Содержание

[Используемые термины и условные обозначения 3](#_Toc105467373)

[Введение 4](#_Toc105467374)

[1 Аналитическая часть 5](#_Toc105467375)

[1.1 Описание предметной области 5](#_Toc105467376)

[1.2 Анализ требований к приложению с использованием CASE средств 5](#_Toc105467377)

[1.3 Описание модели данных 11](#_Toc105467378)

[1.4 Описание архитектуры приложения 13](#_Toc105467379)

[1.5 Проектирование интерфейса 14](#_Toc105467380)

[1.6 Определение средств разработки 19](#_Toc105467381)

[2 Технологическая часть 23](#_Toc105467382)

[2.1 Описание функций приложения 23](#_Toc105467383)

[2.2 Описание организации данных 24](#_Toc105467384)

[2.3 Описание логики приложения 28](#_Toc105467385)

[2.4 Описание уровня представления приложения 31](#_Toc105467386)

[2.5 Организация защиты информации 36](#_Toc105467387)

[2.6 Сценарии и протоколы тестирования 36](#_Toc105467388)

[3 Экономическая часть 38](#_Toc105467389)

[3.1 Описание программного продукта 38](#_Toc105467390)

[3.2 Описание качества программного продукта 38](#_Toc105467391)

[3.3 Оценка проекта 39](#_Toc105467392)

[Заключение 44](#_Toc105467393)

[Список используемых источников 45](#_Toc105467394)

### Используемые термины и условные обозначения

**Программное обеспечение (ПО) -** программа или множество программ, используемых для управления компьютером;

**Система управления базами данных** (**СУБД**) - совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных.

**Объектно-ориентированное программирование (ООП)** — методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определённого класса, а классы образуют иерархию наследования.

**Компиляция -** процесс создания исполняемой программы с помощью кода, написанного на скомпилированном языке программирования.

**Фреймворк** - программная платформа, определяющая структуру программной системы; программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта.

**Серверная сторона (Server-side) –** процедуры и процессы, выполняемые непосредственно на стороне сервера.

**Common Language Runtime (CLR)** – исполняющая среда программной платформы Microsoft .NET Framework, интерпретирующая и контролирующая выполнение программного кода, представленного промежуточным CIL кодом.

### Введение

Целью дипломного проекта является разработка десктоп приложения «Управление договорными отношениями».

Приложение «Управление договорными отношениями» предназначено для эффективной работы менеджера с клиентом. Приложение позволяет просматривать заключенные договора с компаниями-клиентами, управлять ими и отслеживать выполнение обязательств клиента по договорам.

Сегодня большинство крупных компаний используют различные системы управления договорными отношениями, что позволяет им быть гораздо продуктивнее.

Цель создания десктоп приложения заключается в том, чтобы, используя приложение, менеджер мог эффективнее работать с клиентами или компаниями, подписывать с ними договора анализировать их и формировать отчеты для упрощения дальнейшей работы с данными компаниями или физическими лицами.

С помощью приложения менеджер сможет рациональнее использовать свое время и обрабатывать большее количество заявок от потенциальных клиентов.

Основанием для разработки является приказ директора БПОУ Омавиат №250-ОД от 26.04.2022.

### 1 Аналитическая часть

### 1.1 Описание предметной области

[Договорные отношения](https://construction_materials.academic.ru/2815/%D0%94%D0%BE%D0%B3%D0%BE%D0%B2%D0%BE%D1%80%D0%BD%D1%8B%D0%B5_%D0%BE%D1%82%D0%BD%D0%BE%D1%88%D0%B5%D0%BD%D0%B8%D1%8F) - отношения, возникающие в соответствии с условиями обязательств, определенных в договоре подряда и требованиями закона, иных правовых и нормативных актов, которые обязательны для выполнения.

Управление договорными отношениями, подразумевает собой большой круг задач по взаимодействию менеджера с клиентом. Для эффективной работы с клиентом менеджеры крупных компаний используют приложения для управления договорными отношениями.

Менеджер может подписать договор на оказание компанией услуги с клиентом. Приложение же позволит менеджеру управлять договорами на всех его этапах начиная с подписания и заканчивая выполнением обязательств с обоих сторон. После подписания и согласования договора с юристом компании менеджер сможет вести эффективную работу с клиентом, просматривать внесенные платежи по договорам и предупреждать клиента о будущих платежах или их просрочке.

### 1.2 Анализ требований к приложению с использованием CASE средств

Для того, чтобы наглядно представить ожидаемое поведение приложения с взаимодействием пользователей, необходимо построить диаграмму вариантов использования, которая представлена на рисунке 1.

Диаграмма отражает отношения между актерами (пользователями) и прецедентами системы (варианты использования разрабатываемой системы). Она поможет понять, как необходимо построить логику приложения.



Рисунок 1 – Диаграмма вариантов использования

Пользователи разбиты на несколько категорий:

- Менеджер – сотрудник компании, который имеет права на просмотр своих договоров, а также, их создание, редактирование, удаление и согласование. Менеджер может просматривать всю информацию о договоре.

- Юрист – сотрудник компании, который имеет права на просмотр всех договоров со статусом «Не согласован», юрист может просматривать информацию о доступных договорах, и менять их статус на «Согласован».

Выполним описание основных вариантов использования приложения.

1. **Функция «Создание договора»:**

Таблица 1 *-* Вариант использования "Создание договора"

|  |  |
| --- | --- |
| Название прецедента | Создание договора |
| Цель | Создать договор с клиентом компании |
| Актеры | Менеджер |
| Тип | Основной |
| Стиль | Базовый |
| Ссылки (предусловия) | Реализован прецеденты авторизация |
| Результаты (постусловия) | В случае успешного основного сценария передача данных о договоре в базу данных и появление договора в приложении. |

Основной успешный сценарий для данного варианта использования представлен в таблице 2.

Таблица 2 *-* Основной успешный сценарий "Создание договора"

| Действие актера | Отклик системы |
| --- | --- |
| 1.Менеджер переходит на вкладку «Создание договора».  3. Менеджер выбирает тип договора, временные сроки, заполнить сам договор и выставить цену для клиента. | 2.Приложение переводит пользователя на страницу создания договора.  4.Сервер создает запись нового договора в базе данных и отображает информацию о ней в приложении. |

**Альтернативные потоки событий для функции «Создание договора»:**

Не заполнены обязательные поля. Некорректно заполнены поля на форме:

* Приложение выводит ошибку о некорректности заполнения полей;

1. **Функция «Согласование договора»**

Таблица 3 - Вариант использования «Согласования договора»

|  |  |
| --- | --- |
| Название прецедента | Согласование договора |
| Цель | Согласовать договор с юристом компании |
| Актеры | Менеджер, юрист |
| Тип | Основной |
| Стиль | Базовый |
| Ссылки (предусловия) | Реализован прецеденты создание договора |
| Результаты (постусловия) | В случае успешного основного сценария статус договора измениться на согласован |

Основной успешный сценарий для данного варианта использования представлен в таблице 4.

Таблица 4 - Основной успешный сценарий "Согласование договора"

| Действие актера | Отклик системы |
| --- | --- |
| 1.Менеджер отправляет договор на согласование юристом. | 2.Юрист согласовывает договор и изменяет его статус на «Согласованно» |

1. **Функция «Редактирование договора».**

Таблица 5 - Вариант использования «Редактирование договора»

|  |  |  |
| --- | --- | --- |
| Название прецедента | Редактирование договора | |
| Цель | Внести изменения в договор | |
| Актеры | Менеджер | |
| Тип | Основной | |
| Стиль | Базовый | |
| Ссылки (предусловия) | Реализован прецеденты «Создание договора» | |
| Результаты (постусловия) | | В случае успешного основного сценария в приложении отобразиться вся информация о просматриваемом договоре. |

Основной успешный сценарий для данного варианта использования представлен в таблице 6.

Таблица 6 - Основной успешный сценарий «Редактирование договора»

| Действие актера | Отклик системы |
| --- | --- |
| 1.Менеджер переходит на страницу «Договора».  3. Менеджер выбирает нужный договор и нажимает на кнопку «подробнее».  5. Менеджер нажимает на кнопку «Изменить»  7. Менеджер вносит изменения в договор и нажимает на кнопку «Сохранить» | 2. Приложение отображает все договора менеджера.  4. Приложение открывает страницу договора.  6. Приложение открывает страницу редактирования договора.  8. Приложение сохраняет изменения |

**4. Функция «Удаление договора».**

Таблица 7 - Вариант использования «Подписание договора»

|  |  |
| --- | --- |
| Название прецедента | Удаление договора |
| Цель | Удалить договор с клиентом компании |
| Актеры | Менеджер |
| Тип | Основной |
| Стиль | Базовый |
| Ссылки (предусловия) | Реализован прецеденты создание договора |
| Результаты (постусловия) | В случае успешного основного сценария статус договора измениться на подписан. |

Основной успешный сценарий для данного варианта использования представлен в таблице 8.

Таблица 8 - Основной успешный сценарий «Подписание договора»

| Действие актера | Отклик системы |
| --- | --- |
| 1.Менеджер переходит на страницу «Договора».  3. Менеджер выбирает нужный договор и нажимает на кнопку «подробнее».  5. Менеджер нажимает на кнопку «Удалить»  7. Менеджер подтверждает удаление | 2. Приложение отображает все договора менеджера.  4. Приложение открывает страницу договора.  6. Приложение выводит диалоговое окно для подтверждения удаления.  8. Приложение удаляет договор |

### 1.3 Описание модели данных

Структура хранения данных представлена в виде таблиц, которые находятся в Microsoft SQL Server.

При создании проекта в базе должна храниться следующая информация:

* клиент;
* договор;
* менеджер;
* типы договоров;
* вид валюты;
* часть оплаты;
* оплата;
* статус договора;

На рисунке 2 представлены ключевые таблицы структуры хранения

данных. Для визуализации схемы данных используется ER-диаграмма.

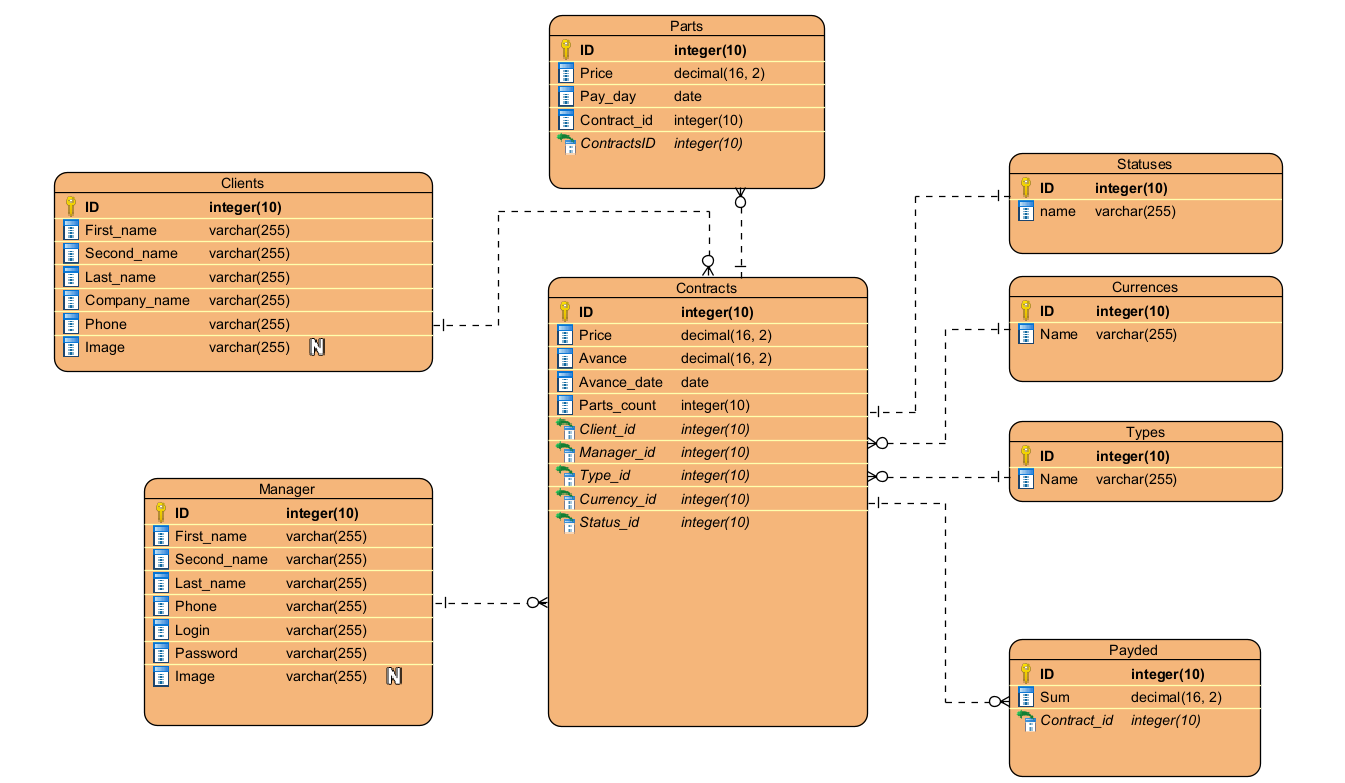


Рисунок 2 – ER-диаграмма

Опишем подробнее назначение и структуру этих таблиц.

**Таблица «Clients»** - в данной таблице хранятся данные о клиенте. Здесь имеются поля: ID (ID\_Клиента), First\_name (фамилия), Second\_name (имя), Last\_name (отчество), Phone (телефон), Company\_name (название компании) и Image (фото клиента). Имеет связи с сущностями «Договор», тип связи один (клиент) ко многим (договорам).

**Таблица «Managers»** - в данной таблице хранятся данные о менеджере. Здесь имеются поля: ID (ID\_Менеджера), First\_name (имя), Second\_name (фамилия), Last\_name (отчество), Phone (телефон), Login (логин), Password (пароль), Image (фото менеджера). Имеет связи с сущностями «Договор», тип связи один (менеджер) ко многим (договорам).

**Таблица «Contracts»** - в данной таблице храниться информация о договорах. Здесь имеются поля: ID, Price (цена), Avance (аванс), First\_part (сумма первой части оплаты), Second\_part (сумма второй части оплаты), Third\_part (сумма третьей части), Fourth\_part (сумма четвертой части), Parts\_count (кол-во частей оплаты), Avance\_date (срок оплаты аванса), First\_part\_date (срок оплаты первой части), Second\_part\_date (срок оплаты второй части), Third\_part\_date (срок оплаты третьей части), Fourth\_part\_date (срок оплаты четвертой части), Client\_id (ID клиента), Manager\_id (ID менеджера), Type\_id (ID типа договора), Currency\_id (ID валюты), Payded (оплаченная сумма). Имеет связи с сущностями «Clients», «Types», «Managers».

**Таблица «Types»** - хранит типы договоров, здесь имеются поля: ID, Name (название).

**Таблица «Statuses»** - хранит виды статусов договора, здесь имеются поля: ID, Name (название).

**Таблица «Payded»** - хранит информации об оплате клиента, здесь имеются поля: ID, Sum (оплаченная сумма), Contract\_id (номер договора).

**Таблица «Currencies»** - хранит виды валют, здесь имеются поля: ID, Name (название).

**Таблица «Parts»** - хранит части оплаты, здесь имеются поля: ID, Pay\_day (день оплаты), Price (сумма по договору), Contract\_id (номер контракта).

### 1.4 Описание архитектуры приложения

Приложение будет построено на клиент-сервер архитектуре. Под «клиент-серверным» приложением мы будем понимать информационную систему, основанную на использовании серверов баз данных.

На стороне клиента выполняется код приложения, в который обязательно входят компоненты, поддерживающие интерфейс с конечным пользователем, производящие отчеты, выполняющие другие специфичные для приложения функции

Клиентская часть приложения взаимодействует с клиентской частью программного обеспечения управления базами данных, которая, фактически, является индивидуальным представителем СУБД для приложения.

К преимуществам архитектуры «клиент-сервер» можно отнести:

* централизованность, поскольку все данные и управление сосредоточены в центральном сервере;
* информационная безопасность, поскольку ресурсы общего пользования администрируются централизованно;
* производительность, использование выделенного сервера повышает скорость работы ресурсов общего пользования;

масштабируемость, количество клиентов и серверов можно увеличивать независимо друг от друга;

Схему работы приложения можно увидеть на рисунке 2.

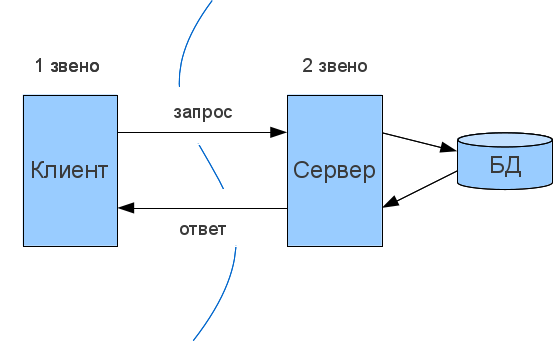


Рисунок 3 - Архитектура приложения

### 1.5 **Проектирование интерфейса**

Открыв приложение, пользователь будет видеть страницу авторизации, на которой он сможет вести логин и пароль, после авторизации пользователь будет попадать на главную страницу, страницу «Договора», на которой будут отображаться договора менеджера и основная информация о них, с этой страницы менеджер сможет перейти на страницу выбранного договора, в ней будет отображаться подробная информация о договоре, так же с данной таблицы можно перейти на страницу редактирования договора.

С главной страницы у пользователя будет возможность перейти на страницу «Профиль», создание договора и клиенты. На странице «Профиль» отображается основная информация о менеджере. На странице «Клиенты» будет отображаться основная информация о клиентах компании. На странице создания договоров пользователь сможет заполнить необходимые поля и создать новый договор.

С главной страницы можно будет перейти в следующие разделы:

* Профиль;
* Договора;
* Клиенты;
* Создание договора;

Более подробное описание всех переходов по приложению отображено на концептуальной модели приложения, представленной на рисунке 4.

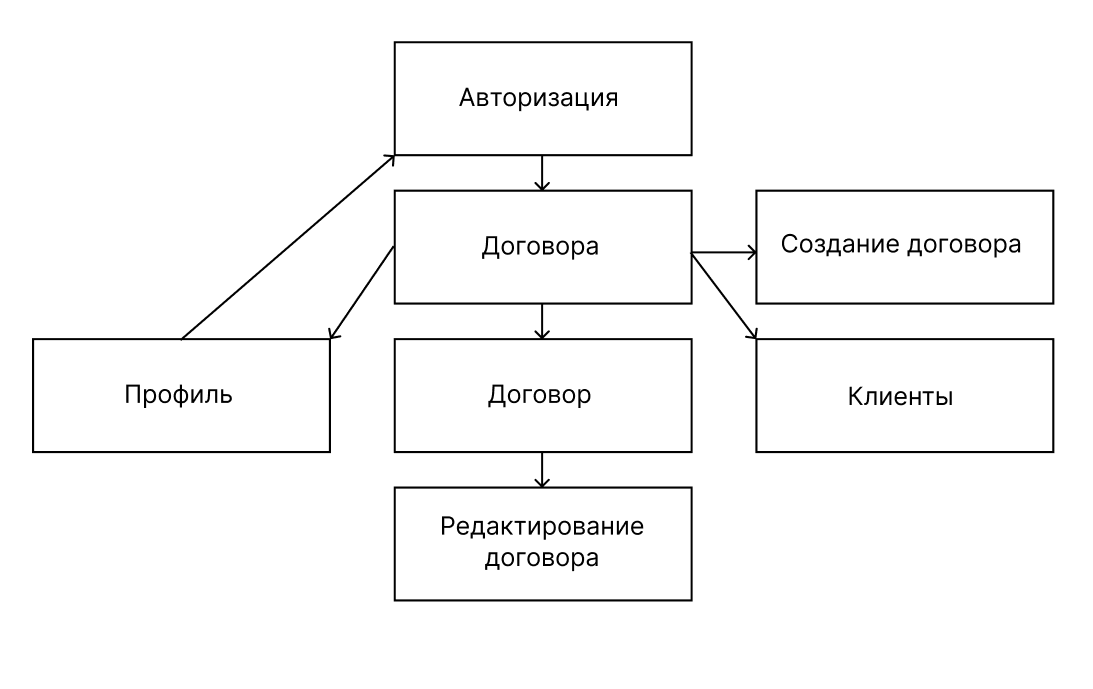


Рисунок 4 - Карта приложения

Макеты страниц представлены на рисунках 5,6,7,8,9,10,11,12.

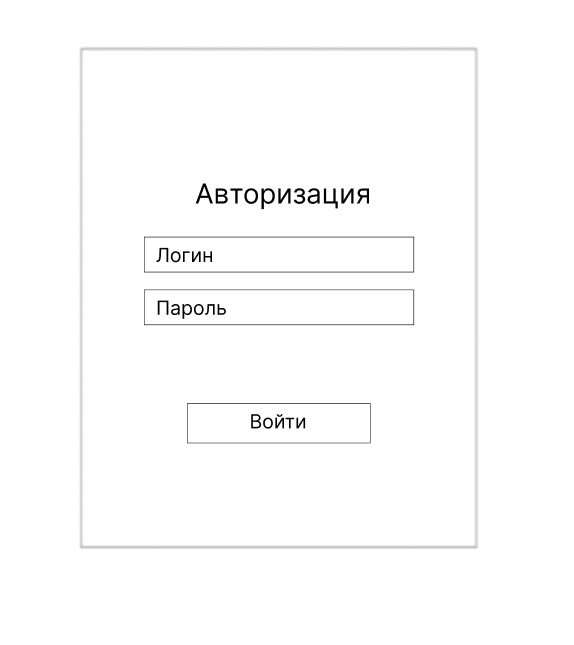


Рисунок 5 – Макет окна авторизации

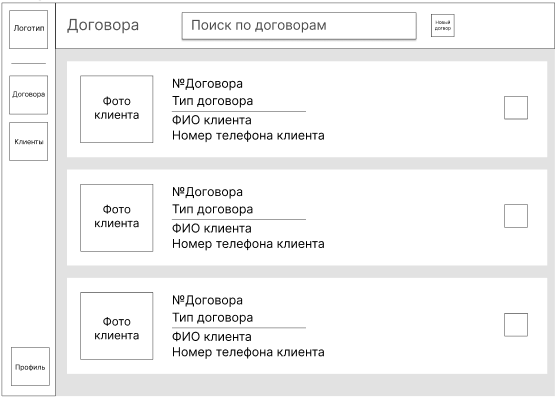


Рисунок 6 – Макет окна договоров

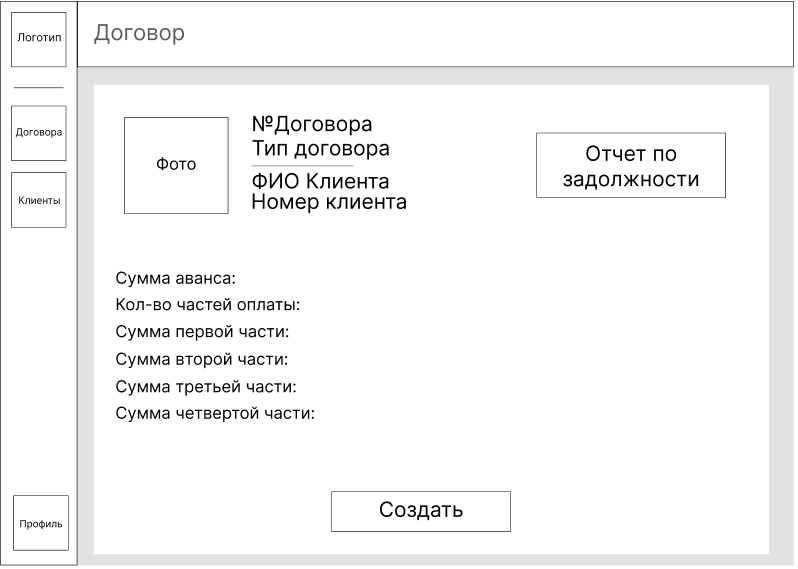


Рисунок 7 – Макет окна подробной информации о договоре

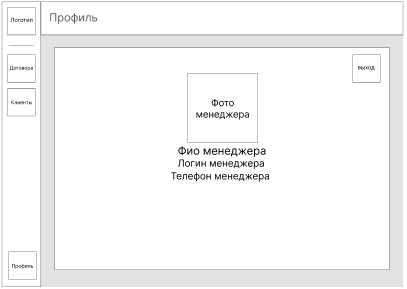


Рисунок 8 – Макет окна профиля

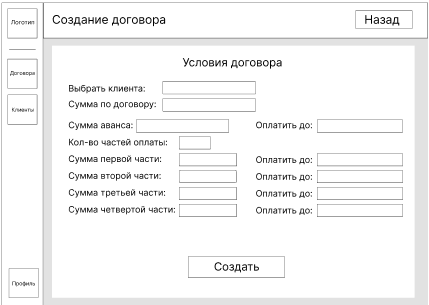


Рисунок 9 – Макет окна создания договора

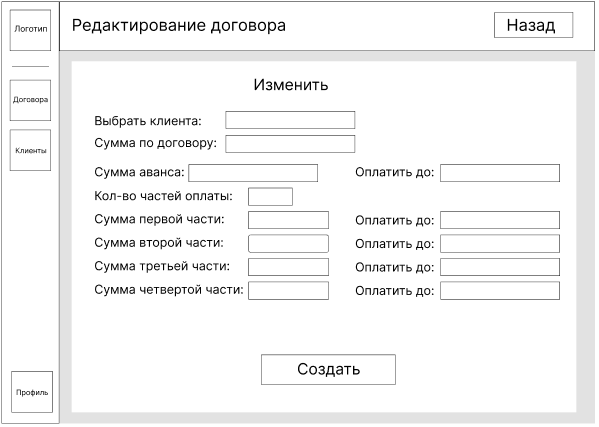


Рисунок 11 - Макет окна редактирования договора

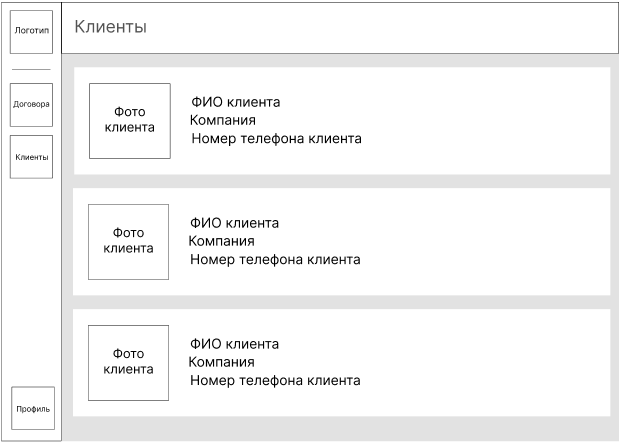


Рисунок 12 - Макет окна клиентов

Прототип приложения:

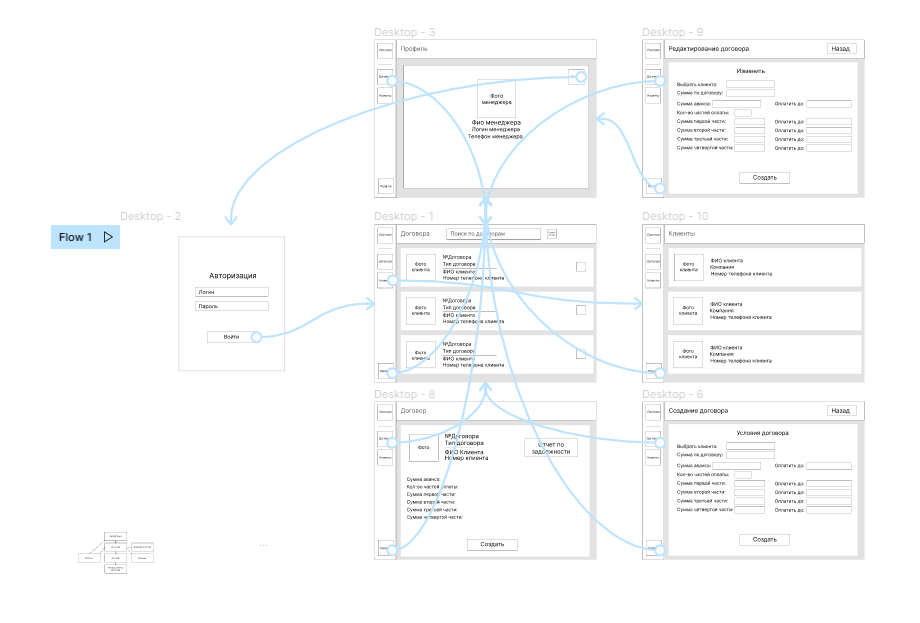


Рисунок 13 – Концептуальная модель приложения

### 1.6 Определение средств разработки

Для разработки приложения можно использовать язык программирования C# и платформу пользовательского интерфейса для создания клиентских приложений для настольных систем WPF в среде разработки Visual Studio. Для разработки базы данных была выбрана СУБД MS Server Management Studio и язык SQL соответственно.

Рассмотрим преимущества и недостатки использования данных средств разработки:

Преимущества C#:

* с# – это объектно-ориентированный, простой и в то же время мощный язык программирования, который позволяет разработчикам создавать многофункциональные приложения;
* с# относиться к языкам компилируемого типа, поэтому он обладает всеми преимуществами таких языков;
* мощная библиотека классов. .NET представляет единую для всех поддерживаемых языков библиотеку классов. Какое бы приложение мы ни собирались писать на C# — текстовый редактор, чат или сложный веб-сайт так или иначе мы задействуем библиотеку классов .NET;
* с# объединяет лучшие идеи современных языков программирования Java, C++, Visual Basic и т.д;
* из-за большого разнообразия синтаксических конструкций и возможности работать с платформой .Net, C# позволяет быстрее, чем любой другой язык, разрабатывать программные решения;
* разнообразие технологий. Общеязыковая среда исполнения CLR и базовая библиотека классов — это основа для целого стека технологий, которые разработчики могут задействовать при создании разных приложений. Например, для баз данных в этом стеке имеется технология ADO.NET и Entity Framework Core. Для графических приложений с насыщенным интерфейсом — технологии WPF и UWP. Для более простых графических приложений — Windows Forms. Для разработки мобильных приложений — Xamarin. Для создания веб-сайтов - ASP.NET и т.д;

Но есть у C# и некоторые недостатки:

* приоритетная ориентированность на платформу Windows;

Преимущества использования WPF:

* **независимость графики от разрешения и устройства.** Основной единицей измерения в графической системе WPF является аппаратно-независимый пиксель, размер которого составляет 1/96 дюйма. Это создает основу для независимой от разрешения и аппаратной платформы отрисовки. Каждый аппаратно-независимый пиксель автоматически масштабируется в соответствии с заданным в системе количеством точек на дюйм (DPI);
* **повышение точности.** Система координат WPF основана на числах двойной точности с плавающей запятой, а не числах одинарной точности. Значения преобразования и прозрачности также выражаются числами двойной точности. Платформа WPF также поддерживает широкую цветовую палитру (scRGB) и имеет встроенную поддержку управления входными данными из разных цветовых схем;
* **расширенная поддержка графики и анимации.** Платформа WPF упрощает программирование графики, автоматически управляя анимированными сценами. Кроме того, WPF обеспечивает поддержку проверки попадания и полную поддержку альфа-версии компоновки;
* **аппаратное ускорение.** Система графики WPF использует возможности графического оборудования, чтобы снизить нагрузку на ЦП;

Недостатки использования WPF:

* плохая документированность;
* чувствительность к драйверам видеокарты;

Microsoft SQL Server — система управления реляционными базами данных, разработанная корпорацией Microsoft. Основной используемый язык запросов — Transact-SQL.

Рассмотрим, какие у Microsoft SQL Server преимущества и недостатки. Это нужно, чтобы вы понимали, насколько данная СУБД подходит для ваших целей.

Основные достоинства:

* Масштабирование системы. Взаимодействовать с ней можно как на простых ноутбуках, так и на ПК с мощным процессором, который способен обрабатывать большой объем запросов.
* Размер страниц – до 8 Кб. Данные извлекаются быстро, а сложную информацию удобнее хранить. Система обрабатывает транзакции в интерактивном режиме, есть динамическая блокировка.
* Автоматизация рутинных административных задач. Например, управление блокировками и памятью, редактура размеров файлов. В программе продуманы настройки, можно создавать профили пользователей.
* Удобный поиск. Его можно осуществлять по фразам, словам, тексту либо создавать ключевые индексы.
* Поддержка работы с другими решениями Майкрософт, в том числе с Excel, Access.
* Также в программе предусмотрена синхронизация, есть репликации через интернет, службы преобразования информации и полноценный web-ассистент для форматирования страниц. Дополнительно в нее интегрирован сервис интерактивного анализа (можно принимать решения, создавать корпоративные отчеты).

Основные минусы:

* Зависимость от ОС. Система работает только с Windows.

Исходя из преимуществ и недостатков было принято решение использовать данные средства разработки для реализации требований приложения. С помощью них мы можем полностью реализовать функционал и визуальную составляющую приложения, на языке C# и XAML, с которыми, как раз, мы уже знакомы, так как изучали его в нашем колледже.

### 2 Технологическая часть

### 2.1 Описание функций приложения

Функции, реализованные в десктоп-приложении «Управление договорными отношениями».

* Авторизация;
* Просмотр договоров;
* Создание новых договоров;
* Изменение договора;
* Удаление договора;
* Просмотр профиля;
* Просмотр клиентов;

Рассмотрим каждую функцию поподробнее:

* 1. Функция «Авторизация»

Для реализации функции «Авторизация» был разработан модуль. Для полноценной работы данной функции был реализован функционал получения логина и пароля с сервера и сравнения их с введенными в приложении.

* 1. Функция «Просмотр договоров»

Для реализации функции «Просмотр договоров» была разработана страница со списком договоров на котором отображается основная информация о договорах, а также станица с подробной информацией о договоре в которой отображена полная информация о договоре.

* 1. Функция «Создание новых договоров»

Для реализации функции создание договоров был разработан модуль. Для полноценной работы данной функции был реализован функционал, в приложении который отправлял запрос на создание нового поля в таблице БД.

* 1. Функция «Изменение статуса договора»

Для реализации данной функции был разработан модуль, который отправлял запрос на изменение атрибута «статус» в таблице выбранного договора.

* 1. Функция «Просмотр профиля»

Для реализации функции «Просмотр профиля» была разработана страница со списком договоров на котором отображается основная информация о авторизовавшемся менеджере.

### 2.2 Описание организации данных

Структура хранения данных представлена в виде таблиц, которые находятся в Microsoft SQL Server.

В базе должна хранится информация о следующих сущностях:

* Managers;
* Contracts;
* Clients;
* Payded;
* Currencies;
* Types;
* Parts;
* Statuses;

Диаграмма базы данных представлена на рисунке 14.

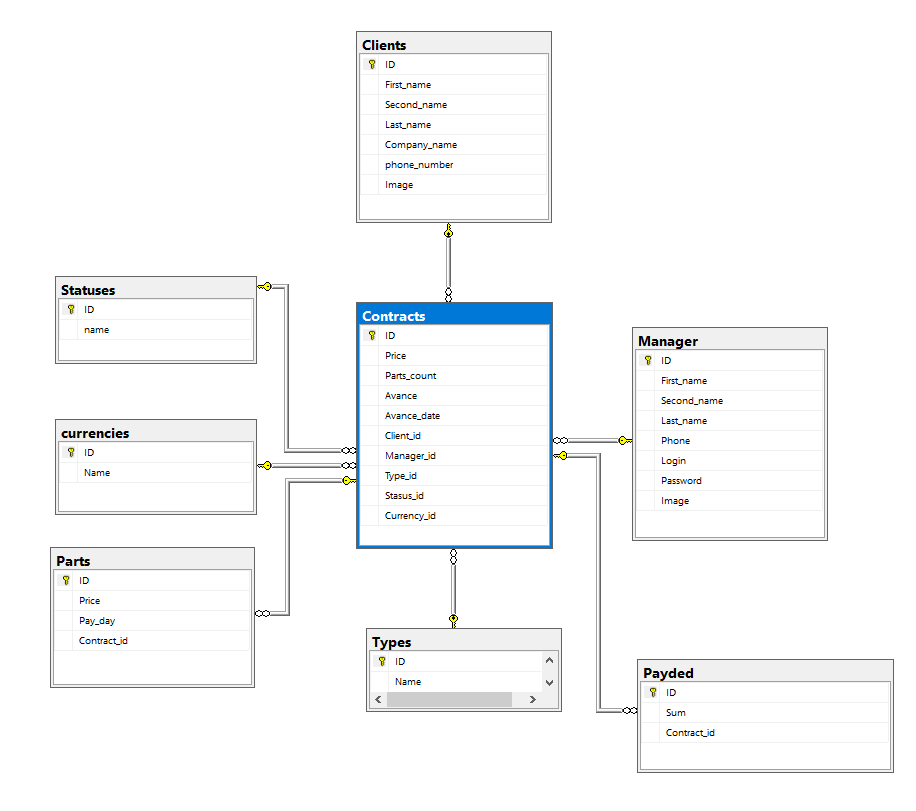


Рисунок 14 – Диаграмма базы данных

Опишем подробнее назначение и структуру этих таблиц.

Таблица «Clients» - в данной таблице хранятся данные о клиенте. Здесь имеются поля: ID (ID\_Клиента), First\_name (фамилия), Second\_name (имя), Last\_name (отчество), Phone (телефон), Company\_name (название компании) и Image (фото клиента). Имеет связи с сущностями «Договор», тип связи один (клиент) ко многим (договорам).

Таблица 9 - Описание полей таблицы «Clients».

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| ID (PK) | Integer | Номер Клиента |
| First\_name | Nvarchar | Фамилия |
| Second\_name | nvarchar | Имя |
| Last\_name | nvarchar | Отчество |
| Phone | nvarchar | Номер телефона |
| Company\_name | nvarchar | Название компании |
| Image | nvarchar | Фото клиента |

Таблица «Managers» - в данной таблице хранятся данные о менеджере. Здесь имеются поля: ID (ID\_Менеджера), First\_name (имя), Second\_name (фамилия), Last\_name (отчество), Phone (телефон), Login (логин), Password (пароль), Image (фото менеджера). Имеет связи с сущностями «Договор», тип связи один (менеджер) ко многим (договорам).

Таблица 10 - Описание полей таблицы "Managers"

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| ID\_Менеджера (PK) | Integer | Номер Менеджера |
| First\_name | nvarchar | Фамилия менеджера |
| Second\_name | nvarchar | Имя менеджера |
| Last\_name | nvarchar | Отчество менеджера |
| Phone | nvarchar | Телефон менежера |
| Login | nvarchar | Логин |
| Password | nvarchar | Пароль |
| Image | nvarchar | Фото менеджера |

Таблица «Contracts» - в данной таблице храниться информация о договорах. Здесь имеются поля: ID, Price (цена), Avance (аванс), First\_part (сумма первой части оплаты), Second\_part (сумма второй части оплаты), Third\_part (сумма третьей части), Fourth\_part (сумма четвертой части), Parts\_count (кол-во частей оплаты), Avance\_date (срок оплаты аванса), First\_part\_date (срок оплаты первой части), Second\_part\_date (срок оплаты второй части), Third\_part\_date (срок оплаты третьей части), Fourth\_part\_date (срок оплаты четвертой части), Client\_id (ID клиента), Manager\_id (ID менеджера), Type\_id (ID типа договора), Currency\_id (ID валюты), Payded (оплаченная сумма). Имеет связи с сущностями «Clients», «Types», «Managers».

Таблица 11 - Описание полей «Contracts».

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| ID\_Договора (РК) | Integer | Номер договора |
| Price | Integer | Стоимость |
| Avance | Integer | Сумма аванса |
| Parts\_count | Integer | Кол-во частей оплаты |
| Avance\_date | Date | Дата оплаты аванса |
| Client\_id (FK) | Integer | Номер клиента |
| Manager\_id (FK) | Integer | Номер менеджера |
| Type\_id (FK) | Integer | Номер типа |
| Currency\_id (FK) | Integer | Номер валюты |
| Status\_id (FK) | Integer | Номер статуса |

Таблица «Types» - хранит типы договоров, здесь имеются поля: ID, Name (название).

Таблица 12 - Описание таблицы «Types».

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| ID (PK) | Integer | Номер типа |
| Name | nvarchar | Название |

Таблица «Statuses» - хранит виды статусов договора, здесь имеются поля: ID, Name (название).

Таблица 13 – Описание таблицы «Statuses»

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| ID (PK) | Integer | Номер статуса |
| Name | nvarchar | Название |

Таблица «Payded» - хранит информации об оплате клиента, здесь имеются поля: ID, Sum (оплаченная сумма), Contract\_id (номер договора).

Таблица 14 – Описание таблицы «Payded»

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| ID (PK) | Integer | Номер оплаты |
| Sum | nvarchar | Сумма |
| Contract\_id | Integer | Номер договора |

Таблица «Currencies» - хранит виды валют, здесь имеются поля: ID, Name (название).

Таблица 15 – Описание таблицы «Currencies»

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| ID (PK) | Integer | Номер валюты |
| Name | nvarchar | Название |

Таблица «Parts» - хранит части оплаты, здесь имеются поля: ID, Pay\_day (день оплаты), Price (сумма по договору), Contract\_id (номер контракта).

Таблица 16 – Описание таблицы «Parts»

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| ID (PK) | Integer | Номер части |
| Pay\_day | nvarchar | Дата оплаты |
| Price | Decimal(16,2) | Сумма |
| Contract\_id | integer | Номер договора |

### 2.3 Описание логики приложения

Для описания логики приложения, мы опишем основные функции и методы для их реализации.

Основные функции приложения:

* Авторизация;
* Создание новых договоров;
* Изменение договора;
* Удаление договора;
  1. Функция «Авторизация»

Для реализации функции авторизация был разработан метод. Для полноценной работы данной функции была реализована сравнение текста из поля логина и пароля с текстом из БД, посредствам использования EntityFramework.

Листинг 1 – Код авторизации.

private void MoveToMainScreen(object sender, RoutedEventArgs e)

{ login = LoginTextBox.Text;

password = PasswordTextBox.Password;

Manager manager;

if (string.IsNullOrWhiteSpace(login))

{ MessageBox.Show("Введите логин!"); return;}

else if (string.IsNullOrWhiteSpace(password))

{ MessageBox.Show("Введите пароль!"); return;}

else if (CM\_Entities.GetContext().Manager.Where(x => x.Login == login && x.Password == password).Count() == 0)

{ MessageBox.Show("Такого пользователя не существует!"); return;}

else if (CM\_Entities.GetContext().Manager.Where(x => x.ID == 3).Count() > 0)

{ manager = CM\_Entities.GetContext().Manager.Where(x => x.Login == login && x.Password == password).First();

ManagerSaver.Login = login;

ManagerSaver.First\_name = manager.First\_name.ToString();

ManagerSaver.Second\_name = manager.Second\_name.ToString();

ManagerSaver.Last\_name = manager.Last\_name.ToString();

ManagerSaver.Phone\_number = manager.Phone.ToString();

ManagerSaver.ID = manager.ID;

MainScreen mainScreen = new MainScreen();

FrameManager.MainFrame.Navigate(new LayerContractsPage());

FrameManager.MainScreen = mainScreen;

FrameManager.MainScreen.Show();

FrameManager.MainWindow.Hide();

LoginTextBox.Text = "Логин";

passwordlable.Text = "Пароль";

PasswordTextBox.Password = null;

} else { manager = CM\_Entities.GetContext().Manager.Where(x => x.Login == login && x.Password == password).First();

ManagerSaver.Login = login;

ManagerSaver.First\_name = manager.First\_name.ToString();

ManagerSaver.Second\_name = manager.Second\_name.ToString();

ManagerSaver.Last\_name = manager.Last\_name.ToString();

ManagerSaver.Phone\_number = manager.Phone.ToString();

ManagerSaver.ID = manager.ID;

MainScreen mainScreen = new MainScreen();

FrameManager.MainFrame.Navigate(new DogovoraPage());

FrameManager.MainScreen = mainScreen;

FrameManager.MainScreen.Show();

FrameManager.MainWindow.Hide();

LoginTextBox.Text = "Логин";

passwordlable.Text = "Пароль";

PasswordTextBox.Password = null; }}

* 1. Создание нового договора

Для реализации функции создания нового договора был разработан метод, который создает новую запись в таблице «Contracts» посредствам использования EntityFramework.

Основной код метода создания нового договора представлен на листинге 2.

Листинг 2 – Код создания нового договора

\_currentContranct.Avance\_date = (DateTime)AvanceDate.SelectedDate;

Contracts currentContract = CM\_Entities.GetContext().Contracts.Add(\_currentContranct);

switch (index)

{case 0:

Parts part = new Parts();

part.Pay\_day = First\_part\_date.SelectedDate;

part.Price = decimal.Parse(TBFirstpart.Text.ToString(),CultureInfo.InvariantCulture.NumberFormat);

part.Contract\_id = currentContract.ID;

CM\_Entities.GetContext().Parts.Add(part); break;

case 1:

Parts part1 = new Parts();

Parts part2 = new Parts();

part1.Pay\_day = First\_part\_date.SelectedDate;

part1.Price = decimal.Parse(TBFirstpart.Text.ToString(), CultureInfo.InvariantCulture.NumberFormat);

part1.Contract\_id = currentContract.ID;

part2.Pay\_day = Second\_part\_date.SelectedDate;

part2.Price = decimal.Parse(TBSecondpart.Text.ToString(), CultureInfo.InvariantCulture.NumberFormat);

part2.Contract\_id = currentContract.ID;

CM\_Entities.GetContext().Parts.Add(part1);

CM\_Entities.GetContext().Parts.Add(part2);

break;

case 2:

Parts part3 = new Parts();

Parts part4 = new Parts();

Parts part5 = new Parts();

part3.Pay\_day = First\_part\_date.SelectedDate;

part3.Price = decimal.Parse(TBFirstpart.Text.ToString(),CultureInfo.InvariantCulture.NumberFormat);

part3.Contract\_id = currentContract.ID;

part4.Pay\_day = Second\_part\_date.SelectedDate;

part4.Price = decimal.Parse(TBSecondpart.Text.ToString(),CultureInfo.InvariantCulture.NumberFormat);

part4.Contract\_id = currentContract.ID;

part5.Pay\_day = Third\_part\_date.SelectedDate;

part5.Price = decimal.Parse(TBThirdpart.Text.ToString(),CultureInfo.InvariantCulture.NumberFormat);

part5.Contract\_id = currentContract.ID;

CM\_Entities.GetContext().Parts.Add(part3);

CM\_Entities.GetContext().Parts.Add(part4);

CM\_Entities.GetContext().Parts.Add(part5);

break;

CM\_Entities.GetContext().SaveChanges();

MessageBox.Show("Данные успешно сохранены!");

* 1. Изменение договора

Для реализации функции изменения договора был разработан метод. Для полноценной работы данной функции был реализован метод, который сохранял изменения, внесенные в необходимые поля, посредствам использования EntityFramework.

Листинг 3 - Код изменения договора

private Contracts \_currentContranct = new Contracts();

private Parts[] parts;

DataContext = contract;

\_currentContranct = contract;

parts = contract.Parts.ToArray();

CM\_Entities.GetContext().SaveChanges();

* 1. Удаление договора

Для реализации функции удаления договора был разработан метод, который удалял договор, после подтверждения удаления пользователем.

Листинг 4 – Код удаления договора.

bool? result = new CustomMessageBox("Вы действительно хотите удалить договор?", MessageType.Confirmation, MessageButtons.YesNo).ShowDialog();

if (result == false)

return;

try

{ CM\_Entities.GetContext().Contract.Remove((sender as Button).DataContext as contract);

CM\_Entities.GetContext().SaveChanges();

new CustomMessageBox("Договор успешно удален!", MessageType.Success, MessageButtons.Ok).ShowDialog();

NavigateManager.MainFrame.Navigate(new DogovoraPage());}

catch (Exception ex)

{new CustomMessageBox(ex.Message.ToString(), MessageType.Error, MessageButtons.Ok).ShowDialog();}

### 2.4 Описание уровня представления приложения

Если запустить приложение, откроется страница авторизации, авторизовавшись на ней можно осуществить переход на остальные страницы приложения. На данной странице пользователь может увидеть поля для ввода логина и пароля.

Страница авторизации, на которую пользователь попадает при входе, изображена на рисунке 9.

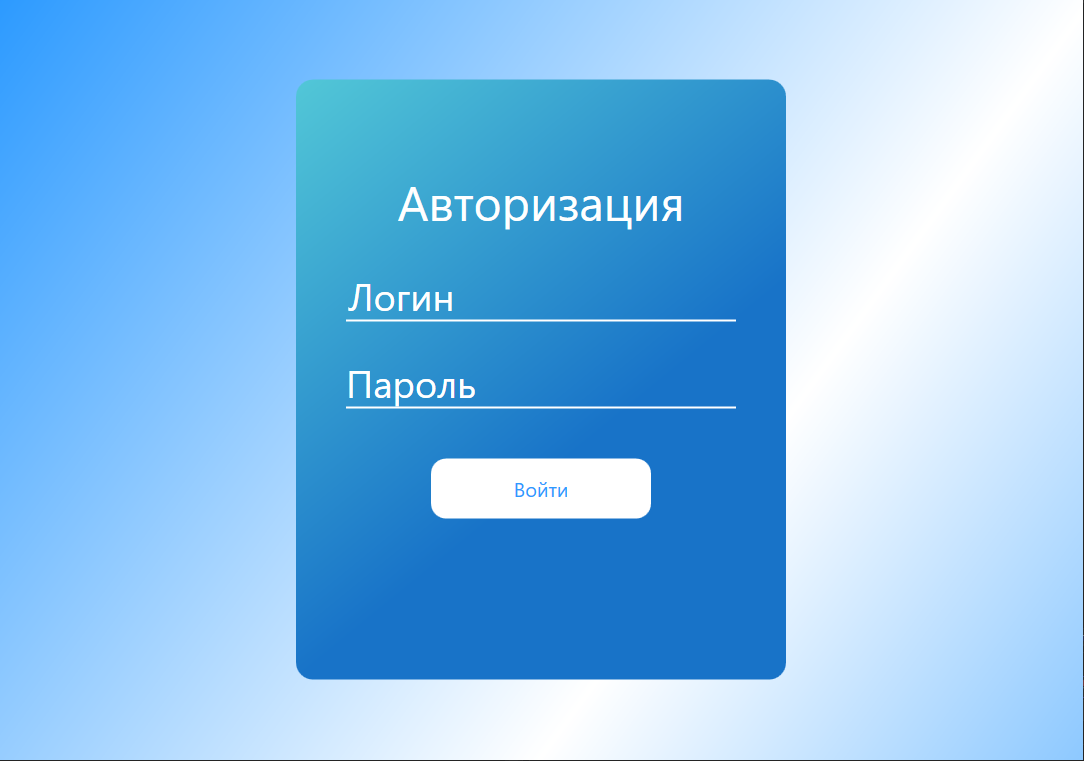


Рисунок 14 - Окно авторизации.

Страница «Договора» отображает все договора менеджера и основную информацию о них, с данной страницы мы можем перейти на страницы «Профиль», «Клиенты» и «Создание договора».

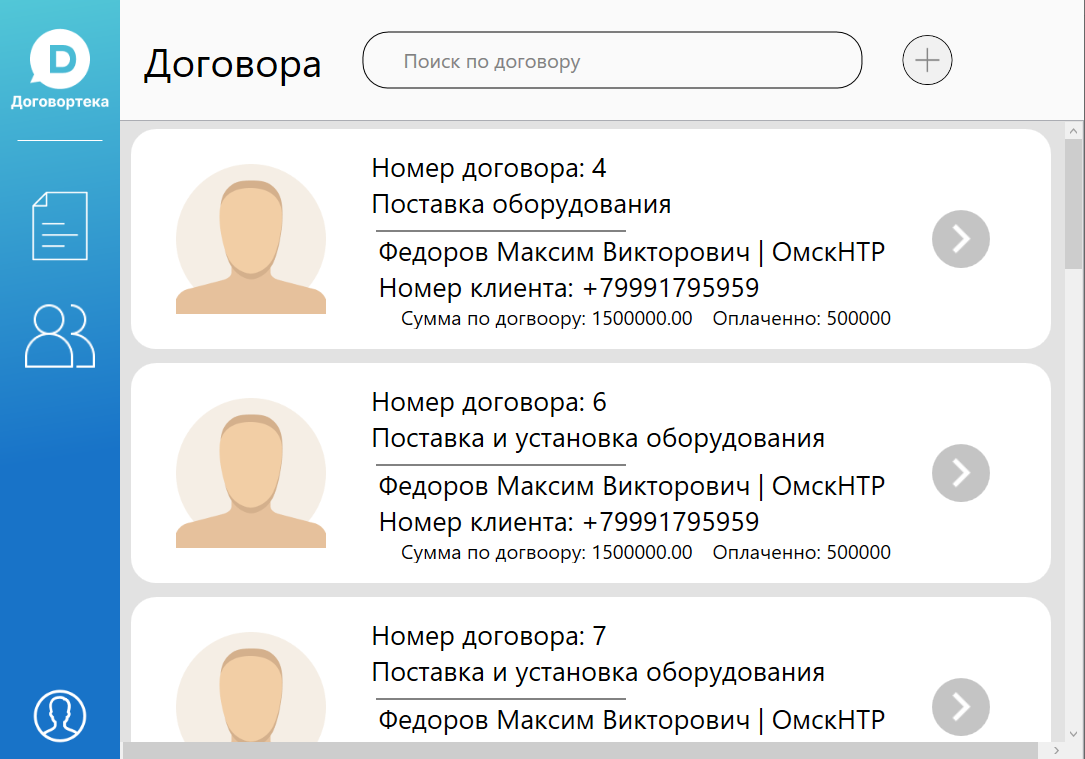


Рисунок 10 - Страница «Договора»

Со страницы «Договора» можно перейти на страницу подробной информации о договоре на которой отображается вся информация о данном договоре.

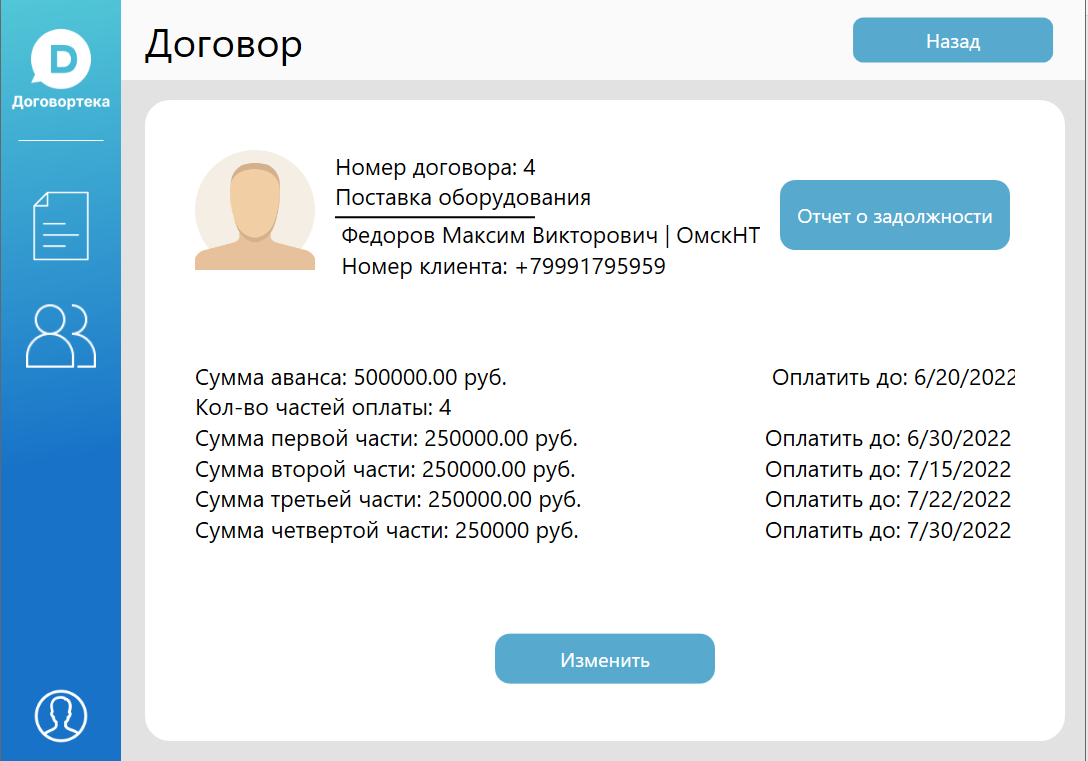


Рисунок 11 – Страница «Договор»

Перейдя на страницу «Профиль» можно увидеть информацию о менеджере, авторизованном в приложении.

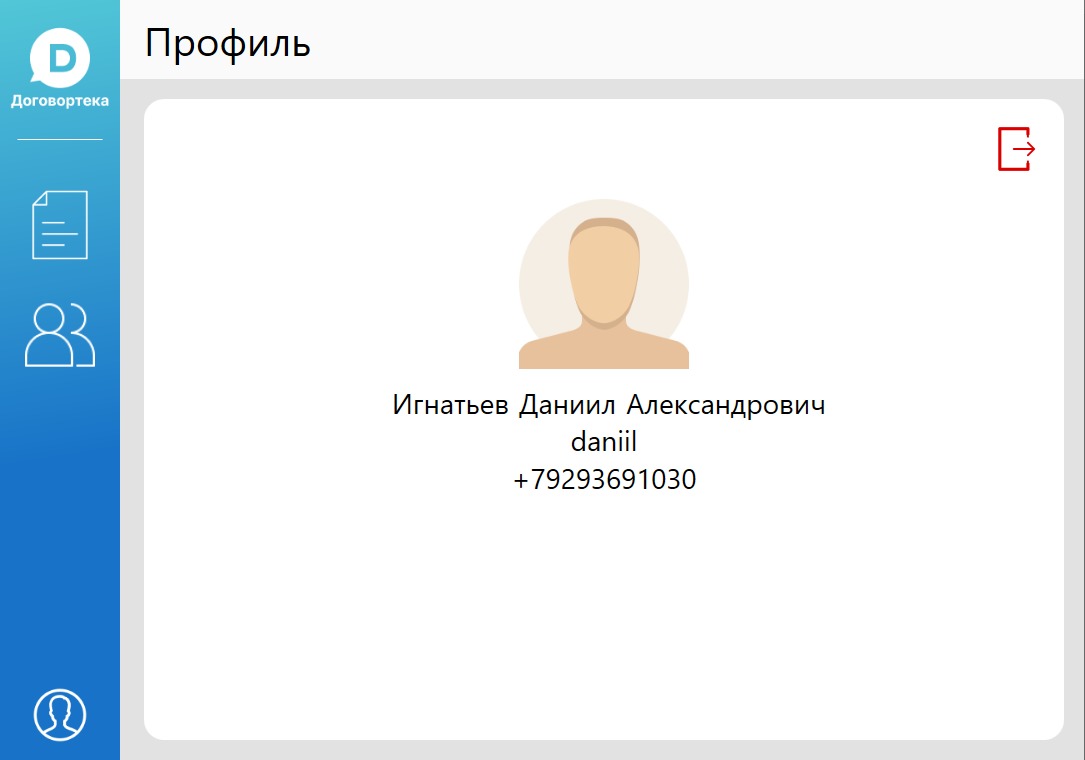


Рисунок 12 - Страница «Профиль»

На странице «Создание договора» реализована основная функция, создание нового договора, здесь мы заполняем всю информацию о договоре в нужных полях и нажимаем на кнопку создать.

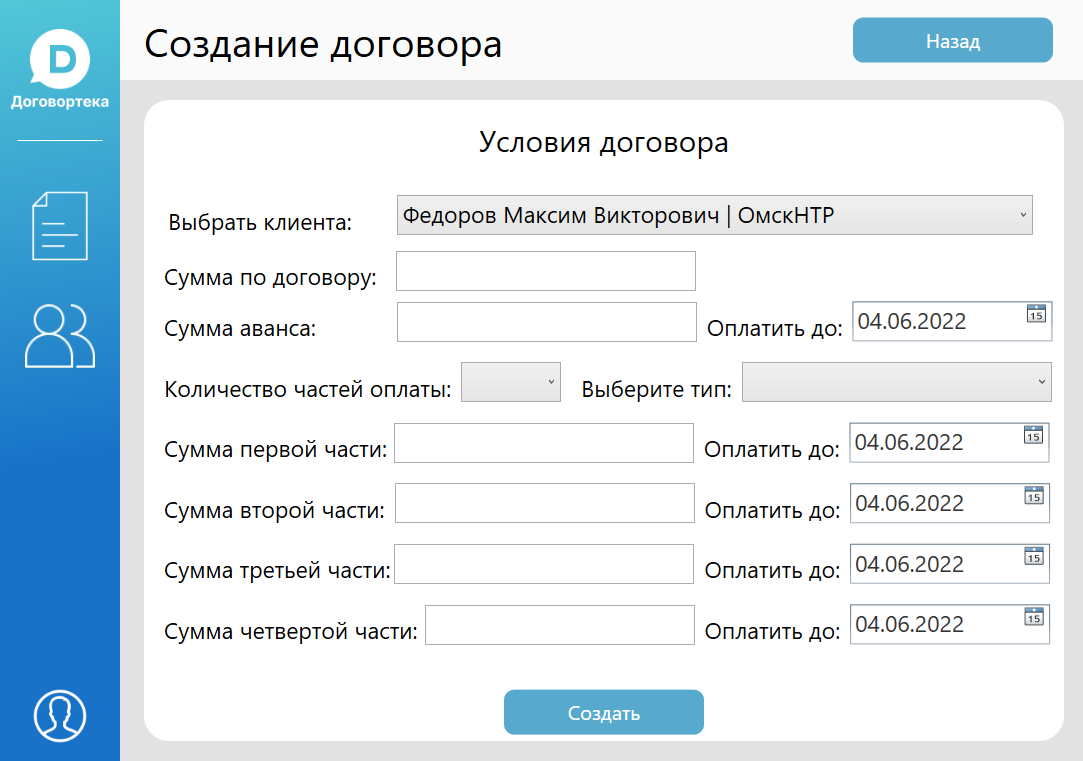


Рисунок 13 - Страница Создания договора

В случае правильного заполнения формы, получаем уведомление что договор успешно создан и переносимся на страницу «Договора».

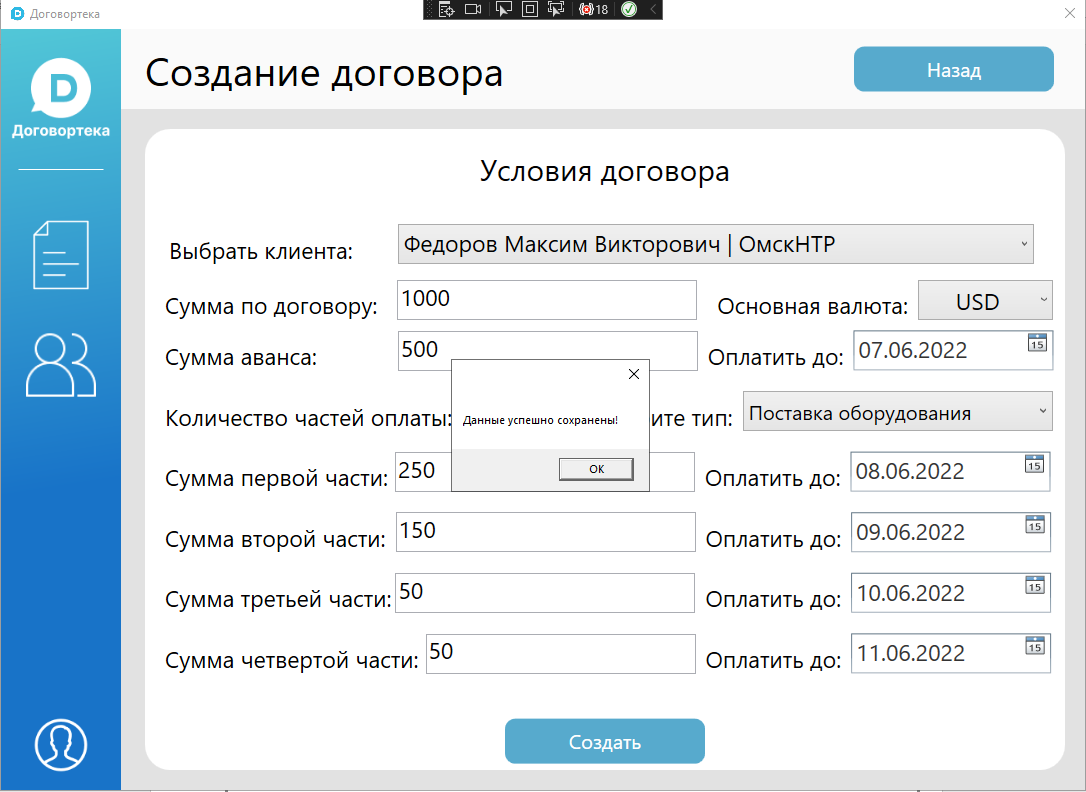


Рисунок 14 – Успешное создание договора.

После создания договора он отобразиться на странице договора.

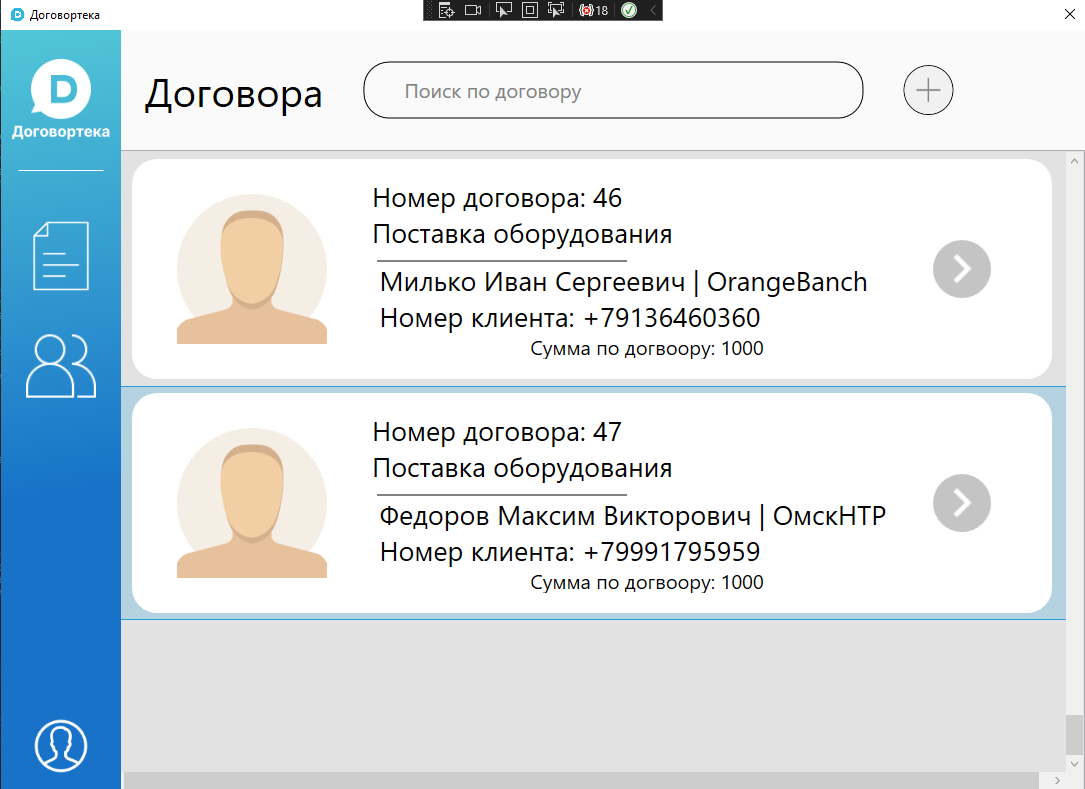
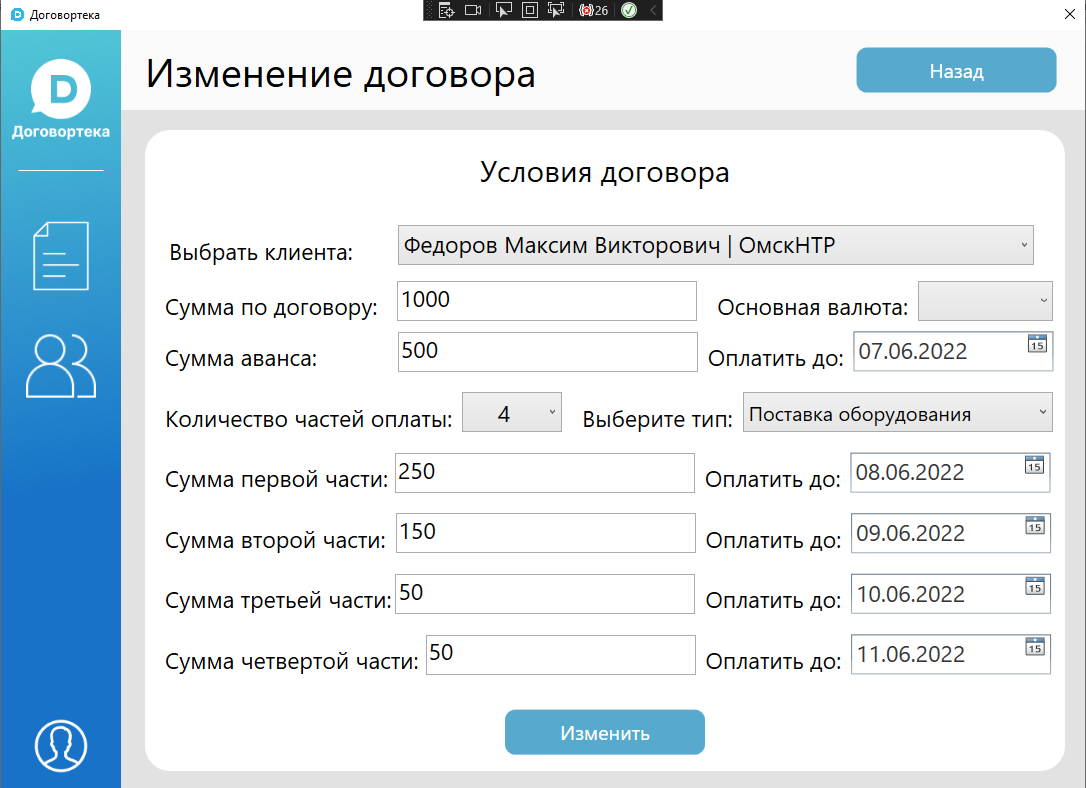


Рисунок 15 – Отображение созданного договора.

Для редактирования договора достаточно нажать на кнопку «изменить» на странице подробной информации о договоре. После чего откроется страница редактирования договора, на которой необходимо заполнить все поля и нажать на кнопку «Изменить». После чего откроется страница с договорами, а информация о договоре измеится.



### 2.5 Организация защиты информации

Для защиты информации были созданы роли: «Менеджер» и «Юрист». Доступ к приложению доступен только авторизированному пользователю. Авторизация происходит через экран авторизации, который открывается при запуске приложения. Авторизация происходит через логин и пароль пользователя, который доступен лишь пользователю и администратору базы данных.

Пользовательские данные для входа в приложение:

Login: admin

Password: admin

### 2.6 Сценарии и протоколы тестирования

Для тестирования приложения будут использоваться тест кейсы

Таблица 17 – Тест кейс «Создание договора»

|  |  |
| --- | --- |
| Номер | Тест-кейс №1 |
| Название | Создание договора |
| Предусловие | Менеджер должен быть авторизован в приложении. |
| Шаги | Нажать на кнопку «+» на экране договора🡪 Заполнить все поля тредубуемыми данными 🡪 Нажать кнопку «Создать». |
| Ожидаемый результат | Договор успешно создан и выводится в списке договоров. |
| Постусловия | Если идентичный договор уже создан, то удалить предыдущий. |

Таблица 18 – Тест кейс «Авторизация»

|  |  |
| --- | --- |
| Номер | Тест-кейс №2 |
| Название | Нарушение валидации при авторизации. |
| Предусловие | Необходимо запустить приложение. |
| Шаги | Запустить приложение  Нажать на кнопку «Войти», поля оставить пустыми. |
| Ожидаемый результат | Ошибка, поля не могут быть пустыми. |
| Постусловия | Если у вас есть аккаунт, то введите свои персональные данные, если нет, то обратитесь к администратору бд. |

Таблица 19 – Тест кейс «Создание договора»

|  |  |
| --- | --- |
| Номер | Тест-кейс №3 |
| Название | Создание договора |
| Предусловие | Менеджер должен быть авторизован в приложении. |
| Шаги | Нажать на кнопку «+» на экране договора🡪 Поля оставить пустыми 🡪 Нажать кнопку «Создать». |
| Ожидаемый результат | Ошибка, поля не могут быть пустыми. |
| Постусловия | Если идентичный договор уже создан, то удалить предыдущий. |

Таблица 20 – Тест кейс «Изменение договора»

|  |  |
| --- | --- |
| Номер | Тест-кейс №3 |
| Название | Создание договора |
| Предусловие | Менеджер должен быть авторизован в приложении. |
| Шаги | Нажать на стрелочку на экране договора🡪 Нажать на кнопку изменить на экране договора 🡪 Заполнить поля некоректными значениями 🡪 Нажать кнопку «Создать». |
| Ожидаемый результат | Ошибка, поля заполенеы не верно. |
| Постусловия | Если идентичный договор уже создан, то удалить предыдущий. |

### 3 Экономическая часть

### 3.1 Описание программного продукта

Наименование десктоп приложения «Договортека».

Возможности программного продукта, доступные конечному пользователю:

Разработанный продукт будет использоваться как десктоп приложение CRM-система для повышения эффективности работы менеджеров компании.

Основные черты приложения:

* простой и понятный интерфейс;
* быстрый отклик системы;
* приятный дизайн, не напрягающий глаза;

Необходимые программные требования:

* версия системы: Windows 7, Windows 8, Windows 8.1, Windows 10;
* минимум 2 гигабайта оперативной памяти;
* 2-х ядерный процессор с частотой 1.5 Ггц;
* оперативная память DDR3 2 гб;
* постоянный интернет со скоростью 100 мб/с.

### 3.2 Описание качества программного продукта

Функциональные возможности:

* функциональная полнота;
* функциональная – согласованность.

Пример важны функций десктоп приложения:

* возможность проходить успешную авторизацию в систему;
* возможность управлять договорами;
* возможность создавать новые договора.

Данные функции должны выполняться корректно в соответствии требованиям данного продукта.

Надежность – приводится описание набора программных и организационных средств, обеспечивающих возможность сохранять качество функционирования программного продукта при установленных условиях за установленный период времени:

* поля для ввода имеют пример, чтобы пользователь ввел корректные данные;
* валидация полей от ошибок пользователя.

Практичность – приводится описание свойств, способствующих быстрому освоению, применению и эксплуатации программного продукта с минимальными трудозатратами с учетом характера решаемых задач и требований к квалификации обслуживающего персонала:

* интерфейс пользователя имеет простую структуру для понимания и простую навигацию.

Эффективность – приводится информация о характере поведения продукта, его степени удовлетворения потребности пользователя в обработке данных с учетом экономических, вычислительных и людских ресурсов:

* время отклика системы на запрос пользователя имеет первоначальный приоритет, так как информация с сервера о задачах должна приходить с минимальной для пользователя задержкой, чтобы избежать ожидания пользователя.

Мобильность - означает способность продукта быть системой, переносимой из одной аппаратно-программной среды в другую:

* Данный продукт не имеет возможности адаптивности под разные операционные системы, данное приложение работает на операционной системе Windows.

### 3.3 Оценка проекта

Оценка данного дипломного проекта произведена методом PERT. Входом для данного метода оценки служит список элементарных пакетов работ.

Перечень элементарных пакетов работ данного мобильного приложения для проведения оценки:

* разработка элементов интерфейса (9 шт.);
* разработка новых методов (10 шт.);
* разработка дизайна приложения в Figma (6 шт.);
* разработка базы данных (10 шт.);

Диапазон неопределенности достаточно охарактеризовать тремя оценками:

* Mi – Наиболее вероятная оценка трудозатрат;
* Oi - Минимально возможные трудозатраты на реализацию пакета работ;
* Pi – Максимально возможные трудозатраты.

Проведем оценку для лучших, худших и наиболее вероятных случаев.

Оценку средней трудоемкости по каждому элементарному пакету можно определить по формуле:

Ei = (Pi+4\*Mi + Oi)/6, (1)

где Pi – Максимально возможные трудозатраты,

Mi – Наиболее вероятная оценка трудозатрат,

Oi – Минимально возможные трудозатраты на реализацию пакета работ.

* 1. Разработка элементов интерфейса

Mi = 4

Oi =3

Pi = 6

Средняя трудоемкость рассчитана по формуле (1):

Еi = (Pi+4\*Mi + Oi)/6 = (4+4\*3+6)/6= 3,6 чел. час

* 1. Разработка новых методов

Mi = 7

Oi = 8

Pi = 10

Средняя трудоемкость рассчитана по формуле (1):

Еi = (Pi+4\*Mi + Oi)/6 = (7+4\*8+10)/6=8,16 чел. час

* 1. Разработка дизайна приложения в Figma

Mi = 2

Oi = 4

Pi = 6

Средняя трудоемкость рассчитана по формуле (1):

Еi = (Pi+4\*Mi + Oi)/6 = (2+4\*4+6)/6= 4 чел. час

* 1. Разработка базы данных

Mi = 2

Oi = 3

Pi = 4

Средняя трудоемкость рассчитана по формуле (1):

Еi = (Pi+4\*Mi + Oi)/6 = (2+4\*3+4)/6=3 чел. час

В таблице 1 показаны оценки наиболее вероятного и ожидаемого случаев.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Компоненты | Наиболее вероятный | Минимально возможный | Худший случай | Ожидаемый случай |
| Разработка элементов интерфейса | 4 | 3 | 6 | 3,6 |
| Разработка новых методов | 7 | 8 | 10 | 8,16 |
| Разработка дизайна приложения | 2 | 4 | 6 | 4 |
| Разработка базы данных | 2 | 3 | 4 | 3 |
| Итого | 45 | 37 | 60 | 46,2 |

Для расчета среднеквадратичного отклонения используется формула:

CKOi = (Pi - Oi)/6, (2)

где Pi – Максимально возможные трудозатраты

Oi - Минимально возможные трудозатраты на реализацию пакета работ.

1. Для разработки элементов интерфейса, среднеквадратичное отклонение рассчитано по формуле (2).

CKOi = (Pi - Oi)/6 = (6-4)/6=0,33 чел. час

1. Для разработки новых методов, среднеквадратичное отклонение рассчитано по формуле (2).

CKOi = (Pi - Oi)/6 = (10-7)/6=0,5 чел. час

1. Для разработки дизайна приложения в Figma, среднеквадратичное отклонение рассчитано по формуле (2).

CKOi = (Pi - Oi)/6 = (6-2)/6= 0,6 чел. час

1. Для разарботки базы данных, среднеквадратичное отклонение рассчитано по формуле (2).

CKOi = (Pi - Oi)/6 = (5-2)/6=0,33 чел. час

Суммарная трудоемкость проекта может быть рассчитана по формуле:

(3)

Е = Σ Ei

Е = Σ Ei = 9\*3,6+10\*8,1+6\*4+10\*3=168,6

Cреднеквадратичное отклонение для оценки суммарной трудоемкости будет составлять:

(4)

CKO = √ (∑CKOi)^2

где CKO – среднеквадратичное отклонение

Cреднеквадратичное отклонение для оценки суммарной трудоемкости рассчитана по формуле (4)

CKO = √ (∑CKOi)^2 =

= 2,69 чел. час

Оценка суммарной трудоемкости проекта, которая с вероятностью 95% точности расчета, можно применить формулу:

(5)

∑〖95%=E+2\*CKO〗

где CKO = Cреднеквадратичное отклонение

E = Ei

∑〖95%=E+2\*CKO〗 = (168,6+2\*2,69) = 174,1 чел. чаc

Полученную оценку трудоемкости кодирования необходимо умножить на четыре, поскольку помним, что кодирование составляет только 25% общих трудозатрат проекта. Поэтому суммарная трудоемкость нашего проекта составит, приблизительно = 696,2 чел.\*час.

В месяц сотрудник будет работать по проекту, примерно 144 чел.\*час/мес.

Следовательно, трудоемкость проекта в человеко-месяцах составит, приблизительно 696,2 /144 ~ 4,83

Тогда согласно формуле Б.Боэма оптимальная продолжительность проекта составит:

T = 2.5 \* (4,83) ^ 1/3=4,22 месяцев.

Средняя численность команды –4,83/4,2 ~ 1 человек.

### Заключение

В ходе разработки дипломного проекта было создано десктоп приложение, которое соответствует поставленным требованиям. Приложение имеет необходимый функционал, создано по современным стандартам разработки, имеет интуитивно понятный интерфейс. Все поставленные задачи были выполнены, цель достигнута.

Дизайн проекта разработан в Figma, разработка осуществлена в среде Visual Studio на языке программирования C#, на основе фреймворка Entity Framework. База данных разработана в Microsoft SQL Management Server.

В проекте также производятся расчеты трудоемкости при разработке данного программного продукта.

Доступ к приложению доступен под учетными данными:

Логин: admin

Пароль: admin

# Список используемых источников

1. ГОСТ 28195-89 «Оценка программных средств. Общие положения.»;
2. ГОСТ Р 59194-2020 «Управление требованиями. Основные положения»;
3. ГОСТ Р ИСО ТО 12182-2002 «Классификация программных средств»;
4. ГОСТ Р 56920-2016 «Системная и программная инженерия. Тестирование ПО»;
5. ГОСТ Р 57100-2016 «Системная и программная инженерия. Описание архитектуры»;
6. ГОСТ Р ИСО 9000-2008 «Системы менеджмента качества»;
7. ГОСТ Р ИСО/МЭК 12207-99 «Процессы жизненного цикла программных средств»;
8. Фленов Михаил Библия C#; БХВ-Петербург - М., 2017 .Гарнаев А. Самоучитель Visual Studio .NET 2017;
9. Лотка Р. C# и CSLA .NET Framework. Разработка бизнес-объектов; Диалектика / Вильямс - М. Алекс Макки Введение в .NET 4.0 и Visual Studio 2017 для профессионалов; Диалектика / Вильямс - М., 2017.
10. Подбельский, В. В. Язык С#. Базовый курс / В.В. Подбельский. - М.: Финансы и статистика, Инфра-М, 2020. - 384 c.
11. Рихтер, Джеффри CLR via C#. Программирование на платформе Microsoft .NET Framework 4.0 на языке C# / Джеффри Рихтер. - М.: Питер, 2019. - 928 c.
12. Орлов С.А. Программная инженерия: учебник для вузов/С.А.Орлов. - СПб.: Питер, 2018, – 640 с.:ил.
13. Тузовский А.Ф. Объектно- ориентированное программирование. Учебное пособие для прикладного бакалавриата – Томск: ЮРАЙТ, 2022. – 206 с.
14. Джон Microsoft Visual C#. Подробное руководство. 8-е изд. — СПб.: Питер, 2017. — 848 с.: ил.
15. Албахари, Джозеф C# 3.0. Справочник / Джозеф Албахари , Бен Албахари. - М.: БХВ-Петербург, 2012. - 944 c.
16. Альфред, В. Ахо Компиляторы. Принципы, технологии и инструментарий / Альфред В. Ахо и др. - М.: Вильямс, 2015. **- 266** c.
17. Работа с EntityFramework - https://metanit.com/sharp/entityframeworkcore/
18. Базовый курс по языку программирования c# - <https://habr.com/ru/c#/>
19. Информация о c# - <https://ru.wikipedia.org/wiki/C_Sharp>
20. Оффициальная документация по работе с MS SERVER - <https://www.microsoft.com/ru-ru/windows-server>