



ДЕПАРТАМЕНТ ОБРАЗОВАНИЯ И НАУКИ ГОРОДА МОСКВЫ
ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ПРОФЕССИОНАЛЬНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ГОРОДА МОСКВЫ
«КОЛЛЕДЖ СВЯЗИ №54»
ИМЕНИ П.М. ВОСТРУХИНА

ДОПУСКАЮ К ЗАЩИТЕ

Заместитель директора по УПР О.В. Корешков

(дата)

ДИПЛОМНАЯ РАБОТА

Создание автоматизированной информационной системы для управления
процессом городского планирования и развития

(тема)

Выпускная квалификационная работа должна быть выполнена в виде:
дипломной работы и демонстрационного экзамена

студентом группы 4ИСП9-14

(номер группы)

Максимом Павловичем Кириллиным

(И. О. Фамилия)

(подпись, дата)

Основная профессиональная образовательная программа по специальности
09.02.07 Информационные системы и программирование

(шифр и наименование специальности)

Форма обучения очная

Руководитель

преподаватель

Олеся Павловна Куропаткина

(ученая степень, должность, И. О.

Фамилия)

(подпись, дата)

Председатель предметной (междисциплинарной, модульной) комиссии

Кирилл Михайлович Бастрыкин

(И. О. Фамилия)

(подпись, дата)

Москва
2024



ДЕПАРТАМЕНТ ОБРАЗОВАНИЯ И НАУКИ ГОРОДА МОСКВЫ
ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ПРОФЕССИОНАЛЬНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ГОРОДА МОСКВЫ
«КОЛЛЕДЖ СВЯЗИ №54»
ИМЕНИ П.М. ВОСТРУХИНА

УТВЕРЖДАЮ

Заместитель директора по УПР О.В. Корешков
02 апреля 2024 года
(дата)

ЗАДАНИЕ НА ДИПЛОМНУЮ РАБОТУ

Студенту Кириллину Максиму Павловичу
(фамилия, имя, отчество полностью)

I. Тема выпускной квалификационной работы Создание автоматизированной информационной системы для управления процессом городского планирования и развития.

II. Срок сдачи студентом законченной работы 06 июня 2024 г.

III. Исходные данные Программные средства: Microsoft Office Visio 2019, СУБД SQL Server Management Studio 2022, среда разработки Visual Studio 2022, среда разработки Android Studio

IV. Перечень подлежащих разработке вопросов

- Анализ особенностей предметной области
- Сравнительный анализ программных средств
- Проектирование и разработка автоматизированной информационной системы (создание Use Case диаграммы, ER диаграммы, прототипа настольного приложения)
- Разработка автоматизированной информационной (создание базы данных в СУБД MS SQL Server, настольного приложения на языке программирования C# в Visual Studio), (мобильного приложения на языке программирования Java в Android studio), (создание API на языке программирования C# в Visual Studio).

V. Перечень графического/иллюстрационного материала

- Use case диаграмма
- ER диаграмма базы данных, разработанная в Microsoft Office Visio
- Диаграмма базы данных, разработанная в СУБД MS SQL Server
- Скриншоты экрана разработанного настольного приложения
- Скриншоты экрана разработанного мобильного приложения
- Презентация

VI. Дата выдачи задания «02» апреля 2024 г.

Руководитель

Задание принял к
исполнению

(подпись)

(подпись)

Куропаткина О.П.

Кириллин М.П.

«02» апреля 2024 г.



ДЕПАРТАМЕНТ ОБРАЗОВАНИЯ И НАУКИ ГОРОДА МОСКВЫ
ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ПРОФЕССИОНАЛЬНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ГОРОДА МОСКВЫ
«КОЛЛЕДЖ СВЯЗИ №54»
ИМЕНИ П.М. ВОСТРУХИНА

КАЛЕНДАРНЫЙ ПЛАН ВКР

для специальности 09.02.07 Информационные системы и программирование
(группа 4ИСП9-14)

Утверждение темы и руководителя дипломной работы	28.03.2024
Выдача задания на дипломную работу	29.03.2024 – 08.04.2024
Сроки преддипломной практики	09.04.2024 – 06.05.2024
Выполнение задания по теме дипломной работы	09.04.2024 – 02.05.2024
Предоставление отчета по практике руководителю	03.05.2024
Аттестация по практике	06.05.2024
Подготовка дипломной работы	07.05.2024 – 10.06.2024
Подбор и анализ исходной информации	07.05.2024 – 13.05.2024
Подготовка и утверждение плана (оглавления) дипломной работы	
Работа над разделами (главами) и устранение замечаний руководителя дипломной работы	14.05.2024 – 27.05.2024
Согласование содержания, устранение замечаний	28.05.2024 – 02.06.2024
Оформление и представление руководителю полного текста работы	03.06.2024 – 06.06.2024
Получение отзыва руководителя дипломной работы	
Сдача демонстрационного экзамена	Согласно отдельному графику
Предзащита дипломной работы	07.06.2024 – 10.06.2024

Руководитель _____ Куропаткина О.П.

План принял к исполнению
«02» апреля 2024 г. _____ Кириллин М.П.

ОГЛАВЛЕНИЕ

Введение.....	5
Глава 1. Анализ предметной области и применяемых программных средств	7
1.1 Анализ особенностей предметной области	7
1.2 Анализ применяемых программных средств	8
1.2.1 Анализ программы для разработки диаграмм Microsoft Visio.....	8
1.2.2 Анализ системы управления баз данных Microsoft SQL Server.....	9
1.2.3 Анализ объектно-ориентированного языка программирования C#	11
1.2.4 Анализ интегрированной среды разработки Visual Studio	12
1.2.5 Анализ интегрированной среды разработки Android Studio	13
Глава 2. Разработка и внедрение программного продукта	15
2.1 Создание Use case диаграммы	15
2.2 Проектирование базы данных.....	16
2.3 Разработка базы данных	17
2.4 Разработка функционала API.....	18
2.5 Разработка функционала настольного приложения	21
2.6 Разработка функционала мобильного приложения.....	27
Заключение	31
Список литературы	32
Приложение 1. Use case диаграмма	
Приложение 2. Логическая модель данных	
Приложение 3. Физическая модель данных	

					09.02.07 – 4ИСП9-14			
Изм.	Лист	№ докум.	Подпись	Дата				
Разраб.		Кириллин М.П.			Создание автоматизированной информационной системы для управления процессом городского планирования и развития.	Лит.	Лист	Листов
Провер.		Куропаткина О.П.					4	35
Рецензент						ГБПОУ КС №54 им. П.М. Вострухина		
Н. Контр.								
Утверд.								

ВВЕДЕНИЕ

Городское планирование — это процесс разработки и реализации стратегий и планов, нацеленных на улучшение использования городской территории, обеспечение экономического развития и качества жизни горожан. В современном мире крайне важной является работа автоматизированных систем, ведь благодаря им большое количество процессов можно оптимизировать, повысить эффективность. Многие процессы в целом невозможны без автоматизированных систем, так как человек просто не может обрабатывать такой объем данных без посторонней помощи компьютера.

Таким образом удастся сэкономить время и исключить ошибки, более того можно автоматизировать и сделать удобным весь процесс управления какой-либо системой.

Для информационной системы управления процессом городского планирования и развития, она может быть полезна следующим:

- понятное и доступное описание всех видов планирования;
- учет объектов недвижимости и земельных участков;
- управление информации о переходе права собственности на недвижимость;
- наглядное описание предоставляемых услуг.

Актуальность данной дипломной работы обусловлена необходимостью создания автоматизированной системы для городского планирования и повышения её эффективности.

Целью дипломной работы является систематизация и закрепление полученных теоретических знаний и практических умений, проектирование и разработка базы данных для учёта планируемых и реализованных объектов недвижимости, для организации, осуществляющей услуги по планированию районов и их урбанизации, создание настольного приложения для учёта и продажи недвижимости, создание мобильного приложения для возможности просмотра недвижимости.

Для достижения цели были поставлены следующие задачи:

- изучение особенностей конкретной предметной области, относящихся к теме курсовой работы;
- анализ программных средств с обоснованием выбора;
- проектирование и разработка базы данных в СУБД;
- проектирование и разработка настольного приложения;
- проектирование и разработка мобильного приложения;
- создание клиент-серверной системы через API, для связи приложений с базой данных;
- анализ полученных результатов работы разработанного программного обеспечения.

Объектом исследования является система управления процессом городского планирования и развития.

Предметом исследования является создание автоматизированной информационной системы для управления процессом городского планирования и развития.

Информационной базой выпускной квалификационной работы является научная и техническая литература, электронные источники.

ГЛАВА 1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И ПРИМЕНЯЕМЫХ ПРОГРАММНЫХ СРЕДСТВ

1.1. Анализ особенностей предметной области

В рамках данной дипломной работы рассматривается предметная область организации, осуществляющей управление процессом городского планирования, развития и продажи недвижимости.

Основной деятельностью организации, осуществляющей процесс городского планирования, развития и продажи недвижимости является предоставление информации о недвижимости, земельных участках и возможности её приобретения. К таким услугам относятся:

- возможность просмотра доступных строений и земельных участков со стороны организации;
- возможность планировки строений со стороны организации;
- просмотр информации о недвижимости со стороны организации и клиента;
- покупка недвижимости со стороны клиента.

Данная автоматизированная система предназначена для оптимизации работы сотрудников и для удобства выбора недвижимости организации. Сотрудники будут разделены в соответствии с их должностями.

Роль менеджера – учёт, распределение и продажа недвижимости клиентам (физические и юридические лица).

Кадастровый инженер – это специалист, который производит межевание территории и вносит актуальную информацию в систему.

Архитектор – специалист, который создаёт планировки домов и квартир, внутри системы архитектор сможет добавлять планировки домов и квартир, указывать информацию о них.

Клиент – юридическое или физическое, на которого ориентирован весь проект. Внутри системы он сможет просматривать информацию о районе, домах и земельных участках, в следствии выбрать недвижимость для покупки или аренды.

Планируется реализация клиент-серверной части через разработку API на базе ASP.NET, удобного инструмента внутренней инфраструктуры Visual Studio. API необходима для привязки к серверу как настольного, так и мобильного приложения.

Анализ предметной области для учета планируемых и реализованных объектов недвижимости включает в себя анализ факторов, влияющих на успешность реализации объектов недвижимости. Кроме того, важно иметь систему отчетности, которая позволяет отслеживать выполнение плановых показателей и анализировать результаты проектов недвижимости.

1.2. Анализ применяемых программных средств

Для разработки информационной системы будет использоваться ряд программных средств, а именно:

- сервис проектирования Microsoft Visio;
- СУБД Microsoft SQL Server 2019;
- язык программирования C#;
- среда разработки Visual Studio 2019;
- среда разработки Android Studio.

Необходимо рассмотреть данные продукты более подробно.

1.2.1 Анализ программы для разработки диаграмм Microsoft Visio

Microsoft Visio - векторный графический редактор, редактор диаграмм и блок-схем, который является частью составного пакета Microsoft Office. Программа предназначена для создания различного вида чертежей: от схем до календарей.

В программе представлено множество различных фигур, как простых, так и сложных. Каждый шаблон предназначен для определенной цели - от создания планов водопроводных сетей до компьютерных сетей. Для поиска сведений о предназначении шаблона является специальное окошко "Поиск". Программа

					09.02.07 – 4ИСП9-14	Лист
						8
Изм.	Лист	№ докум.	Подпись	Дата		

выпускается в двух комплектациях Office Visio Professional и Office Visio Standard. Для черчения схем вполне достаточно Office Visio Standard.

Основные функции:

Программа Visio является мощным инструментом для создания и редактирования UML диаграмм, которые используются для визуализации структуры и поведения программных систем. С ее помощью пользователи могут легко создавать диаграммы классов, которые показывают классы, их атрибуты и методы, а также связи между ними.

Диаграммы последовательности позволяют отобразить взаимодействие объектов в определенном порядке, показывая какие операции выполняются в каком порядке. Диаграммы активностей помогают представить серию действий или процессов, включая условия, циклы и разделение потоков выполнения.

Visio также предоставляет возможность создавать диаграммы состояний, которые отображают все возможные состояния объекта и переходы между ними. Программа имеет различные инструменты форматирования, выравнивания и связывания элементов, что делает процесс создания UML диаграмм более удобным и эффективным.

1.2.2 Анализ системы управления баз данных Microsoft SQL Server

SQL Server – это программа, которая предназначена для хранения и обработки данных. При взаимодействии с ней пользователи могут отправлять запросы и получать ответы – причем как локально, так и по сети. Функционирует программа следующим образом: открывает сетевой порт, принимает команды и выдает результат.

Рассмотрим, какие у Microsoft SQL Server преимущества и недостатки.

Основные достоинства:

1. Масштабирование системы. Взаимодействовать с ней можно как на простых ноутбуках, так и на ПК с мощным процессором, который способен обрабатывать большой объем запросов.

2. Размер страниц – до 8 Кб. Данные извлекаются быстро, а сложную информацию удобнее хранить. Система обрабатывает транзакции в интерактивном режиме, есть динамическая блокировка.
3. Автоматизация рутинных административных задач. Например, управление блокировками и памятью, редактора размеров файлов. В программе продуманы настройки, можно создавать профили пользователей.
4. Удобный поиск. Его можно осуществлять по фразам, словам, тексту либо создавать ключевые индексы.
5. Поддержка работы с другими решениями Майкрософт, в том числе с Excel, Access.
6. 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:

1. Лицензирование и стоимость: SQL Server - коммерческое программное обеспечение, и лицензия на его использование может быть довольно дорогой, особенно для крупных предприятий или организаций.
2. Сложность настройки и управления: SQL Server имеет довольно сложный процесс установки, настройки и обслуживания. Для поддержки базы данных требуется специальная квалификация и опыт.
3. Ограничения бесплатной версии: SQL Server имеет бесплатную версию Express Edition, но она имеет ограничения по объему данных, производительности и функциональным возможностям по сравнению с полной версией.

1.2.3 Анализ объектно-ориентированного языка программирования C#

C# — это объектно-ориентированный язык программирования. Его разработка велась в 1998-2001 годах под руководством группы программистов из корпорации Microsoft. Изначально он рассматривался как средство создания утилит для платформ Microsoft .NET Framework и .NET Core.

Некоторые полагают, что C# – это просто версия C или C++. Данное утверждение неверное. Соответствующий язык программирования был создан «с нуля».

C# изначально был придуман компанией Microsoft для собственных целей и служб. Он предусматривает следующие преимущества:

- строгую типизацию;
- сохранение концепций объектно-ориентированного программирования;
- функциональность;
- достаточно мощный инструментарий;
- стабильную работу через Visual Studio;
- компактный и легко читаемый код;
- понятный даже новичкам синтаксис.

При использовании этого языка можно насладиться обработкой исключений, а также наличием сборщика мусора. Здесь все продумано так, чтобы программисту было легко писать и считывать итоговые кодификации.

Синтаксис языка чем-то напоминает не только C и C++, но и Java.

Автоматическая «сборка мусора» Это значит, что нам в большинстве случаев не придётся заботиться об освобождении памяти. Вышеупомянутая общезыконовая среда CLR сама вызовет сборщик мусора и очистит память.

Низкий порог вхождения. Синтаксис C# имеет много схожего с другими языками программирования, благодаря чему облегчается переход для программистов. Язык C# часто признают наиболее понятным и подходящим для новичков.

1.2.4 Анализ интегрированной среды разработки Visual Studio

Microsoft Visual Studio — это программная среда по разработке приложений для ОС Windows, как консольных, так и с графическим интерфейсом.

Интегрированная среда разработки (IntegratedDevelopmentEnvironment - IDE) Visual Studio предлагает ряд высокоуровневых функциональных возможностей, которые выходят за рамки базового управления кодом.

Ниже перечислены основные преимущества IDE-среды Visual Studio.

Встроенный Web-сервер. Для обслуживания Web-приложения ASP.NET необходим Web-сервер, который будет ожидать Web-запросы и обрабатывать соответствующие страницы. Наличие в Visual Studio интегрированного Web-сервера позволяет запускать Web-сайт прямо из среды проектирования, а также повышает безопасность, исключая вероятность получения доступа к тестовому Web-сайту с какого-нибудь внешнего компьютера, поскольку тестовый сервер может принимать соединения только с локального компьютера.

Меньше кода для написания. Для создания большинства приложений требуется приличное количество стандартного стереотипного кода, и Web-страницы ASP. NET тому не исключение. Например, добавление Web-элемента управления, присоединение обработчиков событий и корректировка форматирования требует установки в разметке страницы ряда деталей. В Visual Studio такие детали устанавливаются автоматически.

Интуитивный стиль кодирования. По умолчанию Visual Studio форматирует код по мере его ввода, автоматически вставляя необходимые отступы и применяя цветовое кодирование для выделения элементов типа комментариев. Такие незначительные отличия делают код более удобным для чтения и менее подверженным ошибкам. Применяемые Visual Studio автоматически параметры форматирования можно даже настраивать, что очень удобно в случаях, когда разработчик предпочитает другой стиль размещения скобок (например, стиль K&R, при котором открывающая скобка размещается на той же строке, что и объявление, которому она предшествует).

					09.02.07 – 4ИСП9-14	Лист
Изм.	Лист	№ докум.	Подпись	Дата		12

Более высокая скорость разработки. Многие из функциональных возможностей Visual Studio направлены на то, чтобы помогать разработчику делать свою работу как можно быстрее. Удобные функции, вроде функции IntelliSense (которая умеет перехватывать ошибки и предлагать правильные варианты), функции поиска и замены (которая позволяет отыскивать ключевые слова как в одном файле, так и во всем проекте) и функции автоматического добавления и удаления комментариев (которая может временно скрывать блоки кода), позволяют разработчику работать быстро и эффективно.

1.2.5 Анализ интегрированной среды разработки Android Studio

Android Studio – это интегрированная среда разработки для работы с платформой Android. ПО было анонсировано в 2013 году компанией Google. Софт помогает разрабатывать разнообразные приложения и игры под Android.

Это – официальная среда программирования, находящаяся под поддержкой Google. В основе лежит IntelliJ IDEA от JetBrains.

Выделяется «студия» следующими особенностями:

- наличие встроенного эмулятора;
- мощный функционал и инструментарий для разработчиков;
- встроенный отладчик;
- понятный и хорошо продуманный интерфейс;
- документация на русском языке;
- множество уроков, при помощи которых удастся быстро освоить платформу;
- «горячие клавиши»;
- возможность настроить Android Studio за несколько минут;
- совместимость с большинством популярных ОС.

Среда Android Studio предназначена как для небольших команд разработчиков мобильных приложений (даже в количестве одного человека), или же крупных международных организаций с GIT или другими подобными системами управления версиями. Опытные разработчики смогут выбрать

инструменты, которые больше подходят для масштабных проектов. Решения для Android разрабатываются в Android Studio с использованием Java или C++.

Вывод

Данные программные средства, необходимые для написания дипломной работы, являются хорошими системами, в которых можно быстро и удобно сформировать запланированный проект, как его описание и визуализацию, так и программную часть. Каждая программа отвечает за свой функционал и одна из главных задач при разработке, понять, как всех их сопоставить вместе.

ГЛАВА II. РАЗРАБОТКА И ВНЕДРЕНИЕ ПРОГРАММНОГО ПРОДУКТА

2.1. Создание USE CASE диаграммы

Use case (сценарий использования) представляет собой описание того, как пользователи взаимодействуют с программным продуктом для достижения конкретной цели. В сценарии взаимодействия указывается:

- кто конкретно использует сайт или приложение;
- что именно пользователь планирует сделать;
- какая цель у пользователя, которую он хочет достичь;
- какие шаги пользователь выполняет, чтобы достичь цели и выполнить определенное действие;
- описание того, как само приложение реагирует на действия пользователя.

В моей системе пользователи имеют 4 роли: менеджер; кадастровый инженер; архитектор; клиент.

Менеджер, кадастровый инженер и архитектор отвечают за функциональность настольного приложения, клиент взаимодействует только с мобильным приложением.

Функциональная доля менеджера состоит в следующих вещах:

- авторизация в системе;
- просмотр всей существующей в системе недвижимости;
- добавление и удаление фотографий недвижимости;
- организация перехода права собственности от одного клиента к другому клиенту;
- просмотр всех существующих клиентов.

Кадастровый инженер может взаимодействовать с системой следующим образом:

- просмотр всей существующей в системе недвижимости;
- добавление и удаление фотографий недвижимости;

- после произведённого инженером межевания, он может добавить новый объект недвижимости в систему с форматом - участок;

Архитектор в системе имеет схожие возможности:

- просмотр всей существующей в системе недвижимости;
- добавление и удаление фотографий недвижимости;
- после произведённого планирования архитектуры объекта недвижимости архитектор может добавить новый объект с форматом – дом или участок;
- выбор к какому объекту недвижимости принадлежит создаваемый объект, например – квартира к дому, а дом к участку.

Клиент имеет иную позицию входа в систему через мобильное приложение, имея в нем следующие возможности:

- регистрация как физическое лицо;
- регистрация как юридическое лицо;
- авторизация в системе;
- просмотр всей недвижимости в системе;
- просмотр более подробной информации об объекте недвижимости.

Диаграмма прецедентов, разработанная в Microsoft Visio, представлена в Приложении №1.

2.2. Проектирование базы данных

Разработка проекта базы данных предполагает несколько этапов. Первый – концептуальный, в рамках которого осуществляется анализ предметной области, выделяются основные процессы и объекты, подлежащие учету. Далее следует этапы логического и физического проектирования базы данных, рассмотрим их подробнее.

Моя логическая модель базы данных была разработана в приложении Visio. Организации нужно хранить информацию о всех сделках, проведённых между клиентами, а также информацию о учёте объектов недвижимости. Клиент может быть как физическим, так и юридическим лицом, поэтому у клиента есть данные, принадлежащие как физическим, так и юридическим лицам. Объекты

					09.02.07 – 4ИСП9-14	Лист
Изм.	Лист	№ докум.	Подпись	Дата		16

недвижимости представляют собой участки, строения и квартиры, при этом необходимо учитывать, какая конкретная квартира находится в конкретном доме, который находится на конкретном участке, при этом дома могут не иметь квартир, а участки могут не иметь домов. Предусмотрена возможность добавления новых объектов недвижимости архитекторами, которые могут добавлять фотографии объектам. На основе этой задачи были выделены ключевые сущности: объект недвижимости, клиент, работник, сделка. И дополнительные: пол, должность работника, почтовый индекс, форма собственности, тип объекта, фотографии

Логическая модель базы данных, разработанная в Microsoft Visio, представлена в Приложении №2.

2.3. Разработка базы данных

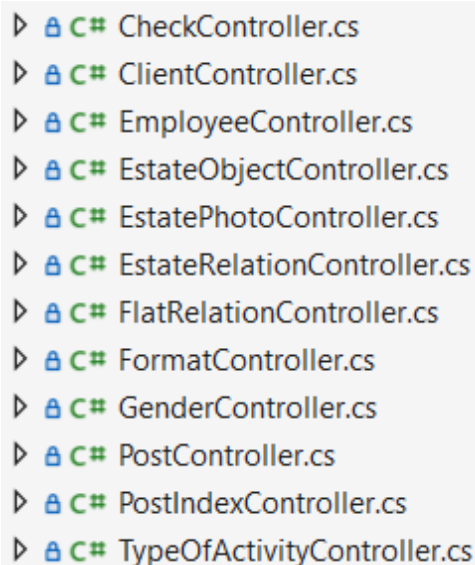
Физическая модель базы данных в СУБД Microsoft SQL Server.

Сущности создавались посредством визуальных инструментов - через создание диаграммы базы данных в приложении Microsoft SQL Server Management studio. Для текстовых полей использовались типы данных nvarchar (информация нефиксированной длины с использованием кодировки UNICODE) и char (информация фиксированной длины). Для числовых полей - integer (ID сущностей) и decimal (цена, площадь). К таким датам как день рождения и дата применялись типы данных date и datetime. К полям, предполагающим значения логического типа, применялся тип bit (1 - активен, 0 - неактивен). На диаграмме можно увидеть ограничения первичного ключа, внешнего и запрет на ввод пустых значений. Ограничение первичного ключа необходимо для идентификации записей, а ограничение внешнего для обеспечения целостности данных.

Таким образом, разработана база данных, которая содержит 12 сущностей, полноразмерная диаграмма базы данных, разработанная в Microsoft SQL Server Management Studio, представлена в Приложении №3.

2.4. Разработка функционала API

В ходе разработки API были разработаны 12 контроллеров (Рисунок 2.1):



```
▸ C# CheckController.cs
▸ C# ClientController.cs
▸ C# EmployeeController.cs
▸ C# EstateObjectController.cs
▸ C# EstatePhotoController.cs
▸ C# EstateRelationController.cs
▸ C# FlatRelationController.cs
▸ C# FormatController.cs
▸ C# GenderController.cs
▸ C# PostController.cs
▸ C# PostIndexController.cs
▸ C# TypeOfActivityController.cs
```

Рисунок 2.1 Контроллеры Api

Каждый контроллер отвечает за свой объект и его обращения к базе данных. Таким образом все контроллеры имеют внутри себя «GET» запрос обращения к базе данных для получения информации. Некоторые из контроллеров имеют внутри себя «POST» запросы, для заполнения или удаления информации внутри базы данных.

Рассмотрим более подробно структуру данных контроллеров (рисунок 2.2):

Рассмотрим «GET» запросы на примере PostController:

Данный контроллер работает следующим образом:

- установка соединения с SQL сервером через специальную учётную запись подключения, имеющую пароль;
- обращение к таблице внутри базы данных инструкцией «Select»;
- преобразование полученных данных в соответствующий класс «Post»;
- проверка на корректное получение данных, в случае успеха контроллер возвращает json файл с телом, внутри которого все объекты класса Post. В противном случае возвращается экземпляр класса «Response» для отображения ошибки.

Остальные «GET» запросы работают по такой же технологии.

```
[Route("api/[controller]")]
[ApiController]
Ссылка: 0
public class PostController : Controller
{
    [HttpGet]
    [Route("GetAllPosts")]
    Ссылка: 0
    public string GetPosts()
    {
        SqlConnection sqlConnection = new SqlConnection(gConnectionString);
        SqlDataAdapter da = new SqlDataAdapter("SELECT * FROM Post", sqlConnection);
        DataTable dt = new DataTable();
        da.Fill(dt);
        List<Post> posts = new List<Post>();
        if (dt.Rows.Count > 0)
        {
            for (int i = 0; i < dt.Rows.Count; i++)
            {
                Post post = new Post();
                post.IDPost = Convert.ToInt32(dt.Rows[i]["IDPost"]);
                post.PostTitle = Convert.ToString(dt.Rows[i]["PostTitle"]);
                posts.Add(post);
            }
        }
        if (posts.Count > 0)
        {
            return JsonConvert.SerializeObject(posts);
        }
        else
        {
            return JsonConvert.SerializeObject(new Response() { Id = "100", Title = "Data not found" });
        }
    }
}
```

Рисунок 2.2 Контроллер «PostController»

Рассмотрим «POST» запросы на примере EstatePhotoController. Данный вид запросов нужен для заполнения и удаления информации внутри базы данных.

Рассмотрим пример заполнения информации (рисунок 2.3).

Данный запрос работает следующим образом:

- установка соединения с SQL сервером через специальную учётную запись подключения, имеющую пароль;
- обращение к базе данных инструкцией «Insert into»;
- динамическое подставление данных из принимаемого объекта «EstatePhoto» внутрь запроса;
- проверка на корректный ответ от базы данных, в случае успеха контроллер возвращает строку «Data is added». В противном случае возвращается строка «Data isn't added».

```
[HttpPost, ActionName("AddNewEstatePhoto")]
[Route("AddNewEstatePhoto")]
```

Ссылка: 0

```
public string AddNewEstatePhoto(EstatePhoto estatePhoto)
{
    SqlConnection sqlConnection = new SqlConnection(gConnectionString);
    SqlCommand sc = new SqlCommand($"Insert into EstatePhoto (PhotoPath, IDEstateObject, IDEmployee) " +
        $"Values('{estatePhoto.PhotoPath}', {estatePhoto.IDEstateObject}, {estatePhoto.IDEmployee})", sqlConnection);
    sqlConnection.Open();
    int i = sc.ExecuteNonQuery();
    sqlConnection.Close();
    if (i > 0)
    {
        return "Data is added";
    }
    else
    {
        return "Data isn't added";
    }
}
```

Рисунок 2.3 Добавление данных с помощью контроллера «EstatePhotoController»

Рассмотрим пример удаления информации (рисунок 2.4).

```
[HttpPost, ActionName("DeleteEstatePhoto")]
[Route("DeleteEstatePhoto")]
```

Ссылка: 0

```
public string DeleteEstatePhoto(EstatePhoto estatePhoto)
{
    SqlConnection sqlConnection = new SqlConnection(gConnectionString);
    SqlDataAdapter da = new SqlDataAdapter("SELECT * FROM EstatePhoto", sqlConnection);
    DataTable dt = new DataTable();
    da.Fill(dt);
    List<EstatePhoto> estatePhotos = new List<EstatePhoto>();
    if (dt.Rows.Count > 0)
    {
        for (int j = 0; j < dt.Rows.Count; j++)
        {
            EstatePhoto estatePhoto2 = new EstatePhoto();
            estatePhoto2.IDEstatePhoto = Convert.ToInt32(dt.Rows[j]["IDEstatePhoto"]);
            estatePhoto2.IDEmployee = Convert.ToInt32(dt.Rows[j]["IDEmployee"]);
            estatePhoto2.IDEstateObject = Convert.ToInt32(dt.Rows[j]["IDEstateObject"]);
            estatePhoto2.PhotoPath = Convert.ToString(dt.Rows[j]["PhotoPath"]);
            estatePhotos.Add(estatePhoto2);
        }
    }
    var o = estatePhotos.Where(j => j.IDEstatePhoto == estatePhoto.IDEstatePhoto).FirstOrDefault();
    if (o == null)
    {
        return "EstatePhoto id not found";
    }
    SqlCommand sc = new SqlCommand("Delete from EstatePhoto where IDEstatePhoto=" + estatePhoto.IDEstatePhoto, sqlConnection);
    sqlConnection.Open();
    int i = sc.ExecuteNonQuery();
    sqlConnection.Close();
    if (i > 0)
    {
        return "Data is deleted";
    }
    else
    {
        return "Data isn't deleted";
    }
}
```

Рисунок 2.4 Удаление данных с помощью контроллера «EstatePhotoController»

Данный запрос работает следующим образом:

- установка соединения с SQL сервером через специальную учётную запись подключения, имеющую пароль;
- обращение к таблице внутри базы данных инструкцией «Select»;

Изм.	Лист	№ докум.	Подпись	Дата

09.02.07 – 4ИСП9-14

Лист

20

- преобразование полученных данных в соответствующий класс «EstatePhoto»;
- проверка на существование удаляемых данных;
- обращение к базе данных инструкцией «Delete from»;
- динамическое подставление данных из принимаемого объекта «EstatePhoto» внутрь запроса;
- проверка на корректный ответ от базы данных, в случае успеха контроллер возвращает строку «Data is deleted». В противном случае возвращается строка «Data isn't deleted».

Остальные «POST» запросы работают по такой же технологии.

2.5. Разработка функционала настольного приложения

Для начала стоит описать структуру поведения модели данных внутри настольного приложения.

Модель данных состоит из 12 классов, соответствующих таблицам внутри базы данных. Внутри каждого класса находятся его свойства, соответствующие свойствам сущностей из базы данных. Однако, помимо этого, существуют виртуальные свойства, реализованные через смежный с данной сущностью класс. Такая система позволяет реализовать навигацию внутри модели данных. Как пример такого класса – «Check» (рисунок 2.5).

```
public partial class Check
{
    Ссылка: 0
    public int IDCheck { get; set; }
    Ссылка: 1
    public System.DateTime DateOfTheSale { get; set; }
    Ссылка: 1
    public decimal FullCost { get; set; }
    Ссылка: 2
    public int IDEmployee { get; set; }
    Ссылка: 4
    public int IDClient { get; set; }
    Ссылка: 4
    public int IDEstateObject { get; set; }

    Ссылка: 2
    public virtual Client Client { get; set; }
    Ссылка: 2
    public virtual Employee Employee { get; set; }
    Ссылка: 2
    public virtual EstateObject EstateObject { get; set; }
}
```

Рисунок 2.5 Класс – «Check»

Следом следует описать такой класс как «Context».

Данный класс является ключевым объектом для взаимодействия данных между собой внутри модели данных.

```
public static class Context
{
    public static ObservableCollection<Check> Checks = new ObservableCollection<Check>();
    public static ObservableCollection<Client> Clients = new ObservableCollection<Client>();
    public static ObservableCollection<Employee> Employees = new ObservableCollection<Employee>();
    public static ObservableCollection<EstateObject> EstateObjects = new ObservableCollection<EstateObject>();
    public static ObservableCollection<EstatePhoto> EstatePhotos = new ObservableCollection<EstatePhoto>();
    public static ObservableCollection<EstateRelation> EstateRelations = new ObservableCollection<EstateRelation>();
    public static ObservableCollection<FlatRelation> FlatRelations = new ObservableCollection<FlatRelation>();
    public static ObservableCollection<Format> Formats = new ObservableCollection<Format>();
    public static ObservableCollection<Gender> Genders = new ObservableCollection<Gender>();
    public static ObservableCollection<Post> Posts = new ObservableCollection<Post>();
    public static ObservableCollection<Postindex> Postindices = new ObservableCollection<Postindex>();
    public static ObservableCollection<TypeOfActivity> TypeOfActivities = new ObservableCollection<TypeOfActivity>();
}
```

Рисунок 2.6 Класс – «Context»

Данный класс реализует в своих свойствах 12 коллекций, соответствующих каждому классу. В дальнейшем, чтобы обратиться к модели базы данных, я обращался именно к этим свойствам.

Данный статический класс имеет внутри себя ключевой метод «GetData». Этот метод позволяет заполнить свежими данными все коллекции, описанные выше. Как пример заполнения, возьму заполнение коллекции – «Genders» (рисунок 2.7).

```
HttpRequest httpRequest;
HttpResponse httpResponse;
Stream stream;
StreamReader sr;
string json;

httpRequest = (HttpRequest)WebRequest.Create(APP_PATH + "/api/Gender/GetAllGenders");
httpResponse = (HttpResponse)httpRequest.GetResponse();
stream = httpResponse.GetResponseStream();
sr = new StreamReader(stream);
json = sr.ReadToEnd();
ObservableCollection<Gender> genders = new ObservableCollection<Gender>();
genders = JsonConvert.DeserializeObject<ObservableCollection<Gender>>(json);
```

Рисунок 2.7 Обращение к API

Данный код отправляет запрос к API описанной выше. Получая ответ от неё, преобразует ответ из Json строки в коллекцию «genders». Аналогичным образом заполняются все остальные коллекции, образуя рабочую модель данных.

Данный метод воспроизводится перед самым стартом приложения, обеспечивая доступ к данным в любой позиции приложения. Данный метод воспроизводится вновь при изменении каких-либо данных внутри системы для достижения актуальности информации.

Эксплуатация приложения пользователем начинается с того, что приложение нужно установить, так как предполагается, что приложением смогут пользоваться сразу множество пользователей. Был разработан специальный установщик приложения, с целью более удобного распространения. Он представляет из себя 2 файла, «UrbanPlanning.msi» и «Setup.exe». Первый файл содержит в себе данные приложения в сжатом виде, второй файл – установщик, открыв который начнется установка приложения в удобную для пользователя директорию (рисунок 2.8).

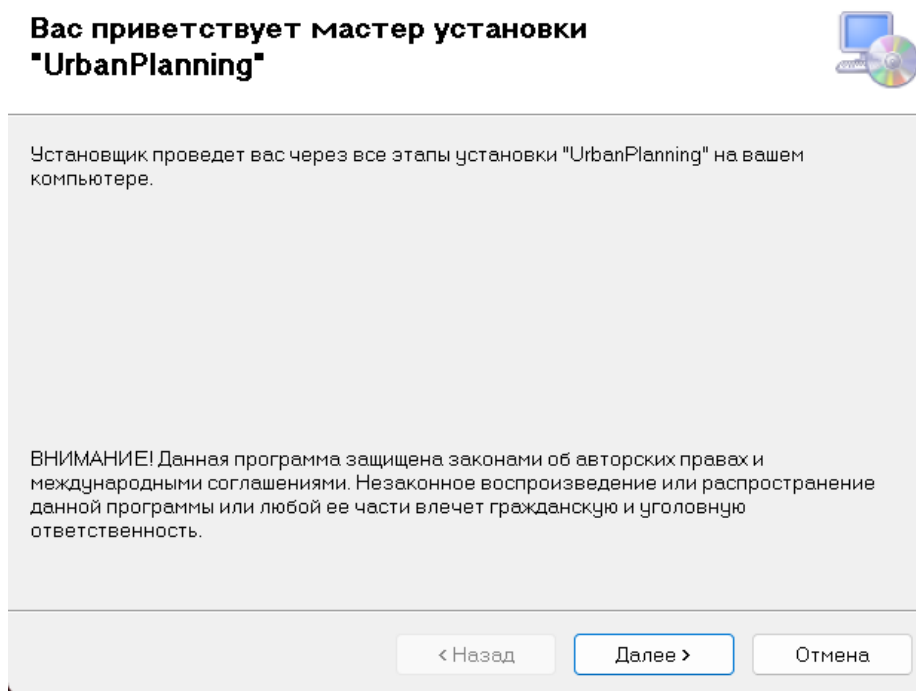


Рисунок 2.8 Мастер установки приложения

После установки приложения и его запуска, пользователю будет предложено авторизоваться в системе (Рисунок 2.9). При успешной авторизации, в зависимости от должности пользователя, он будет перенаправлен на соответствующую его должности страницу.

Информационная система кадастрового учёта объектов недвижимости

Введите логин:

Введите пароль:

Рисунок 2.9 Окно авторизации

Каждый сотрудник имеет доступ к просмотру недвижимости (рисунок 2.10), в окне просмотра имеется интерактивный поиск объекта недвижимости по адресу и кадастровому номеру. Также есть возможность открыть объект в новом окне (рисунок 2.11) для более детального рассмотрения, где можно просмотреть, добавить и удалить фотографии объекта.

Недвижимость

<p>Кадастровый номер: 1 Площадь: 561.0000 Цена: 22440000.00 Дата утверждения проекта: 24.02.1991 Дата ввода в эксплуатацию: 07.01.1993 Номер объекта: 8 Адрес: г. Красноярск, ул. Лермонтова, уч 8 Почтовый индекс: 662599 Тип объекта: Участок Формат объекта: Частная</p>	<p>Кадастровый номер: 2 Площадь: 9166.0000 Цена: 366640000.00 Дата утверждения проекта: 13.04.2010 Дата ввода в эксплуатацию: 16.05.2011 Номер объекта: 198 Адрес: г. Тюльетки, ул. Спердилова, уч 198 Почтовый индекс: 752424 Тип объекта: Участок Формат объекта: Частная</p>	<p>Кадастровый номер: 3 Площадь: 236.0000 Цена: 9440400.00 Дата утверждения проекта: 13.08.1999 Дата ввода в эксплуатацию: 30.03.1996 Номер объекта: 81 Адрес: г. Красноярск, ул. Лермонтова, уч 8, д. 4, кв. 3 Почтовый индекс: 214315 Тип объекта: Квартира Формат объекта: Частная</p>	<p>Кадастровый номер: 4 Площадь: 3124.0000 Цена: 132418600.00 Дата утверждения проекта: 24.08.2013 Дата ввода в эксплуатацию: 18.04.2016 Номер объекта: 63 Адрес: г. Красноярск, ул. Лермонтова, уч 8, д. 4 Почтовый индекс: 674345 Тип объекта: Дом Формат объекта: Государственная</p>
<p>Кадастровый номер: 5 Площадь: 3343.0000 Цена: 133720000.00 Дата утверждения проекта: 25.11.2021 Дата ввода в эксплуатацию: 26.03.2024 Номер объекта: 62 Адрес: г. Екатеринбург, ул. Энергетиков, уч 62 Почтовый индекс: 986345 Тип объекта: Участок Формат объекта: Муниципальная</p>	<p>Кадастровый номер: 6 Площадь: 9637.0000 Цена: 385480000.00 Дата утверждения проекта: 01.07.1998 Дата ввода в эксплуатацию: 04.12.1999 Номер объекта: 14 Адрес: г. Саратов, ул. Фурунов, уч 14 Почтовый индекс: 440394 Тип объекта: Участок Формат объекта: Государственная</p>	<p>Кадастровый номер: 7 Площадь: 9614.0000 Цена: 384560000.00 Дата утверждения проекта: 21.05.2007 Дата ввода в эксплуатацию: 04.02.2009 Номер объекта: 23 Адрес: г. Ростов на Дону, ул. Южная, уч 23 Почтовый индекс: 212105 Тип объекта: Участок Формат объекта: Муниципальная</p>	

Рисунок 2.10 Окно просмотра всей недвижимости

Объект: 53

Текущий владелец:	Баженев Александр Маркович
Кадастровый номер:	53
Площадь:	30.0000
Цена:	3000000.00
Дата утверждения проекта:	01.08.2008
Дата ввода в эксплуатацию:	20.12.2008
Номер объекта:	13
Адрес:	г.Тула 11, 12, 13
Почтовый индекс:	954504
Тип объекта:	Квартира
Формат объекта:	Частная

Рисунок 2.11 Окно просмотра выбранной недвижимости

Кадастровый инженер и архитектор имеют возможность добавлять новые объекты недвижимости в систему (Рисунок 2.12). Инженер только участки, а архитектор – дома и квартиры. При этом архитектор также имеет возможность указать к какому объекту недвижимости принадлежит дом или квартира (Рисунок 2.13).


Добавить объект недвижимости

Введите цену:	<input type="text"/>	Введите номер объекта:	<input type="text"/>
Введите площадь:	<input type="text"/>	Введите адрес:	<input type="text"/>
Введите дату утверждения проекта:	<input type="text" value="Выбор даты"/>	Выберите почтовый индекс:	<input type="text" value="130558"/>
Введите дату ввода в эксплуатацию:	<input type="text" value="Выбор даты"/>	Выберите формат объекта:	<input type="text" value="Частная"/>
<input type="button" value="Зарегистрировать объект"/>		Выберите тип объекта:	<input type="text" value="Дом"/>

Рисунок 2.12 Окно добавления объекта недвижимости

Оформление перехода права собственности

Переход права собственности от "Колосова Василиса Сергеевна" к "Яковлева Альбина Данииловна".



Кадастровый номер: 12
Площадь: 1793.0000
Цена: 71720000.00
Дата утверждения проекта: 13.09.2012
Дата ввода в эксплуатацию: 25.01.2015
Номер объекта: 60
Адрес: г. Красноярск, ул. Молодежная, уч. 60
Почтовый индекс: 931953
Тип объекта: Участок
Формат объекта: Муниципальная

ФИО: Яковлева Альбина Данииловна
Дата рождения: 17.07.2005
Телефон: 85994952037
Сопров. паспорт: 5689
Номер паспорта: 949910
Название компании:
ИНН:
КПП:
ОГРН:
Платежный счет:
Корреспондентский счет:
БИК:
Плат:
Логин: Thetexa

Оформить переход права собственности

Итоговая цена: 71720000,00

Рисунок 2.15 Окно перехода права собственности

Таким образом был продемонстрирован весь функционал и разработка настольного приложения системы.

2.6. Разработка функционала мобильного приложения

Мобильное приложение имеет иной функционал, так как он полностью завязан на клиенте. Однако он имеет ту же структуру обращений к API и формирования модели данных, за исключением того, что модель данных меньше, так как для клиента не имеет смысла получать все данные системы.

Приложение без проблем можно установить на Android устройство через APK файл, что также делает его очень легким в передаче кому-либо. После установки, клиент окажется в окне авторизации (рисунок 2.16). В случае если клиент пользуется системой впервые, он может пройти в окно регистрации (рисунок 2.17), где может зарегистрироваться как физическое, так и юридическое лицо.



Рисунок 2.16 Окно авторизации

На все поля присутствует валидация, поэтому занести неверные данные не получится.

После успешной регистрации пользователь может войти в приложение, а менеджер сможет увидеть нового клиента.

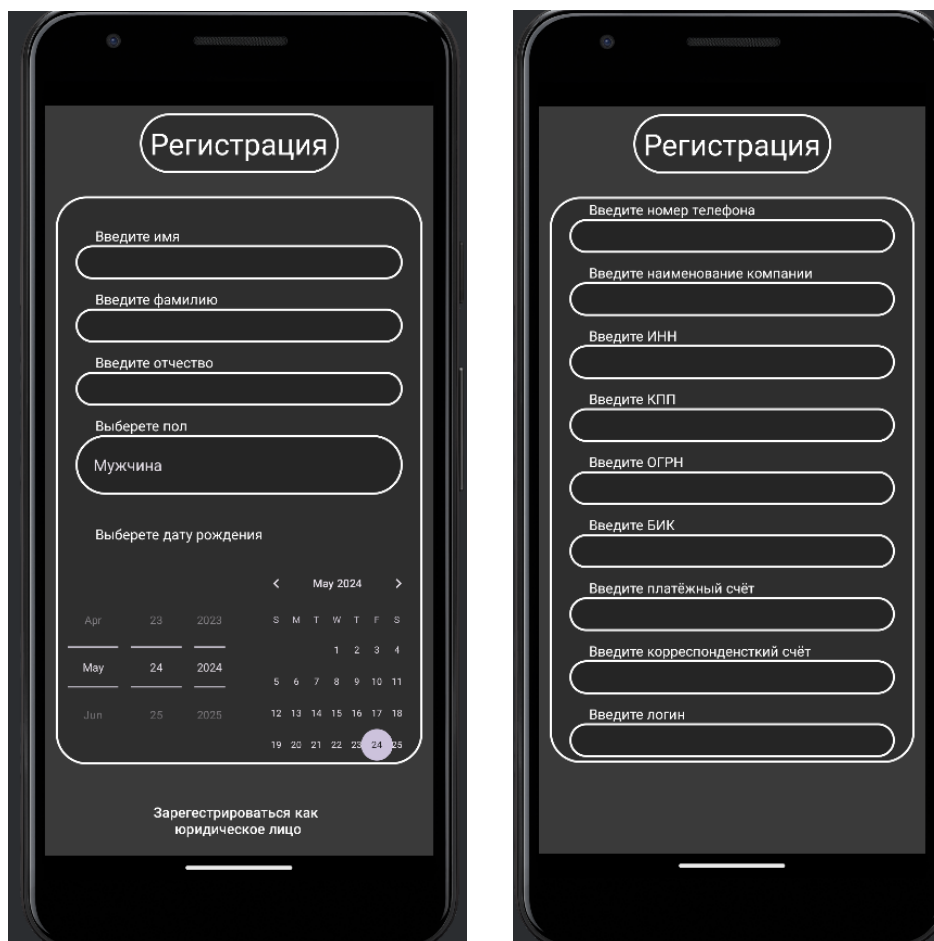


Рисунок 2.17 Окно регистрации, слева физическое лицо, справа – юридическое.

Теперь клиент может просмотреть всю недвижимость (Рисунок 2.18), также он может нажать на любой объект и просмотреть его подробные характеристики и все фотографии.

Также объекты можно искать в поисковике и сортировать.

Изм.	Лист	№ докум.	Подпись	Дата



Рисунок 2.18 Окно просмотра недвижимости, слева все объекты, справа – подробная информация.

Выводы

Таким образом система успешно может функционировать. Клиент, зарегистрировавшийся в приложении, может просматривать все объекты недвижимости. При необходимости он может связаться с компанией и текущим владельцем объекта. Менеджер сможет указать зарегистрированного клиента как покупателя и оформить переход права собственности. Анализ предметной области сформировал понимание как должна работать эта система, в связи с чем был спроектирован функционал системы, состоящий из: Физической базы данных, API для общения базы с приложениями, Мобильное и настольное приложения для работников и клиентов. Применение данной системы позволит автоматизировать учет объектов недвижимости.

ЗАКЛЮЧЕНИЕ

Таким образом, проектирование и разработка системы для управления процессом городского планирования и развития позволяет эффективно управлять данными, сокращать временные затраты на процессы учета и анализа информации.

В проекте использовался нестандартный стек технологий, который был успешно освоен. Организация клиент-серверной структуры позволило получить навыки в сфере архитектуры и администрирования подобных систем управления. Были получены навыки в сфере работы с сетевой структурой и интернет-протоколами, а также закреплены знания в мобильной и настольной разработке систем.

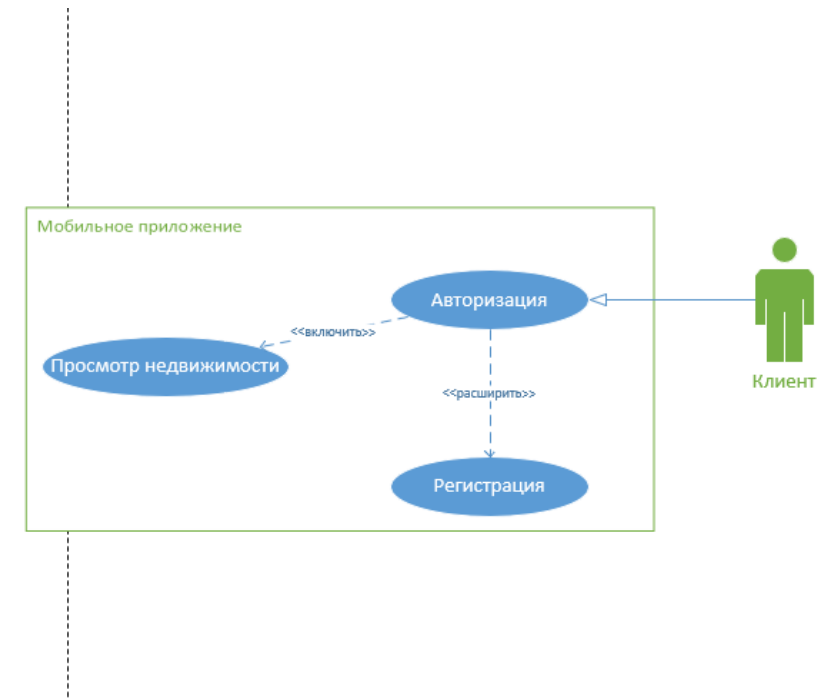
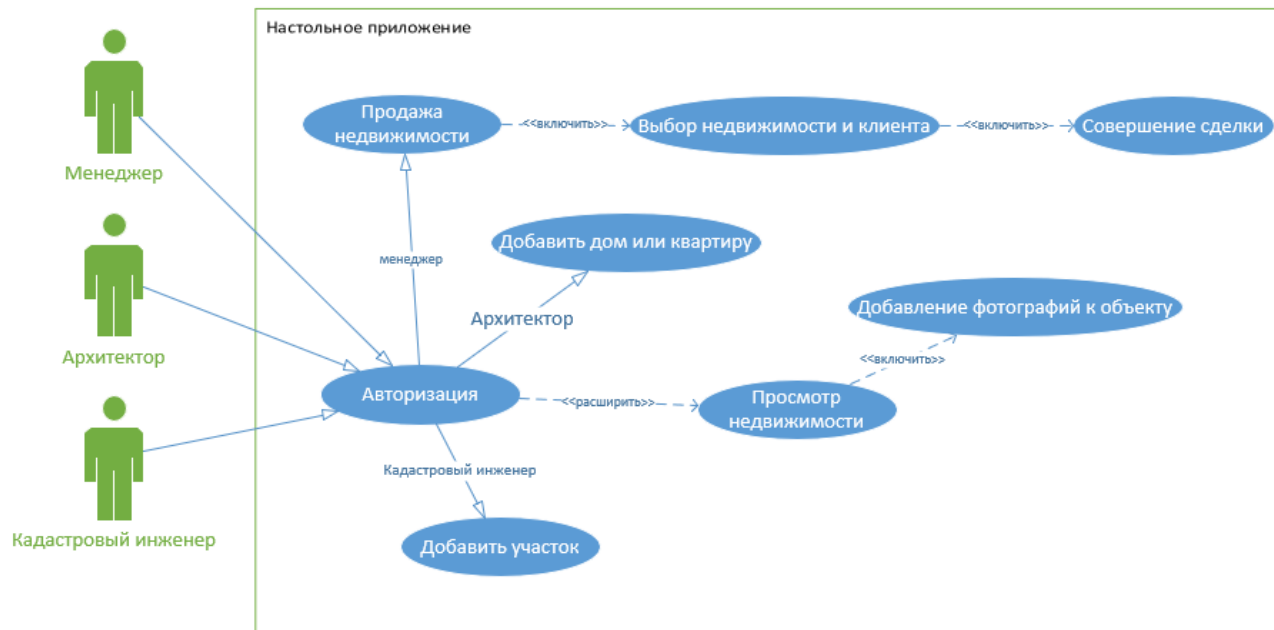
Система удобна и проста в использовании, средства валидации не позволят занести не верную информацию внутрь. Также система легко распространяема за счет её установщиков.

Все поставленные цели были достигнуты, однако система может быть расширена для более точного контроля и учёта объектов недвижимости со стороны настольного приложения. В дальнейшем планируются устранение ошибок и поддержка, с реализацией дополнительных функций по требованию.

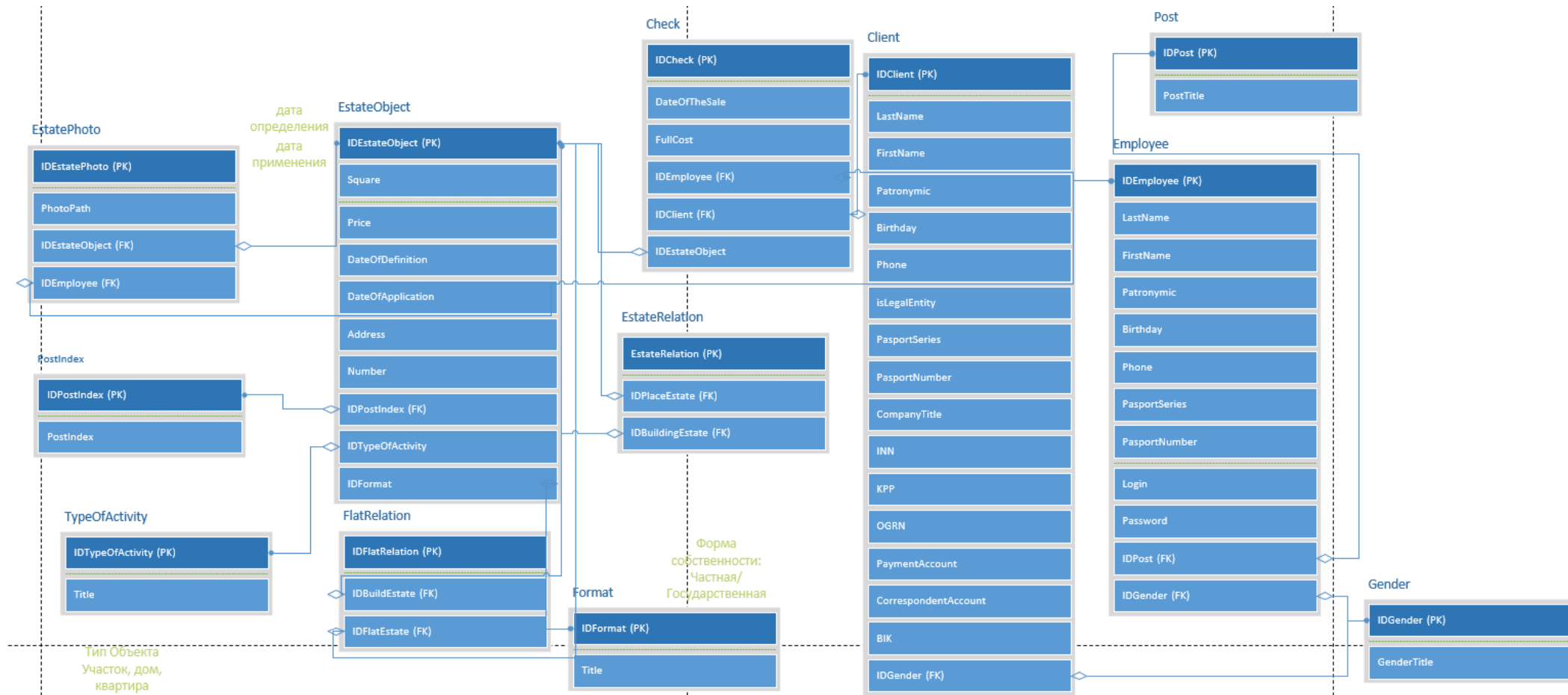
СПИСОК ЛИТЕРАТУРЫ

1. Федоров Г.Н., Разработка, администрирование и защита баз данных: учебник. / Г.Н. Федорова. – Москва: Академия, 2021 г. – 342 с.
 2. Волк В.К., Базы данных. Проектирование, программирование, управление и администрирование. / В.К. Волк. – Москва: Лань, 2021 г. – 423 с.
 3. Карпова Т. С., Базы данных: модели, разработка, реализация: учебное пособие. / Т. С. Карпова. – Москва: ИНТУИТ, 2022 г. – 345 с.
- Электронные ресурсы
4. Юрина Т.А., Программирование и алгоритмизация: учебно-методическое пособие. / Т. А. Юрина. – Сибирский государственный университет, 2021г. – 365 с.
- Электронные ресурсы
5. Росреестр / Публичная кадастровая карта:
<https://pkk.rosreestr.ru/#/search/55.75158999311336,37.61697544168833/19/@5w3tr1rm2> (Дата обращения 7.05.2024).
 6. Росреестр / Кадастровый учёт и регистрация прав:
<https://rosreestr.gov.ru/activity/okazanie-gosudarstvennykh-uslug/kadaastrovyy-uchet-i-ili-registratsiya-prav/> (Дата обращения 7.05.2024).
 7. Иннотер / Статья «Городское планирование, цифровой двойник города»
<https://innoter.com/articles/gorodskoe-planirovanie> (Дата обращения 7.05.2024).
 8. КодНет / Статья «Объекты и концепции базы данных»
<http://www.codenet.ru/db/interbase/ibsql/objs.php> (Дата обращения 7.05.2024).
 9. Молодой учёный / Статья «Городское планирование как система организации устойчивого развития городов»
<https://moluch.ru/archive/115/31116/> (Дата обращения 7.05.2024).

Приложение 1. Use case диаграмма



Приложение 2. Полноразмерная схема базы данных, разработанная в Microsoft Office Visio 2022



Приложение 3. Диаграмма базы данных, разработанная в Microsoft SQL Server Management Studio

