструментальные средства разработки и отладки.

С каждым годом появляется все больше и больше интегрированных инструментов разработки. Одной из широко используемых для разработки программ на C # является система MS Visual Studio.

Microsoft Visual Studio — это линейка продуктов Microsoft, включающая интегрированную среду разработки программного обеспечения и несколько других инструментов. Эти продукты позволяют разрабатывать

как консольные приложения, так и приложения с графическим интерфейсом, в том числе поддерживающие технологию Windows Forms, а также веб-сайты, веб-приложения и веб-службы как в собственном, так и в управляемом кодах для всех платформ, поддерживаемых Microsoft Windows, Windows Mobile, Windows CE, .NET Framework, .NET Compact Framework и Microsoft Silverlight.

Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и как отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода (как например, Subversion и Visual SourceSafe), добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода на предметно-ориентированных языках программирования или инструментов для прочих аспектов цикла разработки программного обеспечения (например, клиент Team Explorer для работы с Team Foundation Server).

При разработке программного обеспечения используется .NET Framework. .NET Framework — программная платформа, выпущенная компанией Microsoft в 2002 году. Основой платформы является исполняющая среда Common Language Runtime (CLR), способная выполнять как обычные программы, так и серверные веб-приложения. .NET Framework поддерживает создание программ, написанных на разных языках программирования.

Считается, что платформа .NET Framework явилась ответом компании Microsoft на набравшую к тому времени большую популярность платформу Java компании Sun Microsystems (ныне принадлежит Oracle).

Хотя .NET является патентованной технологией корпорации Microsoft и официально рассчитана на работу под операционными системами семейства Microsoft Windows, существуют независимые проекты (прежде всего это Mono и Portable.NET), позволяющие запускать программы .NET на многих других операционных системах. — программная платформа, выпущенная компанией Microsoft в 2002 году. Основой платформы является исполняющая среда Common Language Runtime (CLR), способная выполнять как обычные программы, так и серверные веб-приложения. NET Framework поддерживает создание программ, написанных на разных языках программирования.

Разработка платформы началась в 1999 году. Официально о разработке новой технологии было объявлено 13 января 2000 года, в день, когда Билл Гейтс официально объявил о передаче поста главы Microsoft Стиву Балмеру. В этот день руководством корпорации была озвучена новая стратегия компании, получившая название Next Generation Windows Services (сокр. NGWS, рус. Новое поколение служб Windows). Новая стратегия должна была объединить в единый набор существующие и будущие разработки Microsoft для предоставления возможности пользователям работать с Всемирной паутиной с беспроводных устройств, обладающих доступом в Интернет, как со стационарных компьютеров.

Во время проходящей в тот день пресс-конференции Балмер заявил, что несмотря на огромные возможности домашних компьютеров, корпорация считает важным обеспечение гарантированной работы служб нового поколения и на устройствах, отличных от ПК. Что касается Гейтса, то аналитик компании Patricia Seybold Group Энн Томас Мейнс, проанализировав его слова, заявила о том, что компании будет полезен отход от ориентации исключительно на стационарные компьютеры и переход к разработке под

мобильные устройства. По её оценке, Гейтс провозгласил курс на переход к новой эре работы с компьютером, которая не будет жёстко привязана к Windows.

Ввиду небольшой мощности источников питания мобильных устройств, хранение и передача приложений должна осуществляться серверами, тогда как на тот момент практически вся пользовательская информация и ПО хранились на стационарных компьютерах локально. Тогда идея перехода к «сервероцентрической» модели имела крепкую поддержку среди руководителей крупнейших IT-компаний. Так, например, Скотт МакНилли, глава Sun Microsystems объявил о приобретении компании офисного ПО, конкурировавшего с Microsoft Office, для перехода к сервер-клиентскому ПО.

У Microsoft на тот момент было множество причин перехода к новой стратегии. Компания доминировала на рынке операционных систем и веб-браузеров, обладала множеством наработок в области ПО для Интернета, включая порталы MSN и WebTV, а также имела долю в компаниях, занимавшихся предоставлением ПО в прокат через Интернет. Кроме того, как позднее вспоминал Андерс Хейлсберг, у корпорации имелось множество различных (и зачастую несовместимых между собой) сред и технологий программирования, поскольку разработка инструментов для программистов была языкоориентированной, то есть для Visual Basic существовал свой набор приложений, а для C++ — свой. Поэтому одной из целей разработки новой платформы, было объединение всех наиболее удачных наработок в рамках единой платформы и их унификация. Кроме того, ставилась задача следования всем актуальным тенденциям в области программирования на тот момент. Так, например, новая платформа должна была напрямую поддерживать объектно-ориентированность, безопасность типов, сборку мусора и структурную обработку исключений. Кроме того, корпорации необходимо

было предоставить свой ответ набиравшей популярность платформе Java от Sun Microsystems.

Согласно озвученным на тот момент планам, в них входила разработка новой версии операционной системы, а также новой версии среды разработки для программистов, которая была бы предназначена для разработки веб-ориентированных приложений. Кроме того, переведён на новую платформу должен был быть и Microsoft Office. Пол Моритц, вице-президент Microsoft, в качестве одной из будущих служб новой стратегии привёл в пример службу Passport, представлявшую собой электронный бумажник, позволяющий пользователям зарегистрировавшись однажды совершать онлайн-покупки без необходимости ввода на различных сайтах номеров кредитных карт и персональных данных. Тогда же, на конференции, Балмер привёл пример реализации новой стратегии, которая позволит пациентам решать множество их проблем через Интернет, начиная с чтения своей медицинской карты и заканчивая оплатой счетов от врачей за медицинские услуги.

По словам Моритца, основой для новой стратегии компании послужила разработка Windows DNA 2000, представленная в сентябре 1999 года. Windows DNA 2000 объединила в себе новую версию СУБД SQL Server и новые программные инструменты для облегчения разработки веб-приложений.

В феврале 2000 года Стив Балмер, выступая на конференции VBITS 2000 в Сан-Франциско, рассказал о новой версии Visual Basic и Visual C++, являвшихся до той поры краеугольными камнями Visual Studio. Поскольку график выхода версий Visual Studio к тому моменту был изрядно нарушен (Visual Studio 6.0 вышла в сентябре 1998 года, а до этого выпуски входящих в её состав инструментов происходили ежегодно), то информация о нововведениях вызывала изрядный интерес.

Тогда же Балмер и представил концепцию стратегии NGWS (рабочее название которой Балмер счёл весьма ужасным и пообещал сменить его к

моменту выпуска) в виде некоей программной платформы, которая позволит пользователям Windows получать доступ к множеству веб-служб. Для этого Microsoft рассчитывала значительно изменить Windows-платформу. Первым шагом к новой версии платформы стала Windows 2000, которую Балмер назвал основой «пользовательского знакомства с Интернет». Windows 2000 являлась одной из частей технологии Windows Distributed interNet Applications (DNA) 2000, в рамках которой была представлена последняя на тот момент схема трёхуровневой модели программирования, впервые представленная на Windows NT 4.0.

На тот момент данная схема состояла из клиентской части (Windows-или веб-приложение), среднего уровня, базировавшегося на компонентах COM+, и базы данных (как правило SQL Server. NGWS должна была сменить путь разработки Windows DNA на исключительное использование XML как стандарта обмена данными. XML был выбран потому, что являлся развитием языка разметки HTML, ставшего основным языком разметки во Всемирной паутине, а кроме того, он предоставлял удобный способ описания данных. В модели NGWS каждая часть трёхзвенной структуры (клиентское приложение, промежуточный слой и база данных) должна была взаимодействовать с остальными при помощи XML-пакетов. Для упрощения разработки кода с использованием XML предполагалось, что новая версия Visual Studio будет генерировать XML-код, освобождая разработчиков от необходимости ручного написания XML-кода.

Так, например, планировалось, что разработчики смогут писать веб-сайт для электронной коммерции целиком на новой версии Visual Basic, а благодаря тому, что обмен информацией будет происходить при помощи XML, разработчики смогут создавать клиентские приложения, функционирующие на Linux, Solaris и Mac OS. То есть для того, чтобы приложение или операционная система могли взаимодействовать друг с другом нужна, была лишь поддержка стандарта с их стороны.

Для демонстрации своего замысла Microsoft запустила веб-сайт Passport, зайдя на который, пользователь мог бесплатно завести себе электронный кошелёк (eWallet), упрощавший процесс совершения покупок в интернет-магазинах.

Основной идеей при разработке .NET Framework являлось обеспечение свободы разработчика за счёт предоставления ему возможности создавать приложения различных типов, способные выполняться на различных типах устройств и в различных средах.

Вторым принципом стало ориентирование на системы, работающие под управлением семейства операционных систем Microsoft Windows.

Программа для .NET Framework, написанная на любом поддерживаемом языке программирования, сначала переводится компилятором в единый для .NET промежуточный байт-код Common Intermediate Language (CIL) (ранее назывался Microsoft Intermediate Language, MSIL). В терминах .NET получается сборка, англ. assembly. Затем код либо исполняется виртуальной машиной CLR, либо транслируется утилитой NGen.exe в исполняемый код для конкретного целевого процессора. Использование виртуальной машины предпочтительно, так как избавляет разработчиков от необходимости заботиться об особенностях аппаратной части. В случае использования виртуальной машины CLR, встроенный в неё JIT-компилятор «на лету» (just in time) преобразует промежуточный байт-код в машинные коды нужного процессора. Современная технология динамической компиляции позволяет достигнуть высокого уровня быстродействия. Виртуальная машина CLR также сама заботится о базовой безопасности, управлении памятью и системе исключений, избавляя разработчика от части работы.

Архитектура .NET Framework описана и опубликована в спецификации Common Language Infrastructure (CLI), разработанной Microsoft и утверждённой ISO и ECMA. В CLI описаны типы данных .NET, формат

метаданных о структуре программы, система исполнения байт-кода и многое другое.

Объектные классы .NET, доступные для всех поддерживаемых языков программирования, содержатся в библиотеке Framework Class Library (FCL). В FCL входят классы Windows Forms, ADO.NET, ASP.NET, Language Integrated Query, Windows Presentation Foundation, Windows Communication Foundation и другие. Ядро FCL называется Base Class Library (BCL).

3.4 База данных

Функционирование современного медицинского учреждения все в большей степени связано с обработкой огромного количества данных.

Практически все аспекты медицинской деятельности в той или иной степени связаны с обработкой и хранением большого количества информации. Поэтому все чаще встает вопрос о скорости доступа к данным и надежности их хранения.

Решением проблемы скорости доступа к данным и надежности их хранения является применение технологии SQL Server.

Microsoft SQL Server – комплексная платформа баз данных, обеспечивающая управление данными в масштабе предприятия и оснащенная интегрированными средствами управления интеллектуальными ресурсами предприятия (BI). Ядро СУБД SQL Server обеспечивает более безопасное и надежное хранение реляционных и структурированных данных, позволяя создавать и обслуживать высокопроизводительные приложения обработки данных класса предприятия.

Основой системы обработки данных предприятия является ядро базы данных SQL Server. Microsoft SQL Server также включает лучшие разработки в области анализа, подготовки отчетов, интеграции и рассылки уведомлений. Это позволяет создавать и развертывать экономически эффективные решения BI, которые позволят сотрудникам вашей компании доставлять данные в

любое место предприятия при помощи показателей, цифровых панелей, веб-служб и мобильных устройств.

3.5 Преимущества Microsoft SQL Server

Важным преимуществом Microsoft SQL Server является тесная интеграция с Microsoft Visual Studio, Microsoft Office System и новым комплектом средств разработки, включая Business Intelligence Development Studio. Являетесь ли вы разработчиком, администратором баз данных, информационным работником или ответственным лицом, SQL Server 2005 предоставляет решения, которые помогут вам использовать информацию более эффективно.

Масштабируемость и надежность. SQL Server обеспечивает практически неограниченный рост объемов данных за счет увеличения надежности и масштабируемости системы, используя все преимущества мультипроцессорной обработки данных. SQL Server Enterprise Edition под управлением ОС Windows Server Datacenter Edition обеспечивает параллельность обработки данных на 32 CPU и может использовать объем ОЗУ до 64 Гб.

Скорость построения решений. SQL уменьшает время создания, развертывания и выхода на рынок современных приложений для задач бизнеса, электронной коммерции, использует встроенный отладчик T-SQL. Совершенствует и ускоряет процесс поиска данных, упрощает управление, позволяет использовать создаваемые пользователем функции в других приложениях, предоставляет широкие возможности для создания Web приложений.

Рекордные показатели скорости. Еще до окончательного выхода на рынок система SQL Server установила новый мировой рекорд по производительности, далеко опередив конкурирующие решения на различных платформах.

Полная Web-ориентированность. Осуществление запросов, анализ и управление данными через Web. Использование языка XML для обмена данными между удаленными системами. Простой и безопасный доступ к данным с помощью Web-браузеров с использованием Firewall, быстрый поиск необходимых документов. Анализ потоков данных и получение информации о пользователях, в том числе и через Web.

Вывод

В данном проекте производилась разработка и отладка программного обеспечения, разработанного на языке программирования C#. Основная сложность при разработке программного обеспечения заключается в создании корректно работающей клиентской и серверной части. Для решения этой задачи необходимо не только знание правил связывания сервера приложения с сервером БД MSSQL, но и умение правильно настроить взаимодействие клиентской и серверной части приложения.

БД, использовавшаяся для отладки и тестирования операционной среды, согласно заданию на дипломный проект должна иметь сложную структуру: несколько сущностей, логически связанных между собой. Для проектирования базы данных была выбрана предметная область – медицинская история болезни. При проектировании использовалось CASE-средство – программный продукт ERWin; использовалась методология IDEF1X. База данных была реализована с применением сервера баз данных Microsoft SQLServer версии 2005.

4 ПСИХОЛОГИЯ ТРУДА ВРАЧА И ПАЦИЕНТА ПРИ ПРИМЕНЕНИИ В РАБОТЕ ЭЛЕКТРОННОЙ КАРТЫ

Введение

Трудовая психология, отрасль прикладной психологии, которая изучает психологические аспекты и закономерности человеческого труда. Психология труда начала формироваться на рубеже 19 и 20 веков, так как именно в это время активно стали применяться новые технологии в производстве и автоматизация повлияла на человечество очень значительно.

Возникновение психологии труда связано с началом научной организации труда. На первом этапе развития наиболее важной проблемой была проблема выбора профессий для новых рабочих мест. Анализ различий в производительности труда среди работников, прошедших примерно одинаковую подготовку, привел к мысли о существовании более или менее стойких индивидуальных различий в сфере так называемых профессиональных способностей. Специалистами были разработаны специальные тесты, которые должны были определять склонности соискателей и их способности к той или иной работе. Из этого необходимость изучения психологии профессий вышла на первый план. Были выявлены различия в профессиональных склонностях, интересах и мотивах, побуждающих людей предпочитать одни профессии другим, были организованы специальные консультационные бюро для помощи подросткам в выборе профессии.

Возникла специальная отрасль психологии труда - профориентация и профессиональные консультации. Появились специальные исследования закономерностей развития профессиональных навыков и качеств, важных для различных видов труда. Целью данного раздела психологии труда является разработка рекомендаций по совершенствованию методов обучения и применению специальных методов упражнений и тренировок.

Важной областью психологии труда было изучение колебаний трудоспособности, связанных с утомляемостью, ежедневным ритмом, и обоснование оптимального режима работы, при котором производительность и качество работы претерпят наименьшие изменения в течение рабочего дня, рабочей недели. и др. Современная психология труда разрабатывает специальные методики для измерения утомления и степени снижения работоспособности. В этой области это тесно связано с физиологией труда.

В психологии труда накоплен огромный материал о проблемах трудоспособности и утомляемости, влиянии на человека условий труда, характере выполняемых операций, монотонности и опасности труда, необычных и экстремальных условиях труда, мотивации труда, развитии потребностей и способностей человека в процессе коллективной работы и т. д. Одной из задач психологии труда является рациональное переустройство профессий, выяснение психологически оптимального сочетания включенных в них операций, научное обоснование их целесообразной автоматизации, которая важна для повышения производительности труда. Она координирует свои усилия со специалистами в области механизации и автоматизации. Изучение психологических причин возникновения чрезвычайных ситуаций привело к разработке специальных инструментов для профессионального отбора и предотвращения ошибочных действий с использованием специальных методов упражнений и тренировок..

Изучение психологических особенностей конкретных видов труда, составление профессиограмм (содержательное описание профессий и

профессиональной деятельности с точки зрения включения и использования психических свойств и способностей человека), определение набора профессионально значительных личностных качества способствовали формированию специальных направлений психологии профессий (например, психологии авиационной, космической, водительских профессий, конвейерного труда, с.-х. профессий и т.д.).

Наряду с экспериментальным методом аналитические методы занимают большое место в психологии труда. Применяется метод специальных упражнений, связанный с использованием различных устройств, имитирующих основные особенности профессиональной деятельности. Важную роль играют методы вариационной статистики.

В условиях современной научно-технической революции психология труда призвана изучать новые условия, формы и возможные стимулы трудовой деятельности, новые профессии и требования к технически оснащенному труду. Трудовая психология имеет тесные контакты с трудовой социологией, социальной психологией, инженерной психологией, организационной и экономической психологией, специфической экономикой, промышленной этикой, эргономикой, физиологией и гигиеной труда, кибернетикой, с набором управленческих дисциплин, прикладной математикой, квалиметрией, технической эстетикой и т.д.

4.1 Средства труда или средства производства

Средства труда или средства производства — это совокупность средств и предметов труда, используемых человеком при производстве материальных благ. Средства производства составляют материальный фактор производительных сил; В том числе технологии производства, формируют материально-техническую базу общества. К средствам труда и, прежде всего, к инструментам, относятся машины, машины, инструменты, с помощью

которых человек воздействует на природу, а также промышленные здания, земля, каналы, дороги и т. Д. Использование и создание средств труда является характерной чертой человека. трудовая деятельность.

Предметы труда — это субстанция природы, на которую человек воздействует в процессе труда, чтобы приспособить его для личного или промышленного потребления. Субъект труда, который уже подвергся воздействию человеческого труда, но предназначен для дальнейшей переработки, называется сырьем. Некоторые готовые продукты могут также входить в производственный процесс как предмет труда (например, виноград в винодельческой промышленности, животный жир в кондитерской промышленности). Определяющая роль в средствах производства принадлежит орудиям труда. По мере их развития и совершенствования растет техническое оснащение труда, изменяется роль человека в производственном процессе, увеличивается его господство над природой.

Уровень развития средств труда является важнейшим показателем технического прогресса. Их улучшение приводит к глубоким качественным изменениям в технике и технологии производства, к изменению производственных отношений; определяет переход от одного способа производства к другому. Научно-техническая революция внесла фундаментальные изменения в орудия труда, заменив машины традиционного типа комплексами автоматов, содержащих элемент автоматического регулирования и управления производственным процессом; вызваны качественные сдвиги в предметах труда, заключающиеся в использовании искусственных синтетических материалов, ослабляющие зависимость производства от предметов труда, данных природой. Форма собственности на средства производства характеризует общественные и производственные отношения людей, классовую структуру общества и то, как производитель связан со средствами производства.

В этом дипломном проекте было разработано программное обеспечение, которое облегчает работу врача, в данном случае уролога. В настоящее время в связи с ростом объема информации, бурным развитием компьютерных технологий и реализацией президентских национальных проектов они активно развивают и поддерживают использование компьютерных систем в медицине.

Есть много веских причин для перевода существующей структуры медицинских учреждений на компьютерную основу. Сегодня, благодаря широкому развитию и внедрению цифровых компьютерных технологий и Интернета во всех сферах жизни человека, в том числе в медицинской практике, невозможно представить медицинский диагностический процесс без той или иной формы использования этих технологий. В последние годы программы ведения бухгалтерского учета и отчетности по медицинской документации, диагностики различных патологических процессов, прогнозирования развития и основных заболеваний активно используются и совершенствуются в различных областях медицины. Как уже упоминалось выше, уровень развития средств труда является важнейшим показателем технического прогресса. Улучшение результатов приводит к глубоким качественным изменениям в технологии и технологии производства, к изменению производственных отношений; определить переход от одного способа производства к другому. Актуальность этого проекта однозначна.

Благодаря внедрению электронной карты клиента, достигаются следующие результаты:

- Профилактика населения упрощается, становится проще и быстрее;
- Предоставляет лечащему врачу возможность быстрого и удобного ввода информации о пациенте;

- Обеспечивает безопасность доступа к ЕМС с учетом прав доступа пользователей к медицинской информации, утвержденных медицинским учреждением;
- Позволяет просматривать ЭМС пациента и быстро находить необходимую информацию в больших объемах медицинской документации;
- Позволяет создавать на основе ЕМС различные выписки, сертификаты, эпикризы, распечатывать их и хранить копии этих документов;
- Обеспечивает возможность визуального просмотра медицинских данных пациента: диагнозы, список рецептов, составление различных графиков и т.п.;
- Упрощает и ускоряет процесс диагностики, сбора данных пациента, назначения лечения;
- Просмотр и редактирование карт пациентов с любого компьютера или телефона, в любой точке мира;
- Учет всех жалоб, симптомов, анализов и выполненных процедур. Запись результатов;
- Повышение уровня обслуживания.

Факторы, влияют на уровень работы, результат работы и, соответственно, на эмоционально-психологическое состояние врача и пациента.

4.2 Методы психологии труда

Существуют различные методы психологии труда. Ю. В. Котелова приводит следующую классификацию методов психологии труда. Она разделяет всё разнообразие методов на три больших раздела А, Б и В.

Раздел А, в свою очередь, подразделяется на три блока. К первому относятся различные варианты метода наблюдения.

Метод наблюдения:

- Метод непосредственного наблюдения профессиональной деятельности;
- Трудовой метод – участие самого психолога в конкретном виде труда.

В связи с трудностью применения этих методов, к ним прилагаются вспомогательные методы и средства:

- Измерения различных параметров процесса труда;
- Алгоритмический способ описания трудовой деятельности;
- Анализ продуктов трудовой деятельности;
- Анализ ошибочных действий, несчастных случаев и аварий;
- Экспериментально - психологические и физиологические пробы в ходе протекания профессиональной деятельности.

Второй блок раздела А содержит разновидности метода опроса:

- Беседа;
- Устный опрос (интервью);
- Письменный опрос (Анкета);
- Биографические и автобиографические данные (психологический анализ);
- Метод обобщения независимых характеристик

Третий блок методик раздела А – это анализ различной производственной документации помогает в работе методикам из I и II блока.

Далее в классификации методов идёт раздел Б, содержащий виды экспериментальных методик:

- Лабораторный эксперимент

- Абстрактный;
- Модель отдельных элементов трудового процесса;
- Модели профессиональной деятельности в целом;
- Реальная трудовая деятельность в условиях лаборатории;
- Метод тестов
- Производственный эксперимент
- Внесение разнообразных изменений в оборудование, технологию, приёмы, темп и ритм работы.
- Метод искусственной деавтоматизации.

К разделу В Котелова Ю. В. относит разнообразные статистические методы работы с данными. Этот раздел динамично связывает разделы А и Б.

Е. А. Климов в своей классификации методов психологии труда к уже перечисленным эмпирико-познавательным методам из классификации Ю. В. Котеловой добавляет методы построения теории:

- Методы построения простых и сложных теоретических объектов;
- Методы интерпретации;
- Номологизации;
- Определения;
- Верификации;
- Доказательства;
- Объяснения.

Преобразующие или конструктивные методы психологии труда:

- методы обучения, развития субъекта труда, развития профессионально-важных свойств (тренажеры, деловые игры);
- консультирование – метод обогащения и преобразования знаний и представлений человека о себе, о профессиях, о соотношении своих

возможностей и интересов с определенной группой или несколькими группами профессий;

- методы коррекции поведения, состояния субъекта труда;
- методы реконструкции - преобразования рабочего пространства, органов управления и средств индикации, режима труда и отдыха, способов планирования труда, нормирования и контроля (в рамках организационного проектирования и реконструкции трудовых задач и форм организации труда).

4.3 Задачи психологии труда

Основные задачи психологии труда на современном этапе напрямую связаны с социальными задачами улучшения производственных отношений и улучшения качества труда, улучшения условий жизни, ликвидации чрезвычайных ситуаций, демократизации и формирования психологического типа работника, соответствующего рабочей культуре. Различают следующие основные направления исследований: - способы повышения производительности труда - рационализация труда и отдыха, - формирование профессиональной мотивации, - оценка профессиональной пригодности, - оптимизация отношений в трудовых коллективах, - психологическая подготовка молодежи к работе, - проблемы и кризисы профессионального развития - профессиональные стрессы и конфликты - психология безопасности.

В качестве методов исследования в психологии труда используются: - натурный и лабораторный эксперимент, - наблюдение, - собеседования, - опросники, - психологические тесты, - тренажеры, - метод функциональных тестов и другие методы.

Проникновение передовых технологий в сектор медицинской помощи, широкое использование новейших технологий, а также применение принципов эффективного управления срочно требуют учета личных факторов

как в учебном процессе, так и на всем профессиональном пути субъекта. медицинская деятельность. Сегодня есть достаточные основания рассматривать профессиональное становление субъекта медицинского труда как процесс, состоящий из двух частей, включая формирование совокупности знаний, навыков, с одной стороны, и профессионально значимых психологических качеств личности - с другой. Первому из этих компонентов традиционно уделялось значительное внимание, что было подтверждено в последние годы концепцией непрерывного медицинского образования, однако систематических исследований проблем психологии медицинского труда и психологической поддержки развития личности врача на всех этапах профессионального генеза. В то же время никто не сомневается, что эффективность специалиста в области здравоохранения во многом определяется не только уровнем его профессиональных навыков, но и характером профессионального самоопределения, отношением к профессии, профессиональной идентификацией. Принадлежность к профессиям субъектно-субъектного типа, деятельность врача осуществляется в условиях повышенных социально-психологических потребностей и связана с высоким психическим и психоэмоциональным стрессом. Активность врача в условиях профессионального стресса, особенности эмоционального реагирования на различные ситуации профессиональной деятельности, факторы, влияющие на профессиональное и личностное развитие субъекта лечебного процесса, индивидуальные психологические особенности, присущие врачам различных специализаций, являются проблемами, которые имеют недостаточно изучены, несмотря на высокую степень востребованности их научно-практического отражения. Можно сказать, что вопросы психологии медицинского труда и профессионального генезиса личности врача относятся к числу наиболее важных и малоизученных проблем теоретической и прикладной психологии в целом и психологии труда, в частности.

Высокая социальная значимость медицинской работы связана с факторами дестабилизирующего воздействия на личность врача, с одной стороны, и отсутствием научно обоснованной системы психологической поддержки развития субъекта медицинского труда на всех этапах его профессиональной деятельности. А жизненный путь, с другой стороны, является сущностью проблемной ситуации, обуславливающей назревшую социальную необходимость определения психологической специфики профессиональной деятельности врача и формирования на этой основе концепции последовательной психологической поддержки развития, профессиональная личность. Другими словами: теоретическая, методологическая и практическая актуальность системного психологического исследования медицинского труда как направления современной трудовой психологии очевидна.

Вывод

В этом дипломном проекте была разработана и внедрена программа, направленная на улучшение и облегчение работы врача, а также на повышение уровня ухода за пациентами. В процессе было выявлено, что данное программное обеспечение положительно влияет на эмоциональное и психологическое состояние пользователя. В результате использования электронной карты клиента выяснилось, что процент ошибок и потеря данных были уменьшены, а время обслуживания пациента уменьшено.

Проникновение передовых технологий в сектор медицинского обслуживания, широкое использование новейших технологий, а также применение эффективных принципов управления срочно требуют учета личных факторов как в процессе обучения, так и на протяжении всего профессионального пути субъекта. медицинской деятельности.

Вопрос психологии труда изучался и учитывался при разработке программного обеспечения.

ЗАКЛЮЧЕНИЕ

В результате выполнения дипломного проекта была разработана распределенная система ведения истории болезни, предназначенная для медицинских целей в качестве средства ведения централизованной базы данных пациентов.

В процессе выполнения было создано серверное приложение, изучены технологии разработки и отладки программ в среде MS Visual Studio.

Проектирование выполнено с использованием моделирования на языке UML.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. СанПиН 2.2.2/2.4.1340-03 «Гигиенические требования к персональным электронно-вычислительным машинам и организации работы» 30 мая 2003г.
2. *Маклаков С.В.*, ERWin, BPWin. CASE-средства разработки информационных систем, М: Диалог-МИФИ, 1999. – 256 с.: ил.
3. *Боггс У.* UML Rational Rose. / *У. Боггс, М. Боггс.* – М.: Издательство «Лори», 2000. – 582 с., ил.
4. *Буч Г.* Язык UML. Руководство пользователя / *Г. Буч, Д. Рамбо, А. Джекобсон.* – М.: ДМК, 2000. – 432 с., ил.
5. Эндрю Троелсен. Язык программирования C# 2010 и платформа .NET 4.0 = Pro C# 2010 and the .NET 4.0 Platform. — 5-е изд. — М.: Вильямс, 2010.
6. Герберт Шилдт. C# 4.0: полное руководство = C# 4.0 The Complete Reference. — М.: «Вильямс», 2010.
7. Джон Скит. C#: программирование для профессионалов, 2-е издание = C# in Depth, 2nd Edition. — М.: «Вильямс», 2011.
8. Кристиан Нейгел, Карли Уотсон и др. Visual C# 2010: полный курс = Beginning Microsoft Visual C# 2010. — М.: Диалектика, 2010.
9. Трей Нэш. C# 2010: ускоренный курс для профессионалов = Accelerated C# 2010. — М.: Вильямс, 2010
10. Кристиан Нейгел, Билл Ивсен, Джей Глинн, Карли Уотсон, Морган Скиннер. C# 4: Платформа .NET 4 для профессионалов = Professional C# 4 and .NET 4. — М.: Диалектика, 2010

ПРИЛОЖЕНИЕ А

ЛИСТИНГ ПРОГРАММЫ

Program.cs

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;

namespace MedProg
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new MainForm());
        }
    }
}
```

Patients.cs

```
using System;
using System.Collections.Generic;
using System.Text;
using System.IO;
using System.Xml.Serialization;

namespace MedProg
{
    public class Patients:IDisposable
    {
        public List<Patient> PatientList = new List<Patient>();
        bool Disposed = false;
        public DateTime LastDrop;

        public void Serialize()
        {
            XmlSerializer xml = new XmlSerializer(typeof(Patients));

            TextWriter writer = new StreamWriter("Base.xml");
            xml.Serialize(writer, this);
            writer.Close();
        }
    }
}
```

```

public static Patients DeSerialize()
{
    return DeSerialize("Base.xml");
}

public static Patients DeSerialize(string FromFile)
{
    Patients readedbook;

    XmlSerializer xml = new XmlSerializer(typeof(Patients));

    TextReader reader = new StreamReader(FromFile);
    readedbook = (Patients)xml.Deserialize(reader);
    reader.Close();

    return readedbook;
}

public void CreateBackup()
{
    Directory.CreateDirectory("Backups");
    File.Copy("Base.xml", "Backups\\Base_" +
DateTime.Now.ToString().Replace(":", "-") + ".xml");
}

public void Dispose()
{
    if (!Disposed)
    {
        Disposed = true;
        Serialize();

        CreateBackup();
    }
}

~Patients()
{
    if (!Disposed)
    {
        Dispose();
        Disposed = true;
    }
}

public List<Patient> Select()
{
    return PatientList;
}

public Patient FindPatient(Guid PatientId)
{
    foreach (Patient p in PatientList)
    {
        if (p.PatientIdField == PatientId)
            return p;
    }

    return null;
}
}

```

```
}
```

MyWorkSheet.cs

```
using System;
using System.Reflection; // For Missing.Value and BindingFlags
using System.Runtime.InteropServices; // For COMException
using Microsoft.Office.Interop.Excel;

namespace MedProg
{
    class MyWorkSheet
    {
        _Worksheet worksheet;
        public MyWorkSheet(_Worksheet w)
        {
            worksheet = w;
        }

        public void SetCell(int row, int column, object value)
        {
            const int Base = 26;
            string colstr = "";
            int correction = 0;

            row++;

            if (column < Base)
            {
                colstr = colstr.PadLeft(1, (char)('A' + column));
            }
            else
            {
                while (column > 0)
                {
                    colstr = colstr.PadLeft(colstr.Length + 1,
(char)('A' + (column % Base) - correction));
                    column = column / Base;
                    correction = 1;
                }
            }

            SetCell(colstr + row.ToString(), value);
        }

        public void SetCell(string Address, object value)
        {
            Range r = worksheet.get_Range(Address, Address);
            r.Value2 = value;
        }
    }
}
```

ExcelExport.cs

```
using System;
using System.Reflection; // For Missing.Value and BindingFlags
using System.Runtime.InteropServices; // For COMException
using Microsoft.Office.Interop.Excel;
```

```

using System.Diagnostics;
using System.IO;
using System.Collections.Generic;

namespace MedProg
{
    class ExcellExport
    {
        static public bool Output(Patients book, string filename)
        {
            bool Success = false;

            Application app = new Application();
            if (app == null)
            {
                return Success;
            }

            app.Visible = false;

            Workbooks workbooks = app.Workbooks;

            _Workbook workbook =
workbooks.Add(XlWBATemplate.xlWBATWorksheet);

            Sheets sheets = workbook.Worksheets;

            _Worksheet worksheet = (_Worksheet)sheets.get_Item(1);
            if (worksheet == null)
            {
                return Success;
            }

            ExportData(book, worksheet);

            string DirName = Environment.CurrentDirectory + @"\Excel\";
            if (!Directory.Exists(DirName))
            {
                Directory.CreateDirectory(DirName);
            }

            string FileName = DirName + "Base_" +
DateTime.Now.ToString().Replace(".", "_").Replace(":", "-");
            workbook.SaveAs(FileName);

            if (filename.Length > 0)
                workbook.SaveAs(filename);

            try
            {
                // If user interacted with Excel it will not close when
the app object is destroyed, so we close it explicitly
                workbook.Saved = true;
                app.UserControl = false;
                app.Quit();

                Success = true;
            }
        }
    }
}

```

```

        catch (COMException)
        {
            Console.WriteLine("User closed Excel manually, so we
don't have to do that");
        }

        return Success;
    }

    static void ExportHeader(MyWorkSheet mw)
    {
        //////////////////////////////////////
        // Основное

        mw.SetCell(0, 0, "Фамилия");
        mw.SetCell(0, 1, "Имя");
        mw.SetCell(0, 2, "Отчество");
        mw.SetCell(0, 3, "Дата рождения");
        mw.SetCell(0, 4, "Адрес");
        mw.SetCell(0, 5, "Телефон домашний");
        mw.SetCell(0, 6, "Телефон сотовый");

        //////////////////////////////////////
        // Жалобы

        mw.SetCell(0, 7, "Локализация боли");
        mw.SetCell(0, 8, "Локализация боли в мочевом пузыре");
        mw.SetCell(0, 9, "Локализация боли в уретре");
        mw.SetCell(0, 10, "Локализация боли в области вульвы");
        mw.SetCell(0, 11, "Локализация боли во влагалище");
        mw.SetCell(0, 12, "Локализация боли в мошонке");
        mw.SetCell(0, 13, "Локализация боли в промежности");
        mw.SetCell(0, 14, "Локализация боли тазовая боль");

        mw.SetCell(0, 15, "Боль связана с наполнением мочевого
пузыря");
        mw.SetCell(0, 16, "Боль связана с мочеиспусканием");
        mw.SetCell(0, 17, "Боль связана с движением");
        mw.SetCell(0, 18, "Боль связана с половым актом");

        mw.SetCell(0, 19, "Утечка мочи при позыве");
        mw.SetCell(0, 20, "Утечка мочи при кашле");
        mw.SetCell(0, 21, "Утечка мочи при чихании");
        mw.SetCell(0, 22, "Утечка мочи затруднились ответить");
        mw.SetCell(0, 23, "Утечка мочи ночью во сне");
        mw.SetCell(0, 24, "Утечки мочи нет");

        mw.SetCell(0, 25, "Учащение мочеиспускания");
        mw.SetCell(0, 26, "Ургентность");
        mw.SetCell(0, 27, "Затруднение при мочеиспускании");
        mw.SetCell(0, 28, "Чувство неполного опорожнения");
        mw.SetCell(0, 29, "Необходимость натуживания");
        mw.SetCell(0, 30, "Прерывистость струй мочи");
        mw.SetCell(0, 31, "Ноктурия");

        //////////////////////////////////////
        // Анамнез
        //

```

```

mw.SetCell(0, 32, "Роды");
mw.SetCell(0, 33, "Беременности");
mw.SetCell(0, 34, "Менопауза");
mw.SetCell(0, 35, "Эндометриоз");
mw.SetCell(0, 36, "Миома матки");

mw.SetCell(0, 37, "Болезни нервной системы");
mw.SetCell(0, 38, "Нет болезней нервной системы");
mw.SetCell(0, 39, "Рассеянный склероз");
mw.SetCell(0, 40, "Болезнь Паркинсона");
mw.SetCell(0, 41, "Осложнение остеохондроза позвоночника");
mw.SetCell(0, 42, "Грыжа диска");

mw.SetCell(0, 43, "Обострения цистита");

mw.SetCell(0, 44, "Нет инсульта");
mw.SetCell(0, 45, "Мозговой инсульт");
mw.SetCell(0, 46, "Спинальный инсульт");

////////////////////////////////////
// Результат обследования
//

mw.SetCell(0, 47, "Количество мочеиспусканий");
mw.SetCell(0, 48, "Ноктурия");
mw.SetCell(0, 49, "Средний объем");

mw.SetCell(0, 50, "Максимальная цистометрическая емкость");
mw.SetCell(0, 51, "При исследовании порога абд. давления
подтекание мочи на");
mw.SetCell(0, 52, "Подъемы детрузорного давления");

mw.SetCell(0, 53, "QMax");
mw.SetCell(0, 54, "при Det");
mw.SetCell(0, 55, "V мочи");
mw.SetCell(0, 56, "Остаточной мочи");

////////////////////////////////////
// Лечение
//

mw.SetCell(0, 57, "Тиббиальная нейромодуляция на аппарате
БиоБраво");
mw.SetCell(0, 58, "Экстрокорпоральная электромагнитная
стимуляция мышц тазового дна на аппарате Неотонус");

mw.SetCell(0, 59, "Инъекции ботулотоксина в сфинктер.");
mw.SetCell(0, 60, "Инъекции ботулотоксина в детрузор.");
mw.SetCell(0, 61, "Вапоризация лейкоплакии.");
mw.SetCell(0, 62, "Введение формообразующего геля");
mw.SetCell(0, 63, "Операция Берча");
mw.SetCell(0, 64, "Установка слинга");
mw.SetCell(0, 65, "Уретротомия");
mw.SetCell(0, 66, "ТУР простаты");

}

```

```

static string FindExclusiveField(List<string> list, List<string>
fulllist)
{
    foreach (string s in list)
    {
        if (!fulllist.Contains(s))
            return s;
    }

    return " ";
}

static bool IsInList(List<string> list, string s)
{
    return list.Contains(s);
}

static void ExportData(Patients p, _Worksheet w)
{
    MyWorkSheet mw = new MyWorkSheet(w);

    ExportHeader(mw);

    for (int i = 0; i < p.PatientList.Count; i++)
    {
        Patient cp = p.PatientList[i];
        int row = i + 1;

////////////////////////////////////
        // Основное

        mw.SetCell(row, 0, cp.SernameField);
        mw.SetCell(row, 1, cp.NameField);
        mw.SetCell(row, 2, cp.SecondNameField);
        mw.SetCell(row, 3, cp.BirthdayField);
        mw.SetCell(row, 4, cp.AddressField);
        mw.SetCell(row, 5, cp.PhoneHome);
        mw.SetCell(row, 6, cp.PhoneMobile);

////////////////////////////////////
        // Жалобы
        //

        List<string> tmplist = new List<string>();
        tmplist.Add("в мочевом пузыре");
        tmplist.Add("в уретре");
        tmplist.Add("в области вульвы");
        tmplist.Add("во влагалище");
        tmplist.Add("в мошонке");
        tmplist.Add("в промежности");
        tmplist.Add("тазовая боль");

        mw.SetCell(row, 7,
FindExclusiveField(cp.PainLocalization, tmplist));
        mw.SetCell(row, 8, IsInList(cp.PainLocalization,
tmplist[0]));
        mw.SetCell(row, 9, IsInList(cp.PainLocalization,
tmplist[1]));
        mw.SetCell(row, 10, IsInList(cp.PainLocalization,
tmplist[2]));

```

```

tmplist[3]));
tmplist[4]));
tmplist[5]));
tmplist[6]));

mw.SetCell(row, 11, IsInList(cp.PainLocalization,
tmplist[3]));
mw.SetCell(row, 12, IsInList(cp.PainLocalization,
tmplist[4]));
mw.SetCell(row, 13, IsInList(cp.PainLocalization,
tmplist[5]));
mw.SetCell(row, 14, IsInList(cp.PainLocalization,
tmplist[6]));

tmplist.Clear();
tmplist.Add("С наполнением мочевого пузыря");
tmplist.Add("С мочеиспусканием");
tmplist.Add("С движением");
tmplist.Add("С половым актом");

mw.SetCell(row, 15, IsInList(cp.PainLink, tmplist[0]));
mw.SetCell(row, 16, IsInList(cp.PainLink, tmplist[1]));
mw.SetCell(row, 17, IsInList(cp.PainLink, tmplist[2]));
mw.SetCell(row, 18, IsInList(cp.PainLink, tmplist[3]));

tmplist.Clear();
tmplist.Add("При позыве");
tmplist.Add("При кашле");
tmplist.Add("При чихании");
tmplist.Add("Затрудняюсь ответить");
tmplist.Add("Ночью во сне");
tmplist.Add("Нет");

mw.SetCell(row, 19, IsInList(cp.UrineLeak, tmplist[0]));
mw.SetCell(row, 20, IsInList(cp.UrineLeak, tmplist[1]));
mw.SetCell(row, 21, IsInList(cp.UrineLeak, tmplist[2]));
mw.SetCell(row, 22, IsInList(cp.UrineLeak, tmplist[3]));
mw.SetCell(row, 23, IsInList(cp.UrineLeak, tmplist[4]));
mw.SetCell(row, 24, IsInList(cp.UrineLeak, tmplist[5]));

mw.SetCell(row, 25, cp.FastUrinationField);
mw.SetCell(row, 26, cp.UrgentionField);
mw.SetCell(row, 27, cp.ProblemUrinationField);
mw.SetCell(row, 28, cp.NotFullBowelEmptyingField);
mw.SetCell(row, 29, cp.NeedStrainingEffortField);
mw.SetCell(row, 30, cp.SinusoidUrineStreamField);
mw.SetCell(row, 31, cp.NokturiField);

////////////////////////////////////
// Анамнез
//

mw.SetCell(row, 32, cp.LaborField);
mw.SetCell(row, 33, cp.PregnancyField);
mw.SetCell(row, 34, cp.Menopause);
mw.SetCell(row, 35, cp.Endometrioz);
mw.SetCell(row, 36, cp.MiomaMatki);

tmplist.Clear();
tmplist.Add("Нет");
tmplist.Add("Рассеянный склероз");
tmplist.Add("Болезнь Перкинсона");
tmplist.Add("Осложнения остеохондроза позвоночника");
tmplist.Add("Грыжа диска");

```

```

        mw.SetCell(row, 37,
FindExclusiveField(cp.NervousSystemDisease, tmplist));
        mw.SetCell(row, 38, IsInList(cp.NervousSystemDisease,
tmplist[0]));
        mw.SetCell(row, 39, IsInList(cp.NervousSystemDisease,
tmplist[1]));
        mw.SetCell(row, 40, IsInList(cp.NervousSystemDisease,
tmplist[2]));
        mw.SetCell(row, 41, IsInList(cp.NervousSystemDisease,
tmplist[3]));
        mw.SetCell(row, 42, IsInList(cp.NervousSystemDisease,
tmplist[4]));

        mw.SetCell(row, 43, cp.Cistit);

        tmplist.Clear();
        tmplist.Add("Нет");
        tmplist.Add("Мозговой");
        tmplist.Add("Спинальный");

        mw.SetCell(row, 44, IsInList(cp.Insult, tmplist[0]));
        mw.SetCell(row, 45, IsInList(cp.Insult, tmplist[1]));
        mw.SetCell(row, 46, IsInList(cp.Insult, tmplist[2]));

////////////////////////////////////
        // Результат обследования
        //

        mw.SetCell(row, 47, cp.DiuresisUrinationCountField);
        mw.SetCell(row, 48, cp.DiuresisNokturiField);
        mw.SetCell(row, 49, cp.DiuresisAverageValueField);

        mw.SetCell(row, 50, cp.MaxCistometryVolume);
        mw.SetCell(row, 51, cp.VolumeCistometryLeakField);
        mw.SetCell(row, 52, cp.VolumeCistometryPressureField);

        mw.SetCell(row, 53, cp.MikcionCistometryQMax);
        mw.SetCell(row, 54, cp.MikcionCistometryQmaxDet);
        mw.SetCell(row, 55, cp.MikcionCistometryV);
        mw.SetCell(row, 56, cp.MikcionCistometryRemain);

////////////////////////////////////
        // Лечение
        //

        tmplist.Clear();
        tmplist.Add("Тибиальная нейромодуляция на аппарате
БиоБраво 1 сеанс в неделю № 8 -10.");
        tmplist.Add("Экстрокорпоральная электромагнитная
стимуляция мышц тазового дна на аппарате Неотонус 2 раза в неделю №8-10.");

        mw.SetCell(row, 57, IsInList(cp.PhisioCure,
tmplist[0]));
        mw.SetCell(row, 58, IsInList(cp.PhisioCure,
tmplist[1]));

```

```

        mw.SetCell(row, 59, cp.OperCure1);
        mw.SetCell(row, 60, cp.OperCure2);
        mw.SetCell(row, 61, cp.OperCure3);
        mw.SetCell(row, 62, cp.OperCure4);
        mw.SetCell(row, 63, cp.OperCure5);
        mw.SetCell(row, 64, cp.OperCure6);
        mw.SetCell(row, 65, cp.OperCure7);
        mw.SetCell(row, 66, cp.OperCure8);
    }

}

}

```

FontStr.cs

```

using System;
using System.Collections.Generic;
using System.Text;
using System.Drawing;

namespace MedProg
{
    class FontDef
    {
        public FontDef(string s)
        {
            font = new Font("Times New Roman", 14, FontStyle.Regular);
            str = s + "\r\n";
        }
        public FontDef(Font f, string s)
        {
            font = f;
            str = s + "\r\n";
        }

        public Font font;
        public string str;
    }

    class FontHeader1 : FontDef
    {
        static Font cf = new Font("Times New Roman", 24, FontStyle.Regular,
GraphicsUnit.Point);
        public FontHeader1(string s)
            : base(cf, s) {}
    }

    class FontHeader2 : FontDef
    {
        static Font cf = new Font("Times New Roman", 18, FontStyle.Regular,
GraphicsUnit.Point);
        public FontHeader2(string s)
            : base(cf, s) { }
    }

    class FontHeader3 : FontDef
    {
        static Font cf = new Font("Times New Roman", 14, FontStyle.Bold,
GraphicsUnit.Point);
        public FontHeader3(string s)
            : base(cf, s) { }
    }
}

```

```
    }  
}
```

Patient.cs

```
using System;  
using System.Collections.Generic;  
using System.Text;  
using System.Xml.Serialization;  
  
namespace MedProg  
{  
    public class Patient  
    {  
        public Patient()  
        {  
            PatientId = Guid.NewGuid();  
        }  
  
        static string StrForReport(ref List<string> list)  
        {  
            string outs = "";  
            foreach (string s in list)  
            {  
                outs += s + "\n";  
            }  
  
            return outs;  
        }  
  
        public static string AgeLiteralSuffix(int years)  
        {  
            if ((years >= 10 &&  
                years <= 20) ||  
                years % 10 >= 5 ||  
                years % 10 == 0)  
            {  
                return "лет";  
            }  
  
            if (years % 10 == 1)  
            {  
                return "год";  
            }  
  
            return "года";  
        }  
  
        public string AgeLiteral()  
        {  
            TimeSpan age = DateTime.Now - BirthdayField;  
            return (age.Days / 365).ToString() + " " +  
AgeLiteralSuffix(age.Days / 365);  
        }  
  
        private Guid PatientId;  
  
        public Guid PatientIdField  
        {  
            get { return PatientId; }  
        }  
    }  
}
```

```

        set { PatientId = value; }
    }
    // 2

    // Фамилия
    private string Surname;

    public string SurnameField
    {
        get { return Surname; }
        set { Surname = value; }
    }

    // Имя
    private string Name;

    public string NameField
    {
        get { return Name; }
        set { Name = value; }
    }

    // Отчество
    private string SecondName;

    public string SecondNameField
    {
        get { return SecondName; }
        set { SecondName = value; }
    }

    // Дата Рождения
    private DateTime Birthday;

    public DateTime BirthdayField
    {
        get { return Birthday.Date; }
        set { Birthday = value; }
    }

    // Адрес
    private string Address;

    public string AddressField
    {
        get { return Address; }
        set { Address = value; }
    }

    // Телефон домашний
    public string PhoneHome;

    // Телефон мобильный
    public string PhoneMobile;

    // Специальность
    public string Speciality;

```

```

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

```

```

    // Жалобы

```

```

private string Complaints;

public string ComplaintsField
{
    get { return Complaints; }
    set { Complaints = value; }
}

// Боль
private string Pain;

public string PainField
{
    get { return Pain; }
    set { Pain = value; }
}

// Локализация (список)
public List<string> PainLocalization = new List<string>();

// Интенсивность (список)
public List<string> PainInt = new List<string>();

// Боль связана с... (список)
public List<string> PainLink = new List<string>();

// Учащение мочеиспускания
private bool FastUrination;
public bool FastUrinationField
{
    get { return FastUrination; }
    set { FastUrination = value; }
}

// Ургентность
private bool Urgention;
public bool UrgentionField
{
    get { return Urgention; }
    set { Urgention = value; }
}

// Утечка мочи(список)
public List<string> UrineLeak = new List<string>();

// Затруднение при мочеиспускании
private bool ProblemUrination;

public bool ProblemUrinationField
{
    get { return ProblemUrination; }
    set { ProblemUrination = value; }
}

// Чувство неполного опорожнения
private bool NotFullBowelEmptying;

public bool NotFullBowelEmptyingField
{
    get { return NotFullBowelEmptying; }
    set { NotFullBowelEmptying = value; }
}

```

```

// Необходимость натуживания
private bool NeedStrainingEffort;

public bool NeedStrainingEffortField
{
    get { return NeedStrainingEffort; }
    set { NeedStrainingEffort = value; }
}

// Прерывистость струй мочи
private bool SinusoidUrineStream;

public bool SinusoidUrineStreamField
{
    get { return SinusoidUrineStream; }
    set { SinusoidUrineStream = value; }
}

// Ноктурия
private bool Nokturi;

public bool NokturiField
{
    get { return Nokturi; }
    set { Nokturi = value; }
}

////////////////////////////////////
// 3

// Анамнез
private string Anamnes;

public string AnamnesField
{
    get { return Anamnes; }
    set { Anamnes = value; }
}

// Длительность жалоб
private string ComplaintsDuration;

public string ComplaintsDurationField
{
    get { return ComplaintsDuration; }
    set { ComplaintsDuration = value; }
}

// Роды
private string Labor;

public string LaborField
{
    get { return Labor; }
    set { Labor = value; }
}

// Беременности
private string Pregnancy;

public string PregnancyField

```

```

{
    get { return Pregnancy; }
    set { Pregnancy = value; }
}

// Менопауза лет
public string Menopause;

// Эндометриоз
public string Endometrioiz;

// Миома матки
public string MiomaMatki;

// Обострения цистита
public string Cistit;

// Операции
private string Surgery;

public string SurgeryField
{
    get { return Surgery; }
    set { Surgery = value; }
}

// Травмы
private string Trauma;

public string TraumaField
{
    get { return Trauma; }
    set { Trauma = value; }
}

// Болезни нервной системы (список)
public List<string> NervousSystemDisease = new List<string>();

// Инсульт
public List<string> Insult = new List<string>();

// Препараты
private string Drugs;

public string DrugsField
{
    get { return Drugs; }
    set { Drugs = value; }
}

// 4 Осмотр

// Наружное отверстие уретры
string ExternalUrethralOpening;

public string ExternalUrethralOpeningField
{
    get { return ExternalUrethralOpening; }
    set { ExternalUrethralOpening = value; }
}

// Пропалс влагалища

```

```

string VaginaProlapse;

public string VaginaProlapseField
{
    get { return VaginaProlapse; }
    set { VaginaProlapse = value; }
}

// Проллапс тазовых органов
private string PelvicProlapse;

public string PelvicProlapseField
{
    get { return PelvicProlapse; }
    set { PelvicProlapse = value; }
}

// Кашлевая проба
private bool CoughProbe;

public bool CoughProbeField
{
    get { return CoughProbe; }
    set { CoughProbe = value; }
}

// Проба Вальсальвы
private bool ValsalvaProbe;

public bool ValsalvaProbeField
{
    get { return ValsalvaProbe; }
    set { ValsalvaProbe = value; }
}

// Внеуретральная утечка
private bool ExternalUrethral;

public bool ExternalUrethralField
{
    get { return ExternalUrethral; }
    set { ExternalUrethral = value; }
}

// Результаты обследований

// Дневник мочеиспусканий
// Диурез (diuresis)
//Diuresis diuresis = new Diuresis();
// Дневной
private int DiuresisDay;

public int DiuresisDayField
{
    get { return DiuresisDay; }
    set { DiuresisDay = value; }
}

// Ночной
private int DiuresisNight;

public int DiuresisNightField
{

```

```

        get { return DiuresisNight; }
        set { DiuresisNight = value; }
    }
    // Суточный
    private int DiuresisDaily;

    public int DiuresisDailyField
    {
        get { return DiuresisDaily; }
        set { DiuresisDaily = value; }
    }
    // Количество мочеиспусканий
    private string DiuresisUrinationCount;

    public string DiuresisUrinationCountField
    {
        get { return DiuresisUrinationCount; }
        set { DiuresisUrinationCount = value; }
    }
    // Ноктурия
    private string DiuresisNokturi;

    public string DiuresisNokturiField
    {
        get { return DiuresisNokturi; }
        set { DiuresisNokturi = value; }
    }
    // Средний объем
    private int DiuresisAverageValue;

    public int DiuresisAverageValueField
    {
        get { return DiuresisAverageValue; }
        set { DiuresisAverageValue = value; }
    }

    // Уродинамическое обследование
    // Урофлоуметрия

    // Цистометрия наполнения
    //VolumeCistometry vc = new VolumeCistometry();
    // Первое ощущение
    private int VolumeCistometryFirstFeel;

    public int VolumeCistometryFirstFeelField
    {
        get { return VolumeCistometryFirstFeel; }
        set { VolumeCistometryFirstFeel = value; }
    }

    // Первый позыв
    private int VolumeCistometryFirstCall;

    public int VolumeCistometryFirstCallField
    {
        get { return VolumeCistometryFirstCall; }
        set { VolumeCistometryFirstCall = value; }
    }
    private int VolumeCistometryFirstCallDet;

    public int VolumeCistometryFirstCallDetField

```

```

    {
        get { return VolumeCistometryFirstCallDet; }
        set { VolumeCistometryFirstCallDet = value; }
    }

    // Второй позыв
    private int VolumeCistometrySecondCall;

    public int VolumeCistometrySecondCallField
    {
        get { return VolumeCistometrySecondCall; }
        set { VolumeCistometrySecondCall = value; }
    }
    private int VolumeCistometrySecondCallDet;

    public int VolumeCistometrySecondCallDetField
    {
        get { return VolumeCistometrySecondCallDet; }
        set { VolumeCistometrySecondCallDet = value; }
    }

    // Третий позыв
    private int VolumeCistometryThirdCall;

    public int VolumeCistometryThirdCallField
    {
        get { return VolumeCistometryThirdCall; }
        set { VolumeCistometryThirdCall = value; }
    }
    private int VolumeCistometryThirdCallDet;

    public int VolumeCistometryThirdCallDetField
    {
        get { return VolumeCistometryThirdCallDet; }
        set { VolumeCistometryThirdCallDet = value; }
    }

    private int VolumeCistometryMaximal;

    public int VolumeCistometryMaximalField
    {
        get { return VolumeCistometryMaximal; }
        set { VolumeCistometryMaximal = value; }
    }

    private int VolumeCistometryLeak;

    public int VolumeCistometryLeakField
    {
        get { return VolumeCistometryLeak; }
        set { VolumeCistometryLeak = value; }
    }

    private string VolumeCistometryPressure;

    public string VolumeCistometryPressureField
    {
        get { return VolumeCistometryPressure; }
        set { VolumeCistometryPressure = value; }
    }

    // Микционная цистометрия

```

```

//MikcionCistometry mc = new MikcionCistometry();

public int MikcionCistometryQMax;
public int MikcionCistometryQmaxDet;
public int MikcionCistometryV;
public int MikcionCistometryRemain;

// Заключение

public string Conclusion;

// Чувствительность мочевого пузыря
public string BladderSensitivity;

// Функция детрузора при наполнении
public List<string> DetusorFunction = new List<string>();

// Комплаентность
public string Complaent;

// При исследовании порога абдоминального давления подтекание мочи
public string PressureLeak;
public int PressureLeakDet;
// Максимальная цистометрическая емкость
public int MaxCistometryVolume;

// Сократительная способность детрузора
public string Detusor;
public string DetusorAddiction;

// Инфравезикальная обструкция
public string Obstruction;
public string ObstructionAddiction;
// Уродинамическое стрессовое недержание мочи
public bool StressLeak;
// Цистоскопия
public string Cystoscope;

////////////////////////////////////
// Диагноз
//

public string Diagnoz;

////////////////////////////////////
// Лечение
//

// Консервативное
public List<string> ConservativeCure = new List<string>();
public List<string> BaklofenDose = new List<string>();
public string UroGialNumber;

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

public bool OperCure1;
public string OperCure1Add;
public bool OperCure2;

```

```

        public string OperCure2Add;
        public bool OperCure3;
        public string OperCure3Add;
        public bool OperCure4;
        public string OperCure4Add;
        public bool OperCure5;
        public string OperCure5Add;
        public bool OperCure6;
        public string OperCure6Add;
        public bool OperCure7;
        public string OperCure7Add;
        public bool OperCure8;
        public string OperCure8Add;

        public List<string> AnotherCure = new List<string>();
        public string AnotherCureAdd;

        //////////////////////////////////////
        // Дополнительно
        public string FullAddiction;

    }
}

```

PatientEditForm.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace MedProg
{
    public partial class PatientEditForm : Form
    {
        Patient CurPatient;
        bool bAccepted = false;

        public PatientEditForm()
        {
            InitializeComponent();

            textBoxDetusorAddiction.Visible =
                comboBoxDetusor.Text !=
comboBoxDetusor.Items[0].ToString() &&
                comboBoxDetusor.Text != "";

            textBoxObstructionAddiction.Visible =
                comboBoxObstruction.Text !=
comboBoxObstruction.Items[0].ToString() &&
                comboBoxObstruction.Text != "";
        }

        public Patient patient
        {
            get { return CurPatient; }
        }
    }
}

```

```

        set { CurPatient = value; UpdateForm(); }
    }

    public bool Accepted
    {
        get { return bAccepted; }
    }

    private string SetCheckBox(ref List<string> list, ref CheckedListBox
listbox)
    {
        for (int i = 0; i < listbox.Items.Count; i++ )
        {
            listbox.SetItemChecked(i,
list.Contains(listbox.Items[i].ToString()));
        }

        foreach (string s in list)
        {
            if (!listbox.Items.Contains(s))
            {
                return s;
            }
        }

        return "";
    }

    private string ToMultiline(string str)
    {
        if (str == null)
            return "";
        return str.Replace("\r", "").Replace("\n",
Environment.NewLine);
    }

    private void UpdateForm()
    {
        if (CurPatient == null)
            return;

        // Вкладка Общая информация
        textBoxSername.Text = CurPatient.SernameField;
        textBoxName.Text = CurPatient.NameField;
        textBoxSecondName.Text = CurPatient.SecondNameField;

        try
        {
            dateTimePickerBirthday.Value = CurPatient.BirthdayField;
        }
        catch (ArgumentOutOfRangeException)
        {
        }

        textBoxAddress.Text = CurPatient.AddressField;
        textBoxPhoneHome.Text = CurPatient.PhoneHome;
        textBoxPhoneMobile.Text = CurPatient.PhoneMobile;
        textBoxSpeciality.Text = CurPatient.Speciality;

        // Вкладка жалобы

```

```

        textBoxComplaints.Text =
ToMultiline(CurPatient.ComplaintsField);
        textBoxPain.Text = CurPatient.PainField;

        textBoxPainLocalizations.Text = SetCheckBox(ref
CurPatient.PainLocalization, ref checkedListBoxPainLocalizations);
        SetCheckBox(ref CurPatient.PainInt, ref
checkedListBoxPainIntensity);
        SetCheckBox(ref CurPatient.PainLink, ref
checkedListBoxPainLink);
        SetCheckBox(ref CurPatient.UrineLeak, ref
checkedListBoxUrineLeak);

        checkBoxFastUrination.Checked = CurPatient.FastUrinationField;
        checkBoxUrgention.Checked = CurPatient.UrgentionField;
        checkBoxProblemUrination.Checked =
CurPatient.ProblemUrinationField;
        checkBoxNotFullBowelEmptying.Checked =
CurPatient.NotFullBowelEmptyingField;
        checkBoxNeedStrainingEffort.Checked =
CurPatient.NeedStrainingEffortField;
        checkBoxSinusoidUrineStream.Checked =
CurPatient.SinusoidUrineStreamField;
        checkBoxNokturi.Checked = CurPatient.NokturiField;

        // Вкладка Анамнез
        textBoxAnamnes.Text = ToMultiline(CurPatient.AnamnesField);
        textBoxComplaintsDuration.Text =
CurPatient.ComplaintsDurationField;
        textBoxLabor.Text = CurPatient.LaborField;
        textBoxPregnancy.Text = CurPatient.PregnancyField;
        textBoxSurgery.Text = CurPatient.SurgeryField;
        textBoxTrauma.Text = CurPatient.TraumaField;
        textBoxNervousSystemDisease.Text = SetCheckBox(ref
CurPatient.NervousSystemDisease, ref checkedListBoxNervousSystemDisease);
        textBoxInsult.Text = SetCheckBox(ref CurPatient.Insult, ref
checkedListBoxInsult);
        textBoxDrugs.Text = CurPatient.DrugsField;
        textBoxMenopause.Text = CurPatient.Menopause;
        textBoxEndometrioz.Text = CurPatient.Endometrioz;
        textBoxMiomatki.Text = CurPatient.Miomatki;
        comboBoxCistit.Text = CurPatient.Cistit;

        // Вкладка осмотр
        comboBoxExternalUrethralOpening.Text =
CurPatient.ExternalUrethralOpeningField;
        comboBoxVaginaProlapse.Text = CurPatient.VaginaProlapseField;
        comboBoxPelvicProlapse.Text = CurPatient.PelvicProlapseField;
        checkBoxCoughProbe.Checked = CurPatient.CoughProbeField;
        checkBoxValsalvaProbe.Checked = CurPatient.ValsalvaProbeField;
        checkBoxExternalUrethral.Checked =
CurPatient.ExternalUrethralField;

        // Вкладка результат обследования
        textBoxDiuresisDay.Text =
CurPatient.DiuresisDayField.ToString();
        textBoxDiuresisNight.Text =
CurPatient.DiuresisNightField.ToString();
        textBoxDiuresisDaily.Text =
CurPatient.DiuresisDailyField.ToString();

```

```

        textBoxDiuresisUrinationCount.Text =
CurPatient.DiuresisUrinationCountField;
        textBoxDiuresisNokturi.Text = CurPatient.DiuresisNokturiField;
        textBoxDiuresisAverageValue.Text =
CurPatient.DiuresisAverageValueField.ToString();

        textBoxVolumeCistometryFirstFeel.Text =
CurPatient.VolumeCistometryFirstFeelField.ToString();
        textBoxFirstCall.Text =
CurPatient.VolumeCistometryFirstCallField.ToString();
        textBoxFirstCallDet.Text =
CurPatient.VolumeCistometryFirstCallDetField.ToString();
        textBoxSecCall.Text =
CurPatient.VolumeCistometrySecondCallField.ToString();
        textBoxSecCallDet.Text =
CurPatient.VolumeCistometrySecondCallDetField.ToString();
        textBoxThirdCall.Text =
CurPatient.VolumeCistometryThirdCallField.ToString();
        textBoxThirdCallDet.Text =
CurPatient.VolumeCistometryThirdCallDetField.ToString();
        textBoxMaximal.Text =
CurPatient.VolumeCistometryMaximalField.ToString();
        textBoxLeak.Text =
CurPatient.VolumeCistometryLeakField.ToString();
        textBoxVolumeCistometryPressure.Text =
CurPatient.VolumeCistometryPressureField;

        textBoxQMax.Text =
CurPatient.MikcionCistometryQMax.ToString();
        textBoxQMaxDet.Text =
CurPatient.MikcionCistometryQmaxDet.ToString();
        textBoxMikcionCistometryV.Text =
CurPatient.MikcionCistometryV.ToString();
        textBoxMikcionCistometryRemain.Text =
CurPatient.MikcionCistometryRemain.ToString();

        // Заключение

        textBoxConclusion.Text = ToMultiline(CurPatient.Conclusion);
        comboBoxBladderSensitivity.Text =
CurPatient.BladderSensitivity;
        textBoxDetusorFunction.Text = SetCheckBox(ref
CurPatient.DetusorFunction, ref checkedListBoxDetusorFunction);
        textBoxComplaent.Text = CurPatient.Complaent;
        comboBoxPressureLeak.Text = CurPatient.PressureLeak;
        textBoxPressureLeakDet.Text =
CurPatient.PressureLeakDet.ToString();
        textBoxMaxCistometryVolume.Text =
CurPatient.MaxCistometryVolume.ToString();
        textBoxDetusorFunction.Text = SetCheckBox(ref
CurPatient.DetusorFunction, ref checkedListBoxDetusorFunction);
        textBoxComplaent.Text = CurPatient.Complaent;
        comboBoxPressureLeak.Text = CurPatient.PressureLeak;
        textBoxPressureLeakDet.Text =
CurPatient.PressureLeakDet.ToString();
        textBoxMaxCistometryVolume.Text =
CurPatient.MaxCistometryVolume.ToString();
        comboBoxDetusor.Text = CurPatient.Detusor;
        textBoxDetusorAddiction.Text = CurPatient.DetusorAddiction;
        comboBoxObstruction.Text = CurPatient.Obstruction;
        textBoxObstructionAddiction.Text =
CurPatient.ObstructionAddiction;

```

```

        checkBoxStressLeak.Checked = CurPatient.StressLeak;
        textBoxCystoscope.Text = CurPatient.Cystoscope;

        // Диагноз
        textBoxDiagnoz.Text = ToMultiline(CurPatient.Diagnoz);

        // Лечение
        SetCheckBox(ref CurPatient.ConservativeCure, ref
checkedListBoxConservativeCure);
        SetCheckBox(ref CurPatient.BaklofenDose, ref
checkedListBoxBaklofenDose);
        textBoxUroGialNumber.Text = CurPatient.UroGialNumber;

        SetCheckBox(ref CurPatient.PhisioCure, ref
checkedListBoxPhisioCure);

        SetOperCure(ref CurPatient.OperCure1, ref
CurPatient.OperCure1Add, ref checkBoxOperCure1, ref textBoxOperCure1);
        SetOperCure(ref CurPatient.OperCure2, ref
CurPatient.OperCure2Add, ref checkBoxOperCure2, ref textBoxOperCure2);
        SetOperCure(ref CurPatient.OperCure3, ref
CurPatient.OperCure3Add, ref checkBoxOperCure3, ref textBoxOperCure3);
        SetOperCure(ref CurPatient.OperCure4, ref
CurPatient.OperCure4Add, ref checkBoxOperCure4, ref textBoxOperCure4);
        SetOperCure(ref CurPatient.OperCure5, ref
CurPatient.OperCure5Add, ref checkBoxOperCure5, ref textBoxOperCure5);
        SetOperCure(ref CurPatient.OperCure6, ref
CurPatient.OperCure6Add, ref checkBoxOperCure6, ref textBoxOperCure6);
        SetOperCure(ref CurPatient.OperCure7, ref
CurPatient.OperCure7Add, ref checkBoxOperCure7, ref textBoxOperCure7);
        SetOperCure(ref CurPatient.OperCure8, ref
CurPatient.OperCure8Add, ref checkBoxOperCure8, ref textBoxOperCure8);

        SetCheckBox(ref CurPatient.AnotherCure, ref
checkedListBoxAnotherCure);
        textBoxAnotherCure.Text =
ToMultiline(CurPatient.AnotherCureAdd);

        // Дополнительно
        textBoxFullAddiction.Text =
ToMultiline(CurPatient.FullAddiction);

        textBoxDetusorAddiction.Visible =
            comboBoxDetusor.Text !=
comboBoxDetusor.Items[0].ToString() &&
            comboBoxDetusor.Text != "";

        textBoxObstructionAddiction.Visible =
            comboBoxObstruction.Text !=
comboBoxObstruction.Items[0].ToString() &&
            comboBoxObstruction.Text != "";

    }

    private void ApplyCheckBox(ref List<string> list, ref CheckedListBox
listbox)
    {
        ApplyCheckBox(ref list, ref listbox, "");
    }

```

```

        private void ApplyCheckBox(ref List<string> list, ref CheckedListBox
listbox, string addition)
        {
            list.Clear();

            if (addiction.Length > 0)
            {
                list.Add(addiction);
            }

            foreach (string str in listbox.CheckedItems)
            {
                list.Add(str);
            }
        }

        private int ToIntConverter(string str)
        {
            if (str.Length > 0)
            {
                return Convert.ToInt32(str);
            }
            else
            {
                return 0;
            }
        }

        private bool ApplyChanges()
        {
            // Вкладка Общая информация
            CurPatient.SernameField = textBoxSername.Text;
            CurPatient.NameField = textBoxName.Text;
            CurPatient.SecondNameField = textBoxSecondName.Text;
            CurPatient.BirthdayField = dateTimePickerBirthday.Value;
            CurPatient.AddressField = textBoxAddress.Text;
            CurPatient.PhoneHome = textBoxPhoneHome.Text;
            CurPatient.PhoneMobile = textBoxPhoneMobile.Text;
            CurPatient.Speciality = textBoxSpeciality.Text;

            // Вкладка жалобы
            CurPatient.ComplaintsField = textBoxComplaints.Text;
            CurPatient.PainField = textBoxPain.Text;

            ApplyCheckBox(ref CurPatient.PainLocalization, ref
checkedListBoxPainLocalizations, textBoxPainLocalizations.Text);
            ApplyCheckBox(ref CurPatient.PainInt, ref
checkedListBoxPainIntensity);
            ApplyCheckBox(ref CurPatient.PainLink, ref
checkedListBoxPainLink);
            ApplyCheckBox(ref CurPatient.UrineLeak, ref
checkedListBoxUrineLeak);

            CurPatient.FastUrinationField = checkBoxFastUrination.Checked;
            CurPatient.UrgentionField = checkBoxUrgention.Checked;
            CurPatient.ProblemUrinationField =
checkBoxProblemUrination.Checked;
            CurPatient.NotFullBowelEmptyingField =
checkBoxNotFullBowelEmptying.Checked;
            CurPatient.NeedStrainingEffortField =
checkBoxNeedStrainingEffort.Checked;

```

```

        CurPatient.SinusoidUrineStreamField =
checkBoxSinusoidUrineStream.Checked;
        CurPatient.NokturiField = checkBoxNokturi.Checked;

        // Вкладка Анамнез
        CurPatient.AnamnesField = textBoxAnamnes.Text;
        CurPatient.ComplaintsDurationField =
textBoxComplaintsDuration.Text;
        CurPatient.LaborField = textBoxLabor.Text;
        CurPatient.PregnancyField = textBoxPregnancy.Text;
        CurPatient.SurgeryField = textBoxSurgery.Text;
        CurPatient.TraumaField = textBoxTrauma.Text;
        ApplyCheckBox(ref CurPatient.NervousSystemDisease, ref
checkedListBoxNervousSystemDisease, textBoxNervousSystemDisease.Text);
        ApplyCheckBox(ref CurPatient.Insult, ref checkedListBoxInsult,
textBoxInsult.Text);
        CurPatient.DrugsField = textBoxDrugs.Text;
        CurPatient.Menopause = textBoxMenopause.Text;
        CurPatient.Endometrioz = textBoxEndometrioz.Text;
        CurPatient.MiomaMatki = textBoxMiomaMatki.Text;
        CurPatient.Cistit = comboBoxCistit.Text;

        // Вкладка осмотр
        CurPatient.ExternalUrethralOpeningField =
comboBoxExternalUrethralOpening.Text;
        CurPatient.VaginaProlapseField = comboBoxVaginaProlapse.Text;
        CurPatient.PelvicProlapseField = comboBoxPelvicProlapse.Text;
        CurPatient.CoughProbeField = checkBoxCoughProbe.Checked;
        CurPatient.ValsalvaProbeField = checkBoxValsalvaProbe.Checked;
        CurPatient.ExternalUrethralField =
checkBoxExternalUrethral.Checked;

        // Вкладка результат обследования
        CurPatient.DiuresisDayField =
ToIntConverter(textBoxDiuresisDay.Text);
        CurPatient.DiuresisNightField =
ToIntConverter(textBoxDiuresisNight.Text);
        CurPatient.DiuresisDailyField =
ToIntConverter(textBoxDiuresisDaily.Text);
        CurPatient.DiuresisUrinationCountField =
textBoxDiuresisUrinationCount.Text;
        CurPatient.DiuresisNokturiField = textBoxDiuresisNokturi.Text;
        CurPatient.DiuresisAverageValueField =
ToIntConverter(textBoxDiuresisAverageValue.Text);

        CurPatient.VolumeCistometryFirstFeelField =
ToIntConverter(textBoxVolumeCistometryFirstFeel.Text);
        CurPatient.VolumeCistometryFirstCallField =
ToIntConverter(textBoxFirstCall.Text);
        CurPatient.VolumeCistometryFirstCallDetField =
ToIntConverter(textBoxFirstCallDet.Text);
        CurPatient.VolumeCistometrySecondCallField =
ToIntConverter(textBoxSecCall.Text);
        CurPatient.VolumeCistometrySecondCallDetField =
ToIntConverter(textBoxSecCallDet.Text);
        CurPatient.VolumeCistometryThirdCallField =
ToIntConverter(textBoxThirdCall.Text);
        CurPatient.VolumeCistometryThirdCallDetField =
ToIntConverter(textBoxThirdCallDet.Text);
        CurPatient.VolumeCistometryMaximalField =
ToIntConverter(textBoxMaximal.Text);

```

```

        CurPatient.VolumeCistometryLeakField =
ToIntConverter(textBoxLeak.Text);
        CurPatient.VolumeCistometryPressureField =
textBoxVolumeCistometryPressure.Text;

        CurPatient.MikcionCistometryQMax =
ToIntConverter(textBoxQMax.Text);
        CurPatient.MikcionCistometryQmaxDet =
ToIntConverter(textBoxQMaxDet.Text);
        CurPatient.MikcionCistometryV =
ToIntConverter(textBoxMikcionCistometryV.Text);
        CurPatient.MikcionCistometryRemain =
ToIntConverter(textBoxMikcionCistometryRemain.Text);

        // Заключение

        CurPatient.Conclusion = textBoxConclusion.Text;
        CurPatient.BladderSensitivity =
comboBoxBladderSensitivity.Text;
        ApplyCheckBox(ref CurPatient.DetusorFunction, ref
checkedListBoxDetusorFunction, textBoxDetusorFunction.Text);
        CurPatient.Complaent = textBoxComplaent.Text;
        CurPatient.PressureLeak = comboBoxPressureLeak.Text;
        CurPatient.PressureLeakDet =
ToIntConverter(textBoxPressureLeakDet.Text);
        CurPatient.MaxCistometryVolume =
ToIntConverter(textBoxMaxCistometryVolume.Text);
        ApplyCheckBox(ref CurPatient.DetusorFunction, ref
checkedListBoxDetusorFunction, textBoxDetusorFunction.Text);
        CurPatient.Complaent = textBoxComplaent.Text;
        CurPatient.PressureLeak = comboBoxPressureLeak.Text;
        CurPatient.PressureLeakDet =
ToIntConverter(textBoxPressureLeakDet.Text);
        CurPatient.MaxCistometryVolume =
ToIntConverter(textBoxMaxCistometryVolume.Text);
        CurPatient.Detusor = comboBoxDetusor.Text;
        CurPatient.DetusorAddiction = textBoxDetusorAddiction.Text;
        CurPatient.Obstruction = comboBoxObstruction.Text;
        CurPatient.ObstructionAddiction =
textBoxObstructionAddiction.Text;
        CurPatient.StressLeak = checkBoxStressLeak.Checked;
        CurPatient.Cystoscope = textBoxCystoscope.Text;

        // Диагноз
        CurPatient.Diagnoz = textBoxDiagnoz.Text;

        // Лечение
        ApplyCheckBox(ref CurPatient.ConservativeCure, ref
checkedListBoxConservativeCure);
        ApplyCheckBox(ref CurPatient.BaklofenDose, ref
checkedListBoxBaklofenDose);
        CurPatient.UroGialNumber = textBoxUroGialNumber.Text;

        ApplyCheckBox(ref CurPatient.PhisioCure, ref
checkedListBoxPhisioCure);

        ApplyOperCure(ref CurPatient.OperCure1, ref
CurPatient.OperCure1Add, ref checkBoxOperCure1, ref textBoxOperCure1);
        ApplyOperCure(ref CurPatient.OperCure2, ref
CurPatient.OperCure2Add, ref checkBoxOperCure2, ref textBoxOperCure2);
        ApplyOperCure(ref CurPatient.OperCure3, ref
CurPatient.OperCure3Add, ref checkBoxOperCure3, ref textBoxOperCure3);

```

```

        ApplyOperCure(ref CurPatient.OperCure4, ref
CurPatient.OperCure4Add, ref checkBoxOperCure4, ref textBoxOperCure4);
        ApplyOperCure(ref CurPatient.OperCure5, ref
CurPatient.OperCure5Add, ref checkBoxOperCure5, ref textBoxOperCure5);
        ApplyOperCure(ref CurPatient.OperCure6, ref
CurPatient.OperCure6Add, ref checkBoxOperCure6, ref textBoxOperCure6);
        ApplyOperCure(ref CurPatient.OperCure7, ref
CurPatient.OperCure7Add, ref checkBoxOperCure7, ref textBoxOperCure7);
        ApplyOperCure(ref CurPatient.OperCure8, ref
CurPatient.OperCure8Add, ref checkBoxOperCure8, ref textBoxOperCure8);

        ApplyCheckBox(ref CurPatient.AnotherCure, ref
checkedListBoxAnotherCure);
        CurPatient.AnotherCureAdd = textBoxAnotherCure.Text;

        // Дополнительно
        CurPatient.FullAddiction = textBoxFullAddiction.Text;

        return true;
    }

    private bool IsBaklofenChecked()
    {
        return
checkedListBoxConservativeCure.CheckedItems.Contains("Баклофен.");
    }

    private void ApplyOperCure(ref bool cure, ref string cureadd, ref
CheckBox cbx, ref TextBox tb)
    {
        cure = cbx.Checked;
        if (cure)
        {
            cureadd = tb.Text;
        }
    }

    private void SetOperCure(ref bool cure, ref string cureadd, ref
CheckBox cbx, ref TextBox tb)
    {
        cbx.Checked = cure;
        if (cure)
        {
            tb.Text = cureadd;
        }
    }

    private void buttonApply_Click(object sender, EventArgs e)
    {
        bAccepted = ApplyChanges();
        Close();
    }

    private void buttonCancel_Click(object sender, EventArgs e)
    {
        bAccepted = false;
        Close();
    }

    private void buttonPrint_Click(object sender, EventArgs e)
    {
        ApplyChanges();
    }

```

```

        PrintPreviewForm ppf = new PrintPreviewForm();

        ppf.PrintPatient = CurPatient;

        ppf.ShowDialog();
    }

    private void comboBoxDetusor_TextUpdate(object sender, EventArgs e)
    {
        textBoxDetusorAddiction.Visible =
            comboBoxDetusor.Text !=
comboBoxDetusor.Items[0].ToString() &&
            comboBoxDetusor.Text != "";
    }

    private void comboBoxDetusor_SelectionChangeCommitted(object sender,
EventArgs e)
    {
        textBoxDetusorAddiction.Visible =
(string)comboBoxDetusor.SelectedItem != comboBoxDetusor.Items[0].ToString() ;
    }

    private void comboBoxObstruction_SelectionChangeCommitted(object
sender, EventArgs e)
    {
        textBoxObstructionAddiction.Visible =
(string)comboBoxObstruction.SelectedItem !=
comboBoxObstruction.Items[0].ToString();
    }

    private void comboBoxObstruction_TextUpdate(object sender, EventArgs
e)
    {
        textBoxObstructionAddiction.Visible =
            comboBoxObstruction.Text !=
comboBoxObstruction.Items[0].ToString() &&
            comboBoxObstruction.Text != "";
    }

    private void checkedListBoxConservativeCure_ItemCheck(object sender,
ItemCheckEventArgs e)
    {
        if (e.Index ==
checkedListBoxConservativeCure.Items.IndexOf("Баклофен."))
        {
            if (e.CurrentValue == CheckState.Unchecked ||
                e.NewValue == CheckState.Unchecked)
            {
                for (int i = 0; i <
checkedListBoxBaklofenDose.Items.Count; i++ )
                {
                    checkedListBoxBaklofenDose.SetItemChecked(i,
false);
                }
            }

            checkedListBoxBaklofenDose.Visible = e.NewValue ==
CheckState.Checked;
        }
    }

```

```

        if (e.Index ==
checkedListBoxConservativeCure.Items.IndexOf("Лечение препаратом «Уро-гиал», 1
инстиляция в неделю №"))
        {
            if (e.CurrentValue == CheckState.Unchecked ||
                e.NewValue == CheckState.Unchecked)
            {
                textBoxUroGialNumber.Text = "";
            }

            textBoxUroGialNumber.Visible = e.NewValue ==
CheckState.Checked;
        }
    }
}

```

MainForm.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Reflection; // For Missing.Value and BindingFlags
using System.Runtime.InteropServices; // For COMException
using Microsoft.Office.Interop.Excel;

namespace MedProg
{
    public partial class MainForm : Form
    {
        public Patients book;

        public MainForm()
        {
            try
            {
                Cursor = Cursors.WaitCursor;
                book = Patients.DeSerialize();
                if ((DateTime.Now - book.LastDrop).Hours > 24)
                {
                    ExcellExport.Output(book, "");
                    book.LastDrop = DateTime.Now;
                }
            }
            catch
            {
            }

            Cursor = Cursors.Default;

```

```

        if (book == null)
        {
            book = new Patients();
        }

        InitializeComponent();

    }

    private void MainForm_Load(object sender, EventArgs e)
    {
        foreach (Patient p in book.PatientList)
        {
            ListViewItem lvi = new ListViewItem();

            lvi.Text = p.PatientIdField.ToString();
            lvi.SubItems.Add(p.SernameField);
            lvi.SubItems.Add(p.NameField);
            lvi.SubItems.Add(p.SecondNameField);
            lvi.SubItems.Add(p.BirthdayField.ToLongDateString());
            lvi.SubItems.Add(p.AgeLiteral());

            listView1.Items.Add(lvi);
        }
    }

    private void buttonAddPatient_Click(object sender, EventArgs e)
    {
        PatientEditForm pef = new PatientEditForm();

        pef.patient = new Patient();

        pef.ShowDialog();

        if (pef.Accepted)
        {
            book.PatientList.Add(pef.patient);

            Patient p = pef.patient;
            ListViewItem lvi = new ListViewItem();

            lvi.Text = p.PatientIdField.ToString();
            lvi.SubItems.Add(p.SernameField);
            lvi.SubItems.Add(p.NameField);
            lvi.SubItems.Add(p.SecondNameField);
            lvi.SubItems.Add(p.BirthdayField.ToLongDateString());
            lvi.SubItems.Add(p.AgeLiteral());

            listView1.Items.Add(lvi);
        }
    }

    private void EditRow()
    {
        try
        {
            Guid g = new Guid(listView1.SelectedItems[0].Text);
            Patient p = book.FindPatient(g);

            if (p != null)

```

```

        {
            PatientEditForm pef = new PatientEditForm();

            pef.patient = p;
            pef.ShowDialog();

            if (pef.Accepted)
            {
                listView1.SelectedItems[0].SubItems[1].Text
= p.SernameField;
                listView1.SelectedItems[0].SubItems[2].Text
= p.NameField;
                listView1.SelectedItems[0].SubItems[3].Text
= p.SecondNameField;
                listView1.SelectedItems[0].SubItems[4].Text
= p.BirthdayField.ToLongDateString();
                listView1.SelectedItems[0].SubItems[5].Text
= p.AgeLiteral();
            }
        }

    }
    catch
    {
    }

}

private void DeleteRow()
{
    try
    {
        Guid g = new Guid(listView1.SelectedItems[0].Text);
        Patient p = book.FindPatient(g);

        if (p != null)
        {
            listView1.Items.Remove(listView1.SelectedItems[0]);
            book.PatientList.Remove(p);
        }
    }
    catch
    {
    }
}

private void listView1_MouseDoubleClick(object sender,
MouseEventArgs e)
{
    EditRow();
}

private void listView1_ColumnWidthChanging(object sender,
ColumnWidthChangingEventArgs e)
{
    e.Cancel = true;
    e.NewWidth = listView1.Columns[e.ColumnIndex].Width;
}

```

```

    }

    private void buttonEditPatient_Click(object sender, EventArgs e)
    {
        EditRow();
    }

    private void buttonDelPatient_Click(object sender, EventArgs e)
    {
        DeleteRow();
    }

    private void buttonPrint_Click(object sender, EventArgs e)
    {
        try
        {
            Guid g = new Guid(listView1.SelectedItems[0].Text);
            Patient p = book.FindPatient(g);

            if (p != null)
            {
                PrintPreviewForm ppf = new PrintPreviewForm();

                ppf.PrintPatient = p;

                ppf.ShowDialog();
            }
        }
        catch
        {
        }
    }

e) private void excelToolStripMenuItem_Click(object sender, EventArgs
    {
        SaveFileDialog sfd = new SaveFileDialog();

        sfd.OverwritePrompt = false;

        if (sfd.ShowDialog() == DialogResult.OK)
        {
            try
            {
                Cursor = Cursors.WaitCursor;
                if (ExcelExport.Output(book, sfd.FileName))
                {
                    MessageBox.Show("База экспортирована
успешно");

                    book.LastDrop = DateTime.Now;
                }
                else
                {
                    MessageBox.Show("Проблема экспорта базы в
Excel формат. База не экспортирована");
                }
            }
            catch

```

```

        {
            MessageBox.Show("Проблема экспорта базы в Excel
формат. База не экспортирована");
        }

        Cursor = Cursors.Default;
    }
}

private void создатьАрхивнуюКопиюToolStripMenuItem_Click(object
sender, EventArgs e)
{
    MessageBox.Show(
        "Архивные копии внутренней базы создаются при каждом
закрытии базы в папке Backups. Для создания Excel версии воспользуйтесь
экспортом. Автоматический экспорт в Excel осуществляется раз в сутки при запуске
программы.",
        "Создание архивных копий",
        MessageBoxButtons.OK,
        MessageBoxIcon.Information);
}

private void открытьToolStripMenuItem_Click(object sender, EventArgs
e)
{
    OpenFileDialog ofd = new OpenFileDialog();

    ofd.DefaultExt = "xml";
    ofd.Filter = "Внутренняя база пациентов|*.xml";
    ofd.InitialDirectory = Environment.CurrentDirectory +
"\\"Backups";

    ofd.Multiselect = false;
    ofd.Title = "Открытие резервной базы пациентов";

    if (ofd.ShowDialog() == DialogResult.OK)
    {
        listView1.Items.Clear();

        book.CreateBackup();
        book = Patients.DeSerialize(ofd.FileName);

        MainForm_Load(sender, e);
    }
}
}

```

PrintPreviewForm.cs

```

using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Drawing.Printing;

```

```

namespace MedProg
{
    public partial class PrintPreviewForm : Form
    {
        Patient patient;
        string stringToPrint;
        List<FontDef> strings = new List<FontDef>();
        int curstrindex;

        public PrintPreviewForm()
        {
            InitializeComponent();

            printDocument1.DefaultPageSettings.Margins.Left = 30;
            printDocument1.DefaultPageSettings.Margins.Top = 30;
            printDocument1.DefaultPageSettings.Margins.Right = 30;
            printDocument1.DefaultPageSettings.Margins.Bottom = 30;

        }

        public Patient PrintPatient
        {
            set { patient = value; RefreshInfo(); }
            get { return patient; }
        }

        private void RefreshInfo()
        {
            PatientBindingSource.DataSource = patient;
            printDocument1.DocumentName = patient.SernameField;

        }

        private void PrintPreviewForm_Load(object sender, EventArgs e)
        {

        }

        private void printDocument1_PrintPage(object sender,
System.Drawing.Printing.PrintPageEventArgs e)
        {
            int charactersOnPage = 0;
            int linesPerPage = 0;
            Rectangle curRect = e.MarginBounds;

            for (; ; )
            {
                // Sets the value of charactersOnPage to the number of
characters
                // of stringToPrint that will fit within the bounds of
the page.

                e.Graphics.MeasureString(
                    strings[curstrindex].str,
                    strings[curstrindex].font,
                    curRect.Size,
                    StringFormat.GenericTypographic,
                    out charactersOnPage, out linesPerPage);

                // Draws the string within the bounds of the page
                e.Graphics.DrawString(
                    strings[curstrindex].str,

```

```

        strings[curstrindex].font,
        Brushes.Black,
        curRect,
        StringFormat.GenericTypographic);

        // Remove the portion of the string that has been
printed.
        strings[curstrindex].str =
strings[curstrindex].str.Substring(charactersOnPage);

        // Check to see if more pages are to be printed.

        // Если недопечатали текущую строку, ждем след события
if (strings[curstrindex].str.Length > 0)
{
    e.HasMorePages = true;
    break;
}

        // Если допечатали и строк больше нет, выходим
if (curstrindex == strings.Count - 1)
{
    e.HasMorePages = false;
    break;
}

        // Здесь мы если допечатали строку, но строки еще есть

        //Смешаем квадрат печати
curRect.Y +=
(int)strings[curstrindex].font.GetHeight(e.Graphics) * linesPerPage;
curRect.Height -=
(int)strings[curstrindex].font.GetHeight(e.Graphics) * linesPerPage;

        // Переходим к след строке
curstrindex++;
    }
}

private void buttonPrintSetup_Click(object sender, EventArgs e)
{
    try
    {
        PrintDialog pd = new PrintDialog();

        pd.Document = printDocument1;
        pd.ShowDialog();
        printPreviewControl1.InvalidatePreview();

    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}

private void buttonPageSetup_Click(object sender, EventArgs e)
{
    try
    {
        pageSetupDialog1.Document = printDocument1;
        pageSetupDialog1.ShowDialog();
    }
}

```

```

        printPreviewControl1.InvalidatePreview();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}

private void buttonPrint_Click(object sender, EventArgs e)
{
    printDocument1.Print();
}

private void PrintPreviewForm_ResizeEnd(object sender, EventArgs e)
{
    printPreviewControl1.AutoZoom = false;
    printPreviewControl1.AutoZoom = true;
}

private void PrintPreviewForm_SizeChanged(object sender, EventArgs
e)
{
    printPreviewControl1.AutoZoom = false;
    printPreviewControl1.AutoZoom = true;
}

void PrepareStrings()
{
    strings.Clear();

    strings.Add(new FontHeader1("Пациент"));

    // ФИО, возраст

    TimeSpan age = DateTime.Now - PrintPatient.BirthdayField;

    stringToPrint =
        PrintPatient.SurnameField + " " +
        PrintPatient.NameField + " " +
        PrintPatient.SecondNameField + ", " +
        (age.Days / 365).ToString() + " " +
Patient.AgeLiteralSuffix(age.Days / 365);

    strings.Add(new FontDef(stringToPrint));

    if (PrintPatient.Conclusion.Length > 0)
    {
        strings.Add(new FontHeader2("\n\rЗаключение\n\r"));
        strings.Add(new FontDef(PrintPatient.Conclusion));
    }

    if (PrintPatient.Diagnoz.Length > 0)
    {
        strings.Add(new FontHeader2("\n\rДиагноз\n\r"));
        strings.Add(new FontDef(PrintPatient.Diagnoz));
    }

    strings.Add(new FontHeader2("\n\rЛечение\n\r"));

    if (PrintPatient.ConservativeCure.Count > 0)

```

```

        {
            strings.Add(new FontHeader3("Консервативное\n\r"));
            stringToPrint = "";
            foreach (string s in PrintPatient.ConservativeCure)
            {
                stringToPrint += s;

                if (s == "Лечение препаратом «Уро-гиал», 1
инстиляция в неделю №")
                {
                    stringToPrint += " " +
PrintPatient.UroGialNumber;
                }

                stringToPrint += "\n\r";

                if (s == "Баклофен.")
                {
                    stringToPrint += "      Рекомендованная схема
приема:\n\r";
                    foreach (string sb in
PrintPatient.BaklofenDose)
                    {
                        stringToPrint += "      " + sb + "\r\n";
                    }
                }

                strings.Add(new FontDef(stringToPrint));
            }

            if (PrintPatient.PhisioCure.Count > 0)
            {
                strings.Add(new
FontHeader3("Физиотерапевтическое\n\r"));
                stringToPrint = "";
                foreach (string s in PrintPatient.PhisioCure)
                {
                    stringToPrint += s + "\r\n";
                }

                strings.Add(new FontDef(stringToPrint));
            }

            if (PrintPatient.OperCure1 ||
PrintPatient.OperCure2 ||
PrintPatient.OperCure3 ||
PrintPatient.OperCure4 ||
PrintPatient.OperCure5 ||
PrintPatient.OperCure6 ||
PrintPatient.OperCure7 ||
PrintPatient.OperCure8)
            {
                strings.Add(new FontHeader3("Оперативное\n\r"));

                stringToPrint = "";

                if (PrintPatient.OperCure1)
                {
                    stringToPrint += "Инъекции ботулотоксина в
сфинктер. ";

                    if (PrintPatient.OperCure1Add.Length > 0)

```

```

        {
            stringToPrint += PrintPatient.OperCure1Add;
        }

        stringToPrint += "\n\r";
    }

    if (PrintPatient.OperCure2)
    {
        stringToPrint += "Инъекции ботулотоксина в
детрузор. ";

        if (PrintPatient.OperCure2Add.Length > 0)
        {
            stringToPrint += PrintPatient.OperCure2Add;
        }

        stringToPrint += "\n\r";
    }

    if (PrintPatient.OperCure3)
    {
        stringToPrint += "Вапоризация лейкоплакии. ";
        if (PrintPatient.OperCure3Add.Length > 0)
        {
            stringToPrint += PrintPatient.OperCure3Add;
        }

        stringToPrint += "\n\r";
    }

    if (PrintPatient.OperCure4)
    {
        stringToPrint += "Введение формообразующего геля.
";

        if (PrintPatient.OperCure4Add.Length > 0)
        {
            stringToPrint += PrintPatient.OperCure4Add;
        }

        stringToPrint += "\n\r";
    }

    if (PrintPatient.OperCure5)
    {
        stringToPrint += "Операция Берча. ";
        if (PrintPatient.OperCure5Add.Length > 0)
        {
            stringToPrint += PrintPatient.OperCure5Add;
        }

        stringToPrint += "\n\r";
    }

    if (PrintPatient.OperCure6)
    {
        stringToPrint += "Установка слинга. ";
        if (PrintPatient.OperCure6Add.Length > 0)
        {
            stringToPrint += PrintPatient.OperCure6Add;
        }

        stringToPrint += "\n\r";
    }

```

```

    }

    if (PrintPatient.OperCure7)
    {
        stringToPrint += "Уретротомиа. ";
        if (PrintPatient.OperCure7Add.Length > 0)
        {
            stringToPrint += PrintPatient.OperCure7Add;
        }

        stringToPrint += "\n\r";
    }

    if (PrintPatient.OperCure8)
    {
        stringToPrint += "ТУП простаты. ";
        if (PrintPatient.OperCure8Add.Length > 0)
        {
            stringToPrint += PrintPatient.OperCure8Add;
        }

        stringToPrint += "\n\r";
    }

    strings.Add(new FontDef(stringToPrint));
}

if (PrintPatient.AnotherCure.Count > 0)
{
    stringToPrint = "";
    foreach (string s in PrintPatient.PhisioCure)
    {
        stringToPrint += s + "\r\n";
    }

    strings.Add(new FontHeader3(stringToPrint));
}

if (PrintPatient.AnotherCureAdd.Length > 0)
{
    strings.Add(new FontHeader3("Дополнительно"));
    strings.Add(new FontDef(PrintPatient.AnotherCureAdd));
}
}

private void printDocument1_BeginPrint(object sender, PrintEventArgs
e)
{
    curstrindex = 0;
    PrepareStrings();
}

private void PrintPreviewForm_KeyDown(object sender, KeyEventArgs e)
{
    if (e.KeyData == Keys.PageDown)
    {
        printPreviewControll1.StartPage++;
        printPreviewControll1.InvalidatePreview();
    }

    if (e.KeyData == Keys.PageUp)
    {

```

```

        if (printPreviewControll1.StartPage > 0)
            printPreviewControll1.StartPage--;
        printPreviewControll1.InvalidatePreview();
    }

}

private void PrintPreviewForm_MouseDown(object sender,
MouseEventArgs e)
{
}

private void printPreviewControll1_MouseDown(object sender,
MouseEventArgs e)
{
    if (e.Delta > 0)
    {
        printPreviewControll1.StartPage++;
        printPreviewControll1.InvalidatePreview();
    }
    else if (e.Delta < 0)
    {
        if (printPreviewControll1.StartPage > 0)
            printPreviewControll1.StartPage--;
        printPreviewControll1.InvalidatePreview();
    }
}

}

}

tem.Windows.Forms.CheckBox checkBoxOperCure2;
private System.Windows.Forms.TextBox textBoxOperCure8;
private System.Windows.Forms.TextBox textBoxOperCure7;
private System.Windows.Forms.TextBox textBoxOperCure6;
private System.Windows.Forms.TextBox textBoxOperCure5;
private System.Windows.Forms.TextBox textBoxOperCure4;
private System.Windows.Forms.TextBox textBoxOperCure3;
private System.Windows.Forms.TextBox textBoxOperCure2;
private System.Windows.Forms.TextBox textBoxOperCure1;
private System.Windows.Forms.CheckBox checkBoxOperCure1;
private System.Windows.Forms.CheckedListBox
checkedListBoxAnotherCure;
private System.Windows.Forms.TextBox textBoxAnotherCure;
}
}

```

ПРИЛОЖЕНИЕ В ПРЕЗЕНТАЦИЯ

ТЕМА РАБОТЫ: РАСПРЕДЕЛЕННОЕ СЕТЕВОЕ ПРИЛОЖЕНИЕ ДЛЯ ПРЕДПРИЯТИЯ И ЕГО ФИЛИАЛОВ

Автор работы: студент группы 18ВВм1 Жуков В.И.
Научный руководитель: д.т.н., профессор Зинкин С.А.

Актуальность проблемы

- Интернет и автоматизация работы проникает во многие сферы жизни
- Медицина не является исключением, благодаря наличию документооборота
- Медицинские информационные системы активно развиваются как в России, так и за рубежом



Цели и задачи

Цель работы – создание медицинской информационной системы для индивидуального использования специалистом с наличием сетевых функций

Задачи:

- Изучить предметную область
- Проанализировать предъявляемые требования
- Выбрать языки, технологии и средства для разработки приложения
- Разработать приложение с графическим интерфейсом
- Протестировать приложение и выполнить его отладку

Анализ требований к приложению

- Создание документов осмотра пациента с возможностью использования шаблонов
- Редактирование шаблонов внутри приложения
- Синхронизация данных с внешней базой данных по сети



Выбор языков, технологий и средств

Языки:

- C#
- XML
- XAML

Технологии и средства:

- .NET Framework
- Windows Presentation Foundation
- Open Database Connectivity
- Microsoft SQL Server
- Microsoft Visual Studio



Разработка структур шаблонов

```
<druglist>
  <category>
    <categoryName>Антиагреганты</categoryName>
    <drugItem>
      <name>Кардио-Аспирин</name>
      <generic>Аспинат, Аспирин-Кардио, КардиАСК, Тромбо АСС, Аспирин</generic>
      <dosage>100 мг внутрь 1 раз в день вечером перед едой ежедневно постоянно</dosage>
      <expectedEffect>препятствует тромбообразованию, снижая риск инфаркта и инсульта</expectedEffect>
      <receptionFeatures>При АД выше 160/90 мм рт.ст. не принимать (высокий риск кровотечений)!</receptionFeatures>
    </drugItem>
  </category>
</druglist>
```

- Разработка шаблонов велась на основе расширяемого языка разметки XML, предоставляющий широкие возможности по созданию различных структурированных данных

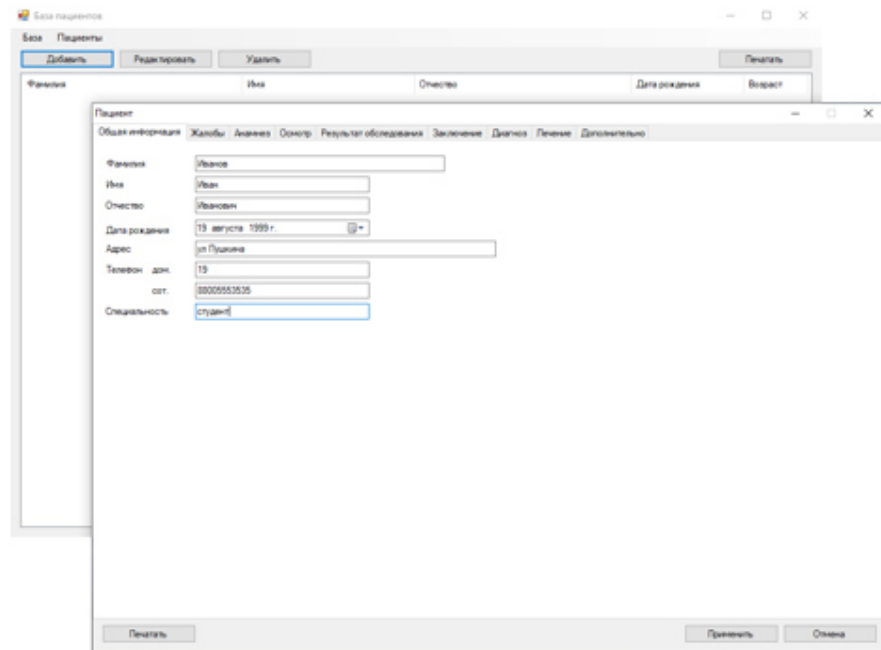


Разработка модулей и интерфейса приложения

- Для разработки модулей была использована Visual Studio 2019
- Для разработки интерфейса приложения был использован редактор графических интерфейсов Visual Studio Blend 2019
- Данные инструменты являются самыми мощными и функциональными для разработки WPF-приложений



Разработка модулей и интерфейса приложения



Тестирование приложения

- Отладка отдельных модулей
- Отладка отдельных групп интерфейсов
- Комплексное тестирование взаимодействия интерфейса и модулей
- Исправление найденных в процессе тестирования ошибок
- Финальное общее тестирование



Выводы

В результате проделанной работы:

- Составлены и реализованы алгоритмы работы МИС для индивидуального использования специалистом
- Разработано и протестировано приложение с графическим интерфейсом
- Оформлена соответствующая документация

Спасибо за внимание!