МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования

Гомельский государственный технический университет имени П.О.Сухого

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

специальность 1-40 05 01-01 Информационные системы и технологии

(в проектировании и производстве)

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовому проекту

по дисциплине «Разработка приложений баз данных для информационных систем»

на тему «Разработка *web*-приложения баз данных «Интернет-провайдер»»

Исполнитель: студент гр. ИТП-31

Пронуза М.Ю.

Руководитель: ассистент

Малиновский И.Л.

Дата проверки: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата допуска к защите: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата защиты: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оценка работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подписи членов комиссии

по защите курсового проекта: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Гомель 2024

**СОДЕРЖАНИЕ**

Введение 3

1 Логическая и физическая структура базы данных интернет-провайдера 5

1.1 Информационно-логическая модель приложения базы данных интернет-провайдера 5

1.2 Нормализация таблиц 8

1.3 Физическая структура базы данных 9

2 Аппаратное и программное обеспечение информационной системы интернет-провайдера 12

3 Структура приложения базы данных интернет-провайдера 14

3.1 Ядро приложения 14

3.2 Инфраструктура 15

3.3 *Web*-интерфейс 16

3.3 Тесты 17

4 Руководство пользователя 20

4.1 Перечень и описание всех выполняемых функций 20

4.2 Рекомендации по освоению 23

5 Руководство программиста 25

Заключение 27

Список использованных источников 28

Приложение А Листинг программы 29

Приложение Б Листинг модульных тестов 77

Приложение В Функциональная схема приложения 95

**ВВЕДЕНИЕ**

В современную эпоху цифровизации и стремительного развития информационных технологий создание *web*-приложений становится основой для построения эффективных информационных систем. Эти приложения предоставляют пользователям удобный доступ к данным, способствуют автоматизации рутинных процессов и обеспечивают высокую степень интеграции различных компонентов инфраструктуры. В условиях глобализации информационных потоков *web-*приложения позволяют компаниям обмениваться данными в реальном времени, снижать издержки и повышать точность обработки информации, что значительно улучшает производительность.

Особенно актуальны такие решения в сфере телекоммуникационных услуг, где обработка и хранение данных требуют строгого соблюдения стандартов безопасности и надёжности. Интернет-провайдеры и телекоммуникационные компании ежедневно сталкиваются с огромным количеством информации, включающей данные о клиентах, тарифных планах, техническом обслуживании и мониторинге качества предоставляемых услуг. Эти данные необходимо не только обрабатывать быстро, но и надёжно защищать от несанкционированного доступа. Эффективное управление данными становится неотъемлемой частью успешного функционирования таких компаний.

Для успешного внедрения подобных решений требуется разработка интуитивно понятных интерфейсов, которые обеспечат лёгкость использования систем всеми категориями сотрудников. Это особенно важно в телекоммуникационной отрасли, где сотрудники могут обладать разным уровнем технической подготовки. Простота взаимодействия с системой способствует снижению времени на обучение персонала и минимизации операционных ошибок.

Помимо обеспечения обработки и хранения больших объёмов информации, современные *web*-приложения предоставляют возможность многопользовательского доступа с разграничением прав и полномочий. Это позволяет каждому сотруднику компании эффективно выполнять свои задачи в рамках должностных обязанностей, минимизируя риски утечки данных и исключая доступ к неактуальной или лишней информации. Такой подход не только укрепляет безопасность, но и улучшает организацию рабочих процессов, что способствует общему повышению производительности.

Более того, гибкость и адаптивность *web*-приложений делают их незаменимыми в условиях постоянных изменений рынка. Возможность быстрой модификации функционала и интеграции новых модулей позволяет компаниям оставаться конкурентоспособными, оперативно реагируя на изменения спроса и внедряя инновации. Например, система управления тарифами может быть быстро обновлена в соответствии с новыми предложениями, а интерфейс для клиентов – улучшен на основании их отзывов.

Таким образом, разработка специализированного *web*-приложения для телекоммуникационной компании является ключевым шагом на пути к оптимизации процессов, повышения качества обслуживания клиентов и увеличения эффективности работы сотрудников. Этот процесс позволяет не только сократить затраты, но и создать основу для долгосрочного роста и развития.

**1 ЛОГИЧЕСКАЯ И ФИЗИЧЕСКАЯ СТРУКТУРА БАЗЫ ДАННЫХ**

**1.1 Информационно-логическая модель приложения базы данных интернет-провайдера**

Информационно-логическая модель системы обработки данных в телекоммуникационной компании позволяет выделить ключевые сущности, однозначность определения которых является фундаментом для разработки базы данных. Каждая из сущностей играет важную роль в обеспечении целостности и функциональности базы данных, благодаря чему достигается способность эффективного управления информацией об услугах оператора.

Сущность "Тарифные планы" представляет собой комплекс атрибутов, описывающих различные параметры тарифов, предлагаемых оператором мобильной связи. Она включает в себя такие важные характеристики, как название тарифного плана (*TariffName*), что позволяет идентифицировать и различать различные предложения компании. Абонентская плата (*SubscriptionFee*) определяет фиксированную стоимость обслуживания, которую клиент оплачивает регулярно. Стоимость минуты разговора при местной, междугородней и международной связи (*LocalCallRate*, *LongDistanceCallRate*, *InternationalCallRate*) предоставляет подробную информацию о тарифах на различные виды звонков, что важно для прозрачности и расчета расходов абонентов. Атрибут *IsPerSecond* указывает тип тарификации, позволяя системе знать, является ли тарификация поминутной или посекундной, что напрямую влияет на точность расчета стоимости звонков. Дополнительно, стоимость *SMS* и *MMS* сообщений (*SmsRate*, *MmsRate*), а также стоимость передачи 1 МБ данных (*DataRatePerMB*) позволяют абонентам и системе оценивать расходы на текстовые сообщения и использование мобильного интернета. Такое детальное описание тарифных планов обеспечивает гибкость в управлении предложениями компании, позволяет быстро адаптироваться к изменениям рынка и предпочтениям клиентов, а также обеспечивает прозрачность и доверие со стороны абонентов.

Сущность "Абоненты" содержит критически важные персональные данные клиентов, необходимые для их идентификации и обеспечения юридической чистоты сделок. Ключевыми атрибутами здесь являются полное имя абонента (*FullName*), которое используется для официальной идентификации клиента во всех документах и взаимодействиях. Домашний адрес (*HomeAddress*) предоставляет информацию о местоположении клиента, что может быть важно для обслуживания, маркетинговых кампаний и соответствия законодательным требованиям. Паспортные данные (*PassportData*) являются обязательными для подтверждения личности клиента и предотвращения мошенничества, а также соответствуют требованиям законодательства в сфере телекоммуникаций и персональных данных. Наличие этих данных в системе обеспечивает возможность индивидуального подхода к каждому клиенту, способствует улучшению качества обслуживания и укрепляет доверие между компанией и абонентами.

Сущность "Договора на оказание услуг" служит для фиксации юридически значимых договорных отношений между оператором и абонентом. В этой сущности содержатся такие атрибуты, как дата заключения договора (*ContractDate*), что важно для отслеживания срока действия соглашения и истории взаимодействия с клиентом. Дата окончания договора (*ContractEndDate*), если она применима, позволяет управлять продлениями и завершениями контрактов. Присвоенный абоненту телефонный номер (*PhoneNumber*) является уникальным идентификатором услуги, предоставляемой клиенту, и важен для учета потребления услуг и выставления счетов. Связь с сущностями "Абоненты" и "Тарифные планы" через внешние ключи *SubscriberID* и *TariffPlanID* обеспечивает интегрированность данных, позволяет отслеживать, какие абоненты пользуются какими тарифами, и анализировать эффективность различных тарифных предложений. Кроме того, атрибут *StaffID* связывает договор с конкретным сотрудником, оформившим его, что важно для управления персоналом и оценки эффективности работы сотрудников.

Сущность "Сотрудники" включает информацию о персонале компании, ответственном за обслуживание клиентов и оформление договоров. Атрибуты этой сущности, такие как полное имя сотрудника (*FullName*), позволяют идентифицировать каждого работника индивидуально. Ссылка на должность (*PositionID*) из сущности "Должности сотрудников" помогает определить роль и обязанности каждого сотрудника внутри организации, что важно для распределения задач и управления ресурсами. Информация об образовании (*Education*) предоставляет дополнительные сведения о квалификации сотрудника, что может быть полезно для планирования обучения, повышения квалификации и развития персонала. Такая детализация способствует эффективному управлению человеческими ресурсами, повышению качества обслуживания клиентов и достижению стратегических целей компании.

Сущности "Звонки", "Сообщения" и "Использование интернета" отражают подробные данные о потреблении услуг абонентами и являются критически важными для биллинговых систем, аналитики и принятия управленческих решений. В сущности "Звонки" фиксируются данные о дате и времени каждого звонка (*CallDate*), продолжительности разговора в секундах (*CallDuration*) и связи с конкретным договором через *ContractID*. Эта информация позволяет точно рассчитывать стоимость звонков, анализировать нагрузку на сеть и поведение клиентов. Сущность "Сообщения" содержит данные о дате и времени отправки каждого сообщения (*MessageDate*), а также атрибут *IsMMS*, указывающий, является ли сообщение *SMS* или *MMS*, что влияет на тарифы и учет расходов. Сущность "Использование интернета" включает дату и время сессии (*UsageDate*), объем отправленных и полученных данных в мегабайтах (*DataSentMB*, *DataReceivedMB*), что необходимо для расчета стоимости интернет-трафика и анализа потребления данных абонентами. Связь этих сущностей с договором через *ContractID* обеспечивает полную прослеживаемость услуг, потребленных каждым клиентом, позволяет формировать точные счета и предоставляет ценную информацию для маркетингового анализа и улучшения качества услуг.

Ключи в базе данных играют фундаментальную роль в обеспечении целостности, уникальности и согласованности записей. Каждая таблица имеет первичный ключ, такой как *TariffPlanID*, *SubscriberID*, *ContractID*, который уникально идентифицирует каждую запись, предотвращая дублирование и обеспечивая быстрый доступ к данным. Внешние ключи устанавливают связи между таблицами, гарантируя, что связанные записи существуют и корректны. Например, *SubscriberID* в таблице "Договора на оказание услуг" ссылается на *SubscriberID* в таблице "Абоненты", что гарантирует, что каждый договор связан с существующим абонентом. Аналогично, *TariffPlanID* связывает договор с конкретным тарифным планом, а *StaffID* указывает на сотрудника, оформившего договор. Использование ограничений целостности данных и каскадных операций при удалении или обновлении записей способствует поддержанию согласованности данных, предотвращает появление "осиротевших" записей и облегчает администрирование базы данных.

Связи между сущностями в основном являются отношениями "один ко многим", что точно отражает реальные бизнес-процессы и организационную структуру компании. Например, один абонент может иметь несколько договоров, что может происходить в случае использования нескольких услуг или номеров, перехода на новые тарифные планы или обновления условий обслуживания, но каждый договор относится только к одному абоненту. Сотрудник может оформить множество договоров с разными абонентами, что отражает его деятельность и эффективность работы. Такая организация данных позволяет эффективно управлять информацией, минимизировать избыточность, обеспечивать быстрый доступ к необходимым данным и поддерживать высокую производительность системы.

Основные характеристики ключей и связей тщательно продуманы для обеспечения целостности данных и поддержки бизнес-логики системы. Это достигается посредством использования строгих ограничений целостности, индексирования и оптимизации запросов. Система спроектирована таким образом, чтобы предотвратить некорректные или противоречивые данные, обеспечить безопасность и конфиденциальность информации, а также соответствовать всем необходимым законодательным и отраслевым стандартам. Такой подход способствует надежности и устойчивости базы данных в целом, обеспечивает возможность масштабирования и адаптации системы к будущим требованиям и изменениям в бизнес-процессах.

**1.2 Нормализация таблиц**

Нормализация базы данных – это один из важнейших процессов проектирования современных информационных систем, обеспечивающий эффективное управление данными. Её применение позволяет устранить избыточность, минимизировать ошибки при обновлении информации и гарантировать целостность структуры данных. Особенно актуальна нормализация для компаний, работающих с большими объёмами данных, где дублирование и несогласованность информации могут привести к финансовым потерям. Структурированный подход к хранению данных упрощает их обработку, улучшает производительность системы и повышает надёжность операций, что делает нормализацию неотъемлемой частью успешного функционирования информационной системы и, в частности, базы данных.

Проектируемая база данных соответствует первой нормальной форме, в связи с тем, что все таблицы базы данных содержат только атомарные значения. Поля не содержат повторяющихся групп и вложенных данных. Каждая таблица имеет уникальный первичный ключ. Например, в таблице абонентов поле "*FullName*" содержит только одно значение – ФИО абонента, а не список имён. Поля тарифных планов, такие как "*LocalCallRate*" или "*SmsRate*", также содержат конкретные числовые значения. Таблицы не имеют многозначных атрибутов или вложенных структур.

Соответствие второй нормальной форме в том, что база данных полностью удовлетворяет требованиям первой нормальной формы и отсутствуют частичные зависимости, где неключевые атрибуты зависели бы только от части составного первичного ключа. Например, таблица звонков содержит такие атрибуты, как дата и длительность звонка, которые зависят только от уникального идентификатора звонка "*CallID*". В других таблицах данные разделены по сущностям: информация о тарифах, сотрудниках и абонентах хранится в отдельных таблицах и не пересекается.

Третья нормальная форма в базе данных гарантируется тем, что таблицы базы данных соответствуют требованиям второй нормальной формы и не содержат транзитивных зависимостей, при которых неключевые атрибуты зависели бы друг от друга. Например, информация о должностях сотрудников выделена в отдельную таблицу, что позволяет избежать повторения данных о названиях должностей в таблице сотрудников. Атрибуты зависят только от первичного ключа, к которому они относятся.

Для устранения избыточности данных применены внешние ключи, которые реализуют связи между таблицами. Например, таблицы абонентов и договоров связаны через "*SubscriberID*", что позволяет избежать дублирования данных абонентов в таблице договоров. Аналогично, информация о тарифных планах ссылается на таблицу договоров через "*TariffPlanID*".

Соответствие нормализации до третьей нормальной формы обеспечивает минимизацию избыточности данных, целостность и удобство обновления. Это делает базу данных более устойчивой к ошибкам и снижает вероятность аномалий при изменении данных.

**1.3 Физическая структура базы данных**

Физическая структура базы данных разработана с учетом особенностей выбранной системы управления базами данных (СУБД) *Microsoft SQL Server*. База данных представляет собой совокупность файлов, включающих основной файл данных и файл журнала транзакций. Основной файл данных, именуемый "*OperatorDB*.*mdf*", содержит все объекты базы данных, включая таблицы, индексы и хранимые процедуры. Файл журнала транзакций, названный "*OperatorDB\_log.ldf*", хранит историю всех транзакций, что обеспечивает возможность восстановления базы данных в случае сбоев и поддерживает целостность данных.

Типы файлов соответствуют стандартам *SQL Server*: основной файл данных имеет расширение .*mdf*, а файл журнала транзакций – .*ldf*. Расположение файлов обычно задается в каталоге данных *SQL Server* по умолчанию, однако имеется возможность размещать их на различных физических дисках для повышения производительности и надежности системы. Объем основного файла данных изначально установлен в 100 МБ с авторасширением по мере необходимости, что позволяет эффективно управлять дисковым пространством. Объем файла журнала транзакций меньше и составляет 50 МБ, также с возможностью авторасширения.

Состав базы данных включает ряд таблиц, каждая из которых спроектирована для оптимального хранения и доступа к данным в соответствии с требованиями телекоммуникационной среды. В таблице *StaffPosition* хранятся данные о должностях сотрудников. Поле *PositionID* (*INT*, *PRIMARY KEY*, *IDENTITY*) служит уникальным идентификатором каждой должности, обеспечивая быстрый поиск и связь с другими таблицами. Поле *PositionName* (*NVARCHAR(100), NOT NULL*) хранит название должности, используя тип *NVARCHAR* для поддержки международных символов и предотвращения проблем с кодировкой.

Таблица *Staff* содержит информацию о сотрудниках компании. Поле *StaffID (INT, PRIMARY KEY, IDENTITY*) уникально идентифицирует каждого сотрудника. Поле *FullName (NVARCHAR(150), NOT NULL*) хранит полное имя, что необходимо для идентификации и внутренних коммуникаций. Поле *PositionID (INT, NOT NULL*) является внешним ключом, связывающим сотрудника с его должностью в таблице *StaffPosition*. Поле *Education (NVARCHAR(100*)) хранит информацию об образовании сотрудника, что может быть полезно для кадровой политики и планирования обучения.

В таблице *TariffPlans* собраны данные о тарифных планах. Поле *TariffPlanID (INT, PRIMARY KEY, IDENTITY*) служит уникальным идентификатором тарифа. Поля, такие как *TariffName (NVARCHAR(100), NOT NULL), SubscriptionFee, LocalCallRate, LongDistanceCallRate, InternationalCallRate, SmsRate, MmsRate, DataRatePerMB (DECIMAL(10, 2), NOT NULL*), содержат соответствующие тарифные ставки и финансовые параметры. Поле *IsPerSecond (BIT, NOT NULL*) указывает на тип тарификации, где значение 0 соответствует поминутной, а 1 – посекундной тарификации.

Таблица *Subscribers* предназначена для хранения данных об абонентах. Поле *SubscriberID (INT, PRIMARY KEY, IDENTITY*) уникально идентифицирует каждого абонента. Поля *FullName (NVARCHAR(150), NOT NULL), HomeAddress (NVARCHAR(255)), PassportData (NVARCHAR(100))* содержат персональную информацию, необходимую для заключения договоров и предоставления услуг. Поля адреса и паспортных данных допускают *NULL* значения, учитывая возможность неполного предоставления информации абонентом.

Таблица *Contracts* фиксирует договорные отношения между компанией и абонентами. Поле *ContractID (INT, PRIMARY KEY, IDENTITY*) уникально идентифицирует каждый договор. Поля *SubscriberID (INT, NOT NULL)* и *TariffPlanID (INT, NOT NULL)* являются внешними ключами, связывающими договор с конкретным абонентом и тарифным планом соответственно. Поле *ContractDate (DATE, NOT NULL)* хранит дату заключения договора, а *ContractEndDate (DATE)* – дату его окончания, если она установлена. Поле *PhoneNumber (NVARCHAR(20), NOT NULL*) содержит присвоенный абоненту номер телефона. Поле *StaffID (INT, NOT NULL)* связывает договор с сотрудником, оформившим его, что важно для учета работы персонала.

Таблица *Calls* содержит информацию о совершенных звонках. Поле *CallID (INT, PRIMARY KEY, IDENTITY)* уникально для каждого звонка. Поле *ContractID (INT, NOT NULL)* связывает запись с конкретным договором. Поля *CallDate (DATETIME, NOT NULL*) и *CallDuration (INT, NOT NULL)* фиксируют дату, время и продолжительность каждого звонка, что необходимо для биллинга и аналитики.

Таблица *Messages* хранит данные об отправленных сообщениях. Поле *MessageID (INT, PRIMARY KEY, IDENTITY)* уникально идентифицирует каждое сообщение. Поле *ContractID (INT, NOT NULL)* связывает сообщение с договором. Поле *MessageDate (DATETIME, NOT NULL)* фиксирует дату и время отправки, а поле *IsMMS (BIT, NOT NULL)* указывает тип сообщения – *SMS* или *MMS*.

Таблица *InternetUsage* отражает использование интернет-трафика. Поле *UsageID (INT, PRIMARY KEY, IDENTITY)* служит уникальным идентификатором записи. Поле *ContractID (INT, NOT NULL)* связывает использование с конкретным договором. Поля *UsageDate (DATETIME, NOT NULL), DataSentMB* и *DataReceivedMB (DECIMAL(10, 2), NOT NULL)* фиксируют дату использования и объем переданных и полученных данных, что важно для расчета стоимости услуг и анализа поведения абонентов.

Типы данных полей выбраны с учетом оптимального использования ресурсов и обеспечения точности хранения информации. Использование типа *INT* для идентификаторов обеспечивает эффективный доступ и манипуляцию данными. Тип *NVARCHAR* выбран для текстовых полей ввиду необходимости поддержки юникод-символов и предотвращения проблем с локализацией. Для финансовых и количественных показателей использован тип *DECIMAL(10, 2),* позволяющий хранить значения с двумя знаками после запятой, что критично для точных финансовых расчетов.

Все внешние ключи снабжены ограничениями ссылочной целостности (*FOREIGN KEY* с *ON DELETE CASCADE*), что предотвращает появление несвязанных данных и обеспечивает согласованность между таблицами. Это особенно важно для операций удаления, где связанные записи автоматически удаляются, сохраняя целостность базы данных.

Оптимальное распределение данных и продуманная физическая структура базы данных способствуют повышению производительности системы, обеспечивая быстрый доступ к данным и эффективное выполнение запросов. Такое решение также упрощает задачи резервного копирования и восстановления, а также масштабирование системы при увеличении объема данных.

**2 АППАРАТНОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ**

Для обеспечения работы информационной системы определяются требования к аппаратному и программному обеспечению как на стороне серверов, так и на стороне клиента. Серверная часть отвечает за выполнение операций с базой данных, обработку бизнес-логики и предоставление веб-интерфейса пользователям. Клиентская часть предоставляет доступ к системе через веб-браузер и работает на пользовательских устройствах.

Серверная часть системы требует высокопроизводительного оборудования. Для работы в условиях небольшой нагрузки (до 1000 пользователей) рекомендуется виртуальная машина или физический сервер с процессором от 4 ядер и тактовой частотой не менее 2.5 ГГц. Для более интенсивной работы следует использовать сервер с 8 ядрами или более. Оперативная память должна составлять минимум 16 ГБ для базовых условий и 32–64 ГБ для высокой нагрузки. Хранилище данных должно быть основано на *SSD*-дисках с объёмом от 512 ГБ и выше, желательно с использованием *RAID*-массивов для обеспечения отказоустойчивости. Сетевые карты должны обеспечивать пропускную способность не менее 1 Гбит/с для быстрого обмена данными.

В качестве операционной системы сервера используется *Windows Server* версии 2019 или 2022, что обеспечивает совместимость с платформой .*NET* и *Microsoft SQL Server*. Альтернативно возможно использование *Linux* (например, *Ubuntu* 22.04 *LTS*). Управление базой данных осуществляется через *Microsoft SQL Server* версии 2019 или выше. Минимальные требования для *SQL Server* включают 4 ГБ оперативной памяти для базового функционирования, однако рекомендуемые требования не менее 8 ГБ для стабильной работы. Требуемое дисковое пространство для *SQL Server* составляет около 6 ГБ для установки и минимум 200 ГБ для хранения данных в условиях стандартной нагрузки.

Для предоставления веб-интерфейса используется веб-сервер *IIS (Internet Information Services*) в среде *Windows Server*. *IIS* позволяет запускать приложения, созданные на базе *ASP.NET Core*, и управлять безопасностью соединений. На *Linux* в качестве веб-сервера можно использовать *Nginx*, который обеспечивает аналогичную функциональность. Для запуска веб-приложения требуется платформа .*NET Core* версии 6.0 или выше, а для работы с данными используется *Entity Framework Core*. Приложение поддерживает безопасное соединение *HTTPS*, для чего требуется настройка *SSL*-сертификата.

Клиентская часть системы ориентирована на работу с любыми современными устройствами, такими как настольные компьютеры, ноутбуки, планшеты или смартфоны. Оборудование пользователя должно соответствовать минимальным требованиям: процессор с двумя ядрами и тактовой частотой от 1.8 ГГц, оперативная память от 4 ГБ. Устройства могут работать под управлением операционных систем *Windows* 10/11, *macOS*, *Android* или *iOS*. Для доступа к системе используются современные браузеры: *Google Chrome, Mozilla Firefox, Microsoft Edge* или *Safari*. Все указанные браузеры поддерживают последние версии *HTML5* и обеспечивают корректное отображение веб-интерфейса.

Программное окружение сервера начинается с установки операционной системы. На сервер устанавливается веб-сервер (*IIS* или *Nginx*), создаётся пул приложений, указывается каталог для файлов веб-приложения, а также настраивается *SSL*-сертификат для шифрования соединения. После этого производится установка СУБД, создание базы данных и настройка её параметров. Для доступа к базе данных задаются права пользователей, конфигурируются параметры подключения и резервное копирование. Затем переносятся файлы приложения на сервер, настраивает конфигурационные файлы приложения, такие как *appsettings.json*, для указания параметров подключения к базе данных, а также проверяет работоспособность системы через веб-браузер.

При развертывании на облачных платформах, таких как *Azure* или *AWS*, действия системного администратора включают настройку виртуальных машин, развёртывание приложения через *App Service* (в случае *Azure*), настройку базы данных, подключение доменного имени и добавление *SSL*-сертификата для шифрования. На арендуемых серверах выполняются аналогичные шаги, с учётом ограничений, предоставляемых хостинг-провайдером.

Такое сочетание оборудования, программного обеспечения и настроек обеспечивает надёжную работу серверной части системы и удобный доступ для пользователей через веб-браузеры на клиентских устройствах. Система будет работать стабильно даже при увеличении нагрузки за счёт использования современных технологий и масштабируемой архитектуры.

**3 СТРУКТУРА ПРИЛОЖЕНИЯ БАЗЫ ДАННЫХ ИНТЕРНЕТ-ПРОВАЙДЕРА**

**3.1 Ядро приложения**

Ядро приложения представлено совокупностью модельных классов, отражающих сущности предметной области, а также кодом, обеспечивающим их использование для расчета и представления данных. В основе лежит контекст данных *TelecomDbContext*, связанный с *Microsoft SQL Server*, что позволяет связывать объекты предметной области на таблицы базы данных. Модельные классы, такие как *Subscriber, TariffPlan, Contract, Call, Message, InternetUsage, Staff, StaffPosition*, спроектированы для точного соответствия схемам хранения данных и логике предметной области телекоммуникационной компании. Каждый класс содержит поля, отражающие структурно и семантически значимые характеристики объектов: *Subscriber* определен для хранения персональных данных абонентов, включая ФИО, адрес и паспортную информацию; *TariffPlan* задает параметры тарифов, таких как стоимость различных типов звонков, *SMS, MMS*, а также передачи данных; *Contract* фиксирует информацию о договорных отношениях между абонентом и компанией, включая привязку к выбранному тарифу, телефонному номеру и сотруднику, оформившему договор; *Call, Message* и *InternetUsage* содержат временные метки и количественные показатели, отражающие фактическое использование услуг, необходимых для последующего тарифного расчета; *Staff* и *StaffPosition* определяют организационную структуру кадрового ресурса оператора, связывая сотрудников с их должностями [1].

Все модельные классы связаны через первичные и внешние ключи, обеспечивая целостность данных и формируя иерархию отношений "один ко многим" и "один к одному", характерную для телекоммуникационных сервисов. При обращении к базе данных через *TelecomDbContext* используются *LINQ*-запросы и механизмы объектно-реляционного отображения для извлечения, фильтрации и модификации данных без прямого написания *SQL*-кода. Бизнес-логика сосредоточена на обработке сведений о пользователях и их активности, применяя тарифные ставки и параметры тарифных планов к фактическим данным о звонках, сообщениях и интернет-трафике. Логика расчета основана на параметрах тарифов, длительности звонков и объемах переданных данных, а также на факте использования *SMS* и *MMS*, что позволяет определять итоговую стоимость обслуживания. Для вычислений и проверки корректности данных предусмотрен четкий набор свойств и типов данных, учитывающих необходимость точных финансовых расчетов и поддержки различных форматов дат и числовых значений [2]. За счет строго определенных свойств модельных классов упрощается реализация методов бизнес-логики, которые приводят к формированию итоговых показателей, агрегированной статистики и возможности предоставлять систему аналитических отчётов. Таким образом, ядро приложения состоит из набора модельных классов, тесно интегрированных с логикой расчета стоимости и анализа использования услуг, опираясь на строгую структуру данных и механизмы *ORM* фреймворка для обеспечения корректного и предсказуемого поведения системы.

**3.2 Инфраструктура**

Архитектура приложения построена с использованием паттерна проектирования MVC (Model-View-Controller), который является ключевым принципом в создании структурированных и масштабируемых приложений. Паттерн MVC обеспечивает разделение ответственности между логикой данных, пользовательским интерфейсом и обработкой пользовательских событий. В рамках данного проекта модель включает в себя классы сущностей, управляемые Entity Framework Core, представление реализуется через интерфейсы и веб-страницы, а контроллеры отвечают за обработку запросов и вызов соответствующих операций модели. Такой подход не только упрощает разработку и тестирование приложения, но и повышает его устойчивость к изменениям, позволяя гибко адаптироваться под изменяющиеся требования бизнеса.

Entity Framework Core играет важную роль в обеспечении взаимодействия между моделью предметной области и реляционной базой данных, функционирующей под управлением Microsoft SQL Server. Центральным элементом этой технологии является класс контекста данных TelecomDbContext. Он предоставляет удобный интерфейс для выполнения операций извлечения, обновления и удаления данных без необходимости написания сложных SQL-запросов.

Класс контекста конфигурируется при инициализации приложения посредством его регистрации в сервисном контейнере, что позволяет динамически использовать объекты DbContext во время выполнения. Это решение обеспечивает не только удобство работы с данными, но и гибкость настройки системы. При добавлении новых сущностей или изменении их свойств разработчики могут использовать механизм миграций EF Core, который автоматически обновляет структуру базы данных в соответствии с изменениями в модели. Такой подход значительно упрощает процесс сопровождения и развития приложения, позволяя быстро и безопасно вносить изменения в схему данных и гарантировать согласованность между моделью и базой данных. Миграции можно выполнять с помощью встроенных инструментов EF Core, что упрощает контроль версионности и управление структурными изменениями базы данных, обеспечивая поддержку долгосрочного развития проекта.

Особенностью данного проекта является отсутствие явного репозиторного слоя. Доступ к данным осуществляется напрямую через DbContext и его наборы сущностей (DbSet). Взаимодействие с ними производится на уровне контроллеров и сервисов, которые используют возможности ORM для инкапсуляции низкоуровневого взаимодействия с базой данных. Такой подход позволяет сосредоточиться на реализации бизнес-логики и использовании агрегированных операций, избегая лишних технических сложностей, связанных с формированием SQL-запросов, управлением соединениями и обработкой транзакций. Инфраструктура EF Core берет на себя все эти задачи, включая обеспечение целостности данных.

Совместное использование MVC и EF Core создаёт мощный каркас для разработки приложения, объединяя структурированность, модульность и гибкость. Это позволяет обеспечить высокую производительность, безопасность данных и удобство работы с системой как для разработчиков, так и для конечных пользователей.

**3.3 *Web*-интерфейс**

В слое web-интерфейса используется инфраструктура *ASP.NET Core MVC*, включающая контроллеры, представления (*Views*) и дополнительные *Razor Pages*. В файле *Program.cs* настраивается подключение к контексту данных *TelecomDbContext*, интеграция *ASP.NET Identity* для аутентификации и авторизации, поддержка кэширования, сессий, а также запуск механизма инициализации базы данных [5]. Внедрение зависимостей (*Dependency Injection*) обеспечивает доступ к сервисам и контексту данных во всех частях приложения, упрощая логику взаимодействия с базой данных и авторизационными механизмами. Для управления ролями и пользователями применяется класс *IdentitySeedData*, что позволяет при старте автоматически создавать учетные записи и роли.

Контроллеры, размещенные в пространстве имен *Controllers* и *Controllers.tables*, отражают логику предметной области и соответствуют ключевым сущностям базы данных. *CallsController* отвечает за операции с данными о звонках, *ContractsController* о договорах, *InternetUsagesController* о трафике, *MessagesController* о сообщениях, *StaffPositionsController* и *StaffsController* о должностях и сотрудниках, *SubscribersController* о данных абонентов, *TariffPlansController* о тарифных планах. Каждый контроллер обрабатывает *HTTP*-запросы, выполняет выборку, обновление и удаление данных с помощью контекста *TelecomDbContext*, а при необходимости накладывает ограничения доступа с помощью атрибутов [*Authorize*] [6]. Такой подход позволяет разделить ответственность между разными контроллерами, сделать код более понятным и упростить дальнейшее сопровождение приложения.

Представления (*Views*), связанные с контроллерами, используют *Razor* для динамической генерации *HTML*-кода и отображения данных, извлеченных из базы. В них реализована поддержка пагинации, фильтрации и сортировки, что делает взаимодействие с большими наборами данных удобным. Типовые представления, такие как *index*, *details, edit, create* и *delete*, обеспечивают все необходимые операции *CRUD* для сущностей, а технология *Razor Pages* применяется для упрощения типичных сценариев аутентификации и управления учетными записями пользователей. Такой симбиоз контроллеров, представлений и *Razor Pages* формирует понятный и расширяемый *web*-интерфейс.

Для автоматической подготовки базы данных к работе используется специальный инициализатор *DbInitializer* и промежуточное ПО *DbInitializerMiddleware*, зарегистрированное в конвейере обработки запросов. При старте приложения этот механизм проверяет состояние базы данных и при необходимости загружает исходные данные из *SQL*-скриптов. Это упрощает процесс развертывания на новых серверах или после очистки базы данных, минимизируя ручные действия и исключая риск пропуска важных этапов настройки окружения.

Вся система *web*-интерфейса, включая контроллеры, представления, *Razor Pages*, механизмы аутентификации, авторизации и инициализации базы данных, взаимодействует через единый инфраструктурный слой. Это обеспечивает целостность архитектуры, предсказуемость поведения и удобство развития приложения, позволяя добавлять новые возможности или изменять логику работы без нарушения принципов модульности и сопровождаемости кода.

**3.4 Тесты**

Модульные и интеграционные тесты играют ключевую роль в поддержании стабильности и предсказуемости поведения системы, особенно в условиях постоянного расширения функциональности и усложнения логики предметной области. В данном случае реализован отдельный проект тестирования под названием *APITest*, в котором сосредоточены проверки корректности работы контроллеров, методов обработки данных и результатов взаимодействия с базой. Подобный подход повышает уверенность в том, что любые изменения или обновления кода не нарушат ожидаемое поведение приложения, а также обеспечивает удобство при обнаружении и локализации проблем.

В качестве примера можно привести тесты, нацеленные на проверку корректности работы *SubscribersController*, где модульные тесты сосредоточены на ключевых операциях *CRUD*, осуществляемых при взаимодействии с сущностью *Subscriber*. Реализация подобных тестов опирается на инструменты, позволяющие максимально приблизить условия исполнения проверяемого кода к реальной работе приложения, но при этом сохранить полный контроль над средой. Для этого используется встроенная возможность конфигурации *DbContext* на основе *InMemoryDatabase*, что избавляет от необходимости обращаться к реальному хранилищу и обеспечивает детерминированное и повторяемое поведение тестов, упрощая их запуск в любых условиях. Тестовый код, написанный с использованием фреймворка *xUnit*, создает изолированное окружение для каждого набора проверок, гарантируя чистоту эксперимента и отсутствие побочных эффектов от предыдущих тестов.

В процессе тестирования проверяются результаты вызовов методов контроллера, связанных с отображением списков абонентов, деталей конкретного абонента, а также функций создания, редактирования и удаления записей в базе. Например, тестовые методы убеждаются, что метод *Index* корректно возвращает представление с постраничным списком *Subscriber*-объектов, а метод *Details* корректно обрабатывает как валидный идентификатор абонента, предоставляя доступ к нужным данным, так и ситуации, когда идентификатор отсутствует или невалиден, возвращая ожидаемые результаты (например, *NotFoundResult*). Аналогичным образом проверяется, что методы *Create, Edit* и *Delete* реагируют предсказуемо: при корректных входных данных перенаправляют на нужные представления или адреса и фактически изменяют состояние базы данных (в случае *InMemoryDatabase* – виртуального хранилища), а при ошибочных условиях возвращают представления с моделью, позволяя определить и исправить проблемы на уровне валидации данных или обработки логики.

Особое внимание уделяется поддержанию целостности и согласованности данных: тесты, выполняющие операции изменения состояния, после своего завершения очищают *InMemoryDatabase*, используя методы вроде *Dispose*, чтобы гарантировать отсутствие остаточных данных, способных повлиять на результаты последующих тестовых сценариев. Это обеспечивает удобство проведения целых батчей тестов последовательно или параллельно, исключая непредвиденные конфликты. Систематическое применение подобных методик способствует максимальной прозрачности процесса тестирования и упрощает анализ результатов: при любом сбое можно быть уверенным, что причина кроется именно в проверяемом участке логики, а не в побочных фактора.

Интеграционные тесты, дополняя модульные проверки, могут распространяться на более широкий контекст, включающий не только контроллеры и данные, но и механизмы авторизации, взаимодействие с внешними сервисами или сетевыми вызовами. При необходимости они могут разворачиваться в приближенном к "боевому" окружении, используя специальные средства тестового хоста и *HTTP*-клиента для моделирования реальной работы приложения. В сочетании с модульными тестами, интеграционные проверки формируют многоуровневую систему контроля качества, удерживающую приложение в состоянии стабильного развития, где каждая новая функция проходит проверку на совместимость с уже существующими компонентами.

Таким образом, наличие проекта *APITest*, использование *InMemoryDatabase* для имитации состояния хранилища, проверка работы контроллеров через тестовые методы, написанные с помощью *xUnit*, а также применение стратегий очистки и изоляции данных отражают комплексный и продуманный подход к тестированию [7]. Данная стратегия интегрируется в общую структуру приложения, повышая общую надежность и обеспечивая предсказуемость поведения всей системы на протяжении всего жизненного цикла разработки и внедрения новых возможностей.

**4 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

**4.1 Перечень и описание всех выполняемых функций**

Приложение предоставляет широкий набор функций для работы с данными о тарифных планах, абонентах, договорах на оказание услуг, использовании услуг связи, кадровом составе организации и статистикой потребления услуг за определенные периоды. Эти функции обеспечивают полный цикл управления записями, включая их создание, просмотр, корректировку и удаление, а также позволяют просматривать и анализировать ключевые показатели обслуживания абонентов, сведения о тарифных планах и детализацию расходов.

Система управления тарифными планами позволяет вносить новые тарифы, указывать их название, абонентскую плату, стоимость звонков (местных, междугородних, международных), а также стоимость текстовых (*SMS*) и мультимедийных (*MMS*) сообщений и тарификацию интернет-трафика. Предусмотрена возможность редактировать тарифные планы при изменении экономических условий или маркетинговой стратегии, удалять устаревшие тарифы и детально просматривать параметры уже существующих планов. Все изменения немедленно отражаются в базе данных, что дает актуальную информацию для формирования счетов и анализа экономической эффективности. Панель сведений о тарифных планах представлена на рисунке 4.1.

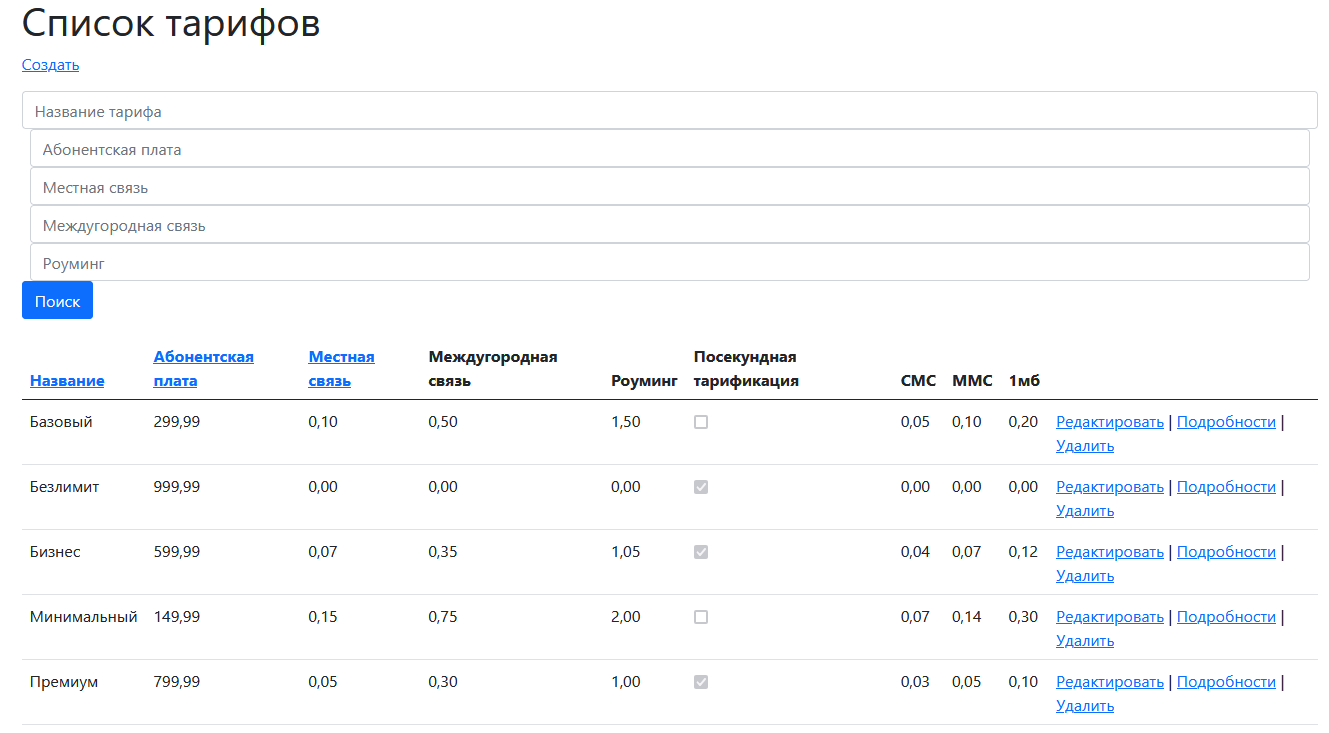


Рисунок 4.1 – Список тарифных планов

Работа с данными о клиентах (абонентах) и их договорах предоставляет функции добавления новых абонентов, ввода их персональных данных (ФИО, адрес, паспортные данные), а также заключения договоров, привязанных к выбранному тарифному плану. Есть возможность указывать дату заключения договора, телефонный номер, оформленный для абонента, и сотрудника, ответственного за оформление. Предусмотрены операции по редактированию записей, чтобы обновлять устаревшие данные – например, изменения в адресе или замена тарифного плана, – а также удалять договоры, завершившие свое действие. При необходимости доступен просмотр всех договоров, заключенных за определенный период, выделение количества абонентов, подключившихся в заданный временной интервал, и формирование статистики по востребованности конкретных тарифных планов. Детализация счетов абонента представленна на рисунке 4.2. Фильтрация абонентов по дате подключения первого тарифа представлена на рисунке 4.3.

Для анализа фактического потребления услуг предусмотрены функции отображения и подсчета статистики звонков, сообщений и интернет-трафика. Можно просматривать длительность конкретных телефонных разговоров, количество отправленных *SMS* и *MMS*, объем переданных данных. Данные о потреблении услуг служат основой для формирования конечной стоимости обслуживания абонентов. Приложение предоставляет механизмы для детализации расходов за указанный период, позволяя получить наглядное представление об объеме и стоимости потребленных услуг, оценить суммарные затраты абонента за месяц, включая абонентскую плату и расходы на трафик, звонки и сообщения. Эти данные могут быть использованы как для выставления счетов, так и для анализа пользовательских предпочтений.

Функции кадрового учета позволяют управлять информацией о сотрудниках и их должностях. Можно вносить новые должности, описывать их наименования, добавлять сотрудников, указав их ФИО и уровень образования, а также связывать работника с определенной должностью. Система предоставляет механизмы просмотра списков персонала, фильтрации и выборки по признакам образования или должности, что облегчает работу отдела кадров при планировании персонала, аналитике внутренней структуры компании или подготовке управленческой отчетности. Интерфейс страницы кадрового учета представлен на рисунке 4.4.

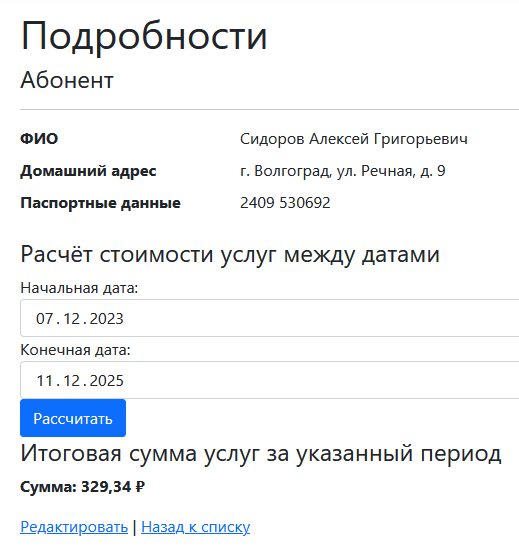


Рисунок 4.2 – Детализация счета

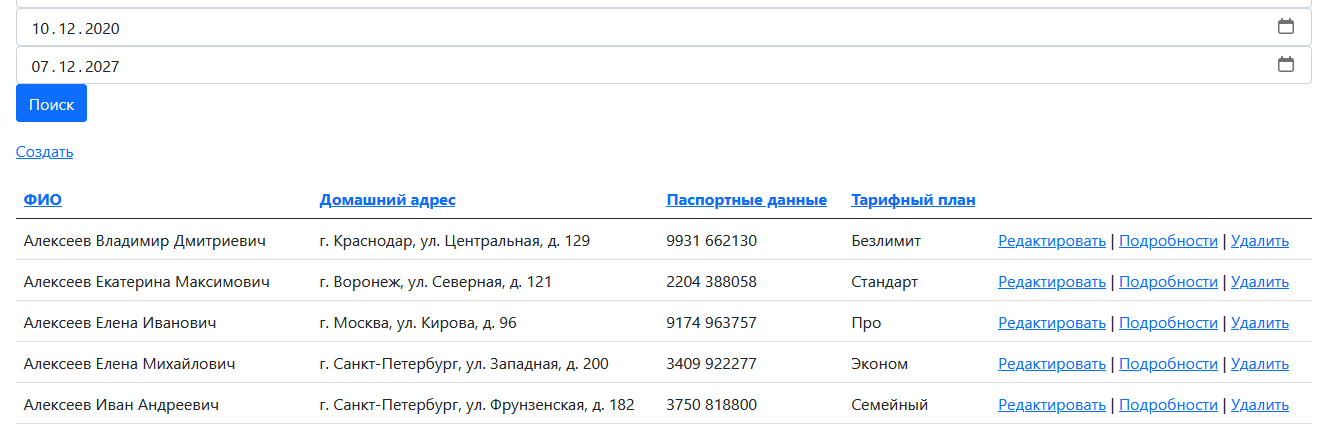


Рисунок 4.3 – Фильтрация абонентов по дате подключения тарифа

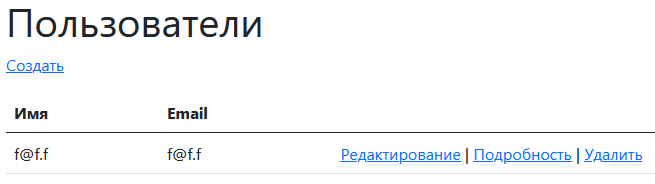


Рисунок 4.4 – Страница кадрового учета

Таким образом, приложение обеспечивает комплексный набор функций для манипуляций с данными – от добавления и обновления записей до формирования сводных отчетов и аналитики. Любые внесенные изменения сохраняются в базе данных, доступны для последующей обработки и отображения, а предоставляемые средства фильтрации, сортировки и детализации информации помогают своевременно анализировать и оценивать состояние телекоммуникационных услуг, кадрового состава и экономических показателей, связанных с предоставлением услуг связи.

**4.2 Рекомендации по освоению**

В процессе эксплуатации веб-приложения осуществляется доступ к функциональности посредством стандартного веб-браузера и предварительно предоставленных учетных данных для аутентификации. Форма аутентификации представлена на рисунке 4.5. После успешного входа открываются основные разделы с учетом выданных прав доступа и ролей, что позволяет работать с сущностями предметной области в удобном графическом интерфейсе. Создание контракта представлено на рисунке 4.6. Предусмотрена возможность создания новых записей, таких как тарифные планы, для которых указываются наименования, абонентские платы, параметры тарификации звонков различного уровня, стоимости *SMS*, *MMS* и передачи данных. Аналогичным образом реализованы операции добавления сведений об абонентах с фиксацией их ФИО, домашнего адреса, паспортных данных и инициации договоров, связанных с конкретными тарифными планами, телефонными номерами и персоналом, оформившим договор. В режиме просмотра списка абонентов или договоров достигается удобство фильтрации, сортировки и пагинации, что облегчает навигацию даже при большом объеме данных. При необходимости изменения записей открываются формы редактирования, позволяющие обновлять и корректировать ранее сохраненные данные, сохранять их, а при утрате актуальности или необходимости удаления отдельных позиций применяются средства удаления. Отдельная часть функционала предназначена для анализа статистики предоставленных услуг: возможно отслеживание продолжительности звонков, количества отправленных сообщений, объемов переданного интернет-трафика, а также формирование сводных отчетов о расходах абонента за определенные периоды, включая расчет стоимости звонков, сообщений, данных и абонентской платы. Панели управления и анализа предоставления услуг телефонии, СМС/ММС связи и интернет трафика представлены на рисунках 4.7-4.9. Предусмотрен доступ к кадровым сведениям, включающим данные о сотрудниках и их должностях, что обеспечивает выборку персонала по критериям должности или уровня образования. При работе с этими инструментами различным подразделениям предоставляется возможность оперативно оценивать состояние клиентской базы, уточнять условия тарифов, контролировать использование услуг, оформлять новые договоры и обновлять сведения об абонентах или персонале. Запуск приложения осуществляется путем обращения к заданному сетевому адресу в браузере, после чего вводятся учетные данные для авторизации; при успешной проверке становятся доступны все инструменты в рамках заданной роли. Отсутствие необходимости в специализированной технической подготовке делает эксплуатацию приложения доступной при наличии базовых навыков работы с веб-интерфейсами, а понимание предметной области связи и тарифообразования лишь упрощает ориентирование и повышает эффективность использования предоставленных функций. В целом весь процесс взаимодействия сводится к интуитивному поиску нужных разделов в меню, переходу к соответствующим формам и таблицам, работе с данными (создание, редактирование, удаление), применению фильтров и изучению статистических отчетов, что обеспечивает полноценный и контролируемый цикл управления информацией.

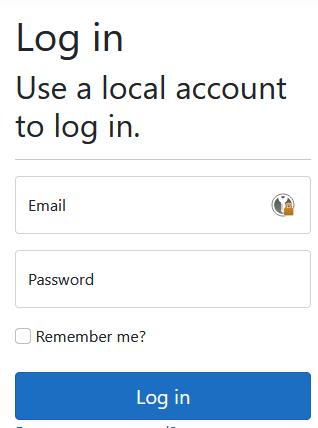
****

Рисунок 4.5 – Форма аутентификации

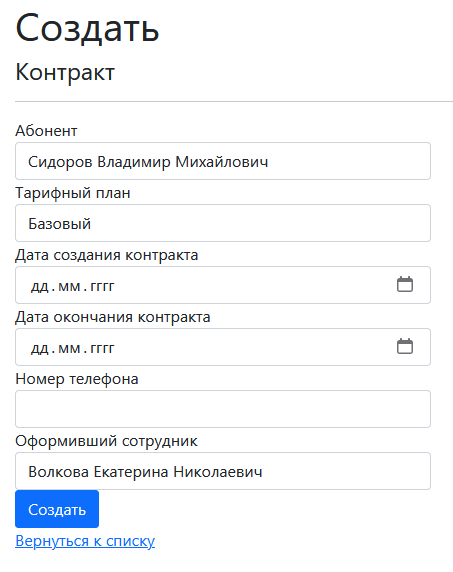


Рисунок 4.6 – Создание контракта

**5 РУКОВОДСТВО ПРОГРАММИСТА**

Программный комплекс автоматизирует работу телекоммуникационной компании, позволяя учитывать абонентов, управлять тарифными планами и договорами, а также анализировать использование услуг. Система обрабатывает данные абонентов, настраивает параметры тарифов и формирует отчеты. Для работы необходим сервер с установленной операционной системой *Windows Server* или *Linux*, веб-сервером *IIS* или *Nginx*, базой данных *Microsoft SQL Server* и средой выполнения .*NET Runtime* версии 6.0 или выше. Применение *MVC*-архитектуры обеспечивает четкое разделение логики обработки данных, интерфейса и доступа к базе данных. Программа управляет записями об абонентах, договорах и тарифах, а также анализирует параметры звонков, сообщений и интернет-трафика. Для контроля корректности данных используется валидация форм на стороне клиента и проверки на стороне сервера, а для диагностики и устранения проблем ведутся логи, фиксирующие действия и сбои. Перед началом работы настраивается учетная запись администратора, в файле *appsettings.json* в разделе *IdentitySeed* указывается *email* и пароль администратора. После сохранения настроек приложение размещается на сервере, инициализируется база данных командой *dotnet ef database update*. При первом запуске создается учетная запись администратора, при ее наличии повторное создание не требуется. Вход в систему осуществляется через веб-браузер по указанному адресу с использованием предоставленных учетных данных. Управление абонентами, тарифами и договорами происходит через соответствующие разделы меню приложения. Например, при создании абонента заполняется форма с персональными данными, при настройке тарифов указываются стоимость звонков, сообщений и передачи данных, а для добавления договора вводятся данные о выбранном абоненте, тарифном плане и дате заключения соглашения. При возникновении проблем анализируются логи, хранящиеся в каталоге приложения, они помогают выявить и устранить причины ошибок. Если в процессе запуска или работы программы обнаруживаются ошибки, их необходимо проанализировать и исправить. В случае отсутствия *email* или пароля администратора в *appsettings.json* лог зафиксирует «*InvalidOperationException: Admin email or password is not configured in appsettings.json*», что требует проверки значений *AdminEmail* и *AdminPassword* в секции *IdentitySeed* с последующим сохранением файла и перезапуском приложения. Если не удается создать пользователя-администратора, логи укажут «*Failed to create admin user*» и список ошибок, тогда проверяется соответствие пароля минимальным требованиям безопасности и при необходимости заменяются либо удаляются коллизирующие записи из базы данных. При попытке создать администратора с уже существующим *email* запись пропускается, а для работы используется имеющийся пользователь; при забытом пароле допускается восстановление либо ручная правка в базе данных. При ошибках подключения к базе данных логи сообщат «*Unable to connect to the database»*, после чего следует проверить доступность базы, корректность строки подключения и, при необходимости, заново применить миграции. После успешной настройки и запуска рекомендуется протестировать функционал путем проверки входа под учетной записью администратора, создания тестовых абонентов, тарифов и договоров, использования фильтрации, сортировки и разбивки данных на страницы, а также контролирования фиксации изменений при создании, редактировании и удалении записей. В случае любых затруднений или вопросов всегда можно обратиться к логам в каталоге приложения для анализа причин сбоев и их дальнейшего устранения.

**ЗАКЛЮЧЕНИЕ**

В процессе разработки *web*-приложения удалось создать эффективный инструмент, способствующий оптимизации управления данными и налаживанию взаимодействия между сотрудниками интернет-провайдера. Данный проект сочетает в себе широкий спектр возможностей для учёта, обработки и анализа информации, предоставляя пользователям интуитивно понятный интерфейс и высокий уровень надёжности.

При разработке были использованы современные технологии, включая язык программирования *C#* и фреймворк *ASP.NET*, что позволило создать удобное программное обеспечение. Основой приложения стала масштабируемая база данных, которая обеспечивает его стабильную работу и позволяет обрабатывать значительные объёмы информации без потери производительности.

Особое внимание уделялось архитектуре системы. Принципы *SOLID*, разделение на уровни, применение паттерна проектирования *MVC (Model-View-Controller*) и модульный подход к проектированию обеспечили структурированность кода и упростили его дальнейшее сопровождение. Подход MVC позволил разделить ответственность между логикой данных, пользовательским интерфейсом и обработкой пользовательских событий, что упрощает поддержку кода, добавление нового функционала и повышает тестируемость приложения. Такой структурный подход гарантирует устойчивость системы к изменениям и удобство её масштабирования.

Разработанное приложение способно решать задачи различной сложности и адаптироваться под индивидуальные потребности предприятий. Оно автоматизирует рутинные операции, оптимизирует рабочие процессы и способствует повышению производительности труда. Гибкость, безопасность и удобство использования делают данный проект актуальным и востребованным инструментом для предприятий, стремящихся к цифровой трансформации и улучшению качества работы.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Мюллер, Д. С# для чайников / Джон Мюллер, Билл Семпф, Чак Сфер. – Издательство «Диалектика-Вильямс», 2019. – 608 с.
2. Прайс, М. С# 10 и .NET 6. Современная кроссплатформенная разработка / М.Прайс – Издательство «Питер», 2023. – 1846 с.
3. Обзор языка C# – Руководство по C#: Microsoft Docs. – Электрон. данные. – Режим доступа: <https://docs.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp/>. – Дата доступа: 28.03.2024.
4. Принципы SOLID в C#. – Электрон. данные. – Режим доступа: <https://professorweb.ru/my/it/blog/net/solid.php>. – Дата доступа: 04.04.2024.
5. Гриффитс, И. Программируем на C# 8.0 / И. Гриффитс – Издательство «Питер», 2021. – 944 с.
6. Моргунов, E. PostgreSQL. Основы языка SQL / Е.Моргунов – Россия, 2024. – 336 с.
7. Ivanov, N. Портфолио разработчика / N.Ivanov – Россия, 2023. – 246 с.