Главное управление по образованию

Минского областного исполнительного комитета

Учреждение образования

«Новопольский государственный аграрно-экономический колледж»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ**

**по учебной дисциплине «Базы данных и системы управления базами данных»**

**на тему «Программное средство для учета работы больницы»**

Учащегося группы 41ПО И.Д. Недень

Руководитель А.А. Чивель

д. Новое Поле 2020

**СОДЕРЖАНИЕ**

Изм.

Лист

№ докум.

Подпись

Дата

Лист

2

**КППО41.18.102.081ПЗ**

Разраб.

Недень И.Д.

Провер.

Чивель А.А.

Реценз.

Н. Контр.

Утверд.

.

Программное средство для учета работы больницы»

Лит.

Листов

30

НГАЭК, 2020

[ВВЕДЕНИЕ 3](#_Toc26972351)

[ОСНОВНАЯ ЧАСТЬ 5](#_Toc26972352)

[1 Назначение и область применения 5](#_Toc26972353)

[1.1 Характеристика предметной области 5](#_Toc26972354)

[1.2 Назначение программного средства 12](#_Toc26972355)

[1.3 Описание аналогов 12](#_Toc26972356)

[2 Технические характеристики 14](#_Toc26972357)

[2.1 Постановка задачи на разработку Программного средства 14](#_Toc26972358)

[2.2 Описание Программного средства 14](#_Toc26972359)

[2.2.1 Общие сведения 14](#_Toc26972360)

[2.2.2 Функциональное назначение 17](#_Toc26972361)

[2.2.3 Описание логической структуры 17](#_Toc26972362)

[2.2.4 Используемые технические средства 22](#_Toc26972363)

[2.2.5 Вызов и загрузка 22](#_Toc26972364)

[2.2.6 Входные данные 23](#_Toc26972365)

[2.2.7 Выходные данные 23](#_Toc26972366)

[2.2.8 Рекомендации по использованию 23](#_Toc26972367)

[ЗАКЛЮЧЕНИЕ 24](#_Toc26972368)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 25](#_Toc26972369)

ПРИЛОЖЕНИЯ

# **ВВЕДЕНИЕ**

Технологии баз данных увели Интернет-приложения далеко от простых брошюрных публикаций, которые характеризовали ранние приложения. В то же время Интернет-технология обеспечивает пользователям стандартизированные и доступные средства публикации содержимого баз данных. Однако в последние годы всплеск популярности Интернета и бурное развитие новых технологий для Интернета сделали знание технологии баз данных для многих одним из актуальнейших путей карьеры. Правда, ни одна из этих новых разработок не отменяет необходимости в классических приложениях баз данных, которые появились еще до развития Интернета для нужд бизнеса. Это только расширяет важность знания баз данных. Многие студенты считают этот предмет приятным и интересным, даже несмотря на его сложность. Проектирование и разработка базы данных требуют и искусства, и умения. Понимание пользовательских требований и перевод их в эффективный проект базы данных можно назвать творческим процессом. Преобразование этих проектов в физические базы данных с помощью функционально полных и высокопроизводительных приложений – инженерный процесс. Оба процесса полны сложностей и приятных интеллектуальных головоломок.

Основная задача программного средства – предоставить пользователю базы данных возможность работать с ней, не вникая в детали на уровне аппаратного обеспечения (пользователь более отстранен от этих деталей, чем прикладной программист, использующий среду программирования). Иными словами, программное средство позволяет конечному пользователю рассматривать базу данных как объект более высокого уровня по сравнению с аппаратным обеспечением, а также предоставляет в его распоряжение набор операций, выражаемых в терминах языка высокого уровня.

Объектом исследования является деятельность больницы.

Предметом исследования является программное средство для учета работы больницы.

Целью работы является разработка программы «Управление данными больницы».

В рамках работы были поставлены следующие задачи:

* анализ предметной области программы «Управление реестром книг»;
* проектирование базы данных в исследуемой предметной области.

Назначение проектируемой базы:

* хранение информации о пациентах;
* хранение информации о врачах;
* хранение информации о медикаментах;
* хранение информации о вызовах врачей.

# **ОСНОВНАЯ ЧАСТЬ**

# **Назначение и область применения**

# **1.1 Характеристика предметной области**

База данных — представленная в объективной форме совокупность самостоятельных материалов, систематизированных таким образом, чтобы эти материалы могли быть найдены и обработаны с помощью электронной вычислительной машины. Для создания и ведения БД (обновления, обеспечение доступа к ним по запросам и выдачи их пользователю) используется набор языковых и программных средств, называемой системой управления базами данных (СУБД). Такие средства обычно включаются в состав СУБД, но они могут существовать и отдельно. СУБД дают возможность пользователям осуществлять непосредственное управление данными, а программистам быстро разрабатывать более совершенные программные средства их обработки.

Система программ, позволяющая создавать БД, обновлять хранимую в ней информацию, обеспечивающая удобный доступ к ней с целью просмотра и поиска, называется СУБД.

Типы баз данных:

По способу установления связей между данными различают:

– реляционную;

– иерархическую;

– сетевую модели.

Реляционная модель является простейшей и наиболее привычной формой представления данных в виде таблицы. В теории множеств таблице соответствует термин отношение (relation), который и дал название модели. Для нее имеется развитый математический аппарат – реляционное исчисление и реляционная алгебра, где для баз данных (отношений) определены такие хорошо известные теоретико-множественные операции, как объединение, вычитание, пересечение, соединение и др.

Достоинством реляционной модели является сравнительная простота инструментальных средств ее поддержки, недостатком – жесткость структуры данных (невозможность, например, задания строк таблицы произвольной длины) и зависимость скорости ее работы от размера базы данных. Для многих операций, определенных в такой модели, может оказаться необходимым просмотр всей базы.

Иерархическая и сетевая модели предполагают наличие связей между данными, имеющими какой-либо общий признак. В иерархической модели такие связи могут быть отражены в виде дерева-графа, где возможны только односторонние связи от старших вершин к младшим. Это облегчает доступ к необходимой информации, но только если все возможные запросы отражены в структуре дерева. Никакие иные запросы удовлетворены быть не могут.

Указанный недостаток снят в сетевой модели, где, по крайней мере теоретически, возможны связи «всех со всеми». Поскольку на практике это, естественно, невозможно, приходится прибегать к некоторым ограничениям. Использование иерархической и сетевой моделей ускоряет доступ к информации в базе данных. Но поскольку каждый элемент данных должен содержать ссылки на некоторые другие элементы, требуются значительные ресурсы как дисковой, так и основной памяти ЭВМ. Недостаток основной памяти, конечно, снижает скорость обработки данных. Кроме того, для таких моделей характерна сложность реализации СУБД.

Хотя известны попытки создания СУБД, поддерживающих сетевую модель для персональных компьютеров, в настоящее время реляционные системы лучше соответствуют их техническим возможностям и вполне удовлетворяют большинство пользователей. Скоростные характеристики этих СУБД поддерживаются специальными средствами ускоренного доступа к информации индексированием баз данных.

СУБД (система управления базами данных) – это программа, которая управляет данными, осуществляет хранение, извлечение, поиск, редактирование информации, хранимой в базе данных.

Таблица – это набор записей (строк), состоящих из отдельных полей (столбцов), в которых хранится информация и составляющая, собственно, содержание базы данных.

Таблицы являются основной формой представления информации, содержащейся в базе данных. Без таблицы нельзя спроектировать формуляр, на базе таблиц составляются запросы и отчеты.

Таблицы баз данных, как правило, допускают работу с гораздо большим количеством разных типов данных. Так, например, базы данных Microsoft Access работают со следующими типами данных.

Текстовый – тип данных, используемый для хранения обычного неформатированного текста ограниченного размера (до 255 символов).

Числовой – тип данных для хранения действительных чисел.

Поле Мемо – специальный тип данных для хранения больших объемов текста (до 65 535 символов). Физически текст не хранится в поле. Он храниться в другом месте базы данных, а в поле храниться указатель на него, но для пользователя такое разделение заметно не всегда.

Дата/время – тип данных для хранения календарных дат и текущего времени.

Денежный – тип данных для хранения денежных сумм. Теоретически, для их записи можно было бы пользоваться и полями числового типа, но для денежных сумм есть некоторые особенности (например, связанные с правилами округления), которые делают более удобным использование специального типа данных, а не настройку числового типа.

Счетчик – специальный тип данных для уникальных (не повторяющихся в поле) натуральных чисел с автоматическим наращиванием. Естественное использование – для порядковой нумерации записей.

Логический – тип для хранения логических данных (могут принимать только два значения, например, да или нет).

Гиперссылка – специальное поле для хранения адресов URL Web-объектов Интернета. При щелчке на ссылке автоматически происходит запуск браузера и воспроизведение объекта в его окне.

Мастер подстановок – это не специальный тип данных. Это объект, настройкой которого можно автоматизировать ввод данных в поле так, чтобы не вводить их вручную, а выбирать их из раскрывающегося списка.

При размещении базы данных на персональный компьютер (ПК), который не находится в сети, база данных всегда используется в монопольном режиме. Даже если базу данных используют несколько пользователей, они могут работать с ней только последовательно. Однако работа на изолированном ПК с небольшой базой данных в настоящий момент становится уже не характерной для большинства приложений. База данных отражает информационную модель реального программного обеспечения (ПО), она растет по объему, следовательно, резко увеличивается количество задач, решаемых с помощью этой базы данных и в соответствии с этим увеличивается количество приложений, работающих с единой базой данных. ПК объединяются в локальные сети, и необходимость распределения приложений, работающих с единой базой данных по сети, является несомненной.

Параллельный доступ к одной базе данных нескольких пользователей, в том случае, если она расположена на одной машине, соответствует режиму распределенного доступа к центральной базе данных. Такие системы называются системами распределенной обработки данных.

Если же база данных расположена на нескольких ПК, распределенных в сети, и к ней возможен параллельный доступ нескольких пользователей, то мы имеем дело с параллельным доступом к распределенным базам данных. Такие системы называются системами распределенных баз данных.

В наиболее полном варианте СУБД должна содержать следующие составные части:

– среда пользователя, дающая возможность непосредственного управления данными с клавиатуры;

– алгоритмический язык для программирования прикладных систем обработки данных, реализованный как интерпретатор. Последнее позволяет быстро создавать и отлаживать программы;

– компилятор для придания завершенной программе вида готового коммерческого продукта в форме независимого ЕХЕ-файла;

– программы-утилиты быстрого программирования рутинных операций (генераторы отчетов, экранов, меню и других приложений).

Собственно говоря, СУБД – это оболочка пользователя. Ввиду того, что такая среда ориентирована на немедленное удовлетворение его запросов, это всегда система-интерпретатор.

Наличие в СУБД языка программирования позволяет создавать сложные системы обработки данных, ориентированные под конкретные задачи и даже под конкретного пользователя. Есть также СУБД, которые имеют только язык и не имеют оболочки пользователя. Они предназначены исключительно для программистов, и это системы компилирующего типа. Такие пакеты лишь с оговорками могут быть названы СУБД. Обычно их называют просто компиляторами.

Быстрое развитие потребностей применений БД выдвигает новые требования к СУБД:

– поддержка широкого спектра типов, представляемых данных и операций над ними;

– естественные и эффективные представления в БД разнообразных отношений между объектами предметных областей (например, пространственно-временных с обеспечением визуализации данных);

– поддержка непротиворечивости данных и реализация дедуктивных БД;

– обеспечение целостности БД в широком диапазоне разнообразных предметных областей и операционных обстановок;

– управление распределенными БД, интеграция неоднородных баз данных;

– существенное повышение надежности функционирования БД.

Вместе с тем традиционная программная реализация многочисленных функций современных СУБД на ЭВМ общего назначения приводит к громоздким и непроизводительным системам с недостаточно высокой надежностью. Тем более затруднительным оказывается наращивание программных средств, обеспечивающих перечисленные выше требования.

Это обусловлено рядом причин:

– фон-неймановская архитектура ЭВМ неадекватна требованиям СУБД, в частности реализации поиска, обновления, защиты данных, обработки транзакций только программным способом неэффективны как по производительности, так и по стоимости;

– многоуровневое и сложное программное обеспечение СУБД снижает эффективность и надежность функционирования БД;

– универсальная ЭВМ оказывается перегруженной функциями управлениями базами данных, что снижает эффективность функционирования собственно прикладных систем;

– централизация и интеграция данных в сетях персональных и профессиональных ЭВМ нереализуема с приемлемой стоимостью без включения в состав сетей, специализированных ЭВМ для поддержки функции СУБД.

Функции СУБД:

– определения данных – СУБД должна допускать определения данных (внешние схемы, концептуальную схему, внутреннюю схему, а также все связанные отображения) в исходной форме и преобразовывать эти определения в форму соответствующих объектов, т.е. СУБД должна включать в себя компонент языкового процессора для различных языков определения данных;

– обработка данных – СУБД должна уметь обрабатывать запросы пользователя на выборку, изменение или удаление соответствующих данных в базе данных или на добавление новых данных в базу данных, т.е. СУБД должна включать в себя компонент процессора языка обработки данных (ЯОД).

Запросы бывают «планируемые» и «не планируемые».

Планируемый запрос – это запрос, необходимость которого предусмотрена заранее. Администратор должен настроить физический проект базы данных таким образом, чтобы гарантировать достаточное быстродействие для таких запросов. Он характерен для «операционных приложений».

Не планируемый запрос – это специальный запрос, необходимость которого не была предусмотрена заранее. Он характерен для приложений «поддержки решений».

– безопасность и целостность данных – СУБД должна контролировать пользовательские запросы и пресекать попытки нарушения правил безопасности и целостности, определенных АБД.

Для пользователей информационной системы недостаточно, чтобы база данных просто отражала объекты реального мира. Важно, чтобы такое отражение было однозначным и непротиворечивым.

Для того чтобы гарантировать корректность и взаимную непротиворечивость данных, на базу данных накладываются некоторые ограничения, которые называют ограничениями целостности.

– восстановление данных и дублирование – восстановление и дублирование данных осуществляется СУБД или администратором;

– введение словаря данных – словарь данных – это исчерпывающий набор таблиц или файлов, представляющий собой каталог всех описаний данных. Также он может содержать информацию о пользователях, привилегиях и т.д., доступную только администратору базу данных. Словарь данных является центральным источником информации для СУБД, АБД всех пользователей;

– производительность – СУБД должна выполнять все указанные функции с максимально возможной эффективностью.

В целом назначением СУБД является предоставление пользовательского интерфейса с базой данных.

На рынке программного обеспечения можно выделить такие наиболее востребованные СУБД, как Мicrosoft Ассеss, МariaDB, Воrland Paradox, Огас1е, МуSQL, SQLite.

MySQL – [свободная](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%D0%BE%D0%B5_%D0%9F%D0%9E) [реляционная система управления базами данных](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%A1%D0%A3%D0%91%D0%94). Разработку и поддержку MySQL осуществляет корпорация [Oracle](https://ru.wikipedia.org/wiki/Oracle), получившая права на торговую марку вместе с поглощённой [Sun Microsystems](https://ru.wikipedia.org/wiki/Sun_Microsystems), которая ранее приобрела шведскую компанию [MySQL AB](https://ru.wikipedia.org/wiki/MySQL_AB). Продукт распространяется как под [GNU General Public License](https://ru.wikipedia.org/wiki/GNU_General_Public_License), так и под собственной коммерческой лицензией. Помимо этого, разработчики создают функциональность по заказу лицензионных пользователей. Именно благодаря такому заказу почти в самых ранних версиях появился механизм [репликации](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BF%D0%BB%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F_(%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%82%D0%B5%D1%85%D0%BD%D0%B8%D0%BA%D0%B0)).

MySQL является решением для малых и средних приложений. Входит в состав серверов [WAMP](https://ru.wikipedia.org/wiki/WAMP), [AppServ](https://ru.wikipedia.org/w/index.php?title=AppServ&action=edit&redlink=1), [LAMP](https://ru.wikipedia.org/wiki/LAMP) и в портативные сборки серверов [Денвер](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%BD%D0%B2%D0%B5%D1%80_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0)), [XAMPP](https://ru.wikipedia.org/wiki/XAMPP), [VertrigoServ](https://ru.wikipedia.org/w/index.php?title=VertrigoServ&action=edit&redlink=1). Обычно MySQL используется в качестве сервера, к которому обращаются локальные или удалённые клиенты, однако в дистрибутив входит библиотека внутреннего сервера, позволяющая включать MySQL в автономные программы.

Гибкость СУБД MySQL обеспечивается поддержкой большого количества типов таблиц: пользователи могут выбрать как таблицы типа [MyISAM](https://ru.wikipedia.org/wiki/MyISAM), поддерживающие полнотекстовый поиск, так и таблицы [InnoDB](https://ru.wikipedia.org/wiki/InnoDB), поддерживающие транзакции на уровне отдельных записей. Более того, СУБД MySQL поставляется со специальным типом таблиц EXAMPLE, демонстрирующим принципы создания новых типов таблиц. Благодаря открытой архитектуре и GPL-лицензированию, в СУБД MySQL постоянно появляются новые типы таблиц.

[Sun Microsystems](https://ru.wikipedia.org/wiki/Sun_Microsystems) приобрела [MySQL AB](https://ru.wikipedia.org/wiki/MySQL_AB) 26 февраля 2008 года за 1 млрд долларов, 27 января 2010 года [Oracle](https://ru.wikipedia.org/wiki/Oracle)приобрела [Sun Microsystems](https://ru.wikipedia.org/wiki/Sun_Microsystems) за 7,4 млрд долларов и включила MySQL в свою линейку [СУБД](https://ru.wikipedia.org/wiki/%D0%A1%D0%A3%D0%91%D0%94).

Сообществом разработчиков MySQL созданы различные ответвления кода, такие как [Drizzle](https://ru.wikipedia.org/w/index.php?title=Drizzle&action=edit&redlink=1), [OurDelta](https://ru.wikipedia.org/w/index.php?title=OurDelta&action=edit&redlink=1), [Percona Server](https://ru.wikipedia.org/w/index.php?title=Percona_Server&action=edit&redlink=1) и [MariaDB](https://ru.wikipedia.org/wiki/MariaDB). Все эти ответвления уже существовали на момент [поглощения](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D0%B8%D1%8F%D0%BD%D0%B8%D1%8F_%D0%B8_%D0%BF%D0%BE%D0%B3%D0%BB%D0%BE%D1%89%D0%B5%D0%BD%D0%B8%D1%8F) компании Sun корпорацией Oracle.

MySQL имеет [API](https://ru.wikipedia.org/wiki/API) и коннекторы для языков [Delphi](https://ru.wikipedia.org/wiki/Delphi_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)), [C](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)), [C++](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8_%D0%BF%D0%BB%D1%8E%D1%81_%D0%BF%D0%BB%D1%8E%D1%81), [Эйфель](https://ru.wikipedia.org/wiki/%D0%AD%D0%B9%D1%84%D0%B5%D0%BB%D1%8C_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)), [Java](https://ru.wikipedia.org/wiki/Java), [Perl](https://ru.wikipedia.org/wiki/Perl), [PHP](https://ru.wikipedia.org/wiki/PHP), [Python](https://ru.wikipedia.org/wiki/Python), [Ruby](https://ru.wikipedia.org/wiki/Ruby), [Smalltalk](https://ru.wikipedia.org/wiki/Smalltalk), [компонентный Паскаль](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%BE%D0%BD%D0%B5%D0%BD%D1%82%D0%BD%D1%8B%D0%B9_%D0%9F%D0%B0%D1%81%D0%BA%D0%B0%D0%BB%D1%8C) и [Tcl](https://ru.wikipedia.org/wiki/Tcl), библиотеки для языков платформы [.NET](https://ru.wikipedia.org/wiki/.NET_Framework), а также обеспечивает поддержку для [ODBC](https://ru.wikipedia.org/wiki/ODBC) посредством ODBC-драйвера [MyODBC](https://ru.wikipedia.org/wiki/MyODBC).

В данной курсовой работе я решил рассмотреть и организовать работу базы данных больницы, так как услуги учреждений здравоохранения являются актуальными ежедневно.

Предполагается, что база данных будет использоваться сотрудниками больницы, а также пациентами.

Организация рабочего процесса в больнице является довольно трудоемкой.

Больница — это организация, оказывающая услуги по лечению, оздоровлению и консультации пациентов

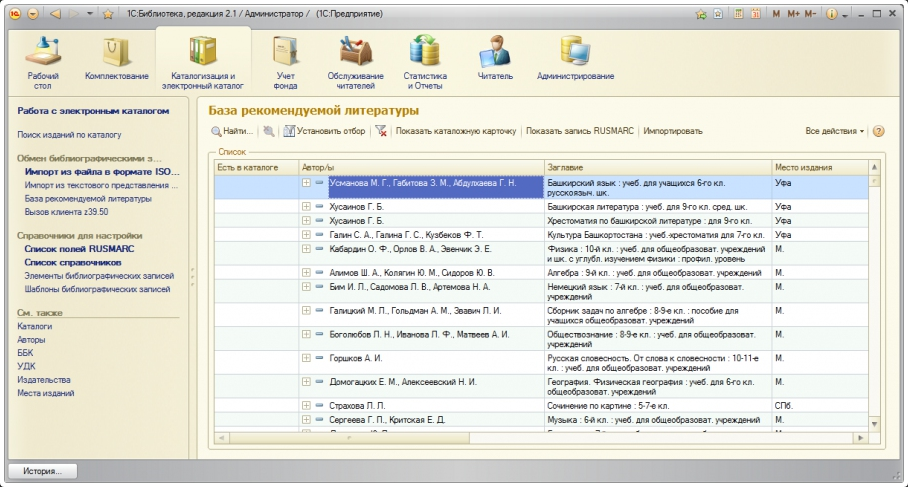
.

# **Назначение программного средства**

Обоснование необходимости в создании программного средства «Учет работы больницы» является упрощение работы сотрудников учреждения здравоохранения, сокращение времени обработки информации, улучшение процесса производства, уменьшении вероятности возникновения ошибок.

# **Описание аналогов**

1С Библиотека (рисунок 1.3.1);



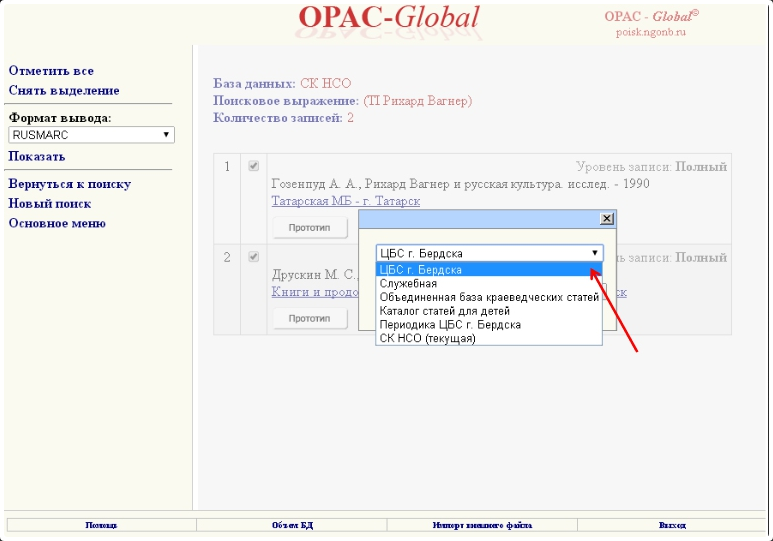
**Рисунок 1.3.1 – Главное окно аналога** «**1С Библиотека**»

Отличия данного аналога программного средства «Управления реестром книг» от представленного заключается в следующих возможностях:

– наличие большего количества функций;

– расширяемость и поддержка производителя.

OPAC-Global (рисунок 1.3.2);



**Рисунок 1.3.2 – Главное окно аналога** «[**OPAC-global**](https://www.livebusiness.ru/tool/2021/)»

Отличия данного аналога программного средства «Управление реестром книг» от представленного заключается в следующих возможностях:

– наличие контактов для обратной связи;

– наличие облачной базы.

# **Технические характеристики**

# **2.1 Постановка задачи на разработку программного средства**

Исходя из современных требований, предъявляемых к качеству работы организации, нельзя не отметить, что его эффективная работа зависит от уровня оснащения компании информационными средствами.

Задача проектируемой базы данных состоит в информационной поддержке библиотеки в целом и для сотрудников в частности. В соответствии с поставленными требованиями автоматизированная информационная система должна содержать данные о пациентах.

Автоматизированная информационная система создается для работы с клиентами. Открывая приложение, работники могут просмотреть список имеющихся карточек и выдать необходимую пациенту или своему коллеге.

# **Описание программного средства**

# **Общие сведения**

Средой разработки послужила Microsoft Visual Studio 2019.

Языком программирования послужил С#, для работы с базой данных использовался фреймворк Entity Framework Core.

При разработке базы данных была использована СУБД (система управления реляционными базами данных) «MySQL».

**2.2.2 Назначение и область применения**

База данных представляет собой организованную структуру, предназначенную для хранения, изменения и обработки взаимосвязанной информации, преимущественно больших объемов. [Базы данных](https://hostiq.ua/wiki/migrate-mysql-database/) активно используются для динамических сайтов со значительными объемами данных – часто это интернет-магазины, порталы, корпоративные сайты. Такие сайты обычно разработаны с помощью серверного языка программирования (как пример, PHP) или на основе CMS (как пример, WordPress), и не имеют готовых страниц с данными по аналогии с HTML-сайтами. Страницы динамических сайтов формируются «на лету» в результате взаимодействия скриптов и баз данных после соответствующего запроса клиента к веб-серверу.

Программное средство для автоматического вызова сотрудника больницы – это система, содержащая в себе базу всех врачей больницы, а также основную информацию о них.

Программное средство является системой учета работы больницы. Так же данное программное средство предоставляет возможность вызвать медработника на дом при помощи нажатия одной кнопки. Предоставляется интерфейс для управления персоналом, пациентами, конференциями и историей вызовов медработника.

Программное средство состоит из реляционной базы данных, программного обеспечения, которое взаимодействует с базой данных, и трех графических пользовательских интерфейсов (один для пациентов, второй для врача, третий для администрации).

Отдельные функции программного средства представляют собой функционально законченные модули, которые объединены в общий интерфейс. Примерный перечень модулей включает:

1. Личный кабинет;
2. Добавление и удаление записей;
3. Учет выписанных лекарств;
4. Интерфейс для администратора;
5. Интерфейс для вызова медработника.

# **Описание логической структуры**

Для корректного функционирования программы необходимо следующее программное обеспечение:

1. Операционная система не ниже Windows 10;
2. .Net Framework версии не ниже 4.6.

Разработанное программное средство предназначено для работы с клиентами организации.

Основные функции программы:

1. Добавление/удаление пользователя;
2. Добавление/удаление сотрудника;
3. Вызвать врача;
4. Заказать лекарства;

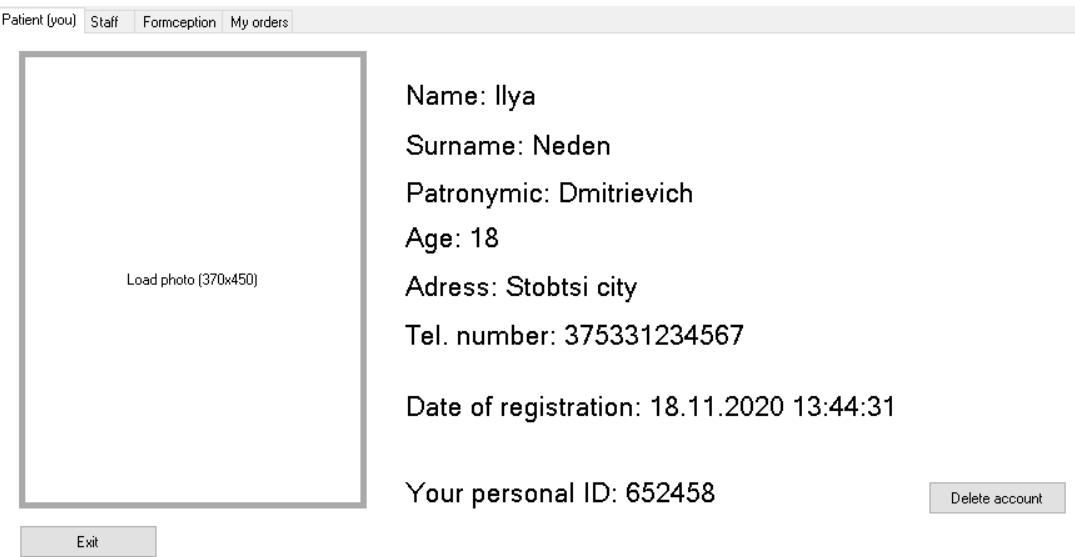
Логическая структура разработанного программного средства (таблица 2.2.3.1).

Таблица 2.2.3.1 – Логическая структура разработанного программного средства

|  |  |
| --- | --- |
| Наименование компонента | Событие |
| 1 | 2 |
| Форма авторизации | |
| Компонент textBox | Используются для ввода логина и пароля |
| Компонент button | Используется для входа в приложение после ввода логина и пароля, и перехода на форму регистрации |
| Компонент label | Используются для визуальной части формы авторизации |
| Форма регистрации | |
| Компонент label | Используются для визуальной части формы регистрации |
| Компонент button | Используется для перехода на следующий шаг регистрации/окончания регистрации и перехода на форму авторизации |
| Компонент textBox | Используется для ввода данных |
| Формы отображения таблиц | |
| Компонент DataGridView | Используется для отображения всех таблиц программного средства |
| Компонент label | Используется для оформления пользовательского интерфейса |
| Компонент button | Используется для добавления/удаления/обновления данных, для выхода из аккаунта и его удаления. Так же для добавления фотографии в кабинете пациента |
| Компонент textBox | Используется для ввода данных |
| Компонент tabControl | Используется для отображения меню и всех компонентов по каждому пункту меню |
| Формы добавления данных | |
| Компонент label | Используется для оформления пользовательского интерфейса |
| Компонент button | Используется для добавления данных |
| Компонент textBox | Используется для ввода данных |

Входными данными в данном проекте являются символы, введённые с клавиатуры (цифры, буквы, знаки). Выходные данные – информационная система, интерфейс и таблицы, предоставляемые пользователю.

Разработанное программное средство (рисунок 2.2.3.1).

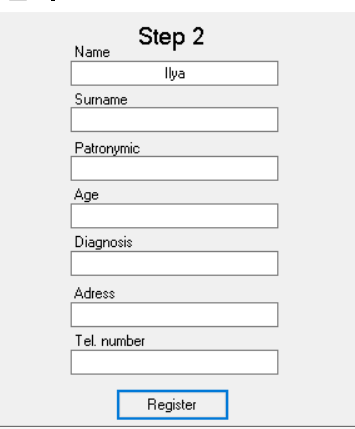


**Рисунок 2.2.3.1 – Внешний личного кабинета пользователя**

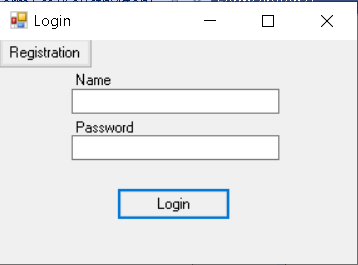
Окно регистрации и авторизации пользователя (рисунок 2.2.3.2).



**Рисунок 2.2.3.2 – Окно регистрации, шаг 1**

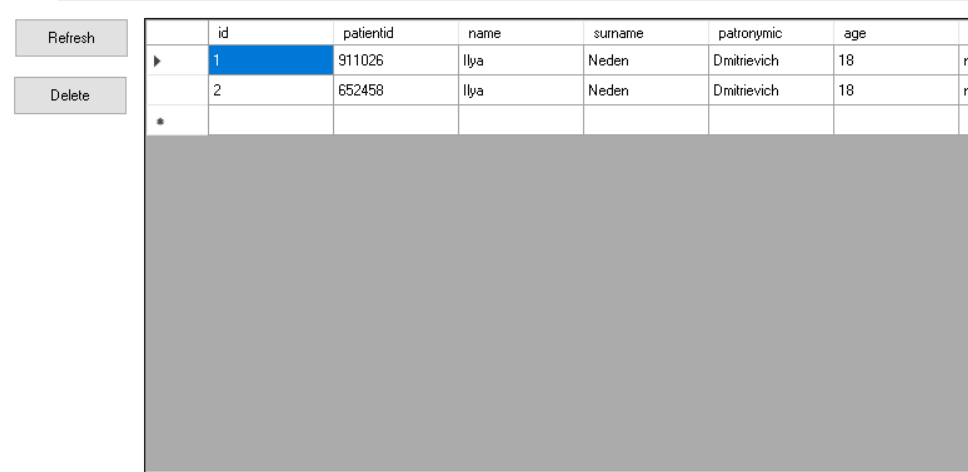


**Рисунок 2.2.3.3 – Окно регистрации, шаг 2**



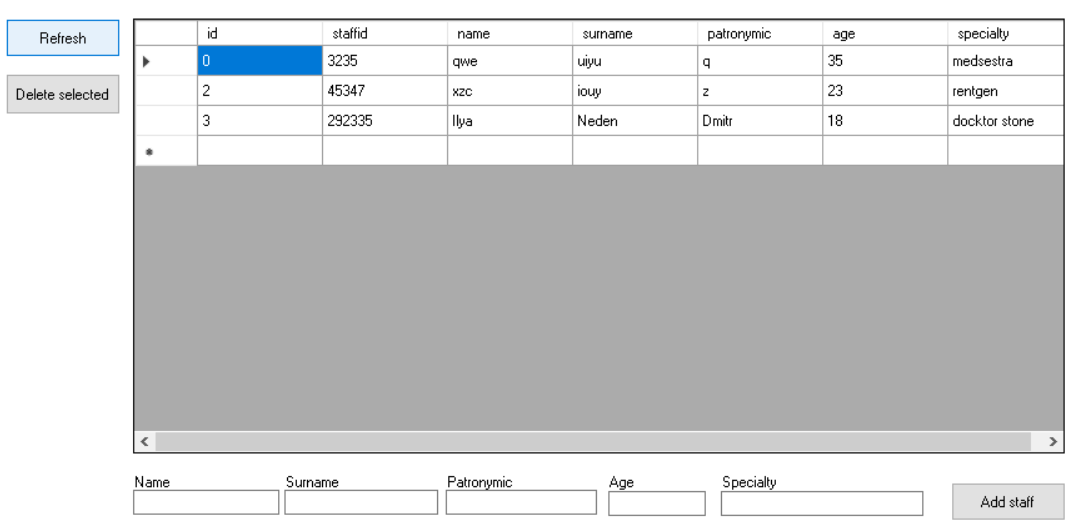
**Рисунок 2.2.3.4 – Окно авторизации**

Внешний вид данных аккаунтов пользователей для администратора (рисунок 2.2.3.5).



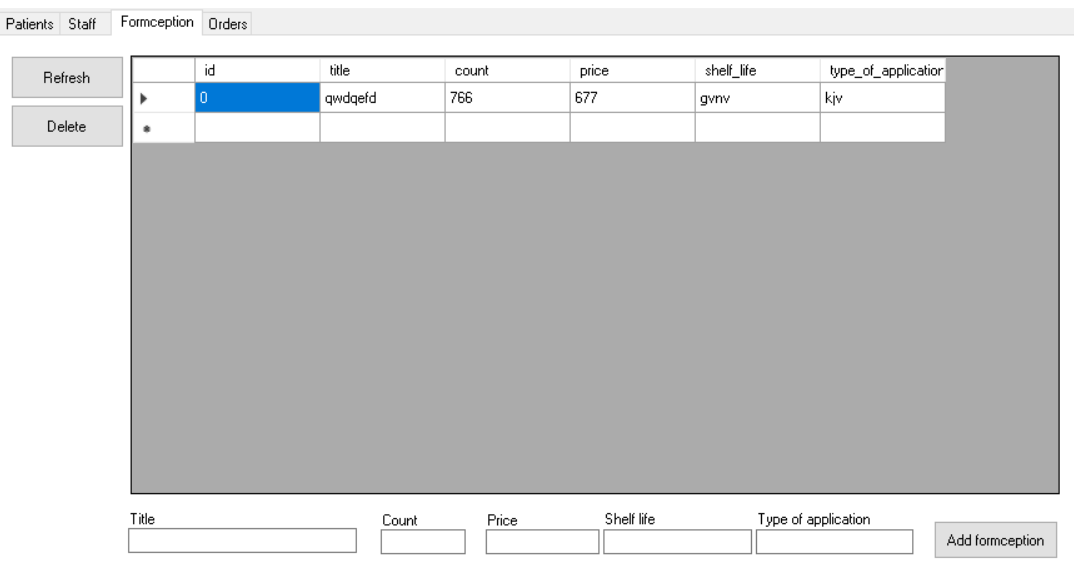
**Рисунок 2.2.3.5 – Внешний вид таблицы всех пользователей**

Внешний вид таблицы персонала, а также форма для их добавления (рисунок 2.2.3.6).



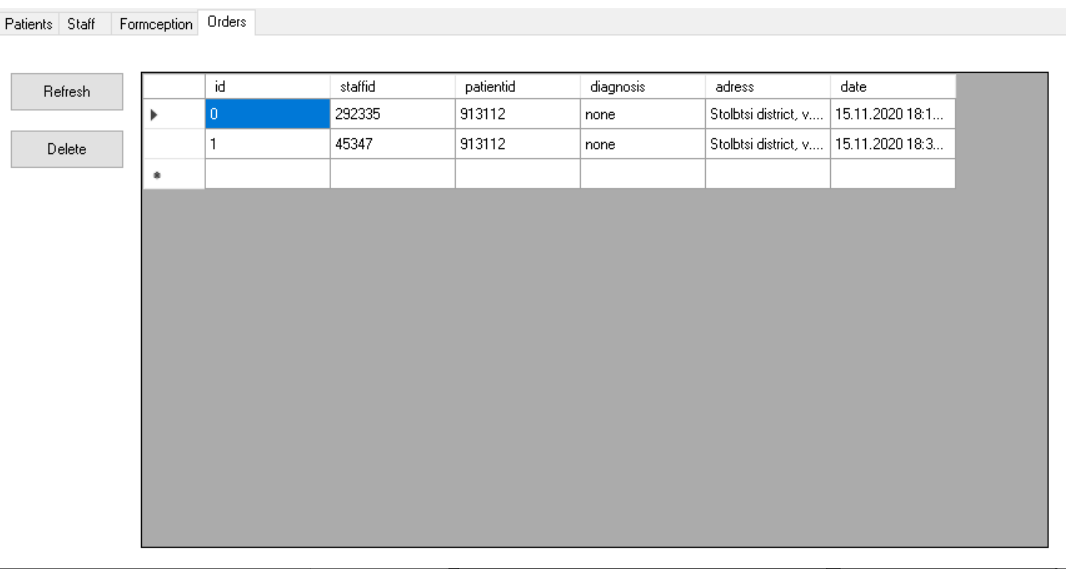
**Рисунок 2.2.3.6 – Таблица персонала**

Внешний вид таблицы лекарств, а также форма для их добавления (рисунок 2.2.3.7).



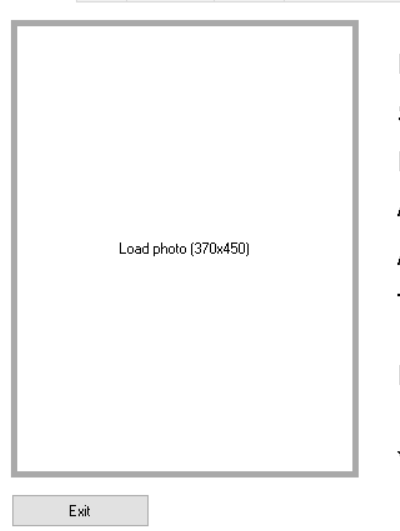
**Рисунок 2.2.3.7 – Таблица лекарств**

Список вызовов врачей (рисунок 2.2.3.8).

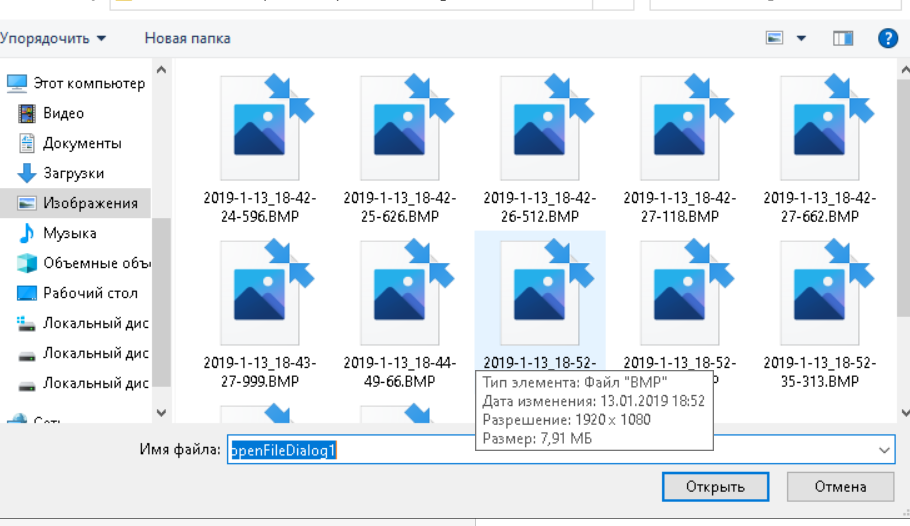


**Рисунок 2.2.3.8 – Таблица вызовов врачей**

Кнопка, для добавления фотографии в личный кабинет (рисунок 2.2.3.9).



**Рисунок 2.2.3.9 – Кнопка добавления фотографии**



**Рисунок 2.2.3.10 – Форма выбора изображения**

# **Используемые технические средства**

Описание системных характеристик компьютера, на котором разрабатывался данный проект (таблица 2.2.4.1).

Таблица 2.2.4.1 – Описание характеристик устройства

|  |  |
| --- | --- |
| **Параметр** | **Конфигурация** |
| **1** | **2** |
| **Процессор** | |
| Модель процессора | AMD E2 |
| Тактовая частота | 1.8 ГГц |
| Энергопотребление | 95 Вт |
| **Оперативная память** | |
| Объем | 4096 Мб |
| Частота | 2666 МГц |
| **Графическая подсистема** | |
| Объем памяти | 2048 МБ |
| Ширина шины памяти | 128 бит |
| Мощность блока питания | 250 Вт |
| **Дисковая подсистема** | |
| Модель | ASX6000LNP-256GT-C |
| Объем | 512 ГБ |

# **Вызов и загрузка**

Для доступа к программе необходимо скачать и установить программную платформу .NET, далее переместить с диска программу или запускать сразу с диска, при помощи запуска иконки «Hospital\_IS\_Managment.exe».

# **Входные данные**

Входной является информация, которую пользователь вносит в файл базы данных, заполняя необходимые поля выбранной таблицы, а также вводя информацию в базу данных с помощью запросов SQL.

Для разрабатываемого программного продукта входной будет служить следующая информация:

* ФИО пациента;
* данные о пациенте;
* данные о персонале;
* данные о лекарствах;

# **Выходные данные**

Выходная информация – результат выполнения запросов, фильтрации данных, вывод необходимой информации в отчет, печать информации. Информация, которая несет вывод либо обобщает указанные данные в общем виде или по определенному критерию. В данной программе выходными данными являются таблицы:

* Информация о книгах;
* Информация об издательствах;
* Информация об издании;

# **Рекомендации по использованию**

Данный проект рекомендуется для использования больницам, для удобного хранения, просмотра и редактировании информации, а также для быстрой выдачи и приема карточек. Данный проект является вспомогательным для различных медицинских учреждений, для более легкого поиска информации в базе данных больницы.

# **ЗАКЛЮЧЕНИЕ**

При написании курсового проекта было спроектировано и реализовано программное средство «учет работы больницы».

Интерфейс созданной программы удобен и прост, наглядно отображает ее возможности. Программа оснащена помощью для пользователя, где подробно описана инструкция по эксплуатации программного продукта.

В результате разработки была достигнута поставленная цель – программное средство «учет работы больницы». Для этого были разработаны алгоритмы и функции для реализации данного приложения с использованием современной технологии программирования. Программное приложение реализовано с использованием объектно-ориентированной технологии программирования. Тестирование показало отличные результаты в работоспособности, целостности и структурированности данных во работы с программой «учет работы больницы».

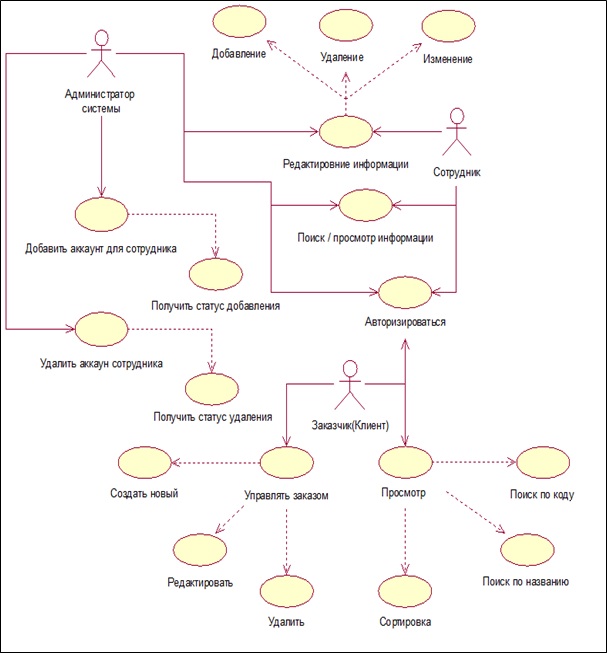
# **СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ**

1. Б. Лоусон, Р. Шарп — HTML 5 / Б. Лоусон, Р. Шарпю – Санкт-Петербург, 2016. – 720 с.
2. Гайдамакин, Н.А. Автоматизированные информационные системы, базы и банки данных. Вводный курс: Учебное пособие / Н.А. Гайдамакин. – М.: Гелиос АРВ, 2002. – 368 с.
3. Дакетт Д. HTML 5. CSS 3 и Web 2.0. Разработка современных. Web-сайтов / 2013 – 480 с.
4. Дронов В.А. HTML 5, CSS 3 и Web 2.0. Разработка современных Web-сайтов. – БХВ-Петербург – Москва: 2011. – 416 c.
5. Дуванов А. А. Web-конструирование. – Петербург: 2009. – 384 с.
6. Едомский Ю. Е. Техника Web-дизайна для студента. – Петербург: 2010. – 491 с.
7. Крамер Э. HTML: курс Web-дизайна. - Киев: 2009. – 304 с.
8. Леонтьев Б. Web-дизайн. Руководство. – Киев: 2011. – 384 с.
9. Корнеев, В.В., Киселев А.В. Современные микропроцессоры /А. Кисилев, В. Корнеев — 2-е изд. — М.: НОЛИДЖ, 2000. – 320 с.
10. Попов, И. И., Партыка, Т. Л. Языки программирования: Учебное пособие /Т. Партыка, И. Попов – М.: ФОРУМ: ИНФРА-М, 2008.
11. Фаронов, В. В., Создание приложений с помощью C#: Руководство программиста /В. Фараонов — М.: Эксмо, 2008.
12. Попов А. А. Программирование в среде СУБД FoxPro 2.0. Построение систем обработки данных / А. Попов. — М.: Издательство Март, 1996 — 352 с.: илл.
13. Матросов А.В., Сергеев А.О., Чаунин М.П. HTML 4.0. – Санкт- Петербург, 1999.
14. Печников В. Н. Создание Web-сайтов. – М.: Триумф: 2010. – 370 с.
15. Роббинс Дж. HTML 5, CSS 3 и javascript / Дж. Роббинс. – Исчерпывающее руководство. 4-е изд., - 2014. – 528 с.
16. Смирнова И. Е. Начала web-дизайна. –Петербург: 2010. – 491 с.
17. Рихтер, Джеффри CLR via C#. Программирование на платформе Microsoft .NET Framework 4.7 на языке C# / Джеффри Рихтер. – М.: Питер, 2013. – 928 c.
18. Форум программистов и сисадминов Киберфорум [Электронный ресурс] / Форум программистов и сисадминов Киберфорум – Режим доступа: www.cyberforum.ru. – Дата доступа: 25.10.2019.
19. Петров, В.Н. Информационные системы / В.Н. Петров. – СПб.: Питер, 2014. – 203 с.
20. Далека, В.Д. Модели структуры данных / В.Д. Далека, О.Г. Кравец, Л.Е. Тимановская. – Харьков.: ХГПУ, 2013. – 54с.
21. Фараонов, В.В. Система программирования С# / В.В. Фараонов. – СПб.: БХВ – Петербург, 2012. – 912 с.

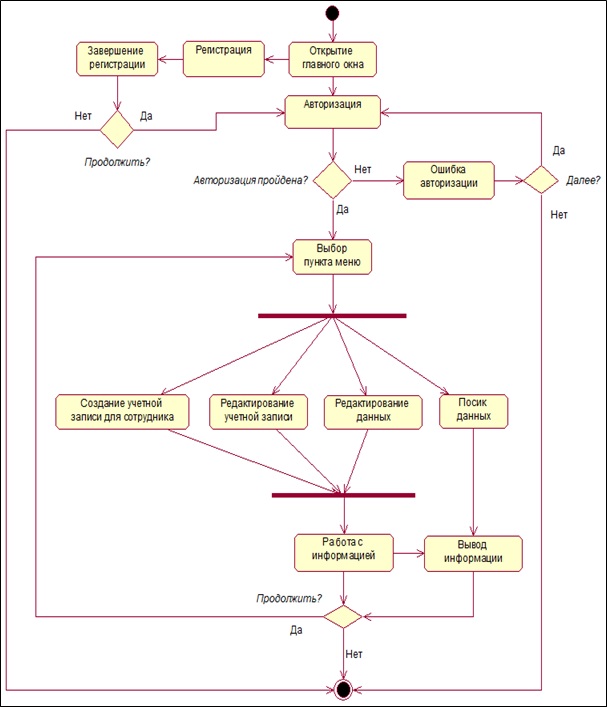
# **ПРИЛОЖЕНИЯ**

# **ПРИЛОЖЕНИЕ А**

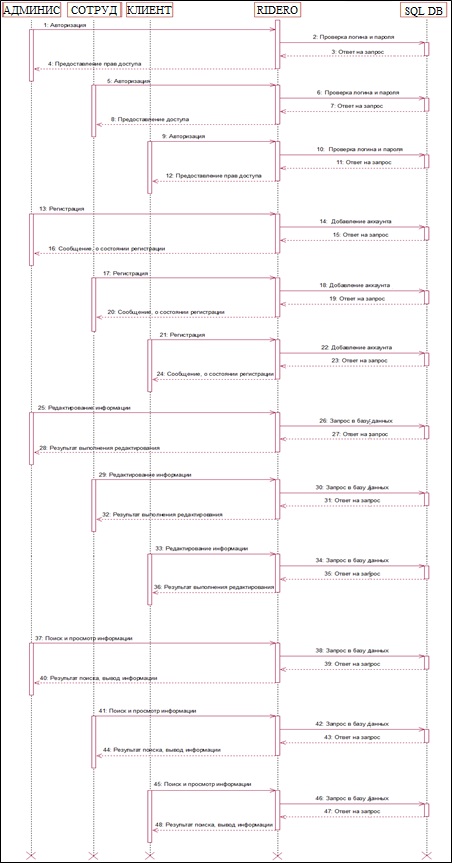
**ДИАГРАММА ВАРИАНТОВ ИСПОЛЬЗОВАНИЯ**

**ПРИЛОЖЕНИЕ Б**

**ДИАГРАММА ДЕЯТЕЛЬНОСТИ**

**ПРИЛОЖЕНИЕ В**

**ДИАГРАММА ПОСЛЕДОВАТЕЛЬНОСТИ**

****

**ПРИЛОЖЕНИЕ Г**

**ЛИСТИНГ ПРОГРАММНОГО СРЕДСТВА**

Модуль Form1

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Hospital\_IS\_Managment

{

public partial class Form1 : Form

{

private const string ADMIN\_LOGIN = "admin";

private const string ADMIN\_PASSWORD = "qwe123";

private DataBaseController db = null;

private Patient p = null;

private Hospital h = null;

bool changestate = false;

public Form1()

{

db = new DataBaseController();

InitializeComponent();

button3.Hide();

label3.Hide();

textBox3.Hide();

label4.Hide();

textBox4.Hide();

label5.Hide();

this.Height = 218;

}

private void button1\_Click(object sender, EventArgs e)

{

if (textBox1.Text.Trim() != "" && textBox2.Text.Trim() != "")

{

if (textBox1.Text == ADMIN\_LOGIN && textBox2.Text == ADMIN\_PASSWORD)

{

//admin

}

if (db.Login(textBox1.Text, textBox2.Text) != null)

{

p = db.Login(textBox1.Text, textBox2.Text);

h = new Hospital(p, this);

h.Show();

refresh();

this.Hide();

} else{

MessageBox.Show("Invalid login data");

warn();

return;

}

}else{

warn();

}

}

private void warn()

{

if (textBox1.Text.Trim() == "")

textBox1.BackColor = (Color)System.ComponentModel.TypeDescriptor.GetConverter(typeof(Color)).ConvertFromInvariantString("#FFFFB9A1");

if (textBox2.Text.Trim() == "")

textBox2.BackColor = (Color)System.ComponentModel.TypeDescriptor.GetConverter(typeof(Color)).ConvertFromInvariantString("#FFFFB9A1");

if (textBox3.Text.Trim() == "")

textBox3.BackColor = (Color)System.ComponentModel.TypeDescriptor.GetConverter(typeof(Color)).ConvertFromInvariantString("#FFFFB9A1");

return;

}

private void refresh()

{

textBox1.Text = ""; textBox2.Text = ""; textBox3.Text = ""; textBox4.Text = ""; textBox5.Text = ""; textBox6.Text = ""; textBox7.Text = "";

button3.Hide();

label3.Hide();

textBox3.Hide();

label4.Hide();

textBox4.Hide();

label5.Hide();

this.Height = 218;

}

private void textBox1\_Enter(object sender, EventArgs e)

{

textBox1.BackColor = (Color)System.ComponentModel.TypeDescriptor.GetConverter(typeof(Color)).ConvertFromInvariantString("Window");

}

private void textBox2\_Enter(object sender, EventArgs e)

{

textBox2.BackColor = (Color)System.ComponentModel.TypeDescriptor.GetConverter(typeof(Color)).ConvertFromInvariantString("Window");

}

private void textBox3\_Enter(object sender, EventArgs e)

{

textBox3.BackColor = (Color)System.ComponentModel.TypeDescriptor.GetConverter(typeof(Color)).ConvertFromInvariantString("Window");

}

private void button2\_Click(object sender, EventArgs e)//change state

{

changestate = !changestate;

if (changestate)

{

button2.Text = "Login";

button1.Hide();

button3.Show();

textBox3.Show();

label3.Show();

label4.Show();

}

else

{

button2.Text = "Registration";

button1.Show();

button3.Hide();

textBox3.Hide();

label3.Hide();

label4.Hide();

}

}

string login = ""; string pass = "";

private void button3\_Click(object sender, EventArgs e)

{

if (textBox1.Text.Trim() != "" && textBox2.Text.Trim() != "" && textBox3.Text.Trim() != "")

{

login = textBox1.Text.Trim();

if (textBox2.Text.Trim() == textBox3.Text.Trim())

{

pass = textBox2.Text.Trim();

button4.Show();

label4.Text = "Step 2";

label1.Text = "Name"; label5.Show(); label7.Show(); textBox6.Show(); textBox3.Text = ""; button3.Hide();

label2.Text = "Surname"; label8.Show(); textBox4.Show(); textBox7.Show(); button2.Hide();

label3.Text = "Patronymic"; label6.Show(); textBox5.Show(); textBox2.Text = "";

this.Height = 370;

}

else

warn();

}

else

{

warn();

}

}

private void button4\_Click(object sender, EventArgs e)

{

if (textBox1.Text.Trim() != "" && textBox2.Text.Trim() != "" && textBox3.Text.Trim() != "" && textBox4.Text.Trim() != "" && textBox5.Text.Trim() != "" && textBox6.Text.Trim() != "" && textBox7.Text.Trim() != "")

{

p = new Patient(new Random().Next(1000000), textBox1.Text.Trim(), textBox2.Text.Trim(), textBox3.Text.Trim(), Convert.ToInt32(textBox4.Text.Trim()), textBox5.Text.Trim(), textBox6.Text.Trim(), textBox7.Text.Trim(), DateTime.Now + "", pass);

if (db.Register(p))

{

h = new Hospital(p, this);

h.Show();

refresh();

this.Hide();

}else{

MessageBox.Show("Registration failed!");

}

}else{

warn();

}

}

private void button5\_Click(object sender, EventArgs e)

{

if (db.Login("Ilya", "123") != null)

{

p = db.Login("Ilya", "123");

h = new Hospital(p, this);

h.Show();

refresh();

this.Hide();

}

}

}

}

Модуль DataBaseController

using MySql.Data.MySqlClient;

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Hospital\_IS\_Managment

{

class DataBaseController

{

private MySqlConnection connect = null;

public DataBaseController()

{

connect = new MySqlConnection("server=localhost;username=root;password=;database=hospital");

connect.Open();

}

public bool Register(Patient p = null)

{

try

{

string sql = string.Format($"Insert Into `hospital`.`{p.TABLE\_NAME}` (id, patientid, name, surname, patronymic, age, diagnosis, adress, tel\_number, date\_of\_visit, pass, medicine, photo) " +

"Values(" + getIndex(p.TABLE\_NAME) + "," + p.Id + ",'" + p.Name + "','" + p.Surname + "','" + p.Patronymic + "', " + p.Age + ",'" + p.Diagnosys + "','" + p.Adress + "','" + p.TelNum + "','" + p.DateOfVisit + "', '" + p.Password + "', '','');");

MySqlCommand cmd = new MySqlCommand(sql, connect);

cmd.ExecuteNonQuery();

return true;

}

catch (Exception ex)

{

MessageBox.Show(ex.StackTrace);

return false;

}

}

public Patient Login(string name, string pass)

{

DataTable dt = new DataTable();

try

{

string sql = $"Select \* From `hospital`.`patients` where name='{name}' and pass='{pass}'";

MySqlCommand cmd = new MySqlCommand(sql, connect);

MySqlDataReader dr = cmd.ExecuteReader();

dt.Load(dr);

if (dt.Rows.Count == 0)

return null;

else

return new Patient(Convert.ToInt32(dt.Rows[0][1].ToString()), dt.Rows[0][2] + "", dt.Rows[0][3] + "", dt.Rows[0][4] + "", Convert.ToInt32(dt.Rows[0][5].ToString()), dt.Rows[0][6] + "", dt.Rows[0][7] + "", dt.Rows[0][8] + "", dt.Rows[0][9] + "", dt.Rows[0][10] + "", dt.Rows[0][11] + "", dt.Rows[0][12] + "");

}

catch (Exception ex) { return null; }

}

public void SaveImage(string path, int id)

{

try

{

path = path.Replace("\\", "|");

string sql = string.Format($"update `hospital`.`patients` set photo = '{path}' where patientid = {id};");

MySqlCommand cmd = new MySqlCommand(sql, connect);

cmd.ExecuteNonQuery();

}

catch (Exception ex)

{

return;

}

}

public void addStaff(Staff s = null)

{

try

{

string sql = string.Format($"Insert Into `hospital`.`{s.TABLE\_NAME}` (id, staffid, name, surname, patronymic, age, specialty) " +

"Values(" + getIndex(s.TABLE\_NAME) + "," + s.Id + ",'" + s.Name + "','" + s.Surname + "','" + s.Patronymic + "', " + s.Age + ",'" + s.Specialty + "');");

MySqlCommand cmd = new MySqlCommand(sql, connect);

cmd.ExecuteNonQuery();

}

catch (Exception ex)

{

MessageBox.Show(ex.StackTrace);

return;

}

}

public DataTable selectStaff()

{

DataTable dt = new DataTable();

string sql = "Select \* From `hospital`.`staff`";

try

{

MySqlCommand cmd = new MySqlCommand(sql, connect);

MySqlDataReader dr = cmd.ExecuteReader();

dt.Load(dr);

if (dt.Rows.Count == 0)

return null;

return dt;

}

catch (Exception e) { return null; }

}

public void deleteStaff(int id)

{

try

{

string sql = "delete from `hospital`.`staff` where staffid=" + id;

MySqlCommand cmd = new MySqlCommand(sql, connect);

cmd.ExecuteNonQuery();

}

catch (Exception ex) { return; }

}

public void orderToHome(Order o)

{

try

{

string sql = string.Format($"Insert Into `hospital`.`{o.TABLE\_NAME}` (id, staffid, patientid, diagnosis, adress, date) " +

"Values(" + getIndex(o.TABLE\_NAME) + "," + o.Staffid + "," + o.Patientid + ",'" + o.Diagnosis + "','" + o.Adress + "', '" + o.Date + "');");

MySqlCommand cmd = new MySqlCommand(sql, connect);

cmd.ExecuteNonQuery();

}

catch (Exception ex)

{

MessageBox.Show(ex.StackTrace);

return;

}

}

public DataTable selectOrders()

{

DataTable dt = new DataTable();

string sql = "Select \* From `hospital`.`orders`";

try

{

MySqlCommand cmd = new MySqlCommand(sql, connect);

MySqlDataReader dr = cmd.ExecuteReader();

dt.Load(dr);

if (dt.Rows.Count == 0)

return null;

return dt;

}

catch (Exception e) { return null; }

}

public void deleteOrder(int userId, int staffid = 0)

{

try

{

string sql = "delete from `hospital`.`orders` where patientid=" + userId + " and staffid=" + staffid;

MySqlCommand cmd = new MySqlCommand(sql, connect);

cmd.ExecuteNonQuery();

}

catch (Exception ex) { return; }

}

public void addFormception(Formception f)

{

try

{

string sql = string.Format($"Insert Into `hospital`.`{f.TABLE\_NAME}` (id, title, count, price, shelf\_life, type\_of\_application) " +

"Values(" + getIndex(f.TABLE\_NAME) + ",'" + f.Title + "'," + f.Count + ", '" + f.Price + "','" + f.ShelfLife + "','" + f.TypeOfAppl + "');");

MySqlCommand cmd = new MySqlCommand(sql, connect);

cmd.ExecuteNonQuery();

}

catch (Exception ex)

{

MessageBox.Show(ex.StackTrace);

return;

}

}

public DataTable selectFormception()

{

DataTable dt = new DataTable();

string sql = "Select \* From `hospital`.`formception`";

try

{

MySqlCommand cmd = new MySqlCommand(sql, connect);

MySqlDataReader dr = cmd.ExecuteReader();

dt.Load(dr);

if (dt.Rows.Count == 0)

return null;

return dt;

}

catch (Exception e) { return null; }

}

public void deleteFormception(int id)

{

try

{

string sql = "delete from `hospital`.`formception` where id=" + id;

MySqlCommand cmd = new MySqlCommand(sql, connect);

cmd.ExecuteNonQuery();

}

catch (Exception ex) { return; }

}

public void deleteUser(int userId)

{

try

{

string sql = "delete from `hospital`.`patients` where patientid=" + userId;

deleteOrder(userId);

MySqlCommand cmd = new MySqlCommand(sql, connect);

cmd.ExecuteNonQuery();

}

catch (Exception ex) { return; }

}

public int getIndex(string table)

{

DataTable dt = new DataTable();

string sql = "Select \* From " + table;

using (MySqlCommand cmd = new MySqlCommand(sql, connect))

{

MySqlDataReader dr = cmd.ExecuteReader();

dt.Load(dr);

dr.Close();

}

int id;

if (dt.Rows.Count > 0)

id = (int)dt.Rows[dt.Rows.Count - 1][0] + 1;

else

return 0;

return id;

}

}

}

Модуль Hospital

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Hospital\_IS\_Managment

{

public partial class Hospital : Form

{

private Form1 main = null;

private Patient user = null;

private DataBaseController db = null;

public Hospital()

{

InitializeComponent();

}

public Hospital(Patient p, Form1 f)//main init

{

InitializeComponent();

main = f;

user = p;

db = new DataBaseController();

loadData();

}

private void loadData()

{

if (user != null)

{

label1.Text += user.Name;

label2.Text += user.Surname;

label3.Text += user.Patronymic;

label4.Text += user.Id;

label5.Text += user.Age;

label6.Text += user.Adress;

label7.Text += user.TelNum;

label8.Text += user.DateOfVisit;

try

{

if (user.Photo != "")

button3.BackgroundImage = Image.FromFile(user.Photo);

}

catch (Exception ex){return; }

dataGridView1.DataSource = db.selectStaff();

dataGridView2.DataSource = db.selectOrders();

dataGridView3.DataSource = db.selectFormception();

}

}

private void button2\_Click(object sender, EventArgs e)//Exit from account

{

user = null;

main.Show();

this.Dispose();

this.Close();

}

private void button1\_Click(object sender, EventArgs e)//delete account

{

db.deleteUser(user.Id);

user = null;

main.Show();

this.Dispose();

this.Close();

}

private void button3\_Click(object sender, EventArgs e)

{

try

{

if (openFileDialog1.ShowDialog() == DialogResult.Cancel) return;

string filename = openFileDialog1.FileName;

button3.BackgroundImage = Image.FromFile(filename);

db.SaveImage(filename, user.Id);

}catch(Exception ex) { return; }

}

private void button4\_Click(object sender, EventArgs e)//refresh staff

{

dataGridView1.DataSource = db.selectStaff();

}

private void button5\_Click(object sender, EventArgs e)//add new staff

{

db.addStaff(new Staff(new Random().Next(1000000), textBox1.Text.Trim(), textBox2.Text.Trim(), textBox3.Text.Trim(), Convert.ToInt32(textBox4.Text.Trim()), textBox5.Text.Trim()));

}

private void button6\_Click(object sender, EventArgs e)//delete selected staff

{

db.deleteStaff(Convert.ToInt32(dataGridView1.SelectedRows[0].Cells[1].Value));

}

private void button7\_Click(object sender, EventArgs e)//order to home

{

db.orderToHome(new Order(Convert.ToInt32(dataGridView1.SelectedRows[0].Cells[1].Value), user.Id, user.Diagnosys, user.Adress, DateTime.Now+""));

dataGridView2.DataSource = db.selectOrders();

}

private void button8\_Click(object sender, EventArgs e)//refresh orders

{

dataGridView2.DataSource = db.selectOrders();

}

private void button9\_Click(object sender, EventArgs e)//delete order

{

db.deleteOrder(Convert.ToInt32(dataGridView2.SelectedRows[0].Cells[2].Value), Convert.ToInt32(dataGridView2.SelectedRows[0].Cells[1].Value));

}

private void button10\_Click(object sender, EventArgs e)//refresh formception

{

dataGridView3.DataSource = db.selectFormception();

}

private void button11\_Click(object sender, EventArgs e)//delete formception

{

db.deleteFormception(Convert.ToInt32(dataGridView2.SelectedRows[0].Cells[0].Value));

}

private void button12\_Click(object sender, EventArgs e)//add formception

{

db.addFormception(new Formception(textBox10.Text.Trim(), Convert.ToInt32(textBox9.Text.Trim()), textBox8.Text.Trim(), textBox7.Text.Trim(), textBox6.Text.Trim()));

}

}

}