

Министерство науки и высшего образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

по курсу «Программирование на языке Java»

на тему «Разработка программы для построения графиков функции»

21.05.26
хорошо
[подпись]

Выполнил:

Студент группы 23ВВИ1

Пыльнов Н.Н.

Приняли:

к.т.н. доцент Юрова О.В.

к.т.н. доцент Карамышева Н.С.

Пенза 2026

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Факультет Вычислительной техники
Кафедра "Вычислительная техника"

"УТВЕРЖДАЮ"

Зав. кафедрой ВТ

профессор М. А. Митрохин

« » 20 г

ЗАДАНИЕ

на курсовое проектирование по курсу

Разработка кроссплатформенных приложений
Студенту Полькову Никите Николаевичу Группа 23ВВН1
Тема проекта Разработка программы для построения
графиков функций

Исходные данные (технические требования) на проектирование

Разработать приложение для построения графиков функций
Языки программирования Java, среда разработки NetBeans.
Приложение должно обладать графическим интерфейсом и использовать
следующие технологии:

1. Java Collections Framework
2. Механизм обработки исключительных ситуаций
3. Java Stream API
4. Java Multithreading
5. Сетевое взаимодействие

Объем работы по курсу

1. Расчетная часть

2. Графическая часть

Диаграмма эрота UML; диаграмма вариантов использования;
диаграмма классов; диаграмма деятельности; диаграмма
развертывания; диаграмма последовательности

3. Экспериментальная часть

Тестирование многомодульного приложения

Срок выполнения проекта по разделам

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8

Дата выдачи задания "12" февраля 2026

Дата защиты проекта " " "

Руководитель Карамышева Надежда Сергеевна

Задание получил "19" марта 2026 г.

Студент Толынов Никита Николаевич

Содержание

Введение.....	5
1 Постановка задачи	7
2 Теоретическая часть задания	8
2.1 Система программирования.....	8
2.2 Программа для построения графиков функции.....	9
3 Описание алгоритма работы	10
3.1 Описание клиент-серверной части.....	10
3.2 Описание работы приложения.....	10
4 Описание программы.....	12
4.1 Клиентская часть.....	12
4.2 Серверная часть.....	12
5 Руководство пользователя	14
5.1 Сервер.....	14
5.2 Клиент	14
Заключение	21
Список литературы	22
Приложение А. Листинг программы	23
Приложение А.1.	23
Приложение Б. UML-диаграммы	39
Приложение Б.1. UML-диаграмма вариантов использования	39
Приложение Б.2. UML-диаграмма классов.....	40
Приложение Б.3. UML- диаграмма деятельности	41
Приложение Б.4. UML-диаграмма последовательности	42
Приложение Б.5. UML-диаграмма развертывания.....	43

Введение

На сегодняшний момент язык Java является одним из самых распространенных и популярных языков программирования.

Ключевой особенностью языка Java является то, что его код сначала транслируется в специальный байт-код, независимый от платформы. А затем этот байт-код выполняется виртуальной машиной JVM (Java Virtual Machine).

Подобная архитектура обеспечивает кроссплатформенность и аппаратную переносимость программ на Java, благодаря чему подобные программы без перекомпиляции могут выполняться на различных платформах - Windows, Linux, Mac OS и т.д. Для каждой из платформ может быть своя реализация виртуальной машины JVM, но каждая из них может выполнять один и тот же код.

Java является языком с Си-подобным синтаксисом и близок в этом отношении к C/C++ и C#. Поэтому, если вы знакомы с одним из этих языков, то овладеть Java будет легче.

Еще одной ключевой особенностью Java является то, что она поддерживает автоматическую сборку мусора. А это значит, что вам не надо освобождать вручную память от ранее использовавшихся объектов, как в C++, так как сборщик мусора это сделает автоматически за вас.

Java является объектно-ориентированным языком. Он поддерживает полиморфизм, наследование, статическую типизацию. Объектно-ориентированный подход позволяет решить задачи по построению крупных, но в тоже время гибких, масштабируемых и расширяемых приложений.

В данном курсовом проекте будет использована клиент-серверная архитектура. Клиент-сервер (Client-server) – вычислительная или сетевая архитектура, в которой задания или сетевая нагрузка распределены между поставщиками услуг (сервисов), называемыми серверами, и заказчиками услуг, называемыми клиентами.

Нередко клиенты и серверы взаимодействуют через компьютерную

сеть и могут быть как различными физическими устройствами, так и программным обеспечением.

Сама технология очень проста. Пользователь делает запрос, а сервер выдает ответ.

1 Постановка задачи

Разработать программу для построения графиков функции. Язык программирования Java, среда разработки Apache NetBeans IDE20. Приложение должно обладать графическим интерфейсом и использовать следующие технологии:

1. Java Collection Framework;
2. Механики обработки исключительных ситуаций;
3. Java Stream API;
4. Java Multithreading;
5. Сетевое взаимодействие.

2 Теоретическая часть задания

2.1 Система программирования

NetBeans IDE — это бесплатная среда разработки с открытым исходным кодом, которая помогает создавать различные виды программных приложений. В контексте этой книги мы сосредоточимся конкретно на IDE как среде для разработки Java-приложений. Среда IDE обеспечивает поддержку всех типов Java-приложений, помогая вам разрабатывать проекты, использующие все технологии Java — Java SE, Java EE, встроенные и облачные.

Среда IDE является модульной и может быть расширена с помощью плагинов. Функции, включенные в пакеты загрузки программы, с которыми вы познакомитесь в этой главе, достаточно обширны, так что вам не нужно будет использовать какие-либо плагины, по крайней мере, на начальном этапе.

Java — это язык, на котором написана среда IDE. Она может работать в операционных системах, в которых установлен пакет разработки Java (JDK). Установщики, которые помогут вам в процессе установки и настройки среды IDE, доступны для Windows, Linux и Mac OSX. IDE также можно загрузить в виде файла ZIP или TAR, что может быть полезно, если вы хотите использовать IDE в операционной системе, для которой не предусмотрен установщик.

Для целей данной книги IDE предназначена для упрощения процесса разработки приложений на Java. Без IDE цикл редактирования-компиляции-отладки, присущий разработке на Java, является громоздким и подверженным ошибкам. Чтобы преодолеть эти проблемы и помочь в процессе разработки, в среду IDE встроены инструменты для выполнения этих действий. В частности, среда IDE выполняет следующие действия:

- Уведомляет вас о проблемах в вашем коде и выделяет их в редакторе Java;
- Помогает вам быстро и эффективно кодировать, используя такие функции редактора, как шаблоны кода, завершение кода и подсказки в редакторе Java;
- При вводе текста в редакторе Java отображается документация по классам и методам;
- Включает визуальную навигационную поддержку, включая функции навигатора и редактора, такие как сворачивание кода, а также множество сочетаний клавиш для ускорения написания кода;
- Предоставляет гиперссылки в окне вывода, позволяющие перейти от ошибок компиляции к строкам в редакторе Java, где находится соответствующий проблемный код;
- Помогает управлять изменениями в коде вашего приложения, управляя ссылками на названия пакетов, классов и методов по всему коду. При перемещении или переименовании кода среда IDE находит связанный код, на который повлияли внесенные вами изменения, и позволяет вам контролировать, следует ли автоматически изменять связанный код.

2.2 Программа для построения графиков функции

Данное приложение представляет собой систему визуализации данных с клиент-серверной авторизацией. Программа позволяет загружать и отображать в виде интерактивных графиков данные из Excel и CSV файлов с поддержкой различных кодировок (UTF-8, windows-1251, CP866), управлять видимостью отдельных параметров, масштабировать и панорамировать графики. Серверная часть обеспечивает централизованную проверку учетных записей пользователей с генерацией сессионных токенов, что позволяет контролировать доступ к приложению и поддерживать одновременную работу нескольких клиентов.

3 Описание алгоритма работы

3.1 Описание клиент-серверной части

Разработанная программа состоит из клиентской части и серверной. Сначала необходимо запустить сервер, который создает конечную точку для обслуживания TCP/IP соединений на определенном порту (по умолчанию 8080). Сервер привязывает сокет к локальной конечной точке и переходит в состояние прослушивания, ожидая входящие попытки соединения от клиентов. Клиентская часть при запуске отображает диалог авторизации, где пользователь вводит логин, пароль и адрес сервера. При нажатии кнопки входа клиент устанавливает HTTP-соединение с сервером и отправляет POST-запрос с учетными данными. Сервер принимает запрос, проверяет наличие пользователя в базе и соответствие пароля, после чего генерирует уникальный токен сессии и отправляет подтверждение клиенту. При успешной авторизации клиент получает доступ к основному функционалу приложения. Сервер поддерживает одновременную работу нескольких клиентов за счет многопоточной обработки запросов.

3.2 Описание работы приложения

Приложение после успешной авторизации предоставляет пользователю графический интерфейс для работы с данными. Основной функционал включает загрузку и визуализацию данных из файлов форматов Excel (.xlsx, .xls) и CSV с автоматическим определением структуры и заголовков. Пользователь может управлять отображением отдельных параметров на графике, включая и отключая их видимость с помощью чекбоксов с цветовой индикацией. Реализована поддержка нескольких популярных кодировок (UTF-8, windows-1251, CP866, ISO-8859-1) для корректного чтения файлов. Приложение позволяет масштабировать график с помощью колесика мыши и кнопок управления, а также сбрасывать масштаб до исходного состояния. Данные на графике можно очистить для загрузки нового файла. После

завершения работы пользователь может выйти из системы и пройти повторную авторизацию.

4 Описание программы

4.1 Клиентская часть

4.1.1 Файл ChartWithExcel.java

Основной файл клиентской части, содержащий всю логику работы приложения. В данном файле реализован графический интерфейс на основе библиотеки Swing, методы авторизации пользователя через HTTP-соединение с сервером, загрузка и обработка данных из файлов форматов Excel (.xlsx, .xls) и CSV с поддержкой нескольких кодировок (UTF-8, windows-1251, CP866, ISO-8859-1). Также включает встроенный класс ColorManager для управления цветовой схемой отображаемых серий данных, методы построения интерактивных графиков с помощью библиотеки JFreeChart, управления видимостью отдельных параметров через чекбоксы с цветовой индикацией, а также функции масштабирования и очистки графика. Метод main() обеспечивает запуск клиентского приложения с установкой системного стиля оформления.

4.2 Серверная часть

4.2.1 Файл AuthServer.java

Главный и единственный файл серверной части, реализующий HTTP-сервер для авторизации пользователей. В данном файле описаны создание и запуск сервера на порту 8080 с использованием встроенного в JDK класса HttpServer, хранение базы учетных записей пользователей и активных сессий в потокобезопасных коллекциях ConcurrentHashMap, обработка POST-запросов авторизации через внутренний класс LoginHandler, проверка учетных данных пользователя с извлечением параметров из тела запроса, генерация уникальных токенов сессий с помощью UUID, а также отправка ответов клиенту. Тестовые учетные записи инициализируются в статическом блоке при запуске сервера. Метод main() создает и запускает сервер с пулом из десяти потоков для одновременной обработки клиентских подключений.

4.3 Общие принципы взаимодействия

Клиентская и серверная части взаимодействуют по протоколу HTTP. Клиент отправляет POST-запрос с логином и паролем на сервер, сервер проверяет данные и возвращает ответ с токеном сессии. Для передачи данных используется текстовый формат с разделителями. Обе части используют кодировку UTF-8 для корректной обработки запросов и ответов.

5 Руководство пользователя

5.1 Сервер

Сервер не имеет графического интерфейса. Требуется только запустить его, дальше он сам будет принимать запросы и обслуживать их.

5.2 Клиент

При запуске клиента, открывается окно приложения для подключения к серверу (рисунок 1).

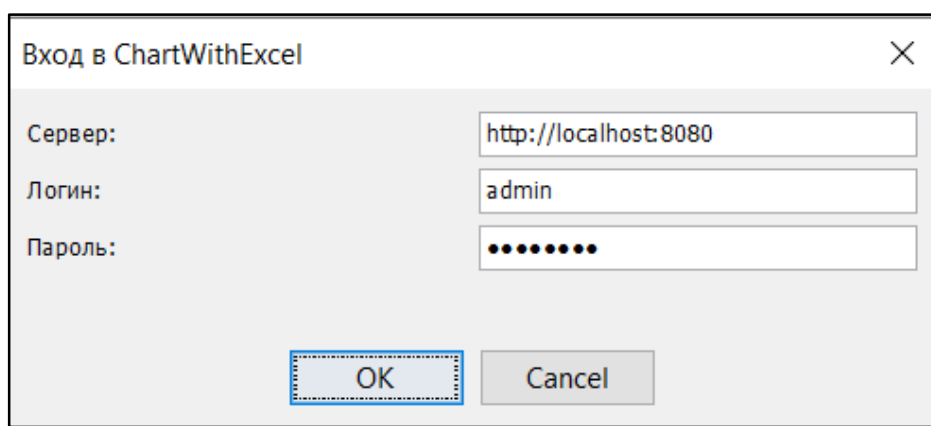


Рисунок 1 – Окно подключения

Далее пользователь вводит пару логин/пароль и при успешном вводе и нажатии на «ОК» входит в приложение (рисунок 2).

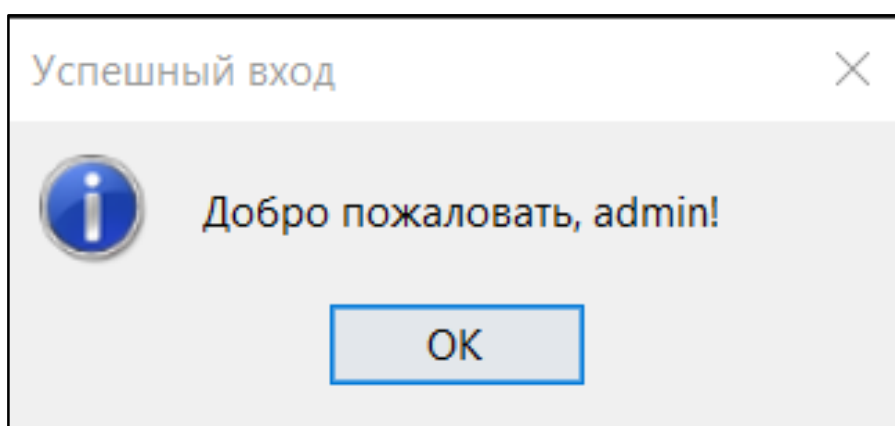


Рисунок 2 – Успешный вход

При неверном вводе учетных данных появляется сообщение об ошибке и предлагается повторить попытку входа (рисунок 3).

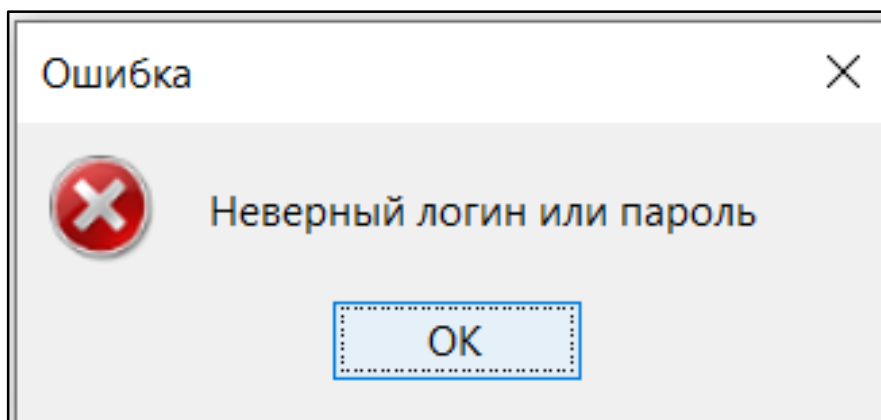


Рисунок 3 – Ошибка авторизации

После успешной авторизации открывается главный экран приложения, на котором отображается панель управления и область построения графиков (рисунок 4). В верхней части панели управления отображается имя текущего пользователя.

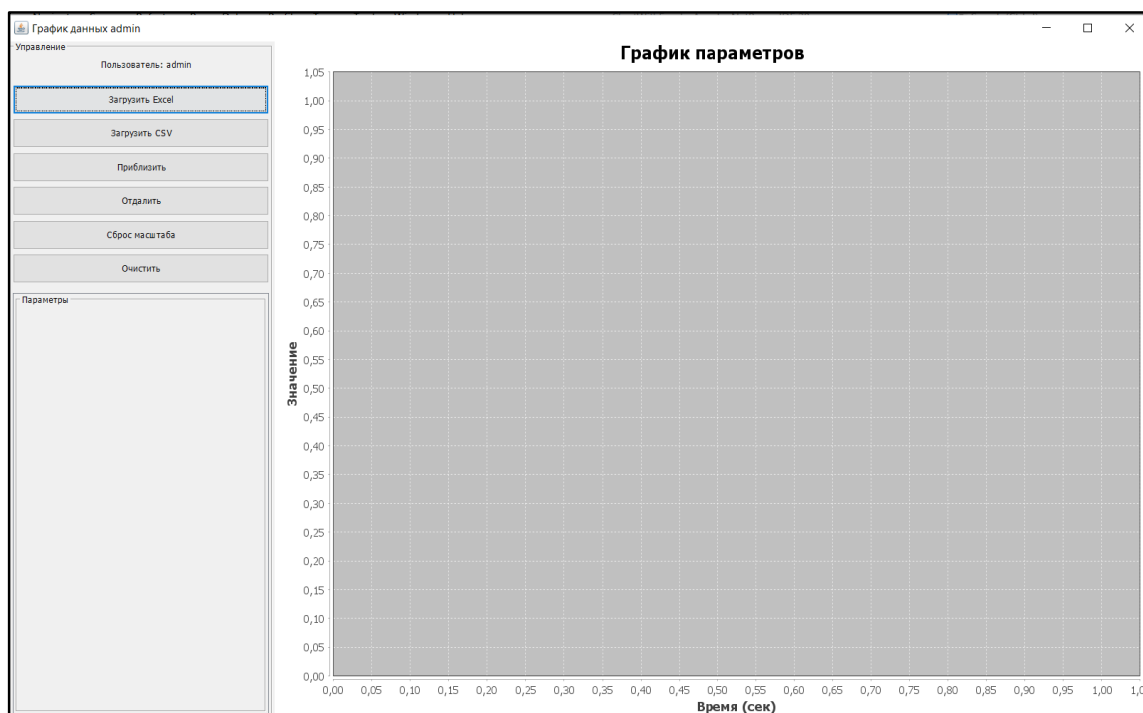


Рисунок 4 – Главный экран приложения

Для загрузки данных из Excel-файла необходимо нажать кнопку «Загрузить Excel», после чего откроется диалог выбора файла с фильтром по расширениям .xlsx и .xls (рисунок 5). После выбора файла данные загружаются и отображаются в виде графиков.

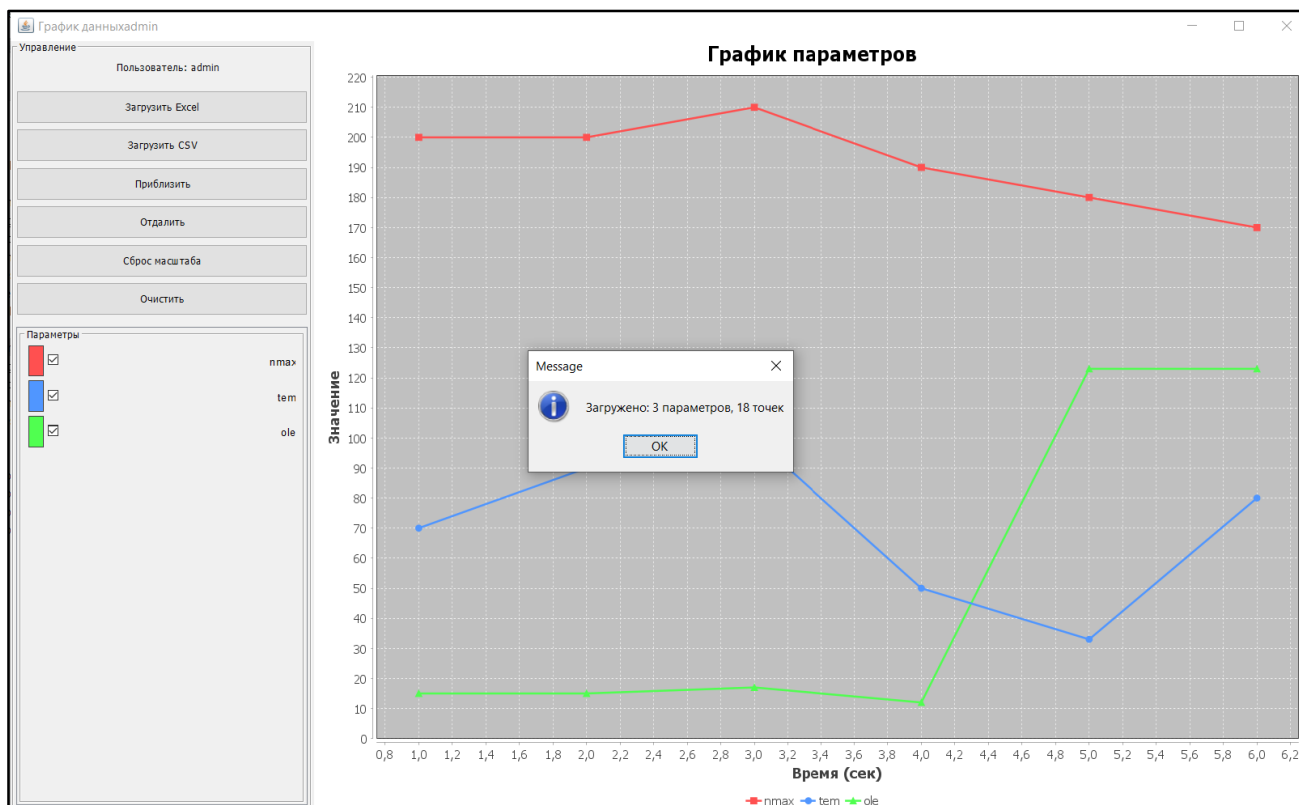


Рисунок 5 – Загрузка данных из Excel-файла

Аналогично выполняется загрузка данных из CSV-файлов. При нажатии на кнопку «Загрузить CSV» открывается диалог выбора файла, где можно выбрать файлы с расширениями .csv (рисунок 6). Приложение автоматически определяет кодировку файла из списка поддерживаемых (UTF-8, windows-1251, CP866, ISO-8859-1).

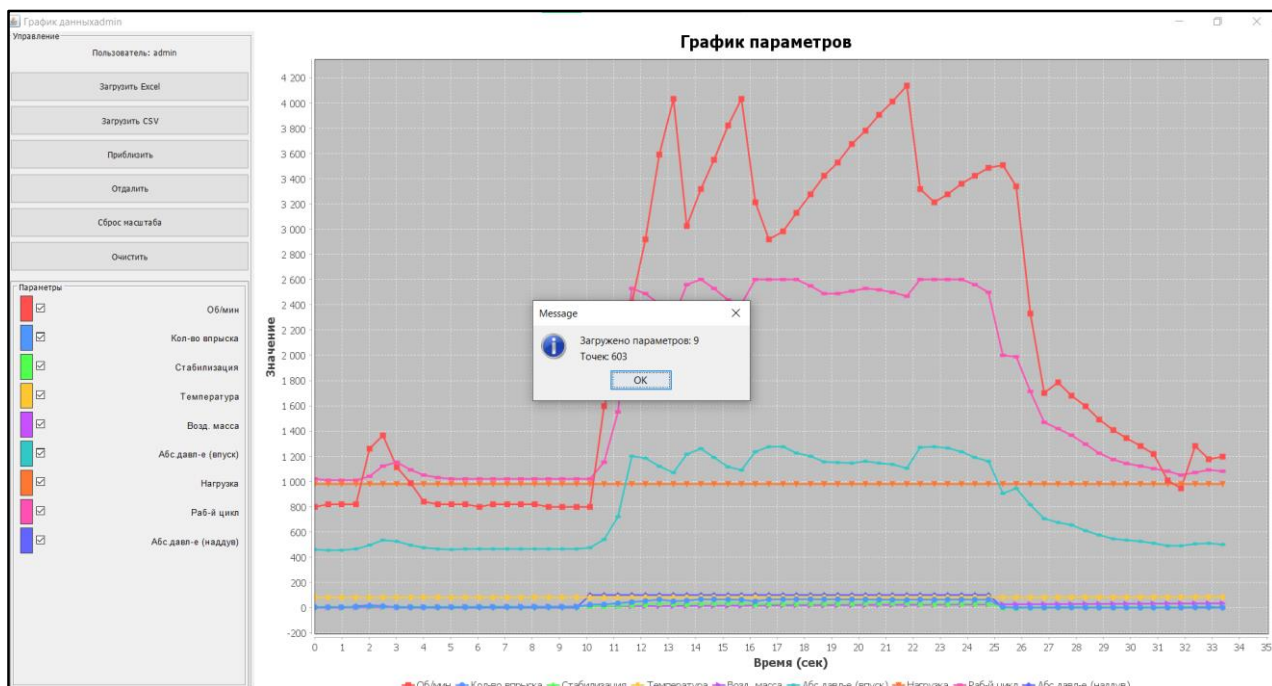


Рисунок 6 – Загрузка данных из CSV-файла

После успешной загрузки данных на графике отображаются все найденные параметры в виде цветных линий. Каждый параметр подписан и имеет уникальный цвет для визуального различия (рисунок 7).

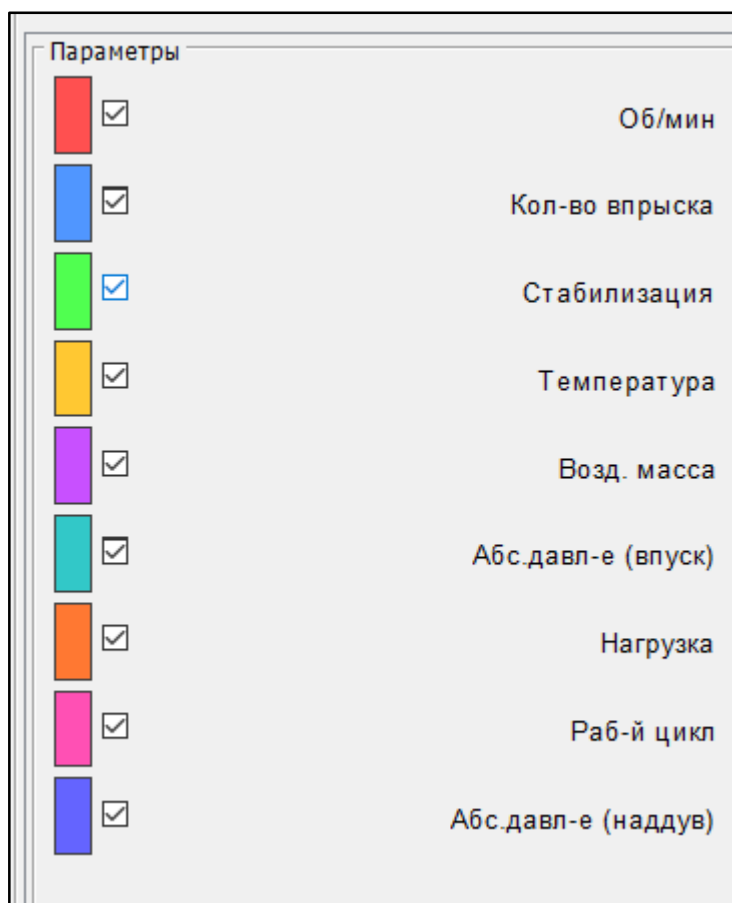


Рисунок 7 – Отображение загруженных параметров на графике

В левой части окна расположена панель со списком параметров, где каждый параметр имеет цветовой индикатор и флажок для управления видимостью. При снятии флажка соответствующая линия скрывается с графика, при установке – появляется вновь с сохранением исходного цвета (рисунок 8).

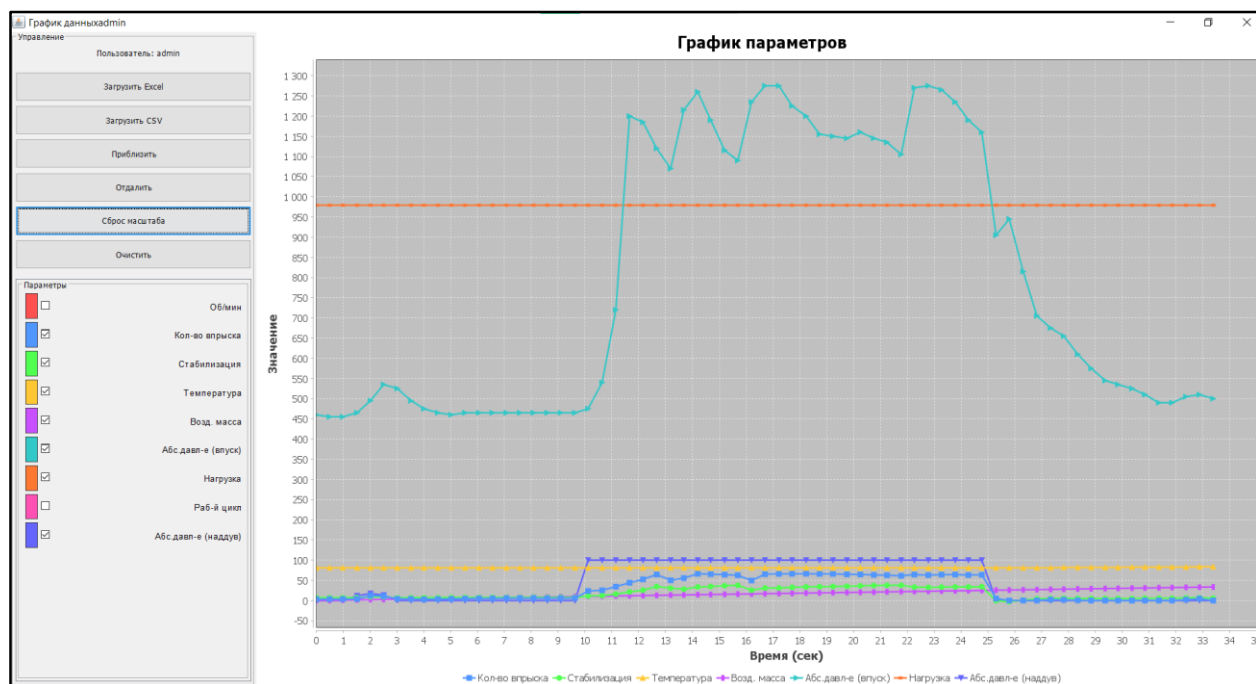


Рисунок 8 – Управление видимостью параметров на графике

Для изменения масштаба графика можно использовать кнопки на панели управления. Кнопка «Приблизить» увеличивает масштаб графика, кнопка «Отдалить» уменьшает его (рисунок 9).

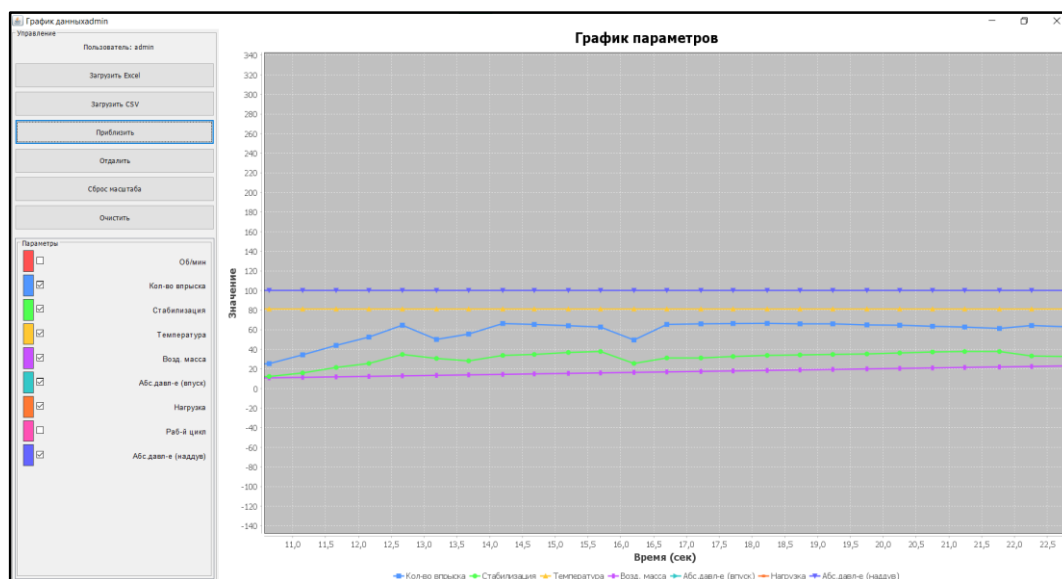


Рисунок 9– Масштабирование графика

Для возврата к исходному масштабу после приближения или отдаления используется кнопка «Сброс масштаба», которая восстанавливает автоматические границы графика (рисунок 10).

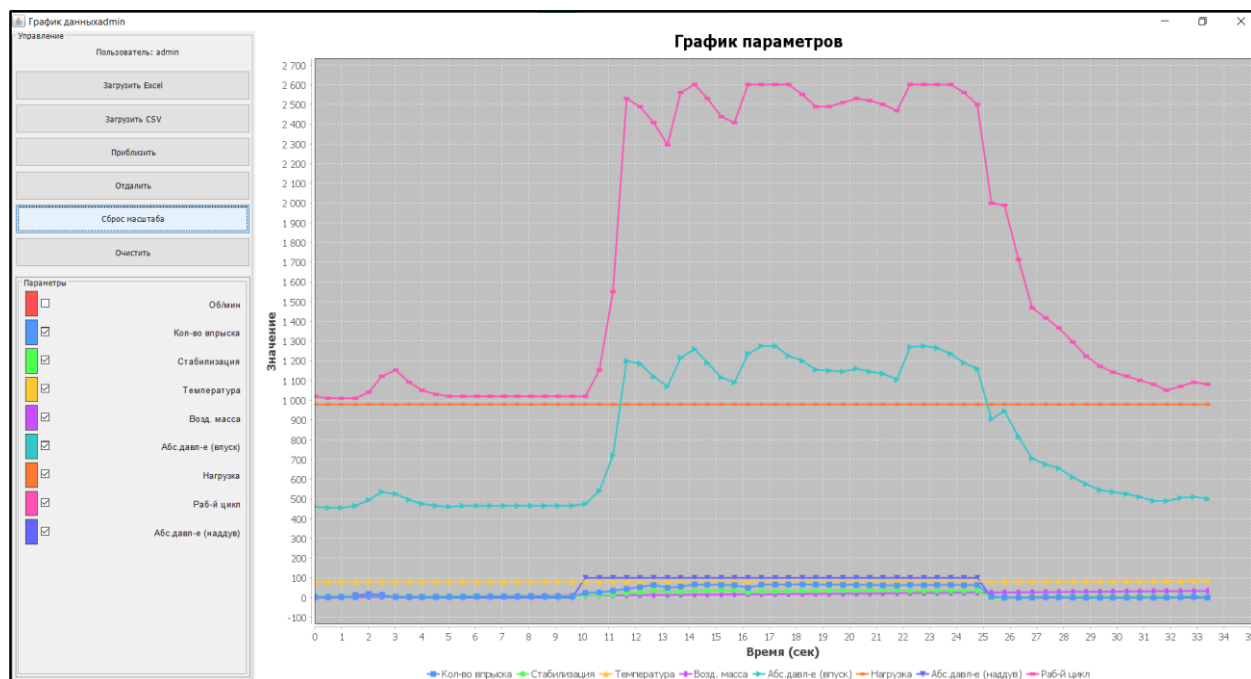


Рисунок 10 – Сброс масштаба графика

При необходимости очистить график от загруженных данных используется кнопка «Очистить», которая удаляет все серии данных, сбрасывает список параметров и подготавливает приложение к загрузке нового файла (рисунок 11).

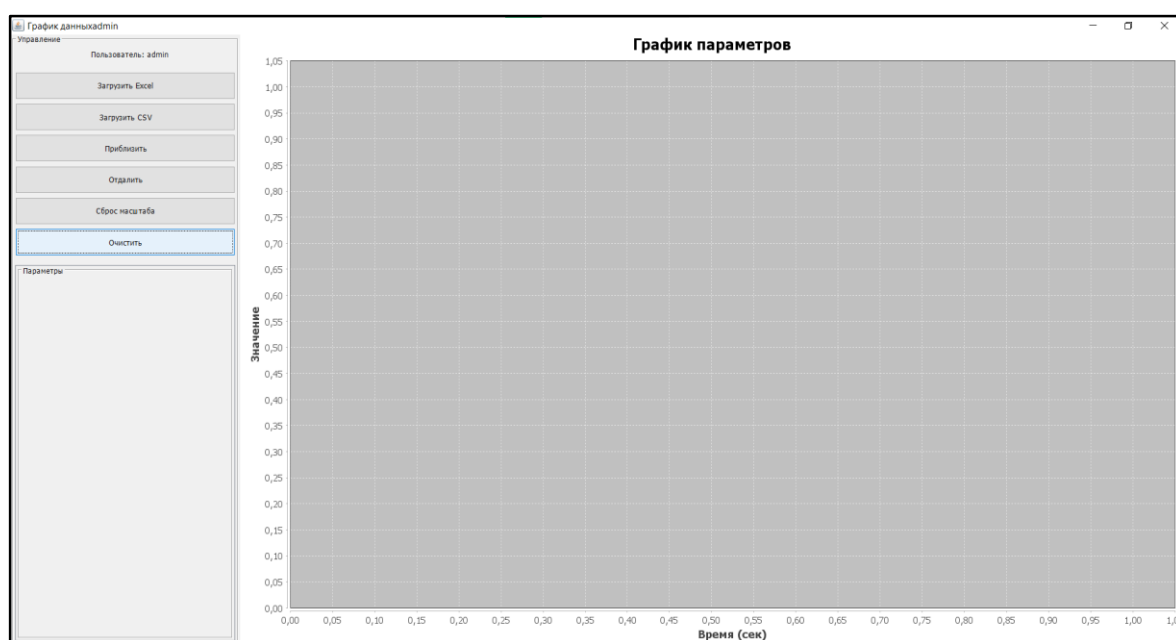


Рисунок 11 – Очистка графика

Таким образом, приложение предоставляет пользователю полный набор функций для загрузки, визуализации и анализа данных из различных источников с возможностью гибкой настройки отображения и навигации по графику.

Заключение

При выполнении данного курсового проекта были изучены принципы работы протоколов TCP/IP. Были получены навыки разработки многомодульного приложения с пользовательским интерфейсом и работы в среде Apache NetBeans IDE. Получены навыки разработки клиент-серверных приложений.

В результате выполнения данной курсовой работы была разработана программа для построения графиков функции, позволяющее пользователям работать с файлами.

Список литературы

1. Wielenga G. Beginning netbeans ide: For java developers. – Apress, 2015. (Дата обращения: 15.05.26)
<https://books.google.com/books?hl=ru&lr=&id=9HeBCgAAQBAJ&oi=fnd&pg=PR6&dq=netbeans+ide+java+programming+language&ots=9FVAOvrCXP&sig=6vnK9tpxpCFm5iIXjsPFPIdb2ws>
2. Язык Java. Курс программирования / Уолтер Савитч. - М.: Вильямс, 2015. - 928 с. (Дата обращения: 15.05.26) [Язык Java. Курс программирования — Савитч, Уолтер | Livelib](#)
3. Niemeyer P., Knudsen J. Learning java. – " O'Reilly Media, Inc.", 2005. (Дата обращения: 15.05.26)
https://books.google.com/books?hl=ru&lr=&id=T9abAgAAQBAJ&oi=fnd&pg=PR5&dq=java&ots=vcOkdGO2rD&sig=i_jo6SWoXGw77eld_lrDJnSG-Cg
4. Форум программистов и сисадминов Киберфорум.
<http://www.cyberforum.ru/>. (Дата обращения: 15.05.26)
5. Stack Overflow <https://stackoverflow.com/> . (Дата обращения: 15.05.26)

Приложение А. Листинг программы

Приложение А.1.

Файл ChartWithExcel.java

```
package folder;

import org.apache.poi.ss.usermodel.Cell;
import org.apache.poi.ss.usermodel.CellType;
import org.apache.poi.ss.usermodel.Row;
import org.apache.poi.ss.usermodel.Sheet;
import org.apache.poi.ss.usermodel.Workbook;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.chart.plot.XYPlot;
import org.jfree.chart.renderer.xy.XYLineAndShapeRenderer;
import org.jfree.data.xy.XYSeries;
import org.jfree.data.xy.XYSeriesCollection;

import javax.swing.BorderFactory;
import javax.swing.Box;
import javax.swing.BoxLayout;
import javax.swing.JButton;
import javax.swing.JCheckBox;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JPasswordField;
import javax.swing.JTextField;
import javax.swing.SwingUtilities;
import javax.swing.UIManager;
import javax.swing.filechooser.FileNameExtensionFilter;
import java.awt.BasicStroke;
import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.GridLayout;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.net.URL;
import java.nio.charset.Charset;
import java.nio.charset.StandardCharsets;
import java.util.ArrayList;
import java.util.LinkedHashMap;
import java.util.List;
```

```

import java.util.Map;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.regex.Pattern;
import java.util.stream.Collectors;
import java.util.stream.IntStream;

public class ChartWithExcel extends JFrame {

    // Класс для управления цветами
    private static class ColorManager {
        private static final Color[] COLORS = {
            new Color(255, 80, 80),    // Красный
            new Color(80, 150, 255),   // Синий
            new Color(80, 255, 80),    // Зеленый
            new Color(255, 200, 50),    // Желтый
            new Color(200, 80, 255),    // Фиолетовый
            new Color(50, 200, 200),    // Бирюзовый
            new Color(255, 120, 50),    // Оранжевый
            new Color(255, 80, 180),    // Розовый
            new Color(100, 100, 255),   // Светло-синий
            new Color(100, 255, 150),   // Мятный
            new Color(255, 150, 100),   // Коралловый
            new Color(180, 100, 255)    // Лавандовый
        };

        private Map<String, Integer> colorIndexMap;
        private int nextIndex;

        public ColorManager() {
            colorIndexMap = new LinkedHashMap<>();
            nextIndex = 0;
        }

        public Color getColor(String seriesName) {
            if (!colorIndexMap.containsKey(seriesName)) {
                colorIndexMap.put(seriesName, nextIndex % COLORS.length);
                nextIndex++;
            }
            return COLORS[colorIndexMap.get(seriesName)];
        }

        public void clear() {
            colorIndexMap.clear();
            nextIndex = 0;
        }
    }

    private XYSeriesCollection dataset;
    private JFreeChart chart;
    private ChartPanel chartPanel;
    private XYLineAndShapeRenderer renderer;
    private Map<String, XYSeries> seriesMap;
    private ColorManager colorManager;
    private List<String> seriesOrder;
    private JPanel listPanel;

    // Данные для сервера
    private String serverUrl = "http://localhost:8080";
    private String authToken = null;
    private String currentUser = null;

```

```

// Пул потоков для фоновых задач
private final ExecutorService executor = Executors.newFixedThreadPool(3);

public ChartWithExcel() {
    seriesMap = new LinkedHashMap<>();
    colorManager = new ColorManager();
    seriesOrder = new ArrayList<>();
    dataset = new XYSeriesCollection();
    renderer = new XYLineAndShapeRenderer();

    // Показываем диалог входа
    if (!showLoginDialog()) {
        executor.shutdown();
        System.exit(0);
    }

    initUI();
}

private boolean showLoginDialog() {
    JTextField serverField = new JTextField("http://localhost:8080", 20);
    JTextField loginField = new JTextField("admin", 20);
    JPasswordField passwordField = new JPasswordField("admin123", 20);

    JPanel panel = new JPanel(new GridLayout(4, 2, 5, 5));
    panel.add(new JLabel("Сервер:"));
    panel.add(serverField);
    panel.add(new JLabel("Логин:"));
    panel.add(loginField);
    panel.add(new JLabel("Пароль:"));
    panel.add(passwordField);
    panel.add(new JLabel(""));

    while (true) {
        int result = JOptionPane.showConfirmDialog(
            null, panel, "Вход в ChartWithExcel",
            JOptionPane.OK_CANCEL_OPTION, JOptionPane.PLAIN_MESSAGE
        );

        if (result != JOptionPane.OK_OPTION) {
            return false;
        }

        serverUrl = serverField.getText().trim();
        String login = loginField.getText().trim();
        String password = new String(passwordField.getPassword());

        try {
            URL url = new URL(serverUrl + "/api/login");
            HttpURLConnection conn = (HttpURLConnection)
url.openConnection();
            conn.setRequestMethod("POST");
            conn.setDoOutput(true);
            conn.setConnectTimeout(3000);
            conn.setReadTimeout(3000);

            String body = "login=" + login + "&password=" + password;

            try (OutputStream os = conn.getOutputStream()) {
                os.write(body.getBytes(StandardCharsets.UTF_8));
            }
        }
    }
}

```

```

        int responseCode = conn.getResponseCode();

        if (responseCode == 200) {
            try (BufferedReader br = new BufferedReader(
                new InputStreamReader(conn.getInputStream(),
StandardCharsets.UTF_8))) {
                String response =
br.lines().collect(Collectors.joining());

                if (response != null && response.startsWith("OK:")) {
                    String[] parts = response.split(":");
                    if (parts.length >= 3) {
                        authToken = parts[1];
                        currentUser = parts[2];

                        JOptionPane.showMessageDialog(null,
                            "Добро пожаловать, " + currentUser + "!",
                            "Успешный вход",
JOptionPane.INFORMATION_MESSAGE);

                        return true;
                    }
                }
            }
        } else {
            JOptionPane.showMessageDialog(null,
                "Неверный логин или пароль",
                "Ошибка", JOptionPane.ERROR_MESSAGE);
        }

        } catch (Exception e) {
            JOptionPane.showMessageDialog(null,
                "Не удалось подключиться к серверу:\n" + e.getMessage(),
                "Ошибка подключения", JOptionPane.ERROR_MESSAGE);
        }
    }

    private void initUI() {
        setTitle("График данных" + currentUser);
        setSize(1300, 800);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new BorderLayout());

        renderer.setDefaultShapesVisible(true);
        renderer.setDefaultLinesVisible(true);
        renderer.setDefaultShape(new java.awt.geom.Ellipse2D.Double(-3, -3, 6,
6));

        chart = ChartFactory.createXYLineChart(
            "График параметров",
            "Время (сек)",
            "Значение",
            dataset,
            PlotOrientation.VERTICAL,
            true,
            true,
            false
        );

        XYPlot plot = chart.getXYPlot();

```

```

plot.setRenderer(renderer);

// ChartPanel с панорамированием
chartPanel = new ChartPanel(chart) {
    private int lastX, lastY;
    private boolean isPanning = false;

    @Override
    public void mousePressed(java.awt.event.MouseEvent e) {
        if (e.getButton() == java.awt.event.MouseEvent.BUTTON1) {
            lastX = e.getX();
            lastY = e.getY();
            isPanning = true;
        }
    }

    @Override
    public void mouseReleased(java.awt.event.MouseEvent e) {
        if (e.getButton() == java.awt.event.MouseEvent.BUTTON1) {
            isPanning = false;
            setCursor(java.awt.Cursor.getDefaultCursor());
        } else {
            super.mouseReleased(e);
        }
    }

    @Override
    public void mouseDragged(java.awt.event.MouseEvent e) {
        if (isPanning) {
            int dx = e.getX() - lastX;
            int dy = e.getY() - lastY;

            XYPlot p = chart.getXYPlot();

            org.jfree.data.Range domainRange =
p.getDomainAxis().getRange();
            org.jfree.data.Range rangeRange =
p.getRangeAxis().getRange();

            java.awt.Rectangle dataArea = (java.awt.Rectangle)
getChartRenderingInfo().getPlotInfo().getDataArea();
            double scaleX = domainRange.getLength() /
dataArea.getWidth();
            double scaleY = rangeRange.getLength() /
dataArea.getHeight();

            p.getDomainAxis().setRange(
                domainRange.getLowerBound() - dx * scaleX,
                domainRange.getUpperBound() - dx * scaleX
            );
            p.getRangeAxis().setRange(
                rangeRange.getLowerBound() + dy * scaleY,
                rangeRange.getUpperBound() + dy * scaleY
            );

            lastX = e.getX();
            lastY = e.getY();
            repaint();
        }
    }
}

```

```

        } else {
            super.mouseDragged(e);
        }
    }
};

// Отключаем зум выделением и колесиком
chartPanel.setDomainZoomable(false);
chartPanel.setRangeZoomable(false);
chartPanel.setMouseWheelEnabled(false);

JPanel controlPanel = new JPanel();
controlPanel.setLayout(new BoxLayout(controlPanel, BoxLayout.Y_AXIS));
controlPanel.setBorder(BorderFactory.createTitledBorder("Управление"));
controlPanel.setPreferredSize(new Dimension(300, 0));

// Информация о пользователе
JLabel userLabel = new JLabel("Пользователь: " + currentUser);
userLabel.setBorder(BorderFactory.createEmptyBorder(5, 5, 5, 5));
controlPanel.add(userLabel);
controlPanel.add(Box.createRigidArea(new Dimension(0, 10)));

JButton loadExcelBtn = new JButton("Загрузить Excel");
loadExcelBtn.addActionListener(e -> loadExcelFile());

JButton loadCSVBtn = new JButton("Загрузить CSV");
loadCSVBtn.addActionListener(e -> loadCSVFile());

JButton zoomInBtn = new JButton("Приблизить");
zoomInBtn.addActionListener(e -> smoothZoom(0.9));

JButton zoomOutBtn = new JButton("Отдалить");
zoomOutBtn.addActionListener(e -> smoothZoom(1.1));

JButton resetZoomBtn = new JButton("Сброс масштаба");
resetZoomBtn.addActionListener(e -> chartPanel.restoreAutoBounds());

JButton clearChartBtn = new JButton("Очистить");
clearChartBtn.addActionListener(e -> clearChart());

JPanel buttonPanel = new JPanel(new GridLayout(6, 1, 5, 5));
buttonPanel.add(loadExcelBtn);
buttonPanel.add(loadCSVBtn);
buttonPanel.add(zoomInBtn);
buttonPanel.add(zoomOutBtn);
buttonPanel.add(resetZoomBtn);
buttonPanel.add(clearChartBtn);

controlPanel.add(buttonPanel);
controlPanel.add(Box.createRigidArea(new Dimension(0, 10)));

listPanel = new JPanel();
listPanel.setLayout(new BoxLayout(listPanel, BoxLayout.Y_AXIS));
listPanel.setBorder(BorderFactory.createTitledBorder("Параметры"));

JScrollPane scrollPane = new JScrollPane(listPanel);
scrollPane.setPreferredSize(new Dimension(280, 400));
controlPanel.add(scrollPane);

add(controlPanel, BorderLayout.WEST);
add(chartPanel, BorderLayout.CENTER);

```

```

        setLocationRelativeTo(null);
    }

    // Плавный зум с сохранением центра графика
    private void smoothZoom(double factor) {
        XYPlot plot = chart.getXYPlot();

        org.jfree.data.Range domainRange = plot.getDomainAxis().getRange();
        org.jfree.data.Range rangeRange = plot.getRangeAxis().getRange();

        double domainCenter = domainRange.getCentralValue();
        double rangeCenter = rangeRange.getCentralValue();

        double newDomainLength = domainRange.getLength() * factor;
        double newRangeLength = rangeRange.getLength() * factor;

        plot.getDomainAxis().setRange(
            domainCenter - newDomainLength / 2,
            domainCenter + newDomainLength / 2
        );
        plot.getRangeAxis().setRange(
            rangeCenter - newRangeLength / 2,
            rangeCenter + newRangeLength / 2
        );
    }

    private void loadExcelFile() {
        JFileChooser fileChooser = new JFileChooser();
        fileChooser.setFileFilter(new FileNameExtensionFilter("Excel Files",
            "xlsx", "xls"));

        if (fileChooser.showOpenDialog(this) == JFileChooser.APPROVE_OPTION) {
            CompletableFuture.runAsync(() -> {
                try {
                    loadDataFromExcel(fileChooser.getSelectedFile());
                } catch (Exception ex) {
                    SwingUtilities.invokeLater(() ->
                        JOptionPane.showMessageDialog(this, "Ошибка: " +
                            ex.getMessage()));
                }
            }, executor);
        }
    }

    private void loadCSVFile() {
        JFileChooser fileChooser = new JFileChooser();
        fileChooser.setFileFilter(new FileNameExtensionFilter("CSV Files",
            "csv", "txt"));

        if (fileChooser.showOpenDialog(this) == JFileChooser.APPROVE_OPTION) {
            CompletableFuture.runAsync(() -> {
                try {
                    loadDataFromCSV(fileChooser.getSelectedFile());
                } catch (Exception ex) {
                    SwingUtilities.invokeLater(() ->
                        JOptionPane.showMessageDialog(this, "Ошибка: " +
                            ex.getMessage()));
                }
            }, executor);
        }
    }
}

```

```

private void loadDataFromExcel(File file) throws Exception {
    clearChart();

    FileInputStream fis = new FileInputStream(file);
    Workbook workbook = new XSSFWorkbook(fis);
    Sheet sheet = workbook.getSheetAt(0);

    int startRow = findDataStartRow(sheet);
    Map<String, List<Double[]>> dataMap = new LinkedHashMap<>();

    Row headerRow = sheet.getRow(startRow - 1);
    Map<Integer, String> headers = new LinkedHashMap<>();
    if (headerRow != null) {
        IntStream.range(1, headerRow.getLastCellNum())
            .filter(col -> headerRow.getCell(col) != null)
            .forEach(col -> headers.put(col,
headerRow.getCell(col).toString()));
    }

    for (int rowNum = startRow; rowNum <= sheet.getLastRowNum(); rowNum++)
    {
        Row row = sheet.getRow(rowNum);
        if (row == null) continue;

        Cell timeCell = row.getCell(0);
        if (timeCell != null && timeCell.getCellType() == CellType.NUMERIC)
        {
            double time = timeCell.getNumericCellValue();

            for (int col = 1; col < row.getLastCellNum(); col++) {
                Cell cell = row.getCell(col);
                if (cell != null && cell.getCellType() == CellType.NUMERIC)
                {
                    String colName = headers.getOrDefault(col, "Колонка " +
col);

                    dataMap.computeIfAbsent(colName, k -> new
ArrayList<>());
                    dataMap.get(colName).add(new Double[]{time,
cell.getNumericCellValue()});
                }
            }
        }

        workbook.close();
        fis.close();

        long totalPoints =
dataMap.values().stream().mapToLong(List::size).sum();

        addSeriesToChart(dataMap);
        JOptionPane.showMessageDialog(this,
            "Загружено: " + dataMap.size() + " параметров, " + totalPoints + "
точек");
    }

    private void loadDataFromCSV(File file) throws Exception {
        clearChart();

        String[] charsetNames = {"UTF-8", "windows-1251", "ISO-8859-1",
"CP866"};
    }
}

```

```

String content = null;

for (String charsetName : charsetNames) {
    try {
        content = readFileWithCharset(file, charsetName);
        if (content != null && content.contains(",")) break;
    } catch (Exception e) {}
}

if (content == null) {
    content =
String(java.nio.file.Files.readAllBytes(file.toPath()),
StandardCharsets.UTF_8);
}

String[] lines = content.split("\\r?\\n");

List<String> dataLines = java.util.Arrays.stream(lines)
    .filter(line -> !line.trim().isEmpty())
    .filter(line -> line.startsWith(",") ||
        (line.length() > 0 && Pattern.matches("^\\d,.-]+$",
            line.substring(0, Math.min(5, line.length()))))
    .collect(Collectors.toList());

if (dataLines.isEmpty()) {
    throw new Exception("Нет данных в CSV");
}

String[] paramNames = {
    "Об/мин", "Кол-во впрыска", "Стабилизация", "Температура",
    "Возд. масса", "Абс.давл-е (впуск)", "Нагрузка", "Раб-й цикл",
    "Абс.давл-е (наддув)"
};

Map<String, List<Double[]>> dataMap = new LinkedHashMap<>();

for (String line : dataLines) {
    String[] values = line.split(",");
    if (values.length < 2) continue;

    double time = 0;
    int startIdx = 0;
    while (startIdx < values.length &&
values[startIdx].trim().isEmpty()) startIdx++;
    if (startIdx >= values.length) continue;

    try {
        time = Double.parseDouble(values[startIdx].trim());
    } catch (NumberFormatException e) {
        continue;
    }

    int paramIndex = 0;
    for (int i = startIdx + 1; i < values.length && paramIndex <
paramNames.length; i++) {
        String val = values[i].trim();
        if (!val.isEmpty() && !val.equals("-") && !val.equals("")) {
            try {
                double numVal = Double.parseDouble(val);
                String paramName = paramNames[paramIndex];

```

```

        dataMap.computeIfAbsent(paramName, k -> new
ArrayList<>());
        dataMap.get(paramName).add(new Double[]{time, numVal});
    } catch (NumberFormatException e) {}
    }
    paramIndex++;
}

if (dataMap.isEmpty()) {
    throw new Exception("Не удалось распарсить данные");
}

long totalPoints =
dataMap.values().stream().mapToLong(List::size).sum();
long totalParams = dataMap.size();

addSeriesToChart(dataMap);
JOptionPane.showMessageDialog(this,
    "Загружено параметров: " + totalParams + "\nТочек: " +
totalPoints);
}

private void addSeriesToChart(Map<String, List<Double[]>> dataMap) {
    for (Map.Entry<String, List<Double[]>> entry : dataMap.entrySet()) {
        String paramName = entry.getKey();

        XYSeries series = new XYSeries(paramName);
        for (Double[] point : entry.getValue()) {
            series.add(point[0], point[1]);
        }

        seriesMap.put(paramName, series);
        seriesOrder.add(paramName);
    }

    IntStream.range(0, seriesOrder.size()).forEach(i -> {
        String name = seriesOrder.get(i);
        XYSeries series = seriesMap.get(name);
        dataset.addSeries(series);

        Color color = colorManager.getColor(name);
        renderer.setSeriesPaint(i, color);
        renderer.setSeriesShapesVisible(i, true);
        renderer.setSeriesLinesVisible(i, true);
        renderer.setSeriesStroke(i, new BasicStroke(2.0f));
    });

    updateLineList();
}

private void updateLineList() {
    listPanel.removeAll();

    IntStream.range(0, seriesOrder.size()).forEach(i -> {
        final String seriesName = seriesOrder.get(i);
        final XYSeries series = seriesMap.get(seriesName);
        final Color color = colorManager.getColor(seriesName);

        JPanel linePanel = new JPanel(new BorderLayout());
        linePanel.setMaximumSize(new Dimension(280, 35));
    });
}

```

```

        linePanel.setBorder(BorderFactory.createEmptyBorder(2, 5, 2, 5));

        JCheckBox checkBox = new JCheckBox();
        checkBox.setSelected(true);

        checkBox.addActionListener(e -> {
            boolean isSelected = checkBox.isSelected();

            if (isSelected) {
                dataset.addSeries(series);
                int newIndex = dataset.indexOf(series);
                if (newIndex != -1) {
                    renderer.setSeriesPaint(newIndex, color);
                    renderer.setSeriesShapesVisible(newIndex, true);
                    renderer.setSeriesLinesVisible(newIndex, true);
                    renderer.setSeriesStroke(newIndex, new
BasicStroke(2.0f));
                }
            } else {
                int idx = dataset.indexOf(series);
                if (idx != -1) {
                    dataset.removeSeries(idx);
                }
            }

            refreshAllSeriesColors();
            chartPanel.repaint();
            listPanel.repaint();
        });

        JLabel nameLabel = new JLabel(seriesName);
        nameLabel.setFont(new Font("Dialog", Font.PLAIN, 11));

        JPanel colorPanel = new JPanel();
        colorPanel.setBackground(color);
        colorPanel.setPreferredSize(new Dimension(15, 15));

        colorPanel.setBorder(BorderFactory.createLineBorder(Color.DARK_GRAY));

        linePanel.add(colorPanel, BorderLayout.WEST);
        linePanel.add(checkBox, BorderLayout.CENTER);
        linePanel.add(nameLabel, BorderLayout.EAST);

        listPanel.add(linePanel);
    });

    listPanel.revalidate();
    listPanel.repaint();
}

private void refreshAllSeriesColors() {
    IntStream.range(0, dataset.getSeriesCount()).forEach(i -> {
        String seriesName = dataset.getSeries(i).getKey().toString();
        Color color = colorManager.getColor(seriesName);
        renderer.setSeriesPaint(i, color);
        renderer.setSeriesShapesVisible(i, true);
        renderer.setSeriesLinesVisible(i, true);
        renderer.setSeriesStroke(i, new BasicStroke(2.0f));
    });
}

```

```

        private String readFileWithCharset(File file, String charsetName) throws
IOException {
            try (BufferedReader reader = new BufferedReader(
                new
                    InputStreamReader(new
                        FileInputStream(file),
Charset.forName(charsetName)))) {
                return reader.lines().collect(Collectors.joining("\n"));
            }
        }

        private int findDataStartRow(Sheet sheet) {
            return IntStream.rangeClosed(0, Math.min(10, sheet.getLastRowNum()))
                .filter(i -> {
                    Row row = sheet.getRow(i);
                    if (row != null) {
                        Cell firstCell = row.getCell(0);
                        return firstCell != null && firstCell.getCellType() ==
CellType.NUMERIC;
                    }
                    return false;
                })
                .findFirst()
                .orElse(1);
        }

        private void clearChart() {
            IntStream.iterate(dataset.getSeriesCount() - 1, i -> i >= 0, i -> i -
1)
                .forEach(i -> dataset.removeSeries(i));

            seriesMap.clear();
            colorManager.clear();
            seriesOrder.clear();
            listPanel.removeAll();
            listPanel.revalidate();
            listPanel.repaint();
            chartPanel.repaint();
        }

        @Override
        public void dispose() {
            executor.shutdown();
            super.dispose();
        }

        public static void main(String[] args) {
            SwingUtilities.invokeLater(() -> {
                try {
                    UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
                } catch (Exception e) {
                    e.printStackTrace();
                }
                new ChartWithExcel().setVisible(true);
            });
        }
    }
}

```

Файл AuthServer.java

```

package folder;

import com.sun.net.httpserver.HttpServer;
import com.sun.net.httpserver.HttpExchange;

```

```

import com.sun.net.httpserver.HttpHandler;
import java.io.*;
import java.net.InetSocketAddress;
import java.nio.charset.StandardCharsets;
import java.util.*;
import java.util.concurrent.*;
import java.util.stream.Collectors;

public class AuthServer {

    private static final Map<String, User> users = new ConcurrentHashMap<>();
    private static final Map<String, Session> sessions = new
ConcurrentHashMap<>();

    private static final int PORT = 8080;

    // Класс пользователя
    public static class User {
        private final String login;
        private final String passwordHash;
        private final String role;
        private final long createdAt;

        public User(String login, String password, String role) {
            this.login = login;
            this.passwordHash = Base64.getEncoder().encodeToString(
                password.getBytes(StandardCharsets.UTF_8));
            this.role = role;
            this.createdAt = System.currentTimeMillis();
        }

        public String getLogin() { return login; }
        public String getPasswordHash() { return passwordHash; }
        public String getRole() { return role; }
        public long getCreatedAt() { return createdAt; }

        public boolean checkPassword(String password) {
            String hash = Base64.getEncoder().encodeToString(
                password.getBytes(StandardCharsets.UTF_8));
            return passwordHash.equals(hash);
        }
    }

    // Класс сессии
    public static class Session {
        private final String token;
        private final String login;
        private final long createdAt;
        private final long expiresAt;

        public Session(String token, String login) {
            this.token = token;
            this.login = login;
            this.createdAt = System.currentTimeMillis();
            this.expiresAt = createdAt + TimeUnit.HOURS.toMillis(24);
        }

        public String getToken() { return token; }
        public String getLogin() { return login; }
        public long getCreatedAt() { return createdAt; }
        public long getExpiresAt() { return expiresAt; }
        public boolean isExpired() { return System.currentTimeMillis() >

```

```

    expiresAt; }
    }

    static {
        // Инициализация пользователей через Stream API
        List<User> defaultUsers = Arrays.asList(
            new User("admin", "admin123", "ADMIN"),
            new User("user1", "pass123", "USER"),
            new User("user2", "pass456", "USER"),
            new User("demo", "demo", "GUEST")
        );

        defaultUsers.forEach(user -> users.put(user.getLogin(), user));
    }

    public static void main(String[] args) {
        try {
            ExecutorService executor = Executors.newFixedThreadPool(10);

            0);

            HttpServer server = HttpServer.create(new InetSocketAddress(PORT),

            server.createContext("/api/login", new LoginHandler());
            server.createContext("/api/users", new UsersHandler());
            server.createContext("/api/sessions", new SessionsHandler());

            server.setExecutor(executor);
            server.start();

            System.out.println("=====");
            System.out.println(" Сервер запущен на порту " + PORT);
            System.out.println("=====");

            users.entrySet().stream()
                .sorted(Map.Entry.comparingByKey())
                .forEach(entry ->
                    System.out.printf(" %-12s [%s]\n",
                        entry.getKey(), entry.getValue().getRole()));

            System.out.println("=====");

            Runtime.getRuntime().addShutdownHook(new Thread(() -> {
                System.out.println("\nЗавершение сервера...");
                executor.shutdown();
                try {
                    if (!executor.awaitTermination(5, TimeUnit.SECONDS)) {
                        executor.shutdownNow();
                    }
                } catch (InterruptedException e) {
                    executor.shutdownNow();
                }
                server.stop(0);
            }));

        } catch (IOException e) {
            System.err.println("Ошибка запуска сервера: " + e.getMessage());
            e.printStackTrace();
        }
    }

    // Обработчик входа
    static class LoginHandler implements HttpHandler {

```

```

@Override
public void handle(HttpExchange exchange) throws IOException {
    try {
        if (!"POST".equals(exchange.getRequestMethod())) {
            sendResponse(exchange, 405, "ERROR: Method not allowed");
            return;
        }

        String body = new String(
            exchange.getRequestBody().readAllBytes(),
            StandardCharsets.UTF_8
        );

        // Stream API - парсим параметры
        Map<String, String> params = Arrays.stream(body.split("&"))
            .map(param -> param.split("="))
            .filter(kv -> kv.length == 2)
            .collect(Collectors.toMap(
                kv -> kv[0],
                kv -> kv[1],
                (v1, v2) -> v2
            ));

        String login = params.get("login");
        String password = params.get("password");

        if (login == null || login.isEmpty()) {
            sendResponse(exchange, 400, "ERROR: Login required");
            return;
        }

        if (password == null || password.isEmpty()) {
            sendResponse(exchange, 400, "ERROR: Password required");
            return;
        }

        // Stream API - поиск пользователя
        User user = Optional.ofNullable(users.get(login))
            .orElse(null);

        if (user == null || !user.checkPassword(password)) {
            System.out.println("Неудачный вход: " + login);
            sendResponse(exchange, 401, "ERROR: Invalid credentials");
            return;
        }

        // Создаем сессию
        String token = UUID.randomUUID().toString();
        Session session = new Session(token, login);
        sessions.put(token, session);

        System.out.println("/ Успешный вход: " + login + " | Роль: " +
user.getRole());

        String response = String.format("OK:%s:%s:%s", token, login,
user.getRole());
        sendResponse(exchange, 200, response);

    } catch (Exception e) {
        System.err.println("Ошибка сервера: " + e.getMessage());
        sendResponse(exchange, 500, "ERROR: Internal server error");
    }
}

```

```

    }
}

// Обработчик для просмотра пользователей
static class UsersHandler implements HttpHandler {
    @Override
    public void handle(HttpExchange exchange) throws IOException {
        try {
            // Stream API - формируем список пользователей
            String userList = users.values().stream()
                .sorted(Comparator.comparing(User::getLogin))
                .map(user -> String.format("%s (%s)", user.getLogin(),
user.getRole()))
                .collect(Collectors.joining("\n"));

            String response = "=== Пользователи ===\n" + userList +
                "\nBcero: " + users.size();

            sendResponse(exchange, 200, response);
        } catch (Exception e) {
            sendResponse(exchange, 500, "ERROR: " + e.getMessage());
        }
    }
}

// Обработчик активных сессий
static class SessionsHandler implements HttpHandler {
    @Override
    public void handle(HttpExchange exchange) throws IOException {
        try {
            // Удаляем истекшие сессии
            sessions.values().removeIf(Session::isExpired);

            // Stream API - фильтруем и сортируем активные сессии
            String sessionList = sessions.values().stream()
                .sorted(Comparator.comparing(Session::getCreatedAt))
                .map(session -> String.format("%s -> %s (создана: %s)",
                    session.getLogin(),
                    session.getToken().substring(0, 8),
                    new Date(session.getCreatedAt())))
                .collect(Collectors.joining("\n"));

            String response = "=== Активные сессии ===\n" +
                (sessionList.isEmpty() ? "Нет активных сессий" :
sessionList) +
                "\nBcero: " + sessions.size();

            sendResponse(exchange, 200, response);
        } catch (Exception e) {
            sendResponse(exchange, 500, "ERROR: " + e.getMessage());
        }
    }
}

private static void sendResponse(HttpExchange exchange, int code, String
response) throws IOException {
    exchange.sendResponseHeaders(code, response.getBytes().length);
    try (OutputStream os = exchange.getResponseBody()) {
        os.write(response.getBytes());
    }
}
}

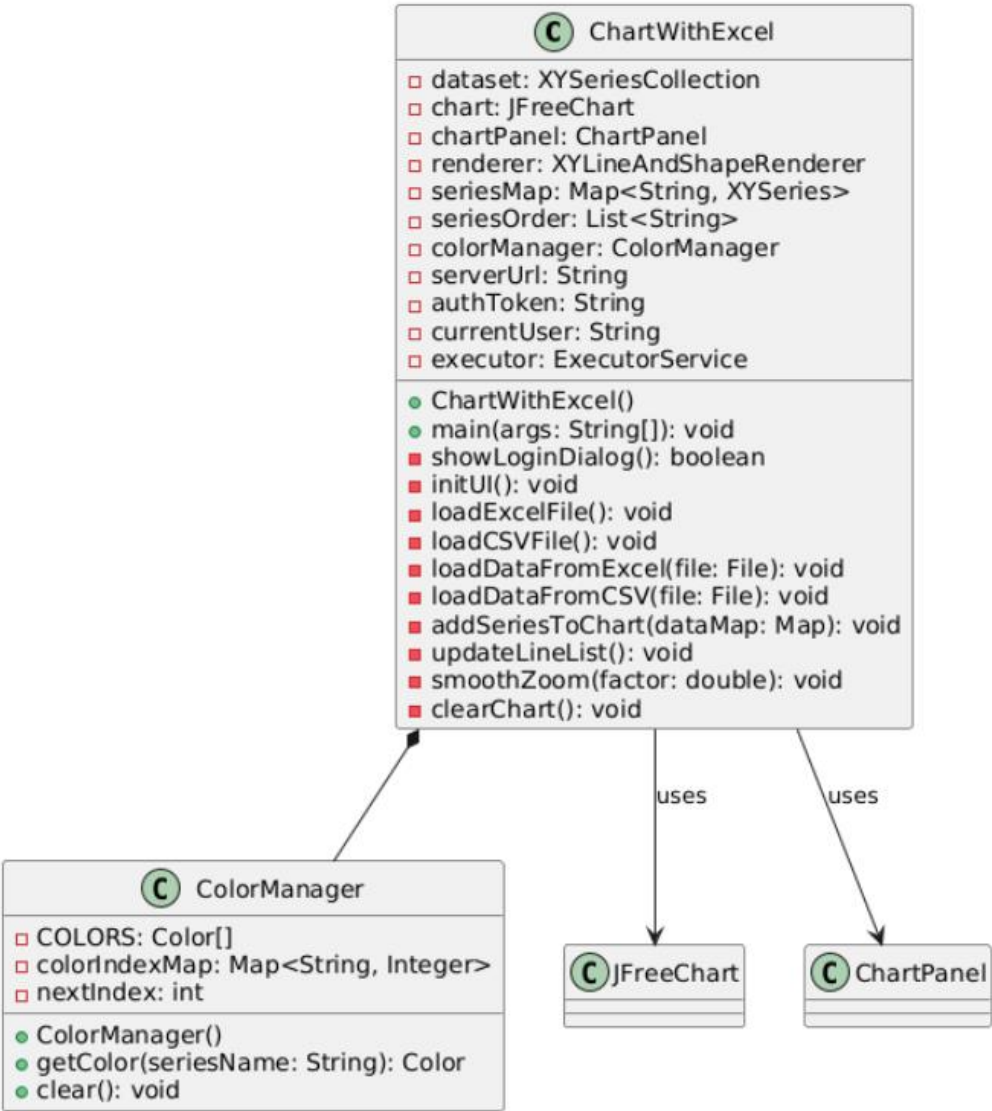
```

Приложение Б. UML-диаграммы

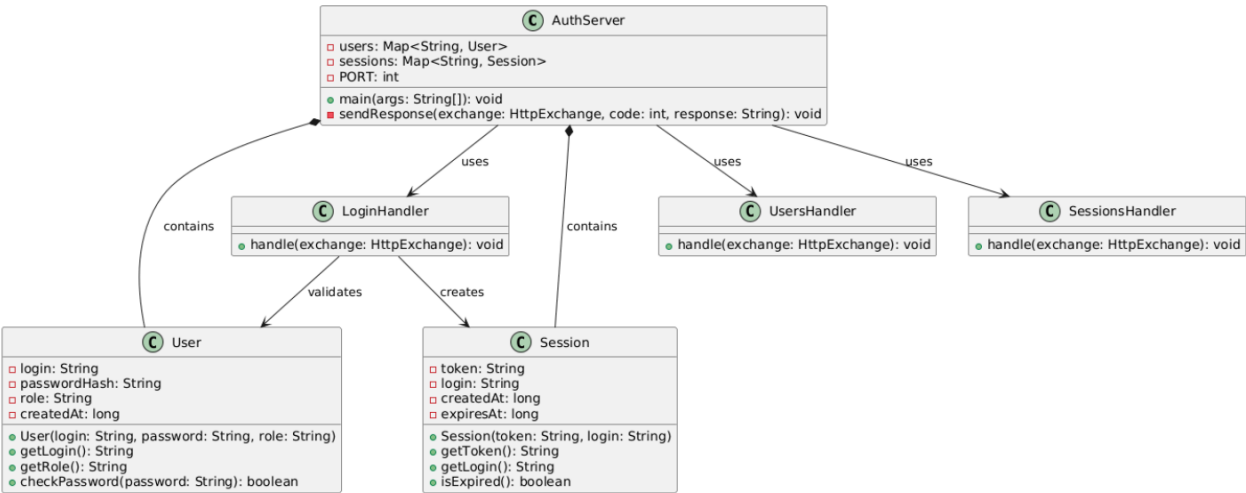
Приложение Б.1. UML-диаграмма вариантов использования



Приложение Б.2. UML-диаграмма классов

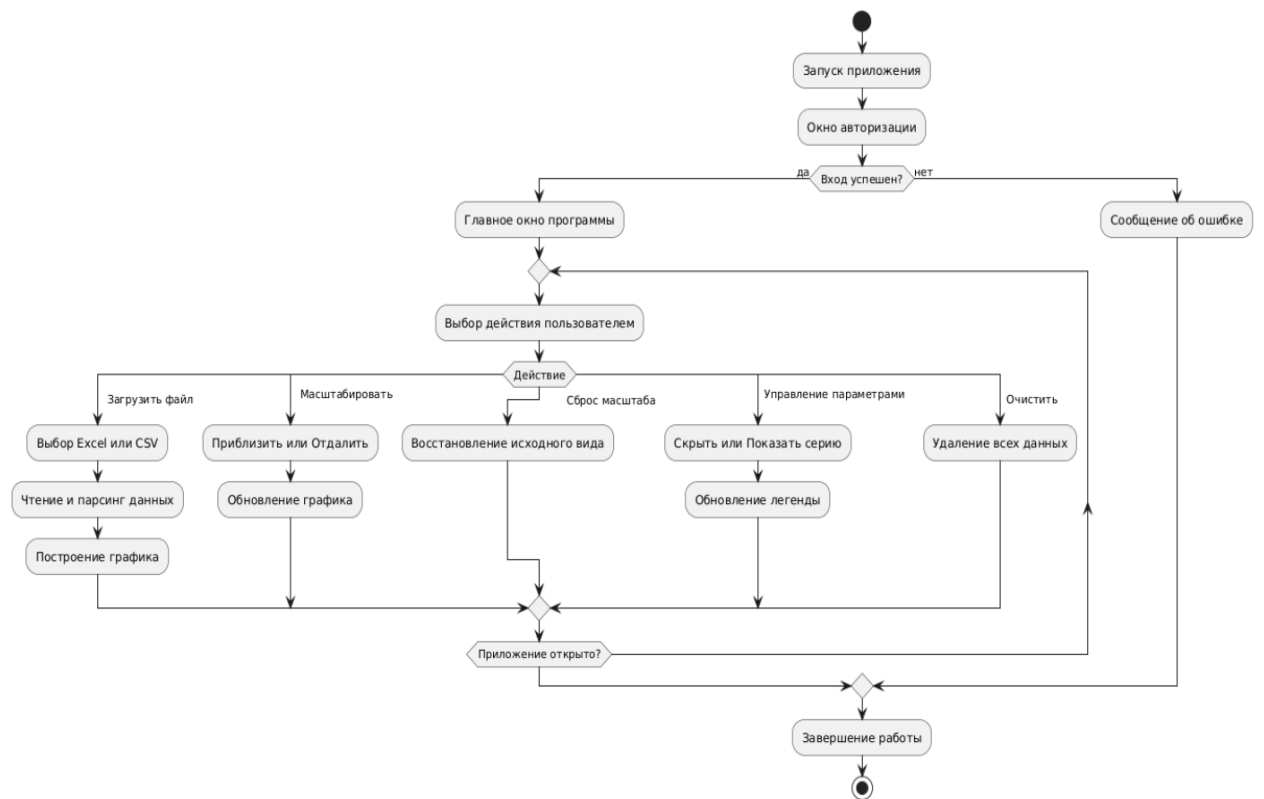


Клиент

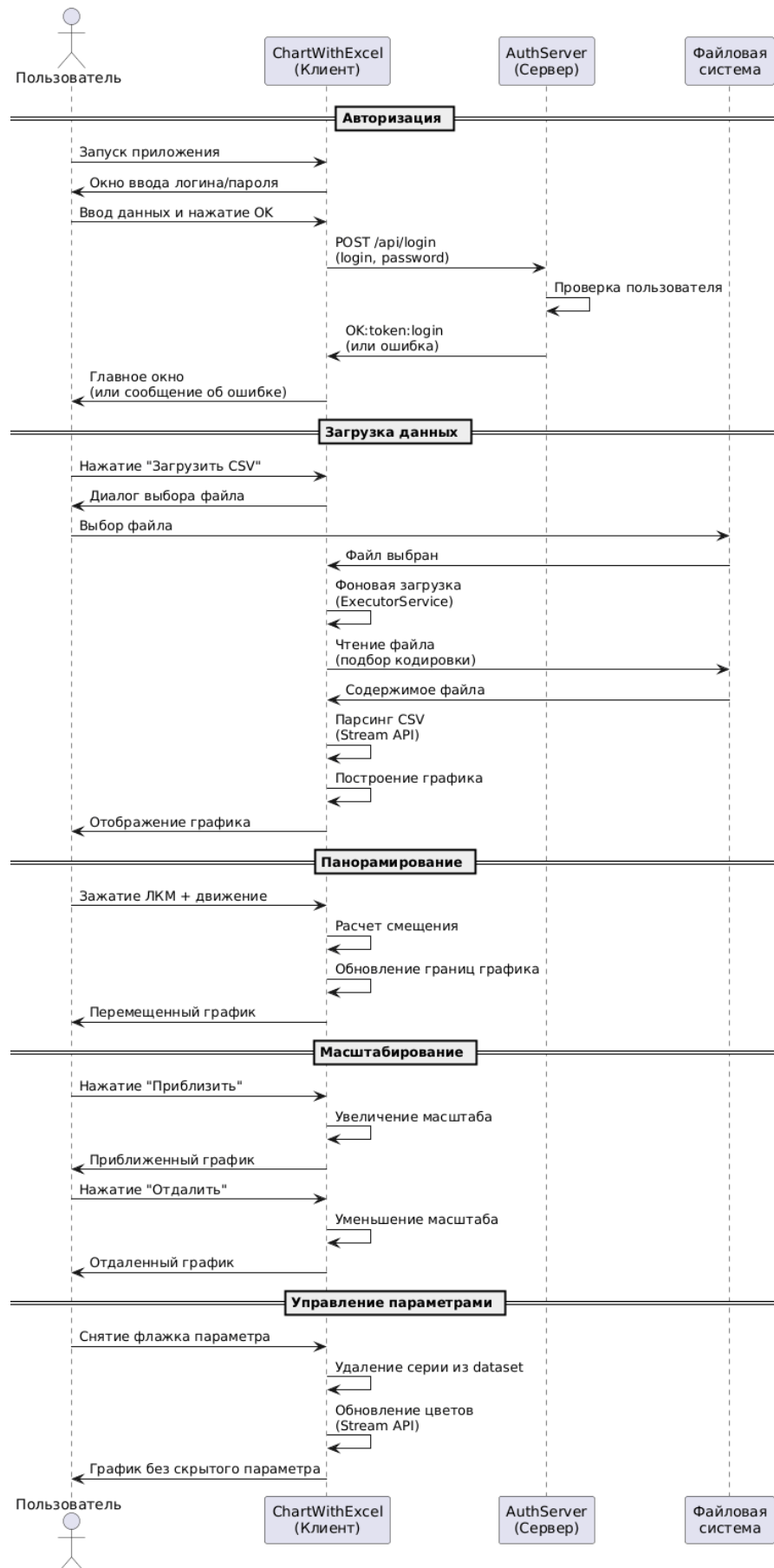


Сервер

Приложение Б.3. UML- диаграмма деятельности



Приложение Б.4. UML-диаграмма последовательности



Приложение Б.5. UML-диаграмма развертывания

