Министерство цифрового развития государственного управления, информационных технологий и связи Республики Татарстан

государственное автономное профессиональное образовательное учреждение

«Международный центр компетенций –

Казанский техникум информационных технологий и связи»

Курсовая работа выполнена

и защищена с оценкой \_\_\_\_\_\_\_\_\_\_

Руководитель КП \_\_\_\_\_\_\_\_\_

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_г

курсовой ПРОЕКТ

по МДК 01.01 Разработка программных модулей

на тему: Разработка информационной системы выдачи талонов на запись к врачу

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.В. Евсеев

подпись, дата

Исполнитель,

студент группы 320 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ З.Р. Давлеткирова

ИСПП подпись, дата

Казань, 2021

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc73146324)

[1 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА 4](#_Toc73146325)

[1.1 Язык UML 4](#_Toc73146326)

[1.2 Диаграмма USE CASE 4](#_Toc73146327)

[1.3 Диаграмма классов 5](#_Toc73146328)

[1.4 Диаграмма ERD 6](#_Toc73146329)

[2 ОБОСНОВАНИЕ ВЫБОРА ЯЗЫКА ПРОГРАММИРОВАНИЯ 8](#_Toc73146330)

[2.1 Обоснование выбора СУБД. 8](#_Toc73146331)

[2.2 Обоснование выбора платформы программирования 10](#_Toc73146332)

[3. РЕАЛИЗАЦИЯ ПРОЕКТА 12](#_Toc73146333)

[3.1 Создание базы данных 12](#_Toc73146334)

[3.2 Создание приложения 15](#_Toc73146335)

[4 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 18](#_Toc73146336)

[4.1 Запуск программы 18](#_Toc73146337)

[4.2 Работа с программой 18](#_Toc73146338)

[5 ТЕСТИРОВАНИЕ 21](#_Toc73146339)

[5.1 Тест-кейсы 21](#_Toc73146340)

[ЗАКЛЮЧЕНИЕ 24](#_Toc73146341)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 25](#_Toc73146342)

[ПРИЛОЖЕНИЕ A 27](#_Toc73146343)

[ПРИЛОЖЕНИЕ B 37](#_Toc73146344)

[ПРИЛОЖЕНИЕ С 46](#_Toc73146345)

# ВВЕДЕНИЕ

Регистратура больницы является одним из важнейших структурных подразделений медицинского учреждения. Актуальность создания информационной системы в поликлинике обусловлена необходимостью использования больших объемов информации.

На сегодняшний день существует множество медицинских учреждений, нуждающихся в информационной системе для облегчения работы сотрудников. Информационные технологии автоматизируют процессы лечебного учреждения, тем самым облегчают работу сотрудникам учреждения и пациентам.

Информационная система разрабатывается для частной поликлиники. Для выдачи талона системой, требуется заполнить данные пациента, а именно: ФИО, дата рождения и номер полиса. После заполнения данных требуется выбрать врача и время посещения.

Данная программа должна быть простой в освоении и использовании, что будет удобно пациентам любых возрастов.

# 1 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА

## 1.1 Язык UML

При решении сложных задач необходимо заблаговременно запланировать и смоделировать программный продукт. Кроме того, вносить изменения в диаграммы – это легче, чем в код. Главным достоинством проектирования является возможность посмотреть на задачу с разных точек зрения, так другим программистам легче понять способ ее реализации, не смотря на код.

UML (Unified Modeling Language) ­является стандартизированным языком моделирования, состоящим из интегрированного набора диаграмм, которые разработаны, чтобы помочь разработчикам систем и программного обеспечения в определении, визуализации, конструировании и документировании артефактов программных систем.

## 1.2 Диаграмма USE CASE

Диаграмма USE CASE или диаграмма прецедентов, описывает функциональные требования системы с точки зрения прецедентов. То есть, это модель предполагаемой функциональности системы и ее среды. Прецеденты позволяют связать то, что нам нужно от системы с тем, как система удовлетворяет эти потребности.

Разработка диаграммы прецедентов преследует цели:

* Определение общих границ и контекста моделируемой предметной области на начальных этапах проектирования системы;
* Формулировка общих требований к функциональному поведению проектируемой системы;
* Разработка концептуальной модели системы для ее последующей детализации в форме логических и физических моделей;
* Подготовка исходной документации для взаимодействия разработчиков системы с ее заказчиками и пользователями.

Действующим лицом в диаграмме называется любая сущность, взаимодействующая с системой извне. В свою очередь вариант использования определяет некоторый набор действий, совершаемый системой при диалоге со средой.

В проекте пациенту предоставляется выбор врача, выбор удобного для него времени и получение электронного талона. Пациент является действующим лицом (Рисунок 1.1).

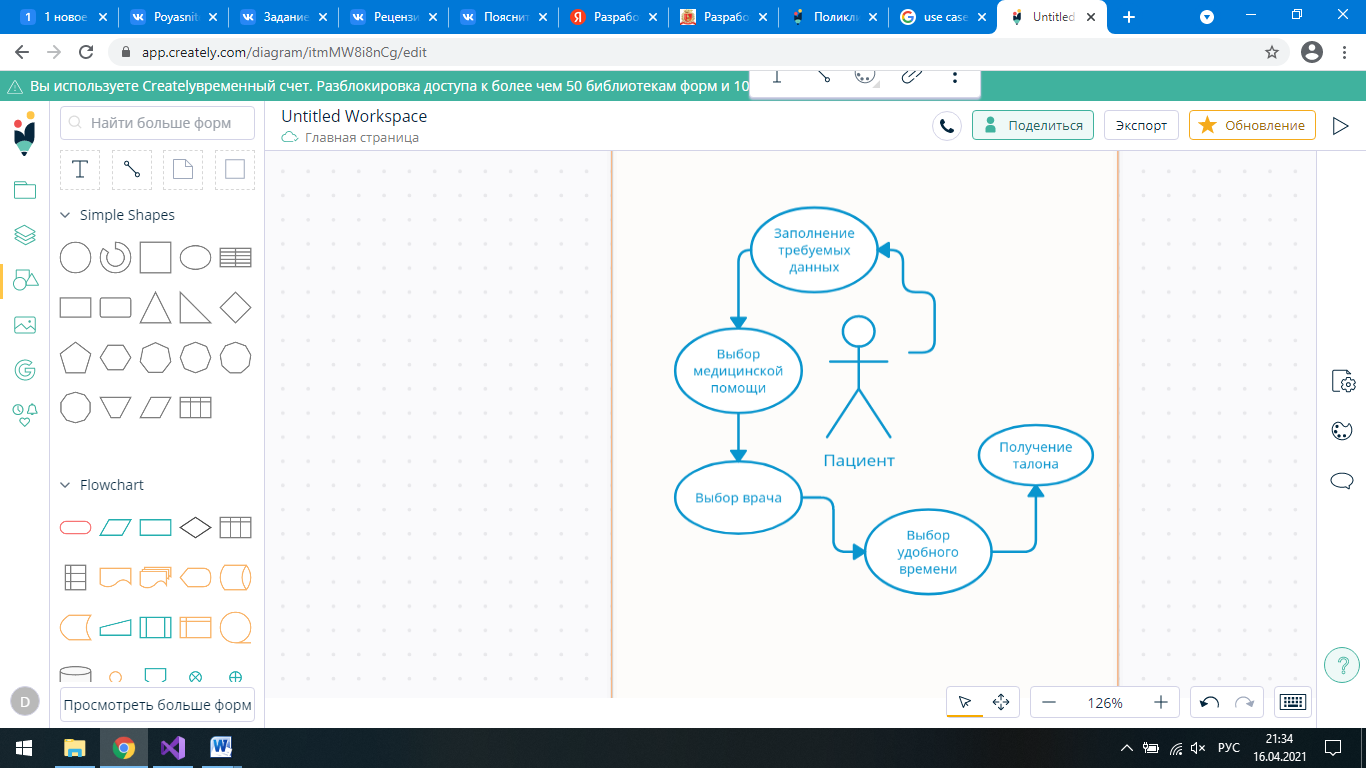


Рисунок 1.1 – Диаграмма Use Case

## 1.3 Диаграмма классов

Диаграммы классов иллюстрируют статичные структуры внутри системы, включая классы, атрибуты, операции и объекты. В их состав могут входить как концептуальные, так и организационные данные в форме классов реализации и логических классов, соответственно. Стоит отметить, что данные две группы могут пересекаться.

Диаграмма дает наиболее полное и развернутое представление о структуре и связях в программном коде. Диаграммы классов применяются при схематизации крупных систем, так как позволяют объединять классы в группы. Понимание принципов построения данной диаграммы позволяет кратко и прозрачно выражать свои мысли и идеи. Самыми значимыми достоинствами этой диаграммы являются:

* Экономия времени при объяснении задачи другим программистам;
* Более точное и наглядное представление структуры основных элементов системы.

В диаграмме указаны классы, отношения, операции и объекты соответствующие базе данных, с уточнением типа данных (Рисунок 1.2).

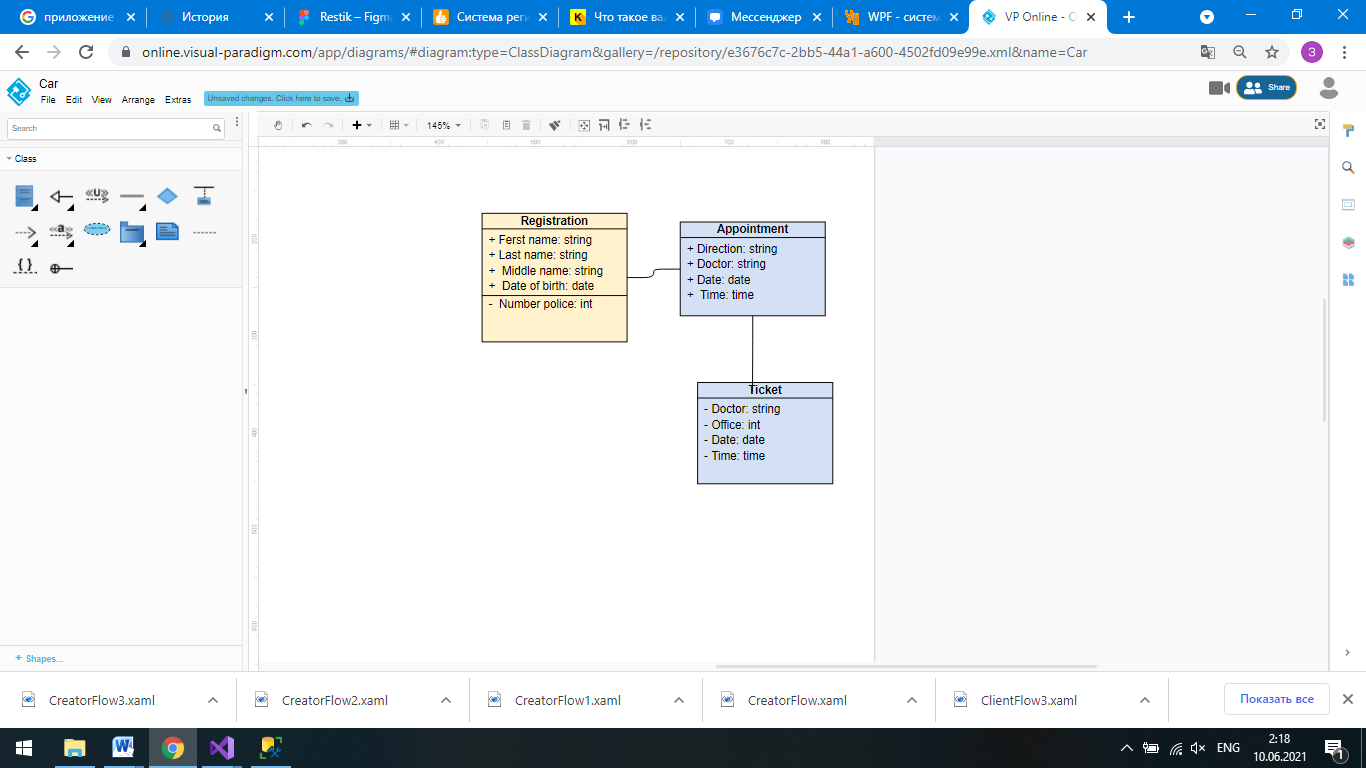


Рисунок 1.2 – Диаграмма классов

## 1.4 Диаграмма ERD

ERD (Entity-Relationship diagram)– диаграмма позволяет рассмотреть систему целиком и выяснить требования, необходимые для ее разработки, касающиеся хранения информации. ER-диаграммы применяются для моделирования и проектирования реляционных баз данных. В сфере разработки программного обеспечения ER-диаграммы также применяются для моделирования конкретных баз данных. Реляционная база данных сопровождается соответствующей реляционной таблицей.

Первым шагом создания диаграммы является выявление сущности, то есть таблицы, под которой понимается набор или класс однотипных физических или абстрактных экземпляров, которые важны для рассматриваемой предметной области. Следующим шагом является определение связей, они показывают, как сущности взаимодействуют между собой. Затем добавляем атрибуты, это свойства, которым может быть охарактеризована сущность или связь. Они иллюстрируют конкретные характеристики сущности, фокусируя внимание на важной информации в контексте модели. Сущностями являются классы: специалист, специализация, пациент, время, талон. Связи между сущностями один ко многим (Рисунок 1.3).

Для создания логичной и правильной ER-диаграммы нужно избегать лишних сущностей и связей, но при этом, все сущности должны быть связаны между собой.

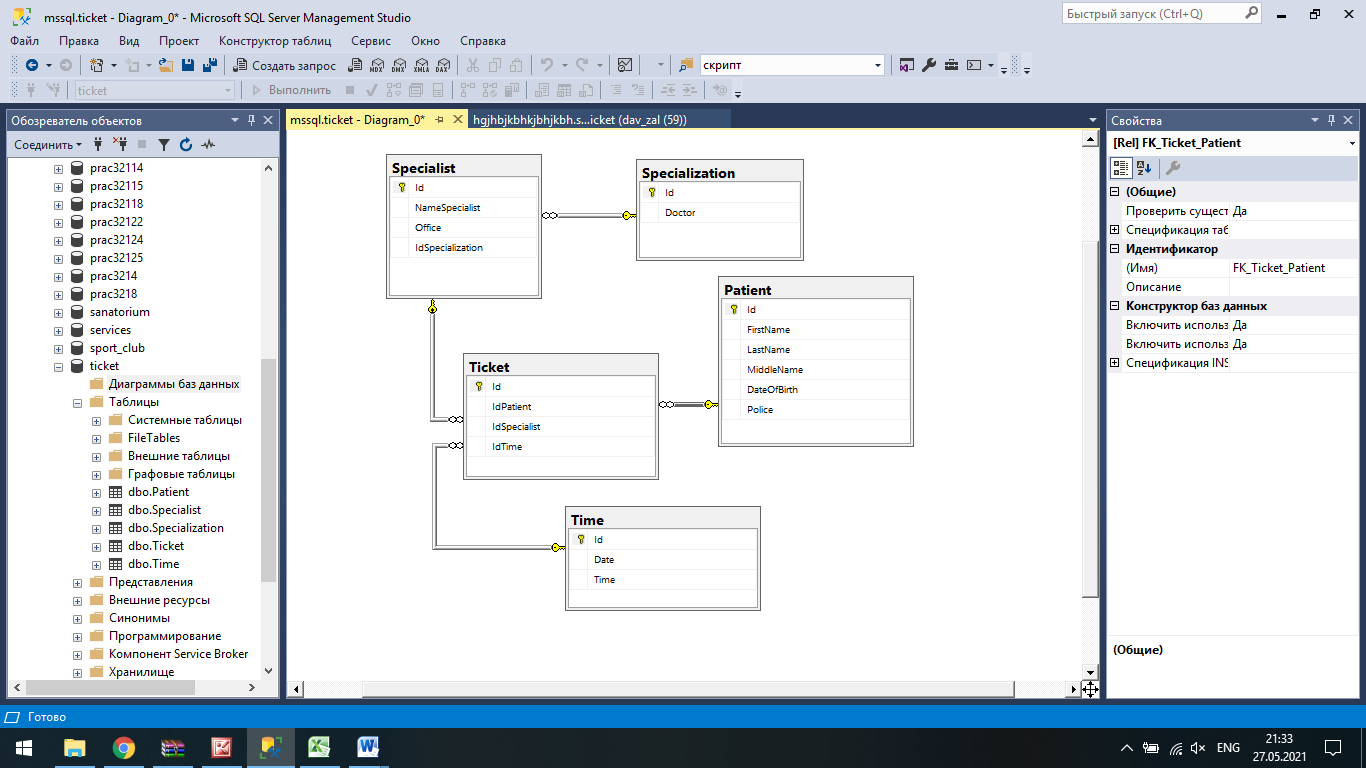


Рисунок 1.3 – Диаграмма ERD

# 2 ОБОСНОВАНИЕ ВЫБОРА ЯЗЫКА ПРОГРАММИРОВАНИЯ

## 2.1 Обоснование выбора СУБД.

Система управления базами данных (СУБД) – это программа, предназначенная для создания, ведения и совместного использования базы данных несколькими пользователями. Основными функциями СУБД являются создание, поиск и изменение необходимых данных. Базы данных представляют собой набор данных, которые связаны друг с другом.

База данных (БД) – это специально созданное хранилище важной информации, неотъемлемым атрибутом которого является удобный доступ к хранящимся данным. Для обеспечения этого доступа используются СУБД, представляющие собой особые приложения на базе определенной программной архитектуры.

Практически все разработчики современных приложений, предусматривающих связь с системами баз данных, ориентируются на реляционные СУБД. В реляционных базах данные хранятся в виде таблиц, состоящих из строк и столбцов. Каждая таблица имеет собственный, заранее определенный набор именованных полей. Столбцы таблиц реляционной базы могут содержать скалярные данные фиксированного типа, например числа, строки или даты. К реляционным СУБД относятся MySQL, SQLite, Microsoft SQL Server.

Можно выделить ряд критериев, по которым следует проводить выбор СУБД:

* особенности архитектуры и функциональные возможности;
* особенности разработки приложений;
* производительность;
* надежность;
* требования к рабочей среде.

SQL Server – система управления реляционными базами данных. Она является надежной базой данных для любых целей,  может продолжать расширяться по мере  наполнения информацией, без заметного уменьшения быстродействия операций с записями в многопользовательском режиме. Обеспечивается максимальная безопасность.  Поскольку безопасность на уровне пользователя, они могут иметь ограниченный доступ к записи данных, тем самым защищая их от модификации или поиска, указав доступ на уровне пользовательских привилегией.  SQL Server упрощает развертывание, передачу и интеграцию больших данных и отличается гибкостью в отношении выбора платформы, языка программирования и средства доставки.

SQLite – это система управления базами данных, отличительной особенностью которой является ее встраиваемость в приложения. Большинство СУБД являются самостоятельными приложениями, взаимодействие с которыми организовано по принципу клиент-сервер. SQLite не имеет каких-либо определенных функций управления пользователями и, следовательно, не подходит для многопользовательского доступа. SQLite подходит для небольших баз данных. По мере роста базы данных требования к памяти также увеличиваются при использовании SQLite.

MySQL – [свободная](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%D0%BE%D0%B5_%D0%9F%D0%9E) [реляционная система управления базами данных](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%A1%D0%A3%D0%91%D0%94). Он обычно используется с PHP для веб-разработки и обладает обширным разнообразием доступных инструментов для создания приложений.

Помимо универсальности и распространенности СУБД MySQL обладает целым комплексом важных преимуществ перед другими системами. В частности следует отметить такие качества как:

* Простота в использовании;
* Обширный функционал. Система MySQL обладает практически всем необходимым инструментарием, который может понадобиться в реализации практически любого проекта;
* Безопасность. Система изначально создана таким образом, что множество встроенных функций безопасности в ней работают по умолчанию.

В данной курсовой работе проект выполняется на MS SQL, так как она проверенная временем и масштабируемая платформа для облачных сред и центров обработки данных. Эта платформа выдержит самые ресурсоемкие рабочие нагрузки. Благодаря своим возможностям восстановления, она хорошо защищена от перебоев в обслуживании.

## 2.2 Обоснование выбора платформы программирования

При изучении материалов для выбора платформы необходимо обращать внимание на сложность ее освоения. Для разработки на WPF и Xamarin достаточно знать язык C# с его стилем написания программ и изучить несколько классов, связанных с конкретными платформами.

WPF (Windows Presentation Foundation) – это платформа пользовательского интерфейса для создания приложений.

В приложении WPF существует два способа создания пользовательского интерфейса для приложения. Одним из них является перетаскивание элементов пользовательского интерфейса из панели инструментов в окно дизайна. Второй способ – создать свой пользовательский интерфейс, написав теги XAML для элементов пользовательского интерфейса. Visual Studio обрабатывает теги XAML, когда для проектирования пользовательского интерфейса используется функция перетаскивания. В XAML очень легко создавать, инициализировать и устанавливать свойства объектов с иерархическими отношениями. Он в основном используется для разработки графических интерфейсов.

WPF поддерживает гибкий поток, размещающий элементы управления на основе их содержимого. В результате получается пользовательский интерфейс, который может быть адаптирован для отображения высокодинамичного содержимого или к разным языкам. Глобализация и локализация приложений WPF упрощена.

Xamarin – это современный кроссплатформенный Framework разработки мобильных приложений. Разработчики используют Xamarin для создания не только небольших, но и для масштабных проектов.

Главным преимуществом Xamarin является возможность переиспользовать общий код для трех платформ, то есть, позволяет создавать одну единственную логику приложения с применением C# и .NET сразу для всех трех платформ - Android, iOS, Windows Mobile. Приложения, разработанные с использованием платформы Xamarin, кажутся больше. Управление такими тяжелыми приложениями требует дополнительных усилий и времени на настройку. Есть много-много разных функций и основных библиотек, с которыми нужно иметь дело. Это может привести к увеличению размера.

При сравнивании Xamarin и WPF было принято решение выбрать WPF за счет простоты в использовании в разработке приложений. Также стоит пометить что затраты на разработку и обслуживание снижаются, так как разметка, определяющая внешний вид, не связана тесно с кодом, обуславливающим поведение.

# 3. РЕАЛИЗАЦИЯ ПРОЕКТА

## 3.1 Создание базы данных

В проекте в качестве хранения и обработки информации используется MS SQL Server 2019.

Первый этап создания базы данных – вход в SQL Server Management Studio (Рисунок 3.1). Необходимо войти в SQL Server с помощью ввода имени сервера, имени пользователя и пароля.

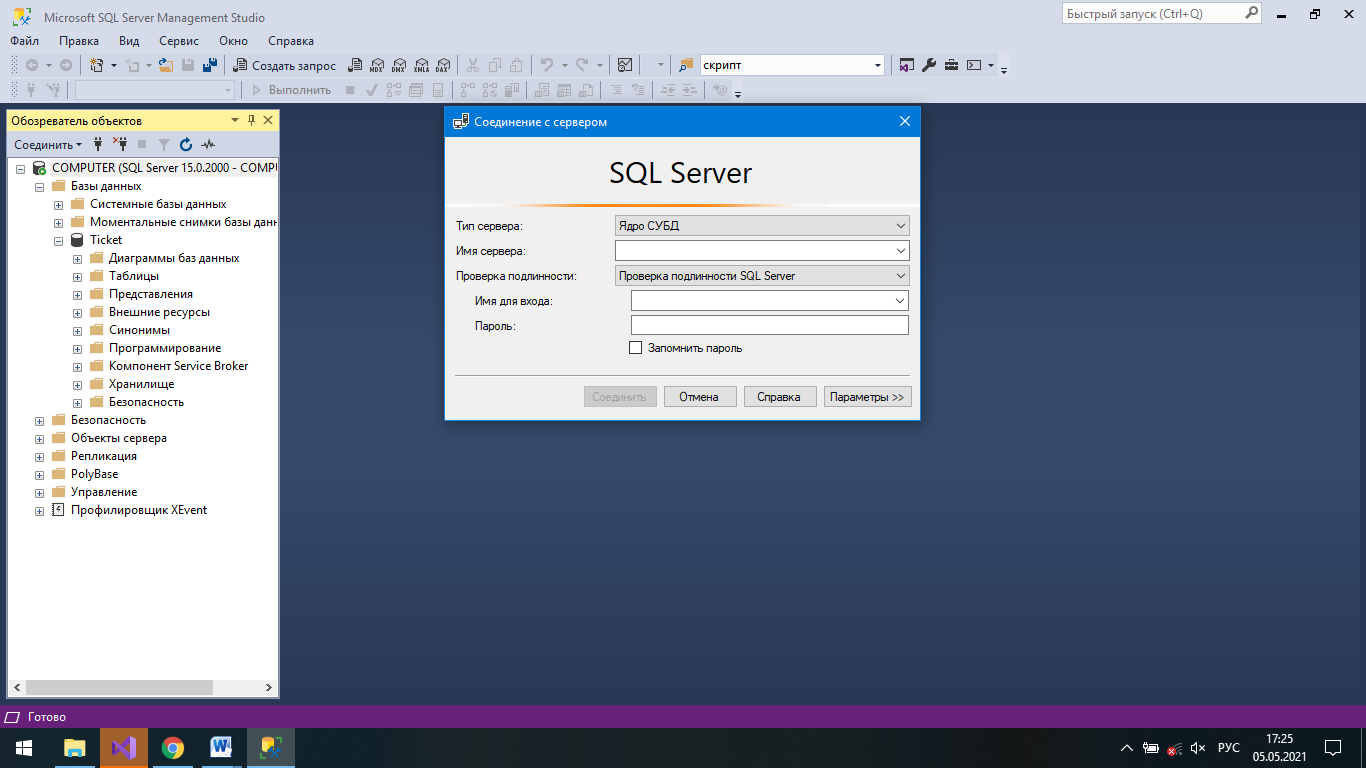


Рисунок 3.1 – Вход в SQL Server

Дальнейшим этапом следует создание самой базы данных, где будут храниться все необходимые данные (Рисунок 3.2).

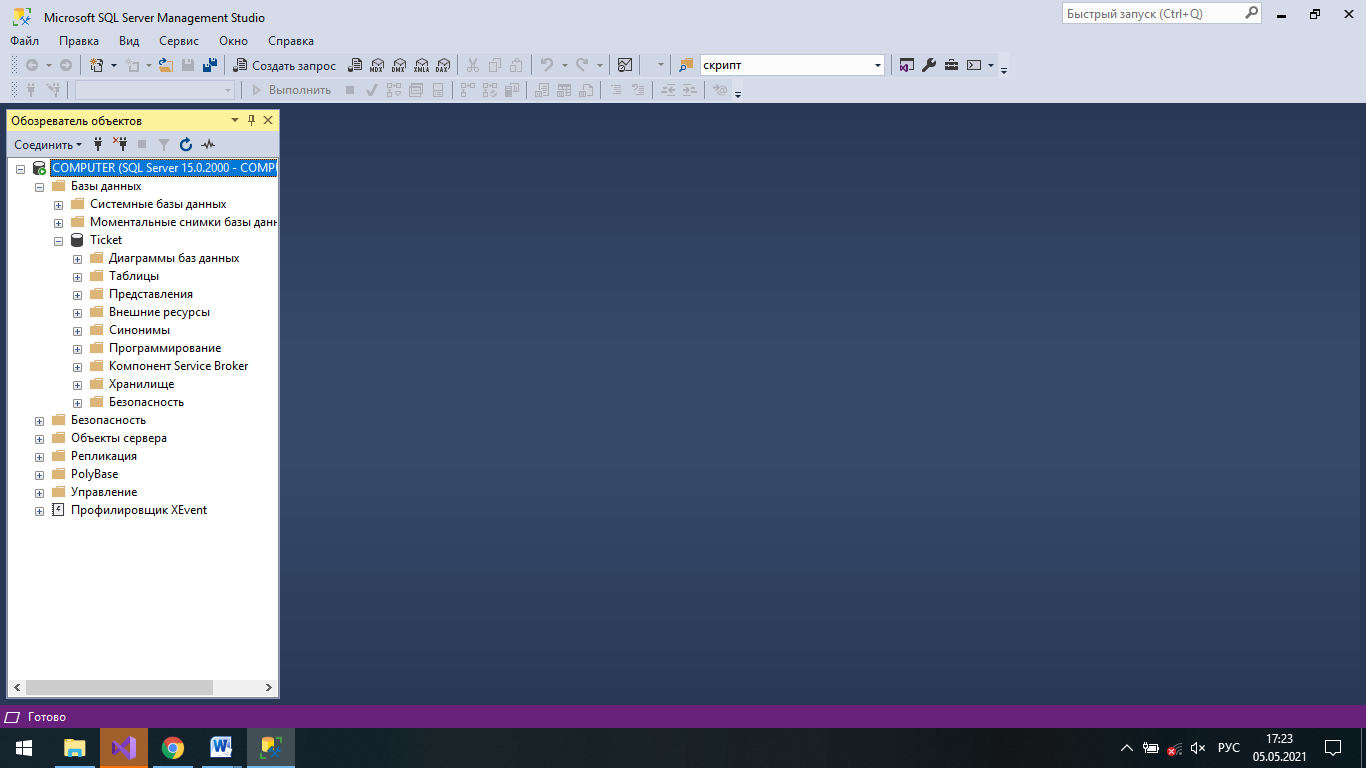


Рисунок 3.2 – Создание базы данных

Следующий этап – создание таблиц, заполнение их необходимыми атрибутами, то есть столбцами, которые в свою очередь должны быть заполнены.

В базе данных содержатся следующие таблицы:

* Patient;
* Specialist;
* Specialization;
* Time;
* Ticket.

Затем следует этап создания связей по внешним ключам.

Внешний ключ представляет один или несколько столбцов из одной таблицы, который одновременно является потенциальным ключом из другой таблицы. Как правило, внешний ключ из зависимой таблицы указывает на первичный ключ из главной таблицы.

Типы связи:

* Один к одному;
* Один ко многим;
* Многие ко многим.

Связь один к одному образуется, когда ключевой столбец присутствует в другой таблице, в которой тоже является ключом либо свойствами столбца задана его уникальность.

В типе связей один ко многим одной записи первой таблицы соответствует несколько записей в другой таблице.

Если нескольким записям из одной таблицы соответствует несколько записей из другой таблицы, то такая связь называется «многие ко многим» и организовывается посредством связывающей таблицы.

Связи выполняют более важную роль, чем просто информация размещения данных по таблицам. Прежде всего, они требуются разработчикам для поддержания целостности баз данных. Правильно настроив связи, можно быть уверенным, что ничего не потеряется.

Вся информация о таблицах сведена в рисунках 3.3-3.7 с более подробным описанием, а также указанием типов данных.

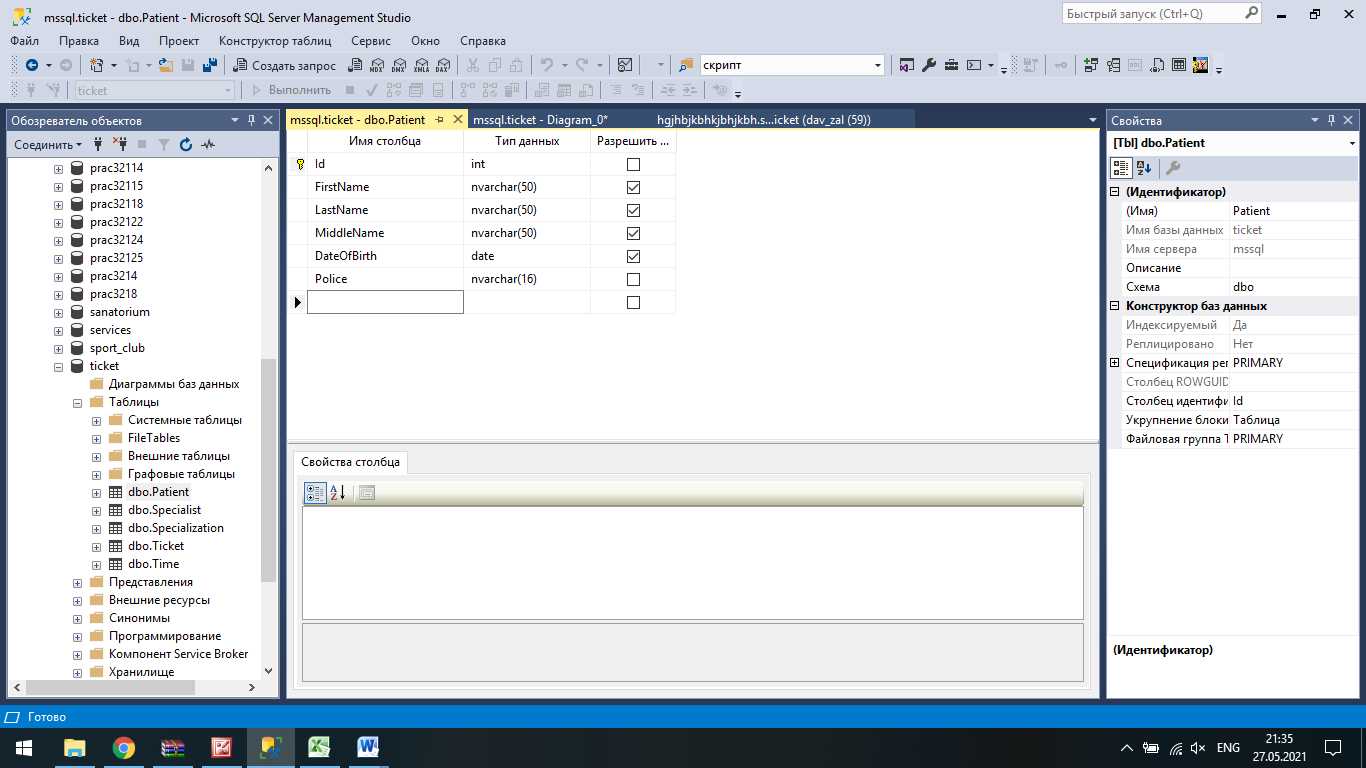


Рисунок 3.3 – Таблица Patient

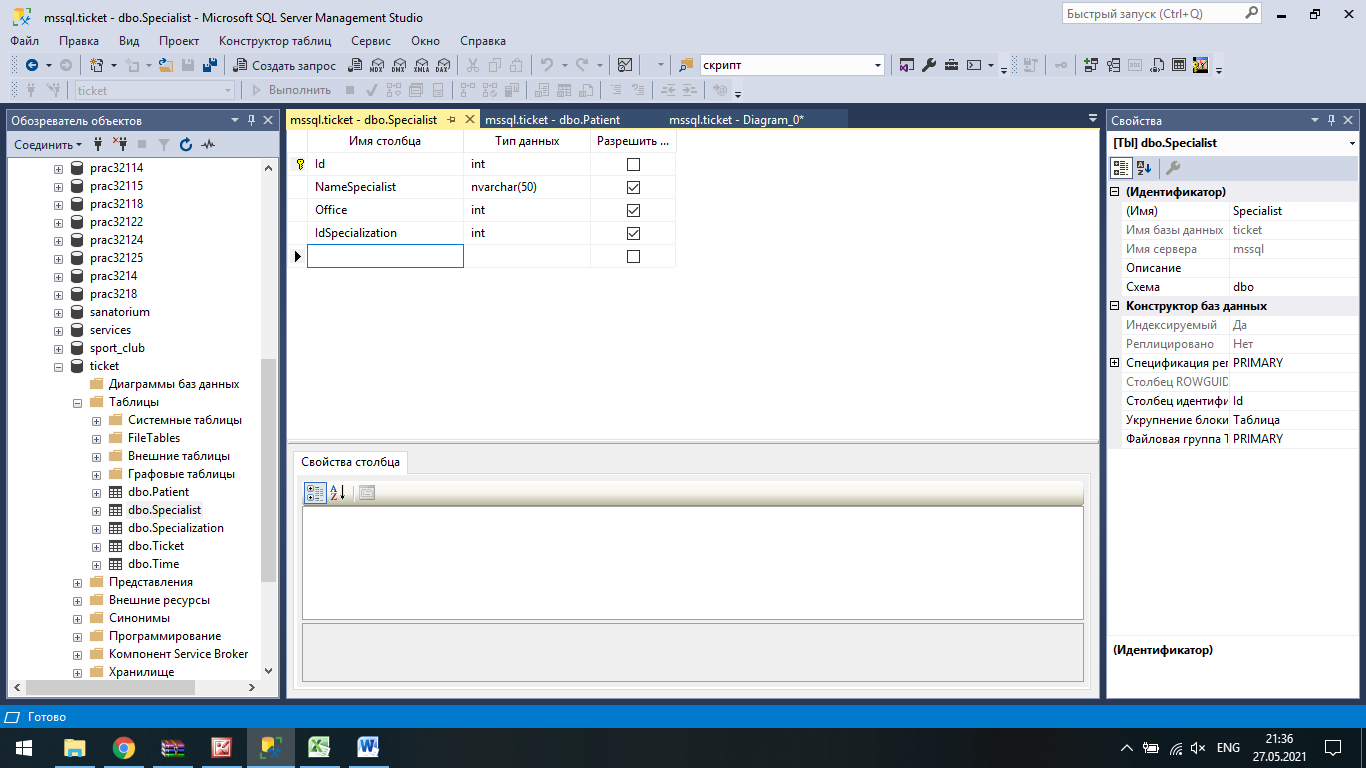


Рисунок 3.4 – Таблица Specialist

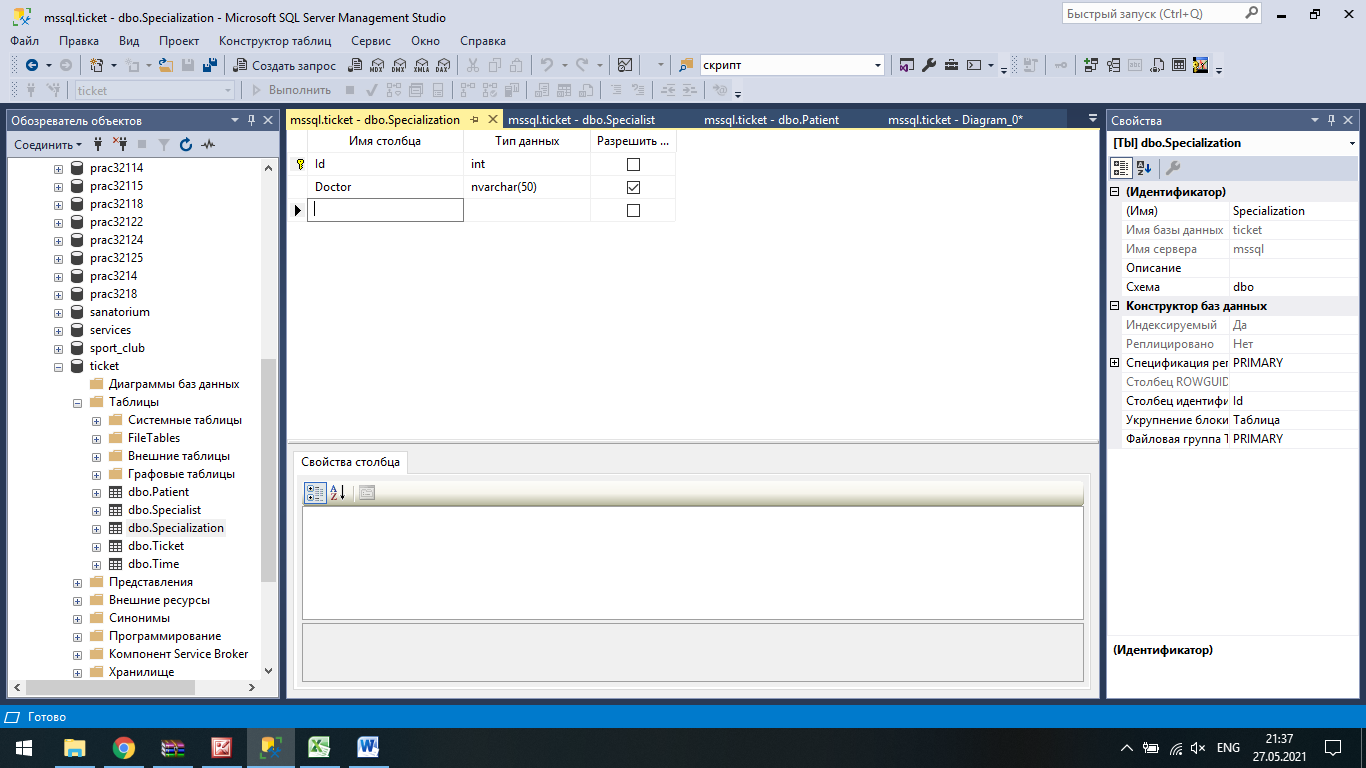


Рисунок 3.5 – Таблица Specialization

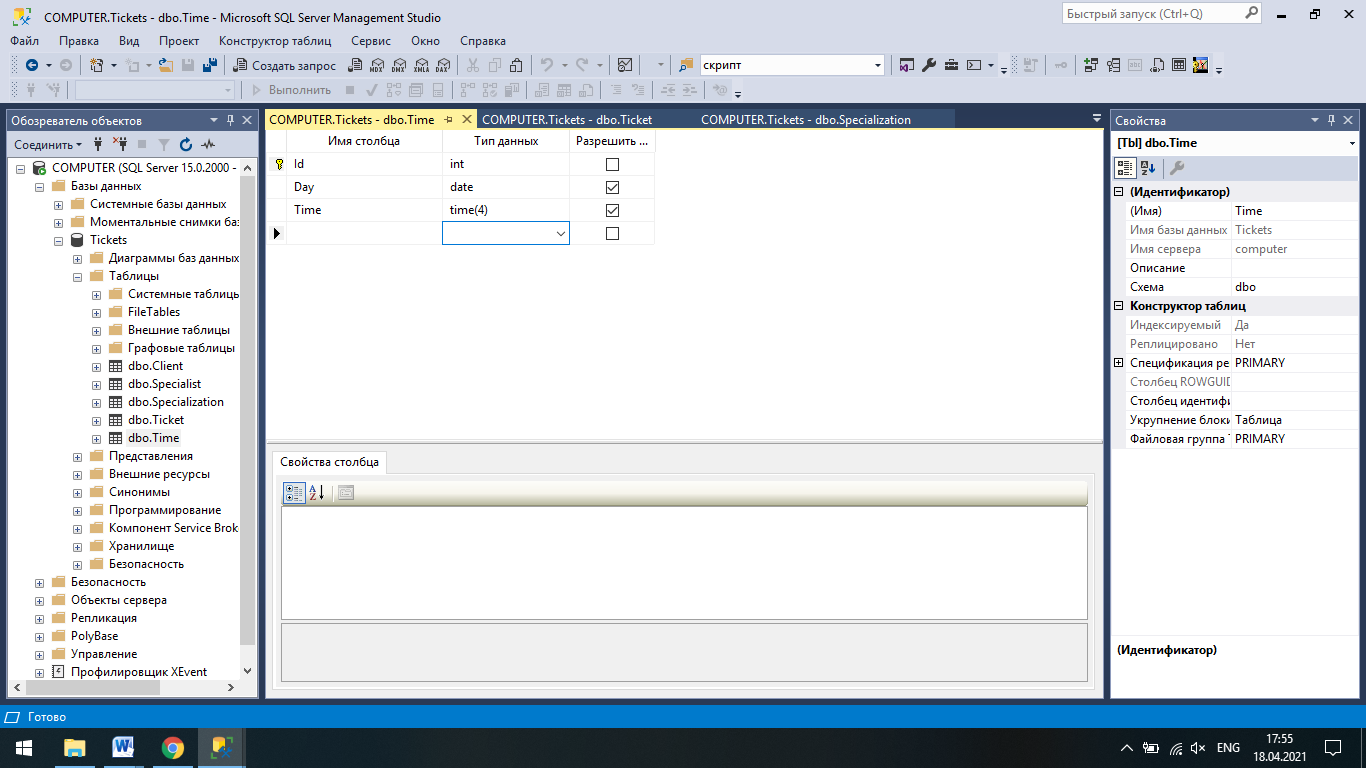


Рисунок 3.6 – Таблица Time

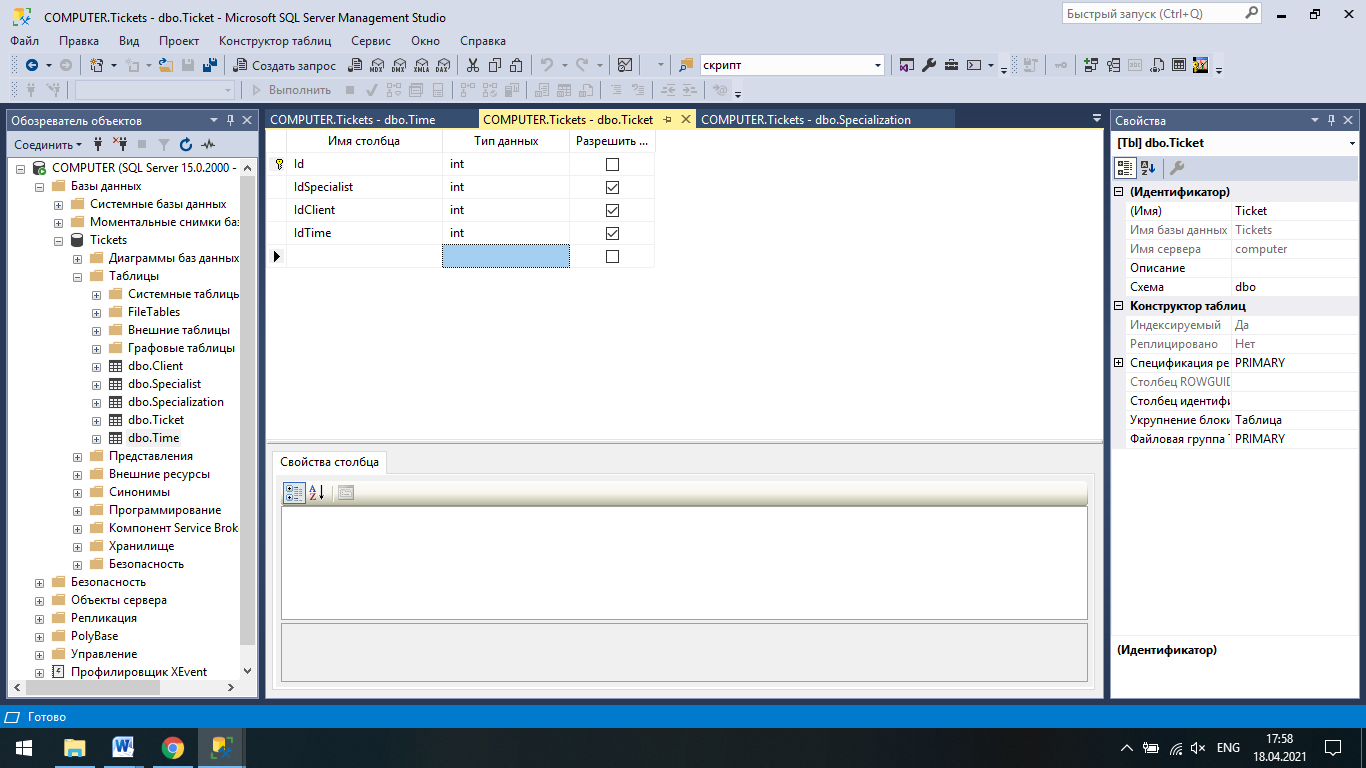


Рисунок 3.7– Таблица Tickets

Последним этапом происходит заполнение таблиц необходимыми данными. Вся информация о данных сведена в рисунках 3.8-3.9.

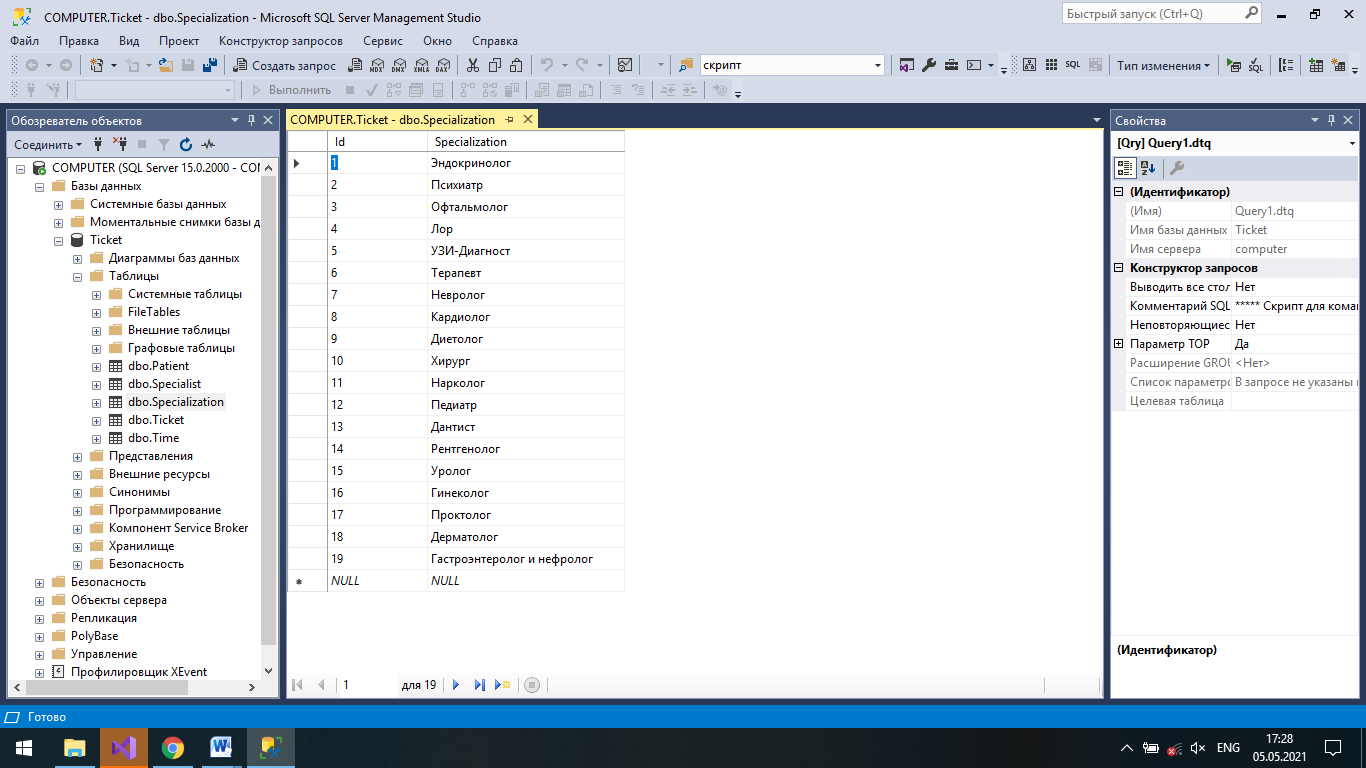


Рисунок 3.8 – Заполнение данными таблицы Specialization

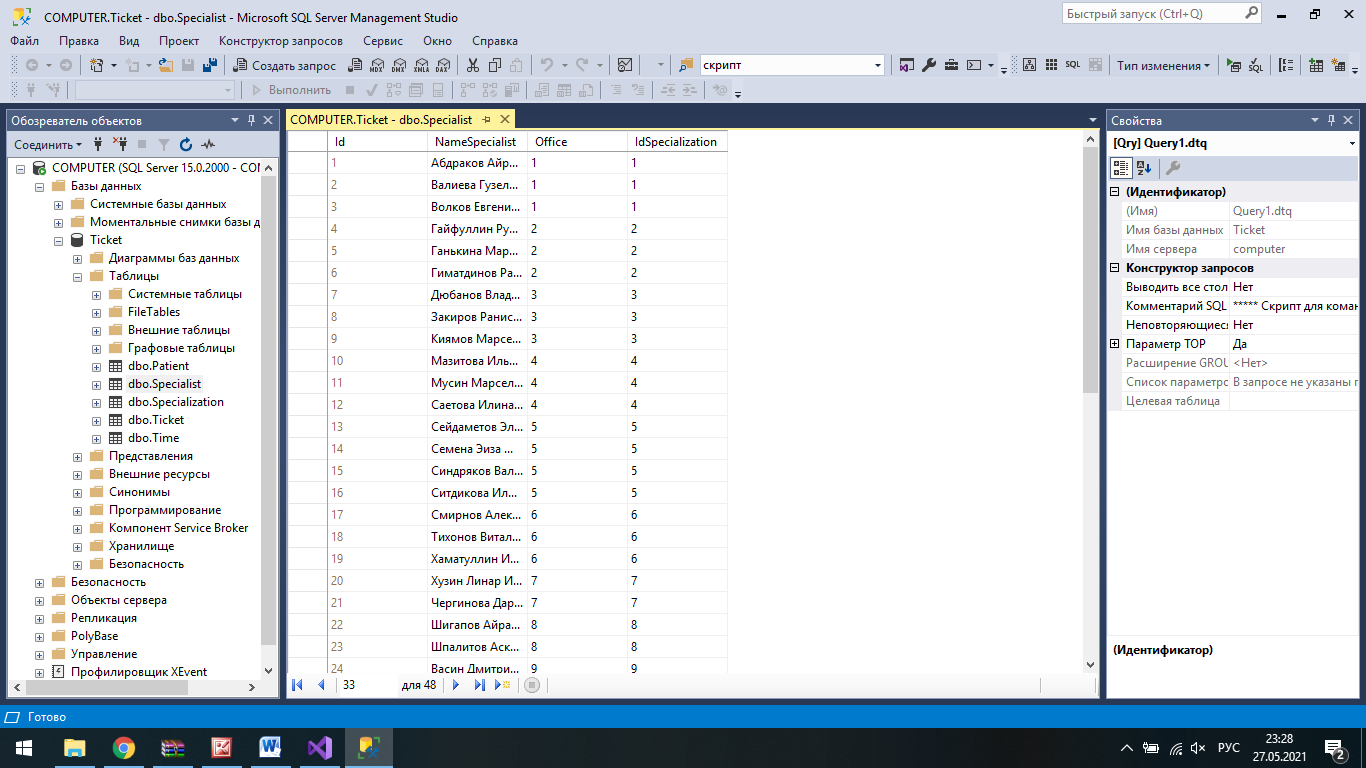


Рисунок 3.9 – Заполнение данными таблицы Specialist

Таблицы Time, Ticket, Patient заполняются автоматически пользователями.

## 3.2 Создание приложения

Для создания приложения необходимо сориентироваться в элементах будущего интерфейса. Для этого нужно исследовать предметную область.

В курсовом проекте приложение разрабатывается в приложении WPF.

Entity Framework – это решение для работы с базами данных, которое используется в программировании на языках семейства .NET. Оно позволяет взаимодействовать с СУБД с помощью сущностей, а не таблиц. Подключить Entity Framework можно к любому проекту. Нужно открыть менеджер пакетов NuGet (Рисунок 3.10)

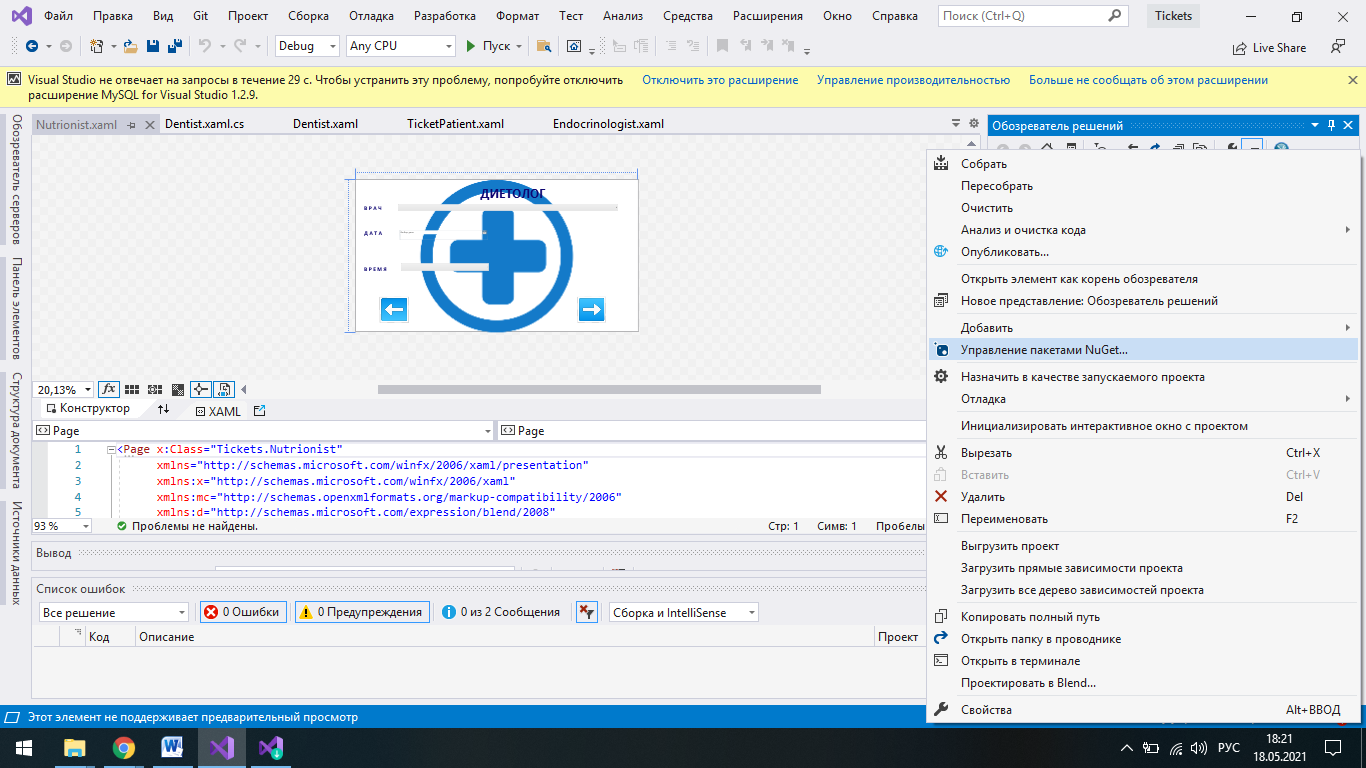


Рисунок 3.10 – Открытие NuGet

Затем необходимо скачать пакет EntityFramework (Рисунок 3.11)

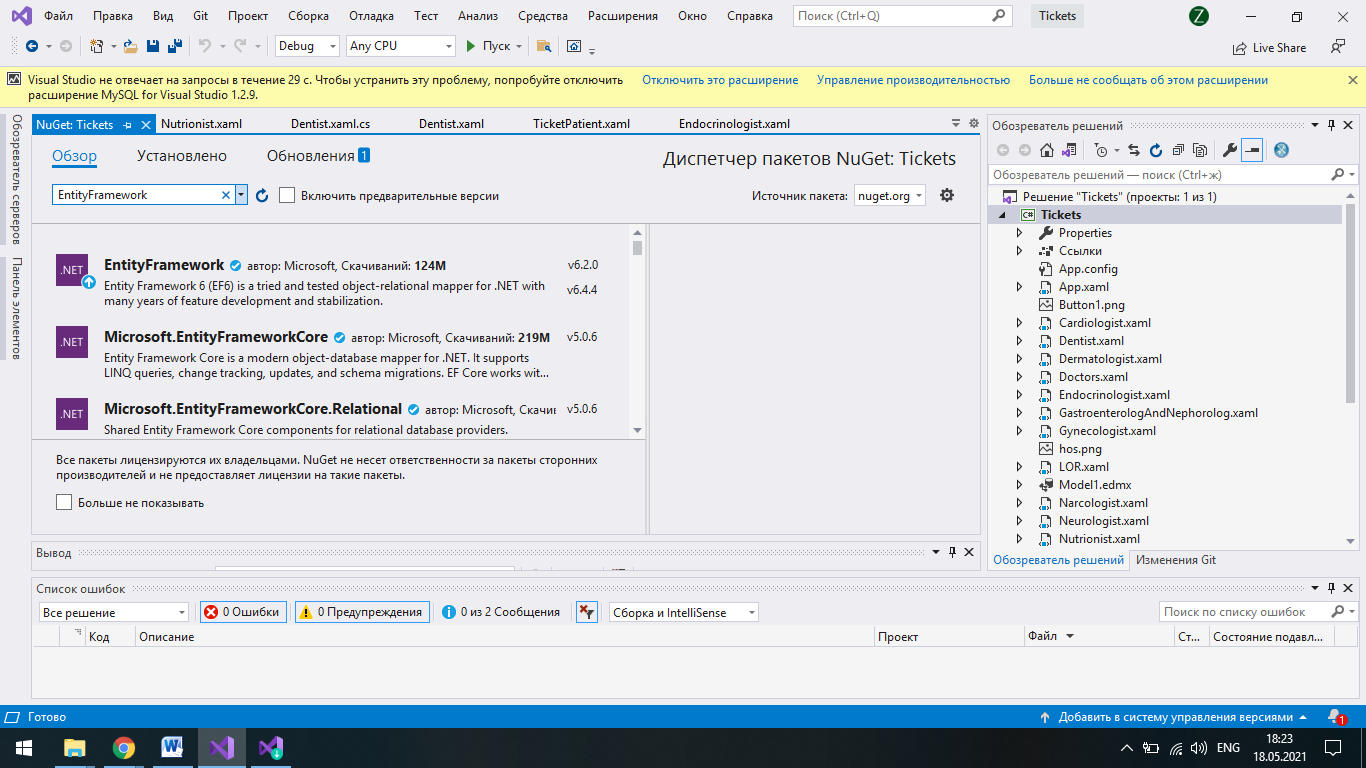


Рисунок 3.11 – Скачивание пакета EntityFramework

Далее необходимо создать модель базы данных через ADO.NET.

ADO.NET – это инструмент, используемый для доступа к базе данных. ADO.NET последнее воплощение поставляется с "Entity Framework", объектно-реляционным картографом (Рисунок 3.12-3.13).

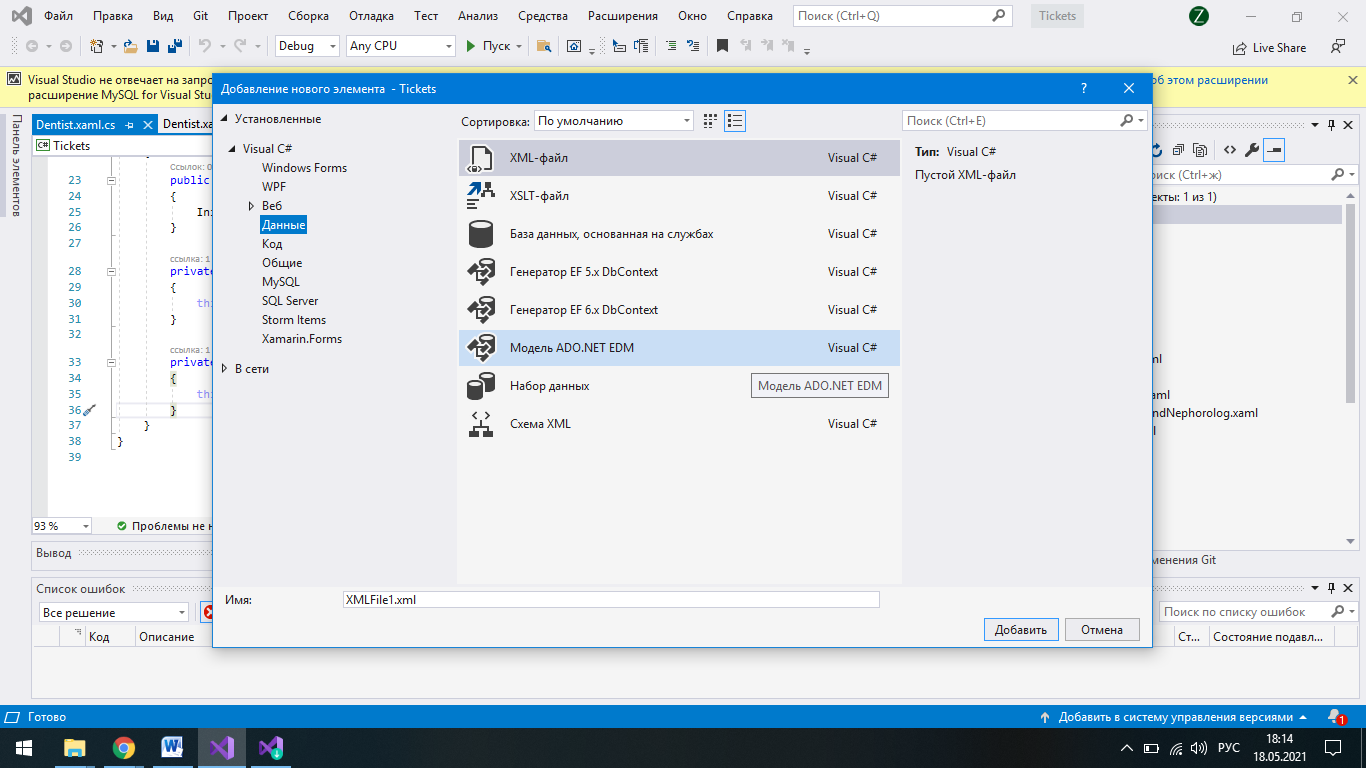


Рисунок 3.12 – Добавление модели ADO.NET

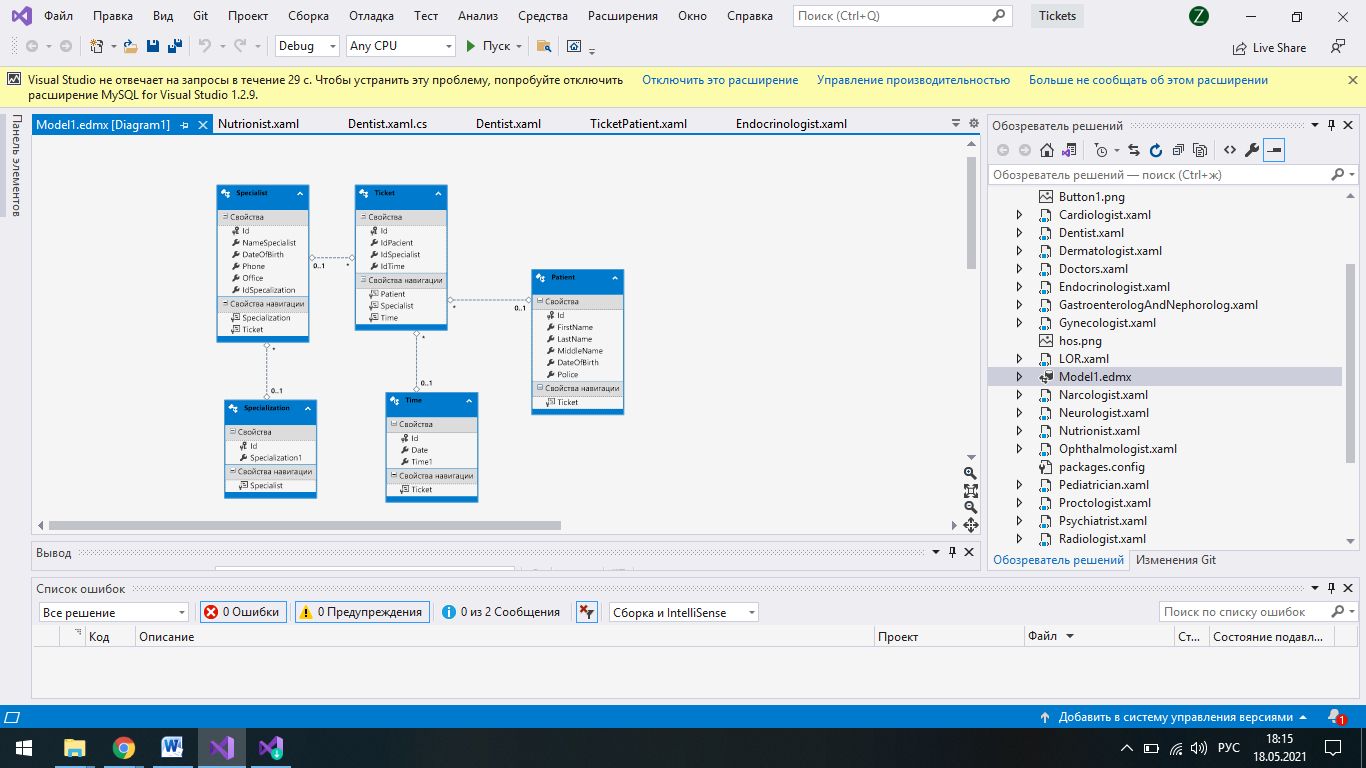


Рисунок 3.13 – Модель ADO.NET

После подключения базы данных реализуется желаемый интерфейс и разработка проекта с поставленными задачами.

# 4 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

## 4.1 Запуск программы

Для использования программы пользователю необходимо иметь стабильный интернет.

## 4.2 Работа с программой

Первым пользователя встречает страница регистрации (Рисунок 4.1). Здесь необходимо ввести данные пользователя. При нажатии кнопки данные пользователя передаются в базу данных.

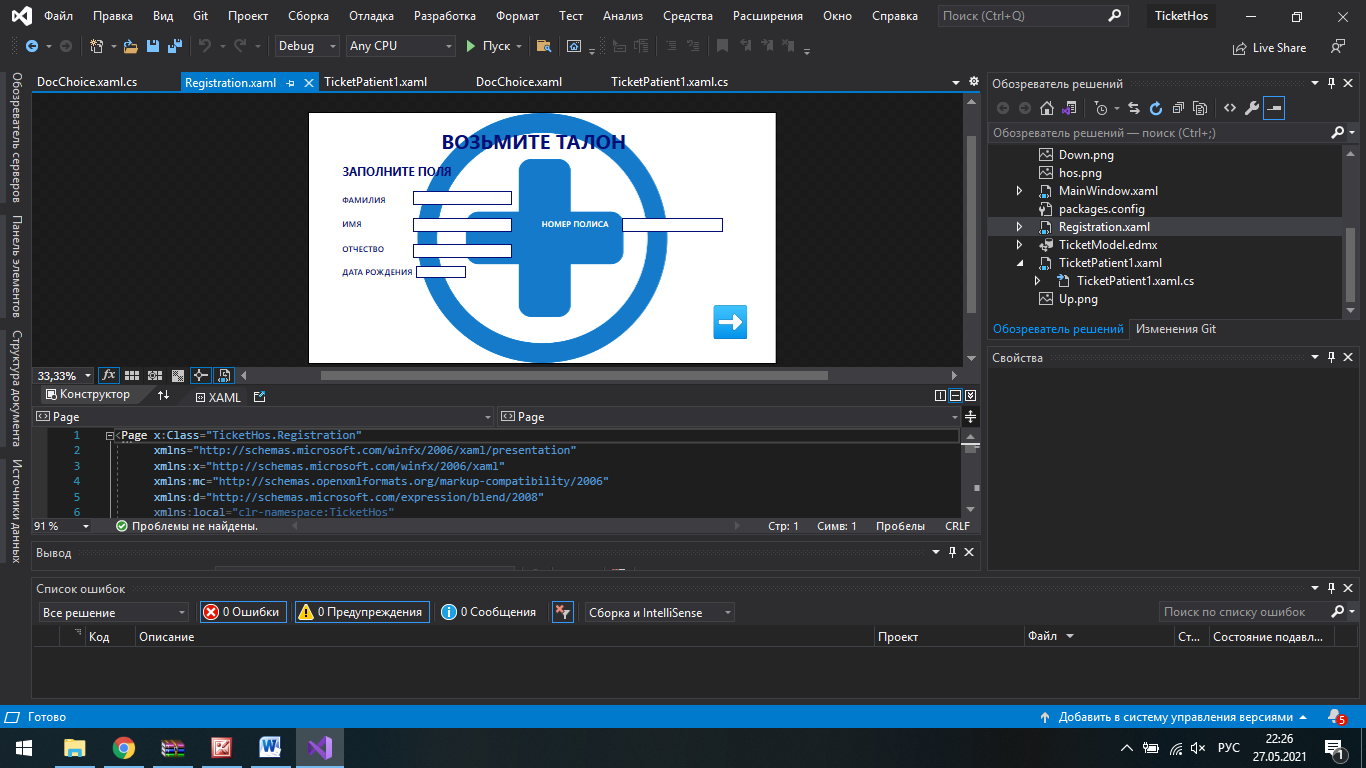


Рисунок 4.1 – Страница регистрации

Если пользователь не заполнил все поля – выходит ошибка с просьбой заполнить их (Рисунок 4.2).

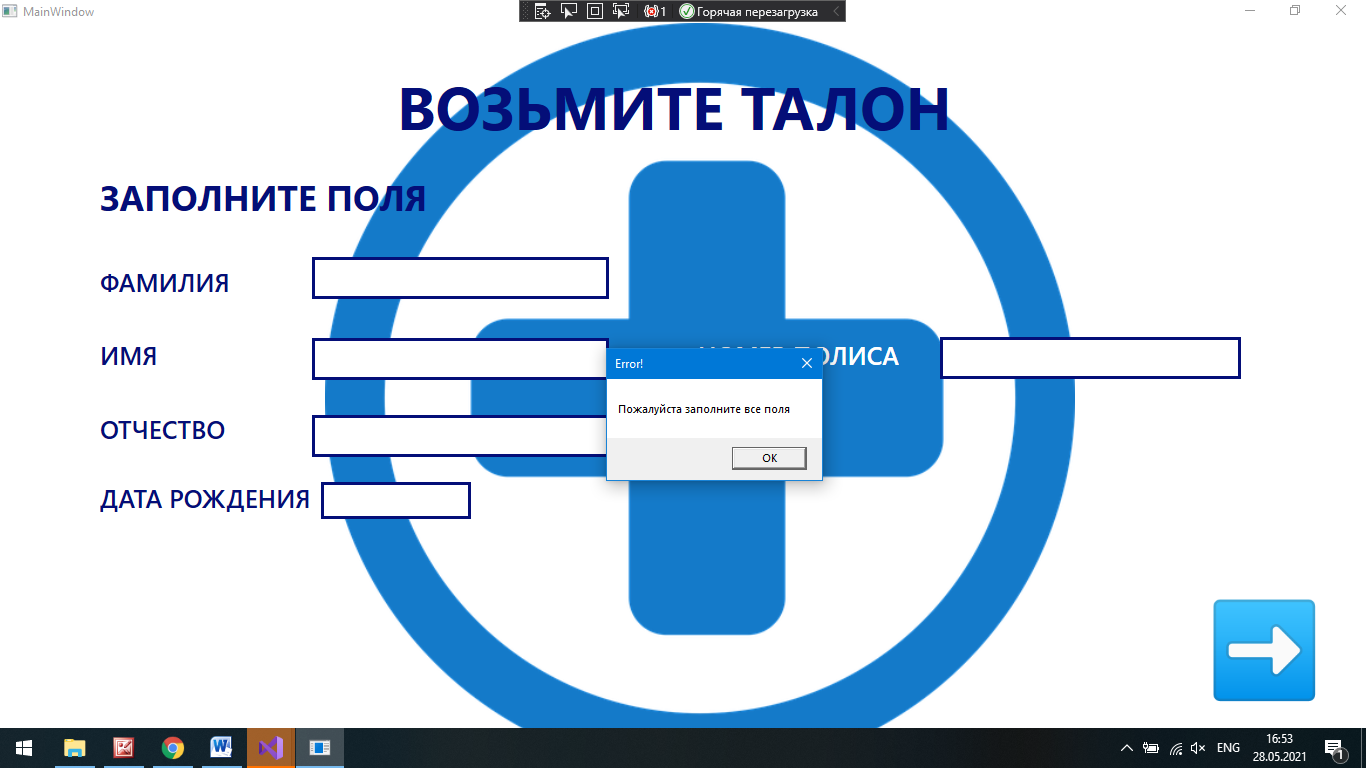


Рисунок 4.2 – Сообщение

Далее по нажатию кнопки, после заполнения всех полей, которая находится в правом нижнем углу, происходит переход на другую страницу с выбором направления, врача и свободного времени. При выборе направления в списке врачей можно выбрать только врача с выбранным направлением. Если пользователь не заполняет все требуемые данные, так же выходит сообщение с просьбой заполнить данные. Если уже есть запись на конкретную дату и на конкретное время, то время не отображается. При нажатии кнопки в нижнем правом углу выбранные данные записываются в базу данных и переходит на другую страницу (Рисунок 4.3).

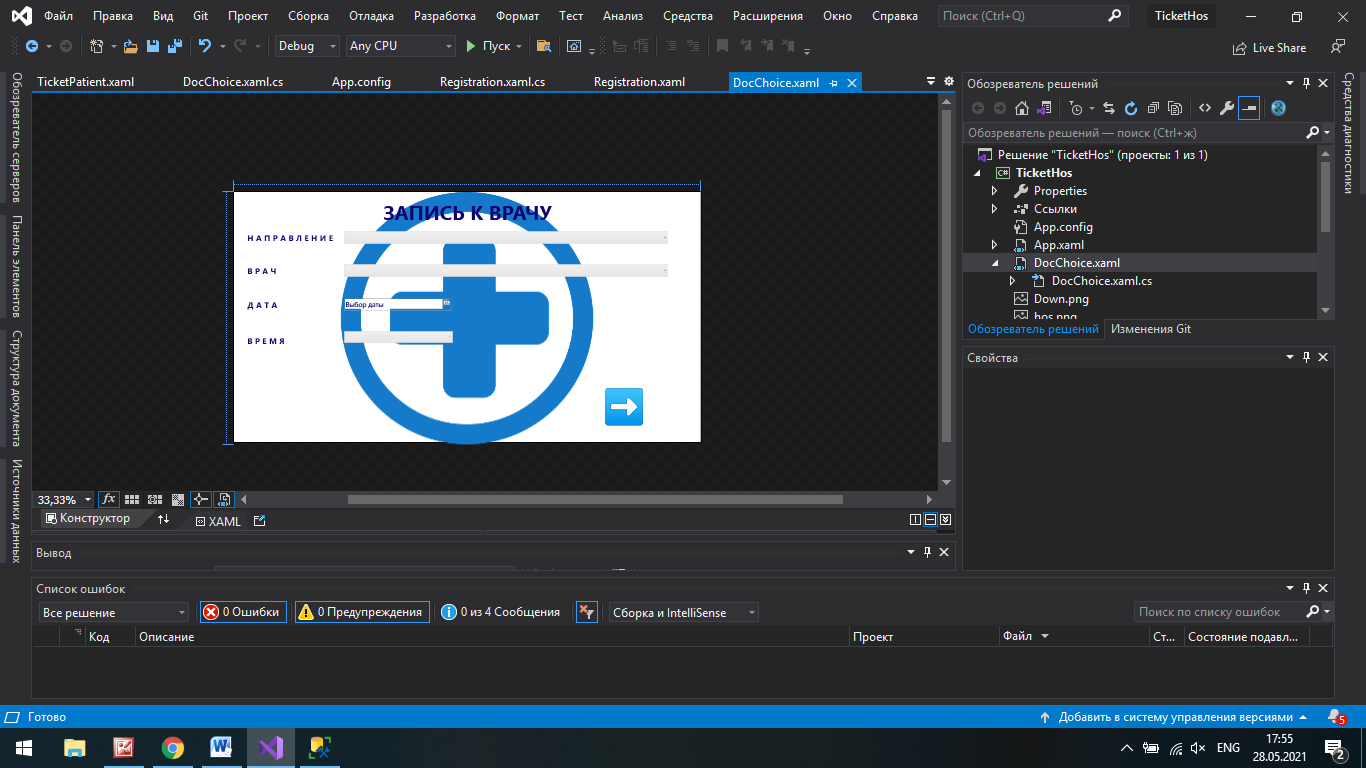


Рисунок 4.3– Страница выбора врача и свободного времени

Далее открывается страница, где изображен талон, с которым он приходит к врачу уже по записи. В талоне прописана необходимая информация для пациента: врач, кабинет врача, дата и время, адрес расположения больницы. Данные берутся уже из базы данных, где это все прописано. При нажатии кнопки справа в нижнем углу приложение возвращается на изначальную страницу (Рисунок 4.4).

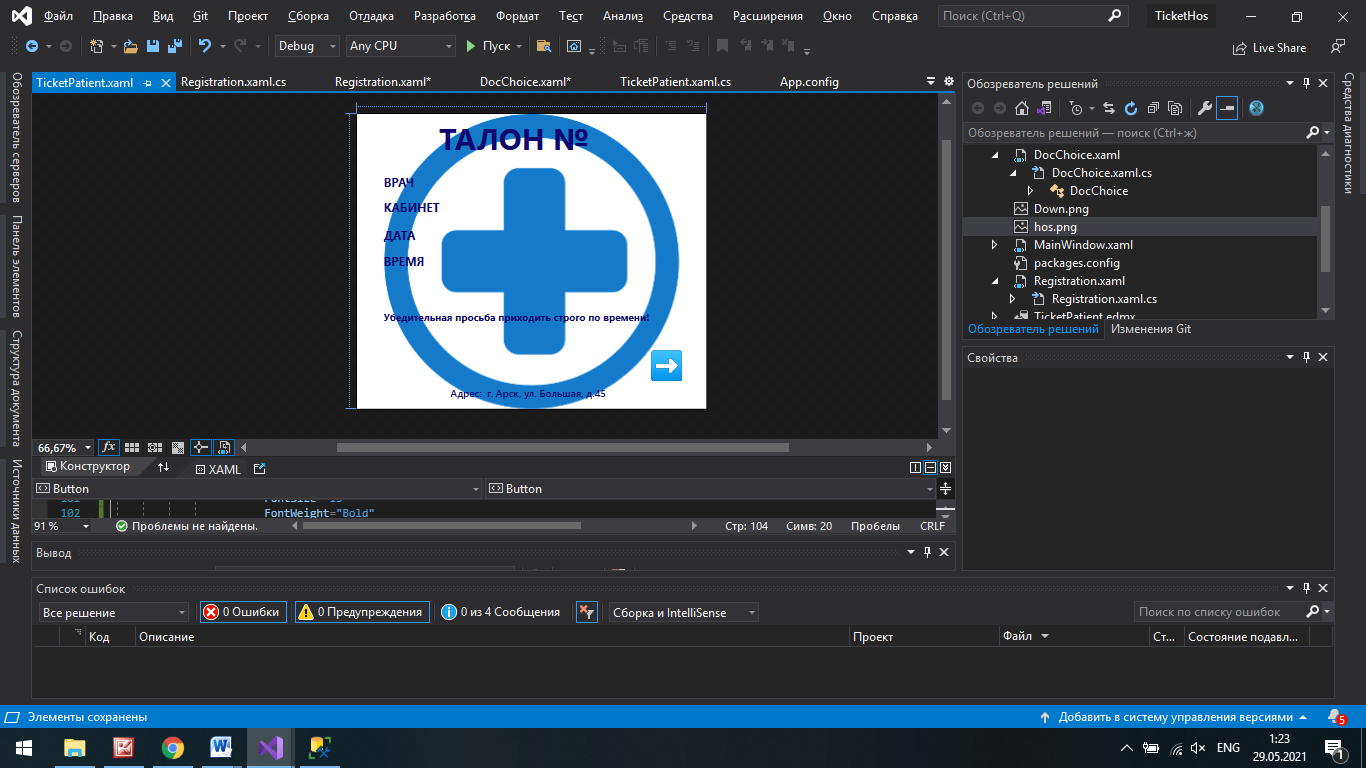


Рисунок 4.4 – Талон

# 5 ТЕСТИРОВАНИЕ

## 5.1 Тест-кейсы

Тестирование — это процесс выполнения программного обеспечения с намерением найти в нем ошибку - довольно необычный процесс, так как этот процесс разрушительный. Целью тестирования является повышение вероятности того, что приложение, предназначенное для тестирования, будет функционировать правильно и соответствовать всем описанным требованиям. Так же целью является предоставление актуальной информации о состоянии продукта на данный момент.

Тест-кейс – это профессиональная документация тестировщика, последовательность действий направленная на проверку какого-либо функционала, описывающая как прийти к фактическому результату. Тест-кейс служит помощником тестировщикам для проверки продукта без ознакомления со всей документацией. Удобный в поддержке тест-кейс сэкономит много времени тестировщикам.

Каждый выполненный тест-кейс должен предоставлять один из трех результатов:

* Положительный результат, если фактический результат равен ожидаемому результату;
* Отрицательный результат, если фактический результат не равен ожидаемому результату. В этом случае, найдена ошибка;
* Выполнение теста блокировано, если после одного из шагов продолжение теста невозможно. В этом случае также, найдена ошибка.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID – test1 | | | | |
| Приоритет – medium | | | | |
| Краткое описание: При выборе конкретного определенного направления, пациенту должны отобразиться врачи только выбранной специальности. | | | | |
| Входное значение | Шаги | Ожидаемый результат | Фактический результат | Статус |
| Направление: Кардиолог | 1. На странице с предложенными врачами найти список направлений 2. Выставить ограничения по направлению 3. Нажать на список врачей | В списке с врачами должны отобразиться врачи конкретной специальности | В списке с врачами отображаются врачи конкретной специальности | Успешно |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID – test2 | | | | |
| Приоритет – medium | | | | |
| Краткое описание: При выборе конкретного определенного направления, врача и даты повторно два раза, пациенту должны отобразиться часы приема не выбранные заранее другим пациентом. | | | | |
| Входное значение | Шаги | Ожидаемый результат | Фактический результат | Статус |
| Направление: Кардиолог  Врач: Шигапов Айрат Фаргатович  Дата: 28.05.2021  Время: 08:00 | 1. На странице с предложенными врачами найти список направлений, врачей, даты и времени 2. Выставить ограничения 3. Нажать на кнопку далее 4. Заново пройти авторизацию для получения талона 5. Ввести такие же ограничения в направлении, во врачах и в дате 6. Открыть список времени | В списке с часами приема не должны отображаться часы уже выбранные другим пациентом | В списке с часами приема не отображаются часы уже выбранные другим пациентом | Успешно |

# ЗАКЛЮЧЕНИЕ

Задачей курсового проекта было создать программу, предназначенную для автоматизации рабочего места регистратора поликлиники, обеспечивающая, уменьшение ручного труда регистратора поликлиники и увеличение пропускной способности пациентов.

После ознакомления с основными правилами учета пациентов, были выделены главные задачи и подзадачи для разработки программного продукта. Так была создана база данных в среде MS SQL, состоящая из 5 таблиц, между которыми есть связь. База данных, в свою очередь, включает в себя данные о врачах, пациентах, приеме и расписаниях, которые необходимы для работы поликлиники. После чего был создан удобный и простой в использовании проект, в программе Visual Studio.

В проекте выполнены и реализованы все поставленные задачи.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. К.ДЖ. Дейт: Введение в системы баз данных / К.ДЖ. Дейт; пер. с англ. – М.: Издательский дом "Вильямс", 2005. — 1328 с;
2. UML [Электронный ресурс]: Википедия. Свободная энциклопедия. – URL: https://ru.wikipedia.org/wiki/UML (дата обращения: 03.05.2021);
3. Р.Дж. Мюллер: [Базы данных и UML. Проектирование](http://progbook.ru/uml/918-myuller-bazy-dannyh-i-uml-proektirovanie.html)/ Р.Дж. Мюллер; пер. с англ. Е.Молодцова, 2002 – 420 с;
4. Э.Троелсен, Ф.Джепикс: Язык программирования С# 6.0 и платформа .NET 4.6 / Э.Троелсен, Ф.Джепикс, 2016 – 1140 с;
5. Xamarin [Электронный ресурс]: Википедия. Свободная энциклопедия. – URL: https://ru.wikipedia.org/wiki/Xamarin (дата обращения: 27.03.2021);
6. WPF [Электронный ресурс]: Microsoft. Свободная энциклопедия. – URL: https://docs.microsoft.com/ru-ru/dotnet/desktop/wpf/?view=netdesktop-5.0 (дата обращения: 27.03.2021);
7. Рихтер Д. CLR via C#: программирование на платформе Microsoft .NET Framework 4.5 на языке C# / Д. Рихтер; пер. с англ. Е. Матвеев. – 4-е изд. – М.: Питер, 2019. – 895 с;
8. А.Д. Хомоненко, М.Г. Мальцев, В.М. Цыганков: Базы данных. Учебник для высших учебных заведений. Раздел про проектирование БД/ А.Д. Хомоненко, М.Г. Мальцев, В.М. Цыганков; 2009 – 736 с;
9. Дж.Р. Грофф, Пол Н. Вайнберг, Э.Дж. Оппель: SQL. Полное руководство/ Дж.Р. Грофф, Пол Н. Вайнберг, Э.Дж. Оппель; пер с англ. М: ООО «И.Д. Вильямс», 2016 – 960 с;
10. SQL [Электронный ресурс]: Википедия. Свободная энциклопедия. – URL: https://ru.wikipedia.org/wiki/SQL (дата обращения: 01.04.2021);
11. MySQL [Электронный ресурс]: Википедия. Свободная энциклопедия. – URL: https://ru.wikipedia.org/wiki/MySQL (дата обращения 02.04.2021);
12. П. Дюбуа: MySQL. Сборник рецептов/ П. Дюбуа; пер с англ. СПб:Символ – плюс, 2007 – 1056 с;
13. С.М. Старолетов: Основы тестирования и верификации программного обеспечения / С.М. Старолетов, 2020 – 344 с;
14. Мэтью Мак-Дональд: WPF. Windows Presentation Foundation в .NET 4.5 с примерами на C# 5.0 / Мэтью Мак-Дональд, 2013 – 1024 с;

# ПРИЛОЖЕНИЕ A

1. ВВЕДЕНИЕ

Разработать приложение выдачи талонов с использованием баз данных и язык программирования С#. Приложение предназначено для облегчения работы в поликлинике.

1. НАЗВАНИЕ ПРОГРАММЫ И ОБЛАСТЬ ПРИМЕНЕНИЯ

Наименование работы: Разработка информационной системы выдачи талонов на запись к врачу

Информационная система предназначена для записи к врачу в электронном формате.

1. ОСНОВАНИЕ ДЛЯ РАЗРАБОТКИ

Основанием для разработки является задание к курсовому проекту по дисциплине МДК.01.01 Разработка программных модулей, выданное учебным учреждением «МЦК – КТИТС»

Плановые сроки начала работы: 14.01.21

Плановый срок окончания работы: 28.05.21

1. НАЗНАЧЕНИЕ РАЗРАБОТКИ

Данная информационная система позволяет облегчить процесс записи к врачу путем электронных талонов. Данная информационная система предназначена для пациента.

Функциональные требования:

Данная программа может быть использована на всех ЭВМ, где установлено программное и аппаратное обеспечение, удовлетворяющие минимальным требованиям к программным и аппаратным средствам для нормального функционирования приложения.

1. ТРЕБОВАНИЯ К ПРОГРАММЕ ИЛИ ПРОГРАММНОМУ ИЗДЕЛИЮ

4.1 Требования к функциональным характеристикам

Разрабатываемый программный продукт должен обладать следующими функциями:

* добавление данных в страницу регистрации;
* база данных должна хранить информацию о врачах;
* база данных должна хранить информацию о свободном времени врача;
* иметь доступный и простой интерфейс.

4.2

Требования к надежности.

Надежность системы в целом зависит от надежности используемой операционной системы. Проект должен обслуживать без сбоев одновременное взаимодействие с программой нескольких пользователей на операционных системах Windows 7, 8, 10. Серверная часть должна обслуживать без сбоев одновременное подключение и работу до 32000 пользователей. Обе части должны без потерь передавать информацию по каналу связи между клиентом и сервером.

1. УСЛОВИЯ ЭКСПЛУАТАЦИИ

Стандартные условия эксплуатации программных продуктов.

1. СТАДИИ И ЭТАПЫ РАЗРАБОТКИ

Разработка должна быть проведена в три стадии:

1. Разработка технического задания

Необходимо выполнить действия, такие как: постановка задачи, определение требований к техническим средствам и к информационной системе, так же определение этапов и сроков реализации проекта и пояснительной записки к данному проекту.

2. Рабочее проектирование

Необходимо выполнить действия, такие как: разработка СУБД и информационной системы, тестирование данной информационной системы.

3. Ввод в действие

Необходимо выполнить действия, такие как: подготовка, в последующем передача проекта с документацией в эксплуатацию.

1. ТРЕБОВАНИЯ К СОСТАВУ И ПАРАМЕТРАМ ТЕХНИЧЕСКИХ СРЕДСТВ

7.1 Требования к функциональным характеристикам.

Для нормального отображения приложения необходимо:

* наличие адаптера подключенного к сети;
* ОС Windows 7, 8, 10;
* процессор x86 с тактовой частотой, не менее 1 ГГц;
* оперативная память объемом, не менее 1 Гб;

Один компьютер для СУБД (основной), включающий в себя:

* процессор x86 с тактовой частотой, не менее 1 ГГц;
* оперативную память объемом, не менее 1 Гб;
* постоянную память не менее 500 Гб.

7.2 Требования к информационной и программной совместимости.

В качестве хранения информации используется удаленный сервер, т.е. необходимо подключение к интернету

1. ТРЕБОВАНИЯ К ПРОГРАММНОЙ ДОКУМЕНТАЦИИ

Разрабатываемая система должна включать справочную информацию о работе системы и подсказки пользователю.

1. ПОРЯДОК КОНТРОЛЯ И ПРИЕМКИ

Испытание системы производится путем проверки функционала всех элементов данного проекта, переходы между страницами, нажатие на кнопки, реализованные в программе.

На основании протокола испытаний исполнитель, совместно с заказчиком, подписывают акт приемки­–сдачи программы в эксплуатацию.

Испытание представленного приложения и контроль качества его работы:

* подключение к базе данных;
* авторизация/регистрация;
* проверка выбора направления;
* проверка выбора врача;
* проверка вывода свободного времени врача;
* проверка вывода талона.

# ПРИЛОЖЕНИЕ B

**Registration.xaml**

<Page x:Class="TicketHos.Registration"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:TicketHos"

mc:Ignorable="d"

Background="White"

Height="750" Width="1400" >

<Grid>

<Grid.Background>

<ImageBrush ImageSource="hos.png"

TileMode="Tile"

Stretch="Uniform"/>

</Grid.Background>

<StackPanel>

<StackPanel>

<Label FontSize="60"

FontWeight="Bold"

Foreground="#FF040E78"

Margin="392,37,266,-18"

Height="91" >

ВОЗЬМИТЕ ТАЛОН

</Label>

</StackPanel>

<StackPanel Height="92">

<TextBlock

Foreground="#FF040E78"

Text="ЗАПОЛНИТЕ ПОЛЯ"

Margin="100,40,813,301"

FontSize="35"

FontWeight="Bold"

Height="53" />

</StackPanel>

<StackPanel Height="103">

<TextBlock Foreground="#FF040E78"

Margin="100,40,1071,0"

FontSize="25"

FontWeight="DemiBold"

Text="ФАМИЛИЯ"

Height="52"/>

<TextBox x:Name="Last\_name"

FontSize="22"

HorizontalAlignment="Left"

Height="42"

Margin="312,-60,0,0"

VerticalAlignment="Top"

Width="297"

BorderBrush="#FF040E78"

BorderThickness="3"/>

</StackPanel>

<StackPanel Height="74">

<TextBlock Foreground="#FF040E78"

Margin="100,10,1191,256"

FontSize="25"

FontWeight="DemiBold"

Text="ИМЯ"

Height="44"/>

<TextBox x:Name="First\_name"

FontSize="22"

HorizontalAlignment="Left"

Height="42"

Margin="312,-300,0,0"

VerticalAlignment="Top"

Width="297"

BorderBrush="#FF040E78"

BorderThickness="3"/>

<TextBlock Foreground="White"

Margin="698,-300,297,309"

FontSize="25"

FontWeight="DemiBold"

Text="НОМЕР ПОЛИСА"

Height="50"/>

<TextBox x:Name="Polis"

FontSize="22"

HorizontalAlignment="Left"

Height="42"

Margin="940,-360,0,0"

VerticalAlignment="Top"

Width="301"

MaxLength="16"

BorderBrush="#FF040E78"

BorderThickness="3"/>

</StackPanel>

<StackPanel Height="70">

<TextBlock Foreground="#FF040E78"

Margin="100,10,1001,309"

FontSize="25"

FontWeight="DemiBold"

Text="ОТЧЕСТВО"

Height="44"/>

<TextBox x:Name="Middle\_name"

FontSize="22"

HorizontalAlignment="Left"

Height="42"

Margin="312,-350,0,0"

VerticalAlignment="Top"

Width="297"

BorderBrush="#FF040E78"

BorderThickness="3"/>

</StackPanel>

<StackPanel Height="70"

Margin="100,0,0,0"

Orientation="Horizontal">

<TextBlock Foreground="#FF040E78"

FontSize="25"

FontWeight="DemiBold"

Text="ДАТА РОЖДЕНИЯ"

Height="51"/>

<TextBox x:Name="DayOfBirth"

FontSize="22"

Margin="10"

Text="{Binding Patient, StringFormat='{}{0:dd/MM/yyyy}'}"

HorizontalAlignment="Left"

Height="37"

VerticalAlignment="Top"

Width="150"

BorderBrush="#FF040E78"

BorderThickness="3"/>

</StackPanel>

<StackPanel Height="203" >

<Button x:Name="Doc"

FontSize="40"

FontWeight="Bold"

HorizontalAlignment="Left"

Height="112"

Margin="1206,50,0,20"

Width="112"

Click="Button\_Click"

BorderBrush="{x:Null}"

Foreground="{x:Null}"

Background="{x:Null}">

<Image Source="Up.png"

Height="112"

Width="112"/>

</Button>

</StackPanel>

</StackPanel>

</Grid>

</Page>

**Registration.xaml.cs**

using System**;**

using System.Collections.Generic;

using System.Linq;

using System.Data;

using System.Data.SqlClient;

using System.Data.Entity;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace TicketHos

{

/// <summary>

/// Логика взаимодействия для Registration.xaml

/// </summary>

public partial class Registration : Page

{

public static TicketEntities ticketEntities = new TicketEntities();

Patient contextPatient;

public Registration()

{

InitializeComponent();

contextPatient = new Patient();

DataContext = contextPatient;

}

private void Button\_Click(object sender, RoutedEventArgs e)

{

if (First\_name.Text != "" && Last\_name.Text != "" && Middle\_name.Text != "" && Polis.Text != "")

{

try

{

contextPatient.FirstName = First\_name.Text;

contextPatient.LastName = Last\_name.Text;

contextPatient.MiddleName = Middle\_name.Text;

contextPatient.DateOfBirth = Convert.ToDateTime(DayOfBirth.Text);

contextPatient.Police = Convert.ToString(Polis.Text);

ticketEntities.Patient.Add(contextPatient);

ticketEntities.SaveChanges();

NavigationService.Navigate(new DocChoice(contextPatient));

}

catch (Exception er)

{

MessageBox.Show(er.Message);

}

}

else

{

MessageBox.Show("Пожалуйста заполните все поля", "Error!");

}

}

}

}

DocChoise.xaml

<Page x:Class="TicketHos.DocChoice"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:TicketHos"

mc:Ignorable="d"

d:DesignHeight="750" Width="1400"

Background="White">

<Grid OpacityMask="WhiteSmoke" Margin="0,0,2,-7">

<Grid.Background>

<ImageBrush ImageSource="hos.png"

TileMode="Tile"

Stretch="Uniform"/>

</Grid.Background>

<StackPanel Margin="0,0,0,10">

<TextBlock Text="ЗАПИСЬ К ВРАЧУ"

Foreground="#FF0E0572"

FontSize="60"

FontWeight="Bold"

Height="102"

Margin="448,20,438,0"/>

</StackPanel>

<StackPanel Margin="0,0,1,276">

<TextBlock Text="Н А П Р А В Л Е Н И Е"

Foreground="#FF0E0572"

FontSize="25"

FontWeight="Bold"

Margin="40,120,1091,364" />

<ComboBox x:Name="Direction"

FontSize="20"

FontWeight="Bold"

Foreground="#FF0E0572"

DisplayMemberPath="Doctor"

SelectedItem="{Binding Specialization}"

SelectionChanged="Direction\_SelectionChanged"

Margin="330,-400,94,362"/>

</StackPanel>

<StackPanel Margin="0,0,1,276">

<TextBlock Text="В Р А Ч"

Foreground="#FF0E0572"

FontSize="25"

FontWeight="Bold"

Margin="40,220,1195,364" />

<ComboBox x:Name="DoctorsBox"

FontSize="20"

FontWeight="Bold"

Foreground="#FF0E0572"

DisplayMemberPath="NameSpecialist"

SelectedItem="{Binding Specialist}"

Margin="330,-400,94,362"/>

</StackPanel>

<StackPanel Margin="0,250,10,333">

<TextBlock Text="Д А Т А"

Foreground="#FF0E0572"

FontSize="25"

FontWeight="Bold"

Margin="40,70,1195,364" />

<DatePicker x:Name="date\_Birth"

FontSize="20"

FontWeight="Medium"

Foreground="#FF0E0572"

Language="ru"

Height="38"

Margin="330,-760,734,0"

SelectedDateChanged="date\_Birth\_SelectedDateChanged"/>

</StackPanel>

<StackPanel Margin="0,378,0,234">

<TextBlock Text="В Р Е М Я" Foreground="#FF0E0572"

FontSize="25"

FontWeight="Bold"

Margin="40,50,1195,364" />

<ComboBox x:Name="TimeBox"

FontSize="20"

FontWeight="Medium"

Foreground="#FF0E0572"

Height="38"

Margin="330,-550,742,229">

</ComboBox>

</StackPanel>

<StackPanel Margin="0,523,0,25" >

<Button x:Name="Doc1"

FontSize="40"

FontWeight="Bold"

HorizontalAlignment="Left"

Height="128"

Margin="1100,0,0,20"

Width="141"

Click="Doc1\_Click" Background="{x:Null}" BorderBrush="{x:Null}" Foreground="{x:Null}">

<Image Source="Up.png" />

</Button>

</StackPanel>

</Grid>

</Page>

DocChoise.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Data;

using System.Data.SqlClient;

using System.Data.Entity;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace TicketHos

{

/// <summary>

/// Логика взаимодействия для DocChoice.xaml

/// </summary>

public partial class DocChoice : Page

{

Ticket contextTicket;

public DocChoice(Patient postPatient)

{

InitializeComponent();

contextTicket = new Ticket() { Patient = postPatient};

DataContext = contextTicket;

Direction.ItemsSource = Registration.ticketEntities.Specialization.ToList();

date\_Birth.BlackoutDates.Add(new CalendarDateRange(new DateTime(1990, 1, 1), DateTime.Now.AddDays(-1)));

}

private void Doc1\_Click(object sender, RoutedEventArgs e)

{

if (Direction.Text != "" && DoctorsBox.Text != "" && date\_Birth.Text != "" && TimeBox.Text != "")

{

ChoiseDoctor();

NavigationService.Navigate(new TicketPatient(contextTicket));

}

else

{

MessageBox.Show("Введите все данные", "Ошибка");

}

Registration.ticketEntities.Ticket.Add(contextTicket);

Registration.ticketEntities.SaveChanges();

}

private void Direction\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

if (Direction.SelectedItem != null)

{

DoctorsBox.ItemsSource = ((Direction.SelectedItem) as Specialization).Specialist.ToList();

}

}

private void ChoiseDoctor()

{

if (Direction.Text != "" && DoctorsBox.Text != "" && date\_Birth.Text != "" && TimeBox.Text != "")

{

contextTicket.Time = new Time() { Time1 = (TimeSpan)TimeBox.SelectedItem, Date = date\_Birth.SelectedDate };

}

else

{

MessageBox.Show("Введите все данные", "Ошибка");

}

}

private void date\_Birth\_SelectedDateChanged(object sender, SelectionChangedEventArgs e)

{

var times = new List<TimeSpan>();

for (int i = 8; i <= 18; i++)

{

times.Add(TimeSpan.FromHours(i));

}

foreach (var time in times)

{

if (Registration.ticketEntities.Time.FirstOrDefault(t => t.Time1.Value == time && t.Date == date\_Birth.SelectedDate) != null)

continue;

TimeBox.Items.Add(time);

}

}

}

}

TicketPatient.xaml

<Page x:Class="TicketHos.TicketPatient"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:TicketHos"

mc:Ignorable="d"

Background="White" Width="524" Height="442">

<Grid>

<Grid.Background>

<ImageBrush ImageSource="hos.png"

TileMode="Tile"

Stretch="Uniform"/>

</Grid.Background>

<StackPanel Margin="0,0,0,338">

<Label Content="ТАЛОН №"

HorizontalAlignment="Center"

Foreground="#FF0E0572"

FontSize="45"

FontWeight="Bold"

Margin="114,0,125,0"

Width="277"

Height="83"/>

<Label Content="{Binding Patient.Id}"

Height="78"

FontSize="45"

Foreground="#FF0E0572"

FontWeight="Bold"

Margin="360,-90,90,0" />

</StackPanel>

<StackPanel Margin="0,80,1,297">

<TextBlock Text="ВРАЧ"

Foreground="#FF0E0572"

FontSize="18"

FontWeight="Bold"

Margin="40,10,394,364"

Height="27" />

<TextBox x:Name="DocTBox"

IsReadOnly="True"

Margin="153,-400,103,364"

FontSize="15"

Foreground="#FF0E0572"

FontWeight="Medium"

Text="{Binding Specialist.NameSpecialist}"

Height="22" Background="{x:Null}" BorderBrush="{x:Null}">

</TextBox>

</StackPanel>

<StackPanel Margin="0,127,0,235">

<TextBlock Text="КАБИНЕТ"

Foreground="#FF0E0572"

FontSize="18"

FontWeight="Bold"

Margin="40,0,394,364" Height="27" />

<TextBox x:Name="CabBox"

IsReadOnly="True"

Margin="153,-400,103,364"

Height="22"

FontSize="15"

Foreground="#FF0E0572"

Text="{Binding Specialist.Office}"

FontWeight="Medium" BorderBrush="{x:Null}" Background="{x:Null}"/>

</StackPanel>

<StackPanel Margin="0,150,10,206">

<TextBlock Text="ДАТА"

Foreground="#FF0E0572"

FontSize="18"

FontWeight="Bold"

Margin="40,20,346,364" />

<TextBox x:Name="DateTBox"

IsReadOnly="True"

Margin="153,-400,94,364"

Height="22"

FontSize="15"

Foreground="#FF0E0572"

FontWeight="Medium"

Text="{Binding Time.Date}" BorderBrush="{x:Null}" Background="{x:Null}">

</TextBox>

</StackPanel>

<StackPanel Margin="0,189,0,161" >

<TextBlock Text="ВРЕМЯ"

Foreground="#FF0E0572"

FontSize="18"

FontWeight="Bold"

Margin="40,20,320,364" />

<TextBox x:Name="TimeTBox"

IsReadOnly="True"

Margin="153,-400,103,364"

Height="22"

FontSize="15"

Foreground="#FF0E0572"

FontWeight="Medium"

Text="{Binding Time.Time1}"

Background="{x:Null}"

BorderBrush="{x:Null}">

</TextBox>

</StackPanel>

<StackPanel Margin="0,294,0,10">

<TextBlock Text="Убедительная просьба приходить строго по времени!"

Foreground="#FF0E0572"

FontSize="15"

FontWeight="Bold"

Margin="40,0,10,364" />

<Button Margin="433,-650,28,-50"

Height="60"

Background="{x:Null}"

BorderBrush="{x:Null}"

Foreground="{x:Null}"

Click="Button\_Click">

<Image Source="Up.png" Width="51" Height="69"/>

</Button>

<TextBlock Text="Адрес: г. Арск, ул. Большая, д.45"

Foreground="#FF0E0572"

FontSize="15"

FontWeight="Medium"

Margin="140,-270,91,364" />

</StackPanel>

</Grid>

</Page>

TicketPatient.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace TicketHos

{

/// <summary>

/// Логика взаимодействия для TicketPatient.xaml

/// </summary>

public partial class TicketPatient : Page

{

Ticket contextTicket;

public TicketPatient(Ticket postTicket)

{

InitializeComponent();

contextTicket = postTicket;

DataContext = contextTicket;

}

private void Button\_Click(object sender, RoutedEventArgs e)

{

this.NavigationService.Navigate(new Uri("Registration.xaml", UriKind.RelativeOrAbsolute));

}

}

}

# ПРИЛОЖЕНИЕ С

1. Папка с приложением;

2. Скрипт для загрузки базы данных.