

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Кафедра «Вычислительная техника»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проектированию
по курсу «Программирование на языке Java»
на тему «Разработка многомодульного приложения на языке Java»

04.06.26
x ооооо
флв

Выполнил:

Студент группы 23ВВП1

Точнов Е.Е.

Приняли:

Юрова О.В.

Карамышева Н.С.

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

Программирование на языке Java

Студенту Июхнову Евгению Евгеньевичу Группа 23ВВ71

Тема проекта Разработка многомодульного приложения на языке Java

Исходные данные (технические требования) на проектирование

Разработать приложение текстовый редактор с поддержкой различных кодировок

Язык 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

Диаграммы языка UML:

диаграмма вариантов использования;

диаграмма классов;

диаграмма деятельности;

диаграмма развёртывания;

диаграмма последовательности.

3. Экспериментальная часть

Тестирование многомодульного приложения

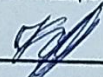
Срок выполнения проекта по разделам

1	
2	
3	
4	
5	
6	
7	
8	

Дата выдачи задания "14" 02. 2026

Дата защиты проекта " " "

Руководитель Карамышева Н.С.

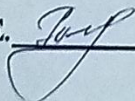


Задание получил

"14" марта

2026г.

Студент Точнов Е.Е.



РЕФЕРАТ	6
ВВЕДЕНИЕ.....	7
1 Анализ предметной области	8
1.1 Обзор предметной области.....	8
1.2 Функциональные требования	9
1.3 Выбор технологического стека	11
2 Архитектура приложения	12
2.1 Диаграммы (работоспособность системы).....	12
2.1.1 Диаграмма вариантов использования	12
2.1.2 Диаграмма классов.....	13
2.1.3 Диаграмма последовательности	14
2.1.4 Диаграмма деятельности	15
2.1.5 Диаграмма развёртывания.....	16
2.2 Структура модулей.....	16
2.3 Описание ключевых компонентов.....	17
2.4 Многопоточная архитектура	17
3 Реализация ключевых компонентов	19
3.1 Реализация сетевого взаимодействия	19
3.2 Многопоточная модель и отзывчивость интерфейса	19
3.3 Обработка текстовых кодировок в редакторе.....	20
3.4 Валидация данных и обработка исключений	20
3.5 Представление удалённой файловой системы	21
4 Описание интерфейса приложения	22
4.1 Окно авторизации.....	22
4.2 Окно представления удалённой файловой системы.....	22
4.3 Диалог создания нового файла	23
4.4 Окно редактора	24
5 Тестирование приложения	26

5.1 Сценарии тестирования	26
5.2 Выявленные ограничения	27
ЗАКЛЮЧЕНИЕ	28
СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ	29
ПРИЛОЖЕНИЕ А	31
A1. Файл FTPInteractions.java	32
A2. Файл FTPTableModel.java.....	34
A3. Файл MainApp.java.....	36
A4. Файл MainEditor.java.....	51
A5. Файл HostAddressException.java	57
A6. Файл InvalidLoginException.java	57
A7. Файл LoginException.java	57
ПРИЛОЖЕНИЕ Б.....	58
Б1. Диаграмма вариантов использования	59
Б2. Диаграмма классов	60
Б3. Диаграмма последовательности	61
Б4. Диаграмма деятельности.....	62
Б5. Диаграмма развёртывания	63

РЕФЕРАТ

JAVA, SWING, СЕТЕВОЕ ПРОГРАММИРОВАНИЕ, ФАЙЛОВЫЙ СЕРВЕР, FTP, ТЕКСТОВЫЙ РЕДАКТОР, ТЕКСТОВЫЕ КОДИРОВКИ, КЛИЕНТ-СЕРВЕРНАЯ АРХИТЕКТУРА

Целью работы является разработка многомодульного графического приложения текстового редактора на языке Java с поддержкой современных и устаревших текстовых кодировок для работы с файлами напрямую на сервере.

В ходе выполнения курсовой работы спроектированы и реализованы: авторизация пользователя на сервере (loginPanel); перемещение по удалённой файловой системе (explorerPanel); обработка исключений связанных с FTP соединением (HostAddressException, LoginException); взаимодействие с FTP сервером (FTPInteractions) и табличная представление каталога удалённой файловой системы (FTPTableModel).

Приложение обеспечивает удобный графический интерфейс для взаимодействия с текстовыми файлами на удалённой файловой системе.

ВВЕДЕНИЕ

Разработка и администрирование программных систем неразрывно связаны с необходимостью управления файлами на удаленных серверах. Несмотря на наличие мощных интегрированных сред разработки (IDE) с поддержкой удаленной работы, сохраняется высокая потребность в легких, специализированных графических приложениях. Такие инструменты позволяют быстро просматривать и редактировать конфигурационные файлы, логи или скрипты непосредственно на сервере, избегая ручного процесса скачивания файла на локальную машину, его изменения и последующей обратной загрузки. Протокол FTP (File Transfer Protocol) остается одним из наиболее распространенных и универсальных стандартов для решения подобных задач.

Целью данной курсовой работы является разработка графического приложения на языке Java, представляющего собой текстовый редактор с интегрированным FTP-клиентом, который позволяет осуществлять навигацию по файловой системе удаленного сервера, а также открывать, редактировать и сохранять текстовые файлы непосредственно в удаленном хранилище.

Для достижения поставленной цели в ходе выполнения работы необходимо решить следующие задачи:

- Провести анализ предметной области и изучить особенности взаимодействия с FTP-серверами.
- Сформулировать функциональные требования к приложению (авторизация, навигация по каталогам, чтение и запись файлов, поддержка различных текстовых кодировок).
- Обосновать выбор технологического стека для реализации проекта.
- Спроектировать архитектуру приложения, обеспечив разделение ответственности между модулями сетевого взаимодействия, хранения данных и пользовательского интерфейса.
- Реализовать механизмы многопоточности для выполнения сетевых операций без блокировки графического интерфейса (GUI).
- Реализовать систему обработки ошибок и исключительных ситуаций.
- Провести тестирование разработанного приложения на корректность выполнения основных сценариев использования.

1 Анализ предметной области

1.1 Обзор предметной области

Управление файлами на удаленных серверах является одной из ключевых задач при разработке, тестировании и администрировании программных систем. Традиционный рабочий процесс редактирования конфигурационных файлов, скриптов или логов на удаленном хосте часто выглядит следующим образом: подключение через специализированный FTP-клиент, скачивание файла на локальный диск, открытие его во внешнем текстовом редакторе, внесение изменений и последующая обратная загрузка на сервер.

Данный подход имеет ряд существенных недостатков:

- Риск потери актуальности: пока файл редактируется локально, он может быть изменен на сервере другим пользователем или процессом, что приведет к конфликту версий при обратной загрузке.
- Человеческий фактор: высока вероятность забыть загрузить измененный файл обратно на сервер или загрузить его не в ту директорию.
- Избыточность действий: необходимость переключения между несколькими приложениями (FTP-клиент и текстовый редактор) снижает эффективность работы.

Протокол FTP (File Transfer Protocol) остается одним из наиболее распространенных и универсальных стандартов для передачи файлов в сетях TCP/IP. Разработка специализированного графического приложения, объединяющего функционал FTP-клиента и текстового редактора, позволяет нивелировать указанные недостатки. Такое приложение обеспечивает бесшовный рабочий процесс: файл считывается из файловой системы сервера, редактируется и сохраняется обратно по тому же пути, что минимизирует количество ручных операций и снижает вероятность ошибок.

1.2 Функциональные требования

На основе анализа предметной области и специфики разрабатываемого приложения были сформулированы следующие функциональные требования:

Для модуля авторизации и подключения:

- Возможность ввода адреса хоста (IP или доменное имя), логина и пароля.
- Валидация введенных данных на стороне клиента (проверка на пустые значения) с генерацией пользовательских исключений (`HostAddressException`, `LoginException`) и выводом понятных сообщений об ошибках.
- Безопасное хранение пароля в процессе ввода (использование компонента `JPasswordField`).

Для модуля навигации и отображения файловой системы:

- Отображение текущего абсолютного пути на сервере с возможностью его ручного изменения (по нажатию клавиши `Enter`).
- Возможность перехода в родительскую директорию одной кнопкой.
- Представление списка файлов и директорий в виде табличного интерфейса (`JTable`) с колонками: «Имя», «Размер» (с автоматическим форматированием в Б, КБ, МБ), «Дата изменения» и «Тип» (Файл/Каталог).
- Визуальное различие файлов и директорий в интерфейсе.

Для модуля работы с файлами:

- Возможность открытия текстовых файлов двойным кликом для редактирования.
- Возможность создания нового текстового файла в текущей директории с заданием его имени.
- Сохранение изменений непосредственно на FTP-сервер по исходному пути без промежуточного сохранения на локальный диск.

Для модуля текстового редактирования:

- Поддержка множественных текстовых кодировок как при чтении, так и при записи файла (UTF-8, UTF-16, KOI8-R, KOI8-U, DOS-866, DOS-437). Это критически важно для корректного отображения файлов, созданных в различных операционных системах.

- Выбор кодировки исходного файла и кодировки сохранения через элементы меню приложения.

Для модуля многопоточности и стабильности:

- Выполнение всех сетевых операций (подключение, получение списка файлов, скачивание, загрузка) в фоновых потоках с использованием механизмов `SwingWorker` и `ExecutorService`.

- Предотвращение блокировки графического интерфейса (`Event Dispatch Thread`) во время ожидания ответа от сервера.

- Корректная обработка сетевых ошибок (обрыв соединения, таймаут, отказ в доступе) с информированием пользователя через диалоговые окна.

1.3 Выбор технологического стека

Для реализации поставленных задач был выбран следующий стек технологий, обеспечивающий надежность, кроссплатформенность и соответствие функциональным требованиям:

- Язык программирования Java (JDK 26) – основной язык разработки, предоставляющий строгую типизацию, встроенную поддержку многопоточности и богатый набор стандартных библиотек.

- Библиотека Java Swing – стандартный фреймворк для создания графического пользовательского интерфейса (GUI) в Java. Используются компоненты JFrame, JDialog, JTable, CardLayout для организации навигации между экранами авторизации и проводника.

- Библиотека Apache Commons Net – проверенное временем решение с открытым исходным кодом для реализации сетевых протоколов. Класс FTPClient данной библиотеки инкапсулирует низкоуровневую работу с TCP-сокетами и FTP-командами, предоставляя удобный высокоуровневый API для аутентификации, навигации и передачи данных.

- Механизмы многопоточности Java (java.util.concurrent) – классы SwingWorker и ExecutorService используются для асинхронного выполнения операций ввода-вывода. Это гарантирует, что интерфейс приложения останется отзывчивым даже при медленном сетевом соединении.

- Пакет java.nio.charset – используется для преобразования массивов байтов, полученных с сервера, в строки (String) и обратно. Поддержка классов Charset и StandardCharsets позволяет гибко работать с различными кодировками, что является обязательным требованием для текстового редактора.

- Пользовательские исключения (Exception) – создание собственных классов исключений (LoginException, HostAddressException, InvalidLoginException) позволяет четко разделять ошибки валидации ввода пользователя от сетевых ошибок библиотеки Apache Commons Net.

2 Архитектура приложения

2.1 Диаграммы (работоспособность системы)

2.1.1 Диаграмма вариантов использования

На рисунке 1 показана диаграмма вариантов использования, на которой представлены два актёра – Пользователь и FTP-сервер – и варианты использования приложения. Пользователь инициирует ввод данных для авторизации на сервере, прохождение по удалённой файловой системе, редактирование файла и т.д. Сервер отвечает за установление соединения, отправку листинга каталога, файлов и сохранение изменённых файлов у себя. Связи «include» отражают обязательные зависимости: так, нельзя установить соединение с сервером не проверив валидность введенных данных для подключения, а открытие файла подразумевает предварительное скачивание файла с сервера и т.д.

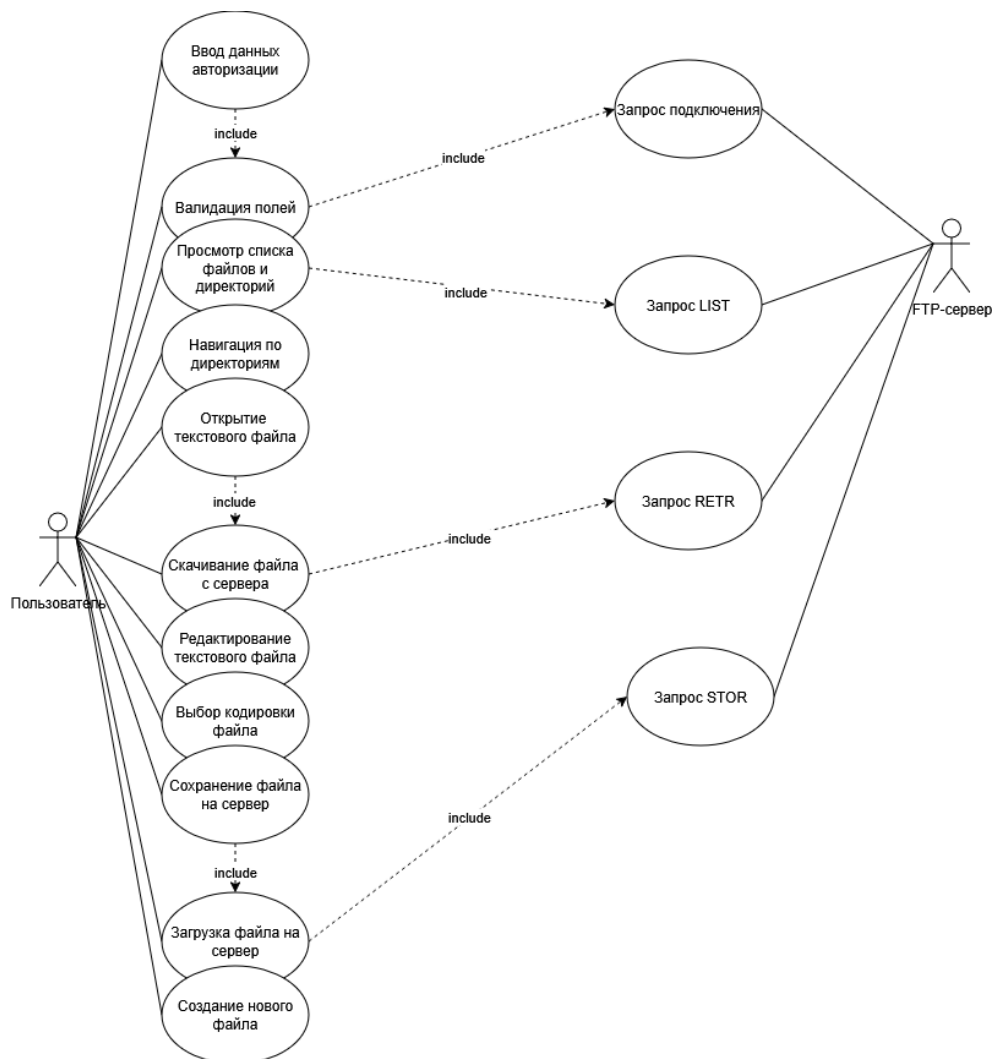


Рисунок 1 - диаграмма вариантов использования

2.1.2 Диаграмма классов

На рисунке 2 изображена диаграмма классов приложения, задействованных в реализации. Так основной класс MainApp содержит в себе класс FTPTableModel, который необходим для табличного представления каталогов удалённой файловой системы. Для получения данных файлов, листинга директорий, переходов по файловой системе он использует FTPInteractions, который в свою очередь делегирует все операции, связанные с взаимодействием с FTP сервером библиотеке FTPClient, которая также используется для установления начального соединения с сервером. Помимо этого, для представления интересующего файла основной класс программы создаёт модальное окно MainEditor, которое также использует FTPInteractions, но уже для сохранения изменённого файла на сервере.

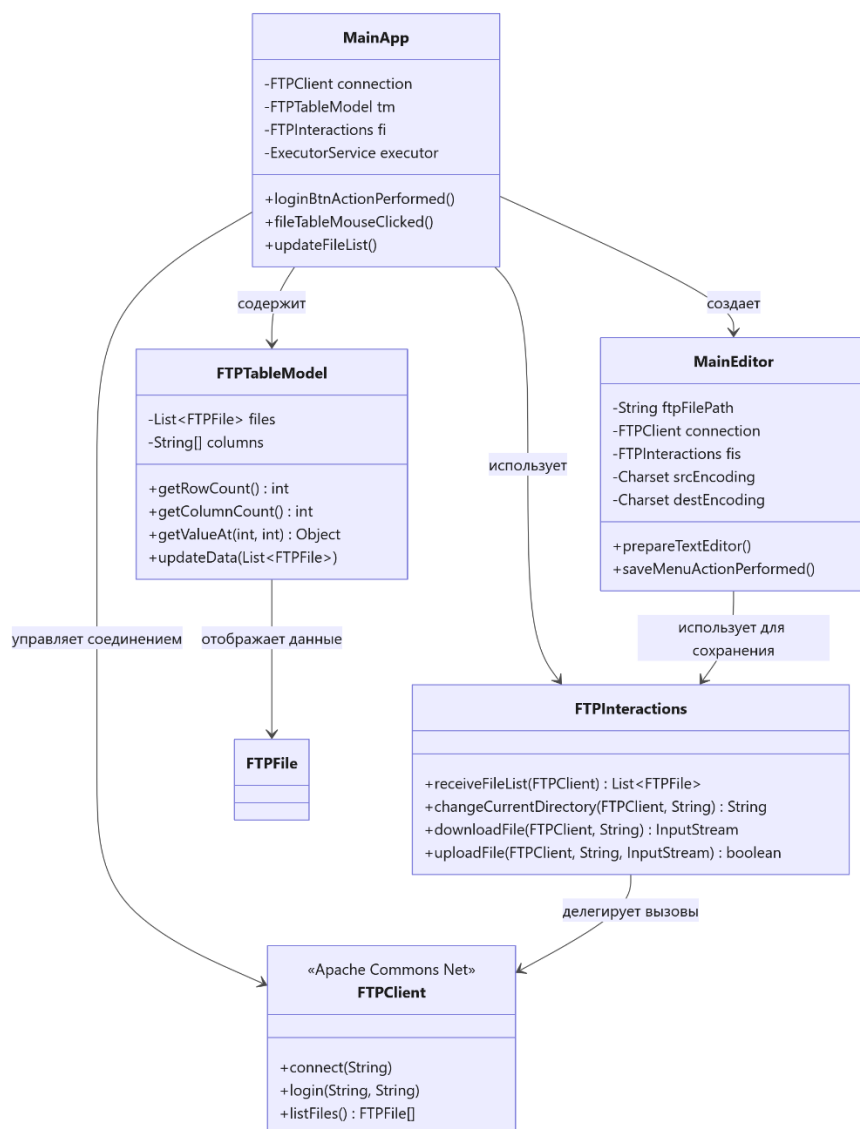


Рисунок 2 - диаграмма классов

2.1.3 Диаграмма последовательности

На рисунке 3 изображена диаграмма последовательности, отражающая сценарий взаимодействия пользователя с программой: в начале пользователь совершает двойной клик по файлу, затем этот файл загружается с сервера в оперативную память клиента в виде набора байтов, после чего этот массив байт передаётся в компонент, который отвечает за отображение текстового редактора, преобразует полученный набор байт в целевую кодировку клиента и отображает итоговое окно с содержимым удалённого файла. После внесения изменений и нажатия кнопки сохранить новые данные преобразуются в целевую кодировку, которая должна храниться на сервере, преобразованные байты передаются в виде потока для сохранения на файловой системе сервера, после чего происходит получение статуса сохранения и в случае если сохранение успешно отображается всплывающее окно с соответствующим сообщением, а табличное представление каталога обновляется, чтобы получить актуальную информацию о каталоге в котором произошло изменение файла.

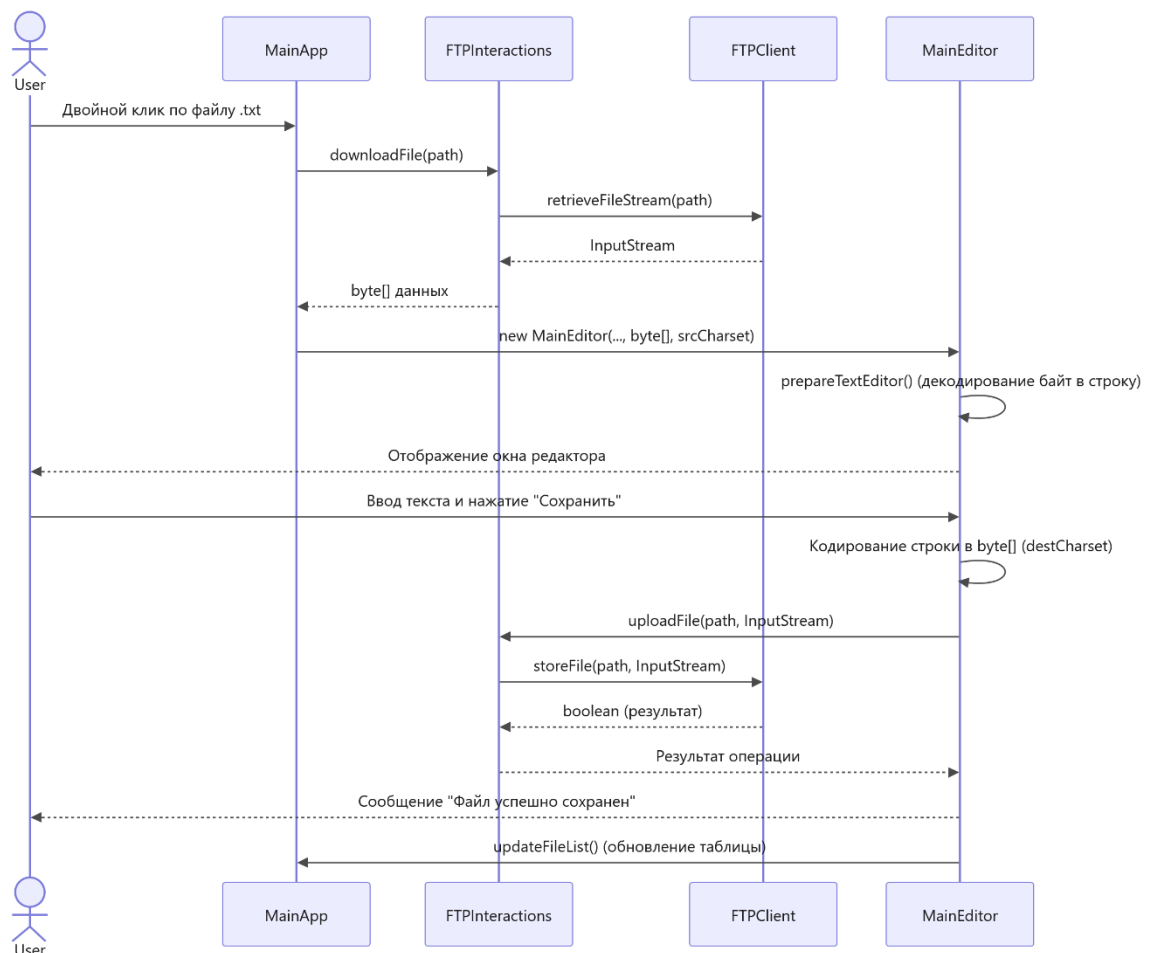


Рисунок 3 - диаграмма последовательности

2.1.4 Диаграмма деятельности

На рисунке 4 показана диаграмма деятельности, которая иллюстрирует процесс работы пользователя с приложением, начиная от авторизации и заканчивая редактированием и сохранением файла на FTP-сервере.

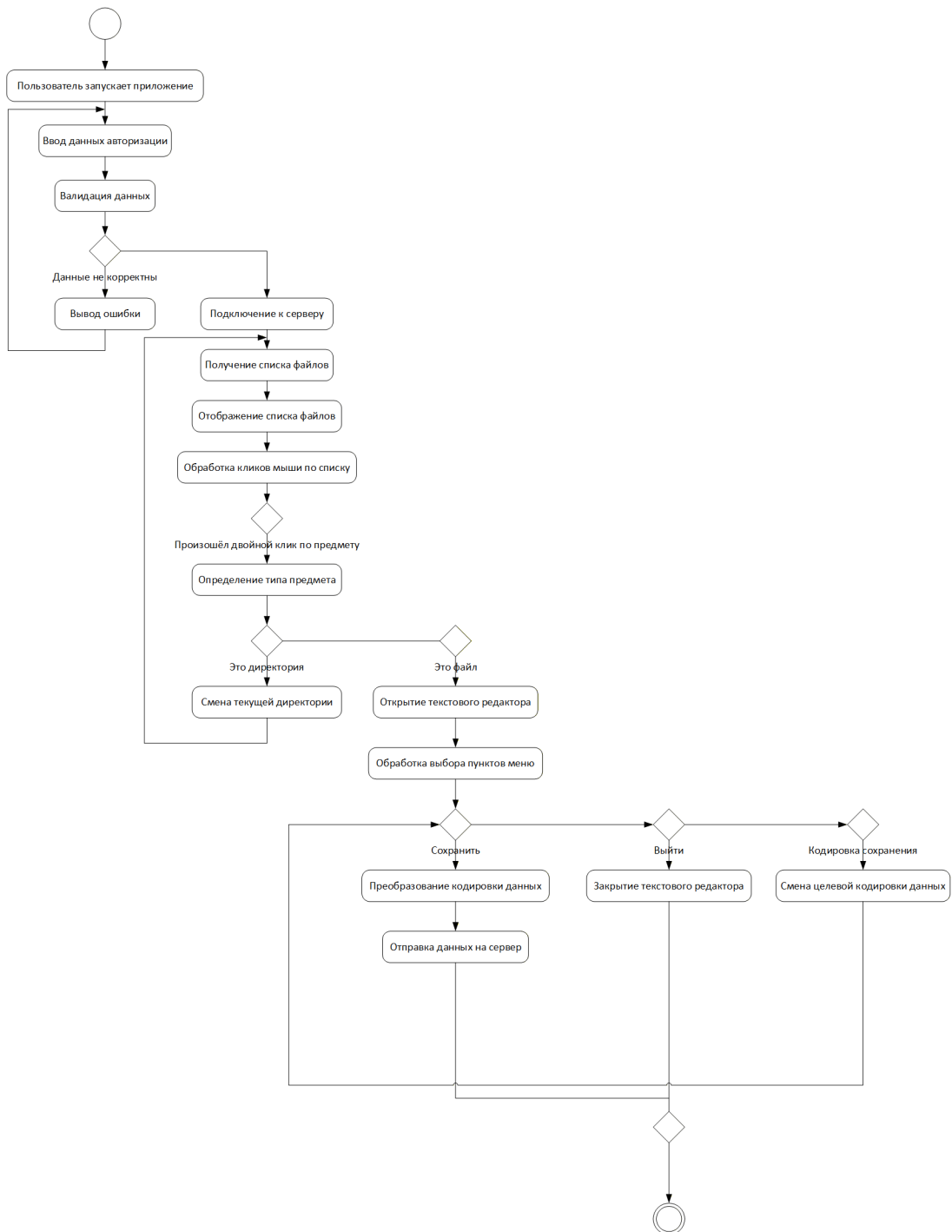


Рисунок 4 – диаграмма деятельности

2.1.5 Диаграмма развёртывания

На рисунке 5 показана диаграмма развёртывания, которая демонстрирует физическую архитектуру системы и распределение программных компонентов по вычислительным узлам. Приложение следует клиент-серверной топологии, где клиентская часть взаимодействует с внешним FTP-сервером через сетевой протокол.

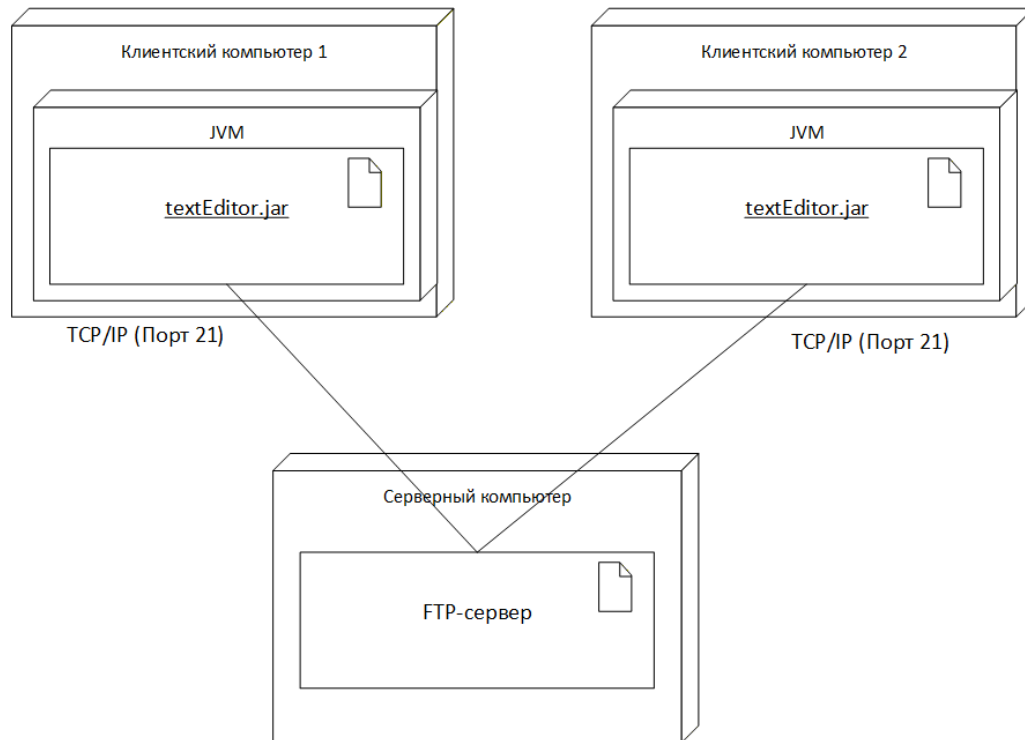


Рисунок 5 – диаграмма развёртывания

2.2 Структура модулей

Разработанное приложение имеет монолитную архитектуру с четким разделением ответственности между компонентами. Проект организован в виде пакетов языка Java, что обеспечивает логическую группировку классов:

- `com.mycompany.java_kursach` – корневой пакет, содержащий основные классы приложения: `MainApp` – главный класс приложения, реализующий основное окно (`JFrame`), управление навигацией между панелями (авторизация и проводник) и координацию пользовательских действий; `MainEditor` – модальное диалоговое окно (`JDialog`) для просмотра и редактирования содержимого текстовых файлов с поддержкой выбора кодировок; `FTPInteractions` – сервисный класс, инкапсулирующий всю логику взаимодействия с FTP-сервером через библиотеку `Apache Commons Net`; `FTPTableModel` – класс модели данных,

наследующий `AbstractTableModel`, предназначенный для отображения списка файлов и директорий в компоненте `JTable`.

- `com.mycompany.java_kursach.Exceptions` – пакет, содержащий пользовательские классы исключений для валидации данных на стороне клиента (`HostAddressException`, `LoginException`).

2.3 Описание ключевых компонентов

Архитектура приложения следует паттерну, близкому к Model-View-Controller (MVC), где:

- Модель (Model) - класс `FTPTableModel` управляет данными о файлах (`List<FTPFile>`), а класс `FTPInteractions` выступает в роли провайдера данных, выполняя запросы к удаленному серверу.

- Представление (View) - графический интерфейс, сгенерированный средствами Java Swing (`MainApp`, `MainEditor`), отвечает исключительно за отображение информации и сбор пользовательского ввода.

- Контроллер (Controller) - методы-обработчики событий (например, `loginBtnActionPerformed`, `fileTableMouseClicked`) в классе `MainApp` координируют вызовы методов модели и обновление представления.

Класс `FTPInteractions` является фасадом для работы с сетью. Он предоставляет высокоуровневые методы (`receiveFileList`, `changeCurrentDirectory`, `downloadFile`, `uploadFile`), скрывая детали работы с объектом `FTPClient` и преобразования типов данных (например, преобразование массива `FTPFile[]` в `List<FTPFile>`).

Класс `MainEditor` реализует гибкую работу с текстовыми кодировками. При открытии файла байтовый поток преобразуется в строку с использованием исходной кодировки (`srcEncoding`), а при сохранении текст кодируется в байтовый массив с использованием целевой кодировки (`destEncoding`) через механизмы `ByteBuffer` и `CharBuffer` пакета `java.nio.charset`.

2.4 Многопоточная архитектура

Критически важным аспектом архитектуры desktop-приложений с сетевым взаимодействием является предотвращение блокировки `Event Dispatch Thread` (EDT) – главного потока, отвечающего за отрисовку графического интерфейса.

В данном проекте для решения этой задачи применяется гибридный подход к многопоточности:

- Для разовых асинхронных задач, результат которых должен быть отображен в графическом интерфейсе (подключение к серверу, смена директории, скачивание/загрузка файла), используется класс `SwingWorker`. Он выполняет тяжелые сетевые операции в фоновом потоке (`doInBackground`), а обновление интерфейса безопасно производит в EDT (`done`).

- Для фоновых задач, не требующих прямого обновления графического интерфейса (например, первоначальное получение списка файлов), используется пул потоков `ExecutorService` (реализация `newSingleThreadExecutor`), что гарантирует упорядоченное выполнение сетевых запросов и экономит ресурсы системы.

3 Реализация ключевых компонентов

3.1 Реализация сетевого взаимодействия

Класс FTPInteractions выступает в роли фасада, инкапсулирующего всю логику работы с протоколом FTP через библиотеку Apache Commons Net. Он оперирует объектом FTPClient, передаваемым ему в качестве аргумента, что обеспечивает слабую связанность компонентов.

Ключевые методы класса:

- receiveFileList(FTPClient): возвращает список файлов и директорий текущего каталога в виде List<FTPFile>, преобразуя массив, возвращаемый методом connection.listFiles().

- downloadFile(FTPClient, String): инициирует поток ввода (InputStream) для чтения данных файла с сервера по указанному пути. После получения потока обязательно вызывается connection.completePendingCommand() для корректного завершения FTP-транзакции.

- uploadFile(FTPClient, String, InputStream): перед записью данных метод явно устанавливает тип передачи FTP.BINARY_FILE_TYPE, что гарантирует сохранность любых текстовых файлов (включая файлы с нестандартными переносами строк) без искажений при передаче. Возвращает булево значение, указывающее на успешность операции.

3.2 Многопоточная модель и отзывчивость интерфейса

Одной из критических задач при разработке GUI-приложений с сетевым взаимодействием является предотвращение блокировки Event Dispatch Thread (EDT) – главного потока, отвечающего за отрисовку интерфейса и обработку событий мыши/клавиатуры.

В проекте MainApp реализован гибридный подход к многопоточности:

- Использование SwingWorker - для операций, результат которых должен быть немедленно отражен в интерфейсе (авторизация, смена директории, скачивание файла), создаются анонимные классы, наследующие SwingWorker. Сетевой запрос выполняется в методе doInBackground(), а обновление GUI (например, вызов tm.updateData() или отображение JOptionPane) происходит в методе done(), который автоматически выполняется в EDT.

- Использование `ExecutorService` - для фоновых задач, не требующих прямого манипулирования компонентами `Swing` (например, первичное получение списка файлов в методе `updateFileList()`), используется пул из одного потока (`Executors.newSingleThreadExecutor()`). Это гарантирует последовательное выполнение сетевых запросов и предотвращает гонки данных при обновлении модели таблицы.

3.3 Обработка текстовых кодировок в редакторе

Класс `MainEditor` реализует гибкий механизм чтения и записи текстовых файлов с поддержкой различных кодировок (`UTF-8`, `UTF-16`, `KOI8-R`, `DOS-866` и др.). Это достигается за счет использования пакетов `java.nio.charset` и `java.nio`.

- Чтение файла - при открытии файла массив байтов (`initialString`), полученный с FTP-сервера, декодируется в строку с использованием исходной кодировки (`srcEncoding`), выбранной пользователем в главном окне.

- Сохранение файла - при нажатии кнопки «Сохранить» текст из компонента `JEditorPane` преобразуется обратно в массив байтов с использованием целевой кодировки (`destEncoding`), которую пользователь может изменить в меню редактора. Полученный `InputStream` передается в метод `uploadFile` класса `FTPInteractions`.

3.4 Валидация данных и обработка исключений

Для повышения надежности приложения и улучшения пользовательского опыта реализована ранняя валидация данных на стороне клиента, до попытки установления сетевого соединения.

В пакете `com.mycompany.java_kursach.Exceptions` определены пользовательские классы исключений:

- `HostAddressException`: генерируется, если поле адреса хоста пусто.
- `LoginException`: генерируется, если поле логина пусто.

В методе `loginBtnActionPerformed` класса `MainApp` эти исключения перехватываются, и пользователю выводится понятное сообщение через `JOptionPane`, что предотвращает выброс неконтролируемых сетевых исключений библиотеки `FTPClient` при заведомо некорректных входных данных.

3.5 Представление удалённой файловой системы

Для отображения списка файлов используется компонент JTable, данные для которого предоставляет класс FTPTableModel, наследующий AbstractTableModel. Класс хранит список объектов FTPFile и переопределяет метод getValueAt, используя конструкцию switch для возврата нужного атрибута файла в зависимости от колонки:

- Для колонки «Размер» вызывается вспомогательный метод formatSize(long size), который преобразует размер из байт в читаемый формат (Б, КБ, МБ) с округлением до одного или двух знаков после запятой.

- Для колонки «Дата» используется класс SimpleDateFormat с паттерном «дд.мм.гггг чч:мм» для локализованного отображения времени последней модификации файла.

- Метод updateData(List<FTPFile> newFiles) заменяет внутренний список и вызывает fireTableDataChanged(), что инициирует перерисовку таблицы в интерфейсе.

4 Описание интерфейса приложения

4.1 Окно авторизации

На рисунке 4 представлено окно для ввода данных авторизации на сервере. Сюда входят поле адреса удалённого сервера, логина и пароля, а также кнопка для входа на удалённый сервер.

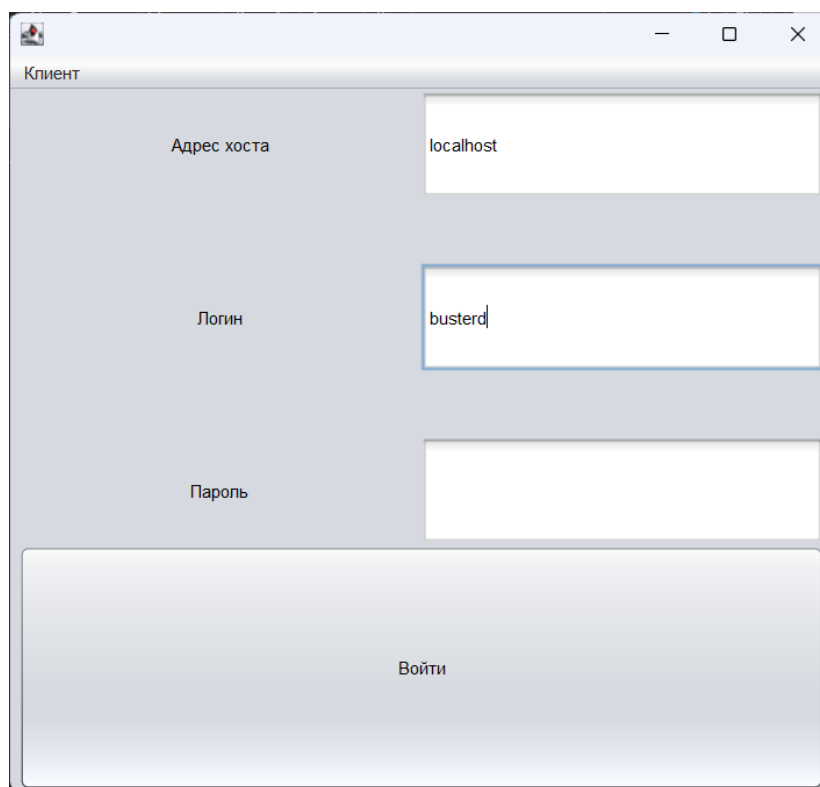


Рисунок 4 - окно авторизации

4.2 Окно представления удалённой файловой системы

На рисунке 5 показано окно, представляющее текущую директорию файловой системы сервера. Оно содержит в себе кнопку для перехода на родительский каталог, адресную строку, представляющую текущий абсолютный путь внутри файловой системы, табличное представление текущего каталога, а также кнопки для создания нового файла и отключения от сервера. Также на рисунке 6 показано меню, которое позволяет выбрать пользователю какую кодировку использовать для чтения файлов на сервере.

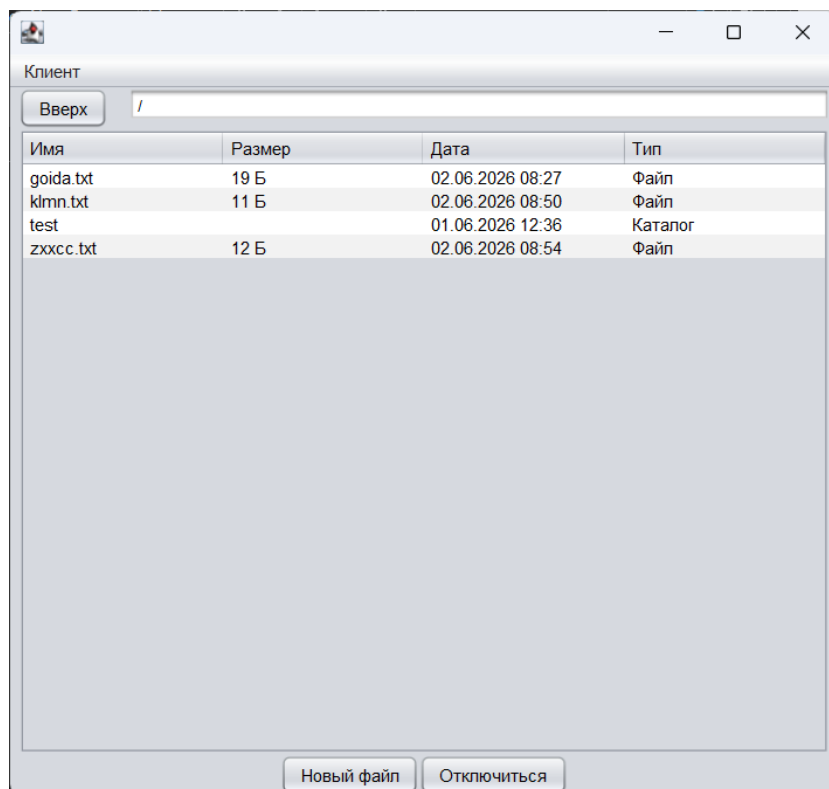


Рисунок 5 – окно удалённой файловой системы

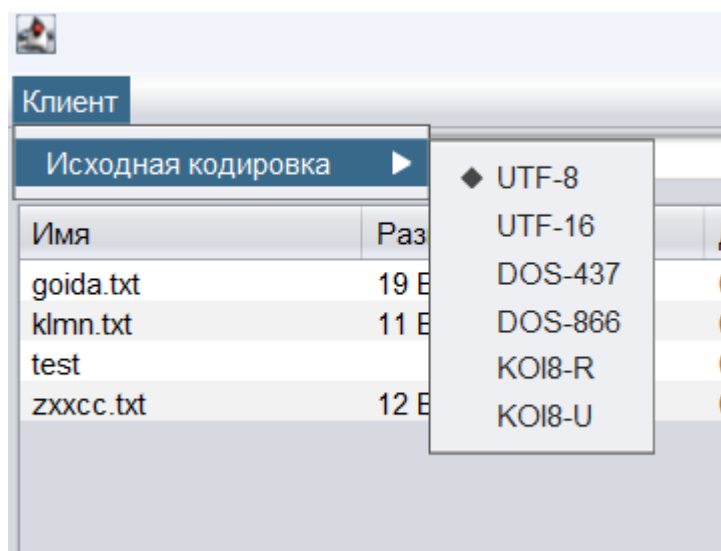


Рисунок 6 – меню выбора кодировки для чтения файлов

4.3 Диалог создания нового файла

Для запроса создания нового файла использовалось встроенное средство Java Swing для создания всплывающих диалоговых окон. На рисунке 7 представлен диалог запрашивающий у пользователя ввод имени для нового файла, состоящий из строки ввода, кнопок подтверждения и отмены.

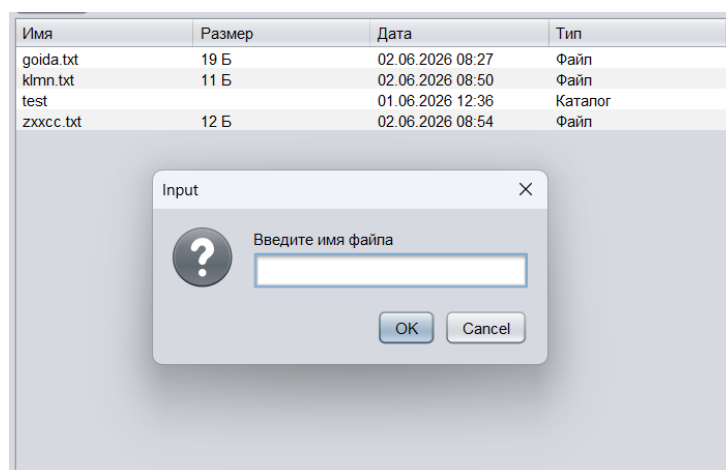


Рисунок 7 – диалог создания нового файла

4.4 Окно редактора

На рисунке 8 представлено окно текстового редактора, которое состоит из текстового поля. Также на рисунках 9 и 10 показаны меню «Файл» и «Изменить».

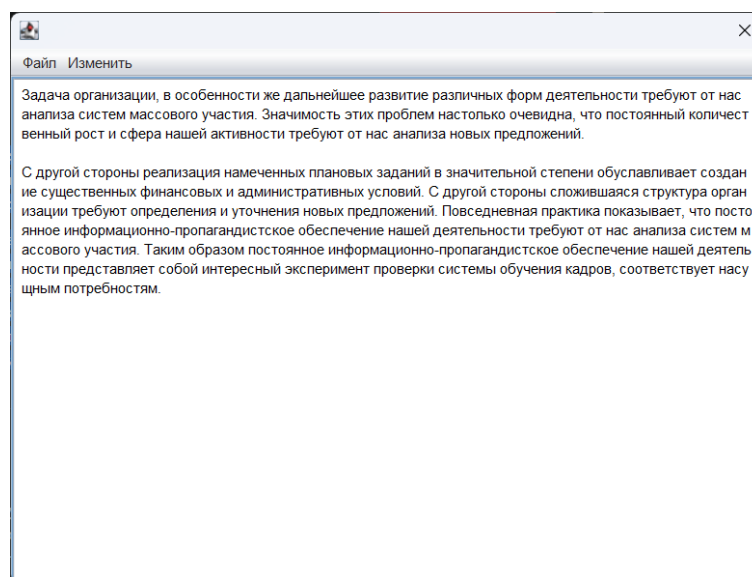


Рисунок 8 – окно текстового редактора

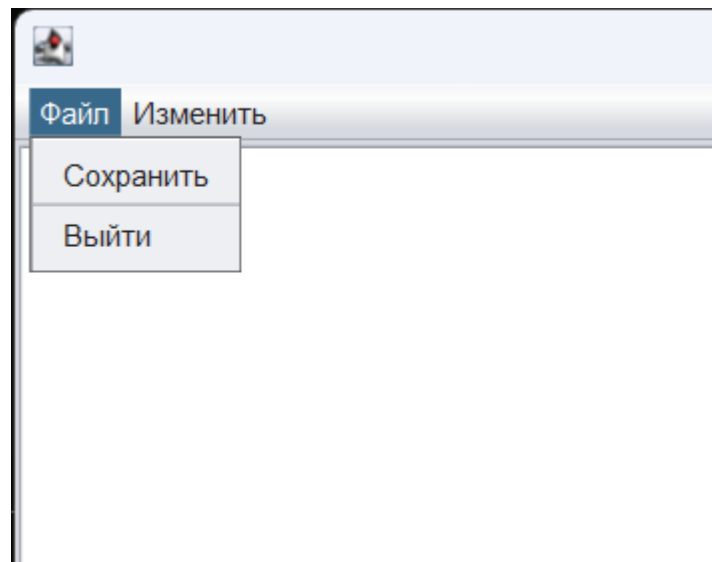


Рисунок 9 – меню «Файл» текстового окна

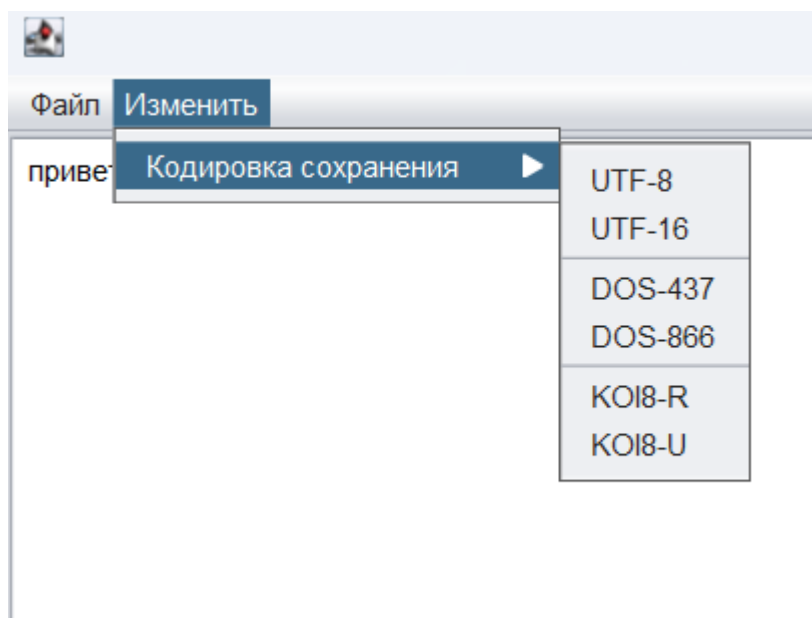


Рисунок 10 – меню «Изменить» текстового окна

5 Тестирование приложения

5.1 Сценарии тестирования

Тестирование проводилось на локальной машине с заранее запущенным FTP-сервером FileZilla. Проверенные сценарии приведены в таблице 1.

Таблица 1 – Результаты тестирования основных сценариев

Сценарий	Ожидаемый результат	Итог
Попытка входа с пустым полем адреса	Отображение всплывающего диалога с предупреждением	Пройден
Попытка входа с пустым полем логина	Отображение всплывающего диалога с предупреждением	Пройден
Попытка входа с некорректными данными	Отображение всплывающего окна с текстом ошибки	Пройден
Попытка входа на несуществующий адрес	Отображение всплывающего окна с текстом ошибки	Пройден
Вход с заранее известными данными	Успешное подключение, смена интерфейса на панель проводника	Пройден
Переход в родительскую директорию или по введенному пути	Обновление текстового поля текущего пути и загрузка нового списка файлов	Пройден
Открытие текстового файла двойным кликом	Появление модального окна текстового редактора с содержимым файла	Пройден
Сохранение файла со сменой кодировки (с UTF-8 на KOI8-R)	Успешное сохранение файла на сервере	Пройден
Открытие файла с изменённой кодировкой (KOI8-R)	Отображение файла без искажений	Пройден
Создание нового текстового файла	Появление диалога ввода имени файла и открытие окна текстового редактора	Пройден
Разрыв сетевого соединения во время операции	Появление диалогового окна с текстом ошибки	Пройден

5.2 Выявленные ограничения

В ходе проведения тестирования были выявлены следующие ограничения и направления для последующей доработки:

- Содержимое удалённых файлов целиком считывается в оперативную память, что может значительно ударить по системам с малым количеством оперативной памяти.
- Поддержка текстовых файлов исключительно с расширением «.txt», в виде жёсткого ограничения. Рекомендуется составить фильтры расширений текстовых файлов для разных целей (просмотр логов, файлов конфигураций и т.д.).
- Отсутствие поддержки шифрованной версии FTP. Рекомендуется добавить поддержку FTPS и SFTP.
- Отсутствие механизма проверки изменений файла на сервере при сохранении. В случае если файл уже был кем-то изменён то произойдёт «тихая» перезапись файла.

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы было разработано графическое приложение на языке Java, объединяющее функционал FTP-клиента и текстового редактора. Программа позволяет осуществлять навигацию по удаленной файловой системе и редактировать текстовые файлы непосредственно на сервере, исключая необходимость их ручного скачивания и обратной загрузки.

Для достижения поставленной цели был выбран технологический стек, включающий стандартные средства Java, библиотеку Java Swing для создания графического интерфейса и Apache Commons Net для сетевого взаимодействия. В результате практической реализации достигнуты следующие результаты:

- Реализована многопоточная архитектура с использованием `SwingWorker` и `ExecutorService`, что гарантирует высокую отзывчивость графического интерфейса (GUI) при выполнении длительных сетевых операций.

- Разработан гибкий механизм работы с текстовыми кодировками (UTF-8, UTF-16, KOI8-R, DOS-866 и др.) на базе пакетов `java.nio.charset`, обеспечивающий корректное декодирование и сохранение файлов, созданных в различных операционных средах.

- Внедрена система ранней валидации пользовательского ввода с использованием кастомных исключений (`HostAddressException`, `LoginException`) и реализовано удобное табличное представление данных (`FTPTableModel`) с автоматическим форматированием размеров файлов и дат их модификации.

Разработанное приложение было успешно протестировано. Проверка основных сценариев использования (авторизация, навигация по директориям, открытие, редактирование, смена кодировки и сохранение файлов) подтвердила корректность работы алгоритмов и устойчивость системы к сетевым ошибкам.

В качестве перспектив дальнейшего развития проекта можно выделить: добавление поддержки защищенных протоколов передачи данных (FTPS или SFTP), реализацию потоковой обработки файлов большого объема для предотвращения ошибок нехватки памяти, а также расширение функционала текстового редактора (например, добавление подсветки синтаксиса).

Подводя итоги, можно констатировать, что все задачи, поставленные во введении, выполнены в полном объеме, а цель курсовой работы достигнута.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Oracle Corporation. Java SE 21 Documentation [Электронный ресурс]. — URL: <https://docs.oracle.com/en/java/javase/21/docs/api/> (дата обращения: 03.06.2026).
2. Horstmann C.S. Core Java: Fundamentals. Vol. 1. 12th ed. — Pearson, 2022. — 936 с.
3. Horstmann C.S. Core Java: Advanced Features. Vol. 2. 12th ed. — Pearson, 2022. — 1056 с.
4. Goetz B., Peierls T., Bloch J. и др. Java Concurrency in Practice. — Addison-Wesley Professional, 2006. — 432 с.
5. Oracle Corporation. Java Swing Tutorial [Электронный ресурс]. — URL: <https://docs.oracle.com/javase/tutorial/uiswing/> (дата обращения: 03.06.2026).
6. Apache Commons Net Project. Commons Net User Guide [Электронный ресурс]. — URL: <https://commons.apache.org/proper/commons-net/> (дата обращения: 03.06.2026).
7. Postel J. File Transfer Protocol (FTP). RFC 959 [Электронный ресурс]. — IETF, 1985. — URL: <https://www.rfc-editor.org/rfc/rfc959> (дата обращения: 03.06.2026).
8. Oracle Corporation. Java Platform, Standard Edition Tools Reference [Электронный ресурс]. — URL: <https://docs.oracle.com/en/java/javase/21/tools/> (дата обращения: 03.06.2026).
9. Блох Дж. Java: эффективное программирование. 3-е изд. — М.: Диалектика, 2019. — 464 с.
10. Oracle Corporation. Java SE Specification: JSR 393 [Электронный ресурс]. — URL: <https://jcp.org/en/jsr/detail?id=393> (дата обращения: 03.06.2026).
11. Nielsen M., Alpert S., Brown N. Refactoring UI. — Refactoring UI, 2018. — 288 с.
12. Oracle Corporation. Trail: Creating a GUI With JFC/Swing [Электронный ресурс]. — URL: <https://docs.oracle.com/javase/tutorial/uiswing/> (дата обращения: 03.06.2026).
13. Apache Software Foundation. Apache Commons Net API Documentation [Электронный ресурс]. — URL: <https://commons.apache.org/proper/commons-net/apidocs/> (дата обращения: 03.06.2026).

14. Stevens W.R. TCP/IP Illustrated. Vol. 1: The Protocols. 2nd ed. — Addison-Wesley Professional, 2011. — 1040 с.
15. Oracle Corporation. Java Platform, Standard Edition Security Developer's Guide [Электронный ресурс]. — URL: <https://docs.oracle.com/en/java/javase/21/security/> (дата обращения: 03.06.2026).

ПРИЛОЖЕНИЕ А

(обязательное)

Исходный код программы

A1. Файл FTPInteractions.java

```
package com.mycompany.java_kursach;

import java.io.IOException;
import java.io.InputStream;
import java.util.Arrays;
import java.util.List;
import org.apache.commons.net.ftp.FTP;
import org.apache.commons.net.ftp.FTPClient;
import org.apache.commons.net.ftp.FTPFile;

/**
 *
 * @author Buster Daemon
 */
public class FTPInteractions {

    public List<FTPFile> receiveFileList(FTPClient connection)
throws IOException {
        return Arrays.asList(connection.listFiles());
    }

    public String receiveCurrentDirectory(FTPClient connection)
throws IOException {
        return connection.printWorkingDirectory();
    }

    public String changeRelativeCurrentDirectory(FTPClient
connection, String dir) throws IOException {
        String curPath = connection.printWorkingDirectory();
        if (dir.isBlank() || dir.isEmpty()) {
            return curPath;
        }

        curPath += dir + "/";
        connection.changeWorkingDirectory(curPath);

        return curPath;
    }

    public String changeCurrentDirectory(FTPClient connection,
String dir) throws IOException {
        if (dir.isBlank() || dir.isEmpty()) {
            dir = "/";
        }
    }
}
```

```

        connection.changeWorkingDirectory(dir);

        return connection.printWorkingDirectory();
    }

    public String changeToParentDirectory(FTPClient connection)
throws IOException {
        String path = "";

        connection.changeToParentDirectory();
        path = connection.printWorkingDirectory();

        return path;
    }

    public InputStream downloadFile(FTPClient connection, String
path) throws IOException {
        InputStream fileData = null;

        if (!(path.isBlank() || path.isEmpty())) {
            fileData = connection.retrieveFileStream(path);
            connection.completePendingCommand();
        }

        return fileData;
    }

    public boolean uploadFile(FTPClient connection, String path,
InputStream fileData) throws IOException {
        connection.setFileType(FTP.BINARY_FILE_TYPE);
        boolean result = connection.storeFile(path, fileData);

        return result;
    }
}

```

A2. Файл FTPTableModel.java

```
package com.mycompany.java_kursach;

import java.text.SimpleDateFormat;
import java.util.List;
import java.util.Locale;
import javax.swing.table.AbstractTableModel;
import org.apache.commons.net.ftp.FTPFile;

/**
 *
 * @author Buster Daemon
 */
public class FTPTableModel extends AbstractTableModel {

    private final String[] columns = {"Имя", "Размер", "Дата",
"Тип"};
    private List<FTPFile> files = List.of();
    private final SimpleDateFormat sdf = new
SimpleDateFormat("dd.MM.yyyy HH:mm", Locale.getDefault());

    @Override
    public int getRowCount() {
        return files.size();
    }

    @Override
    public int getColumnCount() {
        return columns.length;
    }

    @Override
    public String getColumnName(int column) {
        return columns[column];
    }

    @Override
    public Object getValueAt(int row, int column) {
        FTPFile f = files.get(row);
        return switch (column) {
            case 0 ->
                f.getName();
            case 1 ->
                f.isDirectory() ? "" : formatSize(f.getSize());
            case 2 ->
```

```

        f.getTimestamp() != null ?
sdf.format(f.getTimestamp().getTime()) : "-";
        case 3 ->
            f.isDirectory() ? "Каталог" : "Файл";
        default ->
            null;
    };
}

public FTPFile getFTPFileAt(int rowIndex) {
    if (rowIndex >= 0 && rowIndex < files.size()) {
        return files.get(rowIndex);
    }
    return null;
}

public void updateData(List<FTPFile> newFiles) {
    this.files = newFiles;
    fireTableDataChanged();
}

@Override
public Class<?> getColumnClass(int column) {
    return String.class;
}

private String formatSize(long size) {
    if (size < 0) {
        return "?";
    }
    if (size < 1024) {
        return size + " Б";
    }
    if (size < 1024 * 1024) {
        return String.format("%.1f KB", size / 1024.0);
    }
    return String.format("%.2f МБ", size / (1024.0 * 1024.0));
}
}

```

A3. Файл MainApp.java

```
package com.mycompany.java_kursach;

import com.mycompany.java_kursach.Exceptions.HostAddressException;
import com.mycompany.java_kursach.Exceptions.LoginException;
import java.awt.CardLayout;
import java.awt.event.KeyEvent;
import java.io.InputStream;
import java.nio.charset.Charset;
import java.nio.charset.StandardCharsets;
import java.util.List;
import java.util.concurrent.ExecutionException;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.Future;
import javax.swing.JOptionPane;
import javax.swing.SwingWorker;
import org.apache.commons.net.ftp.FTPClient;
import org.apache.commons.net.ftp.FTPFile;

/**
 *
 * @author Buster Daemon
 */
public class MainApp extends javax.swing.JFrame {

    private static final java.util.logging.Logger logger =
java.util.logging.Logger.getLogger(MainApp.class.getName());
    private final FTPClient connection = new FTPClient();
    private final FTPTableModel tm = new FTPTableModel();
    private final FTPInteractions fi = new FTPInteractions();
    private Charset srcCharset = StandardCharsets.UTF_8;
    private final ExecutorService executor =
Executors.newSingleThreadExecutor();

    /**
     * Creates new form LoginForm
     */
    public MainApp() {
        initComponents();
    }

    /**
     * This method is called from within the constructor to
    initialize the form.
```

```

    * WARNING: Do NOT modify this code. The content of this method
is always
    * regenerated by the Form Editor.
    */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jFrame1 = new javax.swing.JFrame();
        SourceEncodingGroup = new javax.swing.ButtonGroup();
        mainPanel = new javax.swing.JPanel();
        loginPanel = new javax.swing.JPanel();
        jPanel3 = new javax.swing.JPanel();
        jLabel3 = new javax.swing.JLabel();
        hostField = new javax.swing.JTextField();
        jLabel1 = new javax.swing.JLabel();
        loginField = new javax.swing.JTextField();
        jLabel2 = new javax.swing.JLabel();
        passwordField = new javax.swing.JPasswordField();
        jPanel2 = new javax.swing.JPanel();
        loginBtn = new javax.swing.JButton();
        explorerPanel = new javax.swing.JPanel();
        jPanel4 = new javax.swing.JPanel();
        upServerBtn = new javax.swing.JButton();
        currentPathText = new javax.swing.JTextField();
        jPanel1 = new javax.swing.JPanel();
        jScrollPane2 = new javax.swing.JScrollPane();
        fileTable = new javax.swing.JTable();
        jPanel5 = new javax.swing.JPanel();
        createFileBtn = new javax.swing.JButton();
        disconnectBtn = new javax.swing.JButton();
        jMenuBar1 = new javax.swing.JMenuBar();
        jMenu1 = new javax.swing.JMenu();
        jMenu2 = new javax.swing.JMenu();
        utf8Src = new javax.swing.JRadioButtonMenuItem();
        utf16Src = new javax.swing.JRadioButtonMenuItem();
        dos437Src = new javax.swing.JRadioButtonMenuItem();
        dos866Src = new javax.swing.JRadioButtonMenuItem();
        koi8RSrc = new javax.swing.JRadioButtonMenuItem();
        koi8USrc = new javax.swing.JRadioButtonMenuItem();

        javax.swing.GroupLayout jFrame1Layout = new
javax.swing.GroupLayout(jFrame1.getContentPane());
        jFrame1.getContentPane().setLayout(jFrame1Layout);
        jFrame1Layout.setHorizontalGroup(

```

```

jFrame1Layout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING)
    .addGap(0, 400, Short.MAX_VALUE)
);
jFrame1Layout.setVerticalGroup(

jFrame1Layout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING)
    .addGap(0, 300, Short.MAX_VALUE)
);

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE
);

mainPanel.setLayout(new java.awt.CardLayout());

loginPanel.setLayout(new javax.swing.BoxLayout(loginPanel,
javax.swing.BoxLayout.PAGE_AXIS));

jPanel3.setLayout(new java.awt.GridLayout(3, 2, 0, 45));

jLabel3.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
jLabel3.setText("Адрес хоста");
jPanel3.add(jLabel3);

hostField.setText("0.0.0.0");
hostField.setPreferredSize(new java.awt.Dimension(64, 10));
jPanel3.add(hostField);

jLabel1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
jLabel1.setText("ЛОГИН");
jPanel3.add(jLabel1);
jPanel3.add(loginField);

jLabel2.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
jLabel2.setText("Пароль");
jPanel3.add(jLabel2);
jPanel3.add(passwordField);

loginPanel.add(jPanel3);

```

```

jPanel2.setMinimumSize(new java.awt.Dimension(72, 16));
jPanel2.setLayout(new java.awt.GridLayout(1, 0));

loginBtn.setText("Войти");
loginBtn.setAlignmentX(0.5F);
loginBtn.addActionListener(this::loginBtnActionPerformed);
jPanel2.add(loginBtn);

loginPanel.add(jPanel2);

mainPanel.add(loginPanel, "card2");

explorerPanel.setLayout(new java.awt.BorderLayout());

jPanel4.setLayout(new
org.netbeans.lib.awtextra.AbsoluteLayout());

upServerBtn.setText("Вверх");

upServerBtn.addActionListener(this::upServerBtnActionPerformed);
jPanel4.add(upServerBtn, new
org.netbeans.lib.awtextra.AbsoluteConstraints(0, 0, -1, -1));

currentPathText.setText("/");
currentPathText.addKeyListener(new
java.awt.event.KeyAdapter() {
    public void keyPressed(java.awt.event.KeyEvent evt) {
        currentPathTextKeyPressed(evt);
    }
});
jPanel4.add(currentPathText, new
org.netbeans.lib.awtextra.AbsoluteConstraints(75, 0, 480, 23));

explorerPanel.add(jPanel4,
java.awt.BorderLayout.LINE_START);

jPanel1.setLayout(new javax.swing.BoxLayout(jPanel1,
javax.swing.BoxLayout.PAGE_AXIS));

fileTable.setModel(this.tm);
fileTable.addMouseListener(new
java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt)
{
        fileTableMouseClicked(evt);
    }
}

```

```

});
jScrollPane2.setViewportViewView(fileTable);

jPanel1.add(jScrollPane2);

jPanel5.setLayout(new java.awt.GridBagLayout());

createFileBtn.setText("Новый файл");

createFileBtn.addActionListener(this::createFileBtnActionPerformed);

jPanel5.add(createFileBtn, new
java.awt.GridBagConstraints());

disconnectBtn.setText("Отключиться");

disconnectBtn.addActionListener(this::disconnectBtnActionPerformed);

jPanel5.add(disconnectBtn, new
java.awt.GridBagConstraints());

jPanel1.add(jPanel5);

explorerPanel.add(jPanel1, java.awt.BorderLayout.PAGE_END);

mainPanel.add(explorerPanel, "card3");

jMenu1.setText("Клиент");

jMenu2.setText("Исходная кодировка");

SourceEncodingGroup.add(utf8Src);
utf8Src.setSelected(true);
utf8Src.setText("UTF-8");
utf8Src.addActionListener(this::utf8SrcActionPerformed);
jMenu2.add(utf8Src);

SourceEncodingGroup.add(utf16Src);
utf16Src.setText("UTF-16");
utf16Src.addActionListener(this::utf16SrcActionPerformed);
jMenu2.add(utf16Src);

SourceEncodingGroup.add(dos437Src);
dos437Src.setText("DOS-437");

dos437Src.addActionListener(this::dos437SrcActionPerformed);

```

```

jMenu2.add(dos437Src);

SourceEncodingGroup.add(dos866Src);
dos866Src.setText("DOS-866");

dos866Src.addActionListener(this::dos866SrcActionPerformed);
jMenu2.add(dos866Src);

SourceEncodingGroup.add(koi8RSrc);
koi8RSrc.setText("KOI8-R");
koi8RSrc.addActionListener(this::koi8RSrcActionPerformed);
jMenu2.add(koi8RSrc);

SourceEncodingGroup.add(koi8USrc);
koi8USrc.setText("KOI8-U");
koi8USrc.addActionListener(this::koi8USrcActionPerformed);
jMenu2.add(koi8USrc);

jMenu1.add(jMenu2);

jMenuBar1.add(jMenu1);

setJMenuBar(jMenuBar1);

javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()
        .addGap(0, 6, Short.MAX_VALUE)
        .addComponent(mainPanel,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(0, 6, Short.MAX_VALUE))
    );
layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()
        .addGap(0, 0, Short.MAX_VALUE)

```

```

        .addComponent(mainPanel,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(0, 0, Short.MAX_VALUE))
    );

    pack();
} // </editor-fold>

private void updateFileList() {
    try {
        Future<List<FTPFile>> future = executor.submit(() -> {
            List<FTPFile> lf =
fi.receiveFileList(this.connection);
            return lf;
        });

        List<FTPFile> fileList = future.get();
        tm.updateData(fileList);
    } catch (InterruptedException | ExecutionException e) {
        JOptionPane.showMessageDialog(rootPane,
e.getMessage());
    }
}

private void loginBtnActionPerformed(java.awt.event.ActionEvent
evt) {
    String user = loginField.getText();
    String password = new String(passwordField.getPassword());
    String host = hostField.getText();

    if (user.isBlank() || user.isEmpty()) {
        try {
            throw new LoginException("Логин не может быть
пустым!");
        } catch (LoginException ex) {
            JOptionPane.showMessageDialog(mainPanel,
ex.getMessage());
            return;
        }
    }

    if (host.isBlank() || host.isEmpty()) {
        try {

```

```

        throw new HostAddressException("Адрес хоста не может
быть пустым!");
    } catch (HostAddressException ex) {
        JOptionPane.showMessageDialog(mainPanel,
ex.getMessage());
        return;
    }
}

SwingWorker<Void, Void> worker = new SwingWorker<Void,
Void>() {
    @Override
    protected Void doInBackground() throws Exception {
        connection.connect(host);
        connection.user(user);
        if (connection.pass(password) == 530) {
            throw new LoginException("Неверный логин и/или
пароль");
        }
        return null;
    }

    @Override
    protected void done() {
        try {
            get();
            updateFileList();
            CardLayout cr = (CardLayout)
mainPanel.getLayout();
            cr.show(mainPanel, "card3");
        } catch (ExecutionException e) {
            Throwable cause = e.getCause();
            String errorMsg = (cause != null) ?
cause.getMessage() : e.getMessage();
            JOptionPane.showMessageDialog(mainPanel,
errorMsg, "Ошибка подключения", JOptionPane.ERROR_MESSAGE);
        } catch (InterruptedException e) {
            JOptionPane.showMessageDialog(mainPanel,
"Операция прервана", "Ошибка", JOptionPane.ERROR_MESSAGE);
        }
    }
};
worker.execute();
}

```

```

private void
disconnectBtnActionPerformed(java.awt.event.ActionEvent evt) {
    SwingWorker<Void, Void> dsWorker = new SwingWorker<Void,
Void>() {
        @Override
        protected Void doInBackground() throws Exception {
            connection.disconnect();
            return null;
        }

        @Override
        protected void done() {
            try {
                get();

                CardLayout cr = (CardLayout)
mainPanel.getLayout();
                cr.show(mainPanel, "card2");
            } catch (ExecutionException e) {
                Throwable t = e.getCause();
                String errMsg = (t != null) ? t.getMessage() :
e.getMessage();
                JOptionPane.showMessageDialog(mainPanel,
errMsg, "Ошибка отключения", JOptionPane.ERROR_MESSAGE);
            } catch (InterruptedException e) {
                JOptionPane.showMessageDialog(mainPanel,
"Операция прервана", "Ошибка", JOptionPane.ERROR_MESSAGE);
            }
        }
    };

    dsWorker.execute();
}

private void
upServerBtnActionPerformed(java.awt.event.ActionEvent evt) {
    SwingWorker<Void, Void> upWorker = new SwingWorker<Void,
Void>() {
        @Override
        protected Void doInBackground() throws Exception {
            currentPathText.setText(fi.changeToParentDirectory(connection));
            return null;
        }
    };
}

```

```

@Override
protected void done() {
    try {
        get();
        updateFileList();
    } catch (ExecutionException e) {
        Throwable t = e.getCause();
        String errMsg = (t != null) ? t.getMessage() :
e.getMessage();
        JOptionPane.showMessageDialog(mainPanel,
errMsg, "Ошибка подключения", JOptionPane.ERROR_MESSAGE);
    } catch (InterruptedException e) {
        JOptionPane.showMessageDialog(mainPanel,
"Операция прервана", "Ошибка", JOptionPane.ERROR_MESSAGE);
    }
}

};

upWorker.execute();
}

private void currentPathTextKeyPressed(java.awt.event.KeyEvent
evt) {
    if (evt.getKeyCode() == KeyEvent.VK_ENTER) {
        SwingWorker<Void, Void> enterWorker = new
SwingWorker<Void, Void>() {
            @Override
            protected Void doInBackground() throws Exception {
currentPathText.setText(fi.changeCurrentDirectory(connection,
currentPathText.getText()));
                return null;
            }

            @Override
            protected void done() {
                try {
                    get();
                    updateFileList();
                } catch (ExecutionException e) {
                    Throwable t = e.getCause();
                    String errMsg = (t != null) ? t.getMessage()
: e.getMessage();
                    JOptionPane.showMessageDialog(mainPanel,
errMsg, "Ошибка подключения", JOptionPane.ERROR_MESSAGE);
                } catch (InterruptedException e) {

```

```

        JOptionPane.showMessageDialog(mainPanel,
"Операция прервана", "Ошибка", JOptionPane.ERROR_MESSAGE);
    }
}

};

enterWorker.execute();
}

}

private void fileTableMouseClicked(java.awt.event.MouseEvent
evt) {
    if (evt.getClickCount() == 2) {
        int row = this.fileTable.rowAtPoint(evt.getPoint());
        if (row >= 0) {
            FTPFile ff = tm.getFTPFileAt(row);
            final String currentPath =
currentPathText.getText();

            if (ff.isDirectory()) {
                SwingWorker<String, Void> dirWorker = new
SwingWorker<>() {
                    @Override
                    protected String doInBackground() throws
Exception {
                        return
fi.changeRelativeCurrentDirectory(connection, ff.getName());
                    }

                    @Override
                    protected void done() {
                        try {
                            String newPath = get();
                            currentPathText.setText(newPath);
                            updateFileList();
                        } catch (ExecutionException e) {
                            String msg = (e.getCause() != null)
? e.getCause().getMessage() : e.getMessage();

JOptionPane.showMessageDialog(rootPane, msg, "Ошибка смены
директории", JOptionPane.ERROR_MESSAGE);
                        } catch (InterruptedException e) {

JOptionPane.showMessageDialog(rootPane, "Операция прервана",
"Ошибка", JOptionPane.ERROR_MESSAGE);
                        }
                    }
                }
            }
        }
    }
}

```

```

        }
    };
    dirWorker.execute();

    } else if (ff.isFile() &&
ff.getName().endsWith(".txt")) {
        final String filePath = currentPath +
ff.getName();

        SwingWorker<byte[], Void> downloadWorker = new
SwingWorker<>() {

            @Override
            protected byte[] doInBackground() throws
Exception {

                try (InputStream f =
fi.downloadFile(connection, filePath)) {
                    return f.readAllBytes();
                }
            }

            @Override
            protected void done() {
                try {
                    byte[] fileContent = get();
                    MainEditor me = new
MainEditor(jFrame1, true, filePath, connection, fileContent, fi,
srcCharset);

                    me.setVisible(true);
                    updateFileList();

                } catch (ExecutionException e) {
                    String msg = (e.getCause() != null)
? e.getCause().getMessage() : e.getMessage();

                    JOptionPane.showMessageDialog(rootPane, msg, "Ошибка скачивания",
JOptionPane.ERROR_MESSAGE);

                } catch (InterruptedException e) {

                    JOptionPane.showMessageDialog(rootPane, "Операция прервана",
"Ошибка", JOptionPane.ERROR_MESSAGE);
                }
            }
        };
        downloadWorker.execute();
    }
}

```

```

    }

    private void
    createFileBtnActionPerformed(java.awt.event.ActionEvent evt) {
        String fileName = JOptionPane.showInputDialog("Введите имя
        файла");
        if (!(fileName.isEmpty() || fileName.isBlank())) {
            final String filePath = this.currentPathText.getText()
+ (fileName.endsWith(".txt") ? fileName : fileName + ".txt");
            SwingWorker<Void, Void> creator = new SwingWorker<Void,
Void>() {
                @Override
                protected Void doInBackground() throws Exception {
                    MainEditor ed = new MainEditor(jFrame1, true,
filePath, connection, new byte[0], fi, srcCharset);
                    ed.setVisible(true);
                    updateFileList();
                    return null;
                }

                @Override
                protected void done() {
                    try {
                        get();
                    } catch (ExecutionException e) {
                        String msg = (e.getCause() != null) ?
e.getCause().getMessage() : e.getMessage();
                        JOptionPane.showMessageDialog(rootPane,
msg, "Ошибка сохранения", JOptionPane.ERROR_MESSAGE);
                    } catch (InterruptedException e) {
                        JOptionPane.showMessageDialog(rootPane,
"Операция прервана", "Ошибка", JOptionPane.ERROR_MESSAGE);
                    }
                }
            };

            creator.execute();
        }

        private void utf8SrcActionPerformed(java.awt.event.ActionEvent
evt) {
            this.srcCharset = StandardCharsets.UTF_8;
        }
    }

```

```

        private void utf16SrcActionPerformed(java.awt.event.ActionEvent
evt) {
            this.srcCharset = StandardCharsets.UTF_16;
        }

        private void
dos437SrcActionPerformed(java.awt.event.ActionEvent evt) {
            this.srcCharset = Charset.forName("CP437");
        }

        private void
dos866SrcActionPerformed(java.awt.event.ActionEvent evt) {
            this.srcCharset = Charset.forName("CP866");
        }

        private void koi8RSrcActionPerformed(java.awt.event.ActionEvent
evt) {
            this.srcCharset = Charset.forName("KOI8-R");
        }

        private void koi8USrcActionPerformed(java.awt.event.ActionEvent
evt) {
            this.srcCharset = Charset.forName("KOI8-U");
        }

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    //<editor-fold defaultstate="collapsed" desc=" Look and feel
setting code (optional) ">
    /* If Nimbus (introduced in Java SE 6) is not available,
stay with the default look and feel.
        *
        * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/pla
f.html
        */
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    }
}

```

```

        }
    } catch (ReflectiveOperationException |
javax.swing.UnsupportedLookAndFeelException ex) {
        logger.log(java.util.logging.Level.SEVERE, null, ex);
    }
}
//</editor-fold>

/* Create and display the form */
java.awt.EventQueue.invokeLater(() -> new
MainApp().setVisible(true));
}

// Variables declaration - do not modify
private javax.swing.ButtonGroup SourceEncodingGroup;
private javax.swing.JButton createFileBtn;
private javax.swing.JTextField currentPathText;
private javax.swing.JButton disconnectBtn;
private javax.swing.JRadioButtonMenuItem dos437Src;
private javax.swing.JRadioButtonMenuItem dos866Src;
private javax.swing.JPanel explorerPanel;
private javax.swing.JTable fileTable;
private javax.swing.JTextField hostField;
private javax.swing.JFrame jFrame1;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JMenu jMenu1;
private javax.swing.JMenu jMenu2;
private javax.swing.JMenuBar jMenuBar1;
private javax.swing.JPanel jPanel1;
private javax.swing.JPanel jPanel2;
private javax.swing.JPanel jPanel3;
private javax.swing.JPanel jPanel4;
private javax.swing.JPanel jPanel5;
private javax.swing.JScrollPane jScrollPane2;
private javax.swing.JRadioButtonMenuItem koi8RSrc;
private javax.swing.JRadioButtonMenuItem koi8USrc;
private javax.swing.JButton loginBtn;
private javax.swing.JTextField loginField;
private javax.swing.JPanel loginPanel;
private javax.swing.JPanel mainPanel;
private javax.swing.JPasswordField passwordField;
private javax.swing.JButton upServerBtn;
private javax.swing.JRadioButtonMenuItem utf16Src;
private javax.swing.JRadioButtonMenuItem utf8Src;
}

```

A4. Файл MainEditor.java

```
package com.mycompany.java_kursach;

import java.io.ByteArrayInputStream;
import java.io.InputStream;
import java.nio.ByteBuffer;
import java.nio.CharBuffer;
import java.nio.charset.Charset;
import java.nio.charset.StandardCharsets;
import java.util.concurrent.ExecutionException;
import javax.swing.JOptionPane;
import javax.swing.SwingWorker;
import org.apache.commons.net.ftp.FTPClient;

/**
 *
 * @author Buster Daemon
 */
public class MainEditor extends javax.swing.JDialog {

    private static final java.util.logging.Logger logger =
java.util.logging.Logger.getLogger(MainEditor.class.getName());
    private final String ftpFilePath;
    private final FTPClient connection;
    private final FTPInteractions fis;
    private final byte initString[];
    private Charset destEncoding = StandardCharsets.UTF_8;
    private Charset srcEncoding = StandardCharsets.UTF_8;

    /**
     * Creates new form MainEditor
     *
     * @param parent
     * @param modal
     * @param ftpFilePath
     * @param con
     * @param srcEncoding
     * @param initString
     * @param fi
     */
    public MainEditor(java.awt.Frame parent, boolean modal, String
ftpFilePath, FTPClient con, byte initString[], FTPInteractions fi,
Charset srcEncoding) {
        super(parent, modal);
        this.ftpFilePath = ftpFilePath;
    }
}
```

```

        this.connection = con;
        this.initialString = initString;
        this.fis = fi;
        this.srcEncoding = srcEncoding;
        initComponents();
    }

    /**
     * This method is called from within the constructor to
    initialize the form.
     * WARNING: Do NOT modify this code. The content of this method
    is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jScrollPane1 = new javax.swing.JScrollPane();
        textEditor = new javax.swing.JEditorPane();
        jMenuItem1 = new javax.swing.JMenuItem();
        saveMenu = new javax.swing.JMenuItem();
        jMenuItem2 = new javax.swing.JMenuItem();
        exitMenu = new javax.swing.JMenuItem();
        jMenuItem3 = new javax.swing.JMenuItem();
        toUTF8 = new javax.swing.JMenuItem();
        toUTF16 = new javax.swing.JMenuItem();
        jMenuItem4 = new javax.swing.JMenuItem();
        toDOS437 = new javax.swing.JMenuItem();
        toDOS866 = new javax.swing.JMenuItem();
        jMenuItem5 = new javax.swing.JMenuItem();
        toKOI8RU = new javax.swing.JMenuItem();
        toKOI8U = new javax.swing.JMenuItem();

        setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);

        setBounds(new java.awt.Rectangle(0, 0, 640, 480));
        setMinimumSize(new java.awt.Dimension(640, 480));
        getContentPane().setLayout(new
            javax.swing.BoxLayout(getContentPane(),
                javax.swing.BoxLayout.LINE_AXIS));

        textEditor.setText(prepareTextEditor());
    }

```

```
jScrollPane1.setViewportViewView(textEditor);

getContentPane().add(jScrollPane1);

jMenu1.setText("Файл");

saveMenu.setText("Сохранить");
saveMenu.addActionListener(this::saveMenuActionPerformed);
jMenu1.add(saveMenu);
jMenu1.add(jSeparator1);

exitMenu.setText("Выйти");
exitMenu.addActionListener(this::exitMenuActionPerformed);
jMenu1.add(exitMenu);

jMenuBar1.add(jMenu1);

jMenu2.setText("Изменить");

jMenu3.setText("Кодировка сохранения");

toUTF8.setText("UTF-8");
toUTF8.addActionListener(this::toUTF8ActionPerformed);
jMenu3.add(toUTF8);

toUTF16.setText("UTF-16");
toUTF16.addActionListener(this::toUTF16ActionPerformed);
jMenu3.add(toUTF16);
jMenu3.add(jSeparator2);

toDOS437.setText("DOS-437");
toDOS437.addActionListener(this::toDOS437ActionPerformed);
jMenu3.add(toDOS437);

toDOS866.setText("DOS-866");
toDOS866.addActionListener(this::toDOS866ActionPerformed);
jMenu3.add(toDOS866);
jMenu3.add(jSeparator3);

toKOI8RU.setText("KOI8-R");
toKOI8RU.addActionListener(this::toKOI8RUActionPerformed);
jMenu3.add(toKOI8RU);

toKOI8U.setText("KOI8-U");
toKOI8U.addActionListener(this::toKOI8UActionPerformed);
jMenu3.add(toKOI8U);
```

```

jMenu2.add(jMenu3);

jMenuBar1.add(jMenu2);

setJMenuBar(jMenuBar1);

pack();
} // </editor-fold>

private String prepareTextEditor() {
    CharBuffer cb =
this.srcEncoding.decode(ByteBuffer.wrap(initialString));
    ByteBuffer bb = this.destEncoding.encode(cb);
    byte[] decoded = new byte[bb.remaining()];
    bb.get(decoded);

    return new String(decoded);
}

private void saveMenuActionPerformed(java.awt.event.ActionEvent
evt) {
    final InputStream is = new
ByteArrayInputStream(this.textEditor.getText().getBytes(this.destE
ncoding));

    SwingWorker<Boolean, Void> saveWorker = new
SwingWorker<Boolean, Void>() {
        @Override
        protected Boolean doInBackground() throws Exception {
            return fis.uploadFile(connection, ftpFilePath, is);
        }

        @Override
        protected void done() {
            try {
                boolean result = get();
                if (result) {
                    JOptionPane.showMessageDialog(rootPane,
"Файл успешно сохранен");
                } else {
                    JOptionPane.showMessageDialog(rootPane, "Не
удалось сохранить файл");
                }
            } catch (ExecutionException e) {

```

```

        JOptionPane.showMessageDialog(rootPane,
e.getCause().getMessage(), "Ошибка сохранения",
JOptionPane.ERROR_MESSAGE);
    } catch (InterruptedException e) {
        JOptionPane.showMessageDialog(rootPane,
"Операция прервана", "Ошибка", JOptionPane.ERROR_MESSAGE);
    }
}

};

saveWorker.execute();
}

private void toUTF8ActionPerformed(java.awt.event.ActionEvent
evt) {
    this.destEncoding = StandardCharsets.UTF_8;
}

private void toUTF16ActionPerformed(java.awt.event.ActionEvent
evt) {
    this.destEncoding = StandardCharsets.UTF_16;
}

private void toDOS437ActionPerformed(java.awt.event.ActionEvent
evt) {
    this.destEncoding = Charset.forName("CP437");
}

private void toDOS866ActionPerformed(java.awt.event.ActionEvent
evt) {
    this.destEncoding = Charset.forName("CP866");
}

private void toKOI8RUActionPerformed(java.awt.event.ActionEvent
evt) {
    this.destEncoding = Charset.forName("KOI8-R");
}

private void toKOI8UActionPerformed(java.awt.event.ActionEvent
evt) {
    this.destEncoding = Charset.forName("KOI8-U");
}

private void exitMenuActionPerformed(java.awt.event.ActionEvent
evt) {
    setVisible(false);
}

```

```

        dispose();
    }

    // Variables declaration - do not modify
    private javax.swing.JMenuItem exitMenu;
    private javax.swing.JMenu jMenu1;
    private javax.swing.JMenu jMenu2;
    private javax.swing.JMenu jMenu3;
    private javax.swing.JMenuBar jMenuBar1;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JPopupMenu.Separator jSeparator1;
    private javax.swing.JPopupMenu.Separator jSeparator2;
    private javax.swing.JPopupMenu.Separator jSeparator3;
    private javax.swing.JMenuItem saveMenu;
    private javax.swing.JEditorPane textEditor;
    private javax.swing.JMenuItem toDOS437;
    private javax.swing.JMenuItem toDOS866;
    private javax.swing.JMenuItem toKOI8RU;
    private javax.swing.JMenuItem toKOI8U;
    private javax.swing.JMenuItem toUTF16;
    private javax.swing.JMenuItem toUTF8;
    // End of variables declaration
}

```

A5. Файл HostAddressException.java

```
package com.mycompany.java_kursach.Exceptions;

/**
 *
 * @author Buster Daemon
 */
public class HostAddressException extends Exception {
    public HostAddressException(String msg) {
        super(msg);
    }
}
```

A6. Файл InvalidLoginException.java

```
package com.mycompany.java_kursach.Exceptions;

/**
 *
 * @author Buster Daemon
 */
public class InvalidLoginException extends Exception {
    public InvalidLoginException(String msg) {
        super(msg);
    }
}
```

A7. Файл LoginException.java

```
package com.mycompany.java_kursach.Exceptions;

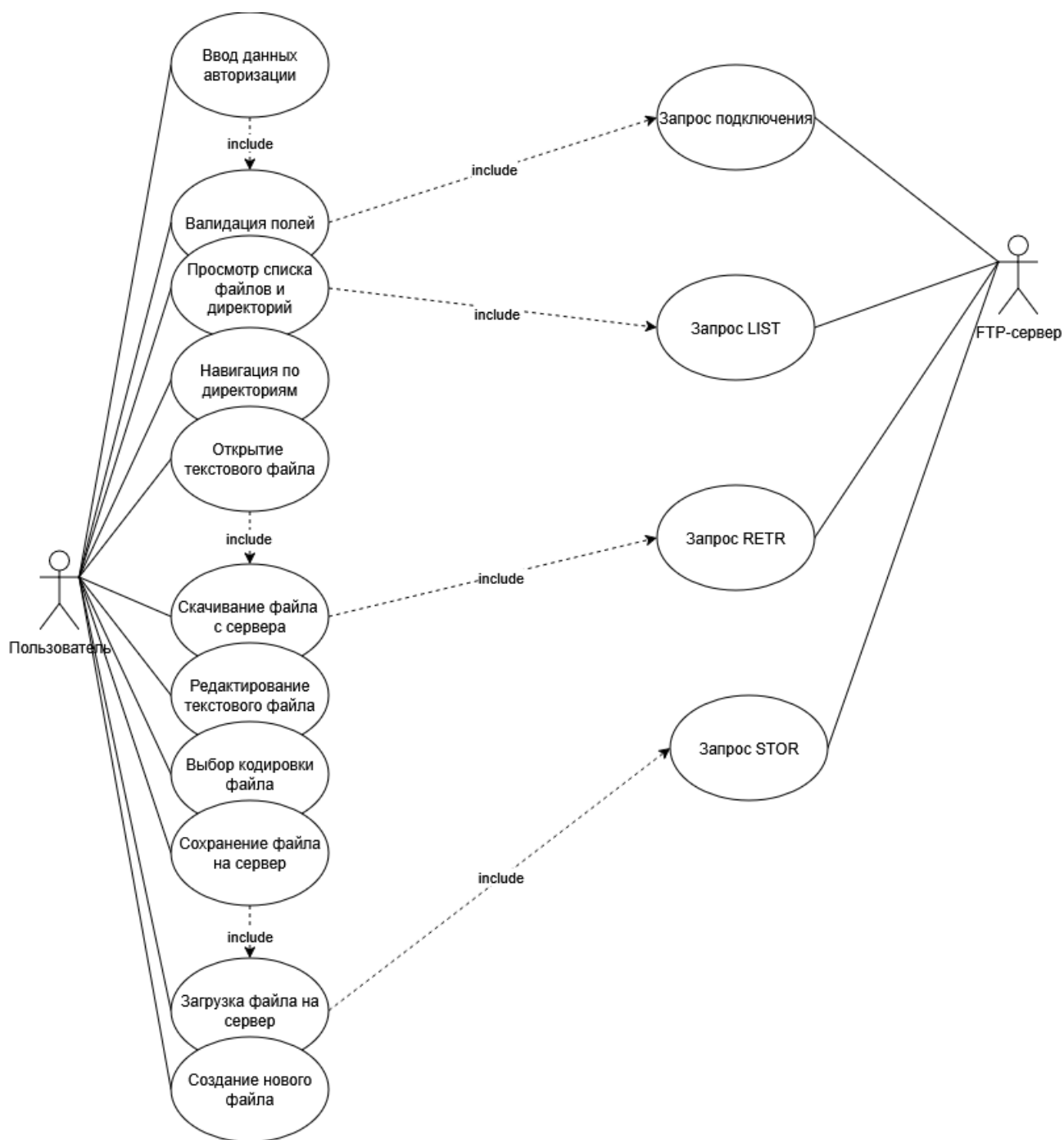
/**
 *
 * @author Buster Daemon
 */
public class LoginException extends Exception{
    public LoginException(String msg) {
        super(msg);
    }
}
```

ПРИЛОЖЕНИЕ Б

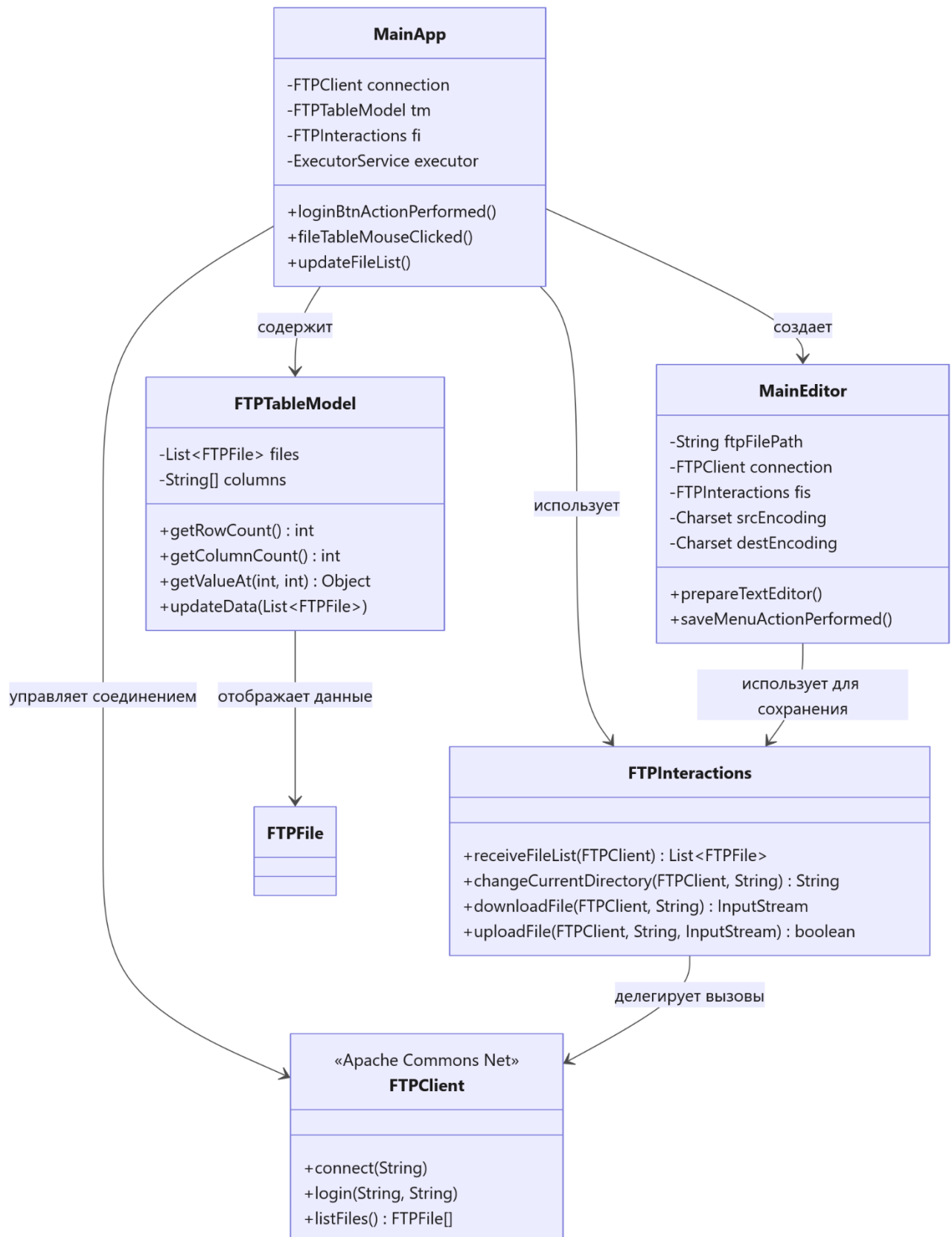
(обязательное)

UML-диаграммы проекта

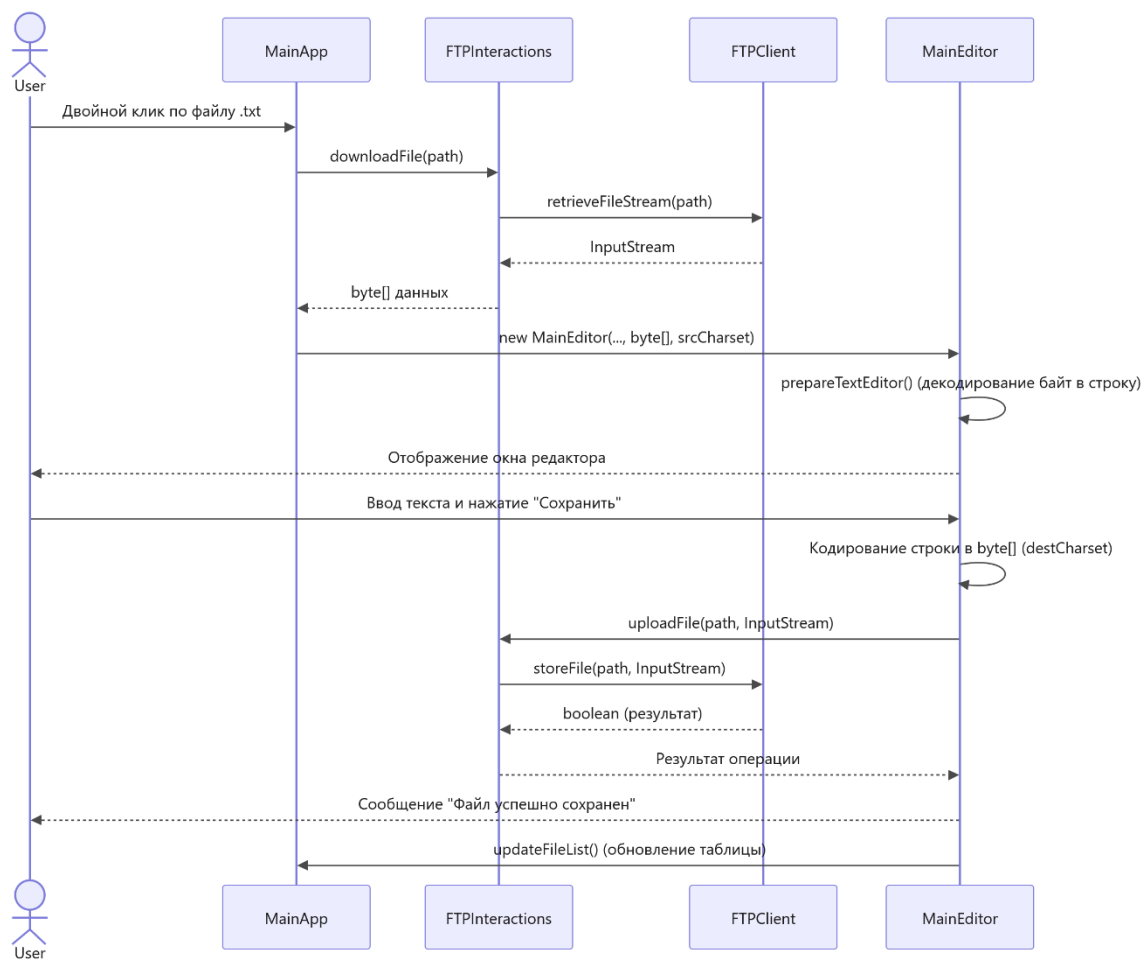
Б1. Диаграмма вариантов использования



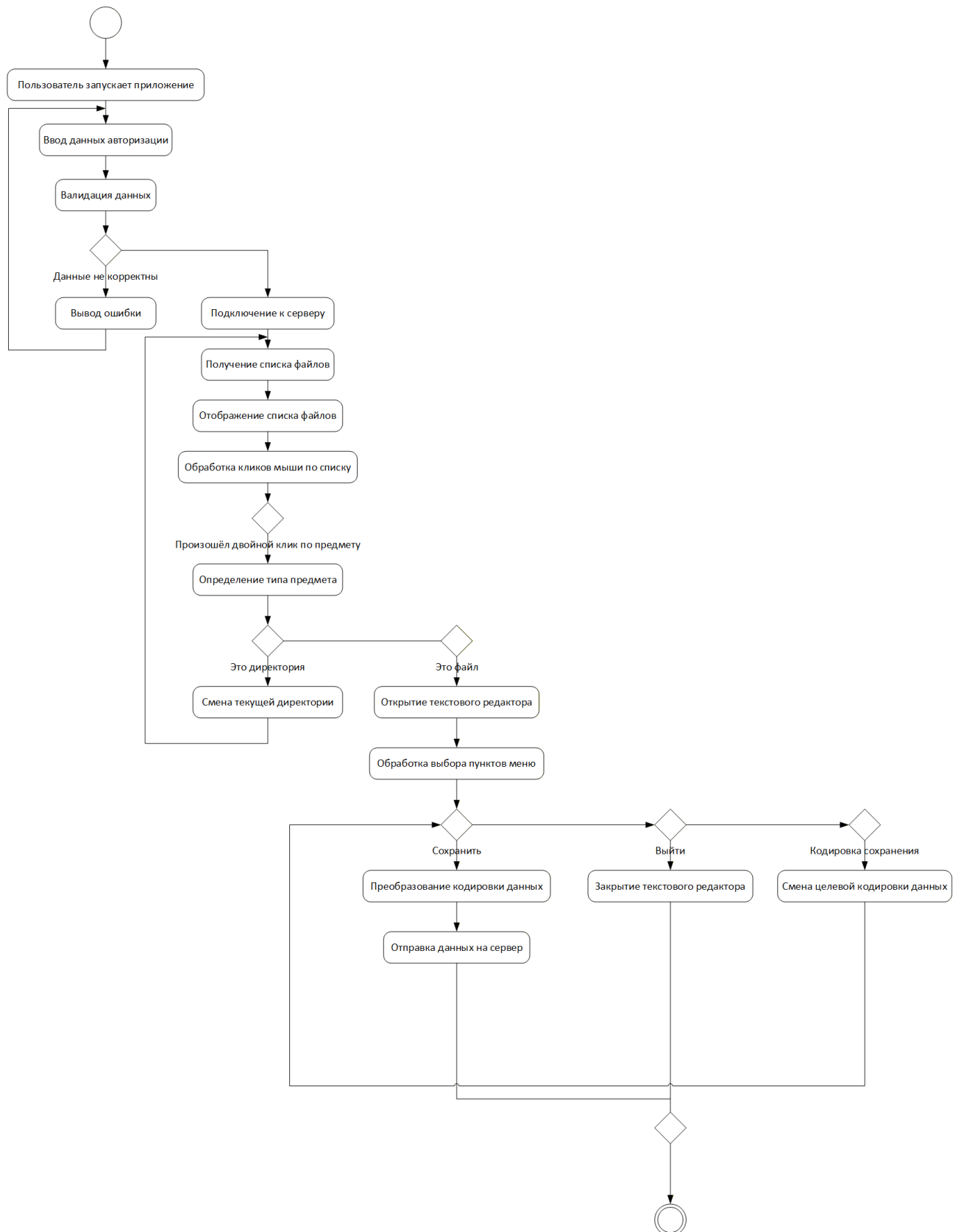
Б2. Диаграмма классов



Б3. Диаграмма последовательности



Б4. Диаграмма деятельности



Б5. Диаграмма развёртывания

