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)**

Пояснительная записка к курсовой работе

по дисциплине

«Технология программирования»

по теме

Информационная система «Больница»

Выполнил: ст-т гр. ВТ-119
Безрядина А.В.
Принял: Сущинина А.А.

Владимир, 2022

Календарные сроки выполнения

1	20 %	4	недели
2	60 %	8	
3	80 %	12	
4	100 %	14	
5	%		
6	%		
7	%		
8	%		
9	%		
10	%		
11	%		
12	%		

Дата выдачи задания " 7 " сентября 2022 г.

Руководитель проектирования Сущинина А.А

Ход выполнения

Дата	4.10												
%выполнения	41												
Подпись руководителя	Сущинина												

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)
Кафедра "Вычислительная техника и системы управления"

УТВЕРЖДАЮ:
Зав. кафедрой _____

ЗАДАНИЕ
на курсовую работу

Студента 4 курса ИИТР Группы _____
тов. Безрядина Алла Владимировна
Тема работы:
Информационная система "Больница"

Срок сдачи законченного проекта " 14 " декабря 2022 г.

Исходные данные

1 Пациент более 100 чел. Функции: просматривать свою историю болезни; просматривать расписание врачей

2 Врач 5-15 чел. Функции: назначать, отменять корректировать лечение; просматривать истории болезней пациентов; оформлять выписки

3 Регистратор 2-5 чел. Функции: оформлять поступление пациентов; составлять график работы врачей; контролировать выход на смену

4

Объем работы

1. Разработать следующие вопросы

1 Видение
2 Словарь предметной области
3 Дополнительные требования
4 Спецификации всех вариантов использования
5 Архитектура программной системы
6 Программная система
7 Тесты

Конструктивно разработать (вычертить)

1	Диаграмма вариантов использования	1 лист
2	Диаграмма деятельности	1 лист
3	Диаграмма коммуникации	1 лист
4	Диаграмма компонентов	1 лист
5	Диаграмма классов	1 лист

Рекомендуемая литература

1 С. Орлов: «Технологии разработки программного обеспечения»
2 Роберт К. Мартин «Быстрая разработка программ»
3 Э. Гамма, Р. Хелм, Р Джонсон, Д. Влиссидес «Приемы объектно-ориентированного проектирования. Паттерны проектирования»

АННОТАЦИЯ

В курсовой работе рассматриваются вопросы создания информационнной системы. Она содержит в себе задачу по разработке информационной системы «Больница», которая в свою очередь взаимодействует с такой системой как «Больница». В данном документе речь пойдет о системе «Больница».

Документация представлена в виде пояснительной записки. Пояснительная записка состоит из 102 страниц, и включает в себя 18 рисунков и 8 таблиц, список приложений.

ANNOTATION

The course work deals with the creation of an information system. It contains the task of developing the "Hospital" information system, which in turn interacts with such a system as the "Hospital". This document will focus on the "Hospital" system.

The documentation is presented in the form of an explanatory note. The explanatory note consists of 102 pages, and includes 18 figures and 8 tables, a list of applications.

Содержание

Введение

Цель данной курсовой работы заключалась в упрощении процесса взаимодействия пациентов и работников больницы, например, у пациентов есть возможность записаться на прием без необходимости звонка или посещения больницы, а врачи могут в любой момент узнать историю посещения пациента без необходимости поиска физического носителя (мед. карточки). Так же разработанная система предоставляет более эффективный доступ к разной информации, например, расписание врачей. Разработанное приложение одинакого полезно для всех пользователей.

В данной ПЗ представлены так же различные необходимые диаграммы.

Язык программирования, который был использован для разработки - C#, ОС – Windows, база данных – MsSQL, сервер – SQLServer, так же был использован подход CodeFirst. Графический интерфейс был разработан с помощью Windows Forms.

1 Видение

1.1 Введение

Целью данного документа является сбор, анализ и определение высокоуровневых потребностей заинтересованных лиц и конечных пользователей системы, а также характеристик системы необходимых для удовлетворения этих потребностей. Детали того, как система отвечает данным потребностям, записываются в модели прецедентов и, возможно, других документах спецификации.

Данный документ распространяется на приложение «Hospital». Цель этого приложения является организация рабочего процесса больницы, запись на прием.ю организация работы и др.

ПЗ содержит в себе описание проблем и их решение с помощью разработанного приложения. Также в настоящем документе описаны типы пользователей программного обеспечения и их потребности.

1.2 Позиционирование

Представленное программное обеспечение разрабатывается для обеспечения организации рабочего процесса в больнице, быстрого доступа и удобной записи информации в базе данных. Далее описаны проблемы, которые решает данный программный продукт:

1. Проблема записи на прием. Для записи на прием нам необходимо либо звонить в больницу, что отнимает время у работников больницы и значительно ограничивает по времени, либо физически придти в больницу, что является еще менее эффективным. Вместо этого приложение предлагает удобную запись в любой момент и просмотр доступного времени для записи. Так же это

производится без какого либо участия сотрудников больницы, что значительно улучшает работу.

2. Доступ к истории посещения. Иногда бывают ситуации, когда нужно получить сведения с какого-либо приема врача, какое было сделано заключение. Обычно для этого нужно как-то связываться с работниками больницы, искать медсицинскую карту, нужный день. Все это отнимает много времени и не всегда бывает вообще успешным. Вместо разработанный программный продукт предоставляет легкий доступ к записи как пациенту, так и врачу.

3. Запись истории посещения. Данная проблема очень схожа с проблемой доступа к истории посещения, такой же поиск физического носителя занимает время и не эффективен, вместо этого сделать запись в приложении занимает намного меньше времени и осуществляется без взаимодействия с другими работниками больницы.

1.3 Описание заинтересованных сторон и пользователей

Данное программное обеспечение значительно упрощает ведение больницы, а также уменьшает затраты на ее содержание и увеличивает эффективность сотрудников.

Исходя из этого следует, что внедрение предложенного программного продукта автоматизирует и ускорит процесс работы, что положительно скажется на организации.

Основными пользователями данного продукта являются: регистраторы, врачи и пациенты. Приложение позволит сократить затрачиваемое количество времени и ресурсов при работе.

1.4 Обзор продукта

Разработанная система связывает пациентов и работников больницы, упрощает их взаимодействие и делает его эффективным.

Время жизни программного обеспечения неограниченно, поскольку он выполняет все поставленные перед ним задачи и в доработке не нуждается.

2. Продумывание вариантов использования

Для определения общего функционала информационной системы используем диаграмму вариантов использования (рис. 1). Каждый вариант использования ниже мы распишем подробно и составим модель «Сущность-Связь» (рис. 2), а из нее уже реляционную модель базы данных (рис. 3).

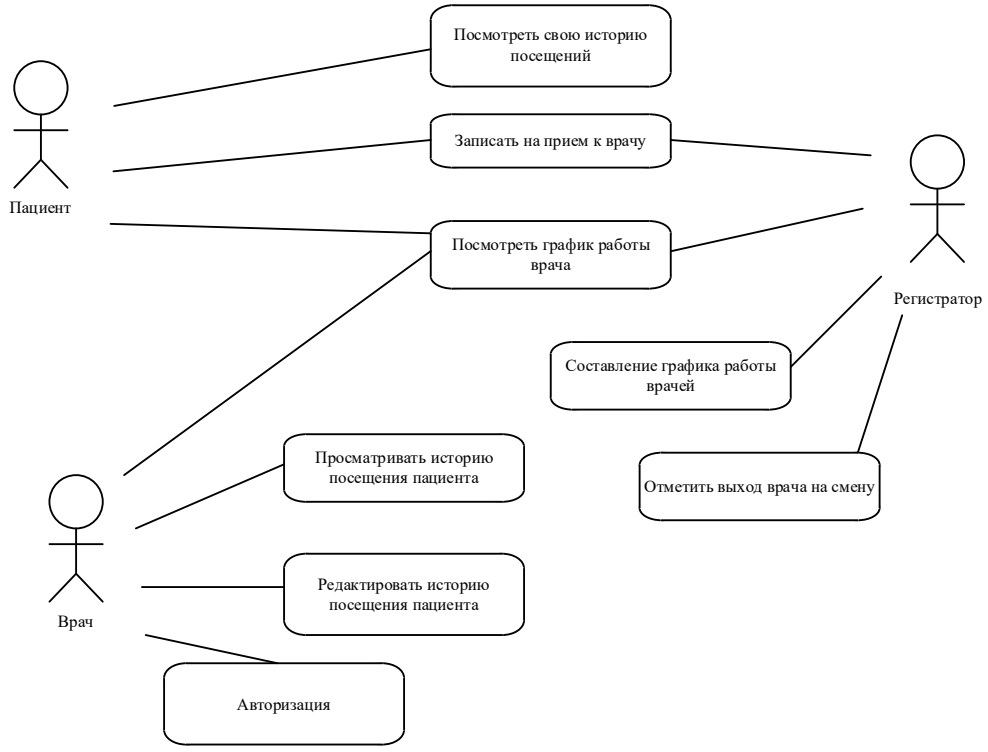
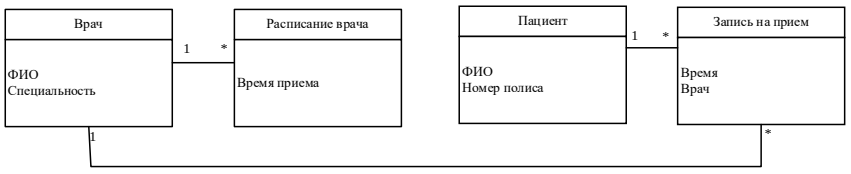
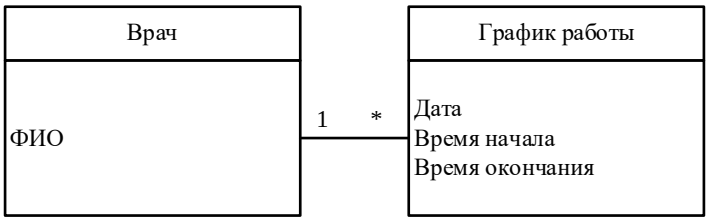


Рисунок 1 – Диаграмма вариантов использования

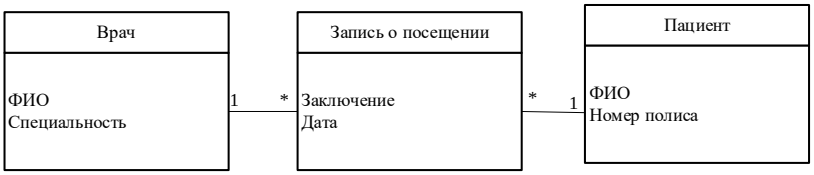
Актант	Регистратор, Пациент
Описание	Актант записывает пациента на прием

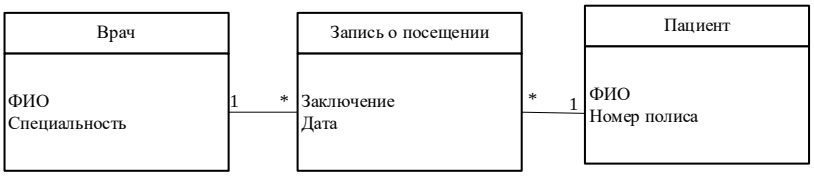
Основной поток событий	1) Актант указывает критерий поиска врача 2) Система предоставляет расписание приема врачей, удовлетворяющих критерию поиска 3) Актант выбирает подходящее время приема 4) Система предоставляет форму для записи на прием к выбранному врачу на выбранное время 5) Актант вносит требуемые данные 6) Система сохраняет данные о записи и сообщает актанту об успешном завершении записи 7) Вариант использования завершается
Альтернативный поток событий	7.1) При необходимости в дополнительной записи перейти к шагу 1. 2.1) Если нет ни одного врача, удовлетворяющего критерию, то перейти к шагу 1. 0) На любом шаге актант может инициировать завершение варианта использования без сохранения данных.
Дополнительные требования	1) Критерий поиска включает в себя фамилию врача, специальность и желаемый интервал времени 2) Требуемые данные: ФИО пациента, номер полиса пациента 3) Выбранное время: конкретная дата и время (напр. 27.11.2022 12:30). 4) Расписание приема врачей: перечень свободных дат и времени для приема
Предусловия	
Постусловия	Выбранная дата бронируется и становится недоступной для записи.
Инфологическая модель	

Актант	Регистратор
Описание	Актант составляет график работы врачей
Основной поток событий	Актант указывает ФИО врача Система предоставляет форму для записи даты и времени работы

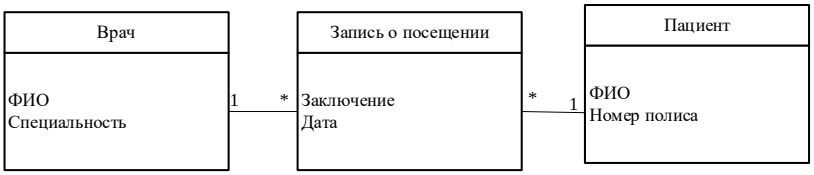
	Актант вносит дату, время начала работы и время окончания Система записывает введенные данные Вариант использование завершается
Альтернативный поток событий	4.1) При необходимости внесения еще одной записи о времени приема – перейти к пункту 1 0) На любом шаге актант может инициировать завершения варианта использования без сохранения данных
Дополнительные требования	
Предусловия	
Постусловия	Введенные данные сохраняются в системе
Инфологическая модель	

Актант	Врач
Описание	Актант редактирует историю посещений пациента
Основной поток событий	1) Актант вводит ФИО пациента для создания записи 2) Система предоставляет список пациентов и их номера полисов 3) Актант выбирает нужного пациента 4) Система предоставляет форму для записи 5) Актант вводит требуемые данные 6) Система сохраняет данные 7) Вариант использования завершается
Альтернативный поток событий	6.1) При необходимости внесения еще одной записи о посещении – перейти к пункту 1 2.1) Если нет ни одного пациента, удовлетворяющего критерию, то перейти к шагу 1. 0) На любом шаге актант может инициировать завершения варианта использования без сохранения данных

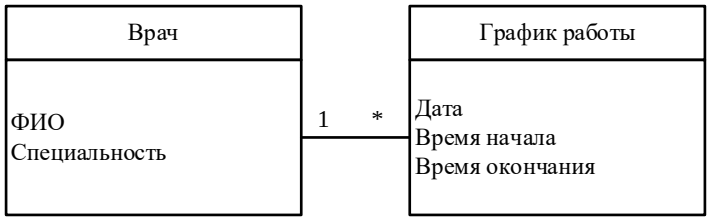
Дополнительные требования	Требуемые данные: Дата, ФИО Врача, Специальность врача, ФИО пациента, Номер полиса пациента, Заключение
Предусловия	
Постусловия	Запись сохраняется в базу
Инфологическая модель	

Актант	Врач
Описание	Актант смотрит запись из истории посещения пациента
Основной поток событий	1) Актант вводит критерий поиска пациента 2) Система выдает список пациентов, подходящих под критерий поиска 3) Актант выбирает нужного пациента и вводит диапазон времени 4) Система выводит данные о посещении 5) Вариант использования завершается
Альтернативный поток событий	2.1) Если нет ни одного пациента, удовлетворяющего критерию, то перейти к шагу 1.
Дополнительные требования	1) Критерий поиска включает в себя ФИО пациента и/или его номер полиса 2) Диапазон времени включает в себя дату начала и дату окончания 3) Данные о посещении представляют из себя дату, ФИО врача, Специальность врача и заключение
Предусловия	
Постусловия	
Инфологическая модель	

Актант	Пациент

Описание	Актант смотрит запись из истории посещения пациента
Основной поток событий	Актант вводит диапазон времени Система выводит данные о посещении Вариант использования завершается
Альтернативный поток событий	2.1) Если нет ни одной записи в введенном диапазоне, то перейти к шагу 1.
Дополнительные требования	Диапазон времени включает в себя дату начала и дату окончания Данные о посещении представляют из себя дату, ФИО врача, Специальность врача и заключение, оказанную услугу
Предусловия	
Постусловия	
Инфологическая модель	

Актант	Врач, Пациент, Регистратор
Описание	Актант смотрит расписание работы врача
Основной поток событий	Актант вводит критерий поиска Система выводит расписание, соответствующее критерию поиска Вариант использования заканчивается
Альтернативный поток событий	1.1) Если нет ни одной записи, удовлетворяющей критерию поиска, то перейти к шагу 1
Дополнительные требования	Критерий поиска: ФИО Врача или специальность врача и диапазон времени Диапазон времени: дата начала и дата окончания
Предусловия	
Постусловия	

Инфологическая модель	

Актант	Регистратор
Описание	Актант отмечает выход врача на смену
Основной поток событий	Актант указывает критерий для поиска Система выдает форму для записи Актант вносит требуемые данные
Альтернативный поток событий	1.1) Если нет ни одной записи, удовлетворяющей критерию поиска, то перейти к шагу 1 0) На любом шаге актант может инициировать завершения варианта использования без сохранения данных
Дополнительные требования	Критерий поиска: ФИО Врача Требуемые данные: ФИО врача, Дата, Отметка
Предусловия	
Постусловия	Запись сохраняется в базе
Инфологическая модель	

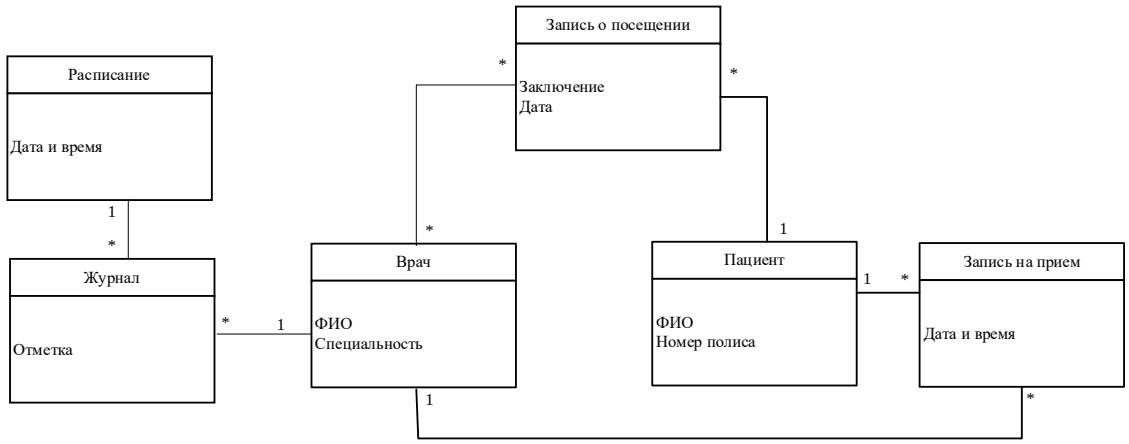


Рисунок 2 – Диаграмма «Сущность-связь»

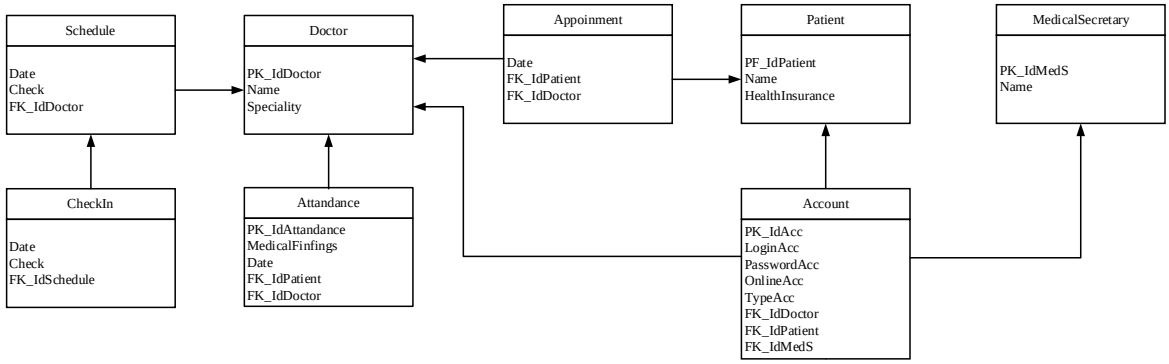


Рисунок 3 – Реляционная модель

3 Разработка диаграммы деятельности

Диаграмма деятельности – диаграмма, на которой показаны действия, состояния которых описаны на диаграмме состояний. Под деятельностью понимается спецификация исполняемого поведения в виде координированного последовательного и параллельного выполнения подчиненных элементов – вложенных видов деятельности и отдельных действий, соединенных между собой потоками, которые идут от выходов одного узла ко входам другого. Диаграммы деятельности используются при моделировании бизнес-процессов, технологических

процессов, последовательных и параллельных вычислений. Диаграммы деятельности представлены ниже, на рисунках 4, 5.

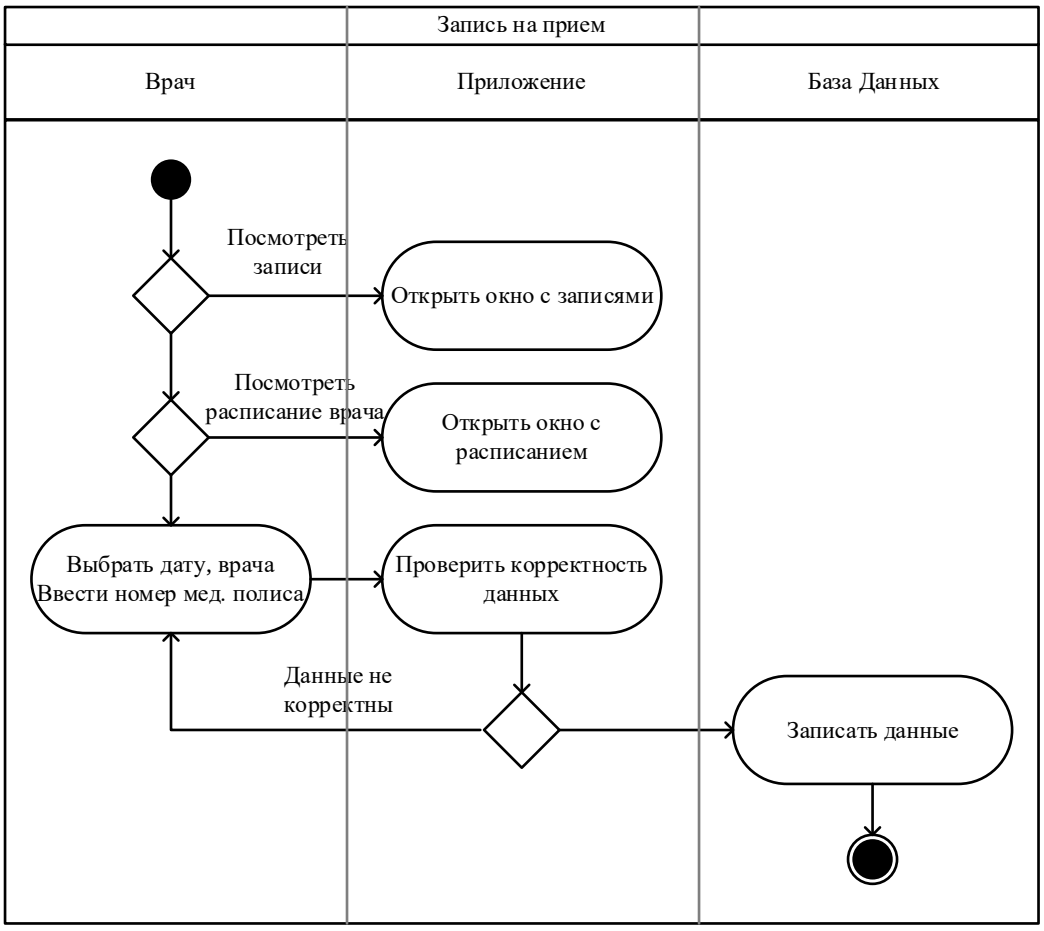


Рисунок 4 – Диаграмма деятельности

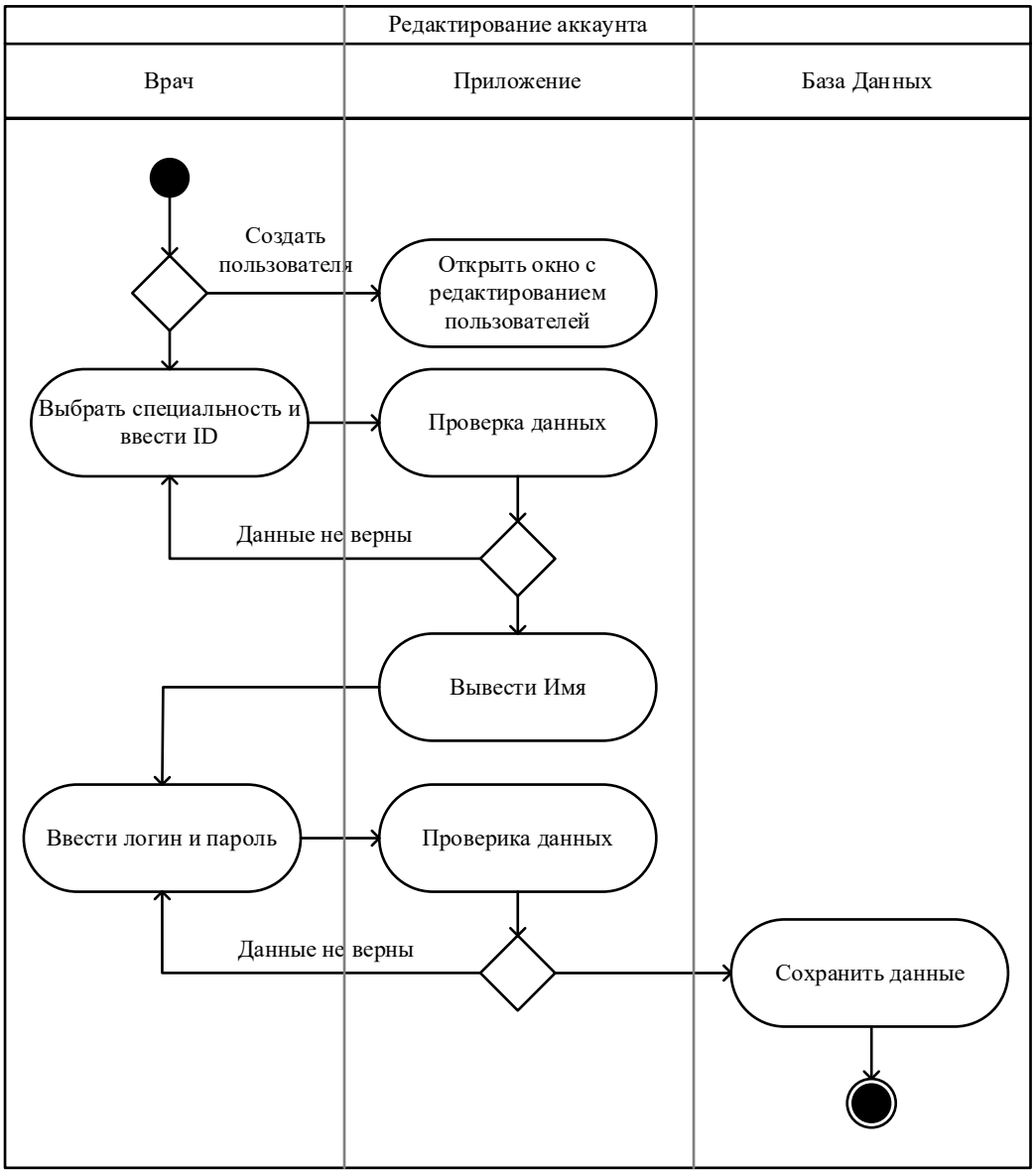


Рисунок 5 – Диаграмма деятельности

4 Диаграмма комуникации

Диаграмма коммуникации – это диаграмма, на которой изображаются взаимодействия между частями композитной структуры или ролями кооперации. В отличие от диаграммы последовательности, на диаграмме коммуникации явно указываются отношения между

объектами, а время как отдельное измерение не используется (применяются порядковые номера вызовов). Ниже, на рисунке 6, представлена диаграмма коммуникации.

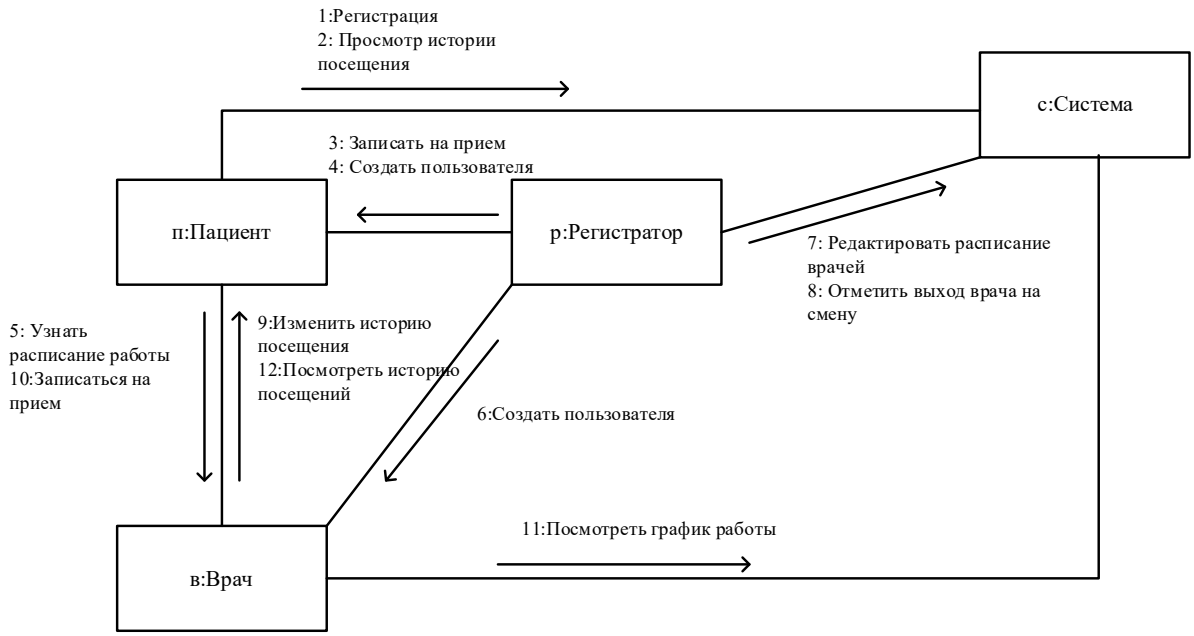


Рисунок 6 – Диаграмма коммуникации

5 Диаграмма компонентов

Диаграмма компонентов – это диаграмма, которая показывает разбиение программной системы на структурные компоненты и связи (зависимости) между компонентами. Ниже, на рисунке 6, изображена диаграмма компонентов.

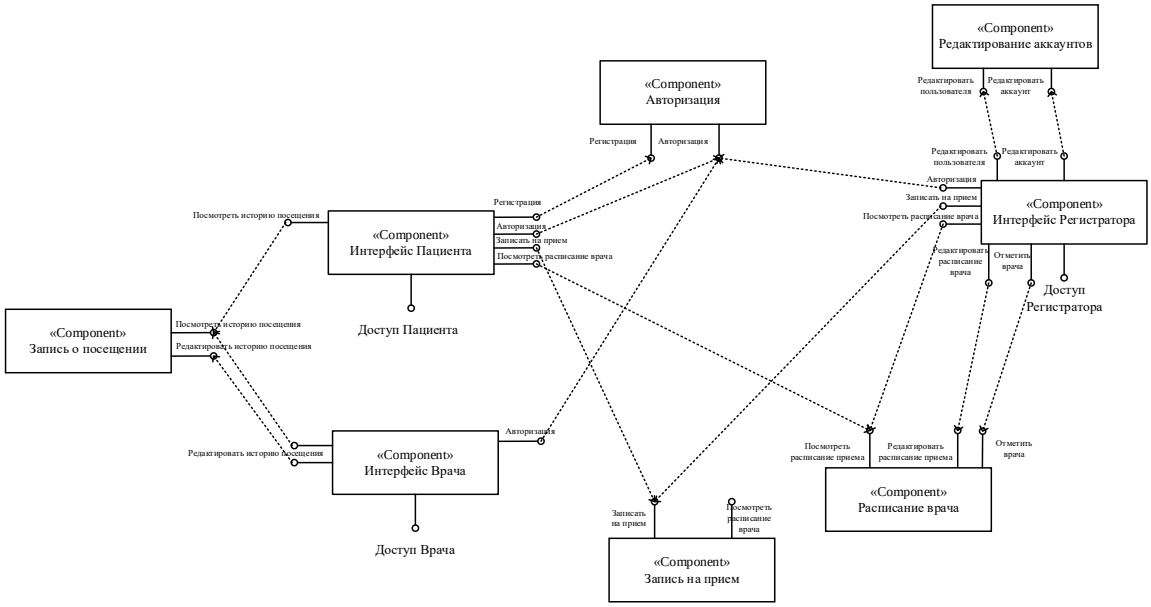


Рисунок 7 – Диаграмма компонентов

6 Диаграмма классов

Диаграмма классов – это диаграмма, демонстрирующая общую структуру иерархии классов системы, их кооперацией, атрибутов (полей), методов, интерфейсов и взаимосвязей (отношений) между ними. Широко применяется не только для документирования и визуализации, но также для конструирования посредством прямого или обратного проектирования. Ниже, на рисунке 8, 9, представлена диаграмма классов.

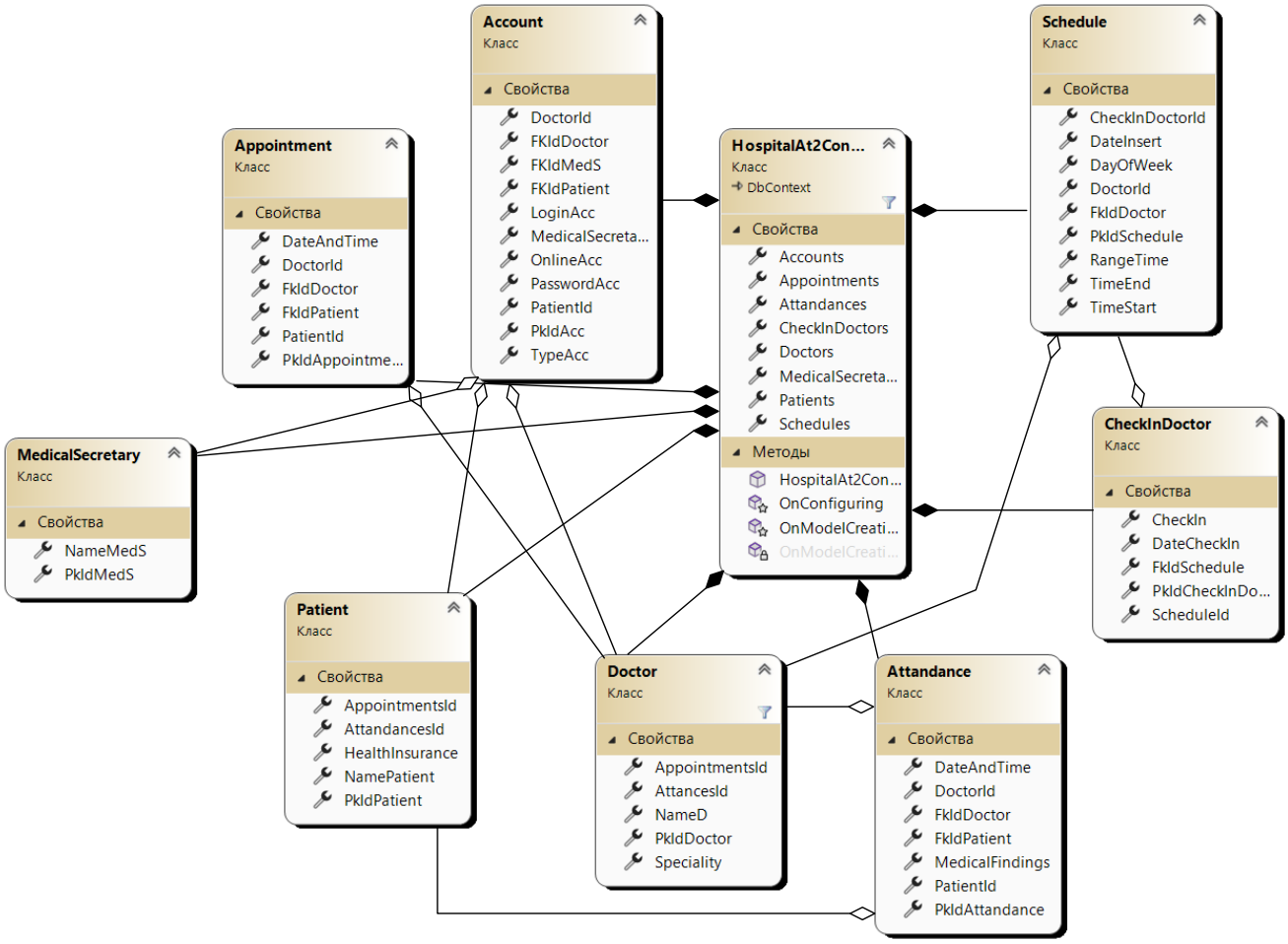


Рисунок 8 – Диаграмма классов

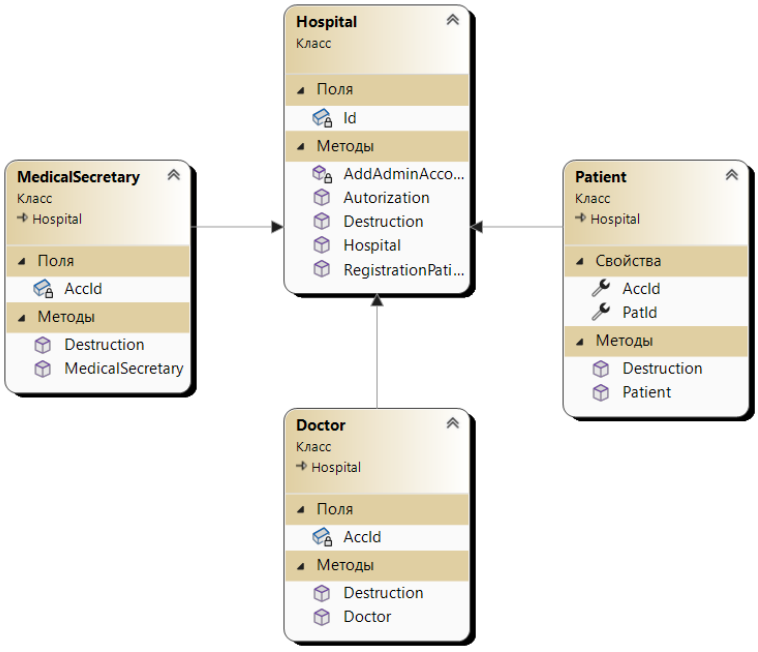


Рисунок 9 – Диаграмма классов

7 Тестирование

Ниже, на рисунках 10-18 представлено тестирование программы, в качестве демонстрации базового функционала программы, а не в качестве выявления различного ошибок.

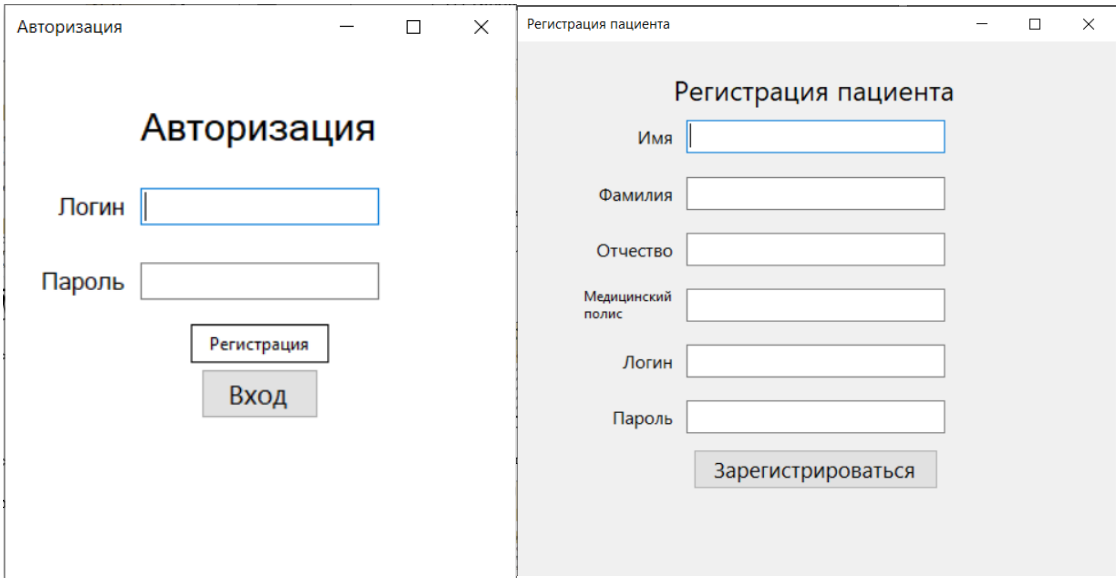
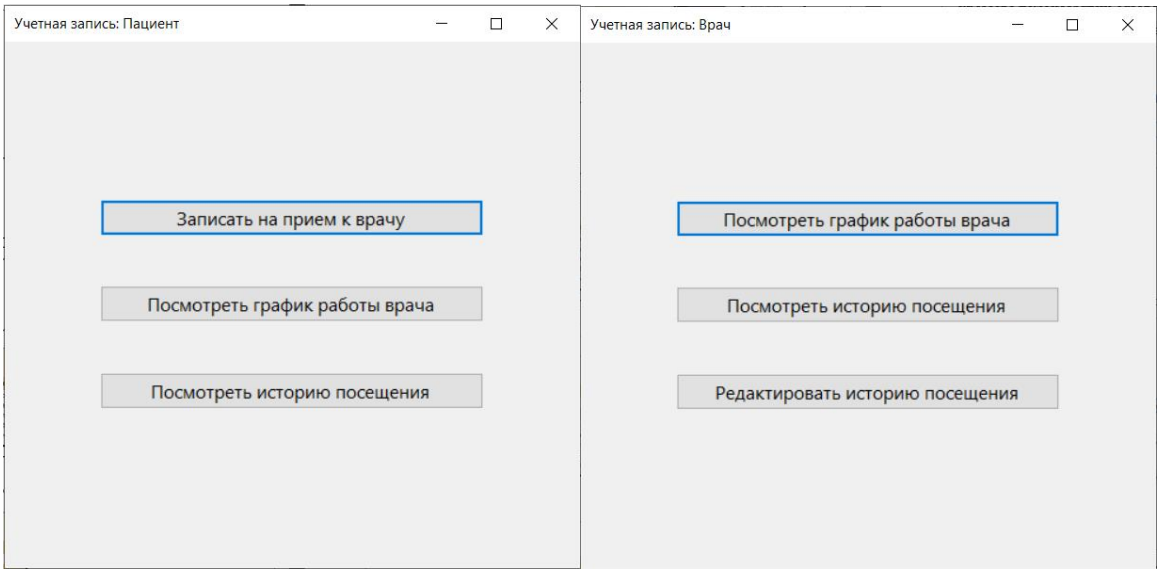


Рисунок 10 – Авторизация и Регистрация



Риснок 11 – Окно пользователя и врача

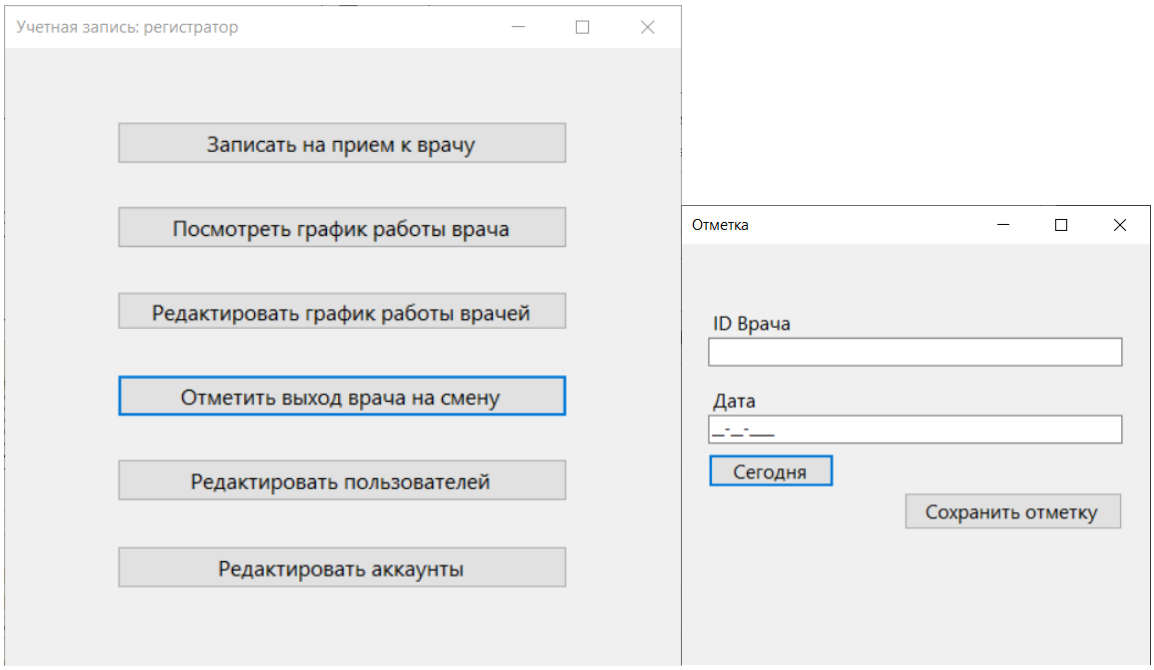


Рисунок 12 – Окно регистратора и окно отметки врача

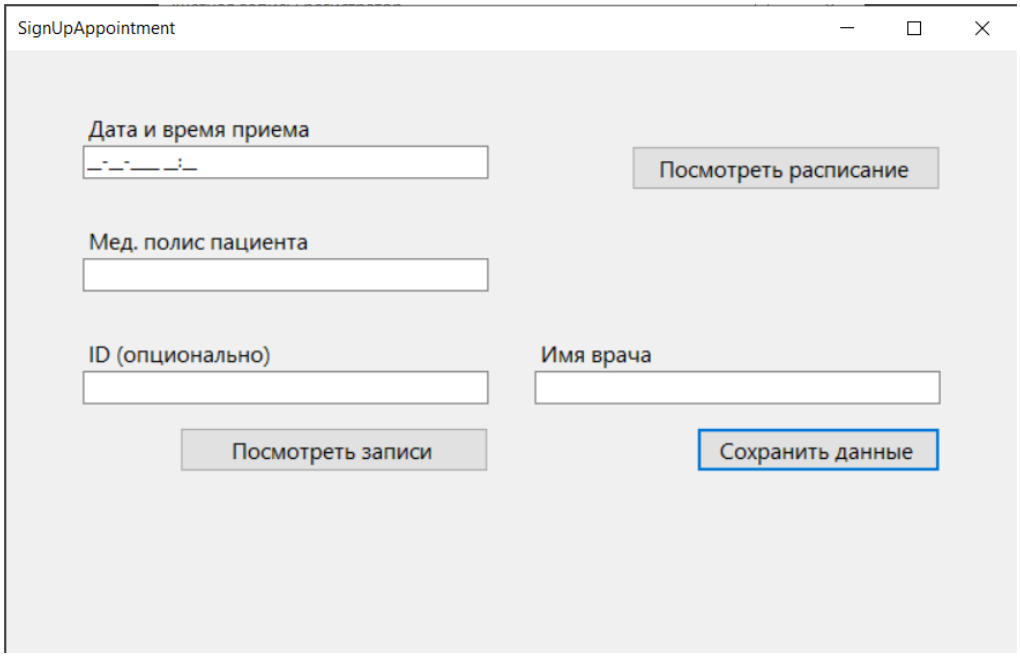


Рисунок 13 – Окно записи на прием

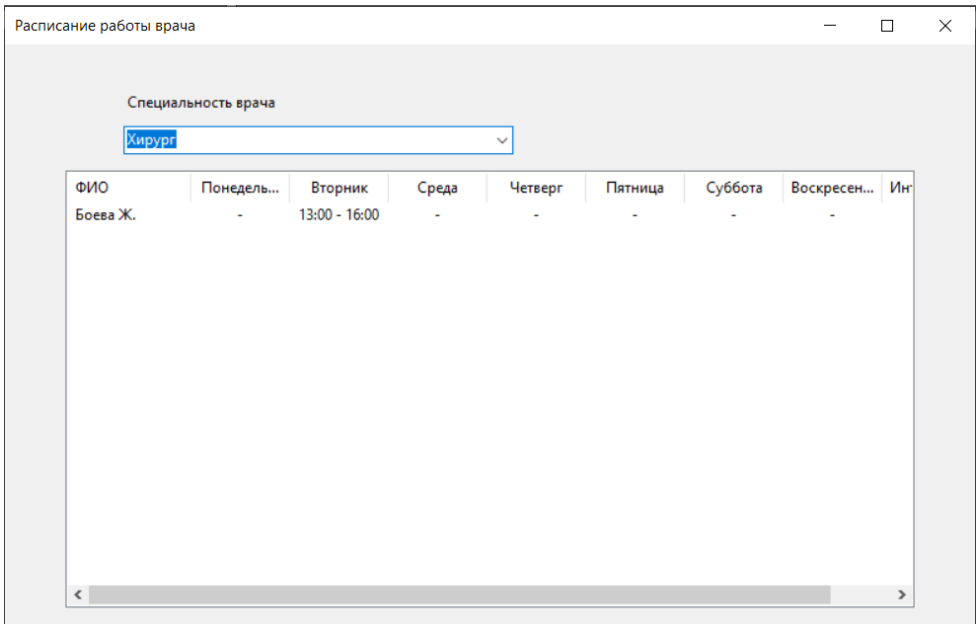


Рисунок 14 – Окно расписания врача

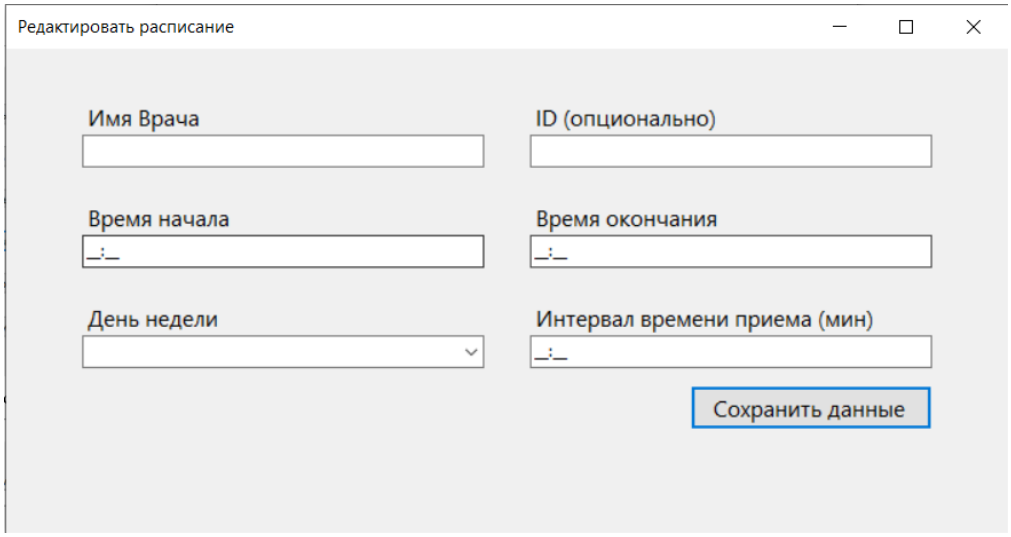


Рисунок 15 – Окно редактирования расписания врача

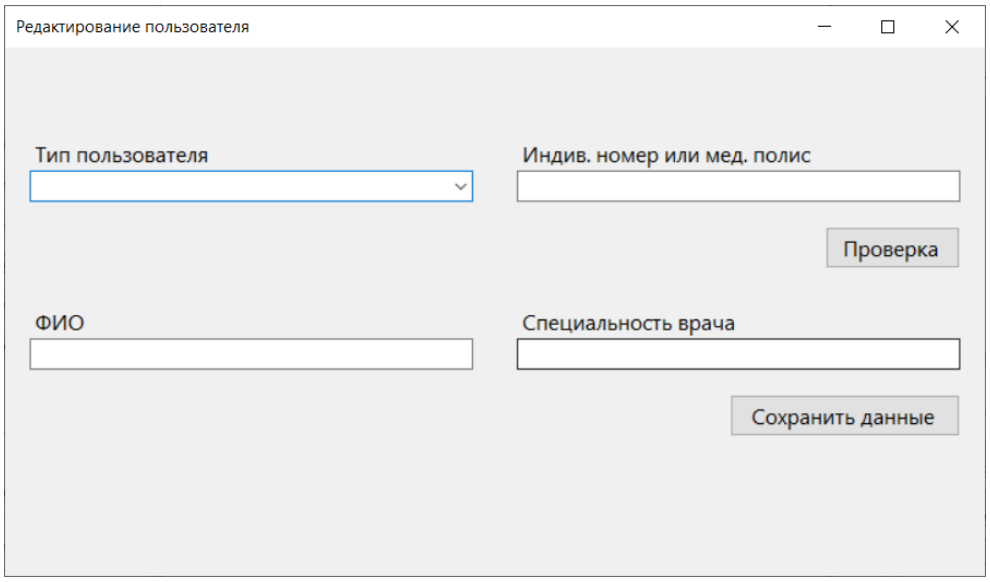


Рисунок 16 – Окно редактирования пользователя

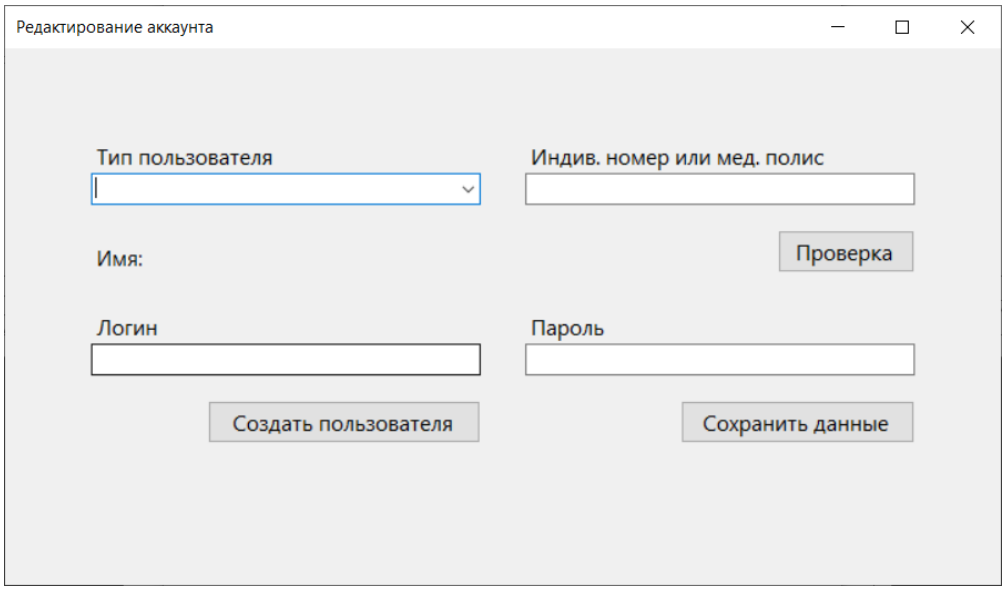


Рисунок 17 – Окно редактирования аккаунта

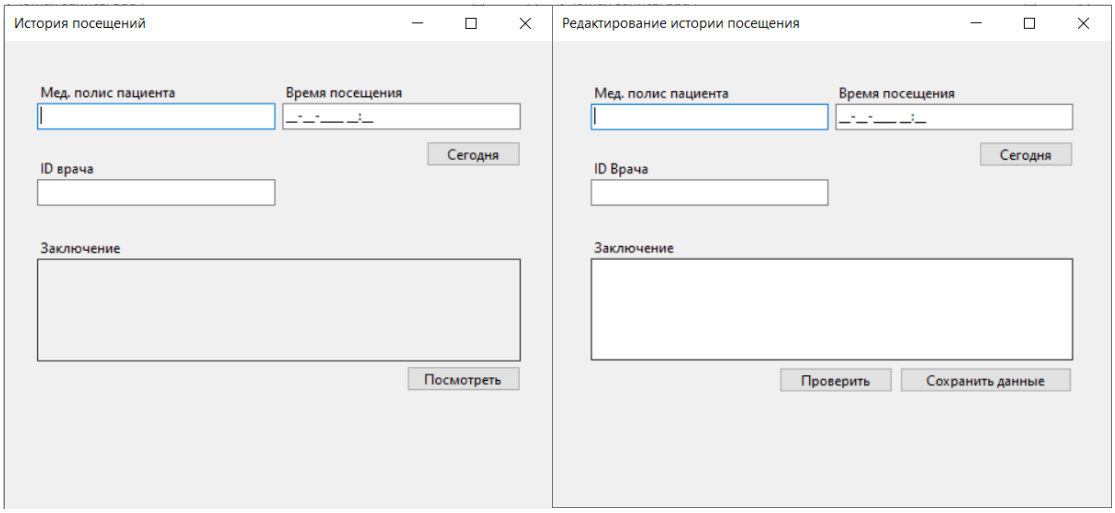


Рисунок 18 – Окно просмотра и редактирования истории посещения

Заключение

В ходе выполнения данной курсовой работы, была разработана система для работы больницы, которая должна обеспечить более высокое быстродействие, а также снизить расходы и увеличить надежность и эффективность хранения информации. Система была разработана на высокоуровневом, кроссплатформенном языке программирования – C#, с использованием кроссплатформенного фреймворка – Entity Framework Core. Интерфейс был реализован с использованием Windows Forms.

Также были разработаны 5 диаграмм: диаграмма вариантов использования, диаграмма деятельности, диаграмма коммуникации, диаграмма компонентов и диаграмма классов. Эти диаграммы описывают функционал программного обеспечения и необходимы для понимания предметной области и требований программного обеспечения.

Приложение

1. Словарь предметной области

Пациент – человек, получающий услуги от больницы.

Врач – работник больницы, оказывающий лечение Пациентам

Регистратор – работник больницы, организующий работу сотрудников.

2. Программный код

В таблице 1 представлен код программы.

Таблица 1 – Код программы

Код

```
//----------------------ClassLogic.Doctor
using CourseWorkAt4.Database;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace CourseWorkAt4.ClassLogic
{
	public class Doctor : Hospital
	{
		private int AccId;
		public Doctor(string login)
		{
			using (HospitalAt2Context db = new HospitalAt2Context())
			{
				// Получение данных из бд
				IEnumerable<Account> dbAcc = db.Accounts.Where(p =>
p.LoginAcc == login);
				foreach (Account acc in dbAcc)
				{
					acc.OnlineAcc = true;
					AccId = acc.PkIdAcc;
				}
				db.SaveChanges();
			}
		}
		public new void Destruction()
		{
```

```
                    using (HospitalAt2Context db = new HospitalAt2Context())
                    {
                        // Получение данных из бд
                        var dbAcc = db.Accounts.Where(p => p.PkIdAcc == AccId);
                        foreach (Account acc in dbAcc)
                        {
                            if (acc.PkIdAcc == AccId)
                            {
                                acc.OnlineAcc = false;
                            }
                        }
                        db.SaveChanges();
                    }
                }
            }
}
//-----------------------ClassLogic.Hospital
using CourseWorkAt4.Database;
using Microsoft.VisualBasic.ApplicationServices;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;

namespace CourseWorkAt4.ClassLogic
{
    public class Hospital
    {
        private int? Id;
        private void AddAdminAccount()
        {
            using (HospitalAt2Context db = new HospitalAt2Context())
            {
                var dbAcc = db.Accounts.Where(p => p.LoginAcc ==
"Admin").SingleOrDefault();
                if (dbAcc == null)
                {
                    //Генерация Primary Key
                    // Получаем самый большой по значению первичный ключ
                    var dbAccId = db.Accounts.OrderBy(p =>
p.PkIdAcc).LastOrDefault();
                    // Если первичного ключа нет, то объявляем его как
наименьший - '0'
                    // Иначе увеличиваем наибольший первичный ключ на 1
                    int idAcc;
                    if (dbAccId != null)
                        idAcc = dbAccId.PkIdAcc + 1;
                    else
                        idAcc = 0;

                    Account admAcc = new Account()
                    {
                        PkIdAcc = idAcc,
                        LoginAcc = "admin",
                        PasswordAcc = "pass",
                        OnlineAcc = false,
                        TypeAcc = "Admin"
                    };
                    db.Accounts.Add(admAcc);
                    db.SaveChanges();
```

```
                        }
                }
        }

        public Hospital()
        {
                AddAdminAccount();
        }

        public string Autorization(string login, string password)
        {
                using (HospitalAt2Context db = new HospitalAt2Context())
                {
                        /* Функция возвращает трехзначное число xyz в виде строки
                         * x = 0/1, 0 - Логин и пароль не правильны, 1 - логин и
пароль верны
                         * y = 0/1, 0 - такой пользователь уже в сети, 1 -
пользователя с такими данными еще нет в сети
                         * z = 0/1/2 - тип пользователя:
                         *                        0 - Medical Secretary
                         *                        1 - Doctor
                         *                        2 - Patient
                         *                        3 - Admin
                         */

                        // Получение данных из бд
                        var dbAcc = db.Accounts.ToList();
                        // Проверка данных
                        foreach (Account acc in dbAcc)
                        {
                                if (acc.LoginAcc == login && acc.PasswordAcc ==
password)
                                {
                                        if (acc.OnlineAcc == false)
                                        // Если введенные данные верны
                                        {
                                                //acc.OnlineAcc = true;
                                                //db.SaveChanges();
                                                this.Id = acc.PkIdAcc;
                                                if (acc.TypeAcc == "MedicalSecretary")
                                                        return "110";
                                                else if (acc.TypeAcc == "Doctor")
                                                        return "111";
                                                else if (acc.TypeAcc == "Patient")
                                                        return "112";
                                                else if (acc.TypeAcc == "Admin")
                                                        return "113";
                                                else
                                                        return "000";
                                        }
                                        else
                                        {
                                                return "100";
                                        }
                                }
                        }
                        return "000";
                }

        }

        // !!!Сделать функцию регистрации универсальной!!!
```

```
            public bool RegistrationPatient(List<string> dataPatient)
            {
                  /*    Функция регистрации Пациентов возвращает значение bool
                   *    true - регистрация прошла успешна
                   *    false - регистрация не удалась
                   */

            //Проверка данных на корректность пациента для добавления в бд
                  string login = dataPatient[0];
                  string password = dataPatient[1];
                  string name = dataPatient[2].ToString() + " " +
dataPatient[3].ToString() + " " + dataPatient[4].ToString();
                  string MedH = dataPatient[5];
            // Проверка на налицие цифр поля name
                  Regex regexName = new Regex(@"(\d)", RegexOptions.IgnoreCase);
                  Match matchesName = regexName.Match(name);
                  if (matchesName.Success)
                        return false;
            // Проверка мед полиса на наличие чего-то кроме цифр
                  Regex regexMedH = new Regex(@"(\D)", RegexOptions.IgnoreCase);
                  Match matchesMedH = regexMedH.Match(MedH);
                  if (matchesMedH.Success)
                        return false;
            // Проверка идентичности Login
                  using (HospitalAt2Context db = new HospitalAt2Context())
                  {
                        var dbAcc = db.Accounts.Where(p => p.LoginAcc ==
login).ToList();
                        if (dbAcc.Count != 0)
                        {
                              return false;
                        }
                  }
                  Database.Patient patient = new Database.Patient();
            // Проверка на наличие пациента в бд и добавление пациента в случае его
отсутствия
                  using (HospitalAt2Context db = new HospitalAt2Context())
                  {
                        var dbPat = db.Patients.Where(p => p.HealthInsurance ==
int.Parse(MedH)).SingleOrDefault();
                        if (dbPat != null)
                        {
                              // Проверка на наличия аккаунта у такого пациента
                              // Если найдется такой пациент, значит у него уже
есть аккаунт
                              var dbAccPat = db.Accounts.Where(p =>
p.PatientId!.PkIdPatient == dbPat.PkIdPatient).SingleOrDefault();
                              if (dbAccPat != null)
                                    return false;
                              // Если пациент уже есть в базе данных, но у него нет
учетной записи,
                              // то его не нужно добавлять в бд повторно
                              else
                              {
                                    patient = dbPat;
                              }
                        }
                        else
                        // Если пациента еще нет в базе данных, то его не нужно
добавить
                        {
                              //Добавление пациента в бд
```

```
                                using (HospitalAt2Context db2 = new
HospitalAt2Context())
                                {
                                        //Генерация Primary Key
                                        // Получаем самый большой по значению первичный
ключ
                                        var dbPatId = db2.Patients.OrderBy(p =>
p.PkIdPatient).LastOrDefault();
                                        // Если первичного ключа нет, то объявляем его
как наименьший - '0'
                                        // Иначе увеличиваем наибольший первичный ключ
на 1
                                        if (dbPatId != null)
                                              patient.PkIdPatient = dbPatId.PkIdPatient
+ 1;
                                        else
                                              patient.PkIdPatient = 0;
                                        patient.NamePatient = name;
                                        patient.HealthInsurance = int.Parse(MedH);
                                        db2.Patients.Add(patient);
                                        db2.SaveChanges();
                                }
                        }
                }
            // Добавление Account
                  using (HospitalAt2Context db = new HospitalAt2Context())
                  {
                        //Генерация Primary Key
                        // Получаем самый большой по значению первичный ключ
                        var dbAccId = db.Accounts.OrderBy(p =>
p.PkIdAcc).LastOrDefault();
                        // Если первичного ключа нет, то объявляем его как
наименьший - '0'
                        // Иначе увеличиваем наибольший первичный ключ на 1
                        int idAcc;
                        if (dbAccId != null)
                              idAcc = dbAccId.PkIdAcc + 1;
                        else
                              idAcc = 0;
                        Account newUser = new Account()
                        {
                              PkIdAcc = idAcc,
                              LoginAcc = login,
                              PasswordAcc = password,
                              OnlineAcc = false,
                              TypeAcc = "Patient",
                              //FKIdPatient = patient.PkIdPatient,
                              /* сделать добавление внешнего ключа на
                                    * на уже существующий элемент
                                    */
                              //PatientId = patient
                        };
                        using (HospitalAt2Context db2 = new HospitalAt2Context())
                        {
                              db2.Accounts.Add(newUser);
                              db2.SaveChanges();
                        }
                        newUser.FKIdPatient = patient.PkIdPatient;
                        newUser.PatientId = patient;
                        db.Accounts.Update(newUser);
                        db.SaveChanges();
                        return true;
```

```
                    }

            }
            public virtual void Destruction() { }
      }
}
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using CourseWorkAt4.Database;
using Microsoft.VisualBasic.Logging;

namespace CourseWorkAt4.ClassLogic
{
      public class MedicalSecretary : Hospital
      {
            private int AccId;
            public MedicalSecretary(string login)
            {
                  using (HospitalAt2Context db = new HospitalAt2Context())
                  {
                        // Получение данных из бд
                        IEnumerable<Account> dbAcc = db.Accounts.Where(p =>
p.LoginAcc == login);
                        foreach (Account acc in dbAcc)
                        {
                              acc.OnlineAcc = true;
                              AccId = acc.PkIdAcc;
                        }
                        db.SaveChanges();
                  }
            }
            public new void Destruction()
            {
                  using (HospitalAt2Context db = new HospitalAt2Context())
                  {
                        // Получение данных из бд
                        var dbAcc = db.Accounts.Where(p => p.PkIdAcc == AccId);
                        foreach (Account acc in dbAcc)
                        {
                              if (acc.PkIdAcc == AccId)
                              {
                                    acc.OnlineAcc = false;
                              }
                        }
                        db.SaveChanges();
                  }
            }

      }
}
using CourseWorkAt4.Database;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace CourseWorkAt4.ClassLogic
{
	public class Patient : Hospital
	{
		public int? PatId { get; }
		public int? AccId { get; }

		public Patient(string login)
		{
			using (HospitalAt2Context db = new HospitalAt2Context())
			{
				// Получение данных из бд
				var dbAcc = db.Accounts
					.Include(p => p.PatientId)
					.Where(p => p.LoginAcc == login)
					.SingleOrDefault();
				if (dbAcc is not null)
				{
					dbAcc.OnlineAcc = true;
					AccId = dbAcc.PkIdAcc;
					PatId = dbAcc.PatientId!.PkIdPatient;
				}
				db.SaveChanges();
			}
		}
		public new void Destruction()
		{
			using (HospitalAt2Context db = new HospitalAt2Context())
			{
				// Получение данных из бд
				var dbAcc = db.Accounts.Where(p => p.PkIdAcc == AccId);
				foreach (Account acc in dbAcc)
				{
					if (acc.PkIdAcc == AccId)
					{
						acc.OnlineAcc = false;
					}
				}
				db.SaveChanges();
			}
		}
	}
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data;
using System.Data.SqlClient;
using CourseWorkAt4;
using CourseWorkAt4.Forms;

//Scaffold-DbContext "Server=DESKTOP-
NL3TB6V;Database=HospitalAt2;Trusted_Connection=True;TrustServerCertificate=True" --
force Microsoft.EntityFrameworkCore.SqlServer

namespace TestForm
{
	internal static class Program
	{
		/// <summary>
```

```
            /// Главная точка входа для приложения.
            /// </summary>
            [STAThread]
            static void Main()
            {
                  Application.EnableVisualStyles();
                  Application.SetCompatibleTextRenderingDefault(false);
                  Application.Run(new Autorization());

            }
      }
}
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace CourseWorkAt4.Database;

public partial class Account
{
      [Key, DatabaseGenerated(DatabaseGeneratedOption.None)]
      public int PkIdAcc { get; set; }

    public string? LoginAcc { get; set; }

    public string? PasswordAcc { get; set; }

    public bool? OnlineAcc { get; set; }

    public string? TypeAcc { get; set; }

      [ForeignKey("FKIdDoctor")]
      public int? FKIdDoctor { get; set; }
      public Doctor? DoctorId { get; set; }

      [ForeignKey("FKIdPatient")]
      public int? FKIdPatient { get; set; }
      public Patient? PatientId { get; set; }

      [ForeignKey("FKIdMedS")]
      public int? FKIdMedS { get; set; }
      public MedicalSecretary? MedicalSecretaryId { get; set; }

}
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace CourseWorkAt4.Database;

public partial class Appointment
{
      [Key, DatabaseGenerated(DatabaseGeneratedOption.None)]
      public int PkIdAppointment { get; set; }
      public DateTime? DateAndTime { get; set; }

      [ForeignKey("FkIdPatient")]
      public int? FkIdPatient { get; set; }
    public Patient? PatientId { get; set; }
```

```
       [ForeignKey("FkIdDoctor")]
       public int? FkIdDoctor { get; set; }
    public Doctor? DoctorId { get; set; }
}
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace CourseWorkAt4.Database;

public partial class Attandance
{
       [Key, DatabaseGenerated(DatabaseGeneratedOption.None)]
       public int PkIdAttandance { get; set; }

    public string? MedicalFindings { get; set; }

    public DateTime? DateAndTime { get; set; }

       [ForeignKey("FkIdDoctor")]
       public int? FkIdDoctor { get; set; }
    public Doctor? DoctorId { get; set; }

       [ForeignKey("FkIdPatient")]
       public int? FkIdPatient { get; set; }
    public Patient? PatientId { get; set; }

}
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations.Schema;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace CourseWorkAt4.Database
{
       public partial class CheckInDoctor
       {
              [Column("PK_IdCheckInDoctor")]
              [Key, DatabaseGenerated(DatabaseGeneratedOption.None)]
              public int PkIdCheckInDoctor { get; set; }
              public DateTime? DateCheckIn { get; set; }
              public bool? CheckIn { get; set; }
              [ForeignKey("FkIdSchedule")]
              public int? FkIdSchedule { get; set; }
              public Schedule? ScheduleId { set; get; }

       }
}
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace CourseWorkAt4.Database;

public partial class Doctor
{
       [Key, DatabaseGenerated(DatabaseGeneratedOption.None)]
```

```
       public int PkIdDoctor { get; set; }

    public string? NameD { get; set; }

    public string? Speciality { get; set; }
       [NotMapped]
       public virtual ICollection<Schedule>? SchedulesId { get; set; }
       [NotMapped]
       public virtual ICollection<Attandance>? AttancesId { get; set; }
       [NotMapped]
       public virtual ICollection<Appointment>? AppointmentsId { get; set; }
}
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations.Schema;
using Microsoft.EntityFrameworkCore;
using Microsoft.VisualBasic.Logging;

namespace CourseWorkAt4.Database;

public partial class HospitalAt2Context : DbContext
{
    public HospitalAt2Context()
    {
            //Database.EnsureDeleted();
            Database.EnsureCreated();
       }

       public HospitalAt2Context(DbContextOptions<HospitalAt2Context> options)
         : base(options)
    {
            //Database.EnsureCreated();
            Database.EnsureDeleted();
       }


       public virtual DbSet<Account> Accounts { get; set; }

    public virtual DbSet<Appointment> Appointments { get; set; }

    public virtual DbSet<Attandance> Attandances { get; set; }

    public virtual DbSet<AttandanceService> AttandanceServices { get; set; }

    public virtual DbSet<CheckInDoctor> CheckInDoctors { get; set; }

    public virtual DbSet<Doctor> Doctors { get; set; }

    public virtual DbSet<DoctorService> DoctorServices { get; set; }

    public virtual DbSet<MedicalSecretary> MedicalSecretaries { get; set; }

    public virtual DbSet<Patient> Patients { get; set; }

    public virtual DbSet<Schedule> Schedules { get; set; }

    public virtual DbSet<ServiceH> ServiceHs { get; set; }
    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
#warning To protect potentially sensitive information in your connection string, you
should move it out of source code. You can avoid scaffolding the connection string by
using the Name= syntax to read it from configuration - see
```

```
https://go.microsoft.com/fwlink/?linkid=2131148. For more guidance on storing
connection strings, see http://go.microsoft.com/fwlink/?LinkId=723263.
            => optionsBuilder.UseSqlServer("Server=DESKTOP-
NL3TB6V;Database=HospitalAt3;Trusted_Connection=True;TrustServerCertificate=True");
    //login=ala;password=IWantNewYear;initial
catalog=HospitalAt2;Trusted_Connection=False
      protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        modelBuilder.Entity<Account>(entity =>
        {
                    entity.HasKey(e =>
e.PkIdAcc).HasName("PK__Account__9DD49AFCFE99F4D6");

            entity.ToTable("Account");

                    entity.Property(e => e.PkIdAcc)
                        .ValueGeneratedNever()
                        .HasColumnName("PK_IdAcc");
            entity.Property(e => e.LoginAcc).HasMaxLength(50);
            entity.Property(e => e.PasswordAcc).HasMaxLength(50);
            entity.Property(e => e.TypeAcc).HasMaxLength(20);
                   entity.Property(e => e.FKIdDoctor).HasColumnName("FK_IdDoctor");
                   entity.Property(e => e.FKIdPatient).HasColumnName("FK_IdPatient");
            entity.Property(e => e.FKIdMedS).HasColumnName("FK_IdMedS");
       });

        modelBuilder.Entity<Appointment>(entity =>
        {
                   entity.HasKey(e => e.PkIdAppointment);

                   entity.ToTable("Appointment");

            entity.Property(e => e.PkIdAppointment)
                         .ValueGeneratedNever()
                         .HasColumnName("PK_IdAppointment");
                   entity.Property(e => e.DateAndTime).HasColumnType("datetime");
            entity.Property(e => e.FkIdDoctor).HasColumnName("FK_IdDoctor");
            entity.Property(e => e.FkIdPatient).HasColumnName("FK_IdPatient");
        });

        modelBuilder.Entity<Attandance>(entity =>
        {
            entity.HasKey(e => e.PkIdAttandance);

            entity.ToTable("Attandance");

            entity.Property(e => e.PkIdAttandance)
                .ValueGeneratedNever()
                .HasColumnName("PK_IdAttandance");
            entity.Property(e => e.DateAndTime).HasColumnType("datetime");
            entity.Property(e => e.FkIdDoctor).HasColumnName("FK_IdDoctor");
            entity.Property(e => e.FkIdPatient).HasColumnName("FK_IdPatient");
            entity.Property(e => e.MedicalFindings).HasMaxLength(100);
        });

        modelBuilder.Entity<AttandanceService>(entity =>
        {
                   entity.HasKey(e => e.PkIdAttandanceService);

                   entity.ToTable("AttandanceService");

                   entity.Property(e => e.PkIdAttandanceService)
```

```
                        .ValueGeneratedNever()
                        .HasColumnName("PK_IdAttandanceService");

                    entity.Property(e =>
e.FkIdAttandance).HasColumnName("FK_IdAttandance");
            entity.Property(e => e.FkIdService).HasColumnName("FK_IdService");
        });

        modelBuilder.Entity<Doctor>(entity =>
        {
            entity.HasKey(e => e.PkIdDoctor).HasName("PK__Doctor__CBDCA36FB9552F3D");

            entity.ToTable("Doctor");

            entity.Property(e => e.PkIdDoctor)
                .ValueGeneratedNever()
                .HasColumnName("PK_IdDoctor");
            entity.Property(e => e.NameD).HasMaxLength(25);
            entity.Property(e => e.Speciality).HasMaxLength(35);
        });

        modelBuilder.Entity<DoctorService>(entity =>
        {

                    entity.HasKey(e => e.PkIdDoctorService);

                    entity.ToTable("DoctorService");

                    entity.Property(e => e.PkIdDoctorService)
                           .ValueGeneratedNever()
                           .HasColumnName("PK_IdDoctorService");
                    entity.Property(e => e.FkIdDoctor).HasColumnName("FK_IdDoctor");
            entity.Property(e => e.FkIdService).HasColumnName("Fk_IdService");
        });

        modelBuilder.Entity<MedicalSecretary>(entity =>
        {
            entity.HasKey(e => e.PkIdMedS).HasName("PK__MedicalS__D31337052BE35F11");

            entity.ToTable("MedicalSecretary");

            entity.Property(e => e.PkIdMedS)
                .ValueGeneratedNever()
                .HasColumnName("PK_IdMedS");
            entity.Property(e => e.NameMedS).HasMaxLength(25);
        });

        modelBuilder.Entity<Patient>(entity =>
        {
            entity.HasKey(e =>
e.PkIdPatient).HasName("PK__Patient__D16397F123E9BC79");

            entity.ToTable("Patient");

            entity.Property(e => e.PkIdPatient)
                .ValueGeneratedNever()
                .HasColumnName("PK_IdPatient");
            entity.Property(e => e.NamePatient).HasMaxLength(25);
        });

        modelBuilder.Entity<Schedule>(entity =>
        {
```

```
            entity
                .HasKey(e => e.PkIdSchedule);
                entity.ToTable("Schedule");

                    entity.Property(e => e.FkIdDoctor).HasColumnName("FK_IdDoctor");
             });

        modelBuilder.Entity<ServiceH>(entity =>
        {
            entity.HasKey(e =>
e.PkIdService).HasName("PK__ServiceH__211C1C609FD7A63C");

            entity.ToTable("ServiceH");

            entity.Property(e => e.PkIdService)
                .ValueGeneratedNever()
                .HasColumnName("PK_IdService");
            entity.Property(e => e.NameService).HasMaxLength(100);
        });

             modelBuilder.Entity<CheckInDoctor>(entity =>
             {
                    entity
                           .HasKey(e => e.PkIdCheckInDoctor);
                    entity.ToTable("CheckInDoctor");

                    entity.Property(e =>
e.FkIdSchedule).HasColumnName("FK_IdSchedule");
                    entity.Property(e => e.CheckIn).HasColumnType("bit");
                    entity.Property(e => e.DateCheckIn).HasColumnType("datetime");
             });

             OnModelCreatingPartial(modelBuilder);

       }

    partial void OnModelCreatingPartial(ModelBuilder modelBuilder);
}
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations.Schema;
using System.ComponentModel.DataAnnotations;

namespace CourseWorkAt4.Database;

public partial class MedicalSecretary
{
       [Key, DatabaseGenerated(DatabaseGeneratedOption.None)]
       public int PkIdMedS { get; set; }

    public string? NameMedS { get; set; }
}
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace CourseWorkAt4.Database;

public partial class Patient
{
       [Key, DatabaseGenerated(DatabaseGeneratedOption.None)]
```

```
      public int PkIdPatient { get; set; }

    public string? NamePatient { get; set; }

    public int? HealthInsurance { get; set; }
      [NotMapped]
      public virtual ICollection<Attandance>? AttandancesId { get; set; }
    [NotMapped]
    public virtual ICollection<Appointment>? AppointmentsId { get; set; }
}
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace CourseWorkAt4.Database;

      //[PrimaryKey(nameof(PkIdSchedule))]
public partial class Schedule
{
      [Column("PK_IdSchedule")]
      [Key, DatabaseGenerated(DatabaseGeneratedOption.None)]
      public int PkIdSchedule { get; set; }
    public TimeSpan? TimeStart { get; set; }
      public TimeSpan? TimeEnd { get; set; }
      [Column(TypeName = "DateTime")]
      public DateTime? DateInsert { get; set; }
      public int? DayOfWeek { get; set; }
      //[Column(TypeName = "time")]
      public TimeSpan? RangeTime { get; set; }

      // Foreign key for Doctor
      [ForeignKey("FkIdDoctor")]
      public int? FkIdDoctor { get; set; }
    public Doctor? DoctorId { set; get; }
      [NotMapped]
      public virtual ICollection<CheckInDoctor>? CheckInDoctorId { get; set; }
}
using CourseWorkAt4.Database;
using Microsoft.EntityFrameworkCore;
using Microsoft.IdentityModel.Tokens;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace CourseWorkAt4.Forms
{
      public partial class AdministrationAcc : Form
      {
            private Form ParentForm;
            private string TypeUser;
            public AdministrationAcc(Form parentForm, string typeUser)
            {
                  InitializeComponent();
```

```
                    ParentForm = parentForm;
                    TypeUser = typeUser;
                    if (typeUser == "Admin")
                          comboBoxUserType.Items.AddRange(new string[] { "Doctor",
"Patient", "Medical Secretary" });
                    else
                          comboBoxUserType.Items.AddRange(new string[] { "Doctor",
"Patient" });

              }
              private List<string> CheckUserTypeAndId(string enteredUserType, string
selectUserType)
              {
                    /* Функция проверяет корректность типа пользователя и его
идентификатора
                     * Возвращает список строк
                     * [0] = 0/1 - верны ли данные (1 - верны, 0 - не верны)
                     * Если [0] = 0, то
                     *           [1] = Сообщение об ошибке
                     *           [2] = флаг, показывающий, что данная ошибка не
является валидной
                     *                           при вызове функции из SaveChanges
                     * Если [0] = 1, то
                     *           [1] = Поле Name
                     *           [[2] = Специальность врача (для пользователей с типом
'Doctor')]
                     */
                    // Проверим введены ли данные в поля
                    if (!(selectUserType == "-1") && !enteredUserType.IsNullOrEmpty())
                    {
                          string enteredId = textBoxId.Text;
                          if (!enteredId.IsNullOrEmpty())
                          {
                                // Проверим, состоит ли Id только из цифр
                                Regex regex = new Regex(@"(\D)",
RegexOptions.IgnoreCase);
                                Match matches = regex.Match(enteredId);
                                if (matches.Success)
                                {   // в поле содержится что-то кроме цифр
                                      return new List<string> { "0", "Поле с
индентификатором должно содержать\nтолько цифры." };
                                }
                                else
                                {
                                      if (selectUserType == "0"
                                            || enteredUserType == "Doctor") // Doctor
                                      {
                                            using (HospitalAt2Context db = new
HospitalAt2Context())
                                            {
                                                  var docUser = db.Doctors.Where(p =>
p.PkIdDoctor == Int32.Parse(enteredId)).SingleOrDefault();
                                                  if (docUser != null)
                                                  {   // Пользователь найден
                                                        return new List<string> {
"1", docUser.NameD, docUser.Speciality };
                                                  }
                                                  else// Пользователя с таким
идентификатором не существует
                                                        return new List<string> {
"0", ("Пользователя с таким идентификатором еще не существует.\n")};
                                            }
```

```
                                    }
                                    else if (selectUserType == "1"
                                          || enteredUserType == "Patient") //
Patient
                                    {
                                          using (HospitalAt2Context db = new
HospitalAt2Context())
                                          {
                                                var patUser = db.Patients.Where(p
=> p.HealthInsurance == Int32.Parse(enteredId)).SingleOrDefault();
                                                if (patUser != null)
                                                {   // Пользователь найден
                                                      return new List<string> {
"1", patUser.NamePatient };
                                                }
                                                else// Пользователя с таким
идентификатором не существует
                                                      return new List<string> {
"0", ("Пользователя с таким идентификатором еще не существует.\n") };
                                          }
                                    }
                                    else if (selectUserType == "2"
                                          || enteredUserType == "Medical
Secretary") // Medical Secretary
                                    {
                                          using (HospitalAt2Context db = new
HospitalAt2Context())
                                          {
                                                var medSUser =
db.MedicalSecretaries.Where(p => p.PkIdMedS ==
Int32.Parse(enteredId)).SingleOrDefault();
                                                if (medSUser != null)
                                                {   // Пользователь найден
                                                      return new List<string> {
"1", medSUser.NameMedS };
                                                }
                                                else// Пользователя с таким
идентификатором не существует
                                                      return new List<string> {
"0", ("Пользователя с таким идентификатором еще не существует.\n") };
                                          }
                                    }
                                    else
                                          return new List<string> { "0",
"Некорректный тип пользователя" };
                              }
                        }
                        else
                              // если идентификатор не введен, то пользователя
пытаются создать
                              return new List<string> { "0", "Введите идентификатор
пользователя"
                                                                        ,
"ThisErrorIsNotForSaveChanges"};
                  }
                  else
                        return new List<string> { "0", "Выберите тип пользователя"
};
            }

            private void buttonCheck_Click(object sender, EventArgs e)
            {
```

```
                    string enteredUserType = comboBoxUserType.Text;
                    string selectUserType = comboBoxUserType.SelectedIndex.ToString();

                    List<string> resultCheck = CheckUserTypeAndId(enteredUserType,
selectUserType);
                    if (resultCheck[0] == "1")
                    {
                          labelName.Text = resultCheck[1];
                          labelMsg.Text = string.Empty;
                    }
                    else
                    {
                          if (resultCheck.Count > 1)
                          {
                                labelMsg.Text = resultCheck[1];
                                labelName.Text = string.Empty;
                          }
                    }

             }

             private void buttonCreateUser_Click(object sender, EventArgs e)
             {
                    AdministrationUser f = new AdministrationUser(this, TypeUser);
                    f.Show();
                    Hide();
             }

             private void AdministrationAcc_FormClosed(object sender,
FormClosedEventArgs e)
             {
                    ParentForm.Show();
             }

             private List<string> CheckUserAccToBe(string typeUser, string idUser)
             {
                    using(HospitalAt2Context db = new HospitalAt2Context())
                    {
                          if (typeUser == "Doctor")
                          {
                                var user = db.Accounts
                                      .Include(p => p.DoctorId)
                                      .Where(p => p.DoctorId!.PkIdDoctor ==
Int32.Parse(idUser))
                                      .SingleOrDefault();
                                if(user is null)
                                      return new List<string> { "0"};
                                else
                                      return new List<string> { "1",
user.PkIdAcc.ToString() };
                          }
                          else if (typeUser == "Patient")
                          {
                                var user = db.Accounts
                                      .Include(p => p.PatientId)
                                      .Where(p => p.PatientId!.HealthInsurance ==
Int32.Parse(idUser))
                                      .SingleOrDefault();
                                if (user is null)
                                      return new List<string> { "0" };
                                else
```

```
                                    return new List<string> { "1",
user.PkIdAcc.ToString() };
                              }
                              else
                              {
                                    var user = db.Accounts
                                          .Include(p => p.MedicalSecretaryId)
                                          .Where(p => p.MedicalSecretaryId!.PkIdMedS ==
Int32.Parse(idUser))
                                          .SingleOrDefault();
                                    if (user is null)
                                          return new List<string> { "0" };
                                    else
                                          return new List<string> { "1",
user.PkIdAcc.ToString() };
                              }
                              return new List<string> { "0" };
                        }
                  }

                  private void buttonSaveChanges_Click(object sender, EventArgs e)
                  {
                        string enteredUserType = comboBoxUserType.Text;
                        string selectUserType = comboBoxUserType.SelectedIndex.ToString();

                        if (!textBoxLogin.Text.IsNullOrEmpty() &&
!textBoxPassword.Text.IsNullOrEmpty())
                        {
                              List<string> resultCheck =
CheckUserTypeAndId(enteredUserType, selectUserType);
                              if (resultCheck[0] == "1")
                              {   // Пользователь с таким Id существует, поэтому мы можем
создать либо изменить аккаунт
                                    List<string> resultCheckAcc =
CheckUserAccToBe(enteredUserType, textBoxId.Text);
                                    if (resultCheckAcc[0] == "1")
                                          // Аккаунт существует и нужно его изменить
                                    {
                                          using (HospitalAt2Context db = new
HospitalAt2Context())
                                          {
                                                // Проверим, есть ли в базе данных
аккаунт с логином
                                                // таким же, как и введенный в форме
                                                var dbAccLog = db.Accounts
                                                      .Where(p => p.LoginAcc ==
textBoxLogin.Text)
                                                      .ToList();
                                                if(dbAccLog.Count != 0)
                                                {
                                                      labelMsg.Text = "Такой логин уже
занят, попробуйте другой";
                                                      return;
                                                }

                                                Database.Account? userAcc = db.Accounts
                                                      .Where(p => p.PkIdAcc ==
Int32.Parse(resultCheckAcc[1]))
                                                      .SingleOrDefault();
                                                if (userAcc != null)
                                                {
```

```
                                                ConfirmActions f = new
ConfirmActions(this
                          , new List<string> // config
                                { "Account"
                                , "Edit"
                                , enteredUserType}
                          , new List<string>  // after
                                { textBoxLogin.Text
                                , textBoxPassword.Text
                                , userAcc.PkIdAcc.ToString()
                                }
                          , new List<string>  // before
                                { userAcc.LoginAcc
                                , userAcc.PasswordAcc
                                , userAcc.PkIdAcc.ToString()
                                }
                          );
                                        f.Show();
                                  }
                                  else
                                        labelMsg.Text = "Что-то не так
пошло где-то...";
                            }

                      }
                      else
                            // Аккаунта нет, его нужно создать
                      {
                            using (HospitalAt2Context db = new
HospitalAt2Context())
                            {
                                  // Проверим, есть ли в базе данных
аккаунт с логином
                                  // таким же, как и введенный в форме
                                  var dbAccLog = db.Accounts
                                        .Where(p => p.LoginAcc ==
textBoxLogin.Text)
                                        .ToList();
                                  if (dbAccLog.Count != 0)
                                  {
                                        labelMsg.Text = "Такой логин уже
занят, попробуйте другой";
                                        return;
                                  }
                            }

                            string id = textBoxId.Text;
```

```
                                        if (enteredUserType == "Patient")
                                        {
                                              using (HospitalAt2Context db = new
HospitalAt2Context())
                                              {
                                                    Patient dbPat = db.Patients.Where(p
=> p.HealthInsurance == Int32.Parse(textBoxId.Text)).SingleOrDefault();
                                                    id = dbPat!.PkIdPatient.ToString();
                                              }
                                        }
                                        ConfirmActions f = new ConfirmActions(this

            , new List<string>  // config

                  { "Account"

                  , "Create"

                  , enteredUserType}

            , new List<string>  // after

                  { textBoxLogin.Text

                  , textBoxPassword.Text

                  , id

                  , labelName.Text}

            );
                                        f.Show();

                                  }


                                  }
                                  else
                                  {   // Возникла какая-то ошибка
                                        labelMsg.Text = resultCheck[1];
                                  }
                  }
                  else
                  {
                        labelMsg.Text = "Введите логин и пароль.";
                  }
            }
      }
}
using CourseWorkAt4.Database;
using Microsoft.IdentityModel.Tokens;
using Microsoft.VisualBasic;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;
using System.Windows.Forms;
```

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

namespace CourseWorkAt4.Forms
{
	public partial class AdministrationUser : Form
	{
		private Form ParentForm;
		private string TypeUser;
		public AdministrationUser(Form parentForm, string typeUser)
		{
			InitializeComponent();
			ParentForm = parentForm;
			TypeUser = typeUser;

			if(typeUser == "Admin")
				comboBoxUserType.Items.AddRange(new string[] { "Doctor",
"Patient", "Medical Secretary" });
			else
				comboBoxUserType.Items.AddRange(new string[] { "Doctor",
"Patient" });

		}

		private List<string> CheckUserTypeAndId(string enteredUserType, string
selectUserType)
		{
			/* Функция проверяет корректность типа пользователя и его
идентификатора
			 * Возвращает список строк
			 * [0] = 0/1 - верны ли данные (1 - верны, 0 - не верны)
			 * Если [0] = 0, то
			 *		[1] = Сообщение об ошибке
			 *		[2] = флаг, показывающий, что данная ошибка не
является валидной
			 *				при вызове функции из SaveChanges
			 * Если [0] = 1, то
			 *		[1] = Поле Name
			 *		[[2] = Специальность врача (для пользователей с типом
'Doctor')]
			 */
			// Проверим введены ли данные в поля
			if (!(selectUserType == "-1") && !enteredUserType.IsNullOrEmpty())
			{
				string enteredId = textBoxId.Text;
				if (!enteredId.IsNullOrEmpty())
				{
					// Проверим, состоит ли Id только из цифр
					Regex regex = new Regex(@"(\D)",
RegexOptions.IgnoreCase);
					Match matches = regex.Match(enteredId);
					if (matches.Success)
					{   // в поле содержится что-то кроме цифр
						return new List<string> { "0", "Поле с
индентификатором должно содержать\nтолько цифры." };
					}
					else
					{
						if (selectUserType == "0"
							|| enteredUserType == "Doctor") // Doctor
						{
							using (HospitalAt2Context db = new
HospitalAt2Context())
```

```
                                        {
                                                var docUser = db.Doctors.Where(p =>
p.PkIdDoctor == Int32.Parse(enteredId)).SingleOrDefault();
                                                if (docUser != null)
                                                {   // Пользователь найден
                                                      return new List<string> {
"1", docUser.NameD, docUser.Speciality };
                                                }
                                                else// Пользователя с таким
идентификатором не существует
                                                      return new List<string> {
"0", ("Пользователя с таким идентификатором еще не существует\n"

            + "Вы можете создать пользователя с таким ID")

            , "ThisErrorIsNotForSaveChanges"};
                                                }
                                    }
                                    else if (selectUserType == "1"
                                          || enteredUserType == "Patient") //
Patient
                                    {
                                          using (HospitalAt2Context db = new
HospitalAt2Context())
                                          {
                                                var patUser = db.Patients.Where(p
=> p.HealthInsurance == Int32.Parse(enteredId)).SingleOrDefault();
                                                if (patUser != null)
                                                {   // Пользователь найден
                                                      return new List<string> {
"1", patUser.NamePatient };
                                                }
                                                else// Пользователя с таким
идентификатором не существует
                                                      return new List<string> {
"0", ("Пользователя с таким идентификатором еще не существует\n"

            + "Вы можете создать пользователя с таким ID")

            , "ThisErrorIsNotForSaveChanges"};
                                                }
                                    }
                                    else if (selectUserType == "2"
                                          || enteredUserType == "Medical
Secretary") // Medical Secretary
                                    {
                                          using (HospitalAt2Context db = new
HospitalAt2Context())
                                          {
                                                var medSUser =
db.MedicalSecretaries.Where(p => p.PkIdMedS ==
Int32.Parse(enteredId)).SingleOrDefault();
                                                if (medSUser != null)
                                                {   // Пользователь найден
                                                      return new List<string> {
"1", medSUser.NameMedS };
                                                }
                                                else// Пользователя с таким
идентификатором не существует
                                                      return new List<string> {
"0", ("Пользователя с таким идентификатором еще не существует\n"
```

```
            + "Вы можете создать пользователя с таким ID")

            , "ThisErrorIsNotForSaveChanges"};
                                        }
                                  }
                                  else
                                        return new List<string> { "0",
"Некорректный тип пользователя" };
                                  }
                            }
                            else
                                  // если идентификатор не введен, то пользователя
пытаются создать
                                  return new List<string> { "0", "Введите идентификатор
пользователя"};
                      }
                      else
                            return new List<string> { "0", "Выберите тип пользователя"
};
            }

            private void buttonCheck_Click(object sender, EventArgs e)
            {
                  string enteredUserType = comboBoxUserType.Text;
                  string selectUserType = comboBoxUserType.SelectedIndex.ToString();

                  List<string> resultCheck = CheckUserTypeAndId(enteredUserType,
selectUserType);
                  if (resultCheck[0] == "1")
                  {
                        textBoxName.Text = resultCheck[1];
                        if (resultCheck.Count > 2)
                        {
                              textBoxSpeciality.Text = resultCheck[2];
                        }
                        labelMsg.Text = "";
                  }
                  else
                  {
                        if(resultCheck.Count > 1)
                        {
                              labelMsg.Text = resultCheck[1];
                        }
                  }

            }

            private List<string> CheckNameAndSpeciality(string userType, string
name, string specialityOrHI)
            {
                  /*    Функция проверяет корректность имени пользователя (и/или
специальности (для врача))
                   *
                        (и/или мед. полис (для пациента))
                   *    Вовращает список типа string
                   *    [0] = 0/1 - не корректные либо корректные данные
                   *    Если [0] = 0, то
                   *          [1] = Сообщение об ошибке
                   */
                  if (name != null)
                  {   // Имя пользователя введено
```

```
                        //      Проверка на корректность поля Name
                        //      Проверка на наличие чего-то кроме букв
                        {
                              Regex regex = new Regex(@"(\d)",
RegexOptions.IgnoreCase);
                              Match matches = regex.Match(name);
                              Regex regex2 = new Regex(@"(\W)",
RegexOptions.IgnoreCase);
                              Match matches2 = regex.Match(name);
                              if (matches.Success || matches2.Success)
                                    return new List<string> { "0", "Имя может
содержать только буквы"};

                        }

                        if (userType == "Doctor")
                        {     // Тип пользователя - Врач
                              // Проверим Speciality на корректность
                              if (specialityOrHI == null)
                                    return new List<string> { "0", "Специальность
врача не введена" };
                              else
                              {   // Проверка на налицие цифр
                                    Regex regexName = new Regex(@"(\d)",
RegexOptions.IgnoreCase);
                                    Match matchesName =
regexName.Match(specialityOrHI);
                                    if (matchesName.Success)
                                          return new List<string> { "0",
"Специальность врача может содержать только буквы" };
                              }
                              return new List<string> { "1"};
                        }
                        else if (userType == "Patient")
                        {     // Тип пользователя - Пациент
                              return new List<string> { "1" };
                        }
                        else
                        {   // Тип пользователя - Регистратор
                              return new List<string> { "1" };
                        }
                  }
                  else
                        return new List<string> { "0", "Имя пользователя не введено"
};
            }
            private void buttonSaveChanges_Click(object sender, EventArgs e)
            {
                  string enteredUserType = comboBoxUserType.Text;
                  string selectUserType = comboBoxUserType.SelectedIndex.ToString();

                  List<string> resultSaveChanges =
CheckNameAndSpeciality(enteredUserType, textBoxName.Text, textBoxSpeciality.Text);
                  if (resultSaveChanges[0] == "1")
                  {
                        List<string> resultCheck =
CheckUserTypeAndId(enteredUserType, selectUserType);
                        if (resultCheck[0] == "1")
                        {   // Пользователь с таким Id существует, поэтому мы
изменяем данные, а не создаем нового пользователя
                              if(enteredUserType == "Doctor")
```

```
                                    {
                                          using (HospitalAt2Context db = new
HospitalAt2Context())
                                          {
                                                Doctor? userDoc = db.Doctors.Where(p =>
p.PkIdDoctor == Int32.Parse(textBoxId.Text)).SingleOrDefault();
                                                if (userDoc != null)
                                                {
                                                      ConfirmActions f = new
ConfirmActions(this
                              , new List<string> // config
                                    { "User"
                                    , "Doctor" }
                              , new List<string>  // after
                                    { textBoxName.Text
                                    , textBoxSpeciality.Text
                                    , userDoc.PkIdDoctor.ToString()}
                              , new List<string>  // before
                                    { userDoc.NameD
                                    , userDoc.Speciality
                                    , userDoc.PkIdDoctor.ToString()}
                              );
                                                      f.Show();
                                                }
                                                else
                                                      labelMsg.Text = "Что-то не так
пошло где-то...";
                                          }
                                    }
                                    else if (enteredUserType == "Patient")
                                    {
                                          using (HospitalAt2Context db = new
HospitalAt2Context())
                                          {
                                                Patient? userPat = db.Patients.Where(p =>
p.HealthInsurance == Int32.Parse(textBoxId.Text)).SingleOrDefault();
                                                if (userPat != null)
                                                {
                                                      ConfirmActions f = new
ConfirmActions(this
                              , new List<string> // config
                                    { "User"
                                    , "Patient" }
                              , new List<string> // after
                                    { textBoxName.Text
```

```
                                , textBoxId.Text

                                , textBoxId.Text}

                          , new List<string> // before

                                { userPat.NamePatient

                                , userPat.HealthInsurance.ToString()

                                , userPat.HealthInsurance.ToString()}

                          );
                                                f.Show();
                                          }
                                          else
                                                labelMsg.Text = "Что-то не так
пошло где-то...";
                                    }
                              }
                              else
                              {
                                    using (HospitalAt2Context db = new
HospitalAt2Context())
                                    {
                                          MedicalSecretary? userMedS =
db.MedicalSecretaries.Where(p => p.PkIdMedS ==
Int32.Parse(textBoxId.Text)).SingleOrDefault();
                                          if (userMedS != null)
                                          {
                                                ConfirmActions f = new
ConfirmActions(this

                          , new List<string> // config

                                { "User"

                                , "Medical Secretary" }

                          , new List<string> // after

                                { textBoxName.Text

                                , ""

                                , userMedS.PkIdMedS.ToString()}

                          , new List<string> // before

                                { userMedS.NameMedS

                                , ""

                                , userMedS.PkIdMedS.ToString()}

                          );
                                                f.Show();
                                          }
                                          else
                                                labelMsg.Text = "Что-то не так
пошло где-то...";
```

```
                                    }
                                }

                            }
                            else if (resultCheck.Count == 2)
                            {   // Возникла какая-то ошибка
                                labelMsg.Text = resultCheck[1];
                            }
                            else
                            {   // Идентификатор пользователя не введен, значит мы
создаем пользователя
                                if (enteredUserType == "Doctor")
                                {
                                    ConfirmActions f = new ConfirmActions(this

            , new List<string> // config

                { "User"

                , "Doctor" }

            , new List<string>  // new

                { textBoxName.Text

                , textBoxSpeciality.Text

                , textBoxId.Text}

            );
                                    f.Show();
                                }
                                else if (enteredUserType == "Patient")
                                {
                                    ConfirmActions f = new ConfirmActions(this

            , new List<string> // config

                { "User"

                , "Patient" }

            , new List<string>  // new

                { textBoxName.Text

                , textBoxId.Text

                , textBoxId.Text}

            );
                                    f.Show();
                                }
                                else
                                {
                                    ConfirmActions f = new ConfirmActions(this

            , new List<string> // config

                { "User"

                , "Medical Secretary" }
```

```
            , new List<string>  // new

                  { textBoxName.Text

                  , ""

                  , textBoxId.Text}

            );
                                    f.Show();
                              }
                        }
                  }
                  else
                  {
                        labelMsg.Text = resultSaveChanges[1];
                  }
            }

            private void AdministrationUser_FormClosed(object sender,
FormClosedEventArgs e)
            {
                  ParentForm.Show();
            }


      }
}
using CourseWorkAt4.ClassLogic;
using Microsoft.IdentityModel.Tokens;
using Microsoft.VisualBasic.Logging;
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 CourseWorkAt4.Forms
{
      public partial class Autorization : Form
      {
            private Hospital User = new Hospital();
            public Autorization()
            {
                  InitializeComponent();
            }

            private void Autorization_FormClosed(object sender, FormClosedEventArgs
e)
            {
                  User.Destruction();

            }

            private void tableLayoutPanelGeneric_Paint(object sender, PaintEventArgs
e)
            {
```

```
            }

            private void tableLayoutPanelGeneric_Paint_1(object sender,
PaintEventArgs e)
            {

            }

            private void buttonLogIn_Click_1(object sender, EventArgs e)
            {
                  string login = textBoxLogin.Text;
                  string password = textBoxPassword.Text;
                  if (!(login.IsNullOrEmpty()) && !(password.IsNullOrEmpty()))
                  {
                        string resultAutorization = User.Autorization(login,
password);
                        if (resultAutorization == "110")            // Пользователь
успешно авторизовался и это Medical Secritary
                        {
                              UserMedS f = new UserMedS(this, login);
                              f.Show();
                              labelMsg.Text = "";
                              Hide();
                        }
                        else if (resultAutorization == "111")   // Пользователь
успешно авторизовался и это Doctor
                        {
                              UserDoctor f = new UserDoctor(this, login);
                              f.Show();
                              labelMsg.Text = "";
                              Hide();
                        }
                        else if (resultAutorization == "112")   // Пользователь
успешно авторизовался и это Patient
                        {
                              UserPatient f = new UserPatient(this, login);
                              f.Show();
                              labelMsg.Text = "";
                              Hide();
                        }
                        else if (resultAutorization == "113")   // Пользователь
успешно авторизовался и это Admin
                        {
                              UserAdmin f = new UserAdmin(this);
                              f.Show();
                              labelMsg.Text = "";
                              Hide();
                        }
                        else if (resultAutorization == "100")   // Login и Password
верны, но пользователь уже в сети
                        {
                              labelMsg.Text = "Этот пользователь уже в сети";
                        }
                        else if (resultAutorization == "000")   // Login или
Password не верны
                        {
                              labelMsg.Text = "Логин или пароль не верны";
                        }
                        else
                        {
                              labelMsg.Text = "Неизвестная ошибка";
```

```
                        }
                    }
                    else labelMsg.Text = "";

                }

                private void buttonSignUp_Click(object sender, EventArgs e)
                {
                    RegistrationPatient f = new RegistrationPatient(this);
                    f.Show();
                    labelMsg.Text = "";
                    Hide();
                }

                private void textBoxLogin_Enter(object sender, EventArgs e)
                {

                }
            }
}
using CourseWorkAt4.Database;
using Microsoft.EntityFrameworkCore;
using Microsoft.IdentityModel.Tokens;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace CourseWorkAt4.Forms
{
    public partial class CheckIn : Form
    {
        Form ParentForm;
        public CheckIn(Form parentForm)
        {
            InitializeComponent();
            ParentForm = parentForm;
        }

        private void buttonDateNow_Click(object sender, EventArgs e)
        {
            DateTime dateTime= DateTime.Now;
            string dateTimeStr = dateTime.ToString(@"dd-MM-yyyy");
            maskedTextBoxTime.Text = dateTimeStr;
        }

        private void CheckIn_FormClosed(object sender, FormClosedEventArgs e)
        {
            ParentForm.Show();
        }

        private void buttonWatchAppoint_Click(object sender, EventArgs e)
        {
            Database.Doctor doctor;
            DateTime dateSch;
            Database.Schedule schedule;
```

```
                    // Проверим корректность введенных данных
                    // Заполнены ли все поля
                    if (!textBoxId.Text.IsNullOrEmpty()
                          && !maskedTextBoxTime.Text.IsNullOrEmpty())
                    {
                          // Проверм корректность введенного ID,
                          // состоит ли Id только из цифр
                          {
                                Regex regex = new Regex(@"(\D)",
RegexOptions.IgnoreCase);
                                Match matches = regex.Match(textBoxId.Text);
                                if (matches.Success)
                                {   // в поле содержится что-то кроме цифр
                                      labelMsg.Text = "Поле с индентификатором должно
содержать\nтолько цифры.";
                                      return;
                                }
                          }
                          // ID состоит только из цифр
                          // Проверим, существует ли врач с таким ID
                          using (HospitalAt2Context db = new HospitalAt2Context())
                          {
                                var dbDoc = db.Doctors.Where(p => p.PkIdDoctor ==
Int32.Parse(textBoxId.Text)).SingleOrDefault();
                                if (dbDoc is not null)
                                {
                                      doctor = dbDoc;
                                }
                                else
                                {
                                      labelMsg.Text = "Врача с таким ID нет";
                                      return;
                                }
                          }
                          // Врач существует

                          dateSch = DateTime.ParseExact(maskedTextBoxTime.Text, "dd-
MM-yyyy",

System.Globalization.CultureInfo.InvariantCulture);
                          // Проверим, есть ли у него часы приема на указанную дату
                          try
                          {
                                DayOfWeek f = dateSch.DayOfWeek;
                                int? myDayOfWeek;
                                switch (f)
                                {
                                      case DayOfWeek.Monday:
                                            myDayOfWeek = 0;
                                            break;
                                      case DayOfWeek.Tuesday:
                                            myDayOfWeek = 1;
                                            break;
                                      case DayOfWeek.Wednesday:
                                            myDayOfWeek = 2;
                                            break;
                                      case DayOfWeek.Thursday:
                                            myDayOfWeek = 3;
                                            break;
                                      case DayOfWeek.Friday:
                                            myDayOfWeek = 4;
                                            break;
```

```
                                        case DayOfWeek.Saturday:
                                              myDayOfWeek = 5;
                                              break;
                                        default:
                                              myDayOfWeek = 6;
                                              break;
                                  }
                                  using (HospitalAt2Context db = new
HospitalAt2Context())
                                  {
                                        // Находим последнюю дату добавления
                                        var schDate = db.Schedules
                                              .Include(p => p.DoctorId)
                                              .Where(p => p.DoctorId!.PkIdDoctor ==
doctor!.PkIdDoctor
                                                    && p.DayOfWeek == myDayOfWeek)
                                              .OrderBy(p => p.DateInsert)
                                              .LastOrDefault();
                                        // Если не нашли, значит врач в этот день не
принимает
                                        if (schDate is null)
                                        {
                                              labelMsg.Text = "Неправильное время
приема";
                                              return;
                                        }
                                        else
                                        {
                                              schedule = schDate;
                                        }

                                  }
                                  // Часы приема на указанную дату есть

                                  // Добавим данных для вставки в бд
                                  bool check = true;
                                  DateTime dateNow = DateTime.Now;

                                  // Все данные есть и верны:
                                  //          Ссылка на расписание - schedule
                                  //          отметка - check
                                  //          Сегодняшняя дата - dateNow

                                  // Вставим данные в бд
                                  using(HospitalAt2Context db = new
HospitalAt2Context())
                                  {

                                        //Генерация Primary Key
                                        // Получаем самый большой по значению первичный
ключ
                                        var dbAccId = db.CheckInDoctors.OrderBy(p =>
p.PkIdCheckInDoctor).LastOrDefault();
                                        // Если первичного ключа нет, то объявляем его
как наименьший - '0'
                                        // Иначе увеличиваем наибольший первичный ключ
на 1
                                        int idAcc;
                                        if (dbAccId != null)
                                              idAcc = dbAccId.PkIdCheckInDoctor + 1;
                                        else
                                              idAcc = 0;
```

```
                                        Database.Schedule? locSch = db.Schedules
                                              .Where(p => p.PkIdSchedule ==
schedule.PkIdSchedule)
                                              .SingleOrDefault();

                                        Database.CheckInDoctor newCheck = new
Database.CheckInDoctor()
                                        {
                                              PkIdCheckInDoctor= idAcc,
                                              DateCheckIn = dateNow,
                                              CheckIn= check,
                                              FkIdSchedule= locSch.PkIdSchedule,
                                              ScheduleId = locSch
                                        };
                                        db.CheckInDoctors.Add(newCheck);
                                        db.SaveChanges();
                                  }
                                  labelMsg.Text = "Отметка успешно добавлена";

                            }
                            catch
                            {
                                  labelMsg.Text = "Время введено неверно";
                            }
                      }
                      else
                      {
                            labelMsg.Text = "Введите все данные";
                      }
                }
          }
}
using CourseWorkAt4.ClassLogic;
using CourseWorkAt4.Database;
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 CourseWorkAt4.Forms
{
      public partial class ConfirmActions : Form
      {
            private Form ParentForm;
            List<string> Config;
            List<string> After;
            public ConfirmActions(Form parentForm,  List<string> config,List<string>
after, List<string>? before = null)
            {
                  InitializeComponent();
                  ParentForm = parentForm;
                  Config = config;
                  After = after;

                  labelChangeAsk.Text = string.Empty;
                  labelBefore.Text = string.Empty;
```

```
                    labelAfter.Text = string.Empty;
                    if (config[0] == "User")
                    {
                          string[] tempUser = new string[] { "Имя: ", "\n", "Спец. или
мед.п.: ", "\n", "ID: ", "\n" };
                          // Заполнение поля After
                          for(int i = 0; i <after.Count; i++)
                          {
                                tempUser[i*2+1] = "\t" + after[i] + "\n";
                          }
                          labelAfter.Text = string.Join("", tempUser);
                    }
                    else
                    {
                          string[] tempUser = new string[] { "Логин: ", "\n", "Пароль:
", "\n", "ID: ", "\n", "Имя: ", "\n" };
                          // Заполнение поля After
                          for (int i = 0; i < after.Count; i++)
                          {
                                tempUser[i * 2 + 1] = "\t" + after[i] + "\n";
                          }
                          labelAfter.Text = string.Join("", tempUser);
                    }

                    if (before is null)
                    {
                          if (config[0] == "User")
                          {
                                labelChangeAsk.Text += "создать пользователя с
следующими данными:\n";
                          }
                          else
                          {
                                labelChangeAsk.Text += "создать аккаунт с следующими
данными:\n";
                          }
                    }
                    else
                    {
                          if (config[0] == "User")
                          {
                                labelChangeAsk.Text += "редактировать пользователя с
следующими данными:\n";

                          }
                          else
                          {
                                labelChangeAsk.Text += "редактировать аккаунт с
следующими данными:\n";
                          }
                          string[] tempUser = new string[] { "Имя: ", "\n", "Спец. или
мед.п.: ", "\n", "ID: ", "\n" };
                          // Заполнение поля After
                          for (int i = 0; i < before.Count; i++)
                          {
                                tempUser[i*2 + 1] = before[i] + "\n";
                          }
                          labelBefore.Text = string.Join("", tempUser);
                    }
              }

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

```
            {
                  // Записать данные в базу данных и выйти
                  if (Config[0] == "User")
                        // Работа с пользователями
                  {
                        if (Config[1] == "Doctor")
                        {
                              // Проверяем, есть ли пользователь с таким id
                              using(HospitalAt2Context db = new
HospitalAt2Context())
                              {
                                    var user = db.Doctors.Where(p => p.PkIdDoctor
== Int32.Parse(After[2])).SingleOrDefault();
                                    if(user is null)
                                          // Пользователя не существует, поэтому
создаем его
                                    {
                                          using (HospitalAt2Context db2 = new
HospitalAt2Context())
                                          {
                                                Database.Doctor userDoc = new
Database.Doctor()
                                                {
                                                      NameD = After[0],
                                                      Speciality = After[1],
                                                      PkIdDoctor =
Int32.Parse(After[2])
                                                };
                                                db2.Doctors.Add(userDoc);
                                                db2.SaveChanges();
                                          }

                                    }
                                    else
                                          // Пользователь существует, поэтому
просто меняем данные
                                    {
                                          using (HospitalAt2Context db2 = new
HospitalAt2Context())
                                          {
                                                Database.Doctor userDoc = new
Database.Doctor()
                                                {
                                                      NameD = After[0],
                                                      Speciality = After[1],
                                                      PkIdDoctor =
Int32.Parse(After[2])
                                                };
                                                db2.Doctors.Update(userDoc);
                                                db2.SaveChanges();
                                          }

                                    }
                              }

                        }
                        else if (Config[1] == "Patient")
                        {
                              // Проверяем, есть ли пользователь с таким id
                              using (HospitalAt2Context db = new
HospitalAt2Context())
                              {
```

```
                                        var user = db.Patients.Where(p =>
p.HealthInsurance == Int32.Parse(After[2])).SingleOrDefault();
                                        if (user is null)
                                        // Пользователя не существует, поэтому создаем
его
                                        {
                                              using (HospitalAt2Context db2 = new
HospitalAt2Context())
                                              {
                                                    //Генерация Primary Key
                                                    // Получаем самый большой по
значению первичный ключ
                                                    var dbAccId = db.Patients.OrderBy(p
=> p.PkIdPatient).LastOrDefault();
                                                    // Если первичного ключа нет, то
объявляем его как наименьший - '0'
                                                    // Иначе увеличиваем наибольший
первичный ключ на 1
                                                    int idAcc;
                                                    if (dbAccId != null)
                                                          idAcc = dbAccId.PkIdPatient +
1;
                                                    else
                                                          idAcc = 0;
                                                    Database.Patient userPat = new
Database.Patient()
                                                    {
                                                          NamePatient = After[0],
                                                          HealthInsurance =
Int32.Parse(After[1]),
                                                          PkIdPatient = idAcc
                                                    };
                                                    db2.Patients.Add(userPat);
                                                    db2.SaveChanges();
                                              }
                                        }
                                        else
                                        // Пользователь существует, поэтому меняем
данные
                                        {
                                              using (HospitalAt2Context db2 = new
HospitalAt2Context())
                                              {
                                                    Database.Patient userPat = new
Database.Patient()
                                                    {
                                                          NamePatient = After[0],
                                                          HealthInsurance =
Int32.Parse(After[1]),
                                                          PkIdPatient =
user.PkIdPatient

                                                    };
                                                    db2.Patients.Update(userPat);
                                                    db2.SaveChanges();
                                              }
                                        }
                                  }
                            }
                            else
                            {
                                  // Проверяем, есть ли пользователь с таким id
```

```
                                using (HospitalAt2Context db = new
HospitalAt2Context())
                                {
                                        var user = db.MedicalSecretaries.Where(p =>
p.PkIdMedS == Int32.Parse(After[2])).SingleOrDefault();
                                        if (user is null)
                                        // Пользователя не существует, поэтому создаем
его
                                        {
                                                using (HospitalAt2Context db2 = new
HospitalAt2Context())
                                                {
                                                        Database.MedicalSecretary userMedS
= new Database.MedicalSecretary()
                                                        {
                                                                NameMedS = After[0],
                                                                PkIdMedS =
Int32.Parse(After[2])
                                                        };

      db2.MedicalSecretaries.Add(userMedS);
                                                        db2.SaveChanges();
                                                }
                                        }
                                        else
                                        // Пользователь существует, поэтому обновляем
данные
                                        {
                                                using (HospitalAt2Context db2 = new
HospitalAt2Context())
                                                {
                                                        Database.MedicalSecretary userMedS
= new Database.MedicalSecretary()
                                                        {
                                                                NameMedS = After[0],
                                                                PkIdMedS =
Int32.Parse(After[2])
                                                        };

      db2.MedicalSecretaries.Update(userMedS);
                                                        db2.SaveChanges();
                                                }
                                        }
                                }
                        }
                }
                else
                        // Работа с Аккаунтами
                {
                        if (Config[1] == "Edit")
                                // Редактировать аккаунт
                        {
                                using(HospitalAt2Context db = new
HospitalAt2Context())
                                {
                                        var user = db.Accounts.Where(p => p.PkIdAcc ==
Int32.Parse(After[2])).SingleOrDefault();
                                        if(user is not null)
                                        {
                                                user.LoginAcc = After[0];
                                                user.PasswordAcc= After[1];
                                                db.Accounts.Update(user);
```

```
                                        db.SaveChanges();
                                  }
                            }
                      }
                      else
                            // Создать аккаунт
                      {
                            if (Config[2] == "Patient")
                            {
                                  using(HospitalAt2Context db = new
HospitalAt2Context())
                                  {
                                        Database.Patient dbPat = db.Patients
                                              .Where(p => p.PkIdPatient ==
Int32.Parse(After[2]))
                                              .SingleOrDefault();

                                        //Генерация Primary Key
                                        // Получаем самый большой по значению
первичный ключ
                                        var dbAccId = db.Accounts.OrderBy(p =>
p.PkIdAcc).LastOrDefault();
                                        // Если первичного ключа нет, то
объявляем его как наименьший - '0'
                                        // Иначе увеличиваем наибольший первичный
ключ на 1
                                        int idAcc;
                                        if (dbAccId != null)
                                              idAcc = dbAccId.PkIdAcc + 1;
                                        else
                                              idAcc = 0;

                                        Database.Account newAcc = new
Database.Account()
                                        {
                                              PkIdAcc = idAcc,
                                              LoginAcc = After[0],
                                              PasswordAcc = After[1],
                                              PatientId = dbPat,
                                              OnlineAcc = false,
                                              TypeAcc = "Patient"
                                        };
                                        db.Accounts.Add(newAcc);
                                        db.SaveChanges();
                                  }
                            }
                            else if (Config[2] == "Doctor")
                            {
                                  using (HospitalAt2Context db = new
HospitalAt2Context())
                                  {
                                        Database.Doctor dbDoc = db.Doctors
                                              .Where(p => p.PkIdDoctor ==
Int32.Parse(After[2]))
                                              .SingleOrDefault();

                                        //Генерация Primary Key
                                        // Получаем самый большой по значению
первичный ключ
                                        var dbAccId = db.Accounts.OrderBy(p =>
p.PkIdAcc).LastOrDefault();
```

```
                                        // Если первичного ключа нет, то
объявляем его как наименьший - '0'
                                        // Иначе увеличиваем наибольший первичный
ключ на 1
                                        int idAcc;
                                        if (dbAccId != null)
                                              idAcc = dbAccId.PkIdAcc + 1;
                                        else
                                              idAcc = 0;

                                        Database.Account newAcc = new
Database.Account()
                                        {
                                              PkIdAcc = idAcc,
                                              LoginAcc = After[0],
                                              PasswordAcc = After[1],
                                              DoctorId = dbDoc,
                                              OnlineAcc = false,
                                              TypeAcc = "Doctor"
                                        };
                                        db.Accounts.Add(newAcc);
                                        db.SaveChanges();
                                  }
                            }
                            else
                            {
                                  using (HospitalAt2Context db = new
HospitalAt2Context())
                                  {
                                        Database.MedicalSecretary dbMedS =
db.MedicalSecretaries
                                              .Where(p => p.PkIdMedS ==
Int32.Parse(After[2]))
                                              .SingleOrDefault();

                                        //Генерация Primary Key
                                        // Получаем самый большой по значению
первичный ключ
                                        var dbAccId = db.Accounts.OrderBy(p =>
p.PkIdAcc).LastOrDefault();
                                        // Если первичного ключа нет, то
объявляем его как наименьший - '0'
                                        // Иначе увеличиваем наибольший первичный
ключ на 1
                                        int idAcc;
                                        if (dbAccId != null)
                                              idAcc = dbAccId.PkIdAcc + 1;
                                        else
                                              idAcc = 0;

                                        Database.Account newAcc = new
Database.Account()
                                        {
                                              PkIdAcc = idAcc,
                                              LoginAcc = After[0],
                                              PasswordAcc = After[1],
                                              MedicalSecretaryId = dbMedS,
                                              OnlineAcc = false,
                                              TypeAcc = "MedicalSecretary"
                                        };
                                        db.Accounts.Add(newAcc);
                                        db.SaveChanges();
```

```
                                        }
                                    }
                                }
                        }

                        this.Close();
                }

                private void buttonNo_Click(object sender, EventArgs e)
                {
                        // Выйти из формы
                        this.Close();
                }
        }
}
using CourseWorkAt4.Database;
using Microsoft.EntityFrameworkCore;
using Microsoft.IdentityModel.Tokens;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace CourseWorkAt4.Forms
{
        public partial class EditAttandance : Form
        {
                Form ParentForm;
                public EditAttandance(Form parentForm)
                {
                        InitializeComponent();
                        ParentForm = parentForm;
                }

                private void EditAttandance_FormClosed(object sender,
FormClosedEventArgs e)
                {
                        ParentForm.Show();
                }

                private void buttonDateNow_Click(object sender, EventArgs e)
                {
                        DateTime dateTime = DateTime.Now;
                        string dateTimeStr = dateTime.ToString(@"dd-MM-yyyy HH:mm");
                        maskedTextBoxDateTime.Text = dateTimeStr;
                }

                private void buttonSaveChange_Click(object sender, EventArgs e)
                {
                        Database.Patient patient;
                        Database.Doctor doctor;
                        DateTime dateTime;
                        string MedFind;
                        // Проверим все данные на корректность
                        // Проверим на заполненность
                        if (!textBoxId.Text.IsNullOrEmpty()
```

```
                        && !textBoxHI.Text.IsNullOrEmpty()
                        && maskedTextBoxDateTime.Text.Length == 16
                        && !textBoxMedFind.Text.IsNullOrEmpty())
                    {
                        MedFind = textBoxMedFind.Text;
                        // Проверим на корректность ID и мед. полис
                        // Проверим, состоит ли Id только из цифр
                        {
                            Regex regex = new Regex(@"(\D)",
RegexOptions.IgnoreCase);
                            Match matches = regex.Match(textBoxId.Text);
                            if (matches.Success)
                            {   // в поле содержится что-то кроме цифр
                                labelMsg.Text = "Поле с индентификатором должно
содержать\nтолько цифры.";
                                return;
                            }
                        }
                        // Проверим, состоит ли мед. полис только из цифр
                        {
                            Regex regex = new Regex(@"(\D)",
RegexOptions.IgnoreCase);
                            Match matches = regex.Match(textBoxHI.Text);
                            if (matches.Success)
                            {   // в поле содержится что-то кроме цифр
                                labelMsg.Text = "Поле с индентификатором должно
содержать\nтолько цифры.";
                                return;
                            }
                        }
                        // Проверим, существует ли врач с таким ID
                        using (HospitalAt2Context db = new HospitalAt2Context())
                        {
                            var dbDoc = db.Doctors.Where(p => p.PkIdDoctor ==
Int32.Parse(textBoxId.Text)).SingleOrDefault();
                            if (dbDoc is not null)
                            {
                                doctor = dbDoc;
                            }
                            else
                            {
                                labelMsg.Text = "Врача с таким ID нет";
                                return;
                            }
                        }
                        // Проверим, существует ли пациент с таким мед. полисом
                        using (HospitalAt2Context db = new HospitalAt2Context())
                        {
                            var dbPat = db.Patients.Where(p => p.HealthInsurance
== Int32.Parse(textBoxHI.Text)).SingleOrDefault();
                            if (dbPat is null)
                            {
                                labelMsg.Text = "Пациента с таким мед. полисом
нет";
                                return;
                            }
                            else
                            {
                                patient = dbPat;
                            }
                        }
```

```
                        // Если все верно, то поля
                        //          doctor - содержит информацию о враче
                        //          patient - содержит информацию о пациенте
                        try
                        {
                              // Проверим поле DateTime на корректность
                              dateTime =
DateTime.ParseExact(maskedTextBoxDateTime.Text, "dd-MM-yyyy HH:mm",

System.Globalization.CultureInfo.InvariantCulture);
                              DayOfWeek f = dateTime.DayOfWeek;
                              int? myDayOfWeek;
                              switch (f)
                              {
                                    case DayOfWeek.Monday:
                                          myDayOfWeek = 0;
                                          break;
                                    case DayOfWeek.Tuesday:
                                          myDayOfWeek = 1;
                                          break;
                                    case DayOfWeek.Wednesday:
                                          myDayOfWeek = 2;
                                          break;
                                    case DayOfWeek.Thursday:
                                          myDayOfWeek = 3;
                                          break;
                                    case DayOfWeek.Friday:
                                          myDayOfWeek = 4;
                                          break;
                                    case DayOfWeek.Saturday:
                                          myDayOfWeek = 5;
                                          break;
                                    default:
                                          myDayOfWeek = 6;
                                          break;
                              }
                              using (HospitalAt2Context db = new
HospitalAt2Context())
                              {
                                    DateTime? maxDateInsert = db.Schedules
                                          .Include(p => p.DoctorId)
                                          .Where(p => p.DoctorId!.PkIdDoctor ==
doctor!.PkIdDoctor
                                                && p.DayOfWeek == myDayOfWeek)
                                          .Max(p => p.DateInsert);
                                    if (maxDateInsert is null)
                                    {
                                          labelMsg.Text = "Неправильное время
приема";
                                          return;
                                    }
                                    TimeSpan locTimeApp = dateTime.TimeOfDay;
                                    Schedule? dbSch = db.Schedules
                                          .Include(p => p.DoctorId)
                                          .Where(p => p.DoctorId!.PkIdDoctor ==
doctor!.PkIdDoctor
                                          && p.DayOfWeek == myDayOfWeek
                                          && p.DateInsert == maxDateInsert)
                                          .SingleOrDefault();
                                    if (dbSch is null)
                                    {
```

```
                                        labelMsg.Text = "Неправильное время
приема";
                                        return;
                                    }
                                    else
                                    {
                                        if (!(locTimeApp <= dbSch.TimeEnd &&
locTimeApp >= dbSch.TimeStart))
                                        {
                                            labelMsg.Text = "Неправильное время
приема";
                                            return;
                                        }
                                    }
                                }
                                // Время введено верно

                                // Все данные верны, можно добавить запись в бд
                                using (HospitalAt2Context db = new
HospitalAt2Context())
                                {

                                    Database.Doctor? locDoc = db.Doctors
                                        .Where(p => p.PkIdDoctor ==
doctor.PkIdDoctor)
                                        .SingleOrDefault();
                                    Database.Patient? locPat = db.Patients
                                        .Where(p => p.PkIdPatient ==
patient.PkIdPatient)
                                        .SingleOrDefault();

                                    // Проверим, есть ли уже такая запись
                                    var dbAt = db.Attandances
                                        .Include(p => p.DoctorId)
                                        .Include(p => p.PatientId)
                                        .Where(p => p.FkIdPatient ==
locPat.PkIdPatient
                                                    && p.DateAndTime ==
dateTime)
                                        .SingleOrDefault();
                                    if(dbAt is not null)
                                    {
                                        if(dbAt.DoctorId!.PkIdDoctor !=
locDoc.PkIdDoctor)
                                        {
                                            labelMsg.Text = "В это время
пациент был у другого врача\n"
                                                + "кажется вы неверно ввели
данные о враче";
                                            return;
                                        }
                                        dbAt.MedicalFindings = MedFind;
                                        db.Attandances.Update(dbAt);
                                        db.SaveChanges();
                                        labelMsg.Text = "Запись успешно
изменена";
                                    }
                                    else
                                    {
                                        //Генерация Primary Key
                                        int idAcc;
                                        {
```

```
                                                // Получаем самый большой по
значению первичный ключ
                                                var dbAccId =
db.Attandances.OrderBy(p => p.PkIdAttandance).LastOrDefault();
                                                // Если первичного ключа нет, то
объявляем его как наименьший - '0'
                                                // Иначе увеличиваем наибольший
первичный ключ на 1
                                                if (dbAccId != null)
                                                      idAcc =
dbAccId.PkIdAttandance + 1;
                                                else
                                                      idAcc = 0;
                                          }

                                          Database.Attandance newAt = new
Database.Attandance()
                                          {
                                                PkIdAttandance = idAcc,
                                                DateAndTime = dateTime,
                                                MedicalFindings = MedFind,
                                                FkIdDoctor = locDoc.PkIdDoctor,
                                                DoctorId = locDoc,
                                                FkIdPatient = locPat.PkIdPatient,
                                                PatientId = locPat
                                          };
                                          db.Attandances.Add(newAt);
                                          db.SaveChanges();
                                          labelMsg.Text = "Запись успешно
добавлена";
                                    }

                              }
                        }
                        catch
                        {
                              labelMsg.Text = "Введенное время некорректно";
                              return;
                        }

                  }
                  else
                  {
                        labelMsg.Text = "Заполните все данные";
                        return;
                  }
            }


            private void buttonCheck_Click(object sender, EventArgs e)
            {
                  Database.Patient patient;
                  Database.Doctor doctor;
                  DateTime dateTime;
                  string MedFind;
                  // Проверим все данные на корректность
                  // Проверим на заполненность
                  if (!textBoxId.Text.IsNullOrEmpty()
                        && !textBoxHI.Text.IsNullOrEmpty()
                        && maskedTextBoxDateTime.Text.Length == 16)
                  {
                        MedFind = textBoxMedFind.Text;
```

```
                        // Проверим на корректность ID и мед. полис
                        // Проверим, состоит ли Id только из цифр
                        {
                              Regex regex = new Regex(@"(\D)",
RegexOptions.IgnoreCase);
                              Match matches = regex.Match(textBoxId.Text);
                              if (matches.Success)
                              {   // в поле содержится что-то кроме цифр
                                    labelMsg.Text = "Поле с индентификатором должно
содержать\nтолько цифры.";
                                    return;
                              }
                        }
                        // Проверим, состоит ли мед. полис только из цифр
                        {
                              Regex regex = new Regex(@"(\D)",
RegexOptions.IgnoreCase);
                              Match matches = regex.Match(textBoxHI.Text);
                              if (matches.Success)
                              {   // в поле содержится что-то кроме цифр
                                    labelMsg.Text = "Поле с индентификатором должно
содержать\nтолько цифры.";
                                    return;
                              }
                        }
                        // Проверим, существует ли врач с таким ID
                        using (HospitalAt2Context db = new HospitalAt2Context())
                        {
                              var dbDoc = db.Doctors.Where(p => p.PkIdDoctor ==
Int32.Parse(textBoxId.Text)).SingleOrDefault();
                              if (dbDoc is not null)
                              {
                                    doctor = dbDoc;
                              }
                              else
                              {
                                    labelMsg.Text = "Врача с таким ID нет";
                                    return;
                              }
                        }
                        // Проверим, существует ли пациент с таким мед. полисом
                        using (HospitalAt2Context db = new HospitalAt2Context())
                        {
                              var dbPat = db.Patients.Where(p => p.HealthInsurance
== Int32.Parse(textBoxHI.Text)).SingleOrDefault();
                              if (dbPat is null)
                              {
                                    labelMsg.Text = "Пациента с таким мед. полисом
нет";
                                    return;
                              }
                              else
                              {
                                    patient = dbPat;
                              }
                        }

                        // Если все верно, то поля
                        //          doctor - содержит информацию о враче
                        //          patient - содержит информацию о пациенте
                        try
                        {
```

```
                                // Проверим поле DateTime на корректность
                                dateTime =
DateTime.ParseExact(maskedTextBoxDateTime.Text, "dd-MM-yyyy HH:mm",

System.Globalization.CultureInfo.InvariantCulture);
                                DayOfWeek f = dateTime.DayOfWeek;
                                int? myDayOfWeek;
                                switch (f)
                                {
                                        case DayOfWeek.Monday:
                                                myDayOfWeek = 0;
                                                break;
                                        case DayOfWeek.Tuesday:
                                                myDayOfWeek = 1;
                                                break;
                                        case DayOfWeek.Wednesday:
                                                myDayOfWeek = 2;
                                                break;
                                        case DayOfWeek.Thursday:
                                                myDayOfWeek = 3;
                                                break;
                                        case DayOfWeek.Friday:
                                                myDayOfWeek = 4;
                                                break;
                                        case DayOfWeek.Saturday:
                                                myDayOfWeek = 5;
                                                break;
                                        default:
                                                myDayOfWeek = 6;
                                                break;
                                }
                                using (HospitalAt2Context db = new
HospitalAt2Context())
                                {
                                        DateTime? maxDateInsert = db.Schedules
                                                .Include(p => p.DoctorId)
                                                .Where(p => p.DoctorId!.PkIdDoctor ==
doctor!.PkIdDoctor
                                                        && p.DayOfWeek == myDayOfWeek)
                                                .Max(p => p.DateInsert);
                                        if (maxDateInsert is null)
                                        {
                                                labelMsg.Text = "Неправильное время
приема";
                                                return;
                                        }
                                        TimeSpan locTimeApp = dateTime.TimeOfDay;
                                        Schedule? dbSch = db.Schedules
                                                .Include(p => p.DoctorId)
                                                .Where(p => p.DoctorId!.PkIdDoctor ==
doctor!.PkIdDoctor
                                                && p.DayOfWeek == myDayOfWeek
                                                && p.DateInsert == maxDateInsert)
                                                .SingleOrDefault();
                                        if (dbSch is null)
                                        {
                                                labelMsg.Text = "Неправильное время
приема";
                                                return;
                                        }
                                        else
                                        {
```

```
                                        if (!(locTimeApp <= dbSch.TimeEnd &&
locTimeApp >= dbSch.TimeStart))
                                        {
                                              labelMsg.Text = "Неправильное время
приема";
                                              return;
                                        }
                                  }
                            }
                            // Время введено верно

                            // Все данные верны, можно вывести заключение
                            using (HospitalAt2Context db = new
HospitalAt2Context())
                            {
                                  Database.Doctor? locDoc = db.Doctors
                                        .Where(p => p.PkIdDoctor ==
doctor.PkIdDoctor)
                                        .SingleOrDefault();
                                  Database.Patient? locPat = db.Patients
                                        .Where(p => p.PkIdPatient ==
patient.PkIdPatient)
                                        .SingleOrDefault();

                                  var dbAt = db.Attandances
                                        .Include(p => p.DoctorId)
                                        .Include(p => p.PatientId)
                                        .Where(p => p.FkIdDoctor ==
locDoc.PkIdDoctor
                                                    && p.FkIdPatient ==
locPat.PkIdPatient
                                                    && p.DateAndTime ==
dateTime)
                                        .SingleOrDefault();
                                  if (dbAt is not null)
                                  {
                                        textBoxMedFind.Text =
dbAt.MedicalFindings;
                                        labelMsg.Text = "Запись найдена";
                                  }
                                  else
                                  {
                                        labelMsg.Text = "Запись не найдена";
                                  }
                            }
                      }
                      catch
                      {
                            labelMsg.Text = "Введенное время некорректно";
                            return;
                      }

                }
                else
                {
                      labelMsg.Text = "Заполните все данные";
                      return;
                }
          }
     }
}
using CourseWorkAt4.Database;
```

```
using Microsoft.IdentityModel.Tokens;
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 CourseWorkAt4.Forms
{
	public partial class EditSchedule : Form
	{
		private Form ParentForm;
		public EditSchedule(Form parentForm)
		{
			InitializeComponent();
			ParentForm = parentForm;

			// Заполним textBox с днями недели
			comboBoxDayOfWeek.Items.AddRange(new string[] { "Понедельник",
"Вторник", "Среда", "Четверг", "Пятница", "Суббота", "Воскресение" });
		}

		private void buttonSaveChanges_Click(object sender, EventArgs e)
		{

			int? docId = null;
			// Проверим содержимое Name и ID
			if (textBoxName.Text.IsNullOrEmpty())
			{
				if (!textBoxId.Text.IsNullOrEmpty())
				{
					using (HospitalAt2Context db = new
HospitalAt2Context())
					{
						var dbDoc = db.Doctors.Where(p => p.PkIdDoctor
== Int32.Parse(textBoxId.Text)).SingleOrDefault();
						if (dbDoc is not null)
						{
							docId = dbDoc.PkIdDoctor;
						}
					}
				}
				else
				{
					labelMsg.Text = "Введите имя врача..";
					return;
				}
			}
			// Проверим если ли врач с таким именем
			// и если их несколько, то проверим еще ID
			else
			{
				using (HospitalAt2Context db = new HospitalAt2Context())
				{
					var dbDoc = db.Doctors.Where(p => p.NameD ==
textBoxName.Text).ToList();
					if (dbDoc is null)
					{
```

```
                                    labelMsg.Text = "Такого врача нет в системе";
                                    return;
                              }
                              else if (dbDoc.Count() > 1)
                              {
                                    if (!textBoxId.Text.IsNullOrEmpty())
                                    {
                                          var dbDoc2 = db.Doctors.Where(p =>
p.PkIdDoctor == Int32.Parse(textBoxId.Text)).SingleOrDefault();
                                          if (dbDoc2 is null)
                                          {
                                                labelMsg.Text = "Врачей с таким
именем и ID не найдено.";
                                                return;
                                          }
                                          else
                                          {
                                                docId = dbDoc2.PkIdDoctor;
                                          }
                                    }
                                    else
                                    {
                                          labelMsg.Text = "Врачей с таким именем
несколько, введите дополнительно ID";
                                          return;
                                    }
                              }
                              else if(dbDoc.Count ==1)
                              {
                                    Database.Doctor doc = dbDoc.Single();
                                    docId = doc.PkIdDoctor;
                              }
                        }
                  }

                  // Проверим поля с временем
                  if (maskedTextBoxEndTime.Text.IsNullOrEmpty()
                        || maskedTextBoxEndTime.Text.Length != 5
                        || maskedTextBoxStartTime.Text.IsNullOrEmpty()
                        || maskedTextBoxStartTime.Text.Length != 5)
                  {
                        labelMsg.Text = "Введите время";
                        return;
                  }


                  // Проверим поле DayOfWeek
                  if (comboBoxDayOfWeek.SelectedIndex == -1)
                  {
                        labelMsg.Text = "Выберете день недели";
                        return;
                  }

                  // Проверим интервал времени
                  if(maskedTextBoxRangeTime.Text.IsNullOrEmpty()
                        || maskedTextBoxRangeTime.Text.Length != 5)
                  {
                        labelMsg.Text = "Введите интервал времени приема";
                        return;
                  }

                  // Все данные проверены и корректны
```

```
                    // Добавим запись в базу данных
                    using (HospitalAt2Context db = new HospitalAt2Context())
                    {
                          Database.Doctor? dbDoc;
                          // Если docId = null, то врача определяем по имени
                          if (docId is null)
                          {
                                dbDoc = db.Doctors.Where(p => p.NameD ==
textBoxName.Text).SingleOrDefault();
                          }
                          // Иначе по id
                          else
                          {
                                dbDoc = db.Doctors.Where(p => p.PkIdDoctor ==
docId).SingleOrDefault();
                          }

                          // Определяем переменные для добавления
                          TimeSpan startTime;
                          TimeSpan endTime;
                          int dayOfWeek;
                          TimeSpan rangeTime;
                          try
                          {
                                startTime =
TimeSpan.Parse(maskedTextBoxStartTime.Text);
                                endTime = TimeSpan.Parse(maskedTextBoxEndTime.Text);
                                dayOfWeek = comboBoxDayOfWeek.SelectedIndex;
                                rangeTime =
TimeSpan.Parse(maskedTextBoxRangeTime.Text);
                                var dbCheck = db.Schedules
                                            .Where(p => p.DoctorId == dbDoc
                                                  && p.TimeStart == startTime
                                                  && p.TimeEnd == endTime
                                                  && p.DayOfWeek == dayOfWeek
                                                  && p.RangeTime ==
rangeTime).SingleOrDefault();
                                if (dbCheck != null)
                                {
                                      dbCheck.DateInsert = DateTime.Now;
                                      db.Schedules.Update(dbCheck);
                                      db.SaveChanges();
                                }
                                else
                                {
                                      int idAcc;
                                      //Генерация Primary Key
                                      // Получаем самый большой по значению первичный
ключ
                                      var dbScheduleId = db.Schedules.OrderBy(p =>
p.PkIdSchedule).LastOrDefault();
                                      // Если первичного ключа нет, то объявляем его
как наименьший - '0'
                                      // Иначе увеличиваем наибольший первичный ключ
на 1
                                      if (dbScheduleId != null)
                                            idAcc = dbScheduleId.PkIdSchedule + 1;
                                      else
                                            idAcc = 0;

                                      Schedule newSchedule = new Schedule()
                                      {
```

```
                                        PkIdSchedule = idAcc,
                                        TimeStart = startTime,
                                        TimeEnd = endTime,
                                        DayOfWeek = dayOfWeek,
                                        DateInsert = DateTime.Now,
                                        RangeTime = rangeTime,
                                        FkIdDoctor = dbDoc!.PkIdDoctor,
                                        DoctorId = dbDoc,
                                  };
                                  db.Schedules.Add(newSchedule);
                                  db.SaveChanges();
                            }

                            labelMsg.Text = "Запись успешно добавлена\nВ
расписание";

                      }
                      catch
                      {
                            labelMsg.Text = "Время введено неверно";
                      }
                }
          }

          private void EditSchedule_FormClosed(object sender, FormClosedEventArgs
e)
          {
                ParentForm.Show();
          }
    }
}

using CourseWorkAt4.ClassLogic;
using Microsoft.IdentityModel.Tokens;
using Microsoft.VisualBasic.ApplicationServices;
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 CourseWorkAt4.Forms
{
    public partial class RegistrationPatient : Form
    {
          private Form ParentForm;
          private Database.Patient DBPatient;
          private Hospital User;
          public RegistrationPatient(Form parentForm)
          {
                InitializeComponent();
                ParentForm = parentForm;
                User = new Hospital();
          }

          private void buttonEnter_Click(object sender, EventArgs e)
          {
                List<string> enterDataPatient;
```

```
                    if (!textBoxLogin.Text.IsNullOrEmpty() &&
                          !textBoxPassword.Text.IsNullOrEmpty() &&
                          !textBoxName.Text.IsNullOrEmpty() &&
                          !textBoxSurname.Text.IsNullOrEmpty() &&
                          !textBoxPatronymic.Text.IsNullOrEmpty() &&
                          !textBoxMedH.Text.IsNullOrEmpty())
                    {
                          enterDataPatient = new List<string>() { textBoxLogin.Text,
                                textBoxPassword.Text, textBoxName.Text,
                                textBoxSurname.Text, textBoxPatronymic.Text,
                                textBoxMedH.Text};

                          bool resultRegistration =
User.RegistrationPatient(enterDataPatient);
                          if (resultRegistration)
                          {
                                labelMsg.Text = "Регистрация прошла успешно";
                          }
                          else
                          {
                                labelMsg.Text = "Регистрация не удалась";
                          }
                    }
                    else
                    {
                          labelMsg.Text = "Введите все данные";
                    }
              }

              private void RegistrationPatient_FormClosed(object sender,
FormClosedEventArgs e)
              {
                    ParentForm.Show();
              }
        }

}
          using CourseWorkAt4.Database;
          using CourseWorkAt4.ClassLogic;
          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;
          using Microsoft.IdentityModel.Tokens;
          using static System.Windows.Forms.VisualStyles.VisualStyleElement;
          using System.Security.Cryptography;
          using Microsoft.EntityFrameworkCore;
          using System.Numerics;
          using System.Text.RegularExpressions;
```

```
namespace CourseWorkAt4.Forms
{
    public partial class SignUpAppointment : Form
    {
        private ClassLogic.MedicalSecretary UserMed;
        private ClassLogic.Patient UserPat;
        private Form ParentForm; // ссылка на форму, которая вызвала текущую форму
        public SignUpAppointment(
            Form parentForm, ClassLogic.MedicalSecretary user)
        {
            InitializeComponent();
            ParentForm = parentForm;
            UserMed = user;
        }
        public SignUpAppointment(
            Form parentForm, ClassLogic.Patient user)
        {
            InitializeComponent();
            ParentForm = parentForm;
            UserPat = user;
        }

        private void listView1_RetrieveVirtualItem(object sender, RetrieveVirtualItemEventArgs e)
        {
        }

        private void maskedTextName_MaskInputRejected(object sender, MaskInputRejectedEventArgs e)
        {

        }


        private void buttonWatchSchedule_Click(object sender, EventArgs e)
        {
            WatchSchedule f = new WatchSchedule(this);
            f.Show();

        }
```

```
                    private void buttonSaveChanges_Click(object sender, EventArgs e)
                    {
                            Database.Doctor? doctor = null;
                            Database.Patient? patient= null;
                            DateTime dateApp;
                            //Проверим поля на заполненность
                            if (maskedTextBoxTime.Text.Length == 16
                                    && !textBoxHIPatient.Text.IsNullOrEmpty()
                                    && !textBoxId.Text.IsNullOrEmpty())
                            {
                                    // Проверим, состоит ли Id только из цифр
                                    {
                                            Regex regex = new Regex(@"(\D)",
RegexOptions.IgnoreCase);
                                            Match matches = regex.Match(textBoxId.Text);
                                            if (matches.Success)
                                            {   // в поле содержится что-то кроме цифр
                                                labelMsg.Text = "Поле с индентификатором
должно содержать\nтолько цифры.";
                                                return;
                                            }
                                    }

                                    //     Проверка на корректность поля Name
                                    //     Проверка на наличие чего-то кроме букв
                                    if (!textBoxName.Text.IsNullOrEmpty())
                                    {
                                            Regex regex = new Regex(@"(\d)",
RegexOptions.IgnoreCase);
                                            Match matches = regex.Match(textBoxName.Text);
                                            Regex regex2 = new Regex(@"(\W)",
RegexOptions.IgnoreCase);
                                            Match matches2 = regex.Match(textBoxName.Text);
                                            if (matches.Success || matches2.Success)
                                            {
                                                labelMsg.Text = "Имя может содержать только
буквы";
                                                return;
                                            }
                                    }



                                    try
```

```
                                {
                                        dateApp                                                 =
DateTime.ParseExact(maskedTextBoxTime.Text, "dd-MM-yyyy HH:mm",

System.Globalization.CultureInfo.InvariantCulture);

                                        //Проверим, есть ли запись на это время
                                        using(HospitalAt2Context       db       =       new
HospitalAt2Context())
                                        {
                                                var   dbApp   =   db.Appointments.Where(p   =>
p.DateAndTime== dateApp).SingleOrDefault();
                                                if(dbApp is not null)
                                                {
                                                        labelMsg.Text = "Запись на это время
уже есть";
                                                        return;
                                                }
                                        }


                                        // Проверим, есть ли врач с такими данными
                                        if (!textBoxName.Text.IsNullOrEmpty())
                                        {
                                                using    (HospitalAt2Context    db    =    new
HospitalAt2Context())
                                                {
                                                        var  dbDoc  =  db.Doctors.Where(p  =>
p.NameD == textBoxName.Text).ToList();
                                                        if (dbDoc is null)
                                                        {
                                                                labelMsg.Text  =  "Такого  врача
нет в системе";
                                                                return;
                                                        }
                                                        else if (dbDoc.Count() > 1)
                                                        {
                                                                if
(!textBoxId.Text.IsNullOrEmpty())
                                                                {
                                                                        var       dbDoc2       =
db.Doctors.Where(p => p.PkIdDoctor == Int32.Parse(textBoxId.Text)).SingleOrDefault();
                                                                        if (dbDoc2 is null)
```

```
                                                {
                                                        labelMsg.Text    =
"Врачей с таким именем и ID не найдено.";
                                                        return;
                                                }
                                                else
                                                {
                                                        doctor = dbDoc2;
                                                }
                                        }
                                        else
                                        {
                                                labelMsg.Text  =  "Врачей
с таким именем несколько, введите дополнительно ID";
                                                return;
                                        }
                                }
                                else if (dbDoc.Count == 1)
                                {
                                        Database.Doctor      doc      =
dbDoc.Single();
                                        doctor = doc;
                                }
                        }
                }
                else if (!textBoxId.Text.IsNullOrEmpty())
                {
                        using    (HospitalAt2Context    db    =    new
HospitalAt2Context())
                        {
                                var  dbDoc  =  db.Doctors.Where(p  =>
p.PkIdDoctor == Int32.Parse(textBoxId.Text)).SingleOrDefault();
                                if (dbDoc is not null)
                                {
                                        doctor = dbDoc;
                                }
                                else
                                {
                                        labelMsg.Text = "Врача с таким
ID нет";
                                        return;
                                }
                        }
```

```
                                        }
                                        else
                                        {
                                                labelMsg.Text = "Введите данные о враче";
                                                return;
                                        }

                                        // Есть ли пациент с таким мед. полисом
                                        using(HospitalAt2Context db = new
HospitalAt2Context())
                                        {
                                                var dbPat = db.Patients.Where(p =>
p.HealthInsurance == Int32.Parse(textBoxHIPatient.Text)).SingleOrDefault();
                                                if(dbPat is null)
                                                {
                                                        labelMsg.Text = "Пациента с таким мед.
полисом нет";
                                                        return;
                                                }
                                                else
                                                {
                                                        patient = dbPat;
                                                }
                                        }

                                        // Проверим корректность заданного времени приема
                                        DayOfWeek f = dateApp.DayOfWeek;
                                        int? myDayOfWeek;
                                        switch (f)
                                        {
                                                case DayOfWeek.Monday:
                                                        myDayOfWeek = 0;
                                                        break;
                                                case DayOfWeek.Tuesday:
                                                        myDayOfWeek = 1;
                                                        break;
                                                case DayOfWeek.Wednesday:
                                                        myDayOfWeek = 2;
                                                        break;
                                                case DayOfWeek.Thursday:
                                                        myDayOfWeek = 3;
                                                        break;
```

```
                                        case DayOfWeek.Friday:
                                                myDayOfWeek = 4;
                                                break;
                                        case DayOfWeek.Saturday:
                                                myDayOfWeek = 5;
                                                break;
                                        default:
                                                myDayOfWeek = 6;
                                                break;
                                }
                                using(HospitalAt2Context       db       =       new
HospitalAt2Context() )
                                {
                                        DateTime? maxDateInsert = db.Schedules
                                                .Include(p => p.DoctorId)
                                                .Where(p => p.DoctorId!.PkIdDoctor ==
doctor!.PkIdDoctor
                                                        && p.DayOfWeek == myDayOfWeek)
                                                .Max(p => p.DateInsert);
                                        if(maxDateInsert is null)
                                        {
                                                labelMsg.Text  =  "Неправильное  время
приема";
                                                return;
                                        }
                                        TimeSpan locTimeApp = dateApp.TimeOfDay;
                                        Schedule? dbSch = db.Schedules
                                                .Include(p => p.DoctorId)
                                                .Where(p => p.DoctorId!.PkIdDoctor ==
doctor!.PkIdDoctor
                                                && p.DayOfWeek == myDayOfWeek
                                                && p.DateInsert == maxDateInsert)
                                                .SingleOrDefault();
                                        if(dbSch is null)
                                        {
                                                labelMsg.Text  =  "Неправильное  время
приема";
                                                return;
                                        }
                                        else
                                        {
                                                if(!(locTimeApp<dbSch.TimeEnd      &&
locTimeApp>dbSch.TimeStart))
```

```
                                                {
                                                        labelMsg.Text = "Неправильное
время приема";
                                                        return;
                                                }
                                        }
                                }

                                // Все данные верны, есть:
                                //            врач
                                //            пациент
                                //            Время для записи
                                // Добавим запись в базу данных

                                using (HospitalAt2Context db = new
HospitalAt2Context())
                                {
                                        Database.Doctor? locDoc = db.Doctors
                                                .Where(p => p.PkIdDoctor ==
doctor.PkIdDoctor)
                                                .SingleOrDefault();
                                        Database.Patient? locPat = db.Patients
                                                .Where(p => p.PkIdPatient ==
patient.PkIdPatient)
                                                .SingleOrDefault();

                                        //Генерация Primary Key
                                        //  Получаем самый большой по значению
первичный ключ
                                        var dbAccId = db.Appointments.OrderBy(p =>
p.PkIdAppointment).LastOrDefault();
                                        // Если первичного ключа нет, то объявляем
его как наименьший - '0'
                                        //  Иначе увеличиваем наибольший первичный
ключ на 1
                                        int idAcc;
                                        if (dbAccId != null)
                                                idAcc = dbAccId.PkIdAppointment + 1;
                                        else
                                                idAcc = 0;
                                        Appointment newApp = new Appointment()
                                        {
                                                PkIdAppointment = idAcc
```

```
                                                ,DateAndTime = dateApp
                                                ,FkIdDoctor = locDoc!.PkIdDoctor
                                                ,DoctorId = locDoc
                                                ,FkIdPatient = locPat.PkIdPatient
                                                ,PatientId = locPat
                                        };
                                        db.Appointments.Add(newApp);
                                        db.SaveChanges();
                                        labelMsg.Text = "Данные успешно сохранены";
                                }
                        }
                        catch
                        {
                                labelMsg.Text = "Время введено неверно";
                        }
                }
                else
                {
                        labelMsg.Text = "Введенные данные не корректны";
                }
        }

        private void buttonWatchAppoinment_Click(object sender, EventArgs e)
        {
                // Для просмотра записей на прием нужны данные:
                //            Врача, к которому производится запись
                //            День, на который планируется запись

                Database.Doctor? doctor = null;
                DateTime dateApp;
                Database.Schedule? schedule = null;
                // Проверим, есть ли врач с такими данными
                if (!textBoxName.Text.IsNullOrEmpty())
                {
                        {
                                Regex      regex      =      new      Regex(@"(\d)",
RegexOptions.IgnoreCase);
                                Match matches = regex.Match(textBoxName.Text);
                                Regex      regex2      =      new      Regex(@"(\W)",
RegexOptions.IgnoreCase);
                                Match matches2 = regex.Match(textBoxName.Text);
                                if (matches.Success || matches2.Success)
                                {
```

```
                                        labelMsg.Text = "Имя может содержать только
буквы";
                                        return;
                                }
                        }
                        if(!textBoxId.Text.IsNullOrEmpty())
                        {
                                Regex regex = new Regex(@"(\D)",
RegexOptions.IgnoreCase);
                                Match matches = regex.Match(textBoxId.Text);
                                if (matches.Success)
                                {   // в поле содержится что-то кроме цифр
                                        labelMsg.Text = "Поле с индентификатором
должно содержать\nтолько цифры.";
                                        return;
                                }

                        }

                        using (HospitalAt2Context db = new HospitalAt2Context())
                        {
                                var dbDoc = db.Doctors.Where(p => p.NameD ==
textBoxName.Text).ToList();
                                if (dbDoc is null)
                                {
                                        labelMsg.Text = "Такого врача нет в системе";
                                        return;
                                }
                                else if (dbDoc.Count() > 1)
                                {
                                        if (!textBoxId.Text.IsNullOrEmpty())
                                        {
                                                var dbDoc2 = db.Doctors.Where(p =>
p.PkIdDoctor == Int32.Parse(textBoxId.Text)).SingleOrDefault();
                                                if (dbDoc2 is null)
                                                {
                                                        labelMsg.Text = "Врачей с таким
именем и ID не найдено.";
                                                        return;
                                                }
                                                else
                                                {
                                                        doctor = dbDoc2;
```

```
                                                }
                                        }
                                        else
                                        {
                                                labelMsg.Text = "Врачей с таким именем
несколько, введите дополнительно ID";
                                                return;
                                        }
                                }
                                else if (dbDoc.Count == 1)
                                {
                                        Database.Doctor doc = dbDoc.Single();
                                        doctor = doc;
                                }
                        }
                }
                else if (!textBoxId.Text.IsNullOrEmpty())
                {
                        using (HospitalAt2Context db = new HospitalAt2Context())
                        {
                                var dbDoc = db.Doctors.Where(p => p.PkIdDoctor ==
Int32.Parse(textBoxId.Text)).SingleOrDefault();
                                if (dbDoc is not null)
                                {
                                        doctor = dbDoc;
                                }
                                else
                                {
                                        labelMsg.Text = "Врача с таким ID нет";
                                        return;
                                }
                        }

                }
                else
                {
                        labelMsg.Text = "Введите данные о враче";
                        return;
                }
                try
                {
                        dateApp   =   DateTime.ParseExact(maskedTextBoxTime.Text,
"dd-MM-yyyy HH:mm",
```

```
System.Globalization.CultureInfo.InvariantCulture);
                                    // Проверим корректность заданного времени приема
                                    //           Время записи должно находится в рамках
времени приема
                                    DayOfWeek f = dateApp.DayOfWeek;
                                    int? myDayOfWeek;
                                    switch (f)
                                    {
                                           case DayOfWeek.Monday:
                                                  myDayOfWeek = 0;
                                                  break;
                                           case DayOfWeek.Tuesday:
                                                  myDayOfWeek = 1;
                                                  break;
                                           case DayOfWeek.Wednesday:
                                                  myDayOfWeek = 2;
                                                  break;
                                           case DayOfWeek.Thursday:
                                                  myDayOfWeek = 3;
                                                  break;
                                           case DayOfWeek.Friday:
                                                  myDayOfWeek = 4;
                                                  break;
                                           case DayOfWeek.Saturday:
                                                  myDayOfWeek = 5;
                                                  break;
                                           default:
                                                  myDayOfWeek = 6;
                                                  break;
                                    }
                                    using (HospitalAt2Context db = new HospitalAt2Context())
                                    {
                                           DateTime? maxDateInsert = db.Schedules
                                                  .Include(p => p.DoctorId)
                                                  .Where(p    =>    p.DoctorId!.PkIdDoctor    ==
doctor!.PkIdDoctor
                                                         && p.DayOfWeek == myDayOfWeek)
                                                  .Max(p => p.DateInsert);
                                           if (maxDateInsert is null)
                                           {
                                                  labelMsg.Text = "Неправильное время приема";
                                                  return;
```

```
                                        }
                                        TimeSpan locTimeApp = dateApp.TimeOfDay;
                                        Schedule? dbSch = db.Schedules
                                               .Include(p => p.DoctorId)
                                               .Where(p    =>    p.DoctorId!.PkIdDoctor    ==
doctor!.PkIdDoctor
                                               && p.DayOfWeek == myDayOfWeek
                                               && p.DateInsert == maxDateInsert)
                                               .SingleOrDefault();
                                        if (dbSch is null)
                                        {
                                               labelMsg.Text = "Неправильное время приема";
                                               return;
                                        }
                                        else
                                        {
                                               schedule = dbSch;
                                               if    (!(locTimeApp    <    dbSch.TimeEnd    &&
locTimeApp > dbSch.TimeStart))
                                               {
                                                      labelMsg.Text  =  "Неправильное  время
приема";
                                                      return;
                                               }
                                        }
                                 }

                                 WatchAppoinment form = new WatchAppoinment(this, doctor,
schedule, dateApp);
                                 form.Show();
                          }
                          catch
                          {
                                 labelMsg.Text = "Время введено неверно";
                          }
                   }
             }

}
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 CourseWorkAt4.Forms
{
	public partial class UserAdmin : Form
	{
		private Form ParentForm;
		public UserAdmin(Form parentForm)
		{
			InitializeComponent();
			ParentForm = parentForm;
		}

		private void UserAdmin_FormClosed(object sender, FormClosedEventArgs e)
		{
			ParentForm.Show();
		}

		private void buttonAddUser_Click(object sender, EventArgs e)
		{
			AdministrationUser f = new AdministrationUser(this, "Admin");
			f.Show();
		}

		private void buttonAddAcc_Click(object sender, EventArgs e)
		{
			AdministrationAcc f = new AdministrationAcc(this, "Admin");
			f.Show();
		}
	}
}
using CourseWorkAt4.ClassLogic;
using Microsoft.VisualBasic.ApplicationServices;
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 CourseWorkAt4.Forms
{
	public partial class UserDoctor : Form
	{
		private Doctor user;
		private Form ParentForm; // ссылка на форму, которая вызвала текущую
форму
		public UserDoctor(Form parentForm, string login)
		{
			InitializeComponent();
			user = new Doctor(login);
			ParentForm = parentForm;
		}
		private void buttonSingUpAppointment_Click(object sender, EventArgs e)
		{
			//SignUpAppointment f = new SignUpAppointment(this, User);
			//f.Show();
		}
```

```
            private void UserDoctor_FormClosed(object sender, FormClosedEventArgs e)
            {
                  user.Destruction();
                  ParentForm.Show();
            }

            private void buttonWatchSchedule_Click(object sender, EventArgs e)
            {
                  WatchSchedule f = new WatchSchedule(this);
                  f.Show();
            }

            private void buttonWatchAttandance_Click(object sender, EventArgs e)
            {
                  WatchAttandance f = new WatchAttandance(this);
                  f.Show();
            }

            private void buttonEditAttandance_Click(object sender, EventArgs e)
            {
                  EditAttandance f = new EditAttandance(this);
                  f.Show();
            }
      }
}
using CourseWorkAt4.Forms;
using CourseWorkAt4.ClassLogic;
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 CourseWorkAt4.Forms
{
      public partial class UserMedS : Form
      {
            private MedicalSecretary User;
            private Form ParentForm; // ссылка на форму, которая вызвала текущую
форму
            public UserMedS(Form parentForm, string login)
            {
                  InitializeComponent();
                  User = new MedicalSecretary(login);
                  ParentForm = parentForm;
            }

            private void UserMedS_FormClosed(object sender, FormClosedEventArgs e)
            {
                  User.Destruction();
                  //Application.OpenForms[0].Show();
                  ParentForm.Show();
            }

            private void buttonSingUpAppointment_Click(object sender, EventArgs e)
            {
                  SignUpAppointment f = new SignUpAppointment(this, User);
                  f.Show();
```

```
            }

            private void buttonWatchSchedule_Click(object sender, EventArgs e)
            {
                  WatchSchedule f = new WatchSchedule(this);
                  f.Show();
            }

            private void buttonAddAcc_Click(object sender, EventArgs e)
            {
                  AdministrationAcc f = new AdministrationAcc(this, "Medical
Secretary");
                  f.Show();
            }

            private void buttonAddUser_Click(object sender, EventArgs e)
            {
                  AdministrationUser f = new AdministrationUser(this, "Medical
Secretary");
                  f.Show();
            }

            private void buttonEditSchedule_Click(object sender, EventArgs e)
            {
                  EditSchedule f = new EditSchedule(this);
                  f.Show();
            }

            private void buttonCheckSchedule_Click(object sender, EventArgs e)
            {
                  CheckIn f = new CheckIn(this);
                  f.Show();
            }
      }
}
using CourseWorkAt4.ClassLogic;
using Microsoft.VisualBasic.ApplicationServices;
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 CourseWorkAt4.Forms
{
      public partial class UserPatient : Form
      {
            private ClassLogic.Patient User;
            private Form ParentForm; // ссылка на форму, которая вызвала текущую
форму
            public UserPatient(Form parentForm, string login)
            {
                  InitializeComponent();
                  User = new Patient(login);
                  ParentForm = parentForm;
            }
            private void buttonSingUpAppointment_Click(object sender, EventArgs e)
            {
```

```
                    SignUpAppointment f = new SignUpAppointment(this, User);
                    f.Show();
             }

             private void UserPatient_FormClosed(object sender, FormClosedEventArgs
e)
             {
                    User.Destruction();
                    ParentForm.Show();
             }

             private void buttonWatchSchedule_Click(object sender, EventArgs e)
             {
                    WatchSchedule f = new WatchSchedule(this);
                    f.Show();
             }

             private void buttonEditService_Click(object sender, EventArgs e)
             {
                    try
                    {
                           WatchAttandance f = new WatchAttandance(this, User);
                           f.Show();
                    }
                    catch
                    {

                    }
             }
       }
}
using CourseWorkAt4.Database;
using Microsoft.EntityFrameworkCore;
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 CourseWorkAt4.Forms
{
       public partial class WatchAppoinment : Form
       {
             Form ParentForm;
             Database.Doctor Doc;
             DateTime DateApp;
             Database.Schedule Sch;
             public WatchAppoinment(Form parentForm, Database.Doctor
doctor,Database.Schedule schedule, DateTime dateApp)
             {
                    InitializeComponent();
                    ParentForm = parentForm;
                    Doc = doctor;
                    DateApp = dateApp;
                    Sch = schedule;

                    labelName.Text = doctor.NameD;
                    labelDate.Text = dateApp.ToString(@"dd-MM-yyyy");
```

```
                    labelTime.Text =
schedule.TimeStart.GetValueOrDefault().ToString(@"hh\:mm")
                                                                    + " - "
                                                                    +
schedule.TimeEnd.GetValueOrDefault().ToString(@"hh\:mm");
                    using(HospitalAt2Context db = new HospitalAt2Context())
                    {
                          var dbApp = db.Appointments
                                .Include(p => p.DoctorId)
                                .Include(p => p.PatientId)
                                .Where(p => p.FkIdDoctor == doctor.PkIdDoctor)
                                .ToList();
                          if (dbApp != null)
                          {
                                foreach (var oneApp in dbApp)
                                {
                                      if (oneApp.DateAndTime.GetValueOrDefault().Date
== dateApp.Date)
                                      {
                                            ListViewItem app =
listViewApp.Items.Add(new ListViewItem(oneApp.PatientId.NamePatient));

      app.SubItems.Add(oneApp.DateAndTime.GetValueOrDefault().ToString(@"HH:mm"));
                                      }
                                }
                          }
                    }
              }

              private void WatchAppoinment_FormClosed(object sender,
FormClosedEventArgs e)
              {
                    ParentForm.Show();
              }
       }
}
using CourseWorkAt4.Database;
using Microsoft.EntityFrameworkCore;
using Microsoft.IdentityModel.Tokens;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace CourseWorkAt4.Forms
{
       public partial class WatchAttandance : Form
       {
              Form ParentForm;
              ClassLogic.Patient? UserPat = null;
              public WatchAttandance(Form parentForm)
              {
                    InitializeComponent();
                    ParentForm = parentForm;
                    labelId.Text = "ID врача";
              }
```

```
            public WatchAttandance(Form parentForm, ClassLogic.Patient user)
            {
                  InitializeComponent();
                  ParentForm = parentForm;
                  UserPat = user;
                  int? HIUser =null;
                  using(HospitalAt2Context db = new HospitalAt2Context())
                  {
                        // Найдем мед полис пациента
                        var dbHIUser = db.Patients
                              .Where(p => p.PkIdPatient == user.PatId)
                              .SingleOrDefault();
                        if (dbHIUser != null)
                        {
                              HIUser = dbHIUser.HealthInsurance;
                        }
                        else
                        {
                              this.Close();
                              return;
                        }
                  }
                  textBoxHI.Text = HIUser!.ToString();
                  textBoxHI.ReadOnly= true;
                  labelMsg.Text = "Введите дату и время, когда у вас был прием";
                  textBoxId.ReadOnly= true;
                  labelId.Text = "Специальность и имя врача";
            }


            private void WatchAttandance_FormClosed(object sender,
FormClosedEventArgs e)
            {
                  ParentForm.Show();
            }

            private void buttonCheck_Click_1(object sender, EventArgs e)
            {
                  Database.Patient patient;
                  Database.Doctor doctor;
                  DateTime dateTime;
                  string MedFind;
                  // Проверим все данные на корректность
                  // Проверим на заполненность
                  if (!textBoxHI.Text.IsNullOrEmpty()
                        && maskedTextBoxDateTime.Text.Length == 16)
                  {
                        MedFind = textBoxMedFind.Text;
                        // Проверим на корректность ID и мед. полис
                        // Проверим, состоит ли мед. полис только из цифр
                        {
                              Regex regex = new Regex(@"(\D)",
RegexOptions.IgnoreCase);
                              Match matches = regex.Match(textBoxHI.Text);
                              if (matches.Success)
                              {   // в поле содержится что-то кроме цифр
                                    labelMsg.Text = "Поле с индентификатором должно
содержать\nтолько цифры.";
                                    return;
                              }
                        }
                        // Проверим, существует ли пациент с таким мед. полисом
```

```
                        using (HospitalAt2Context db = new HospitalAt2Context())
                        {
                              var dbPat = db.Patients.Where(p => p.HealthInsurance
== Int32.Parse(textBoxHI.Text)).SingleOrDefault();
                              if (dbPat is null)
                              {
                                    labelMsg.Text = "Пациента с таким мед. полисом
нет";
                                    return;
                              }
                              else
                              {
                                    patient = dbPat;
                              }
                        }

                        // Если все верно, то поля
                        //          doctor - содержит информацию о враче
                        //          patient - содержит информацию о пациенте
                        try
                        {
                              // Проверим поле DateTime на корректность
                              dateTime =
DateTime.ParseExact(maskedTextBoxDateTime.Text, "dd-MM-yyyy HH:mm",

System.Globalization.CultureInfo.InvariantCulture);
                              DayOfWeek f = dateTime.DayOfWeek;
                              int? myDayOfWeek;
                              switch (f)
                              {
                                    case DayOfWeek.Monday:
                                          myDayOfWeek = 0;
                                          break;
                                    case DayOfWeek.Tuesday:
                                          myDayOfWeek = 1;
                                          break;
                                    case DayOfWeek.Wednesday:
                                          myDayOfWeek = 2;
                                          break;
                                    case DayOfWeek.Thursday:
                                          myDayOfWeek = 3;
                                          break;
                                    case DayOfWeek.Friday:
                                          myDayOfWeek = 4;
                                          break;
                                    case DayOfWeek.Saturday:
                                          myDayOfWeek = 5;
                                          break;
                                    default:
                                          myDayOfWeek = 6;
                                          break;
                              }
                              if (UserPat is not null)
                              {
                                    using(HospitalAt2Context db = new
HospitalAt2Context())
                                    {
                                          // Проверим, есть ли запись в Attandance
                                          // С таким пользователем и таким временем
                                          var dbAt = db.Attandances
                                                .Include(p => p.DoctorId)
                                                .Include(p => p.PatientId)
```

```
                                                .Where(p =>
p.PatientId!.PkIdPatient == patient.PkIdPatient
                                                                        && p.DateAndTime
== dateTime)
                                                .SingleOrDefault();
                                          if(dbAt is not null)
                                          {
                                                // Запись есть и мы можем
определить врача
                                                doctor = dbAt.DoctorId!;
                                                textBoxId.Text =
dbAt.DoctorId!.Speciality  + ", " + dbAt.DoctorId!.NameD;
                                          }
                                          else
                                          {
                                                labelMsg.Text = "На этот день
записи нет";
                                                return;
                                          }
                                    }
                              }
                              else
                              {
                                    if (!textBoxId.Text.IsNullOrEmpty())
                                    {
                                          // Проверим, состоит ли Id только из цифр
                                          {
                                                Regex regex = new Regex(@"(\D)",
RegexOptions.IgnoreCase);
                                                Match matches =
regex.Match(textBoxId.Text);
                                                if (matches.Success)
                                                {   // в поле содержится что-то
кроме цифр
                                                      labelMsg.Text = "Поле с
индентификатором должно содержать\nтолько цифры.";
                                                      return;
                                                }
                                          }
                                          // Проверим, существует ли врач с таким
ID
                                          using (HospitalAt2Context db = new
HospitalAt2Context())
                                          {
                                                var dbDoc = db.Doctors.Where(p =>
p.PkIdDoctor == Int32.Parse(textBoxId.Text)).SingleOrDefault();
                                                if (dbDoc is not null)
                                                {
                                                      doctor = dbDoc;
                                                }
                                                else
                                                {
                                                      labelMsg.Text = "Врача с
таким ID нет";
                                                      return;
                                                }
                                          }
                                    }
                                    else
                                    {
                                          labelMsg.Text = "Введите все данные";
                                          return;
```

```
                                        }
                                }
                                using (HospitalAt2Context db = new
HospitalAt2Context())
                                {
                                        DateTime? maxDateInsert = db.Schedules
                                                .Include(p => p.DoctorId)
                                                .Where(p => p.DoctorId!.PkIdDoctor ==
doctor!.PkIdDoctor
                                                        && p.DayOfWeek == myDayOfWeek)
                                                .Max(p => p.DateInsert);
                                        if (maxDateInsert is null)
                                        {
                                                labelMsg.Text = "Неправильное время
приема";
                                                return;
                                        }
                                        TimeSpan locTimeApp = dateTime.TimeOfDay;
                                        Schedule? dbSch = db.Schedules
                                                .Include(p => p.DoctorId)
                                                .Where(p => p.DoctorId!.PkIdDoctor ==
doctor!.PkIdDoctor
                                                && p.DayOfWeek == myDayOfWeek
                                                && p.DateInsert == maxDateInsert)
                                                .SingleOrDefault();
                                        if (dbSch is null)
                                        {
                                                labelMsg.Text = "Неправильное время
приема";
                                                return;
                                        }
                                        else
                                        {
                                                if (!(locTimeApp <= dbSch.TimeEnd &&
locTimeApp >= dbSch.TimeStart))
                                                {
                                                        labelMsg.Text = "Неправильное время
приема";
                                                        return;
                                                }
                                        }
                                }
                                // Время введено верно

                                // Все данные верны, можно вывести заключение
                                using (HospitalAt2Context db = new
HospitalAt2Context())
                                {
                                        Database.Doctor? locDoc = db.Doctors
                                                .Where(p => p.PkIdDoctor ==
doctor.PkIdDoctor)
                                                .SingleOrDefault();
                                        Database.Patient? locPat = db.Patients
                                                .Where(p => p.PkIdPatient ==
patient.PkIdPatient)
                                                .SingleOrDefault();

                                        var dbAt = db.Attandances
                                                .Include(p => p.DoctorId)
                                                .Include(p => p.PatientId)
                                                .Where(p => p.FkIdDoctor ==
locDoc.PkIdDoctor
```

```
                                                              && p.FkIdPatient ==
locPat.PkIdPatient
                                                              && p.DateAndTime ==
dateTime)
                                                .SingleOrDefault();
                                          if (dbAt is not null)
                                          {
                                                textBoxMedFind.Text =
dbAt.MedicalFindings;
                                                labelMsg.Text = "Запись найдена";
                                          }
                                          else
                                          {
                                                labelMsg.Text = "Запись не найдена";
                                          }
                                    }
                              }
                              catch
                              {
                                    labelMsg.Text = "Введенное время некорректно";
                                    return;
                              }

                        }
                        else
                        {
                              labelMsg.Text = "Заполните все данные";
                              return;
                        }
                  }

                  private void buttonDateNow_Click(object sender, EventArgs e)
                  {
                        DateTime dateTime = DateTime.Now;
                        string dateTimeStr = dateTime.ToString(@"dd-MM-yyyy HH:mm");
                        maskedTextBoxDateTime.Text = dateTimeStr;
                  }
            }
}
using CourseWorkAt4.Database;
using Microsoft.EntityFrameworkCore;
using Microsoft.IdentityModel.Tokens;
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 CourseWorkAt4.Forms
{
      public partial class WatchSchedule : Form
      {
            private class DaySchedule
            {
                  public string? TimeOfWork { get; set; }
                  public int? DayOfWeek { get; set; }
            }
```

```
            private Form ParentForm; // ссылка на форму, которая вызвала текущую
форму
            public WatchSchedule(
                  Form parentForm)
            {
                  InitializeComponent();
                  ParentForm = parentForm;

                  // Заполнение элементами comboBoxSpeciality
                  using(HospitalAt2Context db = new HospitalAt2Context())
                  {
                        var specialityList = db.Doctors.Select(x =>
x.Speciality).Distinct().ToList();
                        foreach(var speciality in specialityList)
                        {
                              comboBoxSpeciality.Items.Add(speciality);
                        }
                  }


                  }

            private void comboBoxSpeciality_SelectedIndexChanged(object sender,
EventArgs e)
            {
                  // Заполнение элементами comboBoxName
                  string specialitySelect =
comboBoxSpeciality.SelectedItem.ToString();

                  // заполняем расписание врачей с выбранной специальностью
                  using(HospitalAt2Context db = new HospitalAt2Context())
                  {
                        listViewSchedule.Items.Clear();
                        var doctorList = db.Doctors.Where(p => p.Speciality ==
specialitySelect).ToList();
                        foreach (var doctor in doctorList)
                        {
                              // weekSchedule содержит расписание текущего врача,
которое есть в базе данных
                              List<DaySchedule> weekSchedule = new
List<DaySchedule>();
                              var dbDOW = db.Schedules
                                    .Include(p => p.DoctorId)
                                    .Where(p => p.DoctorId.PkIdDoctor ==
doctor.PkIdDoctor)
                                    .Select(p => new
                                    {
                                          DayOfWeek = p.DayOfWeek!
                                    })
                                    .Distinct()
                                    .OrderBy(p => p.DayOfWeek)
                                    .ToList();
                              foreach(var day in dbDOW)
                              {
                                    var maxDI = db.Schedules
                                          .Include(p => p.DoctorId)
                                          .Where(p => p.DoctorId.PkIdDoctor ==
doctor.PkIdDoctor
                                                && p.DayOfWeek == day.DayOfWeek)
                                          .Max(p => p.DateInsert);
                                    // Определено:
                                    //          Врач
```

```
                                        //          День недели
                                        //          Момент последнего добавления
расписания
                                        var dbDaySch = db.Schedules
                                              .Include(p => p.DoctorId)
                                              .Where(p => p.DoctorId.PkIdDoctor ==
doctor.PkIdDoctor
                                                    && p.DayOfWeek == day.DayOfWeek
                                                    && p.DateInsert == maxDI)
                                              .SingleOrDefault();

                                        weekSchedule.Add(new DaySchedule()
                                        {
                                              //TimeOfWork = string.Format("{0:t}",
(dbDaySch!.TimeStart.ToString()))
                                              //                + ":" +
string.Format("{0:t}", (dbDaySch.TimeEnd.ToString()))
                                              TimeOfWork =
dbDaySch!.TimeStart.GetValueOrDefault().ToString(@"hh\:mm")
                                                    + " - "
                                                    +
dbDaySch.TimeEnd.GetValueOrDefault().ToString(@"hh\:mm")
                                              ,DayOfWeek = day.DayOfWeek
                                        });
                                  }

                                  // Добавляем данные на экран
                                  ListViewItem doctorView =
listViewSchedule.Items.Add(new ListViewItem(doctor.NameD));
                                  for (int i = 0; i <7; i++)
                                  {
                                        string? timeOfWork = null;
                                        foreach(var loc in weekSchedule)
                                        {
                                              if(i == loc.DayOfWeek)
                                                    timeOfWork= loc.TimeOfWork;
                                        }
                                        if (!timeOfWork.IsNullOrEmpty())
                                              doctorView.SubItems.Add(timeOfWork);
                                        else
                                              doctorView.SubItems.Add("-");
                                  }

                            }
                      }
                }

          }
}
```