**Министерство образования Московской области**

**ГОУ ВО МО «Государственный гуманитарно-технологический университет» (ГГТУ) Ликино-Дулевский политехнический колледж- филиал ГГТУ**

Наименование колледжа/техникума

**О Т Ч Ё Т**

**ПО УЧЕБНОЙ ПРАКТИКЕ**

ПМ.01 Разработка модулей программного обеспечения для компьютерных систем

(код и наименование ПМ)

Обучающегося\_\_\_\_Соловьева Владимира Ивановича\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ф.И.О.

Курс \_\_\_\_3\_\_\_\_\_ группа \_\_\_ИСП.22А\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Специальность 09.02.07 Информационные системы и программирование \_\_\_

Место практики Ликино-Дулевский политехнический колледж – филиал ГГТУ

Период практики 25.11.2024 г. – 07.12.2024 г., 07.04.2025 г. - 19.04.2025 г.

Руководители практики

от колледжа \_\_\_\_\_\_\_\_ Кузьмина Елена Евгеньевна \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Гжегожевский Сергей Владимирович\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ветрова Карина Валерьевна\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Тишкин Валентин Васильевич\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ф.И.О.

г. Ликино-Дулево

2025 г.

Оглавление

[Введение 3](#_Toc184125536)

[1. Разработка технического задания 4](#_Toc184125537)

[2. Проектирование и создание базы данных 6](#_Toc184125538)

[3. Разработка интерфейса приложения 16](#_Toc184125539)

[4. Разработка программных модулей 21](#_Toc184125540)

[5. Разработка Api 44](#_Toc184125541)

[6. Библиотека классов 50](#_Toc184125542)

[Заключение 53](#_Toc184125543)

# Введение

Целью учебной практики является закрепление и углубление знаний, полученных обучающимися в процессе теоретического обучения по профессиональному модулю ПМ.01 «Разработка модулей программного обеспечения для компьютерных систем».

Задачи практики:

1. Получение первичных профессиональных умений и навыков;

2. Подготовка студентов к осознанному и глубокому изучению общепрофессиональных и специальных дисциплин;

3. Получение практических профессиональных умений и навыков по специальности 09.02.07 Информационные системы и программирование.

С целью овладения видом деятельности «Разработка модулей программного обеспечения для компьютерных систем» обучающийся должен обладать соответствующими профессиональными компетенциями:

ПК 1.1. Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием;

ПК 1.2. Разрабатывать программные модули в соответствии с техническим заданием;

ПК 1.3. Выполнять отладку программных модулей с использованием специализированных программных средств;

ПК 1.4. Выполнять тестирование программных модулей;

1. Разработка технического задания

Задание:  
Информационная система для маркетплейса для аренды вещей – площадка для сдачи и аренды различных предметов и оборудования.

1. Ключевые сущности выбранной предметной области:

Клиенты, арендодатели, оборудование, заказ

Процессы: просмотр доступного оборудования, просмотр совершенных заказов, добавление оборудования, оформление и редактирование заказа.

2. Состав диаграммы сущностей и связей (ER-диаграмма), отражающую взаимосвязь основных понятий.

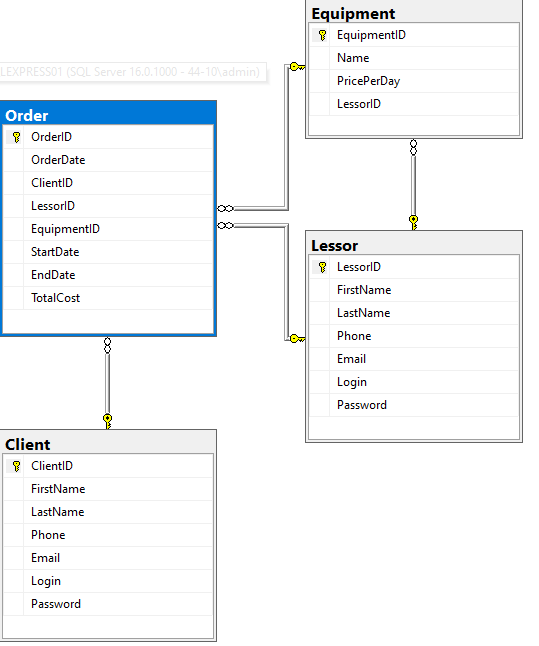


рис.1 «ER-диаграмма»

3. Разработка блок-схемы бизнес-процессов, иллюстрирующие взаимодействие участников системы.

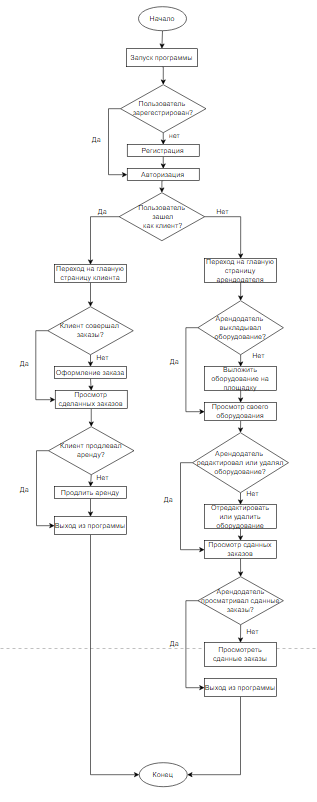


рис.2 «Блок-схема бизнес-процессов»

4. Описание функциональных требований.

1. Регистрация и авторизация
2. Добавление, редактирование и удаление оборудования
3. Оформление, удаление и редактирование заказа
4. Возможность печати информации о заказе

Описание нефункциональных требований.

1. Стабильная работа приложения на всех стадиях.
2. Работа приложения без задержек.

Анализ рисков и ограничений.

Возможны утечки данных пользователей из за халатности обращения к программе  
План тестирования.

Проверка всех функций на работоспособность и корректность

2. Проектирование и создание базы данных  
  
Цель:  
Разработка структуры базы данных для хранения данных выбранного программного продукта.

Задание:  
На основании описания предметной области, созданного ранее, разработайте базу данных, выполнив следующие шаги:  
  
СУБД: MS SQLServer 20  
- ER-диаграмма базы данных.

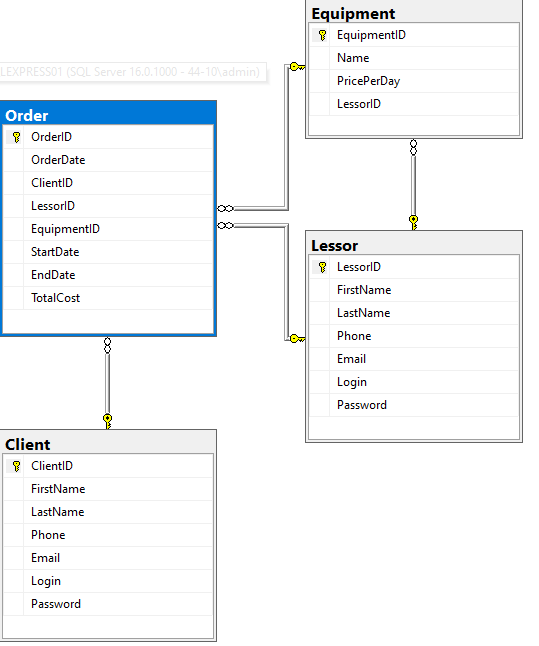


рис.3 «ER-диаграмма»

- Скрипт базы данных.

USE [EquipmentRent]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Client] Script Date: 03.12.2024 2:30:32 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Client](

[ClientID] [int] IDENTITY(1,1) NOT NULL,

[FirstName] [nvarchar](100) NOT NULL,

[LastName] [nvarchar](100) NOT NULL,

[Phone] [nvarchar](15) NULL,

[Email] [nvarchar](100) NULL,

[Login] [nvarchar](50) NOT NULL,

[Password] [nvarchar](255) NOT NULL,

PRIMARY KEY CLUSTERED

(

[ClientID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Equipment] Script Date: 03.12.2024 2:30:32 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Equipment](

[EquipmentID] [int] IDENTITY(1,1) NOT NULL,

[Name] [nvarchar](100) NOT NULL,

[PricePerDay] [decimal](10, 2) NOT NULL,

[LessorID] [int] NOT NULL,

PRIMARY KEY CLUSTERED

(

[EquipmentID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Lessor] Script Date: 03.12.2024 2:30:32 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Lessor](

[LessorID] [int] IDENTITY(1,1) NOT NULL,

[FirstName] [nvarchar](100) NOT NULL,

[LastName] [nvarchar](100) NOT NULL,

[Phone] [nvarchar](15) NULL,

[Email] [nvarchar](100) NULL,

[Login] [nvarchar](50) NOT NULL,

[Password] [nvarchar](255) NOT NULL,

PRIMARY KEY CLUSTERED

(

[LessorID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Order] Script Date: 03.12.2024 2:30:32 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Order](

[OrderID] [int] IDENTITY(1,1) NOT NULL,

[OrderDate] [datetime] NULL,

[ClientID] [int] NOT NULL,

[LessorID] [int] NOT NULL,

[EquipmentID] [int] NOT NULL,

[StartDate] [datetime] NOT NULL,

[EndDate] [datetime] NOT NULL,

[TotalCost] [decimal](10, 2) NOT NULL,

PRIMARY KEY CLUSTERED

(

[OrderID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

SET IDENTITY\_INSERT [dbo].[Client] ON

INSERT [dbo].[Client] ([ClientID], [FirstName], [LastName], [Phone], [Email], [Login], [Password]) VALUES (1, N'Иван', N'Иванов', N'89001112233', N'ivanov@mail.ru', N'ivanov', N'pass123')

INSERT [dbo].[Client] ([ClientID], [FirstName], [LastName], [Phone], [Email], [Login], [Password]) VALUES (2, N'Мария', N'Петрова', N'89004445566', N'petrova@mail.ru', N'petrova', N'qwerty')

INSERT [dbo].[Client] ([ClientID], [FirstName], [LastName], [Phone], [Email], [Login], [Password]) VALUES (3, N'Алексей', N'Смирнов', N'89007778899', N'smirnov@mail.ru', N'smirnov', N'12345678')

INSERT [dbo].[Client] ([ClientID], [FirstName], [LastName], [Phone], [Email], [Login], [Password]) VALUES (4, N'Андрей', N'Сергеев', N'89009991111', N'andrey@mail.ru', N'andrey', N'andrpass')

INSERT [dbo].[Client] ([ClientID], [FirstName], [LastName], [Phone], [Email], [Login], [Password]) VALUES (5, N'Ольга', N'Новикова', N'89001231234', N'novikova@mail.ru', N'novikova', N'olga2024')

INSERT [dbo].[Client] ([ClientID], [FirstName], [LastName], [Phone], [Email], [Login], [Password]) VALUES (6, N'Николай', N'Федоров', N'89003455678', N'fedorov@mail.ru', N'fedorov', N'passfed')

INSERT [dbo].[Client] ([ClientID], [FirstName], [LastName], [Phone], [Email], [Login], [Password]) VALUES (7, N'Володя', N'Соловушкин', NULL, NULL, N'SOLOVUSHKIN', N'ыщдщ123')

INSERT [dbo].[Client] ([ClientID], [FirstName], [LastName], [Phone], [Email], [Login], [Password]) VALUES (8, N'Артём', N'Козлов', N'89157834521', N'kozlov@mail.ru', N'kozlov\_art', N'pass789')

INSERT [dbo].[Client] ([ClientID], [FirstName], [LastName], [Phone], [Email], [Login], [Password]) VALUES (9, N'Евгения', N'Соколова', N'89263457812', N'sokolova@mail.ru', N'sok\_eva', N'eva2024')

INSERT [dbo].[Client] ([ClientID], [FirstName], [LastName], [Phone], [Email], [Login], [Password]) VALUES (10, N'Максим', N'Волков', N'89091234567', N'volkov@mail.ru', N'volkov\_max', N'max123')

INSERT [dbo].[Client] ([ClientID], [FirstName], [LastName], [Phone], [Email], [Login], [Password]) VALUES (11, N'София', N'Лебедева', N'89165438790', N'lebedeva@mail.ru', N'sofia\_leb', N'sof456')

INSERT [dbo].[Client] ([ClientID], [FirstName], [LastName], [Phone], [Email], [Login], [Password]) VALUES (12, N'Кирилл', N'Морозов', N'89257894561', N'morozov\_k@mail.ru', N'kirill\_m', N'kir789')

INSERT [dbo].[Client] ([ClientID], [FirstName], [LastName], [Phone], [Email], [Login], [Password]) VALUES (13, N'Дарья', N'Васильева', N'89164563278', N'vasileva@mail.ru', N'daria\_vas', N'dar234')

INSERT [dbo].[Client] ([ClientID], [FirstName], [LastName], [Phone], [Email], [Login], [Password]) VALUES (14, N'Глеб', N'Никитин', N'89263456789', N'nikitin@mail.ru', N'gleb\_nik', N'gleb567')

INSERT [dbo].[Client] ([ClientID], [FirstName], [LastName], [Phone], [Email], [Login], [Password]) VALUES (15, N'Полина', N'Зайцева', N'89165437890', N'zaiceva@mail.ru', N'polina\_z', N'pol890')

INSERT [dbo].[Client] ([ClientID], [FirstName], [LastName], [Phone], [Email], [Login], [Password]) VALUES (16, N'Тимур', N'Григорьев', N'89257893456', N'grigorev@mail.ru', N'timur\_g', N'tim123')

INSERT [dbo].[Client] ([ClientID], [FirstName], [LastName], [Phone], [Email], [Login], [Password]) VALUES (17, N'Алиса', N'Кузьмина', N'89091237890', N'kuzmina@mail.ru', N'alisa\_k', N'ali456')

SET IDENTITY\_INSERT [dbo].[Client] OFF

GO

SET IDENTITY\_INSERT [dbo].[Equipment] ON

INSERT [dbo].[Equipment] ([EquipmentID], [Name], [PricePerDay], [LessorID]) VALUES (1, N'Ноутбук Dell XPS 15', CAST(1500.00 AS Decimal(10, 2)), 1)

INSERT [dbo].[Equipment] ([EquipmentID], [Name], [PricePerDay], [LessorID]) VALUES (2, N'Фотоаппарат Canon EOS R', CAST(2000.00 AS Decimal(10, 2)), 2)

INSERT [dbo].[Equipment] ([EquipmentID], [Name], [PricePerDay], [LessorID]) VALUES (3, N'Проектор Epson EB-U50', CAST(1000.00 AS Decimal(10, 2)), 3)

INSERT [dbo].[Equipment] ([EquipmentID], [Name], [PricePerDay], [LessorID]) VALUES (4, N'Электрогитара Fender Stratocaster', CAST(2500.00 AS Decimal(10, 2)), 1)

INSERT [dbo].[Equipment] ([EquipmentID], [Name], [PricePerDay], [LessorID]) VALUES (5, N'Штатив Manfrotto Compact', CAST(300.00 AS Decimal(10, 2)), 2)

INSERT [dbo].[Equipment] ([EquipmentID], [Name], [PricePerDay], [LessorID]) VALUES (7, N'Видеокамера Sony FX3', CAST(3000.00 AS Decimal(10, 2)), 7)

INSERT [dbo].[Equipment] ([EquipmentID], [Name], [PricePerDay], [LessorID]) VALUES (8, N'Квадрокоптер DJI Mavic 3', CAST(2500.00 AS Decimal(10, 2)), 8)

INSERT [dbo].[Equipment] ([EquipmentID], [Name], [PricePerDay], [LessorID]) VALUES (9, N'Микрофон Shure SM7B', CAST(800.00 AS Decimal(10, 2)), 9)

INSERT [dbo].[Equipment] ([EquipmentID], [Name], [PricePerDay], [LessorID]) VALUES (10, N'Световой прибор Aputure 600d', CAST(1800.00 AS Decimal(10, 2)), 10)

INSERT [dbo].[Equipment] ([EquipmentID], [Name], [PricePerDay], [LessorID]) VALUES (11, N'Синтезатор Roland Jupiter-X', CAST(2200.00 AS Decimal(10, 2)), 7)

INSERT [dbo].[Equipment] ([EquipmentID], [Name], [PricePerDay], [LessorID]) VALUES (12, N'3D-принтер Prusa i3 MK3S+', CAST(1500.00 AS Decimal(10, 2)), 8)

INSERT [dbo].[Equipment] ([EquipmentID], [Name], [PricePerDay], [LessorID]) VALUES (13, N'Аудиоинтерфейс Universal Audio Apollo', CAST(1200.00 AS Decimal(10, 2)), 9)

INSERT [dbo].[Equipment] ([EquipmentID], [Name], [PricePerDay], [LessorID]) VALUES (14, N'Барабанная установка Pearl Masters', CAST(2800.00 AS Decimal(10, 2)), 10)

INSERT [dbo].[Equipment] ([EquipmentID], [Name], [PricePerDay], [LessorID]) VALUES (15, N'Телескоп Celestron EdgeHD', CAST(3500.00 AS Decimal(10, 2)), 7)

INSERT [dbo].[Equipment] ([EquipmentID], [Name], [PricePerDay], [LessorID]) VALUES (16, N'Игровая приставка PS5', CAST(1000.00 AS Decimal(10, 2)), 8)

SET IDENTITY\_INSERT [dbo].[Equipment] OFF

GO

SET IDENTITY\_INSERT [dbo].[Lessor] ON

INSERT [dbo].[Lessor] ([LessorID], [FirstName], [LastName], [Phone], [Email], [Login], [Password]) VALUES (1, N'Сергей', N'Морозов', N'89001234567', N'morozov@mail.ru', N'morozov', N'lessor123')

INSERT [dbo].[Lessor] ([LessorID], [FirstName], [LastName], [Phone], [Email], [Login], [Password]) VALUES (2, N'Елена', N'Кузнецова', N'89009876543', N'kuznecova@mail.ru', N'kuznecova', N'rent456')

INSERT [dbo].[Lessor] ([LessorID], [FirstName], [LastName], [Phone], [Email], [Login], [Password]) VALUES (3, N'Дмитрий', N'Васильев', N'89004567890', N'vasiliev@mail.ru', N'vasiliev', N'dmipass')

INSERT [dbo].[Lessor] ([LessorID], [FirstName], [LastName], [Phone], [Email], [Login], [Password]) VALUES (4, N'Анна', N'Фролова', N'89007654321', N'frolova@mail.ru', N'frolova', N'annapass')

INSERT [dbo].[Lessor] ([LessorID], [FirstName], [LastName], [Phone], [Email], [Login], [Password]) VALUES (5, N'Олег', N'Рябов', N'89006743210', N'ryabov@mail.ru', N'ryabov', N'ryabpass')

INSERT [dbo].[Lessor] ([LessorID], [FirstName], [LastName], [Phone], [Email], [Login], [Password]) VALUES (6, N'Монстер', N'Монстров', N'89687415266', N'vladimirsolovev817@gmail.com', N'monster', N'monster123')

INSERT [dbo].[Lessor] ([LessorID], [FirstName], [LastName], [Phone], [Email], [Login], [Password]) VALUES (7, N'Роман', N'Белов', N'89267894563', N'belov@mail.ru', N'roman\_b', N'rom789')

INSERT [dbo].[Lessor] ([LessorID], [FirstName], [LastName], [Phone], [Email], [Login], [Password]) VALUES (8, N'Марина', N'Комарова', N'89165432189', N'komarova@mail.ru', N'marina\_k', N'mar234')

INSERT [dbo].[Lessor] ([LessorID], [FirstName], [LastName], [Phone], [Email], [Login], [Password]) VALUES (9, N'Степан', N'Орлов', N'89257891234', N'orlov@mail.ru', N'stepan\_o', N'step567')

INSERT [dbo].[Lessor] ([LessorID], [FirstName], [LastName], [Phone], [Email], [Login], [Password]) VALUES (10, N'Вера', N'Миронова', N'89091234567', N'mironova@mail.ru', N'vera\_m', N'ver890')

INSERT [dbo].[Lessor] ([LessorID], [FirstName], [LastName], [Phone], [Email], [Login], [Password]) VALUES (11, N'Антон', N'Карпов', N'89165437889', N'karpov@mail.ru', N'anton\_k', N'ant123')

INSERT [dbo].[Lessor] ([LessorID], [FirstName], [LastName], [Phone], [Email], [Login], [Password]) VALUES (12, N'Юлия', N'Медведева', N'89267893456', N'medvedeva@mail.ru', N'yulia\_m', N'yul456')

INSERT [dbo].[Lessor] ([LessorID], [FirstName], [LastName], [Phone], [Email], [Login], [Password]) VALUES (13, N'Богдан', N'Соловьев', N'89257894567', N'solovev@mail.ru', N'bogdan\_s', N'bog789')

INSERT [dbo].[Lessor] ([LessorID], [FirstName], [LastName], [Phone], [Email], [Login], [Password]) VALUES (14, N'Ирина', N'Котова', N'89165432178', N'kotova@mail.ru', N'irina\_k', N'ir234')

INSERT [dbo].[Lessor] ([LessorID], [FirstName], [LastName], [Phone], [Email], [Login], [Password]) VALUES (15, N'Павел', N'Жуков', N'89091237856', N'zhukov@mail.ru', N'pavel\_zh', N'pav567')

INSERT [dbo].[Lessor] ([LessorID], [FirstName], [LastName], [Phone], [Email], [Login], [Password]) VALUES (16, N'Алёна', N'Щербакова', N'89267891234', N'scherbakova@mail.ru', N'alena\_sch', N'al890')

SET IDENTITY\_INSERT [dbo].[Lessor] OFF

GO

SET IDENTITY\_INSERT [dbo].[Order] ON

INSERT [dbo].[Order] ([OrderID], [OrderDate], [ClientID], [LessorID], [EquipmentID], [StartDate], [EndDate], [TotalCost]) VALUES (1, CAST(N'2024-11-26T08:53:43.607' AS DateTime), 1, 1, 1, CAST(N'2024-01-12T00:00:00.000' AS DateTime), CAST(N'2024-05-12T00:00:00.000' AS DateTime), CAST(7500.00 AS Decimal(10, 2)))

INSERT [dbo].[Order] ([OrderID], [OrderDate], [ClientID], [LessorID], [EquipmentID], [StartDate], [EndDate], [TotalCost]) VALUES (2, CAST(N'2024-11-26T08:53:43.607' AS DateTime), 2, 2, 2, CAST(N'2024-02-12T00:00:00.000' AS DateTime), CAST(N'2024-07-12T00:00:00.000' AS DateTime), CAST(12000.00 AS Decimal(10, 2)))

INSERT [dbo].[Order] ([OrderID], [OrderDate], [ClientID], [LessorID], [EquipmentID], [StartDate], [EndDate], [TotalCost]) VALUES (3, CAST(N'2024-11-26T08:53:43.607' AS DateTime), 3, 3, 3, CAST(N'2024-03-12T00:00:00.000' AS DateTime), CAST(N'2024-06-12T00:00:00.000' AS DateTime), CAST(3000.00 AS Decimal(10, 2)))

INSERT [dbo].[Order] ([OrderID], [OrderDate], [ClientID], [LessorID], [EquipmentID], [StartDate], [EndDate], [TotalCost]) VALUES (4, CAST(N'2024-11-26T08:53:43.607' AS DateTime), 4, 1, 4, CAST(N'2024-04-12T00:00:00.000' AS DateTime), CAST(N'2024-08-12T00:00:00.000' AS DateTime), CAST(10000.00 AS Decimal(10, 2)))

INSERT [dbo].[Order] ([OrderID], [OrderDate], [ClientID], [LessorID], [EquipmentID], [StartDate], [EndDate], [TotalCost]) VALUES (5, CAST(N'2024-11-26T08:53:43.607' AS DateTime), 5, 2, 5, CAST(N'2024-05-12T00:00:00.000' AS DateTime), CAST(N'2024-09-12T00:00:00.000' AS DateTime), CAST(1200.00 AS Decimal(10, 2)))

INSERT [dbo].[Order] ([OrderID], [OrderDate], [ClientID], [LessorID], [EquipmentID], [StartDate], [EndDate], [TotalCost]) VALUES (17, CAST(N'2024-12-03T02:28:51.943' AS DateTime), 8, 7, 7, CAST(N'2024-01-15T00:00:00.000' AS DateTime), CAST(N'2024-01-20T00:00:00.000' AS DateTime), CAST(7500.00 AS Decimal(10, 2)))

INSERT [dbo].[Order] ([OrderID], [OrderDate], [ClientID], [LessorID], [EquipmentID], [StartDate], [EndDate], [TotalCost]) VALUES (18, CAST(N'2024-12-03T02:28:51.943' AS DateTime), 9, 8, 7, CAST(N'2024-02-01T00:00:00.000' AS DateTime), CAST(N'2024-02-05T00:00:00.000' AS DateTime), CAST(4800.00 AS Decimal(10, 2)))

INSERT [dbo].[Order] ([OrderID], [OrderDate], [ClientID], [LessorID], [EquipmentID], [StartDate], [EndDate], [TotalCost]) VALUES (19, CAST(N'2024-12-03T02:28:51.943' AS DateTime), 10, 9, 8, CAST(N'2024-02-10T00:00:00.000' AS DateTime), CAST(N'2024-02-15T00:00:00.000' AS DateTime), CAST(14000.00 AS Decimal(10, 2)))

INSERT [dbo].[Order] ([OrderID], [OrderDate], [ClientID], [LessorID], [EquipmentID], [StartDate], [EndDate], [TotalCost]) VALUES (20, CAST(N'2024-12-03T02:28:51.943' AS DateTime), 8, 10, 9, CAST(N'2024-03-01T00:00:00.000' AS DateTime), CAST(N'2024-03-07T00:00:00.000' AS DateTime), CAST(24500.00 AS Decimal(10, 2)))

INSERT [dbo].[Order] ([OrderID], [OrderDate], [ClientID], [LessorID], [EquipmentID], [StartDate], [EndDate], [TotalCost]) VALUES (21, CAST(N'2024-12-03T02:28:51.943' AS DateTime), 9, 7, 10, CAST(N'2024-03-15T00:00:00.000' AS DateTime), CAST(N'2024-03-20T00:00:00.000' AS DateTime), CAST(5000.00 AS Decimal(10, 2)))

INSERT [dbo].[Order] ([OrderID], [OrderDate], [ClientID], [LessorID], [EquipmentID], [StartDate], [EndDate], [TotalCost]) VALUES (22, CAST(N'2024-12-03T02:28:51.943' AS DateTime), 10, 8, 7, CAST(N'2024-04-01T00:00:00.000' AS DateTime), CAST(N'2024-04-05T00:00:00.000' AS DateTime), CAST(7500.00 AS Decimal(10, 2)))

INSERT [dbo].[Order] ([OrderID], [OrderDate], [ClientID], [LessorID], [EquipmentID], [StartDate], [EndDate], [TotalCost]) VALUES (23, CAST(N'2024-12-03T02:28:51.943' AS DateTime), 8, 9, 7, CAST(N'2024-04-15T00:00:00.000' AS DateTime), CAST(N'2024-04-20T00:00:00.000' AS DateTime), CAST(6000.00 AS Decimal(10, 2)))

INSERT [dbo].[Order] ([OrderID], [OrderDate], [ClientID], [LessorID], [EquipmentID], [StartDate], [EndDate], [TotalCost]) VALUES (24, CAST(N'2024-12-03T02:28:51.943' AS DateTime), 9, 10, 8, CAST(N'2024-05-01T00:00:00.000' AS DateTime), CAST(N'2024-05-07T00:00:00.000' AS DateTime), CAST(19600.00 AS Decimal(10, 2)))

INSERT [dbo].[Order] ([OrderID], [OrderDate], [ClientID], [LessorID], [EquipmentID], [StartDate], [EndDate], [TotalCost]) VALUES (25, CAST(N'2024-12-03T02:28:51.943' AS DateTime), 10, 7, 9, CAST(N'2024-05-15T00:00:00.000' AS DateTime), CAST(N'2024-05-20T00:00:00.000' AS DateTime), CAST(17500.00 AS Decimal(10, 2)))

INSERT [dbo].[Order] ([OrderID], [OrderDate], [ClientID], [LessorID], [EquipmentID], [StartDate], [EndDate], [TotalCost]) VALUES (26, CAST(N'2024-12-03T02:28:51.943' AS DateTime), 8, 8, 10, CAST(N'2024-06-01T00:00:00.000' AS DateTime), CAST(N'2024-06-05T00:00:00.000' AS DateTime), CAST(4000.00 AS Decimal(10, 2)))

SET IDENTITY\_INSERT [dbo].[Order] OFF

GO

SET ANSI\_PADDING ON

GO

/\*\*\*\*\*\* Object: Index [UQ\_\_Client\_\_5E55825B94D86418] Script Date: 03.12.2024 2:30:32 \*\*\*\*\*\*/

ALTER TABLE [dbo].[Client] ADD UNIQUE NONCLUSTERED

(

[Login] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, SORT\_IN\_TEMPDB = OFF, IGNORE\_DUP\_KEY = OFF, ONLINE = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

GO

SET ANSI\_PADDING ON

GO

/\*\*\*\*\*\* Object: Index [UQ\_\_Lessor\_\_5E55825B2BDBBB9E] Script Date: 03.12.2024 2:30:32 \*\*\*\*\*\*/

ALTER TABLE [dbo].[Lessor] ADD UNIQUE NONCLUSTERED

(

[Login] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, SORT\_IN\_TEMPDB = OFF, IGNORE\_DUP\_KEY = OFF, ONLINE = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Order] ADD DEFAULT (getdate()) FOR [OrderDate]

GO

ALTER TABLE [dbo].[Equipment] WITH CHECK ADD FOREIGN KEY([LessorID])

REFERENCES [dbo].[Lessor] ([LessorID])

GO

ALTER TABLE [dbo].[Order] WITH CHECK ADD FOREIGN KEY([ClientID])

REFERENCES [dbo].[Client] ([ClientID])

GO

ALTER TABLE [dbo].[Order] WITH CHECK ADD FOREIGN KEY([EquipmentID])

REFERENCES [dbo].[Equipment] ([EquipmentID])

GO

ALTER TABLE [dbo].[Order] WITH CHECK ADD FOREIGN KEY([LessorID])

REFERENCES [dbo].[Lessor] ([LessorID])

GO

Подключение к проекту wpf

Nugget Для подключения БД требуются дополнительные пакеты Nugget



рис.4 «Пакеты Nugget»

После установки Nugget пакетов открываем консоль диспетчеров пакета и прописываем след.код EF:  
scaffold-DbContext "Server=44-10\\SQLEXPRESS;Database=EquipmentRent;TrustServerCertificate=True;Trusted\_Connection=True"

Модель базы данных в виде структуры классов:

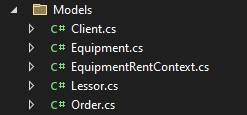


рис.5 « структура БД»

# 3. Разработка интерфейса приложения

Для разработки приложения wpf использовался стандартный шаблон:

Приложение WPF (Майкрософт)

LoginPage

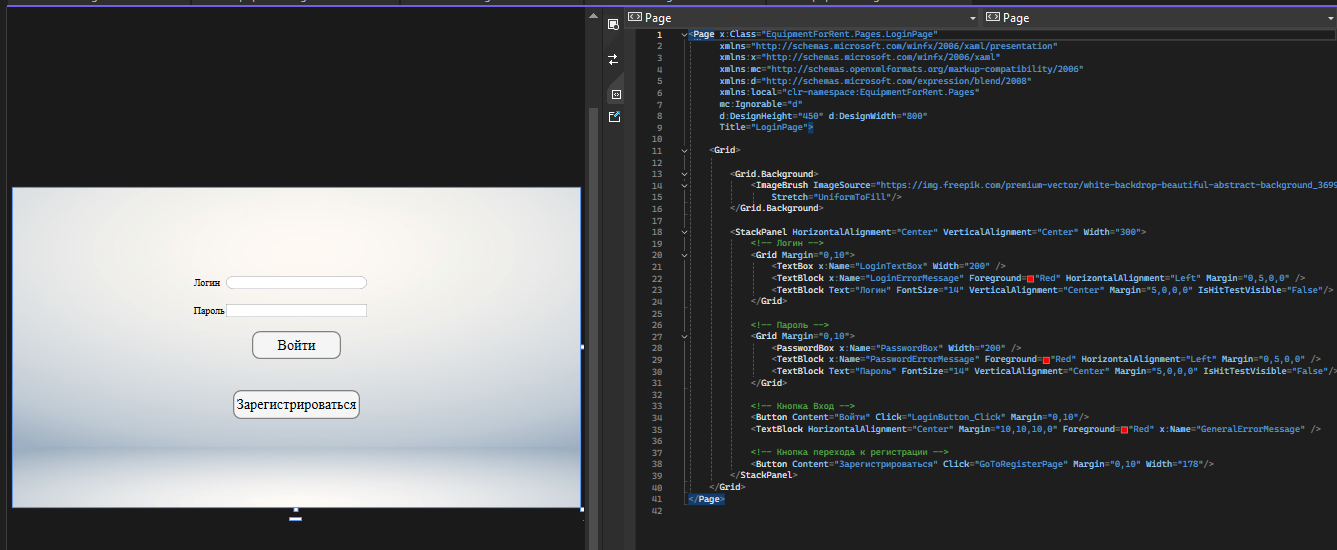


рис.6 «Страница авторизации»

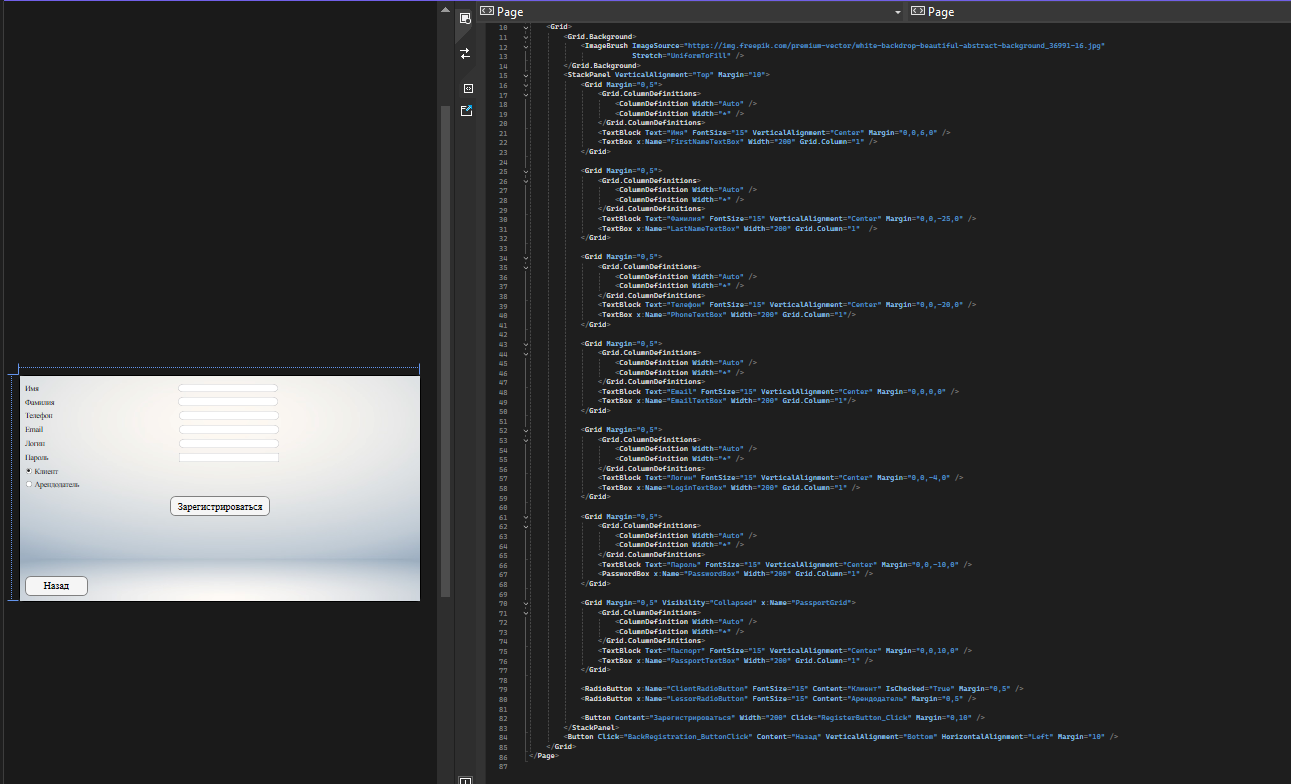
RegistrationPage   


рис.7 «Страница регистрации»

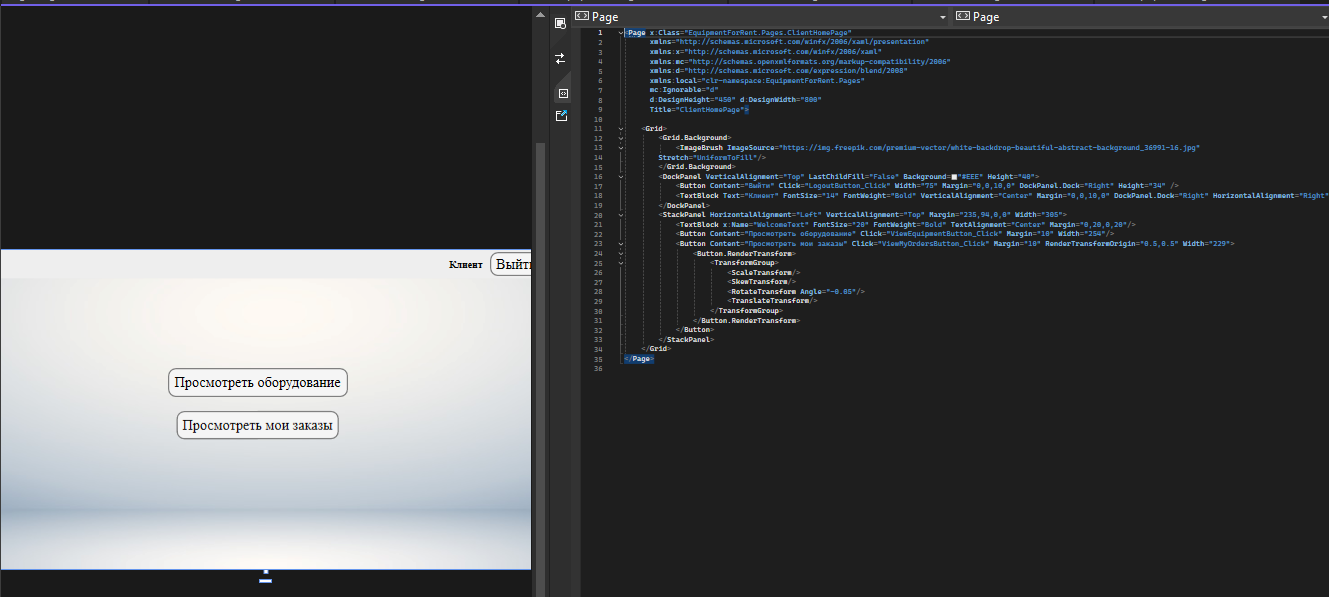
ClientHomePage  


рис.8 «Главная страница клиента»

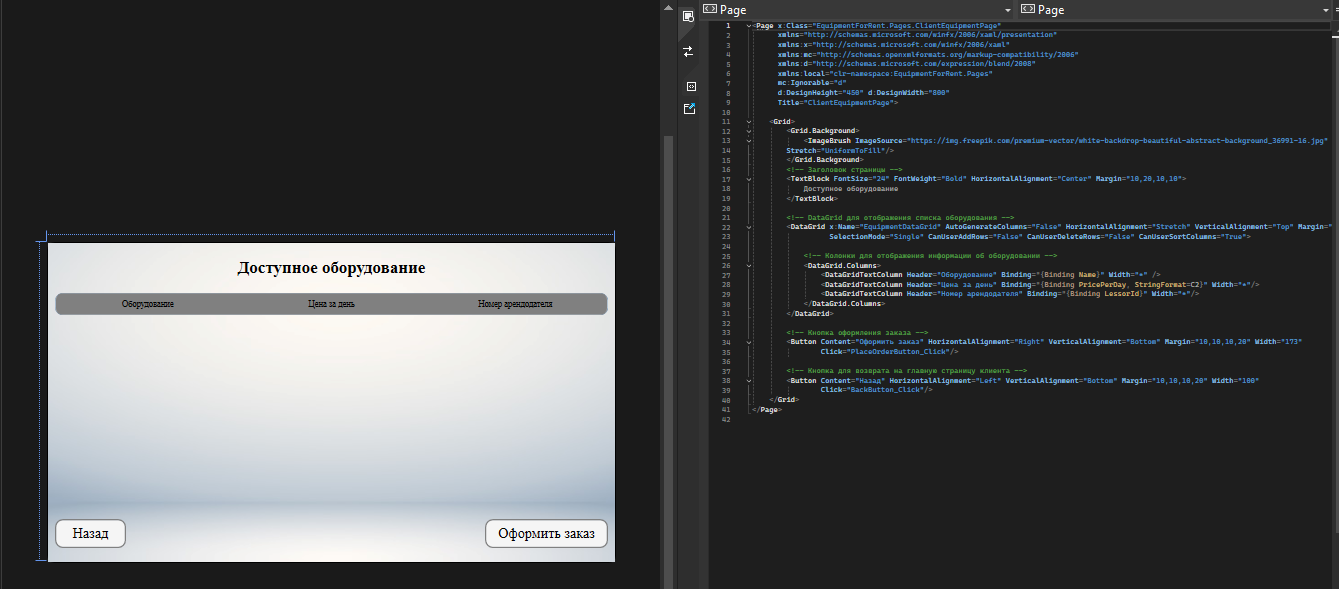
ClientEquipmentPage  


рис.9 «Страница доступное оборудование»

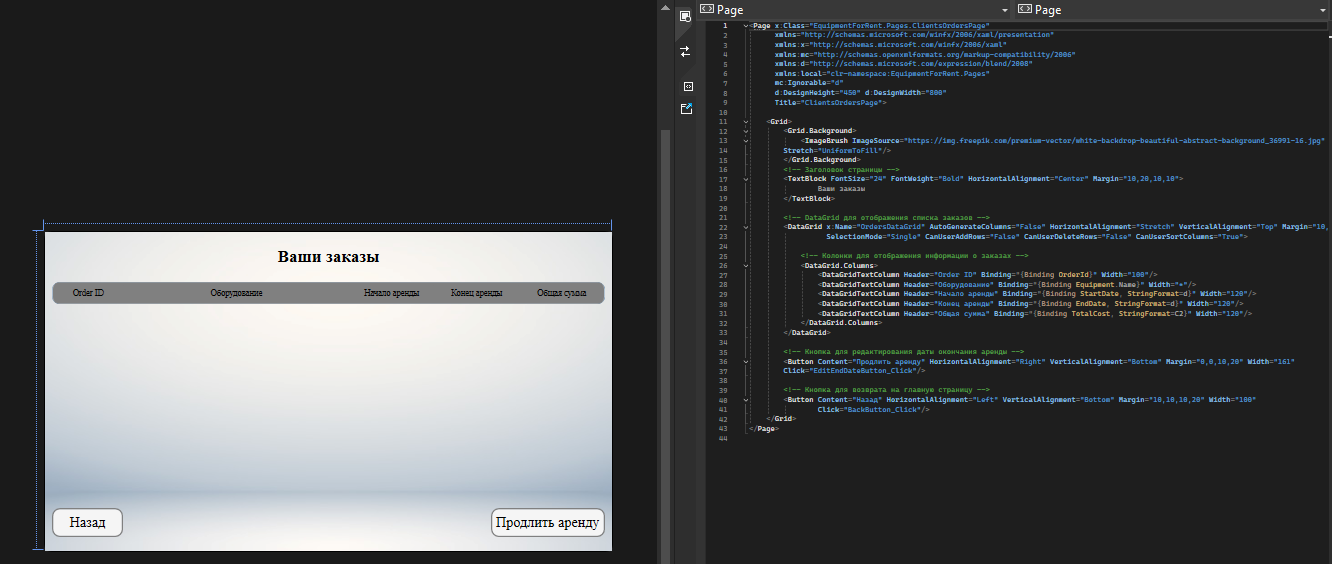
ClientOrdersPage  


рис.10 «Страница мои заказы»

LessorHomePage

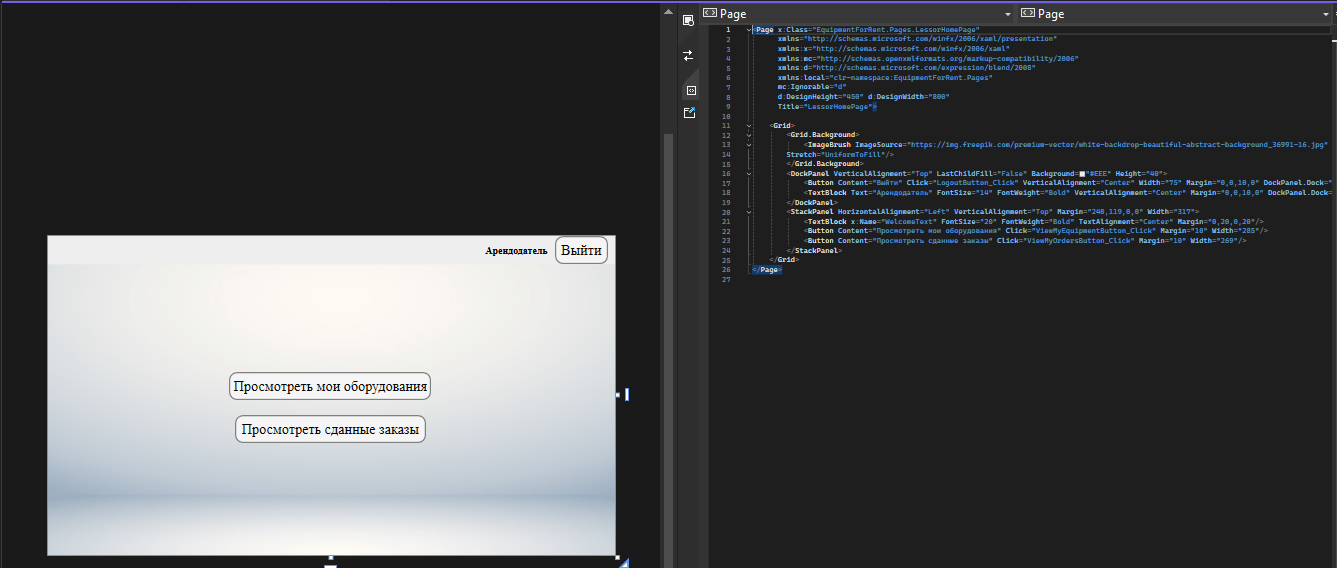


рис.11 «Главная страница Арендодателя»

LessorEquipmentPage

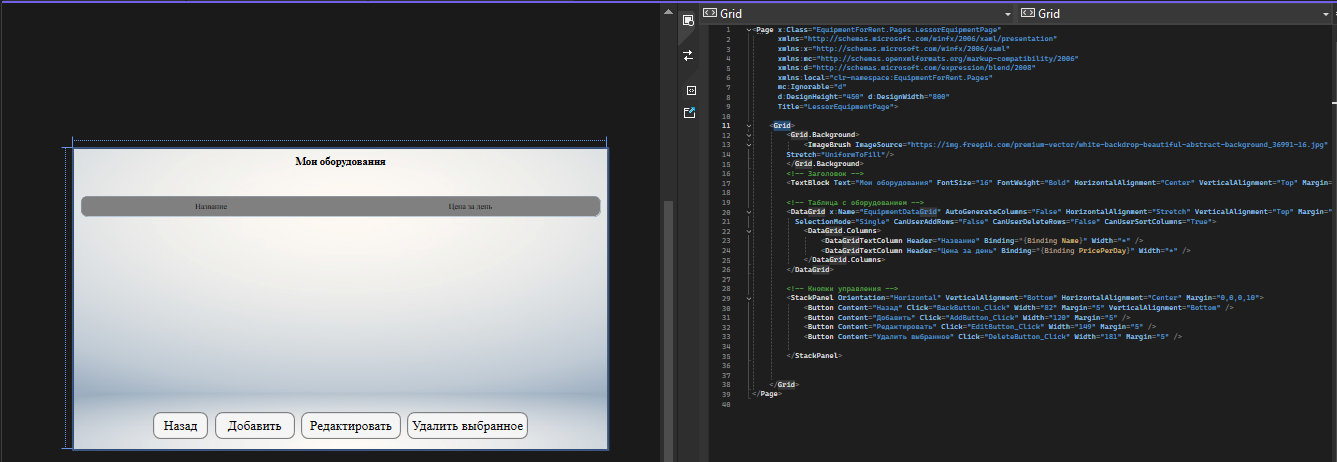


рис.12 «Страница мои оборудования»

LessorOrdersPage

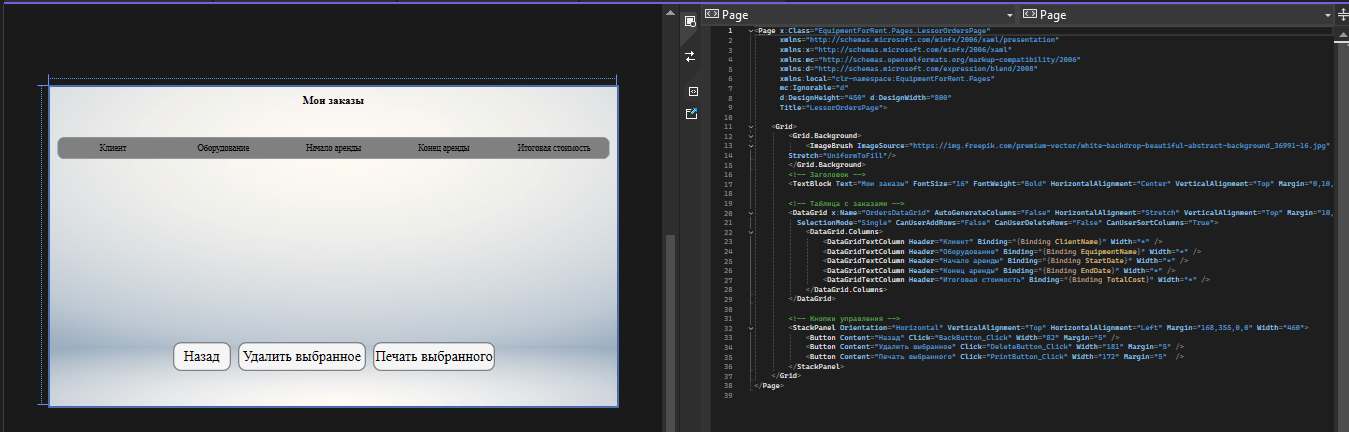


рис.13 «Страница мои оборудования»

DateTimePickerWindow

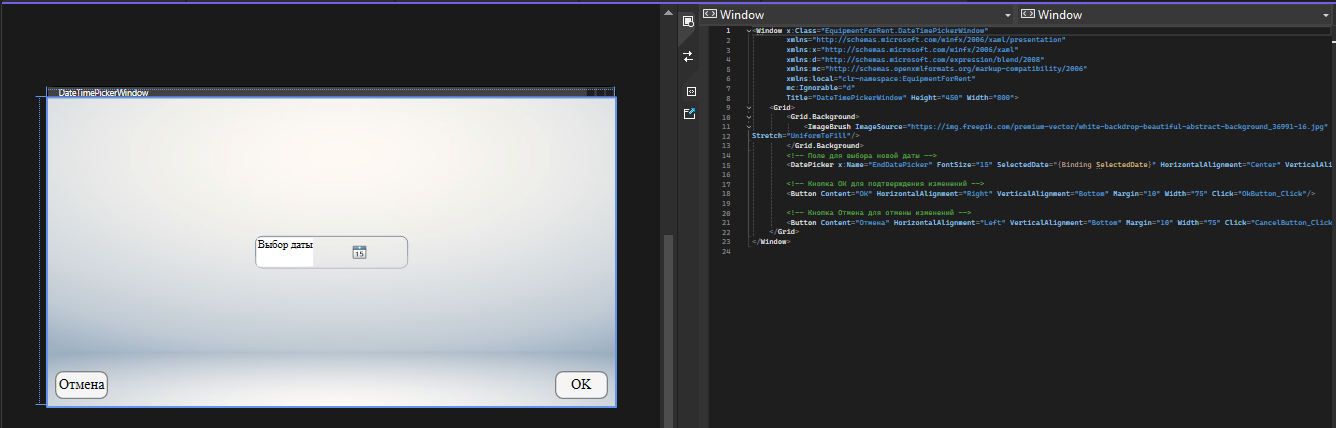


рис.14 «Окно продления аренды»

NewOrderWindow

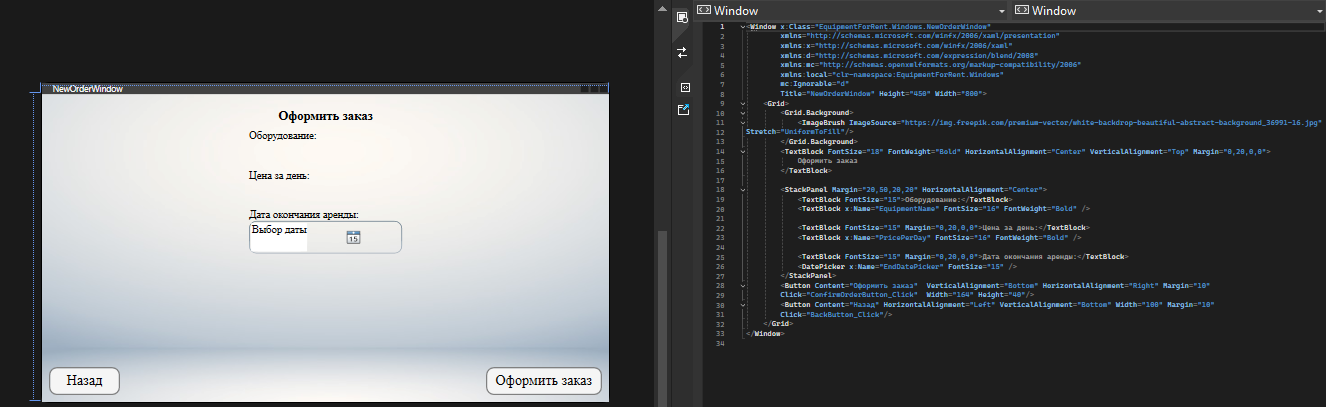


рис.15 «Окно оформления заказа»

NewEquipmentWindow

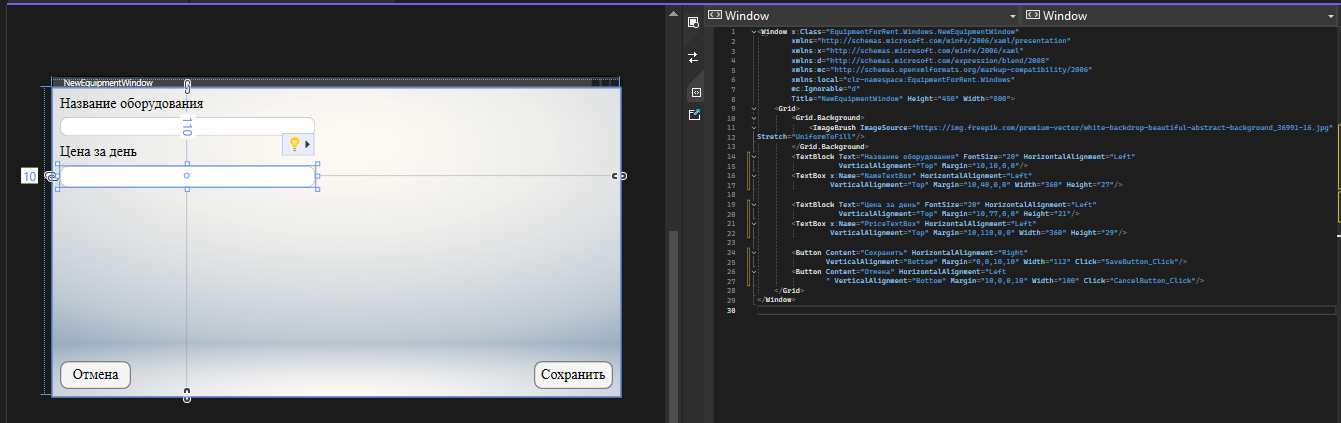


рис.16 «Окно добавления оборудования»

EditEquipmentWindow

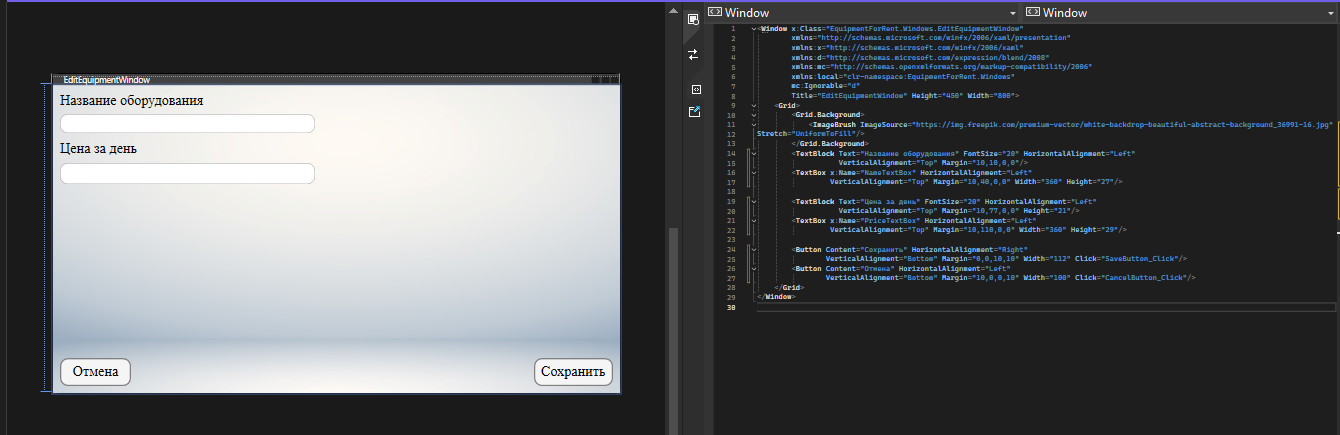


рис.17 «Окно редактирования оборудования»

# 4. Разработка программных модулей

Структура проекта

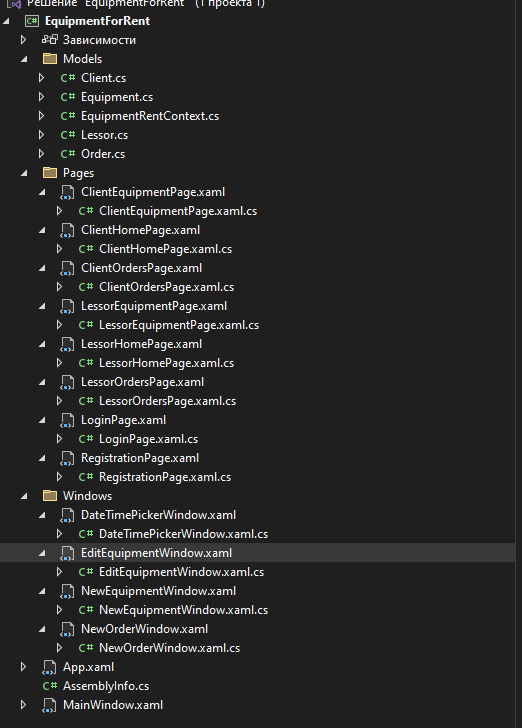


рис.18 «Структура программных модулей»

**Авторизация пользователя на странице LoginPage**

LoginPage.cs  
using EquipmentForRent.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace EquipmentForRent.Pages

{

/// <summary>

/// Логика взаимодействия для LoginPage.xaml

/// </summary>

public partial class LoginPage : Page

{

public LoginPage()

{

InitializeComponent();

}

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

{

string login = LoginTextBox.Text;

string password = PasswordBox.Password;

// Очистка сообщений об ошибках

LoginErrorMessage.Text = "";

PasswordErrorMessage.Text = "";

GeneralErrorMessage.Text = "";

if (string.IsNullOrEmpty(login) || string.IsNullOrEmpty(password))

{

GeneralErrorMessage.Text = "Логин и пароль обязательны!";

return;

}

using (var context = new EquipmentRentContext())

{

var client = context.Clients.FirstOrDefault(c => c.Login == login && c.Password == password);

var lessor = context.Lessors.FirstOrDefault(l => l.Login == login && l.Password == password);

if (client != null)

{

NavigationService.Navigate(new ClientHomePage(client));

}

else if (lessor != null)

{

NavigationService.Navigate(new LessorHomePage(lessor));

}

else

{

GeneralErrorMessage.Text = "Неверный логин или пароль!";

}

}

}

private void GoToRegisterPage(object sender, RoutedEventArgs e)

{

NavigationService.Navigate(new RegistrationPage());

}

}

}

**Регистрация пользователя**

RegistrationPage.cs

using EquipmentForRent.Models;

using Microsoft.Data.SqlClient;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace EquipmentForRent.Pages

{

/// <summary>

/// Логика взаимодействия для RegistrationPage.xaml

/// </summary>

public partial class RegistrationPage : Page

{

public RegistrationPage()

{

InitializeComponent();

}

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

{

// Собираем данные

string firstName = FirstNameTextBox.Text;

string lastName = LastNameTextBox.Text;

string phone = PhoneTextBox.Text;

string email = EmailTextBox.Text;

string login = LoginTextBox.Text;

string password = PasswordBox.Password;

string passport = PassportTextBox.Text;

if (ClientRadioButton.IsChecked == true)

{

// Регистрация клиента

RegisterClient(firstName, lastName, phone, email, login, password);

}

else if (LessorRadioButton.IsChecked == true)

{

// Регистрация арендодателя

RegisterLessor(firstName, lastName, phone, email, login, password, passport);

}

}

private void RegisterClient(string firstName, string lastName, string phone, string email, string login, string password)

{

// Добавление в таблицу Client

using (SqlConnection connection = new SqlConnection("Server=PC;Database=EquipmentRent;TrustServerCertificate=True;Trusted\_Connection=True;"))

{

connection.Open();

string query = "INSERT INTO Client (FirstName, LastName, Phone, Email, Login, Password) VALUES (@FirstName, @LastName, @Phone, @Email, @Login, @Password)";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@FirstName", firstName);

command.Parameters.AddWithValue("@LastName", lastName);

command.Parameters.AddWithValue("@Phone", phone);

command.Parameters.AddWithValue("@Email", email);

command.Parameters.AddWithValue("@Login", login);

command.Parameters.AddWithValue("@Password", password);

command.ExecuteNonQuery();

}

}

MessageBox.Show("Регистрация прошла успешно!", "Успех", MessageBoxButton.OK, MessageBoxImage.Information);

}

private void RegisterLessor(string firstName, string lastName, string phone, string email, string login, string password, string passport)

{

// Добавление в таблицу Lessor

using (SqlConnection connection = new SqlConnection("Server=PC;Database=EquipmentRent;TrustServerCertificate=True;Trusted\_Connection=True;"))

{

connection.Open();

string query = "INSERT INTO Lessor (FirstName, LastName, Phone, Email, Passport, Login, Password) VALUES (@FirstName, @LastName, @Phone, @Email, @Passport, @Login, @Password)";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@FirstName", firstName);

command.Parameters.AddWithValue("@LastName", lastName);

command.Parameters.AddWithValue("@Phone", phone);

command.Parameters.AddWithValue("@Email", email);

command.Parameters.AddWithValue("@Passport", passport);

command.Parameters.AddWithValue("@Login", login);

command.Parameters.AddWithValue("@Password", password);

command.ExecuteNonQuery();

}

}

MessageBox.Show("Регистрация прошла успешно!", "Успех", MessageBoxButton.OK, MessageBoxImage.Information);

}

private void BackRegistration\_ButtonClick(object sender, RoutedEventArgs e)

{

// Возвращаемся на страницу авторизации

NavigationService.Navigate(new LoginPage());

}

}

}

**Главная страница клиента**

ClientHomePage.cs

using EquipmentForRent.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace EquipmentForRent.Pages

{

public partial class ClientHomePage : Page

{

private Client \_loggedClient;

// Конструктор, принимающий объект Client

public ClientHomePage(Client loggedClient)

{

InitializeComponent();

\_loggedClient = loggedClient;

WelcomeText.Text = $"Здравствуйте! {\_loggedClient.FirstName} {\_loggedClient.LastName}";

}

// Конструктор, принимающий clientId (в случае, если нужно создать клиента из ID)

public ClientHomePage(int clientId)

{

InitializeComponent();

using (var context = new EquipmentRentContext())

{

\_loggedClient = context.Clients.FirstOrDefault(c => c.ClientId == clientId);

if (\_loggedClient != null)

{

WelcomeText.Text = $"Здравствуйте! {\_loggedClient.FirstName} {\_loggedClient.LastName}";

}

else

{

MessageBox.Show("Клиент не найден!");

}

}

}

// Обработчик для кнопки просмотра оборудования

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

{

// Передаем clientId в ClientEquipmentPage

NavigationService.Navigate(new ClientEquipmentPage(\_loggedClient.ClientId));

}

// Обработчик для кнопки просмотра заказов

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

{

// Передаем clientId в ClientOrdersPage

NavigationService.Navigate(new ClientsOrdersPage(\_loggedClient.ClientId));

}

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

{

// Возвращаемся на страницу авторизации

NavigationService.Navigate(new LoginPage());

}

}

}

**Главная страница арендодателя**

LessorHomePage.cs

using EquipmentForRent.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace EquipmentForRent.Pages

{

/// <summary>

/// Логика взаимодействия для LessorHomePage.xaml

/// </summary>

public partial class LessorHomePage : Page

{

private Lessor \_loggedLessor;

public LessorHomePage(Lessor loggedLessor)

{

InitializeComponent();

\_loggedLessor = loggedLessor;

WelcomeText.Text = $"Здравствуйте! {\_loggedLessor.FirstName} {\_loggedLessor.LastName}";

}

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

{

NavigationService.Navigate(new LessorEquipmentPage(\_loggedLessor.LessorId));

}

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

{

NavigationService.Navigate(new LessorOrdersPage(\_loggedLessor.LessorId));

}

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

{

// Возвращаемся на страницу авторизации

NavigationService.Navigate(new LoginPage());

}

}

}

**Страница с доступным оборудованием для клиента**

ClientEquipmentPage.cs

using EquipmentForRent.Models;

using EquipmentForRent.Windows;

using Microsoft.EntityFrameworkCore;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace EquipmentForRent.Pages

{

/// <summary>

/// Логика взаимодействия для ClientEquipmentPage.xaml

/// </summary>

public partial class ClientEquipmentPage : Page

{

private int \_clientId;

public ClientEquipmentPage(int clientId)

{

InitializeComponent();

\_clientId = clientId;

LoadEquipment();

}

private void LoadEquipment()

{

using (var context = new EquipmentRentContext())

{

var equipmentList = context.Equipment

.Include(e => e.Lessor) // Включаем данные о арендодателе

.ToList();

EquipmentDataGrid.ItemsSource = equipmentList;

}

}

// Обработчик для оформления заказа

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

{

if (EquipmentDataGrid.SelectedItem is Equipment selectedEquipment)

{

var newOrderWindow = new NewOrderWindow(

\_clientId,

selectedEquipment.EquipmentId,

selectedEquipment,

selectedEquipment.LessorId

);

newOrderWindow.ShowDialog();

}

else

{

MessageBox.Show("Пожалуйста, выберите оборудование для оформления заказа.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);

}

}

// Обработчик для возврата на главную страницу

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

{

NavigationService.Navigate(new ClientHomePage(\_clientId)); // Навигация на главную страницу клиента

}

}

}

**Страница с заказами клиента**

ClientOrdersPage.cs

using EquipmentForRent.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using EquipmentForRent.Pages;

using Microsoft.EntityFrameworkCore;

namespace EquipmentForRent.Pages

{

/// <summary>

/// Логика взаимодействия для ClientsOrdersPage.xaml

/// </summary>

public partial class ClientsOrdersPage : Page

{

private int \_clientId;

public ClientsOrdersPage(int clientId)

{

InitializeComponent();

\_clientId = clientId;

LoadOrders();

}

private void LoadOrders()

{

using (var context = new EquipmentRentContext())

{

var orders = context.Orders

.Where(o => o.ClientId == \_clientId)

.Include(o => o.Equipment) // Включаем данные об оборудовании

.ToList();

OrdersDataGrid.ItemsSource = orders;

}

}

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

{

if (OrdersDataGrid.SelectedItem is Order selectedOrder)

{

// Открываем окно выбора новой даты окончания аренды

var newEndDateWindow = new DateTimePickerWindow(selectedOrder.EndDate);

if (newEndDateWindow.ShowDialog() == true) // Если нажата кнопка "OK"

{

DateTime? newEndDate = newEndDateWindow.SelectedDate;

// Проверяем, что новая дата не раньше даты начала аренды

if (newEndDate.HasValue && newEndDate.Value < selectedOrder.StartDate)

{

MessageBox.Show("Дата окончания не может быть раньше даты начала", "Недопустимая дата", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

// Если дата окончания изменена

if (newEndDate.HasValue && newEndDate.Value != selectedOrder.EndDate)

{

// Перерасчитываем стоимость аренды

int rentalDays = (newEndDate.Value - selectedOrder.StartDate).Days;

if (rentalDays <= 0)

{

MessageBox.Show("Дата окончания должна быть позже даты начала.", "Неверная дата.", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

double dailyRate = (double)selectedOrder.Equipment.PricePerDay; // Используем PricePerDay вместо DailyRentalRate

double newTotalCost = rentalDays \* dailyRate; // Пересчитываем стоимость

// Обновляем дату окончания аренды и пересчитанную стоимость

selectedOrder.EndDate = newEndDate.Value;

selectedOrder.TotalCost = (decimal)newTotalCost; // Обновляем поле TotalCost

// Обновляем заказ в базе данных

using (var context = new EquipmentRentContext())

{

context.Orders.Update(selectedOrder); // Обновляем заказ в контексте

context.SaveChanges(); // Сохраняем изменения в базе данных

}

// Перезагружаем список заказов

LoadOrders();

}

}

else

{

// Если нажата кнопка "Cancel" или окно закрыто, ничего не меняем

MessageBox.Show("Изменения были отменены.", "Отмена", MessageBoxButton.OK, MessageBoxImage.Information);

}

}

}

// Обработчик кнопки "Back" для возврата на главную страницу клиента

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

{

NavigationService.Navigate(new ClientHomePage(\_clientId)); // Навигация на главную страницу клиента

}

}

}

**Страница с оборудованием арендодателя**

LessorEquipmentPage.cs

using EquipmentForRent.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using EquipmentForRent.Pages;

using Microsoft.Data.SqlClient;

using EquipmentForRent.Windows;

namespace EquipmentForRent.Pages

{

/// <summary>

/// Логика взаимодействия для LessorEquipmentPage.xaml

/// </summary>

public partial class LessorEquipmentPage : Page

{

public LessorEquipmentPage(int lessorId)

{

InitializeComponent();

LoadEquipmentData();

}

public void LoadEquipmentData()

{

string connectionString = "Server=PC;Database=EquipmentRent;TrustServerCertificate=True;Trusted\_Connection=True;"; // Укажите строку подключения к базе данных

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "SELECT EquipmentID, Name, PricePerDay FROM Equipment WHERE LessorID = @LessorID";

SqlCommand command = new SqlCommand(query, connection);

command.Parameters.AddWithValue("@LessorID", 1); // Передайте ID арендатора

SqlDataReader reader = command.ExecuteReader();

var equipmentList = new System.Collections.Generic.List<Equipment>();

while (reader.Read())

{

equipmentList.Add(new Equipment

{

EquipmentId = reader.GetInt32(0),

Name = reader.GetString(1),

PricePerDay = reader.GetDecimal(2)

});

}

EquipmentDataGrid.ItemsSource = equipmentList;

}

}

// Обработчик кнопки "Назад"

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

{

// Переход на предыдущую страницу (например, LessorMainPage)

NavigationService.GoBack();

}

// Обработчик кнопки "Добавить"

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

{

// Переход на страницу добавления нового оборудования (NewEquipmentWindow)

NewEquipmentWindow newEquipmentWindow = new NewEquipmentWindow(this); // Передаем текущую страницу как parentPage

newEquipmentWindow.ShowDialog(); // Открыть в диалоговом режиме

}

// Обработчик кнопки "Редактировать"

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

{

// Проверяем, что элемент выбран

var selectedEquipment = EquipmentDataGrid.SelectedItem as Equipment;

if (selectedEquipment == null)

{

MessageBox.Show("Пожалуйста, выберите оборудование для редактирования.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);

return;

}

// Открываем окно редактирования с передачей выбранного оборудования

EditEquipmentWindow editEquipmentWindow = new EditEquipmentWindow(selectedEquipment);

if (editEquipmentWindow.ShowDialog() == true)

{

// Если редактирование было успешно завершено, обновляем данные в таблице

LoadEquipmentData();

}

}

// Обработчик кнопки "Удалить выбранное"

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

{

var selectedEquipment = EquipmentDataGrid.SelectedItem as Equipment;

if (selectedEquipment != null)

{

// Подтверждение удаления

MessageBoxResult result = MessageBox.Show("Вы уверены, что хотите удалить это оборудование?", "Удалить", MessageBoxButton.YesNo);

if (result == MessageBoxResult.Yes)

{

// Удаление из базы данных

string connectionString = "Server=PC;Database=EquipmentRent;TrustServerCertificate=True;Trusted\_Connection=True;"; // Укажите строку подключения

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "DELETE FROM Equipment WHERE EquipmentID = @EquipmentID";

SqlCommand command = new SqlCommand(query, connection);

command.Parameters.AddWithValue("@EquipmentID", selectedEquipment.EquipmentId);

command.ExecuteNonQuery();

}

// Обновить список оборудования

LoadEquipmentData();

}

}

else

{

MessageBox.Show("Пожалуйста, выберите оборудование для удаления.");

}

}

}

}

**Страница с заказами арендодателя**

LessorOrdersPage.cs

using EquipmentForRent.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace EquipmentForRent.Pages

{

/// <summary>

/// Логика взаимодействия для LessorOrdersPage.xaml

/// </summary>

public partial class LessorOrdersPage : Page

{

private int \_lessorID;

public LessorOrdersPage(int lessorID)

{

InitializeComponent();

\_lessorID = lessorID;

LoadOrdersData();

}

private void LoadOrdersData()

{

using (var db = new EquipmentRentContext())

{

// Выгружаем все заказы, относящиеся к текущему арендодателю

var orders = db.Orders

.Where(o => o.LessorId == \_lessorID)

.Select(o => new

{

o.OrderId,

ClientName = o.Client.FirstName + " " + o.Client.LastName,

EquipmentName = o.Equipment.Name,

o.StartDate,

o.EndDate,

o.TotalCost

}).ToList();

// Привязываем данные к таблице

OrdersDataGrid.ItemsSource = orders;

}

}

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

{

// Получаем выделенные строки

var selectedOrders = OrdersDataGrid.SelectedItems.Cast<dynamic>().ToList();

if (selectedOrders.Count == 0)

{

MessageBox.Show("Выберите заказы для удаления.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

var result = MessageBox.Show($"Вы уверены, что хотите удалить {selectedOrders.Count} заказ(ов)?",

"Подтверждение удаления",

MessageBoxButton.YesNo,

MessageBoxImage.Question);

if (result == MessageBoxResult.Yes)

{

using (var db = new EquipmentRentContext())

{

foreach (var order in selectedOrders)

{

int orderId = order.OrderId;

var dbOrder = db.Orders.FirstOrDefault(o => o.OrderId == orderId);

if (dbOrder != null)

{

db.Orders.Remove(dbOrder);

}

}

db.SaveChanges();

}

MessageBox.Show("Выбранные заказы успешно удалены.", "Успех", MessageBoxButton.OK, MessageBoxImage.Information);

// Обновляем таблицу

LoadOrdersData();

}

}

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

{

// Переход на домашнюю страницу арендодателя

NavigationService.GoBack();

}

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

{

// Получаем выбранный заказ

var selectedOrder = OrdersDataGrid.SelectedItem as dynamic;

if (selectedOrder == null)

{

MessageBox.Show("Пожалуйста, выберите заказ для печати.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

// Создаем документ для печати

PrintDialog printDialog = new PrintDialog();

if (printDialog.ShowDialog() == true)

{

FlowDocument document = new FlowDocument

{

PagePadding = new Thickness(50),

FontSize = 14,

FontFamily = new FontFamily("Segoe UI")

};

// Добавляем заголовок

Paragraph header = new Paragraph(new Bold(new Run("Информация о заказе")))

{

FontSize = 18,

TextAlignment = TextAlignment.Center,

Margin = new Thickness(0, 0, 0, 20)

};

document.Blocks.Add(header);

// Добавляем информацию о заказе

document.Blocks.Add(new Paragraph(new Run($"Клиент: {selectedOrder.ClientName}")));

document.Blocks.Add(new Paragraph(new Run($"Оборудование: {selectedOrder.EquipmentName}")));

document.Blocks.Add(new Paragraph(new Run($"Начало аренды: {selectedOrder.StartDate:dd.MM.yyyy}")));

document.Blocks.Add(new Paragraph(new Run($"Конец аренды: {selectedOrder.EndDate:dd.MM.yyyy}")));

document.Blocks.Add(new Paragraph(new Run($"Итоговая стоимость cтоимость: {selectedOrder.TotalCost:C}")));

// Печать документа

IDocumentPaginatorSource idpSource = document;

printDialog.PrintDocument(idpSource.DocumentPaginator, "Печать заказа");

}

}

}

}

**Окно с редактированием(продлением) заказа для клиента**

DateTimePickerWindow.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace EquipmentForRent

{

/// <summary>

/// Логика взаимодействия для DateTimePickerWindow.xaml

/// </summary>

public partial class DateTimePickerWindow : Window

{

public DateTime? SelectedDate { get; set; }

public DateTimePickerWindow(DateTime? initialDate)

{

InitializeComponent();

SelectedDate = initialDate;

EndDatePicker.SelectedDate = initialDate;

}

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

{

// Получаем выбранную дату из DatePicker

SelectedDate = EndDatePicker.SelectedDate;

DialogResult = true; // Закрытие окна с подтверждением изменений

}

// Обработчик кнопки "Cancel" - отменяем изменения и закрываем окно

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

{

DialogResult = false; // Закрытие окна без изменений

}

}

}

**Окно с оформлением заказа для клиента**

NewOrderWindow.cs

using EquipmentForRent.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace EquipmentForRent.Windows

{

/// <summary>

/// Логика взаимодействия для NewOrderWindow.xaml

/// </summary>

public partial class NewOrderWindow : Window

{

private Equipment \_selectedEquipment;

private int \_clientId;

public NewOrderWindow(int clientId, int equipmentId, Equipment equipment, int lessorId)

{

InitializeComponent();

\_selectedEquipment = equipment;

\_clientId = clientId;

// Инициализация данных

EquipmentName.Text = \_selectedEquipment.Name;

PricePerDay.Text = \_selectedEquipment.PricePerDay.ToString("C2");

EndDatePicker.SelectedDate = DateTime.Now.AddDays(1); // По умолчанию завтрашний день

}

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

{

DateTime? endDate = EndDatePicker.SelectedDate;

if (!endDate.HasValue || endDate <= DateTime.Now)

{

MessageBox.Show("Дата окончания аренды должна быть позже сегодняшнего дня.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

// Расчет количества дней аренды, включая начальную и конечную даты

int totalDays = (endDate.Value - DateTime.Now.Date).Days + 1;

// Проверка на корректное количество дней

if (totalDays <= 0)

{

MessageBox.Show("Дата окончания аренды должна быть позже сегодняшнего дня.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

// Расчет общей стоимости

decimal totalCost = \_selectedEquipment.PricePerDay \* totalDays;

var newOrder = new Order

{

ClientId = \_clientId,

EquipmentId = \_selectedEquipment.EquipmentId,

LessorId = \_selectedEquipment.LessorId,

StartDate = DateTime.Now,

EndDate = endDate.Value,

TotalCost = totalCost

};

using (var context = new EquipmentRentContext())

{

context.Orders.Add(newOrder);

context.SaveChanges();

}

MessageBox.Show($"Заказ успешно оформлен!\nИтоговая стоимость аренды: {totalCost:C2}",

"Успех", MessageBoxButton.OK, MessageBoxImage.Information);

Close();

}

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

{

Close(); // Закрытие окна оформления заказа

}

}

}

**Окно с добавлением оборудования для арендодателя**

NewEquipmentWindow.cs

using EquipmentForRent.Pages;

using Microsoft.Data.SqlClient;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace EquipmentForRent.Windows

{

/// <summary>

/// Логика взаимодействия для NewEquipmentWindow.xaml

/// </summary>

public partial class NewEquipmentWindow : Window

{

private LessorEquipmentPage \_parentPage;

public NewEquipmentWindow(LessorEquipmentPage parentPage)

{

InitializeComponent();

\_parentPage = parentPage;

}

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

{

string name = NameTextBox.Text;

string priceText = PriceTextBox.Text;

if (string.IsNullOrEmpty(name) || string.IsNullOrEmpty(priceText) || !decimal.TryParse(priceText, out decimal price))

{

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

return;

}

if (price <= 0)

{

MessageBox.Show("Цена за день должна быть больше нуля.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

// Добавление нового оборудования в базу данных

string connectionString = "Server=PC;Database=EquipmentRent;TrustServerCertificate=True;Trusted\_Connection=True;"; // Укажите строку подключения

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "INSERT INTO Equipment (Name, PricePerDay, LessorID) VALUES (@Name, @PricePerDay, @LessorID)";

SqlCommand command = new SqlCommand(query, connection);

command.Parameters.AddWithValue("@Name", name);

command.Parameters.AddWithValue("@PricePerDay", price);

command.Parameters.AddWithValue("@LessorID", 1); // Пример ID арендатора

command.ExecuteNonQuery();

}

// Отображаем всплывающее окно с уведомлением

MessageBox.Show("Успешно сохранено!", "Информация", MessageBoxButton.OK, MessageBoxImage.Information);

// Обновляем список оборудования в родительской странице

\_parentPage.LoadEquipmentData();

// Закрытие окна после сохранения

this.Close();

}

// Обработчик кнопки "Отмена"

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

{

this.Close();

}

}

}

**Окно с редактированием оборудования для арендодателя**

EditEquipmentWindow.cs

using EquipmentForRent.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace EquipmentForRent.Windows

{

/// <summary>

/// Логика взаимодействия для EditEquipmentWindow.xaml

/// </summary>

public partial class EditEquipmentWindow : Window

{

private Equipment \_equipment;

public EditEquipmentWindow(Equipment equipment)

{

InitializeComponent();

\_equipment = equipment;

// Заполнение полей окна данными выбранного оборудования

NameTextBox.Text = \_equipment.Name;

PriceTextBox.Text = \_equipment.PricePerDay.ToString();

}

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

{

this.Close();

}

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

{

try

{

string name = NameTextBox.Text;

string priceText = PriceTextBox.Text;

if (string.IsNullOrEmpty(name) || string.IsNullOrEmpty(priceText) || !decimal.TryParse(priceText, out decimal price))

{

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

return;

}

if (price <= 0)

{

MessageBox.Show("Цена за день должна быть больше нуля.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

// Обновление данных оборудования

\_equipment.Name = name;

\_equipment.PricePerDay = price;

using (var db = new EquipmentRentContext())

{

var equipmentToUpdate = db.Equipment.FirstOrDefault(eq => eq.EquipmentId == \_equipment.EquipmentId);

if (equipmentToUpdate != null)

{

equipmentToUpdate.Name = \_equipment.Name;

equipmentToUpdate.PricePerDay = \_equipment.PricePerDay;

db.SaveChanges();

}

}

// Сообщение об успешном сохранении

MessageBox.Show("Изменения успешно сохранены!", "Успех", MessageBoxButton.OK, MessageBoxImage.Information);

DialogResult = true;

Close();

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при сохранении изменений: {ex.Message}", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

}

}

# 5. Разработка Api

API — это набор правил и и протоколов, которые позволяют различным системам обмениваться информацией между собой. API — это общий термин, который применяется в контексте совершенно разных систем.

Swagger — это инструмент, который помогает разработчикам создавать, документировать и проверять API.

Для работы с API использовался шаблон Веб-API ASP.NET Core (Майкрософт)

Структура

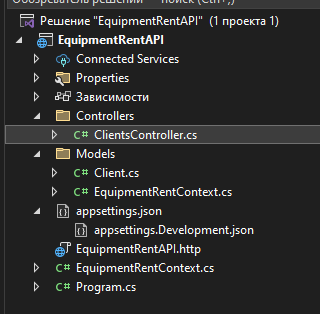


рис.19 «структура API»

Код ClientsController:

using EquipmentRentAPI.Models;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

[Route("api/[controller]")]

[ApiController]

public class ClientsController : ControllerBase

{

private readonly EquipmentRentContext \_context;

public ClientsController(EquipmentRentContext context)

{

\_context = context;

}

// GET: api/clients

/// <summary>

/// Получить список всех клиентов.

/// </summary>

/// <returns>Список клиентов</returns>

[HttpGet]

public async Task<ActionResult<IEnumerable<Client>>> GetAllClients()

{

var clients = await \_context.Clients.ToListAsync();

if (clients == null || !clients.Any())

{

return NotFound("Клиенты не найдены");

}

return Ok(clients); // Возвращаем 200 OK с данными клиентов

}

// GET: api/clients/{id}

/// <summary>

/// Получить клиента по ID.

/// </summary>

/// <param name="id">ID клиента</param>

/// <returns>Клиент с указанным ID</returns>

[HttpGet("{id}")]

public async Task<ActionResult<Client>> GetClientById(int id)

{

var client = await \_context.Clients.FindAsync(id);

if (client == null)

{

return NotFound(); // Возвращаем 404, если клиент не найден

}

return Ok(client); // Возвращаем 200 OK с данными клиента

}

// POST: api/clients

/// <summary>

/// Создать нового клиента.

/// </summary>

/// <param name="client">Данные клиента для создания</param>

/// <returns>Созданный клиент</returns>

[HttpPost]

public async Task<IActionResult> CreateClient(Client client)

{

\_context.Clients.Add(client);

await \_context.SaveChangesAsync();

return CreatedAtAction(nameof(GetClientById), new { id = client.ClientId }, client); // Возвращаем 201 Created

}

// PUT: api/clients/{id}

/// <summary>

/// Обновить данные клиента по ID.

/// </summary>

/// <param name="id">ID клиента</param>

/// <param name="client">Обновленные данные клиента</param>

/// <returns>Результат выполнения операции</returns>

[HttpPut("{id}")]

public async Task<IActionResult> UpdateClient(int id, [FromBody] Client client)

{

if (id != client.ClientId)

{

// Возвращаем ошибку, если ID в URL не совпадает с ID в теле запроса

return BadRequest("ID в маршруте не совпадает с ID в теле запроса.");

}

\_context.Entry(client).State = EntityState.Modified;

try

{

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!\_context.Clients.Any(e => e.ClientId == id))

{

return NotFound();

}

else

{

throw;

}

}

return NoContent();

}

// DELETE: api/clients/{id}

/// <summary>

/// Удалить клиента по ID.

/// </summary>

/// <param name="id">ID клиента</param>

/// <returns>Результат выполнения операции</returns>

[HttpDelete("{id}")]

public async Task<IActionResult> DeleteClient(int id)

{

var client = await \_context.Clients.FindAsync(id);

if (client == null)

{

return NotFound(); // Возвращаем 404, если клиент не найден

}

\_context.Clients.Remove(client);

await \_context.SaveChangesAsync();

return NoContent(); // Возвращаем 204 No Content, если удаление прошло успешно

}

// Проверка, существует ли клиент с указанным ID

private bool ClientExists(int id)

{

return \_context.Clients.Any(e => e.ClientId == id); // Проверка, существует ли клиент с данным ID

}

}

Проверка работы

1. GET all

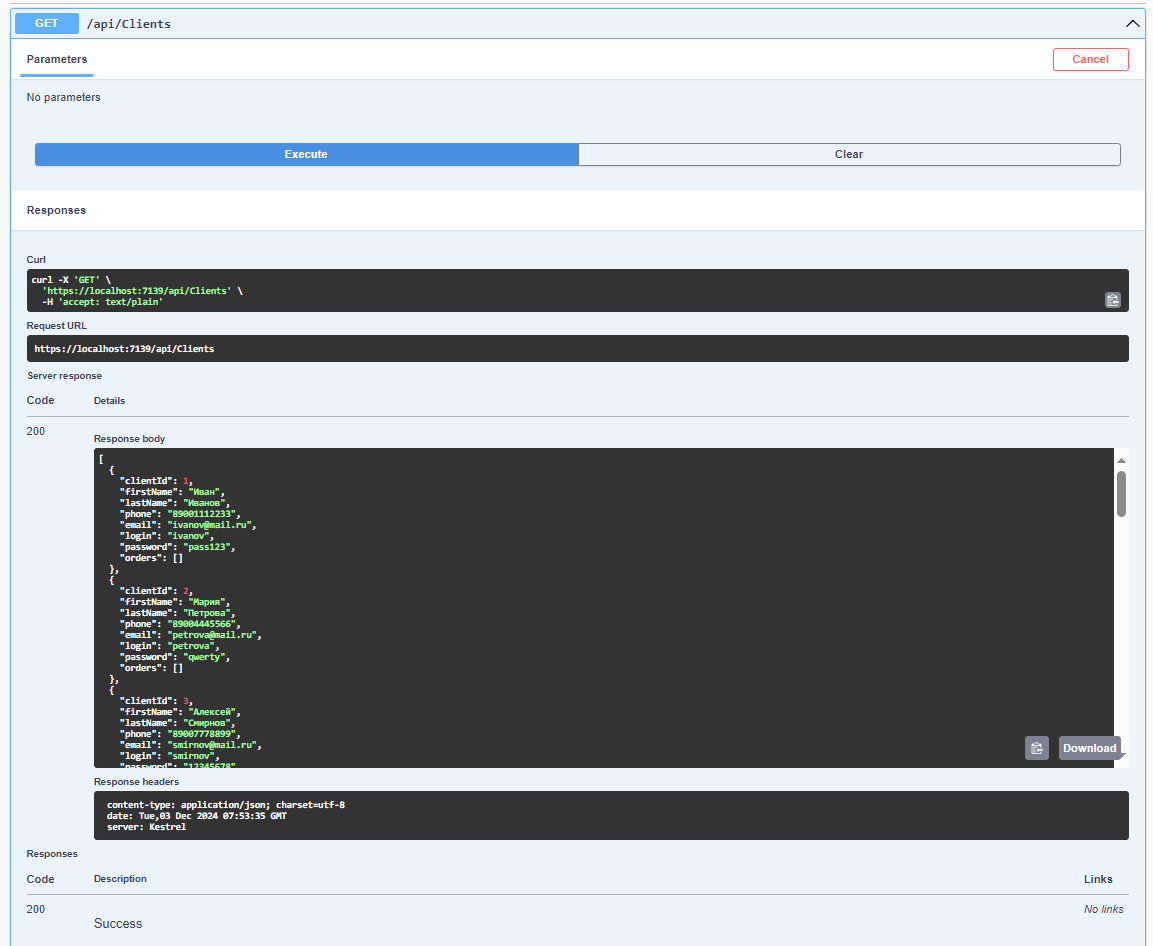


рис.20 «Get all»

2. POST

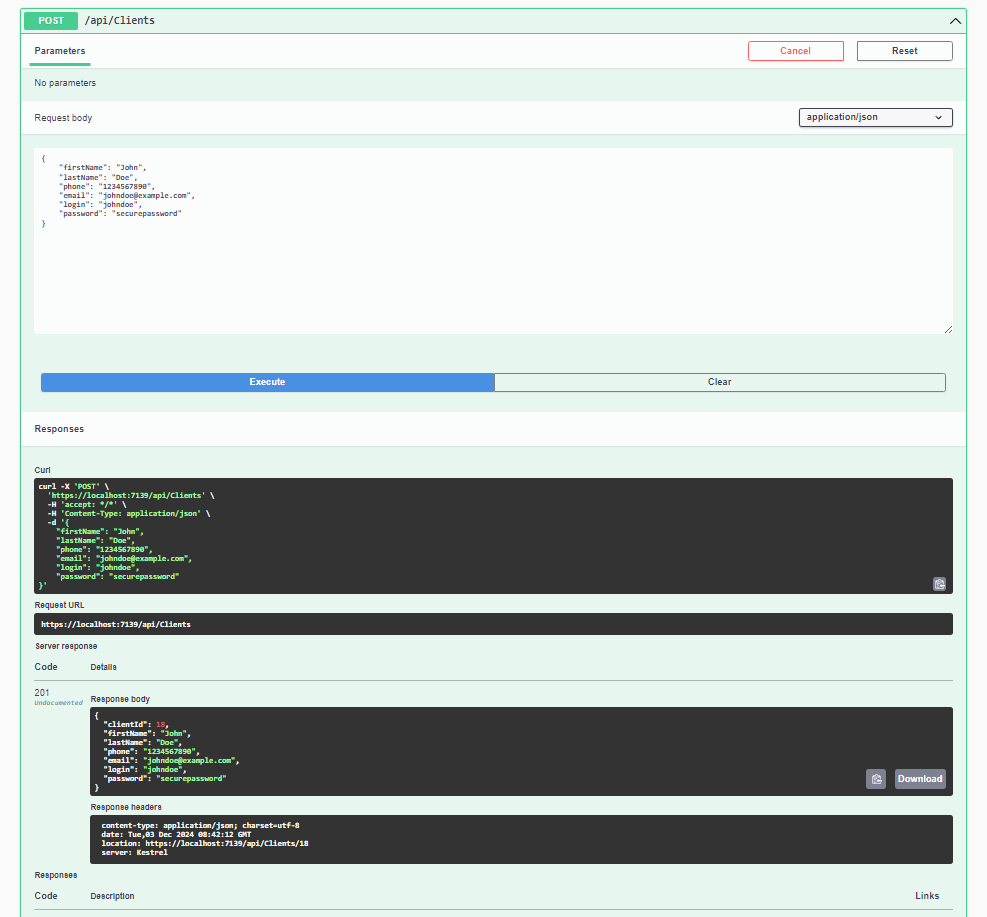


рис.21 «Post»

3.GET id

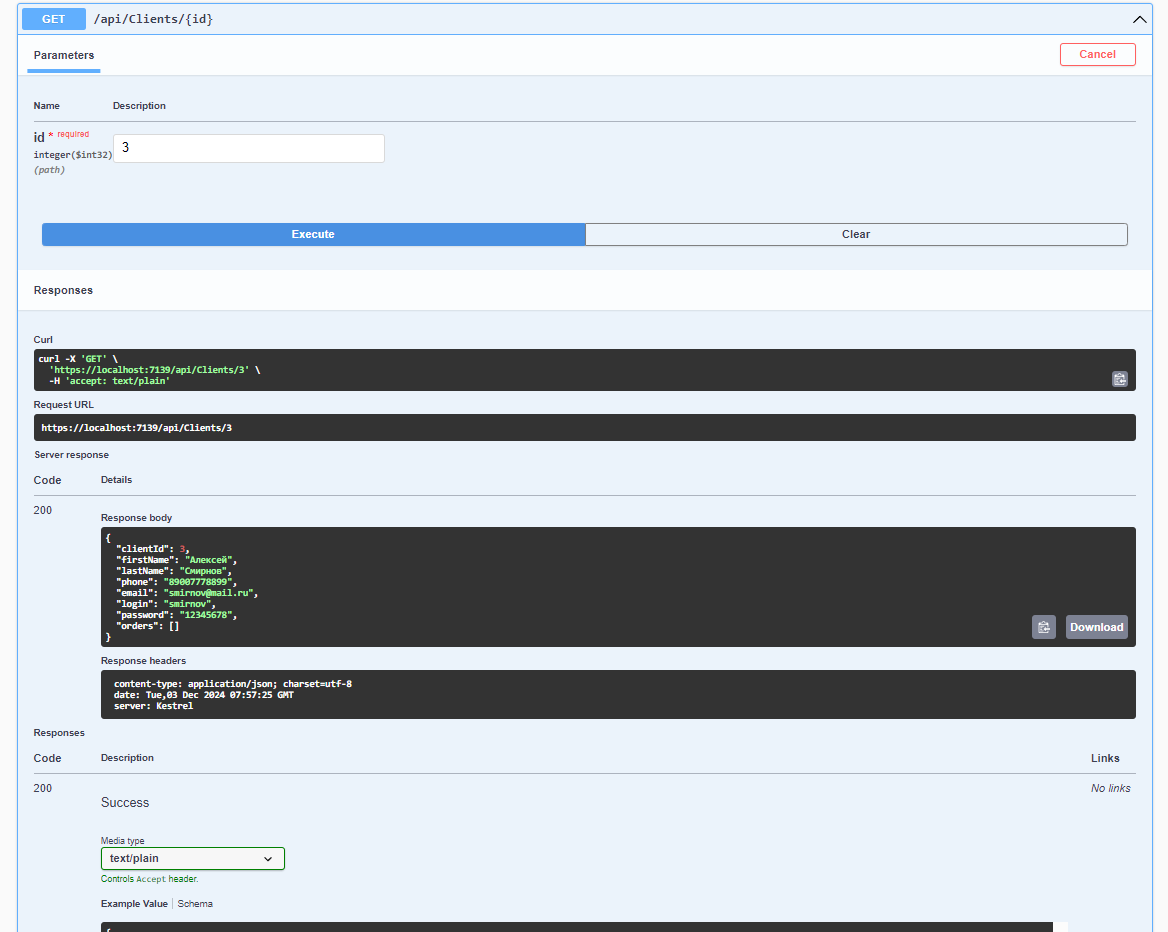


рис.22 «Get id»

4. PUT

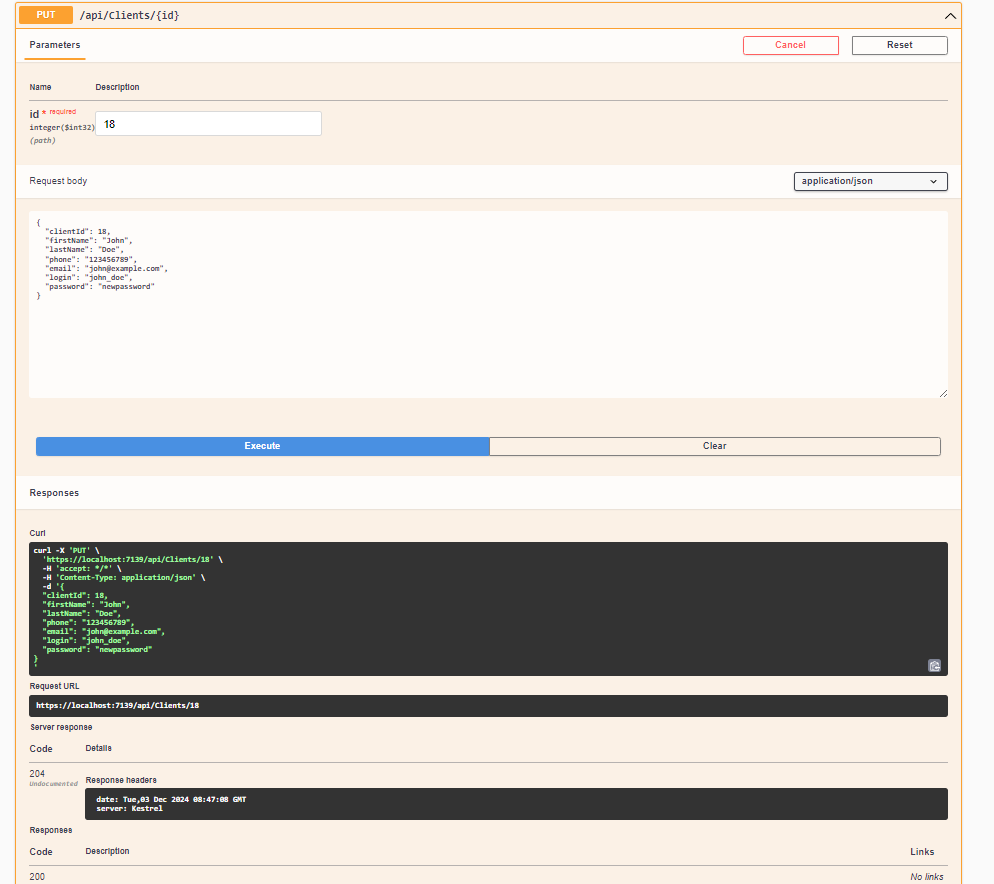


рис.23 «Put»

5.DELETE

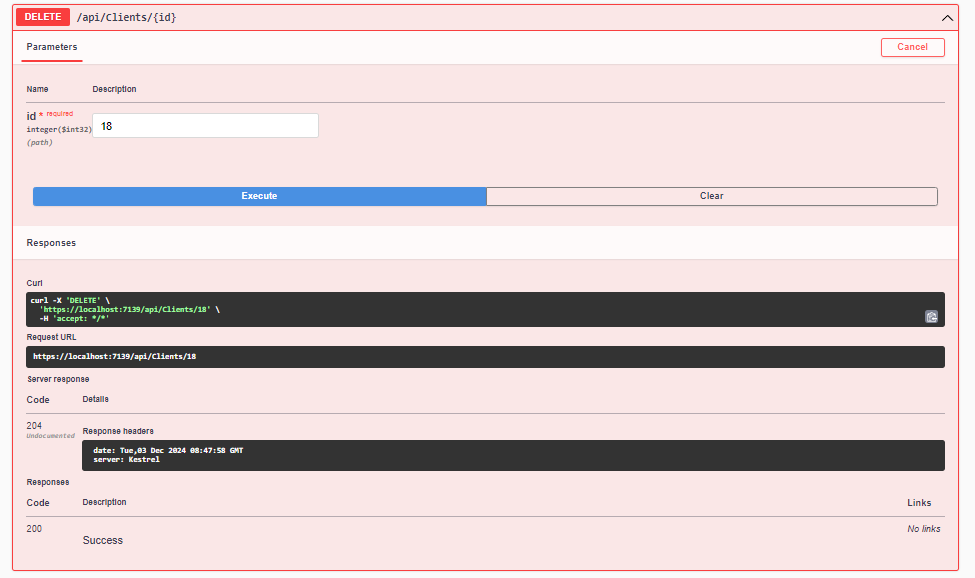


рис.24 «Delete»

# 6. Библиотека классов

Для разработки библиотеки классов использовался шаблон Библиотека классов (Майкрософт)

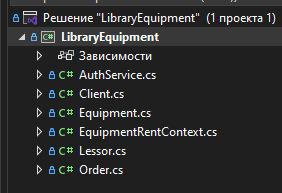


рис.25 «Структура библиотеки классов»

Класс в библиотеки используется в качестве логики и использован для тестов работы авторизации.

AuthService  
using System;

using System.Linq;

using EquipmentForRent.Models;

using Microsoft.EntityFrameworkCore;

namespace LibraryEquipment

{

public class AuthService

{

private readonly EquipmentRentContext \_context;

public AuthService(EquipmentRentContext context)

{

\_context = context;

}

public string Login(string login, string password)

{

if (string.IsNullOrEmpty(login) || string.IsNullOrEmpty(password))

{

return "Логин и пароль обязательны!";

}

var client = \_context.Clients.FirstOrDefault(c => c.Login == login && c.Password == password);

var lessor = \_context.Lessors.FirstOrDefault(l => l.Login == login && l.Password == password);

if (client != null)

{

return "ClientHomePage"; // Возвращаем строку для примера

}

else if (lessor != null)

{

return "LessorHomePage";

}

else

{

return "Неверный логин или пароль!";

}

}

}

}

Для тестов использовался шаблон проекта MSTest

Структура приложения MSTest

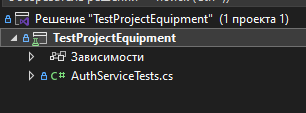


рис.26 «Структура MSTest»

Проект модульного тестирования

using EquipmentForRent.Models;

using LibraryEquipment;

using Microsoft.EntityFrameworkCore;

using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace TestProjectEquipment

{

[TestClass]

public class AuthServiceTests

{

private EquipmentRentContext \_context;

private AuthService \_authService;

[TestInitialize]

public void Initialize()

{

// Настроим SQL Server базу данных для тестов

var options = new DbContextOptionsBuilder<EquipmentRentContext>()

.UseSqlServer("Server=PC;Database=EquipmentRent;TrustServerCertificate=True;Trusted\_Connection=True;")

.Options;

\_context = new EquipmentRentContext(options);

// Убедитесь, что база данных создана

\_context.Database.EnsureCreated();

// Создаем экземпляр AuthService

\_authService = new AuthService(\_context);

}

[TestMethod]

public void Login\_ShouldReturnClientHomePage\_ForValidClientCredentials()

{

// Arrange

string login = "ivanov"; // Логин существующего клиента

string password = "pass123"; // Пароль существующего клиента

// Act

string result = \_authService.Login(login, password);

// Assert

Assert.AreEqual("ClientHomePage", result);

}

[TestMethod]

public void Login\_ShouldReturnLessorHomePage\_ForValidClientCredentials()

{

// Arrange

string login = "morozov"; // Логин существующего арендодателя

string password = "lessor123"; // Пароль существующего арендодателя

// Act

string result = \_authService.Login(login, password);

// Assert

Assert.AreEqual("LessorHomePage", result);

}

}

}

Тест проверяет логин и пароль существующего клиента или арендодателя сравнивая их с данными в базе данных

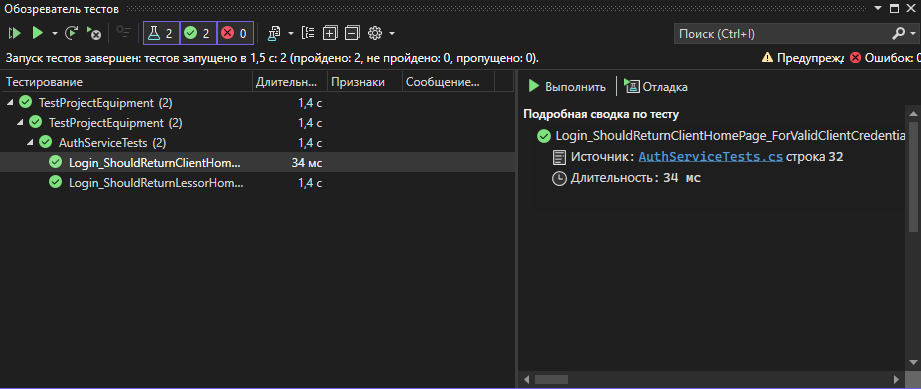


рис.27 «Отчет об удачном завершении теста»

Если указать неправильные данные то тест сразу дает ошибку с результатом – “Неверный логин или пароль”

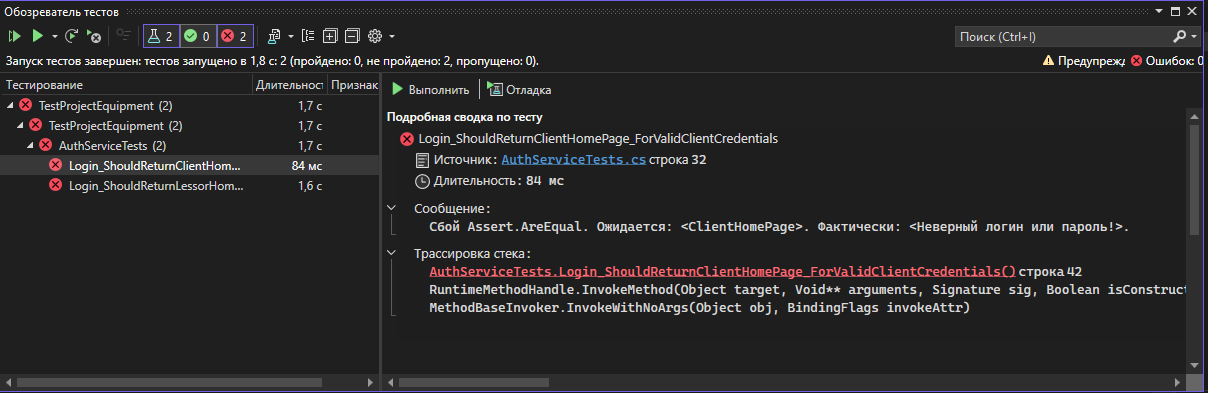


рис.28 «Отчет об неудачном завершении теста»

# Заключение

Приложение выполнено в соответствии с техническим заданием, работает корректно и имеет дополнительное приложение API и библиотеку классов.

В дальнейшем я бы переработал поиск оборудования клиентом, добавил бы больший функционал, например фильтрацию оборудования по цене. Улучшил бы интерфейс убрав баги стилей и сделал бы его более понятным.