Министерство образования и молодежной политики

Свердловской области

ГАПОУ СО «Екатеринбургский колледж транспортного строительства»

Специальность 09.02.07: «Информационные системы и программирование»

Разработка информационной системы «Разработка подсистем для CRM системы мед. центра»

**Пояснительная записка**

к курсовому проекту

КР-ПР-32-01-2023-ПЗ

Разработал:

Студент гр. ПР-32 \_\_\_\_\_\_\_\_\_\_\_\_\_\_/Д.А. Безменов

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_/Д.О. Гариев

2023

Министерство образования и молодежной политики

Свердловской области

ГАПОУ СО «Екатеринбургский колледж транспортного строительства»

Специальность 09.02.07: «Информационные системы и программирование»

Разработка информационной системы «Разработка подсистем для CRM системы мед. центра»

**Курсовой проект**

КП-ПР-32-01-2023

2023

Оглавление

[Введение 4](#_Toc132184531)

[1 Постановка задачи 5](#_Toc132184532)

[1.1 Описание предметной области 5](#_Toc132184533)

[1.2 Требования к программному продукту 7](#_Toc132184534)

[2 Проектирование логической структуры 10](#_Toc132184535)

[3 Техническое проектирование 18](#_Toc132184536)

[3.1 Выбор состава технических программных средств 18](#_Toc132184537)

[3.2 Физическая структура программы 20](#_Toc132184538)

[4 Тестирование 21](#_Toc132184539)

[Заключение 22](#_Toc132184540)

[Список использованной литературы 23](#_Toc132184541)

# Введение

Сейчас в медицинской клинике уже используется CRM система, но она имеет несколько недостатков. В ней очень сложно разобраться, она имеет множество кнопок, выпадающие списки и из-за этого в данной системе не легко ориентироваться. Я собираюсь написать новую, более удобную и практичную систему, состоящую из нескольких подсистем, при этом сохранив весь основной функционал. В моей системе будет максимально легко найти нужную вкладку, страницу.

# 1 Постановка задачи

## Описание предметной области

Клиника семейной медицины «Ваш доктор» — это частный медицинский центр для всей семьи, основанный в 2011 году. В данной клинике работают опытные доктора – терапевты, узкие специалисты, диагносты. В данной клинике уже вылечена ни одна сотня детей и взрослых.

Направления клиники:

* детское отделение. Заботимся о здоровье ребенка от зачатия до совершеннолетия. В клинике принимают педиатры, детский невролог и хирург-ортопед;
* взрослое отделение. Первичная консультация и терапия, а также лечение у узких специалистов - кардиолога и гастроэнтеролога;
* диагностика. Качественная диагностика – первый шаг на пути к выздоровлению и крепкому здоровью;
* косметология. Аппаратная и инъекционная косметология;
* медосмотры;
* физиотерапия;

Записаться на прием можно как по телефону, так и онлайн с помощью сайта клиники.

CRM система или же «управление отношениями с клиентами» - программа, упрощающая и автоматизирующая работу с пациентами в поликлинике.

Перед приемом пациента заносят в базу, а если он уже есть в базе, то его записывают на прием. Так же для каждого пациента создается амбулаторная карта, которую заполняет врач, в ней записаны все посещения приемов, диагнозы, информация о пациенте.

Так же врачи с помощью CRM системы могут отслеживать свое расписание, просматривать пациентов, записанных к ним на прием.

При принятии нового врача на работу, его заносят в базу врачей, выдают уникальный логин и пароль для авторизации в CRM системе.

В подсистеме расписания администратор сможет выбирать дни, в которые будет работать врач, а также время начала и конца работы и время обеда. Врач же сможет только просматривать данную информацию и не будет иметь возможности редактировать ее

В подсистеме расписания администратор сможет составлять график работы врачей на следующую неделю, а именно выбирать рабочие и выходные дни, выбирать время начала и конца работы и время обеда. Врачи же в данной подсистеме смогут только отслеживать свое расписание.

На складе тоже используется данная CRM система. При поступлении товара, его заносят в базу. В базе хранится информация о том, сколько товара сейчас хранится на складе, его цены, информации о поставщиках.

Сейчас в клинике уже есть CRM система, позволяющая администратору смотреть информацию о всех врачах, их расписании, пациентах и их истории лечений, записывать пациентов на прием и регистрировать тех, кто впервые пришел на прием, принимать оплату. Так же можно отслеживать расходные материалы и медикаменты, поступившие и находящиеся на складе, составлять различные отчеты за определенные временные промежутки.

Система, которая используется в данный момент, была написана в 2011 году. За это время она сильно устарела. Ее интерфейс интуитивно не понятен, некоторые функции работают долго, дизайн не красивый. Я напишу подсистему просмотра пациентов и составления расписания.

## Требования к программному продукту

Будет написано две подсистемы для CRM системы клиники. Первая подсистема – пациенты. Она будет связана с подсистемами записей на прием и медицинскими картами. Вторая – расписание врачей. Она будет связана с подсистемами врачей и записями на прием.

Сверху программы будут находится кнопки навигации для переходов между различными экранами.

При переходе между экранами на предыдущем должна запоминаться несохраненная информация для избегания потери информации при случайном нажатии куда-либо. Так же после завершения редактирования какой-либо информации и нажатия кнопок «принять изменения» должно будет появляться окно с вопросом «Действительно ли вы хотите сохранить изменения?».

Для подсистемы пациентов потребуется создать базу данных, в которой будет хранится следующая информация:

* фамилия, имя, отчество пациента в строковом формате;
* пол – ссылочный тип;
* дата рождения в формате даты;
* номер телефона для связи в строковом формате;
* хронические заболевания в строковом формате;
* серия и номер паспорта в строковом формате;
* снилс в строковом формате (по желанию пациента);
* фото пациента (по желанию пациента);
* номер полиса в строковом формате;
* место регистрации – строковый тип;
* история склада – строковый тип;
* медицинская карта – ссылочный тип.

В этой подсистеме должен быть реализован:

* Поиск клиентов;
* Просмотр данных о клиенте;
* Редактирование данных клиента;
* Добавление новых клиентов.

В данной системе будет показан список всех пациентов. При нажатии на определенного пациента откроется окно с выбором действий. Можно будет отредактировать информацию, записать на прием или же открыть медицинскую карту. В окне изменения слева будет показано фото, если клиент согласился сохранить его. Справа прописана информация о пациенте. Так же снизу будет кнопка «подробнее», при нажатии на которую откроется медицинская карта пациента, в которой прописаны все посещения, диагнозы, комментарии врача к посещению, курс лечения.

Подсистема пациентов должна будет связана с подсистемами составления медицинских книг и записей пациентов на прием.

Подсистема расписания врачей будет составлять таблицу расписания, основываясь на базу данных врачей и базу данных записей. Из первой базы будет браться информация о времени работы врачей в определенные дни. Из второй уже занятое время. В итоге будет наглядно видно, когда и к какому врачу можно записать клиента. Окно расписания будет иметься у врача и у администратора. Администратор сможет редактировать таблицу расписания любого врача, выбирая время работы, а врач же сможет только просматривать свое расписание.

В окне «врачи» будет список всех врачей. У каждого врача будет 2 кнопки – «Редактирование» и «Расписание». При нажатии на «Расписание» будет открываться окно расписания определенного врача.

В таблице будут следующие колонки: день недели, работает ли врач в этот день, время начала работы, время конца работы, время начала обеда.

Будет разработано окно с историей выдачи препаратов со складов. Таблица истории будет автоматически заполнятся после завершения. Данная таблица будет содержать следующие поля:

* ID доктора;
* ID клиента;
* время выдачи;
* препарат;
* количество препарата.

Данное окно необходимо для того, чтобы отслеживать кто, когда, кому и что выдавал.

Подсистема логирования склада. В данной подсистеме будут показываться все траты расходных материалов со склада. После завершения приема врач будет указывать сколько и чего было потрачено за время приема, затем эти данные будут добавляться в таблицу StorageLog.

# Проектирование логической структуры

Исходной точкой построения логической схемы данных является представление предметной области в виде одного или нескольких отношений, и на каждом шаге проектирования производится некоторый набор схем отношений (Рисунок 1).



Рисунок 1- структура базы данных

В подсистеме Пациентов главная таблица – «Patients». Она связана с таблицами «Genders» связью многие-к-одному, «MedicalCards» связью многие-ко-многим и «StorageLogs» связью один-ко-многим.

Таблица «Patients» - таблица со всеми добавленными пациентами. В ней хранится вся необходимая информация о пацинтах, а именно:

* PatientId – уникальный идентификатор пациента;
* Name – имя пациента;
* Surname – фамилия пациента;
* Patronymic – отчество пациента;
* GenderCodeNavigationGenderId – идентификатор для связи с таблицей гендеров;
* PhoneNumber – номер телефона;
* ChronicDiseases – хронические заболевания;
* Passport – серия и номер паспорта;
* SNILS – номер снилса;
* PhotoPath – путь до фото;
* PolicyNumber – номер полиса;
* PlaceOfRegistration – место регистрации;
* BirthDay – дата рождения.

Основываясь на эту таблицу в окне просмотра всех пациентов, заполняется список пациентов.

Таблица «Genders» содержит всего 2 поля, а именно GenderId (идентификатор) и сам пол (в нашем случае мужской и женский). Связана она с помощью поля GenderCodeNavigationGenderId в таблице «Patients».

Таблица «Receptions» - таблица записей пациентов на приемы. В ней хранятся все прошедшие записи, а также уже запланированные.

Таблица «StorageLogs» - главная в подсистеме просмотра истории склада. В ней хранится информация о докторе, пациенте и о том, что доктор выдал пациенту во время приема.

Данная таблица содержит следующие поля:

* StorageLogId – уникальный идентификатор записи;
* DoctorId – идентификатор доктора, служит для связи с таблицей Doctors;
* PatientId – идентификатор пациента, служит для связи с таблицей «Patients»;
* LogDate – дата проведения приема;
* DayOfWeek – день недели;
* StorageId – идентификатор выданного товара;
* Count – количество выданного товара.

Данная таблица имеет связи с таблицами «Storages» связью многие-к-одному полем StorageId, «Doctors» связью многие-к-одному полем DoctorId, «Patients» связью многие-к-одному полем PatientId.

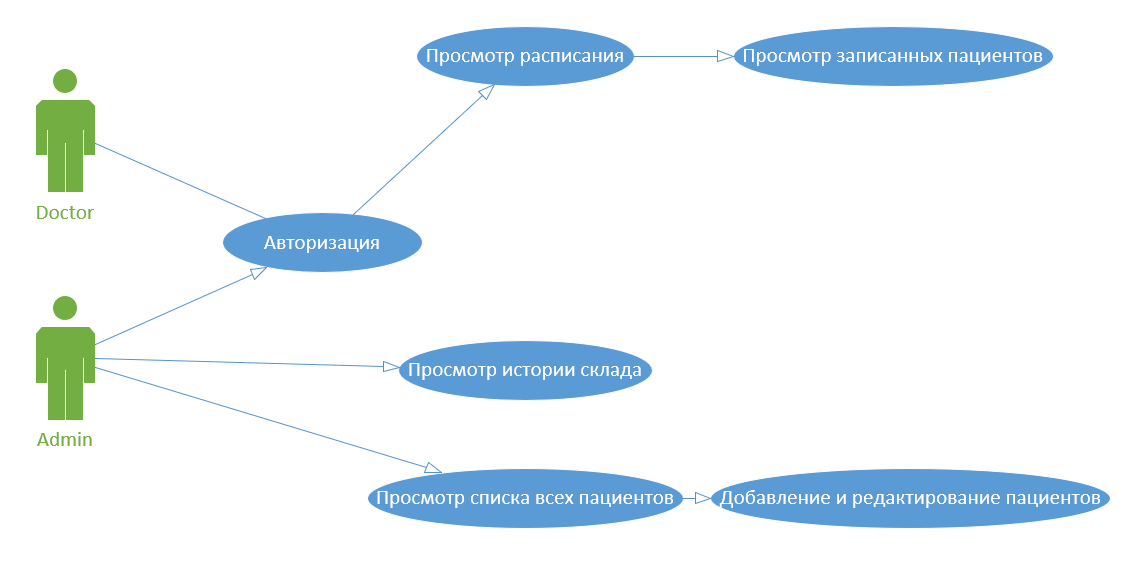


Рисунок 2 - диаграмма прецедентов

Данной программой будут пользоваться 2 актора – Doctor и Admin.

Doctor – доктор, специалист, работающий в клинике. Он принимает пациентов, осматривает их, проводит лечение. При запуске доктор авторизовывается под своим логином и паролем, которые проверяются на верность введённых данных, основываясь на базу данных. Далее доктор может перейти на окно расписания. На данном окне он увидит в какие дни он работает, а также временные промежутки работы и обеда.

Admin – администратор, специалист, работающий в клинике. Работа администратора заключается в том, чтобы добавлять новых пациентов, записывать их на приемы, следить за наличием расходников на складе. Изначально при запуске программы администратор авторизуется с помощью админ-пароля, который записан в базе. После авторизации администратор может перейти на окно просмотра истории склада. Здесь он сможет найти информацию о том, какой доктор выдал лечебные препараты (количество и дату выдачи) какому-либо пациенту.

Так же администратору доступно окно просмотра всех пациентов. На данном окне администратор может увидеть список всех пациентов, который хоть раз посещали клинику. С верху окна находится поле поиска пациентов. Начав вписывать в него, список будет фильтроваться и будут оставаться только те пациенты, фамилии которых содержат введенные буквы. При нажатии на какого-либо пациента откроется окно с подробной информацией о нем, а также ее можно будет отредактировать, в случае необходимости. Когда придется записать на прием пациента, которого нет в базе, администратор нажмет кнопку «Добавить пациента», находящуюся на окне просмотра пациентов. После нажатия на нее откроется форма с данными, необходимыми для добавления пациента. После заполнения всех необходимых полей, администратор нажмет кнопку «Добавить», чтобы клиент добавился в базу данных, появился в списке всех клиентов и его можно было записать на прием

Сверху программы будут находится кнопки для перемещения между подсистемами.

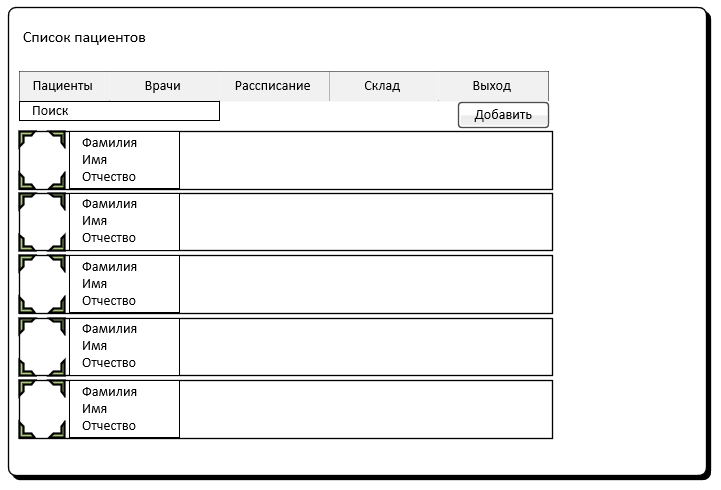


Рисунок 3- Проволочная диаграмма подсистемы пациентов

Окно списка пациентов содержит список клиентов, в который выводится основная информация о человеке. Слева показана фотография (если пациент захотел оставить ее), правее фотографии написаны имя, фамилия и отчество. Сверху страницы находится поле для поиска пациентов и кнопка, для вызова окна добавления новых пациентов.

При нажатии на кнопку добавления или же нажатию по пациенту откроется окно добавления нового, либо редактирования уже имеющегося пациента соответственно.

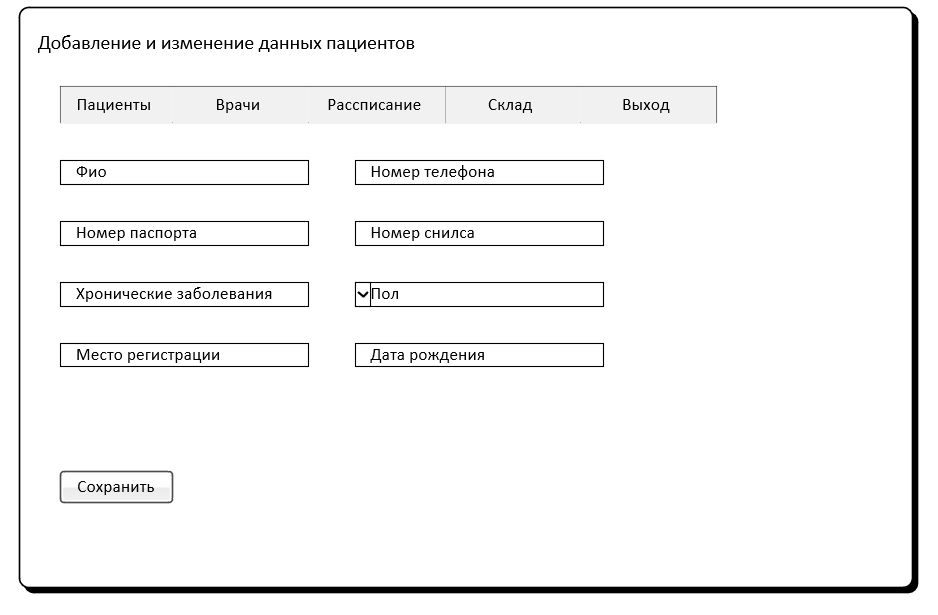


Рисунок 4 - проволочная диаграмма подсистемы добавления и изменения данных пациентов

На окне добавления и изменения данных находятся поля, необходимые для заполнения. Снизу располагается кнопка сохранения. Если заполнены не все поля или какие-то заполнены не верно, то при нажатии на кнопку «Сохранить» появится соответствующая ошибка.

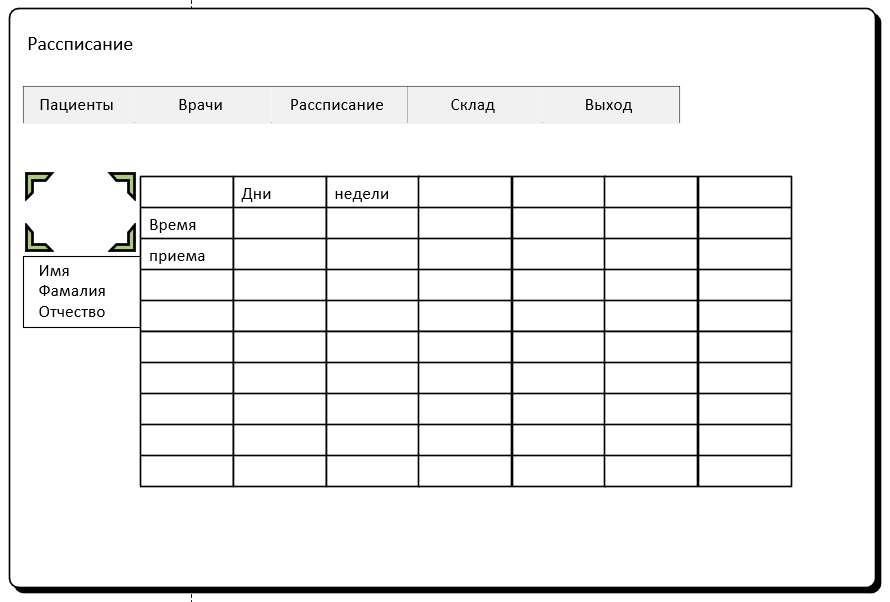


Рисунок 5 - проволочная диаграмма подсистемы расписания

На окне с расписанием будет показано в какие дни и в какое время работает выбранный врач. Слева от таблицы будет написана информация о враче, к которому относится данная таблица.

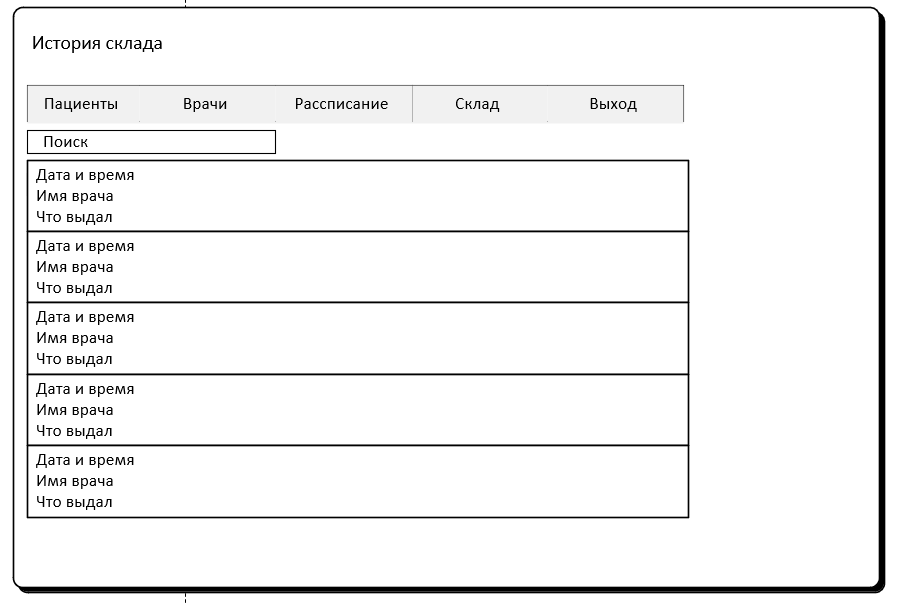


Рисунок 6 - проволочная диаграмма подсистемы истории склада

Окно истории склада содержит информацию о дате и времени выдачи медикаментов, враче и что именно он выдал. Так же сверху окна находится поле поиска.

# 3 Техническое проектирование

## 3.1 Выбор состава технических программных средств

Для написания подсистем для CRM системы я выбрал язык программирования C#. C# - [объектно-ориентированный](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) [язык программирования](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) общего назначения. Разработан в [1998](https://ru.wikipedia.org/wiki/1998)—[2001 годах](https://ru.wikipedia.org/wiki/2001_%D0%B3%D0%BE%D0%B4) группой инженеров компании [Microsoft](https://ru.wikipedia.org/wiki/Microsoft) под руководством [Андерса Хейлсберга](https://ru.wikipedia.org/wiki/%D0%A5%D0%B5%D0%B9%D0%BB%D1%81%D0%B1%D0%B5%D1%80%D0%B3,_%D0%90%D0%BD%D0%B4%D0%B5%D1%80%D1%81) и Скотта Вильтаумота как язык разработки приложений для платформы [Microsoft .NET Framework](https://ru.wikipedia.org/wiki/.NET_Framework) и [.NET Core](https://ru.wikipedia.org/wiki/.NET_Core).

C# относится к семье языков с [C-подобным синтаксисом](https://ru.wikipedia.org/wiki/C-%D0%BF%D0%BE%D0%B4%D0%BE%D0%B1%D0%BD%D1%8B%D0%B9_%D1%81%D0%B8%D0%BD%D1%82%D0%B0%D0%BA%D1%81%D0%B8%D1%81), из них его синтаксис наиболее близок к [C++](https://ru.wikipedia.org/wiki/C%2B%2B) и [Java](https://ru.wikipedia.org/wiki/Java). Язык имеет [статическую типизацию](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F), поддерживает [полиморфизм](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%B8%D0%BC%D0%BE%D1%80%D1%84%D0%B8%D0%B7%D0%BC_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)), [перегрузку операторов](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%B3%D1%80%D1%83%D0%B7%D0%BA%D0%B0_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BE%D0%B2) (в том числе операторов явного и неявного приведения типа), [делегаты](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%BB%D0%B5%D0%B3%D0%B0%D1%82_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), атрибуты, [события](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D0%B1%D1%8B%D1%82%D0%B8%D0%B9%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), [переменные](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), [свойства](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B9%D1%81%D1%82%D0%B2%D0%BE_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), [обобщённые](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D0%BE%D0%B1%D1%89%D1%91%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) типы и методы, [итераторы](https://ru.wikipedia.org/wiki/%D0%98%D1%82%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80), [анонимные функции](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%BE%D0%BD%D0%B8%D0%BC%D0%BD%D0%B0%D1%8F_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F) с поддержкой [замыканий](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%BC%D1%8B%D0%BA%D0%B0%D0%BD%D0%B8%D0%B5_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), [LINQ](https://ru.wikipedia.org/wiki/Language_Integrated_Query), [исключения](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D0%B8%D1%81%D0%BA%D0%BB%D1%8E%D1%87%D0%B5%D0%BD%D0%B8%D0%B9), [комментарии](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BC%D0%B5%D0%BD%D1%82%D0%B0%D1%80%D0%B8%D0%B8_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) в формате [XML](https://ru.wikipedia.org/wiki/XML).

C# довольно прост в изучении, но это не мешает быть ему хорошим языком программирования для написания всевозможных приложений для персональных компьютеров. C# обладает всем необходимым набором функций, необходимых для написания CRM системы.

Технология WPF (Windows Presentation Foundation) является часть экосистемы платформы .NET и представляет собой подсистему для построения графических интерфейсов. Одной из важных особенностей является использование языка декларативной разметки интерфейса XAML, основанного на XML: вы можете создавать насыщенный графический интерфейс, используя или декларативное объявление интерфейса, или код на управляемых языках C#, VB.NET и F#, либо совмещать и то, и другое.

Первая версия - WPF 3.0 вышла вместе с .NET Framework 3.0 и операционной системой Windows Vista в 2006 году. И с тех пор платформа WPF является частью экосистемы .NET и развивается вместе с фреймворком .NET. Например, на сегодняшний день последней версией фреймворка .NET является .NET 7, и WPF полностью поддерживается этой версией фреймворка.

В WPF очень много различных элементов для построения визуальной части, а также есть множество различных пакетов для создания более красивых, удобных элементов.

Для работы с базой данных я использую язык структурированных запросов SQL. SQL – язык запросов, без которого работа с базами данных невозможна. С его помощью можно полностью управлять базой, а именно добавлять, редактировать, удалять записи. Работать с SQL можно с помощью многих языков программирования, в том числе и с помощью C#.

Microsoft SQL Server — [система управления реляционными базами данных (РСУБД)](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%A1%D0%A3%D0%91%D0%94), разработанная корпорацией [Microsoft](https://ru.wikipedia.org/wiki/Microsoft). Основной используемый язык запросов — [Transact-SQL](https://ru.wikipedia.org/wiki/Transact-SQL" \o "Transact-SQL), создан совместно Microsoft и [Sybase](https://ru.wikipedia.org/wiki/Sybase" \o "Sybase). Transact-SQL является реализацией стандарта [ANSI](https://ru.wikipedia.org/wiki/%D0%90%D0%BC%D0%B5%D1%80%D0%B8%D0%BA%D0%B0%D0%BD%D1%81%D0%BA%D0%B8%D0%B9_%D0%BD%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%B8%D0%BD%D1%81%D1%82%D0%B8%D1%82%D1%83%D1%82_%D1%81%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82%D0%BE%D0%B2)/[ISO](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D0%B6%D0%B4%D1%83%D0%BD%D0%B0%D1%80%D0%BE%D0%B4%D0%BD%D0%B0%D1%8F_%D0%BE%D1%80%D0%B3%D0%B0%D0%BD%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F_%D0%BF%D0%BE_%D1%81%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D0%B8) по структурированному языку запросов ([SQL](https://ru.wikipedia.org/wiki/SQL)) с расширениями. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия; конкурирует с другими СУБД в этом сегменте рынка.

T-SQL – это расширение языка SQL, реализованное в Microsoft SQL Server. T-SQL обладает всеми возможностями языка SQL, однако предназначен он для решения задач программирования, не связанных с реляционными данными. Иными словами, на T-SQL Вы можете полноценно программировать, используя переменные, циклы, условные конструкции и другие возможности. T-SQL позволяет нам реализовывать сложные алгоритмы бизнес-логики в виде хранимых процедур и функций, тем самым создавая так называемые «программы» внутри базы данных.

Кроме этого, T-SQL упрощает написание SQL запросов за счет огромного количества встроенных системных функций, а также позволяет администрировать SQL Server и получать системную информацию с помощью системных представлений, процедур и функций.

Для работы с SQL запросами через C# я использую пакеты NuGet:

* Microsoft.EntityFrameworkCore
* Microsoft.EntityFrameworkCore.SqlServer
* Microsoft.EntityFrameworkCore.Tools

С помощью данных пакетов строятся SQL запросы в языке C#.

## 3.2 Физическая структура программы

Для создания более красивого и практичного дизайна был использован пакет NuGet – MaterialDesign. MaterialDesign – один из самых простых и удобных пакетов для создания визуальной части программы.

Каталог Models содержит все классы базы данных CRM\_MED\_DataBase.

Каталог Pages содержит страницы страницы программы. Admin\_PatientPage – страница вывода пациентов при входе через админ-аккаунт. На этой странице выведены все пациенты, при нажатии на которых открывается диалоговое окно с выбором действий с пациентом, а именно изменение данных пациента, запись на прием и просмотр медицинской книжки. Так же на этой странице можно добавлять новых пациентов.

Doctor\_PatientPage – страница выывода пациентов при входе через аккаунт врача. На данной странице будет выводится список всех пациентов, при нажатии на которых будет открываться медицинская книжка.

Doctor\_TimeTablePage – страница просмотра своего расписание при входе через аккаунт врача. Здесь будет выводится таблица, со всей информацией о времени работы врача. Данную таблицу редактировать нельзя.

AdminStorageLogPage – страница, доступная только администратору. На ней выводится вся информация о трате материалов со склада. Редактирование таблицы вывода отключено.

Каталог Views – содержит все окна программы.

AddPatientPage – окно, доступное только администратору, для добавления и редактирования информации о пациентах.

DoctorTimeTableWindow – окно для составления расписания врача. Администратор может редактировать данное окно, меня расписание врача.

PatientDialogWindow – окно, появляющееся при нажатии по пациенту, при входе через админ-аккаунт. В нем находится 3 кнопки с выбором действий с данным пациентом.

PatientMedicalCardWindow – окно с выводом медицинской карты пациента. Сверху показана информация о пациенте, а снизу в таблице вся история посещений этого пациента.

StorageLogReceptionWindow – окно, доступное врачам, появляющееся во время приема. В данном окне врач выбирает что он потратил во время приема и в каком количестве.

Каталог Photos хранит все фотографии врачей и пациентов, а так же Anon.jpg – фото, выводящееся, если не добавлено другое фото.

# 4 Тестирование

# Заключение

В ходе выполнения курсовой работы мною были разработаны подсистемы CRM системы для медицинской клиники. Я сохранил весь основной функционал CRM системы, которая используется сейчас, улучшил дизайн, упростил ориентацию внутри программы.

В подсистеме пациентов реализован вывод всех пациентов из базы данных. Их данные администратор может редактировать, а также переходить на окно записи пациента на прием к врачу и просматривать медицинские книжки. Врачи же просто видят список всех пациентов и могут переходить на их медицинские книжки.

В подсистеме расписания администратор может составлять расписание работы врачей, а именно выбирать работает врач в тот или иной день или нет, выбирать время начала и конца работы, выбирать время обеда. Врачи же могут только просматривать свое расписание и не могут его редактировать.

В подсистеме логирования склада я реализовал вывод информации, о трате материалов со склада. После завершения приема врач указывает что и сколько было потрачено, и эта информация сохраняется в базе данных.

# Список использованной литературы