**Министерство науки и высшего образования Российской Федерации**

Федеральное государственное бюджетное образовательное учреждение высшего образования

«**Владимирский государственный университет имени Александра Григорьевича и Николая Григорьевича Столетовых**»

(ВлГУ)

Колледж инновационных технологий и предпринимательства

**КАФЕДРА ФИЗИКИ И ПРИКЛАДНОЙ МАТЕМАТИКИ**

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОЙ РАБОТЕ**

по дисциплине «Технология разработки и защиты баз данных»

Тема: «Медицинские услуги. Больной, болезнь, история: разработка и администрирование базы данных, разработка клиентского приложения»

Выполнил студент

группы ПКсп-120

Владимиров А.И.

Приняла Кабанова М.Ю.

Владимир, 2023



Содержание

[Введение 3](#_Toc131390847)

[1 Проектирование базы данных 4](#_Toc131390848)

[1.1 ER-диаграмма 4](#_Toc131390849)

[2. Подготовка базы данных 6](#_Toc131390850)

[2.1 Словарь данных 6](#_Toc131390851)

[2.2 Нормализация полученных отношений 8](#_Toc131390852)

[2.3 Процедуры, функции, представления 9](#_Toc131390853)

[3. Резервное копирование и восстановление 12](#_Toc131390854)

[4 УПРАВЛЕНИЕ ДОСТУПОМ 14](#_Toc131390855)

[5 Разработка клиентского приложения 18](#_Toc131390856)

[5.1 Структура приложения 18](#_Toc131390857)

[5.1.1 Форма обычного пользователя 19](#_Toc131390858)

[5.1.2 Форма админа больницы 20](#_Toc131390859)

[5.2 Пользовательский интерфейс 24](#_Toc131390860)

[Заключение 31](#_Toc131390861)

[Список использованных источников 32](#_Toc131390862)

[Приложение А 33](#_Toc131390863)

[Приложение Б 35](#_Toc131390864)

[Приложение В 41](#_Toc131390865)

[Приложение Г 44](#_Toc131390866)

# Введение

SQL Server – это система, созданная для управления набора данных с предопределенными связями между ними, которые организованы в виде таблиц, состоящих из строк и столбцов. SQL Server используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятий, выполняющие функции постоянного хранения данных, поиска по ключевым критериям, чтение и редактирования реляционными базами данных [1)].

В настоящее время базы данных используются для хранения, обработки и структурирования информации. Так же существует возможность множеству людей одновременно использовать базу данных, что делает работу с данными более быстрее и безопасней.

Базы данных являются эффективным способом хранения информации о докторах, пациентах, больницах, в которых они проходят консультацию или лечение, полюсах пациентов. Если хранить данную информацию в базе данных, то вся необходимая информация хранится в структурированном виде [9)]. База данных может использоваться в любой больнице для эффективного хранения и обработки большого количества информации [3)].

# 1 Проектирование базы данных

## 1.1 ER-диаграмма

В базе данных «Медицинские услуги. Больной, болезнь, история.» содержится следующий набор сущностей с присущими им атрибутами:

* + Врач (doctor): Должность; Фамилия; Имя; Отчество; Цена приема;
  + История болезни (medical\_history): Мед. карта; Номер электронной мед. карты;
  + Пациент (patient): Фамилия; Имя; Отчество; Пол; Возраст; Номер телефона; Серия и номер паспорта;
  + Заболевание (disease): Дата заболевания; Температура;
  + Полис (policy): Страховая компания;
  + Поликлиника (hospital): Город; Улица; Номер дома;

Между этими сущностями организованы связи типа «один-ко-многим», «многие-ко-многим» через ассоциативные таблицы [8)].

Диаграмма Чена представлена на рисунке 1.

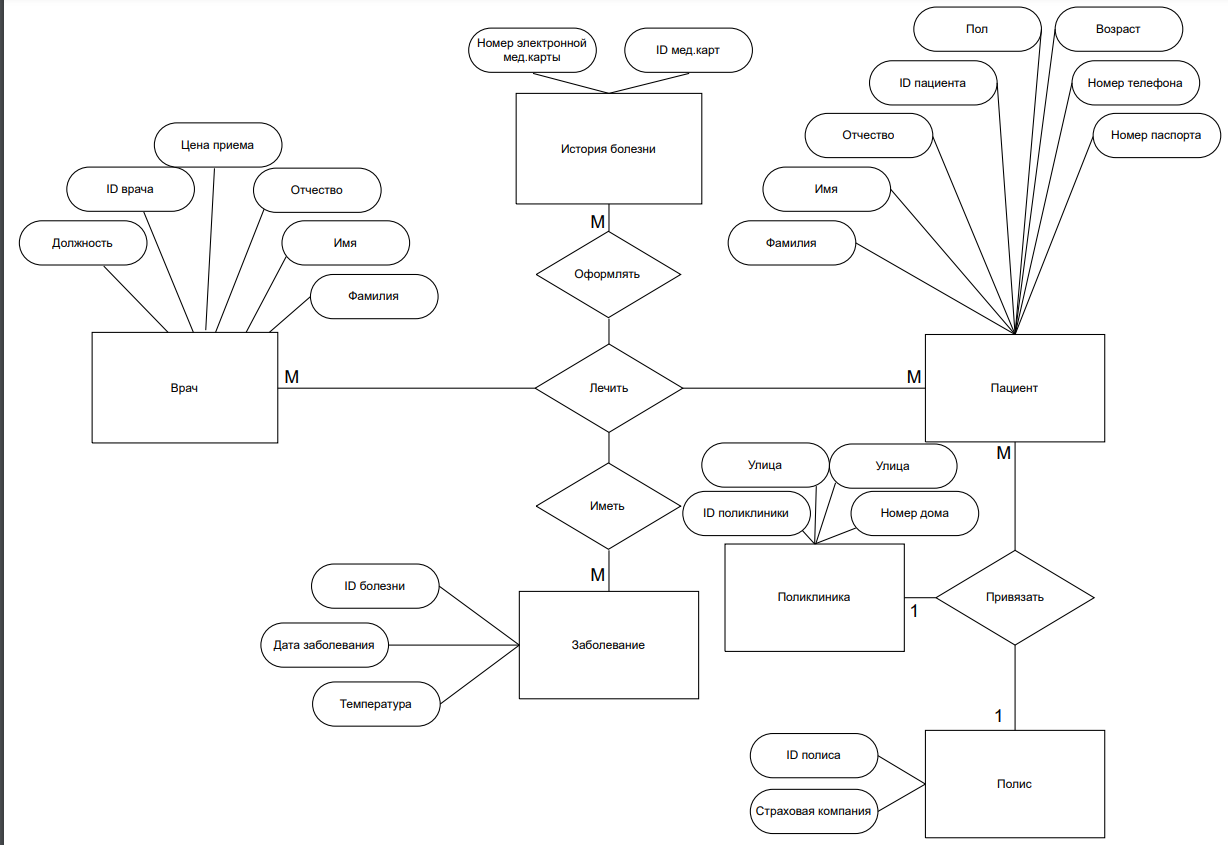


Рисунок 1 – Диаграмма Чена

Диаграмма Мартина представлена на рисунке 2.

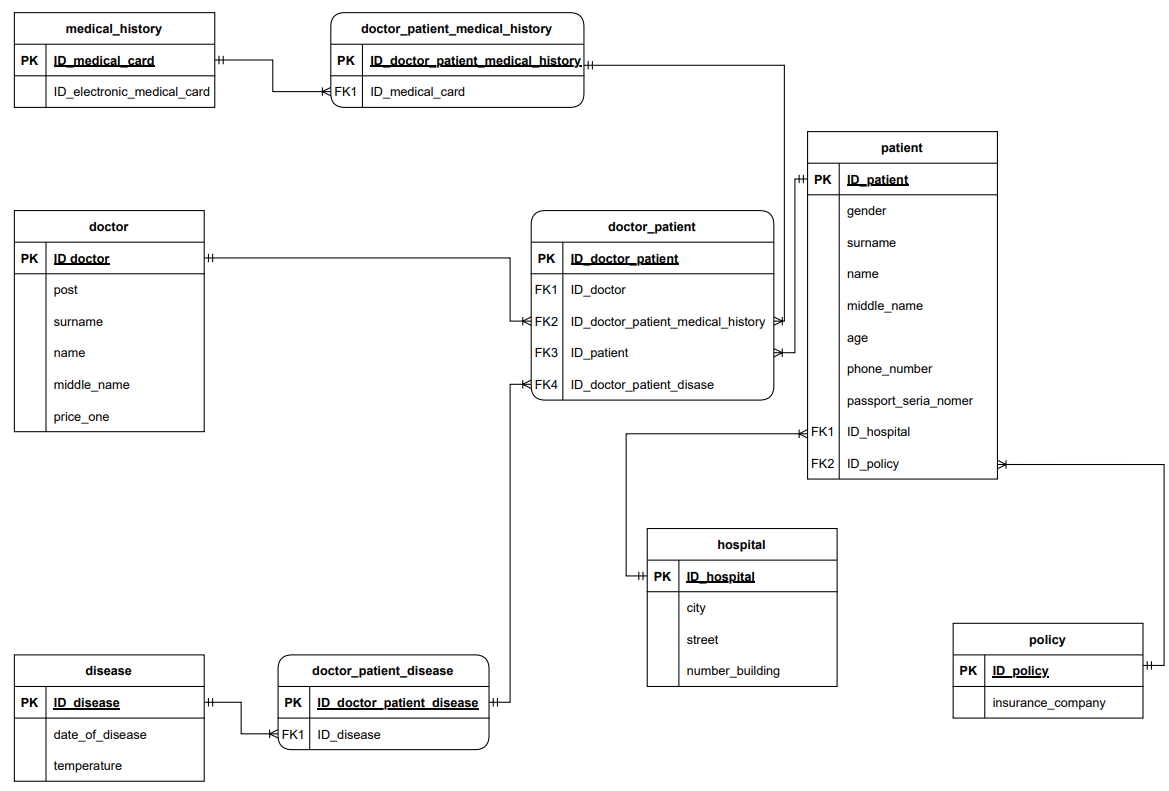


Рисунок 2 – Диаграмма Мартина

Исходя из этих таблиц можно составить словарь данных

# 2. Подготовка базы данных

## 2.1 Словарь данных

Словарь данных для создания базы данных представлен в таблице 1.

Таблица 1 – Словарь данных

|  |  |  |  |
| --- | --- | --- | --- |
| doctor | | | |
| Ключ | Поле | Обязательное (да/нет) | Примечание |
| Первичный ключ | id\_doctor | Да | Id врача |
|  | surname | Да | Фамилия врача |
|  | name | Да | Имя врача |
|  | middle\_name | Нет | Отчество врача |
|  | post | Да | Должность врача |
|  | Price\_one | Да | Цена приема |
| doctor\_patient | | | |
| Первичный ключ | id\_doctor\_patient | Да | Id врача и пациента |
| Внешний ключ | id\_doctor | Да | Внешний ключ на таблицу doctor |
| Внешний ключ | id\_doctor\_patient\_medical\_history | Да | Внешний ключ на doctor\_patient\_medical\_history |
| Внешний ключ | id\_patient | Да | Внешний ключ на таблицу patient |
| Внешний ключ | id\_doctor\_patient\_disease | Да | Внешний ключ на doctor\_patient\_disease |
| medical\_history | | | |
| Первичный ключ | id\_medical\_card | Да | Id мед. карты |
|  | number\_electronic\_medical\_card | Да | Номер электронной мед. карты |
| doctor\_patient\_medical\_history | | | |

Продолжение таблицы 1

|  |  |  |  |
| --- | --- | --- | --- |
| Первичный ключ | id\_doctor\_patient\_medical\_history | Да | Id врача, пациента и истории болезни |
| Внешний ключ | id\_medical\_card | Да | Внешний ключ на таблицу medical\_card |
| disease | | | |
| Первичный ключ | id\_disease | Да | Id болезни |
|  | date\_of\_disease | Да | Дата заболевания |
|  | temperature | Нет | Значение температуры |
| doctor\_patient\_disease | | | |
| Первичный ключ | ID\_doctor\_patient\_disease | Да | Id врача, пациента и болезни |
| Внешний ключ | id\_disease | Да | Внешний ключ на таблицу disease |
| patient | | | |
| Первичный ключ | ID\_patient | Да | Id пациента |
|  | gender | Да | Пол пациента |
|  | surname | Да | Фамилия |
|  | name | Да | Имя |
|  | middle\_name | Нет | Отчество |
|  | age | Да | Возраст |
|  | phone\_number | Да | Номер телефона |
|  | passport\_seria\_nomer | Да | Серия и номер паспорта |
| Внешний ключ | ID\_hospital | Да | Внешний ключ на таблицу hospital |
| Внешний ключ | ID\_policy | Да | Внешний ключ на таблицу policy |

Продолжение таблицы 1

|  |  |  |  |
| --- | --- | --- | --- |
| hospital | | | |
| Первичный ключ | ID\_hospital | Да | Id поликлиники |
|  | street | Да | Улица |
|  | number\_building | Да | Номер дома |
|  | city | Да | Город |
| policy | | | |
| Первичный ключ | ID\_policy | Да | Id полиса |
|  | insurance\_company | Да | Страховая компания полиса |

Итоговая диаграмма базы данных, сгенерированная после создания всех таблиц представлена на рисунке 3.

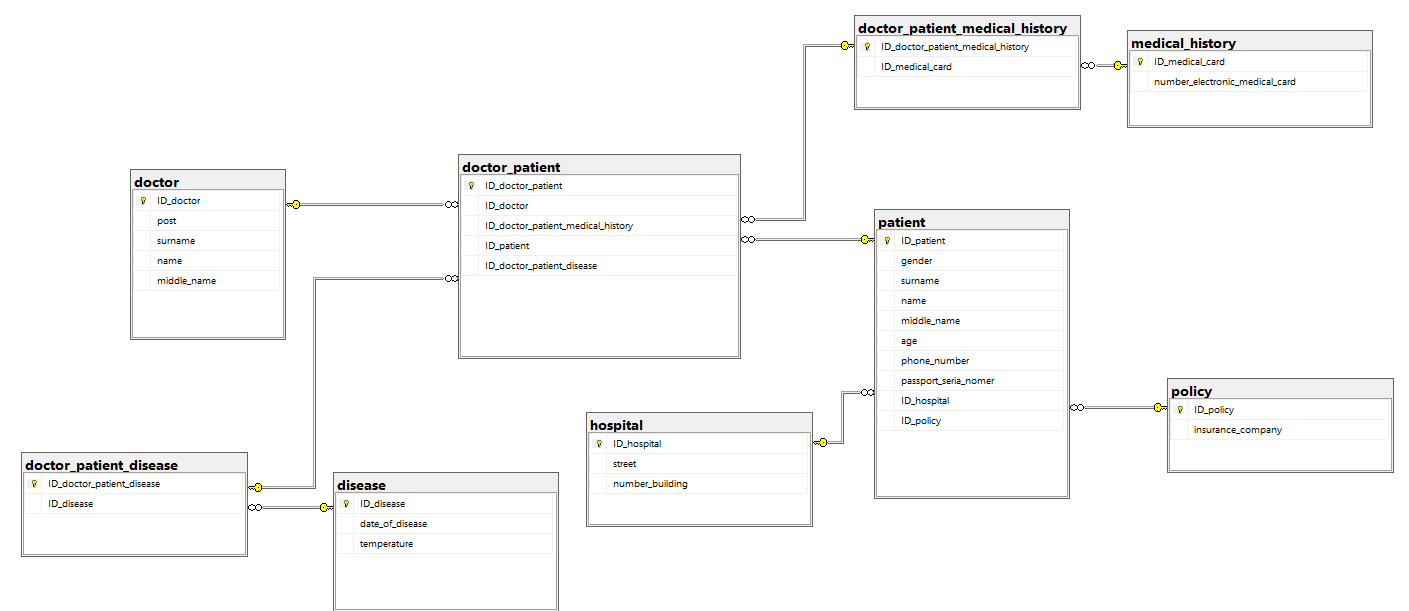


Рисунок 3 – Диаграмма базы данных

Код создания таблиц представлен в Приложении А. Код заполнения базы данных представлен в приложении Б.

## 2.2 Нормализация полученных отношений

Разработанная схема базы данных находится в:

* Первой нормальной форме, так как в качестве доменов выступают только скалярные значения, и информация в таблицах не дублируется. Во всех таблицах есть идентификатор id в качестве первичного ключа
* Второй и третьей нормальных формах, каждый не ключевой атрибут не приводимо и не транзитивно зависит от первичного ключа. Схема базы данных находится в нормальной форме Бойса-Кодда.

## 2.3 Процедуры, функции, представления

В базе данных реализована хранимая процедура, скалярная функция, представление, триггеры.

В базе данных реализована скалярная функция fio\_doctor1, которая рассчитывает цену приемов при покупке трех сразу, с учетом скидки 500 рублей. У данной функции имеется входной параметр, в который записывается фамилия доктора. После чего происходит вызов значения из таблицы doctor, умножается на три, далее вычитается 500.

Выходным параметром данной функции является числовое значение, которое определяет цену за три приема, с учетом скидки 500 рублей.

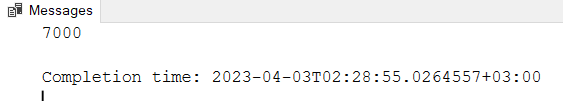


Рисунок 4 – Результат работы скалярной функции

Хранимая процедура имеет название: count\_patient. Она предназначена для того, чтобы вывести в одну таблицу информацию о определенном пациенте. Данная процедура путём объединения нескольких таблиц patient, doctor, policy, doctor\_patient облегчает поиск информации. На выходе из хранимой процедуры формируется выборка, в которую будут входить такие поля, как name, surname, middle\_name, passport\_seria\_nomer из таблицы patient, name, surname, middle\_name из таблицы doctor, insurance\_company из таблицы policy.

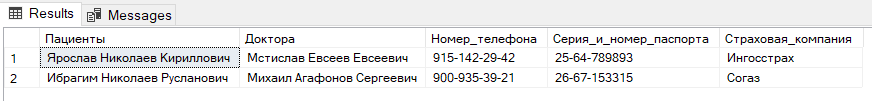


Рисунок 5 – Результат работы хранимой процедуры

В базе данных содержится представление с названием all.

Оно предназначено для того, чтобы вывести в одну таблицу информацию о ФИО доктора, его должности, его пациентов, страховщик пациента и адрес больницы. Данное представление путём объединения нескольких таблиц doctor, patient, policy и hospital облегчает поиск нужной информации. На выходе из представления формируется выборка, в которую будут входить такие поля, как surname, name, middle\_name и post из таблицы doctor, surname, name, middle\_name из таблицы patient, insurance\_company из таблицы policy, city, street, number\_building из таблицы hospital.

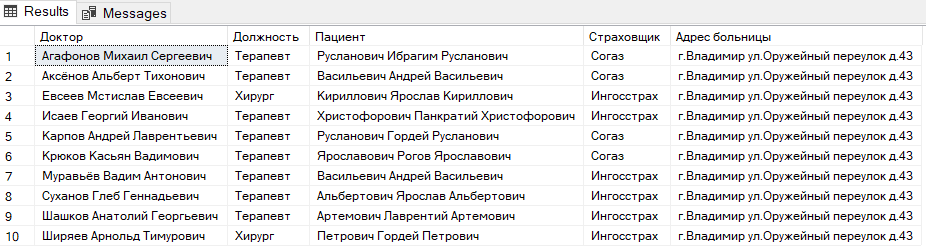


Рисунок 6 – Результат работы представления

В базе данных разработаны триггеры:

* insert\_trig. Используется для добавления.
* del\_rollback. Используется для удаления.
* update. Используется для добавления записи в таблицу History о обновлении таблицы doctor.

Данные триггеры при добавлении, удалении и обновлении данных в таблице doctor дублируют информацию, которая добавляется, удаляется или обновляется, в таблицу History, а также вносится информация кто совершил действие, во сколько это произошло и что именно произошло, добавление, обновление или удаление. Если информация была добавлена в поле CONVERT пишется INSERT, если удалена DELETE, если обновлена UPDATE.

Коды создания скалярной функции, хранимой процедуры, представления и триггеров представлены в приложении В.

# 3. Резервное копирование и восстановление

Чтобы выполнить резервное копирование базы данных с использованием командного файла, нужно написать SQL-скрипт, который реализует резервное копирование базы, затем сохранить данный скрипт в файле с расширением .sql. Скрипт для резервного копирования представлен в листинге 1.

Листинг 1

BACKUP DATABASE [Med\_Yslugi]

TO DISK = N'C:\SQL Buckup\Med\_Yslugi.bak' WITH NOFORMAT, NOINIT, NAME = N'Med\_Yslugi-Full Database Backup', SKIP, NOREWIND, NOUNLOAD, STATS = 10

GO

Для запуска данного скрипта необходимо с помощью любого текстового редактора создать скрипт файл с расширением .bat, внутри которого написать код, представленный в листинге 2.

Листинг 2

sqlcmd -s laptop-clvpr4q1 -i C:\SQL Buckup\back.bat

Теперь при запуске файла будет создаваться резервная копия базы данных в указанной папке.

Так же был создан SQL-скрипт и командный файл для запуска восстановления базы данных из резервной копии. SQL-скрипт для восстановления представлен в листинге 3.

Листинг 3

USE [master]

BACKUP LOG [Med\_Yslugi] TO DISK = N'C:\Program Files\Microsoft SQL Server\MSSQL16.MSSQLSERVER\MSSQL\Backup\Med\_Yslugi\_LogBackup\_2023-03-09\_11-13-42.bak' WITH NOFORMAT, NOINIT, NAME = N'Med\_Yslugi\_LogBackup\_2023-03-09\_11-13-42', NOSKIP, NOREWIND, NOUNLOAD, NORECOVERY , STATS = 5

RESTORE DATABASE [Med\_Yslugi] FROM DISK = N'C:\SQL Buckup\Med\_Yslugi.bak' WITH FILE = 1, NOUNLOAD, STATS = 5

GO

Код командного файла представлен в листинге 4.

Листинг 4

sqlcmd -s laptop-clvpr4q1 -i C:\SQL Buckup\res.sql

Теперь база данных содержит резервную копию, которой можно воспользоваться в любой момент и открыть на других компьютерах [4)].

# 4 УПРАВЛЕНИЕ ДОСТУПОМ

Для управления доступом в базе банных с помощью sql скриптов были созданы две роли:

* sysAdm1, которая не имеет ограничений в работе с данными в базе данных [5)].
* visiter, которая может лишь просматривать данные в таблицах doctor и hospital, соответственно для этой роли ограничены возможности добавления, удаления и редактирования данных.

Для каждой роли были созданы пользователи:

* admin111, имеет роль sysAdm111
* VISITER1, имеет роль VISITER1

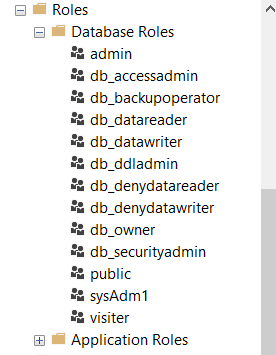


Рисунок 7 – Роли в базе данных

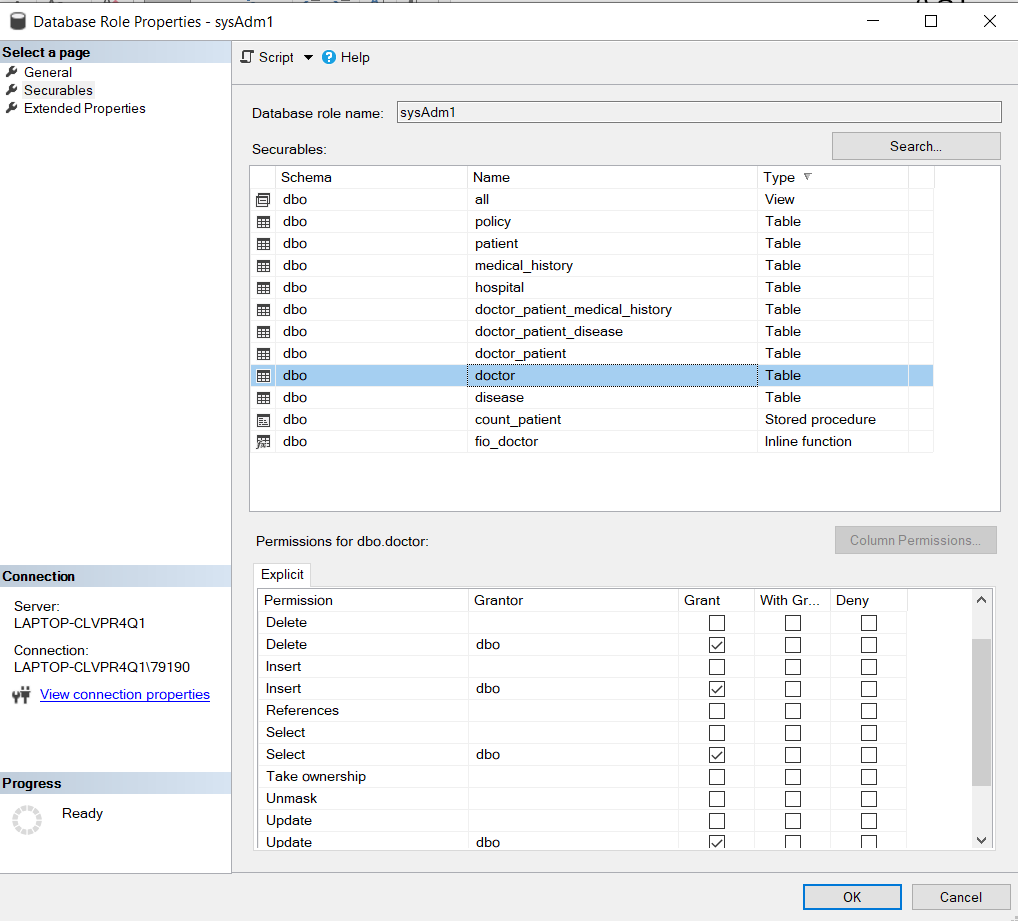


Рисунок 8 – Права роли sysAdm1

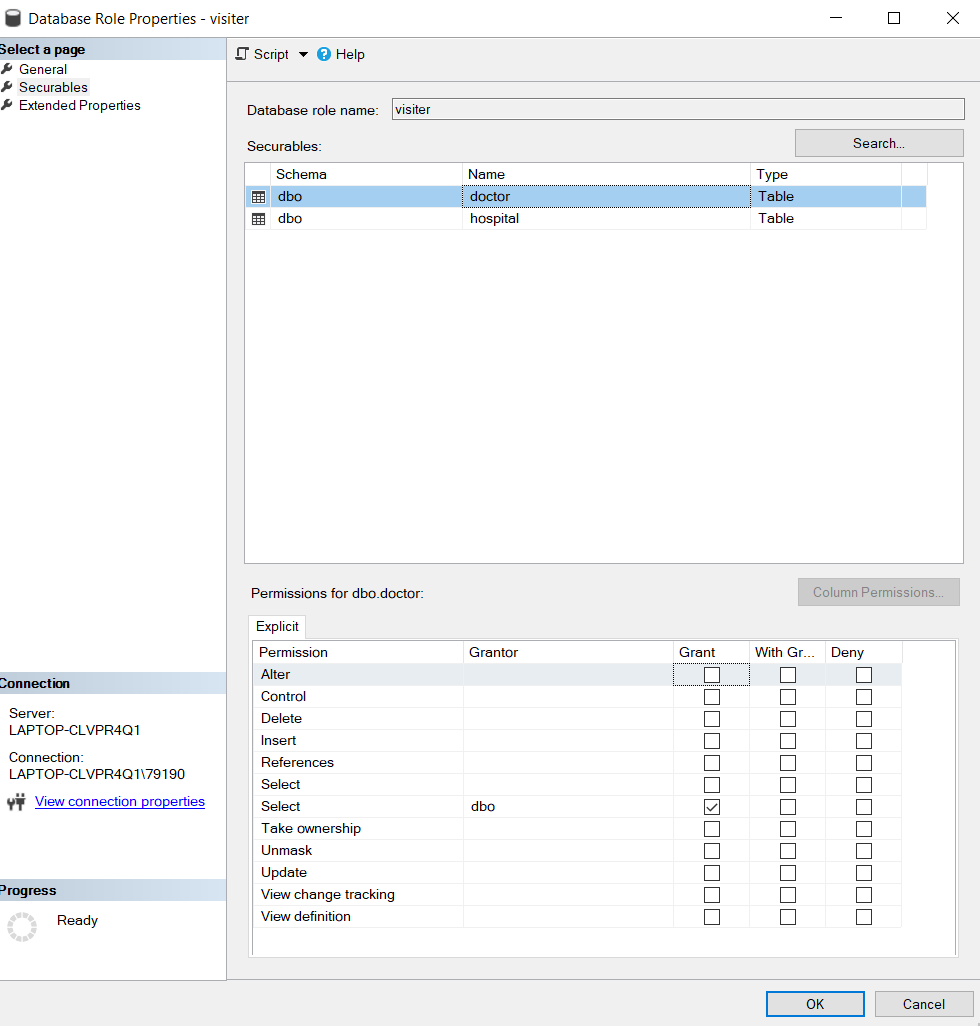


Рисунок 9 – Права роли visiter

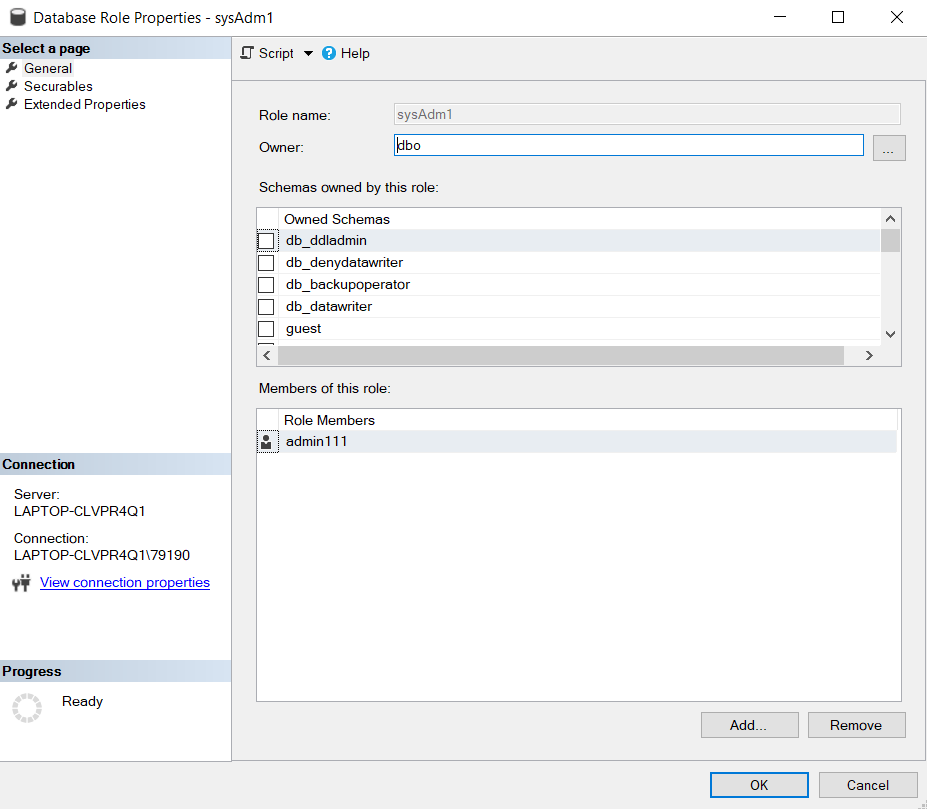


Рисунок 10 – Пользователь с ролью sysAdm1

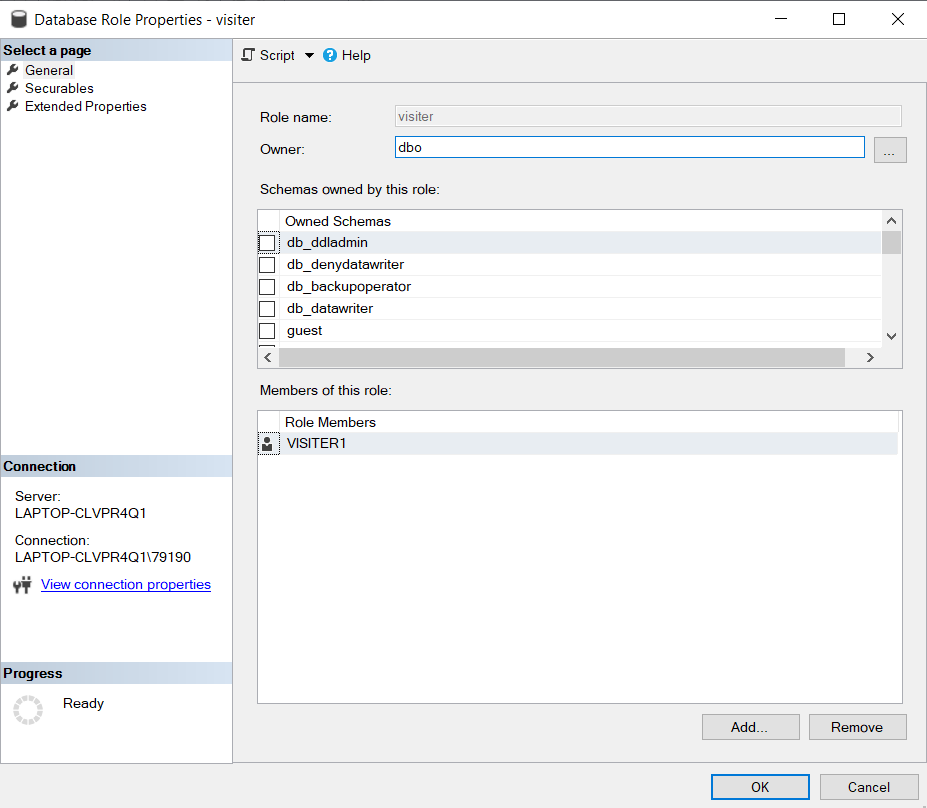


Рисунок 11 – Пользователь с ролью visiter

SQL скрипт выполняющий создание ролей и пользователей представлены в приложении В

# 5 Разработка клиентского приложения

## 5.1 Структура приложения

Диаграмма прецедентов позволяет визуализировать различные типы ролей в системе, она нужна, чтобы показать функционал и поведение каждого участника предметной области. Актёр - любая сущность, которая выполняет роль в системе. Случай использования - набор действий, который может быть использован актёром для взаимодействия с системой. При создании диаграммы следует задать вопросы:

* Что может ещё произойти?
* Что может пойти не так?

Диаграмма прецедентов показана на рисунке 12.

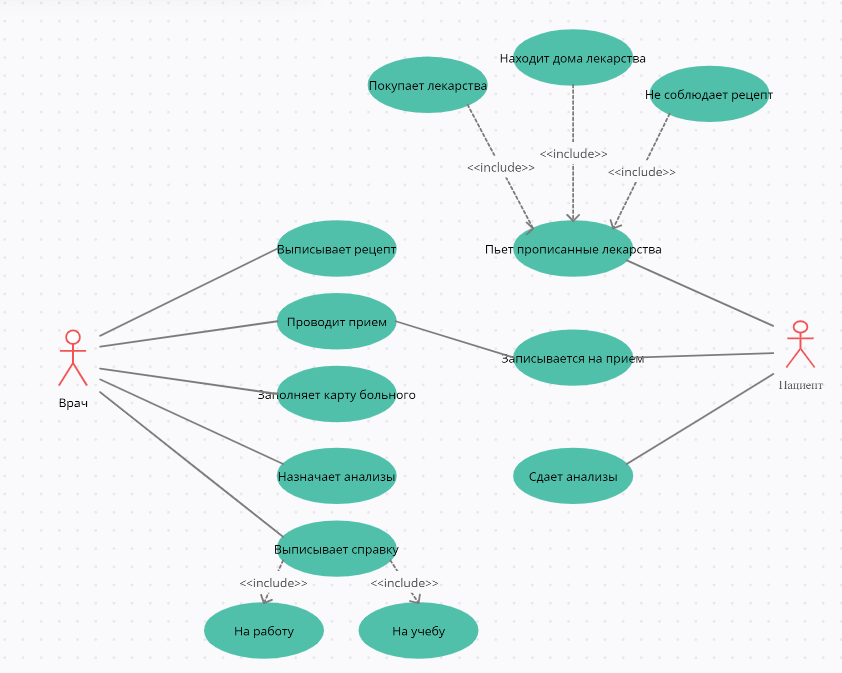


Рисунок 12 – Диаграмма прецедентов

При запуске приложения появляется экран авторизации, где требуется ввести логин и пароль, после чего осуществляется вход в систему по кнопке «Вход». Если пользователя нет в системе он может нажать на кнопку «Регистрация» и перейти на экран регистрации, где после ввода логина, под которым он будет заходить в систему, а также пароля, после он нажимает на кнопку входа и может совершить авторизацию в систему [6)].

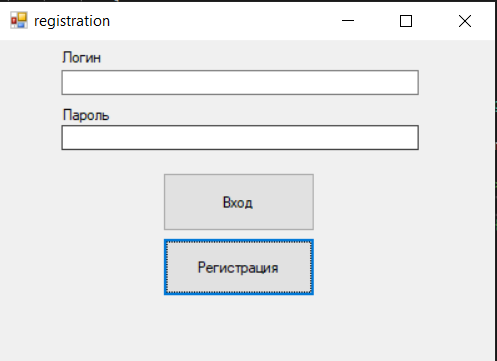


Рисунок 13 – Панель входа

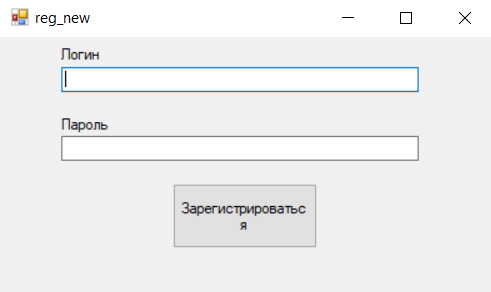


Рисунок 14 – Окно регистрации

После прохождения авторизации пользователь попадает на главную форму в соответствии с тем, какие права он имеет.

### 5.1.1 Форма обычного пользователя

При входе от обычного пользователя появляется окно, которое содержит только некоторые таблицы и все данные доступны только для чтения [7)].

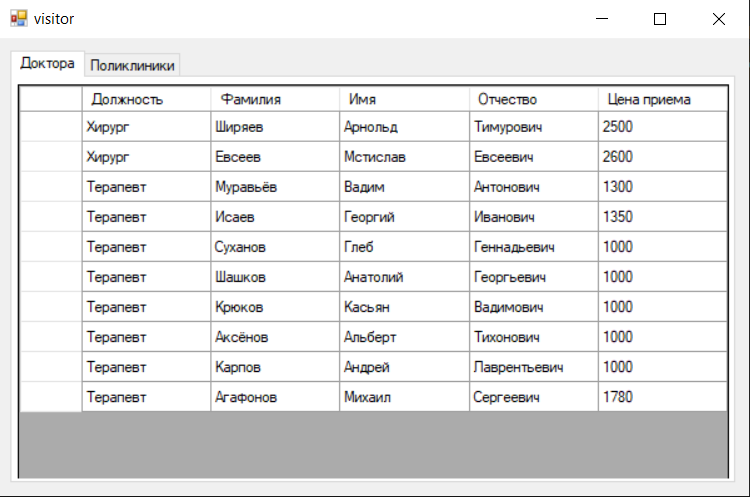


Рисунок 15 – Окно пользователя

После запуска формы, пользователь может прочитать информацию о докторах и поликлиниках.

### 5.1.2 Форма админа больницы

При входе в систему имея права админа главное окно выглядит следующим образом.

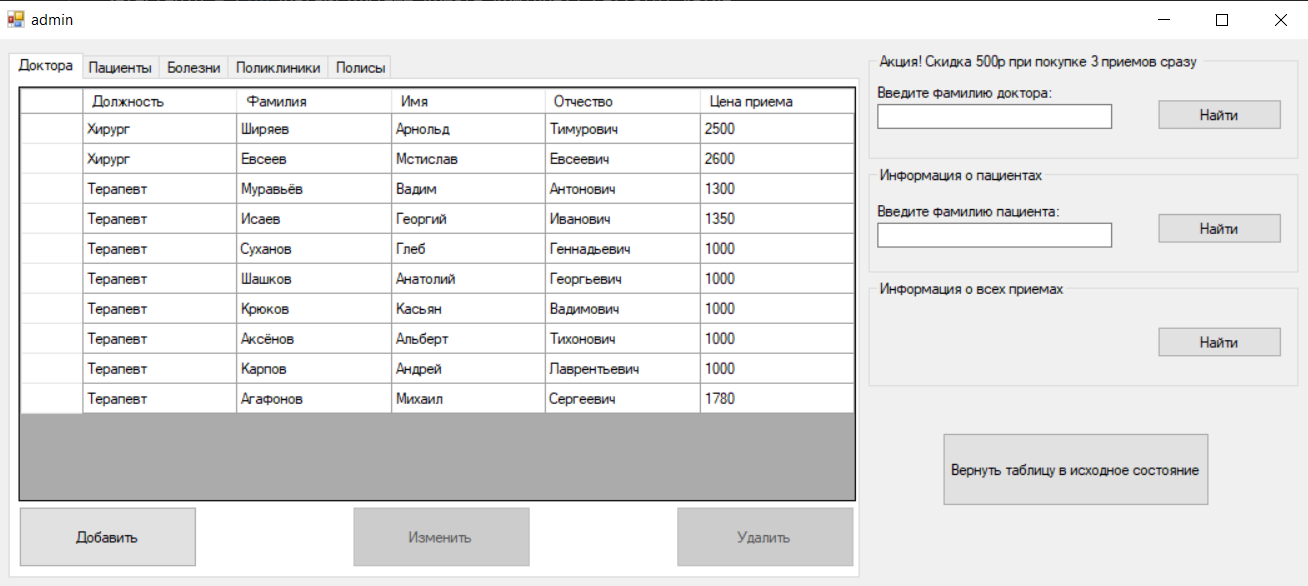


Рисунок 16 – Окно админа

Админ может просматривать больше данных и обладает некоторыми возможностями.

Для добавления нового доктора нужно нажать кнопку «Добавить»

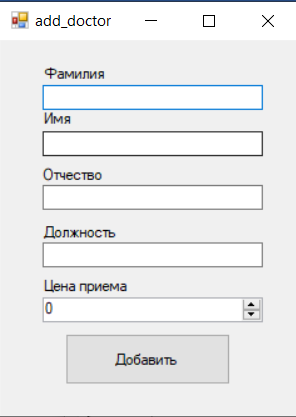


Рисунок 17 – Форма добавления доктора

После ввода значений в соответствующие поля, нужно нажать на кнопку «Добавить» после чего данные добавятся в базу данных и отобразятся в таблице [11)].

Для удаления данных необходимо нажать на поле в таблице, которое нужно удалить, после чего разблокируется кнопка «Удалить», после этого можно будет удалить запись.

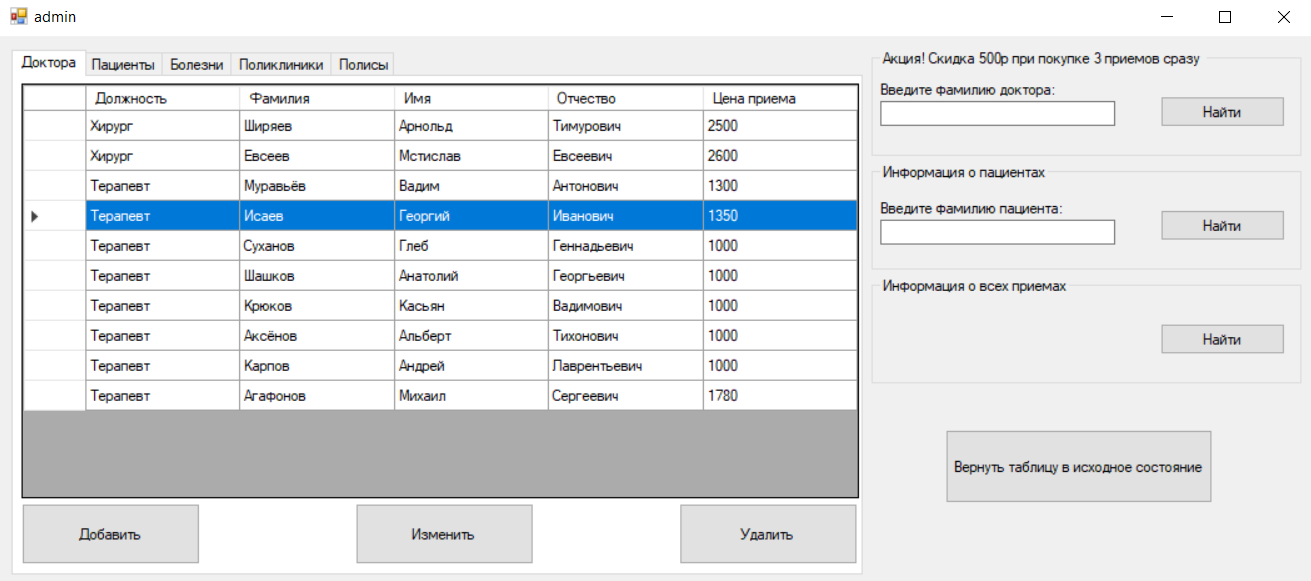


Рисунок 18 – Пример разблокировки кнопок изменения и удаления

Для изменения данных необходимо нажать на поле в таблице, которое нужно изменить, после чего разблокируется кнопка «Изменить», после нажатия которой, откроется форма изменения, в которой данные будут уже вписаны.

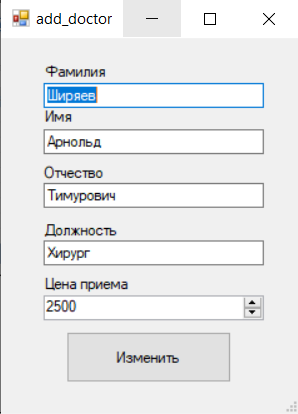


Рисунок 19 – Форма изменения

Так же по фамилии доктора можно узнать цену трех приемов с учетом скидки 500 рублей

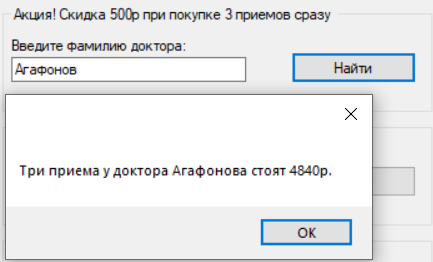


Рисунок 20 – Пример использования акции

Далее при вводе фамилии уже пациента можно узнать информацию о нем. Указаны такие столбцы как: «Пациенты», «Доктора», «Номер телефона пациента», «Серия и номер паспорта пациента» и «Страховой компании пациента».

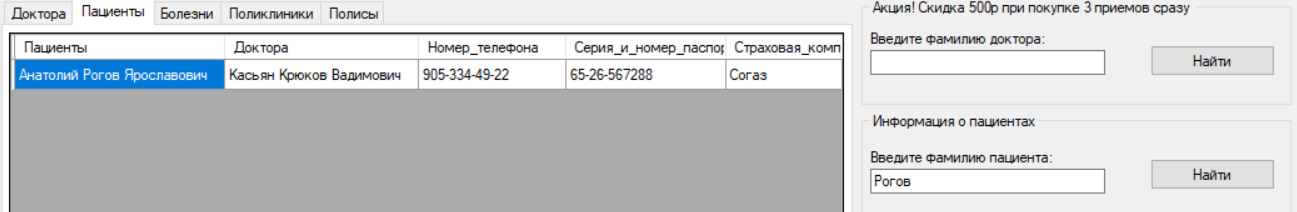


Рисунок 21 – Пример поиска информации о пациенте

Есть возможность просмотреть информацию о всех существующих приемах в базе данных.



Рисунок 22 – Вывод информации о всех приемах

Для очистки поля вывода информации в исходный вид используется кнопка «Вернуть таблицу в исходное состояние».

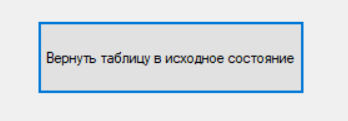


Рисунок 23 – Кнопка возврата таблицы в исходное состояние

В вкладках «Пациенты», «Болезни», «Поликлиники» и «Полисы» можно осуществлять, добавление, изменение, удаление и просмотр данных по алгоритму, описанному выше.

## 5.2 Пользовательский интерфейс

При открытии приложения пользователю необходимо пройти регистрацию или авторизацию, если он есть в системе. При регистрации пользователь должен ввести в первое поле логин, во второе пароль.

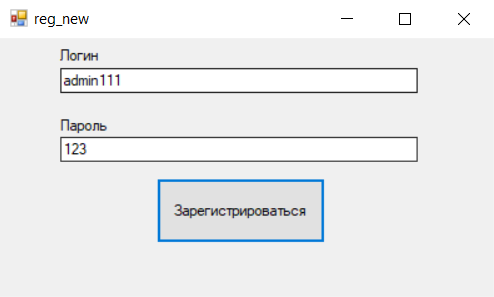


Рисунок 24 – Пример регистрации

После нажатия на кнопку «Зарегистрироваться» в системе проходит проверка, если такие данные в базе данных уже существуют, то пользователь видит сообщение о существующем пользователе.

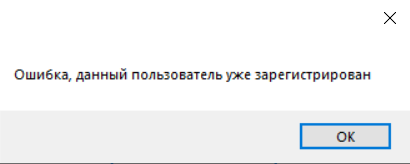


Рисунок 25 – Сообщение о уже существующем пользователе

Если логина и пароля в базе данных не существует, то регистрация прошла успешна и пользователь получит сообщении о завершении регистрации.

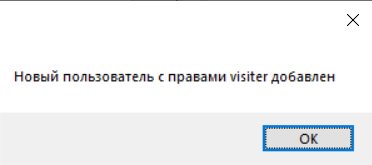


Рисунок 26 – Сообщение о успешной регистрации

После прохождения регистрации нужно авторизоваться в системе, для этого необходимо вести логин и пароль, после чего система проверит корректный ввод и при несовпадении выдаст сообщение что вход не выполнен.

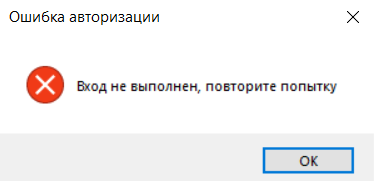


Рисунок 27 – Сообщение о ошибке авторизации

Если данные были введены правильно, тогда пользователь переходит на главную форму. Поскольку в данном приложение две главные роли, то при авторизации выясняется роль, которая была присвоена пользователю, который авторизовался и в зависимости от этого открывается соответствующая ему форма.

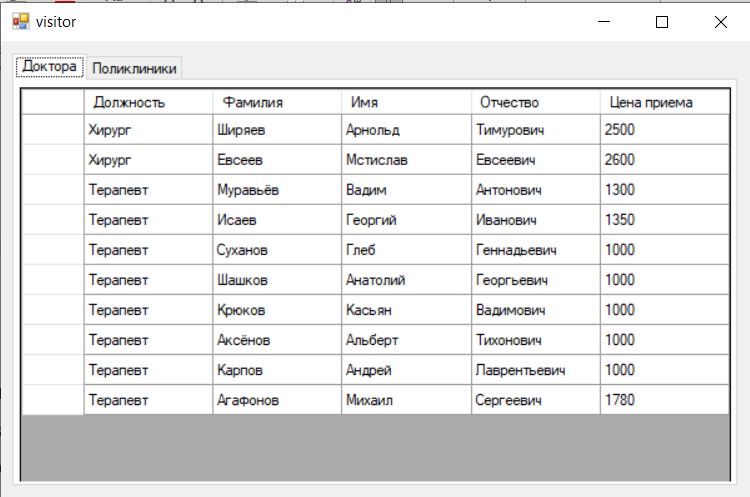


Рисунок 28 – Форма обычного пользователя

Данная форма открывается если пользователь имеет роль обычного пользователя и может только просматривать данные, но не изменять их, добавлять или удалять их.

Для выхода из данной учетной записи требуется закрыть программу, после чего перед пользователем появится окно авторизации. В данной программе существует роль админа, которая обладает практически всеми правами, главное окно выглядит следующим образом.

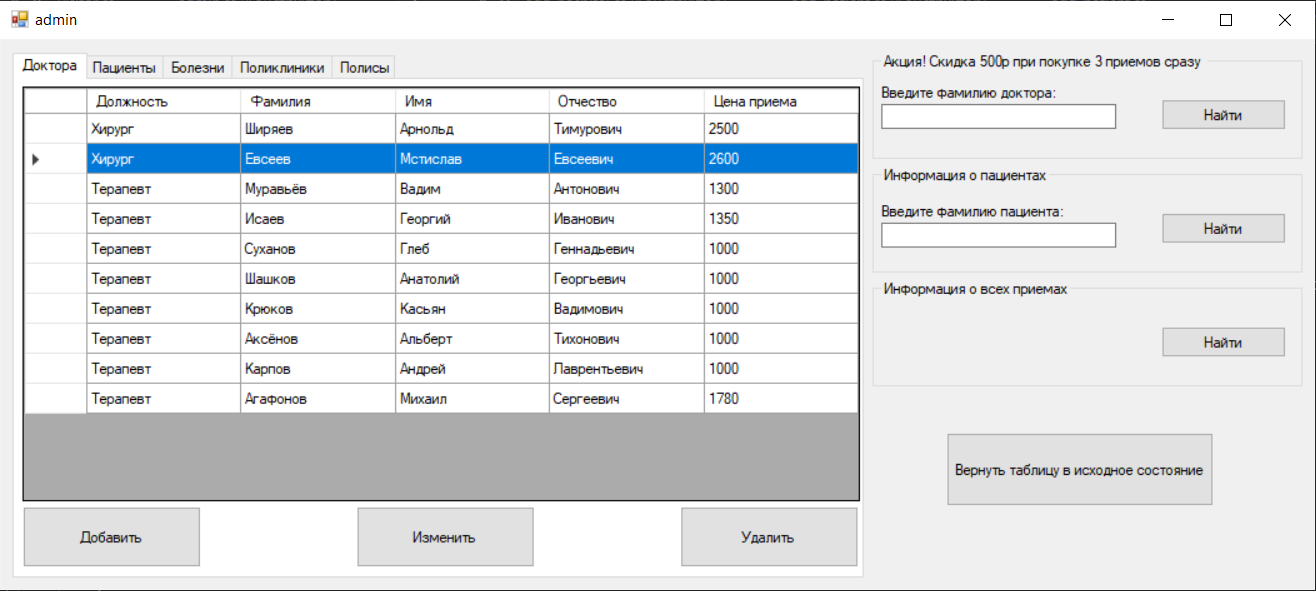


Рисунок 29 – Форма пользователя админа

Данная роль может добавлять, просматривать, удалять, изменять данные в любой таблице, так же вызывать скалярную функцию, хранимую процедуру и представление для облегчения работы с данными.

При добавлении данных с помощью кнопки «Добавить», следует заполнять все поля, иначе же пользователь получит сообщение с требованием заполнить поля.

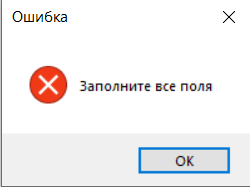


Рисунок 30 – Форма с требованием заполнить поля

В вкладке пациентов подобное добавление чуть иное, первая цифра в поле номера телефона не может быть нулем.

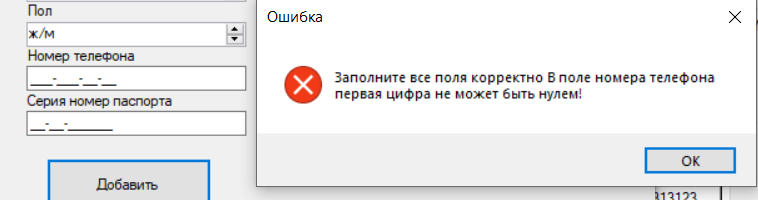


Рисунок 31 – Форма с требованием заполнить поля корректно

Для того чтобы выполнить удаление нужно нажать на строку, которую нужно удалить, после чего кнопка «Удалить» станет активной. При её выполнении программа получает ID выбранной строки, который является скрытым и выполняет удаление строки из таблицы, также удаляются строки из других таблиц, где есть соответствующий ID выбранной таблицы.

При выполнении изменения нужно выбрать и нажать кнопку «Изменить». После произойдёт открытие окна с вписанными данными выбранной строки. Данная операция происходит также путём получения ID выбранной строки, только в данном случае достаются построчно данные и записываются в соответствующие поля открываемого окна.

В остальных вкладках те же функции производятся аналогично.

Скалярная функция при вводе фамилии доктора выводит цену трех его платных посещений со скидкой в 500 рублей.

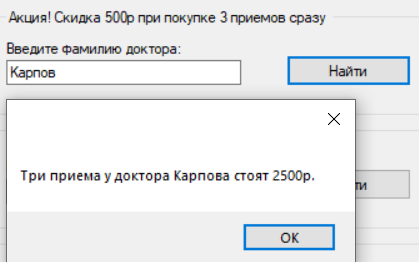


Рисунок 32 – Реализация акции

Если в поле ввода фамилии доктора написать несуществующую фамилию, то пользователь получит сообщении об этом.

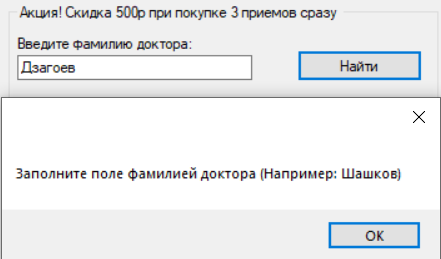


Рисунок 33 – Ввод несуществующего доктора

На других вкладках функций столько же, каждое из которых содержит проверку на пустые поля, а также свои проверки для корректного добавления или изменения данных, соответственно добавление, изменение и удаление данных из таблицы

# Заключение

В ходе выполнения данной работы был рассмотрен теоретически материал для создания базы данных больницы и последующей её внедрения в пользовательское приложения, для простого взаимодействия.

Целью данной работы было разработать базы данных на тему «Медицинские услуги. Больной, болезнь, история: разработка и администрирование базы данных, разработка клиентского приложения.

Цель была достигнута путём успешного выполнения основных задач курсовой работы: создание ER-модели реляционной модели данных, создание диаграммы Мартина, создания триггеров, хранимых процедур, представления и разработка клиентского приложения с внедрением в него ранее разработанную базу данных[2)].

В процессе выполнения работы были приобретены навыки анализа производства, обретены и применены полученные знания в области разработки системы управления базами данных для решения конкретной задачи с последующем внедрением её в клиентское приложение [10)].

Разработанная база данных может существенно облегчить работу сотрудников больниц, а также упростить процедуру получения информации, необходимой для сотрудников.

# Список использованных источников

1. Руководство по Entity Framework Core 6 // METANIT URL: <https://metanit.com/sharp/entityframework/1.1.php> (Дата обращения: 20.02.2023).
2. Реляционные базы данных и язык SQL // METANIT URL: <https://metanit.com/sql/> (Дата обращения: 28.02.2023).
3. Руководство по ADO.NET и работе с базами данных в .NET 6 // METANIT URL: <https://metanit.com/sharp/adonetcore/> (Дата обращения: 09.03.2023).
4. Руководство по MS SQL Server 2022 // METANIT URL: <https://metanit.com/sql/sqlserver/> (Дата обращения 18.02.2023)
5. Камалиева Ирина РинатовнаДоверие в модели отношения «врач – пациент» в современной медицине URL: <http://philsoc.psu.ru/issue-1/214-nauka/vestnik-2019/filosofiya/1329-doverie-kak-osnovanie-otnosheniya-vrach-patsient-v-sovremennoj-meditsine-filosofsko-antropologicheskaya-interpretatsiya> (Дата обращения: 18.03.2023).
6. Юридический отдел «Факультета медицинского права» Назначение и рекомендации пациенту URL: <https://kormed.ru/baza-znaniy/pravila-okazaniya-meduslug/rekomendacii-i-naznacheniya-pacientu/rekomendacii-i-naznacheniya-pacientu/> (Дата обращения: 29.03.2023)
7. Ишкова, Э. А. Самоучитель С#. Начала программирования / Э.А. Ишкова. - М.: Наука и техника, 2013. - 496 c. (Дата обращения: 27.03.2023)
8. Медицинские базы данных URL: <http://ilab.xmedtest.net/?q=node/4185> (Дата обращения: 23.03.2023).
9. Глушков СВ., Ломотько Д.В. Базы данных: Учебный курс. - Харьков: Фолно; М.: «Издательство ACT», 2003 (Дата обращения: 27.03.2023).
10. Крёнке Д. Теория и практика построения баз данных. - СПб: Питер, 2003. (Дата обращения: 28.03.2023).
11. Карпова Т.С. Базы данных: модели, разработка, реализация. - СПб: Питер, 2002 (Дата обращения: 20.03.2023).

# Приложение А

Создание таблицы doctor

CREATE TABLE doctor

(

ID\_doctor INT NOT NULL IDENTITY(1,1) PRIMARY KEY,

post VARCHAR(40) NOT NULL,

surname VARCHAR(40) NOT NULL,

[name] VARCHAR(30) NOT NULL ,

middle\_name VARCHAR(50) ,

one\_price int not null

)

Создание таблицы policy

CREATE TABLE [policy]

(

ID\_policy INT NOT NULL IDENTITY(1,1) PRIMARY KEY,

insurance\_company VARCHAR(50) NOT NULL,

)

Создание таблицы patient

CREATE TABLE patient

(

ID\_patient INT NOT NULL IDENTITY(1,1) PRIMARY KEY,

gender varchar (1) NOT NULL check (gender in ('м', 'М', 'ж', 'Ж')),

surname VARCHAR(40) NOT NULL,

[name] VARCHAR(30) NOT NULL ,

middle\_name VARCHAR(50)

age int NOT NULL,

phone\_number char (13) NOT NULL check (phone\_number like '[1-9][0-9][0-9]-[0-9][0-9][0-9]-[0-9][0-9]-[0-9][0-9]'),

passport\_seria\_nomer char (12) NOT NULL check (passport\_seria\_nomer like '[0-9][0-9]-[0-9][0-9]-[0-9][0-9][0-9][0-9][0-9][0-9]'),

ID\_hospital int not null

CONSTRAINT fk\_ID\_hospital FOREIGN KEY (ID\_hospital) REFERENCES hospital,

ID\_policy int not null

CONSTRAINT fk\_ID\_policy FOREIGN KEY (ID\_policy) REFERENCES [policy]

)

Создание таблицы medical\_history

CREATE TABLE medical\_history

(

ID\_medical\_card INT NOT NULL IDENTITY(1,1) PRIMARY KEY,

number\_electronic\_medical\_card INT NOT NULL,

)

Создание таблицы hospital

CREATE TABLE hospital

(

ID\_hospital INT NOT NULL IDENTITY(1,1) PRIMARY KEY,

city VARCHAR(50) NOT NULL,

street VARCHAR(50) NOT NULL,

number\_building VARCHAR(50) NOT NULL,

)

Создание таблицы doctor\_patient\_medical\_history

CREATE TABLE doctor\_patient\_medical\_history

(

ID\_doctor\_patient\_medical\_history INT NOT NULL IDENTITY(1,1) PRIMARY KEY,

ID\_medical\_card INT NOT NULL,

CONSTRAINT fk\_ID\_medical\_card FOREIGN KEY (ID\_medical\_card) REFERENCES medical\_history

)

Создание таблицы doctor\_patient\_disease

CREATE TABLE doctor\_patient\_disease

(

ID\_doctor\_patient\_disease INT NOT NULL IDENTITY(1,1) PRIMARY KEY,

ID\_disease INT NOT NULL,

CONSTRAINT fk\_ID\_disease FOREIGN KEY (ID\_disease) REFERENCES disease

)

Создание таблицы doctor\_patient

CREATE TABLE doctor\_patient

(

ID\_doctor\_patient INT NOT NULL IDENTITY(1,1) PRIMARY KEY,

ID\_doctor INT NOT NULL,

CONSTRAINT fk\_ID\_doctor FOREIGN KEY (ID\_doctor) REFERENCES doctor,

ID\_doctor\_patient\_medical\_history INT NOT NULL,

CONSTRAINT fk\_ID\_doctor\_patient\_medical\_history FOREIGN KEY (ID\_doctor\_patient\_medical\_history) REFERENCES doctor\_patient\_medical\_history,

ID\_patient INT NOT NULL,

CONSTRAINT fk\_ID\_patient FOREIGN KEY (ID\_patient) REFERENCES patient,

ID\_doctor\_patient\_disease INT NOT NULL,

CONSTRAINT fk\_ID\_doctor\_patient\_disease FOREIGN KEY (ID\_doctor\_patient\_disease) REFERENCES doctor\_patient\_disease

)

Создание таблицы desease

CREATE TABLE disease

(

ID\_disease INT NOT NULL IDENTITY(1,1) PRIMARY KEY,

date\_of\_disease date NOT NULL,

temperature decimal (3,1)f

)

# Приложение Б

Заполнение таблицы doctor

insert into doctor (post, surname, [name], middle\_name, price\_one)

values ('Хирург', 'Ширяев', 'Арнольд', 'Тимурович', '2500')

insert into doctor (post, surname, [name], middle\_name, price\_one)

values ('Хирург', 'Евсеев', 'Мстислав', 'Евсеевич', '2600')

insert into doctor (post, surname, [name], middle\_name, price\_one)

values ('Терапевт', 'Муравьёв', 'Вадим', 'Антонович', '1300')

insert into doctor (post, surname, [name], middle\_name, price\_one)

values ('Терапевт', 'Исаев ', 'Георгий ', 'Иванович', '1350')

insert into doctor (post, surname, [name], middle\_name, price\_one)

values ('Терапевт', 'Суханов', 'Глеб ', 'Геннадьевич', '1000')

insert into doctor (post, surname, [name], middle\_name, price\_one)

values ('Терапевт', 'Шашков ', 'Василий ', 'Георгьевич', '1000')

insert into doctor (post, surname, [name], middle\_name, price\_one)

values ('Терапевт', 'Крюков', 'Касьян ', 'Вадимович', '1000')

insert into doctor (post, surname, [name], middle\_name, price\_one)

values ('Терапевт', 'Аксёнов', 'Альберт', 'Тихонович', '1000')

insert into doctor (post, surname, [name], middle\_name, price\_one)

values ('Терапевт', 'Карпов', 'Адриан ', 'Лаврентьевич', '1000')

insert into doctor (post, surname, [name], middle\_name, price\_one)

values ('Терапевт', 'Агафонов ', 'Михаил ', 'Сергеевич', '1780')

Заполнение таблицы policy

insert into [policy] (insurance\_company)

values ('Ингосстрах')

insert into [policy] (insurance\_company)

values ('Ингосстрах')

insert into [policy] (insurance\_company)

values ('Ингосстрах')

insert into [policy] (insurance\_company)

values ('Ингосстрах')

insert into [policy] (insurance\_company)

values ('Ингосстрах')

insert into [policy] (insurance\_company)

values ('Ингосстрах')

insert into [policy] (insurance\_company)

values ('Согаз')

insert into [policy] (insurance\_company)

values ('Согаз')

insert into [policy] (insurance\_company)

values ('Согаз')

insert into [policy] (insurance\_company)

values ('Согаз')

Заполнение таблицы patient

insert into patient (gender, surname, [name], middle\_name, age, phone\_number, passport\_seria\_nomer, ID\_hospital, ID\_policy)

values ('м', 'Савельев', 'Гордей', 'Петрович', '32', '919-002-39-93', '39-24-754893', 1, 1)

insert into patient (gender, surname, [name], middle\_name, age, phone\_number, passport\_seria\_nomer, ID\_hospital, ID\_policy)

values ('м', 'Николаев ', 'Ярослав ', 'Кириллович', '39', '915-142-29-42', '25-64-789893', 1, 1)

insert into patient (gender, surname, [name], middle\_name, age, phone\_number, passport\_seria\_nomer, ID\_hospital, ID\_policy)

values ('м', 'Голубев ', 'Андрей ', 'Васильевич', '29', '910-742-29-43', '69-34-798753', 1, 1)

insert into patient (gender, surname, [name], middle\_name, age, phone\_number, passport\_seria\_nomer, ID\_hospital, ID\_policy)

values ('м', 'Зайцев ', 'Панкратий ', 'Христофорович', '53', '915-773-29-58', '23-77-249191', 1, 1)

insert into patient (gender, surname, [name], middle\_name, age, phone\_number, passport\_seria\_nomer, ID\_hospital, ID\_policy)

values ('м', 'Ширяев ', 'Ярослав ', 'Альбертович', '33', '905-345-79-82', '45-24-787351', 1, 1)

insert into patient (gender, surname, [name], middle\_name, age, phone\_number, passport\_seria\_nomer, ID\_hospital, ID\_policy)

values ('м', 'Нестеров ', 'Лаврентий ', 'Федорович', '39', '915-987-49-74', '25-64-875392', 1, 1)

insert into patient (gender, surname, [name], middle\_name, age, phone\_number, passport\_seria\_nomer, ID\_hospital, ID\_policy)

values ('м', 'Рогов ', 'Анатолий ', 'Ярославович', '36', '905-334-49-22', '65-26-567288', 1, 1)

insert into patient (gender, surname, [name], middle\_name, age, phone\_number, passport\_seria\_nomer, ID\_hospital, ID\_policy)

values ('м', 'Воробьёв ', 'Андрей ', 'Васильевич', '63', '900-985-29-64', '73-63-876535', 1, 1)

insert into patient (gender, surname, [name], middle\_name, age, phone\_number, passport\_seria\_nomer, ID\_hospital, ID\_policy)

values ('м', 'Ильин ', 'Гордей ', 'Русланович', '32', '955-373-73-42', '62-86-656847', 1, 1)

insert into patient (gender, surname, [name], middle\_name, age, phone\_number, passport\_seria\_nomer, ID\_hospital, ID\_policy)

values ('м', 'Николаев ', 'Ибрагим', 'Русланович', '51', '900-935-39-21', '26-67-153315', 1, 1)

Заполнение таблицы medical\_history

insert into medical\_history (number\_electronic\_medical\_card)

values (1)

insert into medical\_history (number\_electronic\_medical\_card)

values (2)

insert into medical\_history (number\_electronic\_medical\_card)

values (3)

insert into medical\_history (number\_electronic\_medical\_card)

values (4)

insert into medical\_history (number\_electronic\_medical\_card)

values (5)

insert into medical\_history (number\_electronic\_medical\_card)

values (6)

insert into medical\_history (number\_electronic\_medical\_card)

values (7)

insert into medical\_history (number\_electronic\_medical\_card)

values (8)

insert into medical\_history (number\_electronic\_medical\_card)

values (9)

insert into medical\_history (number\_electronic\_medical\_card)

values (10)

Заполнение таблицы hospital

insert into hospital (street, number\_building, city)

values ('Оружейный переулок', '43', 'Владимир')

insert into hospital (street, number\_building, city)

values ('Оружейный переулок', '43', 'Владимир')

insert into hospital (street, number\_building, city)

values ('Оружейный переулок', '43', 'Владимир')

insert into hospital (street, number\_building, city)

values ('Оружейный переулок', '43', 'Владимир')

insert into hospital (street, number\_building, city)

values ('Оружейный переулок', '43', 'Владимир')

insert into hospital (street, number\_building, city)

values ('Оружейный переулок', '43', 'Владимир')

insert into hospital (street, number\_building, city)

values ('Оружейный переулок', '43', 'Владимир')

insert into hospital (street, number\_building, city)

values ('Оружейный переулок', '43', 'Владимир')

insert into hospital (street, number\_building, city)

values ('Оружейный переулок', '43', 'Владимир')

insert into hospital (street, number\_building, city)

values ('Оружейный переулок', '43', 'Владимир')

Заполнение таблицы doctor\_patient\_medical\_history

insert into doctor\_patient\_medical\_history (ID\_medical\_card)

values (1)

insert into doctor\_patient\_medical\_history (ID\_medical\_card)

values (2)

insert into doctor\_patient\_medical\_history (ID\_medical\_card)

values (3)

insert into doctor\_patient\_medical\_history (ID\_medical\_card)

values (4)

insert into doctor\_patient\_medical\_history (ID\_medical\_card)

values (5)

insert into doctor\_patient\_medical\_history (ID\_medical\_card)

values (6)

insert into doctor\_patient\_medical\_history (ID\_medical\_card)

values (7)

insert into doctor\_patient\_medical\_history (ID\_medical\_card)

values (8)

insert into doctor\_patient\_medical\_history (ID\_medical\_card)

values (9)

insert into doctor\_patient\_medical\_history (ID\_medical\_card)

values (10)

Заполнение таблицы doctor\_patient\_disease

insert into doctor\_patient\_disease (ID\_disease)

values (1)

insert into doctor\_patient\_disease (ID\_disease)

values (2)

insert into doctor\_patient\_disease (ID\_disease)

values (3)

insert into doctor\_patient\_disease (ID\_disease)

values (4)

insert into doctor\_patient\_disease (ID\_disease)

values (5)

insert into doctor\_patient\_disease (ID\_disease)

values (6)

insert into doctor\_patient\_disease (ID\_disease)

values (7)

insert into doctor\_patient\_disease (ID\_disease)

values (8)

insert into doctor\_patient\_disease (ID\_disease)

values (9)

insert into doctor\_patient\_disease (ID\_disease)

values (10)

Заполнение таблицы doctor\_patient

insert into doctor\_patient (ID\_doctor, ID\_doctor\_patient\_medical\_history, ID\_patient, ID\_doctor\_patient\_disease)

values (1, 1, 21, 31)

insert into doctor\_patient (ID\_doctor, ID\_doctor\_patient\_medical\_history, ID\_patient, ID\_doctor\_patient\_disease)

values (2, 2, 22, 32)

insert into doctor\_patient (ID\_doctor, ID\_doctor\_patient\_medical\_history, ID\_patient, ID\_doctor\_patient\_disease)

values (3, 3, 23, 33)

insert into doctor\_patient (ID\_doctor, ID\_doctor\_patient\_medical\_history, ID\_patient, ID\_doctor\_patient\_disease)

values (4, 4, 24, 34)

insert into doctor\_patient (ID\_doctor, ID\_doctor\_patient\_medical\_history, ID\_patient, ID\_doctor\_patient\_disease)

values (5, 5, 25, 35)

insert into doctor\_patient (ID\_doctor, ID\_doctor\_patient\_medical\_history, ID\_patient, ID\_doctor\_patient\_disease)

values (6, 6, 26, 36)

insert into doctor\_patient (ID\_doctor, ID\_doctor\_patient\_medical\_history, ID\_patient, ID\_doctor\_patient\_disease)

values (7, 7, 27, 37)

insert into doctor\_patient (ID\_doctor, ID\_doctor\_patient\_medical\_history, ID\_patient, ID\_doctor\_patient\_disease)

values (8, 8, 28, 38)

insert into doctor\_patient (ID\_doctor, ID\_doctor\_patient\_medical\_history, ID\_patient, ID\_doctor\_patient\_disease)

values (9, 9, 29, 39)

insert into doctor\_patient (ID\_doctor, ID\_doctor\_patient\_medical\_history, ID\_patient, ID\_doctor\_patient\_disease)

values (10, 10, 30, 40)

Заполнение таблицы desease

insert into disease (date\_of\_disease, temperature)

values ('2023-01-11', 36.5)

insert into disease (date\_of\_disease, temperature)

values ('2023-01-2', 36.9)

insert into disease (date\_of\_disease, temperature)

values ('2023-01-13', 38.4)

insert into disease (date\_of\_disease, temperature)

values ('2023-02-1', 37.9)

insert into disease (date\_of\_disease, temperature)

values ('2023-01-21', 36.5)

insert into disease (date\_of\_disease, temperature)

values ('2023-01-26', 38.9)

insert into disease (date\_of\_disease, temperature)

values ('2023-01-19', 38.1)

insert into disease (date\_of\_disease, temperature)

values ('2023-02-15', 40.0)

insert into disease (date\_of\_disease, temperature)

values ('2023-01-20', 38.2)

insert into disease (date\_of\_disease, temperature)

values ('2023-02-5', 39.8)

# Приложение В

Создание функции

alter FUNCTION fio\_doctor1 (@fio varchar(155))

RETURNS int

begin

declare @p int

select @p = (select d.price\_one) from doctor d

where d.surname like @fio

set @p = @p \* 3 - 500

return @p

end

Создание процедуры

alter view [all] as

(

select distinct 'Доктор' = d.surname + ' ' + d.[name] + ' ' + d.middle\_name,

'Должность' = d.post,

'Пациент' = p.middle\_name + ' ' + p.[name] + ' ' + p.middle\_name,

'Страховщик' = po.insurance\_company,

'Адрес больницы' = 'г.' + h.city + ' ул.' + h.street + ' д.' + h.number\_building

from doctor d, patient p, [policy] po, doctor\_patient dp, hospital h

where p.ID\_policy = po.ID\_policy and dp.ID\_doctor = d.ID\_doctor and dp.ID\_patient = p.ID\_patient

and h.ID\_hospital = p.ID\_hospital

)

Создание представления

alter view [all] as

(

select distinct 'Доктор' = d.surname + ' ' + d.[name] + ' ' + d.middle\_name,

'Должность' = d.post,

'Пациент' = p.middle\_name + ' ' + p.[name] + ' ' + p.middle\_name,

'Страховщик' = po.insurance\_company,

'Адрес больницы' = 'г.' + h.city + ' ул.' + h.street + ' д.' + h.number\_building

from doctor d, patient p, [policy] po, doctor\_patient dp, hospital h

where p.ID\_policy = po.ID\_policy and dp.ID\_doctor = d.ID\_doctor and dp.ID\_patient = p.ID\_patient

and h.ID\_hospital = p.ID\_hospital

)

Триггер insert

create trigger insert\_trig on doctor

after insert as

begin

insert into History

SELECT ID\_doctor, post, surname, [name], middle\_name, price\_one, SUSER\_NAME(),CONVERT(varchar,'ADD',1),GETDATE()

from inserted

end;

Триггер delete

alter trigger del\_rollback on doctor

after delete as

begin

insert into History

SELECT ID\_doctor, post, surname, [name], middle\_name, price\_one, SUSER\_NAME(),CONVERT(varchar,'DELETE',1),GETDATE()

from deleted

print 'Удаление'

end;

Триггер update

alter trigger [update] on doctor

after update as

begin

insert into History

SELECT ID\_doctor, post, surname, [name], middle\_name, price\_one, SUSER\_NAME(),CONVERT(varchar,'UPDATE',1),GETDATE()

from INSERTED

End

Создание роли visiter

exec sp\_addrole 'visiter'

GRANT select

on doctor to visiter

GRANT select

on hospital to visiter

Создание роли sysAdm

exec sp\_addrole 'sysAdm1'

GRANT select, update , insert , delete

on doctor to sysAdm1

GRANT select, update , insert , delete

on disease to sysAdm1

GRANT select, update , insert , delete

on doctor\_patient to sysAdm1

GRANT select, update , insert , delete

on doctor\_patient\_disease to sysAdm1

GRANT select, update , insert , delete

on doctor\_patient\_medical\_history to sysAdm1

GRANT select, update , insert , delete

on hospital to sysAdm1

GRANT select, update , insert , delete

on medical\_history to sysAdm1

GRANT select, update , insert , delete

on patient to sysAdm1

GRANT select, update , insert , delete

on [policy] to sysAdm1

grant select on dbo.fio\_doctor1 to sysAdm1

grant execute on dbo.count\_patient to sysAdm1

grant select on dbo.[all] to sysAdm1

Создание пользователя admin1

exec sp\_addlogin 'admin111', '123' , 'Med\_Yslugi'

use Med\_Yslugi

exec sp\_adduser 'admin111' , 'admin111'

exec sp\_addrolemember 'sysAdm1' , 'admin111'

Создание пользователя visiter1

exec sp\_addlogin 'VISITER1', '321' , 'Med\_Yslugi'

use Med\_Yslugi

exec sp\_adduser 'VISITER1' , 'VISITER1'

exec sp\_addrolemember 'visiter' , 'VISITER1'

# Приложение Г

Код главной формы

Код главной формы обычного пользователя

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 Med\_Yslugi

{

public partial class visitor : Form

{

public string connectionString;

public visitor()

{

InitializeComponent();

}

public visitor(string con)

{

InitializeComponent();

connectionString = con;

}

public void UpdateDataGrid()

{

using (context db = new context(connectionString))

{

var doctors = db.doctors;

dataGridView1.DataSource = doctors.ToList();

dataGridView1.Columns[0].Visible = false;

dataGridView1.Columns[6].Visible = false;

dataGridView1.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;

dataGridView1.Columns[1].HeaderText = "Должность";

dataGridView1.Columns[2].HeaderText = "Фамилия";

dataGridView1.Columns[3].HeaderText = "Имя";

dataGridView1.Columns[4].HeaderText = "Отчество";

dataGridView1.Columns[5].HeaderText = "Цена приема";

var hospitals = db.hospitals;

dataGridView2.DataSource = hospitals.ToList();

dataGridView2.Columns[0].Visible = false;

dataGridView2.Columns[4].Visible = false;

dataGridView2.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;

dataGridView2.Columns[1].HeaderText = "Улица";

dataGridView2.Columns[2].HeaderText = "Номер дома";

dataGridView2.Columns[3].HeaderText = "Город";

}

}

private void visitor\_FormClosed(object sender, FormClosedEventArgs e)

{

Application.Exit();

}

private void visitor\_Load(object sender, EventArgs e)

{

UpdateDataGrid();

}

}

}

Код главной формы администратора больницы

using Med\_Yslugi.add;

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.Entity;

using System.Data.SqlClient;

using System.Linq;

using System.Reflection.Emit;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace Med\_Yslugi

{

public partial class admin : Form

{

public admin()

{

InitializeComponent();

}

public string connectionString;

public admin(string con)

{

InitializeComponent();

connectionString = con;

}

public int? SelectedId = null;

public string SelectedTitle = null;

public void UpdateDataGrid()

{

using (context db = new context(connectionString))

{

var doctors = db.doctors;

dataGridView1.DataSource = doctors.ToList();

dataGridView1.Columns[0].Visible = false;

dataGridView1.Columns[6].Visible = false;

dataGridView1.Columns[1].HeaderText = "Должность";

dataGridView1.Columns[2].HeaderText = "Фамилия";

dataGridView1.Columns[3].HeaderText = "Имя";

dataGridView1.Columns[4].HeaderText = "Отчество";

dataGridView1.Columns[5].HeaderText = "Цена приема";

var patients = db.patients;

dataGridView2.DataSource = patients.ToList();

dataGridView2.Columns[0].Visible = false;

dataGridView2.Columns[8].Visible = false;

dataGridView2.Columns[9].Visible = false;

dataGridView2.Columns[10].Visible = false;

dataGridView2.Columns[11].Visible = false;

dataGridView2.Columns[12].Visible = false;

dataGridView2.Columns[1].HeaderText = "Пол";

dataGridView2.Columns[2].HeaderText = "Фамилия";

dataGridView2.Columns[3].HeaderText = "Имя";

dataGridView2.Columns[4].HeaderText = "Отчество";

dataGridView2.Columns[5].HeaderText = "Возраст";

dataGridView2.Columns[6].HeaderText = "Номер телефона";

dataGridView2.Columns[7].HeaderText = "Серия и номер паспорта";

var diseases = db.diseases;

dataGridView3.DataSource = diseases.ToList();

dataGridView3.Columns[0].Visible = false;

dataGridView3.Columns[3].Visible = false;

dataGridView3.Columns[1].HeaderText = "Дата заболевания";

dataGridView3.Columns[2].HeaderText = "Температура";

var hospitals = db.hospitals;

dataGridView4.DataSource = hospitals.ToList();

dataGridView4.Columns[0].Visible = false;

dataGridView4.Columns[4].Visible = false;

dataGridView4.Columns[1].HeaderText = "Улица";

dataGridView4.Columns[2].HeaderText = "Номер дома";

dataGridView4.Columns[3].HeaderText = "Город";

var policies = db.policies;

dataGridView5.DataSource = policies.ToList();

dataGridView5.Columns[0].Visible = false;

dataGridView5.Columns[2].Visible = false;

dataGridView5.Columns[1].HeaderText = "Страховая компания";

dataGridView1.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;

dataGridView2.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;

dataGridView3.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;

dataGridView4.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;

dataGridView5.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;

}

}

private void admin\_Load(object sender, EventArgs e)

{

UpdateDataGrid();

alter\_buttonClick.Enabled = false;

button3.Enabled = false;

button6.Enabled = false;

button9.Enabled = false;

button12.Enabled = false;

delete\_buttonClick.Enabled = false;

button2.Enabled = false;

button5.Enabled = false;

button8.Enabled = false;

button11.Enabled = false;

}

private void admin\_FormClosed(object sender, FormClosedEventArgs e)

{

Application.Exit();

}

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

{

add\_doctor add\_doctor = new add\_doctor(connectionString, this);

add\_doctor.button2.Visible = false;

add\_doctor.Show();

}

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

{

add\_patient add\_patient = new add\_patient(connectionString, this);

add\_patient.button2.Visible = false;

using (context db = new context(connectionString))

{

List<hospital> hos = db.hospitals.ToList();

add\_patient.comboBox1.DataSource = hos;

add\_patient.comboBox1.ValueMember = "ID\_hospital";

add\_patient.comboBox1.DisplayMember = "street";

List<policy> pol = db.policies.ToList();

add\_patient.comboBox2.DataSource = pol;

add\_patient.comboBox2.ValueMember = "ID\_policy";

add\_patient.comboBox2.DisplayMember = "insurance\_company";

}

add\_patient.Show();

}

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

{

add\_disease add\_disease = new add\_disease(connectionString, this);

add\_disease.button2.Visible = false;

add\_disease.Show();

}

private void button7\_Click(object sender, EventArgs e)

{

add\_hospital add\_hospital = new add\_hospital(connectionString, this);

add\_hospital.button2.Visible = false;

add\_hospital.Show();

}

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

{

add\_policy add\_policy = new add\_policy(connectionString, this);

add\_policy.button2.Visible = false;

add\_policy.Show();

}

private void dataGridView1\_CellClick(object sender, DataGridViewCellEventArgs e) //

{

try

{

SelectedId = (int)dataGridView1.Rows[e.RowIndex].Cells[0].Value;

SelectedTitle = (string)dataGridView1.Rows[e.RowIndex].Cells[1].Value;

alter\_buttonClick.Enabled = true;

delete\_buttonClick.Enabled = true;

}

catch { return; };

}

private void dataGridView2\_CellClick(object sender, DataGridViewCellEventArgs e)

{

try

{

SelectedId = (int)dataGridView2.Rows[e.RowIndex].Cells[0].Value;

SelectedTitle = (string)dataGridView2.Rows[e.RowIndex].Cells[1].Value;

button3.Enabled = true;

button2.Enabled = true;

}

catch { return; };

}

private void dataGridView3\_CellClick(object sender, DataGridViewCellEventArgs e)

{

try

{

SelectedId = (int)dataGridView3.Rows[e.RowIndex].Cells[0].Value;

SelectedTitle = (string)dataGridView3.Rows[e.RowIndex].Cells[1].Value;

button6.Enabled = true;

button5.Enabled = true;

}

catch { return; };

}

private void dataGridView4\_CellClick(object sender, DataGridViewCellEventArgs e)

{

try

{

SelectedId = (int)dataGridView4.Rows[e.RowIndex].Cells[0].Value;

SelectedTitle = (string)dataGridView4.Rows[e.RowIndex].Cells[1].Value;

button9.Enabled = true;

button8.Enabled = true;

}

catch { return; };

}

private void dataGridView5\_CellClick(object sender, DataGridViewCellEventArgs e)

{

try

{

SelectedId = (int)dataGridView5.Rows[e.RowIndex].Cells[0].Value;

SelectedTitle = (string)dataGridView5.Rows[e.RowIndex].Cells[1].Value;

button12.Enabled = true;

button11.Enabled = true;

}

catch { return; };

}

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

{

using (context db = new context(connectionString))

{

var delete = db.doctors.Where(d => d.ID\_doctor == SelectedId).FirstOrDefault();

db.doctors.Remove(delete);

db.SaveChanges();

UpdateDataGrid();

}

}

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

{

using (context db = new context(connectionString))

{

var delete = db.patients.Where(p => p.ID\_patient == SelectedId).FirstOrDefault();

db.patients.Remove(delete);

db.SaveChanges();

UpdateDataGrid();

}

}

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

{

using (context db = new context(connectionString))

{

var delete = db.diseases.Where(dis => dis.ID\_disease == SelectedId).FirstOrDefault();

db.diseases.Remove(delete);

db.SaveChanges();

UpdateDataGrid();

}

}

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

{

using (context db = new context(connectionString))

{

var delete = db.hospitals.Where(dis => dis.ID\_hospital == SelectedId).FirstOrDefault();

db.hospitals.Remove(delete);

db.SaveChanges();

UpdateDataGrid();

}

}

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

{

using (context db = new context(connectionString))

{

var delete = db.policies.Where(dis => dis.ID\_policy == SelectedId).FirstOrDefault();

db.policies.Remove(delete);

db.SaveChanges();

UpdateDataGrid();

}

}

private void alter\_buttonClick\_Click(object sender, EventArgs e)

{

add\_doctor fm = new add\_doctor(connectionString, this);

using (context db = new context(connectionString))

{

doctor doc = db.doctors.Find(SelectedId);

try

{

fm.textBox1.Text = doc.surname;

fm.textBox2.Text = doc.name;

fm.textBox3.Text = doc.middle\_name;

fm.textBox4.Text = doc.post;

fm.numericUpDown1.Value = doc.price\_one;

fm.button1.Visible = false;

}

catch { MessageBox.Show("Заполните поля корректно", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error); }

try

{

DialogResult dr = fm.ShowDialog();

if (dr == DialogResult.OK)

{

doc.surname = fm.textBox1.Text;

doc.name = fm.textBox2.Text;

doc.middle\_name = fm.textBox3.Text;

doc.post = fm.textBox4.Text;

doc.price\_one = (int)fm.numericUpDown1.Value;

db.SaveChanges();

}

UpdateDataGrid();

fm.Close();

}

catch

{

MessageBox.Show("Заполните поля корректно!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

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

{

add\_patient fm = new add\_patient(connectionString, this);

using (context db = new context(connectionString))

{

patient pat = db.patients.Find(SelectedId);

List<hospital> hos = db.hospitals.ToList();

fm.comboBox1.DataSource = hos;

fm.comboBox1.ValueMember = "ID\_hospital";

fm.comboBox1.DisplayMember = "street";

List<policy> pol = db.policies.ToList();

fm.comboBox2.DataSource = pol;

fm.comboBox2.ValueMember = "ID\_policy";

fm.comboBox2.DisplayMember = "insurance\_company";

try

{

fm.textBox1.Text = pat.surname;

fm.textBox2.Text = pat.name;

fm.textBox3.Text = pat.middle\_name;

fm.numericUpDown1.Value = pat.age;

fm.domainUpDown1.Text = pat.gender;

fm.maskedTextBox1.Text = pat.phone\_number;

fm.maskedTextBox2.Text = pat.passport\_seria\_nomer;

fm.comboBox1.SelectedIndex = pat.ID\_hospital - 1;

fm.comboBox2.SelectedIndex = pat.ID\_policy - 1;

fm.button1.Visible = false;

}

catch

{

MessageBox.Show("Заполните поля корректно! В поле номера телефона первая цифра не может быть нулем!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

try

{

DialogResult dr = fm.ShowDialog();

if (dr == DialogResult.OK)

{

hospital hos1 = fm.comboBox1.Items[fm.comboBox1.SelectedIndex] as hospital;

policy pol1 = fm.comboBox2.Items[fm.comboBox2.SelectedIndex] as policy;

pat.surname = fm.textBox1.Text;

pat.name = fm.textBox2.Text;

pat.middle\_name = fm.textBox3.Text;

pat.age = (int)fm.numericUpDown1.Value;

pat.gender = fm.domainUpDown1.Text;

pat.phone\_number = fm.maskedTextBox1.Text;

pat.passport\_seria\_nomer = fm.maskedTextBox2.Text;

pat.ID\_hospital = hos1.ID\_hospital;

pat.ID\_policy = pol1.ID\_policy;

db.SaveChanges();

}

UpdateDataGrid();

fm.Close();

}

catch

{

MessageBox.Show("Заполните поля корректно! В поле номера телефона первая цифра не может быть нулем!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

private void button6\_Click(object sender, EventArgs e)

{

add\_disease fm = new add\_disease(connectionString, this);

using (context db = new context(connectionString))

{

disease dis = db.diseases.Find(SelectedId);

try

{

fm.dateTimePicker1.Value = dis.date\_of\_disease;

fm.numericUpDown1.Value = (decimal)dis.temperature;

fm.button1.Visible = false;

}

catch

{

MessageBox.Show("Заполните поля корректно!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

try

{

DialogResult dr = fm.ShowDialog();

if (dr == DialogResult.OK)

{

dis.date\_of\_disease = fm.dateTimePicker1.Value;

dis.temperature = fm.numericUpDown1.Value;

db.SaveChanges();

}

UpdateDataGrid();

fm.Close();

}

catch

{

MessageBox.Show("Заполните поля корректно!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

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

{

add\_hospital fm = new add\_hospital(connectionString, this);

using (context db = new context(connectionString))

{

hospital hos = db.hospitals.Find(SelectedId);

try

{

fm.textBox1.Text = hos.street;

fm.numericUpDown1.Text = hos.number\_building;

fm.textBox2.Text = hos.city;

fm.button1.Visible = false;

}

catch

{

MessageBox.Show("Заполните поля корректно!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

try

{

DialogResult dr = fm.ShowDialog();

if (dr == DialogResult.OK)

{

hos.street = fm.textBox1.Text;

hos.number\_building = fm.numericUpDown1.Text;

hos.city = fm.textBox2.Text;

db.SaveChanges();

}

UpdateDataGrid();

fm.Close();

}

catch

{

MessageBox.Show("Заполните поля корректно!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

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

{

add\_policy fm = new add\_policy(connectionString, this);

using (context db = new context(connectionString))

{

policy pol = db.policies.Find(SelectedId);

try

{

fm.textBox1.Text = pol.insurance\_company;

fm.button1.Visible = false;

}

catch

{

MessageBox.Show("Заполните поля корректно!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

try

{

DialogResult dr = fm.ShowDialog();

if (dr == DialogResult.OK)

{

pol.insurance\_company = fm.textBox1.Text;

db.SaveChanges();

}

UpdateDataGrid();

fm.Close();

}

catch

{

MessageBox.Show("Заполните поля корректно!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

private void button13\_Click(object sender, EventArgs e)

{

using (context db = new context(connectionString))

{

try

{

var scal = db.Database.SqlQuery<int>($"select dbo.fio\_doctor1 (@name)", new SqlParameter("@name", textBox1.Text)).FirstOrDefault();

MessageBox.Show($"Три приема у доктора {textBox1.Text}а стоят {scal}р.");

}

catch { MessageBox.Show("Заполните поле фамилией доктора (Например: Шашков)"); }

//dataGridView1.DataSource = scal;

//dataGridView2.DataSource = scal;

//dataGridView3.DataSource = scal;

//dataGridView4.DataSource = scal;

//dataGridView5.DataSource = scal;

}

}

private void button14\_Click(object sender, EventArgs e)

{

using (context db = new context(connectionString))

{

try

{

var proc = db.Database.SqlQuery<Procedure>($"count\_patient @name", new SqlParameter("@name", textBox2.Text)).ToList();

dataGridView1.DataSource = proc;

dataGridView2.DataSource = proc;

dataGridView3.DataSource = proc;

dataGridView4.DataSource = proc;

dataGridView5.DataSource = proc;

}

catch { MessageBox.Show("Заполните поле фамилией пациента (Например: Савельев)"); }

}

}

private void button15\_Click(object sender, EventArgs e)

{

using (context db = new context(connectionString))

{

var view = db.all.ToList();

dataGridView1.DataSource = view;

dataGridView2.DataSource = view;

dataGridView3.DataSource = view;

dataGridView4.DataSource = view;

dataGridView5.DataSource = view;

}

}

private void button16\_Click(object sender, EventArgs e)

{

UpdateDataGrid();

}

}

}

Код формы добавления/редактирования

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Runtime.Remoting.Contexts;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Med\_Yslugi.add

{

public partial class add\_doctor : Form

{

public string connectionString;

admin admin;

public add\_doctor(string con,admin op)

{

InitializeComponent();

connectionString = con;

admin = op;

}

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

{

using (context db = new context(connectionString))

{

try

{

doctor d = new doctor()

{

name = textBox1.Text,

surname = textBox2.Text,

middle\_name = textBox3.Text,

post = textBox4.Text,

price\_one = (int)numericUpDown1.Value

};

db.doctors.Add(d);

db.SaveChanges();

admin.UpdateDataGrid();

MessageBox.Show("Запись добавлена");

Close();

}

catch { MessageBox.Show("Заполните все поля", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error); }

}

}

}

}

Код авторизации

using Med\_Yslugi.adm\_vis\_reg;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.Entity;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Med\_Yslugi

{

public partial class registration : Form

{

public registration()

{

InitializeComponent();

}

private enum Role

{

ADMIN,

PATIENT

}

Role role;

public static string GetRemoteConnectionString(string login, string password)

{

SqlConnectionStringBuilder sqlString = new SqlConnectionStringBuilder()

{

DataSource = $"(local)",

InitialCatalog = "Med\_Yslugi", //Database

IntegratedSecurity = false,

MultipleActiveResultSets = true,

ApplicationName = "EntityFramework",

UserID = login,

Password = password

};

return sqlString.ToString();

}

public string connecttionString;

public string mode;

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

{

connecttionString = GetRemoteConnectionString(textBox1.Text, textBox2.Text);

using (context db = new context(connecttionString))

{

try

{

if (db.Database.SqlQuery<int>("SELECT is\_member('VISITER')").FirstOrDefault() == 1)

mode = "Посетитель";

if (db.Database.SqlQuery<int>("SELECT is\_member('sysAdm1')").FirstOrDefault() == 1)

mode = "Админ";

switch (mode)

{

case "Админ":

Form admin = new admin(connecttionString);

this.Hide();

admin.Show();

break;

case "Посетитель":

Form visiter = new visitor(connecttionString);

this.Hide();

visiter.Show();

break;

}

}

catch

{

MessageBox.Show("Вход не выполнен, повторите попытку", "Ошибка авторизации", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

}

}

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

{

Form reg = new reg\_new(connecttionString);

this.Hide();

reg.Show();

}

}

}

Код регистрации

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Med\_Yslugi.adm\_vis\_reg

{

public partial class reg\_new : Form

{

public string connection;

public reg\_new(string con)

{

InitializeComponent();

connection = con;

}

public static string GetConnectionString(string login, string password)

{

SqlConnectionStringBuilder cx = new SqlConnectionStringBuilder()

{

DataSource = $"LAPTOP-CLVPR4Q1",

InitialCatalog = "Med\_Yslugi",

IntegratedSecurity = true,

MultipleActiveResultSets = true,

ApplicationName = "EntityFramework",

UserID = login,

Password = password

};

return cx.ToString();

}

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

{

try

{

string login = textBox1.Text;

string password = textBox2.Text;

if (textBox2.Text == String.Empty)

{

MessageBox.Show("Задайте логин и пароль!");

return;

};

connection = GetConnectionString("LAPTOP-CLVPR4Q1\\User", "");

SqlConnection con3 = new SqlConnection(connection);

SqlCommand cmd3 = new SqlCommand();

con3.Open();

cmd3.Connection = con3;

cmd3.CommandText = $"EXEC sp\_addlogin '{login}','{password}','Med\_Yslugi'";

cmd3.ExecuteNonQuery();

cmd3.CommandText = $"USE Med\_Yslugi";

cmd3.CommandText = $"EXEC sp\_adduser '{login}','{login}'";

cmd3.ExecuteNonQuery();

cmd3.CommandText = $"EXEC sp\_addrolemember 'visiter','{login}'";

cmd3.ExecuteNonQuery();

con3.Close();

MessageBox.Show("Новый пользователь с правами visiter добавлен");

Form reg = new registration();

reg.Show();

this.Close();

}

catch { MessageBox.Show("Ошибка, данный пользователь уже зарегистрирован"); }

}

private void reg\_new\_FormClosed(object sender, FormClosedEventArgs e)

{

Form reg = new registration();

Form reg2 = new reg\_new("");

reg2.Close();

}

}

}