*Министерство образования, науки и молодежной политики Республики Коми*

*Государственное профессиональное образовательное учреждение*

*«Воркутинский горно-экономический колледж»*

*Специальность: «Информационные системы и программирование»*

*К защите допущен:*

*Зам. директора по УПР*

*Н.Н. Кукош*

*ДИПЛОМНЫЙ ПРОЕКТ*

*«Разработка информационной системы «NevaTelecom».*

*Пояснительная записка**ДП.09.02.07.00.00.05.ПЗ*

*Руководитель проекта:*

*Д.П. Егоров*

*Выполнил студент группы: ИСП-18*

*А.В Костура*

*Воркута*

*2022*

*Воркута*

*2022*

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc105630470)

[ГЛАВА 1. ПРОЕКТИРОВАНИЕ ТРЕБОВАНИЙ К ИНФОРМАЦИОННОЙ СИСТЕМЕ 4](#_Toc105630471)

[1.1 Анализ Диаграммы вариантов использования (Usecase) 4](#_Toc105630472)

[1.2 Анализ диаграммы деятельности (Activity) 5](#_Toc105630473)

[1.3 Анализ диаграммы последовательностей (Sequence) 8](#_Toc105630474)

[1.4 Анализ диаграммы состояний (StateMachine) 10](#_Toc105630475)

[ГЛАВА 2. ПРОЕТИРОВАНИЕ БАЗЫ ДАННЫХ 12](#_Toc105630476)

[2.1 Анализ диаграммы ERD 12](#_Toc105630477)

[2.2 Выбор инструментария. 15](#_Toc105630478)

[2.3 Разработка базы данных 16](#_Toc105630479)

[ГЛАВА 3. РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ 21](#_Toc105630480)

[3.1 Выбор инструментария 21](#_Toc105630481)

[3.2 Разработка интерфейса информационной системы 23](#_Toc105630482)

[3.3 Программирование информационной системы 42](#_Toc105630483)

[ЗАКЛЮЧЕНИЕ 57](#_Toc105630484)

[СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 58](#_Toc105630485)

Изм.

Лист

№ докум.

Подпись

Дата

Лист

2

ДП.09.02.07.00.00.05.ПЗ

Разраб.

*Костура А.В. А.В*

Руководит..

*Егоров Д.П. Д.П*

*«Разработка информационной системы «NevaTelecom».*

Лит.

Листов

58

ИСП-18

# ВВЕДЕНИЕ

На данным момент в каждой компании есть свой отдел кадров, который выполняет определенные функции. С каждым годом появляются новые задачи в работе и поэтому, на рынке программных продуктов имеется множества программ, которые могут быть использованы для автоматизации работы, но универсальное программное обеспечение, не может учитывать требование определенных компаний, что является существенным минусом.

Основная особенность специализированных программных продуктов – это их функциональность и универсальность. Разработчики предлагают универсальные программных решения, но без учета особенностей самих предприятий, а также тех функций и задач, которым стоит уделить особое внимание.

Часто имеющиеся функции ИС в предлагаемых на рынке программных продуктах избыточны. Их наличие ведет к повышению системных требований для обеспечения функционирования данных систем и к значительному увеличению расходов учреждения на приобретение и эксплуатации данных программных продуктов.

Предполагается, что разработка узкоспециализированной автоматизированной системы, выполненная с учетом требований заказчиков, позволяет упростить работу, не завышая системные требования и при этом сократив расходы компании на приобретение программного продукта.

Целью дипломного проекта является создания информационной системы, содержащей такие модули как:

* Модуль «Абоненты» - позволяет, добавлять новых абонентов в систему, редактировать данные абонентов, а также отслеживать «Активных» и «Не активных» пользователей.
* Модуль «Документы» - позволяет, создавать и работать с документами прям в приложении, а также вывести документы на печать.
* Модуль «CRM» - модуль для работы с заявками абонентов по предоставлению услуг.

Для достижения цели проекта нужно выполнить следующие задачи:

1. На основании анализа требований заказчика к информационной системе, составить основные разделы технического задания.
2. В соответствии с техническим заданием выполнить проектирование и разработку информационной системы в выбранной среде программирования.

# ГЛАВА 1. ПРОЕКТИРОВАНИЕ ТРЕБОВАНИЙ К ИНФОРМАЦИОННОЙ СИСТЕМЕ

Построение uml-диаграмм является важным этапом при проектировании ИС, так как обеспечивает поддержку всех этапов жизненного цикла проекта.

Uml - это язык графического описания для объектного моделирования в области разработки программного обеспечения, для моделирования бизнес-процессов, системного проектирования и отображения организационных структур.

Согласно техническому заданию необходимо построить 4 основных uml-диаграммы:

* Диаграмма вариантов использования (*use case*);
* Диаграмма деятельности (*activity*);
* Диаграмма последовательности (*sequence*);
* Диаграмма состояний (*state machine*).

# 1.1 Анализ Диаграммы вариантов использования (Usecase)

Use Case diagram – это диаграмма вариантов использования. Диаграмма, отражающая отношения между актерами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне.

Прецедент (вариант использования) — это описание поведения системы, когда она взаимодействует с кем-то (или чем-то) из внешней среды.

Актер (Use Case) – это роль объекта вне системы, который прямо взаимодействует с ее частью — конкретным элементом.

При моделировании системы с помощью диаграммы usecase необходимо:

* чётко отделить систему от её окружения;
* определить действующих лиц (актеров), их взаимодействие с системой и ожидаемую функциональность системы;
* определить в глоссарии предметной области понятия, относящиеся к детальному описанию функциональности системы (то есть прецедентов).

При работе с usecase важно помнить несколько простых правил:

* каждый прецедент относится как минимум к одному действующему лицу;
* каждый прецедент имеет инициатора;
* каждый прецедент приводит к соответствующему результату.

На диаграмме (Рис.1.1) изображены 3 основных актера ИС: абонент, технический отдел и отдел по работе с клиентами. Все актеры взаимосвязаны между собой прецедентами, которые расположены в овалах:

1. Человек приходит подает заявление на подключение услуг. Сотрудник отдела по работе с клиентами, заносит данные о новом абоненте в БД.
2. Абонент звонит в отдел по работе с клиентами, делает запрос оказание услуг. После чего формируется заявка и данные добавляются в БД.
3. Сотрудник отдела по работе с клиентами может производить работу с документами, а также редактировать данные об абоненте и заявок на оказание услуг.
4. Сотрудники технического отдела и отдела по работе с клиентами могут просматривать список заявок.

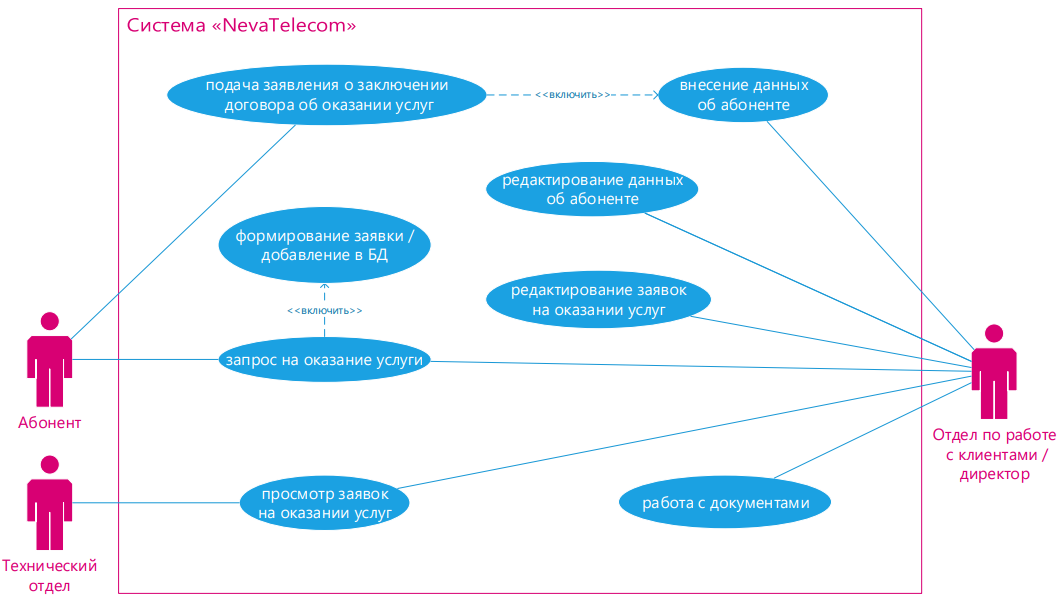


Рисунок.1.1 «Диаграмма use-case»

# 1.2 Анализ диаграммы деятельности (Activity)

Activity diagram (диаграмма деятельности) – это технология, позволяющая описывать логику процедур, бизнес-процессы и потоки работ. Диаграмма для демонстрации рабочего процесса некоторой деятельности, основанной на поэтапных действиях и действиях с поддержкой выбора и параллелизма.

Диаграмма деятельности очень похожа на обычную блок-схему, но основным ее отличием отблок-схемы является активная поддержка параллельных процессов, что объясняет применение диаграммы деятельности для моделирования потоков работ.

Диаграммы деятельности состоят из ограниченного количества фигур:

* Прямоугольники с закруглениями — действия(операция). Узел управления — это абстрактный узел действия, которое координирует потоки действий.
* Ромбы— решения.Узел решения предназначен для определения правила ветвления и различных вариантов дальнейшего развития сценария. В точку ветвления входит ровно один переход, а выходит — два или более.
* Круг— начало процесса (начальное состояние деятельности), является узлом управления, в котором начинается поток (или потоки) при вызове данной деятельности извне.
* Круг с обводкой— окончание процесса (конечный узел), является узлом управления, который останавливает все потоки данной диаграммы деятельности. На диаграмме может быть более одного конечного узла.
* Стрелки идут от начала к концу процесса и показывают потоки управления или потоки объектов (данных).

На диаграмме (Рис.1.2) можно увидеть общий «бассейн», который включает в себя 3 дорожки, которые символизируют роль пользователя или организационное подразделение, осуществляющее определенные действия в рамках данной деятельности:

* абонент;
* технический отдел;
* отдел по работе с клиентами.

Данная диаграмма демонстрирует процесс работы предприятия «NevaTelecom». В первом варианте все начинается с того, что приходит человек и подает заявление на подключение услуг. Сотрудник отдела по работе с клиентами печатает заявление, которое заполняет человек. После чего сотрудник заносит данные о новом абоненте в систему. Второй вариант начинается тогда, когда абонент хочет оформить заявку на оказание услуг. Он звонит в отдел кадров по работе с клиентами. Сотрудник проверяет данные абонента. Если абонент не найден происходит отказ в оформлении заявки. Если абонент найден, то сотрудник вносит данные о заявки в БД. После оформления, сотрудник технического отдела просматривает заявки. Если статус заявки «Требует выезда», то происходит выезд к абоненту, иначе просто решение проблемы абонента. Когда проблема решена сотрудник отдела по работе с клиентами изменяет статус заявки на «Закрыта».

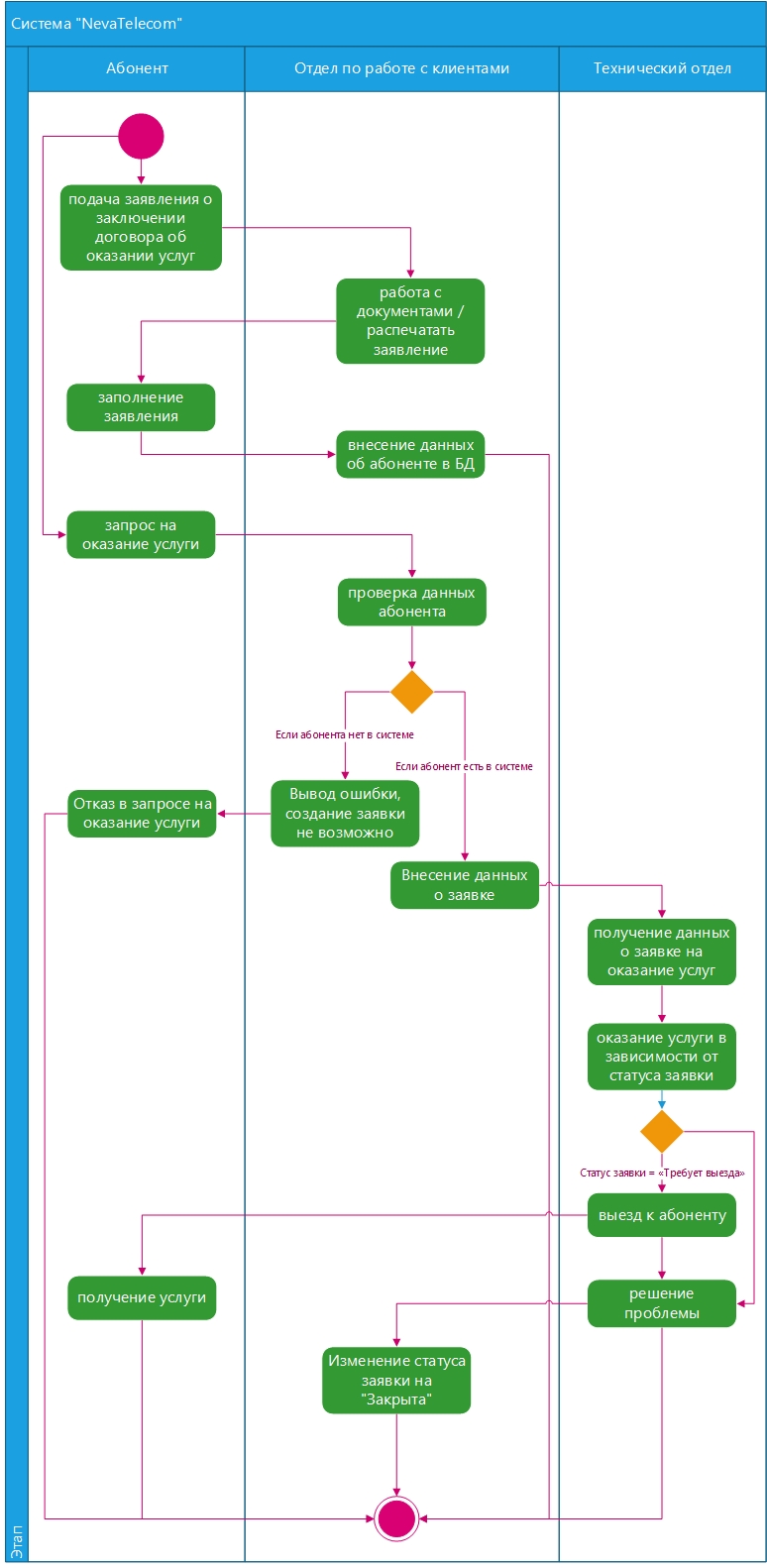


Рисунок.1.2 «Диаграмма Activity»

# 1.3 Анализ диаграммы последовательностей (Sequence)

Sequence Diagram (диаграмма последовательностей) — это диаграмма, на которой для некоторого набора объектов на единой временной оси показан жизненный цикл какого-либо определенного объекта и взаимодействие актеров информационной системы в рамках какого-либо определённого прецедента. Проще говоря, это графическая модель, которая для определенного сценария варианта использования показывает динамику взаимодействия объектов во времени.

Особенности диаграммы последовательностей:

* диаграмма последовательности позволяет изобразить поведение нескольких объектов врамках одного прецедента;
* диаграмма последовательности удобна для представления взаимодействия объектов, нонедля точного определения ихповедения;
* диаграмма показывает экземпляры объектов и сообщения, которыми обмениваются экземпляры в рамках одного прецедента.

Диаграммы последовательностей обычно содержат объекты, которые взаимодействуют в рамках сценария, сообщения, которыми они обмениваются, и возвращаемые результаты, связанные с сообщениями.

* Объекты обозначаются прямоугольниками с подчеркнутыми именами (чтобы отличить их от классов).
* Сообщения (вызовы методов) - линиями со стрелками.
* Возвращаемые результаты - пунктирными линиями со стрелками.
* Актеры изображаются в виде человечков.
* «Линии жизни», отображающие течение времени, изображаются вертикальными пунктирными линиями.
* Прямоугольники, отражающие деятельность объекта или исполнение им определенной функции - прямоугольники на пунктирной «линии жизни».

На диаграмме (Рис.1.3) мы видим, что в сценарии участвуют 3 актера – абонент, технический отдел и отдел по работе с клиентами, но также важными составляющими в данной диаграмме являются приложение и сама база данных.

1. Человек приходит подает заявление на подключение услуг. Сотрудник отдела по работе с клиентами, заносит данные о новом абоненте в БД. Происходит сохранение.
2. Сотрудник отдела по работе с клиентами печатает договор, человек (абонент) заполняет его и отдает сотруднику.
3. Абонент хочет оставить заявку на оказание услуг. Сотрудник отдела по работе с клиентами вводит данные и происходит проверка. Если абонента нет в системе, вывод ошибки и отказ в формировании заявки. Если абонент найден, то сотрудник вводит информацию о заявке. Данные сохраняются в БД. После сотрудник информирует сотрудника тех. отдела.
4. Сотрудник тех. отдела просматривает заявки. Если статус заявки «Требует выезд», то сотрудник выезжает для выполнения заявки, иначе просто выполнение заявки.
5. Сотрудник отдела по работе с клиентами редактирует информацию о заявки и сохраняет изменения в БД.

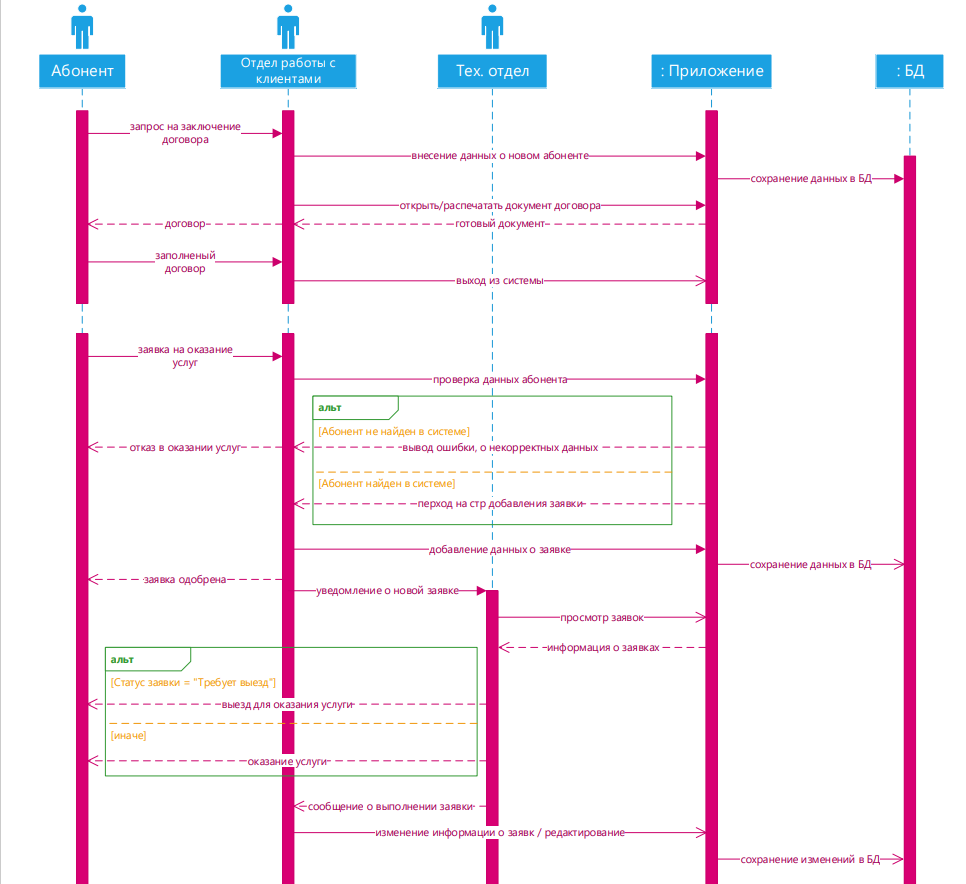


Рисунок.1.3 «Диаграмма Sequence»

# 1.4 Анализ диаграммы состояний (StateMachine)

StateMachineDiagram (диаграмма состояний) – это методология объектно-ориентированного проектирования, предназначенная для представления жизненного цикла объектов в реальном или абстрактном мире.

Диаграммы состояний показывают, как объект переходит из одного состояния в другое. Они служат для моделирования динамических аспектов системы.

Диаграмма «StateMachine»полезна при моделировании жизненного цикла объекта. Данная диаграмма отличается от других диаграмм тем, что она описывает процесс изменения состояний только одного экземпляра определенного класса - одного объекта, причем объекта реактивного, то есть объекта, поведение которого характеризуется его реакцией на внешние события.

Особенности диаграммы состояний:

1. Диаграмма состояний позволяет описывать поведение системы;
2. В объектно-ориентированном подходе разрабатывается диаграмма состояний единственного класса, демонстрирующая поведение одного объекта в течение его жизни;
3. Состояние на диаграмме является более абстрактным понятием, чем состояние объекта (последнее есть комбинация всех данных из полей объекта);
4. Диаграмма позволяет проектировать различные способы реакции на события. Событие – это сообщение, которое возникает в различных точках исполняемого кода при выполнении определенных условий.

Условные обозначения:

1. Круг, обозначающий начальное состояние.
2. Окружность с маленьким кругом внутри, обозначающая конечное состояние (если есть).
3. Скруглённый прямоугольник, обозначающий состояние. Верхушка прямоугольника содержит название состояния. В середине может быть горизонтальная линия, под которой записываются активности, происходящие в данном состоянии.
4. Стрелка, обозначающая переход. Название события (если есть), вызывающего переход, отмечается рядом со стрелкой.

Диаграмма (Рис. 1.4), показывает нам процесс изменения состояний информационной системой при работе с ней сотрудника отдела кадров.

1. Началом является вход сотрудника в систему.
2. Если данные внесены некорректно, то сотрудник соответственно не может зайти в систему.
3. Когда все данные внесены корректно, то сотрудник попадает в гласное меню.
4. Далее происходит работа сотрудника с приложением.
5. После того как сотрудник закончил с работой, он возвращается в главное меню, где может выйти из системы.

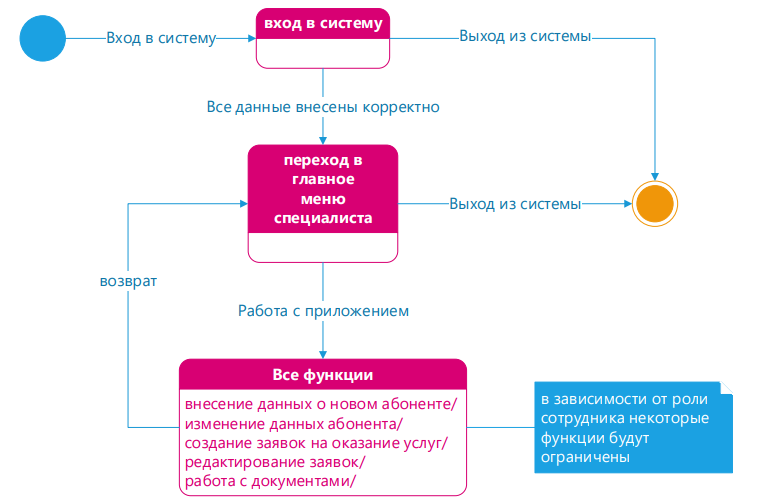


Рисунок.1.4 «Диаграмма State Machine»

# ГЛАВА 2. ПРОЕТИРОВАНИЕ БАЗЫ ДАННЫХ

# 2.1 Анализ диаграммы ERD

Следующим этапом в разработке информационной системы является этап построения диаграммы ERD. Данный этап очень важен, так как значительно упростит работу по построению базы данных приложения.

ERD-схемы используются для проектирования и анализа реляционных баз данных, применяемых в бизнес-процессах.

Реляционные базы данных могут пригодиться в любом бизнес-процессе, где задействованы данные, разбитые на поля, включая сущности, действия и взаимосвязи. Базы данных помогают оптимизировать процессы, извлекать данные и повышать качество результатов.

Диаграмма «сущность-связь» (ERD или ER-диаграмма) — это разновидность блок-схемы, где показано, как разные «сущности» связаны между собой внутри системы.

Под понятием «сущности» подразумеваются объекты или понятия, несущие важную информацию. С точки зрения грамматики, они, как правило, обозначаются существительными.

Связи используются в схемах «сущность-связь» для обозначения взаимодействия между двумя сущностями. Грамматически связи, как правило, выражаются глаголами и несут полезную информацию, которую невозможно получить, опираясь только на типы сущностей.

Первичный ключ — это атрибут или сочетание атрибутов, идентифицирующих один конкретный экземпляр сущности.

Внешний ключ создается каждый раз, когда атрибут привязывается к сущности посредством единичной или множественной связи.

Виды связей:

1. «один-к-одному» или 1:1. Это значит, что одному экземпляру некоторой сущности может соответствовать только один экземпляр другой сущности;
2. «один-ко-многим»или 1:M. Это значит, что одному экземпляру сущности может соответствовать любое количество (M) экземпляров другой сущности. Если известно значение максимального количества экземпляров, то это значение указывается вместо символа М;
3. «много-к-многим» или M:N. Это означает, что нескольким экземплярам одной сущности может соответствовать несколько экземпляров другой сущности.
4. Особенности ERD диаграммы:
5. В основе ERD-диаграмм лежит принцип «рисунок нагляднее текста».
6. ERD-диаграмма графически представляет сущности (entities) предметной области, свойства (attributes) сущностей и связи (relationship) между ними.
7. ERD-диаграммы делятся на концептуальные и физические. В отличие от физических, в концептуальных ERD-диаграммах не учитываются особенности конкретной базы данных. Впоследствии сущности концептуальных ERD-диаграмм становятся таблицами, атрибуты — колонками, а связи реализуются путем миграции ключевых атрибутов родительских сущностей и создания внешних ключей.

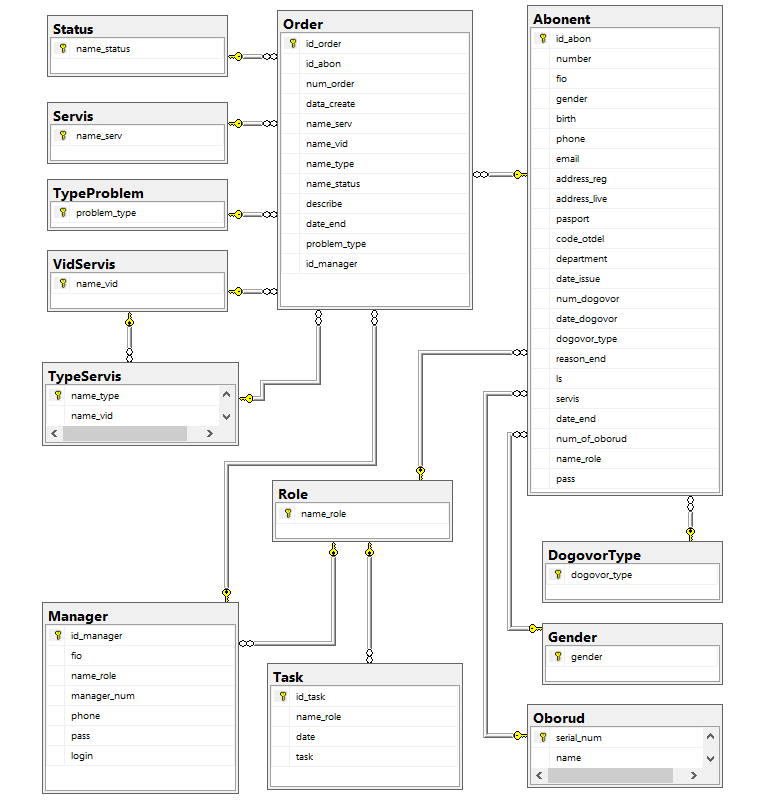


Рисунок.2.1.1 «ERD диаграмма»

ER-диаграмма (Рис. 2.1.1) состоит из 13 сущностей:

1. Сущность «Abonent», содержитвсебеосновныеатрибуты (данные) обабоненте: id\_abon, number, fio, gender, birth, phone, email, address\_reg, address\_live, passport, code\_otdel, department, date\_issue, num\_dogovor, date\_dogovor, dogovor\_type, reason\_end, ls, servis, date\_end, num\_of\_oborud, name\_role, pass.

Ключевым полем данной сущности является атрибут «id\_abon».

Данная сущность взаимосвязана с другими сущностями ER-диаграммы, такими как:Order, Role, DogovorType, Gender и Oborud (оборудование).

1. Сущность «DogovorType», содержит в себе название различных типов договора.

Ключевым полем данной сущности является атрибут «dogovor\_type».

Данная сущность взаимосвязана с сущностью ER-диаграммы, такими как: Abonent.

1. Сущность «Gender».

Ключевым полем данной сущности является атрибут «gender».

Данная сущность взаимосвязана с сущностью ER-диаграммы, такими как: Abonent.

1. Сущность «Oborud», содержит в себе информацию об оборудовании.

Ключевым полем данной сущности является атрибут «serial\_num».

Данная сущность взаимосвязана с сущностью ER-диаграммы, такими как: Abonent.

1. Сущность «Manager», содержит в себе основные атрибуты (данные) об сотрудниках: id\_manager, fio, name\_role, manager\_num, phone, pass, login

Ключевым полем данной сущности является атрибут «id\_manager».

Данная сущность взаимосвязана с другими сущностями ER-диаграммы, такими как: Order, Role.

1. Сущность «Role», содержит в себе названия ролей пользователей системы.

Ключевым полем данной сущности является атрибут «name\_role».

Данная сущность взаимосвязана с другими сущностями ER-диаграммы, такими как: Abonent, Task и Manager.

1. Сущность «Task», содержит в себе основные атрибуты (данные) о задачах сотрудниках: id\_task, name\_role, date, task.

Ключевым полем данной сущности является атрибут «id\_task».

Данная сущность взаимосвязана с сущностьюER-диаграммы, такими как: Task.

1. Сущность «Order», содержит в себе основные атрибуты (данные) о заявках на оказание услуг:id\_order, id\_abon, num\_order, data\_create, name\_serv, name\_vid, name\_type, name\_status, describe, date\_end, problem\_type, id\_manager.

Ключевым полем данной сущности является атрибут «id\_order».

Данная сущность взаимосвязана с другими сущностями ER-диаграммы, такими как: Manager, TypeServis, TypeProblem, Servis, Status.

1. Сущность «Status», содержит в себе название статусов, которые могут принимать заявки.

Ключевым полем данной сущности является атрибут «name\_status».

Данная сущность взаимосвязана с сущностью ER-диаграммы, такими как: Order.

1. Сущность «Servis», содержит в себе название услуг, которые могут оказать абоненту.

Ключевым полем данной сущности является атрибут «name\_servis».

Данная сущность взаимосвязана с сущностью ER-диаграммы, такими как: Order.

1. Сущность «VidServis», содержит в себе название видовуслуг.

Ключевым полем данной сущности является атрибут «name\_vid».

Данная сущность взаимосвязана с сущностью ER-диаграммы, такими как: Order, TypeServis.

1. Сущность «TypeServis», содержит в себе название типов услуг, которые зависят от вида.

Ключевым полем данной сущности является атрибут «name\_type».

Данная сущность взаимосвязана с другими сущностями ER-диаграммы, такими как: Order, VidServis.

1. Сущность «TypeProblem», содержит в себе название типов проблем.

Ключевым полем данной сущности является атрибут «problem\_type».

Данная сущность взаимосвязана с сущностью ER-диаграммы, такими как: Order.

# 2.2 Выбор инструментария

Важной частью при создании и проектировании БД является выбор программы. Мной была выбрана программа «Microsoft SQL Management Studio».

Microsoft SQL Management Studio – это основной, стандартный и полнофункциональный инструмент для работы с Microsoft SQL Server, разработанный компанией Microsoft. С помощью SSMS Вы можете разрабатывать базы данных, выполнять инструкции T-SQL, а также администрировать Microsoft SQL Server.

Функционал:

* подключение к службам SQL Server;
* редактор SQL кода;
* обозреватель решений;
* конструктор баз данных (Диаграммы баз данных);
* конструктор запросов и представлений;
* просмотр свойств объектов;
* мастер создания скриптов;
* присоединение и отсоединение баз данных;
* создание резервных копий баз данных и восстановление баз данных из архива.

Перед началом работы необходимо установить Microsoft SQL Management Studio и Microsoft SQL Server. Это можно сделать, прейдя на официальный сайт (Рис.2.2.4).

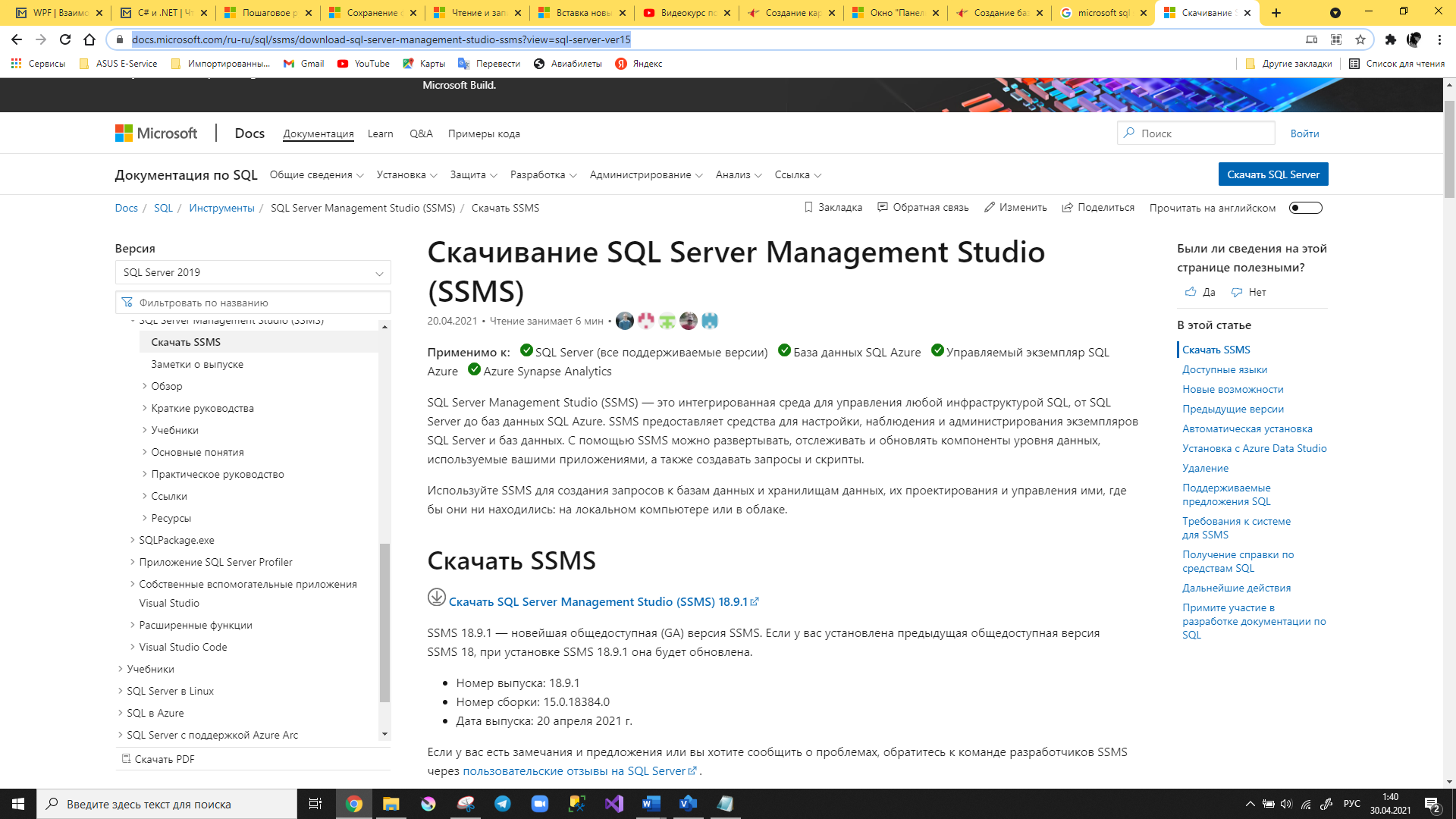


Рис.2.2.4 «Установка SSMS»

# 2.3 Разработка базы данных

База данных (Database) – это организованная структура, предназначенная для хранения, изменения и обработки взаимосвязанной информации, преимущественно больших объемов.

Этапы проектирования БД:

1. Запускаем MSSQLManagementStudio и подключаемся к серверу (Рис.2.3.1).

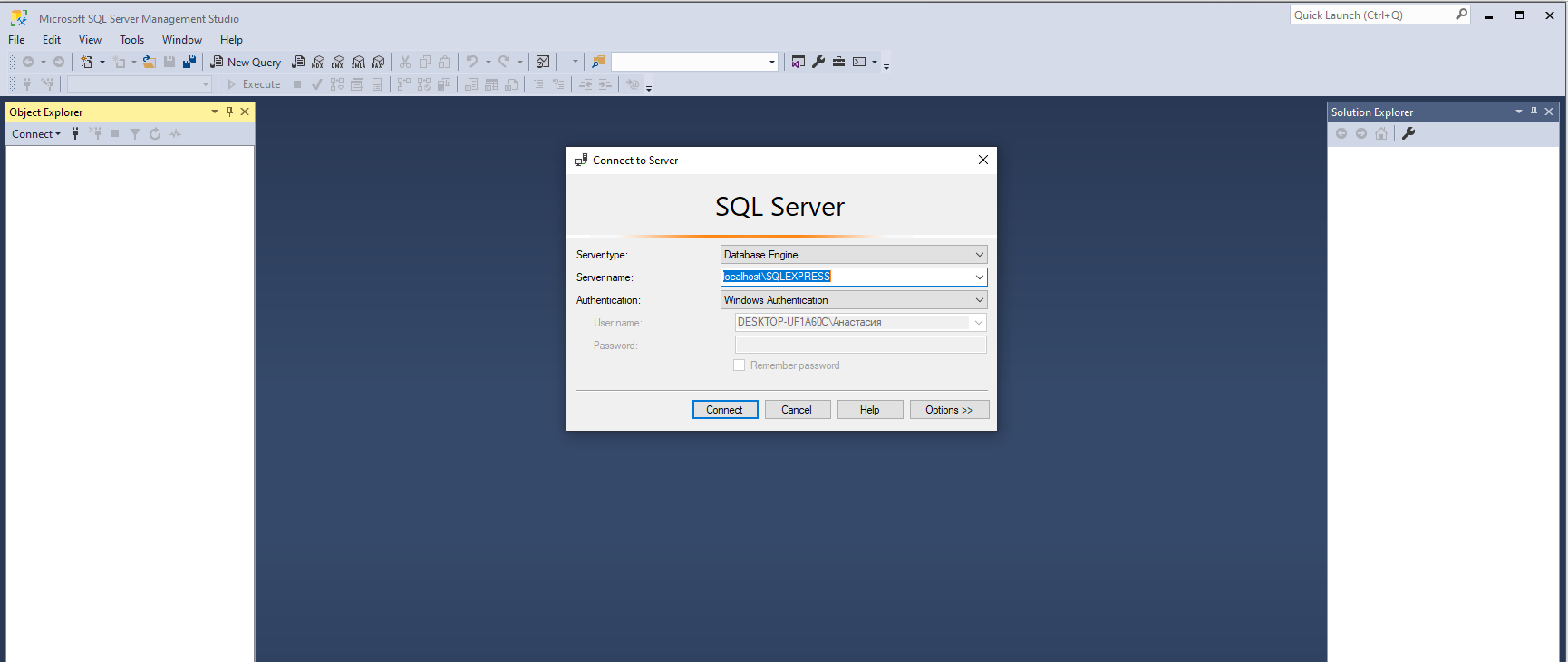


Рисунок.2.3.1 «Подключение к серверу»

1. Создаем новую базу данных(Рис.2.3.2).

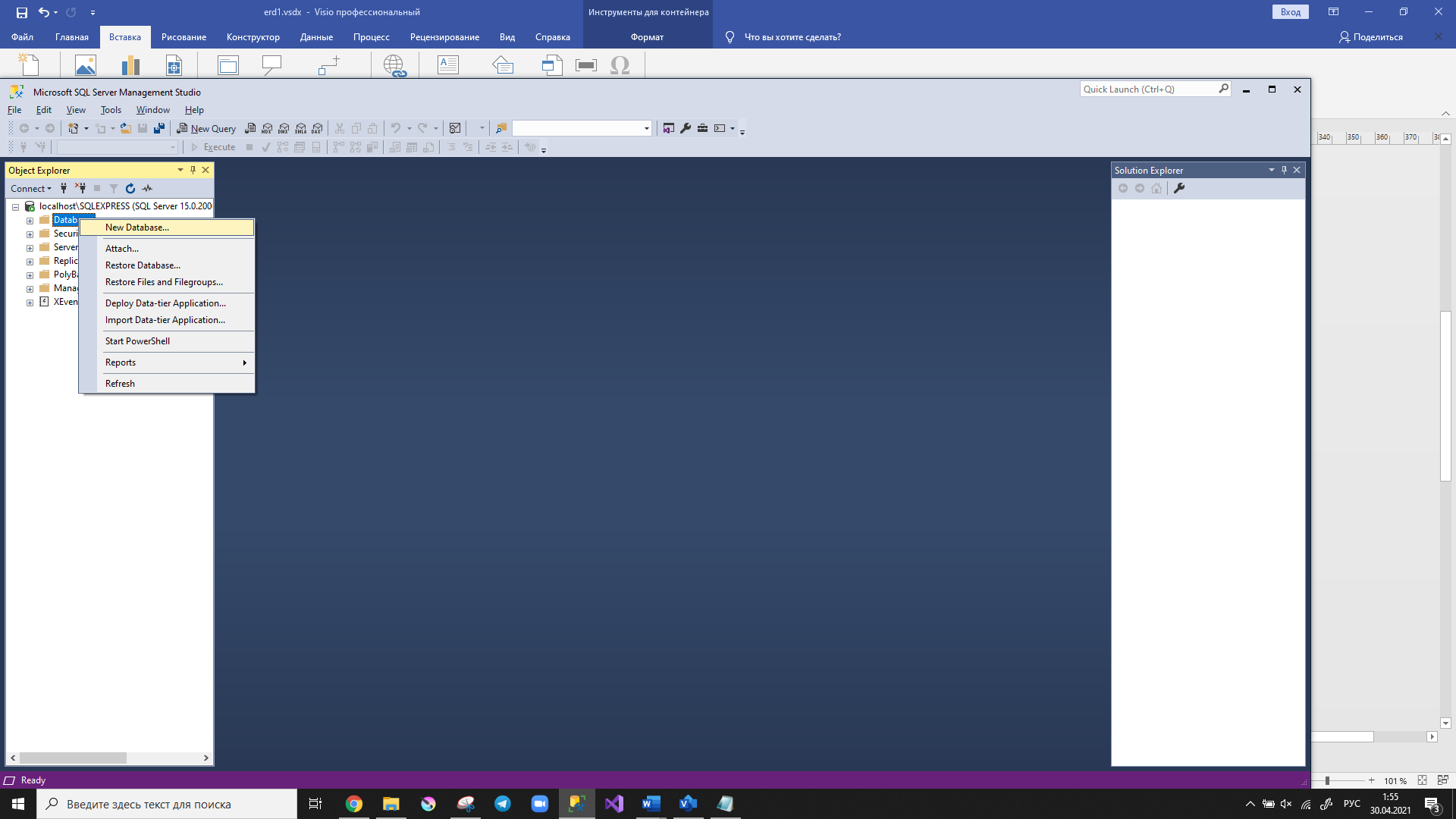


Рисунок.2.3.2 «Создание БД»

1. Создание сущностей (Рис.2.3.3).

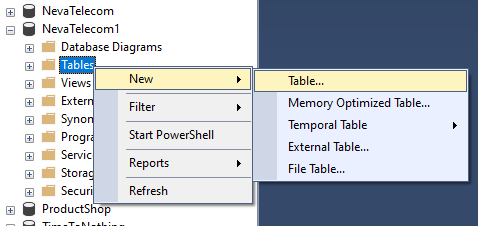


Рисунок.2.3.3 «Создание сущностей»

Согласно построенной ранее ERD диаграмме создаем 13 сущностей. В первом столбце прописывается название атрибута сущности. Во втором столбце необходимо выбрать тип данных атрибутов сущности.

Типы данных:

* Int - представляет целые числа от -2147483648 до 2147483647, занимает 4 байта;
* nvarchar(x) - представляет стоку переменной длины, длина хранимой строки указывается в скобках;
* datetime - хранит даты и время;
* nchar - представляет стоку фиксированной длины.

В третьем столбце есть маркер, отвечающий за обязательность поля. В случае, если мы отметим его галочкой, поле будет необязательным при заполнении данных в таблицу.

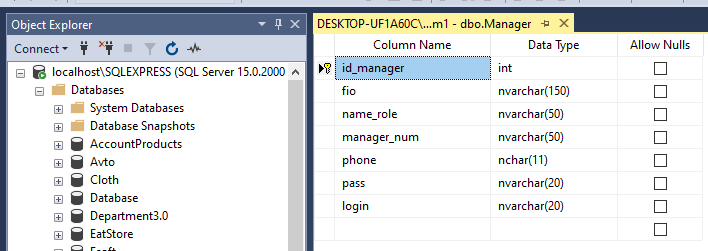


Рисунок.2.3.4 «Таблица Manager»

По примеру создания таблицы «Abonent» (Рис.2.3.4), необходимо создать еще 12 таблиц.

1. Создание диаграммы. Чтобы создать диаграмму, необходимо добавить таблицы (сущности) на поле для построения диаграмм (Рис.2.3.5).

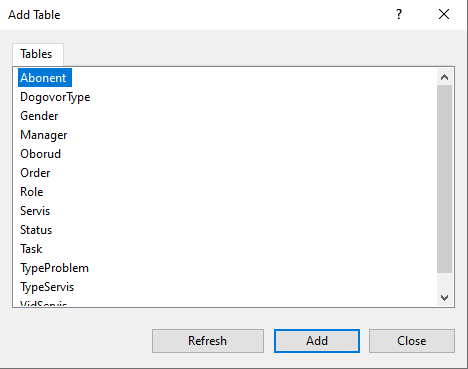
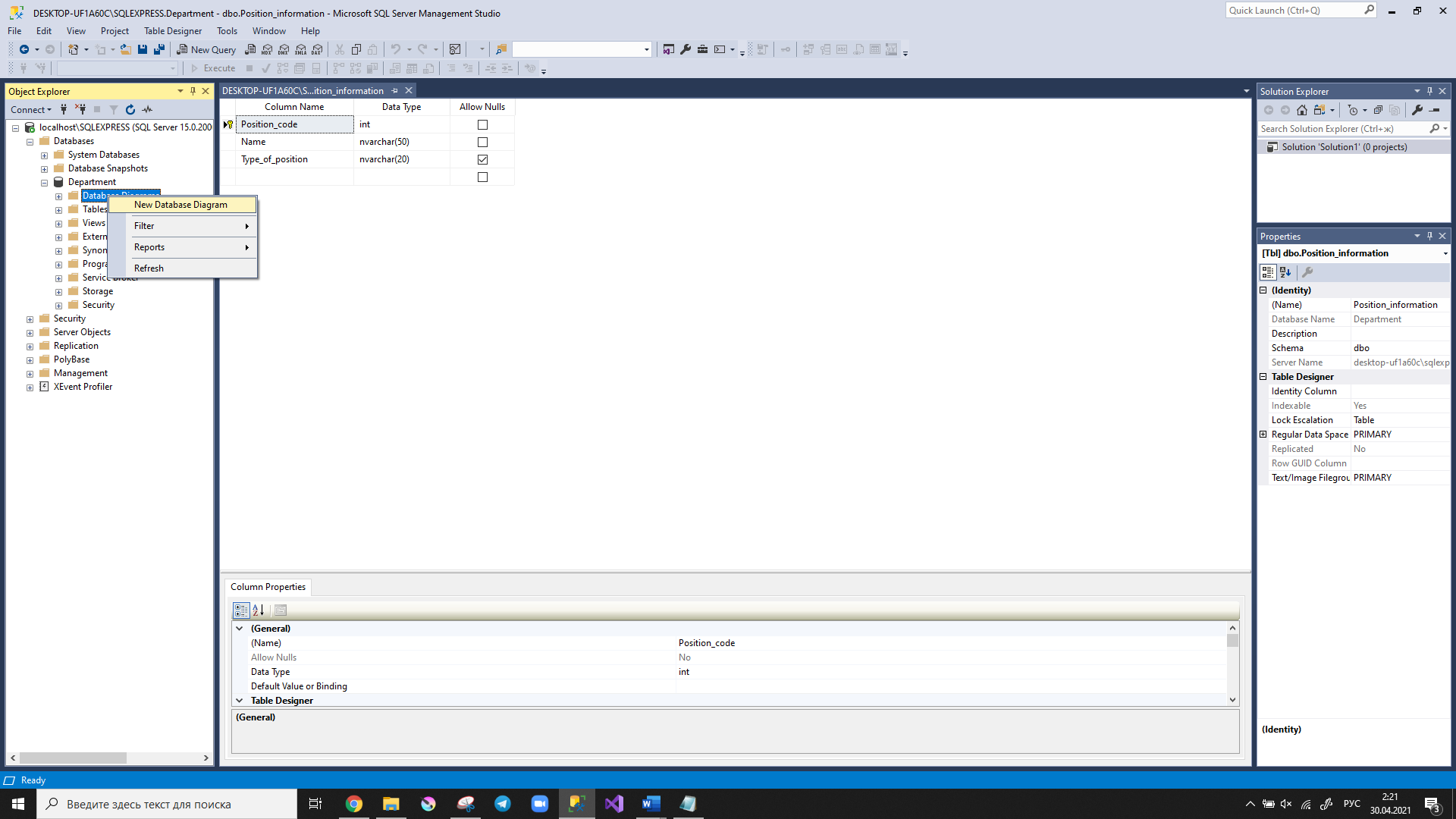


Рисунок.2.3.5 «Добавление таблиц»

1. Установка связей между таблицами.

Чтобы создать в таблицу, с которой необходимо создать соединение, добавляется внешнее ключевое поле, которое такое же по типу данных, как и ключевое поле. Далее от первичного ключа ведется связь к внешнему ключу.

Например: Чтобы связать таблицу «Manager» с таблицей «Role», в таблицу «Manager»был добавлен внешний ключ - name\_role (Рис2.3.6).

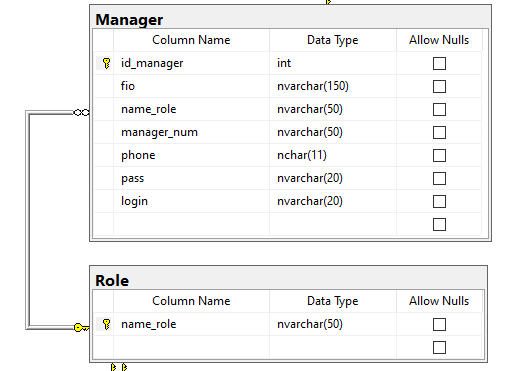


Рисунок.2.3.6 «Создание связи»

По такому принципу создаем связи между таблицами, как на ER-диаграмме. В итоге получаем готовую диаграмму нашей БД (Рис.2.3.7).

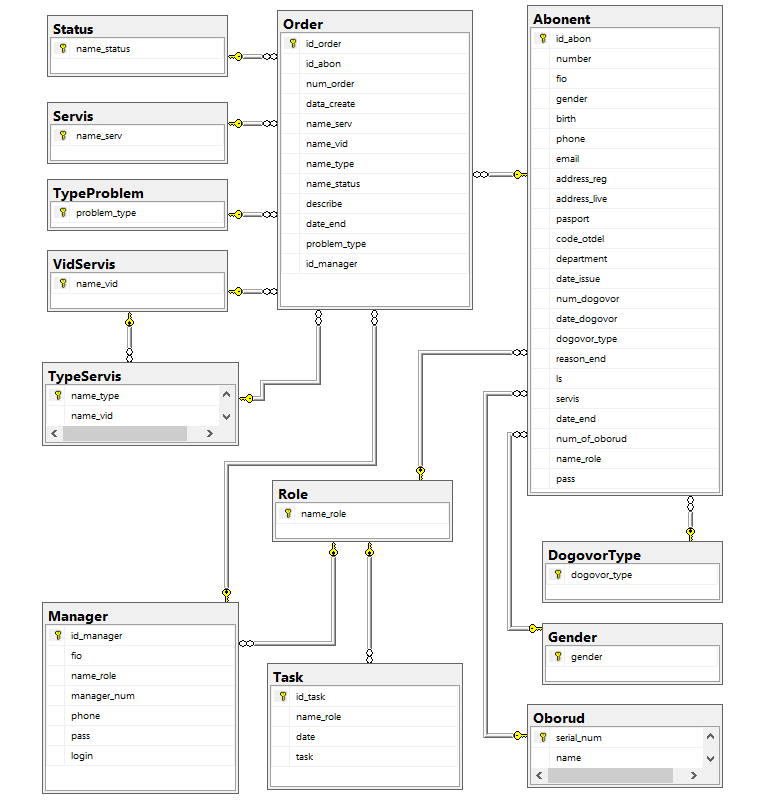


Рисунок.2.3.7 «Готовая диаграмма»

1. Сохранение скрипта базы данных (Рис.2.3.8).

Все созданные базы данных хранятся на сервере. Чтобы перенести базу данных на другой сервер, необходимо правильно ее сохранить. Один из методов - создание скрипта базы данных.

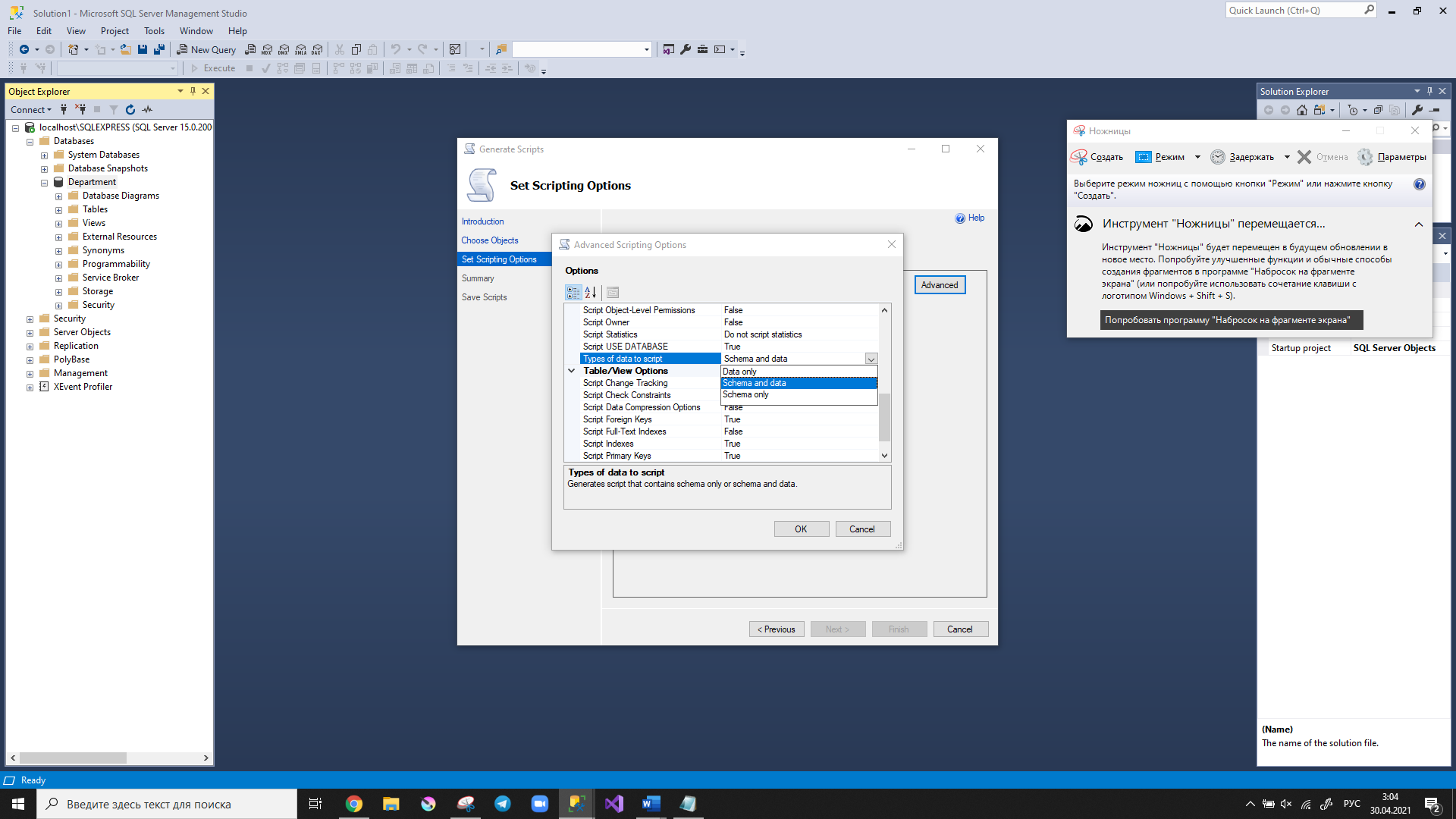
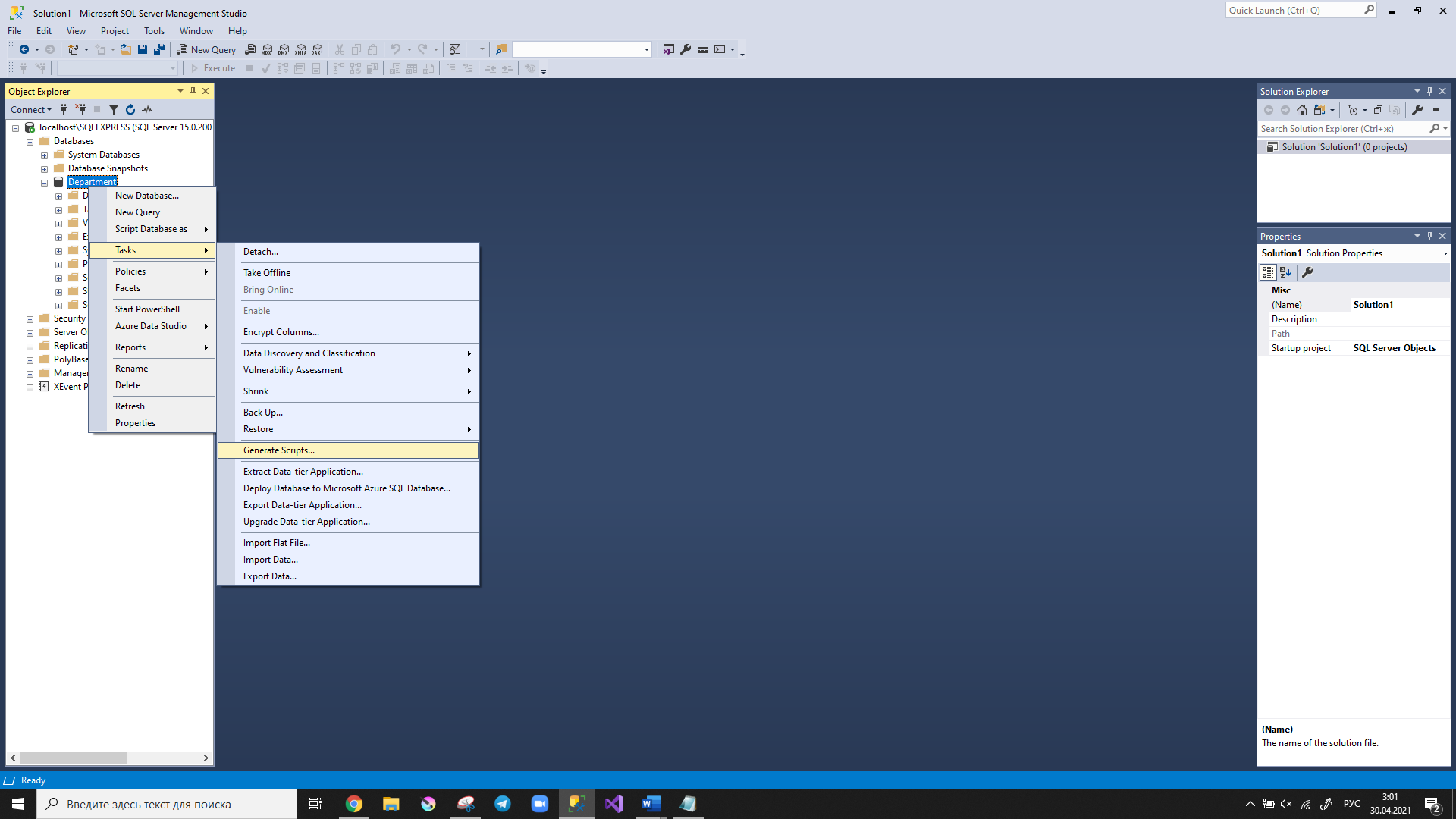


Рисунок.2.3.8 «Сохранение скрипта»

# ГЛАВА 3. РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ

# 3.1 Выбор инструментария

Перед тем как начать разрабатывать приложение необходимо выбрать среду разработки, в которой будет удобно работать. Мной была выбрана VisualStudio (Рис.3.1.1).

Особенности **Visual Studio:**

* **Visual Studio**позволяет разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии WPF.
* В редакторе присутствуют встроенный отладчик, инструменты для работы с Git и средства рефакторинга, навигации по коду, автодополнения типовых конструкций и контекстной подсказки.
* Продукт поддерживает разработку для платформ ASP.NET и Node.js, и считается легковесным решение, которое позволяет обойтись без полной интегрированной среды разработки.

Функционал:

* встроенные инструменты интеграции с GitHub, GIT, а также Visual Studio Team Services для быстрого тестирования, сборки, упаковки и развертывания разных типов приложений;
* поддержка практически всех языков программирования (C++, C#, Python, PHP, JavaScript и других);
* написание кода для конкретной задачи с его последующей интеграцией в проект (с надстройкой или напрямую);
* обширная библиотека шаблонов, готовых фрагментов кода;
* одновременная работа с несколькими проектами (в нескольких окнах);
* интерфейс можно разделить на две панели для сравнения кода;
* функция отладки.

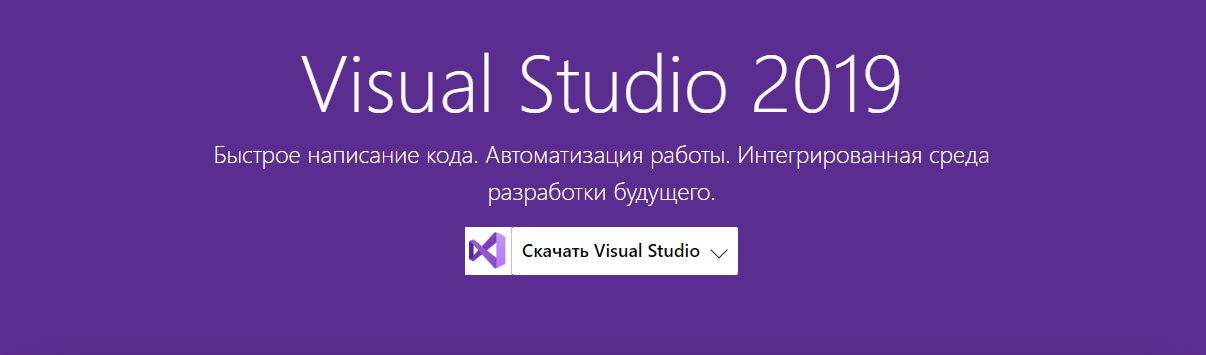


Рисунок.3.1.1 «VisualStudio»

1. Установка VisualStudio и загрузка решений (Рис.3.1.2).

Для установки IDE необходимо зайти на официальный сайт. После запускаVisualStudio, можно выбрать нужные наборы функций. Для разработки приложения,необходимо установить такие решения и библиотекитакие как:

* хранение и обработка данных;
* разработка классических приложений .NET;
* разработка приложений для универсальной платформы Windows;
* разработка классических приложений на C#.

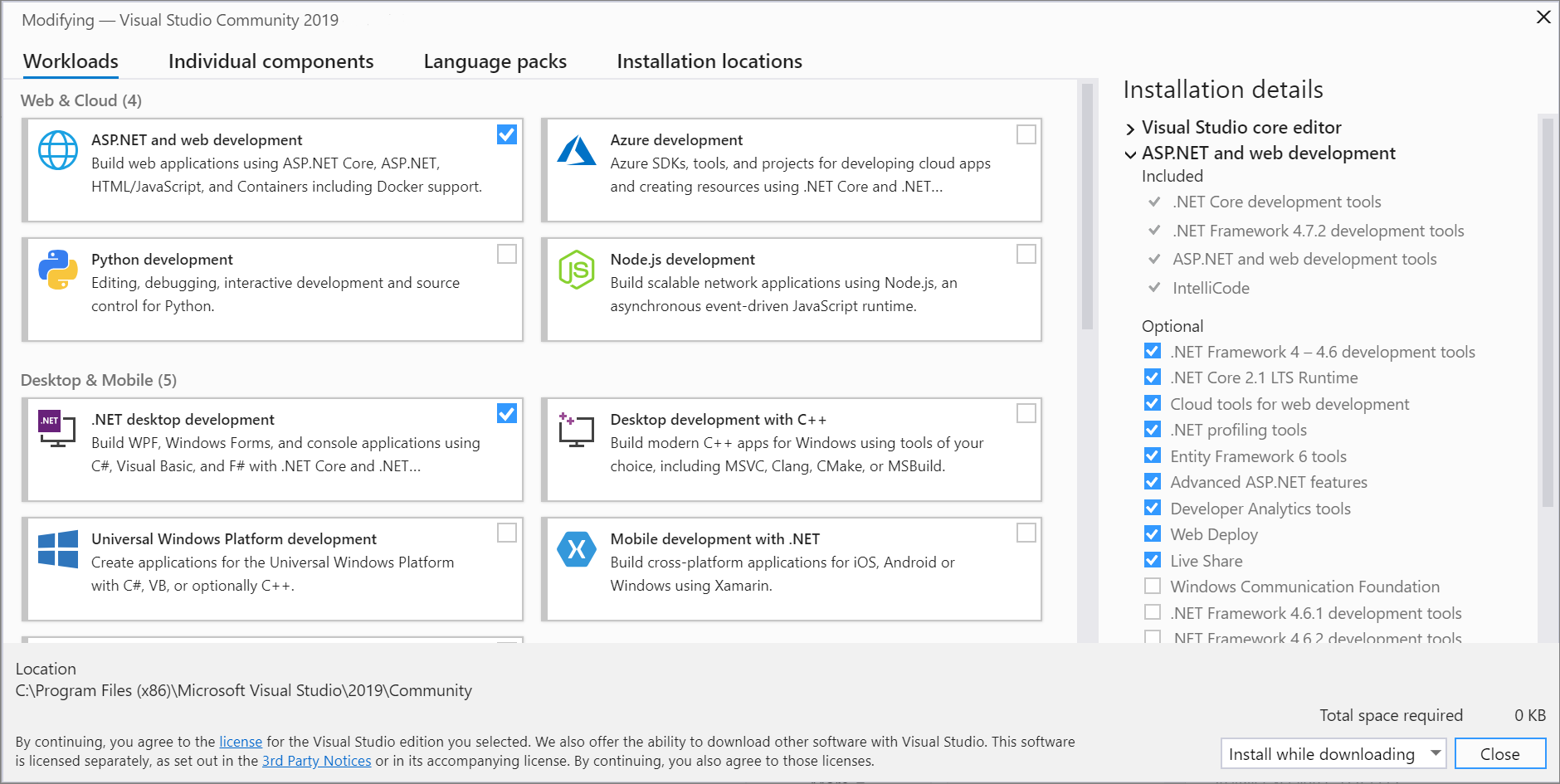


Рисунок.3.1.2 «Установка решений»

1. Начало работы.

* Когда установка Visual Studio завершится, нажмите кнопку Запустить, чтобы приступить к разработке в Visual Studio (Рис.3.1.3).
* На начальном экране выберите «Создать проект» (Рис.3.1.4).
* Выбираем «Приложение WPF (.NET Framework)»

WPF - это система для построения клиентских приложений  с визуально привлекательными возможностями взаимодействия с пользователем, графическая подсистема в составе [.NET Framework](https://ru.wikipedia.org/wiki/.NET_Framework), использующая язык [XAML](https://ru.wikipedia.org/wiki/XAML).

XAML - язык разметки, используемый для инициализации объектов в технологиях на платформе .NET.

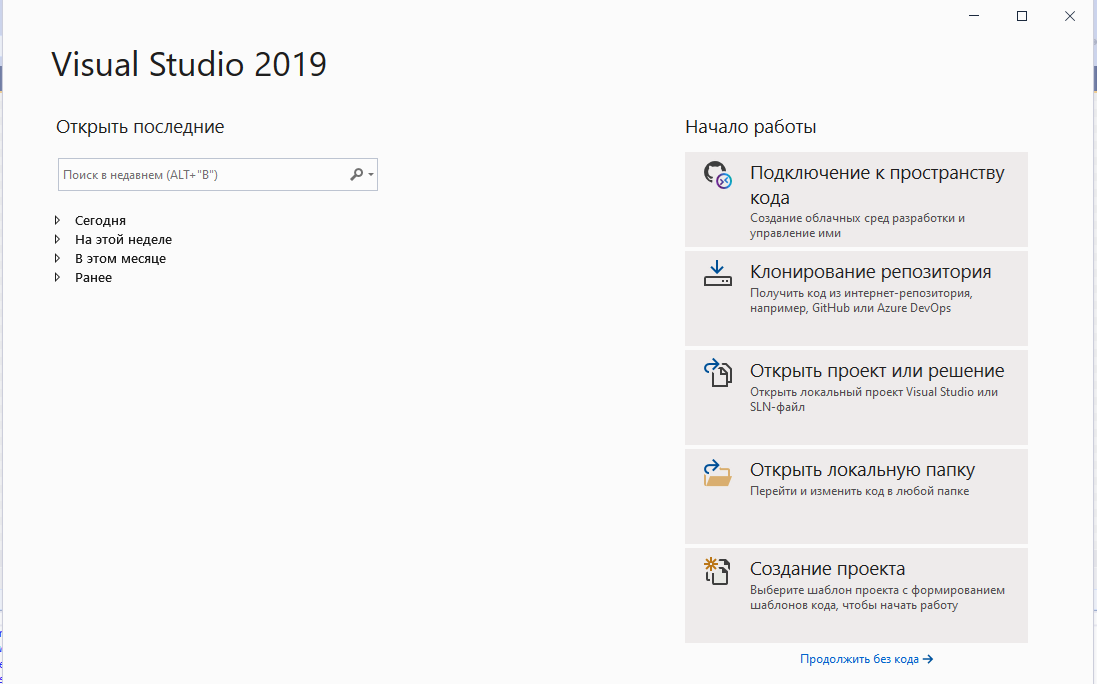


Рисунок.3.1.3 «Запуск программы»

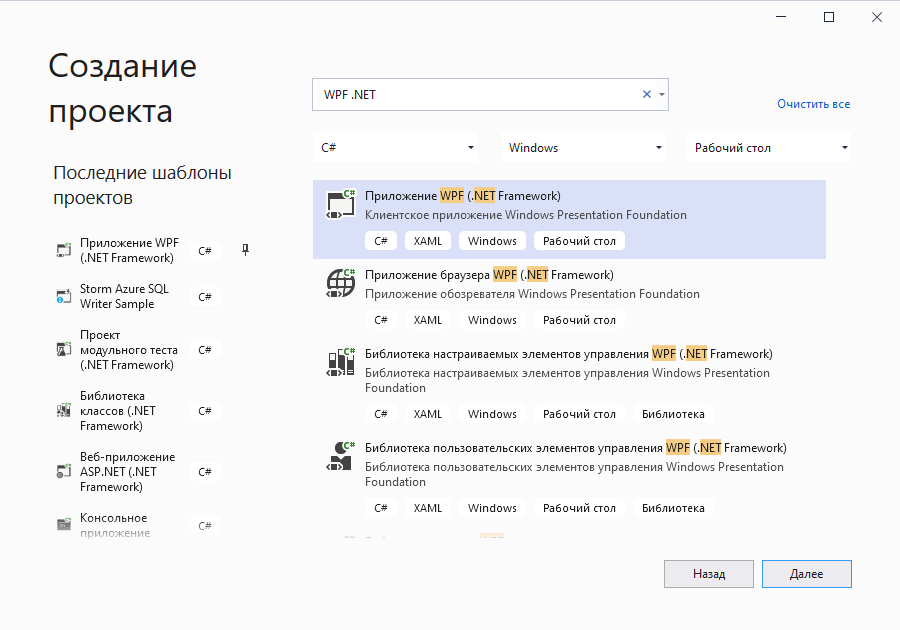


Рис.3.1.6 «Выбор типа приложения»

# 3.2 Разработка интерфейса информационной системы

После создания проекта, открывается главное окно MainWindow.xml – это окно,которое первым открываться при запуске приложения.

В теге <Window/>, для названия приложения используется свойство Title, для логотипа приложения – Icon.Далее в теге <Grid> прописывается тег «Frame» и свой-ваNavigationUIVisibility="Hidden" – для скрытия навигационной панели и Name="MainFame" – в дальнейшем оно понадобиться для реализации отображения страниц.



Рисунок.3.2.1 «Код MainWindow.xml»

Чтобы не прописывать для одинаковых элементов системы, таких как кнопки, поля ввода информации и другие, зададим стили. Они прописываются в файле App.xaml.

Код App.xaml:

<Application.Resources>

<!--Стилидляэлемента "DataGrid"-->

<Style TargetType="DataGrid">

<Setter Property="IsReadOnly" Value="True"/>

<Setter Property="AutoGenerateColumns" Value="False"/>

<Setter Property="FontSize" Value="20"/>

<Setter Property="FontFamily" Value="Candara"/>

<Setter Property="CanUserResizeRows" Value="False"/>

</Style>

<!--Стили для элемента "Button"-->

<Style TargetType="Button">

<Setter Property="Background" Value="#4A67F7"/>

<Setter Property="Foreground" Value="White"/>

<Setter Property="FontSize" Value="22"/>

<Setter Property="BorderBrush" Value="#3CAAFF"/>

<Setter Property="VerticalAlignment" Value="Center"/>

<Setter Property="FontWeight" Value="DemiBold"/>

<Setter Property="FontFamily" Value="Candara"/>

<Setter Property="Padding" Value="10"/>

</Style>

<!--Стилидляэлемента "TextBlock"-->

<Style TargetType="TextBlock">

<Setter Property="VerticalAlignment" Value="Center"/>

<Setter Property="FontWeight" Value="DemiBold"/>

<Setter Property="FontFamily" Value="Candara"/>

</Style>

<!--Стили для элемента "TextBox"-->

<Style TargetType="TextBox">

<Setter Property="Foreground" Value="#FF243238"/>

<Setter Property="FontSize" Value="22"/>

<Setter Property="VerticalAlignment" Value="Center"/>

<Setter Property="FontWeight" Value="DemiBold"/>

<Setter Property="FontFamily" Value="Candara"/>

<Setter Property="Margin" Value="16"/>

<Setter Property="Width" Value="350"/>

</Style>

<!--Стилидляэлемента "PasswordBox"-->

<Style TargetType="PasswordBox">

<Setter Property="Foreground" Value="#FF243238"/>

<Setter Property="FontSize" Value="22"/>

<Setter Property="VerticalAlignment" Value="Center"/>

<Setter Property="FontWeight" Value="DemiBold"/>

<Setter Property="FontFamily" Value="Candara"/>

<Setter Property="Margin" Value="16"/>

<Setter Property="Width" Value="350"/>

</Style>

<!--Стилидляэлемента "ComboBox"-->

<Style TargetType="ComboBox">

<Setter Property="Foreground" Value="#FF243238"/>

<Setter Property="FontSize" Value="22"/>

<Setter Property="VerticalAlignment" Value="Center"/>

<Setter Property="FontWeight" Value="DemiBold"/>

<Setter Property="FontFamily" Value="Candara"/>

<Setter Property="Margin" Value="16"/>

<Setter Property="Width" Value="350"/>

</Style>

<!--Стили для элемента "DatePicker"-->

<Style TargetType="DatePicker">

<Setter Property="Foreground" Value="#FF243238"/>

<Setter Property="FontSize" Value="22"/>

<Setter Property="VerticalAlignment" Value="Center"/>

<Setter Property="FontWeight" Value="DemiBold"/>

<Setter Property="FontFamily" Value="Candara"/>

<Setter Property="Margin" Value="16"/>

<Setter Property="Width" Value="350"/>

</Style>

</Application.Resources>

Теперь создадим все необходимые страницы приложения (Рис.3.2.2).

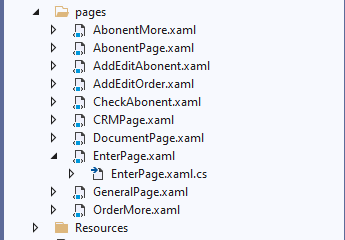


Рисунок.3.2.2 «Список страниц»

1. EnterPage.xaml – страница входа (Рис.3.2.3).

Данная страница служит для отображения входа в систему. Состоит из поля ввода логина, пароля, проверочного кода, кнопок генерации кода, отображения пароля и входа.

Код xaml:

<Grid VerticalAlignment="Center" HorizontalAlignment="Center">

<!--Разметка на стобцы и строки-->

<Grid.RowDefinitions>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

</Grid.RowDefinitions>

<!--Элемент предназначен для вывода текстовой информации.-->

<TextBlock Text="Авторизация" Foreground="#4A67F7" Grid.Row="0" HorizontalAlignment="Center" FontSize="42" FontWeight="Bold" Margin="16"/>

<Grid Grid.Row="1">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="auto"/>

<ColumnDefinition Width="auto"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

</Grid.RowDefinitions>

<!--блок ввода логина пользователя-->

<Image Grid.Row="0" Grid.Column="0" Source="/NevaTelecomv\_1.0;component/Resources/Login.png" Margin="0 10"/>

<TextBox Name="log" Grid.Row="0" Grid.Column="1" HorizontalAlignment="Left" VerticalAlignment="Center" MaxLength="20"/>

<!--блоквводапароляпользователя-->

<Image Grid.Row="1" Grid.Column="0" Source="/NevaTelecomv\_1.0;component/Resources/password.png" Margin="0 10"/>

<StackPanel Orientation="Horizontal" Grid.Row="1" Grid.Column="1" Grid.ColumnSpan="2">

<Grid>

<TextBox Name="text\_pass" HorizontalAlignment="Left" Visibility="Hidden" MaxLength="20" IsEnabled="False"/>

<PasswordBox Name="pass" HorizontalAlignment="Left" Visibility="Visible" MaxLength="20" IsEnabled="False"/>

</Grid>

<!--кнопкапоказатьпароль-->

<Button Name="Show" Click="Show\_Click" Grid.Row="1" Grid.Column="1" Width="40" Height="40" IsEnabled="False">

<Button.Background>

<ImageBrush ImageSource="/NevaTelecomv\_1.0;component/Resources/view.png"/>

</Button.Background>

<Button.BorderBrush>

<ImageBrush/>

</Button.BorderBrush>

</Button>

</StackPanel>

<!--блоквводасгенерированногокода-->

<Image Grid.Row="2" Grid.Column="0" Source="/NevaTelecomv\_1.0;component/Resources/key.png" Margin="0 10"/>

<StackPanel Orientation="Horizontal" Grid.Row="2" Grid.Column="1" Grid.ColumnSpan="2">

<TextBox Name="code" IsEnabled="False" Grid.Row="2" Grid.Column="1" HorizontalAlignment="Left" VerticalAlignment="Center" MaxLength="20"/>

<!--кнопкагенерациикода-->

<Button Name="btnCode" Click="btnCode\_Click" IsEnabled="False" Grid.Row="2" Grid.Column="1" Width="40" Height="40">

<Button.Background>

<ImageBrush ImageSource="/NevaTelecomv\_1.0;component/Resources/refresh.png"/>

</Button.Background>

<Button.BorderBrush>

<ImageBrush/>

</Button.BorderBrush>

</Button>

</StackPanel>

</Grid>

<!--кнопкавходавсистему-->

<Button IsDefault="True" Width="200" Name="enter" Click="enter\_Click" Content="Вход" Grid.Row="3" Padding="5" Margin="0 10"/>

</Grid>

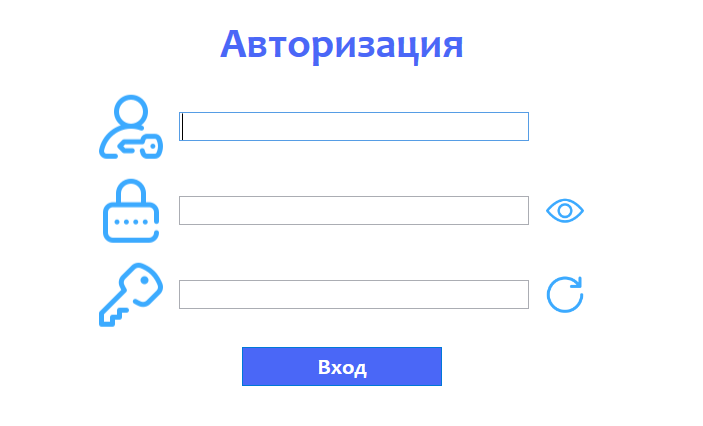


Рисунок.3.2.3 «Отображение EnterPage»

1. GeneralPage.xaml – главная страница (Рис.3.2.4).

Данная страницапредставляет основную страницу с отображением логотипа компании, меню приложения и списком событий для сотрудников, а также здесь будут отображаться выбранные остальные страницы.

Код xaml:

<Grid>

<!--Разметка страницы на стобцы и строки-->

<Grid.RowDefinitions>

<RowDefinition Height="150"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="150"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<Grid Grid.Row="0" Grid.Column="0" Background="#B5DBFF"/>

<!--Элемент предназначен для вывода изображения.-->

<Image Grid.Row="0" Grid.Column="0" Source="/NevaTelecomv\_1.0;component/Resources/logoWithBack.png" Margin="0 10"/>

<!--Блокменюскнопками-->

<StackPanel Background="#B5DBFF" Grid.Column="0" Grid.Row="1">

<Button Content="Абонетны" Margin="5 0 5 10" Name="Btn\_abonent" Click="Btn\_abonent\_Click"/>

<Button Content="CRM" Margin="5 0 5 10" Name="Btn\_CRM" Click="Btn\_CRM\_Click"/>

<Button Content="Документы" Margin="5 0 5 10" Name="Btn\_doc" Click="Btn\_doc\_Click"/>

<Button Content="Выход" Margin="5 0 5 10" Name="Btn\_exit" Click="Btn\_exit\_Click"/>

</StackPanel>

<!--Блок для отображения названия выбранной вкладки-->

<TextBlock Grid.Row="0" Grid.Column="1" Margin="10" Name="Title" VerticalAlignment="Center" FontSize="54" Foreground="#FF4A67F7"/>

<Grid Grid.Column="1" Grid.Row="1">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="auto"/>

</Grid.ColumnDefinitions>

<!--Тег для навигации по страницам-->

<Frame Name="MainFrame1" NavigationUIVisibility="Hidden" Grid.Row="1" Grid.Column="0"/>

<!--блок для отображения событий-->

<GridGrid.Column="1">

<Grid.RowDefinitions>

<RowDefinition Height="auto"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Grid Grid.Row="0" Background="#4A67F7">

<TextBlock Text="События" HorizontalAlignment="Center" Padding="5" FontSize="18" Foreground="White"/>

</Grid>

<!--ListView предназначен для отображения представлений одних и тех же данных.-->

<ListView Name="LvTask" Grid.Row="1" Width="auto">

<ListView.ItemTemplate>

<DataTemplate>

<Border BorderBrush="#4A67F7" BorderThickness="2">

<StackPanel HorizontalAlignment="Center">

<TextBlock Text="{Binding date}" Margin="5" FontSize="18"/>

<TextBlock Text="{Binding task1}" Margin="5" TextWrapping="Wrap" FontSize="18"/>

</StackPanel>

</Border>

</DataTemplate>

</ListView.ItemTemplate>

</ListView>

</Grid>

</Grid>

</Grid>

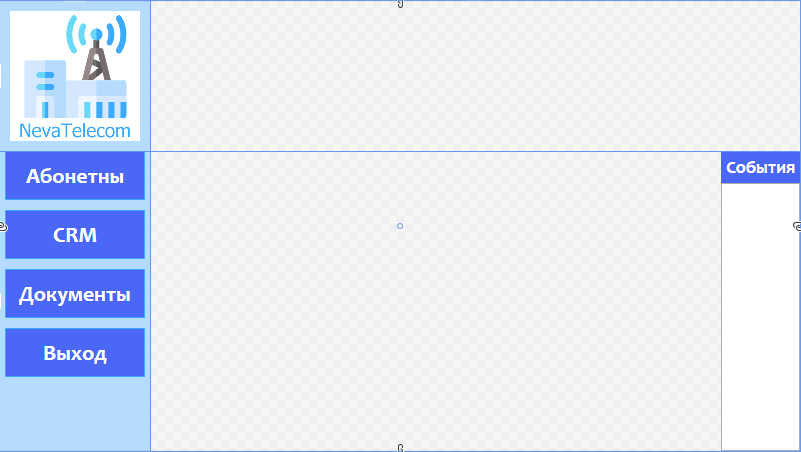


Рисунок.3.2.4 «Отображение GeneralPage»

1. AbonentPage.xaml – страница со списком абонентов компании (Рис.3.2.5).

Страница предназначена, для отображения списка всех абонентов, которые пользуются услугами компании. Так же присутствует возможность просмотра подробной информации, редактирование абонента и поиск по ФИО и лицевому счету.

В разметке используется элемент <DataGrid/>, который представляет собой элемент управления отображением данных, который извлекает информацию из коллекции объектов и визуализирует ее в сетке со строками и ячейками. Каждая строка соответствует отдельному объекту, а каждый столбец — свойству в этом объекте.

Код xaml:

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="auto"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="auto"/>

</Grid.RowDefinitions>

<Grid Grid.Row="0" Background="#7FB5DBFF">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="auto"/>

<ColumnDefinition Width="auto"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

</Grid.RowDefinitions>

<!--блок для поиска данных по фамилии абонента-->

<TextBlock Text="Ф.И.Оабонента" FontSize="18" Grid.Row="0" Grid.Column="0" Margin=" 5 10 5 10" />

<TextBox Name="Tb\_fio" SelectionChanged="Tb\_fio\_SelectionChanged" Grid.Row="0" Grid.Column="1" Margin=" 5 10 5 10"/>

<!--блок для поиска данных по лицевому счету абонента-->

<TextBlock Text="Лицевойсчет" FontSize="18" Grid.Row="1" Grid.Column="0" Margin=" 5 5 5 10"/>

<TextBox Name="Tb\_ls" SelectionChanged="Tb\_ls\_SelectionChanged" Grid.Row="1" Grid.Column="1" Margin=" 5 5 5 10"/>

</Grid>

<!--таблица с отображением данных об абонентах-->

<DataGrid Grid.Row="1" x:Name="DGridAbonent" AutoGenerateColumns="False" IsReadOnly="True" CanUserResizeRows="False" FontSize="18" >

<DataGrid.Columns>

<DataGridTextColumn Header="№ абонента" Binding="{Binding number}"/>

<DataGridTextColumn Header="ФИО" Binding="{Binding fio}"/>

<DataGridTextColumn Header="№ договора" Binding="{Binding num\_dogovor}"/>

<DataGridTextColumn Header="ЛС" Binding="{Binding ls}"/>

<DataGridTextColumn Header="Услуги" Binding="{Binding servis}"/>

<!--кнопка подороднее для всех записей-->

<DataGridTemplateColumn>

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<Button Name="more" Click="more\_Click" Content="Подробнее"/>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

<!--кнопкаредактироватьдлявсехзаписей-->

<DataGridTemplateColumn>

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<Button Name="edit" Click="edit\_Click" Content="Редактировать"/>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

</DataGrid.Columns>

</DataGrid>

<Grid Grid.Row="2" Margin="0 15">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="auto"/>

<ColumnDefinition Width="auto"/>

<ColumnDefinition Width="auto"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<!--блокдляфильтрацииданных-->

<RadioButton Name="BtnActiv" Content="Активные" Checked="BtnActiv\_Checked" FontSize="18" Grid.Column="0" Margin="10 0 50 0" VerticalAlignment="Center"/>

<RadioButton Name="BtnUnActiv" Content="Неактивные" Checked="BtnUnActiv\_Checked" FontSize="18" Grid.Column="1" Margin="10 0 50 0" VerticalAlignment="Center"/>

<RadioButton Name="BtnAll" Content="Все" Checked="BtnAll\_Checked" FontSize="18" Grid.Column="2" VerticalAlignment="Center"/>

<!--кнопкадобавленияабонента-->

<Button Content="Добавить" Name="Btn\_Add\_exit" Click="Btn\_Add\_exit\_Click" HorizontalAlignment="Right" Grid.Column="3" Padding="5" Margin="10 0"/>

</Grid>

</Grid>

Подробнее об использованных элементах:

* <DataGridTemplateColumn.CellTemplate/> – получает или задает шаблон, используемый для отображения содержимого ячейки, находящейся не в режиме редактирования.
* Binding– получение или установка привязки, которая связывает столбец со свойством источника данных. В нашем случае БД.
* <[DataTemplate](https://docs.microsoft.com/ru-ru/dotnet/api/system.windows.datatemplate?view=net-5.0)/> – шаблон, который нужно использовать для отображения содержимого ячейки. В нашем случае для кнопки редактирования информации.

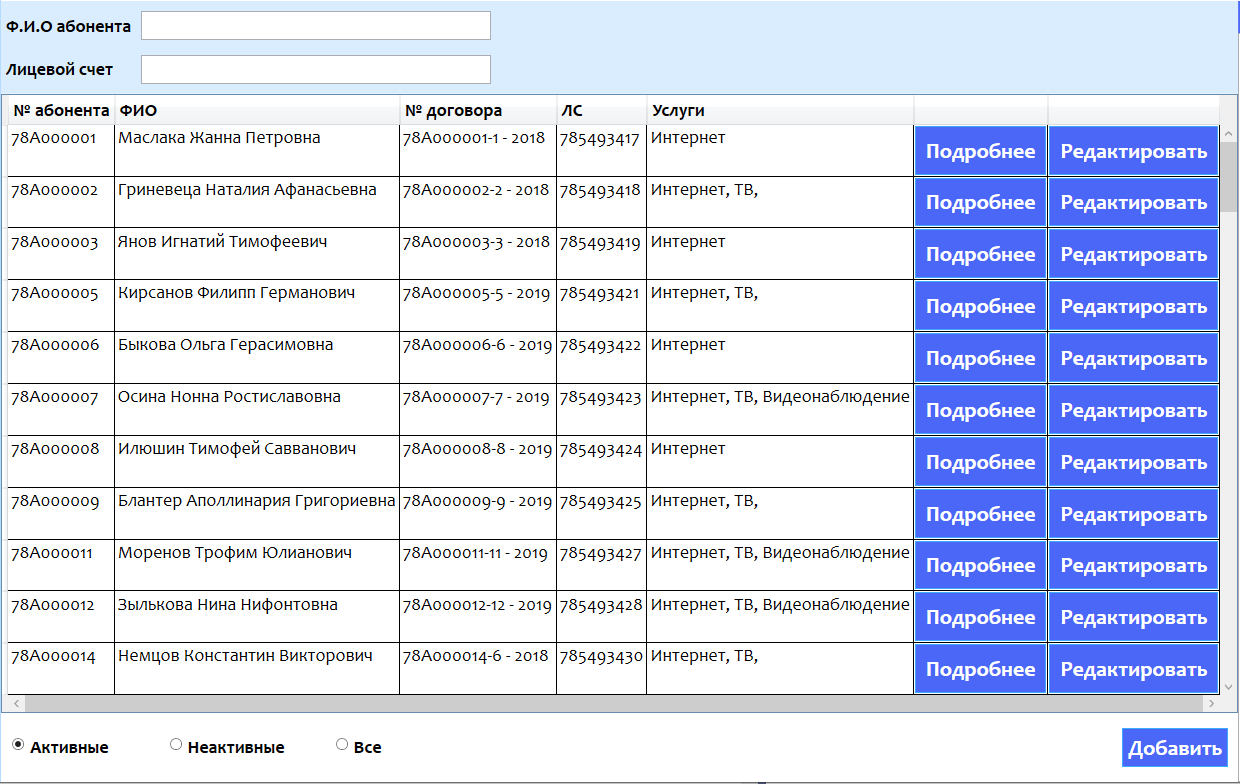


Рисунок.3.2.5 «Отображение AbonentPage»

1. AbonentMore.xaml – страница подробной информации об абоненте (Рис.3.2.6).

Данная страница предназначена для подробного просмотра информации обабоненте.

Код xaml:

<ScrollViewer>

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

</Grid.RowDefinitions>

<StackPanel Orientation="Horizontal" Grid.Row="0" Margin="10">

<TextBlock Text="№ абонента: " FontSize="20"/>

<TextBlock Text="{Binding number}" FontSize="20" FontWeight="Normal"/>

</StackPanel>

<StackPanel Orientation="Horizontal" Grid.Row="1" Margin="10">

<TextBlock Text="ФИОабонента: " FontSize="20"/>

<TextBlock Text="{Binding fio}" FontSize="20" FontWeight="Normal"/>

</StackPanel>

<StackPanel Orientation="Horizontal" Grid.Row="2" Margin="10">

<TextBlock Text="Серияиномер: " FontSize="20"/>

<TextBlock Text="{Binding pasport}" FontSize="20" FontWeight="Normal"/>

</StackPanel>

<StackPanel Orientation="Horizontal" Grid.Row="3" Margin="10">

<TextBlock Text="Датавыдачи: " FontSize="20"/>

<TextBlock Text="{Binding date\_issue, StringFormat=d}" FontSize="20" FontWeight="Normal"/>

</StackPanel>

<StackPanel Orientation="Horizontal" Grid.Row="4" Margin="10">

<TextBlock Text="Кемвыдан: " FontSize="20"/>

<TextBlock Text="{Binding department}" FontSize="20" FontWeight="Normal"/>

</StackPanel>

<StackPanel Orientation="Horizontal" Grid.Row="5" Margin="10">

<TextBlock Text="№ договора: " FontSize="20"/>

<TextBlock Text="{Binding num\_dogovor}" FontSize="20" FontWeight="Normal"/>

</StackPanel>

<StackPanel Orientation="Horizontal" Grid.Row="6" Margin="10">

<TextBlock Text="Дата заключения договора: " FontSize="20"/>

<TextBlock Text="{Binding date\_dogovor, StringFormat=d}" FontSize="20" FontWeight="Normal"/>

</StackPanel>

<StackPanel Orientation="Horizontal" Grid.Row="7" Margin="10">

<TextBlock Text="Типдоговора: " FontSize="20"/>

<TextBlock Text="{Binding dogovor\_type}" FontSize="20" FontWeight="Normal"/>

</StackPanel>

<StackPanel Orientation="Horizontal" Grid.Row="8" Margin="10">

<TextBlock Text="Лицевойсчет: " FontSize="20"/>

<TextBlock Text="{Binding ls}" FontSize="20" FontWeight="Normal"/>

</StackPanel>

<StackPanel Orientation="Horizontal" Grid.Row="9" Margin="10">

<TextBlock Text="Адрес: " FontSize="20"/>

<TextBlock Text="{Binding address\_reg}" FontSize="20" FontWeight="Normal"/>

</StackPanel>

<StackPanel Orientation="Horizontal" Grid.Row="10" Margin="10">

<TextBlock Text="Перечень подключенных услуг: " FontSize="20"/>

<TextBlock Text="{Binding servis}" FontSize="20" FontWeight="Normal"/>

</StackPanel>

<StackPanel Orientation="Horizontal" Grid.Row="11" Margin="10">

<TextBlock Text="Данные об оборудовании: " FontSize="20"/>

<TextBlock Text="{Binding num\_of\_oborud}" FontSize="20" FontWeight="Normal"/>

</StackPanel>

<StackPanel Orientation="Horizontal" Grid.Row="12" Margin="10" Name="DogovorDateEnd">

<TextBlock Text="Дата расторжения договора: " FontSize="20"/>

<TextBlock Text="{Binding date\_end, StringFormat=d}" FontSize="20" FontWeight="Normal"/>

</StackPanel>

<StackPanel Orientation="Horizontal" Grid.Row="13" Margin="10" Name="DogovorReasonEnd">

<TextBlock Text="Причина расторжения договора: " FontSize="20"/>

<TextBlock Text="{Binding reason\_end}" FontSize="20" FontWeight="Normal"/>

</StackPanel>

</Grid>

</ScrollViewer>

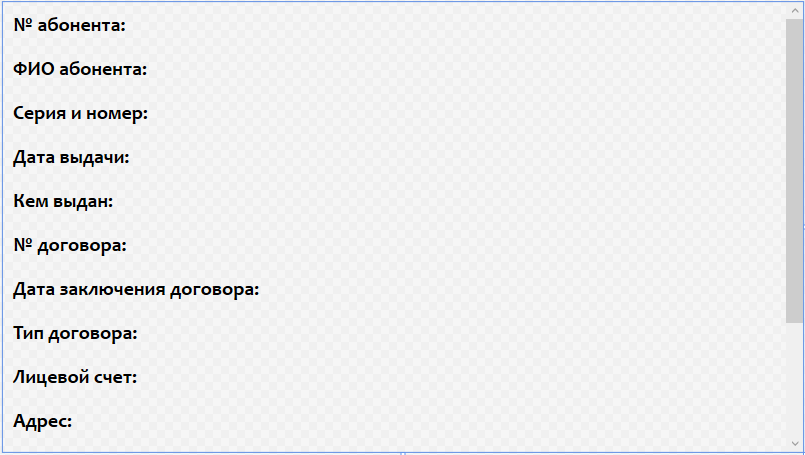


Рисунок.3.2.6 «Отображение AbonentMore»

1. AddEditAbonent.xaml – страница для добавления или редактированияинформации обабоненте (Рис.3.2.7).

Код xaml:

<ScrollViewer>

<Grid Margin="10 0">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="auto"/>

<ColumnDefinition Width="auto"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

</Grid.RowDefinitions>

<TextBlock Text="№ абонента: " FontSize="20" Grid.Row="0" Grid.Column="0"/>

<TextBox Text="{Binding number}" HorizontalAlignment="Left" MaxLength="50" FontSize="20" FontWeight="Normal" Grid.Row="0" Grid.Column="1" />

<TextBlock Text="ФИОабонента: " FontSize="20" Grid.Row="1" Grid.Column="0"/>

<TextBox Text="{Binding fio}" HorizontalAlignment="Left" MaxLength="150" FontSize="20" FontWeight="Normal" Grid.Row="1" Grid.Column="1"/>

<TextBlock Text="Пол: " FontSize="20" Grid.Row="2" Grid.Column="0"/>

<ComboBox Name="Cb\_gender" HorizontalAlignment="Left" SelectedItem="{Binding Gender1}" DisplayMemberPath="gender1" FontSize="20" FontWeight="Normal" Grid.Row="2" Grid.Column="1"/>

<TextBlock Text="Датарождения: " FontSize="20" Grid.Row="3" Grid.Column="0"/>

<DatePicker HorizontalAlignment="Left" SelectedDate="{Binding birth}" Name="birth" FontSize="20" FontWeight="Normal" Grid.Row="3" Grid.Column="1"/>

<TextBlock Text="Телефон: " FontSize="20" Grid.Row="4" Grid.Column="0"/>

<TextBox HorizontalAlignment="Left" Text="{Binding phone}" FontSize="20" FontWeight="Normal" Grid.Row="4" Grid.Column="1" MaxLength="11"/>

<TextBlock Text="Email: " FontSize="20" Grid.Row="5" Grid.Column="0"/>

<TextBox HorizontalAlignment="Left" Text="{Binding email}" FontSize="20" FontWeight="Normal" Grid.Row="5" Grid.Column="1" MaxLength="50"/>

<TextBlock Text="Адресрегистрации: " FontSize="20" Grid.Row="6" Grid.Column="0"/>

<TextBox HorizontalAlignment="Left" Text="{Binding address\_reg}" FontSize="20" FontWeight="Normal" Grid.Row="6" Grid.Column="1" MaxLength="150"/>

<TextBlock Text="Адреспроживания: " FontSize="20" Grid.Row="7" Grid.Column="0"/>

<TextBox HorizontalAlignment="Left" Text="{Binding address\_live}" FontSize="20" FontWeight="Normal" Grid.Row="7" Grid.Column="1" MaxLength="150"/>

<TextBlock Text="Серияиномер: " FontSize="20" Grid.Row="8" Grid.Column="0"/>

<TextBox HorizontalAlignment="Left" Text="{Binding pasport}" FontSize="20" FontWeight="Normal" Grid.Row="8" Grid.Column="1" MaxLength="11"/>

<TextBlock Text="Кодподразделения: " FontSize="20" Grid.Row="9" Grid.Column="0"/>

<TextBox HorizontalAlignment="Left" Text="{Binding code\_otdel}" FontSize="20" FontWeight="Normal" Grid.Row="9" Grid.Column="1" MaxLength="7"/>

<TextBlock Text="Кемвыдан: " FontSize="20" Grid.Row="10" Grid.Column="0"/>

<TextBox HorizontalAlignment="Left" Text="{Binding department}" FontSize="20" FontWeight="Normal" Grid.Row="10" Grid.Column="1" MaxLength="150"/>

<TextBlock Text="Датавыдачи: " FontSize="20" Grid.Row="11" Grid.Column="0"/>

<DatePicker HorizontalAlignment="Left" SelectedDate="{Binding date\_issue}" Name="date\_issue" FontSize="20" FontWeight="Normal" Grid.Row="11" Grid.Column="1"/>

<TextBlock Text="№ договора: " FontSize="20" Grid.Row="12" Grid.Column="0"/>

<StackPanel Orientation="Horizontal" Grid.Row="12" Grid.Column="1">

<TextBox Text="{Binding num\_dogovor}" Name="dogovor" FontSize="20" FontWeight="Normal" MaxLength="50" IsReadOnly="True"/>

<Button Name="gen\_num" Click="gen\_num\_Click" Margin="0 10" Padding="0" Width="30" Height="30" BorderBrush="{x:Null}">

<Button.Background>

<ImageBrush ImageSource="/NevaTelecomv\_1.0;component/Resources/refresh.png"/>

</Button.Background>

</Button>

</StackPanel>

<TextBlock Text="Датадоговора: " FontSize="20" Grid.Row="13" Grid.Column="0"/>

<DatePicker HorizontalAlignment="Left" SelectedDate="{Binding date\_dogovor}" Name="date\_dogovor" FontSize="20" FontWeight="Normal" Grid.Row="13" Grid.Column="1" IsEnabled="False"/>

<TextBlock Text="Типдоговора: " FontSize="20" Grid.Row="14" Grid.Column="0"/>

<ComboBox HorizontalAlignment="Left" Name="Cb\_dogovortype" SelectedItem="{Binding DogovorType}" DisplayMemberPath="dogovor\_type" FontSize="20" FontWeight="Normal" Grid.Row="14" Grid.Column="1"/>

<TextBlock Text="Лицевойсчет: " FontSize="20" Grid.Row="15" Grid.Column="0"/>

<TextBox HorizontalAlignment="Left" Text="{Binding ls}" FontSize="20" FontWeight="Normal" Grid.Row="15" Grid.Column="1" MaxLength="50"/>

<TextBlock Text="Услуги: " FontSize="20" Grid.Row="16" Grid.Column="0"/>

<TextBox HorizontalAlignment="Left" Text="{Binding servis}" Name="servis" FontSize="20" FontWeight="Normal" Grid.Row="16" Grid.Column="1"/>

<TextBlock Text="№ оборудования: " FontSize="20" Grid.Row="17" Grid.Column="0"/>

<ComboBox HorizontalAlignment="Left" Name="Cb\_oborud" SelectedItem="{Binding Oborud}" DisplayMemberPath="serial\_num" FontSize="20" FontWeight="Normal" Grid.Row="17" Grid.Column="1" />

<TextBlock Text="Парольдлявхода: " FontSize="20" Grid.Row="18" Grid.Column="0"/>

<TextBox HorizontalAlignment="Left" Text="{Binding pass}" FontSize="20" FontWeight="Normal" Grid.Row="18" Grid.Column="1" MaxLength="50"/>

<TextBlock Text="Датазавершения: " FontSize="20" Grid.Row="19" Grid.Column="0"/>

<DatePicker HorizontalAlignment="Left" SelectedDate="{Binding date\_end}" Name="date\_end" FontSize="20" FontWeight="Normal" Grid.Row="19" Grid.Column="1"/>

<TextBlock Text="Причиназавершения: " FontSize="20" Grid.Row="20" Grid.Column="0" />

<TextBox HorizontalAlignment="Left" Text="{Binding reason\_end}" FontSize="20" FontWeight="Normal" Grid.Row="20" Grid.Column="1" Height="100" TextWrapping="Wrap" TextBox.AcceptsReturn="True"/>

<StackPanel Grid.Row="21" Orientation="Horizontal" Margin="5 10" Grid.ColumnSpan="2">

<Button Content="Сохранить" Margin="0 10" Name="Btn\_save" Click="Btn\_save\_Click"/>

<Button Content="Назад" Margin="10" Name="Btn\_back" Click="Btn\_back\_Click"/>

</StackPanel>

</Grid>

</ScrollViewer>

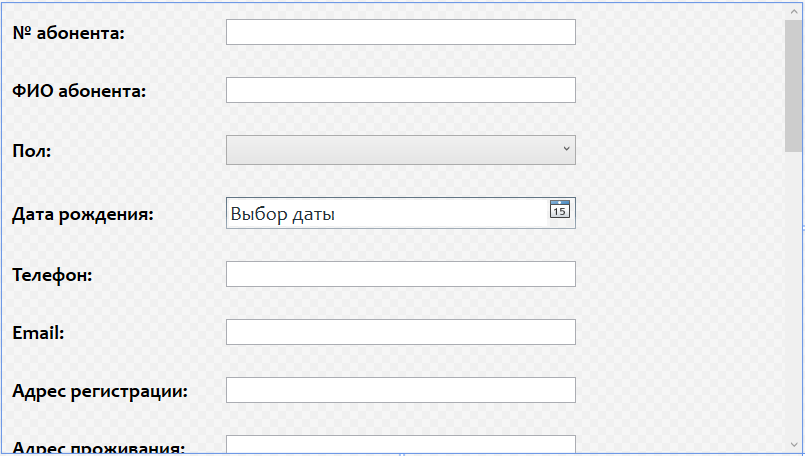


Рисунок.3.2.7 «ОтображениеAddEditAbonent»

1. DocumentPage.xaml – страница для работы с документами (Рис.3.2.8).

Данная страница отображает поле, в котором можно писать текст, открывать документы, редактировать их, а также выводить документы на печать.

Код xaml:

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<!--блоксфункциями-->

<ToolBarTray Background="Black" Grid.Row="0" Width="21cm">

<!--блок с редактированием текста-->

<ToolBar Background="#FFEAEAEA">

<Button Click="new\_ButonClick" FontSize="16" FontWeight="Bold">New</Button>

<Button Click="open\_ButonClick" FontSize="16"FontWeight="Bold">Open</Button>

<Button Click="save\_ButonClick" FontSize="16"FontWeight="Bold">Save</Button>

<Button Click="print\_ButonClick" FontSize="16"FontWeight="Bold">Print</Button>

</ToolBar>

<ToolBar FontFamily="Times New Roman" Background="#FFEAEAEA">

<Button FontWeight="Bold" Command="EditingCommands.ToggleBold" FontSize="16" Foreground="Black">B</Button>

<Button FontStyle="Italic" Command="EditingCommands.ToggleItalic" FontSize="16" Foreground="Black">I</Button>

<Button Command="EditingCommands.ToggleUnderline" FontSize="16" Foreground="Black">U</Button>

</ToolBar>

</ToolBarTray>

<!--блок для отображения текста-->

<RichTextBox x:Name="richTextBox" Grid.Row="1" BorderBrush="Black" BorderThickness="2" Width="21cm"/>

</Grid>

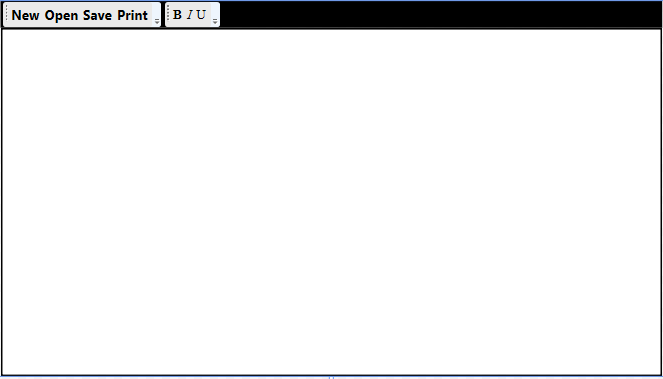


Рисунок.3.2.8 «Отображение DocumentPage»

1. CRMPage.xaml – страница для отображения списка заявок(Рис.3.2.9).

Данная страница отображает список всех заявок на оказание услуг, так же есть кнопки для создания заявки, редактирования, а также просмотра подробной информации.

Код xaml:

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="auto"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<TextBlock Text="Заявки" FontSize="24" Margin="10" Grid.Row="0"/>

<!--кнопкасозданиязаявки-->

<Button Content="Создатьзаявку" Name="AddOrder" Click="AddOrder\_Click" VerticalAlignment="Center" HorizontalAlignment="Right" Margin="10" Padding="2" Grid.Row="0"/>

<!--таблицасотображениемзаявок-->

<DataGrid Grid.Row="1" x:Name="DGOrder" AutoGenerateColumns="False" CanUserResizeRows="False" IsReadOnly="True" FontSize="16">

<DataGrid.Columns>

<DataGridTextColumn Header="Фиоабонента" Binding="{Binding Abonent.fio}"/>

<DataGridTextColumn Header="Телефонабонента" Binding="{Binding Abonent.phone}"/>

<DataGridTextColumn Header="№ заявки" Binding="{Binding num\_order}"/>

<DataGridTextColumn Header="Статусзаявки" Binding="{Binding name\_status}"/>

<DataGridTextColumn Header="Типпроблемы" Binding="{Binding problem\_type}"/>

<DataGridTextColumn Header="Сотрудник" Binding="{Binding Manager.fio}"/>

<!--кнопка для всех записей таблицы-->

<DataGridTemplateColumn x:Name="more">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<Button Name="more" Click="more\_Click" Content="Подробнее"/>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

<!--кнопкадлявсехзаписейтаблицы-->

<DataGridTemplateColumn x:Name="edit">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<Button Name="edit" Click="edit\_Click" Content="Редактировать"/>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

</DataGrid.Columns>

</DataGrid>

</Grid>

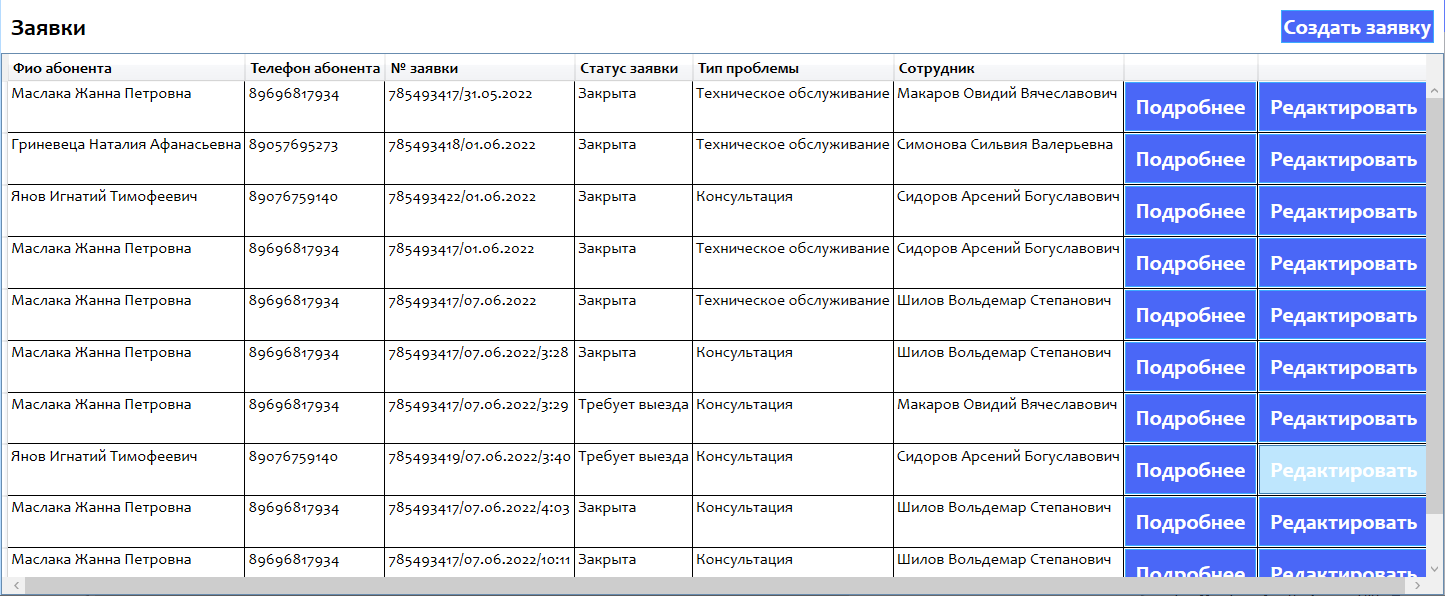


Рисунок.3.2.9 «Отображение CRMPage»

1. CheckAbonent.xaml(Рис.3.2.10).

Страницанеобходима для проверки данных об абоненте, который хочет подать заявку.

Код xaml:

<Grid Margin="10 0">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="auto"/>

<ColumnDefinition Width="auto"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

</Grid.RowDefinitions>

<!--блок для ввода телефона позвонившего абонента-->

<TextBlock Text="Номертелефона: " FontSize="20" Grid.Row="0" Grid.Column="0"/>

<TextBox Text="{Binding Abonent.phone}" Name="phone" HorizontalAlignment="Left" MaxLength="50" FontSize="20" FontWeight="Normal" Grid.Row="0" Grid.Column="1"/>

<!--блок для ввода фио позвонившего абонента-->

<TextBlock Text="ФИОабонента: " FontSize="20" Grid.Row="1" Grid.Column="0"/>

<TextBox Text="{Binding Abonent.fio}" Name="fio" HorizontalAlignment="Left" MaxLength="150" FontSize="20" FontWeight="Normal" Grid.Row="1" Grid.Column="1"/>

<StackPanel Grid.Row="2" Orientation="Horizontal" Grid.ColumnSpan="2">

<Button Name="BtnBack" Click="BtnBack\_Click" Content="Назад" Padding="3"/>

<Button Name="BtnCheckOrder" Click="BtnCheckOrder\_Click" Content="Проверка" Padding="3" Margin="10"/>

</StackPanel>

</Grid>



Рисунок.3.2.10 «Отображение CheckAbonent»

1. AddEditOrder.xaml – страница для добавления или редактирования информации о заявках (Рис.3.2.11).

Код xaml:

<ScrollViewer>

<Grid Margin="10 0">

<Grid.RowDefinitions>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="auto"/>

<ColumnDefinition Width="auto"/>

</Grid.ColumnDefinitions>

<TextBlock Text="ФИО: " FontSize="20" Grid.Row="0" Grid.Column="0" Margin="0 10"/>

<TextBlock Name="fio" HorizontalAlignment="Left" FontSize="20" FontWeight="Normal" Grid.Row="0" Grid.Column="1" Margin="10"/>

<TextBlock Text="Номерабонента:" FontSize="20" Grid.Row="1" Grid.Column="0" Margin="0 10"/>

<TextBlock Name="ab\_num" HorizontalAlignment="Left" FontSize="20" FontWeight="Normal" Grid.Row="1" Grid.Column="1" Margin="10"/>

<TextBlock Text="Номерзаявления: " FontSize="20" Grid.Row="2" Grid.Column="0" Margin="0 10"/>

<TextBlock Name="num\_order" Text="{Binding num\_order}" HorizontalAlignment="Left" FontSize="20" FontWeight="Normal" Grid.Row="2" Grid.Column="1" Margin="10"/>

<TextBlock Text="Датасоздания: " FontSize="20" Grid.Row="3" Grid.Column="0" Margin="0 10"/>

<TextBlock Name="data\_create" Text="{Binding data\_create, StringFormat=d}" HorizontalAlignment="Left" FontSize="20" FontWeight="Normal" Grid.Row="3" Grid.Column="1" Margin="10"/>

<TextBlock Text="Лицевойсчет: " FontSize="20" Grid.Row="4" Grid.Column="0" Margin="0 10"/>

<TextBlock Name="ls" HorizontalAlignment="Left" FontSize="20" FontWeight="Normal" Grid.Row="4" Grid.Column="1" Margin="10"/>

<TextBlock Text="Услуга: " FontSize="20" Grid.Row="5" Grid.Column="0" Margin="0 10"/>

<ComboBox SelectedItem="{Binding Servi}" Name="CbServ" DisplayMemberPath="name\_serv" Margin="10" HorizontalAlignment="Left" FontSize="20" FontWeight="Normal" Grid.Row="5" Grid.Column="1"/>

<TextBlock Text="Видуслуги: " FontSize="20" Grid.Row="6" Grid.Column="0" Margin="0 10"/>

<ComboBox Width="auto" SelectedItem="{Binding VidServi}" Name="CbVidServ" DisplayMemberPath="name\_vid" SelectionChanged="CbVidServ\_SelectionChanged" Margin="10" HorizontalAlignment="Left" FontSize="20" FontWeight="Normal" Grid.Row="6" Grid.Column="1"/>

<TextBlock Text="Типуслуги" FontSize="20" Grid.Row="7" Grid.Column="0" Margin="0 10" />

<StackPanel Orientation="Horizontal" Grid.Row="7" Grid.Column="1" Margin="0 10">

<ComboBox Width="auto" SelectedItem="{Binding TypeServi}" Name="CbTypeServ" DisplayMemberPath="name\_type" Margin="10" HorizontalAlignment="Left" FontSize="20" FontWeight="Normal" Grid.Row="7" Grid.Column="1"/>

<Button Click="refrash\_Click" Name="refrash" Width="40" Height="40">

<Button.Background>

<ImageBrush ImageSource="/NevaTelecomv\_1.0;component/Resources/view.png"/>

</Button.Background>

<Button.BorderBrush>

<ImageBrush/>

</Button.BorderBrush>

</Button>

</StackPanel>

<TextBlock Text="Статусзаявки" FontSize="20" Grid.Row="8" Grid.Column="0" Margin="0 10"/>

<ComboBox SelectedItem="{Binding Status}" Name="CbStatus" DisplayMemberPath="name\_status" Margin="10" HorizontalAlignment="Left" FontSize="20" FontWeight="Normal" Grid.Row="8" Grid.Column="1"/>

<TextBlock Text="Типпроблемы" FontSize="20" Grid.Row="9" Grid.Column="0" Margin="0 10"/>

<ComboBox SelectedItem="{Binding TypeProblem}" Name="CbTypeProblem" DisplayMemberPath="problem\_type" Margin="10" HorizontalAlignment="Left" FontSize="20" FontWeight="Normal" Grid.Row="9" Grid.Column="1"/>

<TextBlock Text="Сотрудник" FontSize="20" Grid.Row="10" Grid.Column="0" Margin="0 10"/>

<ComboBox SelectedItem="{Binding Manager}" Name="CbManager" DisplayMemberPath="fio" Margin="10" HorizontalAlignment="Left" FontSize="20" FontWeight="Normal" Grid.Row="10" Grid.Column="1"/>

<TextBlock Text="Описание: " FontSize="20" Grid.Row="11" Grid.Column="0" />

<TextBox Name="describe" Text="{Binding describe}" HorizontalAlignment="Left" FontSize="20" FontWeight="Normal" Grid.Row="11" Grid.Column="1" TextWrapping="Wrap" TextBox.AcceptsReturn="True"/>

<TextBlock Text="Датазавершения " FontSize="20" Grid.Row="12" Grid.Column="0" Margin="0 10"/>

<TextBlock Name="date\_end" Text="{Binding date\_end, StringFormat=d}" HorizontalAlignment="Left" FontSize="20" FontWeight="Normal" Grid.Row="12" Grid.Column="1" Margin="10"/>

<Button Name="btnSave" Click="btnSave\_Click" Content="Сохранить" Grid.Row="13" Margin="0 10" HorizontalAlignment="Left"/>

</Grid>

</ScrollViewer>

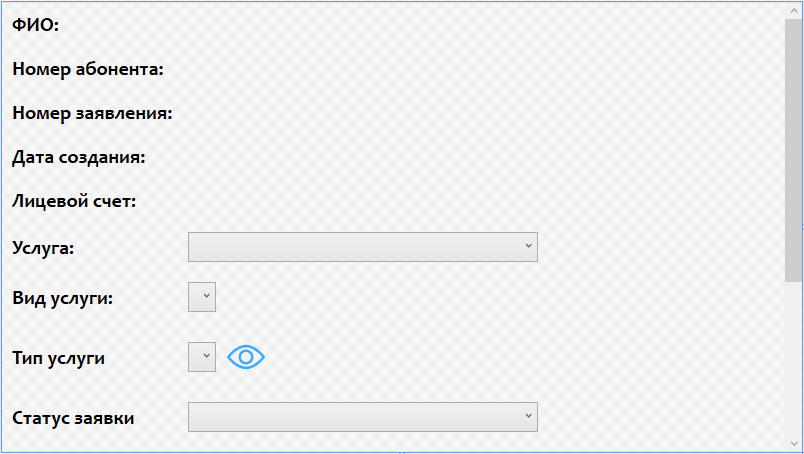


Рисунок.3.2.11 «Отображение AddEditOrder»

1. OrderMore.xaml – страница подробной информации о заявке (Рис.3.2.12).

Данная страница предназначена для подробного просмотра информации об заявке.

Код xaml:

<ScrollViewer>

<Grid Margin="10 0">

<Grid.RowDefinitions>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="auto"/>

<ColumnDefinition Width="auto"/>

</Grid.ColumnDefinitions>

<TextBlock Text="ФИО: " FontSize="20" Grid.Row="0" Grid.Column="0" Margin="0 10"/>

<TextBlock Text="{Binding Abonent.fio}" HorizontalAlignment="Left" FontSize="20" FontWeight="Normal" Grid.Row="0" Grid.Column="1" Margin="10"/>

<TextBlock Text="Номерабонента:" FontSize="20" Grid.Row="1" Grid.Column="0" Margin="0 10"/>

<TextBlock Text="{Binding Abonent.number}" HorizontalAlignment="Left" FontSize="20" FontWeight="Normal" Grid.Row="1" Grid.Column="1" Margin="10"/>

<TextBlock Text="Номерзаявления: " FontSize="20" Grid.Row="2" Grid.Column="0" Margin="0 10"/>

<TextBlock Name="num\_order" Text="{Binding num\_order}" HorizontalAlignment="Left" FontSize="20" FontWeight="Normal" Grid.Row="2" Grid.Column="1" Margin="10"/>

<TextBlock Text="Датасоздания: " FontSize="20" Grid.Row="3" Grid.Column="0" Margin="0 10"/>

<TextBlock Name="data\_create" Text="{Binding data\_create, StringFormat=d}" HorizontalAlignment="Left" FontSize="20" FontWeight="Normal" Grid.Row="3" Grid.Column="1" Margin="10"/>

<TextBlock Text="Лицевойсчет: " FontSize="20" Grid.Row="4" Grid.Column="0" Margin="0 10"/>

<TextBlock Text="{Binding Abonent.ls}" HorizontalAlignment="Left" FontSize="20" FontWeight="Normal" Grid.Row="4" Grid.Column="1" Margin="10"/>

<TextBlock Text="Услуга: " FontSize="20" Grid.Row="5" Grid.Column="0" Margin="0 10"/>

<TextBlock Text="{Binding name\_serv}" Margin="10" HorizontalAlignment="Left" FontSize="20" FontWeight="Normal" Grid.Row="5" Grid.Column="1"/>

<TextBlock Text="Видуслуги: " FontSize="20" Grid.Row="6" Grid.Column="0" Margin="0 10"/>

<TextBlock Text="{Binding name\_vid}" Margin="10" HorizontalAlignment="Left" FontSize="20" FontWeight="Normal" Grid.Row="6" Grid.Column="1"/>

<TextBlock Text="Типуслуги" FontSize="20" Grid.Row="7" Grid.Column="0" Margin="0 10" />

<TextBlock Text="{Binding name\_type}" Margin="10" HorizontalAlignment="Left" FontSize="20" FontWeight="Normal" Grid.Row="7" Grid.Column="1"/>

<TextBlock Text="Статусзаявки" FontSize="20" Grid.Row="8" Grid.Column="0" Margin="0 10"/>

<TextBlock Text="{Binding name\_status}" Margin="10" HorizontalAlignment="Left" FontSize="20" FontWeight="Normal" Grid.Row="8" Grid.Column="1"/>

<TextBlock Text="Типпроблемы" FontSize="20" Grid.Row="9" Grid.Column="0" Margin="0 10"/>

<TextBlock Text="{Binding problem\_type}" Margin="10" HorizontalAlignment="Left" FontSize="20" FontWeight="Normal" Grid.Row="9" Grid.Column="1"/>

<TextBlock Text="Сотрудник" FontSize="20" Grid.Row="10" Grid.Column="0" Margin="0 10"/>

<TextBlock Text="{Binding Manager.fio}" Margin="10" HorizontalAlignment="Left" FontSize="20" FontWeight="Normal" Grid.Row="10" Grid.Column="1"/>

<TextBlock Text="Описание: " FontSize="20" Grid.Row="11" Grid.Column="0" />

<TextBlock Text="{Binding describe}" HorizontalAlignment="Left" FontSize="20" FontWeight="Normal" Grid.Row="11" Grid.Column="1"/>

<TextBlock Text="Датазавершения " FontSize="20" Grid.Row="12" Grid.Column="0" Margin="0 10"/>

<TextBlock Text="{Binding date\_end, StringFormat=d}" HorizontalAlignment="Left" FontSize="20" FontWeight="Normal" Grid.Row="12" Grid.Column="1" Margin="10"/>

<Button Name="btnBack" Click="btnBack\_Click" Content="Назад" Grid.Row="13" Margin="0 10" HorizontalAlignment="Left"/>

</Grid>

</ScrollViewer>

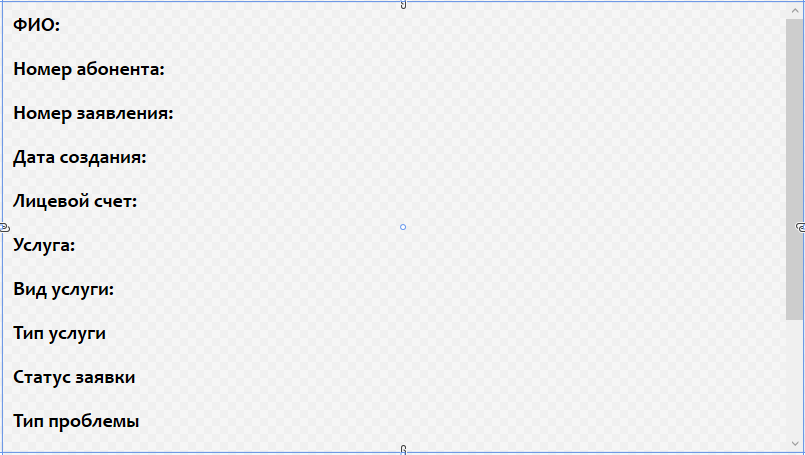


Рисунок.3.2.12 «Отображение OrderMore»

# 3.3 Программирование информационной системы

Для того чтобы приложение заработало, необходимо прописать для каждой страницы логику, подключить Базу Данных и создать класс для переходов между страницами.

Подключение БД:

1. Выбираем ADO.NET Entity Data Model иназываем BaseModel (Рис.3.3.1).
2. Выбираем EF Designer from database.
3. Создаем новое подключение и выбираем базу данных. Имя сервера: localhost\SQLEXPRESS, название БД: NevaTelecom1(Рис.3.3.2).
4. Выбираем таблицы, которые необходимо добавить в модель. И модель добавлена.

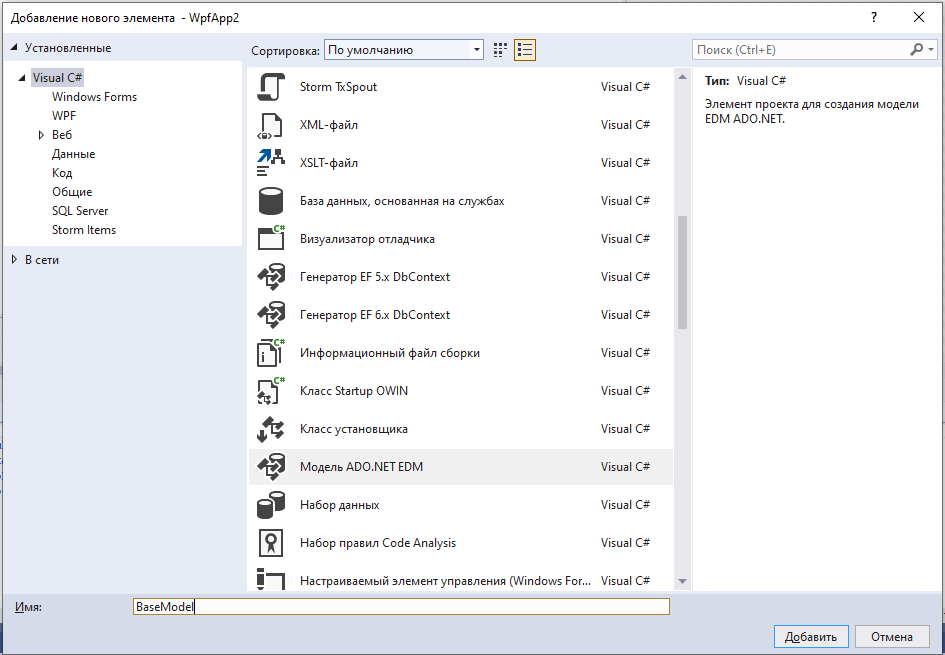


Рисунок.3.3.1 «Создание элемента BaseModel»

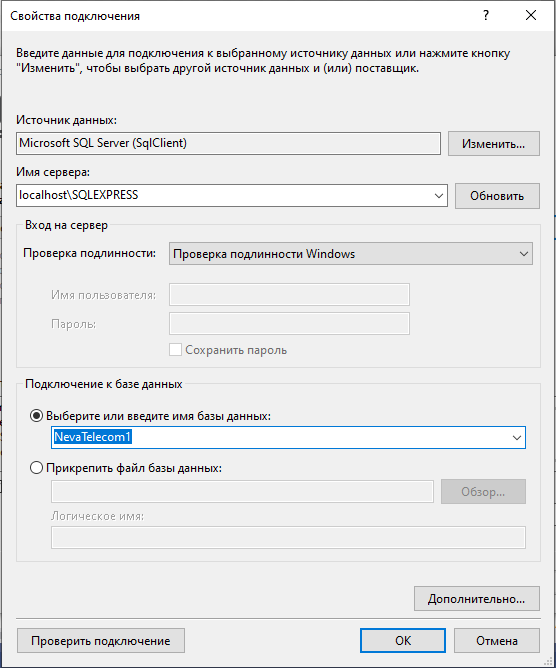


Рисунок.3.3.2 «Подключение BaseModel»

Создание класса Navigation.cs:

1. Правой кнопкой на название проекта — Add — Class.
2. Создаем класс Navigation и в нем статичное свойство MainFrame (Рис.3.3.3).
3. Создаем класс Navigation1 и в нем статичное свойство MainFrame1 (Рис.3.3.3).

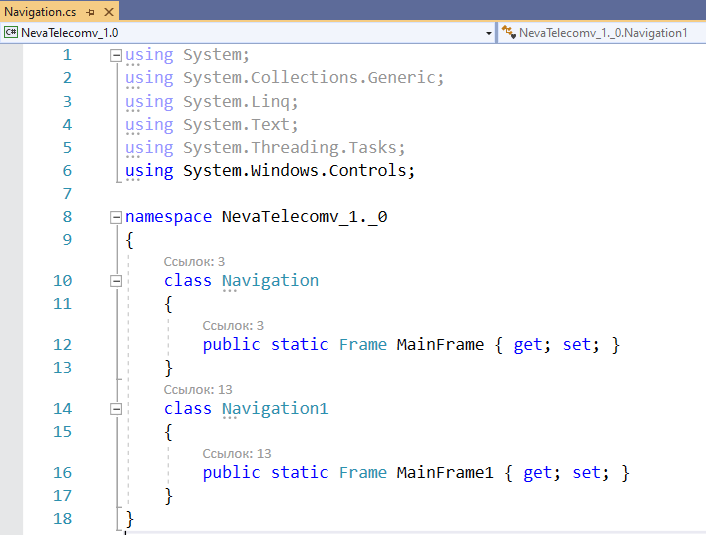


Рисунок.3.3.3 «Создание классов»

В логике каждой страницы будет:

1. InitializeComponent(); — генерируется автоматически при компиляции приложения.
2. privatevoidНазвание\_кнопки\_Click(objectsender, RoutedEventArgse)  — функция для обработки события при нажатии на кнопку.

Код логики страницы EnterPage.xaml:

publicEnterPage()

{

InitializeComponent();

Global.codes = null; //присваиваем значение глобальной переменной

}

publicstaticclassGlobal//Создание глобальных переменных

{

public static int manager\_id;

public static string manager\_role;

public static string title;

public static string codes;

}

private void gencode()//методгенерациикода

{

Random random = new Random();

Global.codes = random.Next(1000, 9999).ToString();

MessageBox.Show("Ваш код: " + Global.codes); //вывод сообщения с кодом

}

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

{

gencode(); //вызов метода генерации кода

}

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

{

using (var db = new NevaTelecom1Entities())

{

// поиск совпадений введенного логина и логинов в БД

var login = db.Managers.AsNoTracking().FirstOrDefault(m => m.login == log.Text.Trim());

// поиск совпадений введенного логина и логинов в БД, а так же паролей

var manager = db.Managers.AsNoTracking().FirstOrDefault(m => m.login == log.Text.Trim() & (m.pass == pass.Password || m.pass == text\_pass.Text));

if (login != null)

{

Show.IsEnabled = true; // делаемкнопкиактивными

pass.IsEnabled = true;

text\_pass.IsEnabled = true;

}

else

{

MessageBox.Show("Логинневерный");// блокируемкнопки

Show.IsEnabled = false;

pass.IsEnabled = false;

text\_pass.IsEnabled = false;

code.IsEnabled = false;

pass.IsEnabled = false;

btnCode.IsEnabled = false;

Global.codes = null;

}

if (manager != null && Global.codes == null)

{

code.IsEnabled = true;

pass.IsEnabled = true;

btnCode.IsEnabled = true;

gencode();

}

else if (login != null && manager == null)

{

MessageBox.Show("Парольневерный");

Global.codes = null;

}

if (Global.codes != null)

{

if (Global.codes == code.Text & manager != null)

{

//заносим данные пользователя глобальную переменную в переменную

Global.manager\_role = manager.name\_role;

Global.manager\_id = manager.id\_manager;

if (manager.name\_role == "Технический специалист" || manager.name\_role == "Руководитель технического департамента")

Global.title = "CRM - заявки";

else

Global.title = "Абоненты";

//переходнаглавнуюстрприлодения

Navigation.MainFrame.Navigate(new GeneralPage());

}

else

MessageBox.Show("Кодневерный");

}

}

}

//обработчиксобытияпоказа/скрытияпароля

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

{

if (text\_pass.Visibility == Visibility.Hidden)

{

text\_pass.Text = pass.Password;

text\_pass.Visibility = Visibility.Visible;

pass.Visibility = Visibility.Hidden;

}

else

{

pass.Password = text\_pass.Text;

pass.Visibility = Visibility.Visible;

text\_pass.Visibility = Visibility.Hidden;

}

}

Кодлогикистраницы GeneralPage.xaml:

public GeneralPage()

{

InitializeComponent();

Navigation1.MainFrame1 = MainFrame1;

//Разграничение ролей, в зависимости от роли доступны те или иные кнопки

if (EnterPage.Global.manager\_role == "Технический специалист" || EnterPage.Global.manager\_role == "Руководитель технического департамента")

{

Btn\_abonent.IsEnabled = false;//скрытиекнопок

Btn\_doc.IsEnabled = false;

Title.Text = EnterPage.Global.title;//отображаемпеременнуювзаголовке

MainFrame1.Navigate(new pages.CRMPage());//отображение страницы при переходе на главую стр

}

else

{

Title.Text = EnterPage.Global.title;

MainFrame1.Navigate(new pages.AbonentPage());

}

//задаем коллекцию данных в блок события, в зависимости от роли сотрудника

LvTask.ItemsSource = NevaTelecom1Entities.GetContext().Tasks.Where(p => p.name\_role == EnterPage.Global.manager\_role).ToList();

}

public static class Global

{

public static string role;

public static string title;

public static int id\_abonenta;

public static string vid\_serv;

}

private void Btn\_doc\_Click(object sender, RoutedEventArgs e)

{

MainFrame1.Navigate(new pages.DocumentPage());//переходнастрдокументы

Title.Text = "Документы";

}

private void Btn\_CRM\_Click(object sender, RoutedEventArgs e)

{

MainFrame1.Navigate(new pages.CRMPage()); //переходнастрработассзаявками

Title.Text = "CRM - заявки";

}

private void Btn\_abonent\_Click(object sender, RoutedEventArgs e)

{

MainFrame1.Navigate(new pages.AbonentPage()); //переходнастрабонентов

Title.Text = "Абоненты";

}

private void Btn\_exit\_Click(object sender, RoutedEventArgs e)

{

//всплывающее окно (предупреждение)

if (MessageBox.Show($"Вы действительно хотите выйти из системы", "Внимание",

MessageBoxButton.YesNo, MessageBoxImage.Question) == MessageBoxResult.Yes)

{

Navigation.MainFrame.Navigate(new EnterPage());//переходнастрвхода

}

}

КодлогикистраницыAbonentPage.xaml:

public AbonentPage()

{

InitializeComponent();

BtnActiv.IsChecked = true;//кнопкаактивныхабонентоввыбрана

update(); //вызов метода

}

// метод который позволяет обновить данные DataGrid учитывая данные введеные для поиска

private void update()

{

//переменная с данными абонентов, где дата расторжения null

var \_curentAbonent = NevaTelecom1Entities.GetContext().Abonents.Where(p => p.date\_end == null).ToList();

//присваиваем данные, где данные абонентав содержат введенные данные (поиск по фамилии и лицевому счету)

\_curentAbonent = \_curentAbonent.Where(p => p.ls.ToLower().Contains(Tb\_ls.Text.ToLower())).ToList();

\_curentAbonent = \_curentAbonent.Where(p => p.fio.ToLower().Contains(Tb\_fio.Text.ToLower())).ToList();

//присваиваем списов абонентов таблице для отображения данных

DGridAbonent.ItemsSource = \_curentAbonent;

}

//переход на стр с подробной информацией

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

{

Navigation1.MainFrame1.Navigate(new pages.AbonentMore((sender as Button).DataContext as Abonent)); //передаемобъектсданнымисвыбраннойстрокитаблицы

}

private void BtnActiv\_Checked(object sender, RoutedEventArgs e)

{

//отображениевтаблицеактивныхпользователей

DGridAbonent.ItemsSource = NevaTelecom1Entities.GetContext().Abonents.Where(p => p.date\_end == null).ToList();

}

private void BtnUnActiv\_Checked(object sender, RoutedEventArgs e)

{

//отображениевтаблиценеактивныхпользователей

DGridAbonent.ItemsSource = NevaTelecom1Entities.GetContext().Abonents.Where(p => p.date\_end != null).ToList();

}

private void BtnAll\_Checked(object sender, RoutedEventArgs e)

{

//отображениевтаблицевсехпользователей

DGridAbonent.ItemsSource = NevaTelecom1Entities.GetContext().Abonents.ToList();

}

private void Btn\_Add\_exit\_Click(object sender, RoutedEventArgs e)

{

//переход на страницу добавления/редактирования данных и передаем пустой объект

Navigation1.MainFrame1.Navigate(new pages.AddEditAbonent(null));

}

//обработчик события при вводе информации в поле поиск по фамилии

private void Tb\_fio\_SelectionChanged(object sender, RoutedEventArgs e)

{

update();

}

//обработчик события при вводе информации в поле поиск по личному счету

private void Tb\_ls\_SelectionChanged(object sender, RoutedEventArgs e)

{

update();

}

//кнопкаредактированияданных

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

{

//получаем объект данных выбранной строки таблицы

Abonent \_currentAbonent = (sender as Button).DataContext as Abonent;

//если договор расторгнут то редактирование запрещено

if (\_currentAbonent.date\_end != null)

{

MessageBox.Show("Договор расторгнут \nНельзя вносить изменения");

}

//инае переход на страницу добавления/редактирования данных передаем

else

Navigation1.MainFrame1.Navigate(new pages.AddEditAbonent((sender as Button).DataContext as Abonent));

}

//Если видимость страницы isVisible, мы будем обращаться к контексту с помощью свойства ChangeTracker ко всем сущностям, которые есть. И для каждой из них будем выполнять метод перезагрузки и вывода актуальных данных.

private void Page\_IsVisibleChanged(object sender, DependencyPropertyChangedEventArgs e)

{

if (Visibility == Visibility.Visible)

{

NevaTelecom1Entities.GetContext().ChangeTracker.Entries().ToList().ForEach(p => p.Reload());

update();

}

}

КодлогикистраницыAbonentMore.xaml:

public Abonent \_abonent = new Abonent(); // создаемобъект Abonent

public AbonentMore(Abonent selectedAbonent)

{

InitializeComponent();

//если переданный объект не пустой, значит происходит редактирование

if (selectedAbonent != null)

//привязываем полученные данные с полями

\_abonent = selectedAbonent;

//привязываем полученные данные с полями

DataContext = \_abonent;

//условие если дата расторжения договора пустая

if (\_abonent.date\_end == null)

{

//скрываем данные о дате расторжения и причине

DogovorDateEnd.Visibility = Visibility.Hidden;

DogovorReasonEnd.Visibility = Visibility.Hidden;

}

}

Кодлогикистраницы AddEditAbonent.xaml:

private Abonent \_currentAbonent = new Abonent(); // создаемобъект Abonent

public AddEditAbonent(Abonent selectedAbonent)

{

InitializeComponent();

//если переданный объект не пустой, значит происходит редактирование

if (selectedAbonent != null)

{

// присваиваем полученные данные созданному ранее объекту

\_currentAbonent = selectedAbonent;

//отключаем кнопку генерации номера договора

gen\_num.Visibility = Visibility.Hidden;

}

//если переданный объект пустой, значит происходит добавление

if (selectedAbonent == null)

{

//присваеваем сегоднняшнее число дате создания договора

date\_dogovor.Text = DateTime.Now.ToShortDateString();

\_currentAbonent.date\_dogovor = DateTime.Now;

}

//привязываем полученные данные с полями

DataContext = \_currentAbonent;

//задаем значения для выпадающих списков

Cb\_gender.ItemsSource = NevaTelecom1Entities.GetContext().Genders.ToList();

Cb\_dogovortype.ItemsSource = NevaTelecom1Entities.GetContext().DogovorTypes.ToList();

Cb\_dogovortype.ItemsSource = NevaTelecom1Entities.GetContext().DogovorTypes.ToList();

Cb\_oborud.ItemsSource = NevaTelecom1Entities.GetContext().Oboruds.ToList();

}

//кнопкасохраненияданных

private void Btn\_save\_Click(object sender, RoutedEventArgs e)

{

//контейнерошибок

StringBuilder errors = new StringBuilder();

//проверкавведенныхданных

if (string.IsNullOrWhiteSpace(\_currentAbonent.number) )

errors.AppendLine("Введите № абонента");

//если введеные данные фио содержат цифры

if (string.IsNullOrWhiteSpace(\_currentAbonent.fio) || Regex.Match(\_currentAbonent.fio, "[0-9]").Success)

errors.AppendLine("ВведитеФИОабонента");

if (\_currentAbonent.Gender1 == null)

errors.AppendLine("Выберитепол");

if (\_currentAbonent.birth == null)

errors.AppendLine("Выберитедатурождения");

//если введеные данные фио содержат буквы

if (string.IsNullOrWhiteSpace(\_currentAbonent.phone) || \_currentAbonent.phone.Length != 11 ||

Regex.Match(\_currentAbonent.phone, "[А-Я]").Success || Regex.Match(\_currentAbonent.phone, "[а-я]").Success ||

Regex.Match(\_currentAbonent.phone, "[a-z]").Success || Regex.Match(\_currentAbonent.phone, "[A-Z]").Success)

errors.AppendLine("Введите номер телефона абонента");

if(string.IsNullOrWhiteSpace(\_currentAbonent.email) || (Regex.Match(\_currentAbonent.email, "[@]").Success != true))

errors.AppendLine("Введите email");

if (string.IsNullOrWhiteSpace(\_currentAbonent.address\_reg))

errors.AppendLine("Введите адрес регистрации");

if (string.IsNullOrWhiteSpace(\_currentAbonent.address\_live))

errors.AppendLine("Введите адрес жительства");

//если введеные данные фио содержат буквы

if (string.IsNullOrWhiteSpace(\_currentAbonent.pasport) || \_currentAbonent.pasport.Length != 11 ||

Regex.Match(\_currentAbonent.pasport, "[А-Я]").Success || Regex.Match(\_currentAbonent.pasport, "[а-я]").Success ||

Regex.Match(\_currentAbonent.pasport, "[a-z]").Success || Regex.Match(\_currentAbonent.pasport, "[A-Z]").Success)

errors.AppendLine("Введите номер (4) и серия (6) через пробел");

//если введеные данные фио содержат буквы

if (string.IsNullOrWhiteSpace(\_currentAbonent.code\_otdel) || \_currentAbonent.code\_otdel.Length != 7 ||

Regex.Match(\_currentAbonent.code\_otdel, "[А-Я]").Success || Regex.Match(\_currentAbonent.code\_otdel, "[а-я]").Success ||

Regex.Match(\_currentAbonent.code\_otdel, "[a-z]").Success || Regex.Match(\_currentAbonent.code\_otdel, "[A-Z]").Success)

errors.AppendLine("Введите код подразделения (ххх-ххх)");

if (string.IsNullOrWhiteSpace(\_currentAbonent.department))

errors.AppendLine("Введите кем выдан паспорт");

if (\_currentAbonent.date\_issue == null)

errors.AppendLine("Выберите дату выдачи паспорта");

if (string.IsNullOrWhiteSpace(\_currentAbonent.num\_dogovor))

errors.AppendLine("Сформируйте номер договора");

if (\_currentAbonent.date\_dogovor == null)

errors.AppendLine("Введите дату создания договора");

if (\_currentAbonent.DogovorType == null)

errors.AppendLine("Выберитетипдоговора");

//если введеные данные фио содержат буквы

if (string.IsNullOrWhiteSpace(\_currentAbonent.ls) ||

Regex.Match(\_currentAbonent.ls, "[А-Я]").Success || Regex.Match(\_currentAbonent.ls, "[а-я]").Success ||

Regex.Match(\_currentAbonent.ls, "[a-z]").Success || Regex.Match(\_currentAbonent.ls, "[A-Z]").Success)

errors.AppendLine("Введите личный счет");

if (string.IsNullOrWhiteSpace(\_currentAbonent.servis))

errors.AppendLine("Введите услуги (Интернет, ТВ, Видеонаблюдение)");

if (\_currentAbonent.Oborud == null)

errors.AppendLine("Выберите № оборудования");

if (string.IsNullOrWhiteSpace(\_currentAbonent.pass))

errors.AppendLine("Введитепароль");

\_currentAbonent.name\_role = "Пользователь";

if (date\_end.SelectedDate != null)

{

if (string.IsNullOrWhiteSpace(\_currentAbonent.reason\_end))

errors.AppendLine("Введите причину расторжения");

}

//если в контейнере есть ошибки то выводим их

if (errors.Length > 0)

{

MessageBox.Show(errors.ToString());

return;

}

//добавляем данные объекта в таблицу бд

if (\_currentAbonent.id\_abon == 0) // происходит во время добавления

{

//добавляем данные объекта в таблицу бд

NevaTelecom1Entities.GetContext().Abonents.Add(\_currentAbonent);

}

//конструктор для отлавливания системных ошибок (не было сбоя системы)

try

{

//сохраняемизмененияБД

NevaTelecom1Entities.GetContext().SaveChanges();

MessageBox.Show("Данныесохранены");

Navigation1.MainFrame1.Navigate(new pages.AbonentPage());

}

catch(Exception ex)

{

MessageBox.Show(ex.ToString());

}

}

//кнопка для перехода на стр абонентов

private void Btn\_back\_Click(object sender, RoutedEventArgs e)

{

Navigation1.MainFrame1.Navigate(new pages.AbonentPage());

}

//кнопкагенерацииномерадоговора

private void gen\_num\_Click(object sender, RoutedEventArgs e)

{

if (string.IsNullOrWhiteSpace(\_currentAbonent.number))

MessageBox.Show("Введите номер абонента");

else

{

// создаем номер договора, который состоит из номера абонента, месяца и года сосздания договора

dogovor.Text = \_currentAbonent.number + "-" + DateTime.Now.Month.ToString() + " - " + DateTime.Now.Year.ToString();

\_currentAbonent.num\_dogovor = dogovor.Text;

}

}

КодлогикистраницыDocumentPage.xaml:

public DocumentPage()

{

InitializeComponent();

}

private void save\_ButonClick(object sender, RoutedEventArgs e)

{

Microsoft.Win32.SaveFileDialog save = new Microsoft.Win32.SaveFileDialog();

save.Filter =

"Файл RTF-файл (\*.rtf)|\*.rtf";

if (save.ShowDialog() == true)

{

// Создание контейнера TextRange для всего документа

TextRange documentTextRange = new TextRange(

richTextBox.Document.ContentStart, richTextBox.Document.ContentEnd);

// Если такой файл существует, он перезаписывается,

using (FileStream fs = File.Create(save.FileName))

{

if (System.IO.Path.GetExtension(save.FileName).ToLower() == ".rtf")

{

documentTextRange.Save(fs, DataFormats.Rtf);

}

else

{

documentTextRange.Save(fs, DataFormats.Xaml);

}

}

}

}

private void open\_ButonClick(object sender, RoutedEventArgs e)

{

Microsoft.Win32.OpenFileDialog openFile =

new Microsoft.Win32.OpenFileDialog();

openFile.Filter = "RichText files (\*.rtf)|\*.rtf|All files (\*.\*)|\*.\*";

if (openFile.ShowDialog() == true)

{

TextRange tr = new TextRange(

richTextBox.Document.ContentStart, richTextBox.Document.ContentEnd);

using (FileStream fs = File.Open(openFile.FileName, FileMode.Open))

{

tr.Load(fs, DataFormats.Rtf);

}

}

// Копирование содержимого документа в MemoryStream.

using (MemoryStream stream = new MemoryStream())

{

TextRange range = new TextRange(richTextBox.Document.ContentStart,

richTextBox.Document.ContentEnd);

range.Save(stream, DataFormats.Xaml);

stream.Position = 0;

}

}

private void new\_ButonClick(object sender, RoutedEventArgs e)

{

richTextBox.Document = new FlowDocument();

}

private void print\_ButonClick(object sender, RoutedEventArgs e)

{

PrintDialog pd = new PrintDialog();

if ((pd.ShowDialog() == true))

{

//use either one of the below

pd.PrintVisual(richTextBox as Visual, "printing as visual");

pd.PrintDocument((((IDocumentPaginatorSource)richTextBox.Document).DocumentPaginator), "printing as paginator");

}

}

Код логики страницы CRMPage.xaml:

public CRMPage()

{

InitializeComponent();

//разграничение ролей

if (EnterPage.Global.manager\_role == "Технический специалист")

{

//отображаем заявки пренадлежащие определенному тех. спец.

DGOrder.ItemsSource = NevaTelecom1Entities.GetContext().Orders.Where(p => p.id\_manager == EnterPage.Global.manager\_id).ToList();

AddOrder.Visibility = Visibility.Hidden;//скрываемкнопки (ограничиваемправа)

edit.Visibility = Visibility.Hidden;

}

else

{

//присваиваем списов заявок таблице для отображения данных

DGOrder.ItemsSource = NevaTelecom1Entities.GetContext().Orders.ToList();

//скрываем кнопки (ограничиваем права)

if (EnterPage.Global.manager\_role == "Руководитель технического департамента")

{

AddOrder.Visibility = Visibility.Hidden;

edit.Visibility = Visibility.Hidden;

}

}

}

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

{

Navigation1.MainFrame1.Navigate(new CheckAbonent());//переходнастрпроверкиабонентавсистеме

}

//переходнастрредактированиязаявки

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

{

//передаем значения в видео объекта, хранящиеся в выбранной записи таблицы

Navigation1.MainFrame1.Navigate(new AddEditOrder((sender as Button).DataContext as Order));

}

//преход на стр с подробной информацией о заявки

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

{

//передаем значения в видео объекта, хранящиеся в выбранной записи таблицы

Navigation1.MainFrame1.Navigate(new OrderMore((sender as Button).DataContext as Order));

}

//Если видимость страницы isVisible, мы будем обращаться к контексту с помощью свойства ChangeTrackerко всем сущностям, которые есть. И для каждой из них будем выполнять метод перезагрузки и выводаактуальных данных.

private void Page\_IsVisibleChanged(object sender, DependencyPropertyChangedEventArgs e)

{

if (Visibility == Visibility.Visible)

{

NevaTelecom1Entities.GetContext().ChangeTracker.Entries().ToList().ForEach(p => p.Reload());

}

}

}

}

КодлогикистраницыCheckAbonent.xaml:

public CheckAbonent()

{

InitializeComponent();

}

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

{

using (var db = new NevaTelecom1Entities())

{

//поиск на совпадение введенных данных в БД

var abonent = db.Abonents.AsNoTracking().FirstOrDefault(p => p.phone == phone.Text && fio.Text == p.fio);

if (abonent != null)

{

GeneralPage.Global.id\_abonenta = abonent.id\_abon;

//переход на стр добавления, передаем пустой объект

Navigation1.MainFrame1.Navigate(new AddEditOrder(null));

}

else

MessageBox.Show("Абонентненайден");

}

}

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

{

Navigation1.MainFrame1.Navigate(new CRMPage());//обработчиксобытиянакнопкуназад

}

КодлогикистраницыAddEditOrder.xaml:

private Order \_curentOrder = new Order(); //созданиеобъекта Order

public AddEditOrder(Order selectedOrder)

{

InitializeComponent();

//если переданный объект не пустой, значит происходит редактирование

if (selectedOrder != null)

{

//присваиваем \_curentOrder переданныеданные

\_curentOrder = selectedOrder;

//находимабонентакоторыйпрошелпроверку

var abonent = NevaTelecom1Entities.GetContext().Abonents.FirstOrDefault(p => p.id\_abon == \_curentOrder.id\_abon);

//присваиваем значения в textbox о абоненте

fio.Text = " " + abonent.fio;

ab\_num.Text = " " + abonent.number;

ls.Text = " " + abonent.ls;

GeneralPage.Global.id\_abonenta = abonent.id\_abon;

//отключаем поля которые не нужно редактировать

CbServ.IsEnabled = false;

CbVidServ.IsEnabled = false;

CbTypeServ.IsEnabled = false;

CbTypeProblem.IsEnabled = false;

if(\_curentOrder.date\_end != null)

{

CbStatus.IsEnabled = false;

describe.IsEnabled = false;

CbManager.IsEnabled = false;

}

}

if (selectedOrder == null)

{

//находимабонентакоторыйпрошелпроверку

var abonent = NevaTelecom1Entities.GetContext().Abonents.FirstOrDefault(p => p.id\_abon == GeneralPage.Global.id\_abonenta);

//присваиваем значения в textbox о абоненте

fio.Text = " " + abonent.fio;

ab\_num.Text = " " + abonent.number;

//формированиеномеразаявки

num\_order.Text = abonent.ls + "/" + DateTime.Now.ToShortDateString() + "/" + DateTime.Now.ToShortTimeString(); ;

data\_create.Text = " " + DateTime.Now.ToShortDateString();

ls.Text = " " + abonent.ls;

}

//привязываем полученные данные с полями

DataContext = \_curentOrder;

//задаем значения для выпадающих списков

CbServ.ItemsSource = NevaTelecom1Entities.GetContext().Servis.ToList();

CbVidServ.ItemsSource = NevaTelecom1Entities.GetContext().VidServis.ToList();

CbStatus.ItemsSource = NevaTelecom1Entities.GetContext().Status.ToList();

CbTypeProblem.ItemsSource = NevaTelecom1Entities.GetContext().TypeProblems.ToList();

//отображаемтолькотехническихспециалистов

CbManager.ItemsSource = NevaTelecom1Entities.GetContext().Managers.Where(p => p.name\_role == "Техническийспециалист").ToList();

}

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

{

//получаем текст из выпадающего списка (вид услуги)

GeneralPage.Global.vid\_serv = CbVidServ.Text;

}

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

{

//получаем текст из выпадающего списка (вид услуги)

GeneralPage.Global.vid\_serv = CbVidServ.Text;

//отображаем данные выпадающего списка типов услуг в зависимости от вида

CbTypeServ.ItemsSource = NevaTelecom1Entities.GetContext().TypeServis.Where(p => p.name\_vid == GeneralPage.Global.vid\_serv).ToList();

}

//обработчик при нажатии на кнопку сохранить

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

{

//контейнерошибок

StringBuilder error = new StringBuilder();

//присваиваемданные

\_curentOrder.num\_order = num\_order.Text;

//получениесегодняшнейдаты

\_curentOrder.data\_create = DateTime.Now;

\_curentOrder.id\_abon = GeneralPage.Global.id\_abonenta;

//проверка данных (если не введены)

if (\_curentOrder.Servi == null)

error.AppendLine("Выбeритеуслугу");

if (\_curentOrder.VidServi == null)

error.AppendLine("Выбeритевидуслуги");

if (\_curentOrder.TypeServi == null)

error.AppendLine("Выбeритетипуслуги");

if (\_curentOrder.Status == null)

error.AppendLine("Выбeритестатусзаявки");

if (\_curentOrder.TypeProblem == null)

error.AppendLine("Выбeритетиппроблемы");

if (\_curentOrder.Manager == null)

error.AppendLine("Выбeритесотрудника");

if (CbStatus.SelectedIndex == 0)

{

date\_end.Text = DateTime.Now.ToShortDateString();

\_curentOrder.date\_end = DateTime.Now;

}

if (CbStatus.SelectedIndex != 0)

{

\_curentOrder.date\_end = null;

date\_end.Text = null;

}

//если в контейнере есть ошибки то выводим их

if (error.Length > 0)

{

MessageBox.Show(error.ToString());

return;

}

//добавляем данные объекта в таблицу бд

if (\_curentOrder.id\_order == 0)

NevaTelecom1Entities.GetContext().Orders.Add(\_curentOrder);

//конструктор для отлавливания системных ошибок (не было, сбоя системы)

try

{

//сохраняемизменения

NevaTelecom1Entities.GetContext().SaveChanges();

Navigation1.MainFrame1.Navigate(new CRMPage());//преходнастрсзаявками

MessageBox.Show("Заявкасохранена");

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());//выводсистемныхошибок

}

}

Код логики страницы OrderMore.xaml:

private Order \_curentOrder = new Order();

public OrderMore(Order selectedOrder)

{

InitializeComponent();

//если переданный объект не пустой, значит происходит редактирование

if (selectedOrder != null)

//присваиваем \_curentOrder переданныеданные

\_curentOrder = selectedOrder;

//привязываем полученные данные с полями

DataContext = \_curentOrder;

}

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

{

Navigation1.MainFrame1.Navigate(new CRMPage());

}

# ЗАКЛЮЧЕНИЕ

ООО «Телеком Нева Связь» представлен в Адмиралтейском, Петроградском и Василеостровском районах Санкт-Петербурга – районах с плотной городской застройкой и превалирующим количеством жилых домов с толщиной стен от 80 см (1800-1930 годы строительства). В связи с этим оператор работает с несколькими технологиями: xDSL,  PON.

В связи с резким ростом количества абонентов в 2020 и 2021 годах оператор принял решении о реализации проекта цифровой трансформации компании. Одним из этапов трансформации является внедрение в работу компании бизнес систем.

В связи с тем, что «коробочные» бизнес решения, существующие на рынке, достаточно дорогие и иногда избыточны по функционалу, руководством компании было принято решение о написании собственной системы по управлению сетью связи.

В ходе выполнения дипломной работы были выполнены следующие задачи: на основании анализа требований заказчика к информационной системе, были составлены основные разделы технического задания, а также в соответствии с техническим заданием было выполнено проектирование и разработка информационной системы в выбранной среде программирования.

Выполняя дипломнуюю работу, стало понятно, что работа каждого предприятия индивидуальна, и каждая информационная система должна быть приспособлена под организацию и должна упрощать работу. Поэтому разработка ИС очень важна и интересна.

Таким образом, цель работы достигнута, поставленные задачи выполнены.

# СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

Литература

1. **Александр Леоненков**Самоучитель UML 2. — [БХВ-Петербург](https://www.ozon.ru/publisher/bhv-peterburg-1098685/): 2020 — 576 с.
2. **МаркДж. Прайс** C# 7.1 and .NET Core 2.0: Modern Cross-Platform Development. — [Питер](https://www.ozon.ru/publisher/piter-856134/): 2021 — 440 с.
3. **Мюллер Д. П.** C# для чайников. — [Диалектика-Вильямс](https://www.ozon.ru/publisher/dialektika-vilyams-135585305/): 2019 — 608 с.
4. **Адам Натан**WPF 4: Unleashed. — [Символ-Плюс](https://www.ozon.ru/publisher/simvol-plyus-856498/): 2021 — 880 с.

Интернет-ресурсы

1. **Itproger Изучение C# WPF на практике**: [Электронный ресурс]. —https://itproger.com/course/csharp-wpf (дата обращения: 02.05.2022)
2. **METANIT.COM:** [Электронный ресурс]. — https://metanit.com/sharp/wpf/ (дата обращения: 15.05.2022)
3. **Microsoft Пошаговое руководство.**: [Электронный ресурс]. —https://docs.microsoft.com/ru-ru/dotnet/desktop/winforms/advanced/walkthrough-creating-new-wpf-content-on-windows-forms-at-design-time?view=netframeworkdesktop-4.8 (дата обращения: 11.05.2022)
4. **Worldskills Программные решения для бизнеса.**: [Электронный ресурс]. —https://nationalteam.worldskills.ru/skills/programmnye-resheniya-dlya-biznesa/ (дата обращения: 17.05.2022)