

Министерство образования Республики Беларусь
Учреждение образования «Белорусский государственный университет
информатики и радиоэлектроники»

Инженерно-экономический факультет
Кафедра проектирования информационно-компьютерных систем
Дисциплина «Программирование сетевых приложений»

«К ЗАЩИТЕ ДОПУСТИТЬ»

Руководитель курсового проекта
старший преподаватель

_____.Т.М. Унучек
_____._____.2021

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему:

**«АНАЛИЗ И ПРОГНОЗИРОВАНИЕ ДАННЫХ НА РЫНКЕ АРЕНДЫ
ЖИЛЬЯ НА ОСНОВЕ НЕЙРОННЫХ СЕТЕЙ»**

БГУИР КП 1-40 05 01-10 009 ПЗ

Выполнил студент группы 914301
Ковалёв Данила Владимирович

(подпись студента)

Курсовой проект представлен на
проверку _____._____.2021

(подпись студента)

Минск 2021

СОДЕРЖАНИЕ

Введение.....	6
1 Описание предметной области.....	7
2 Постановка задачи и обзор методов её решения.....	12
3 Функциональное моделирование на основе стандарта IDEF0.....	15
4 Информационная модель системы и её описание.....	19
5 Описание алгоритмов, реализующих бизнес-логику серверной части проектируемой системы.....	21
6 Руководство пользователя.....	24
6.1 Алгоритм работы форм регистрации и авторизации.....	24
6.2 Алгоритм работы пользователя.....	26
6.3 Алгоритм работы администратора.....	28
7 Архитектурный шаблон проектирования.....	34
7.1 Разработка диаграммы использования.....	34
7.2 Разработка диаграммы классов.....	35
7.3 Разработка диаграммы компонентов.....	36
7.4 Разработка диаграммы последовательности.....	37
7.5 Разработка диаграммы состояний.....	38
7.6 Разработка диаграммы развертывания.....	39
8 Результаты тестирования разработанной системы.....	40
Заключение.....	42
Список использованных источников.....	43
Приложение А (рекомендуемое) Антиплагиат.....	44
Приложение Б (обязательное) Листинг алгоритмов, реализующих бизнес-логику.....	45
Приложение В (обязательное) Листинг основных элементов программы.....	48
Приложение Г (обязательное) Листинг скрипта генерации базы данных.....	60
Приложение Д (обязательное) Ведомость курсового проекта.....	63

ВВЕДЕНИЕ

Задача регрессии или же прогнозирования значений сейчас используется компаниями для подсчета будущей выручки на основании каких либо известных показателей. В двадцать первом веке для этого применяется методы машинного обучения. Для регрессии можно применять такие модели как линейная регрессия или искусственные нейронные сети.

Актуальность темы данной работы заключается в том, что в современных условиях совершенствование системы прогнозирования и планирования развития экономики приобретает особую актуальность. Такая система позволяет не только строить достоверные прогнозы развития экономики, но и своевременно вносить соответствующие поправки в различные планы и программы развития.

Цель: создание клиент-серверного приложения прогнозирования цен на жилье на основе нейронных сетей с организацией взаимодействия с базой данных на объектно-ориентированном языке Java.

Задачи:

- разработка серверной части;
- разработка базы данных;
- разработка клиентской части;
- разработка диаграмм последовательности и состояния;
- разработка блок-схем;
- разработка функциональная и инфомационные модели;
- разработка диаграмм состояний;
- разработка диаграмм последовательности;
- разработка диаграммы вариантов использования;
- разработка диаграмм классов;
- разработка диаграммы компонентов;
- разработка диаграммы развёртывания.

Процент уникальности работы в системе антиплагиат – 94,59 %.

Проверка производилась с помощью сайта: <https://www.antiplagiat.ru/>

Скриншот приведен ниже в приложении А.

1 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

Рассмотрим предметную область проекта.

Прогнозирование – метод, в котором на основании опыта, накопленного в прошлом, знаний настоящего делаются предположения о развитии событий в будущем.

Для прогнозирования в настоящее время чаще всего используются методы и модели машинного обучения такие как:

- линейная регрессионная модель
- полиномиальная регрессионная модель
- регрессия методом опорных векторов
- регрессия на основе случайных деревьев
- искусственная нейронная сеть

Если нужно учесть последовательности прошлых показателей эффективности, тогда можно использовать рекуррентные нейронные сети, сверточные нейронные сети или преобразователи.

Линейная регрессия – модель зависимости переменной x от одной или нескольких других переменных (факторов, регрессоров, независимых переменных) с линейной функцией зависимости.

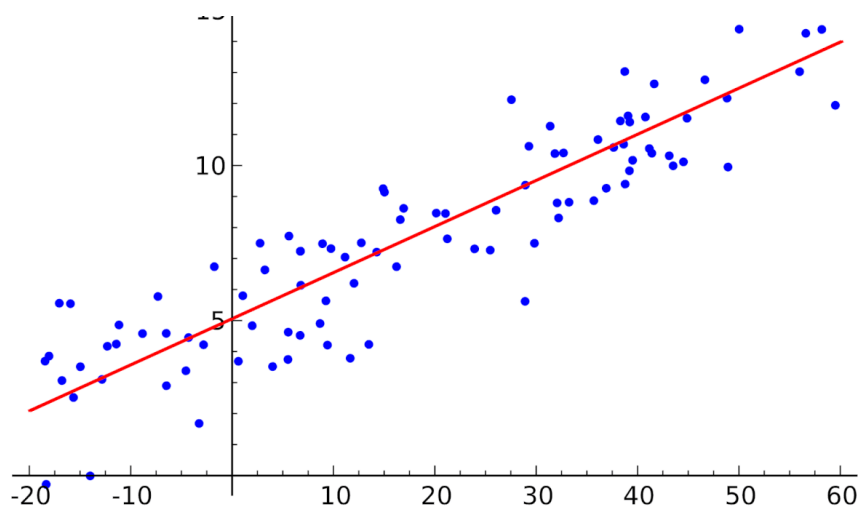


Рисунок 1.1– Модель линейной регрессии

Линейная регрессия относится к задаче определения “линии наилучшего соответствия” через набор точек данных и стала простым предшественником нелинейных методов, которые используют для обучения нейронных сетей

Полиномиальная регрессия – это форма линейной регрессии, в которой взаимосвязь между независимой переменной x и зависимой переменной y моделируется как полином n -ой степени

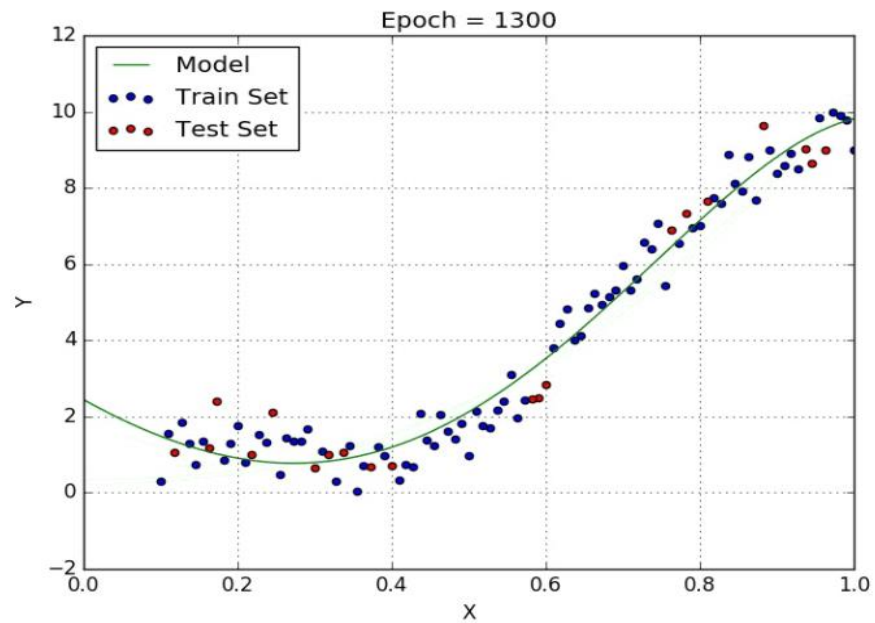


Рисунок 1.2 – Модель полиномиальной регрессии

Полиномиальная регрессия моделирует нелинейно разделенные данные (чего не может линейная регрессия). Она более гибкая и может моделировать сложные взаимосвязи.

Метод опорных векторов или SVM (от англ. Support Vector Machines) – это линейный алгоритм используемый в задачах классификации и регрессии. Данный алгоритм имеет широкое применение на практике и может решать как линейные так и нелинейные задачи. Суть работы SVM проста: алгоритм создает линии или гиперплоскость, которая разделяет данные на классы.

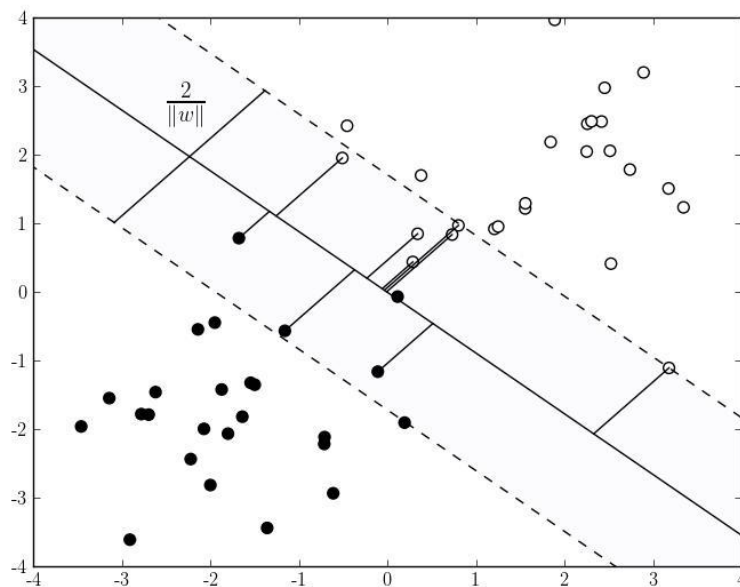


Рисунок 1.3 – Метод опорных векторов

Алгоритм случайного леса (Random forest) – универсальный алгоритм машинного обучения, суть которого состоит в использовании ансамбля решающих деревьев. Само по себе решающее дерево предоставляют крайне невысокое качество классификации, но из-за большого их количества результат значительно улучшается. Также это один из немногих алгоритмов, который можно использовать в абсолютном большинстве задач.

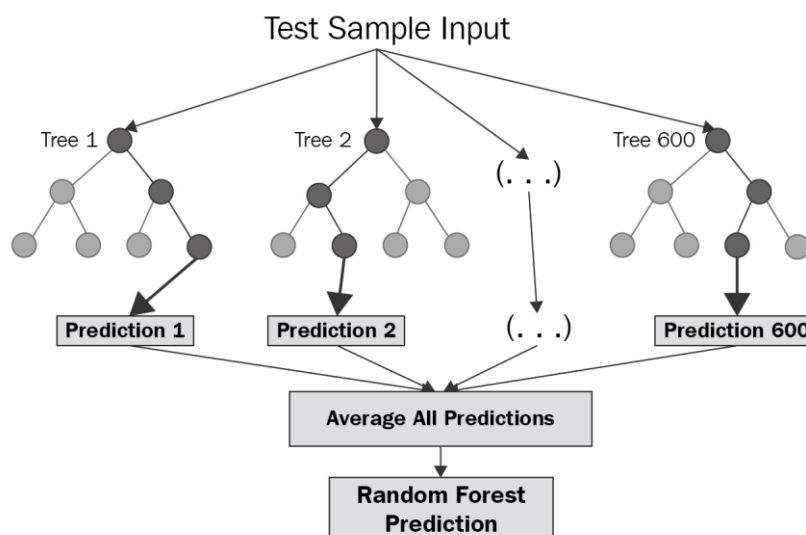


Рисунок 1.4 – Метод случайного леса

Искусственная нейронная сеть – попытка с помощью математических моделей воспроизвести работу человеческого мозга создания машин, обладающих искусственным интеллектом.

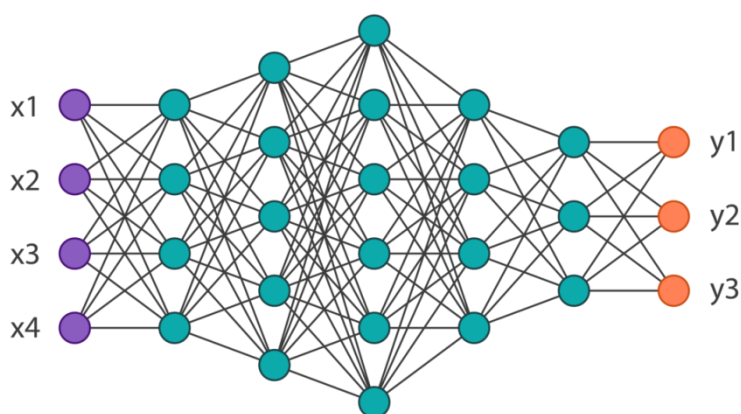


Рисунок 1.5 – Искусственная нейронная сеть

Искусственная нейронная сеть обычно обучается с учителем. Это означает наличие обучающего набора (датасета), который содержит примеры с истинными значениями: тегами, классами, показателями.

Например, если вы хотите создать нейросеть для оценки тональности текста, датасетом будет список предложений с соответствующими каждому эмоциональными оценками. Тональность текста определяют признаки (слова фразы, структура предложения), которые придают негативную или позитивную окраску. Веса признаков в итоговой оценке тональности текста (позитивный, негативный, нейтральный) зависят от математической функции, которая вычисляется во время обучения нейронной сети.

Рекуррентные нейронные сети (Recurrent Neural Networks, RNNs) – популярные модели, используемые в обработке естественного языка (NLP). Во-первых, они оценивают произвольные предложения на основе того, насколько часто они встречались в текстах. Это дает нам меру грамматической и семантической корректности. Такие модели используются в машинном переводе. Во-вторых, языковые модели генерируют новый текст. Обучение модели на поэмах Шекспира позволит генерировать новый текст, похожий на Шекспира.

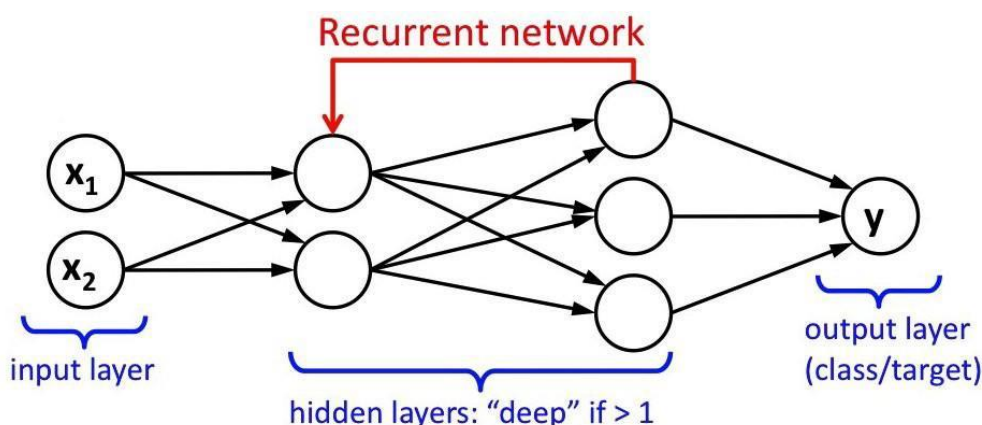


Рисунок 1.6 – Модель рекуррентной нейронной сети

Идея RNN заключается в последовательном использовании информации. В традиционных нейронных сетях подразумевается, что все входы и выходы независимы. Но для многих задач это не подходит. Если вы хотите предсказать следующее слово в предложении, лучше учитывать предшествующие ему слова. RNN называются рекуррентными, потому что они выполняют одну и ту же задачу для каждого элемента последовательности, причем выход зависит от предыдущих вычислений. Еще одна интерпретация RNN: это сети, у которых есть «память», которая учитывает предшествующую информацию. Теоретически RNN могут использовать информацию в произвольно длинных последовательностях, но на практике они ограничены лишь несколькими шагами.

2 ПОСТАНОВКА ЗАДАЧИ И ОБЗОР МЕТОДОВ ЕЁ РЕШЕНИЯ

В данном проекте необходимо произвести прогноз цен на жилье с помощью применения нейронной сети.

Для реализации данного проекта необходимо выполнить следующие задачи:

- реализовать клиент-серверное приложение;
- разработать базу данных;
- создать модель нейронной сети
- обучить модель нейронной сети
- создать удобный интерфейс для пользователя;
- распределить роли (администратор и пользователь);
- разработать собственную иерархию классов;
- реализовать не менее двух паттернов проектирования;
- предусмотреть обработку исключительных ситуаций;
- протестировать полученное приложение.

Со стороны администратора программа должна предусматривать следующие возможности:

- добавление пользователей;
- просмотр пользователей;
- удаление пользователей;
- блокировка пользователей;
- изменение пользователя;
- просмотр данных пользователей;
- просмотр статистики приложения;

Со стороны пользователя программа должна предусматривать следующие возможности:

- регистрация;
- просмотр своей статистики;
- ввод значения для прогноза;
- просмотр видов значений;
- получение результатов прогнозирования;

Для решения поставленных задач используются PostgreSQL, IntelliJ IDEA, PyCharm, язык программирования Java и его платформа JavaFX, язык программирования Python и его библиотеки Flask и Keras.

PostgreSQL была выбрана из-за ряда преимуществ, таких как:

- поддержка БД неограниченного размера;
- мощные и надежные механизмы транзакций и репликации;

- расширяемая система встроенных языков программирования и поддержка загрузки С-совместимых модулей;
- наследование;
- легкая расширяемость;

Также для реализации поставленной задачи используется объектно-ориентированный язык Java. При создании серверных приложений с помощью Java встречается минимальное число проблем.

Для создания сервиса с нейронной сетью был использован высокоуровневый язык программирования общего назначения Python.

Для работы с языком Java используется IntelliJ IDEA – мощная кроссплатформенная среда программирования

Для работы с языком Python используется PyCharm.

Для разработки интерфейса используется JavaFX. Технология JavaFX обеспечивает создание мощного графического интерфейса пользователя для крупномасштабных приложений, ориентированных на обработку данных, насыщенных медиаприложений, поставляющих разнообразный медиаконтент пользователю.

Для разработки сервиса с нейронной сетью используется микрофреймворк Flask – который предоставляет лишь самые базовые возможности для разработки веб-приложений.

Для создания и обучения нейросети используется библиотека Keras. Keras – открытая библиотека, написанная на языке Python и обеспечивающая взаимодействие с искусственными нейронными сетями. Она представляет собой надстройку на фреймворке Tensorflow.

Преимущества Keras:

- простой в использовании с высокой скоростью развертывания
- качественная документация и поддержка от сообщества
- поддержка разных движков и модульность
- натренированные модели
- поддержка нескольких GPU

Соединение между серверной и клиентскими частями должно устанавливаться с помощью протокола TCP/IP. Этот протокол обладает одним важным преимуществом: он обеспечивает аппаратную независимость. Основное преимущество TCP/IP — надёжность.

Соединение между сервером и сервисом устанавливается с помощью протокола HTTP.

Также используются следующие шаблоны проектирования:

- одиночка
- строитель
- декоратор

Одиночка – это порождающий паттерн проектирования, который гарантирует, что у класса есть только один экземпляр, и предоставляет к нему глобальную точку доступа.

Преимущества:

- гарантирует наличие единственного экземпляра класса;
- предоставляет к нему глобальную точку доступа;
- реализует отложенную инициализацию объекта-одиночки;

Строитель — это порождающий паттерн проектирования, который позволяет создавать сложные объекты пошагово. Строитель дает возможность использовать один и тот же код строительства для получения разных представлений объектов.

Преимущества:

- позволяет создавать продукты пошагово;
- позволяет использовать один и тот же код для создания различных продуктов;
- изолирует сложный код сборки продукта от его основной бизнес-логики

Декоратор — это структурный паттерн проектирования, который позволяет динамически добавлять объектам новую функциональность, оборачивая их в полезные «обёртки».

Преимущества:

- Большая гибкость, чем у наследования.
- Позволяет добавлять обязанности на лету.
- Можно добавлять несколько новых обязанностей сразу.
- Позволяет иметь несколько мелких объектов вместо одного объекта на все случаи жизни.

3 ФУНКЦИОНАЛЬНОЕ МОДЕЛИРОВАНИЕ НА ОСНОВЕ СТАНДАРТА IDEF0

На диаграмме IDEF0 представлено описание процесса прогнозирования цены на жилье. На рисунке 2.1 представлена контекстная диаграмма верхнего уровня.

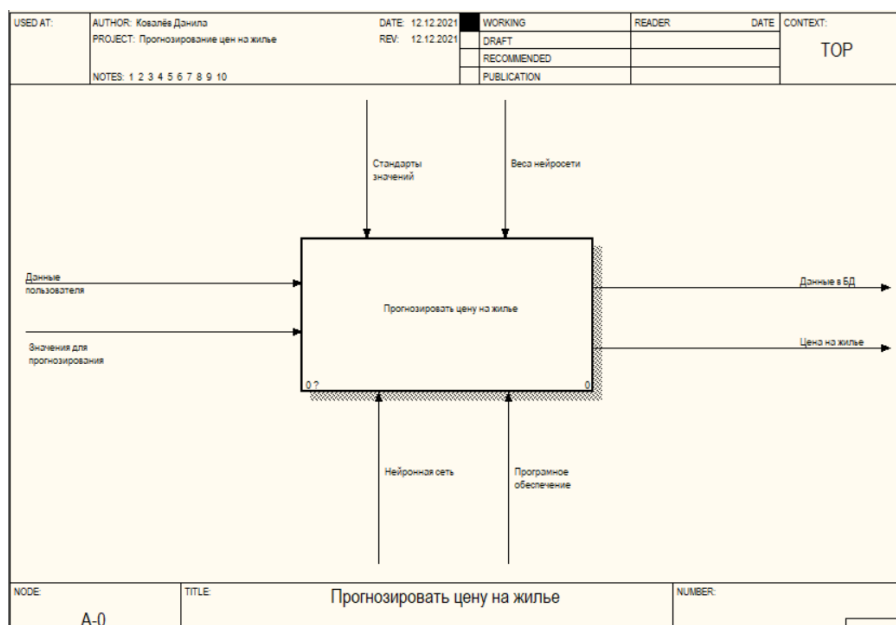


Рисунок 2.1 – Контекстная диаграмма функциональной модели

Входными данными являются: данные пользователя и значения для прогнозирования. Результатом работы программы будет являться записи в БД и спрогнозированная цена на жилье. В качестве механизма выступают программное обеспечение и нейронная сеть. Управлением будут являться веса нейронной сети и стандарты значений.

На следующем рисунке 2.2 представлена декомпозиция процесса «Прогнозировать цену на жилье».

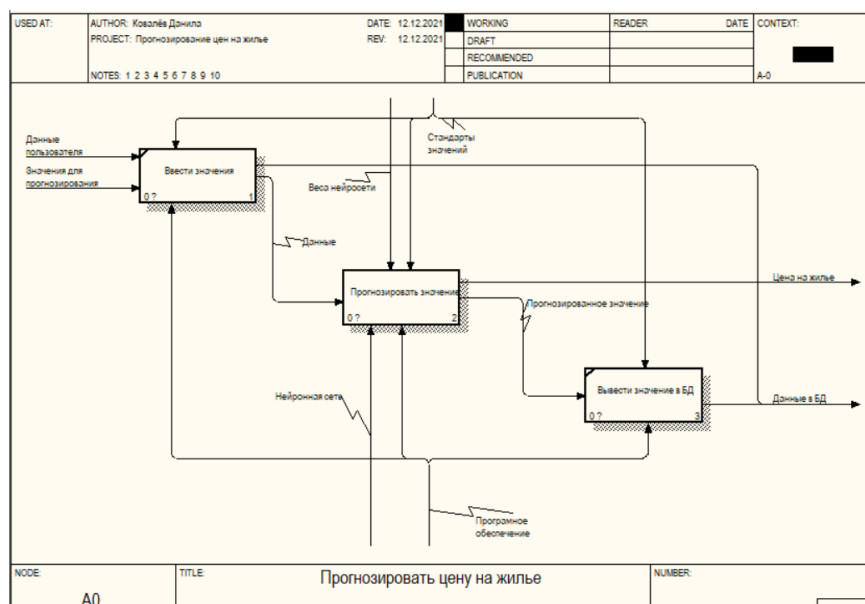


Рисунок 2.2 – Декомпозиция контекстной диаграммы

Данный уровень разбит на 3 функциональных блока: ввести значения, прогнозировать значения, вывести значения в БД.

Стрелка управления «Стандарты значений» входит во все 3 блока, стрелка управления «Веса нейросети» и стрелка механизма «Нейронная сеть» входят только в функциональный блок «Прогнозировать значения». Стрелка механизма «Программное обеспечение» взодит во все 3 функциональных блока.

Функциональный блок «Ввести значения» передаёт следующему данные, по которым будет производиться расчет цены на жилье.

Выходные значения из функционального блока «Прогнозировать значения» передаются на выход всей диаграммы и в следующий функциональный блок. В следующем функциональном блоке переданные данные заносятся в БД.

Перейдём на 3 уровень декомпозиции блока A2 (см. рисунок 2.3).

Диаграмма представлена 4 блоками: A21 – получить данные, A22 – проверить количество данных, A23 – проверить данные на валидность, A24 – прогнозировать цену. В качестве управления и механизма остаются те же, что и на 1 уровне, «Веса нейросети» и «Нейронная сеть» входят только в функциональный блок A24 .

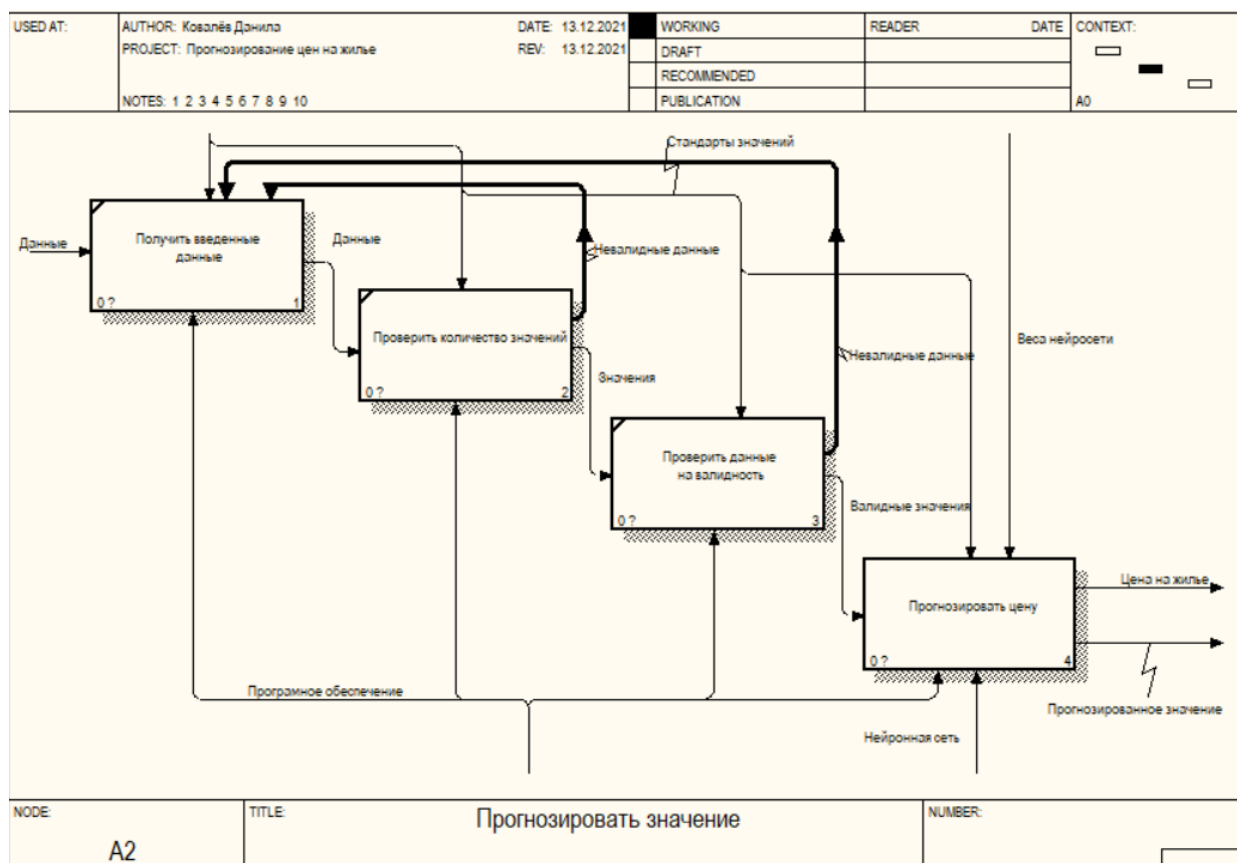


Рисунок 2.3 – Декомпозиция функционального блока «Прогнозировать значения»

Из функции A21 данные передаются в блок «проверить количество данных», далее в блок A23 и, в конечном итоге, в A24 а затем и на выход. Если данные не являются верными, то из блоков A22 и A23 они переходят в A21. Процесс начинается заново.

Рассмотрим 4 уровень построенной IDEF0. Декомпозиция функционального блока «Прогнозировать цену» представлена на рисунке 2.4.

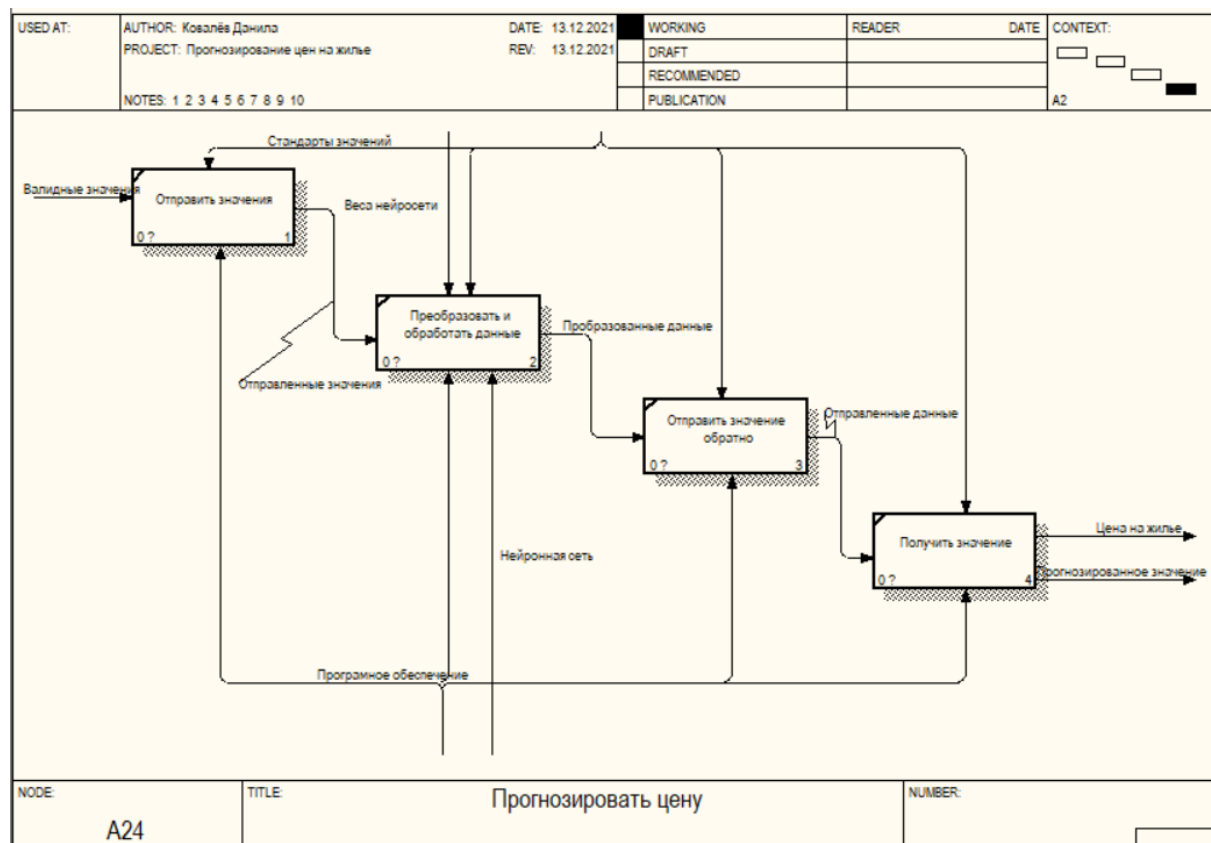


Рисунок 2.4 – Декомпозиция функционального блока «Прогнозировать цену»

На данном этапе декомпозиция состоит из 4 блоков:отправить значения, преобразовать и обработать данные, отправить значение обратно, получить значение. Управление и механизм соответствует уровню 3.

На вход поступают валидные данные, далее данные обрабатываются в первой функции A32 и передаются в A33 затем в A34.

4 ИНФОРМАЦИОННАЯ МОДЕЛЬ СИСТЕМЫ И ЕЁ ОПИСАНИЕ

Рассмотрим разработанную логическую модель системы (см. рисунок 2.1).

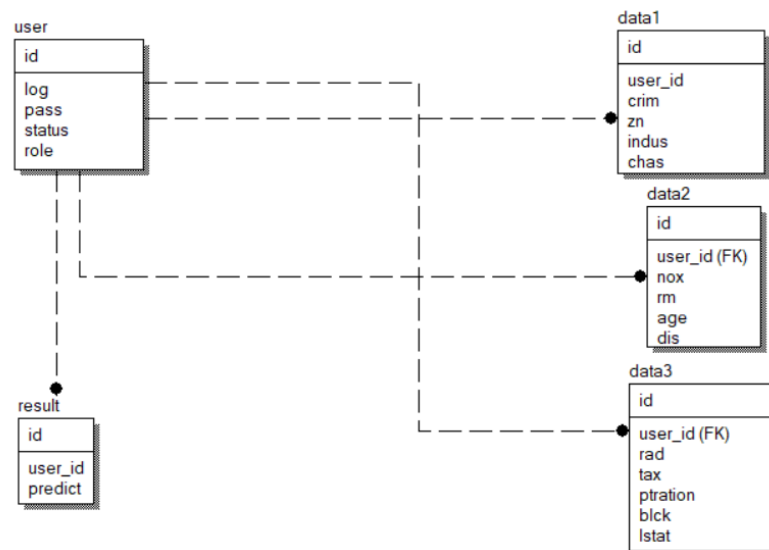


Рисунок 3.1 – Логическая модель системы

Логическая модель системы разработана с помощью средства AllFusion ERwin Data Modeler. Система содержит 5 сущностей: пользователи (user), первая часть значений (data1), вторая часть значений (data2), третья часть значений (data3), результат (result). У каждой сущности есть первичный ключ, который хранит уникальный номер.

Рассмотрим каждую из сущностей.

Сущность «пользователи» имеет 5 атрибутов: id пользователя, логин, пароль, статус и роль. Статус хранит информацию о том, заблокирован пользователь или нет. Роль хранит информацию о том является пользователь администратором или обычным пользователем. Сущности «data1», «data2», и «data3» хранят значения запросов, а так же поле «user_id» отвечает за то какому пользователю принадлежат значения.

Последним этапом является физическое проектирование. Это описание физической структуры базы данных. Физическая модель системы создана при помощи программы PgAdmin4 (см. рисунок 2.2).

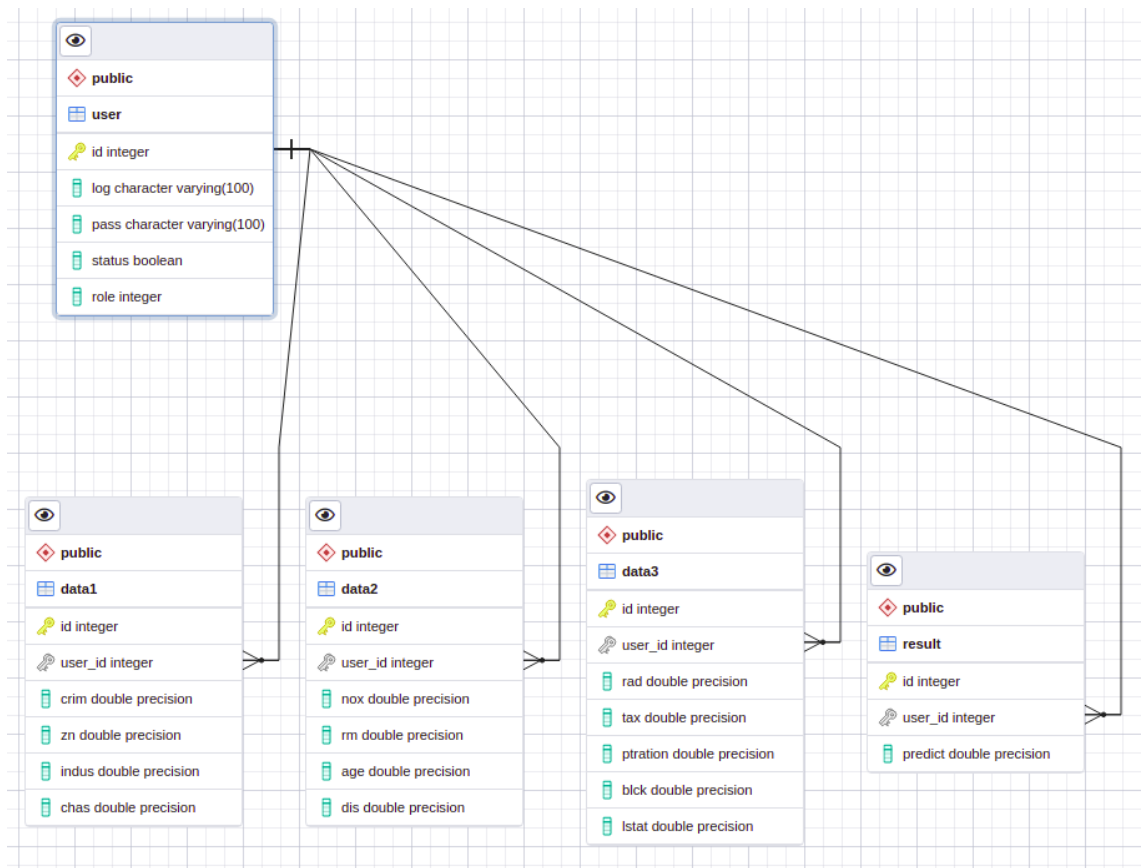


Рисунок 3.2 – Физическая модель системы

База данных была приведена к третьей нормальной форме, так как у каждой таблицы имеется всего один первичный ключ, а каждое не ключевое поле не транзитивно зависит от первичного ключа, то есть изменив значение в одном столбце, не потребуется изменение в другом столбце.

5 ОПИСАНИЕ АЛГОРИТМОВ, РЕАЛИЗУЮЩИХ БИЗНЕС-ЛОГИКУ СЕРВЕРНОЙ ЧАСТИ ПРОЕКТИРУЕМОЙ СИСТЕМЫ

Рассмотрим алгоритм механизма прогнозирования цен.

На первом этапе пользователь вводит собственные значения, далее производится проверка введённых данных – проверяется количество введенных цифр и все ли значения являются числами (см. рисунок 4.1).

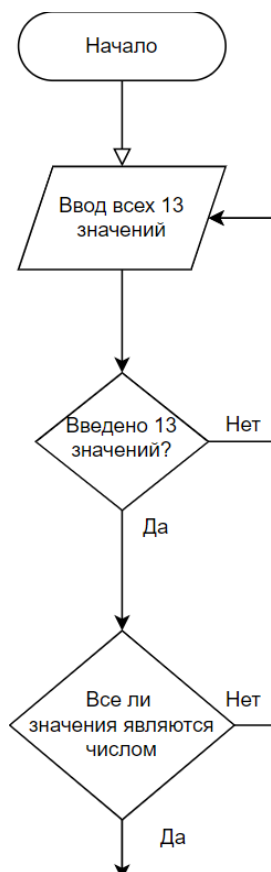


Рисунок 4.1 – Ввод и проверка значений

Далее производится преобразование значений в строку, а затем в JSON, после данные отправляются на сервер (см. рисунок 4.2).

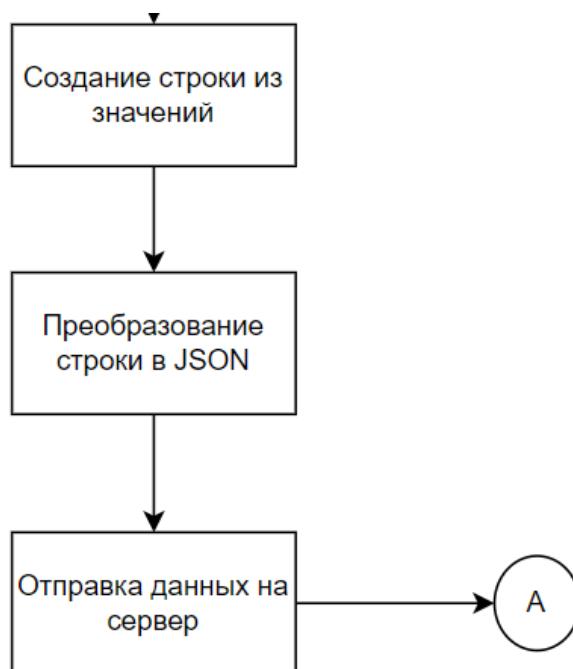


Рисунок 4.2 – Преобразование и отправка данных

После получения данных сервер преобразует полученные данные в JSON формате в массив из дробных чисел, которые затем передаются нейронной сети, и уже после нейросеть обрабатывает значения и выдаёт соответственно результат (см. рисунок 4.3).

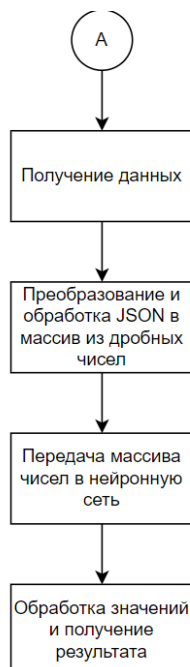


Рисунок 4.3 – Обработка данных

После получения результата данные отправляются обратно пользователю и выводятся в текстовое окно, а затем эти данные заносятся в базу данных(см. рисунок 4.4) .

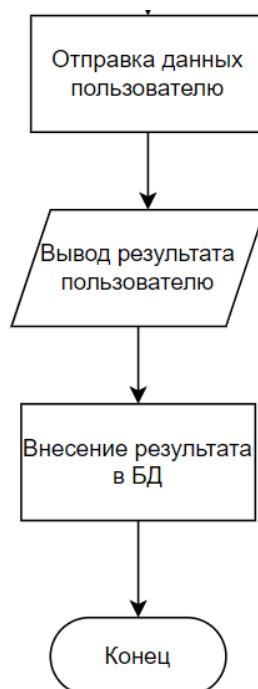


Рисунок 4.4 – Расчёт зарплаты по комиссионной система, но не менее фиксированного оклада

После, пользователь может повторно произвести операцию расчета стоимости, не выходя из программы.

6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Данное программное обеспечение представляет собой систему по прогнозированию цен на жилье.

6.1 Алгоритм работы форм регистрации и авторизации

После запуска приложения открывается форма главного меню(см. рисунок 5.1), где пользователь может выбрать пункт «Авторизация»(см. рисунок 5.2) или «Регистрация»(см. рисунок 5.3).

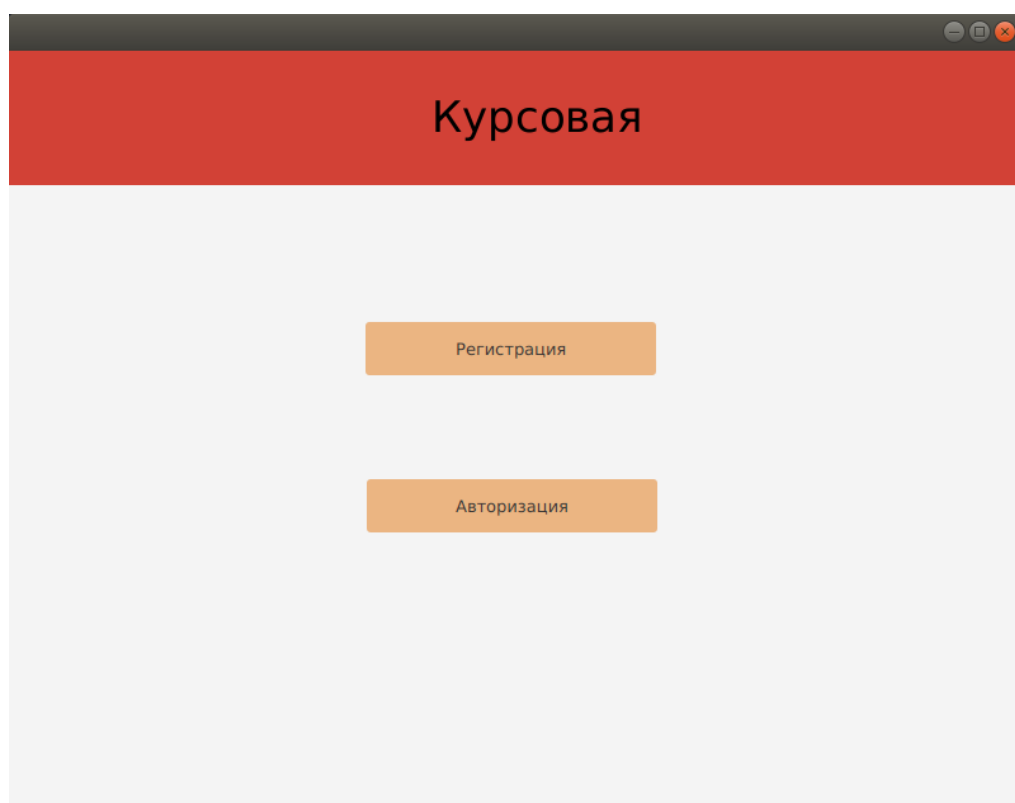
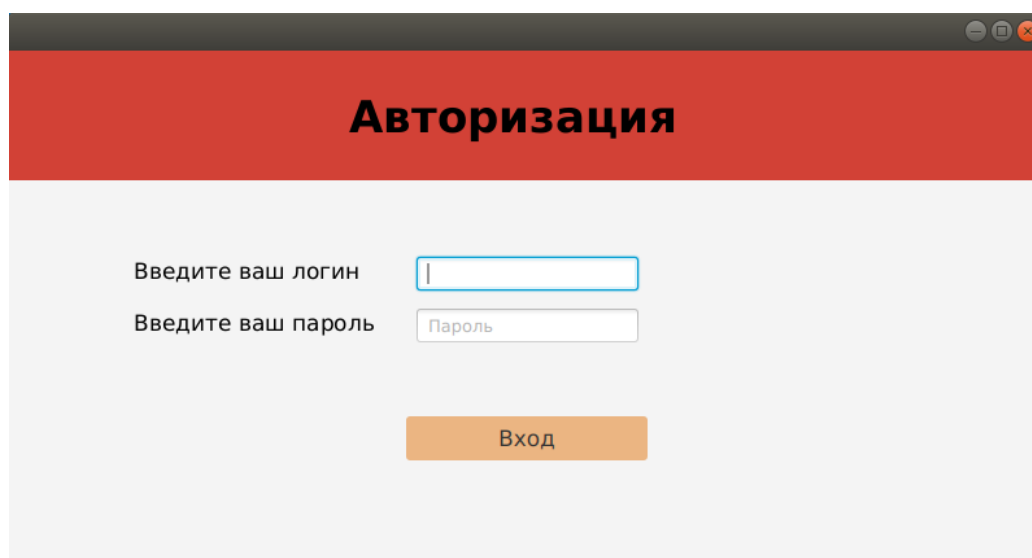


Рисунок 5.1 – Главное меню

После нажатия на пункт «Авторизация», открывается форма где можно ввести логин и пароль. Если данные будут верные вас перенесет на форму соответствующей роли.



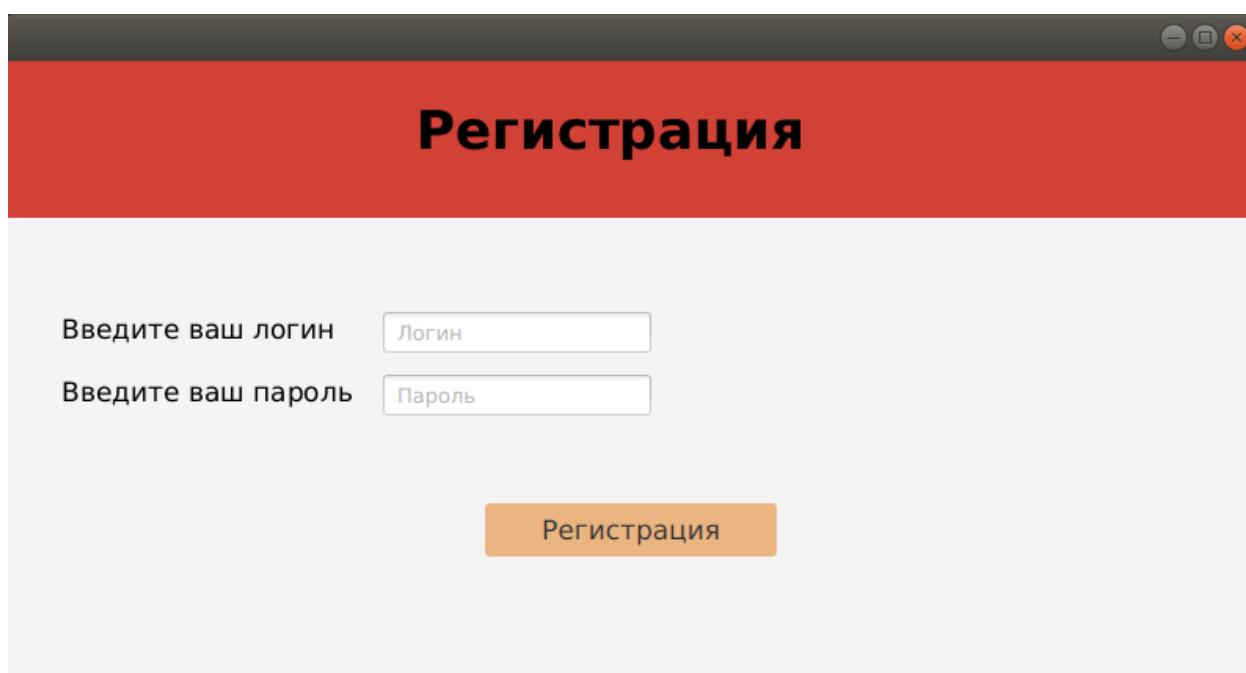
Авторизация

Введите ваш логин

Введите ваш пароль

Рисунок 5.2 – Авторизация

После выбора пункта «Регистрация», открывается форма регистрации(см. рисунок 4.3), где можно ввести свой новый логин и новый пароль. После чего вас перенесет на форму авторизации.



Регистрация

Введите ваш логин

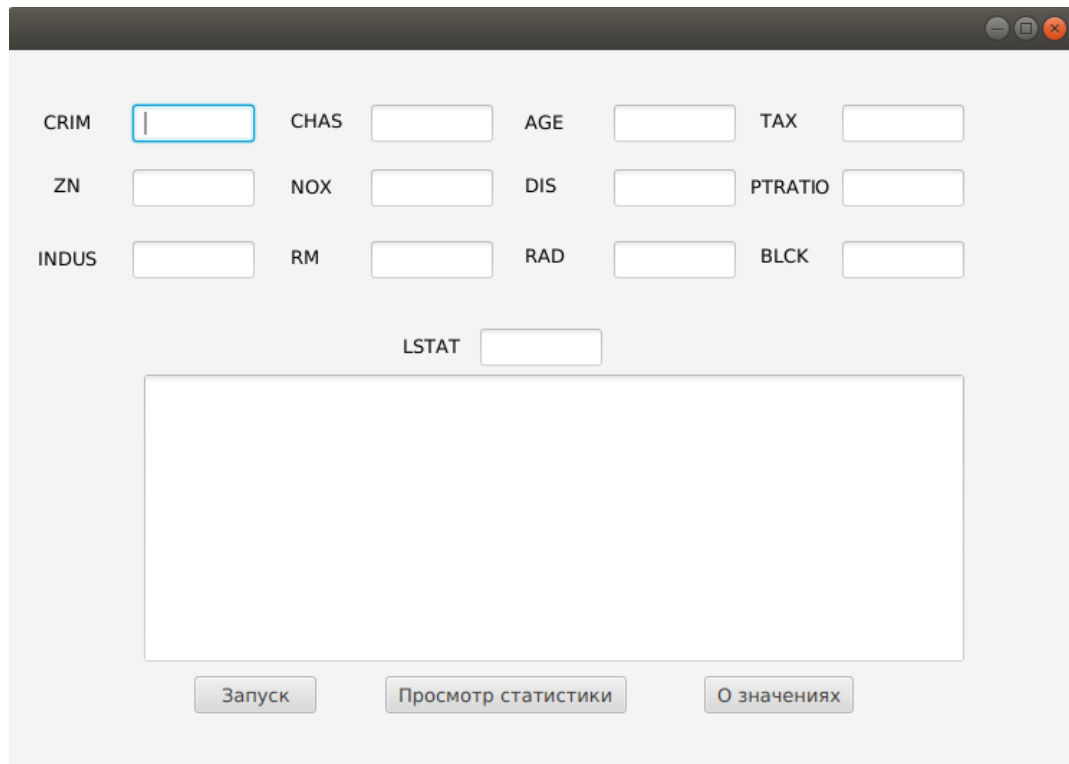
Введите ваш пароль

Рисунок 5.3 – Регистрация

Если логин уже существует, то пользователю выдаст соответствующее уведомление.

6.2 Алгоритм работы пользователя

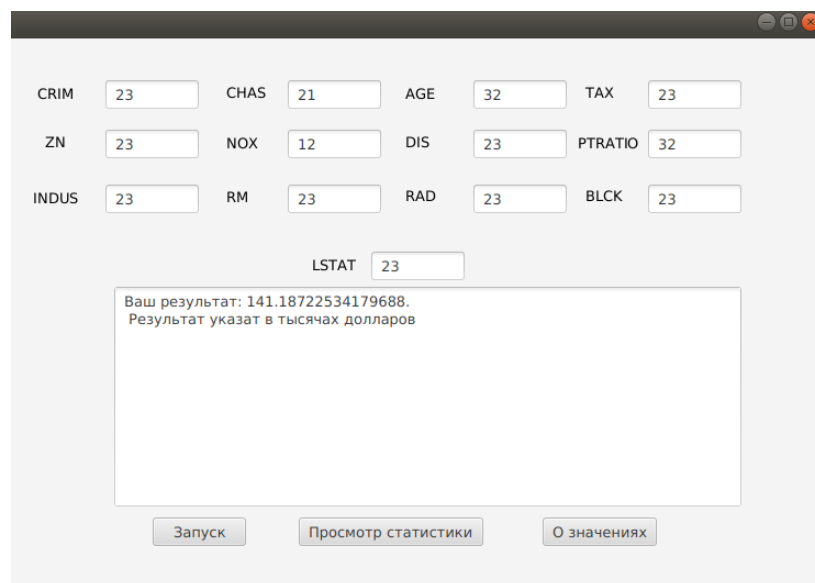
При авторизации как пользователь открывается соответствующее окно(см. рисунок 5.4).



The screenshot shows a window titled "Меню пользователя" (User Menu). It contains several input fields for variables: CRIM, CHAS, AGE, TAX, ZN, NOX, DIS, PTRATIO, INDUS, RM, RAD, and BLCK. Each variable has a corresponding input field. Below these fields is a large empty box for the result. At the bottom of the window are three buttons: "Запуск" (Run), "Просмотр статистики" (View Statistics), and "О значениях" (About Values).

Рисунок 5.4 – Меню пользователя

Пользователь может вводить значения, производить расчет, просматривать свою статистику, а так же информацию о значениях.



The screenshot shows the same window as Figure 5.4, but with values entered in the input fields. The variables and their values are: CRIM (23), CHAS (21), AGE (32), TAX (23), ZN (23), NOX (12), DIS (23), PTRATIO (32), INDUS (23), RM (23), RAD (23), and BLCK (23). The LSTAT field is also filled with 23. The large box for the result now contains the text: "Ваш результат: 141.18722534179688. Результат указат в тысячах долларов" (Your result: 141.18722534179688. The result is indicated in thousands of dollars). The buttons at the bottom remain the same.

Рисунок 5.5 – Ввод значений и произведенный расчет

После обязательного ввода тринадцати значений пользователь может запустить основной механизм программного обеспечения и получить прогнозируемую цену на жилье, который будет выведен в текстовое окно(см рисунок 5.5).

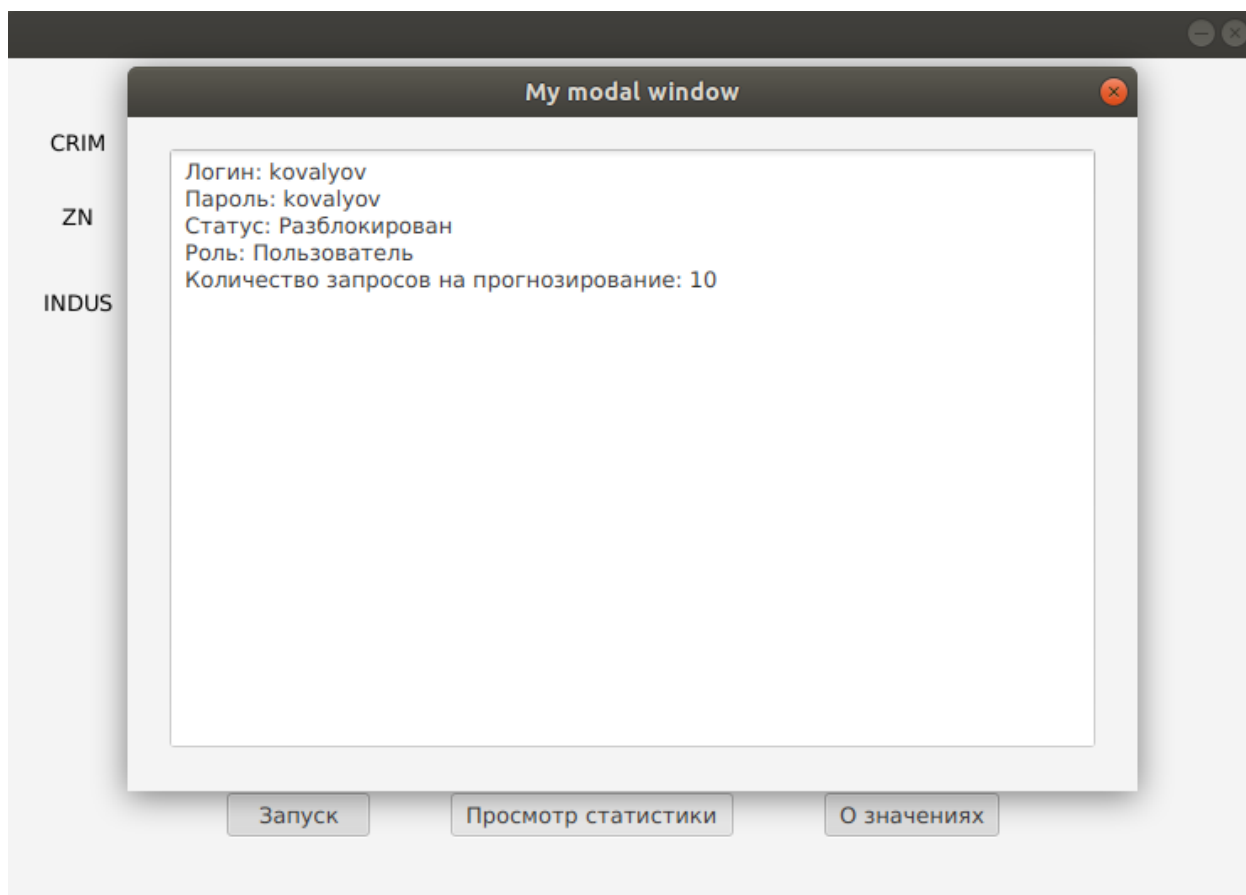


Рисунок 5.6 – Просмотр статистики

Просмотр статистики выводит на экран окно со следующими данными: логин, пароль, статус, роль, а так же количество запросов на прогноз(см. рисунок 5.6).

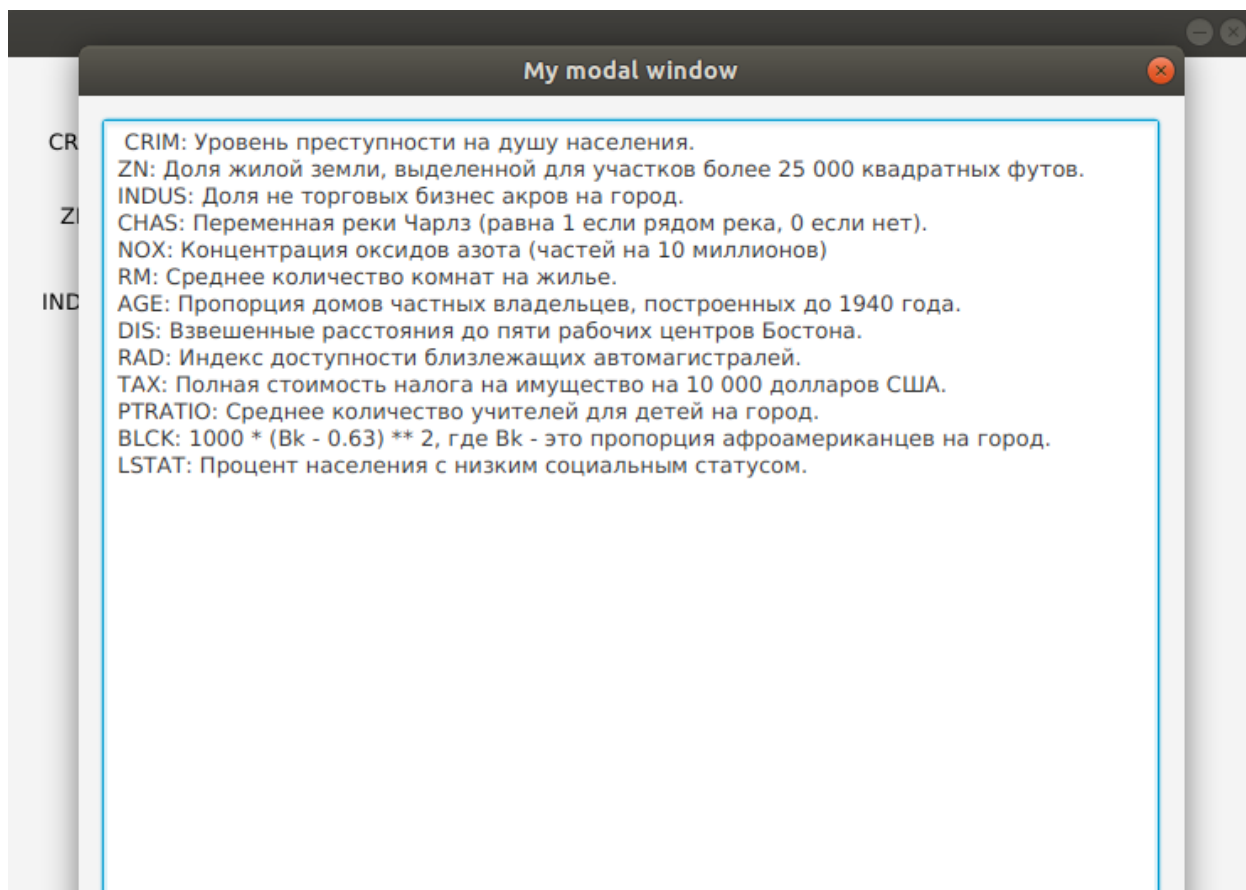


Рисунок 5.7 – Просмотр информации о значениях

С помощью просмотра информации о значениях пользователь может узнать описание значений, чтобы понять какую информацию надо вводить в поля(см. рисунок 5.7).

6.3 Алгоритм работы администратора

При авторизации как администратор, открывается соответствующее окно (см. рисунок 5.8).

id	Логин	Пароль	Статус	Роль
No content in table				

id Логин Пароль Статус Роль

Просмотр пользователей

Изменить Удаление Добавить

Заблокировать Разблокировать Просмотр статистики Проверка пользователя

Рисунок 5.8 – Меню администратора

Администратор может выполнять больше функций, чем пользователь: просматривать всех пользователей(см. рисунок 5.9), изменять информацию о пользователе, удалять пользователя, добавлять пользователя, блокировать пользователя, разблокировать пользователя, просмотреть статистику сервера, просмотреть более подробную информацию о пользователе.

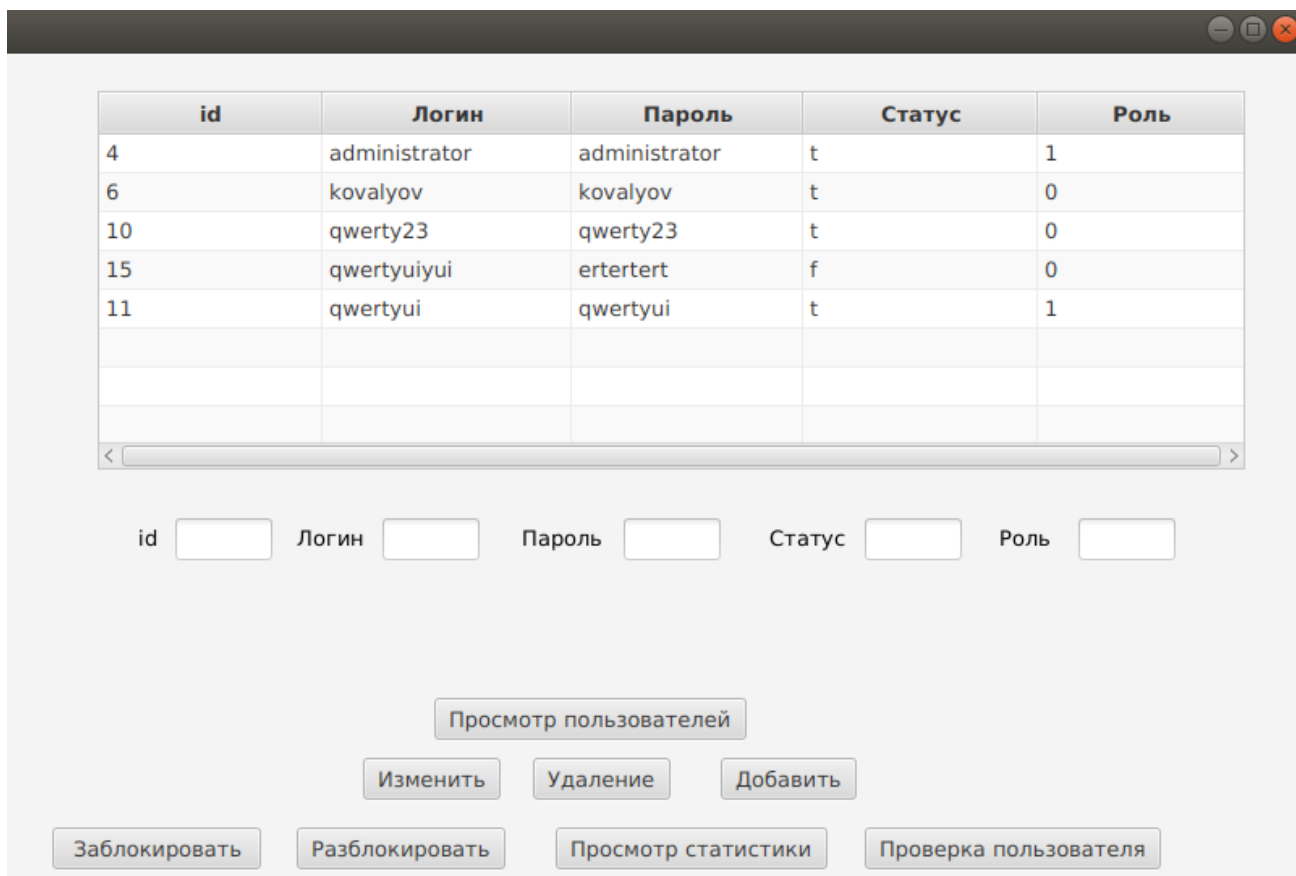


Рисунок 5.9 – Просмотр пользователей

При выполнении функции «просмотреть пользователей»(см. рисунок 5.9), выводится список всех пользователей которые содержатся в базе данных. Предоставляется следующая информация: id пользователя, логин пользователя, пароль пользователя, статус пользователя, роль пользователя.

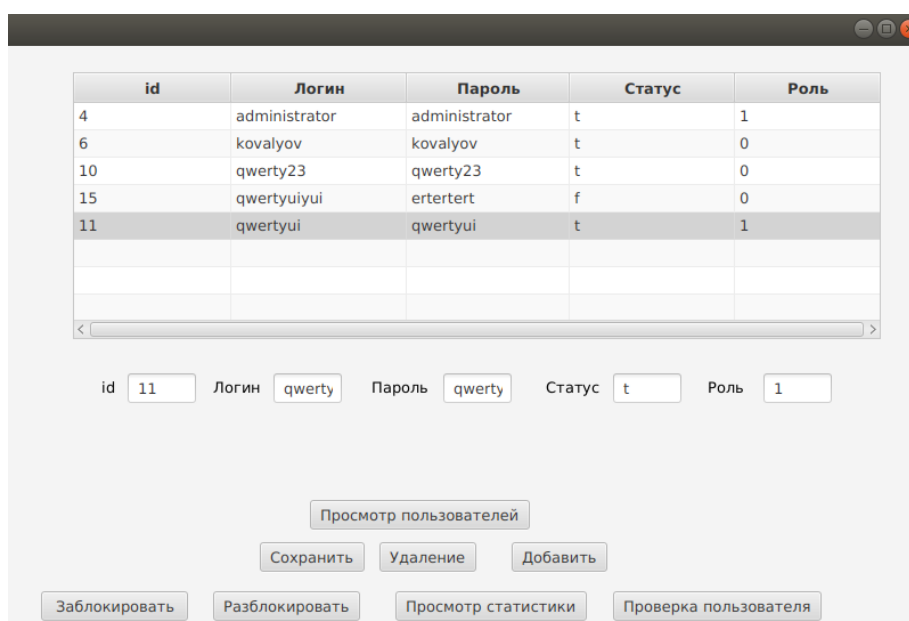


Рисунок 5.10 – Редактирование пользователей

Для функции редактирования(см. рисунок 5.10) требуется, чтобы пользователь был выбран из списка доступных, после нужно нажать на соответствующий элемент на форме, ввести нужные значения, а после нажать на кнопку «Сохранить», после чего опять просмотреть пользователей и убедиться в изменениях.

The screenshot shows a web application window with a table of users and a form below it. The table has five columns: id, Логин, Пароль, Статус, and Роль. Below the table is a form with input fields for each column, and a set of buttons for user management.

id	Логин	Пароль	Статус	Роль
4	administrator	administrator	t	1
6	kovalyov	kovalyov	t	0
10	qwerty23	qwerty23	t	0
11	qwrtyui	qwertyui	t	1

Below the table, the form shows the selected user (id 11) with the following values:

id: 11 Логин: qwrtyui Пароль: qwerty Статус: t Роль: 1

Buttons available:

- Просмотр пользователей
- Изменить
- Удаление
- Добавить
- Заблокировать
- Разблокировать
- Просмотр статистики
- Проверка пользователя

Рисунок 5.11 – Удаление пользователя

Удаления пользователя происходит следующим образом: администратор выбирает пользователя, нажимает на соответствующую кнопку, а затем обновляет список пользователей(см. рисунок 5.11).

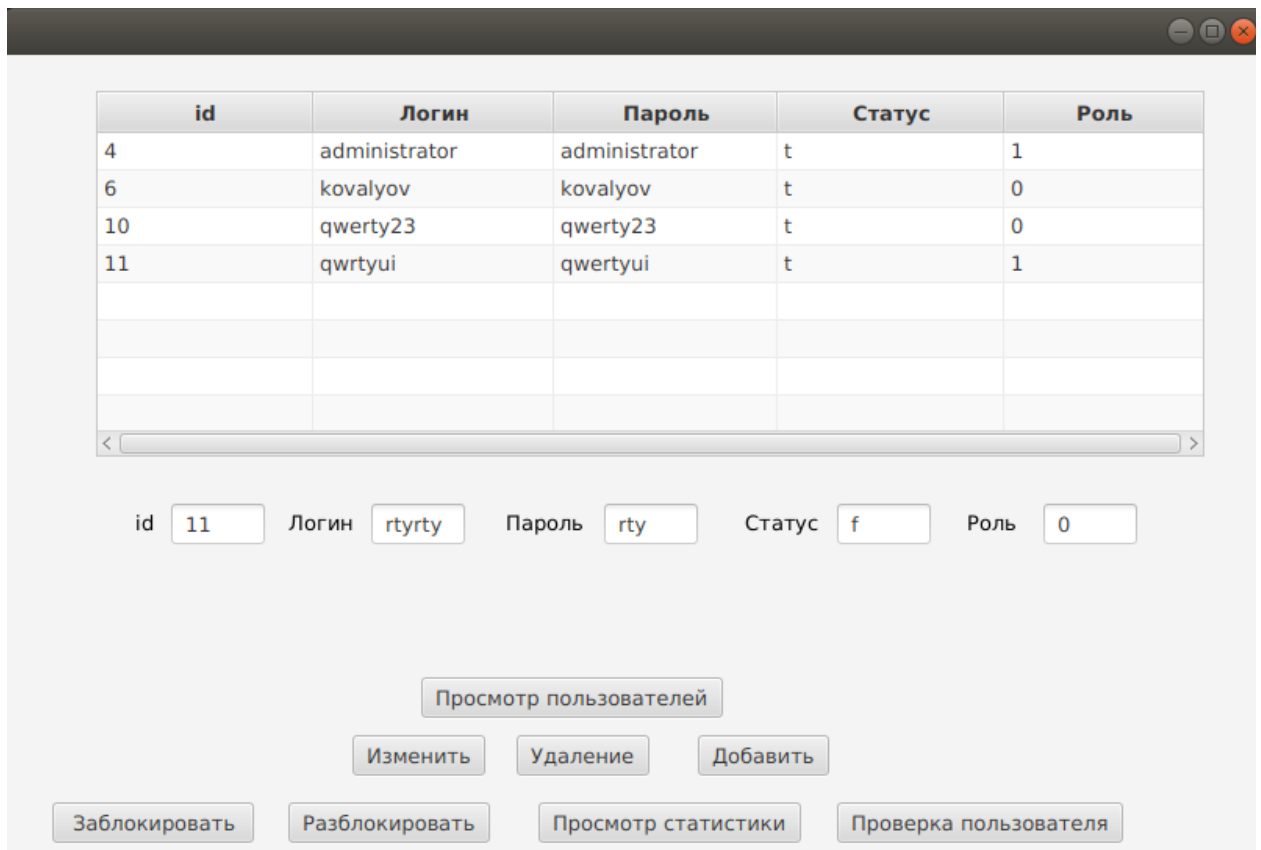


Рисунок 5.12 – Добавление пользователя

Функция добавления пользователя для администратора очень просто реализована просто: администратор вводит новый логин, пароль, устанавливает статус, где «t» – разблокирован, а «f» – заблокирован, выставляет роль где «1» – соответствует администратору, а «0» – обычному пользователю.

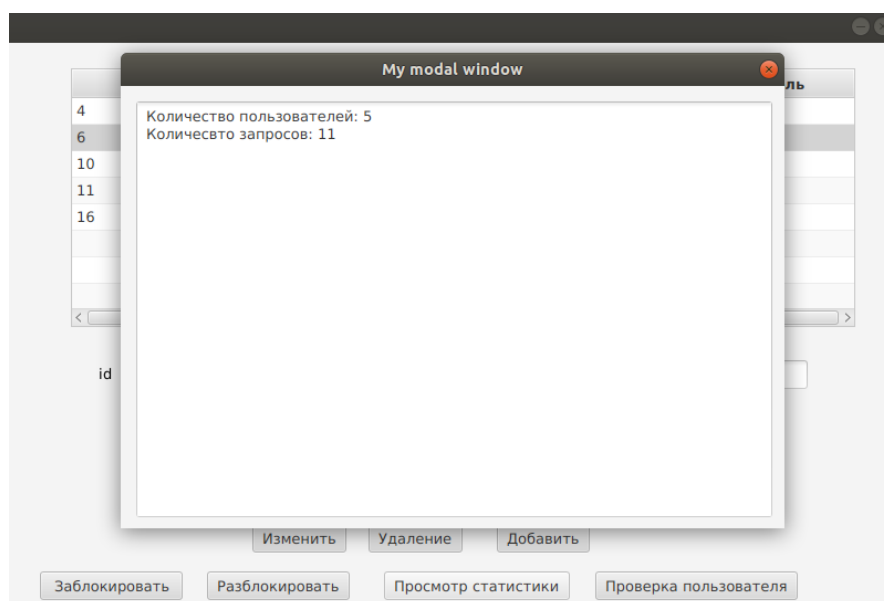


Рисунок 5.13 – Просмотр статистики сервера

Функция просмотра статистики сервера предоставляет статистику по всему серверу: количество пользователей, количество запросов. После нажатия на соответствующий пункт администратору выводится окно с неизменяемым текстовым полем, где написана вся информация(см. рисунок 5.13).

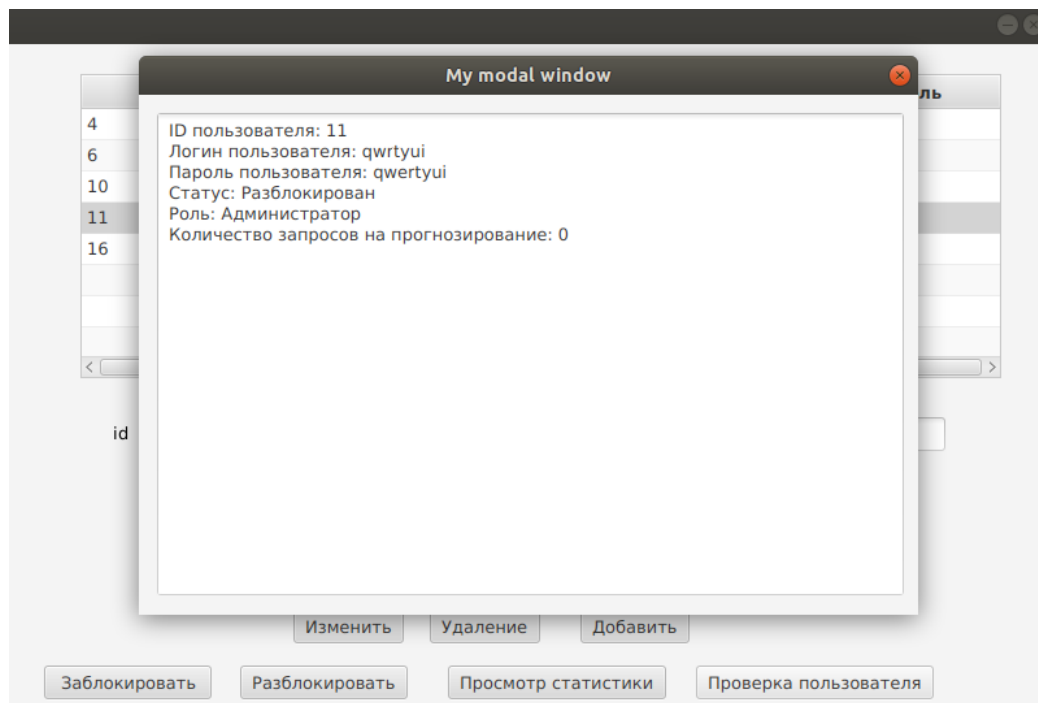


Рисунок 5.14 – Просмотр информации о конкретном пользователе

После выбора пользователя администратор может просмотреть более подробную информацию о пользователе или администраторе(см. рисунок 5.14).

7 АРХИТЕКТУРНЫЙ ШАБЛОН ПРОЕКТИРОВАНИЯ

7.1 Диаграмма вариантов использования

Диаграмма вариантов использования (use case diagram) — диаграмма, на которой изображаются отношения между актерами и вариантами использования.

Назначение данной диаграммы состоит в следующем: проектируемая программная система представляется в форме так называемых вариантов использования, с которыми взаимодействуют внешние сущности или актеры. При этом актером или действующим лицом называется любой объект, субъект или система, взаимодействующая с моделируемой бизнес-системой извне. Это может быть человек, техническое устройство, программа или любая другая система, которая служит источником воздействия на моделируемую систему так, как определит разработчик. Вариант использования служит для описания сервисов, которые система предоставляет актеру. Другими словами, каждый вариант использования определяет набор действий, совершаемый системой при диалоге с актером. При этом ничего не говорится о том, каким образом будет реализовано взаимодействие актеров с системой и собственно выполнение вариантов использования. Рассмотрим диаграмму прецедентов данного курсового проекта. (рисунок 7.1).

В данной диаграмме есть два действующих лица – это пользователь, и администратор.

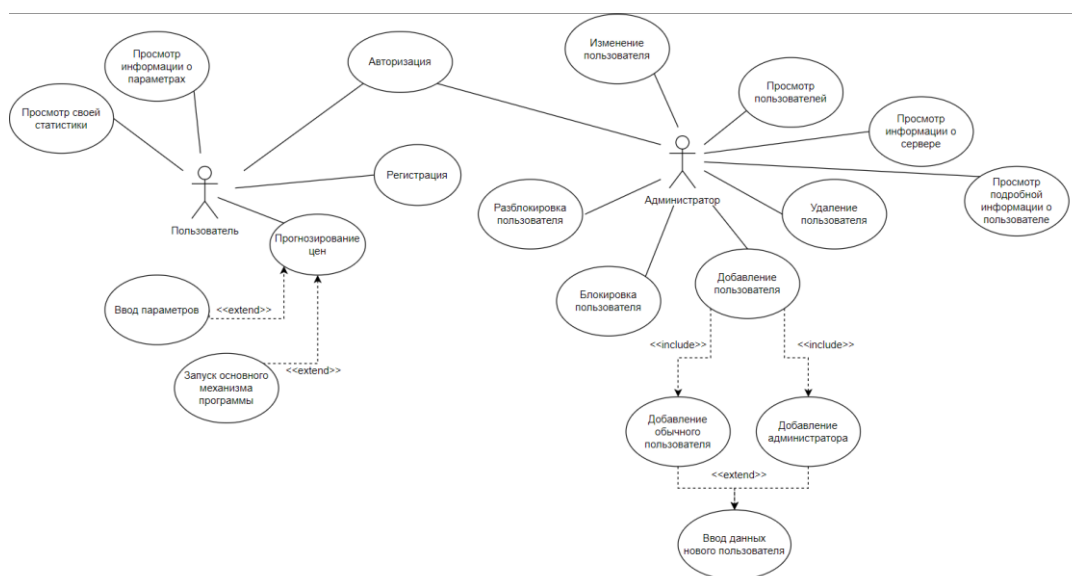


Рисунок 6.1 – Диаграмма вариантов использования

Администратор может: изменять, удалять, добавлять, блокировать и разблокировать, просматривать всех пользователей, может просматривать подробную информацию о пользователе, а так же информацию о сервере.

Возможности пользователя: вводить значения для прогнозирования, запускать основной механизм программного обеспечения, а так же просматривать информацию о своем аккаунте.

7.2 Диаграмма классов

Диаграмма классов определяет типы классов системы и различного рода статические связи, которые существуют между ними. На диаграммах классов изображаются также атрибуты классов, операции классов и ограничения, которые накладываются на связи между классами.

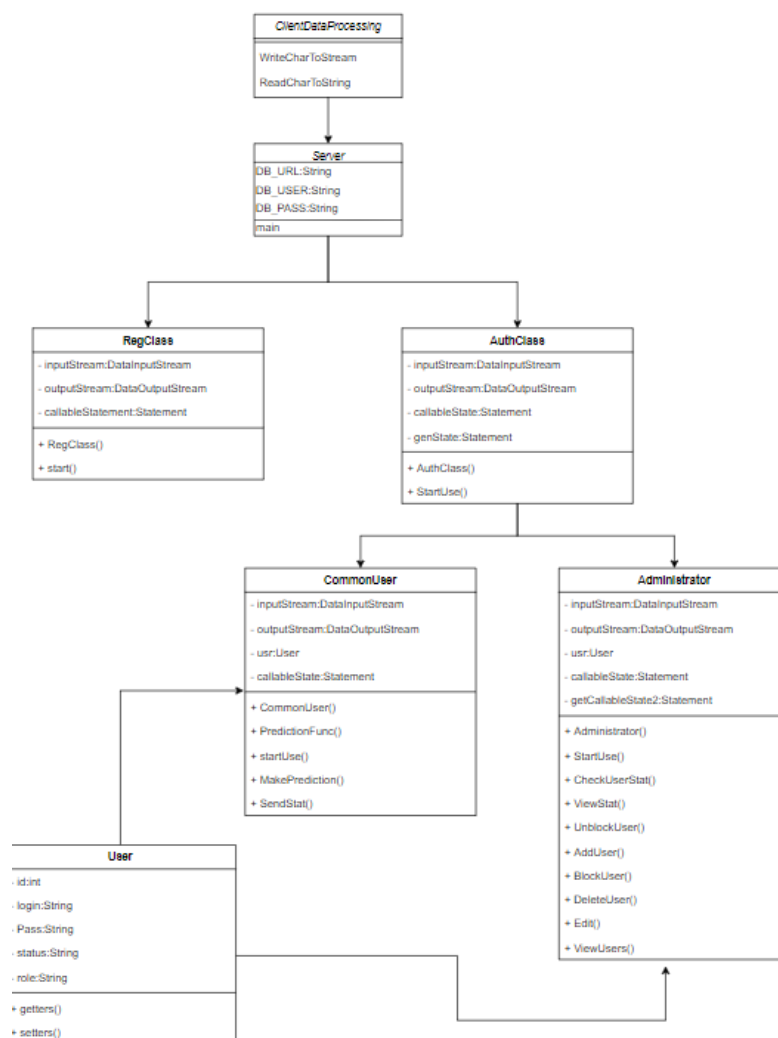


Рисунок 6.2 – Диаграмма классов сервера

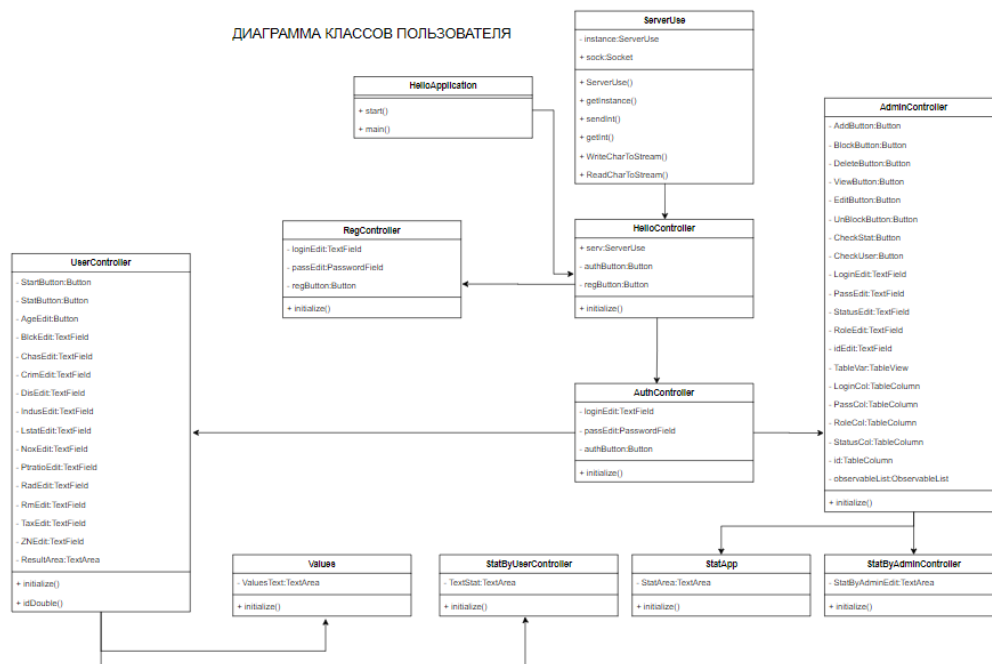


Рисунок 6.3 – Диаграмма классов пользователя

7.3 Диаграмма компонентов

Эта диаграмма позволяет определить архитектуру разрабатываемой системы, установив зависимости между программными компонентами. В качестве таких компонентов могут выступать файлы, библиотеки, модули, исполняемые файлы, пакеты и так далее. Рассмотрим диаграмму компонентов для системы учёта оказания платных услуг поликлиникой, которая изображена на рисунке 6.4.

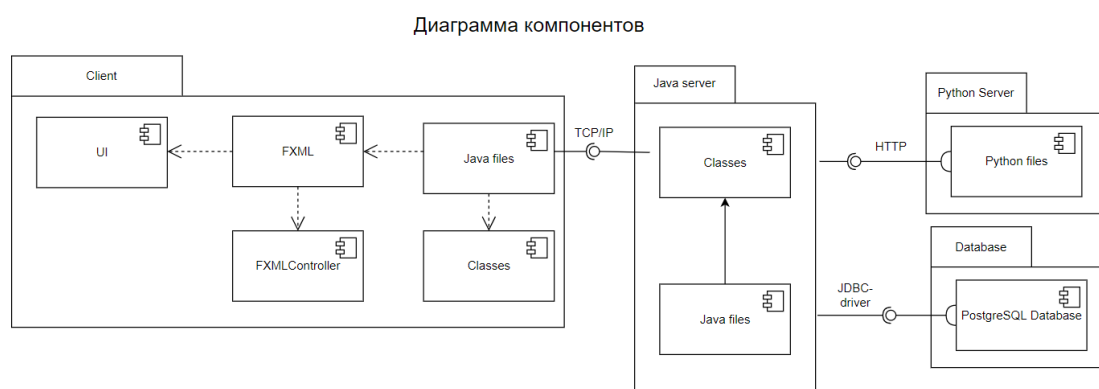


Рисунок 6.4 – Диаграмма компонентов

7.4 Диаграмма последовательности

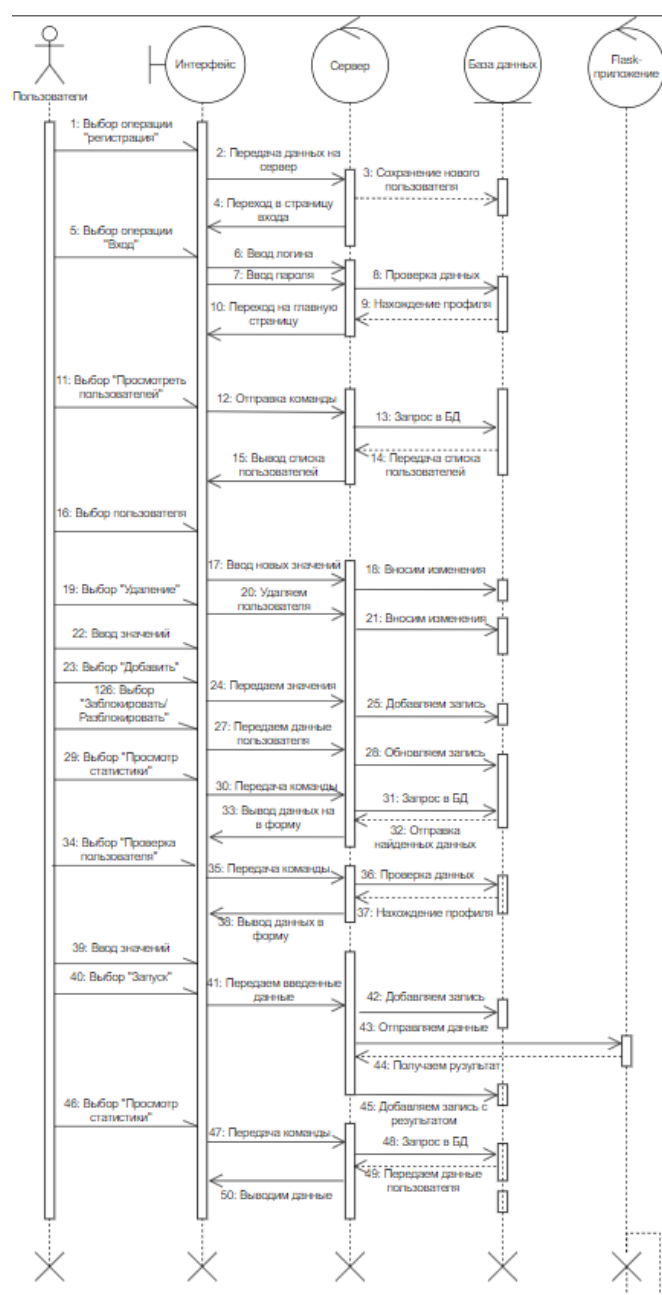


Рисунок 6.5 – Диаграмма последовательности

Эта диаграмма является видом диаграмм взаимодействия, которые описывают отношения объектов в различных условиях. Условия взаимодействия задаются сценарием, полученным на этапе разработки диаграмм вариантов использования. Диаграмма последовательностей показана на рисунке 6.5.

7.5 Диаграмма состояний

Диаграмма состояний является широко известным средством описания поведения систем. Она определяет все возможные состояния, в которых может находиться конкретный объект, а также процесс смены состояний объекта в результате влияния некоторых событий. Данная диаграмма полезна при моделировании жизненного цикла объекта. Диаграмма состояний разрабатываемой системы представлена на рисунке 6.6.

Действия связаны с переходами и рассматриваются, как мгновенные и непрерываемые. Деятельности связаны с состояниями и могут длиться достаточно долго. Деятельность может быть прервана в результате наступления некоторого события.

Из конкретного состояния в данный момент времени может быть осуществлен только один переход; таким образом, условия являются взаимно исключающими для любого события. Существует два особых состояния: вход и выход. Любое действие, связанное с событием входа, выполняется, когда объект входит в данное состояние. Событие выхода выполняется в том случае, когда объект выходит из данного состояния. Диаграммы состояний хорошо использовать для описания поведения некоторого объекта в нескольких различных вариантах использования. Они не слишком пригодны для описания поведения ряда взаимодействующих объектов.

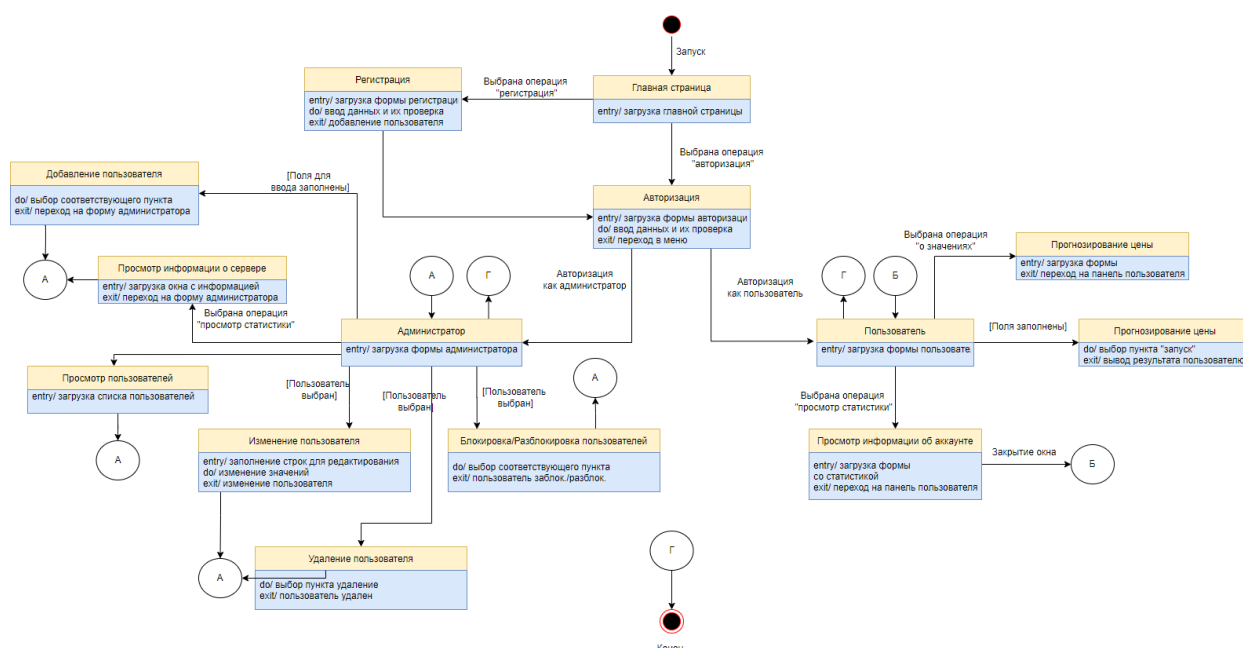


Рисунок 6.6 – Диаграмма состояний

7.6 Диаграмма развертывания

Диаграмма развертывания показывает топологию системы и распределение компонентов системы по её узлам, а также соединения – маршруты передачи информации между аппаратными узлами. На рисунке 6.7 показана диаграмма развёртывания разрабатываемой системы.

Диаграмма развёртывания

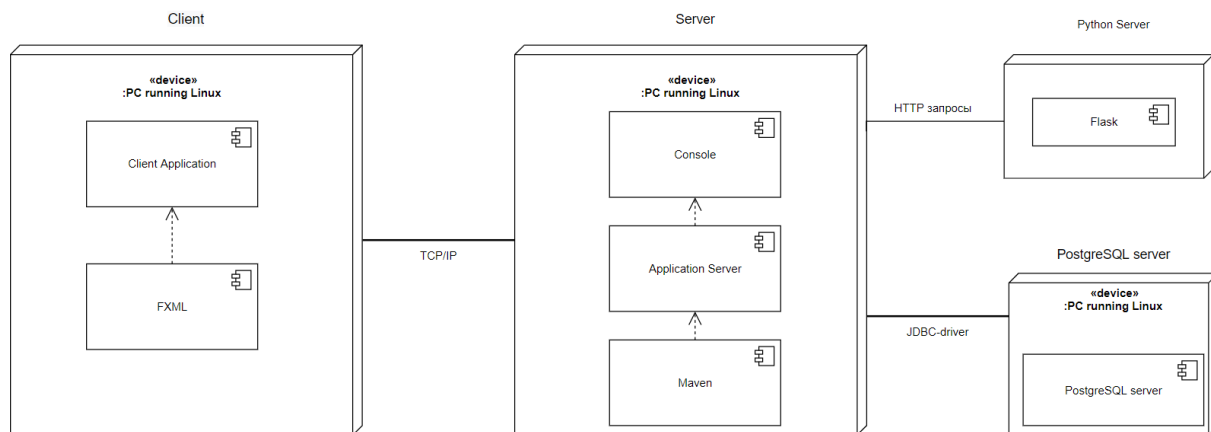


Рисунок 6.7 – Диаграмма развертывания

Как можно заметить, основными элементами являются клиентская и серверная части приложения. Так же присутствуют элементы под названием Python Server и PostgreSQL server.

Взаимодействие между клиентским и серверным узлом осуществляется посредством стека протоколов TCP/IP. Для работы с базой данных используется JDBC-driver. Для соединения с Python Server используется HTTP.

8 РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ РАЗРАБОТАННОЙ СИСТЕМЫ

В процессе работы данного приложения по разным причинам могут возникать различные ошибки. Для устойчивого функционирования необходимо предусмотреть обработку исключительных ситуаций.

Если в окне авторизации или регистрации введен некорректный пароль или логин то высветится ошибка, (рисунок 7.1,7.2).

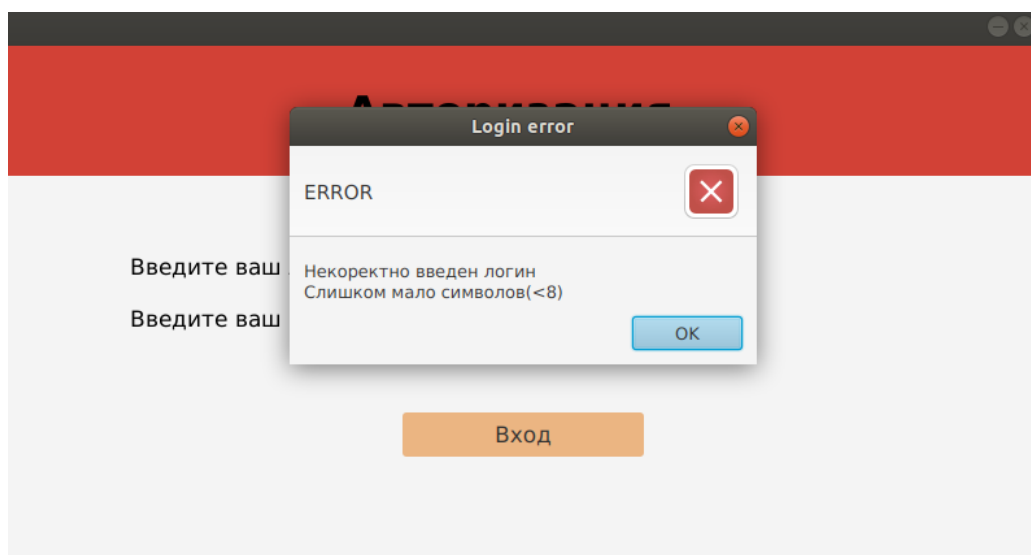


Рисунок 7.1 – Ошибка авторизации. Некорректный логин

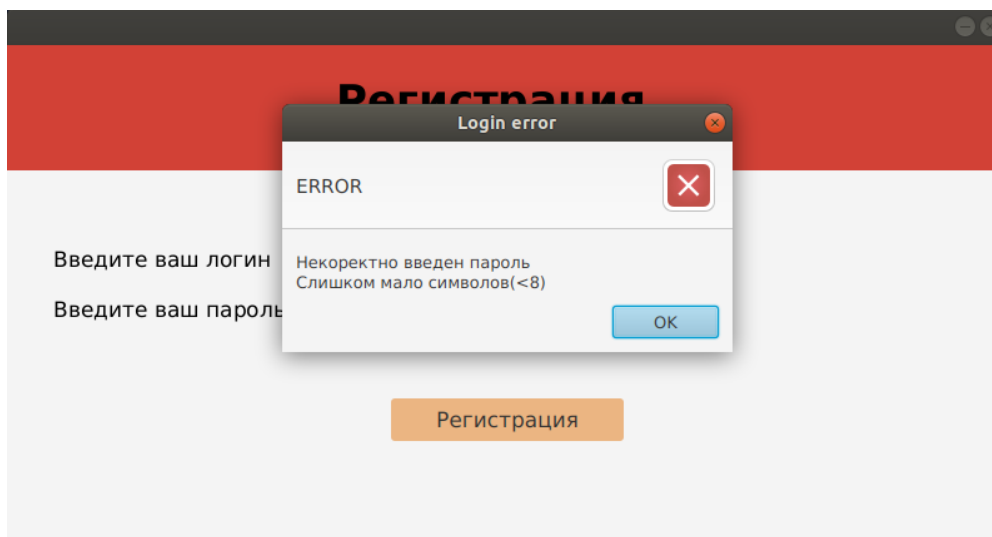


Рисунок 7.2 – Ошибка авторизации. Некорректный пароль

Так же был протестирован основной механизм программного обеспечения, путем введения некорректного числа(см. рисунок 7.3).

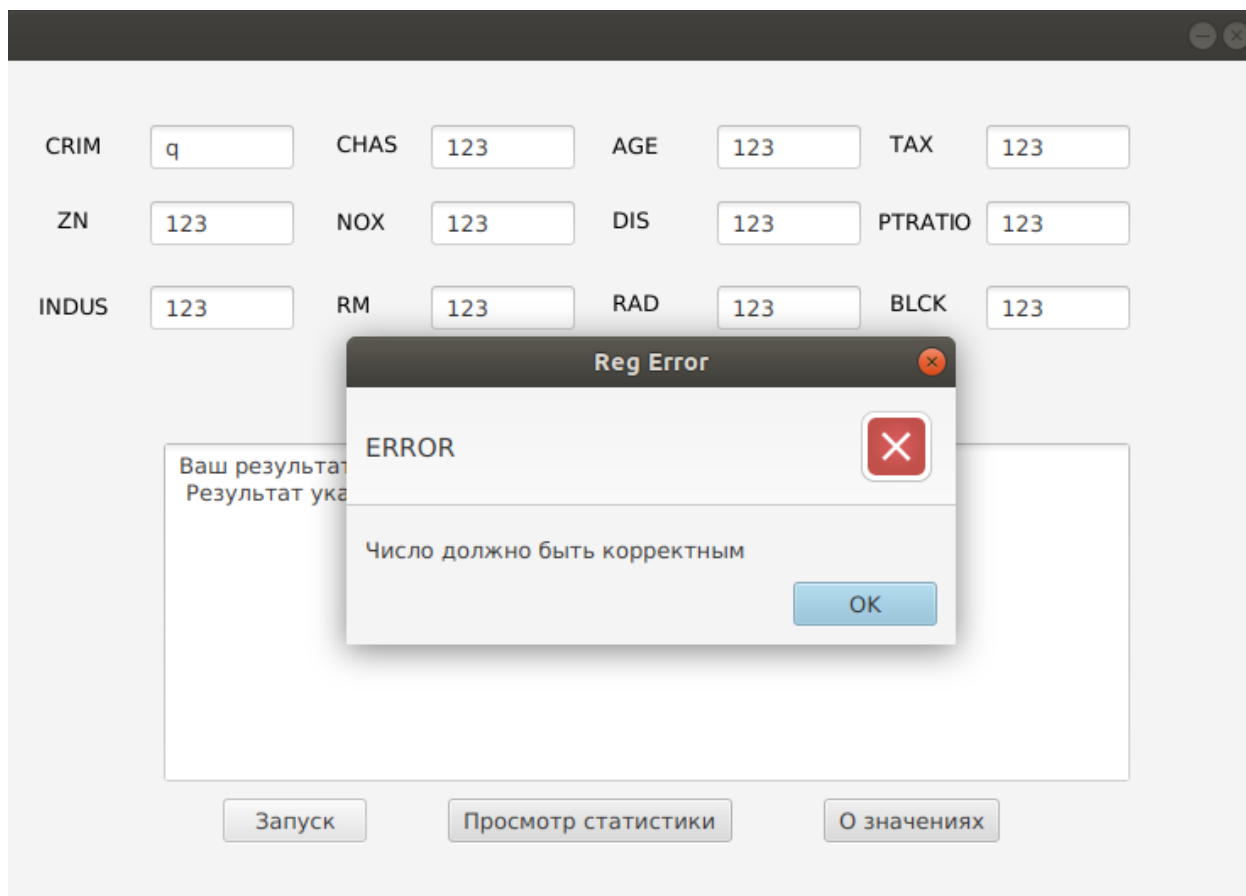


Рисунок 7.3 – Ошибка чтения числа

Таким образом были протестированы основные механизмы программы.

ЗАКЛЮЧЕНИЕ

В курсовой работе была разработана программа прогнозирования .

Программа позволяет пользователю осуществлять регистрацию, авторизацию в ролях администратора и пользователя, редактирование пользователя, удаления, блокировки, есть возможность просматривать всю информацию о пользователях, возможность просматривать подробную информацию о пользователях, пользователь также может информацию о своем аккаунте, а так же использовать основной механизм программного средства.

Был разработан простой и удобный интерфейс, который облегчает пользователю работу в данном приложении.

В процессе разработки были изучены способы работы с JavaFX и PostgreSQL. Также были решены все поставленные в начале разработки задачи.

В ходе выполнения курсовой работы по созданию клиент-серверного приложения прогнозирования цен на жильё на основе нейронных сетей были созданы:

- код программы на языке Java с использованием основных принципов ООП;
- код программы на языке Python
- функциональная и информационные модели;
- диаграммы состояний;
- диаграммы последовательности;
- диаграмма вариантов использования;
- диаграммы классов;
- диаграмма компонентов;
- блок-схемы алгоритмов;
- диаграмма развёртывания.

В будущем возможно рассмотрение вопроса о расширении функционала программы или же усовершенствования имеющегося. Это обеспечит расширение спектра применения разработанного программного средства.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- [1] Глубокие нейросети для проблем регрессии [Электронный ресурс]. – Режим доступа: <https://www.machinelearningmastery.ru/deep-neural-networks-for-regression-problems-81321897ca33/>
- [2] Proglib [Электронный ресурс]. – Режим доступа: <https://proglib.io/p/mashinnoe-obuchenie-dlya-nachinayushchih-algorithm-sluchaynogo-lesa-random-forest-2021-08-12>
- [3] Портал Знаний [Электронный ресурс]. – Режим доступа: <http://statistica.ru/branches-maths/metod-opornykh-vektorov-supported-vector-machine-svm/>
- [4] Машина опорных векторов [Электронный ресурс]. – Режим доступа: <http://www.machinelearning.ru/wiki/index.php?title=SVM>
- [5] Портал Знаний [Электронный ресурс]. – Режим доступа: <http://statistica.ru/theory/osnovy-lineynoy-regressii/>
- [6] Батин, Н.В. Проектирование баз данных / Батин Н.В., Минск: Беларусь -2007. – 56 с.)
- [7] Черемных, С. Моделирование и анализ систем. IDEF-технологии / Черемных С.В., Семенов И.О., Ручкин В.С, Москва: Россия. – 2006. – 192 с.
- [8] UML-диаграммы. Диаграмма последовательности [Электронный ресурс]. – Режим доступа: www.it-gost.ru/articles/view_articles/94
- [9] UML-диаграммы классов: сущности, связи, интерфейсы [Электронный ресурс]. – Режим доступа: <https://prog-cpp.ru/uml-classes>

ПРИЛОЖЕНИЕ А

(рекомендуемое)

Проверка на антиплагиат

Начало проверки: 15.12.2021 00:07:58

Длительность проверки: 00:00:03

Модуль поиска	Заимствования	Самоцитирования	Цитирования	Оригинальность
Интернет Free	5.41%	0%	0%	94.59%

ПРИЛОЖЕНИЕ Б

(обязательное)

Листинг алгоритмов, реализующих бизнес-логику

UserController.java

```
StartButton.setOnAction(actionEvent -> {
    serv.sendInt(1);
    String crim = isDouble(CrimEdit.getText());
    String zn = isDouble(ZNEdit.getText());
    String indus = isDouble(IndusEdit.getText());
    String chas = isDouble(ChasEdit.getText());
    String nox = isDouble(NoxEdit.getText());
    String rm = isDouble(RmEdit.getText());
    String age = isDouble(AgeEdit.getText());
    String dis = isDouble(DisEdit.getText());
    String rad = isDouble(RadEdit.getText());
    String Tax = isDouble(TaxEdit.getText());
    String ptratio = isDouble(PtratioEdit.getText());
    String blk = isDouble(BlckEdit.getText());
    String lstat = isDouble(LstatEdit.getText());
    try {
        serv.WriteCharToStream(crim);
        serv.WriteCharToStream(zn);
        serv.WriteCharToStream(indus);
        serv.WriteCharToStream(chas);
        serv.WriteCharToStream(nox);
        serv.WriteCharToStream(rm);
        serv.WriteCharToStream(age);
        serv.WriteCharToStream(dis);
        serv.WriteCharToStream(rad);
        serv.WriteCharToStream(Tax);
        serv.WriteCharToStream(ptratio);
        serv.WriteCharToStream(blk);
        serv.WriteCharToStream(lstat);

        } catch (IOException e) {
            e.printStackTrace();
        }
        try {
            String resultVar = serv.ReadCharToString();
            ResultArea.setText("Ваш результат: " + resultVar + "." +
"\n Результат указат в тысячах долларов");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
});
```

CommonUser.java

```
public Double PredictionFunc(String string) throws IOException,
ParseException {
    StringBuilder result = new StringBuilder();
    JSONParser parser = new JSONParser();
    URL url = new URL("http://127.0.0.1:5000/helloworld");
    JSONObject senddata = new JSONObject();
    senddata.put("data", string);
    System.out.println(senddata);
    HttpURLConnection conn = (HttpURLConnection) url.openConnection();
```

```

        conn.setRequestMethod("PUT");
        conn.setDoOutput(true);
        OutputStream out = conn.getOutputStream();
        byte[] out1 = senddata.toJSONString().getBytes(StandardCharsets.UTF_8);
        out.write(out1);
        out.close();
        try (BufferedReader reader = new BufferedReader(
            new InputStreamReader(conn.getInputStream()))) {
            for (String line; (line = reader.readLine()) != null; ) {
                result.append(line);
            }
        }
        String res = result.toString();
        JSONObject object = (JSONObject) parser.parse(res);
        Double dat = (Double) object.get("name");
        return dat;
    }

    public void MakePrediction(){
        try {
            ArrayList<String> arrayList = new ArrayList<>();
            StringBuilder stringBuilder = new StringBuilder();
            for (int i = 0; i < 13; i++) {
                arrayList.add(ReadCharToString(inputStream));
                System.out.println(i);
            }
            stringBuilder.append('[');
            for (int i = 0; i < 13; i++) {
                if (i == 12) {
                    stringBuilder.append(arrayList.get(i));
                } else
                    stringBuilder.append(arrayList.get(i) + ",");
            }
            stringBuilder.append("]");
            Double predictionVar = PredictionFunc(stringBuilder.toString());
            arrayList.add(String.valueOf(predictionVar));
            System.out.println(predictionVar);
            System.out.println(arrayList);
            WriteCharToStream(predictionVar.toString(), outputStream);
            callableState.executeUpdate(String.format("insert into
\"data1\"(user_id,crim,zn,indus,chas) values (%d,%s,%s,%s,%s);", usr.getId(),
arrayList.get(0), arrayList.get(1), arrayList.get(2), arrayList.get(3)));
            callableState.executeUpdate(String.format("insert into
\"data2\"(user_id,nox,rm,age,dis) values (%d,%s,%s,%s,%s);", usr.getId(),
arrayList.get(4), arrayList.get(5), arrayList.get(6), arrayList.get(7)));
            callableState.executeUpdate(String.format("insert into
\"data3\"(user_id,rad,tax,ptration,blk,lstat) values (%d,%s,%s,%s,%s,%s);",
usr.getId(), arrayList.get(8), arrayList.get(9), arrayList.get(10),
arrayList.get(11), arrayList.get(12)));
            callableState.executeUpdate(String.format("insert into
\"result\"(user_id,predict) values (%d,%s);", usr.getId(),
arrayList.get(13)));
        } catch (IOException | ParseException | SQLException e) {
            e.printStackTrace();
        }
    }
}

FlaskServer
from flask import Flask, request

```

```

import json
from flask_restful import Api, Resource
import keras
import numpy as np
from keras.datasets import boston_housing

app = Flask(__name__)
api = Api(app)

class HelloWorld(Resource):
    def put(self):
        my_json = request.get_data().decode('utf8')
        data_raw = json.loads(my_json)
        data = data_raw.get("data")
        data = data.replace('[', ' ')
        data = data.replace(']', ' ')
        data = data.replace(']', ' ')
        print(data)
        data = data.split(',')
        print(data)
        data = np.asarray(data, dtype=np.float64)
        (train, train_target), (test, test_target) =
boston_housing.load_data()
        mean = train.mean(axis=0)
        std = train.std(axis=0)
        model = keras.models.load_model('model.h5')
        data -= mean
        data /= std
        data = data.reshape(1,13)
        print(data)
        prediction = model.predict(data)
        prediction = prediction.astype(float)
        prediction = prediction[0][0]
        return {"name": prediction}

api.add_resource(HelloWorld, "/helloworld")
if __name__ == "__main__":
    app.run(debug=True)

```

ПРИЛОЖЕНИЕ В

(обязательное)

Листинг основных элементов программы

```
Auth
package AuthReg;

import java.io.IOException;
import java.net.Socket;
import java.net.URL;
import java.util.ResourceBundle;

import com.example.client.HelloController;
import com.example.client.ServerUse;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Alert;
import javafx.scene.control.Button;
import javafx.scene.control.PasswordField;
import javafx.scene.control.TextField;
import javafx.stage.Stage;

public class AuthController extends HelloController {

    @FXML
    private ResourceBundle resources;

    @FXML
    private URL location;

    @FXML
    private TextField loginEdit;

    @FXML
    private PasswordField passEdit;

    @FXML
    private Button authButton;

    @FXML
    void initialize() {
        authButton.setOnAction(actionEvent -> {
            Integer userRole;
            String UserStatus;
            String login = loginEdit.getText();
            String pass = passEdit.getText();
            if (login.length() < 8) {
                Alert alert = new Alert(Alert.AlertType.ERROR);
                alert.setTitle("Login error");
                alert.setHeaderText("ERROR");
                alert.setContentText("Некоректно введен логин\n" + "Слишком
мало символов(<8)");
                alert.show();
                loginEdit.setText("");
            }
            return;
        });
    }
}
```

```

    }
    if (pass.length() < 8) {
        Alert alert = new Alert(Alert.AlertType.ERROR);
        alert.setTitle("Login error");
        alert.setHeaderText("ERROR");
        alert.setContentText("Некоректно введен пароль\n" + "Слишком
мало символов(<8)");
        alert.show();
        passEdit.setText("");
        return;
    }
    serv.sendInt(2);
    try {
        serv.WriteCharToStream(login);
        serv.WriteCharToStream(pass);
    } catch (IOException e) {
        e.printStackTrace();
    }
    try {
        UserStatus = serv.ReadCharToString();
        String UserRole = serv.ReadCharToString();
        System.out.println(UserStatus);
        if (UserStatus.equals("f")) {
            Alert alert = new Alert(Alert.AlertType.ERROR);
            alert.setTitle("Auth error");
            alert.setHeaderText("ERROR");
            alert.setContentText("Данный пользователь заблокирован");
            alert.show();
        } else if (UserStatus.equals("error")) {
            Alert alert = new Alert(Alert.AlertType.ERROR);
            alert.setTitle("Auth error");
            alert.setHeaderText("ERROR");
            alert.setContentText("Такого пользователя не
существует");
            alert.show();
        } else {
            if (UserRole.equals("0")) {
                authButton.getScene().getWindow().hide();
                FXXMLLoader loader = new FXXMLLoader();

                loader.setLocation(getClass().getResource("/com/example/client/User.fxml"));
                try {
                    loader.load();
                } catch (IOException e) {
                    e.printStackTrace();
                }
                Parent root = loader.getRoot();
                Stage stage = new Stage();
                stage.setScene(new Scene(root));
                stage.show();
            }
            if (UserRole.equals("1")) {
                authButton.getScene().getWindow().hide();
                FXXMLLoader loader = new FXXMLLoader();

                loader.setLocation(getClass().getResource("/com/example/client/Admin.fxml"));
                try {
                    loader.load();
                } catch (IOException e) {

```

```

        e.printStackTrace();
    }
    Parent root = loader.getRoot();
    Stage stage = new Stage();
    stage.setScene(new Scene(root));
    stage.show();
}
}
//        userRole = serv.getInt();
    } catch (IOException e) {
        e.printStackTrace();
    }
    });
}

}

REG
package AuthReg;

import java.io.IOException;
import java.net.URL;
import java.util.ResourceBundle;

import com.example.client.HelloController;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Alert;
import javafx.scene.control.Button;
import javafx.scene.control.PasswordField;
import javafx.scene.control.TextField;
import javafx.stage.Stage;

public class RegistrationController extends HelloController {
    @FXML
    private ResourceBundle resources;

    @FXML
    private URL location;

    @FXML
    private TextField loginEdit;

    @FXML
    private PasswordField passEdit;

    @FXML
    private Button regButton;

    @FXML
    void initialize() {
        regButton.setOnAction(actionEvent -> {
            if (loginEdit.getText().length() < 8) {
                Alert alert = new Alert(Alert.AlertType.ERROR);
                alert.setTitle("Login error");
                alert.setHeaderText("ERROR");
            }
        });
    }
}

```

```

        alert.setContentText("Некоректно введен логин\n" + "Слишком
мало символов(<8)");
        alert.show();
        loginEdit.setText("");
        return;
    }
    if (passEdit.getText().length() < 8) {
        Alert alert = new Alert(Alert.AlertType.ERROR);
        alert.setTitle("Login error");
        alert.setHeaderText("ERROR");
        alert.setContentText("Некоректно введен пароль\n" + "Слишком
мало символов(<8)");
        alert.show();
        passEdit.setText("");
        return;
    }
    String login = loginEdit.getText();
    String pass = passEdit.getText();
    System.out.println(login);
    System.out.println(pass);
    serv.sendInt(1);
    try {
        serv.WriteCharToStream(login);
        serv.WriteCharToStream(pass);
        if (serv.getInt() == 0){
            Alert alert = new Alert(Alert.AlertType.ERROR);
            alert.setTitle("Reg Error");
            alert.setHeaderText("ERROR");
            alert.setContentText("Такой      пользователь      уже
существует");
            alert.show();
            passEdit.setText("");
            loginEdit.setText("");
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
    regButton.getScene().getWindow().hide();
    FXMLLoader loader = new FXMLLoader();

    loader.setLocation(getClass().getResource("/com/example/client/Auth.fxml"));
    try {
        loader.load();
    } catch (IOException e) {
        e.printStackTrace();
    }
    Parent root = loader.getRoot();
    Stage stage = new Stage();
    stage.setScene(new Scene(root));
    stage.show();
    });
}

}

Admin
package UserAdmin;

```

```

import java.io.DataOutputStream;
import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.ResourceBundle;

import com.example.client.HelloController;
import com.example.client.User;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Node;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.stage.Modality;
import javafx.stage.Stage;

public class AdminController extends HelloController {

    @FXML
    private ResourceBundle resources;

    @FXML
    private URL location;

    @FXML
    private Button AddButton;

    @FXML
    private Button BlockButton;

    @FXML
    private Button DeleteButton;

    @FXML
    private Button EditButton;

    @FXML
    private TableColumn<User, String> LoginCol;

    @FXML
    private TextField LoginEdit;

    @FXML
    private TableColumn<User, String> PassCol;

    @FXML
    private TextField PassEdit;

    @FXML
    private TableColumn<User, String> RoleCol;

    @FXML
    private TextField RoleEdit;

    @FXML

```



```

private TableColumn<User, String> StatusCol;
@FXML
private TableColumn<User, Integer> id;
@FXML
private TextField StatusEdit;

@FXML
private TableView<User> TableVar;

@FXML
private Button UnblockButton;

@FXML
private Button ViewButton;

@FXML
private Button CheckStat;

@FXML
private Button CheckUser;

@FXML
private TextField idEdit;
ObservableList<User> observableList =
FXCollections.observableArrayList();

@FXML
void initialize() {
    ViewButton.setOnAction(actionEvent -> {
        observableList.clear();
        serv.sendInt(1);
        System.out.println(123);
        Integer count = serv.getInt();
        for (int i = 0; i < count; i++) {
            try {
                Integer tempID =
Integer.parseInt(serv.ReadCharToString());
                String tempLogin = serv.ReadCharToString();
                String tempPass = serv.ReadCharToString();
                String tempStatus = serv.ReadCharToString();
                String tempRole = serv.ReadCharToString();
                observableList.add(new User(tempID, tempLogin, tempPass,
tempStatus, tempRole));

            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        id.setCellValueFactory(new PropertyValueFactory<User,
Integer>("id"));
        LoginCol.setCellValueFactory(new PropertyValueFactory<User,
String>("login"));
        PassCol.setCellValueFactory(new PropertyValueFactory<User,
String>("Pass"));
        StatusCol.setCellValueFactory(new PropertyValueFactory<User,
String>("status"));
        RoleCol.setCellValueFactory(new PropertyValueFactory<User,
String>("role"));
        TableVar.setItems(observableList);
    });
}

```

```

});

DeleteButton.setOnAction(actionEvent -> {
    serv.sendInt(3);
    User deleteData = TableVar.getSelectionModel().getSelectedItem();
    serv.sendInt(deleteData.getId());
});

BlockButton.setOnAction(actionEvent -> {
    serv.sendInt(5);
    User BlockData = TableVar.getSelectionModel().getSelectedItem();
    serv.sendInt(BlockData.getId());
});

UnBlockButton.setOnAction(actionEvent -> {
    serv.sendInt(6);
    User                                UnBlockData                                =
TableVar.getSelectionModel().getSelectedItem();
    serv.sendInt(UnBlockData.getId());
});

EditButton.setOnAction(actionEvent -> {
    User EditData;
    if (EditButton.getText().equals("Изменить")) {
        serv.sendInt(2);
        EditData = TableVar.getSelectionModel().getSelectedItem();
        try {
            boolean b = EditData.getId() == (int) EditData.getId();
            idEdit.setText(String.valueOf(EditData.getId()));
            LoginEdit.setText(EditData.getLogin());
            PassEdit.setText(EditData.getPass());
            RoleEdit.setText(EditData.getRole());
            StatusEdit.setText(EditData.getStatus());
            EditButton.setText("Сохранить");
        } catch (NullPointerException e) {
            Alert alert = new Alert(Alert.AlertType.ERROR);
            alert.setTitle("Ошибка");
            alert.setHeaderText("Ошибка");
            alert.setContentText("Вы не выбрали пользователя");
            alert.show();
        }
    } else {
        EditButton.setText("Изменить");
        String IdVar = idEdit.getText();
        System.out.println(idEdit.getText());
        String LoginVar = LoginEdit.getText();
        String PassVar = PassEdit.getText();
        String StatusVar = StatusEdit.getText();
        String RoleVar = RoleEdit.getText();
        try {
            serv.WriteCharToStream(IdVar);
            serv.WriteCharToStream(LoginVar);
            serv.WriteCharToStream(PassVar);
            serv.WriteCharToStream(StatusVar);
            serv.WriteCharToStream(RoleVar);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

```

```

    }
}
});

AddButton.setOnAction(actionEvent -> {
    if (LoginEdit.getText().length() != 0) {
        if (PassEdit.getText().length() != 0) {
            if (StatusEdit.getText().length() != 0) {
                if (RoleEdit.getText().length() != 0) {
                    try {

                        serv.sendInt(4);
                        serv.WriteCharToStream(LoginEdit.getText());
                        serv.WriteCharToStream(PassEdit.getText());
                        serv.WriteCharToStream(StatusEdit.getText());
                        serv.WriteCharToStream(RoleEdit.getText());
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                }
            }
        }
    } else {
        Alert alert = new Alert(Alert.AlertType.INFORMATION);
        alert.setTitle("Information Dialog");
        alert.setHeaderText(null);
        alert.setContentText("Error");

        alert.showAndWait();
    }
});

CheckStat.setOnAction(actionEvent -> {
    serv.sendInt(7);
    ArrayList<String> arrayList = new ArrayList<>();
    Stage stage = new Stage();
    Parent root = null;
    FXMLLoader fxmlLoader = new FXMLLoader();

    fxmlLoader.setLocation(getClass().getResource("/com/example/client/StatApp.fxml"));

    try {
        fxmlLoader.load();
    } catch (IOException e) {
        e.printStackTrace();
    }
    root = fxmlLoader.getRoot();
    stage.setScene(new Scene(root));
    stage.setTitle("My modal window");
    stage.initModality(Modality.WINDOW_MODAL);
    stage.initOwner(
        ((Node)actionEvent.getSource()).getScene().getWindow() );
    stage.show();
});

CheckUser.setOnAction(actionEvent -> {
    serv.sendInt(8);
    User usr;
    usr = TableVar.getSelectionModel().getSelectedItem();
    serv.sendInt(usr.getId());
    Stage stage = new Stage();

```

```

        Parent root = null;
        FXMLLoader fxmlloader = new FXMLLoader();

fxmlloader.setLocation(getClass().getResource("/com/example/client/StatByAdmin.fxml"));
        try {
            fxmlloader.load();
        } catch (IOException e) {
            e.printStackTrace();
        }
        root = fxmlloader.getRoot();
        stage.setScene(new Scene(root));
        stage.setTitle("My modal window");
        stage.initModality(Modality.WINDOW_MODAL);
        stage.initOwner(
            ((Node)actionEvent.getSource()).getScene().getWindow() );
        stage.show();
    });

}

}

package UserAdmin;

import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.ResourceBundle;

import ModalControllers.StatByUserController;
import com.example.client.HelloController;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Node;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Alert;
import javafx.scene.control.Button;
import javafx.scene.control.TextArea;
import javafx.scene.control.TextField;
import javafx.stage.Modality;
import javafx.stage.Stage;

public class UserController extends HelloController {

    @FXML
    private ResourceBundle resources;

    @FXML
    private URL location;

    @FXML
    private TextField AgeEdit;

    @FXML
    private TextField BlckEdit;

    @FXML
    private TextField ChasEdit;

```

```

@FXML
private TextField CrimEdit;

@FXML
private TextField DisEdit;

@FXML
private TextField IndusEdit;

@FXML
private TextField LstatEdit;

@FXML
private TextField NoxEdit;

@FXML
private TextField PtratioEdit;

@FXML
private TextField RadEdit;

@FXML
private TextArea ResultArea;

@FXML
private TextField RmEdit;

@FXML
private Button StartButton;

@FXML
private TextField TaxEdit;

@FXML
private Button StatButton;

@FXML
private TextField ZNEdit;

@FXML
private Button ValuesButton;
@FXML
void initialize() {
    StartButton.setOnAction(actionEvent -> {
        serv.sendInt(1);
        String crim = isDouble(CrimEdit.getText());
        String zn = isDouble(ZNEdit.getText());
        String indus = isDouble(IndusEdit.getText());
        String chas = isDouble(ChasEdit.getText());
        String nox = isDouble(NoxEdit.getText());
        String rm = isDouble(RmEdit.getText());
        String age = isDouble(AgeEdit.getText());
        String dis = isDouble(DisEdit.getText());
        String rad = isDouble(RadEdit.getText());
        String Tax = isDouble(TaxEdit.getText());
        String ptratio = isDouble(PtratioEdit.getText());
        String blk = isDouble(BlckEdit.getText());
        String lstat = isDouble(LstatEdit.getText());
    });
}

```

```

try {
    serv.WriteCharToStream(crim);
    serv.WriteCharToStream(zn);
    serv.WriteCharToStream(indus);
    serv.WriteCharToStream(chas);
    serv.WriteCharToStream(nox);
    serv.WriteCharToStream(rm);
    serv.WriteCharToStream(age);
    serv.WriteCharToStream(dis);
    serv.WriteCharToStream(rad);
    serv.WriteCharToStream(Tax);
    serv.WriteCharToStream(ptratio);
    serv.WriteCharToStream(blck);
    serv.WriteCharToStream(lstat);

    } catch (IOException e) {
        e.printStackTrace();
    }
    try {
        String resultVar = serv.ReadCharToString();
        ResultArea.setText("Ваш результат: " + resultVar + "." +
"\n Результат указат в тысячах долларов");
    } catch (IOException e) {
        e.printStackTrace();
    }
});

StatButton.setOnAction(actionEvent -> {
    System.out.println("DONE");
    serv.sendInt(2);
    ArrayList<String> arrayList = new ArrayList<>();
    Stage stage = new Stage();
    Parent root = null;
    FXXMLLoader fxmLoader = new FXXMLLoader();

fxmLoader.setLocation(getClass().getResource("/com/example/client/StatByUser
.fxml"));

    try {
        fxmLoader.load();
    } catch (IOException e) {
        e.printStackTrace();
    }
    root = fxmLoader.getRoot();
    stage.setScene(new Scene(root));
    stage.setTitle("My modal window");
    stage.initModality(Modality.WINDOW_MODAL);
    stage.initOwner(

((Node) actionEvent.getSource()).getScene().getWindow() );
    stage.show();
});
ValuesButton.setOnAction(actionEvent -> {
    Stage stage = new Stage();
    Parent root = null;
    FXXMLLoader fxmLoader = new FXXMLLoader();

fxmLoader.setLocation(getClass().getResource("/com/example/client/Values.fxm
l"));

    try {

```

```

        fxmlloader.load();
    } catch (IOException e) {
        e.printStackTrace();
    }
    root = fxmlloader.getRoot();
    stage.setScene(new Scene(root));
    stage.setTitle("My modal window");
    stage.initModality(Modality.WINDOW_MODAL);
    stage.initOwner(

((Node)actionEvent.getSource()).getScene().getWindow() );
    stage.show();
    });
}

public String isDouble(String str){
    try{
        Double doubl = Double.parseDouble(str);
        return str;
    }catch (Exception e){
        Alert alert = new Alert(Alert.AlertType.ERROR);
        alert.setTitle("Reg Error");
        alert.setHeaderText("ERROR");
        alert.setContentText("Число должно быть корректным \nНеправильное
число считается равное нулю");
        alert.show();
        return "0";
    }
}

}

```

ПРИЛОЖЕНИЕ Г

(обязательное)

Листинг скрипта генерации базы данных

```
CREATE DATABASE kursovaya
WITH
  OWNER = admin
  ENCODING = 'UTF8'
  LC_COLLATE = 'ru_RU.UTF-8'
  LC_CTYPE = 'ru_RU.UTF-8'
  TABLESPACE = pg_default
  CONNECTION LIMIT = -1;
CREATE TABLE IF NOT EXISTS public.data1
(
  id integer NOT NULL GENERATED ALWAYS AS IDENTITY ( INCREMENT 1 START 1
MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),
  user_id integer,
  crim double precision,
  zn double precision,
  indus double precision,
  chas double precision,
  CONSTRAINT data1_pkey PRIMARY KEY (id),
  CONSTRAINT fk_user_id FOREIGN KEY (user_id)
    REFERENCES public."user" (id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
)
WITH (
  OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE IF EXISTS public.data1
  OWNER to admin;
CREATE TABLE IF NOT EXISTS public.data2
(
  id integer NOT NULL GENERATED ALWAYS AS IDENTITY ( INCREMENT 1 START 1
MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),
  user_id integer,
  nox double precision,
  rm double precision,
  age double precision,
  dis double precision,
  CONSTRAINT data2_pkey PRIMARY KEY (id),
  CONSTRAINT fk_user_id FOREIGN KEY (user_id)
    REFERENCES public."user" (id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
)
WITH (
  OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE IF EXISTS public.data2
  OWNER to admin;
CREATE TABLE IF NOT EXISTS public.data3
(
```



```

        id integer NOT NULL GENERATED ALWAYS AS IDENTITY ( INCREMENT 1 START 1
MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),
        user_id integer,
        rad double precision,
        tax double precision,
        ptration double precision,
        blk double precision,
        lstat double precision,
        CONSTRAINT data3_pkey PRIMARY KEY (id),
        CONSTRAINT fk_user_id FOREIGN KEY (user_id)
            REFERENCES public."user" (id) MATCH SIMPLE
            ON UPDATE NO ACTION
            ON DELETE NO ACTION
    )
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE IF EXISTS public.data3
    OWNER to admin;
CREATE TABLE IF NOT EXISTS public.result
(
    id integer NOT NULL GENERATED ALWAYS AS IDENTITY ( INCREMENT 1 START 1
MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),
    user_id integer,
    predict double precision,
    CONSTRAINT result_pkey PRIMARY KEY (id),
    CONSTRAINT fk_user_id FOREIGN KEY (user_id)
        REFERENCES public."user" (id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE IF EXISTS public.result
    OWNER to admin;
CREATE TABLE IF NOT EXISTS public."user"
(
    id integer NOT NULL GENERATED ALWAYS AS IDENTITY ( INCREMENT 1 START 1
MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),
    log character varying(100) COLLATE pg_catalog."default" NOT NULL,
    pass character varying(100) COLLATE pg_catalog."default" NOT NULL,
    status boolean NOT NULL,
    role integer NOT NULL,
    CONSTRAINT "User_pkey" PRIMARY KEY (id)
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE IF EXISTS public."user"
    OWNER to admin;
ALTER TABLE IF EXISTS public."user"
    ALTER COLUMN log SET STATISTICS 1;

```