МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет вычислительной техники

Кафедра «Вычислительная техника»

Специальность: 09.05.01 «Применение и эксплуатация автоматизированных систем специального назначения»

Специализация: «Эксплуатация вычислительных машин, комплексов, систем и сетей специального назначения»

ДИПЛОМНЫЙ ПРОЕКТ

на тему

ПРОГРАММА СБОРА ДАННЫХ ОБ ОБЪЕКТАХ НЕДВИЖИМОСТИ

Студент _____ (подпись, дата) 13.06.2023 Водянова Ирина Алексеевна

Руководитель _____ (подпись, дата) 13.06.2023 Дубравин А.В.

Нормоконтролёр _____ (подпись, дата) 13.06.2023 Бычков А.С.

Рецензент нач-к отдела, фил-л ПАО „Россети Волга" - „Пензаэнерго" (должность, место работы) _____ (подпись, дата) 16.06.2023 (фамилия, инициалы) Севастьянов М.

Работа допущена к защите (протокол заседания кафедры от 2.06.2023 г. № 14)

Заведующий кафедрой _____ (подпись) Митрохин М.А.

Работа защищена с отметкой отлично (протокол заседания ГЭК от 19.06.2023 № 1)

Секретарь ГЭК _____ (подпись) Гурин Е.И.

Пенза 2023

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
Вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
_____ Митрохин М.А.
«26» 04 2023г.

ЗАДАНИЕ
НА ДИПЛОМНЫЙ ПРОЕКТ

1. Студент Водянова Ирина Алексеевна гр. 18ВОЭ1 факультета вычислительной техники.

2.Тема работы: «Программа сбора данных об объектах недвижимости».

Тема утверждена приказом ПГУ № 458/0 от 26.04 2023 г.

3. Руководитель дипломного проекта к.т.н., доцент каф. ВТ «ПГУ» Дубравин А.В.

4. Задание на проект (назначение разработки, исходные данные и т.п.).

Назначение разработки – Программа предназначена для сбора, хранения и обработки данных об объектах государственных учреждений. В программе необходимо предусмотреть возможность локального хранения данных полученных от ФГИС ЕГРН. Возможность выполнения операций с записями об объектах недвижимости по схеме CRUD. Программа помимо работы с локальными данными должна обладать интерфейсом API для взаимодействия с ФГИС ЕГРН и интерфейсом создания отчетов по объектам недвижимости.

5. Перечень подлежащих разработке вопросов.

Требуется произвести анализ предметной области, определить функциональный состав программы, выбрать стек технологий для реализации программы, разработать алгоритм работы программы, разработать диаграммы: прецедентов, активностей, последовательностей, с последующим написанием программного кода.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области	10%	26.04.2023 28.04.2023	
2	Выбор стека технологий для реализации программы	10%	29.04.2023 30.04.2023	
3	Проектирование и составление UMLдиаграмм	20%	31.04.2023 03.05.2023	
4	Реализация базы данных	10%	04.05.2023 08.05.2023	
5	Реализация клиентской и серверной части программы	10%	09.05.2023 13.05.2023	
6	Отладка системы	10%	16.05.2023 23.05.2023	
7	Оформление выпускной работы	30%	24.05.2023 02.06.2023	

Дата выдачи задания « 26 » 04 2023 года.

Руководитель дипломного проекта _____ Дубравин А.В.
(подпись, дата)

Задание к исполнению принял студент _____ Водянова И. А.
(подпись, дата)

Работу к защите допустить
Декан факультета _____ 19.06.2023 Фионова Л.Р.
(подпись, дата)

Реферат

Дипломный проект содержит 102 лист., 43 рис., 39 табл., 6 источ., 3 прил.

СПЕЦИАЛЬНЫЙ УЧЕТ ОБЪЕКТОВ НЕДВИЖИМОСТИ, ОБРАБОТКА ИНФОРМАЦИИ, РЕЕСТ, КЛИЕНТСЕРВЕРНАЯ АРХИТЕКТУРА, *WEB* СЕРВИС

Цель дипломного проекта – разработать программное обеспечение сбора данных об объектах недвижимости, позволяющее ответственным сотрудникам юридического лица проводить учет специализированных объектов недвижимости принадлежащих данному юридическому лицу.

В ходе выполнения дипломного проекта были разработаны:

Приложение для формирования информации об объектах недвижимости принадлежащих юридическому лицу и его подконтрольных организаций с возможностью передачи актуальный данных в ФГИС ЕГРН.

В результате дипломного проекта разработано клиент-серверное приложение, полностью соответствующее поставленной задаче, а именно:

- разработана серверная часть приложения, которая обрабатывает и хранит информацию об объектах недвижимости;
- разработана СУБД для хранения требуемой пользователю программного обеспечения информации;
- клиентская часть представлена в виде «тонкого» клиента.

В пояснительной записке приведен анализ предметной области, процесс разработки программного обеспечения, обзор технологии разработки и отладки программы.

Графическая часть содержит диаграммы, полностью описывающие представление СУБД и реализованного программного обеспечения.

Содержание

Перечень сокращений и обозначений 5
Введение 6
1 Программный анализ предметной области 8
1.1 Анализ предметной области 8
1.2 Техническое задание на разработку программного обеспечения 14
1.2.1 Назначение разработки 14
1.3 Требования к программе 14
1.3.1 Требования к функциональным характеристикам ПО 14
1.3.2 Требования к надежности 15
1.3.3 Требования к составу и параметрам технических средств 15
1.3.4 Требования к информационной и программной совместимости 16
1.3.5 Стадии и этапы разработки 16
1.4 Анализ целесообразности использования заданных средств разработки 17
1.5 Проектирование автоматизированной системы 23
1.6 Диаграммы деятельности 24
1.7 Диаграмма последовательности 33
1.8 Диаграмма развертывания 34
1.9 Выводы по предпроектному анализу 34
2 Проектирование и реализация программы сбора данных об объектах недвижимости 35
2.1 Проектирование базы данных приложения 35
2.1.1 Проектирование логической модели базы данных приложения 36
2.1.2 Проектирование физической модели базы данных приложения 39
2.2 Проектирование интерфейса приложения 45
2.3 Проектирование серверной части приложения 62
3 Технология настройки *VPS* сервера для работы *WEB* приложения 69
4 Экономическое обоснование эффективности разработки и внедрения программы сбора данных об объектах недвижимости 74
5 Факторы освещения, воздействующие на оператора персонального компьютера 79

Заключение 81
Список используемых источников 82
Приложение А – *UML*-диаграммы 83
Приложение А.1 – Диаграмма вариантов использования 84
Приложение А.2 – Диаграмма развертывания. 85
Приложение А.3 – Диаграмма последовательности 86
Приложение А.4 – Диаграмма деятельности 87
Приложение А.5 – Диаграмма данных 88
Приложение Б – Листинг программы 89
Приложение В – Презентация 95

Перечень сокращений и обозначений

В данном проекте применяются следующие сокращения и обозначения

ЕГРН – Единый государственный реестр недвижимости

ФГИС – Федеральная государственная информационная система

РФ – Российская Федерация

БД – база данных

СУБД – система управления базами данных

ИС – информационная система

АИС – автоматизированная информационная система

ЭВМ – электронная вычислительная машина

ПО – программное обеспечение

ОС – операционная система

IP (Internet Protokol) – уникальный числовой идентификатор устройства в компьютерной сети

LAMP (Linux Apache MySQl PHP) – набор серверного программного обеспечения

SQL (Structured Query Language) – язык структурированных запросов

PHP (Personal Home Page) – инструменты для создания персональных веб-страниц

UML (Unified Modeling Language) – унифицированный язык моделирования

Введение

Государственные учреждения имеют в пользовании объекты недвижимости специального и общего назначения, которые требуется содержать в надлежащем состоянии.

Для актуализации данных об объектах существует штатная единица, а иногда и целые подразделения, отвечающие за объекты недвижимости. В связи с территориальной удаленностью объектов недвижимости и текучкой сотрудников каждому региональному и федеральному учреждению требуется наличие ведомственного реестра (с интеграцией с БД Росреестра и Росимущества) и программного обеспечения, которое позволит содержать базу данных о недвижимости в актуальном состоянии.

Наличие собственного программного обеспечения позволяет:

- избежать ведомственных запросов для актуализации данных, которые иногда обрабатываются до 30 календарных дней;

- иметь функционал программного обеспечения шире, чем предлагает портал Росреестра и Росимущества;

- избежать использования импортного программного обеспечения;

- иметь полный контроль над передачей информации, которая касается данного юридического лица.

Реализуемое программное обеспечение должно:

- формировать отчет о требуемом объекте недвижимости;

- редактировать информацию о требуемом объекте недвижимости;

- иметь возможность загрузить и отображать в отчетах фотографии об объекте недвижимости;

- иметь возможность формировать *XML* файлы с данными об объектах недвижимости с последующей возможностью передачи информации в единый государственный реестр недвижимости (ЕГРН);

- соответствовать требованиям Федерального закона «О персональных данных» от 27.07.2006 N 152-ФЗ.

Программное обеспечение реализовано с помощью следующего инструментария:

- *Yandex* браузер или иной отечественный аналог с поддержкой протокола *TLS*;

- ведомственный сервер или ЦОД провайдера одобренного для обработки данных нормативными актами РФ;

- бесплатный веб-фреймворк с открытым кодом *Laravel*;

- *JavaScript*-библиотека с открытым исходным кодом для разработки пользовательских интерфейсов *React*;

- операционная система *Linux*.

Программное обеспечение реализуется с использованием практики *Agile*. Для этого в рамках проекта применен итеративный подход к разработке программного обеспечения, который помогает реализовать минимальный работоспособный программный продукт.

Вместо того, чтобы выпускать весь продукт целиком, следуя принципам *Agile*, планируется выполнять работу в рамках небольших, но удобных итераций, по окончании которых конечные пользователи могут воспользоваться реализованным функционалом.

1 Предпроектный анализ предметной области

1.1 Анализ предметной области

В настоящий момент в Российской Федерации существует единое информационное пространство содержащее информацию о земельных участках и объектах недвижимости принадлежащих тому или иному государственному ведовству. Данное информационное пространство представлено Федеральной Государственной Информационной Системой ЕГРН или ЕГРОН (далее - ФГИС ЕГРН).

ФГИС ЕГРН была реализована в три этапа и решила три задачи, которые стоят перед государственными органами:

- создание Единого государственного реестра объектов недвижимости;

- объединение разрозненных баз данных о недвижимости в единое информационное пространство. В результате чего были объединены в единый государственный реестр ИС Госкадастра недвижимости (ГКН) и АИС «Юстиция»;

- реализация проекта только на отечественных и открытых технологиях, в качестве СУБД используется *PostgreSQL*. До этого использовался зарубежный программный продукт СУБД *Oracle*.

Создание ФГИС ЕГРН создало необходимость в создании клиентского программного обеспечения, которое бы позволило пользоваться данными ФГИС ЕГРН и при необходимости и/или по запросу предоставлять в него актуальные данные об объектах недвижимости тем государственным организациям, которым данная недвижимость принадлежит.

Рассмотрим образцы отечественного программного обеспечения типа: программа сбора данных об объектах недвижимости (таблица 1.1), которые соответствуют минимальным функциональным требованиям необходимым для обслуживания потребности государственного учреждения при формировании актуального отчета в Росреестр и Росимущество:

- возможность формирования отчета об объекте недвижимости;

- возможность редактирования информации об объекте недвижимости;

- возможность загрузки фотографий об объекте недвижимости;

- возможность формирования *XML* файлы с данными об объектах недвижимости в соответствии с требованиями ФГИС ЕГРН;

- соответствовать требованиям Федерального закона «О персональных данных» от 27.07.2006 N 152-ФЗ.

Таблица 1.1 - Сравнительный анализ аналогов

Наименование	Функционал	
	плюсы	минусы
1С Предприятие 8 ООО «1С»	- отечественное программное обеспечение; - входит в реестр российского программного обеспечения; - модульная система	- реализовано только в виде «толстого» клиента; - слабая техническая поддержка; - каждый модуль имеет дополнительную стоимость; - реконфигурация возможна силами специалистов; - платное и проприетарное ПО; - отсутствует функция загрузки и просмотра фотографий объекта недвижимости
Платформастроительный сервис ООО «ПСС»	- отечественное программное обеспечение; - входит в реестр российского программного обеспечения; - реконфигурация возможна силами конечного пользователя; - наличие «тонкого» клиента	- реализовано в виде «облачного» сервиса, без доступа у СУБД сторонним ПО; - реализовано в виде платной подписки; - отсутствует функция загрузки и просмотра фотографий объекта недвижимости

Наименование	Функционал	
	плюсы	минусы
Панорама АО КБ «Панорама»	- отечественное программное обеспечение; - входит в реестр российского программного обеспечения; - реконфигурация возможна силами конечного пользователя; - наличие «тонкого» клиента	- реализовано только в виде «толстого» клиента; - сохраняет отчеты в специальном формате; - для перевода специального файла в *XML*требуется покупка дополнительного модуля; - отсутствует функция загрузки и просмотра фотографий объекта недвижимости
Pragmacore. Облачная платформа управления строительными проектами ООО «Прагма»	- отечественное программное обеспечение; - входит в реестр российского программного обеспечения; - наличие мобильного клиентского приложения	- реализовано в виде «облачного» сервиса; - отсутствие *API* для работы со сторонним приложением; - платное программное обеспечение; - отсутствует функция загрузки и просмотра фотографий объекта недвижимости

Все представленное в таблице 1.1 программное обеспечение обладает функционалом, описанным выше, но данное программное обеспечение в основном проприетарное и доступ и контроль исходного кода для технического персонала государственного ведомства отсутствует, а значит своими силами поддержку ПО они осуществить не смогут.

В части позиций отсутствует «тонкий» клиент, а значит, компьютеры учреждения на которых можно инсталлировать только отечественные

браузеры (с поддержкой *TLS*) без возможности установки полноценной программы клиента не могут работать в системе.

Все выше сказанное, показывает необходимость собственной разработки программного обеспечения для сбора данных об объектах недвижимости в рамках свободно распространяемого и отечественного инструментария, чтобы максимально не зависеть от производителей ПО и оперативно реагировать на изменения данных о недвижимости с функционалом загрузки и просмотра фотографий об объекте недвижимости.

С помощью диаграммы вариантов использования (рисунок 1.1) показаны основные роли в реализуемом приложении.

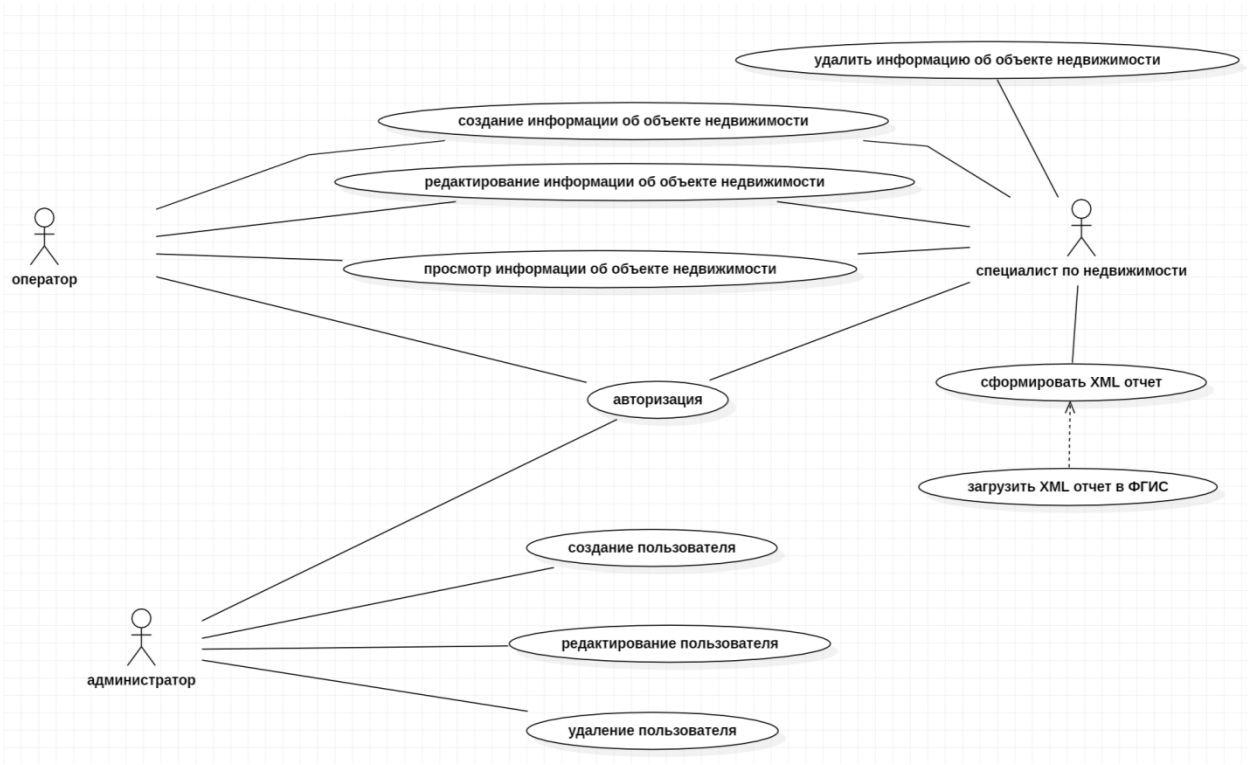


Рисунок 1.1 – Диаграмма вариантов использования

На рисунке 1.1 показана диаграмма вариантов использования, где видно, что в программе могут работать три вида авторизированный пользователей: администратор, оператор и специалист по недвижимости.

Функция самостоятельной регистрации в программе отсутствует, в связи с тем, что в программе могут работать только пользователи сотрудники учреждения. Только администратор имеет право создавать, редактировать и

удалять пользователя. Создание нового пользователя, как и иная операция с учетной записью пользователя, происходит только с письменного разрешения начальника службы безопасности на основании служебной записки от руководителя подразделения в чьи должностные обязанности входит работа с программой сбора данных об объектах недвижимости.

Основную работу в системе по работе с данными об объектах недвижимости выполняют два вида пользователей: оператор и специалист по недвижимости.

Роль оператор выполняет следующие функции:

- авторизация - оператор на странице авторизации вводит логин и пароль, выданный ему администратором системы, после успешной авторизации оператору будут доступны рабочие функции: просмотр, создание, редактирование информации об объекте недвижимости;

- просмотр информации об объекте недвижимости - предоставляет оператору следующую информацию в режиме чтения (запрет на редактирование): регион, район, город, внутригородская территория, улица, дом, кадастровый номер объекта недвижимости, учреждение которому данный объект принадлежит;

- редактирование информации об объекте недвижимости - предоставляет оператору возможность редактировать информацию об объекте недвижимости: регион, район, город, внутригородская территория, улица, дом, кадастровый номер объекта недвижимости, учреждение которому данный объект принадлежит;

- создание информации об объекте недвижимости предоставляет оператору возможность создать информационное поле об объекте недвижимости: регион, район, город, внутригородская территория, улица, дом, кадастровый номер объекта недвижимости, учреждение которому данный объект принадлежит с возможностью заполнения данных информационных полей.

Роль специалиста имеет доступ к функциям роли оператор, плюс имеет следующий дополнительный функционал:

- удалить информацию об объекте недвижимости;

- сформировать отчет в формате *XML*, информационные поля которого соответствуют требованиям к отчетам Федерального закона «О государственной регистрации недвижимости» от 13.07.2015 N 218-ФЗ (пример полей приведен в таблице 1.2).

Таблица 1.2 – Пример полей БД и отчета *XML*

Название	Тип	Формат	Обязательность	Описание
region	*string*	кириллица и цифры арабского алфавита	Да	Код региона
district	*string*	кириллица и цифры арабского алфавита	Да	Район
city	*string*	кириллица и цифры арабского алфавита	Да	Город
urbanDistrict	*string*	кириллица и цифры арабского алфавита	Нет	Внутригородской район
locality	*string*	кириллица и цифры арабского алфавита	Нет	Населенный пункт
street	*string*	кириллица и цифры арабского алфавита	Нет	Улица
house	*string*	кириллица и цифры арабского алфавита	Нет	Дом
building	*string*	кириллица и цифры арабского алфавита	Да	Здание
structure	*string*	кириллица и цифры арабского алфавита	Нет	Сооружение
apartment	*string*	кириллица и цифры арабского алфавита	Нет	Квартира
okato	*string*	кириллица и цифры арабского алфавита	Нет	ОКАТО
oktmo	*string*	кириллица и цифры арабского алфавита	Нет	ОКТМО
postalCode	*string*	кириллица и цифры арабского алфавита	Нет	Почтовый индекс
Kladr	*string*	кириллица и цифры арабского алфавита	Нет	Код КЛАДР

На основании вышеописанного, возможно сформировать требования к функциональным характеристикам программы сбора данных об объектах недвижимости.

1.2 Техническое задание на разработку программного обеспечения

1.2.1 Назначение разработки

Программное обеспечение сбора данных об объектах недвижимости предназначено для взаимодействия государственных учреждений с ФГИС ЕГРН, для поддержания информации о своих объектах недвижимости общего и специального назначения в актуальном виде. Программа предназначена для формирования актуального списка объектов недвижимости принадлежащих юридическому лицу с возможностью формирования отчетной документации для Росреестра и Росимущества.

Программа имеет трехзвенную архитектуру, реализованную на базе паттерна *MVC* в виде *WEB* приложения.

1.3 Требования к программе

1.3.1 Требования к функциональным характеристикам ПО

Программное обеспечение сбора данных об объектах недвижимости должно обеспечивать выполнение следующих функций:

- поиск объектов недвижимого имущества по кадастровому номеру или адресу;

- регистрация прав и обременений на объект недвижимости (взаимодействие в Росриестром);

- получения отчета по объекту недвижимости (взаимодействие в Росриестром);

- создание объекта недвижимости;

- получение данных об объекте недвижимости;

- редактирование данных об объекте недвижимости;

- удаление данных об объекте недвижимости;

- создание нового пользователя;

- редактирование данных о пользователе;

- удаление пользователя.

Входными данными являются:

- информация об объекте недвижимости (идентификатор объекта, вид объекта недвижимости, кадастровый номер, площадь объекта недвижимости, адрес объекта недвижимости, вид разрешенного использования, назначение объекта недвижимости, фото отчеты);

- информация о пользователях программы (роль пользователя, полномочия в рамках программы, логины и пароли);

- информация о шаблонах отчетов в формате *XML* для взаимодействия с ФГИС.

Выходными данными являются:

- хранимые данные в локальной БД приложения;

- сформированные отчеты, включающие в себя информацию, указанную пользователем.

1.3.2 Требования к надежности

При функционировании программного обеспечения должен осуществляться контроль входной и выходной информации. В случае ошибки должны выдаваться предупреждающие сообщения и предоставляться возможность исправления ошибок.

1.3.3 Требования к составу и параметрам технических средств

Во исполнение постановления правительства РФ от 3 декабря 2020г. №2013 «О минимальной доле закупок товаров российского происхождения» система предназначена для работы на персональных компьютерах российского производителя «Аквариус» (или эквивалент) со следующей минимальной конфигурацией:

– процессор с частотой 2.4 ГГц и выше;

– оперативная память 8Гбайт и выше;

– жесткий диск тип *SSD*;

– монитор не менее 24 дюйма.

1.3.4 Требования к информационной и программной совместимости

Программное обеспечение сбора данных об объектах недвижимости должно работать в системе, аттестованной по требованиям Федерального закона «О персональных данных» от 27.07.2006 N 152-ФЗ организаций имеющей право на аттестацию информационных систем (ИСПДн).

Клиентская часть программы должна быть представлена «тонким» клиентом – браузером российского производства, с поддержкой протокола *TLS* (например, *Yandex* браузер). Операционной средой на клиентском ПК должна быть ОС российского производства (например *ОС Linux*).

Серверная часть приложения должна быть должна быть ОС российского производства (например *ОС Linux*).

На клиентской и серверном ПК должны быть установлены средства антивирусной защиты российского производства.

1.3.5 Стадии и этапы разработки

Стадии и этапы разработки приложения представлены в таблице 1.3.

Таблица 1.3 - Стадии и этапы разработки

Наименование этапов работы	Объем работы	Срок выполнения
Анализ предметной области	9%	
Составление технического задания	5%	
Выбор языка программирования	10%	
Проектирование системы	10%	
Разработка программы	30%	
Разработка СУБД	20%	
Выполнение задания по БЖД	5%	
Выполнение задания по экономике	5%	
Оформление пояснительной записки	6%	

1.4 Анализ целесообразности использования заданных средств разработки

Реализуемое приложение содержит три слоя – клиентскую, серверную части и СУБД.

Для реализации клиентской части используется язык *JavaScript*. *JavaScript* язык программирования, который добавляет интерактивности web-сайту. Интерактивность необходима в поведении ответов при нажатии кнопок или при вводе данных в формы.

*Я*зык *JavaScript* многоцелевой инструмент, поддерживающий функциональные и объектно-ориентированные стили программирования. Язык *JavaScript* включает *API* для работы с массивами, выражениями и текстами, но не имеет никакой поддержки ввода-вывода, такой как средства хранения, мультимедиа, сетевые и графические средства.

Существуют аналоги и диалекты языка язык *JavaScript*, в таблице 1.4 приведен сравнительный анализ который показывает преимущества языка *JavaScript* перед аналогами.

Таблица 1.4 - Сравнительный анализ языков программирования

Наименование инструмента	Функциональные возможности	
	плюсы	минусы
JavaScript	- простой в освоении и выполнении; - может быть встроен в любую веб-страницу; - несмотря на расширение файла явное и простое описание синтаксических ошибок	- безопасность на стороне клиента отсутствует
CoffeeScript	- более лаконичное описание программы по сравнению с *JavaScript*	- не явное описание синтаксических ошибок

Наименование инструмента	Функциональные возможности	
	плюсы	минусы
TypeScript	- реализованы механизмы безопасности на стороне клиента; - строго типизирован; - имеет инструменты ООП	- является надстройкой *JavaScript*, что увеличивает время исполнения

Как видно из таблицы 1.4, языки программирования *CoffeeScrip*t и *TypeScript* являются или надстройкой или диалектом языка *JavaScript*, которые не дают явных преимуществ, при реализации минимального программного продукта. Поэтому основным инструментом реализации клиентской части выбран язык *JavaScript.*

После выбора языка программирования для реализации клиента требуется выбрать фрейворк, который содержит дополнительные инструменты для реализации клиента: *AngularvsReactvsVue*.

Фрейворк - готовая модель в программировании для быстрой разработки, на основе которой можно реализовать собственный код. Он задает структуру, определяет правила и предоставляет необходимый набор инструментов для создания проекта.

Согласно опросу *StackOverflow* 2022 [1], *React* является популярным фреймворком - 40,14% разработчиков, *Angular* - 22,96% и *Vue* - 18,97% разработчиков соответственно (рисунок 1.2).

Angular - это полноценный фреймворк веб-приложения, основанный на языке *TypeScript*, бесплатный и с открытым исходным кодом. Он широко используется для создания одностраничных приложений так называемых *SPA. Angular* привлек внимание многих разработчиков благодаря своему функциональному набору и технологической поддержке корпорации *Google*.

Angular не будет использоваться в данном проекте, так как *Angula r*поддерживает только*TypeScript* (рисунок 1.3).

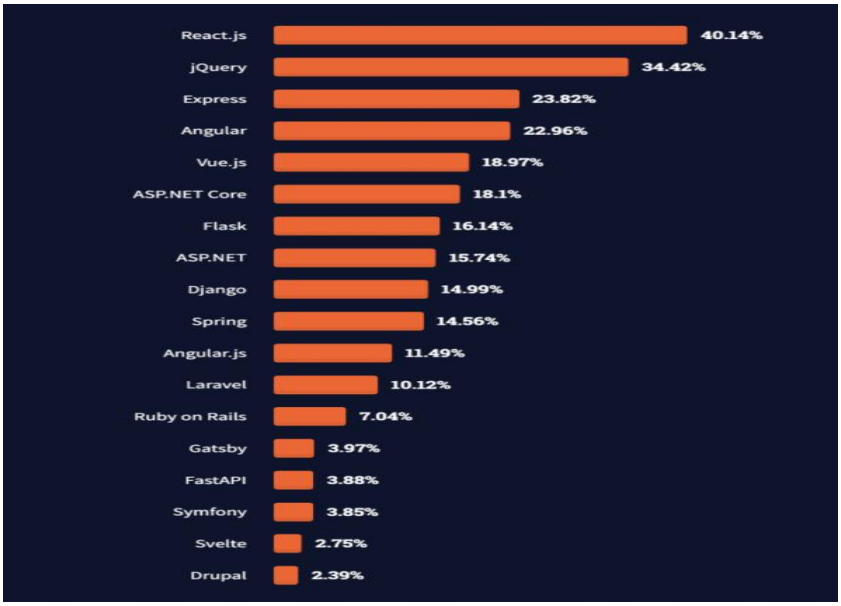


Рисунок 1.2 – Сравнительный анализ фреймворков на основе опроса сайта *StackOverflow*

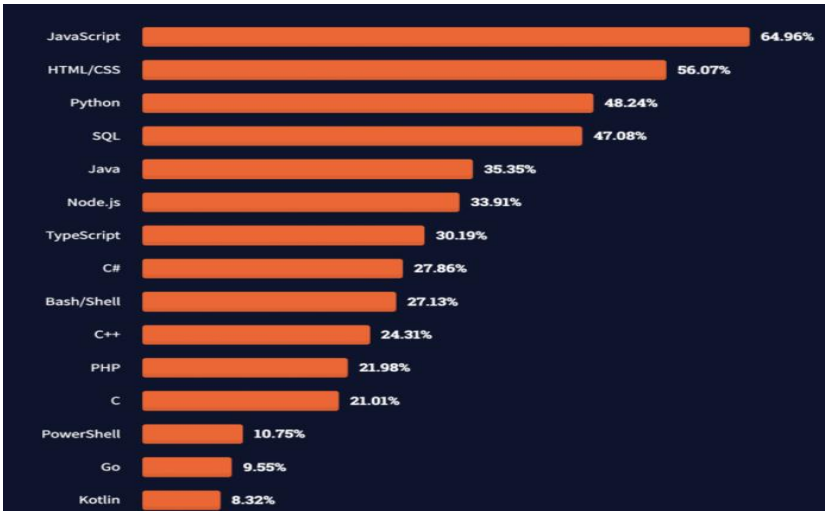


Рисунок 1.3 – Сравнительный анализ языков программирования на основе опроса сайта *StackOverflow*

Vue - это полноценный фреймворк с открытым исходным кодом, получивший широкую популярность для создания одностраничных приложений (*SPA*). Разработчики фреймворка *Vue* взяли за основу *Angular*, но несмотря на легкость в освоении и пригодности для реализации малых проектов, *Vue* обладает недостатками критичными при реализации дипломного проекта: отсутствие прямой технологической поддержки поддерживаются *Google* и *Facebook*, отсутствие корректной работы со

старыми версиями операционных систем и веб-браузеров, что важно для конечных пользователей государственных учреждений.

React - это интерфейсная библиотека *JavaScript* с открытым исходным кодом, используемая для разработки пользовательских интерфейсов на основе компонентов. В настоящее время поддерживается сообществом с открытым исходным кодом и *Facebook*. Это наиболее широко используемая библиотека на основе *JavaScript* для создания web-приложений. Более того, помимо создания пользовательского интерфейса, он предлагает несколько функций, таких как *Flux* и *ReactNative*. Одним из самых больших преимуществ использования *React* является то, что можно создавать webи мобильное приложения, используя одну и ту же кодовую базу.

В связи с тем, что *React* поддерживает язык *JavaScript* и соответствует требованиям для реализации программы сбора данных об объектах недвижимости он выбран в качестве инструмента реализации клиентской части приложения.

Для реализации серверной части приложения требуется выбрать иной инструмент среди фреймворков: *Laravel*, *Django* и *RubyonRails*.

Laravel - это *PHP*-фреймворк веб-разработки с открытым исходным кодом. Он следует шаблону архитектуры *ModelViewController* (*MVC*). Создан и лицензирован *MIT* лицензией. Является прямым конкурентом *RubyonRails*.

Django - это веб-фреймворк *Python*, который позволяет разрабатывать безопасные и поддерживаемые веб-сайты. *Django* бесплатный с открытым исходным кодом, имеет активное сообщество, документацию и вариантs бесплатной и платной поддержки.

RubyonRails - это платформа на основе *MVC* (*modelviewcontroller*), используемая для веб-разработки и программирования приложений на стороне сервера приложений. *Rails* - это фреймворк, написанный на языке *Ruby* и расширяющий возможности *Ruby* для решения задач разработки серверной части приложения.

Сравнительный анализ фреймворков приведен в таблице 1.5.

Таблица 1.5- Сравнительный анализ фреймворков

Инструменты	*Laravel*	*Django*	*RubyonRails*
Язык программирования	*PHP*	*Python*	*Ruby*
Сложность реализации кода и конструкций фреймворка	Удобный для начинающих разработчиков	Требует фундаментальных знаний *Python*	Требует фундаментальных знаний *Ruby*
Масштабируемость	Имеется	Имеется	Имеется
Архитектура приложения	*MVC*	*MVT*	*MVC*
Техническая поддержка	Обладает обширной экосистемой инструментов и пакетов	Поддерживающее сообщество профессионалов	Богатую экосистему библиотек, инструментов и пакетов
Наличие поддержки*API* для ФГИС ЕГРН	Имеется	Нет	Нет
Технологии работы с данными	*ORM*	*Eloquent ORM*	*ActiveRecord*

В связи с тем, что *Laravel* является более легким для изучения и поддерживается *API* информационной платформы ФГИС ЕГРН он был выбран в качестве инструмента реализации серверной части приложения.

Третьим этапом исследования является выбор СУБД для хранения локальных данных.

СУБД *SQL* и *NoSQL* типа являются двумя основными категориями баз данных.

Ниже описаны основные различия между СУБД *SQL* и *NoSQL* типа:

- структура данных: *SQL* базы данных имеют жесткую схему данных, где необходимо определить поля и типы данных заранее для каждой таблицы. *NoSQL* СУБД имеют гибкую схему данных, где можно хранить данные без определения схемы;

- запросы: *SQL* использует язык запросов *SQL* для получения данных из базы данных, в то время как *NoSQL* использует специальные запросы для доступа к данным;

- масштабируемость: *NoSQL* легче масштабируются, они могут расширяться по горизонтали и вертикали. *SQL* СУБД имеют определенные ограничения на масштабирование.

В связи с тем, что база данных ФГИС ЕГРН реализована на СУБД *SQL*, данный тип базы данных будет использован в дипломном проекте. Проведем сравнительный анализ трех СУБД *SQL*: *MySQL, PostgreSQL* и *Oracle*.

СУБД *MySql* - бесплатная и открытая СУБД, доступна для использования как в коммерческих, так и в некоммерческих целях.

Достоинства:

- простота администрирования и поддержки;

- быстрая и легковесная СУБД, которая хорошо подходит для небольших проектов или сайтов.

Недостатки - недостаточный функционал для крупных проектов.

СУБД *PostgreSQL* - бесплатная и открытая СУБД, которая хорошо подходит для больших проектов и транзакций.

Достоинства:

- более широкие возможности и функционал по сравнению с *MySQL*;

- сильные механизмы безопасности и контроля целостности данных.

Недостатки:

- более трудна в использовании и настройке;

- большое потребление ресурсов на выполнение запросов.

СУБД *Oracle* - профессиональная СУБД, которую используют крупные компании и некоммерческие структуры.

Достоинства:

- широкий функционал, возможность обработки больших данных;

- мощные функциональные возможности и техническая поддержка.

Недостатки:

- высокая цена для коммерческой версии;

- сложная интеграция с другими системами;

- зарубежное производство и техническая поддержка.

Так как проекту требуется бесплатная СУБД легкая во внедрении и с хорошей бесплатной технической поддержкой выбор сделан в пользу СУБД *MySQL*.

1.5 Проектирование автоматизированной системы

Проектирование с помощью *UML* диаграмм возможно на разных уровнях детализации.

Некоторые диаграммы описывать систему с более высокого, более абстрактного уровня, в то время как другие модели обеспечивают большую детализацию.

Модели *UML* содержат элементы модели, такие как актеры, варианты использования, классы и пакеты, а также один или несколько диаграммы, которые показывают конкретную перспективу системы. Модель может также содержат другие, более подробные модели.

Содержимое проекта моделирования подразделяется на три типа логических папок: диаграмм, моделей и профилей. Эта структура отображает логическое удержание элементов модели *UML*, независимо о том, где они хранятся физически.

Разработчик может использовать модели для выполнения следующих действий:

- визуально представьте систему, которую вы хотите создать;
- донесите свое видение системы до клиентов и коллег;
- разработка и тестирование архитектуры системы;
- используйте диаграммы *UML* для непосредственной генерации кода.

Основными типами диаграмм для реализации программного проекта являются:

- структурная схема: диаграмма классов, диаграмма объектов, диаграмма пакетов, диаграмма компонентов, диаграмма развертывания;
- поведенческие диаграммы: диаграмма деятельности, диаграмма вариантов использования, диаграмма конечных автоматов;

- схема взаимодействия: временная диаграмма, диаграмма последовательности, диаграмма совместной работы.

1.6 Диаграммы деятельности

Ниже используется диаграмма действий для визуального представления действий в системе, аналогично блок-схеме или диаграмме потока данных. Моделируемые действия могут быть последовательными и параллельными.

Диаграммы деятельности представлены на рисунках 1.4 — 1.12.

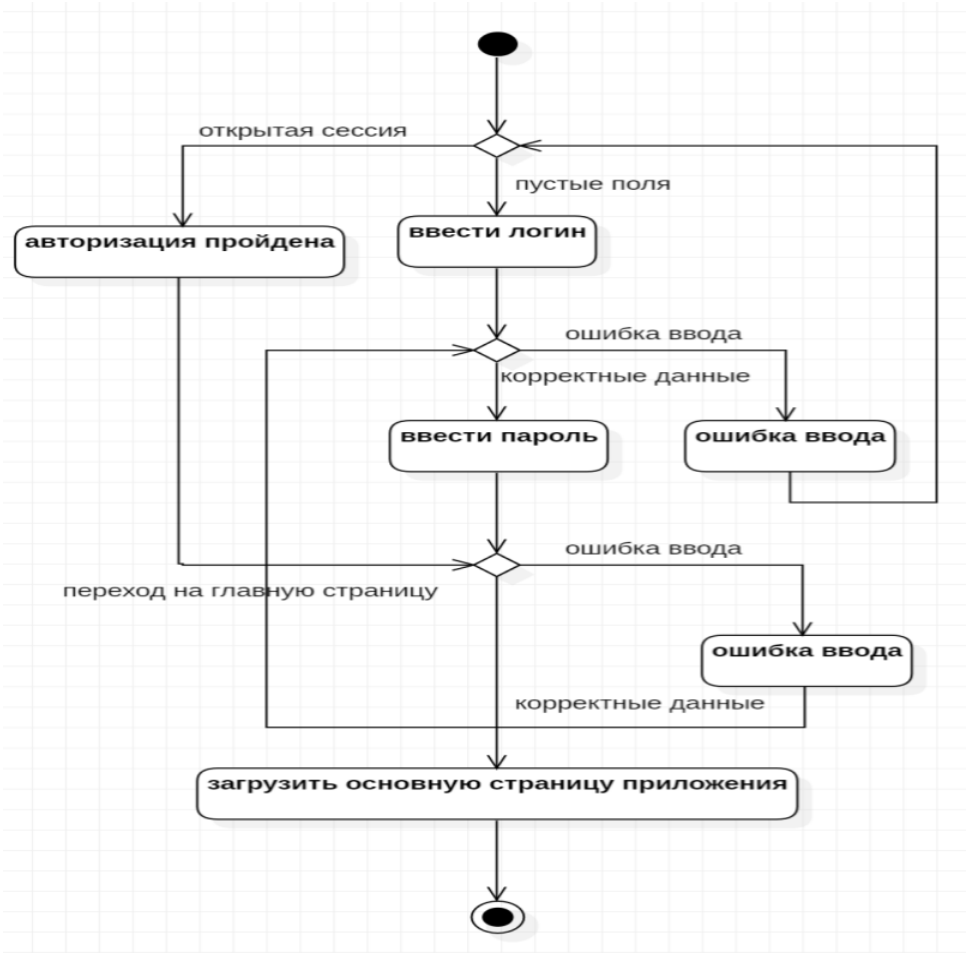


Рисунок 1.4 –Диаграмма деятельности: авторизация

Авторизация заключается в выполнении следующих шагов:

- отображение страницы авторизации;

- ввод учетных данных пользователя;

- если данные верны, то переход на основную страницу приложения, иначе вывод информации об ошибке.

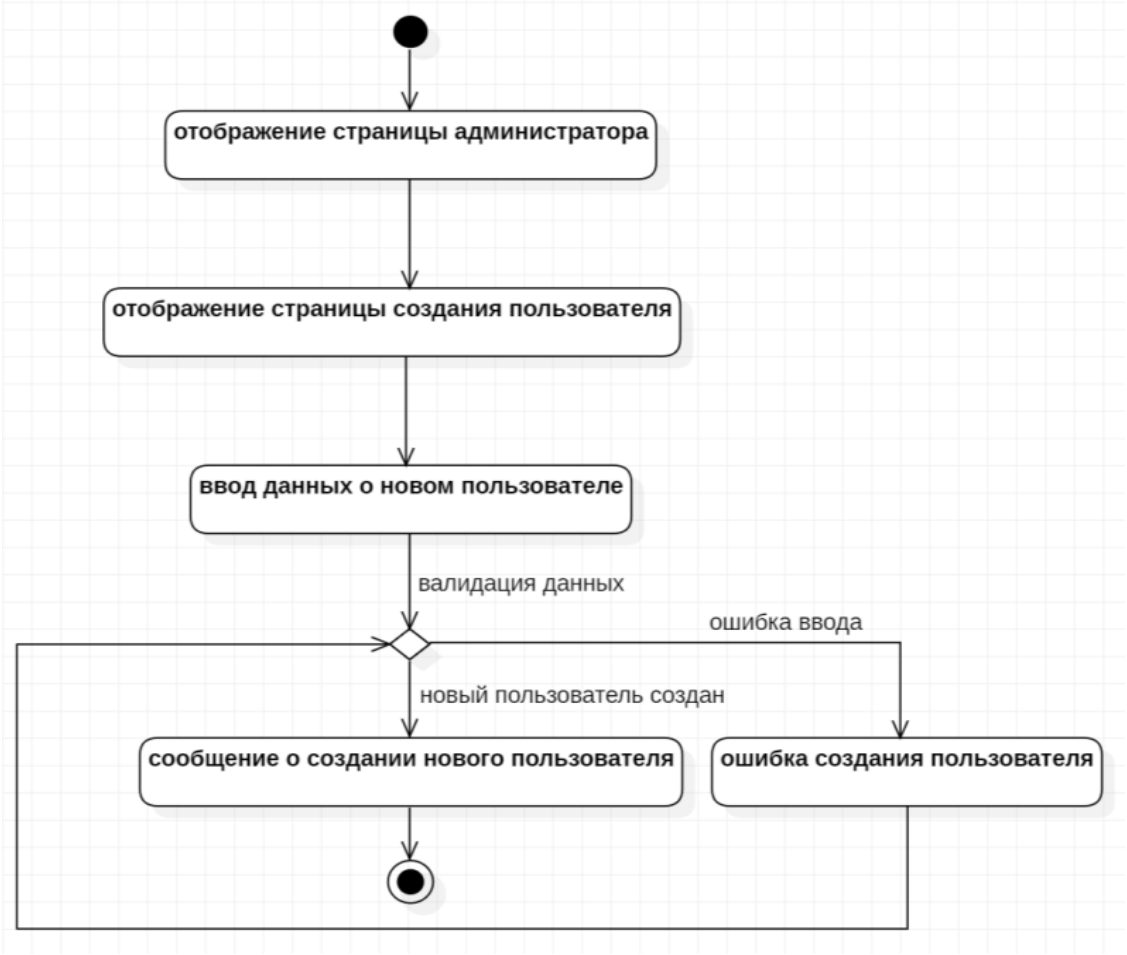


Рисунок 1.5 –Диаграмма деятельности: создание пользователя от имени администратора

Создание пользователя заключается в выполнении следующих шагов:

- отображение страницы администратора;
- переход на страницу создания пользователя;
- ввод данных о новом пользователе;
- если данные верны, то сгенерировать сообщение о создании нового пользователя, иначе вывод информации об ошибке.

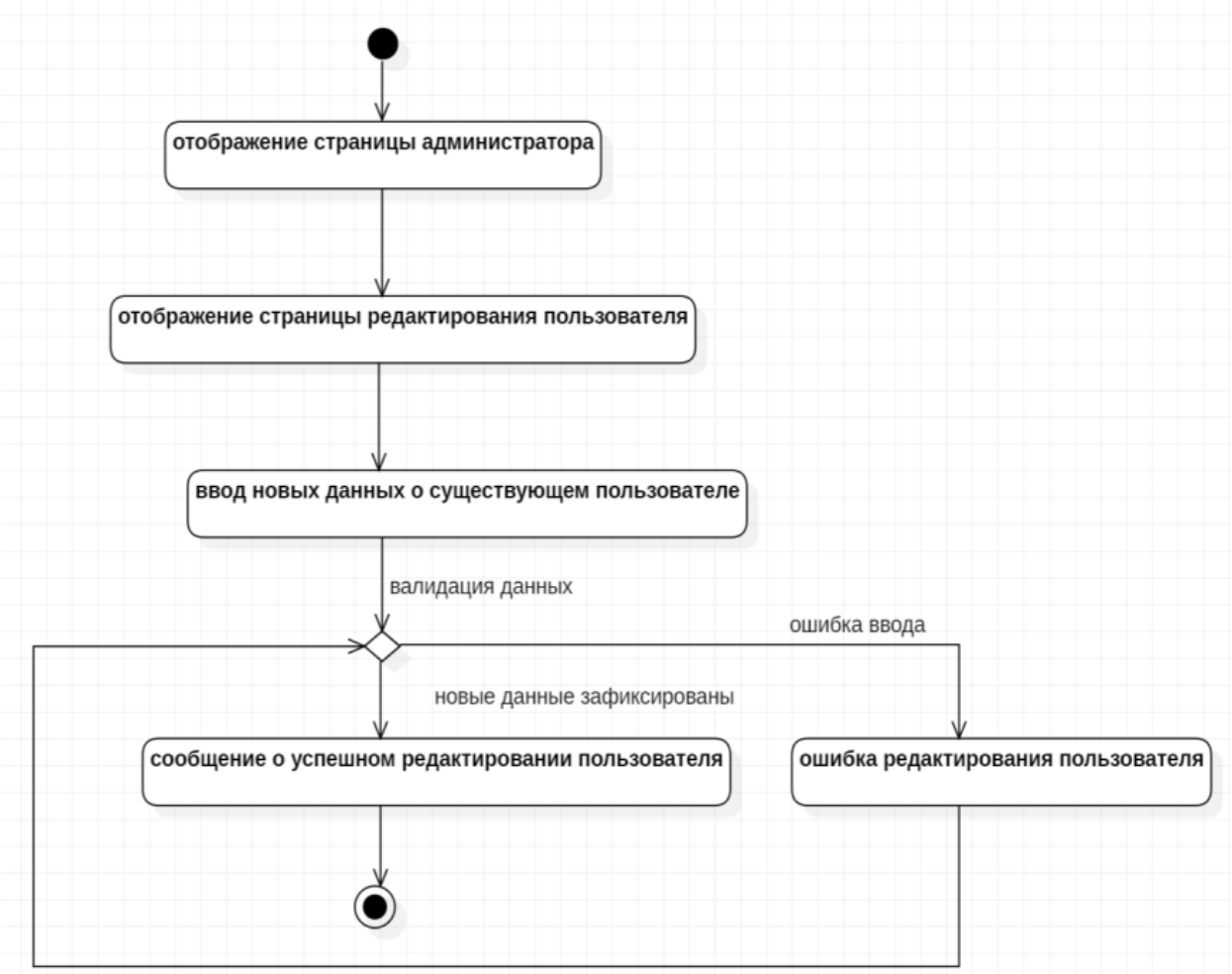


Рисунок 1.6 –Диаграмма деятельности: редактирование пользователя от имени администратора

Редактирование пользователя заключается в выполнении следующих шагов:

- отображение страницы администратора;
- переход на страницу редактирования пользователя;
- ввод новых данных о существующем пользователе;
- если данные верны, то перейти на главную страницу, иначе вывод информации об ошибке.

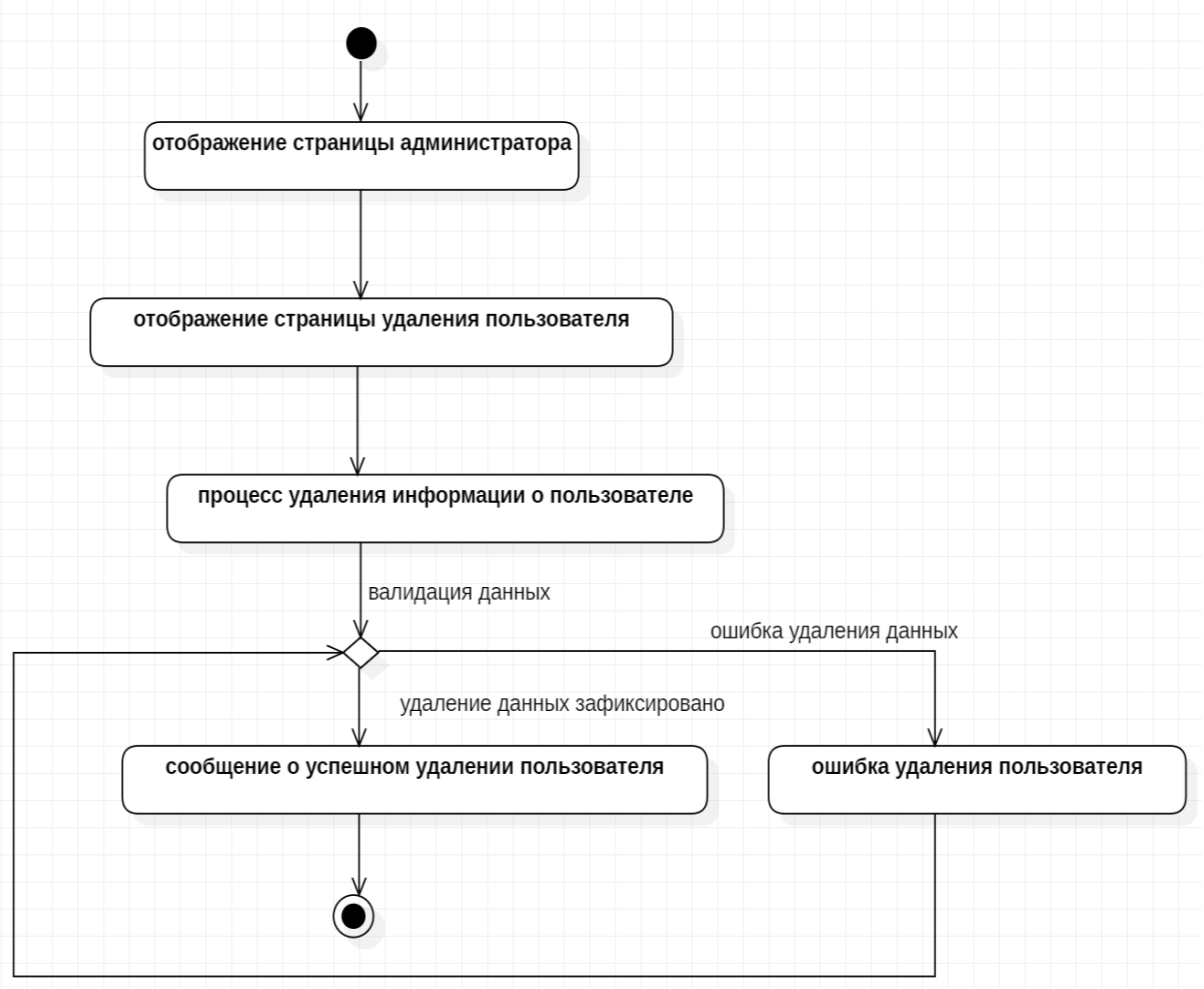


Рисунок 1.7 –Диаграмма деятельности: удаление пользователя от имени администратора

Удаление пользователя заключается в выполнении следующих шагов:

- отображение страницы администратора;

- переход на страницу удаления пользователя;

- если процесс удаления пользователя был успешен, то сгенерировать сообщение об удалении пользователя, иначе вывод информации об ошибке.

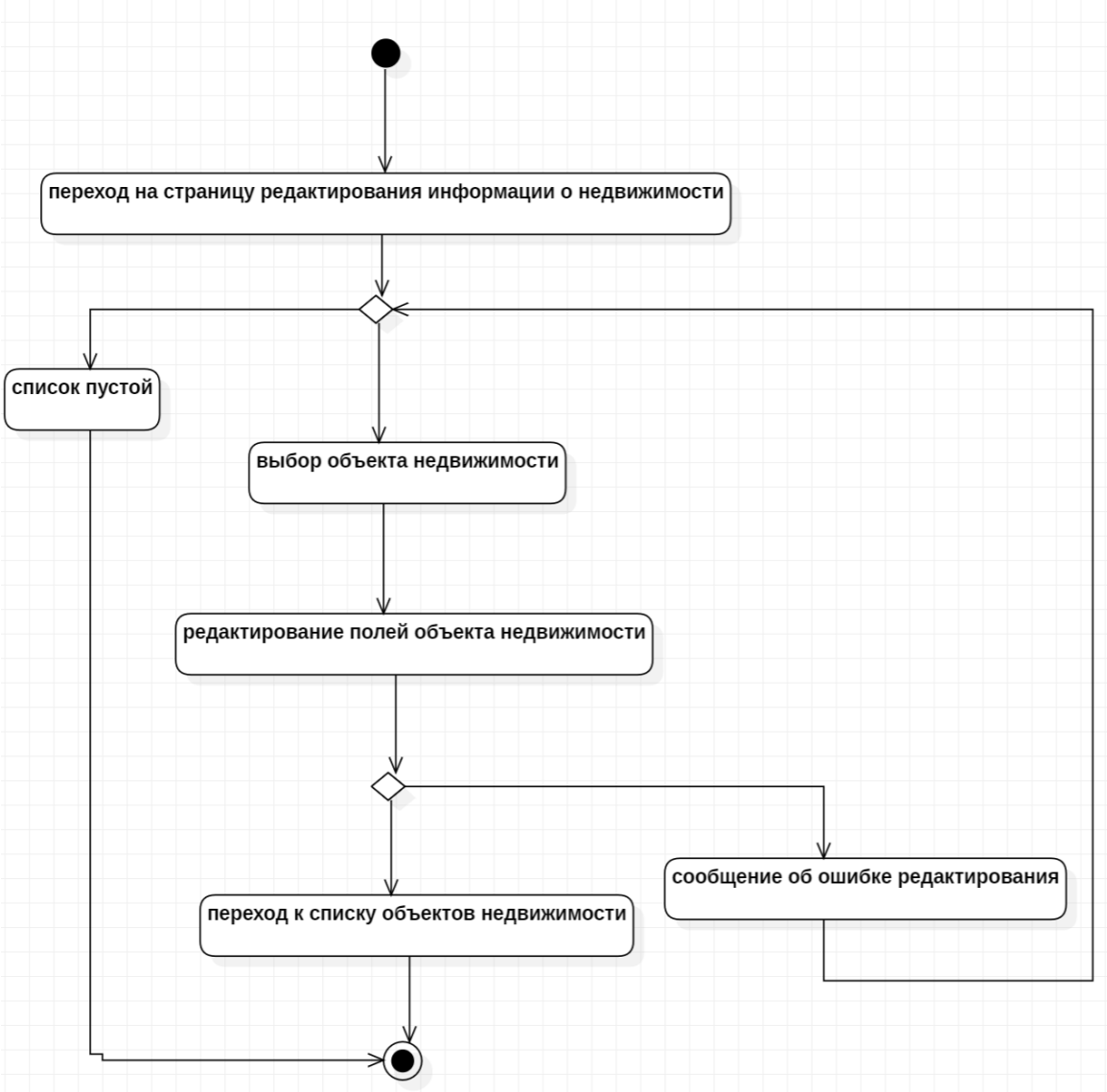


Рисунок 1.8 –Диаграмма деятельности: редактирование информации о недвижимости от имени пользователя

Редактирование информации о недвижимости заключается в выполнении следующих шагов:

- отображение главной страницы приложения (вход под пользователем);

- переход на страницу редактирования информации об объекте недвижимости;

- процесс редактирования полей об объекте недвижимости;

- если процесс редактирования был успешен, то перейти к списку объектов недвижимости, иначе вывод информации об ошибке.

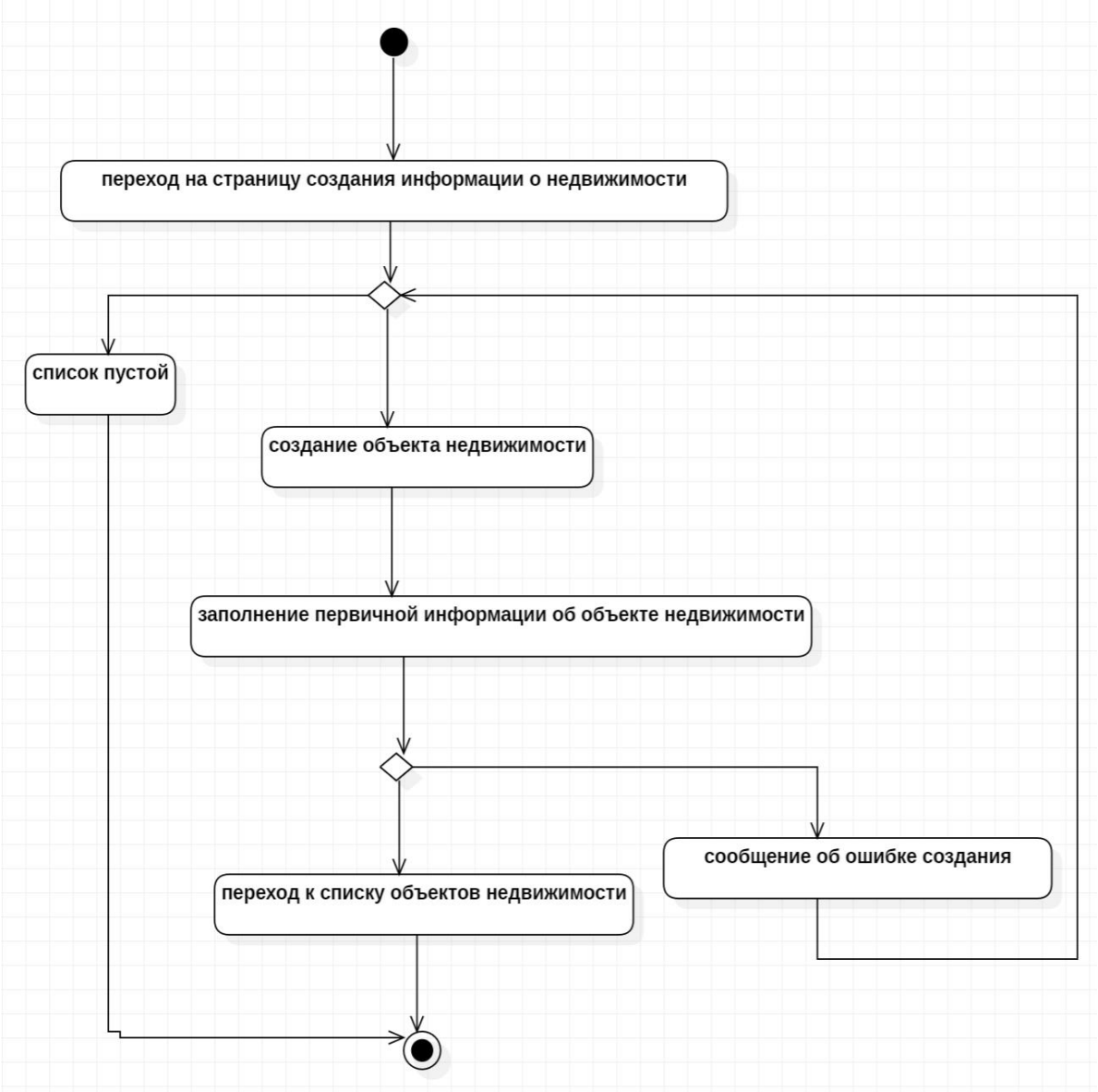


Рисунок 1.9 –Диаграмма деятельности: создание информации о недвижимости от имени пользователя

Создание информации о недвижимости заключается в выполнении следующих шагов:

- отображение главной страницы приложения (вход под пользователем);
- переход на страницу создания информации об объекте недвижимости;
- процесс создания объекта недвижимости и информации о нем;
- если процесс создания был успешен, то перейти к списку объектов недвижимости, иначе вывод информации об ошибке.

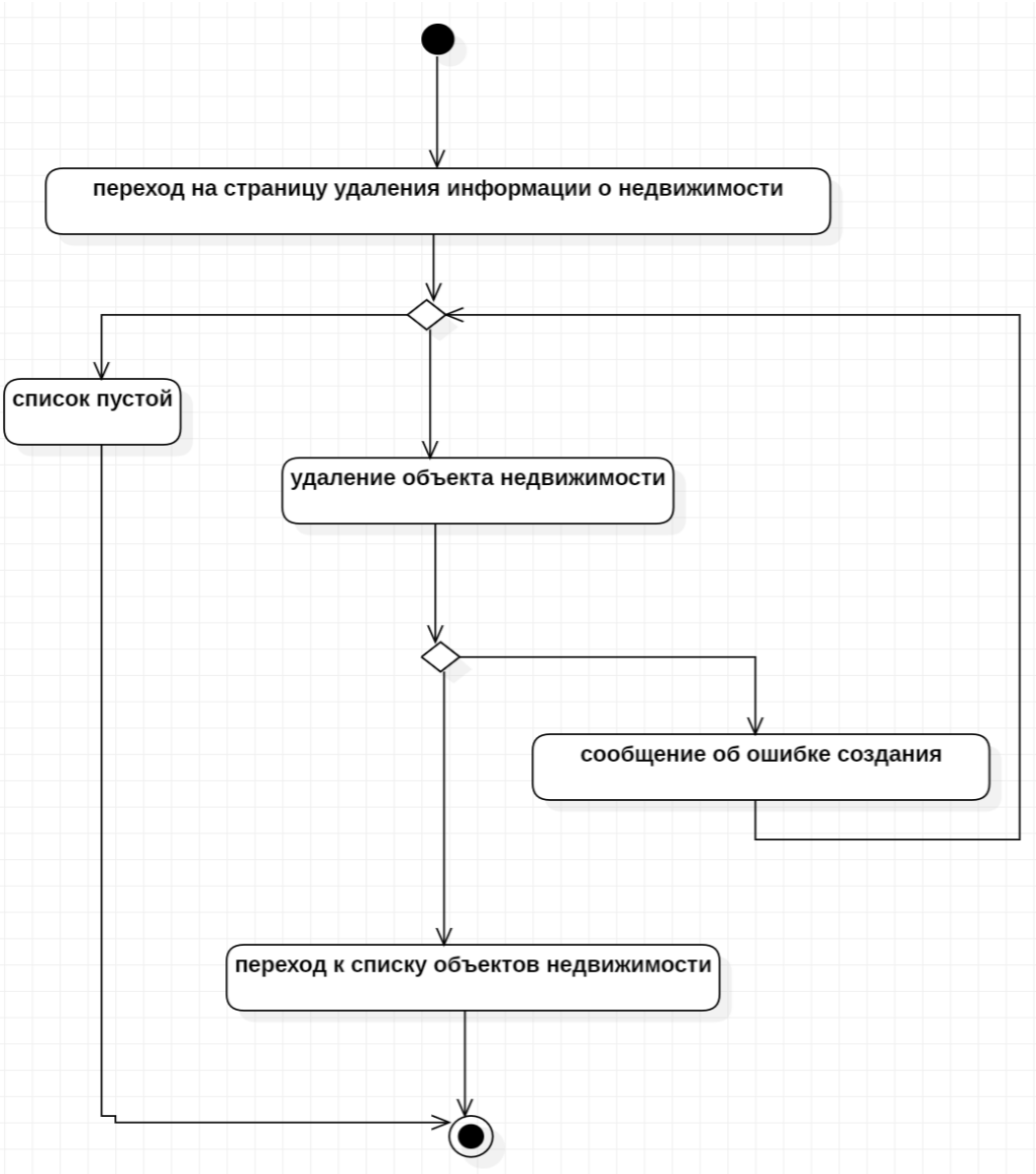


Рисунок 1.10 –Диаграмма деятельности: удаление информации о недвижимости от имени пользователя

Удаление информации о недвижимости заключается в выполнении следующих шагов:

- отображение главной страницы приложения (вход под пользователем);
- переход на страницу удаления информации об объекте недвижимости;
- процесс удаления объекта недвижимости и информации о нем;
- если процесс удаления был успешен, то перейти к списку объектов недвижимости, иначе вывод информации об ошибке.

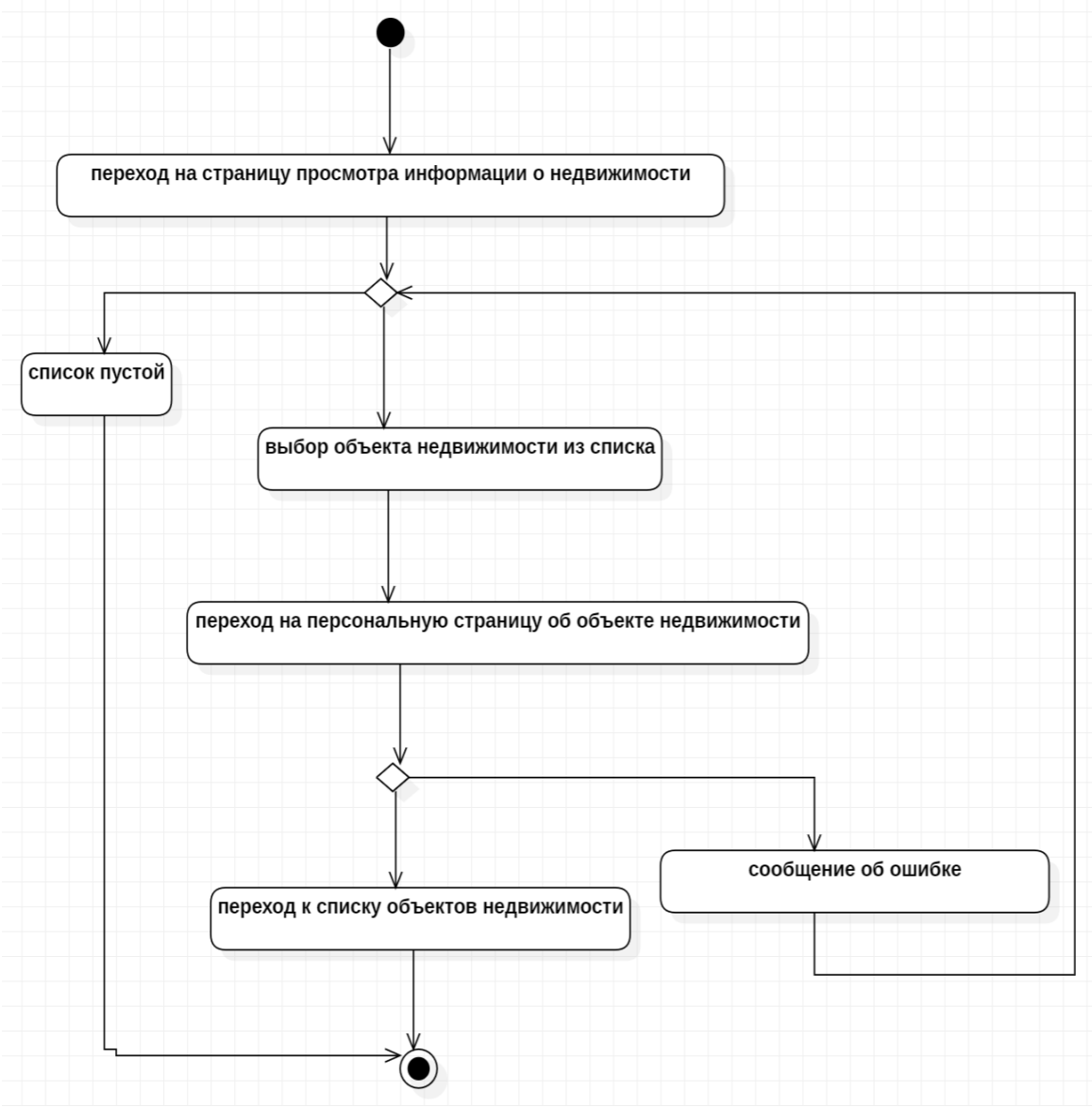


Рисунок 1.11 –Диаграмма деятельности: просмотр информации о недвижимости от имени пользователя

Просмотр информации о недвижимости заключается в выполнении следующих шагов:

- отображение главной страницы приложения (вход под пользователем);
- выбор объекта недвижимости из списка;
- переход на персональную страницу объекта недвижимости;
- процесс просмотра объекта недвижимости и информации о нем;
- если процесс просмотра был успешен, то перейти к списку объектов недвижимости, иначе вывод информации об ошибке.

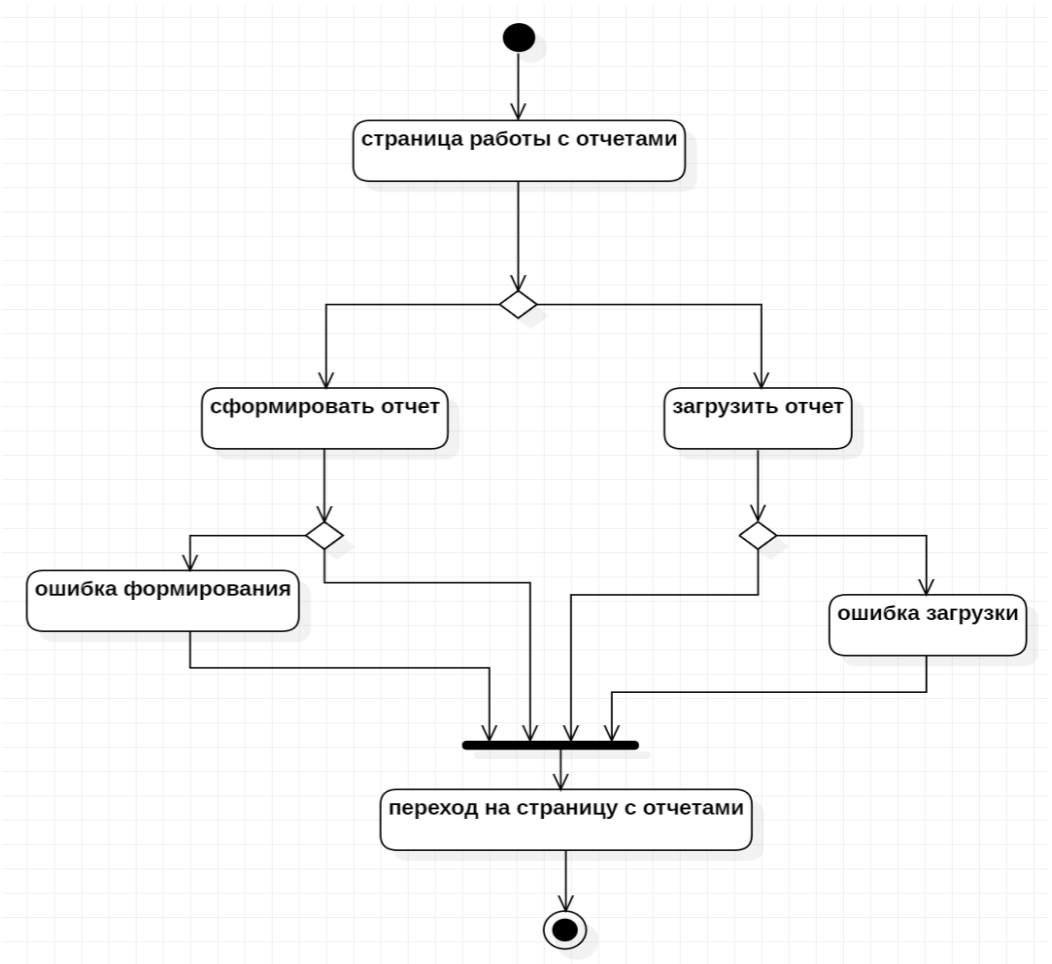


Рисунок 1.12 –Диаграмма деятельности: просмотр информации о недвижимости от имени пользователя (специалиста по недвижимости)

Работа с отчетами заключается в выполнении следующих шагов:

- отображение страницы отчетов приложения (вход под пользователем – специалист по недвижимости);

- выбор операции;

- если операция «сформировать отчет» перейти на страницу формирования отчета;

- на странице формирования отчета выбрать объект недвижимости и сформировать *XML* файл отчета;

- если операция «загрузить отчет» перейти на страницу загрузки отчета;

- на странице загрузить отчет выбрать *XML* файл отчета и отправить его;

- если процесс просмотра был успешен, то перейти к списку объектов недвижимости, иначе вывод информации об ошибке.

1.7 Диаграмма последовательности

Диаграмма последовательности представляет поток сообщений в системе и также называется диаграммой событий. Это помогает представить динамичные сценарии. Диаграмма последовательности требуется для моделирования высокоуровневого взаимодействия между активными объектами внутри системы (рисунок 1.13).

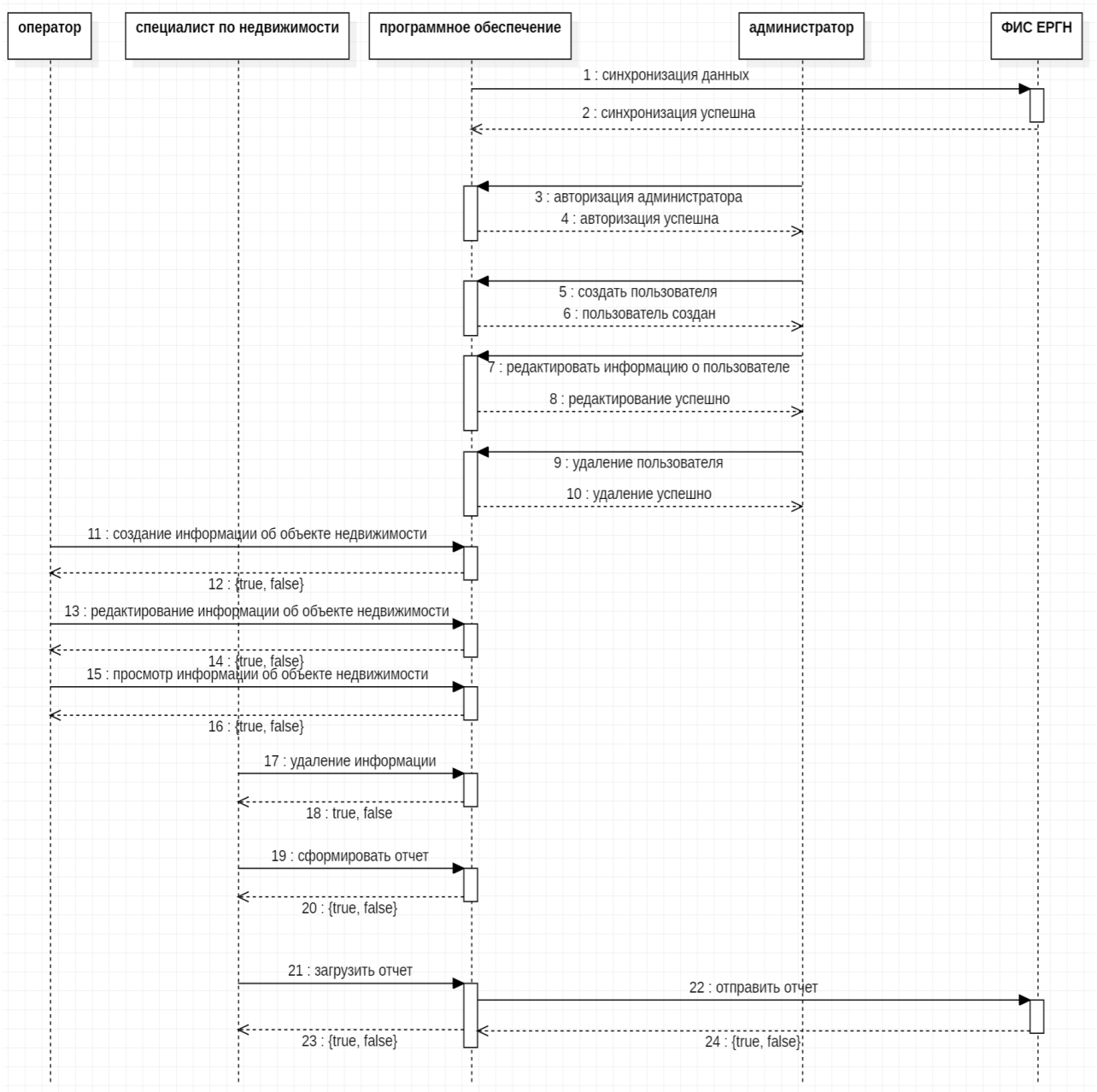


Рисунок 1.13 –Диаграмма последовательности программной системы

1.8 Диаграмма развертывания

На диаграмме развертывания показаны основные узлы, на которых развернуты программные компоненты. Она отображает общую конфигурацию и топологию шины, показывает расположение компонентов на каждом узле в системе, указывает на наличие физических соединений и информационных путей между аппаратными устройствами. Согласно техническому заданию была разработана диаграмма развертывания, представленная на рисунке 1.14.

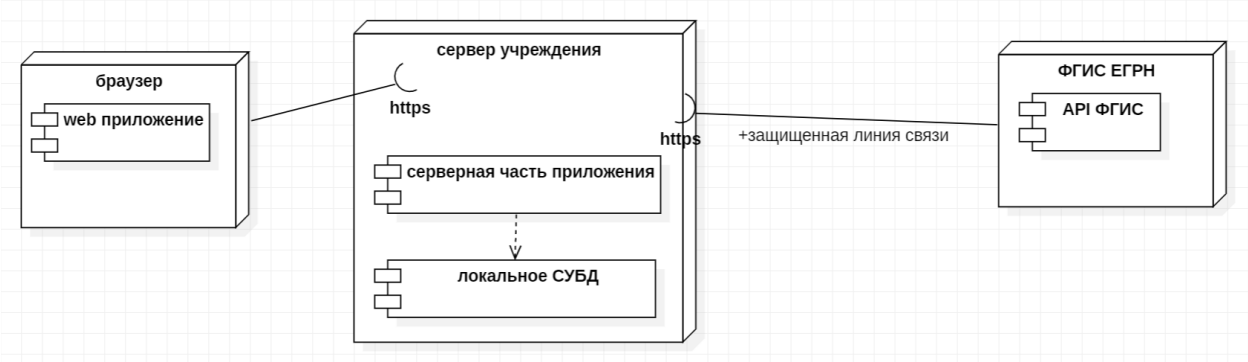


Рисунок 1.14 –Диаграмма развертывания

На диаграмме развертывания на рисунке 1.14 показана система, в рамках которой реализовано приложение.

Каналы связи между сервером учреждения и ФГИС ЕГРН защищен сертифицырованным ФСЭК программно-аппаратным шлюзом безопасности *ViPNetCoordinator HW* 1000.

1.9 Выводы по предпроектному анализу

В первой главе дипломного проекта проведен анализ предметной области, показана необходимость в разработке программного обеспечения с описанными функциональными возможностями.

С помощью *UML* диаграмм проведено проектирование программного обеспечения.

2 Проектирование и реализация программы сбора данных об объектах недвижимости

Реализованное приложение сбора данных об объектах недвижимости имеет следующую архитектуру (рисунок 2.1): «тонкий клиент», фронтенд приложения, серверная часть, отцветающая за обработку запросов со стороны клиентской части приложения и базы данных.

На стороне серверной части реализованы механизмы взаимодействия с ФГИС с помощью внешних *API.*

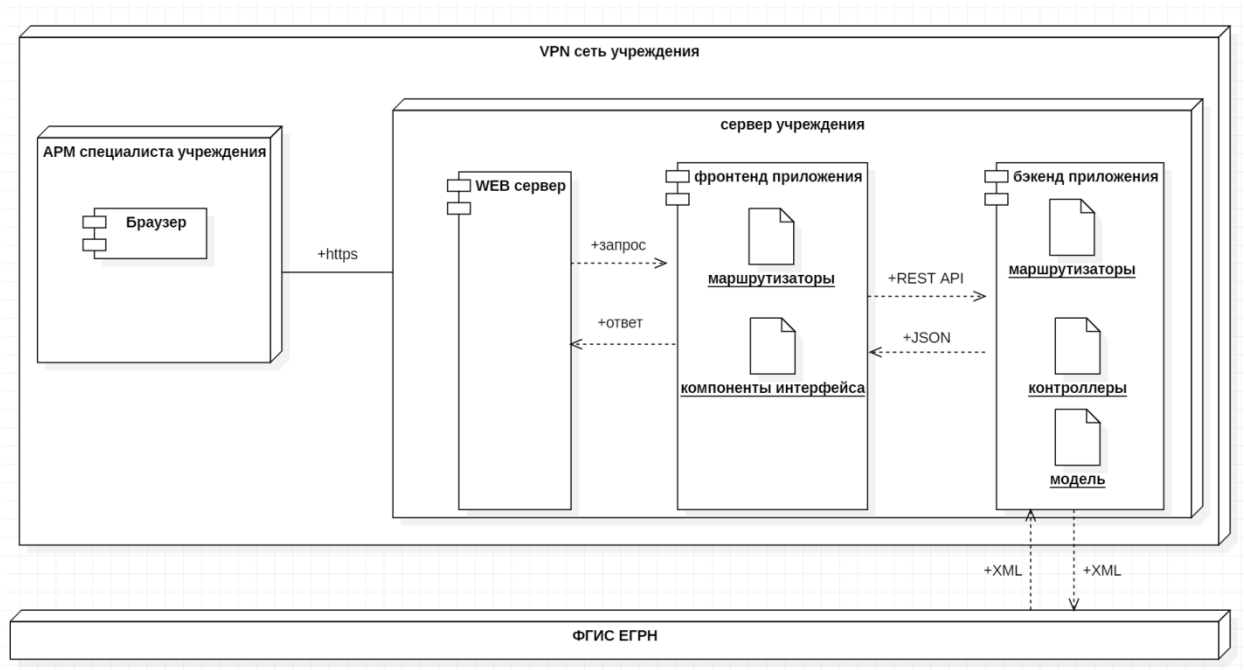


Рисунок 2.1 –Диаграмма развертывания. Архитектура приложения

2.1 Проектирование базы данных приложения

Корректное проектирование и реализация базы данных является залогом для реализации серверной части приложения и интерфейса. Проект базы данных предоставляет собой структуры данных и связи между ними.

Для проектирования БД на концептуальном уровне существуют объектно-ролевые модели (*ORM*) и диаграмм взаимосвязи сущностей (*ERD).* Диаграммы *ORM* и *ERD* описывают, как связаны различные объекты или сущности между собой.

Разработка концептуальной модели данных приводит к тому, что фиксируется обобщенная информацию об отношениях сущностей. Сущности помимо данных, взаимосвязаны между собой следующими способами:

- многие ко многим - одна или несколько записей базы данных относятся к элементам в другой таблице;

- один ко многим - родительский узел подключен к нескольким дочерним узлам;

- один к одному - объект в таблице может относиться только к одному объекту в другой таблице.

Как только определен масштаб проекта, данные и взаимосвязи между сущностями, необходимо перейти к логической модели. Логическая модель данных предоставляют детальную информацию о структуре данных и взаимосвязях сущностей. В моделях выделяются первичные ключи и внешние ключи, которые определяют связи между типами данных. Логическая модель показывает атрибуты данных конкретного объекта. Пропуск этапа логического моделирования данных может привести к пропуску определенных требований, что приведет к некачественному дизайну физической базы данных.

Физическая модель данных предлагает окончательный дизайн и схему того, как данные хранятся в базе данных. Модель включает в себя элементы, специфичные для базы данных, такие как процедуры, индексы, ограничения, представления, столбцы, типы данных и таблицы.

2.1.1 Проектирование логической модели базы данных приложения

Ниже представлены таблицы логической модели реализованной БД, с описанием сущностей (таблицы 2.1 – 2.8).

Таблица 2.1 – Атрибуты сущности «Страны»

Наименование атрибута	Тип	Обязательный
Идентификатор страны	Целый	Да
Наименование страны	Строка	Да
Код страны	Целый	Да

Таблица 2.2 – Атрибуты сущности «Города»

Наименование атрибута	Тип	Обязательный
Идентификатор города	Целый	Да
Наименование города	Строка	Да
Почтовый индекс города	Целый	Да

Таблица 2.3 – Атрибуты сущности «Объект недвижимости»

Наименование атрибута	Тип	Обязательный
Идентификатор объекта недвижимости	Целый	Да
Наименование объекта недвижимости	Строка	Да
Ссылка на город	Целый	Да
Тип объекта недвижимости	Целый	Да
Общая площадь	Целый	Да
Признак исторического объекта	Целый	Да
Количество квартир или частей	Целый	Да
Количество гаражей	Целый	Да
Количество паркинговых мест	Целый	Да
Общая информация об объекте недвижимости	текст	Нет
Статус объекта недвижимости	Целый	Да
ОКАТО	Строка	Да
ОКТМО	Строка	Да
Стоимость объекта недвижимости	Целый	Да
Налог на объект недвижимости	Целый	Да
Кадастровый номер объекта недвижимости	Целый	Да

Таблица 2.4 – Атрибуты сущности «Сотрудник учреждения»

Наименование атрибута	Тип	Обязательный
Идентификатор сотрудника	Целый	Да
Фамилия сотрудника	Строка	Да
Отчество сотрудника	Строка	Да
Имя сотрудника	Строка	Да
Должность сотрудника	Строка	Да
Статус сотрудника	Целый	Да

Таблица 2.5 – Атрибуты сущности «Запрос отчета»

Наименование атрибута	Тип	Обязательный
Идентификатор запроса	Целый	Да
Идентификатор должностного лица сделавшего запрос	Целый	Да

Наименование атрибута	Тип	Обязательный
Идентификатор сотрудника обрабатывающего запрос	Целый	Да
Периодичность запроса	Целое	Да
Количество ответов	Целое	Да
Дата подписания ответа на запрос	Временная метка	Да
Дата начала работы с запросом	Временная метка	Да
Дата окончания работы с запросом	Временная метка	Да

Таблица 2.6 – Атрибуты сущности «Инициатор запроса»

Наименование атрибута	Тип	Обязательный
Идентификатор инициатора	Целый	Да
Официальное название организации	Целый	Да
Адрес организации	Целый	Да
Контактное лицо	Целое	Да
телефон официальный	Целое	Да
Мобильный телефон	Целое	Да
Адрес электронной почты	Строка	Да
Описание инициатора запроса	Текст	Да

Таблица 2.7 – Атрибуты сущности «Транзакции запроса»

Наименование атрибута	Тип	Обязательный
Идентификатор транзакции	Целый	Да
Идентификатор инициатора запроса	Целый	Да
Счетчик запросов	Целый	Да
Описание отправителя запроса	Текст	Да
Описание того кто обрабатывает запрос	Текст	Да
Детали запроса	Текст	Да
Ссылка на официальный запрос (скан копия)	Файл	Нет
Дата создания запроса	Временная метка	Да
Дата отправки запроса	Временная метка	Да

Таблица 2.8 – Атрибуты сущности «Настройки»

Наименование атрибута	Тип	Обязательный
Идентификатор	Целый	Да
Хеш для шифрования	Целый	Да
Номер сети криптошлюза для связи с ФГИС	Целый	Да
актуальный IP адрес ФГИС	Целый	Да

В результате этапа логического проектирования БД, было выделено 8 сущностей, которые отвечают за следующие данные:

- информация об объектах недвижимости учреждения;

- сотрудники учреждения, отвечающих за поддержание информации об объектах недвижимости в актуальном состоянии;

- запросы со стороны юридических лиц, в том числе и контролирующих органов, по вопросам связанным с объектами недвижимости;

- настройки программы для связи с ФГИС.

2.1.2 Проектирование физической модели базы данных приложения

На основании логической модели БД реализуется физическая модель, содержащая таблицы, строки с данными и типами данных, с указанием атрибутов первичных и внешних ключей. Физическая модель содержит большее число таблиц в связи с тем, что для реализации некоторых сущностей требуется основная таблица и вспомогательные таблицы: словари и таблицы связи.

Физическая модель (рисунок 2.2) данных состоит из трех основных предметных областей: данные о недвижимости, данные об отчетах об объектах недвижимости и настройки сети.

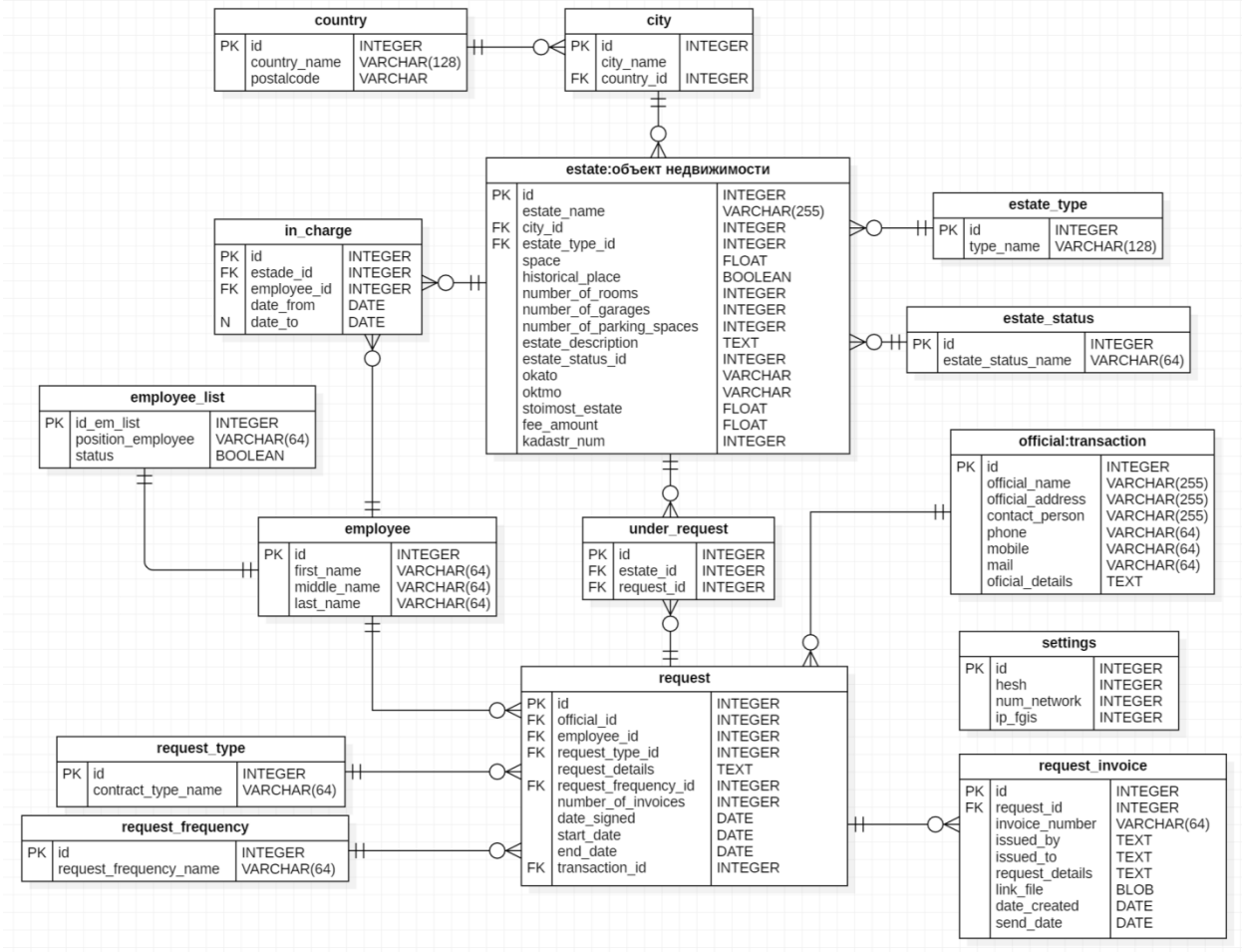


Рисунок 2.2 –Физическая модель базы данных

Область модели «объект недвижимости» состоит из 6 таблиц. Центральной таблицей в этой предметной области является таблица «*estate*» (таблица 2.11). Она содержит список всех объектов недвижимости и их характеристики.

Данная область содержит два словаря, которые относятся к таблице «*estate*»: *estate_type* (таблица 2.12) и *estate_status* (таблица 2.13). Оба словаря содержат только идентификатор и атрибут *name*.

В таблице словаре «*estate_type*» (таблица 2.12) хранятся такие значения, как квартира, дом, поле и т.д. В таблице словаре «*estate_status*» (таблица 2.13) содержатся значения, указывающие, доступна ли недвижимость в данный момент учреждению или нет: «сдана в аренду», «взята в аренду», «переведена на баланс», «снята с баланса».

Местоположение каждого объекта недвижимости определяется использованием двух таблиц словарей: *country* (таблица 2.9) и *city* (таблица 2.10).

Таблица *«in_charge»* (таблица 2.14) необходима для определения сотрудника ответственного за объект недвижимости, в том числе и по вопросам ответа на запросы или для формирования запросов в надлежащие инстанции по вопросам данного объекта. В данной таблице также определен временной интервал, в рамках которого сотрудник отвечает за объект.

В области физической модели «Запрос» центральной таблицей является таблица «*contract*». В ней хранятся детали запроса об объекте недвижимости, и существует связь сотрудника и инициатора запроса.

В таблице «*under_contract*» реализована связь запросов и имущественных прав, данная связь реализована парой внешних ключей, *estate_id* и *contract_id*.

Информация о документах запроса, в том числе и скан копии запроса, хранится в таблице «*request_invoce*» (таблица 2.21).

В процессе проектирования будут использоваться еще два словаря для описания запроса: *request_type* (таблица 2.19) и *request _frequency* (таблица 2.20). Таблица «*request_type*» (таблица 2.19) «физического лица», «посредничество», ««запрос контролирующего органа». Таблица *request_frequency* (таблица 2.20) описывает, как часто создается отчет по запросу: «ежемесячно», «ежеквартально», «раз в год».

Таблица «*Official*» (таблица 2.22) содержит информацию о субъекте сделавшим запрос по данному объекту недвижимости.

Таблица 2.9 – Описание полей таблицы «*Country*»

Имя поля	Тип	Примечание
id	*int*	Первичный ключ, обязательный
country_name	*varchar*(128)	Да
postalcode	*varchar*	Да

Таблица 2.10 – Описание полей таблицы «*City*»

Имя поля	Тип	Примечание
id	*int*	Первичный ключ, обязательный
city_name	*varchar*(128)	Да
country_id	*int*	Вторичный ключ, обязательный

Таблица 2.11 – Описание полей таблицы «*Estate*»

Имя поля	Тип	Примечание
id	*int*	Первичный ключ, обязательный
estate_name	*varchar*(255)	Да
city_id	*int*	Внешний ключ, обязательный
estate_type_id	*int*	Внешний ключ, обязательный
space	*float*	Да
historical_place	*boolen*	Да
number_of_rooms	*int*	Да
number_of_garages	*int*	Да
number_of_parking_ spaces	*int*	Да
estate_description	*text*	Да
estate_status_id	*int*	Внешний ключ, обязательный
okato	*int*	Да
oktmo	*int*	Да
stoimost_estate	*int*	Да
fee_amount	*float*	Да
kadastr_num	*int*	Да

Таблица 2.12 – Описание полей таблицы «*estate_type*»

Имя поля	Тип	Примечание
id	*int*	Первичный ключ, обязательный
type_name	*varchar*(128)	Да

Таблица 2.13 – Описание полей таблицы «*estate_status*»

Имя поля	Тип	Примечание
id	*int*	Первичный ключ, обязательный
estate_status_name	*varchar*(64)	Да

Таблица 2.14 – Описание полей таблицы «*in_charge*»

Имя поля	Тип	Примечание
id	*int*	Первичный ключ, обязательный
estade_id	*varchar*(64)	Внешний ключ, обязательный
employee_id	*int*	Да
date_from	*data*	Да
date_to	*data*	Да

Таблица 2.15 – Описание полей таблицы «*employee*»

Имя поля	Тип	Примечание
id	*int*	Первичный ключ, обязательный
first_name	*varchar*(64)	Да
middle_name	*varchar*(64)	Да
last_name	*varchar*(64)	Да

Таблица 2.16 – Описание полей таблицы «*employee_list*»

Имя поля	Тип	Примечание
id	*int*	Первичный ключ, обязательный
position_employee	*varchar*(64)	Да
status	*boolean*	Да

Таблица 2.17 – Описание полей таблицы «*request*»

Имя поля	Тип	Примечание
id	*int*	Первичный ключ, обязательный
official_id	*int*	Внешний ключ, обязательный
employee_id	*int*	Внешний ключ, обязательный
request_type_id		Внешний ключ, обязательный
request_details	*text*	Да
request_frequency_id	*int*	Да
number_of_invoices	*int*	Да
date_signed	*date*	Да
start_date	*date*	Да
end_date	*date*	Да
transaction_id	*int*	Да

Таблица 2.18 – Описание полей таблицы «*under_request*»

Имя поля	Тип	Примечание
id	*int*	Первичный ключ, обязательный
estate_id	*int*	Внешний ключ, обязательный
request_id	*int*	Внешний ключ, обязательный

Таблица 2.19 – Описание полей таблицы «*request_type*»

Имя поля	Тип	Примечание
id	*int*	Первичный ключ, обязательный
contract_type_name	*varchar*(64)	Да

Таблица 2.20 – Описание полей таблицы «*request_frequency*»

Имя поля	Тип	Примечание
id	*int*	Первичный ключ, обязательный
request_frequency_name	*varchar*(64)	Да

Таблица 2.21 – Описание полей таблицы «*request_invoice*»

Имя поля	Тип	Примечание
id	*int*	Первичный ключ, обязательный
request_id	*int*	Внешний ключ, обязательный
invoice_number	*varchar*(64)	Да
issued_by	*text*	Да
issued_to	*text*	Да
request_details	*text*	Да
link_file	*blob*	Нет
date_created	*date*	Да
send_date	*date*	Да

Таблица 2.22 – Описание полей таблицы «*official*»

Имя поля	Тип	Примечание
id	*int*	Первичный ключ, обязательный
official_name	*varchar*(255)	Да
official_address	*varchar*(255)	Да
contact_person	*varchar*(255)	Да
phone	*varchar*(64)	Да
mobile	*varchar*(64)	Да
mail	*varchar*(64)	Да
official_details	*text*	Да

Таблица 2.23– Описание полей таблицы «*settings*»

Имя поля	Тип	Примечание
id	*int*	Первичный ключ, обязательный
hesh	*int*	Да
num_network	*int*	Да
ip_fgis	*int*	Да

В результате проведенного в данном разделе проектирования логической и физической модели БД, была реализована схема базы данных (рисунок 2.2) необходимая для проектирования и реализации интерфейса и серверной части приложения.

2.2 Проектирование интерфейса приложения

Дизайн пользовательского интерфейса - это процесс проектирования того, как эти интерфейсы выглядят и ведут себя. Пользовательский интерфейс состоит из следующих элементов:

- элементы управления вводом: флажки, кнопки, текстовые поля и выпадающие списки;

- навигационные элементы: поля поиска, ползунки и меню для гамбургеров;

- информационные компоненты: окна сообщений, уведомления и индикаторы выполнения.

- контейнеры - используются для группировки содержимого в значимые разделы.

Дизайн пользовательского интерфейса учитывает выше представленные элементы и то, как они работают вместе, для создания интерфейсов, которые одновременно удобны в навигации и визуально приятны.

Дизайн пользовательского интерфейса охватывает:

- интерактивность - как ведет себя и функционирует пользовательский интерфейс и его различные элементы;

- визуальный дизайн - как выглядит интерфейс с учетом таких факторов, как цвет, графика, логотипы, дизайн значков и интервалы между ними;

- информационная архитектура - как организовано и маркировано содержимое в пользовательском интерфейсе.

Исходя из требований, представленных в данном разделе, ниже представлена структура клиентской части приложения (рисунки 2.3 – 2.5)

На рисунке 2.3 показана структура сайта доступных пользователю «Администратор» страниц и функций.

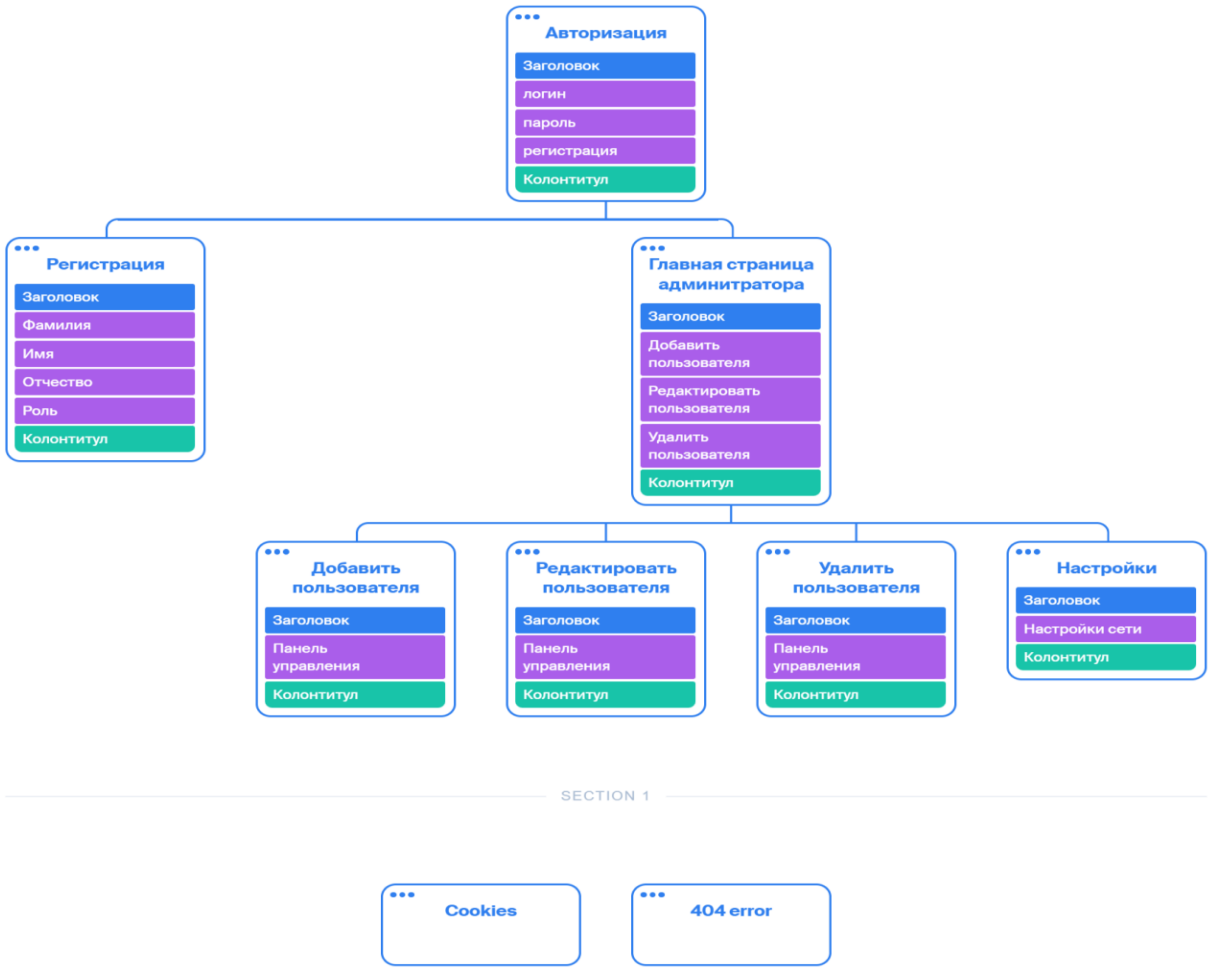


Рисунок 2.3 – Структура страниц для роли «Администратор»

На первой странице пользователю предлагается пройти авторизацию или зарегистрироваться. Если пользователь выбрал регистрацию, ему потребуется заполнить поля идентификации, после чего пользователю потребуется ожидать подтверждения регистрации от службы безопасности учреждения, которая проверит пользователя и активизирует его роль. После чего на

корпоративную почту пользователя придет уведомление о его доступе в систему или отказ с формулировкой дальнейших действий.

После авторизации администратор входит на главную страницу приложения и получает доступ к управлению пользователями. В соответствии с необходимостью администратор может перейти на страницы: добавить пользователя, редактировать пользователя, удалить пользователя. Также администратор отвечает за установку настроек доступа приложения в ФГИС ЕГРН: номер сети ФГИС ЕГРН и рабочий адрес доступа в ФГИС.

На рисунке 2.4 показана структура сайта доступных пользователю «Специалист по недвижимости» страниц и функций.

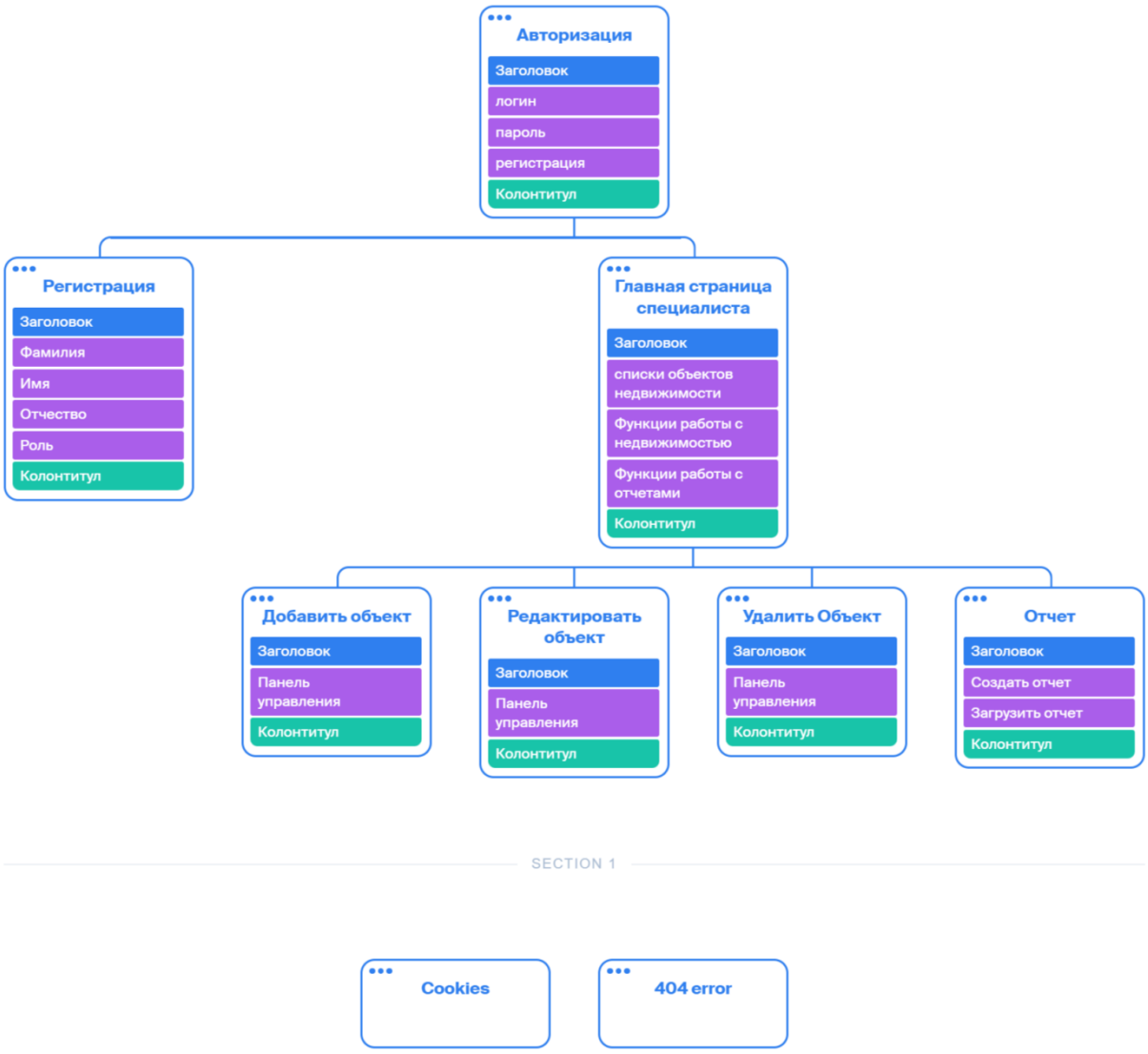


Рисунок 2.4 – Структура страниц для роли «Специалист по недвижимости»

Регистрация пользователя проходит аналогично вышеописанного пользователя «Администратор». Авторизация дает право пользователю войти на главную страницу и получить доступ к следующим функциям: просмотреть список недвижимости принадлежащей учреждению, возможность добавить, редактировать, удалить объект недвижимости из списка доступный, посмотреть отчеты и запросы, а также сформировать отчет об объекте недвижимости.

На рисунке 2.5 показана структура сайта доступных пользователю «Оператор» страниц и функций.

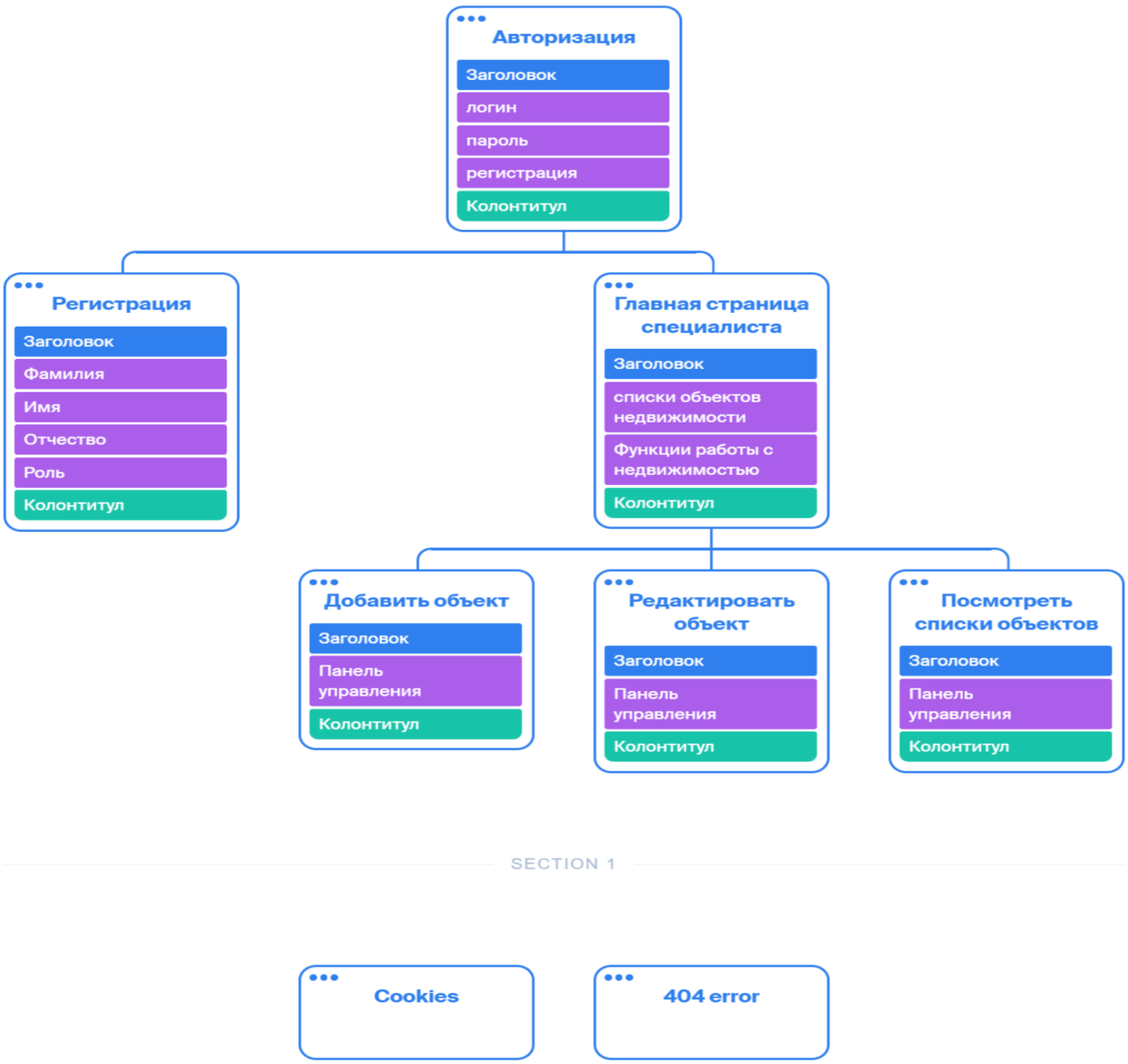


Рисунок 2.5 – Структура страниц для роли «Оператор»

Регистрация пользователя проходит аналогично вышеописанного пользователя «Администратор». Авторизация дает право пользователю «Оператор» войти на главную страницу и получить доступ к следующим функциям: просмотреть список недвижимости принадлежащей учреждению, возможность добавить, редактировать объекты недвижимости учреждения.

После детального описания структуры сайта были разработаны макеты страниц, представленные на рисунках 2.6 – 2.21.

На рисунках 2.8 – 2.12 представлены макеты страниц работы в режиме «Администратор».

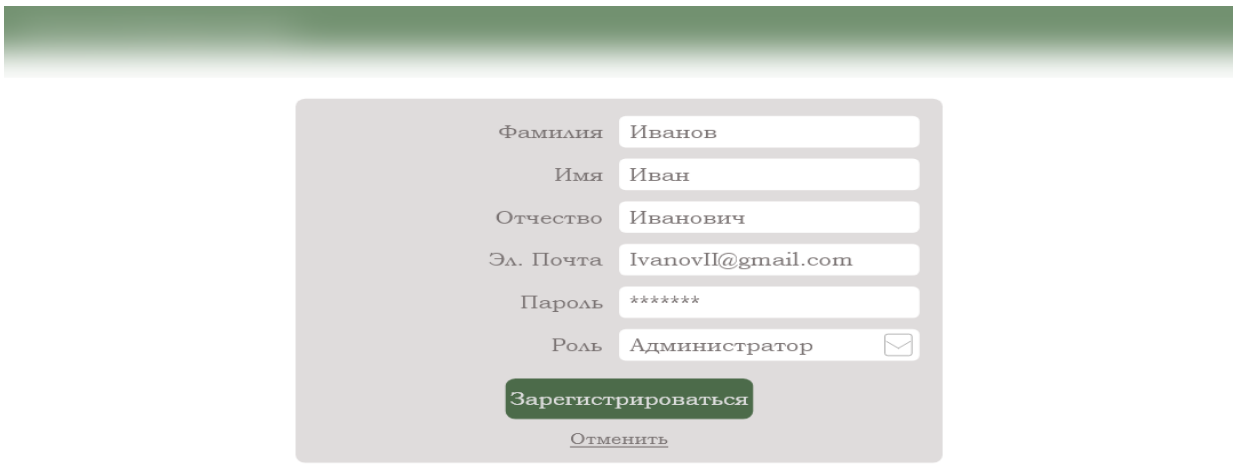


Рисунок 2.6 – Макет страницы регистрации

Элементами страницы регистрации являются (рисунок 2.6):

– форма регистрации – тип *Form*;

– поле ввода имени - тип *Inputtype*;

– поле ввода отчества - тип *Inputtype*;

– поле ввода фамилии - тип *Inputtype*;

– поле ввода логина - тип *Inputtype*;

– поле ввода пароля - тип *Inputtype*;

– поле ввода роли пользователя - тип *Inputtype*;

– кнопка регистрации - тип *Button*;

– ссылка на форму авторизации - тип *href*.

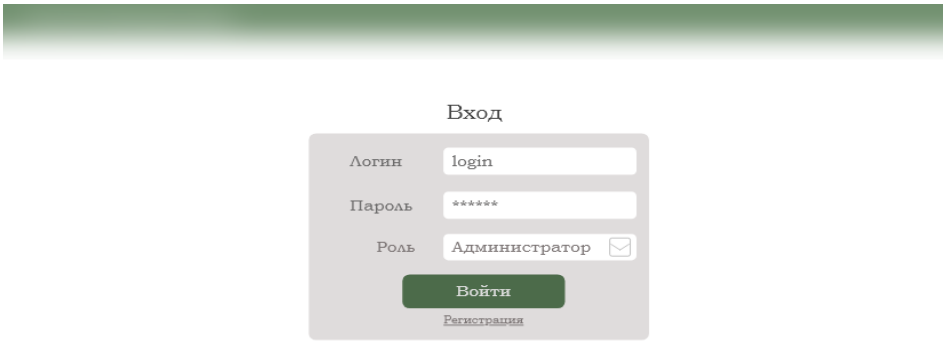


Рисунок 2.7 – Макет страницы авторизации

Элементами страницы авторизации являются (рисунок 2.7):

– форма регистрации – тип *Form*;

– поле ввода логина - тип *Inputtype*;

– поле ввода пароля - тип *Inputtype*;

– поле ввода роли пользователя - тип *Inputtype*;

– кнопка сброса - тип *Button*;

– ссылка на форму регистрации – тип *href*.

Рисунок 2.8 – Макет главной страницы администратора

Элементами главной страницы администратора являются (рисунок 2.8):

– форма страницы – тип *Form*;

– счетчик зарегестрированных пользователей – тип *label*;

– счетчик неподтвержденных пользователей – тип *label*;

– ссылки навигации - тип h*ref*;

– таблица с информацией о пользователях – тип *table*;

– навигация – тип *href*.

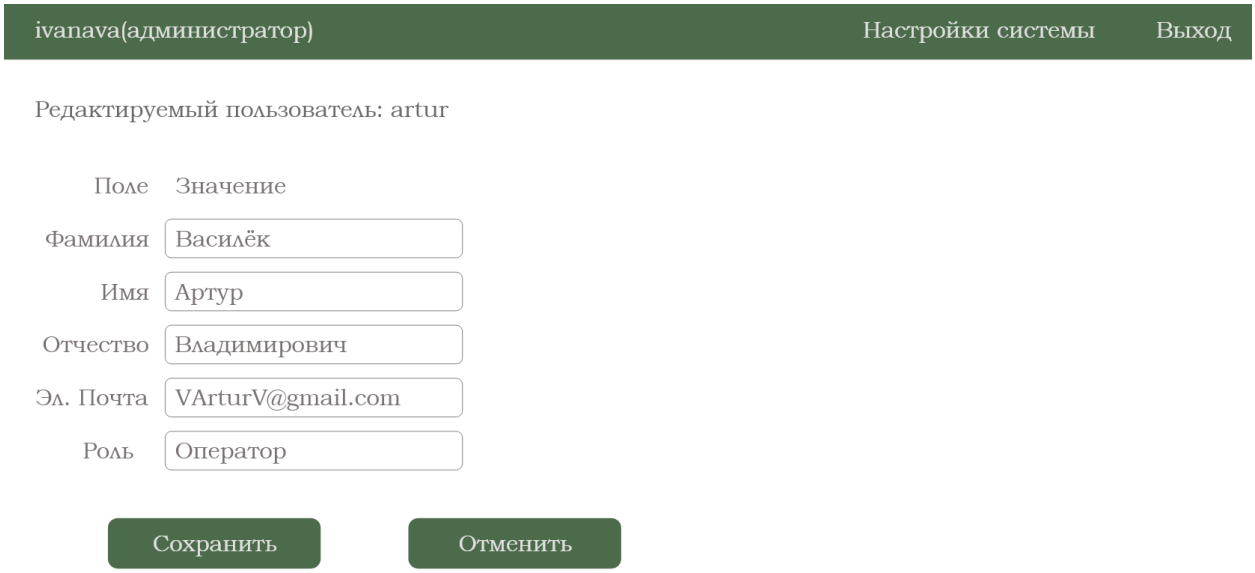


Рисунок 2.9 – Макет страницы редактировать пользователя (администратор)

Элементами страницы редактирования информации о пользователе являются (рисунок 2.9):

– форма страницы – тип *Form*;

– информационное поле – тип *label*;

– ссылки навигации - тип *href*;

– таблица с информацией о пользователе в режиме редактора – тип *table*;

– навигация – тип *href*.

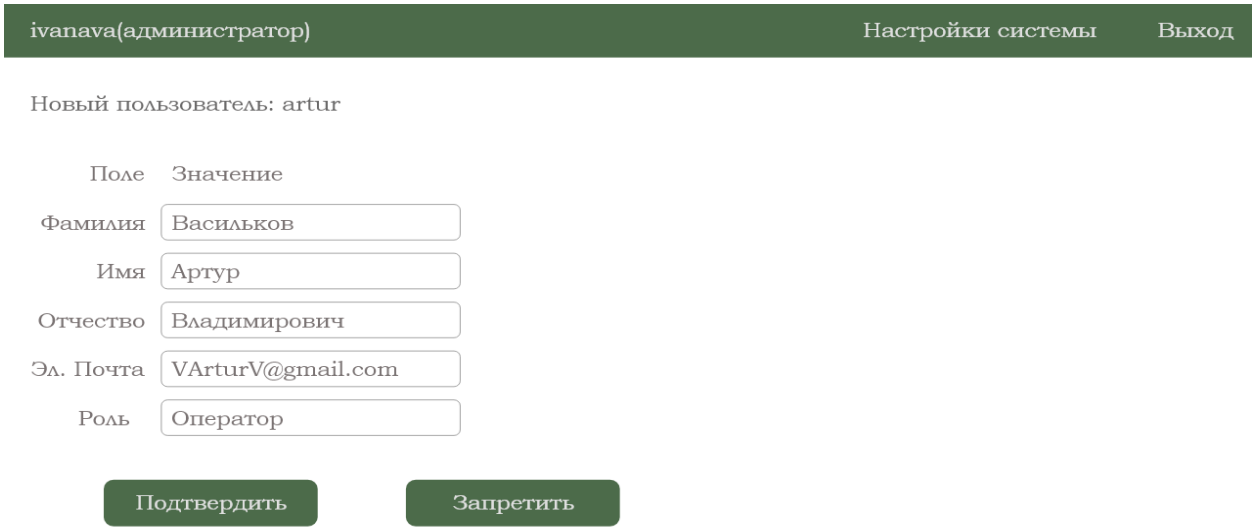


Рисунок 2.10 – Макет страницы подтверждение пользователя (администратор)

Элементами страницы «Новый пользователь»являются (рисунок 2.10):

– форма страницы – тип *Form*;

– информационное поле – тип *label*;

– ссылки навигации - тип *href*;

– таблица с информацией о пользователе в режиме чтения – тип *table*;

– навигация – тип *href*.

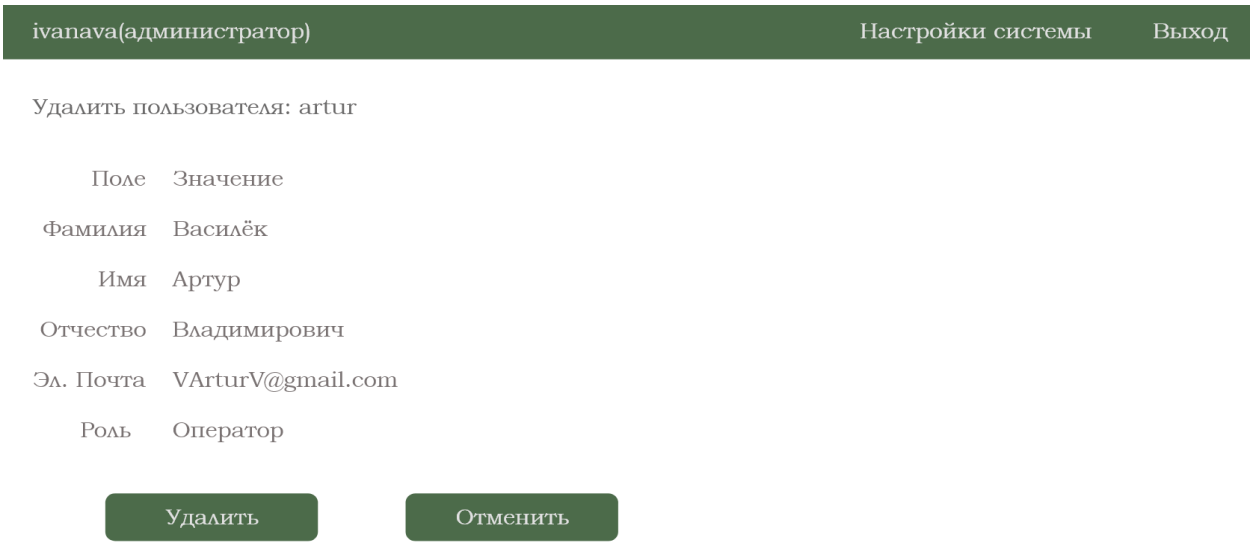


Рисунок 2.11 – Макет страницы удаление пользователя (администратор)

Элементами страницы «Удаление пользователя» являются (рисунок 2.11):

– форма страницы – тип *Form*;

– информационное поле – тип *label*;

– ссылки навигации - тип *href*;

– таблица с информацией об удаляемом пользователе в режиме чтения – тип *table*;

– навигация – тип *Button*;

– навигация – тип *href*.

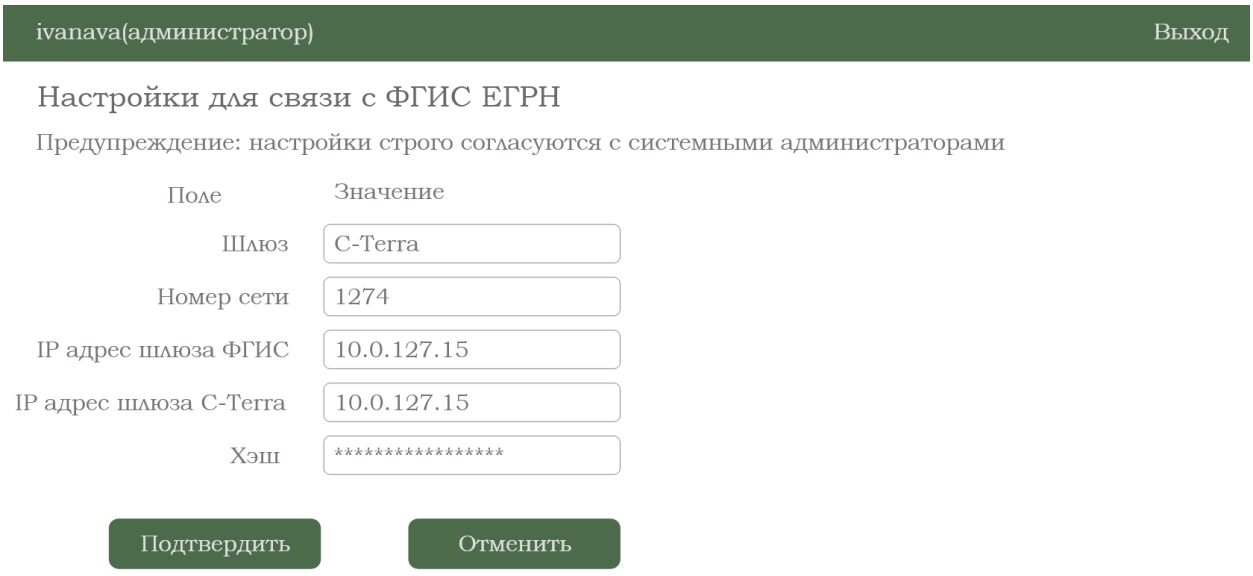


Рисунок 2.12 – Макет страницы настроек системы (администратор)

Элементами страницы «Настройки связи» являются (рисунок 2.12):

– форма страницы – тип *Form*;

– информационное поле – тип *label*;

– ссылки навигации - тип *href*;

– таблица с информацией о сети для подключения к ФГИС ЕГРН в режиме чтения – тип *table*;

– навигация – тип *Button*;

– навигация – тип *href*.

На рисунках 2.13 – 2.21 представлены макеты страниц работы в режиме «Специалист по недвижимости».

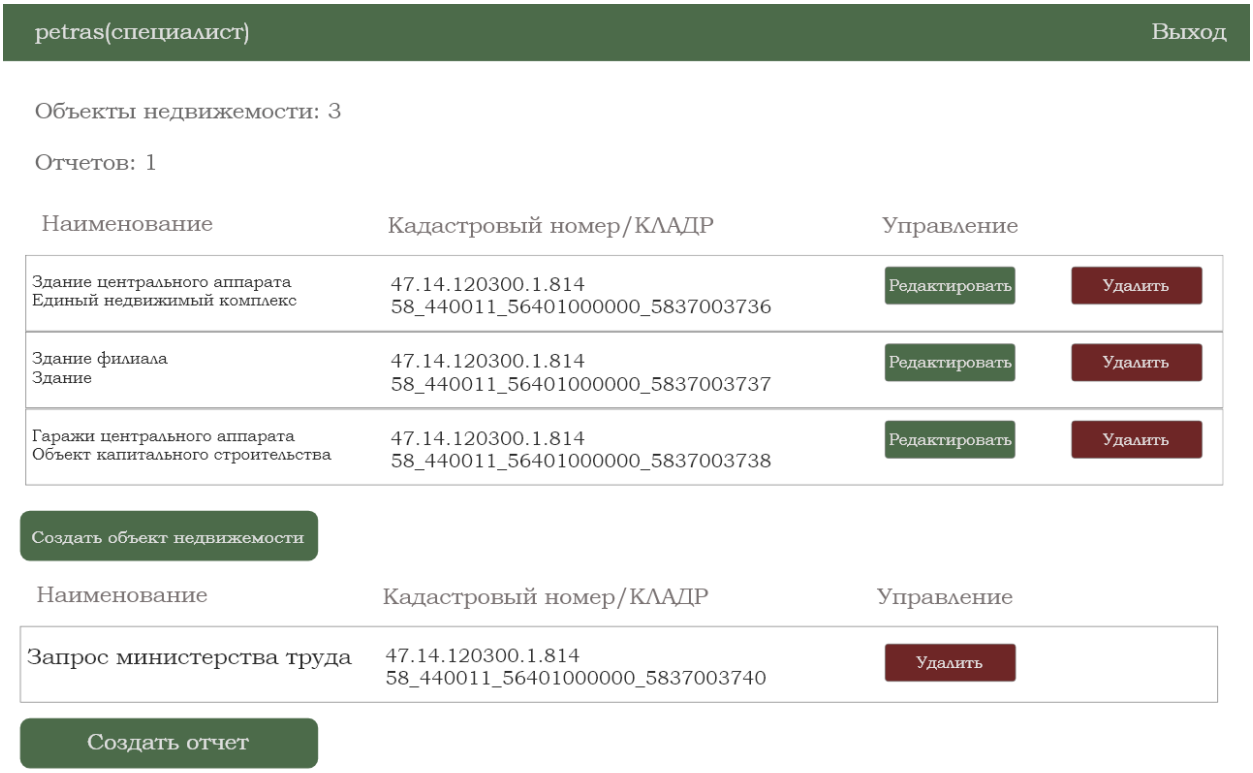


Рисунок 2.13 – Макет главной страницы в режиме специалист

Элементами главной страницы в режиме «Специалист по недвижимости» являются (рисунок 2.13):

– форма страницы – тип *Form*;

– информационное поле – тип *label*;

– ссылки навигации - тип *href*;

– таблица с информацией об объектах недвижимости – тип *table*;

– таблица с информацией об отчетах – тип *table*;

– навигация – тип *Button*;

– навигация – тип *href*.

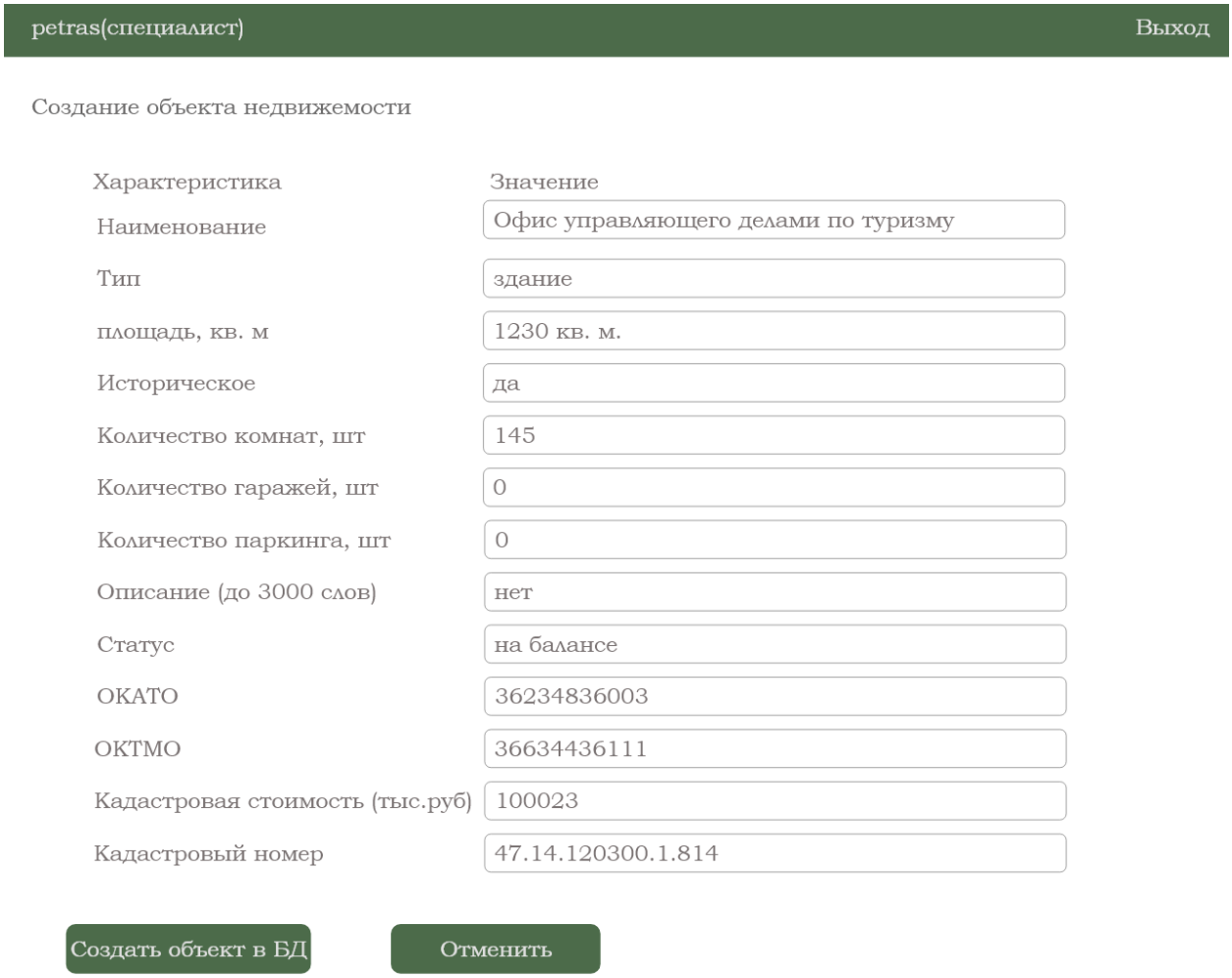


Рисунок 2.14 – Макет страницы «Создать объект недвижимости» в режиме специалист

Элементами страницы «Создать объект недвижимости» являются (рисунок 2.14):

– форма страницы – тип *Form*;

– информационное поле – тип *label*;

– ссылки навигации - тип *href*;

– таблица с информацией об объектах недвижимости (с обязательным полем ввода для кадастрового номера) – тип *table*;

– навигация – тип *Button*;

– навигация – тип *href*.

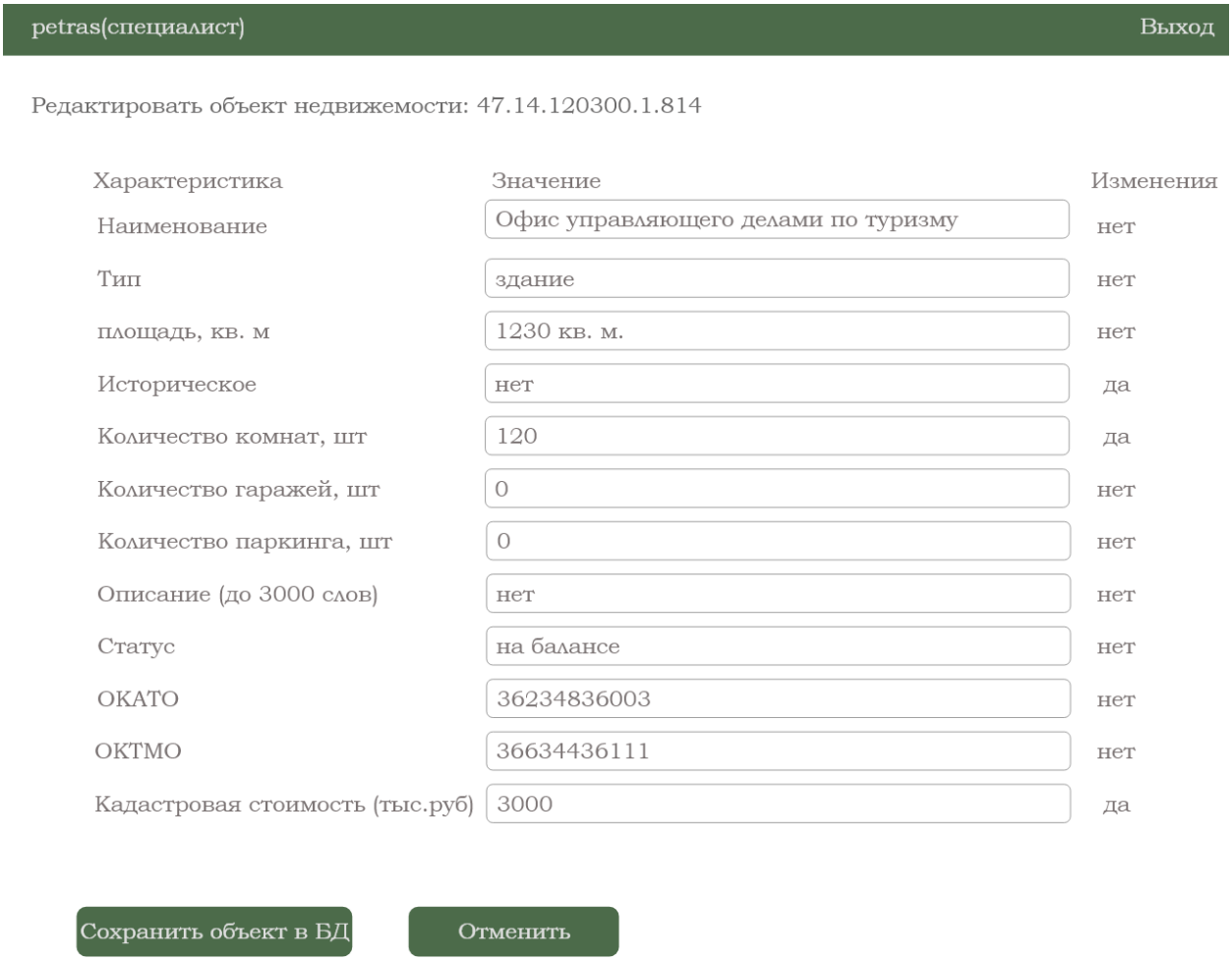


Рисунок 2.15 – Макет страницы «Редактировать объект недвижимости» в режиме специалист

Элементами страницы «Редактировать объект недвижимости» являются (рисунок 2.15):

– форма страницы – тип *Form*;

– информационное поле – тип *label*;

– ссылки навигации - тип *href*;

– таблица с информацией об объектах недвижимости – тип *table*;

– навигация – тип *Button*;

– навигация – тип *href*.

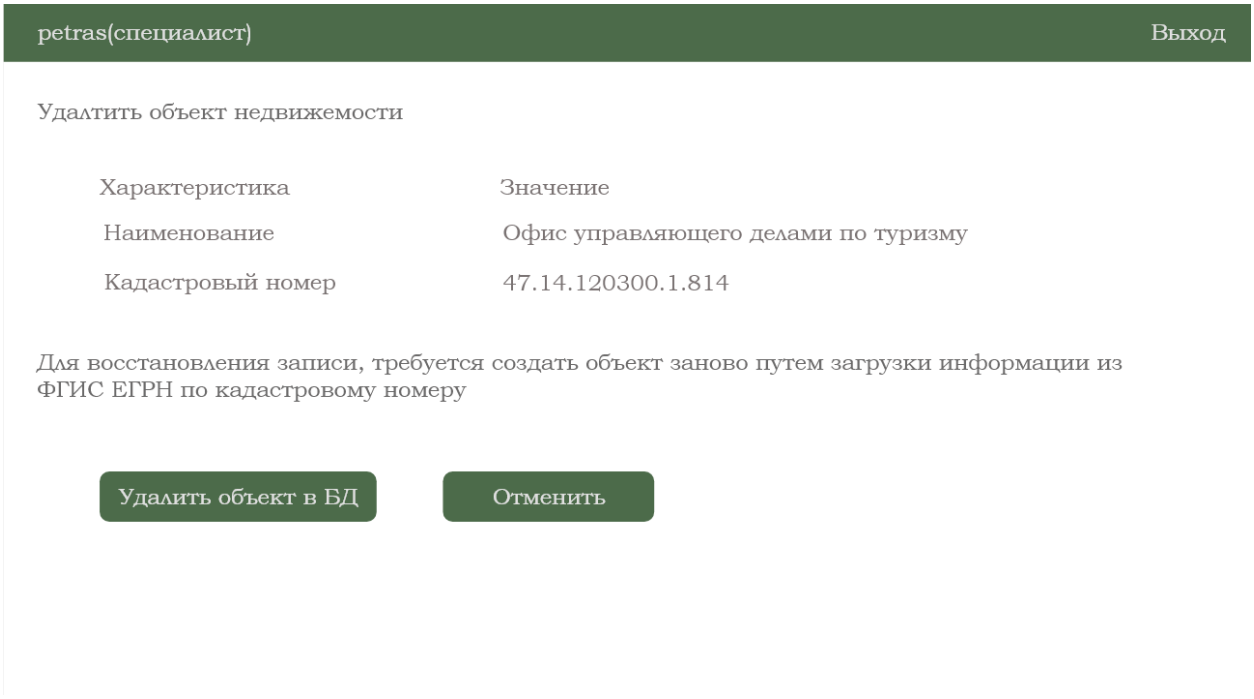


Рисунок 2.16 – Макет страницы «Удалить объект недвижимости» в режиме специалист

Элементами страницы «Удалить объект недвижимости» являются (рисунок 2.16):

– форма страницы – тип *Form*;

– информационное поле – тип *label*;

– ссылки навигации - тип *href*;

– таблица с информацией об удаляемом объекте недвижимости – тип *table*;

– навигация - тип *Button*.

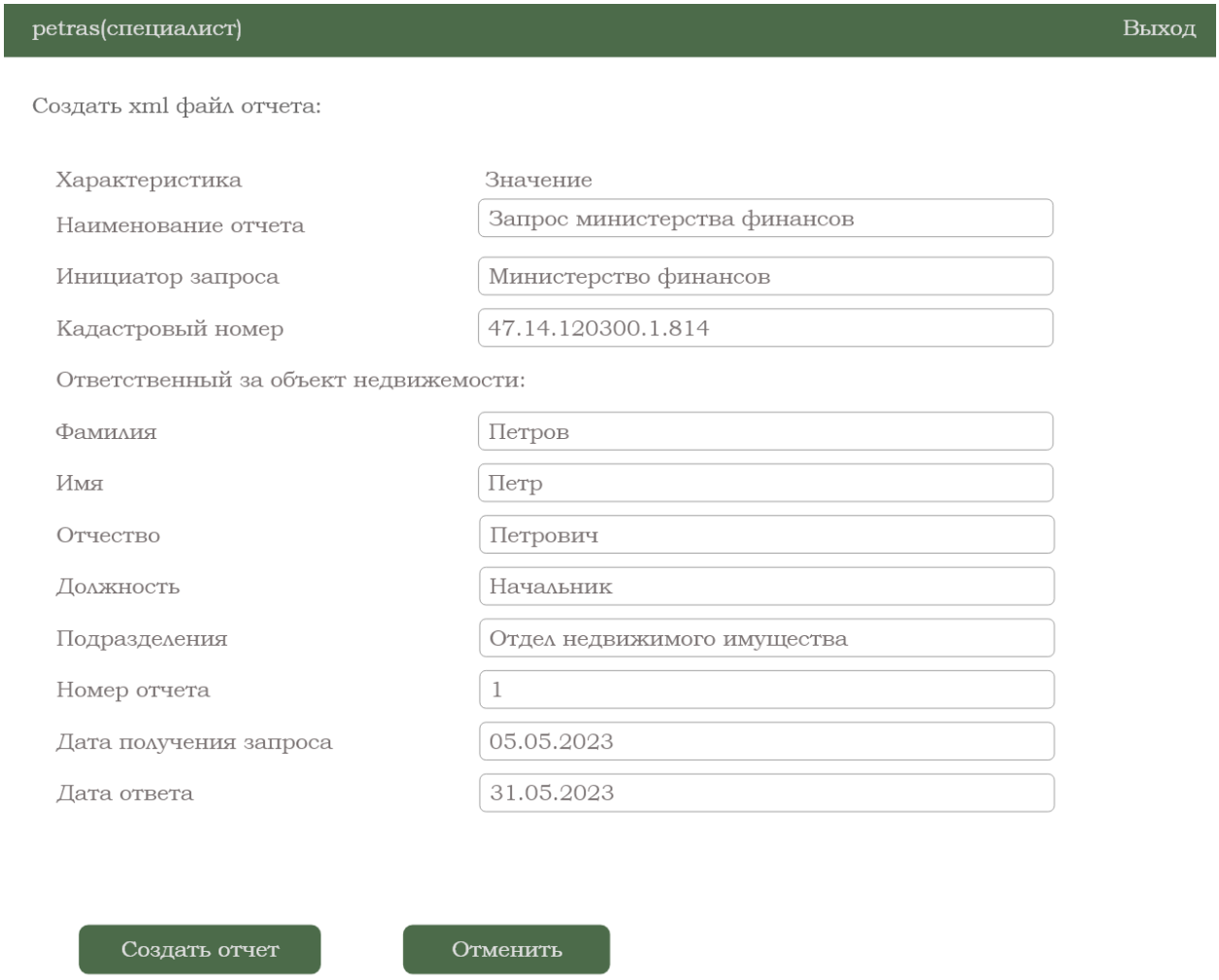


Рисунок 2.17 – Макет страницы «Создать отчет» в режиме специалист

Элементами страницы «Создать отчет»являются (рисунок 2.17):

– форма страницы – тип *Form*;

– информационное поле – тип *label*;

– ссылки навигации - тип *href*;

– таблица с информацией об отчете – тип *table*;

– навигация – тип *Button*.

На рисунках 2.18 – 35 представлены макеты страниц работы в режиме «Оператор».

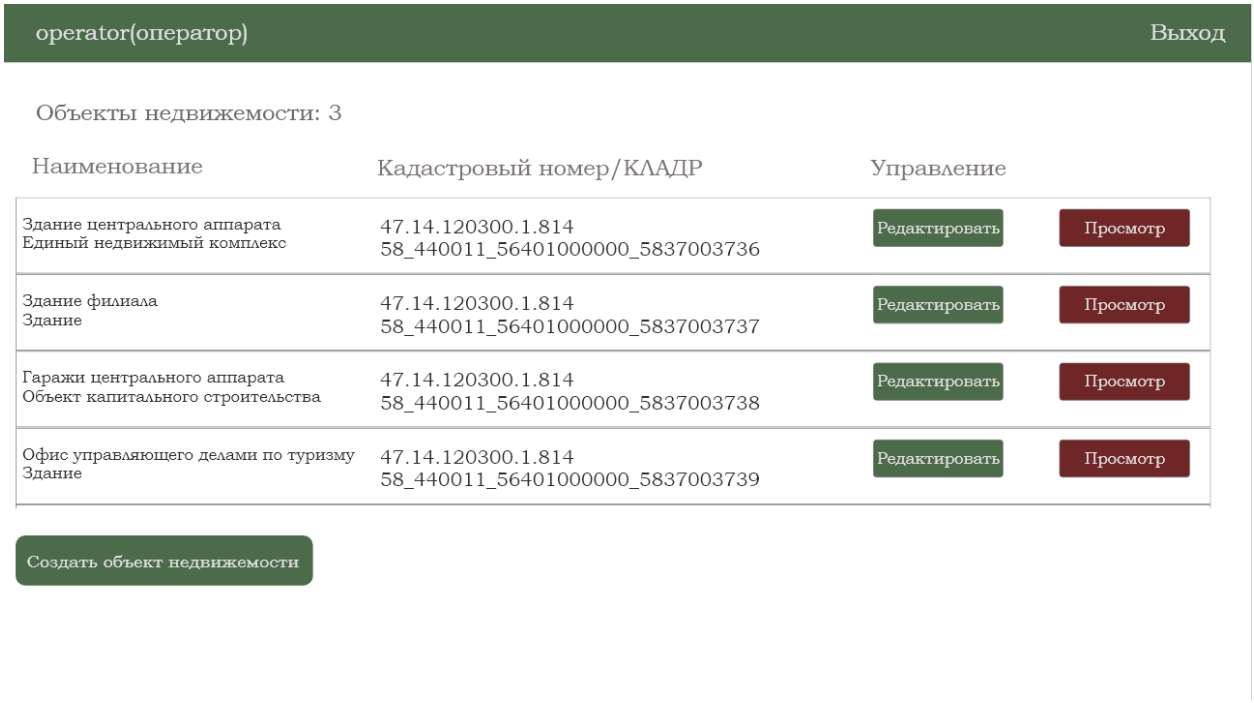


Рисунок 2.18 – Макет главной страницы в режиме оператор

Элементами главной страницы «Оператор» являются (рисунок 2.18):

– форма страницы – тип *Form*;

– информационное поле – тип *label*;

– ссылки навигации - тип *href*;

– таблица с информацией об объектах недвижимости – тип *table*;

– навигация - тип *Button*.

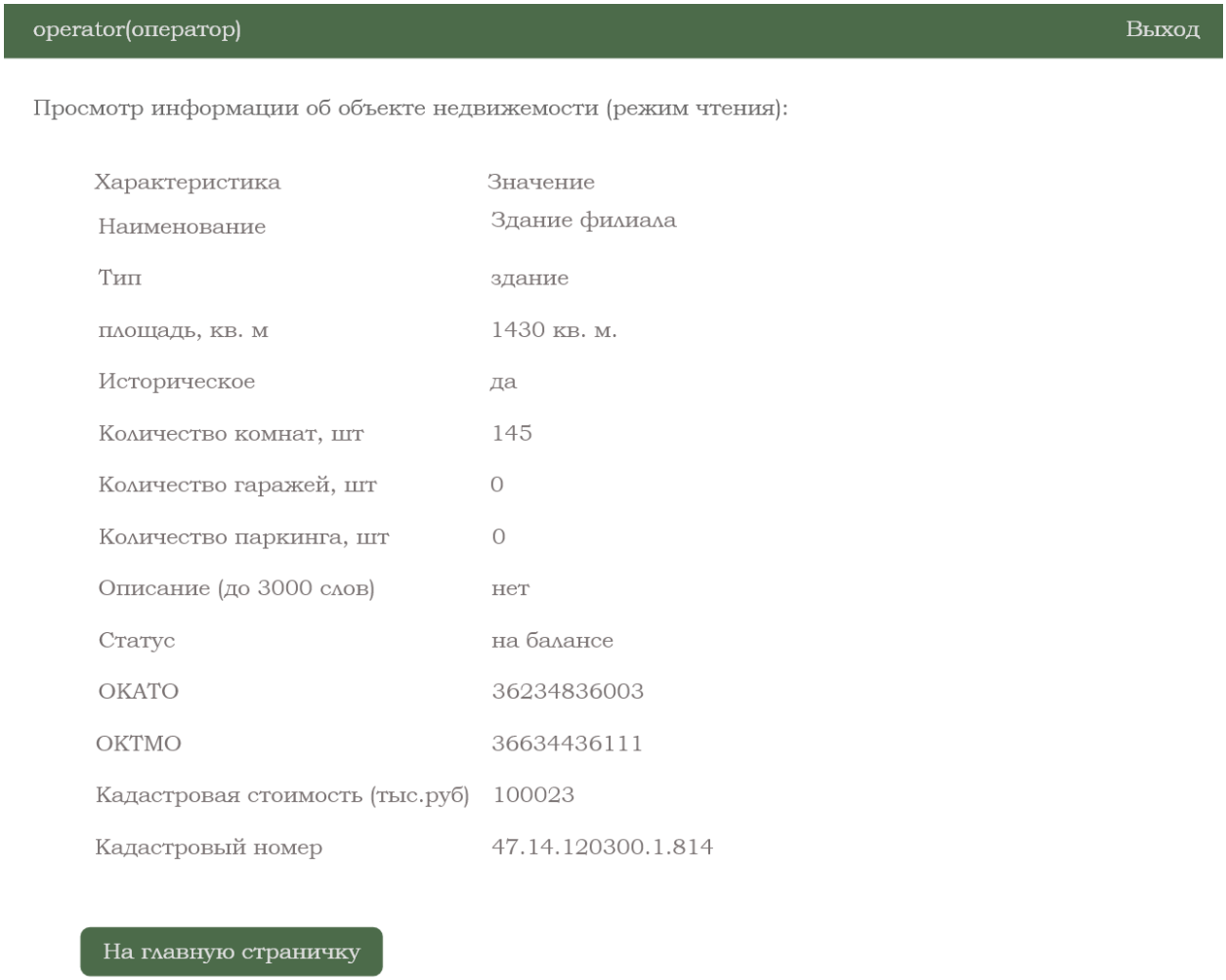


Рисунок 2.19 – Макет страницы просмотра в режиме оператор

Элементами главной страницы «Просмотр информации…»являются (рисунок 2.19):

– форма страницы – тип *Form*;

– информационное поле – тип *label*;

– ссылки навигации - тип *href*;

– таблица с информацией о выбранном объекте недвижимости (в режиме чтения) – тип *table*;

– навигация – тип *Button*.

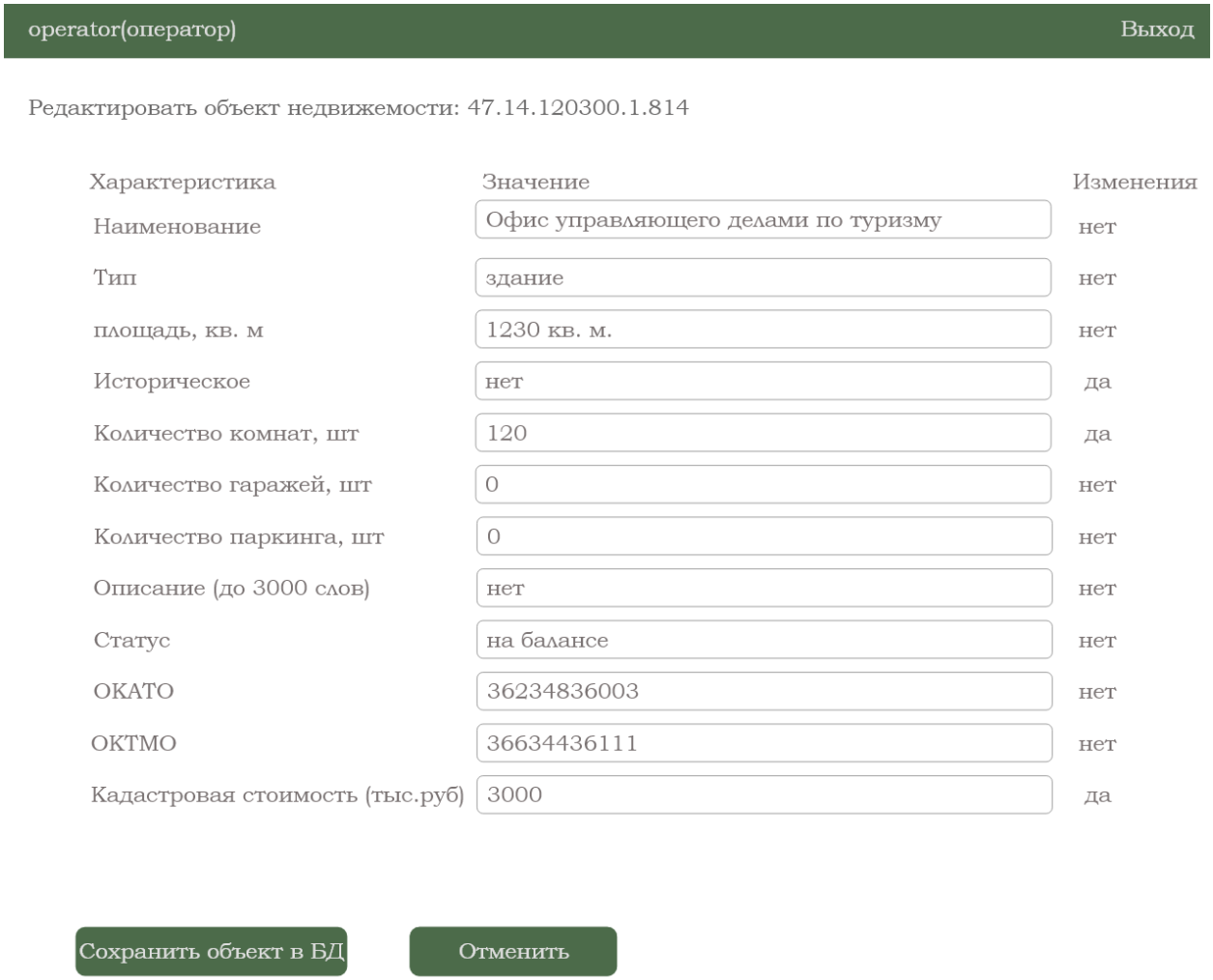


Рисунок 2.20 – Макет главной страницы в режиме специалист

Элементами главной страницы «Редактировать…»являются (рисунок 2.20):

– форма страницы – тип *Form*;

– информационное поле – тип *label*;

– ссылки навигации - тип *href*;

– таблица с информацией о выбранном объекте недвижимости (в режиме редактирования) – тип *table*;

– навигация – тип *Button*.

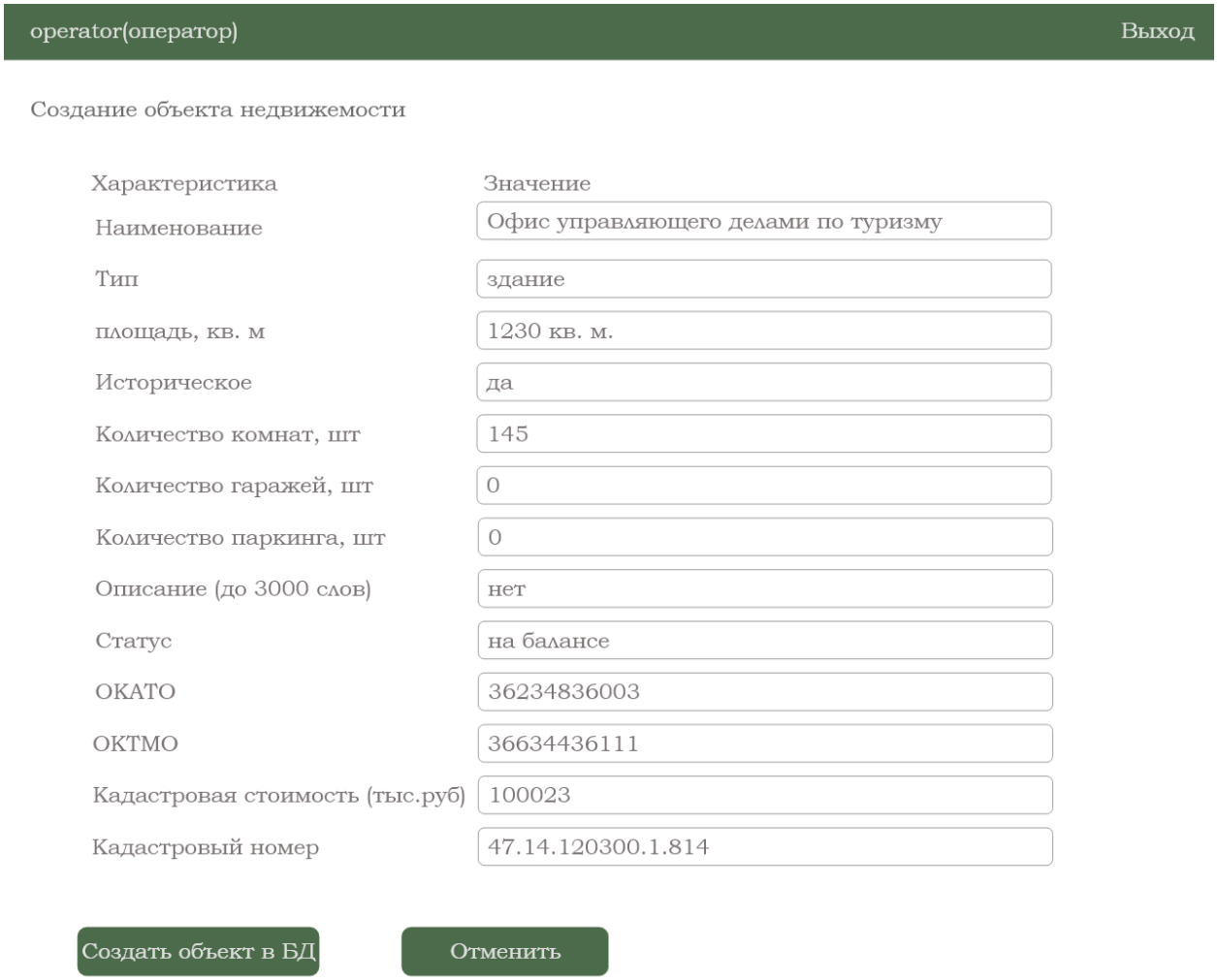


Рисунок 2.21 – Макет главной страницы в режиме специалист

В результате проделанной работы был сделан проект, и реализация пользовательского интерфейса клиентской части приложения.

2.3 Проектирование серверной части приложения

Серверная часть приложения реализована с помощью фреймворка *Laravel*. Фреймворк *Laravel* необходим для реализации приложения в рамках паттерна *MVC*.

Для реализации и развертывания приложения вконсоли среды разработки требуется создать каталог проекта с файлами шаблонов и настроек приложения, для этого необходима утилита *composer* и следующая команда: `composercreate-projectlaravel/laravelphp-laravel-project-crud`.

Вторым шагом реализации приложения будет работа с настройками проекта. В корне проекта `php-laravel-project-crud` содержится файл с расширением*.env*. Это файлы конфигурации приложения для установки имени приложения, домашнего *URL* и иных параметров. В частности в нем хранятся данные о БД, с которой будет работать приложение. Настройки реализованного приложения выглядят следующим образом:

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=php-laravel-crud
```

С помощью утилиты *artisan* была произведена миграция схемы базы данных в созданную БД и экспортированы данные подготовленные с помощью *backup* файла.

```
phpartisanmake:migrationcreatedbphp-laravel-crud
php artisan migrate backup.sqlphp-laravel-crud
```

Далее последовательно требуется реализация интерфейсов, контроллеров и моделей приложения в соответствии с заявленным в техническом задании функционалом.

Для создания моделей требуется утилита artisanи последовательное выполнение следующей команды для создания всех необходимых моделей: `phpartisanmake:modelEstate.`

В данном случае создана модель объекта недвижимости, поля данных которой будут совпадать с полями таблицы созданной базы данных (рисунок 2.2). Команда создает класс модели *Estate* в каталоге *app/Model*. Файл содержит массив заполняемых полей, защищенных от неправильного ввода данных.

```
useIlluminate\Database\Eloquent\Factories\HasFactory;
useIlluminate\Database\Eloquent\Model;
classEstateextendsModel
{
useHasFactory;
    protected $fillable =
```

```
    ['estate_name',          'space',          'historical_place',
'number_of_rooms',                              'number_of_garages',
'number_of_parking_spaces',     'estate_description',     'okato',
'oktmo', 'stoimost_estate', 'fee_amount', 'kadastr_num'];
    }
```

Создание контроллера связанного с созданным классом модели в *Laravel* имеет то же соглашение об именовании, но имя класса должно быть с суффиксом *Controller*. В данном случае это *EstateController.php*. Команда *PHPartisan* для создания контроллера выглядит следующим образом:

```
phpartisanmake:controllerEstateController
```

Все классы *controller* приложения включают в себя функции для выполнения действий по чтению, сохранению, обновлению, удалению данных. Каждая функция связана с соответствующими методами запроса и маршрутным путем. Значение внутри фигурных скобок {} в *URI* является уникальным идентификатором для извлечения записи из БД.

Так для класса *EstateController* реализованы функции показанные в таблице 2.24.

Таблица 2.24 – Описание функций реализуемых контроллером.

Функция	Метод	Описание
index()	*GET*	Перечислите результаты(вывести данные об имеющихся объектах недвижимости)*URI: /estate*
create()	*GET*	Показать форму добавления объекта недвижимости*URI: /estate/create*
store()	*POST*	Добавьте запись о новом объекте недвижимости*URI: /estate /create*
show()	*GET*	Просмотр записи по идентификатору. *URI: /estate /{estateid}*
edit()	*GET*	Переход на форму редактирования выбранного объекта недвижимости *URI: / estate /{ estateid}/edit*
update()	*PUT*	Сохранить измененные данные об определенном объекте недвижимости*URI: / estate /{estateid}*
destroy()	*POST*	Удалить запись о выбранном объекте недвижимости*URI: / estate /{ estateid}*

Приведем примеры описанных в таблице 2.24 функций контроллера работы с объектами недвижимости.

Функция *EstateController* добавления новой записи при получении значения переменной *kadastr_num* идентифицирует объект в ФГИС ЕРГН и позволяет открыть форму заполнения данных о нем.

```
publicfunctionstore(Request $request)
{
$newPost = Estate::create([
'title' =>$request->title,
'short_notes' =>$request->short_notes,
'kadastr_num' =>$request->kadastr_num
        ]);
returnredirect('estate/' . $newPost->id '/edit');
}
```

Функция *EstateController* обновления информации об объекте недвижимости по кадастровому номеру предоставляет форму с заполненными полями данных об объекте с возможностью их редактирования.

```
publicfunctionupdate(Request $request, Estate $product)
{
$product->update([
'title' =>$request->title,
'short_notes' =>$request->short_notes,
' kadastr_num ' =>$request->kadastr_num
        ]);
returnredirect('estate/' . $estate->id .'/edit');
    }
```

Функция *EstateController* удаления информации об объекте недвижимости по кадастровому номеру какую запись в базе данных требуется удалить.

```
publicfunctiondestroy(Estate $estate)
{
' kadastr_num ' =>$request->kadastr_num
$estate->delete();
returnredirect('product/');}
```

После реализации функционала контроллеров требуется реализовать функционал слоя Вид. Данный слой реализован на основании стандартного шаблона *Laravelapp.layout*с добавлением дочерних шаблонов расположенных в файлах с расширением *blade.php.* Стандартный шаблон *Laravel* заключает весь дочерний html код реализуемых страниц во вложение *@section-@ensection*. Таким образом, позволяя в одном файле работать с кодом htmlи данными и функциями *php*.В частности на главной странице специалиста по недвижимости (рисунок 2.13) для вывода списка объектов недвижимости и возможных операций с ними используются элементы таблицы и цикл phpforelse:

```
@forelse($estate as $esate)
<tr>
      <td>{!! $estate->title !!}</td>
      <td class="pr-5 text-right">{$estate-
>estate_name}</td>
      <td>{!! $esate->kadastr_num!!}</td>
      <td><a href="estate/{$estate->id !!}/edit"
      class="btnbtn-outline-primary">Редактировать</a>
      <button type="button" class="btn
"onClick='showModel({$estate->id !!})'>Удалить</button></td>
      </tr>
@endforelse
```

Для работы с *API*ФГИС ЕГРН требуется наличие в контроллерах приложения кода настройки на работу с конкретным видом *API*. В данном приложении за настройку отвечает функция *callAPI.*

```
functioncallAPI($method, $url, $data){
$curl = curl_init();
   switch ($method){
      case "POST":
curl_setopt($curl, CURLOPT_POST, 1);
         if ($data)
curl_setopt($curl, CURLOPT_POSTFIELDS, $data);
         break;
```

```
        case "PUT":
       curl_setopt($curl, CURLOPT_CUSTOMREQUEST, "PUT");
                if ($data)
       curl_setopt($curl, CURLOPT_POSTFIELDS, $data);
                break;
             default:
                if ($data)
                   $url = sprintf("%s?%s", $url,
http_build_query($data));
          }
          // OPTIONS:
       curl_setopt($curl, CURLOPT_URL, $url);
       curl_setopt($curl, CURLOPT_HTTPHEADER, array(
            'APIKEY: 12145117811901114311111',
            'portal.orgid: 10.0.100.56',
           'ORGID:1748b89b-7d34-4d47-b479-b06b40ab1a30
        ',
          ));
       curl_setopt($curl, CURLOPT_RETURNTRANSFER, 1);
       curl_setopt($curl, CURLOPT_HTTPAUTH, CURLAUTH_BASIC);
          // EXECUTE:
          $result = curl_exec($curl);
          if(!$result){die("Connection Failure");}
       curl_close($curl);
          return $result;
       }
```

Ключ *API* необходим для связи с ФГИС, которая однозначно определяет по ключу вызовы из доверенной зоны *APIKEY*, идентификационному номеру организации *ORGID*, и по доменному имени или *IP*адресу организации в сети ФГИС ЕГРН.

Данные *APIKEY, ORGID* и *IP* адрес организации в сети ФГИС ЕГРН выдаются по письменному требованию учреждения провайдером ФГИС ЕГРН.

На основании выше описанного пример работы приложения в режиме «Специалист по недвижимости» показан на диаграмме последовательности (рисунок 2.22).

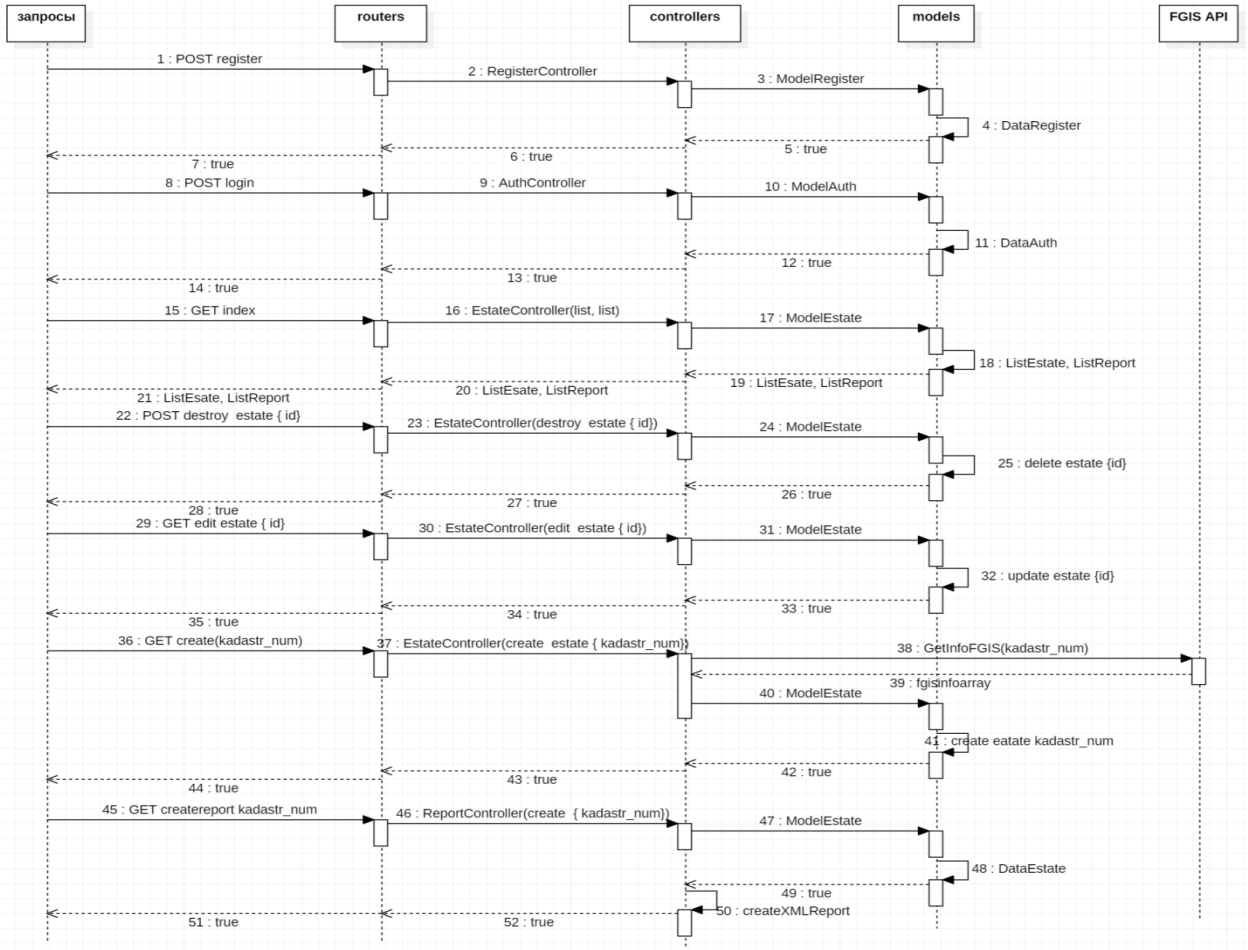


Рисунок 2.22 – Диаграмма последовательности

В данной главе была разработана программа сбора данных об объектах недвижимости: база данных, клиентская и серверные части приложения в соответствии с техническим заданием.

3. Технология настройки VPS сервера для работы WEB приложения

Для работы совеременного *WEB* приложения требуется наличия экосистемы его работы. *VPS* сервера предлагают весь набор требуемого инструментария для работы *WEB* приложения.

Ниже показана последовательность действий для настройки *WEB* сервера *Apache* на базе *VPS* сервера *DigitalOcean* (рисунки 2.23-2.29). Для работы разработанного приложения требуется установить сервер *LAMP*.

Первым шагом для его установки является выбор дистрибутива *LinuxServer* (рисунок 2.24) и затем после его инсталляции настройка *IP* адреса, паролей пользователей и иной конфигурации (рисунок 2.25).

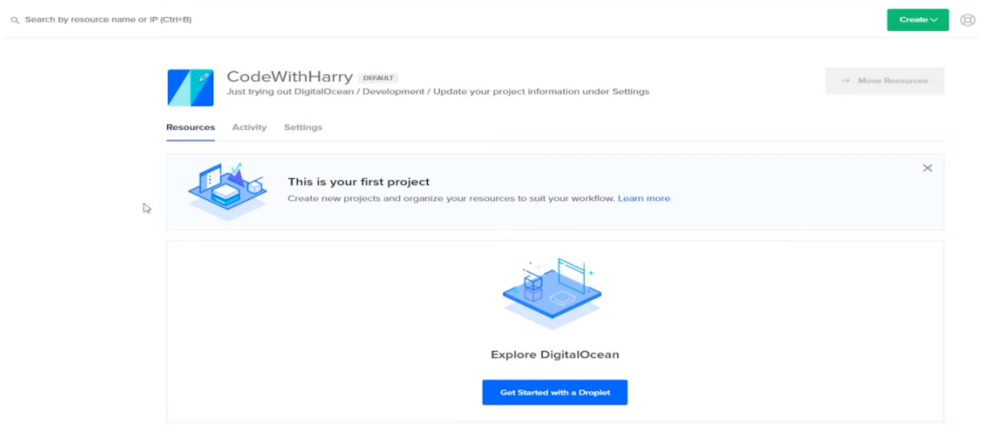


Рисунок 2.23 – Главная станица аккаунта на сайте *DigitalOcean*

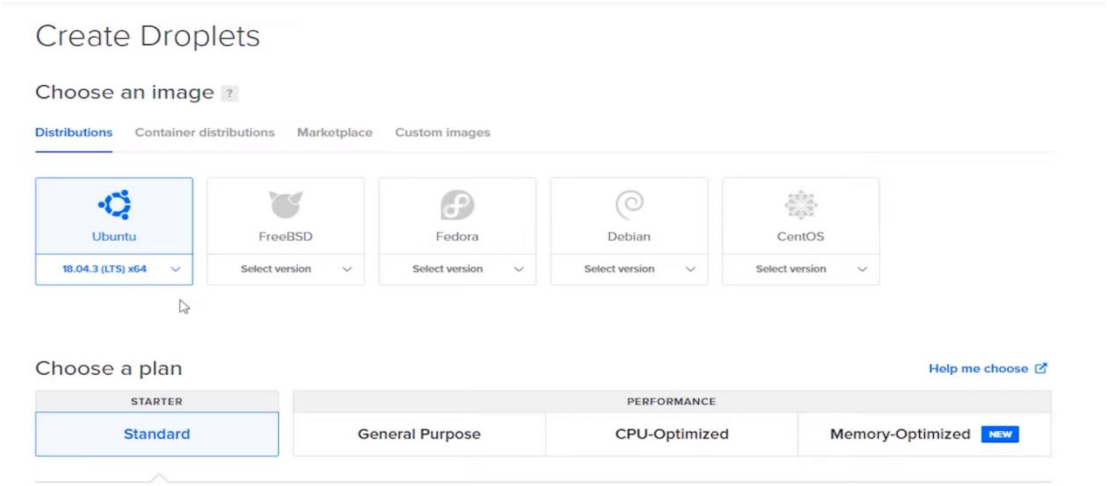


Рисунок 2.24 – Выбор дистрибутива *Linux*сервер для установки

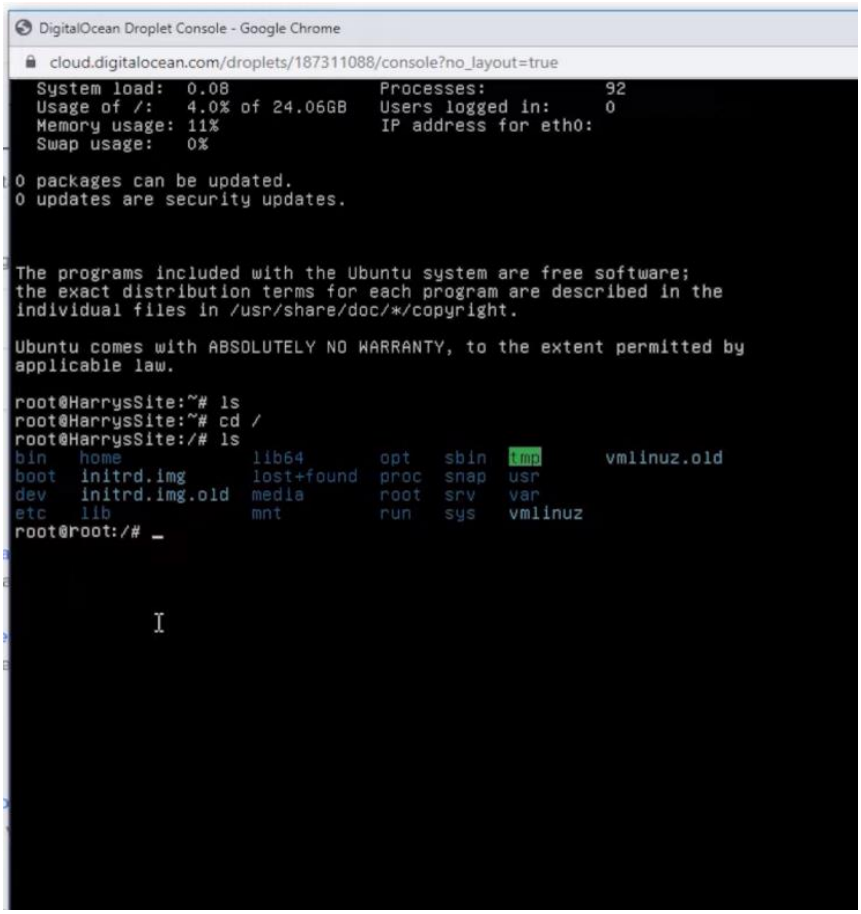


Рисунок 2.25 – Настройка сервера после инсталляции

Для дальнейшей работы с сервером утсновим *SSH* соединение с помощью утилиты *Putty* (рисунок 2.26).

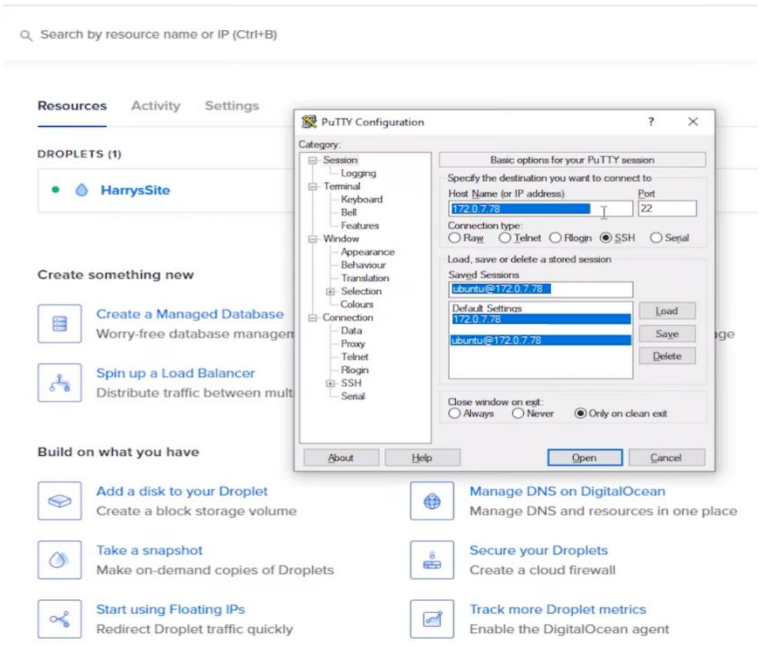


Рисунок 2.26 – Вход на сервер с помощью утилиты *Putty*

После подключения удаленного сеанса требуется установить *WEB* сервер *Apache*. Для чего сначала требуется обновить систему командой *aptupdate* (рисунок 2.27).

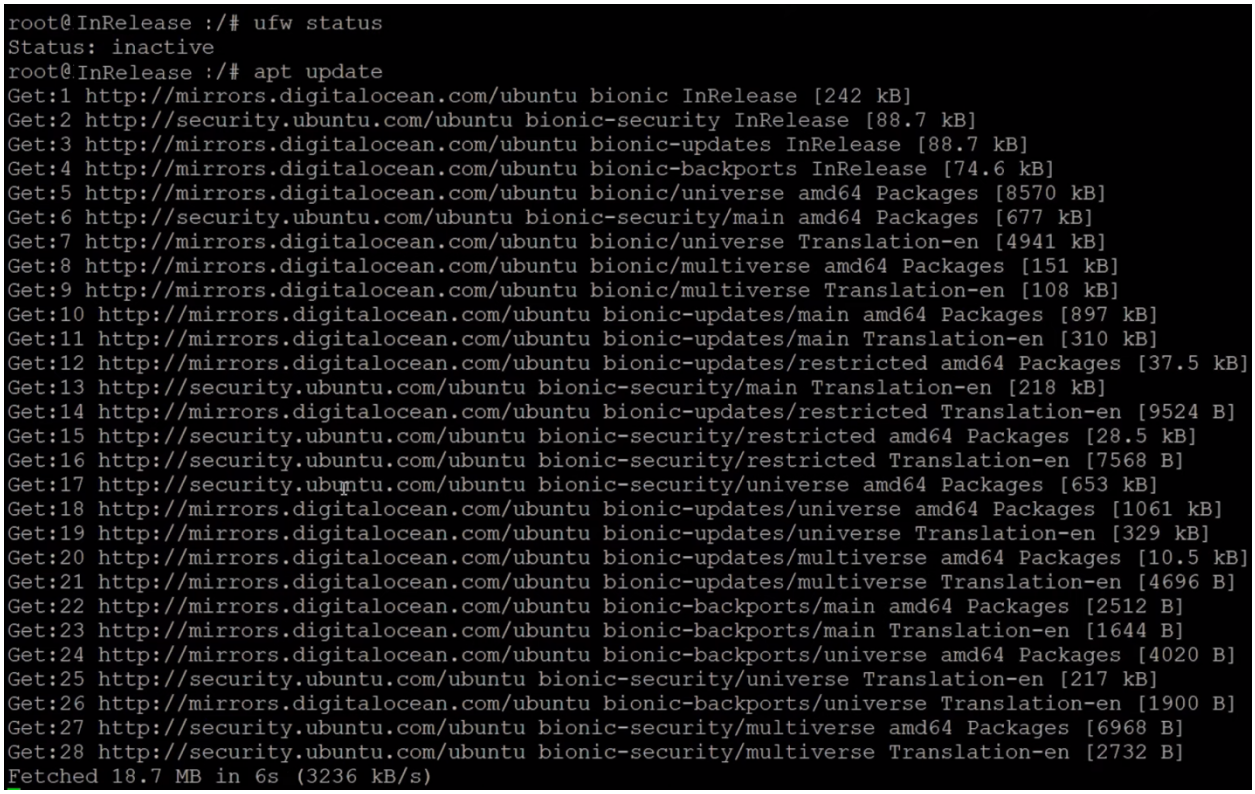


Рисунок 2.27 – Обновление*Linux*до актуального состояния

Затем нужно написать команду *aptinstall apache2***.** После чего начнется процесс установки *HTTP*-сервера *Apache*, бесплатного кроссплатформенного веб-серверного программного обеспечения с открытым исходным кодом. Нужно разрешить входящий трафик, и в результате консоль даст сообщения представленные на рисунке 2.28.

```
Enabling module reqtimeout.
Enabling conf charset.
Enabling conf localized-error-pages.
Enabling conf other-vhosts-access-log.
Enabling conf security.
Enabling conf serve-cgi-bin.
Enabling site 000-default.
Created symlink /etc/systemd/system/multi-user.target.wants/apache2.service → /lib/systemd/system/apache2.service.
Created symlink /etc/systemd/system/multi-user.target.wants/apache-htcacheclean.service → /lib/systemd/system/apache-h
tcacheclean.service.
Processing triggers for libc-bin (2.27-3ubuntu1) ...
Processing triggers for systemd (237-3ubuntu10.31) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
Processing triggers for ufw (0.36-0ubuntu0.18.04.1) ...
Processing triggers for ureadahead (0.100.0-21) ...
root@InRelease:/# sudo ufw app list
Available applications:
  Apache
  Apache Full
  Apache Secure
  OpenSSH
root@InRelease:/# sudo ufw app info "Apache Full"
Profile: Apache Full
Title: Web Server (HTTP,HTTPS)
Description: Apache v2 is the next generation of the omnipresent Apache web
server.

Ports:
  80,443/tcp
root@HarrysSite:/# sudo ufw allow in "Apache Full"
Rules updated
Rules updated (v6)
root@InRelease :/#
```

Рисунок 2.28 – Процесс установки *WEB*сервера *Apache*

Если процесс установки был реализован корректно, то при наборе *IP*адреса выделенного *VPS* провайдером серверу разработки появиться тесовое окно *Apache* (рисунок 2.29)

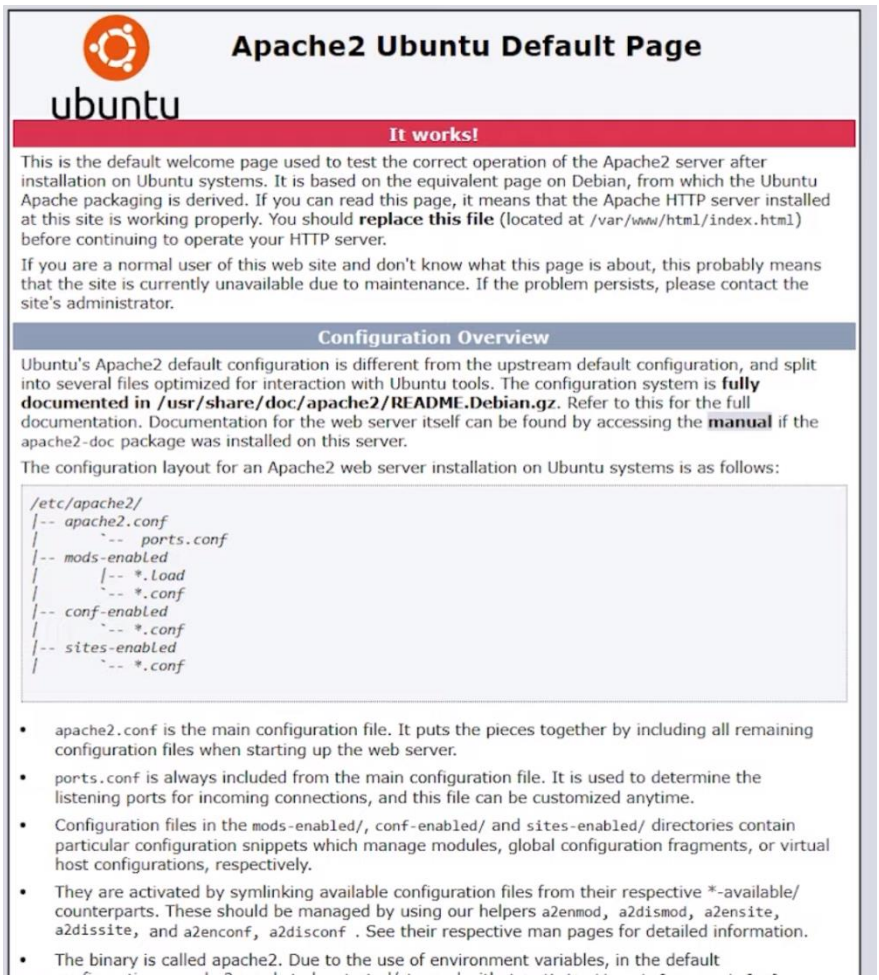


Рисунок 2.29 – Тестовая страница *WEB* сервера *Apache*

Наличие на сервере *Linux*, *MySQL*, *Apache*, *PHP* дает возможность разместить приложение на исполнение. Для использования приложения достаточно наличие браузера и *URL* начальной страницы приложения.

В данной главе была показана технология развертывания на *VPS*сервере *ОС Linux* и *WEB*сервера *Apache* как основу экосистемы *LAMP*.

4. Экономическое обоснование эффективности разработки и внедрения программы сбора данных об объектах недвижимости

Разработка программного обеспечения осуществлена на базе собственной производственной площади общей площадью 60 кв.м. Содержание помещения в месяц составляет 10 000 рублей.

Для разработки программного обеспечения приобретена вычислительная техника, лицензионное программное обеспечение и расходные материалы (обучающие материалы).

Данные по потребности в оборудовании и лицензионном программном обеспечении представлены соответственно в таблице 4.1 и 4.2.

Таблица 4.1 – Потребность в оборудовании

Оборудование	Источник	Кол-во	Ориентировочная цена за единицу, руб	Потребляемая мощность, Ватт
Компьютер *iRU* 525	ООО«ДНС»	5	51000	136
МФУ струйный *HP LaserJet*2750	ООО«ДНС»	1	7200	183
Клавиатура и мышь	ООО«ДНС»	5	890	2,5
Монитор *SamsungSMG*2703	ООО«ДНС»	5	12450	18

Таблица 4.2 – Потребность в лицензионном программном обеспечении

Лицензионное программное обеспечение	Источник	Цена, рублей	Срок действия лицензии, дней
Microsoft Windows 10 *Pro*	интернет-магазин ООО «СофтЛайн»	32000	бессрочная
Антивирус *Kaspersky* 2023/2024	интернет-магазин ООО «СофтЛайн»	1 999	365

Перечень дополнительных материалов, приобретение которых не требуется, приведены в таблице 4.3.

Таблица 4.3 – Потребность в дополнительных материалах

Материал	Источник	Количество, штук
Стол рабочий с надстройкой	ООО «Канцлер»	1
Настольная лампа	ООО «Канцлер»	5
Стол компьютерный	ООО «ДНС»	4
Кресло	ООО «ДНС»	4
Канцтовары	ООО «Канцлер»	5
Бумага	ООО «Канцлер»	5

Разработка программного обеспечения осуществлялась командой разработчиков, состав которой представлен в таблице 4.4

Состав коллектива разработчиков и информация по заработной плате разработчиков программного обеспечения и их количественному составу приведены в таблице 4.4.

Таблица 4.4 – Занятость и заработная плата разработчиков

Специалист	Количество, человек	Сумма заработной платы, рублей
Руководитель проекта	1	80000
Системный архитектор	1	60000
Специалист по защите информации	1	80000
Программист	1	60000
Тестировщик	1	50000

Себестоимость разработки программного продукта включает следующие статьи затрат:

- затраты на приобретение оборудования;
- затраты на приобретение лицензионного программного обеспечения;
- затраты на дополнительные материалы;
- затраты на заработную плату;
- затраты на обслуживание.

Затраты на оборудование в период разработки составят 328900 рублей на основании данных в таблице 4.1.

Затраты на программное обеспечение в период разработки составят 169995 рублей и представлены в таблице 4.2.

Затраты на дополнительные материалы в период разработки составят 0 рублей в соответствии с условием что потребности в закупке нет.

Затраты на заработную плату включают затраты на основную и дополнительную заработную плату разработчиков программного обеспечения и отчислениях на страховые взносы.

Затраты на основную заработную плату разработчиков определяются исходя из установленной заработной платы, количества специалистов и их занятости в процессе разработки программного продукта (продолжительность работы в днях).

Дополнительная заработная плата составляет 15% от затрат на основную заработную плату разработчиков.

С учетом выходных дней при 5-днейвной рабочей неделе количество рабочих дней в месяце в среднем составляет 21 день.

Затраты на основную и дополнительную заработную плату разработчиков программного обеспечения представлены в таблице 4.5.

Таблица 4.5 – Затраты на основную и дополнительную заработную плату разработчиков

Исполнитель	Количество, человек	Сумма заработной Платы, рублей	Период занятости, дней	Основная заработная плата, рублей	Дополнительная заработная плата, рублей	Всего рублей
Руководитель проекта	1	80000	63	240000	36000	276000
Системный архитектор	1	60000	63	180000	27000	207000
Специалист по защите информации	1	80000	63	240000	36000	276000
Программист	1	60000	63	180000	27000	207000
Тестировщик	1	50000	63	150000	22500	172500
Итого	5	156000	315	990000	148500	1138500

Таким образом, затраты на заработную плату разработчиков составят 1138500 рублей.

Отчисления на страховые взносы по нормам, установленным законодательством в ПРФ , ФОМС, ФСС РФ вычисляются исходя из данных представленных ниже.

Отчисления на страховые взносы производится согласно «Налоговому кодексу Российской Федерации (часть вторая)» от 05.08.2000 № 117-ФЗ (ред. от 26.03.2022).

Действующие тарифы на страховые взносы работодателей установлены статьей 34 НК РФ и составляют: ОПС (обязательное пенсионное страхование) – 22%, ВНиМ (страхование временной нетрудоспособности и в связи с

материнством) – 2,9%, ОМС (обязательное медицинское страхование) – 5.1%.

Общая сумма отчислений на страховые взносы составит 341550 рублей.

Общая сумма затрат является суммой: затрат на оплату труда включая налоги и сборы, затраты на закупку оборудования и лицензионного ПО и составляет 1637395 рублей.

При этом учреждение получает собственную техническую поддержку и отпадает необходимость в покупке дополнительных модулей, которые будут разработаны и внедрены собственной командой разработки.

5 Факторы освещения, воздействующие на оператора персонального компьютера

Безопасность жизнедеятельности включает в себя создание условий труда, которые обеспечивают состояние защищенности и отсутствия недопустимого риска нанесения вреда.

В процессе трудовой деятельности на человеческий организм могут воздействовать опасные факторы. Рабочее место программиста представляет собой набор инструментов, в результате работы которых возникает:

- повышенный уровень шума, источниками которого служат вентиляционные устройства ЭВМ, устройства ввода-вывода, агрегаты кондиционирования и вентилирования воздуха, другие электрические приборы;

- высокие уровни электростатического и электромагнитного излучения, источниками которого являются видеотерминалы. Так что, микроклимат, чистота воздуха, освещенность в помещении, влажность, наличие пыли влияют на человека, который находится в данной среде.

Параметры при работе с дисплейными терминалами и персональными компьютерами регламентирует СанПиН 2.2.2/2.4. 1340-03 «Гигиенические требования к персональным электронно-вычислительным машинам и организации работы».Общие рекомендации по электробезопасности, а также особенности воздействия других факторов даны в «Типовой инструкции по охране труда при работе на персональном компьютере» (ТОИ Р-45-084-01).

Согласно СанПиН 1.2.3685-21 «Гигиенические нормативы и требования к обеспечению безопасности и (или) безвредности для человека факторов среды обитания» освещенность на поверхности стола в зоне размещения рабочего документа должна быть 300-500 лк. Освещение не должно создавать бликов на поверхности экрана. Освещенность поверхности экрана не должна быть более 300 лк.

Следует ограничивать неравномерность распределения яркости в поле зрения пользователя ПЭВМ при этом соотношение яркости между рабочими поверхностями не должно превышать 3:1-5:1, а между рабочими поверхностями и поверхностями стен и оборудования - 10:1. Коэффициент пульсации не должен превышать 5%.

Для исключения бликов отражений в экране светильников общего освещения рабочий стол с ПК следует размещать между рядами светильников. При этом светильники должны быть расположены параллельно горизонтальной линии взгляда работающего. Для уменьшения бликов рекомендуется использовать защитный фильтр для видеомониторов. При рядном размещении рабочих столов не допускается расположение экранов дисплеев навстречу друг другу из-за их взаимного отражения, в противном случае между столами следует устанавливать перегородки.

При соблюдении вышеописанных требований рабочее место будет считаться пригодным для работы с персональным компьютером.

Заключение

В ходе выполнения дипломного проекта была изучена предметная область, в результате чего было сформулировано техническое задание на разработку программного обеспечения учета объектов недвижимости.

Был произведен выбор инструментария, выполнено проектирование, в результате которого были разработаны*UML*диаграммы разрабатываемого ПО.

Произведено проектирование и реализация частей системы, в результате, которого были разработаны макеты страниц, структура базы данных и логика серверной части приложения. Приложение реализовано в соответствии с техническим заданием.

В разделе безопасности жизнедеятельности были выявлены основные вредные факторы воздействия компьютера на человека, описаны требования к офисным помещениям для минимизации воздействия этих факторов.

В экономическом разделе был произведен расчет трудоемкости и затрат на разработку программного обеспечения.

Задание на дипломное проектирование выполнено в полном объеме.

Список использованных источников

1. Разработка *WEB*приложений: учеб. пособие / А. В. Мирбаев, С. А. Мамаев, В. Л. Лазарев. Москва: Изд-во МГУ, 2017. 430 с.

3. Фирма Контур: [Электронный ресурс] // Контур API ФГИС ЕГРН. Официальный сайт. *URL: https://reestro-docs.kontur.ru/realty-api/entity/methods/post-subject.html* (Дата обращения: 25.05.2023).

4. Создание баз данных на основе СУБД *MySQL*: учеб. пособие / Г. И. Иванов, Ю. М. Михеев. Екатеринбург: Изд-во Экос, 2022. 159 с.

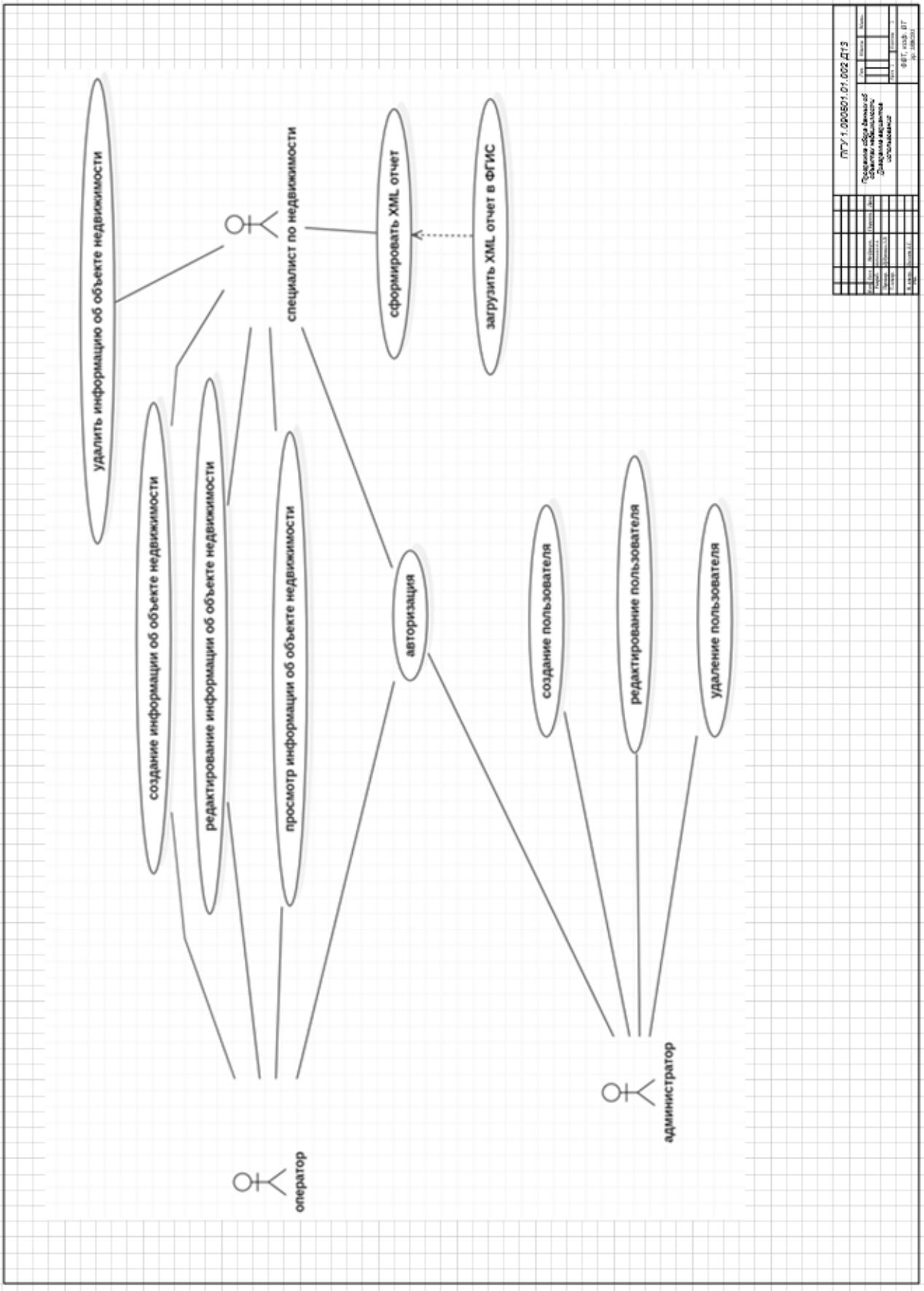
5. СанПиН 1.2.3685-21 «Гигиенические нормативы и требования к обеспечению безопасности и (или) безвредности для человека факторов среды обитания», утвержденные Постановлением Главного государственного санитарного врача РФ от 28.01.2021 № 2.

6. Справочная книга *LAMP* разработчика. / Под ред. Г.Б. Берга. – Л.: Маска, 2016.

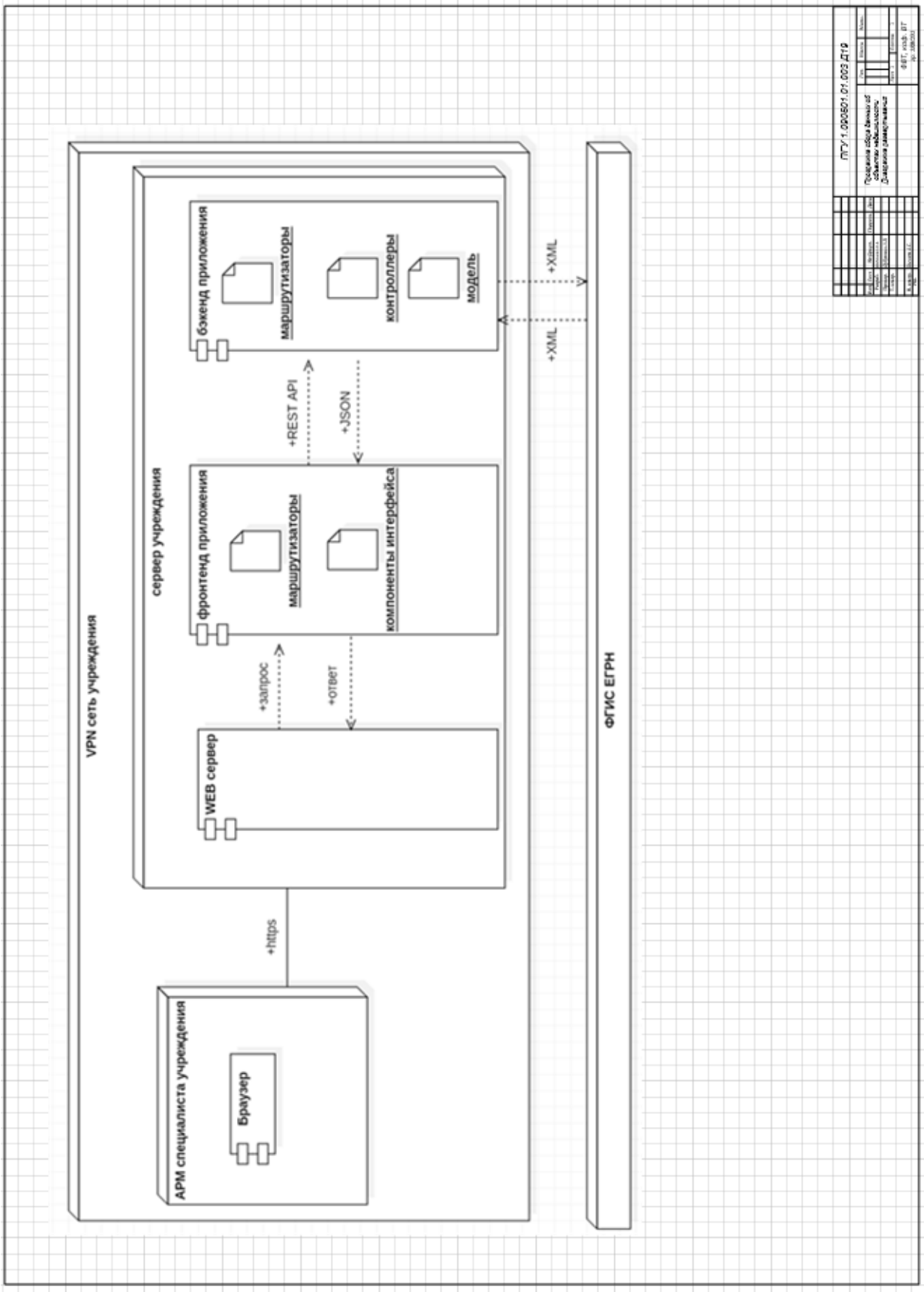
Приложение А

(Обязательное)
Программа сбора данных об объектах недвижимости
UML-диаграммы

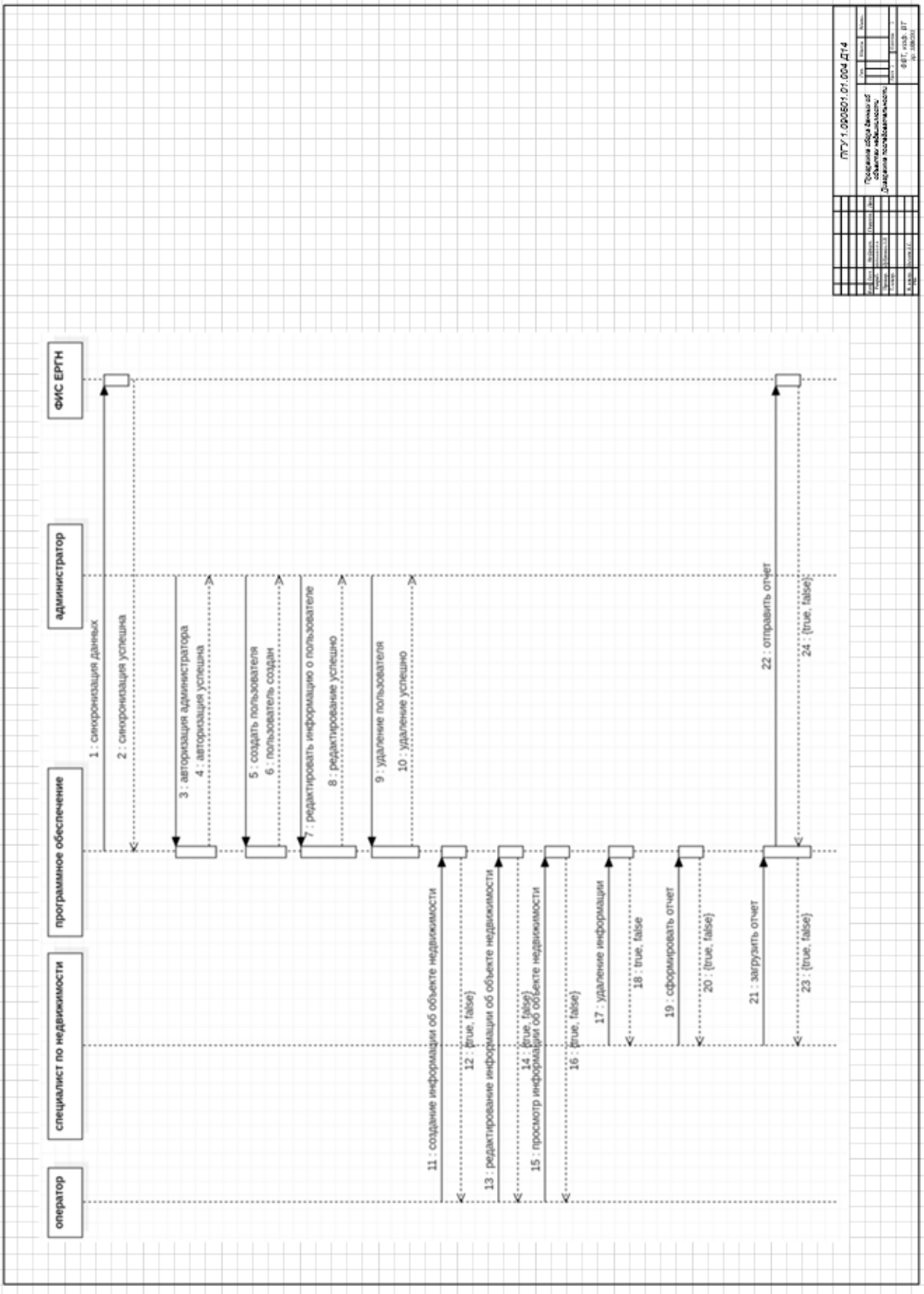
Приложение А.1 – Диаграмма вариантов использования



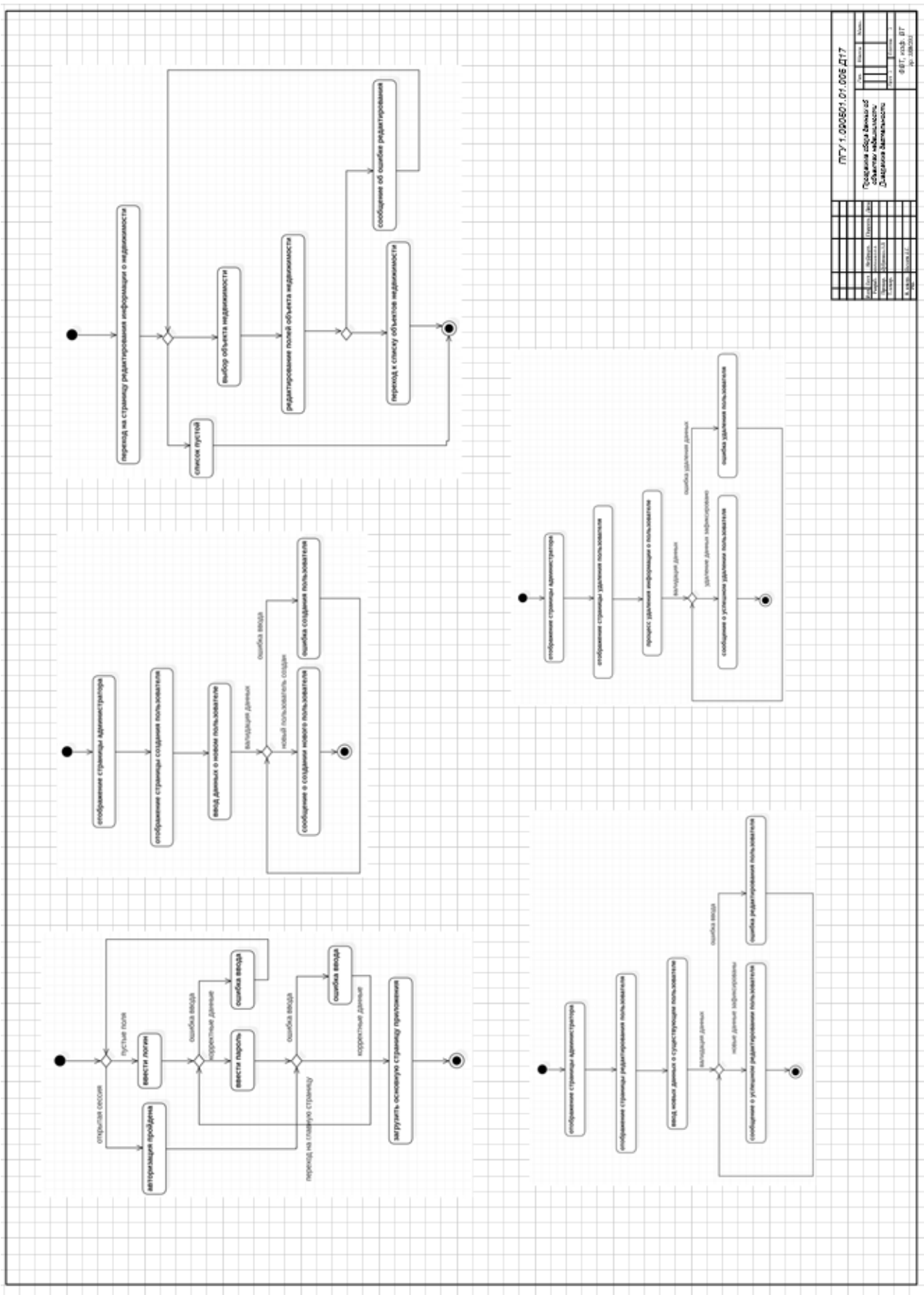
Приложение А.2 – Диаграмма развертывания



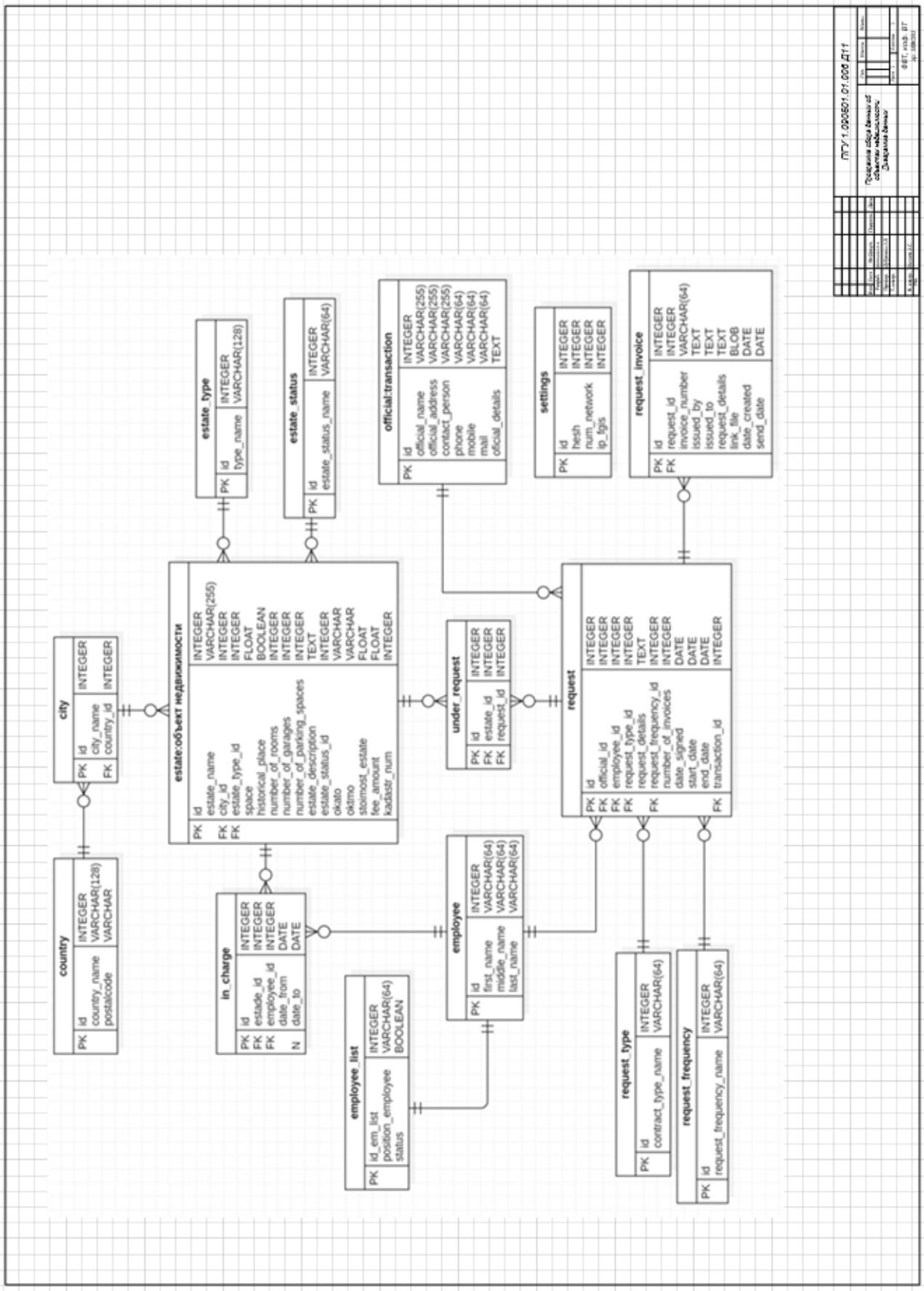
Приложение А.3 – Диаграмма последовательности



Приложение А.4 – Диаграмма деятельности



Приложение А.5 – Диаграмма данных



Приложение Б

(Обязательное)
Программа сбора данных об объектах недвижимости
Листинг программы

МодульMainController.php

```
<?php

class MainController extends BaseController {

    public function __construct()
    {
View::share('locations',Location::orderBy('created_at')->get());
View::share('news',Post::where('post_type','news')-
>orderBy('created_at','desc')->limit(4)->get());
    }

    public function main()
    {
        $data['properties'] = Property::where('status',1)-
>orderBy('created_at','desc')->limit(3)->get();
View::share('page_title','Homepage');
View::share('sliders',Property::where('status',1)-
>orderBy(DB::raw('RAND()'))->limit(10)->get());
View::share('sidebar',true);
View::share('homepage',true);
        return View::make('public.home',$data);
    }

    public function about(){
        $article = Post::find(163);
        if(is_null($article)){ return App::abort('404'); }
View::share('page_title',$article->title);
View::share('post',$article);
        return View::make('public.post');
    }
    public function services(){
        $article = Post::find(10);
        if(is_null($article)){ return App::abort('404'); }
View::share('page_title',$article->title);
View::share('post',$article);
        return View::make('public.post');
    }
    public function news(){
        $archives = Post::where('post_type','news')-
>orderBy('created_at','')->paginate(15);
        return View::make('public.news',array('archives' =>
$archives));
    }
    public function contact()
    {
        $post = Post::find(162);
        if(is_null($post)){ return App::abort('404'); }
View::share('page_title',$post->title);
View::share('post',$post);
        return View::make('public.contact');
    }
    public function contact_post()
```

```
    {
        $rules = array(
            'name' => 'required',
            'email' => 'required|email',
            'message' => 'required'
        );
        $validator = Validator::make(Input::all(),$rules);
        if($validator->fails())
        {
            return Redirect::to('contact-us')->withInput()-
>withErrors($validator);
}else{

            $admin_email = User::find(1)->email;
            $data = array(
                'from' =>Input::get('name'),
                'msg' =>Input::get('message'),
                'email' =>Input::get('email')
            );
Mail::send('mails.contact', $data, function($message)
use($admin_email)
            {
                $message->to($admin_email, 'Выйти')-
>subject('Contact Inquiry');
            });
            return Redirect::to('contact-us')-
>with('success','Message Sent!');
        }
    }
    public function article($id)
    {
        $article = Post::find($id);
        if(is_null($article)){ return App::abort('404'); }
View::share('page_title',$article->title);
View::share('post',$article);
        return View::make('public.post');
}
}
```

Модуль Login.php

```
@extends('layouts.public')
@section('content')
<div class="grid_12" id="left">
<div class="login-panel">
<h3 class="panel-title">Client Login</h3>
<div class="login-wrapper">
<div class="login-inner">
                    @include('layouts.notifications')
                    @if(Session::has('errors'))
<div class="alert alert-danger">
<h4>Warning!</h4>
                        @foreach($errors->all() as $message)
{{ $message }}<br>
```

```
                        @endforeach
</div><!--alert-->
                    @endif
{{ Form::open() }}
<div class="form-group">
{{ Form::label('Email Address:') }}
{{ Form::email('email',null,array('class' => 'form-control'))}}
</div>
<div class="form-group">
{{ Form::label('Password:') }}
{{ Form::password('password',array('class' => 'form-control'))}}
</div>
<div class="form-group">
<label>
{{ Form::checkbox('remember', 'Регистрация') }}
</label>
</div>
<div class="form-group">
{{ Form::submit('Авторизация',array('class' => 'button-yellow'))
}}
<p><br><a href="{{ URL::to('clients/forgot-password') }}"> d?</a>
| <a href="{{ URL::to('clients/register') }}">Регистрация</a></p>
</div>

{{ Form::close() }}
</div>
</div>
</div>
</div>
@stop
```

МодульTransaction.php

```
<?php

class Transaction extends Eloquent {
     protected $guarded = array();

     protected $softDelete = true;

     protected $appends = array('status_text');

     public static $rules = array(
          'status' => 'required',
     );

     public function user()
     {
          return $this->belongsTo('User');
     }

     public function property()
     {
          return $this->belongsTo('Property');
```

```
	}

	public function reservation()
	{
		return $this->belongsTo('Reservation');
	}

	public function getStatusTextAttribute()
	{
		if(strtolower($this->status) == 'paid')
		{
			return '<span class="label label-
success">Paid</span>';
		}

		if(strtolower($this->status) == 'pending')
		{
			return '<span class="label label-
default">Pending</span>';
		}

		if(strtolower($this->status) == 'cancelled')
		{
			return '<span class="label label-
danger">Cancelled</span>';
		}
	}
}
```

МодульUser.php

```
<?php

use Illuminate\Auth\UserInterface;
use Illuminate\Auth\Reminders\RemindableInterface;

class User extends \Cartalyst\Sentry\Users\Eloquent\User
implements UserInterface, RemindableInterface {

	public static $rules = array(
		'first_name'	=> 'required',
		'last_name'	=> 'required',
		'password'		=> 'required',
		'email'			=> 'required|unique:users,email',
		'tin_number'	=>
'required|str_len:12|unique:users,tin_number',
		'mobile'	=> 'required|numeric|str_len:11',
		'landline'		=> 'required|str_len:7|numeric'
	);

	public static function get_rules($id = null)
	{
		$rules = User::$rules;
		if(is_null($id))
		{
```

```
                return $rules;
            }else{
                $rules['password'] = 'min:8';
                $rules['email'] =
'required|unique:users,email,'.$id;
                $rules['tin_number'] =
'required|unique:users,tin_number,'.$id;
            }
            return $rules;
        }

        protected $softDelete = true;

        protected $appends = array('status');

        /**
         * The database table used by the model.
         *
         * @var string
         */
        protected $table = 'users';

        protected $hidden = array('password');

        public function getAuthIdentifier()
        {
            return $this->getKey();
        }
        public function getAuthPassword()
        {
            return $this->password;
        }


        public function getReminderEmail()
        {
            return $this->email;
        }

        public function getStatusAttribute()
        {
            if($this->activated == false || $this->activated == 0)
            {
                return '<span class="label label-
danger">Inactive</span>';
            }else{
                return '<span class="label label-
success">Active</span>';
            }
        }
}
```

Приложение В

(Обязательное)
Программа сбора данных об объектах недвижимости
Презентация

Программа сбора данных об объектах недвижимости

Автор работы:
студент гр. 18ВОЭ1 Водянова Ирина Алексеевна
Научный руководитель:
к.т.н., доцент кафедры ВТ Дубравин Алексей Викторович

Пенза, 2023

Цель работы и объекты

Цель работы – Разработать приложение для оперативного получения информации об объектах недвижимости государственного учреждения.
.

Объектом проекта является клиентское приложение, выполняющие следующие функции:

- подключение к платфоме ФГИС ЕГРН;
- выбор доступных режимов работы;
- работа в режиме авторизированного пользователя, в роли: администратор, специалист по недвижимости, оператор;
- формирование отчетов на основании собранных данных;
- завершение сеанса работы авторизированного пользователя.

Задачи приложения

Программа реализована в виде приложения доступ к которому осуществляется посредством «тонкого клиента», только в рамках защищенной сети учреждения. Работа с данными осуществляется по ключу – «кадастровому номеру» объекта недвижимости.

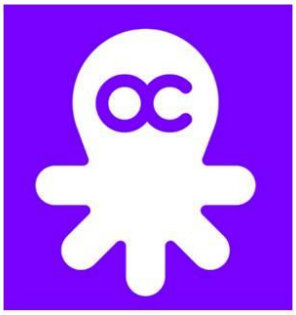
Входными данными являются:
•информация об объекте недвижимости
(идентификатор объекта, вид объекта недвижимости,
кадастровый номер, площадь объекта недвижимости,
адрес объекта недвижимости;
•информация о пользователях программы (роль пользователя);
информация о шаблонах отчетов в формате XML для взаимодействия с ФГИС ЕГРН.

Сравнение аналогов

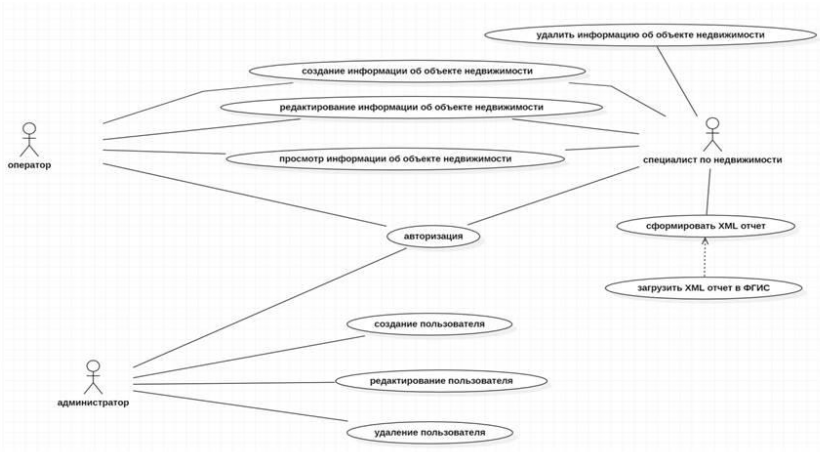
Панорама АО КБ «Панорама»	отечественное программное обеспечение входит в реестр российского программного обеспечения реконфигурация возможна силами конечного пользователя наличие «тонкого» клиента	реализовано только в виде «толстого» клиента, сохраняет отчеты в специальном формате, для перевода специального файла в XML требуется покупка дополнительного модуля отсутствует функция загрузки и просмотра фотографий объекта недвижимости
Pragmacore. Облачная платформа управления строительными проектами ООО «Прагма»	отечественное программное обеспечение входит в реестр российского программного обеспечения наличие мобильного клиентского приложения	реализовано в виде «облачного» сервиса, отсутствие API для работы со сторонним приложением платное программное обеспечение отсутствует функция загрузки и просмотра фотографий объекта недвижимости

Аналоги обладают недостатками:
облачные решения, которые не позволяют развернуть приложение на аппаратной платформе заказчика и полностью ее контролировать;
закрытые системы с отсутствием доступа к исходным кодам;
не все реализованое программное обеспечение кросплатформенное.

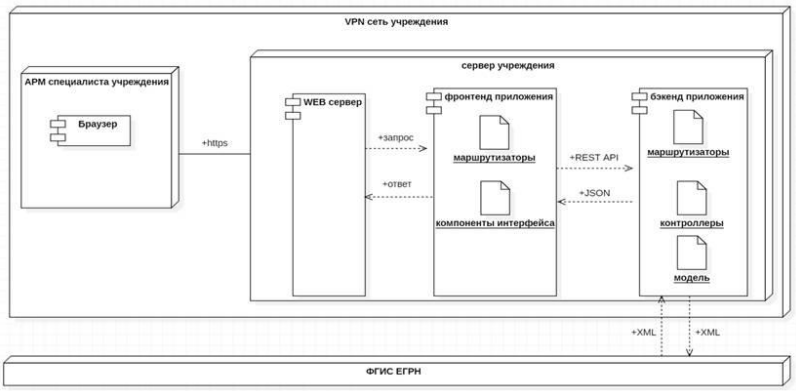
Стек технологий



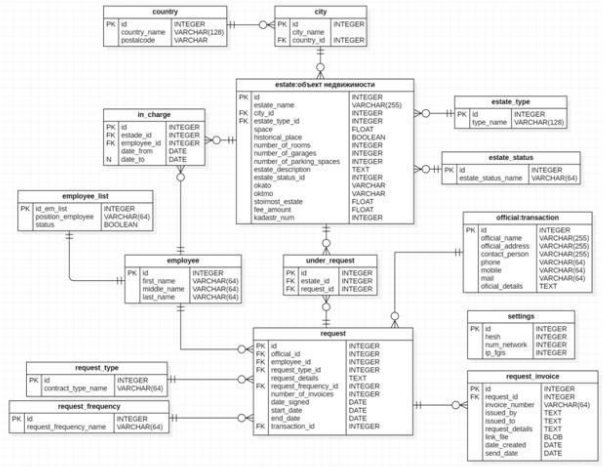
Основные функции приложения



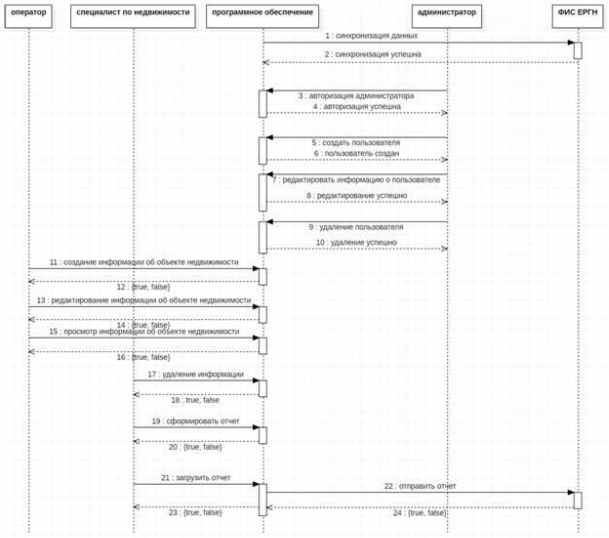
Архитектура приложения



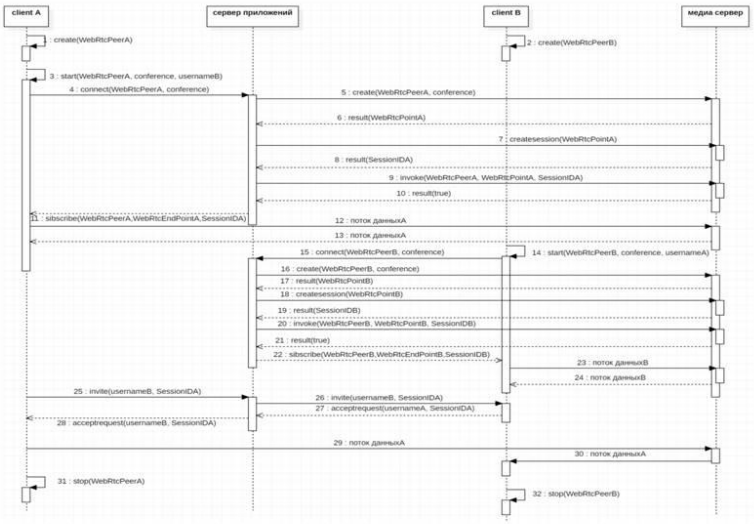
База данных приложения



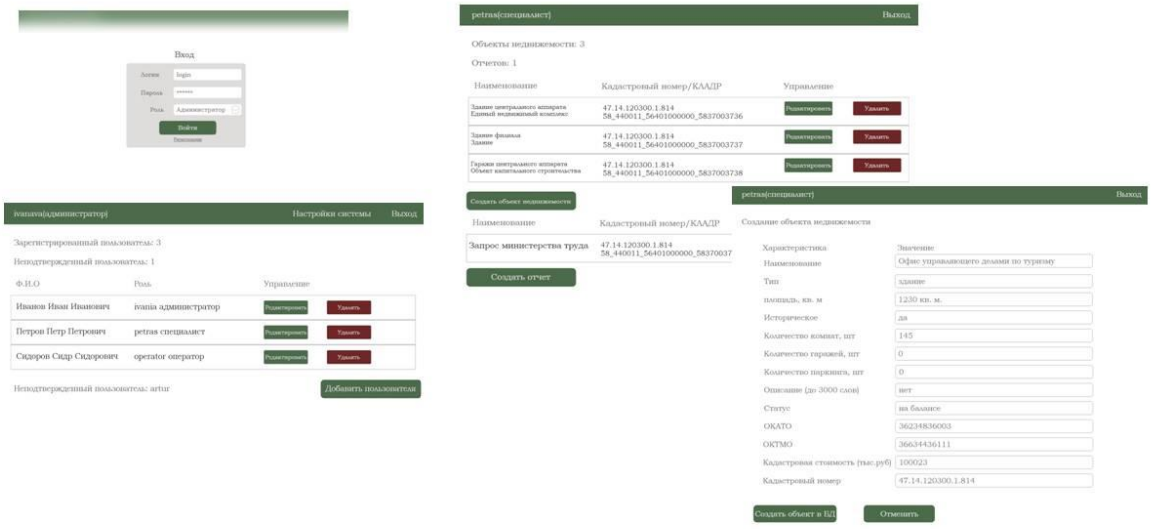
Работа приложения в различных режимах



Режим передачи сообщений



Скриншоты запущенного приложения



Результаты работы

- Разработано приложение полностью соответствующее техническому заданию;
- Проведено функциональное тестирование (успешно);
- Намечены задачи по развитию функциональных возможностей программы.

Спасибо за внимание!