МИНИСТЕРСТВО ПО РАЗВИТИЮ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И КОММУНИКАЦИЙ РЕСПУБЛИКИ УЗБЕКИСТАН

ТАШКЕНТСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

К защите.

Зав. Кафедрой

_____

_____2016 г.

ВЫПУСКНАЯ РАБОТА БАКАЛАВРА

на тему: Создание ИС «электронная клиника» (административная часть)

Выпускник:_____ Лутфуллаев О.Ш

Руководитель:_____

БЖД _____

Рецензент_____

ТАШКЕНТ-2016

ТАШКЕНТСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Факультет КИ кафедра Информационные технологии .

Направление (специальность) Информатика и информационные технологии

УТВЕРЖДАЮ

Зав кафедрой _____

«_____» _____ 2016 г.

ЗАДАНИЕ

на выпускную работу

Лутфуллаев Отабек Шукурулла ўғли

(фамилия, имя, отчество)

1. Тема работы : Создание ИС «электронная клиника» (административная часть)

2. Утверждена приказом по университету от «_____» _____

3. Срок сдачи законченной работы

4. Исходные данные к работе Литературные данные, входящие данные

5. Содержание расчётно – пояснительной записи (перечень подлежащих разработке вопросов) Аннотация, Введение, 1 глава. Современные информационные технологии в медицине Анализ объекта и постановка задачи Глава 2 Логистический подход при решении задач, Глава 3 – Программное обеспечение, Глава 4 – Обеспечение безопасности жизнедеятельности, Заключение, Использованные литературы, Приложение

6. Перечень графического материала; Презентационные материалы

7. Дата выдачи задания _____

Руководитель _____

(подпись)

Задание принял _____

(подпись)

8. Консультанты по отдельным разделам выпускной работы

Раздел	Ф.И.О руководителя	Подпись дата	
		Задание выдал	Задание получил
Основная честь БЖД			

9. График выполнения работы

№	Наименование раздела работы	Срок выполнения	Отметка руководителя о выполнении
1	Введение		
2	Современные информационные технологии в медицине		
3	Проектирование информационной системы		
4	Программная реализация информационной системы		
5	безопасность жизнедеятельности и экология		
6	Заключение		

Выпускник _____ «_____» _____ 2016г.

(подпись)

Руководитель _____ «_____» _____ 2016г.

(подпись)

Аннотация

Цель работы является разработка информационной системы “Электронная клиника (административная часть)” для повышения эффективности деятельности больницы с использованием современных информационних технологий в отделах больницы, на основе требований системы “Электронного правительства”.

Тақдим этилаётган иш, шифохона фаолиятини “электрон ҳукумат” тизими талабларидан келиб чиқиб, шифохона бўлимлари фаолиятига замонавий ахборот-коммуникацион технологияларини жорий қилган ҳолда, шифохона фаолияти самарадорлигини ошириш мақсадида “Электрон шифохона” (адинистратор қисми) ахборот тизимини яратиш мақсад қилиб олинган.

The goal is to develop an information system "Electronic clinic (administrative part)" to improve the efficiency of the hospital with the use of modern information technology in hospital departments, based on the requirements of the system "Electronic Government".

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Компьютеры и информационные технологии уже давно стали неотъемлемой частью самых разных сфер жизни, и медицина не стала исключением. Врачи консультируют пациентов on-line, диагностическая аппаратура оснащена мощными процессорами, конференции и консилиумы проводятся через интернет. И сегодня медицинские информационные технологии приобретают все большую актуальность, а программное обеспечение для медицины становится все более востребованными. Внедрение медицинских информационных систем способно значительно усовершенствовать рабочие процессы на самых разных уровнях: начиная от отдельных рабочих мест врачей в отдельном ЛПУ до комплексной автоматизации поликлиник и стационаров на региональном уровне.

Информационная система Электронная клиника - программный продукт, предназначенный для выполнения различных проектов автоматизации лечебно-профилактических учреждений. ИС отлично подходит для автоматизации поликлиник, многопрофильных стационаров с различными клиническими и диагностическими отделениями, санаториев.

Этот проект предназначен на реализация постановление Президента Республики Узбекистан № ПҚ-1730 «О мерах по дальнейшему внедрению и развитию современных информационно-коммуникационных технологий” и постановление Президента Республики Узбекистан № ПҚ-1989 «О мерах по дальнейшему развитию национальных информационно-коммуникационных систем» в 27-июня 2013 года.

Цель работы является разработка информационной системы “Электронная клиника” для повышения эффективности деятельности болницы с использованием современных информационних технологий в отделах болницы, на основе требований системы “Электронного правительства”.

В выполнение проекта включаются следующие этапы:

Будут определены задачи системы и будут сгруппированы задачи. Будут разработаны базы данных системы. Будут разработаны базы данных системы и программный комплекс для обеспечения управления базы данных. Будут сформулированы части системного администратора. Будет разработан программный комплекс для поиска документов из системы БД. Будет создан контрольный модуль испольняющий задачи. Будут разработаны классы программного комплекса, обеспечивающие поиск информации по модулям и обеспечивающий ввод, вывод и редактирование данных. Будет разработана политика безопастности системы. Будет создан профиль администратора и модули данных. Будет разработан модуль для обработки документов. Будут разработаны формы интерфейса и общий дизайн системы. Будет создана инструкция для пользователей.

Электронная клиника– это сеть универсальных общедоступных медицинских учреждений по оказанию комплексной амбулаторно-поликлинической помощи, использующих в своей работе современные медицинские технологии и высокие профессиональные стандарты диагностики и лечения всех наиболее распространенных заболеваний.

В последнее время уровень конкуренции для компании возрос, так как на рынке увеличился уровень спроса, что связано с открытием новых частных поликлиник, повышением уровня качества лечения и обслуживания в государственных поликлинических учреждениях.

Пояснительная записка выпускной работы состоит из следующих частей: введение, обзорная часть, основная часть, безопасность жизнедеятельности и экология, заключение, списка литературы и приложение.

Во введение описывается значение; информационных технологий и автоматизированных информационных систем, баз данных и компьютеров в жизни современного общества.

В обзорной части рассматриваются методы организации и обработки данных в ИС, требования к организации базы данных, постановка задачи.

В основной части приведены обоснование и выбор языка программирования для решения задач, проектирование базы данных решаемой задачи, описание программы, инструкция по эксплуатации

В заключении выпускной работы приводится итоги проделанной работы и эффективность использования разработанных программных средств.

1. СОВРЕМЕННЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ В МЕДИЦИНЕ

1.1.Описание предметной области.

Характерной чертой нашего времени являются информационные технологии. В понятие «информационные технологии» входят компьютерные технологии (оборудование и программное обеспечение) и технологии связи для передачи информации. Необходимость в информатизации обусловливается многими факторами. Огромными потоками информации, которые циркулируют в различных сферах деятельности, временем, затрачиваемым на процесс обработки этой информации и сложностями, связанными с хранением информации.

Внедрение информационной системы позволяет решить эти задачи. В результате чего сокращается время на обработку информации, осуществляется хранение больших объемов информации, ускоряется поиск необходимой информации, и выдача ее в удобном для пользователя виде. В основе любой информационной системы лежит база данных. База данных представляет собой совокупность специальным образом организованных данных, хранимых в памяти вычислительной системы и отображающих состояние объектов и их взаимосвязей в рассматриваемой предметной области.

Процесс создания программной системы предполагает несколько этапов:

1. Исследование предметной области, той сферы, куда будет внедряться информационная система, сбор сведений об объектах, фактах, событиях и процессах, которые участвуют в рассматриваемом виде деятельности.

2. Проектирование базы данных. Осуществляется выбор модели данных: иерархическая, сетевая или реляционная. Наиболее удобной моделью является реляционная модель. В реляционной базе данных информация организована в виде отношений. Проектирование базы данных предполагает

определение отношений и связей между ними. Определяются типы данных, ограничения на значения, задаются значения по умолчанию. Осуществляется выбор полей.

3. Разработка приложения, для работы с базой данных. Приложение осуществляет следующие задачи: добавление информации в базу данных, удаление информации из базы данных, формирование отчетов, выдача информации в необходимом для пользователя виде, по заданным запросам.

Все поликлиники сети похожи и функционируют по одинаковому принципу, поэтому проектировать ИС можно для одной поликлиники. Современная поликлиника является крупным многопрофильным, специализированным лечебно-профилактическим учреждением, предназначенным оказывать медицинскую помощь и осуществлять комплекс профилактических мероприятий по оздоровлению населения и предупреждению заболеваний.

В ее функции входят:

- оказание первой медицинской помощи при острых и внезапных заболеваниях, травмах;
- лечение больных при обращении в поликлинику и на дому;
- организация и проведение диспансеризации;
- экспертиза временной нетрудоспособности;
- освобождение больных от работы;
- направление на врачебно-трудовую экспертную комиссию лиц с признаками стойкой утраты трудоспособности;
- направление больных на санаторно-курортное лечение;
- своевременная госпитализация нуждающихся в стационарном лечении.

Поликлиника проводит большую профилактическую работу, противоэпидемические мероприятия, санитарно-просветительную работу среди обратившихся пациентов, изучает здоровье населения, выявляет

раннюю заболеваемость, организует статистический учет и анализ показателей состояния здоровья населения.

Организационная структура предприятия

Сеть поликлиник представляет собой 3 комплексных амбулаторно-поликлинических учреждения для детей и взрослых, травмпункт и лабораторно-диагностический центр. Общая структура предприятия представлена ниже (Рисунок 1.1), структура поликлиник идентична (см. Рисунок 1.2).На каждого пациента заводиться общая электронная медицинская карта:

- объединяет всю медицинскую информацию о пациенте, обеспечивая доступ к ней в любой поликлинике сети;
- позволяет любому специалисту сети поликлиник определить диагноз, подобрать лечение;
- исключает потерю данных;
- обеспечивает конфиденциальность обследования (врачебную тайну).

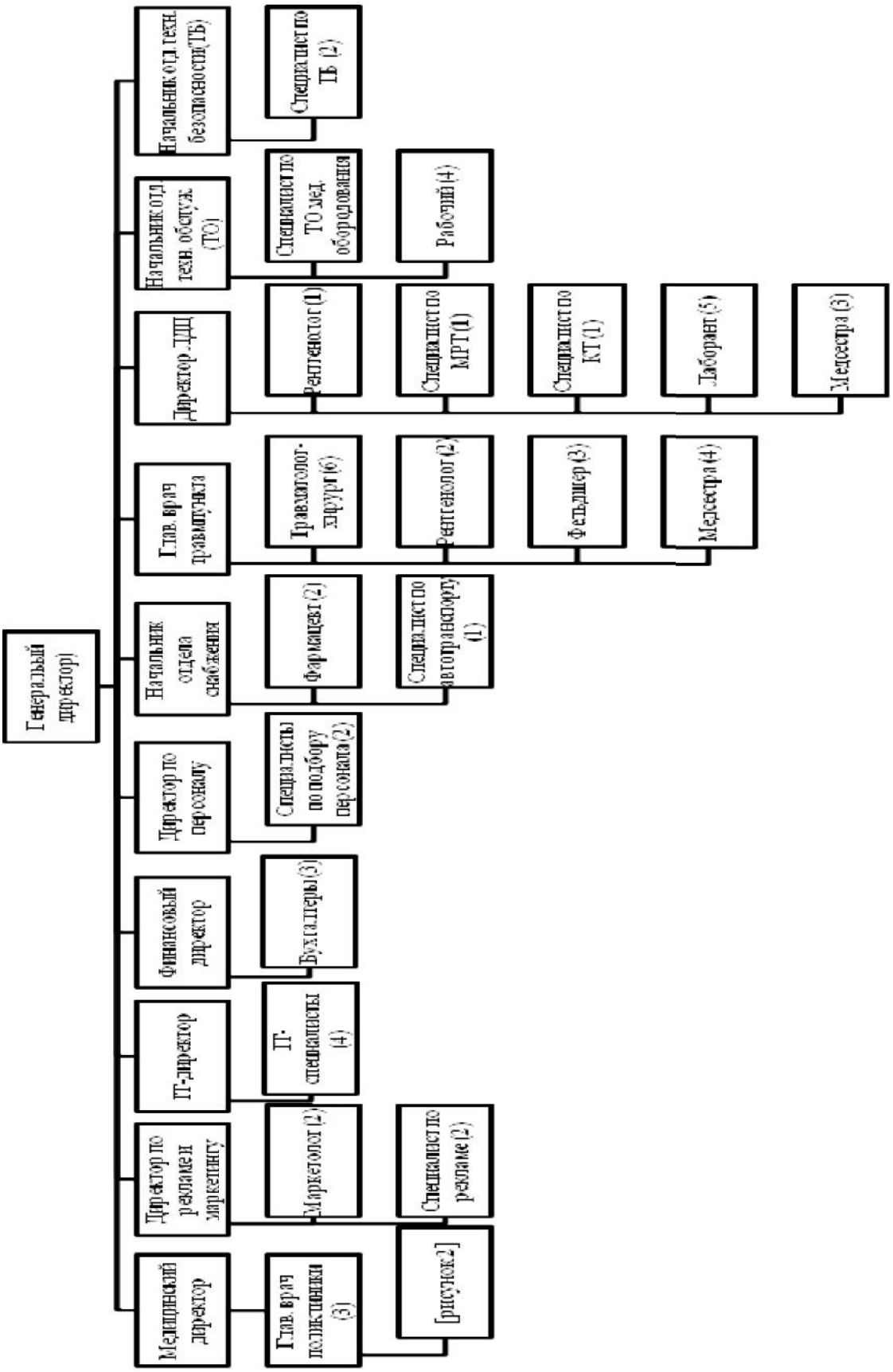

Рисунок 1.1 – Структура сети поликлиник.

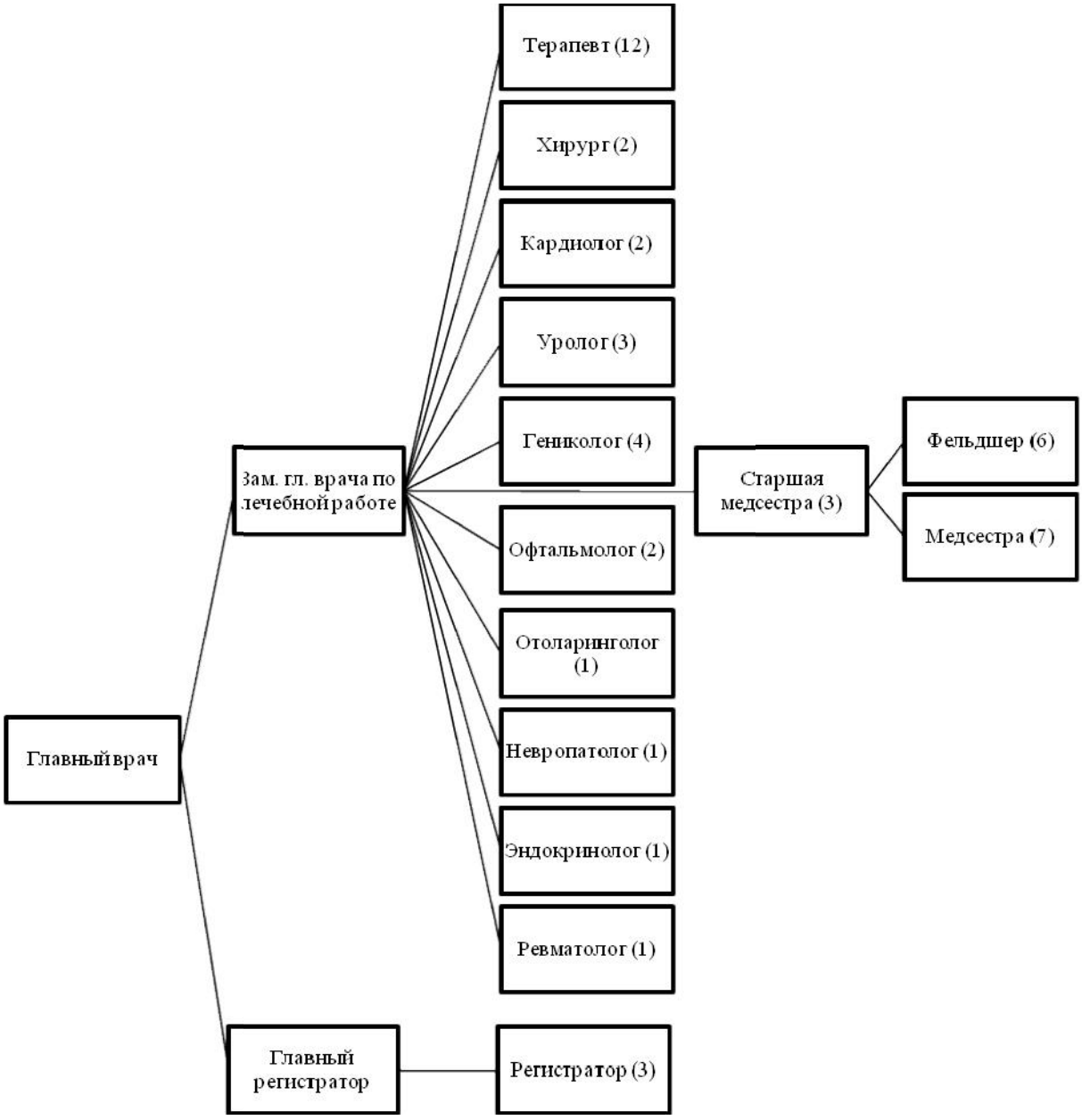

Рисунок 1.2 – Структура поликлиники.

Установленные формы нормативных документов.

Медицинская документация – система документов установленной формы, предназначенных для регистрации данных лечебных, диагностических, профилактических, санитарно-гигиенических и других мероприятий, а также для их обобщения и анализа.

Во всех однотипных медицинских учреждениях ведется унифицированная первичная медицинская документация, установленная перечнем, в котором указаны вид документа (бланк, журнал и т.д.), формат и

сроки его хранения. Образцы учетных форм и правила их заполнения содержатся в альбоме, утвержденном Министерством здравоохранения и социального развития РУз.

Унификация документов значительно облегчает разработку материалов, создает условия для механизированной обработки данных с применением электронно-вычислительной техники.

Медицинский учет отражает объем и характер работы учреждений здравоохранения и необходим для планирования мероприятий по улучшению состояния здоровья и оказания медпомощи населению, оценки качества и эффективности деятельности медицинских учреждений, обеспечения медико-статистической информацией органов управления здравоохранением различных уровней.

По своему назначению учетные документы делятся на несколько групп. Документы первой группы предназначены для записей результатов наблюдения за состоянием больного в период его лечения и лечебно-диагностических назначений. К этим документам в стационарных учреждениях относятся "Медицинская карта стационарного больного", а также ее важнейшие модификации: "Медицинская карта прерывания беременности", "История родов"; история развития новорожденного.

Вторая группа учетных документов предназначена для обеспечения преемственности и взаимосвязи между отдельными этапами (звеньями) оказания медпомощи. Эти документы содержат основные сведения о заболевших и служат оперативным сигналом для проведения необходимых санитарно-профилактических и лечебных мероприятий. К ним относятся "Выписка из медицинской карты амбулаторного, стационарного больного"; "Экстренное извещение об инфекционном заболевании, пищевом, остром профессиональном отравлении, необычной реакции на прививку"; "Извещение о больном с впервые в жизни установленным диагнозом активного туберкулеза, венерической болезни, трихофитии, микроспории, фавуса, чесотки, трахомы, психического заболевания"; "Извещение о больном с

впервые в жизни установленным диагнозом рака или другого злокачественного новообразования" и др.

Третья группа документов отражает в основном объем выполняемой медперсоналом работы. К ним, в частности, относится "Дневник работы врача поликлиники (амбулатории), диспансера, консультации", в котором ежедневно учитываются отработанное время, количество принятых больных и лиц, обследованных с профилактической целью; "Тетрадь учета работы на дому медицинской сестры (акушерки)"; "Журнал учета процедур" и др.

Основные требования, предъявляемые к заполнению медицинской документации: достоверность, медицинская грамотность, полнота и своевременность записей. Медицинская документация по своему характеру относится к документам сугубо служебного назначения и должна быть доступна только лицам, профессионально с ней связанным.

Справочные документы (таблицы, схемы, графики и др.) содержат нормативную информацию, необходимую для управления медицинской службой.

Описание процессов по подразделениям

Успех любого медицинского учреждения во многом зависит от уровня управления.

Особое значение имеет четкое определение функциональных обязанностей всех должностных лиц, работающих в поликлинике, от санитарки до главного врача. Практическая деятельность главного врача и всех должностных лиц регламентируется приказами, распоряжениями и инструкциями Министерства здравоохранения РУз. Функциональные обязанности врачей, среднего и младшего медицинского персонала могут корректироваться на местах с учетом специфики работы медицинского учреждения.

Главный врач. Поликлиникой на правах единоначалия руководит главный врач. На должность главного врача назначаются наиболее квалифицированные врачи, имеющие организаторские способности и навыки.

Главный врач организует и контролирует:

- лечебно-диагностическую и профилактическую деятельность поликлиники;

- диспансеризацию населения;

- соблюдение врачебным персоналом установленного порядка выдачи больничных листов;

- повышение квалификации медицинского персонала;

- вовлечение врачей в научно-исследовательскую работу;

- правильность ведения документации;

- учет и хранение сильнодействующих средств и ядовитых веществ, рецептурных бланков на них в соответствии с действующими инструкциями;

- обеспечение поликлиники медицинским оборудованием, инструментами, хозяйственным и мягким инвентарем;

- регулярный анализ деятельности всех подразделений поликлиники, состояния инфекционной и общей заболеваемости, заболеваемости с временной утратой трудоспособности.

Главный врач утверждает планы повышения квалификации врачебного и среднего медицинского персонала, врачебных конференций, семинарских занятий устанавливает распорядок работы поликлиники, утверждает графики работы персонала. В обязанности главного врача входят поощрение хорошо работающих, инициативных сотрудников и привлечение к дисциплинарной ответственности работников, нарушающих трудовую дисциплину, не выполняющих свои обязанности.

Заместитель главного врача по лечебной работе. Вторым лицом в поликлинике является заместитель главного врача по лечебной работе, который назначается главным врачом из наиболее квалифицированных, обладающих организаторскими способностями врачей. В отсутствие главного врача он исполняет его обязанности.

Заместитель главного врача по медицинской части отвечает за всю медицинскую деятельность поликлиники. Он организует и контролирует

правильность и своевременность обследования и лечения больных в поликлинике и на дому следит за постоянным внедрением в практику работы врачей современных, наиболее эффективных методов профилактики, диагностики и лечения больных, новых организационных форм и методов работы передовых медицинских учреждений. Заместитель главного врача обеспечивает преемственность в обследовании и лечении больных между поликлиникой и стационаром, организацию госпитализации больных, нуждающихся в стационарном лечении. Заместитель главного врача обеспечивает выполнение плана повышения квалификации медицинского персонала направляет врачей на стажировку в стационары, на курсы усовершенствования и специализации организует различные семинары, научно-практические конференции и реферативные обзоры, лечебно-контрольные комиссии с обсуждением состояния лечебно-диагностической работы, расхождений диагнозов и врачебных ошибок, работу кабинета медицинской статистики.

Регистратура. Первое знакомство посетителей с поликлиникой начинается в регистратуре. Она является основным ее структурным подразделением по организации приема больных в поликлинике и на дому. При обращении пациента, регистратор направляет его к нужному врачу или на консультацию к терапевту. В задачи регистратуры входит:

- организация предварительной и неотложной записи больных на прием к врачу как при непосредственном обращении в поликлинику, так и по телефону;
- обеспечение четкого регулирования интенсивности потока населения с целью создания равномерной нагрузки врачей и распределение его по видам оказываемой помощи;
- проведение своевременного подбора и доставки медицинской документации в кабинеты врачей, правильное ведение и хранение картотеки поликлиники.

Рациональная организация приема призвана сократить время ожидания больных на прием к врачам. Среднее время, затраченное на посещение поликлиники, в зависимости от цели посещения и ряда других объективных причин, связанных с особенностями обслуживания тех или иных больных, варьируется от 31,9 мин до 125,2 мин.

Управление сложным потоком больных в поликлинике обеспечивается внедрением прогрессивных форм организации труда врачебного и среднего медицинского персонала, а также путем совершенствования существующих форм работы регистратуры с учетом установленных норм нагрузок. Работа регистратуры должна строиться по централизованной системе и исходить из участковое-территориального принципа обслуживания населения и бригадного метода работы врачей поликлиники.

Работой регистратуры руководит заведующий, назначаемый на эту должность приказом главного врача поликлиники. Медицинскому регистратору отводится большая роль в деле правильной организации приема больных. Он первым встречает больного, беседует с ним, в необходимых случаях помогает больному разобраться в организации приема.Медицинский регистратор должен разбираться в вопросах сортировки больных по медицинским показаниям, что дает возможность правильно решать вопросы регулирования потока больных.

Врач-терапевт. Основными задачами терапевта являются:

- оказание квалифицированной терапевтической помощи пациентам на приеме в поликлинике и на дому;
- организация и непосредственное проведение профилактических мероприятий среди пациентов поликлиники путем пропаганды санитарно-гигиенических знаний, проведения профилактических прививок, организации профилактических осмотров пациентов;
- снижение заболеваемости и смертности пациентов.

В соответствии врач-терапевт обязан обеспечить:

- своевременную терапевтическую помощь в поликлинике (амбулатории) и на дому;

- экстренную медицинскую помощь больным в случае их непосредственного обращения при возникновении острых состояний, травм, отравлений;

- своевременную госпитализацию терапевтических больных с обязательным предварительным обследованием при плановой госпитализации;

- экспертизу временной нетрудоспособности больных;

- консультацию больных в необходимых случаях в поликлинике;

- использование в своей работе современных методов профилактики, диагностики и лечения больных, в том числе комплексной терапии и восстановительного лечения (медикаментозные средства, диетотерапия, лечебная физкультура, массаж, физиотерапия и др.);

- выдачу заключений пациентам, проходящим медицинские осмотры и отъезжающим за рубеж;

- систематическое повышение своей квалификации и уровня медицинских знаний участковой медицинской сестры.

Старшая медицинская сестра. Прямым помощником заместителя главного врача по медицинской части является старшая медицинская сестра, которая организует и контролирует работу среднего медицинского персонала поликлиники, выполнение им своих функциональных обязанностей.

Старшая медицинская сестра поликлиники:

- осуществляет рациональную расстановку и использование среднего и младшего медицинского персонала, а также составление графиков работы и отпусков;
- обеспечивает четкую работу регистратуры, справочно-информационной службы, процедурного и других лечебных кабинетов, работающих без врачей;

- осуществляет контроль над соблюдением дисциплины и направляет работу младшего и среднего медицинского персонала на создание надлежащей культуры в обслуживании больных и выполнением врачебных назначений
- контролирует учет, распределение, расходование и использование медицинского инструментария, медикаментов, бактериальных препаратов, перевязочного материала, а также хранение и учет сильнодействующих ядовитых медикаментов, ведение медицинской документации;
- всемерно содействует внедрению мероприятий по научной организации труда;
- составляет планы повышения квалификации средних и младших медицинских работников, организует и контролирует их выполнение.

Медицинская сестра. Важным помощником участкового врача-терапевта является участковая медицинская сестра. Целесообразно, чтобы с терапевтом работала постоянно одна и та же медицинская сестра. Прежде всего, она помогает врачу при амбулаторном приеме, ей отводится большая роль в организации наблюдения за больными на дому и выполнения назначений врача.

Медицинская сестра должна приходить на работу на 20–30 мин раньше врача. За это время ей необходимо подготовить рабочее место, проверить поступление медицинских карт из регистратуры, наличие всей необходимой документации, выявить нуждающихся в первоочередном приеме, измерить у больных температуру, артериальное давление (АД) (если это не было сделано в кабинете доврачебного приема и т. д.).Это все необходимо провести для того, чтобы врач сразу же начал прием.

Работая под непосредственным руководством и контролем врача и старшей сестры поликлиники, участковая медицинская сестра осуществляет следующую работу:

- оказывает лечебную помощь населению, выполняет назначения врача как в поликлинике, так и на дому;
- по указанию врача измеряет больным артериальное давление, проводит термометрию и другие медицинские манипуляции;
- заполняет статистические талоны, карты экстренного извещения, бланки направлений на лечебно-диагностические исследования, вносит в медицинскую карту данные флюорографического и других исследований, помогает заполнять посыльные листы во ВТЭК, санаторно-курортные карты, выписки из медицинских карт и др., при соответствующих условиях под диктовку врача пишет рецепты и заполняет соответствующие графы листка нетрудоспособности, медицинские карты амбулаторного больного и др.;
- выдает очередные талоны для повторных посетителей;
- объясняет больному способы подготовки к лабораторным, инструментальным и другим исследованиям, методы подготовки и сдачи материала для лабораторных исследований;
- принимает активное участие в организации и проведении диспансерной работы на участке: под контролем и руководством врача заполняет диспансерную карту на вновь выявленных больных ведет картотеку диспансеризуемых в поликлинике больных в соответствии со сроками динамического наблюдения изучает условия жизни диспансерных больных и о результатах наблюдения докладывает участковому врачу подготавливает предварительные материалы для составления отчета по диспансеризации населения участка;
- помогает в осуществлении госпитализации больных;
- комплектует необходимым инструментарием и медикаментами сумку врача для оказания неотложной медицинской помощи при амбулаторном приеме и на дому.

При оказании медицинской помощи на дому участковая сестра проводит следующую работу:

- посещает больных на дому и выполняет манипуляции в соответствии с назначением;
- комплектует соответствующим набором медикаментов, перевязочного материала и медицинского инструментария специальную сумку, предназначенную для посещения больных на участке;
- информирует врача о своевременности выполнения больным диагностических и лечебных процедур;
- при необходимости привлекает общественный актив к уходу за больными на дому;
- проводит профилактические прививки.

Кроме того, медицинская сестра отчитывается об израсходованных медикаментах перед старшей сестрой отделения.

1.2.Терминология, используемая при проектировании системы

Когда говорят о логическом проектировании, употребляют такие термины, как *сущность*, *связь* и*атрибут*.

Сущность - это множество однотипных объектов, называемых экземплярами, при этом каждый экземпляр индивидуален и отличается от всех остальных экземпляров. Типичными примерами сущностей могут быть сущности "Врач", "Пациент", "Специальность", а примерами экземпляров - "Лутфуллаев", "Норматов", "терапевт". Как правило, сущности именуются в единственном числе. Сущности рекомендуется снабжать текстовым описанием (хотя бы для того, чтобы через несколько месяцев не забыть, что имелось в виду).

Атрибут - это характеристика сущности. Атрибут выражает одно законченное и определенное свойство сущности (например, дату рождения пациента). При проектировании данных рекомендуется создавать атомарные атрибуты (например, выделение страны и города в отдельные атрибуты при

описании адреса в дальнейшем позволит анализировать экземпляры сущности "Пациент" по признаку принадлежности к той или иной стране или тому или иному городу).

Связь - это логическое отношение между сущностями, выражающее некоторое ограничение или бизнес-правило. Связь обычно именуется глаголом.

При логическом проектировании нередко также создаются связи "многие ко многим". Типичный пример такой связи - связь врач-пациент, когда любой пациент может лечиться у любого врача. В случае такой связи в общем случае невозможно определить, какой экземпляр одной сущности соответствует выбранному экземпляру другой сущности, что делает неосуществимой физическую (на уровне индексов и триггеров) реализацию такой связи между соответствующими таблицами. Поэтому перед переходом к физической модели все связи "многие ко многим" должны быть переопределены (некоторые CASE-средства, если таковые используются при проектировании данных, делают это автоматически). Обычно для этой цели вводится промежуточная сущность, обладающая ассоциативными атрибутами. В нашем примере объектом, представляющим собой экземпляр подобной сущности, может являться посещение пациентом врача (то есть реальное событие), либо регистрирующий его документ - статистический талон), а ассоциативными атрибутами, отличающими по существу один ее экземпляр от другого, могут быть дата посещения, цель посещения и иные содержащиеся в статистическом талоне сведения.

Говоря о связях, нельзя не остановиться на понятии ключей и ключевых атрибутов. *Первичным ключом* называется атрибут или группа атрибутов, однозначно идентифицирующих каждый экземпляр сущности. Нередко возможны несколько вариантов выбора первичного ключа. Например, в небольшой организации первичными ключами сущности "сотрудник" могут быть как табельный номер, так и комбинация фамилии, имени и отчества (при уверенности, что в организации нет полных тезок), либо номер и серия

паспорта (если паспорта есть у всех сотрудников) . В таких случаях при выборе первичного ключа предпочтение отдается наиболее простым ключам (в данном примере - табельному номеру). Другие кандидаты на роль первичного ключа называются *альтернативными ключами*.

Отметим, что обязательным требованием, предъявляемым к первичному ключу, является его уникальность - два экземпляра сущности не должны иметь одинаковых значений первичного ключа. Другим важным требованием является компактность - сложный первичный ключ не должен содержать ни одного атрибута, удаление которого не приводило бы к утрате уникальности. Помимо этого, первичный ключ не должен содержать пустых значений. При выборе первичного ключа рекомендуется выбирать атрибут, значение которого не меняется в течение всего времени существования экземпляра (в этом случае табельный номер предпочтительнее фамилии, так как ее можно сменить, вступив в брак).

При создании связей между сущностями (например, "один ко многим") в дочернюю сущность передаются атрибуты, составляющие первичный ключ в родительской сущности. Эти атрибуты образуют в дочерней сущности *внешний ключ*.

Нормализация данных. Одним из основополагающих принципов проектирования данных является принцип нормализации, позволяющей существенно сократить объем хранимой информации и устраняющей аномалии в организации хранения данных. Степень нормализации данных может быть различной.

Говорят, что модель данных соответствует *первой нормальной форме*, если в таблицах отсутствуют группы повторяющихся значений. Это соответствие достигается путем выделения атрибутов с повторяющимися значениями в отдельные сущности, созданием или выбором для них новых первичных ключей и установлением связей "один ко многим" от новых сущностей к старым. При этом первичные ключи новых сущностей станут внешними ключами для старой сущности.

Говорят, что модель данных соответствует *второй нормальной форме*, если в сущностях, содержащих составной первичный ключ, неключевые атрибуты зависят от всего первичного ключа. Если же в какой-либо сущности имеется зависимость каких-либо неключевых атрибутов от части ключа, следует выделить их в отдельную сущность, сделав первичным ключом новой сущности ту часть первичного ключа, от которой зависят данные атрибуты, и установить связь "один ко многим" от новой сущности к старой.

Говорят, что модель данных соответствует *третьей нормальной форме*, если в сущностях отсутствует взаимозависимость между неключевыми атрибутами. Это соответствие достигается путем выделения в отдельную сущность атрибутов с одной и той же зависимостью от неключевого атрибута, использования атрибутов, определяющих эту зависимость, в качестве первичного ключа новой сущности и установки связи "один ко многим" от новой сущности к старой сущности.

Результатом нормализации является модель данных, которую легко поддерживать, не содержащая неопределенностей в данных и повторений данных.

Отметим , что проектирование данных гораздо удобнее производить, используя современные средства проектирования баз данных, и, в частности, CASE-средства, так как с их помощью можно автоматизировать рутинную работу по созданию собственно объектов базы данных на основе созданных логической модели и физической моделей данных.

1.3.Постановка задачи

В последнее время уровень конкуренции для медицины возрос, так как на рынке увеличился уровень спроса, что связано с открытием новых частных поликлиник, повышением уровня качества лечения и обслуживания в государственных поликлинических учреждениях.

Требуется создание полнофункциональной информационной системы, использование которой будет способствовать повышению эффективности работы предприятия в целом, переходу сети поликлиник на качественно новый уровень обслуживания и лечения пациентов.

Прохождение лечения пациента в поликлиники подразумевает:

- оформление личной карточки пациента;
- хранение личной карточки пациента, историй всех болезней, поставленных диагнозов, результатов проведения исследований / анализов;
- направление пациента к врачу, на проведение исследования, сдачу анализов;
- оформление справок / больничных листов.

В связи с этим необходимо создание информационной системы, использование которой позволит решить задачу оперативной постановки пациента на учет:

- ✓ упрощение доступа к персональным данным пациента;
- ✓ быстрое доведение результатов проведения исследований / анализов пациента до лечащего врача;
- ✓ сокращение штатной численности отдела регистратуры, расходов на зарплату и сокращение людских и временных затрат на обработку информации;
- ✓ централизованное хранение всех данных о пациенте;
- ✓ уменьшение количества противоречивых данных;
- ✓ упрощение постановки диагноза пациенту.

Выводы

В этом разделе работы рассматривались современные информационные технологии в медицине, Описание предметной области, Терминология, используемая при проектировании системы и постановка задачи.

2. ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ.

2.1 Концептуальная модель проектной области

Диаграмма вариантов использования, моделирующих функциональную (процессную) структуру предметной области посредством вариантов использования и отношений между ними (Рисунок 2.1., Рисунок 2.2.).

Разработка диаграммы вариантов использования

Для создания главной диаграммы вариантов использования в программе Rational Rose необходимо выполнить следующие действия:

Дважды щелкнуть по пункту Main (Главная диаграмма) в разделе Use Case View (Представление прецедентов) в списке браузера, чтобы открыть диаграмму.

В списке браузера выбрать актера или требуемый прецедент и перетащить его на диаграмму с помощью мыши.

Актеры и прецеденты могут быть получены прямо на диаграмме с использованием панели инструментов. Чтобы создать коммуникативные ассоциации в программе Rational Rose необходимо:

На панели инструментов щелкнуть по кнопке Association (Ассоциативная связь) или по кнопке Unidirectional Association (Однонаправленная ассоциативная связь). Если нужная кнопка отсутствует нужно щелкнуть правой кнопкой мыши на панели инструментов, в появившемся контекстно-зависимом меню выбрать команду Customize (Настройка), чтобы добавить кнопку.

Щелкнуть по актеру – инициатору связи – и перетащить возникшую линию связи на нужный прецедент.

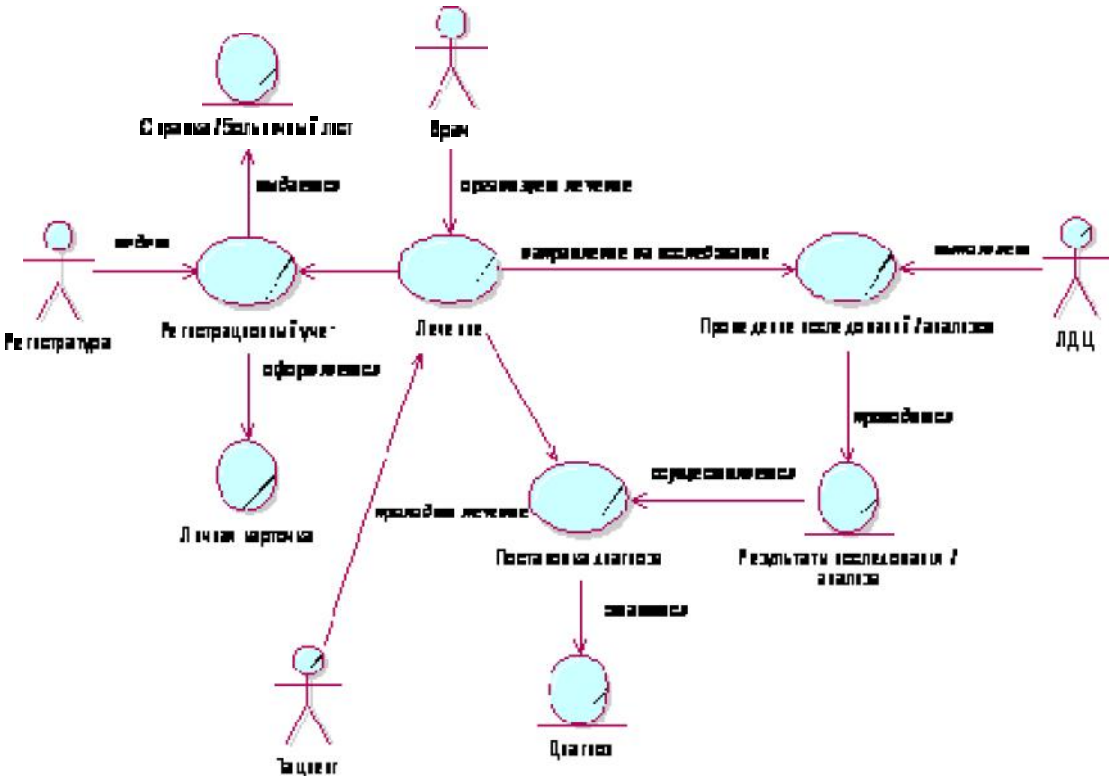

Рисунок 2.1 – Диаграмма вариантов использования. (Usecasediagram)

Так как основной задачей разрабатываемой системы является организация постановки пациента на учет и контроль процесса прохождения лечения, то необходимо рассмотреть диаграмму активности постановки пациента на учет (Рисунок 2.1.).

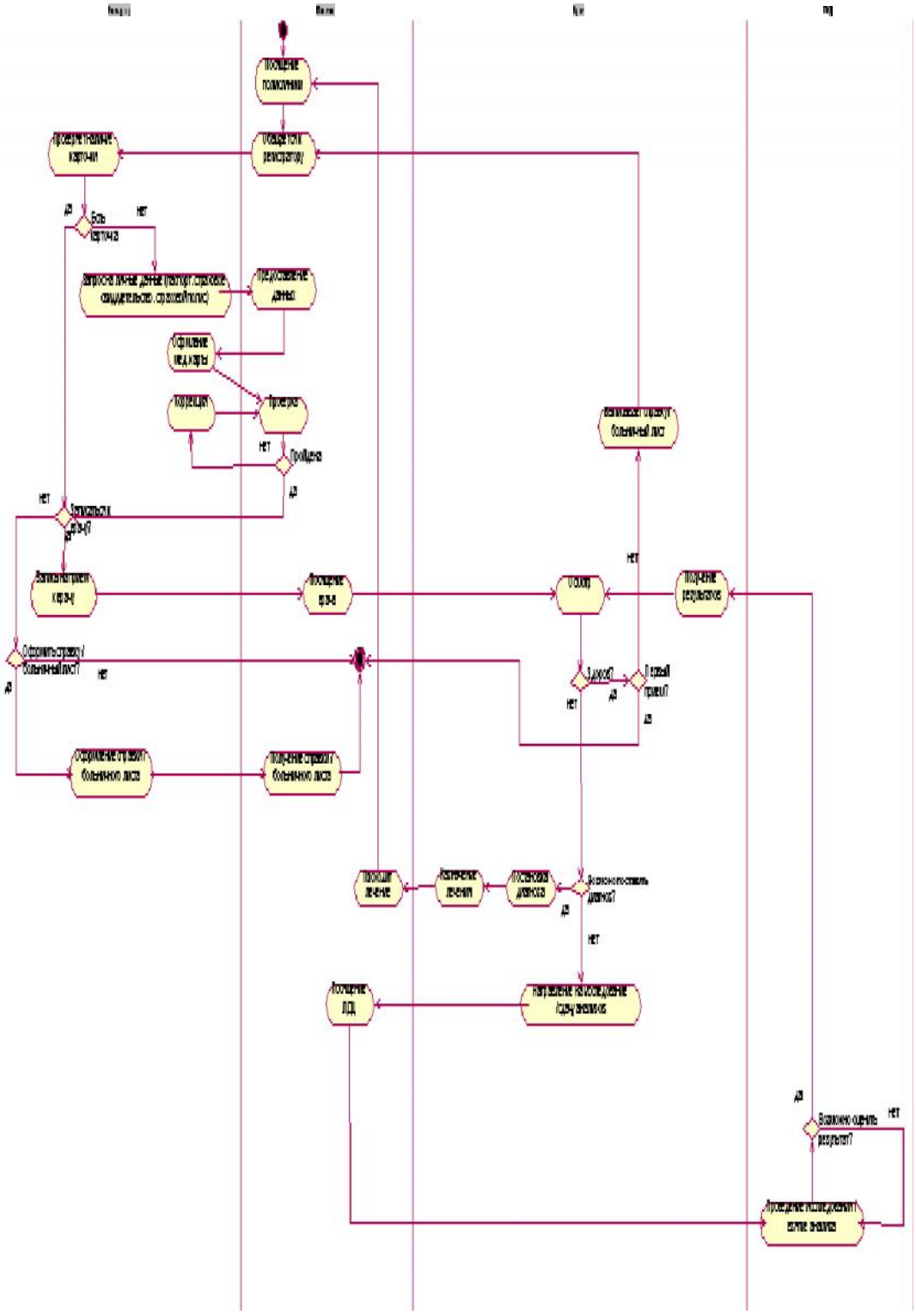

Рисунок 2.2 – Диаграмма активности.(Sequencediagram)

Отношения между ключевыми объектами представлено на диаграмме классов (Рисунок 2.3).

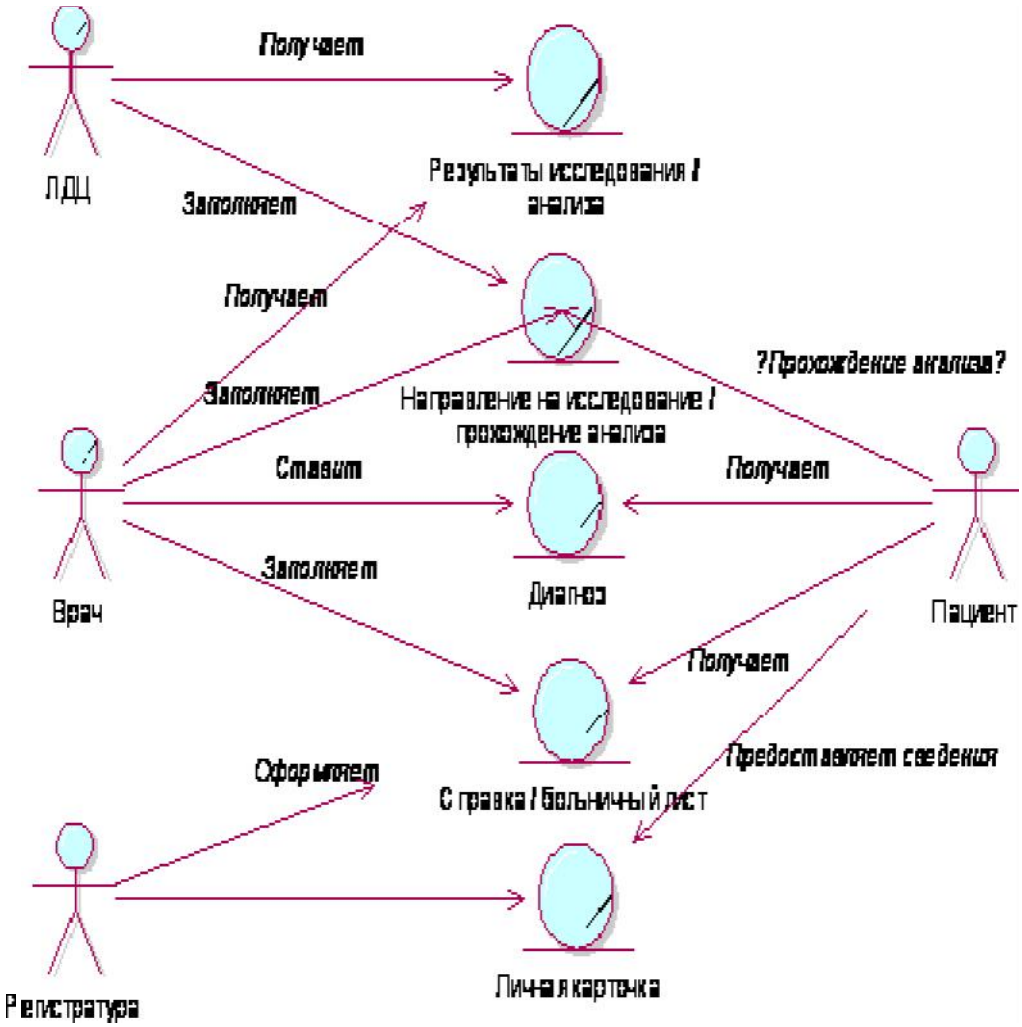

Рисунок 2.3 – Диаграмма классов. (Classdiagram)

Описание проблем и формирование концепции информационной системы.

Решение проблемы предметной области

После проведения исследования предметной области были выявлены следующие проблемы:

- с увеличением количества пациентов сети поликлиник, увеличивается количество информационных потоков, что приводит к снижению управляемости из-за несвоевременного получения информации, ошибочной информации или вовсе из-за отсутствия таковой. Это является недопустимым,

так как может затянуть процесс лечения, привести к неверной и (или) несвоевременной постановки диагноза;

- затраты большого количества временных и человеческих ресурсов на обработку документов и необходимость увеличивать эти ресурсы с ростом количества клиентов;

- тратится много времени на подготовку и поиск необходимых историй болезней, личных карточек, результатов обследований;

- в связи с большим бумажным документооборотом возрастает нагрузка на персонал, в связи с чем возрастает количество ошибок;

- при обращении пациента в различные поликлиники сети могут произойти ошибки при передаче сведений;

- из-за необходимости получать документы и данные из других подразделений возникают временные задержки, связанные с их доставкой.

Концепция информационной системы

Основные понятия**.** Пациент – человек, получающий медицинскую помощь, подвергающийся медицинскому наблюдению и/или лечению по поводу какого-либо заболевания. Регистратура – основное структурное подразделение по организации приема больных в поликлинике и на дому.Врач – специалист с высшим медицинским образованием, который занимается предотвращением (профилактикой), распознанием (диагностикой) и лечением заболеваний и травм.

Лечебно-диагностический центр – специально оборудованный комплекс, приспособленный для проведения различных медицинских исследований и диагностик.

Функциональные требования

Основные требования.

К информационной системе предъявляются следующие основные требования:

- содержать электронные формы стандартных документов (личная медицинская карта, история болезни, результаты анализов / исследований);

- позволять заполнять имеющиеся стандартные формы используя устройства ввода;

- иметь возможность хранения электронных (отсканированных) копий документов;

- иметь возможность формирования различных типовых отчетов (по пациентам, по отделениям, по заболеваниям);

- централизованно хранить информацию для доступа к ней из любой поликлиники.

Обеспечивающие требования

К информационной системе предъявляются следующие Обеспечивающие требования:

- иметь возможность внесения дополнительных форм документов и отчетов;

- обеспечивать защиту информации от несанкционированного доступа и изменения;

- иметь возможность резервного копирования данных;

- проверка на правильность ввода данных.

Вспомогательные требования.

- иметь удобную систему поиска и фильтрации по различным параметрам;

- возможность печати (на принтере) отчетов, форм и электронных копий документов;

- возможность сохранения данных, отчетов, форм в отдельный файл для отправки по факсу или электронной почтой.

Нефункциональные требования.

- работа в операционной системе Windows;

- наглядный пользовательский интерфейс для простоты и удобства работы пользователя;

- возможность хранения большого объема электронных документов.

Концептуальная модель информационной системы

Для разработки архитектуры ИС, целесообразно использовать шаблон трехслойной архитектуры. Основные высказывания:

Слой представления – предоставляет услуги отображения данных, обработки событий пользовательского интерфейса (щелчки мыши, нажатия клавиш). В общем случае, охватывает все, что имеет отношение к общению пользователя с ИС.

Предметная область – Выполняет вычисления на основе вводимых и хранимых данных, проверку всех элементов данных и обработка команд, поступающих от слоя представления, а так же передачу информации стою источника данных.Выполняет обращение к БД, обмен сообщениями, мониторинг транзакций.

Результаты разработки представлены в виде диаграммы классов на Рисунок 2.4. , описание назначения классов по слоям – в таблице 1.

Таблица 2.1 – Назначение классов концептуальной модели

№	Наименование класса	Назначение класса
Слой представления		
	E-UI-Registrator	Граничный класс, отвечающий за отображение формы личной карты пациента, параметров и результатов поиска.
	E-UI-Doctor	Граничный класс, отвечающий за отображение формы истории болезни пациента.
	E-UI-LDC	Граничный класс, отвечающий за отображение формы сведений о прохождении исследовании / сдачи анализов пациентом.
	ControllerTreatment	Управляющий класс, методы которого отвечают за управление приложением в целом
Слой предметной области		
	CallService	Граничный класс, отвечающий за взаимодействие с

		классами слоя предметной области.
	PatientData	Класс хранения, содержащий ключевые данные о пациенте.
	eDiagnose	Класс хранения, содержащий сведения о поставленном диагнозе
	eResult	Класс хранения, содержащий данные результатов исследования.
	eNaprav	Класс хранения, содержащий сведения о направлении пациента на исследования / сдачу анализов.
	eMedcard	Класс хранения, содержащий медицинскую карточку.
	eOperator	Класс хранения, содержащий сведения об операторах, работающих с ИС.
	AccessList	Класс хранения, содержащий права доступа операторов ИС.
Слой источника данных		
	Data	Граничный класс, отвечающий за взаимодействие с БД.

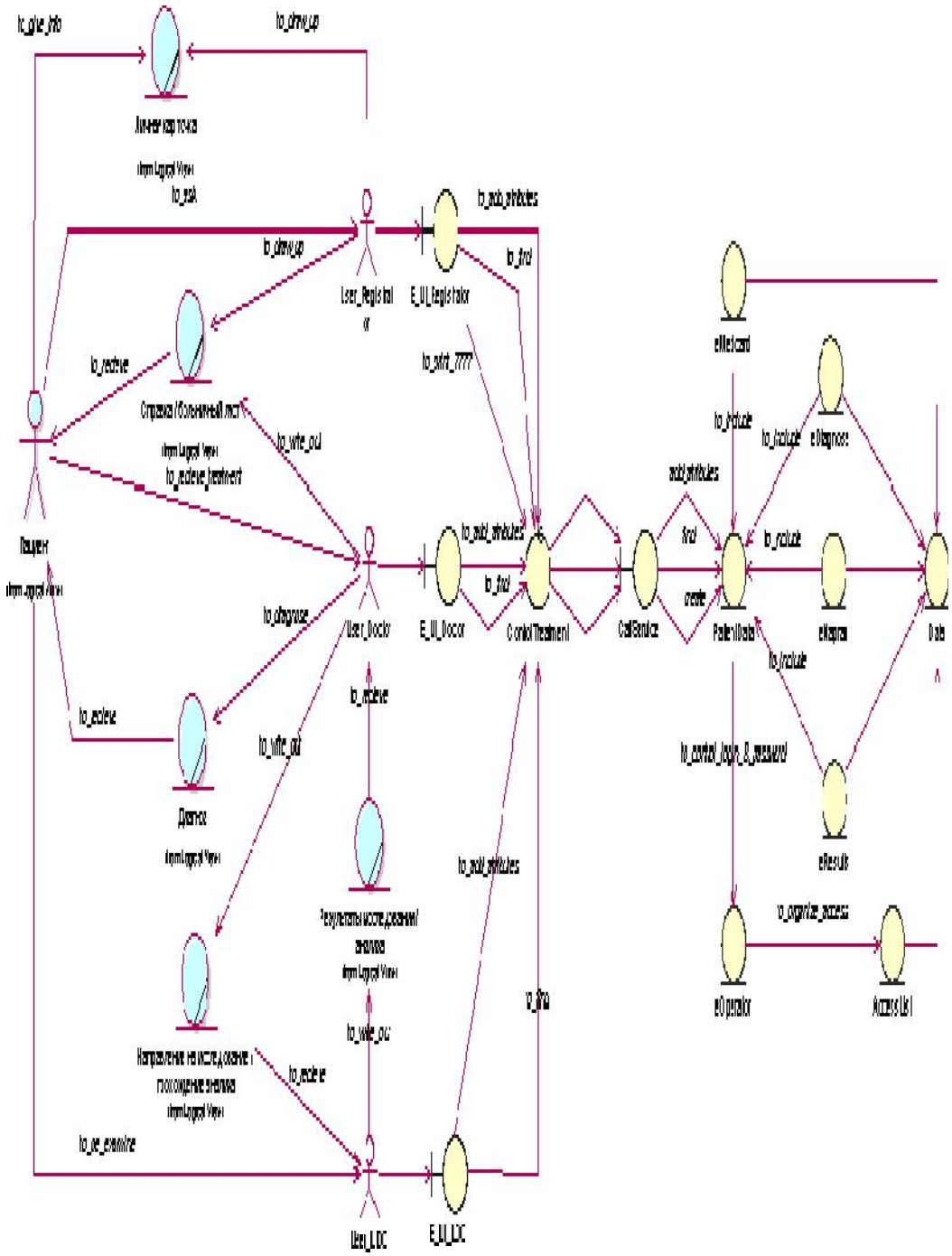

Рисунок 2.4 – Диаграмма классов, моделирующая структуру ИС на концептуальном уровне.

Логическая модель информационной системы

Модель поведения

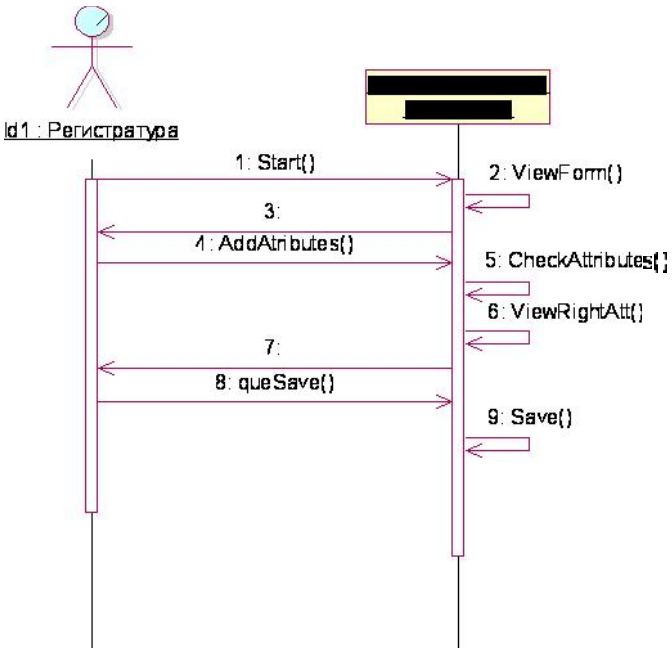

Рисунок 2.5 – Диаграмма последовательности, моделирующая функцию создания новой записи.

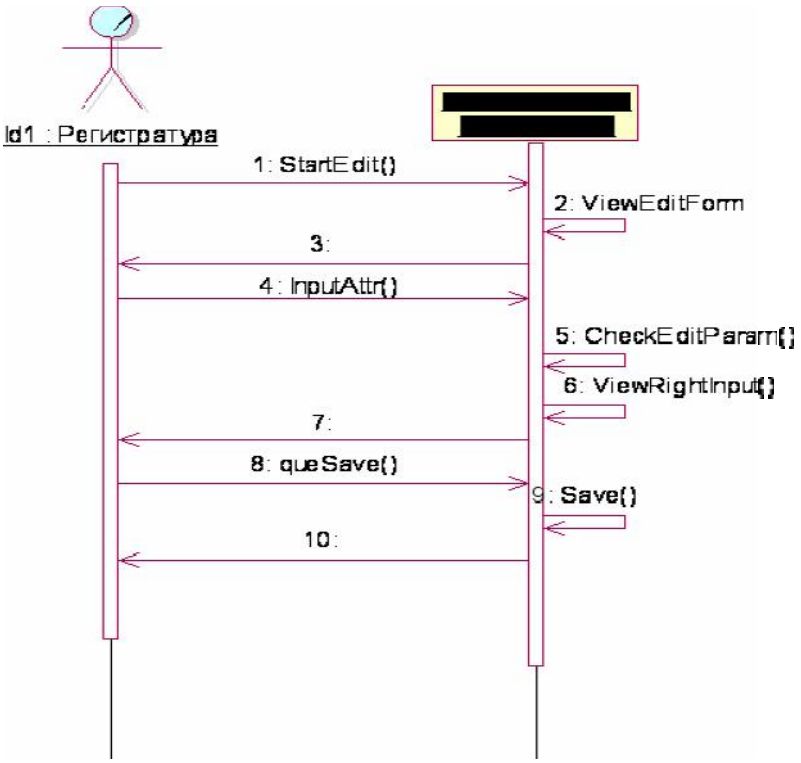

Рисунок 2.6 – Диаграмма последовательности, моделирующая функцию редактирования записи.

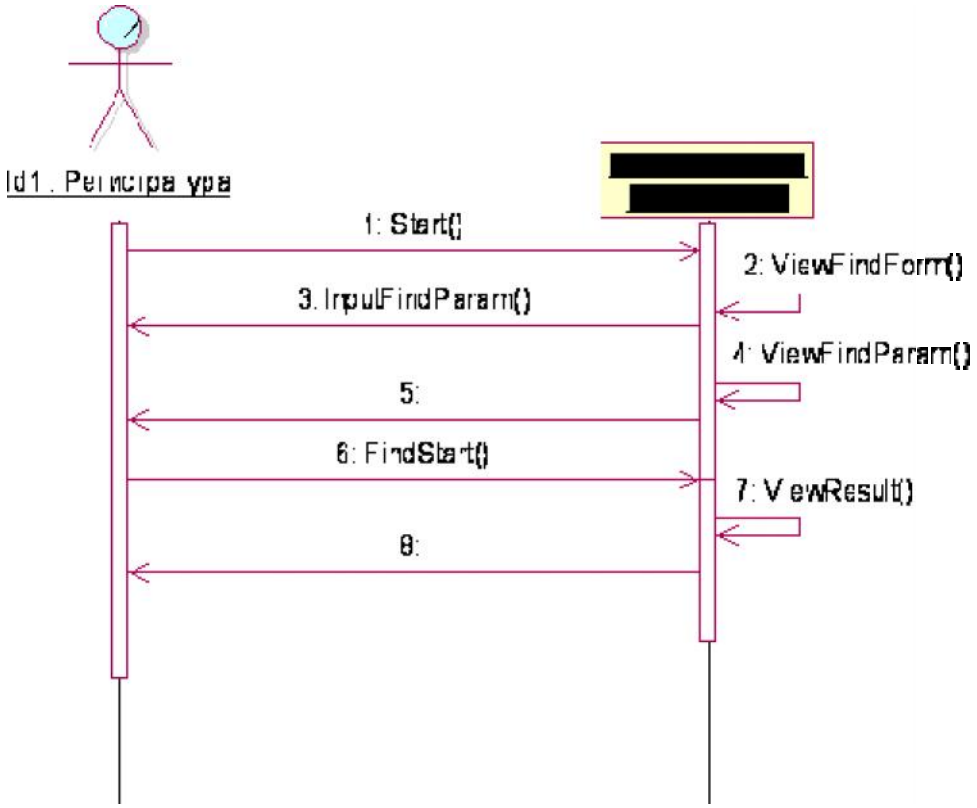

Рисунок 2.7– Диаграмма последовательности, моделирующая функцию поиска записи.

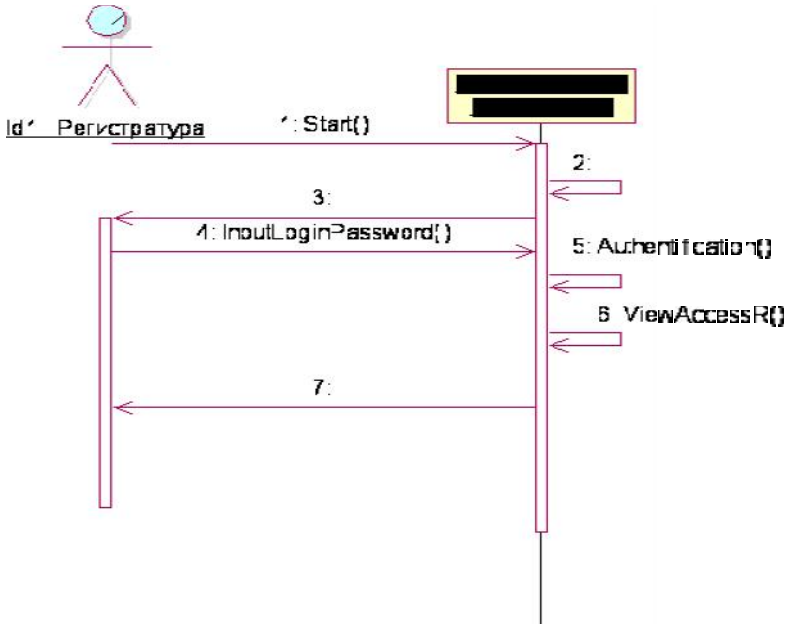

Рисунок 2.8 – Диаграмма последовательности, моделирующая функцию аутентификации оператора.

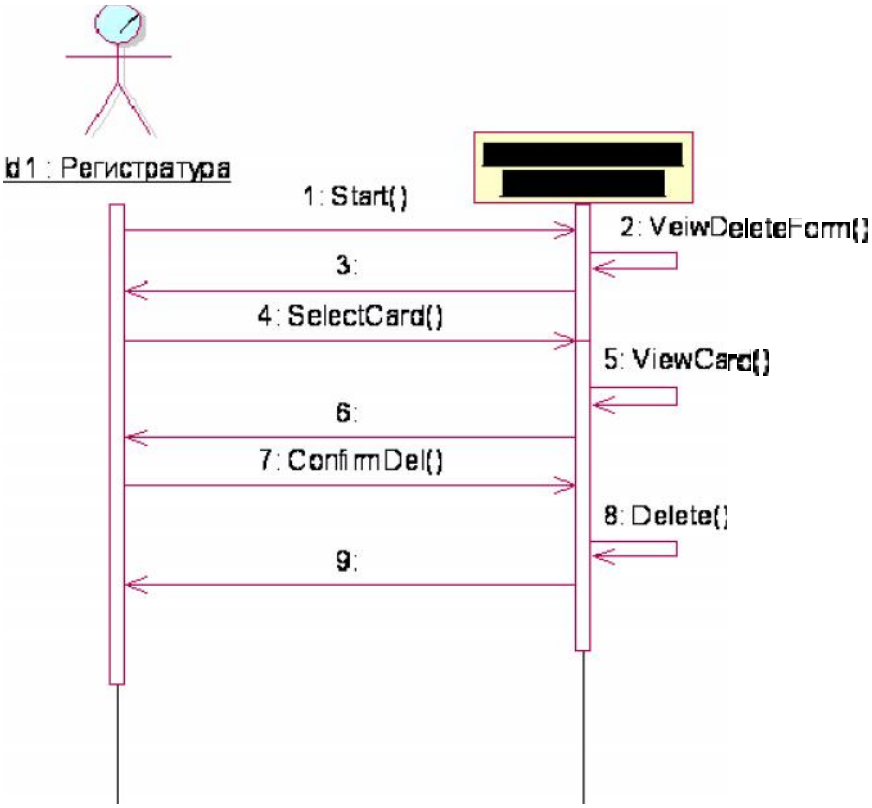

Рисунок 2.9– Диаграмма последовательности, моделирующая функцию удаления.

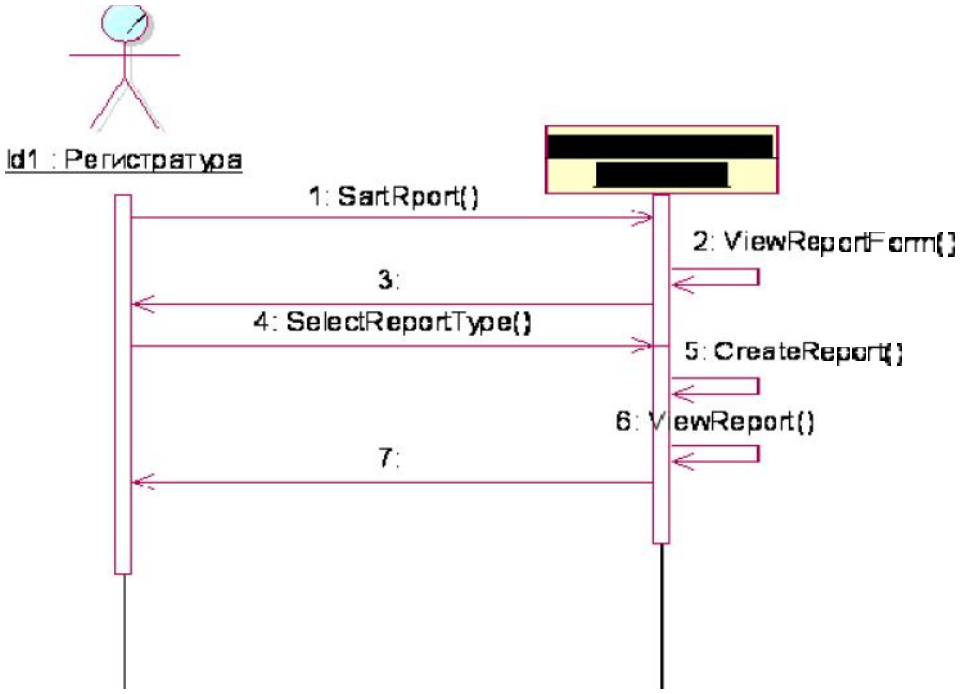

Рисунок 2.10 – Диаграмма последовательности, моделирующая функцию создания отчета.

Модель структуры

На Рисунок 2.11 представлена диаграмма классов ПО ИС, на которой отражены все классы, составляющие ПО ИС постановки пациента на учет в поликлинике. Данная диаграмма представляет всю модель структуры ПО ИС и получена в результате решения задач первой итерации проектирования. При необходимости проектант, получив целевую модель структуры, может вернуться к уточнению требований, сформулированных в концепции и выполнить вторую итерацию проектирования, результатом которой может быть уточненные модели поведения и структуры, как на концептуальном, так и на логическом уровне.

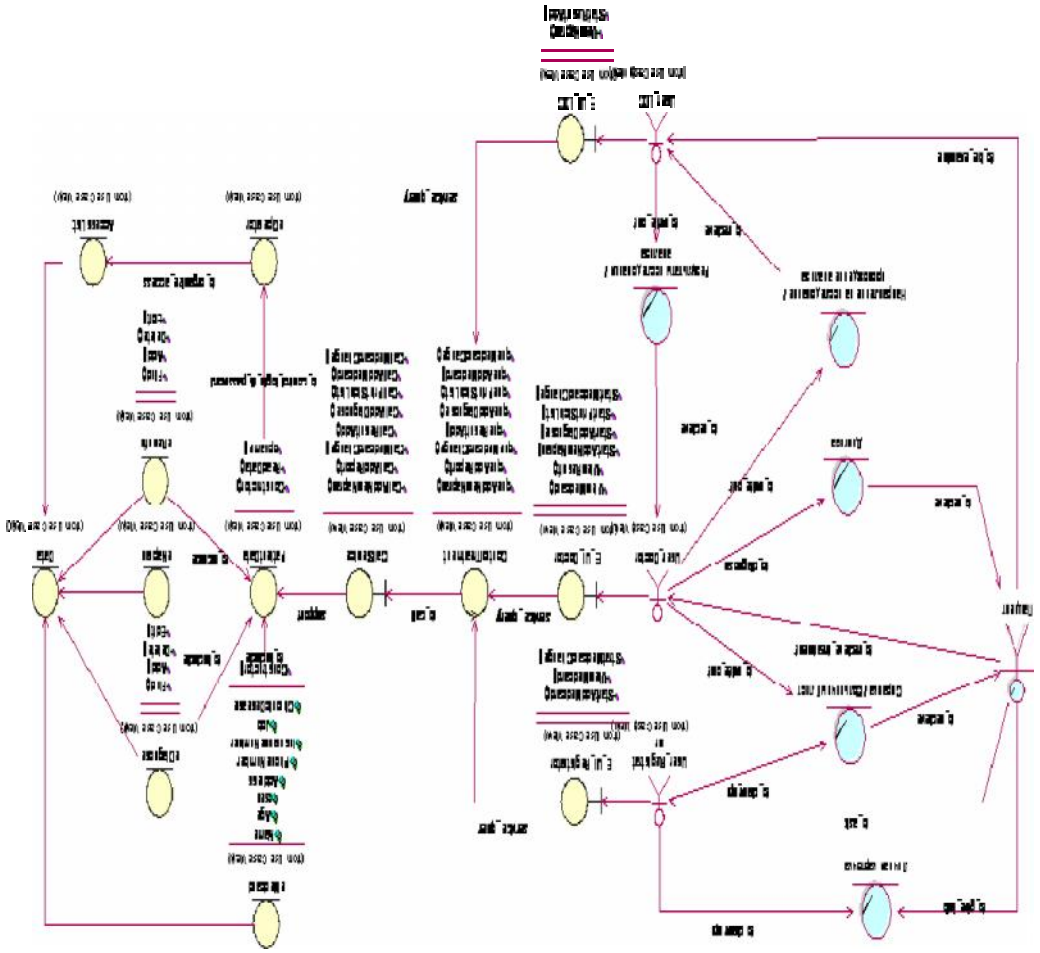

Рисунок 2.11 – Диаграмма классов, моделирующая структуру ПО ИС на логическом уровне.

2.2.Реализация модели в среде CASE-средства

Начало работы над проектом

Запустить программу Rational Rose Enterprise Edition. Создать новый проект: FiIe→New.

После того, как проект будет создан и работа с ним будет завершена, необходимо сохранить полученные диаграммы. Для этого в меню File выбрать пункт Save или Save As, дать имя проекту и сохранить его в файл с расширением *.mdl.

Разработка диаграммы действий

Диаграммы действий в Rational Rose создаются следующим образом:

Щелкнуть правой кнопкой мыши по разделу Use Case View (Представление прецедентов) в списке браузера.

В появившемся контекстно-зависимом меню выбрать команду New → Activity Diagram (Создать →Диаграмма действий). В список будет добавлена новая диаграмма, которой нужно дать имя.

Для того, чтобы открыть диаграмму необходимо дважды щелкнуть по ней мышью в браузере.

Действием называется исполнение определенного поведения в потоке управления системы.

Создание действий:

Щелкнуть по кнопке Activity (Действие) на панели инструментов. Щелкнуть по диаграмме действий, чтобы переместить элемент, изображающий действие, на диаграмму.

Ввести имя нового действия. Переходы используются для изображения пути потока управления от действия к действию.

Создание переходов:

Щелкнуть по кнопке State Transition (Переход) на панели инструментов. Щелкнуть по начальному действию на диаграмме и переместить стрелку перехода на последующее действие.

При моделировании управляющих потоков системы часто требуется показать места их разделения на основе условного выбора. Переходы из элемента выбора содержат ограничительные условия, определяющие, какое направление перехода будет выбрано. Элементы выбора и условия позволяют задавать альтернативные пути потока управления.

Создание элементов выбора. Щелкнуть по кнопке Decision (Элемент выбора) на панели инструментов.

Щелкнуть по диаграмме действий, чтобы переместить на нее элемент выбора. Ввести имя нового элемента.

Секции делят диаграммы действий на несколько участков. Это нужно для того, чтобы показать, кто отвечает за выполнение действий на каждом участке.

Алгоритм создания секции:

Щелкнуть по кнопке Swimlane (Секция) на панели инструментов. Щелкнуть по диаграмме действий, чтобы создать на ней новую секцию.

Дважды щелкнуть по названию новой секции, чтобы открыть диалоговое окно Specification (Параметры) и в поле Name ввести нужное название секции. Нажать кнопку OK.

Для обозначения начального и конечного состояний в потоке управления системы используются специальные символы.

Последовательность создания начального и конечного состояний:

Щелкнуть по кнопке Start State (Начальное состояние) или End State (Конечное состояние) на панели инструментов.

Щелкнуть по диаграмме действия, чтобы поместить не нее символ конечного или начального состояния.

Разработка диаграммы классов

Главная диаграмма классов в логическом представлении модели обычно отображает пакеты системы. Каждый пакет также имеет свою главную диаграмму классов, которая обычно содержит общедоступные классы пакета. Другие диаграммы создаются по необходимости.

Программа Rational Rose автоматически создает главную диаграмму классов в логическом представлении модели.

Чтобы добавить пакеты к главной диаграмме классов нужно дважды щелкнуть по пункту списка Main Diagram (Главная диаграмма) в браузере, чтобы открыть диаграмму, а затем, выбрав нужный пакет, перетащить его на диаграмму.

Для создания главной диаграммы класса пакета в программе Rational Rose необходимо дважды щелкнуть по изображению пакета на диаграмме классов. После того, как пакет откроется, появится главная диаграмма классов. Для добавления классов, необходимо выбрать нужный класс и перетащить его с помощью мыши на диаграмму. Для отображения стереотипа класса на диаграмме можно воспользоваться командой Format → Stereotype Display (Формат → Показать стереотип).

Для установки/сброса видимости класса нужно установить/сбросить флажок Show Visibility (Показать видимость) на вкладке Diagram контекстно-зависимого меню Options (для конкретного класса) или меню Tools → Options (для отображения всех классов).

Модель классов разрабатывается аналогично, но более подробно.

Разработка модели поведения

Для создания диаграммы последовательности действий в программе Rational Rose необходимо добавить в список браузера новую диаграмму. Для этого нужно щелкнуть правой кнопкой мыши по папке Logical View (Логическое представление) и в появившемся контекстно-зависимом меню выбрать команду New → Sequence Diagram (Создать → Диаграмма последовательности действий).

Для создания объектов и сообщений на диаграмме последовательности действий, прежде всего, нужно ее открыть, затем выбрать на панели инструментов сообщение или объект и перетащить его на диаграмму.

Выводы

В этом разделе работы рассматривались проектирование информационной системы, концептуальная модель проектной области, реализация модели в среде CASE-средства.

3.ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ИНФОРМАЦИОННОЙ СИСТЕМЫ

3.1. Реализация законченных результатов разработки

Электронная клиника– это сеть универсальных общедоступных медицинских учреждений по оказанию комплексной амбулаторно-поликлинической помощи, использующих в своей работе современные медицинские технологии и высокие профессиональные стандарты диагностики и лечения всех наиболее распространенных заболеваний.

Ожидаемые результаты от проекта можно использовать в больницах, частных клиниках и сельских медицинских пунктов. А также, сотрудники системы здравоохранения республиканских органов управления, специалисты, участвующие в разработке инфраструктуры информационно-телекоммуникационных систем “Электронного правительства”, могут использовать в процессе разработки.

Структура работы системы

3.2.Рукаводства пользователя

рисунок 3.1-Окно авторизации

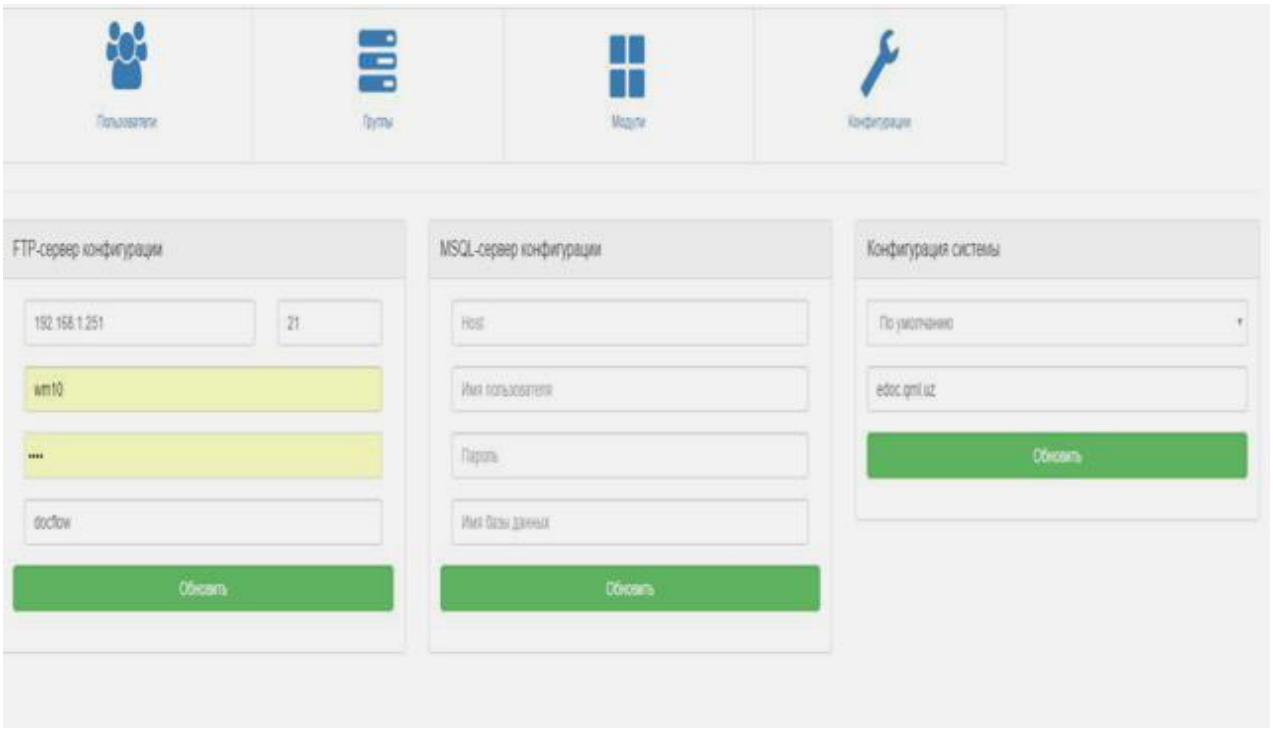

рисунок 3.2-Панель Администратора

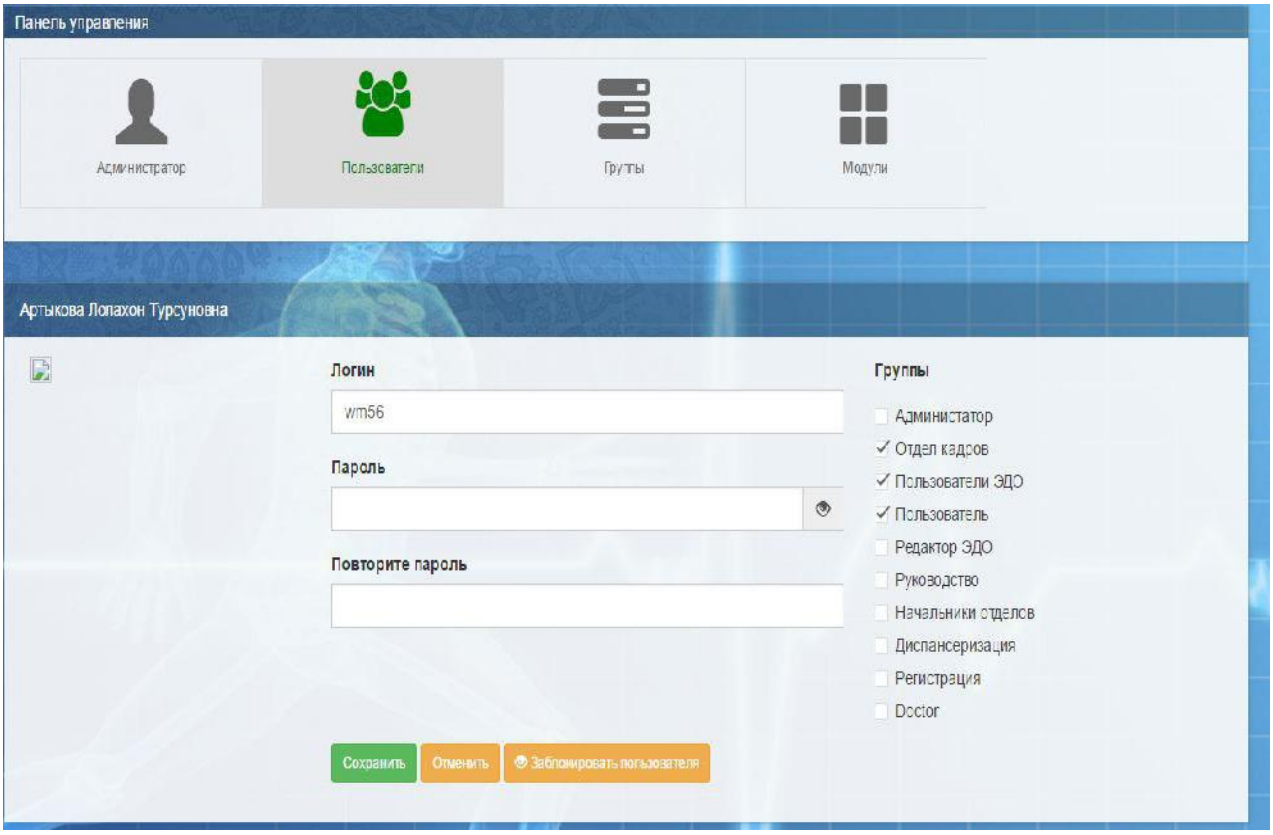

Рисунок 3.3-Окно регистрации нового пользователя

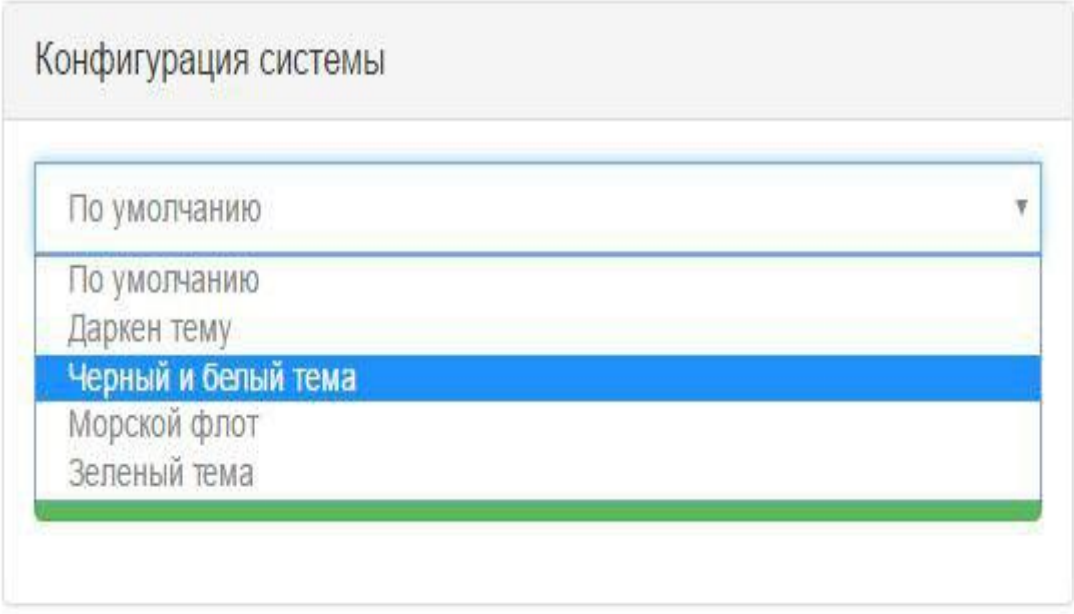

Рисунок 3.4-Окно оформления интерфейса системы

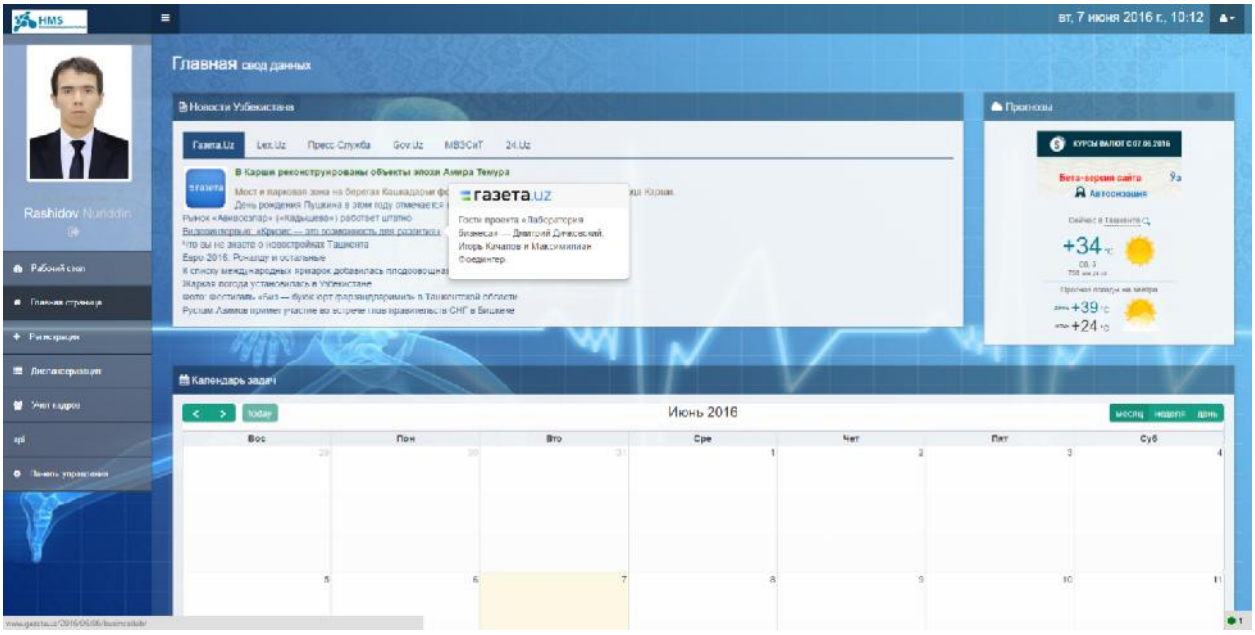

Рисунок 3.5- Окно рабочий стол врача

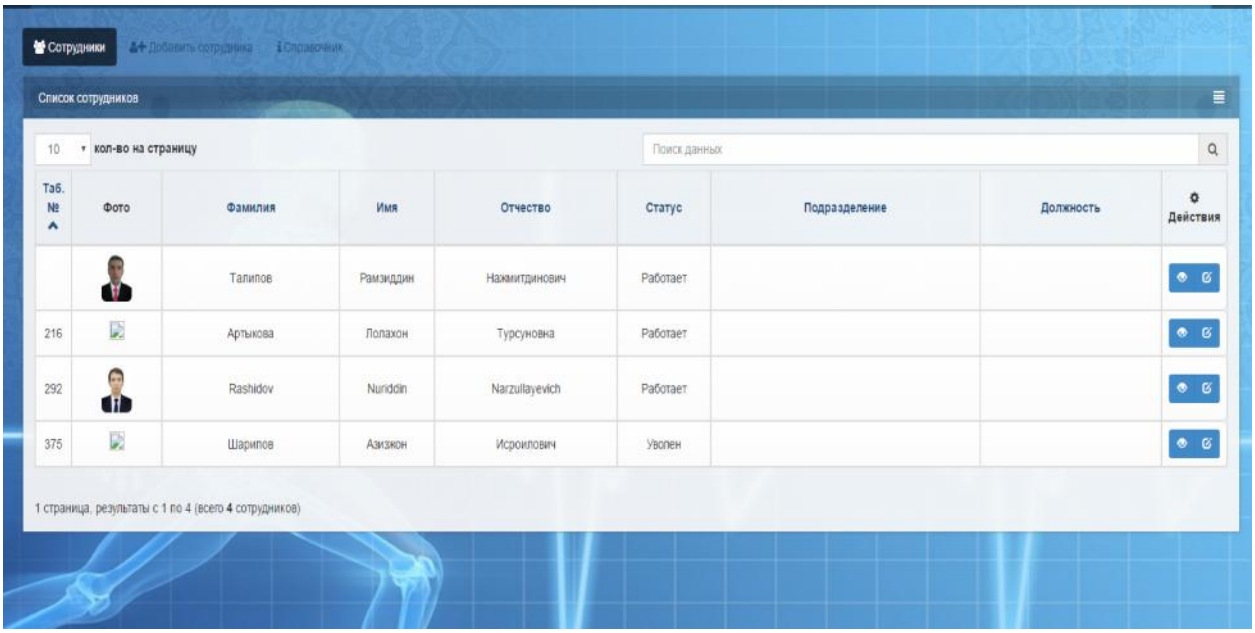

Рисунок 3.6-Окно управления личными данными врачей

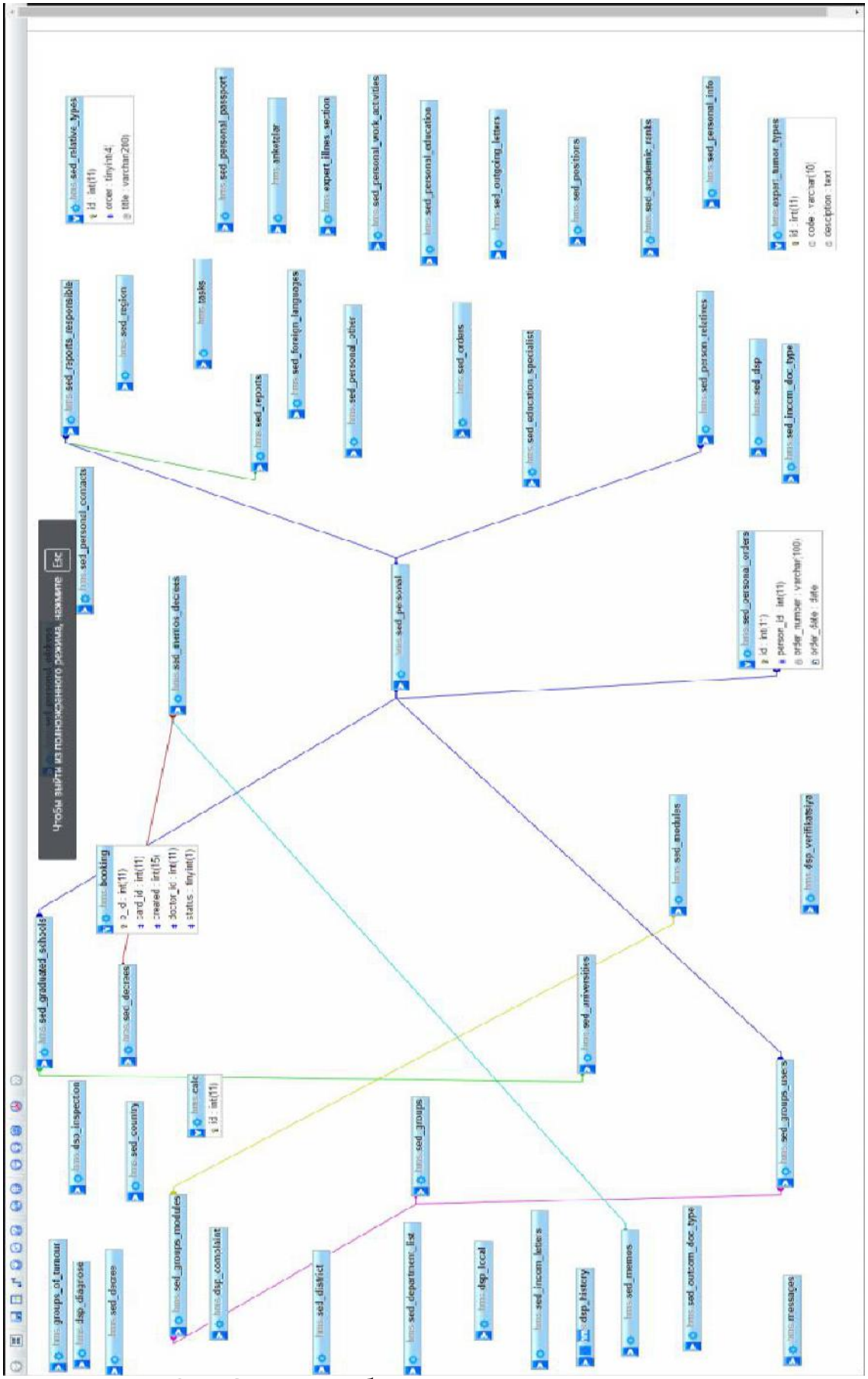

Рисунок 3.7- Структура базы данных

4. БЕЗОПАСНОСТЬ ЖИЗНЕДЕЯТЕЛЬНОСТИ И ЭКОЛОГИЯ

4.1 Создание оптимального микроклимата в производственном помещении

Определенное состояние воздушной среды производственных помещений является необходимым, а часто и решающим условием для стабильной и долговременной работы многих электронных устройств и средств связи.

Поддержание заданных параметров воздушной среды в производственных условиях обеспечивают системы кондиционирования микроклимата, которые являются совокупностью пассивных и активных инженерных средств и устройств.

Пассивные средства включают в себя различного рода ограждения и тепловую изоляцию помещений. В большинстве случаев, при существующих зданиях и сооружениях изменение их конструкции не представляется возможным или экономически невыгодно. К активным средствам обеспечения параметров микроклимата относятся системы вентиляции, отопления и кондиционирования воздуха. Грамотная их организация, в большинстве случаев, позволяет добиться в производственных помещениях оптимальных условий для функционирования технологического оборудования.

Системы кондиционирования воздуха (СКВ) являются активной, обычно регулируемой системой, предназначенной для комплексного поддержания заданных параметров внутреннего воздуха. Система кондиционирования воздуха может работать в здании совместно с системами отопления и системами вентиляции, но зачастую система кондиционирования воздуха берет на себя функции последних и создает в здании или, по крайней мере, в его наиболее ответственных помещениях необходимые климатические условия, как в холодный, так и в теплый период года.

Еще совсем недавно при создании заданных параметров микроклимата в основном ориентировались на использование центральных не автономных

систем кондиционирования и централизованных станций холодоснабжения. В настоящее время наблюдается отход от центральных систем. Подсчитано, что в настоящий момент до 75% систем кондиционирования воздуха разрабатываемых для новых или перестроенных машинных залов, помещений для оборудования средств связи и т. д., ориентированы на использование автономных агрегатированных систем кондиционирования.

Размещение всей системы кондиционирования воздуха в самом обслуживаемом помещении и простое присоединение к внутренним и наружным установкам для отвода тепла, (которые могут находиться в любой безопасной зоне здания) фактически гарантируют полную защиту системы.

В настоящее время значительная часть (15-20%) капитальных вложений на строительство производственных зданий, где размещается электронное оборудование или ведутся современные технологические процессы, приходится на системы кондиционирования воздуха и сопряженные с ними устройства. Затраты на эксплуатацию этих систем могут достигать 60% от общих затрат на эксплуатацию зданий.

Микроклимат помещения — это совокупность факторов, определяющих метеорологическую обстановку в нем. К числу этих факторов относятся: температура и влажность воздуха, потоки лучистого тепла, определяющие радиационную температуру помещений, газовый состав атмосферы, наличие твердых примесей в атмосфере, подвижность воздуха и фактор стабильности вышеуказанных параметров.

Поддержание температурного режима в помещении

Температура воздуха является одним из основных факторов, характеризующих климатические условия помещений.

Большинство производителей электронной техники регламентируют температурные условия ее эксплуатации в пределах 22-26°С. Гигиенические нормы допустимых и оптимальных параметров микроклимата производственных помещений зависят от периода года, категории (тяжести) выполняемой работы и т.д. Для легкой категории работ допустимые нормы

температуры составляют 19-23°С. В любом производственном помещении техники всегда можно разместить больше чем людей. Следовательно, компьютерный зал или производственное помещение с электронным оборудованием имеют удельную тепловую нагрузку существенно большую, чем офисное помещение. Такие помещения требует значительно большей охлаждающей способности в расчете на квадратный метр площади пола. Как правило, требуется одна тонна комфортно кондиционированного воздуха на каждые 25-30 м² площади офиса, и в то же время около одной тонны прецизионно кондиционированного воздуха на каждые 5-10 м² компьютерного зала.

Оборудование для комфортного кондиционирования воздуха обычно разрабатывается с учетом потребной холодопроизводительности в 1 кВт на 8 м², тогда как прецизионный кондиционер разработан с учетом 2.5 кВт на 1 м².

Комнатные кондиционеры, кондиционеры централизованной системы жилого здания, а также системы кондиционирования воздуха служебных помещениях (комфортные систем кондиционирования) работают с коэффициентом ощутимого охлаждения около 0.60-0.70. Это значит, что 60-70% работы системы кондиционирования уходит на понижение температуры воздуха, а 30-40% — на удаление влаги. Такая система вполне оправдывает себя в помещениях с большим числом персонала и умеренным потоком входящих и выходящих людей.

Прецизионные (технологические) системы кондиционирования воздуха, в отличие от комфортных, за счет специальных технических средств имеют значительно более высокий коэффициент охлаждающей способности (ощутимой), равный 0.85-0.95. Примерно 85-95% работы прецизионной системы воздушного кондиционирования уходит на охлаждение воздуха, а 5-15% — на удаление влаги.

Еще одной важной характеристикой является стабильность заданных параметров температурного режима помещения. Прецизионные системы кондиционирования воздуха легко справляются с такими трудностями.

Точность поддержания температуры обычно составляет до 2°С, в отдельных случаях до 0.6°С. Лучшее что можно ожидать от комфортных систем это 5°С.

Поддержание влажностного режима помещения

Во многих производственных помещениях, наряду с жесткими требованиями к тепловому режиму, существует необходимость точно выдерживать и влажностный режим. При высокой относительной влажности, например в компьютерном зале или узле связи, могут возникнуть проблемы с конденсацией атмосферной влаги на частях оборудования, что неизбежно приводит к коррозии различных узлов электронного оборудования и неизбежному выходу его из стоя. Наоборот, при низкой относительной влажности из-за эффекта электризации и возникновении статического электричества (потенциала) могут возникнуть нарушения в работе оборудования, потери части информации вплоть до выхода из строя отдельных элементов электронных плат и иных элементов электронного оборудования.

Практически все производители современного электронного оборудования графу технического паспорта «условия эксплуатации» заносят сведения о влажностном режиме помещения, причем диапазон рекомендуемых значений относительной влажности обычно лежит в пределах 45-55%.

В отличие от комфортного, прецизионный кондиционер предназначен для поддержания заданных параметров микроклимата исходя из требований технологического оборудования, для которого не существует жестких требований по скорости распространения воздушных масс в объеме помещения. Прецизионный кондиционер обрабатывает существенно большие массы воздуха, что при одной и той же холодопроизводительности с комфортным позволяет поднять температуру теплопередающих поверхностей выше температур конденсации атмосферной влаги (выше точки росы), исключив (или снизив до минимума) процесс осушения воздуха. Кроме того, при необходимости существует возможность понизить температуру

поверхности испарителя и осуществлять процесс осушения воздуха. В состав оборудования прецизионного кондиционера входит специальный блок генератора пара (увлажнителя), который может быть бойлерного типа, ультразвуковой, с инфракрасными излучателями и т. д.

В заключении если в помещении где размещено электронное оборудование отказаться от поддержания требуемой влажности воздуха, то в следствии необратимых процессов (разрушения лака на электронных печатных платах, окисления контактов, высыхания изоляции силовой и коммутационной проводки и т. д.) сроки службы оборудования значительно снижаются, а эксплуатационные затраты (ремонт, обслуживание) на основное оборудование резко возрастают.

4.2. Защита от электромагнитного излучения

В процессе эксплуатации электроэнергетических установок - открытых распределительных устройств (ОРУ) и воздушных линий (ВЛ) электропередачи напряжением 400 кВ и выше – отмечено ухудшение состояния здоровья персонала, обслуживающего эти установки.

Субъективно это выражается в ухудшении самочувствия работающих – повышенная утомляемость, вялость, головные боли, плохой сон, боли в сердце и т. п. В электроустановках напряжением менее 400 кВ также возникают электромагнитные поля, но менее интенсивные и, как показывает длительный опыт эксплуатации таких установок, не оказывающие отрицательного влияния на биологические объекты.

Интенсивное электромагнитное поле промышленной частоты вызывает у работающих нарушение функционального состояния центральной нервной системы, сердечной деятельности и системы кровообращения. При этом наблюдаются повышенная утомляемость, снижение точности рабочих движений, изменение кровяного давления и пульса, возникновение болей в сердце, сопровождающихся сердцебиением и аритмией, и т. п.

Эффект воздействия электромагнитного поля на биологический объект принято оценивать количеством электромагнитной энергии, поглощаемой этим объектом при нахождении его в поле. Электромагнитное поле можно рассматривать состоящим из двух полей: электрического и магнитного. Можно также считать, что в электроустановках электрическое поле возникает при наличии напряжения на токоведущих частях, а магнитное - при прохождении тока по этим частям.

При малых частотах, в том числе при 50 Гц, электрическое и магнитное поля практически не связаны между собой, поэтому их можно рассматривать отдельно друг от друга и также отдельно рассматривать влияние, оказываемое ими на биологический объект. Исходя из этого, определена поглощаемая телом человека энергия электрического и магнитного полей. При этом в любой точке электромагнитного поля, возникающего в электроустановках промышленной частоты, поглощенная телом человека энергия магнитного поля примерно в 50 раз меньше поглощенной им энергии электрического поля. Вместе с тем измерениями в реальных условиях установлено, что напряженность магнитного поля в рабочих зонах ОРУ и ВЛ напряжением до 750 кВ включительно не превышает 20 - 25 А/м, в то время как вредное действие магнитного поля на биологический объект проявляется при напряженности 150 - 200 А/м.

Это позволило сделать вывод, что отрицательное действие на организм человека электромагнитного поля в электроустановках промышленной частоты обусловлено электрическим полем; магнитное же поле оказывает незначительное биологическое действие и в практических условиях им можно пренебречь.

Электрическое поле электроустановок частотой 50 Гц можно рассматривать в каждый данный момент как электростатическое поле, т. е. применять к нему законы электростатики. Это поле создается между двумя электродами (телами), несущими заряды разных знаков, на которых начинаются и оканчиваются силовые линии. Поле электроустановок является

неравномерным, т. е. напряженность его изменяется вдоль силовых линий. Вместе с тем оно обычно несимметричное, поскольку возникает между электродами различной формы, например между токоведущей частью и землей или металлической заземленной конструкцией. Поле воздушной линии электропередачи является, кроме того, плоскопараллельным, т.е. форма которого одинакова в параллельных плоскостях, называемых плоскостями поля. В данном случае плоскости поля перпендикулярны оси линии.

Процесс биологического действия электрического поля на организм человека изучен недостаточно. Предполагается, что нарушение регуляции физиологических функций организма - изменение кровяного давления, пульса, нарушение сердечного ритма - обусловлено воздействием поля на различные отделы нервной системы. При этом повышение возбудимости центральной нервной системы происходит за счет рефлекторного действия поля, а тормозной эффект вызывается прямым воздействием поля на структуры головного и спинного мозга. Считается, что кора головного мозга, а также промежуточный мозг особенно чувствительны к воздействию электрического поля.

Предполагается также, что основным материальным фактором, вызывающим такие изменения в организме, является индуцируемый в теле ток и в значительно меньшей мере - электрическое поле.

Наряду с биологическим действием электрическое поле обусловливает

возникновение разрядов между человеком и металлическим предметом, имеющим иной потенциал, чем человек.

Если человек стоит непосредственно на земле или на токопроводящем заземленном основании, то потенциал его тела практически равен нулю, а если он изолирован от земли, то тело оказывается под некоторым потенциалом, достигающим иногда нескольких киловольт.

Очевидно, что прикосновение человека, изолированного от земли, к заземленному металлическому предмету, так же как и прикосновение

человека, имеющего контакт с землей, к металлическому предмету, изолированному от земли, сопровождается прохождением через человека в землю разрядного тока, который может вызвать болезненные ощущения, особенно в первый момент. Часто прикосновение сопровождается искровым разрядом.

В случае прикосновения к изолированному от земли металлическому предмету большой протяженности (трубопровод, проволочная ограда на деревянных стойках и т. п.) или большого размера (например, крыша деревянного здания) ток через человека может достигать значений, опасных для жизни.

4.3. Соизмерения природных и производственных потенциалов территории.

Техногенное загрязнение среды является наиболее очевидной и быстродействующей негативной причинной связью в системе экосферы: «экономика, производство, техника, среда». Оно обусловливает значительную часть природоемкости техносферы и приводит к деградации экологических систем, глобальным климатическим и геохимическим изменениям, к поражениям людей. На предотвращение загрязнения природы и окружающей человека среды направлены основные усилия современной экологии.

Преодоление экологического кризиса требует определения допустимой антропогенной нагрузки на биосферу, соизмерения природных и производственных потенциалов территории, нормирования техногенных воздействий, т.е. экологической регламентации хозяйственной деятельности человека. Не менее важно обеспечить всесторонний и объективный контроль за выполнением экологических регламентов на глобальном, региональном и локальном уровнях, - то, что может быть реализовано еще до глубокой экологизации экономики и производства.

Наиболее полно эти требования могут быть реализованы в пределах такого природно-хозяйственного комплекса, который образует равновесную

эколого-экономическую систему. Понятие эколого-экономической системы (ЭЭС) широко используется в современной экономической и экологической литературе наряду с близкими по смыслу понятиями «природно-экономическая система», «биоэкономическая система* и «природно-техническая система».

В настоящее время существует два уровня интерпретации понятия ЭЭС - глобальный и территориальный. Согласно первому ЭЭС трактуется как тип экологически ориентированной социально-экономической формации. Необходим переход человечества от экономической системы к эколого-экономической системе. Но в глобальном смысле пока что это отдаленная и довольно абстрактная перспектива. Для практической реализации принципа сбалансированного природопользования важно иметь представление об ЭЭС на территориальном уровне - в отдельных регионах и промышленных комплексах.

В такой трактовке эколого-экономическая система - это ограниченная определенной территорией часть техносферы, в которой природные, социальные и производственные структуры и процессы связаны взаимоподдерживающими потоками вещества, энергии и информации. В литературе по инженерной экологии довольно широко употребляется понятие природно-технической системы. Под природно-технической системой (ПТС) понимают совокупность природных и искусственных объектов, сформировавшуюся на какой-то территории в результате строительства и эксплуатации промышленных комплексов, инженерных сооружений и технических средств, взаимодействующих с компонентами природной и социальной среды.

К сожалению, реальные ПТС никто никогда не рассматривал с позиций эколого-экономического баланса. Индустриальное развитие никогда не ставило своей целью создание сбалансированных ЭЭС. А механизмы экологической регламентации хозяйственной деятельности, такие, как оценка предполагаемых воздействий на окружающую среду и экологическая

экспертиза программ и проектов, сами по себе не в состоянии обеспечить практическую реализацию требований сбалансированности. Но это не означает, что такие системы невозможны. Следует только различать понятия «сбалансированная эколого-экономическая система» и «сбалансированное эколого-экономическое развитие». Последнее обычно предполагает коэволюцию живой природы и общества, т.е. по существу согласование скоростей естественной эволюции и общественного прогресса. Вот это действительно невозможно.

Модели ЭЭС: структура и потоки. Сейчас известно много попыток моделирования ЭЭС. Региональные ЭЭС обычно представляются в виде блочных моделей, в которых анализируются связи, но нет подходов к количественной экологической регламентации.

ЭЭС представляет собой сочетание совместно функционирующих экологической и экономической систем, обладающее эмерджентными свойствами. Напомним, что экосистема - это сообщество живых организмов, так взаимодействующих между собой и со средой обитания, что поток энергии создает устойчивую структуру и круговорот веществ между живой и неживой частями системы. В свою очередь экономическая система является организованной совокупностью производительных сил, которая преобразует входные материально-энергетические потоки природных и производственных ресурсов в выходные потоки предметов потребления и отходов производства. Таким образом, часть материальных элементов экологической системы, в том числе и элементов среды обитания человека используется как ресурс экономической системы.

Соизмерение производственных и природных потенциалов территории - одна из актуальных задач промышленной экологии, без решения которой невозможна выработка научно обоснованной системы экологических регламентации. Соизмерение не сводится лишь к подчинению экологическому императиву - требованию природных систем и их защитников уменьшить индустриальную экспансию. Сбалансированность нужна не только природным

комплексам и среде обитания людей, но и самому хозяйству. Она имеет не только природоохранное и гигиеническое значение, но и прямое экономическое: равновесное сопряжение производственных и экологических процессов не столько принуждает к ограничению входных мощностей, сколько предлагает дополнительный экономический инструмент контроля эффективности производства. Экономический рост, превышающий порог допустимых нагрузок, выступает как основной дестабилизирующий фактор для окружающей среды. Именно поэтому соизмерение и согласование экономических и природных потенциалов и формирование эколого-экономической системы должно быть предметом экономической теории и практики.

Сама по себе процедура соизмерения основана на определении и сопоставлении *экологической техноемкости территории* (ЭТТ или ПДТН) и *природоемкости хозяйства территории.* Эта процедура практически совпадает с оценкой безопасности территориальных комплексов.

Необходимость смены техногенного типа развития требует введения экологических ограничений или экологических нормативов. Экологическая техноемкость территории и предельно допустимая техногенная нагрузка по существу являются универсальными территориальными экологическими нормативами, предназначенными для *регламентации хозяйственной деятельности.* Но как раз ЭТТ и ПДТН законодательно не утверждены как нормативы.

Заключение

В результате дипломного работы была разработано информационная система «Электронная клиника» (административная часть)». Основой для создания информационной системы послужили проблемы предметной области. В качестве среды разработки было выбрано CASE-средство фирмы Rational Software Corporation – Rational Rose 2000, предназначенное для автоматизации этапов анализа и проектирования предметной области.

Определены задачи системы и будут сгруппированы задачи. Разработаны базы данных системы. Разработаны базы данных системы и программный комплекс для обеспечения управления базы данных. Сформулированы части системного администратора. Разработан программный комплекс для поиска документов из системы БД. Создан контрольный модуль испольняющий задачи. Разработаны классы программного комплекса, обеспечивающие поиск информации по модулям и обеспечивающий ввод, вывод и редактирование данных. Разработана политика безопастности системы. Создан профиль администратора и модули данных. Разработан модуль для обработки документов. Разработаны формы интерфейса и общий дизайн системы. Создана инструкция для пользователей.

Использование данной ИС упрощает доступ к персональным данным пациента, централизует хранение всех данных о пациенте и уменьшает количество противоречивых данных. Благодаря этому представляется возможность сократить численности административно-управляющего персонала и расходов на зарплату, избежать снижение пропускной способности сети поликлиник при увеличении количества пациентов, повысить уровень качества обслуживания и лечения. Все это приводит к повышению управляемости компанией в целом и, как следствие, к увеличению прибыли.

Литература

1. Дрожжина Д. Подходы к исследованию выпускников: зарубежный опыт // Мониторинг университета. 2012. № 9. С. 7–14.

2. Головничер М.Н. Проектирование информационных систем // Структура информационных систем. 2013. С. 46-78.

3. Мосиенко Н. Л., Черкашина Т. Ю. Мониторинг выпускников вуза: варианты оценки результативности обуче-ния // Вестн. Новосиб. гос. ун-та. Серия: Социально-экономические науки. 2014. Т. 14, вып. 1. С. 127–140.

4. Отчёт Всемирного банка. Модернизация системы высшего образования в Узбекистане. //Возможности модернизации и улучшения системы высшего образования. 2014. № 88606. С. 98-105.

5. Структура сайта Туринского политехнического университета в городе Ташкенте - https://polito.uz/index.php/ru/discover-the-polytechnic. Дата обращения: 25.03.2015 г.

Приложение

```
<?php



class Utility{

    public $lang = 'ru';
    public $config = array();
    public $db = null;
    public $rootPath = null;
    public $person_id = null;
    public $person = array();
    public $personHash = null;
    public static  $path = array();
    public $dictionary = array();
    const PROJECT_DIR = 'lame';
    public $module = null;
    /// ACL libs


    ////

    public $imageTypes = array(
        'png' => 'image/png',
        'jpg' => 'image/jpg',
        'jpeg' => 'image/jpeg',
        'gif' => 'image/gif'
    );
```

```
public static $docExtensions = array(
        'doc',
        'docx',
        'pdf',
        'jpg',
        'jpeg',
        'xls',
        'xlsx',
        'tiff',
        'tif'
    );

public $visitor = array(
    'uid' => null,
    'type' => 'visitor'
);

public $viewTypes = array('blocks','layouts');

public function mergeArrays(){
    $args = func_get_args();
    $arrays = array();
    foreach($args as $array){
        if(is_array($array)){
            foreach($array as $key => $value){
                $arrays[$key] = $value;
            }
        }
    }
    return $arrays;
```

```
}
public function __construct(){
    $this->getRootPath();
    $this->setPaths();
    $this->loadConfig();
    $this->isSession();
    $this->lang();
    $this->dbConnect();
    $this->getVisitor();
    $this->loadVisitor();
    $this->getPerson();

    if(isset($_SESSION['module'])){
        $this->module = $_SESSION['module'];
    }
}

private function setPaths(){
    self::$path = array(
        "config" => $this->rootPath."/config",
        "modules" => $this->rootPath."/modules",
        "function" => $this->rootPath."/function",
        "locale" => $this->rootPath."/function/language"
    );

    return self::$path;
}

public function getModule(){
```

```
    }

    public function getVisitor(){
       if(isset($_SESSION['ID'])){
          $this->visitor['uid'] = $_SESSION['ID'];
       }
       return $this->visitor;
    }

    public function setVisitor($visitor){
       $this->visitor = $visitor;
       $_SESSION['ID'] = $this->visitor;
       return $this->visitor;
    }

    public function loadVisitor(){
        if($this->visitor['uid']!==null){
          $q = $this->db->prepare("SELECT sp.per_id,sp.depar_id,sp.temp_pos,sp.position,sp.name,sp.surename,sp.middle_name,deps.department_number from sed_personal as sp LEFT JOIN sed_department_list as deps ON(deps.did = sp.depar_id) where sp.per_id=:uid limit 1");
          $q->execute(array('uid' => $this->visitor['uid']));
          $q = $q->fetch(PDO::FETCH_ASSOC);
          $r = $this->db->prepare("SELECT * FROM sed_positions WHERE id = :pos limit 1");
          $r->execute(array('pos' => (!empty($q['temp_pos']) && $q['temp_pos']!==null ? $q['temp_pos'] : $q['position'])));
          $r = $r->fetch(PDO::FETCH_ASSOC);
          $this->visitor['rank'] = $r['rank'];
```

```
            $this->visitor['department'] = $q['depar_id'];
            $this->visitor['department_number'] = $q['department_number'];
        }
    }

    public function loadDictionary($type = null){
        if($type!==null){
            $file = self::$path["locale"]."/".$type."-".$this->lang.".json";
            if(file_exists($file)){
                $array = json_decode(file_get_contents($file),true);
                return $array;
            }else{
                throw new ErrorException("Language file doesn't exist...");
            }
        }else{
            throw new ErrorException("No type given for dictionary...");
        }

    }

    public static  function shuffleDate($date = '',$symbol_1 = '',$symbol_2 = ''){
        if(!empty($date) && !empty($symbol_1) && !empty($symbol_2)){
            $date = explode($symbol_1,$date);
            if(count($date)>0){
                $date = array_reverse($date);
                $date = implode($symbol_2,$date);
                return $date;
            }
        }
```

```
    return false;
}

public function translate($str = null,$placeholders = array()){
    if($str!==null && isset($this->dictionary[$str])){
        $str = $this->dictionary[$str];
    }
    if(count($placeholders)>0){
        foreach($placeholders as $key=>$value){
            $str = str_replace("%{$key}%",$value,$str);
        }
    }

    return $str;
}

public function createTable($data = array()){
    $str = '';
    if(is_array($data) && count($data)>0){
        foreach($data as $key => $value){
            $str.='<tr>';
            foreach($value as $val){
                $str.='<td>'.$val.'</td>';
            }
            $str.='</tr>';
        }
    }
    return $str;
}
private function isSession(){
```

```
        if(!isset($_SESSION)){
            session_start();
        }
    }
    private function lang(){
        if($_SESSION && isset($_SESSION['lang'])){
            if(in_array($_SESSION['lang'],$this->config['locales'])){
                $this->lang = $_SESSION['lang'];
            }
            $_SESSION['lang'] = $this->lang;
        }
    }

    public function setLang(){
        if(isset($_GET['lang']) && !empty($_GET['lang'])){
            if($_SESSION){
                $_SESSION['lang'] = $_GET['lang'];
            }
        }
    }

    private function getPerson(){
        if(isset($_SESSION['current_person'])){
            $q = $this->db->prepare("SELECT
per_id,name,surename,middle_name FROM sed_personal where
per_id='".$_SESSION['current_person']."' limit 1");
            $q->execute();
            $row = $q->fetch(PDO::FETCH_ASSOC);
            if($q->rowCount()>0){
                $this->person_id = $row['per_id'];
```

```
                $this->person = array(
                    'first' => $row['name'],
                    'last' => $row['surename'],
                    'middle' => $row['middle_name']
                );
            }
        }
    }

    public function isRecordAllowed(){
        if($this->person_id==null){
            return false;
        }
        return true;
    }

    private function dbConnect($conf = null){
        if(isset($this->config['db'])){
            $this->db = new PDO('mysql:host='.$this->config['db']['host'].';dbname='.$this->config['db']['database'], $this->config['db']['user'], $this->config['db']['password'],array(PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES ".$this->config['db']['charset']));
            $this->db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
            $this->db->setAttribute(PDO::ATTR_DRIVER_NAME,'mysql');
        }else{
            throw new ErrorException("The config parameter was empty");
        }
```

```
        }

        public function getRootPath(){

            if(isset($_SERVER['DOCUMENT_ROOT'])){
                $this->rootPath =
str_replace('\\','/',$_SERVER['DOCUMENT_ROOT']);
            }else{
            if(self::PROJECT_DIR){
                $path = explode(self::PROJECT_DIR,realpath(__FILE__));
                if(count($path)==2){
                    $this->rootPath = str_replace('\\','/',$path[0].self::PROJECT_DIR);
                }else{
                    throw new ErrorException("Invalid project root directory...");
                }
            }else{
                throw new ErrorException("Project root directory is not defined...");
            }

            }

        }

        private function loadConfig(){
            $file = file_get_contents($this->rootPath.'/config/config.json');
            $json = json_decode($file,true);
            if(is_array($json)){
                $this->config = $json;
            }else{
                throw new Exception("The config file is corrupted...");
```

```
    }

}

public function renderView($type,$path = null,$params = array()){
    if($path!==null && $type!==null){
        $mpath = explode('/',$path);
        if(count($mpath)==2 && in_array(strtolower($type),$this->viewTypes)){
        $module =& $mpath[0];
            $view =& $mpath[1];
        $file = $this->rootPath.'/modules/'.$module.'/View/'.ucfirst($type).'/'.$view.'.html';
            $contents = file_get_contents($file);
        if(file_exists($file)){
            if(count($params)>0){
                foreach($params as $var => $value){
                    $contents = str_replace('{%'.$var.'%}',$value,$contents);
                }
            }
            return $contents;
        }else{
            throw new Exception("View path is invalid");
        }
        }else{
            throw new Exception("View file  not found");
        }
    }else{
        throw new Exception("Enter module and view options");
    }
```

```
            return false;
        }

    //>prepare("SELECT * FROM animals WHERE animal_id = :animal_id AND animal_name = :animal_name");
    //
    //    /*** bind the paramaters ***/
    //$stmt->bindParam(':animal_id', $animal_id, PDO::PARAM_INT);
    //$stmt->bindParam(':animal_name', $animal_name, PDO::PARAM_STR, 5);
    //
    //    /*** execute the prepared statement ***/
    //$stmt->execute();
    //
    //    /*** fetch the results ***/
    //$result = $stmt->fetchAll();

        public function find($q = null,$params = array()){
            return false;
        }

        public function htmlSelect($options,$data,$selected = null){
            $select = "<select ";
            foreach($options as $attr => $value){
                $select.=" ".$attr."=\"".$value."\"";
            }
            $select.=">";
            foreach ($data as $value=>$title){
                $select.="<option value=\"".$value."' ".($selected!==null && $selected==$value ? 'selected' : "").">".ucfirst($title)."</option>";
```

```
        }
        $select.=’</select>’;

        return $select;
    }

    public function __destruct(){
        $this->db = null;
    }

    public static function translitString($string) {
$tr = array(
            "А"=>"A","Б"=>"B","В"=>"V","Г"=>"G",
            "Д"=>"D","Е"=>"E","Ё"=>"E","Ж"=>"J","З"=>"Z","И"=>"I",
            "Й"=>"Y","К"=>"K","Л"=>"L","М"=>"M","Н"=>"N",
            "О"=>"O","П"=>"P","Р"=>"R","С"=>"S","Т"=>"T",
            "У"=>"U","Ф"=>"F","Х"=>"H","Ц"=>"TS","Ч"=>"CH",
            "Ш"=>"SH","Щ"=>"SCH","Ъ"=>"","Ы"=>"Y","Ь"=>"",
            "Э"=>"E","Ю"=>"YU","Я"=>"YA","а"=>"a","б"=>"b",
            "в"=>"v","г"=>"g","д"=>"d","е"=>"e","ё"=>"e","ж"=>"j",
            "з"=>"z","и"=>"i","й"=>"y","к"=>"k","л"=>"l",
            "м"=>"m","н"=>"n","о"=>"o","п"=>"p","р"=>"r",
            "с"=>"s","т"=>"t","у"=>"u","ф"=>"f","х"=>"h",
            "ц"=>"ts","ч"=>"ch","ш"=>"sh","щ"=>"sch","ъ"=>"y",
            "ы"=>"y","ь"=>"","э"=>"e","ю"=>"yu","я"=>"ya"
);
        return strtr($string, $tr);
    }

    public static function makeSlug($string) {
```

```
            $string = strip_tags($string);
            iconv("UTF-8", "CP1251", $string);
            $string = self::translitString($string);
            iconv("CP1251", "UTF-8", $string);

            $string = str_replace(array(”’, ”’, "&", "%", ",", "."), ’-’, $string);
            $string = preg_replace(’‘&(amp;)?#?[a-z0-9]+;‘i’, ’-’, $string);
            $string = htmlentities($string, ENT_COMPAT, ’UTF-8’);
            $string = preg_replace( "‘&([a-z])(acute|uml|circ|grave|ring|cedil|slash|tilde|caron|lig|quot|lsquo|rsquo|laquo|raquo);‘i", "\\1", $string );
            $string = preg_replace( array("‘[^a-z0-9]‘i","‘[-]+‘") , "-", $string);
            $string = preg_replace( array("/nbsp/", "/amp/", "/laquo/", "/raquo/", "/lsquo/", "/rsquo/"), "", $string );
            $string = preg_replace( "/--/", "-", $string );
            return mb_substr(strtolower(trim($string, ’-’)), 0, 128, ’UTF-8’);
        }

        function getEducationTypes($types = array()){
            $q = $this->db->prepare("SELECT et.id,et.order,et.title FROM sed_education_types as et ORDER by et.order ASC");
            $q->execute();
            foreach($q->fetchAll(PDO::FETCH_ASSOC) as $row){
               $types[$row[’id’]] = $row[’title’];
            }

            return $types;
        }

        function getSpecialists($types = array()){
```

```
        $q = $this->db->prepare("SELECT es.id,es.order,es.title FROM sed_education_specialist as es ORDER by es.order ASC ");
        $q->execute(array('lang' => $this->lang));
        foreach($q->fetchAll(PDO::FETCH_ASSOC) as $row){
            $types[$row['id']] = $row['title'];
        }

        return $types;
    }

    function getAcademicRanks($types = array()){
        $q = $this->db->prepare("SELECT ar.id,ar.order,ar.title FROM sed_academic_ranks as ar ORDER by ar.order ASC ");
        $q->execute(array('lang' => $this->lang));
        foreach($q->fetchAll(PDO::FETCH_ASSOC) as $row){
            $types[$row['id']] = $row['title'];
        }

        return $types;
    }

    function getGraduationTypes($types = array()){
        $q = $this->db->prepare("SELECT * FROM sed_graduation_types as gt order by gt.order ASC ");
        $q->execute();
        foreach($q->fetchAll(PDO::FETCH_ASSOC) as $row){
            $types[$row['id']] = $row['title'];
        }

        return $types;
```

```
    }

    public function flushView($type,$path = null,$params = array()){
        if($path!==null && $type!==null){
            $mpath = explode('/',$path);
            if(count($mpath)==2 && in_array(strtolower($type),$this->viewTypes)){
                $module =& $mpath[0];
                $view =& $mpath[1];
                $file = $this->rootPath.'/modules/'.$module.'/View/'.ucfirst($type).'/'.$view.'.phtml';
                if(file_exists($file)){
                    if(count($params)>0){
                        extract($params);
                        ///@print($user_action);exit;
                    }
                    ob_start();
                    require($file);
                    return ob_get_clean();
                }else{
                    throw new Exception("View path is invalid");
                }
            }else{
                throw new Exception("View file  not found");
            }
        }else{
            throw new Exception("Enter module and view options");
        }
        return false;
    }
```

```
        public function generateTree($obj = array(),$nodeMarkup = "<li>{%NODE%}</li>",$bundleMarkup = "<ul>{%BUNDLE%}</ul>",$tree = ”){
            if(is_array($obj) && count($obj)>0){
                foreach($obj as $title => $children){
                    $str = $nodeMarkup;
                    $str = str_replace(’{%ID%}’,$children[’id’],$str);
                    $str = str_replace(’{%PARENT%}’,$children[’parent’],$str);
                    $str = str_replace(’{%TITLE%}’,$title,$str);
                    $tree.= "<li>".$str;
                    if(is_array($children[’children’]) && count($children[’children’])>0){
                        $tree.= "<ul>".$this->generateTree($children[’children’],$nodeMarkup)."</ul>";
                    }else{
                        $tree.="<ul></ul>";
                    }
                    $tree.="</li>";
                }
            }

            return $tree;

        }

        public function checkUser($user = array()){
            if(isset($user[’f’]) && isset($user[’l’]) && isset($user[’m’])){
                $q = $this->db->prepare("SELECT per_id,name,surename,middle_name FROM sed_personal where
```

```
LOWER(name)=:first and LOWER(surename)=:last and
LOWER(middle_name)=:middle limit 1");
                $q->bindParam(':first',$user['f']);
                $q->bindParam(':last',$user['l']);
                $q->bindParam(':middle',$user['m']);
                $q->execute();
                if($q->rowCount()>0){
                    $row = $q->fetch(PDO::FETCH_ASSOC);
                    if(isset($_SESSION['current_person'])){
                        unset($_SESSION['current_person']);
                    }
                    $_SESSION['current_person'] = $row['per_id'];
                    $this->person_id = $row['per_id'];
                    $this->person = array(
                        'first' => $row['name'],
                        'last' => $row['surename'],
                        'middle' => $row['middle_name']
                    );
                    return true;
                }
            }

            return false;
        }

        public function dobUnDigitize($dob = null,$del = '-',$reverse = true){
            if($dob!==null){
                $dob = str_split($dob);
                if(count($dob)==8){
                    if($reverse){
```

```
                    $dob =
"{$dob[0]}{$dob[1]}{$dob[2]}{$dob[3]}{$del}{$dob[4]}{$dob[5]}{$del}{$dob[6
]}{$dob[7]}";
                }else{
                    $dob =
"{$dob[4]}{$dob[5]}{$dob[6]}{$dob[7]}{$del}{$dob[2]}{$dob[3]}{$del}{$dob[0
]}{$dob[1]}";
                }

                return $dob;
            }
        }
        return $dob;
    }

    public function whoIs($user = null){

        if($user!==null){
            $userData = $this->db->prepare("SELECT
                user.login,
                user.password,
                user.per_id,
                user.name,
                user.surename,
                user.middle_name,
                user.personal_picture,
                user.status
                FROM sed_personal as user
                WHERE user.per_id=:uid LIMIT 1");
            $userData->execute(array('uid' => $user));
```

```
        $userData = $userData->fetch(PDO::FETCH_ASSOC);

        if(count($userData)>0){
            if(isset($_SESSION['current_person'])){
                unset($_SESSION['current_person']);
            }
            $_SESSION['current_person'] = $user;

            $this->person_id = $userData['per_id'];
            $this->person = array(
                'per_id' => $userData['per_id'],
                'first' => $userData['name'],
                'last' => $userData['surename'],
                'middle' => $userData['middle_name'],
                'personal_picture' => $userData['personal_picture'],
                'login' => $userData['login'],
                'password' => $userData['password'],
                'status' => $userData['status']
            );
            return true;
        }

    }

    return false;
}


public function scanModules(){
    $modules = scandir($this->rootPath.'/modules');
```

```
            foreach($modules as $module){
                if(preg_match('/^[A-Za-z0-9]+$/i',$module)){
                    $module = strtoupper($module);
                    $sortMax = $this->db->prepare("SELECT MAX(sort) FROM sed_modules");
                    $sortMax->execute();
                    $sortMax = $sortMax->fetchColumn(0);
                    $sortMax = $sortMax==0 ? 1 : $sortMax;
                    $q = $this->db->prepare("
                        INSERT INTO sed_modules(code,title,sort,created)
                        SELECT * FROM (SELECT :code,:title,:sort,:created) AS modules
                        WHERE NOT EXISTS (
                          SELECT * FROM sed_modules WHERE code=:code) LIMIT 1");
                    $q->execute(array('code' => $module,'title' => $module."_title",'sort' => $sortMax,'created' => date('Y-m-d H:i:s'))) or die("Mysql error on module insertion");
                }
            }
            $modules = $this->db->prepare("SELECT * FROM sed_modules");
            $modules->execute();
            $modules = $modules->fetchAll(PDO::FETCH_ASSOC);
            foreach($modules as $module){
                if(!is_dir($this->rootPath.'/modules/'.strtolower($module['code']))){
                    $removeModule = $this->db->prepare("DELETE FROM sed_modules WHERE id=:mid");
                    $removeModule->execute(array('mid' => $module['id']));
                }
            }
```

```
        }

        public function getTerm($term = false,$index = false){

            if($term){
                if($index!==false){
                    if(array_key_exists($index,$this->config['terms'][$term])){
                        return $this->config['terms'][$term][$index];
                    }
                }else{
                    if(array_key_exists($term,$this->config['terms'])){
                        return $this->config['terms'][$term];
                    }
                }
            }

            return false;
        }

        public function isUserGroup(){
            $g = $this->db->prepare("SELECT * FROM sed_groups as sg WHERE
LOWER(sg.group) = 'user' LIMIT 1");
            $g->execute();
            if($g->rowCount()>0){
                $g = $g->fetch(PDO::FETCH_ASSOC);
                $g = $g['id'];
            }else{
                $g = $this->db->prepare("INSERT INTO
sed_groups(`group`,`title`,`active`,`created`) values(:code,:title,'1',:now)");
```

```
            $g->execute(array('code' => 'user','title' => 'Пользователь','now' =>
date('Y-m-d H:i:s')));
            $g  = $g->lastInsertId();
        }
        return $g;
    }


    public function deny($msg = ''){
/*      if(extension_loaded('curl')){
            $curl = curl_init('access_error.php');
            curl_setopt($curl, CURLOPT_POSTFIELDS, "msg=".$msg);
            curl_setopt($curl, CURLOPT_FOLLOWLOCATION, true);
            curl_setopt($curl, CURLOPT_CUSTOMREQUEST, "POST");
            curl_exec($curl);
        }else{*/
            header("Location: access_error.php?msg=".urlencode($msg));
//      }
    }




    public function hasPermission($module = false){
        $hasAccess = false;
        if($module && is_string($module) && $this->visitor['uid']!==null){
            $m = $this->db->prepare("SELECT * FROM sed_modules WHERE
LOWER(code)=:m");
            $m->execute(array('m' => strtolower($module)));
```

```
                if($m->rowCount()>0){
                    $m = $m->fetch(PDO::FETCH_ASSOC);
                    if(is_dir($this->rootPath.'/modules/'.strtolower($m['code']))){
                        $userGroups = $this->db->prepare("SELECT * FROM sed_groups_users WHERE user_id=:uid");
                        $userGroups->execute(array('uid' => $this->visitor['uid']));
                        $userGroups = $userGroups->fetchAll(PDO::FETCH_ASSOC);
                        foreach($userGroups as $group){
                            $md = $this->db->prepare("SELECT * FROM sed_groups_modules WHERE module_id=:mid and group_id=:gid");
                            $md->execute(array('gid' => $group['group_id'], 'mid' => $m['id']));
                            if($md->rowCount()>0){
                                $hasAccess = true;
                            }
                        }
                    }
                }
            }

            return $hasAccess;
        }

        public function belongsToGroup($group = null){
            if($group!==null && is_string($group)){
                $q = $this->db->prepare("SELECT sr.id FROM sed_groups as sr WHERE UPPER(sr.group)=:group limit 1");
                $q->execute(array('group' => strtoupper($group)));
                if($q->rowCount()>0){
                    $q = $q->fetch(PDO::FETCH_ASSOC);
```

```
                $u = $this->db->prepare("SELECT count(*) FROM
sed_groups_users where user_id=:uid and group_id=:g");
                $u->execute(array('uid'=>$this->visitor['uid'],'g'=>$q['id']));
                if($u->fetchColumn(0)>0){
                    return true;
                }
            }

        }
        return false;
    }


    public function isAdminDefined(){
        $m = $this->moduleExists('admin');
        if($m){
            $q = $this->db->prepare("SELECT g.group FROM sed_groups as g
WHERE UPPER(g.group)=:g LIMIT 1");
            $q->execute(array('g' => 'ADMIN'));
            if($q->rowCount()==0){
                $q = $this->db->prepare("INSERT INTO
sed_groups(`group`,`title`,`active`,`created`) values(:code,:title,:active,:created)");
                $q->execute(array(
                    'code' => 'ADMIN',
                    'title' => 'Администатор',
                    'active' => 1,
                    'created' => date('Y-m-d H:i:s')
                ));
                $role_id = $this->db->lastInsertId('id');
```

```
                $md = $this->db->prepare("INSERT into
sed_groups_modules(group_id,module_id) values(:g,:m)");
                $md->execute(array(
                    'g' => $role_id,
                    'm' => $m
                ));
                $s = $this->db->prepare("SELECT count(*) FROM sed_personal
where LOWER(login)='admin' LIMIT 1");
                $s->execute();
                if($s->fetchColumn(0)==0){
                    $s = $this->db->prepare("INSERT INTO
sed_personal(login,password,name,surename,middle_name)
values('admin',md5('adminpass'),'Господин','Админ','Администрович')");
                    $s->execute();
                    $uid = $this->db->lastInsertId();
                    $r = $this->db->prepare("INSERT INTO
sed_groups_users(user_id,group_id) values(:uid,:role)");
                    $r->execute(array('uid' => $uid,'role' => $role_id));
                }
            }
            return true;
        }

        return false;

    }

    public function moduleExists($module = false){
        if($module){
            $module = strtoupper($module);
```

```
            $q = $this->db->prepare("SELECT id FROM sed_modules WHERE
UPPER(code)=:module limit 1");
            $q->execute(array('module' => $module));
            $path  = $this->rootPath.'/modules/'.strtolower($module);
            if(is_dir($path)){
               if($q->rowCount()==0){
                  $q          =          $this->db->prepare("INSERT          INTO
sed_modules(active,code,title) values('1',:code,:title)");
                  $q->execute(array('code' => $module,'title' => $module));
                  $id = $this->db->lastInsertId('id');
               }else{
                  $q = $q->fetch(PDO::FETCH_ASSOC);
                  $id = $q['id'];
               }
               return $id;
            }
         }

         return false;
      }


      public function giveQuotes($str){
         return "'".$str."'";
      }

      public function referedNames($name = false){
         $endings = array(
            'ва' => 'ой',
            'ов' => 'ова',
```

```
'на' => 'ной',
            'ин' => 'ина',
            'ич' => 'ича',
'вна' => 'вны'
        );
        if($name){
            if(in_array(substr($name,-2),$endings)){
                return substr($name,1,strlen($name)-2).$endings[substr($name,-2)];
            }
        }

        return trim($name);
    }

    public function makeSqlFiendly($string = false){
        if($string && is_string($string)){
            return $string = addslashes($string);
        }
        return false;
    }

    public function makeHtmlFriendly($string = false){
        if($string && is_string($string)){
            return $string = stripslashes($string);
        }
        return false;
    }

    public function getFileExtension($name = false){
        if($name && is_string($name)){
```

```
            $parse = explode('.', $name);
            return end($parse);
        }
        return false;
    }

    public function extractPattern($pattern = false){
        if($pattern && is_string($pattern)){
            $pattern = array_filter(explode(';',$pattern));
            $pattern = array_map(function($v){
                return preg_replace('/#_/i','',$v);
            },$pattern);
            return $pattern;
        }
        return $pattern;
    }

    public function removeUserJunk($user = false){
        if($user and is_string($user)){
            $user = preg_replace('/#_/i','',$user);
            $user = preg_replace('/;/i','',$user);
            return $user;
        }
        return false;
    }

    function pager($page = 1, $total = 0, $params = array(),$hash = "",$per = 10, $visible = 3){
        $pages = (int) abs(ceil($total/$per));
        $prev = $page>1 ? $page-1 : null;
```

```
$next = $page<$total ? $page+1 : null;
/*  $vpages_start = $pages>$visible ? ($page>ceil($visible/2) ? $page-ceil($visible/2) : 1) : 1;
$vpages_end = $pages<=$visible ? $pages : ($page<$pages-ceil($visible/2) ? ($page<$visible-ceil($visible/2) ? $visible : $page+ceil($visible/2)) : $pages);*/
$vpages_start = $pages>$visible ? ($page>ceil($visible/2) ? $page-ceil($visible/2) : 1) : 1;
$vpages_end = $pages<=$visible ? $pages : ($page<$pages-ceil($visible/2) ? ($page<$visible-ceil($visible/2) ? $visible : $page+ceil($visible/2)) : $pages);
$start = $total>$per ? ($page*$per)-$per : 0;
$from = $start==0 ? 1 : $start+1;
$to =  $total < $start+$per ? $start+$total-$start : $start+$per;
$reverse_from = $total-$start;
//  $reverse_to = $total > $start+$per ? $start+$total-$start : $start+$per;
$pageline = ”;

if($page>1):
  $pageline.=’<li><a href="#’.$hash.’/page-1{%type%}{%records%}{%search%}">&laquo;</a></li>’;
  $pageline.=’<li><a href="#’.$hash.’/page-’.($page-1).’{%type%}{%records%}{%search%}">&larr;</a></li>’;
endif;

if($pages>1){
  for($p = $vpages_start; $p<=$vpages_end; $p++):
    $pageline.=’<li ’.($p==$page ? ’class="active"’ : null).’><a href="#’.$hash.’/page-’.$p.’{%type%}{%records%}{%search%}">’.$p.’</a></li>’;
  endfor;
```

```
        }

        if($page<$pages):
            $pageline.='<li><a href="#'.$hash.'/page-'.($page+1).'{%type%}{%records%}{%search%}">&rarr;</a></li>';
            $pageline.='<li><a href="#'.$hash.'/page-'.$pages.'{%type%}{%records%}{%search%}" data-toggle="tooltip" data-placement="bottom" data-original-title="вконец">&raquo;</a></li>';
        endif;

        if($pages>$visible){
            $pageline.='<li><input type="number" id="page" placeholder="Стр." minlength="1" maxlength="'.$pages."'></li>';
        }

        if(count($params)>0){
            foreach($params as $key => $value){
                $pageline = str_replace("{%".$key."%}","/".$key."-".$value,$pageline);
            }
        }

        $pageline = str_replace('{%search%}','',$pageline);
        $pageline = str_replace('{%records%}','',$pageline);
        $pageline = str_replace('{%type%}','',$pageline);

        return array(
            'per_page' => $per,
            'total_results' => $total,
            'pageline' => $pageline,
```

```
            'total_pages' => $pages,
            'current_page' => $page,
            'start' => $start,
            'end' => $per,
            'results_from' => $from,
            'results_to' => $to,
            'reverse_from' => $reverse_from
        );
    }


    public function getDocCount($type = false,$result = array('all' => 0,'new' => 0)){
        if($type && is_string($type)){
            switch($type){
                case 'incoming':
                    // $sql = "SELECT * FROM sed_incom_letters as incoms";
                    //$sql.= " LEFT JOIN sed_orders as orders ON(orders.link_doc_id = incoms.il_id and orders.link_doc_type='incoming') WHERE (incoms.il_id IS NOT NULL)".($this->belongsToGroup("SED_EDIT") ? "" : " AND (incoms.control_of_uid IS NOT NULL and incoms.responsible_uid IS NOT NULL and incoms.performers_uid IS NOT NULL and incoms.destination_uid IS NOT NULL) AND (incoms.control_of_uid LIKE '%".pattern($this->visitor['uid'])."%' or incoms.destination_uid LIKE '%".pattern($this->visitor['uid'])."%' or incoms.responsible_uid LIKE '%".pattern($this->visitor['uid'])."%' or orders.u_id = ''.$this->visitor['uid']."')");
                    // $sql = "SELECT * FROM sed_orders as orders LEFT JOIN sed_incom_letters as incoms ON(incoms.il_id = orders.link_doc_id and orders.link_doc_type='incoming') WHERE (incoms.control_of_uid IS NOT NULL and incoms.responsible_uid IS NOT NULL and incoms.performers_uid IS NOT
```

```
NULL and incoms.destination_uid IS NOT NULL)".($this->belongsToGroup("SED_EDIT") || $this->visitor['rank']=='1' ? "" : " AND (orders.u_id = '".$utility->visitor['uid']."' or incoms.control_of_uid LIKE '%#_".pattern($this->visitor['uid'])."';%' or incoms.destination_uid LIKE '%#_".pattern($this->visitor['uid'])."';%' or incoms.responsible_uid LIKE '%#_".pattern($this->visitor['uid'])."';%')");
                    // $sql." AND (orders.view IS NULL and orders.close_date IS NULL) GROUP BY incoms.il_id";
                    //exit;
                    $result['all'] = 7400;//$this->db->query($sql." GROUP BY incoms.il_id")->rowCount();
                    $result['new'] = 330;//$this->db->query($sql." AND (orders.view IS NULL and orders.close_date IS NULL) GROUP BY incoms.il_id")->rowCount();
                    break;
                case 'outgoing':
                    $sql = "SELECT * FROM sed_outgoing_letters as outgoing WHERE outgoing.ol_d IS NOT NULL".($this->belongsToGroup('SED_EDIT') || $this->visitor['rank']=='1' ? '' : " AND (chief_id = '".$this->visitor['uid']."' or performer_id LIKE '%".pattern($this->visitor['uid'])."%')");
                    $result['all'] = $this->db->query($sql)->rowCount();
                    break;
                case 'memos':
                    if($this->visitor['rank']=='1' || $this->visitor['rank']=='2'){
                        $q = $this->db->prepare("SELECT count(*) FROM sed_memos WHERE whom = :uid");
                        $q->execute(array('uid' => $this->visitor['uid']));
                        $result['all'] = $q->fetchColumn(0);
                        $n = $this->db->prepare("SELECT count(*) FROM sed_memos WHERE whom = :uid and deadline IS NOT NULL");
```

```
                $n->execute(array('uid' => $this->visitor['uid']));
                $result['new'] = $n->fetchColumn(0);
            }elseif($this->belongsToGroup('SED_EDIT')){
                $q = $this->db->prepare("SELECT count(*) FROM sed_memos");
                $q->execute();
                $result['all'] = $q->fetchColumn(0);
                $n = $this->db->prepare("SELECT count(*) FROM sed_memos WHERE deadline IS NOT NULL");
                $n->execute();
                $result['new'] = $n->fetchColumn(0);
            }else{
                $q = $this->db->prepare("SELECT count(*) FROM sed_memos as memo WHERE memo.where LIKE '%#_:uid;%'");
                $q->execute(array('uid' => $this->visitor['uid']));
                $result['all'] = $q->fetchColumn(0);
                $n = $this->db->prepare("SELECT count(*) FROM sed_memos as memo WHERE memo.where LIKE '%#_:uid%' and deadline IS NOT NULL");
                $n->execute(array('uid' => $this->visitor['uid']));
                $result['new'] = $n->fetchColumn(0);
            }
            break;

        case 'orders':
            $sql = "SELECT
             *,orders.deadline as order_deadline
             FROM sed_orders as orders
             LEFT JOIN sed_incom_letters as incoming ON(incoming.il_id = orders.link_doc_id and orders.link_doc_type='incoming')
```

```
                    LEFT JOIN sed_memos as memos ON(memos.id = orders.link_doc_id and orders.link_doc_type = 'memo')
                    WHERE or_id IS NOT NULL".(
                    $this->belongsToGroup("SED_EDIT") || $this->visitor['rank']=='1' ? "" :
                    " AND (orders.u_id='".$this->visitor['uid']."' or orders.reorder = '".$utility->visitor['uid']."' OR incoming.performers_uid LIKE '%".$this->visitor['uid']."%'
                    OR incoming.control_of_uid LIKE '%".$this->visitor['uid']."%' OR incoming.responsible_uid LIKE '%".$this->visitor['uid']."%'
                    OR memos.executors LIKE '%".$this->visitor['uid']."%' OR memos.where LIKE '%".$this->visitor['uid']."%'
                    OR memos.user_id = '".$this->visitor['uid']."') ");
                    $result['all'] = $this->db->query($sql." GROUP BY orders.link_doc_id")->rowCount();
                    $result['new'] = $this->db->query($sql." AND (orders.close_date IS NOT NULL and orders.view IS NULL)")->rowCount();
                    break;
                case 'decree':
                    $sql = "SELECT * FROM sed_decree";
                    $result['all'] = $this->db->query($sql)->rowCount();
                    break;
            }

            return $result;
        }

        return 0;
    }
```

```
        public function timeWarner($time = false){
            if($time){

            }
            return $time;
        }
        public function getStaffTree($dep = false){
            $outcome = array();
            $sql = "SELECT department_name,did FROM sed_department_list WHERE 1=1";
            if($dep){
                $sql.=" AND (did = '".$dep."')";
            }
            $sql.=" order by did asc";
            $deps = $this->db->query($sql)->fetchAll(PDO::FETCH_ASSOC);
            foreach($deps as $dep){
                $outcome[$dep['department_name']] = $this->db->query("SELECT person.surename,person.name,person.middle_name,person.per_id FROM sed_personal as person
                LEFT JOIN sed_positions as pos ON((person.temp_pos!='' and person.temp_pos=pos.id) or (person.temp_pos='' and person.position=pos.id))
                WHERE (person.depar_id = '".$dep['did']."' or person.temp_dep = '".$dep['did']."')
                and person.status!=0 and person.status!='0' order by pos.rank asc")->fetchAll(PDO::FETCH_ASSOC);
            }
            return $outcome;
        }

        public function makeFullname($l,$f,$m){
```

```
            return
ucfirst(trim($l))." ".ucfirst(mb_substr(trim($f),0,2)).".".ucfirst(mb_substr(tri
m($m),0,2));
        }

        public function getUsersOnline(){
            $content = "";
            $online = $this->db->query("SELECT * FROM sed_personal as person
            LEFT JOIN sed_modules as modules ON(person.current_module =
modules.id)
            WHERE person.per_id!='".$this->visitor['uid']."' and person.logged >
'".(time()-60)."' ")->fetchAll(PDO::FETCH_ASSOC);
            foreach($online as $user){
                $content.= "<li class='show-chat-window' id='chu-
".$user['per_id']."'><i class='fa fa-user'></i>\t".$this-
>makeFullname($user['surename'],$user['name'],$user['middle_name'])." -
        <small><i class='glyphicon glyphicon-
time'></i>\t".date('H:i',strtotime($user['visit']))."</small><br>
            ".(!empty($user['title']) ? "<small class='label label-
default'><small>".$user['title']."</small></small>" : "")."</li>";
            }

            return array(
                'count' => count($online),
                'list' => $content
            );
        }
    public function getUser(){
    if(isset($_SESSION["username"])){
```

```
        $u=$this->db->query("Select per_id from sed_personal where login='".$_SESSION["username"]."' limit 1");
        $u=$u->fetchAll(PDO::FETCH_ASSOC);
        if($u)return $u; else return false;
        } else return false;
        }


        ?>
```