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 «Файловый менеджер»

Выполнил:

студент группы 22ВВП1

Хоссейни Нежад С.А.С.М.

Принял:

Юрова О.В.

Пенза 2025

# **Содержание**

[Содержание 3](#_Toc196565590)

[Введение 4](#_Toc196565591)

[1 Постановка задачи 6](#_Toc196565592)

[2 Теоретическая часть задания 7](#_Toc196565593)

[3 Описание алгоритма работы 10](#_Toc196565594)

[4 Описание программы 13](#_Toc196565595)

[4.1 Сервер 13](#_Toc196565596)

[4.2 Клиент 15](#_Toc196565597)

[4.3 Библиотека 17](#_Toc196565598)

[5 Описание работы программы 19](#_Toc196565599)

[Заключение 21](#_Toc196565600)

[Список используемых источников 23](#_Toc196565601)

[Приложение А Листинг программы 24](#_Toc196565602)

[Приложение А.1 Код сервера 24](#_Toc196565603)

[Приложение А.2 Код клиента 40](#_Toc196565604)

[Приложение А.3 Код библиотеки 83](#_Toc196565605)

[Приложение Б UML-диаграммы 97](#_Toc196565606)

[Приложение Б.1 UML-диаграмма вариантов использования приложения 97](#_Toc196565607)

[Приложение Б.2 UML-диаграмма последовательности протокола взаимодействия клиента и сервера 102](#_Toc196565608)

[Приложение Б.3 UML-диаграмма развертывания 103](#_Toc196565609)

[Приложение Б.4 UML-диаграмма состояний клиента 104](#_Toc196565610)

[Приложение Б.5 UML-диаграмма состояний сервера 106](#_Toc196565611)

[Приложение Б.6 UML-диаграмма классов сервера 108](#_Toc196565612)

[Приложение Б.7 UML-диаграмма классов клиента 109](#_Toc196565613)

# **Введение**

В современных условиях автоматизации бизнес-процессов и удалённого взаимодействия пользователей с файловыми хранилищами эффективность и безопасность работы с файлами приобретают ключевое значение. Клиент-серверные файловые менеджеры позволяют централизованно хранить, передавать и обрабатывать пользовательские данные, обеспечивая при этом разграничение доступа и защищённую передачу информации. Разработка подобной системы особенно актуальна в свете роста объёма распределённых хранилищ, требующих надёжной и удобной оболочки для работы конечного пользователя.

Объектом данной курсовой работы является процесс управления файловой системой в клиент-серверной архитектуре. Предметом — программная реализация клиентской и серверной частей файлового менеджера на языке Java с применением технологий сетевого взаимодействия, шифрования и графического интерфейса пользователя ​.

Цель курсовой работы — спроектировать и реализовать эффективный, безопасный и удобный в использовании файловый менеджер в клиент-серверном режиме.

В работе применяются методы объектно-ориентированного проектирования, сетевого программирования и криптографической защиты данных. В качестве основных технологий использованы язык программирования Java, библиотеки java.net для организации сокет-сообщений, Swing для построения пользовательского интерфейса, криптографические утилиты для генерации и управления ключами RSA/AES, база данных SQLite для хранения данных о пользователях.

Практическая ценность работы заключается в создании прототипа файлового менеджера, готового к расширенному использованию и доработке под конкретные корпоративные или образовательные задачи. Новизна реализованного решения состоит в объединении простого и интуитивно понятного интерфейса с надёжным уровнем защищённой передачи больших объёмов данных через сеть.

# **1 Постановка задачи**

Разработать приложение «Файловый менеджер»

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

Диаграмма вариантов использования представлена в приложении Б.1

# **2 Теоретическая часть задания**

**1. Архитектура клиент-сервер**

Клиент-серверная архитектура разделяет систему на две основные роли:

* **Клиент** — взаимодействует с пользователем, формирует запросы и отображает полученные ответы;
* **Сервер** — обрабатывает поступающие запросы, обеспечивает безопасность и консистентность информации.

**2. Протокол TCP и обмен данными**

**TCP (Transmission Control Protocol)** обеспечивает надёжную, ориентированную на соединение передачу данных:

* **Установление соединения** гарантирует синхронизацию состояния между узлами;
* **Гарантия доставки** — при потере пакета производится повторная отправка, контрольная сумма проверяет целостность;
* **Потоковая передача** — данные считываются и записываются как непрерывный поток байт, что удобно для передачи больших объёмов (например, файлов).  
  В Java TCP-сессия реализуется через Socket и ServerSocket, а для объектов применяется потоковая сериализация (ObjectOutputStream/ObjectInputStream).

**3. Гибридное шифрование: RSA + AES**

Cочетание асимметричного и симметричного шифрования для защиты канала связи:

1. **RSA** (4096-бит) служит для безопасной передачи симметричного ключа. Публичным ключом шифруется сессионный ключ, приватным — расшифровывается.
2. **AES** (128- или 256-бит) применяется для шифрования основного потока данных. Симметричное шифрование быстрее асимметричного и подходит для больших объёмов.  
   Гибридный подход сочетает надёжность RSA при обмене ключами и производительность AES при передаче контента.

**4. Паттерн DTO и сериализация объектов**

**DTO (Data Transfer Object)** — структура без бизнес-логики, предназначенная для переноса данных между компонентами или по сети.

* Объекты DTO инкапсулируют поля запроса (например, команда, параметры) и легко сериализуются в байтовый поток.
* В Java сериализация осуществляется реализацией интерфейса Serializable и передачей через ObjectOutputStream.
* Приёмо-передача DTO упрощает расширение функциональности, поскольку новые типы запросов добавляются без изменения инфраструктуры обмена.

**5. Java Swing для настольных GUI**

**Swing** — библиотека компонентов для создания графических интерфейсов в Java:

* **Лёгковесность**: компоненты полностью рисуются на стороне Java, независимы от ОС;
* **MVC** (Model-View-Controller): чёткое разделение данных (model), отображения (view) и обработки событий (controller);
* **Event Dispatch Thread (EDT)**: специальный поток для всех операций с UI, предотвращающий состояния гонки;
* **Look & Feel**: возможность смены визуального стиля компонентов.  
  Среди основных компонентов Swing — JFrame, JPanel, JButton, JTable, JFileChooser, а также возможность создавать собственные панели и диалоги.

**6. Многопоточность и конкурентный доступ**

Многопоточные модели обеспечивают параллельное обслуживание нескольких клиентов или задач:

* **Thread-per-Connection**: для каждого входящего соединения создаётся свой поток, что упрощает логику, но требует управления числом потоков;
* **Пулы потоков** (ExecutorService): ограничивают количество одновременно активных задач, повторно используют потоки и повышают масштабируемость;
* **Синхронизация**: при совместном доступе к разделяемым ресурсам (файлам, БД) используются synchronized, Lock или коллекции из пакета java.util.concurrent.

# **3 Описание алгоритма работы**

Алгоритм работы можно разбить на несколько стадий:

1. **Инициализация соединения**  
   При запуске клиент создаёт TCP-сокет и устанавливает соединение с сервером (через Socket/ServerSocket), после чего инициализирует потоки для обмена объектами (ObjectOutputStream и ObjectInputStream).
2. **Гибридный обмен ключами**
   * Сервер генерирует пару RSA-ключей (4096 бит) и отправляет клиенту публичный ключ.
   * Клиент генерирует случайный симметричный AES-ключ, шифрует его публичным RSA-ключом и отправляет обратно.
   * После этого все последующие сообщения шифруются AES, что сочетает надёжность RSA и производительность AES.
3. **Аутентификация пользователя**
   * Клиент отправляет RegisterRequest или LoginRequest с учётными данными.
   * Сервер проверяет или регистрирует пользователя с помощью UserDB, создаёт ему отдельный каталог в файловой системе и возвращает результат в ResponseDTO.
   * В случае успеха клиент переходит к основному рабочему режиму, иначе повторяет ввод.
4. **Основной цикл обработки команд**  
   После аутентификации клиент входит в цикл:
   * Пользователь выбирает операцию (просмотр папки, загрузка, скачивание и т. д.).
   * Клиент формирует соответствующий DTO-запрос (ListFilesRequest, UploadRequest, DownloadRequest и др.), сериализует его, шифрует AES и отправляет серверу.
   * Сервер дешифрует, десериализует запрос, обрабатывает команду (работа с файловой системой) и возвращает результат в виде ResponseDTO (с флагом успеха и, при необходимости, данными).
5. **Операция загрузки файла**
   * Клиент отправляет UploadRequest, указывая путь назначения, размер файла и флаг перезаписи.
   * Сервер проверяет наличие файла и при конфликте возвращает сообщение об ошибке. Клиент предлагает перезапись или переименование.
   * При готовности сервер отвечает “READY”.
   * Клиент читает файл блоками по 4096 B, шифрует каждую порцию AES и отправляет серверу.
   * Сервер расшифровывает блоки, записывает в файл и по окончании передаёт итоговый ответ об успешной загрузке.
6. **Операция скачивания файла**
   * Клиент отправляет DownloadRequest с путём к файлу.
   * Сервер проверяет существование, отправляет размер файла в ответе.
   * Затем сервер циклически читает файл, шифрует блоки AES и пересылает клиенту.
   * Клиент принимает, расшифровывает и записывает данные в локальный файл, пока не будет получен весь объём.
7. **Работа с каталогами и файлами**
   * **Просмотр содержимого**: ListFilesRequest возвращает список объектов с метаданными (имя, размер, дата, тип).
   * **Создание папки**: CreateFolderRequest создаёт директорию на сервере.
   * **Удаление**: DeleteRequest рекурсивно удаляет файлы и папки.
   * **Копирование и переименование**: CopyRequest и RenameRequest позволяют дублировать и изменять имена объектов, причём при копировании генерируются уникальные имена при конфликте.
8. **Поиск файлов**  
   Клиент формирует SearchRequest с текстом поиска. Сервер рекурсивно обходит каталог пользователя и формирует список совпадающих FileInfo, который возвращает клиенту.
9. **Обновление интерфейса**  
   После каждого обмена данных клиент обновляет графический интерфейс (таблицу, панель хлебных крошек и т. д.), позволяя пользователю интерактивно работать с актуальным состоянием удалённой файловой системы.

# **4 Описание программы**

## **4.1 Сервер**

**Server**

1. main(String[] args)  
   – настраивает вывод в UTF-8, создаёт корневую папку данных, генерирует RSA-ключи и запускает ServerSocket; для каждого подключения создаёт новый поток с ClientHandler.
2. getBaseDir()  
   – отдаёт путь к корневой директории хранения пользовательских данных.
3. getPublicKey()  
   – возвращает публичный RSA-ключ для передачи клиентам.
4. getPrivateKey()  
   – возвращает приватный RSA-ключ для расшифровки данных.
5. setUtf8Output()  
   – перенастраивает System.out и System.err на кодировку UTF-8.

**ClientHandler**

1. ClientHandler(Socket socket)  
   – конструктор: сохраняет сокет, инициализирует потоки ObjectOutputStream и ObjectInputStream.
2. run()  
   – выполняет обмен ключами (RSA → AES), затем в цикле принимает зашифрованные запросы, десериализует DTO и вызывает соответствующий метод обработки.
3. serialize(Object obj)  
   – превращает объект-DTO в последовательность байтов.
4. deserialize(byte[] data)  
   – восстанавливает DTO из байтов.
5. sendResponse(ResponseDTO resp)  
   – сериализует ответ, шифрует AES и отправляет клиенту.
6. handleRegister(RegisterRequest req)  
   – проверяет наличие пользователя, создаёт для него папку и отправляет результат регистрации.
7. handleLogin(LoginRequest req)  
   – сверяет переданный пароль с БД и отправляет результат аутентификации.
8. handleUpload(UploadRequest req)  
   – проверяет конфликт имени, при одобрении отвечает «READY», принимает AES-блоки и сохраняет файл.
9. handleDownload(DownloadRequest req)  
   – проверяет наличие файла, сообщает его размер и пересылает содержимое шифрованными блоками.
10. handleCreateFolder(CreateFolderRequest req)  
    – создаёт директорию и отправляет статус операции.
11. handleDelete(DeleteRequest req)  
    – рекурсивно удаляет файл или папку, оповещая о результате.
12. handleSearch(SearchRequest req)  
    – обходит директорию, ищет по подстроке в имени и отправляет список найденных элементов.
13. handleCopy(CopyRequest req)  
    – копирует файл или папку, при необходимости генерирует уникальное имя и сообщает результат.
14. handleRename(RenameRequest req)  
    – переименовывает файл или папку и уведомляет о результате.
15. handleListFiles(ListFilesRequest req)  
    – перечисляет содержимое указанной папки и отсылает список FileInfo.
16. deleteRecursively(File) — удаление каталога с вложениями;
17. toFileInfo(File, int) — формирование объекта FileInfo;
18. searchFiles(File, String, List, int) — рекурсивный поиск;
19. getAvailableName(File) — генерация нового имени при конфликте;
20. copyFile(...) и copyDirectory(...) — копирование файлов и директорий.

**UserDB**

1. UserDB()  
   – конструктор: создаёт папку для БД, подключается к SQLite и создаёт таблицу пользователей.
2. userExists(String username)  
   – проверяет, зарегистрирован ли пользователь.
3. insertUser(String username, String password)  
   – хэширует пароль через BCrypt и сохраняет нового пользователя.
4. checkPassword(String username, String password)  
   – извлекает хэш из БД и сверяет его с введённым паролем.

Описание взаимодействия классов представлено в приложении Б.6

## **4.2 Клиент**

**ClientApp**

1. main(String[] args)  
   – создаёт FileManagerClient, запускает Swing GUI (LoginFrame) в потоке EDT.
2. FileManagerClient
3. FileManagerClient(String ip, int port)  
   – конструктор: сохраняет адрес сервера и вызывает connect().
4. connect()  
   – устанавливает TCP-соединение, инициализирует потоки, получает публичный RSA-ключ и обменивается AES-ключом.
5. sendRequest(RequestDTO request)  
   – сериализует DTO, шифрует AES, отправляет на сервер, ждёт ответа, расшифровывает и десериализует ResponseDTO.
6. serialize(Object) и deserialize(byte[])  
   – утилиты для преобразования объектов в байты и обратно.
7. register(...) и login(...)  
   – отправляют запросы на регистрацию и вход.
8. uploadFile(...) и downloadFile(...)  
   – реализуют блочную передачу файлов с AES-шифрованием.
9. createFolder(String folderPath)  
   – формирует и отправляет CreateFolderRequest, затем при успехе обновляет список текущей директории.
10. createFolder(String folderPath)  
    – формирует и отправляет CreateFolderRequest, затем при успехе обновляет список текущей директории.
11. delete(String path)  
    – отправляет DeleteRequest для указанного пути и при успехе перезагружает содержимое папки.
12. search(String query)  
    – отправляет SearchRequest с текстом поиска и заменяет отображаемый список на результаты поиска.
13. copy(String source, String destination)  
    – отправляет CopyRequest с путями исходного и целевого объекта и затем обновляет список файлов.
14. rename(String oldPath, String newName)  
    – формирует RenameRequest с текущим путём и новым именем, после чего обновляет отображение.
15. listFolder(String relativePath)  
    – отправляет ListFilesRequest для заданной директории и выводит полученный список FileInfo.

**Frame**

1. Frame(FileManagerClient client)  
   – конструктор: сохраняет ссылку на клиент, вызывает initTable(), refreshFileList() и initBreadcrumbPanel().
2. initTable()  
   – задаёт модель JTable c колонками «Имя», «Дата», «Тип», «Размер».
3. initBreadcrumbPanel()  
   – создаёт панель навигации по пути.
4. refreshFileList()  
   – запрашивает и отображает содержимое текущей папки.
5. updateTableModel(List<FileInfo>)  
   – обновляет строки таблицы данными.
6. Методы-обработчики действий:
7. uploadFile(), downloadFile() — диалоги выбора и передача файлов;
8. createFolder(), deleteItem(), copyItem(), insertCopiedItem(), renameItem() — операции с файлами и папками;
9. sortFiles(String, boolean)  
   – сортирует отображаемый список по выбранному критерию.
10. setListView(boolean)  
    – переключает режимы отображения в виде списка или таблицы.

**LoginFrame**

1. LoginFrame(FileManagerClient client)  
   – конструктор: сохраняет клиент и инициализирует компоненты.
2. performLogin()  
   – читает логин/пароль, отправляет запрос на аутентификацию, при успехе открывает Frame.
3. performRegistration()  
   – отправляет запрос на регистрацию и показывает результат.

Описание взаимодействия классов представлено в приложении Б.7

## **4.3 Библиотека**

**CryptoUtils**

1. generateAESKey()  
   – создаёт новый AES-ключ для симметричного шифрования.
2. encryptAES(byte[] data, SecretKey key)  
   – шифрует данные в режиме AES/GCM с IV.
3. decryptAES(byte[] encryptedData, SecretKey key)  
   – расшифровывает данные AES/GCM с извлечением IV.
4. encryptRSA(byte[] data, PublicKey key)  
   – шифрует данные RSA/OAEP.
5. decryptRSA(byte[] data, PrivateKey key)  
   – расшифровывает данные RSA/OAEP.
6. keyToString(Key key)  
   – кодирует ключ в Base64-строку.
7. publicKeyFromString(String)  
   – восстанавливает публичный ключ RSA из Base64.

**BreadcrumbPanel**

1. BreadcrumbPanel(BreadcrumbListener listener)  
   – конструктор: сохраняет слушатель клика и настраивает панель.
2. updateBreadcrumb(String path)  
   – пересоздаёт навигацию, добавляя метки и стрелки.
3. createArrowLabel()  
   – создаёт разделитель «>».
4. createLinkLabel(String text, String targetPath)  
   – формирует кликабельную метку сегмента пути.

**Классы CommandType, CopyRequest, CreateFolderRequest, DeleteRequest, DownloadRequest, ListFilesRequest, LoginRequest, RegisterRequest, RenameRequest, RequestDTO, ResponseDTO, SearchRequest, UploadRequest** для передачи информации между сервером и клиентом.

# **5 Описание работы программы**

Разработанная программа представляет собой клиент-серверное приложение для управления файлами, позволяющее пользователям выполнять операции с файлами и папками на удалённом сервере через графический интерфейс. Приложение реализовано на языке Java с использованием библиотеки Swing для клиентской части и базы данных SQLite для хранения информации о пользователях.

**Серверная часть**

Сервер запускается и инициализирует необходимые компоненты, включая генерацию RSA-ключей для обеспечения безопасности передачи данных. Он прослушивает входящие соединения на определённом порту и для каждого нового клиента создаёт отдельный поток обработки. Сервер обрабатывает следующие типы запросов от клиентов:

* **Регистрация и аутентификация пользователей**: Проверка уникальности логина при регистрации и верификация пароля при входе. Данные пользователей хранятся в базе данных SQLite с использованием хэширования паролей для безопасности.
* **Операции с файлами и папками**: Создание, удаление, копирование, переименование и поиск файлов и папок в пользовательской директории на сервере. Все операции сопровождаются соответствующими ответами клиенту о результате выполнения.
* **Передача файлов**: Загрузка и скачивание файлов осуществляется с использованием шифрования AES для обеспечения конфиденциальности данных. Передача происходит поэтапно, блоками данных, что позволяет обрабатывать файлы большого размера.

**Клиентская часть**

Клиентское приложение предоставляет пользователю графический интерфейс для взаимодействия с сервером. Основные функции клиента включают:

* **Регистрация и вход в систему**: Пользователь может зарегистрироваться или войти в систему, введя свои учетные данные. После успешной аутентификации клиент получает доступ к своим файлам и папкам на сервере.
* **Навигация по файловой системе**: Интерфейс позволяет просматривать содержимое папок, переходить между директориями и выполнять операции с файлами и папками.
* **Операции с файлами**: Пользователь может создавать новые папки, удалять, копировать, переименовывать файлы и папки, сортировать по размеру, типу, дате изменения и названию, а также выполнять поиск по имени файлов.
* **Передача файлов**: Клиент может загружать файлы на сервер и скачивать их обратно. Передача данных осуществляется с использованием шифрования для обеспечения безопасности.

**Взаимодействие между клиентом и сервером**

Обмен данными между клиентом и сервером происходит по протоколу TCP/IP. Для обеспечения безопасности используется гибридная схема шифрования: при установлении соединения происходит обмен RSA-ключами, после чего для передачи данных используется симметричное шифрование AES. Все запросы и ответы сериализуются в объекты DTO (Data Transfer Object), что упрощает структуру передаваемых данных и их обработку на обеих сторонах.

**Хранение данных**

Информация о пользователях, включая логины и хэшированные пароли, хранится в базе данных SQLite на сервере. Файлы и папки каждого пользователя размещаются в отдельной директории на сервере, что обеспечивает изоляцию данных между пользователями и упрощает управление файловой системой.

# **Заключение**

В ходе выполнения данного курсового проекта была изучена архитектура клиент-серверных приложений, а также принципы сетевого взаимодействия на основе протокола TCP. Особое внимание было уделено вопросам безопасности передачи данных, где применена гибридная криптографическая схема с использованием алгоритмов RSA и AES для шифрования трафика.

Результатом работы стала программная система — файловый менеджер, реализованный по модели «клиент–сервер». Клиентская часть включает графический интерфейс на базе Java Swing, обеспечивающий интуитивно понятное взаимодействие пользователя с удалённой файловой системой. Серверная часть обрабатывает запросы клиентов, управляет файлами, а также контролирует доступ к данным.

Для хранения данных пользователей и аутентификации в проекте использовалась реляционная база данных SQLite. Пароли пользователей хранятся в зашифрованном виде с применением алгоритма BCrypt, что обеспечивает высокий уровень безопасности.

В процессе разработки были приобретены практические навыки:

* проектирования и реализации клиент-серверных приложений на языке Java;
* работы с TCP-сокетами и объектной передачей данных через сериализацию;
* применения алгоритмов шифрования RSA и AES для защиты данных;
* создания пользовательского интерфейса с помощью Java Swing;
* использования многопоточности для параллельной обработки запросов;
* взаимодействия с базой данных SQLite с помощью SQL-запросов.

Разработка программы велась в среде NetBeans на языке Java. Полученный опыт и реализованные подходы могут быть использованы при создании других защищённых клиент-серверных приложений с пользовательским интерфейсом и хранилищем данных.

# **Список используемых источников**

1. Блинов, А. В., & Соловьёв, В. В. (2019). Java. Библиотека профессионала. Санкт-Петербург: Питер. – 800 с.
2. Хорстманн, К. С. (2018). Core Java. Том I. Основы. Москва: Диалектика. – 928 с.
3. Гамма, Э., Хелм, Р., Джонсон, Р., & Влиссидес, Дж. (2007). Приёмы объектно-ориентированного проектирования. Паттерны проектирования. Санкт-Петербург: Питер. – 256 с.
4. Брус, Б. (2017). Java Swing. Руководство по созданию графических интерфейсов. Москва: Вильямс. – 720 с.
5. Бондаренко, В. А. (2016). Клиент-серверные приложения на Java: основы и практика. Москва: БХВ-Петербург. – 432 с.
6. Кнут, Д. Э. (2015). Искусство программирования. Том 1: Основные алгоритмы. Москва: Вильямс. – 784 с.
7. Шнайер, Б. (2013). Прикладная криптография: протоколы, алгоритмы и исходные тексты на языке C. Москва: Техносфера. – 784 с.
8. Кауфман, Ч., Перлман, Р., & Спек, М. (2012). Сетевая безопасность: частный доступ в публичных сетях. Санкт-Петербург: Питер. – 512 с.
9. Хан, М. (2014). SQLite. Основы работы с базой данных. Москва: Диалектика. – 320 с.
10. Саймон, Р. (2015). Безопасность Java-приложений: от разработки до внедрения. Москва: Вильямс. – 480 с.

# **Приложение А**

(Обязательное)

Листинг Программы

## **Приложение А.1**

Код сервера

**Server.java**

package Server;

import java.io.\*;

import java.net.\*;

import java.io.File;

import java.nio.charset.StandardCharsets;

import java.security.KeyPair;

import java.security.KeyPairGenerator;

import java.security.PrivateKey;

import java.security.PublicKey;

public class Server {

private static final int PORT = 12345;

private static final String BASE\_DIR = "server\_data/catalogs";

private static UserDB userDB;

private static KeyPair rsaKeyPair;

public static void main(String[] args) {

setUtf8Output();

new File(BASE\_DIR).mkdirs();

userDB = new UserDB();

try {

KeyPairGenerator keyGen = KeyPairGenerator.getInstance("RSA");

keyGen.initialize(4096);

rsaKeyPair = keyGen.generateKeyPair();

} catch (Exception e) {

throw new RuntimeException("Failed to generate RSA keys");

}

try (ServerSocket serverSocket = new ServerSocket(PORT)) {

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

while (true) {

Socket clientSocket = serverSocket.accept();

System.out.println("Подключился клиент: " + clientSocket.getInetAddress());

new Thread(new ClientHandler(clientSocket)).start();

}

} catch (Exception e) {

e.printStackTrace();

}

}

public static String getBaseDir() {

return BASE\_DIR;

}

public static PublicKey getPublicKey() {

return rsaKeyPair.getPublic();

}

public static PrivateKey getPrivateKey() {

return rsaKeyPair.getPrivate();

}

private static void setUtf8Output() {

System.setOut(new PrintStream(System.out, true, StandardCharsets.UTF\_8));

System.setErr(new PrintStream(System.err, true, StandardCharsets.UTF\_8));

}

}

**ClientHandler.java**

package Server;

import java.io.\*;

import java.net.\*;

import java.util.\*;

import DTO.\*;

import Security.\*;

import javax.crypto.SecretKey;

import javax.crypto.spec.SecretKeySpec;

public class ClientHandler implements Runnable {

private Socket socket;

private ObjectInputStream ois;

private ObjectOutputStream oos;

private String username = null;

private UserDB userDB;

private SecretKey aesKey;

public ClientHandler(Socket socket) {

this.socket = socket;

this.userDB = new UserDB();

try {

oos = new ObjectOutputStream(socket.getOutputStream());

oos.flush();

ois = new ObjectInputStream(socket.getInputStream());

} catch (IOException e) {

e.printStackTrace();

}

}

@Override

public void run() {

try {

oos.writeObject(CryptoUtils.keyToString(Server.getPublicKey()));

oos.flush();

byte[] encryptedAESKey = (byte[]) ois.readObject();

byte[] aesKeyBytes = CryptoUtils.decryptRSA(encryptedAESKey, Server.getPrivateKey());

aesKey = new SecretKeySpec(aesKeyBytes, "AES");

while (true) {

byte[] encryptedRequest = (byte[]) ois.readObject();

byte[] decrypted = CryptoUtils.decryptAES(encryptedRequest, aesKey);

Object obj = deserialize(decrypted);

if (!(obj instanceof RequestDTO)) {

sendResponse(new ResponseDTO(false, "Invalid request type"));

continue;

}

RequestDTO request = (RequestDTO)obj;

System.out.println("Получена команда: " + request.getCommand());

if (username == null &&

!(request instanceof RegisterRequest) &&

!(request instanceof LoginRequest)) {

sendResponse(new ResponseDTO(false, "Not authenticated"));

continue;

}

switch(request.getCommand()){

case REGISTER:

handleRegister((RegisterRequest) request);

break;

case LOGIN:

handleLogin((LoginRequest) request);

break;

case UPLOAD:

handleUpload((UploadRequest) request);

break;

case DOWNLOAD:

handleDownload((DownloadRequest) request);

break;

case CREATE\_FOLDER:

handleCreateFolder((CreateFolderRequest) request);

break;

case DELETE:

handleDelete((DeleteRequest) request);

break;

case SEARCH:

handleSearch((SearchRequest) request);

break;

case COPY:

handleCopy((CopyRequest) request);

break;

case RENAME:

handleRename((RenameRequest) request);

break;

case LIST\_FILES:

handleListFiles((ListFilesRequest) request);

break;

default:

sendResponse(new ResponseDTO(false, "Unknown command"));

}

}

} catch (Exception ex) {

System.out.println("Клиент отключился: " + socket.getInetAddress());

} finally {

try { socket.close(); } catch(Exception e) {}

}

}

private byte[] serialize(Object obj) throws IOException {

ByteArrayOutputStream bos = new ByteArrayOutputStream();

ObjectOutputStream oos = new ObjectOutputStream(bos);

oos.writeObject(obj);

return bos.toByteArray();

}

private Object deserialize(byte[] data) throws Exception {

ByteArrayInputStream bis = new ByteArrayInputStream(data);

ObjectInputStream ois = new ObjectInputStream(bis);

return ois.readObject();

}

private void handleRegister(RegisterRequest req) throws IOException {

String user = req.getUsername();

String pass = req.getPassword();

if (userDB.userExists(user)) {

sendResponse(new ResponseDTO(false, "User already exists"));

} else {

if (userDB.insertUser(user, pass)) {

username = user;

new File(Server.getBaseDir() + File.separator + username).mkdirs();

sendResponse(new ResponseDTO(true, "OK"));

} else {

sendResponse(new ResponseDTO(false, "Registration failed"));

}

}

}

private void handleLogin(LoginRequest req) throws IOException {

String user = req.getUsername();

String pass = req.getPassword();

if (userDB.checkPassword(user, pass)) {

username = user;

sendResponse(new ResponseDTO(true, "OK"));

} else {

sendResponse(new ResponseDTO(false, "Invalid credentials"));

}

}

private File getUserDir() {

return new File(Server.getBaseDir() + File.separator + username);

}

private void handleUpload(UploadRequest req) throws IOException, ClassNotFoundException {

String destPath = req.getDestinationPath();

long fileSize = req.getFileSize(); // Примечание: фактический размер может отличаться из-за шифрования

boolean overwrite = req.isOverwrite();

File outFile = new File(getUserDir(), destPath);

outFile.getParentFile().mkdirs();

if (outFile.exists() && !overwrite) {

sendResponse(new ResponseDTO(false, "File conflict"));

return;

}

sendResponse(new ResponseDTO(true, "READY"));

try (FileOutputStream fos = new FileOutputStream(outFile, false)) {

long remaining = fileSize;

while (true) {

byte[] encryptedBlock = (byte[]) ois.readObject();

byte[] decryptedBlock;

try {

decryptedBlock = CryptoUtils.decryptAES(encryptedBlock, aesKey);

} catch (Exception e) {

sendResponse(new ResponseDTO(false, "Decryption error"));

return;

}

fos.write(decryptedBlock);

remaining -= decryptedBlock.length;

if (remaining <= 0) break;

}

sendResponse(new ResponseDTO(true, "OK"));

} catch (Exception ex) {

sendResponse(new ResponseDTO(false, "Upload failed: " + ex.getMessage()));

throw ex;

}

}

private void handleDownload(DownloadRequest req) throws IOException {

String filePath = req.getFilePath();

File file = new File(getUserDir(), filePath);

if (!file.exists() || file.isDirectory()) {

sendResponse(new ResponseDTO(false, "File not found"));

return;

}

try (FileInputStream fis = new FileInputStream(file)) {

long fileSize = file.length();

sendResponse(new ResponseDTO(true, "OK", fileSize));

byte[] buffer = new byte[4096];

int bytesRead;

while ((bytesRead = fis.read(buffer)) != -1) {

byte[] blockToEncrypt = Arrays.copyOf(buffer, bytesRead);

byte[] encryptedBlock = CryptoUtils.encryptAES(blockToEncrypt, aesKey);

oos.writeObject(encryptedBlock);

}

oos.flush();

} catch (Exception ex) {

sendResponse(new ResponseDTO(false, "Download failed: " + ex.getMessage()));

}

}

private void handleCreateFolder(CreateFolderRequest req) throws IOException {

String folderPath = req.getFolderPath();

File folder = new File(getUserDir(), folderPath);

boolean created = folder.mkdirs();

sendResponse(new ResponseDTO(created, created ? "OK" : "Unable to create folder"));

}

private void handleDelete(DeleteRequest req) throws IOException {

String path = req.getPath();

File file = new File(getUserDir(), path);

boolean deleted = deleteRecursively(file);

sendResponse(new ResponseDTO(deleted, deleted ? "OK" : "Deletion failed"));

}

private boolean deleteRecursively(File file) {

if (file.isDirectory()) {

File[] files = file.listFiles();

if (files != null) {

for (File child : files) {

deleteRecursively(child);

}

}

}

return file.delete();

}

private FileInfo toFileInfo(File f, int baseLength) {

String fileType;

if (f.isDirectory()) {

fileType = "Папка";

} else {

int dot = f.getName().lastIndexOf(".");

if (dot != -1) {

fileType = f.getName().substring(dot + 1).toLowerCase();

} else {

fileType = "Файл";

}

}

long size = f.isDirectory() ? 0 : f.length();

String name = (baseLength >= 0) ? f.getAbsolutePath().substring(baseLength + 1) : f.getName();

return new FileInfo(name, size, f.lastModified(), fileType);

}

private void handleSearch(SearchRequest req) throws IOException {

String query = req.getQuery();

File baseDir = getUserDir();

List<FileInfo> results = new ArrayList<>();

int baseLength = baseDir.getAbsolutePath().length();

searchFiles(baseDir, query, results, baseLength);

sendResponse(new ResponseDTO(true, "OK", results));

}

private void searchFiles(File dir, String query, List<FileInfo> results, int baseLength) {

File[] files = dir.listFiles();

if (files == null) {

return;

}

for (File f : files) {

if (f.getName().contains(query)) {

results.add(toFileInfo(f, baseLength));

}

if (f.isDirectory()) {

searchFiles(f, query, results, baseLength);

}

}

}

private void handleCopy(CopyRequest req) throws IOException {

String sourcePath = req.getSourcePath();

String destPath = req.getDestinationPath();

File source = new File(getUserDir(), sourcePath);

File dest = new File(getUserDir(), destPath);

if (dest.exists()) {

dest = getAvailableName(dest);

}

boolean success = false;

if (source.isDirectory()) {

success = copyDirectory(source, dest);

} else if (source.isFile()) {

dest.getParentFile().mkdirs();

success = copyFile(source, dest);

}

sendResponse(new ResponseDTO(success, success ? "Copied as: " + dest.getName() : "Copy failed"));

}

private File getAvailableName(File original) {

String name = original.getName();

File parent = original.getParentFile();

String baseName;

String extension = "";

int dotIndex = name.lastIndexOf('.');

if (original.isFile() && dotIndex > 0) {

baseName = name.substring(0, dotIndex);

extension = name.substring(dotIndex);

} else {

baseName = name;

}

int index = 1;

File candidate;

do {

String newName = baseName + " (" + index + ")" + extension;

candidate = new File(parent, newName);

index++;

} while (candidate.exists());

return candidate;

}

private boolean copyFile(File source, File dest) {

try (InputStream in = new FileInputStream(source);

OutputStream out = new FileOutputStream(dest)) {

byte[] buffer = new byte[4096];

int len;

while ((len = in.read(buffer)) > 0) {

out.write(buffer, 0, len);

}

return true;

} catch (IOException e) {

e.printStackTrace();

return false;

}

}

private boolean copyDirectory(File source, File dest) {

if (!dest.exists())

dest.mkdirs();

File[] files = source.listFiles();

if (files == null)

return false;

for (File file : files) {

File destFile = new File(dest, file.getName());

if (file.isDirectory()) {

if (!copyDirectory(file, destFile))

return false;

} else {

if (!copyFile(file, destFile))

return false;

}

}

return true;

}

private void handleRename(RenameRequest req) throws IOException {

String oldPath = req.getOldPath();

String newName = req.getNewName();

File oldFile = new File(getUserDir(), oldPath);

File newFile = new File(oldFile.getParentFile(), newName);

boolean renamed = oldFile.renameTo(newFile);

sendResponse(new ResponseDTO(renamed, renamed ? "OK" : "Rename failed"));

}

private void handleListFiles(ListFilesRequest req) throws IOException {

String relativePath = req.getRelativePath();

File folder = new File(getUserDir(), relativePath);

if (!folder.exists() || !folder.isDirectory()) {

sendResponse(new ResponseDTO(false, "Folder not found"));

return;

}

File[] files = folder.listFiles();

List<FileInfo> list = new ArrayList<>();

if (files != null) {

for (File f : files) {

list.add(toFileInfo(f, -1));

}

}

sendResponse(new ResponseDTO(true, "OK", list));

}

private void sendResponse(ResponseDTO response) throws IOException {

try {

byte[] data = serialize(response);

byte[] encrypted = CryptoUtils.encryptAES(data, aesKey);

oos.writeObject(encrypted);

oos.flush();

} catch (Exception e) {

throw new IOException("Encryption failed", e);

}

}

}

**UserDB.java**

package Server;

import java.io.File;

import java.sql.\*;

import org.mindrot.BCrypt;

public class UserDB {

private static final String BASE\_DIR = "server\_data/users";

private static final String DB\_URL = "jdbc:sqlite:server\_data/users/users.db";

public UserDB() {

new File(BASE\_DIR).mkdirs();

try (Connection conn = DriverManager.getConnection(DB\_URL)) {

if (conn != null) {

String createTable = "CREATE TABLE IF NOT EXISTS users (" +

"username TEXT PRIMARY KEY," +

"password TEXT NOT NULL" +

");";

Statement stmt = conn.createStatement();

stmt.execute(createTable);

}

} catch (SQLException e) {

e.printStackTrace();

}

}

public boolean userExists(String username) {

String query = "SELECT username FROM users WHERE username = ?";

try (Connection conn = DriverManager.getConnection(DB\_URL);

PreparedStatement pstmt = conn.prepareStatement(query)) {

pstmt.setString(1, username);

ResultSet rs = pstmt.executeQuery();

return rs.next();

} catch (SQLException e) {

e.printStackTrace();

return false;

}

}

public boolean insertUser(String username, String password) {

String query = "INSERT INTO users(username, password) VALUES(?, ?)";

try (Connection conn = DriverManager.getConnection(DB\_URL);

PreparedStatement pstmt = conn.prepareStatement(query)) {

// Генерируем хэш пароля с BCrypt. Второй параметр - число раундов.

String hashedPassword = BCrypt.hashpw(password, BCrypt.gensalt(12));

pstmt.setString(1, username);

pstmt.setString(2, hashedPassword);

int affectedRows = pstmt.executeUpdate();

return affectedRows > 0;

} catch (SQLException e) {

e.printStackTrace();

return false;

}

}

public boolean checkPassword(String username, String password) {

String query = "SELECT password FROM users WHERE username = ?";

try (Connection conn = DriverManager.getConnection(DB\_URL);

PreparedStatement pstmt = conn.prepareStatement(query)) {

pstmt.setString(1, username);

ResultSet rs = pstmt.executeQuery();

if (rs.next()) {

String storedHash = rs.getString("password");

// Сравниваем введённый пароль с хэшем

return BCrypt.checkpw(password, storedHash);

}

} catch (SQLException e) {

e.printStackTrace();

}

return false;

}

}

## **Приложение А.2**

Код клиента

**ClientApp.java**

package client;

import javax.swing.SwingUtilities;

public class ClientApp {

public static void main(String[] args) {

FileManagerClient client = new FileManagerClient("127.0.0.1", 12345);

SwingUtilities.invokeLater(new Runnable() {

@Override

public void run() {

LoginFrame loginFrame = new LoginFrame(client);

loginFrame.setTitle("Вход в файловый менеджер");

loginFrame.setDefaultCloseOperation(javax.swing.JFrame.EXIT\_ON\_CLOSE);

loginFrame.setVisible(true);

}

});

}

}

**FileManagerClient.java**

package client;

import java.io.\*;

import java.net.\*;

import java.util.List;

import DTO.\*;

import Security.CryptoUtils;

import java.security.PublicKey;

import java.util.Arrays;

import javax.crypto.SecretKey;

import javax.swing.JOptionPane;

public class FileManagerClient {

private String serverIp;

private int serverPort;

private Socket socket;

private ObjectOutputStream oos;

private ObjectInputStream ois;

private SecretKey aesKey;

private PublicKey serverPublicKey;

public FileManagerClient(String ip, int port) {

this.serverIp = ip;

this.serverPort = port;

connect();

}

private void connect() {

try {

socket = new Socket(serverIp, serverPort);

oos = new ObjectOutputStream(socket.getOutputStream());

oos.flush();

ois = new ObjectInputStream(socket.getInputStream());

serverPublicKey = CryptoUtils.publicKeyFromString((String) ois.readObject());

aesKey = CryptoUtils.generateAESKey();

byte[] encryptedAESKey = CryptoUtils.encryptRSA(aesKey.getEncoded(), serverPublicKey);

oos.writeObject(encryptedAESKey);

oos.flush();

} catch(Exception ex) {

ex.printStackTrace();

}

}

private ResponseDTO sendRequest(RequestDTO request) {

try {

byte[] data = serialize(request);

byte[] encrypted = CryptoUtils.encryptAES(data, aesKey);

oos.writeObject(encrypted);

oos.flush();

byte[] encryptedResponse = (byte[]) ois.readObject();

byte[] decrypted = CryptoUtils.decryptAES(encryptedResponse, aesKey);

return (ResponseDTO) deserialize(decrypted);

} catch(Exception ex) {

ex.printStackTrace();

return new ResponseDTO(false, "Exception: " + ex.getMessage());

}

}

private byte[] serialize(Object obj) throws IOException {

ByteArrayOutputStream bos = new ByteArrayOutputStream();

ObjectOutputStream oos = new ObjectOutputStream(bos);

oos.writeObject(obj);

return bos.toByteArray();

}

private Object deserialize(byte[] data) throws Exception {

ByteArrayInputStream bis = new ByteArrayInputStream(data);

ObjectInputStream ois = new ObjectInputStream(bis);

return ois.readObject();

}

public boolean register(String username, String password) {

RegisterRequest req = new RegisterRequest(username, password);

ResponseDTO resp = sendRequest(req);

return resp.isSuccess();

}

public boolean login(String username, String password) {

LoginRequest req = new LoginRequest(username, password);

ResponseDTO resp = sendRequest(req);

return resp.isSuccess();

}

public boolean uploadFile(File file, String destPath) {

return uploadFile(file, destPath, false);

}

private boolean uploadFile(File file, String destPath, boolean overwrite) {

try {

UploadRequest req = new UploadRequest(destPath, file.length(), overwrite);

ResponseDTO resp = sendRequest(req);

if (!resp.isSuccess() && "File conflict".equals(resp.getMessage())) {

Object[] options = {"Перезаписать", "Переименовать"};

int choice = JOptionPane.showOptionDialog(null,

"Файл с таким именем уже существует. Выберите действие:",

"Конфликт файла",

JOptionPane.YES\_NO\_OPTION,

JOptionPane.QUESTION\_MESSAGE,

null,

options,

options[0]);

if (choice == JOptionPane.CANCEL\_OPTION || choice == -1) {

return false;

}

if (choice == JOptionPane.YES\_OPTION) {

return uploadFile(file, destPath, true);

} else if (choice == JOptionPane.NO\_OPTION) {

File original = new File(destPath);

String currentName = original.getName();

Object input = JOptionPane.showInputDialog(

null,

"Введите новое имя файла:",

"Переименование",

JOptionPane.QUESTION\_MESSAGE,

null,

null,

currentName

);

if (input == null || input.toString().trim().isEmpty()) {

return false;

}

String newName = input.toString();

String parent = original.getParent();

String newDestPath = (parent != null ? parent + File.separator : "") + newName;

return uploadFile(file, newDestPath, false);

}

} else if (!resp.isSuccess() && !"READY".equals(resp.getMessage())) {

return false;

}

FileInputStream fis = new FileInputStream(file);

byte[] buffer = new byte[4096];

int bytesRead;

while ((bytesRead = fis.read(buffer)) != -1) {

byte[] block = Arrays.copyOf(buffer, bytesRead);

byte[] encryptedBlock = CryptoUtils.encryptAES(block, aesKey);

oos.writeObject(encryptedBlock);

}

fis.close();

oos.flush();

byte[] encryptedResponse = (byte[]) ois.readObject();

byte[] decrypted = CryptoUtils.decryptAES(encryptedResponse, aesKey);

ResponseDTO finalResp = (ResponseDTO) deserialize(decrypted);

return finalResp.isSuccess();

} catch (Exception ex) {

ex.printStackTrace();

}

return false;

}

public boolean downloadFile(String filePath, File saveTo) {

try {

DownloadRequest req = new DownloadRequest(filePath);

ResponseDTO resp = sendRequest(req);

if (!resp.isSuccess()) {

return false;

}

long fileSize = (Long) resp.getData();

FileOutputStream fos = new FileOutputStream(saveTo);

byte[] buffer = new byte[4096];

long remaining = fileSize;

while (remaining > 0) {

byte[] encryptedBlock = (byte[]) ois.readObject();

byte[] decrypted = CryptoUtils.decryptAES(encryptedBlock, aesKey);

fos.write(decrypted);

remaining -= decrypted.length;

}

fos.close();

return true;

} catch(Exception ex) {

ex.printStackTrace();

}

return false;

}

public boolean createFolder(String folderPath) {

CreateFolderRequest req = new CreateFolderRequest(folderPath);

ResponseDTO resp = sendRequest(req);

return resp.isSuccess();

}

public boolean delete(String path) {

DeleteRequest req = new DeleteRequest(path);

ResponseDTO resp = sendRequest(req);

return resp.isSuccess();

}

public List<FileInfo> search(String query) {

SearchRequest req = new SearchRequest(query);

ResponseDTO resp = sendRequest(req);

if (!resp.isSuccess()) {

return null;

}

return (List<FileInfo>) resp.getData();

}

public boolean copy(String source, String destination) {

CopyRequest req = new CopyRequest(source, destination);

ResponseDTO resp = sendRequest(req);

return resp.isSuccess();

}

public boolean rename(String oldPath, String newName) {

RenameRequest req = new RenameRequest(oldPath, newName);

ResponseDTO resp = sendRequest(req);

return resp.isSuccess();

}

// Запрос содержимого папки.

public List<FileInfo> listFolder(String relativePath) {

ListFilesRequest req = new ListFilesRequest(relativePath);

ResponseDTO resp = sendRequest(req);

if (!resp.isSuccess()) {

return null;

}

return (List<FileInfo>) resp.getData();

}

}

**Frame.java**

package client;

import DTO.\*;

import Component.\*;

import java.awt.BorderLayout;

import java.io.File;

import java.text.SimpleDateFormat;

import java.util.Collections;

import java.util.Comparator;

import java.util.Date;

import java.util.List;

import javax.swing.JFileChooser;

import javax.swing.JOptionPane;

import javax.swing.table.DefaultTableModel;

public class Frame extends javax.swing.JFrame {

private String currentPath = "";

private String copiedFilePath = null;

private FileManagerClient client;

private DefaultTableModel tableModel;

private java.util.List<FileInfo> fileList;

private boolean sortByNameAscending = true;

private boolean sortBySizeAscending = true;

private boolean sortByTypeAscending = true;

private boolean sortByDateAscending = true;

private BreadcrumbPanel breadcrumbPanel;

/\*\*

\* Creates new form Frame

\*/

public Frame() {

initComponents();

}

public Frame(FileManagerClient client) {

this.client = client;

initComponents();

initTable();

refreshFileList();

initBreadcrumbPanel();

}

private void initTable() {

String[] columnNames = {"Имя", "Дата изменения", "Тип", "Размер"};

tableModel = new DefaultTableModel(columnNames, 0) {

@Override

public boolean isCellEditable(int row, int column) {

return false; // запрет редактирования ячеек

}

};

jTableFiles.setModel(tableModel);

}

private void initBreadcrumbPanel() {

breadcrumbPanel = new BreadcrumbPanel(new BreadcrumbPanel.BreadcrumbListener() {

@Override

public void onBreadcrumbClicked(String path) {

currentPath = path;

refreshFileList();

breadcrumbPanel.updateBreadcrumb(currentPath);

}

});

jPanelBreadcrumb.setLayout(new BorderLayout());

jPanelBreadcrumb.removeAll();

jPanelBreadcrumb.add(breadcrumbPanel, BorderLayout.CENTER);

jPanelBreadcrumb.revalidate();

jPanelBreadcrumb.repaint();

}

/\*\*

\* 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() {

jPopupMenu1 = new javax.swing.JPopupMenu();

jMenu1 = new javax.swing.JMenu();

jPopupMenu2 = new javax.swing.JPopupMenu();

jFrame1 = new javax.swing.JFrame();

jFrame2 = new javax.swing.JFrame();

jMenuItem1 = new javax.swing.JMenuItem();

jMenuItem2 = new javax.swing.JMenuItem();

popupMenu1 = new java.awt.PopupMenu();

menuBar1 = new java.awt.MenuBar();

menu1 = new java.awt.Menu();

menu2 = new java.awt.Menu();

menuBar2 = new java.awt.MenuBar();

menu3 = new java.awt.Menu();

menu4 = new java.awt.Menu();

popupMenu2 = new java.awt.PopupMenu();

jPopupMenu3 = new javax.swing.JPopupMenu();

jPopupMenu4 = new javax.swing.JPopupMenu();

jPopupMenu5 = new javax.swing.JPopupMenu();

jMenuBar1 = new javax.swing.JMenuBar();

jMenu2 = new javax.swing.JMenu();

jMenu3 = new javax.swing.JMenu();

jMenuBar3 = new javax.swing.JMenuBar();

jMenu6 = new javax.swing.JMenu();

jMenu7 = new javax.swing.JMenu();

jMenu8 = new javax.swing.JMenu();

jCheckBoxMenuItem1 = new javax.swing.JCheckBoxMenuItem();

jMenuBar4 = new javax.swing.JMenuBar();

jMenu9 = new javax.swing.JMenu();

jMenu10 = new javax.swing.JMenu();

jMenu11 = new javax.swing.JMenu();

jMenuBar5 = new javax.swing.JMenuBar();

jMenu12 = new javax.swing.JMenu();

jMenu13 = new javax.swing.JMenu();

jPopupMenu6 = new javax.swing.JPopupMenu();

jPanel1 = new javax.swing.JPanel();

jCheckBoxMenuItem2 = new javax.swing.JCheckBoxMenuItem();

jScrollPane1 = new javax.swing.JScrollPane();

jEditorPane1 = new javax.swing.JEditorPane();

jMenuItem3 = new javax.swing.JMenuItem();

filler1 = new javax.swing.Box.Filler(new java.awt.Dimension(0, 0), new java.awt.Dimension(0, 0), new java.awt.Dimension(32767, 0));

jTextFieldSearch = new javax.swing.JTextField();

jButtonBack = new javax.swing.JButton();

jButtonSearch = new javax.swing.JButton();

jScrollPane3 = new javax.swing.JScrollPane();

jTableFiles = new javax.swing.JTable();

jPanelBreadcrumb = new javax.swing.JPanel();

jMenuBar = new javax.swing.JMenuBar();

jMenuSort = new javax.swing.JMenu();

jMenuSortName = new javax.swing.JMenuItem();

jMenuSortSize = new javax.swing.JMenuItem();

jMenuSortType = new javax.swing.JMenuItem();

jMenuSortDate = new javax.swing.JMenuItem();

jMenuView = new javax.swing.JMenu();

jMenuViewTable = new javax.swing.JMenuItem();

jMenuViewList = new javax.swing.JMenuItem();

jMenuUpload = new javax.swing.JMenu();

jMenuDownload = new javax.swing.JMenu();

jMenuDelete = new javax.swing.JMenu();

jMenuCreateFolder = new javax.swing.JMenu();

jMenuRename = new javax.swing.JMenu();

jMenuCopy = new javax.swing.JMenu();

jMenuInsert = new javax.swing.JMenu();

jMenuRefresh = new javax.swing.JMenu();

jMenu1.setText("jMenu1");

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

jFrame1.getContentPane().setLayout(jFrame1Layout);

jFrame1Layout.setHorizontalGroup(

jFrame1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGap(0, 400, Short.MAX\_VALUE)

);

jFrame1Layout.setVerticalGroup(

jFrame1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGap(0, 300, Short.MAX\_VALUE)

);

javax.swing.GroupLayout jFrame2Layout = new javax.swing.GroupLayout(jFrame2.getContentPane());

jFrame2.getContentPane().setLayout(jFrame2Layout);

jFrame2Layout.setHorizontalGroup(

jFrame2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGap(0, 400, Short.MAX\_VALUE)

);

jFrame2Layout.setVerticalGroup(

jFrame2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGap(0, 300, Short.MAX\_VALUE)

);

jMenuItem1.setText("jMenuItem1");

jMenuItem2.setText("jMenuItem2");

popupMenu1.setLabel("popupMenu1");

menu1.setLabel("File");

menuBar1.add(menu1);

menu2.setLabel("Edit");

menuBar1.add(menu2);

menu3.setLabel("File");

menuBar2.add(menu3);

menu4.setLabel("Edit");

menuBar2.add(menu4);

popupMenu2.setLabel("popupMenu2");

jMenu2.setText("File");

jMenuBar1.add(jMenu2);

jMenu3.setText("Edit");

jMenuBar1.add(jMenu3);

jMenu6.setText("File");

jMenuBar3.add(jMenu6);

jMenu7.setText("Edit");

jMenuBar3.add(jMenu7);

jMenu8.setText("jMenu8");

jCheckBoxMenuItem1.setSelected(true);

jCheckBoxMenuItem1.setText("jCheckBoxMenuItem1");

jMenu9.setText("File");

jMenuBar4.add(jMenu9);

jMenu10.setText("Edit");

jMenuBar4.add(jMenu10);

jMenu11.setText("jMenu11");

jMenu12.setText("File");

jMenuBar5.add(jMenu12);

jMenu13.setText("Edit");

jMenuBar5.add(jMenu13);

javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);

jPanel1.setLayout(jPanel1Layout);

jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGap(0, 100, Short.MAX\_VALUE)

);

jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGap(0, 100, Short.MAX\_VALUE)

);

jCheckBoxMenuItem2.setSelected(true);

jCheckBoxMenuItem2.setText("jCheckBoxMenuItem2");

jScrollPane1.setViewportView(jEditorPane1);

jMenuItem3.setText("jMenuItem3");

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

jTextFieldSearch.setBorder(javax.swing.BorderFactory.createEtchedBorder());

jButtonBack.setText("Назад");

jButtonBack.setMaximumSize(new java.awt.Dimension(72, 25));

jButtonBack.setMinimumSize(new java.awt.Dimension(72, 25));

jButtonBack.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButtonBackActionPerformed(evt);

}

});

jButtonSearch.setText("Поиск");

jButtonSearch.setMaximumSize(new java.awt.Dimension(72, 25));

jButtonSearch.setMinimumSize(new java.awt.Dimension(72, 25));

jButtonSearch.setName(""); // NOI18N

jButtonSearch.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButtonSearchActionPerformed(evt);

}

});

jTableFiles.setModel(new javax.swing.table.DefaultTableModel(

new Object [][] {

},

new String [] {

"Имя", "Дата изменения", "Размер", "Тип"

}

));

jTableFiles.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

jTableFilesMouseClicked(evt);

}

});

jScrollPane3.setViewportView(jTableFiles);

jPanelBreadcrumb.setBackground(new java.awt.Color(255, 255, 255));

jPanelBreadcrumb.setBorder(javax.swing.BorderFactory.createEtchedBorder());

javax.swing.GroupLayout jPanelBreadcrumbLayout = new javax.swing.GroupLayout(jPanelBreadcrumb);

jPanelBreadcrumb.setLayout(jPanelBreadcrumbLayout);

jPanelBreadcrumbLayout.setHorizontalGroup(

jPanelBreadcrumbLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGap(0, 0, Short.MAX\_VALUE)

);

jPanelBreadcrumbLayout.setVerticalGroup(

jPanelBreadcrumbLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGap(0, 25, Short.MAX\_VALUE)

);

jMenuBar.setBorderPainted(false);

jMenuBar.setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT\_CURSOR));

jMenuBar.setFont(jMenuBar.getFont());

jMenuBar.setMinimumSize(new java.awt.Dimension(400, 50));

jMenuBar.setPreferredSize(new java.awt.Dimension(400, 35));

jMenuSort.setText("Сортировать");

jMenuSortName.setText("Названию");

jMenuSortName.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jMenuSortNameActionPerformed(evt);

}

});

jMenuSort.add(jMenuSortName);

jMenuSortSize.setText("Размеру");

jMenuSortSize.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jMenuSortSizeActionPerformed(evt);

}

});

jMenuSort.add(jMenuSortSize);

jMenuSortType.setText("Типу");

jMenuSortType.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jMenuSortTypeActionPerformed(evt);

}

});

jMenuSort.add(jMenuSortType);

jMenuSortDate.setText("Дате изменения");

jMenuSortDate.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jMenuSortDateActionPerformed(evt);

}

});

jMenuSort.add(jMenuSortDate);

jMenuBar.add(jMenuSort);

jMenuView.setText("Вид");

jMenuViewTable.setText("Таблица");

jMenuViewTable.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jMenuViewTableActionPerformed(evt);

}

});

jMenuView.add(jMenuViewTable);

jMenuViewList.setText("Список");

jMenuViewList.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jMenuViewListActionPerformed(evt);

}

});

jMenuView.add(jMenuViewList);

jMenuBar.add(jMenuView);

jMenuUpload.setText("Загрузить");

jMenuUpload.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

jMenuUploadMouseClicked(evt);

}

});

jMenuBar.add(jMenuUpload);

jMenuDownload.setText("Скачать");

jMenuDownload.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

jMenuDownloadMouseClicked(evt);

}

});

jMenuBar.add(jMenuDownload);

jMenuDelete.setText("Удалить");

jMenuDelete.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

jMenuDeleteMouseClicked(evt);

}

});

jMenuBar.add(jMenuDelete);

jMenuCreateFolder.setText("Создать папку");

jMenuCreateFolder.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

jMenuCreateFolderMouseClicked(evt);

}

});

jMenuBar.add(jMenuCreateFolder);

jMenuRename.setText("Переименовать");

jMenuRename.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

jMenuRenameMouseClicked(evt);

}

});

jMenuBar.add(jMenuRename);

jMenuCopy.setText("Копировать");

jMenuCopy.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

jMenuCopyMouseClicked(evt);

}

});

jMenuBar.add(jMenuCopy);

jMenuInsert.setText("Вставить");

jMenuInsert.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

jMenuInsertMouseClicked(evt);

}

});

jMenuBar.add(jMenuInsert);

jMenuRefresh.setText("Обновить");

jMenuRefresh.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

jMenuRefreshMouseClicked(evt);

}

});

jMenuBar.add(jMenuRefresh);

setJMenuBar(jMenuBar);

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

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addContainerGap()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addComponent(jButtonBack, javax.swing.GroupLayout.PREFERRED\_SIZE, 94, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(jPanelBreadcrumb, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(jTextFieldSearch, javax.swing.GroupLayout.PREFERRED\_SIZE, 127, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(jButtonSearch, javax.swing.GroupLayout.PREFERRED\_SIZE, 94, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addComponent(jScrollPane3, javax.swing.GroupLayout.DEFAULT\_SIZE, 1087, Short.MAX\_VALUE))

.addContainerGap())

);

layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addContainerGap()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jPanelBreadcrumb, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jTextFieldSearch, javax.swing.GroupLayout.PREFERRED\_SIZE, 29, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jButtonSearch, javax.swing.GroupLayout.PREFERRED\_SIZE, 30, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addComponent(jButtonBack, javax.swing.GroupLayout.PREFERRED\_SIZE, 30, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(jScrollPane3, javax.swing.GroupLayout.DEFAULT\_SIZE, 527, Short.MAX\_VALUE)

.addContainerGap())

);

pack();

}// </editor-fold>

private void jMenuSortNameActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

sortByNameAscending = !sortByNameAscending;

sortFiles("name", sortByNameAscending);

}

private void jMenuSortSizeActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

sortBySizeAscending = !sortBySizeAscending;

sortFiles("size", sortBySizeAscending);

}

private void jMenuSortDateActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

sortByDateAscending = !sortByDateAscending;

sortFiles("date", sortByDateAscending);

}

private void jButtonBackActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

if (!currentPath.isEmpty()) {

int sepIndex = currentPath.lastIndexOf(File.separator);

if (sepIndex == -1)

currentPath = "";

else

currentPath = currentPath.substring(0, sepIndex);

refreshFileList();

breadcrumbPanel.updateBreadcrumb(currentPath);

}

}

private void jMenuUploadMouseClicked(java.awt.event.MouseEvent evt) {

// TODO add your handling code here:

uploadFile();

}

private void jMenuDownloadMouseClicked(java.awt.event.MouseEvent evt) {

// TODO add your handling code here:

downloadFile();

}

private void jMenuDeleteMouseClicked(java.awt.event.MouseEvent evt) {

// TODO add your handling code here:

deleteItem();

}

private void jMenuCreateFolderMouseClicked(java.awt.event.MouseEvent evt) {

// TODO add your handling code here:

createFolder();

}

private void jMenuRenameMouseClicked(java.awt.event.MouseEvent evt) {

// TODO add your handling code here:

renameItem();

}

private void jMenuCopyMouseClicked(java.awt.event.MouseEvent evt) {

// TODO add your handling code here:

copyItem();

}

private void jButtonSearchActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

String query = jTextFieldSearch.getText().trim();

List<FileInfo> results = client.search(query);

if (results != null) {

fileList = results;

updateTableModel(results);

} else {

JOptionPane.showMessageDialog(this, "Поиск не дал результатов");

}

}

private void jMenuInsertMouseClicked(java.awt.event.MouseEvent evt) {

// TODO add your handling code here:

insertCopiedItem();

}

private void jMenuSortTypeActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

sortByTypeAscending = !sortByTypeAscending;

sortFiles("type", sortByNameAscending);

}

private void jTableFilesMouseClicked(java.awt.event.MouseEvent evt) {

// TODO add your handling code here:

if (evt.getClickCount() == 2) {

int selectedRow = jTableFiles.getSelectedRow();

if (selectedRow >= 0) {

String selectedItem = (String) tableModel.getValueAt(selectedRow, 0);

if (selectedItem != null) {

String newPath = currentPath.isEmpty() ? selectedItem : currentPath + File.separator + selectedItem;

List<FileInfo> folderContents = client.listFolder(newPath);

if (folderContents != null) {

currentPath = newPath;

refreshFileList();

breadcrumbPanel.updateBreadcrumb(currentPath);

} else {

JOptionPane.showMessageDialog(Frame.this, "Это не папка или произошла ошибка");

}

}

}

}

}

private void jMenuViewListActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

setListView(true);

}

private void jMenuViewTableActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

setListView(false);

}

private void jMenuRefreshMouseClicked(java.awt.event.MouseEvent evt) {

// TODO add your handling code here:

refreshFileList();

breadcrumbPanel.updateBreadcrumb(currentPath);

}

// Переключение вида отображения

private void setListView(boolean listView) {

if (listView) {

jTableFiles.getColumnModel().getColumn(1).setMinWidth(0);

jTableFiles.getColumnModel().getColumn(1).setMaxWidth(0);

jTableFiles.getColumnModel().getColumn(2).setMinWidth(0);

jTableFiles.getColumnModel().getColumn(2).setMaxWidth(0);

jTableFiles.getColumnModel().getColumn(3).setMinWidth(0);

jTableFiles.getColumnModel().getColumn(3).setMaxWidth(0);

} else {

jTableFiles.setAutoResizeMode(javax.swing.JTable.AUTO\_RESIZE\_ALL\_COLUMNS);

for (int i = 0; i < jTableFiles.getColumnCount(); i++) {

javax.swing.table.TableColumn column = jTableFiles.getColumnModel().getColumn(i);

column.setMinWidth(15);

column.setMaxWidth(Integer.MAX\_VALUE);

column.setPreferredWidth(jTableFiles.getWidth() / jTableFiles.getColumnCount());

}

jTableFiles.doLayout();

}

}

private void sortFiles(String criterion, boolean ascending) {

if (fileList == null) return;

Comparator<FileInfo> comparator;

switch (criterion) {

case "name":

comparator = Comparator.comparing(FileInfo::getName, String.CASE\_INSENSITIVE\_ORDER);

break;

case "size":

comparator = Comparator.comparingLong(FileInfo::getSize);

break;

case "date":

comparator = Comparator.comparingLong(FileInfo::getLastModified);

break;

case "type":

comparator = Comparator.comparing(FileInfo::getFileType, String.CASE\_INSENSITIVE\_ORDER);

break;

default:

comparator = Comparator.comparing(FileInfo::getName, String.CASE\_INSENSITIVE\_ORDER);

}

if (!ascending) {

comparator = comparator.reversed();

}

Collections.sort(fileList, comparator);

updateTableModel(fileList);

}

private String formatFileSize(long size) {

if (size < 1024) {

return size + " Б";

} else if (size < 1024 \* 1024) {

double kb = size / 1024.0;

return String.format("%.1f КБ", kb);

} else if (size < 1024L \* 1024 \* 1024) {

double mb = size / (1024.0 \* 1024);

return String.format("%.1f МБ", mb);

} else {

double gb = size / (1024.0 \* 1024 \* 1024);

return String.format("%.1f Г", gb);

}

}

private void updateTableModel(List<FileInfo> files) {

tableModel.setRowCount(0);

SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

for (FileInfo f : files) {

String name = f.getName();

String lastMod = sdf.format(new Date(f.getLastModified()));

String size = "";

if (!f.getFileType().equalsIgnoreCase("Папка")) {

size = formatFileSize(f.getSize());

}

String type = f.getFileType();

tableModel.addRow(new Object[]{name, lastMod, type, size});

}

}

private void refreshFileList() {

List<FileInfo> files;

if (currentPath.isEmpty()) {

files = client.listFolder("");

} else {

files = client.listFolder(currentPath);

}

if (files != null) {

fileList = files;

updateTableModel(files);

}

}

private void uploadFile() {

JFileChooser fc = new JFileChooser();

int res = fc.showOpenDialog(this);

if (res == JFileChooser.APPROVE\_OPTION) {

File file = fc.getSelectedFile();

String destPath = file.getName();

String relativeDest = currentPath.isEmpty() ? destPath : currentPath + File.separator + destPath;

if (client.uploadFile(file, relativeDest)) {

refreshFileList();

} else {

JOptionPane.showMessageDialog(this, "Ошибка загрузки");

}

}

}

private void downloadFile() {

int selectedRow = jTableFiles.getSelectedRow();

if (selectedRow < 0) {

JOptionPane.showMessageDialog(this, "Выберите файл для скачивания");

return;

}

String fileName = (String) tableModel.getValueAt(selectedRow, 0);

String relativePath = currentPath.isEmpty() ? fileName : currentPath + File.separator + fileName;

JFileChooser fc = new JFileChooser();

fc.setSelectedFile(new File(fileName));

int res = fc.showSaveDialog(this);

if (res == JFileChooser.APPROVE\_OPTION) {

File saveTo = fc.getSelectedFile();

if (client.downloadFile(relativePath, saveTo)) {

JOptionPane.showMessageDialog(this, "Файл успешно скачан");

} else {

JOptionPane.showMessageDialog(this, "Не удалось загрузить файл");

}

}

}

private void createFolder() {

String folderName = JOptionPane.showInputDialog(this, "Введите название папки:");

if (folderName != null && !folderName.trim().isEmpty()) {

String relativePath = currentPath.isEmpty() ? folderName : currentPath + File.separator + folderName;

if (client.createFolder(relativePath)) {

refreshFileList();

} else {

JOptionPane.showMessageDialog(this, "Не удалось создать папку");

}

}

}

private void deleteItem() {

int selectedRow = jTableFiles.getSelectedRow();

if (selectedRow < 0) {

JOptionPane.showMessageDialog(this, "Выберите элемент для удаления");

return;

}

String fileName = (String) tableModel.getValueAt(selectedRow, 0);

String relativePath = currentPath.isEmpty() ? fileName : currentPath + File.separator + fileName;

int confirm = JOptionPane.showConfirmDialog(this, "Вы действительно хотите удалить: " + fileName + "?");

if (confirm == JOptionPane.YES\_OPTION) {

if (client.delete(relativePath)) {

refreshFileList();

} else {

JOptionPane.showMessageDialog(this, "Удаление не удалось");

}

}

}

private void copyItem() {

int selectedRow = jTableFiles.getSelectedRow();

if (selectedRow < 0) {

JOptionPane.showMessageDialog(this, "Выберите элемент для копирования");

return;

}

String fileName = (String) tableModel.getValueAt(selectedRow, 0);

copiedFilePath = currentPath.isEmpty() ? fileName : currentPath + File.separator + fileName;

}

private void insertCopiedItem() {

if (copiedFilePath == null) {

JOptionPane.showMessageDialog(this, "Ни один файл не был скопирован. Пожалуйста, сначала нажмите \"Копировать\".");

return;

}

File f = new File(copiedFilePath);

String fileName = f.getName();

String destination = currentPath.isEmpty() ? fileName : currentPath + File.separator + fileName;

if (client.copy(copiedFilePath, destination)) {

refreshFileList();

} else {

JOptionPane.showMessageDialog(this, "Копирование не удалось");

}

copiedFilePath = null;

}

private void renameItem() {

int selectedRow = jTableFiles.getSelectedRow();

if (selectedRow < 0) {

JOptionPane.showMessageDialog(this, "Выберите элемент для переименования");

return;

}

String oldName = (String) tableModel.getValueAt(selectedRow, 0);

String newName = JOptionPane.showInputDialog(this, "Введите новое имя:", oldName);

if (newName != null && !newName.trim().isEmpty()) {

String relativePath = currentPath.isEmpty() ? oldName : currentPath + File.separator + oldName;

if (client.rename(relativePath, newName)) {

refreshFileList();

} else {

JOptionPane.showMessageDialog(this, "Переименование не удалось");

}

}

}

/\*\*

\* @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/plaf.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 (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(Frame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(Frame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(Frame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(Frame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

}

//</editor-fold>

/\* Create and display the form \*/

java.awt.EventQueue.invokeLater(new Runnable() {

public void run() {

Frame frame = new Frame();

frame.setTitle("Файловый менеджер");

frame.setVisible(true);

}

});

}

// Variables declaration - do not modify

private javax.swing.Box.Filler filler1;

private javax.swing.JButton jButtonBack;

private javax.swing.JButton jButtonSearch;

private javax.swing.JCheckBoxMenuItem jCheckBoxMenuItem1;

private javax.swing.JCheckBoxMenuItem jCheckBoxMenuItem2;

private javax.swing.JEditorPane jEditorPane1;

private javax.swing.JFrame jFrame1;

private javax.swing.JFrame jFrame2;

private javax.swing.JMenu jMenu1;

private javax.swing.JMenu jMenu10;

private javax.swing.JMenu jMenu11;

private javax.swing.JMenu jMenu12;

private javax.swing.JMenu jMenu13;

private javax.swing.JMenu jMenu2;

private javax.swing.JMenu jMenu3;

private javax.swing.JMenu jMenu6;

private javax.swing.JMenu jMenu7;

private javax.swing.JMenu jMenu8;

private javax.swing.JMenu jMenu9;

private javax.swing.JMenuBar jMenuBar;

private javax.swing.JMenuBar jMenuBar1;

private javax.swing.JMenuBar jMenuBar3;

private javax.swing.JMenuBar jMenuBar4;

private javax.swing.JMenuBar jMenuBar5;

private javax.swing.JMenu jMenuCopy;

private javax.swing.JMenu jMenuCreateFolder;

private javax.swing.JMenu jMenuDelete;

private javax.swing.JMenu jMenuDownload;

private javax.swing.JMenu jMenuInsert;

private javax.swing.JMenuItem jMenuItem1;

private javax.swing.JMenuItem jMenuItem2;

private javax.swing.JMenuItem jMenuItem3;

private javax.swing.JMenu jMenuRefresh;

private javax.swing.JMenu jMenuRename;

private javax.swing.JMenu jMenuSort;

private javax.swing.JMenuItem jMenuSortDate;

private javax.swing.JMenuItem jMenuSortName;

private javax.swing.JMenuItem jMenuSortSize;

private javax.swing.JMenuItem jMenuSortType;

private javax.swing.JMenu jMenuUpload;

private javax.swing.JMenu jMenuView;

private javax.swing.JMenuItem jMenuViewList;

private javax.swing.JMenuItem jMenuViewTable;

private javax.swing.JPanel jPanel1;

private javax.swing.JPanel jPanelBreadcrumb;

private javax.swing.JPopupMenu jPopupMenu1;

private javax.swing.JPopupMenu jPopupMenu2;

private javax.swing.JPopupMenu jPopupMenu3;

private javax.swing.JPopupMenu jPopupMenu4;

private javax.swing.JPopupMenu jPopupMenu5;

private javax.swing.JPopupMenu jPopupMenu6;

private javax.swing.JScrollPane jScrollPane1;

private javax.swing.JScrollPane jScrollPane3;

private javax.swing.JTable jTableFiles;

private javax.swing.JTextField jTextFieldSearch;

private java.awt.Menu menu1;

private java.awt.Menu menu2;

private java.awt.Menu menu3;

private java.awt.Menu menu4;

private java.awt.MenuBar menuBar1;

private java.awt.MenuBar menuBar2;

private java.awt.PopupMenu popupMenu1;

private java.awt.PopupMenu popupMenu2;

// End of variables declaration

}

**LoginFrame.java**

package client;

import javax.swing.JOptionPane;

public class LoginFrame extends javax.swing.JFrame {

private FileManagerClient client;

/\*\*

\* Creates new form LoginFrame

\*/

public LoginFrame() {

initComponents();

}

public LoginFrame(FileManagerClient client) {

this.client = client;

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() {

jLabelLogin = new javax.swing.JLabel();

jTextFieldLogin = new javax.swing.JTextField();

jLabelPassword = new javax.swing.JLabel();

jButtonRegistration = new javax.swing.JButton();

jButtonLogin = new javax.swing.JButton();

jPasswordField = new javax.swing.JPasswordField();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

jLabelLogin.setFont(new java.awt.Font("Segoe UI", 0, 18)); // NOI18N

jLabelLogin.setText("Логин:");

jTextFieldLogin.setFont(new java.awt.Font("Segoe UI", 0, 14)); // NOI18N

jLabelPassword.setFont(new java.awt.Font("Segoe UI", 0, 18)); // NOI18N

jLabelPassword.setText("Пароль:");

jButtonRegistration.setText("Регистрация");

jButtonRegistration.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

jButtonRegistrationMouseClicked(evt);

}

});

jButtonLogin.setText("Войти");

jButtonLogin.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

jButtonLoginMouseClicked(evt);

}

});

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

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()

.addContainerGap(104, Short.MAX\_VALUE)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

.addGroup(layout.createSequentialGroup()

.addComponent(jLabelPassword, javax.swing.GroupLayout.PREFERRED\_SIZE, 96, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(jPasswordField))

.addGroup(layout.createSequentialGroup()

.addComponent(jLabelLogin, javax.swing.GroupLayout.PREFERRED\_SIZE, 96, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(jTextFieldLogin, javax.swing.GroupLayout.PREFERRED\_SIZE, 229, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()

.addComponent(jButtonLogin, javax.swing.GroupLayout.PREFERRED\_SIZE, 126, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jButtonRegistration, javax.swing.GroupLayout.PREFERRED\_SIZE, 126, javax.swing.GroupLayout.PREFERRED\_SIZE)))

.addGap(102, 102, 102))

);

layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()

.addContainerGap(82, Short.MAX\_VALUE)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jTextFieldLogin, javax.swing.GroupLayout.PREFERRED\_SIZE, 34, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jLabelLogin, javax.swing.GroupLayout.PREFERRED\_SIZE, 34, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGap(34, 34, 34)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

.addComponent(jLabelPassword, javax.swing.GroupLayout.DEFAULT\_SIZE, 34, Short.MAX\_VALUE)

.addComponent(jPasswordField))

.addGap(30, 30, 30)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jButtonRegistration, javax.swing.GroupLayout.PREFERRED\_SIZE, 35, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jButtonLogin, javax.swing.GroupLayout.PREFERRED\_SIZE, 35, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGap(68, 68, 68))

);

pack();

}// </editor-fold>

private void jButtonLoginMouseClicked(java.awt.event.MouseEvent evt) {

// TODO add your handling code here:

performLogin();

}

private void jButtonRegistrationMouseClicked(java.awt.event.MouseEvent evt) {

// TODO add your handling code here:

performRegistration();

}

private void performLogin() {

String login = jTextFieldLogin.getText().trim();

String password = new String(jPasswordField.getPassword()).trim();

if(client.login(login, password)){

Frame mainFrame = new Frame(client);

mainFrame.setTitle("Файловый менеджер");

mainFrame.setVisible(true);

this.dispose();

} else {

JOptionPane.showMessageDialog(this, "Неверный логин или пароль!", "Ошибка", JOptionPane.ERROR\_MESSAGE);

}

}

private void performRegistration() {

String login = jTextFieldLogin.getText().trim();

String password = new String(jPasswordField.getPassword()).trim();

if(client.register(login, password)){

JOptionPane.showMessageDialog(this, "Регистрация прошла успешно! Теперь вы можете войти.");

} else {

JOptionPane.showMessageDialog(this, "Ошибка регистрации", "Ошибка", JOptionPane.ERROR\_MESSAGE);

}

}

/\*\*

\* @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/plaf.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 (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(LoginFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(LoginFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(LoginFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(LoginFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

}

//</editor-fold>

FileManagerClient client = new FileManagerClient("127.0.0.1", 12345);

/\* Create and display the form \*/

java.awt.EventQueue.invokeLater(new Runnable() {

public void run() {

LoginFrame loginFrame = new LoginFrame(client);

loginFrame.setTitle("Вход в файловый менеджер");

loginFrame.setDefaultCloseOperation(Frame.EXIT\_ON\_CLOSE);

loginFrame.setVisible(true);

}

});

}

// Variables declaration - do not modify

private javax.swing.JButton jButtonLogin;

private javax.swing.JButton jButtonRegistration;

private javax.swing.JLabel jLabelLogin;

private javax.swing.JLabel jLabelPassword;

private javax.swing.JPasswordField jPasswordField;

private javax.swing.JTextField jTextFieldLogin;

// End of variables declaration

}

## **Приложение А.3**

Код библиотеки

**BreadcrumbPanel.java**

package Component;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

import java.io.File;

public class BreadcrumbPanel extends JPanel {

private String currentPath = "";

private BreadcrumbListener listener;

public BreadcrumbPanel(BreadcrumbListener listener) {

this.listener = listener;

setLayout(new FlowLayout(FlowLayout.LEFT, 4, 3));

setBackground(Color.WHITE);

updateBreadcrumb("");

}

public void updateBreadcrumb(String path) {

this.currentPath = path;

removeAll();

JLabel rootLabel = createLinkLabel("Главная", "");

add(rootLabel);

if (!path.isEmpty()) {

String[] parts = path.split("\\Q" + File.separator + "\\E");

String cumulative = "";

for (String part : parts) {

if (part.isEmpty()) continue;

cumulative += File.separator + part;

add(createArrowLabel());

JLabel partLabel = createLinkLabel(part, cumulative);

add(partLabel);

}

}

revalidate();

repaint();

}

private JLabel createArrowLabel() {

JLabel arrow = new JLabel(">");

arrow.setForeground(new Color(120, 120, 120));

arrow.setFont(new Font("Segoe UI", Font.PLAIN, 12));

arrow.setBorder(BorderFactory.createEmptyBorder(0, 2, 0, 2));

return arrow;

}

private JLabel createLinkLabel(String text, final String targetPath) {

JLabel label = new JLabel(text);

label.setForeground(Color.BLACK);

label.setFont(new Font("Segoe UI", Font.PLAIN, 12));

label.setCursor(Cursor.getPredefinedCursor(Cursor.HAND\_CURSOR));

label.setOpaque(true);

label.setBackground(Color.WHITE);

label.setBorder(BorderFactory.createEmptyBorder(2, 4, 2, 4));

label.addMouseListener(new MouseAdapter() {

@Override

public void mouseClicked(MouseEvent e) {

listener.onBreadcrumbClicked(targetPath);

}

@Override

public void mouseEntered(MouseEvent e) {

label.setBackground(new Color(230, 230, 230));

label.setForeground(Color.BLACK);

}

@Override

public void mouseExited(MouseEvent e) {

label.setBackground(Color.WHITE);

label.setForeground(Color.BLACK);

}

});

return label;

}

public interface BreadcrumbListener {

void onBreadcrumbClicked(String path);

}

}

**CommandType.java**

package DTO;

import java.io.Serializable;

public enum CommandType implements Serializable {

REGISTER,

LOGIN,

UPLOAD,

DOWNLOAD,

CREATE\_FOLDER,

DELETE,

SEARCH,

COPY,

RENAME,

SORT,

MOVE,

LIST\_FILES

}

**CopyRequest.java**

package DTO;

public class CopyRequest extends RequestDTO {

private static final long serialVersionUID = 1L;

private final String sourcePath;

private final String destinationPath;

public CopyRequest(String sourcePath, String destinationPath) {

super(CommandType.COPY);

this.sourcePath = sourcePath;

this.destinationPath = destinationPath;

}

public String getSourcePath() { return sourcePath; }

public String getDestinationPath() { return destinationPath; }

}

**CreateFolderRequest.java**

package DTO;

public class CreateFolderRequest extends RequestDTO {

private static final long serialVersionUID = 1L;

private final String folderPath;

public CreateFolderRequest(String folderPath) {

super(CommandType.CREATE\_FOLDER);

this.folderPath = folderPath;

}

public String getFolderPath() { return folderPath; }

}

**DeleteRequest.java**

package DTO;

public class DeleteRequest extends RequestDTO {

private static final long serialVersionUID = 1L;

private final String path;

public DeleteRequest(String path) {

super(CommandType.DELETE);

this.path = path;

}

public String getPath() { return path; }

}

**DownloadRequest.java**

package DTO;

public class DownloadRequest extends RequestDTO {

private static final long serialVersionUID = 1L;

private final String filePath;

public DownloadRequest(String filePath) {

super(CommandType.DOWNLOAD);

this.filePath = filePath;

}

public String getFilePath() { return filePath; }

}

**FileInfo.java**

package DTO;

import java.io.Serializable;

public class FileInfo implements Serializable {

private static final long serialVersionUID = 1L;

private final String name;

private final long size;

private final long lastModified;

private final String fileType;

public FileInfo(String name, long size, long lastModified, String fileType) {

this.name = name;

this.size = size;

this.lastModified = lastModified;

this.fileType = fileType;

}

public String getName() {

return name;

}

public long getSize() {

return size;

}

public long getLastModified() {

return lastModified;

}

public String getFileType() {

return fileType;

}

}

**ListFilesRequest.java**

package DTO;

public class ListFilesRequest extends RequestDTO {

private static final long serialVersionUID = 1L;

private final String relativePath;

public ListFilesRequest(String relativePath) {

super(CommandType.LIST\_FILES);

this.relativePath = relativePath;

}

public String getRelativePath() { return relativePath; }

}

**LoginRequest.java**

package DTO;

public class LoginRequest extends RequestDTO {

private static final long serialVersionUID = 1L;

private final String username;

private final String password;

public LoginRequest(String username, String password) {

super(CommandType.LOGIN);

this.username = username;

this.password = password;

}

public String getUsername() { return username; }

public String getPassword() { return password; }

}

**RegisterRequest.java**

package DTO;

public class RegisterRequest extends RequestDTO {

private static final long serialVersionUID = 1L;

private final String username;

private final String password;

public RegisterRequest(String username, String password) {

super(CommandType.REGISTER);

this.username = username;

this.password = password;

}

public String getUsername() { return username; }

public String getPassword() { return password; }

}

**RenameRequest.java**

package DTO;

public class RenameRequest extends RequestDTO {

private static final long serialVersionUID = 1L;

private final String oldPath;

private final String newName;

public RenameRequest(String oldPath, String newName) {

super(CommandType.RENAME);

this.oldPath = oldPath;

this.newName = newName;

}

public String getOldPath() { return oldPath; }

public String getNewName() { return newName; }

}

**RequestDTO.java**

package DTO;

import java.io.Serializable;

public abstract class RequestDTO implements Serializable {

private static final long serialVersionUID = 1L;

private final CommandType command;

public RequestDTO(CommandType command) {

this.command = command;

}

public CommandType getCommand() {

return command;

}

}

**ResponseDTO.java**

package DTO;

import java.io.Serializable;

public class ResponseDTO implements Serializable {

private static final long serialVersionUID = 1L;

private final boolean success;

private final String message;

private final Object data;

public ResponseDTO(boolean success, String message) {

this(success, message, null);

}

public ResponseDTO(boolean success, String message, Object data) {

this.success = success;

this.message = message;

this.data = data;

}

public boolean isSuccess() { return success; }

public String getMessage() { return message; }

public Object getData() { return data; }

}

**SearchRequest.java**

package DTO;

public class SearchRequest extends RequestDTO {

private static final long serialVersionUID = 1L;

private final String query;

public SearchRequest(String query) {

super(CommandType.SEARCH);

this.query = query;

}

public String getQuery() { return query; }

}

**UploadRequest.java**

package DTO;

import java.io.Serializable;

public class UploadRequest extends RequestDTO implements Serializable {

private static final long serialVersionUID = 1L;

private final String destinationPath;

private final long fileSize;

private final boolean overwrite;

public UploadRequest(String destinationPath, long fileSize, boolean overwrite) {

super(CommandType.UPLOAD);

this.destinationPath = destinationPath;

this.fileSize = fileSize;

this.overwrite = overwrite;

}

public String getDestinationPath() {

return destinationPath;

}

public long getFileSize() {

return fileSize;

}

public boolean isOverwrite() {

return overwrite;

}

}

**CryptoUtils.java**

package Security;

import java.io.ByteArrayOutputStream;

import javax.crypto.\*;

import javax.crypto.spec.\*;

import java.security.\*;

import java.security.spec.\*;

import java.util.Arrays;

import java.util.Base64;

//Шаг 1: Сервер генерирует приватный и публичный ключ

//

//Шаг 1: Клиент подключается к серверу.

//

//Шаг 2: Сервер отправляет клиенту свой публичный RSA-ключ в виде строки Base64.

//

//Шаг 3: Клиент генерирует AES-ключ (256 бит) для симметричного шифрования.

//

//Шаг 4: Клиент шифрует AES-ключ публичным RSA-ключом сервера и отправляет его.

//

//Шаг 5: Сервер расшифровывает AES-ключ своим приватным RSA-ключом.

//

//Обе стороны (клиент и сервер) теперь имеют общий AES-ключ для шифрования данных.

public class CryptoUtils {

private static final String AES\_ALG = "AES/GCM/NoPadding";

private static final String RSA\_ALG = "RSA/ECB/OAEPWithSHA-256AndMGF1Padding";

private static final int GCM\_TAG\_LENGTH = 128;

private static final int IV\_LENGTH = 12;

public static SecretKey generateAESKey() throws NoSuchAlgorithmException {

KeyGenerator keyGen = KeyGenerator.getInstance("AES");

keyGen.init(256);

return keyGen.generateKey();

}

public static byte[] encryptAES(byte[] data, SecretKey key) throws Exception {

Cipher cipher = Cipher.getInstance(AES\_ALG);

byte[] iv = new byte[IV\_LENGTH];

SecureRandom random = new SecureRandom();

random.nextBytes(iv);

GCMParameterSpec spec = new GCMParameterSpec(GCM\_TAG\_LENGTH, iv);

cipher.init(Cipher.ENCRYPT\_MODE, key, spec);

byte[] encrypted = cipher.doFinal(data);

ByteArrayOutputStream outputStream = new ByteArrayOutputStream();

outputStream.write(iv);

outputStream.write(encrypted);

return outputStream.toByteArray();

}

public static byte[] decryptAES(byte[] encryptedData, SecretKey key) throws Exception {

Cipher cipher = Cipher.getInstance(AES\_ALG);

byte[] iv = Arrays.copyOfRange(encryptedData, 0, IV\_LENGTH);

byte[] data = Arrays.copyOfRange(encryptedData, IV\_LENGTH, encryptedData.length);

GCMParameterSpec spec = new GCMParameterSpec(GCM\_TAG\_LENGTH, iv);

cipher.init(Cipher.DECRYPT\_MODE, key, spec);

return cipher.doFinal(data);

}

public static byte[] encryptRSA(byte[] data, PublicKey publicKey) throws Exception {

Cipher cipher = Cipher.getInstance(RSA\_ALG);

cipher.init(Cipher.ENCRYPT\_MODE, publicKey);

return cipher.doFinal(data);

}

public static byte[] decryptRSA(byte[] data, PrivateKey privateKey) throws Exception {

Cipher cipher = Cipher.getInstance(RSA\_ALG);

cipher.init(Cipher.DECRYPT\_MODE, privateKey);

return cipher.doFinal(data);

}

public static String keyToString(Key key) {

return Base64.getEncoder().encodeToString(key.getEncoded());

}

public static PublicKey publicKeyFromString(String key) throws Exception {

byte[] keyBytes = Base64.getDecoder().decode(key);

KeyFactory keyFactory = KeyFactory.getInstance("RSA");

return keyFactory.generatePublic(new X509EncodedKeySpec(keyBytes));

}

}

# **Приложение Б**

(Обязательное)

UML-диаграммы

## **Приложение Б.1**

UML-диаграмма вариантов использования приложения

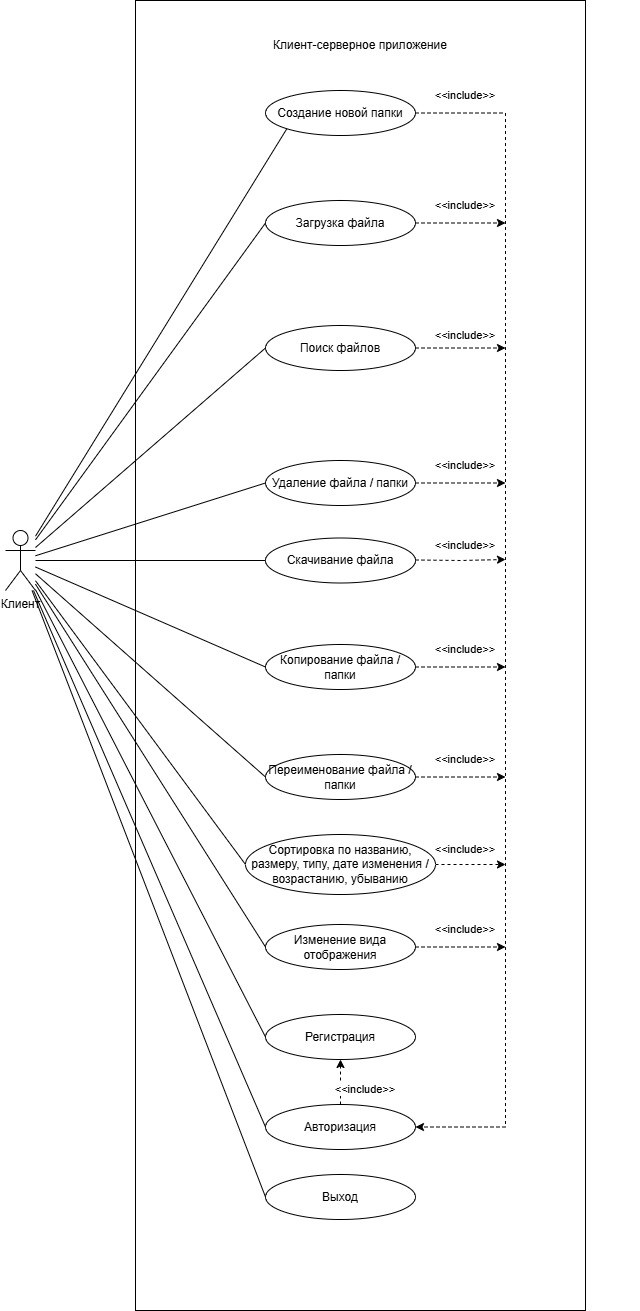


Рисунок 6 - UML-диаграмма вариантов использования приложения

## **Приложение Б.2**

UML-диаграмма последовательностей протокола взаимодействия клиента и сервера

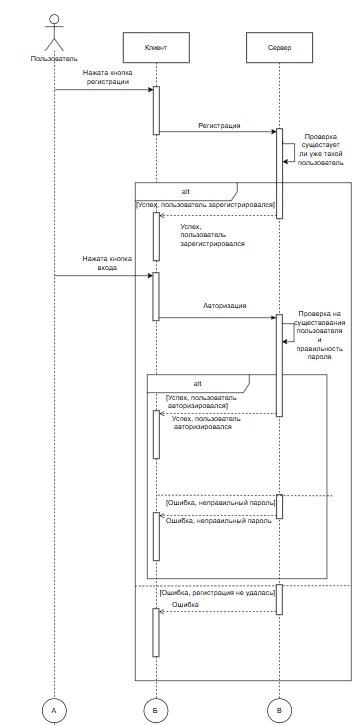


Рисунок 1 - UML-диаграмма последовательностей протокола взаимодействия клиента и сервера часть 1

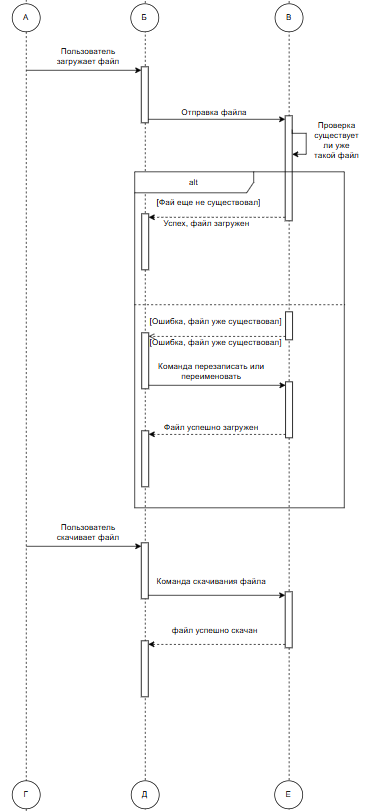
****

Рисунок 2 - UML-диаграмма последовательностей протокола взаимодействия клиента и сервера часть 2

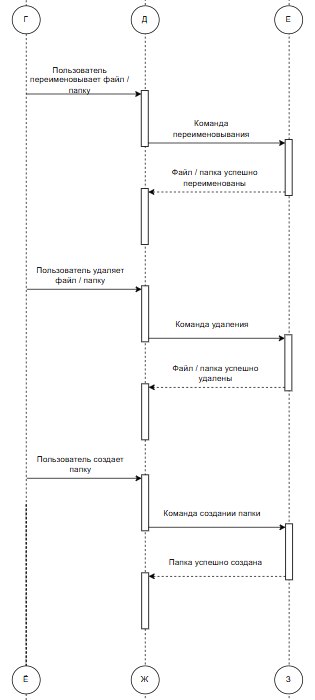
****

Рисунок 3 - UML-диаграмма последовательностей протокола взаимодействия клиента и сервера часть 3

Изображение выглядит как текст, диаграмма, Параллельный, линия

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 4 - UML-диаграмма последовательностей протокола взаимодействия клиента и сервера часть 4

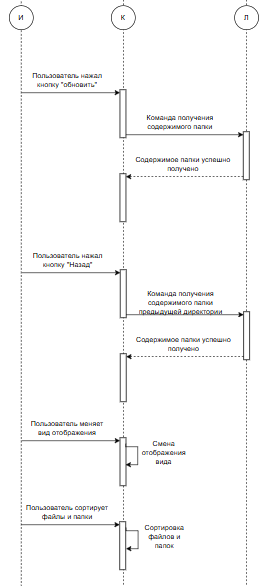


Рисунок 5 - UML-диаграмма последовательностей протокола взаимодействия клиента и сервера часть 5

## **Приложение Б.3**

UML-диаграмма развертывания

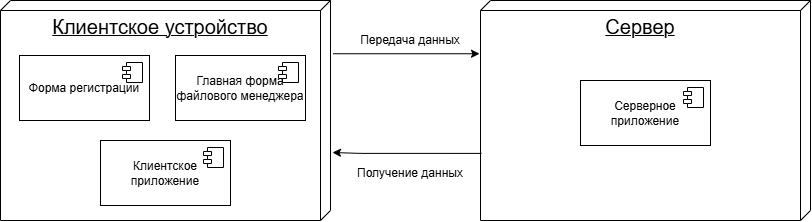


Рисунок 7 - UML-диаграмма развертывания

## **Приложение Б.4**

UML-диаграмма состояний клиента

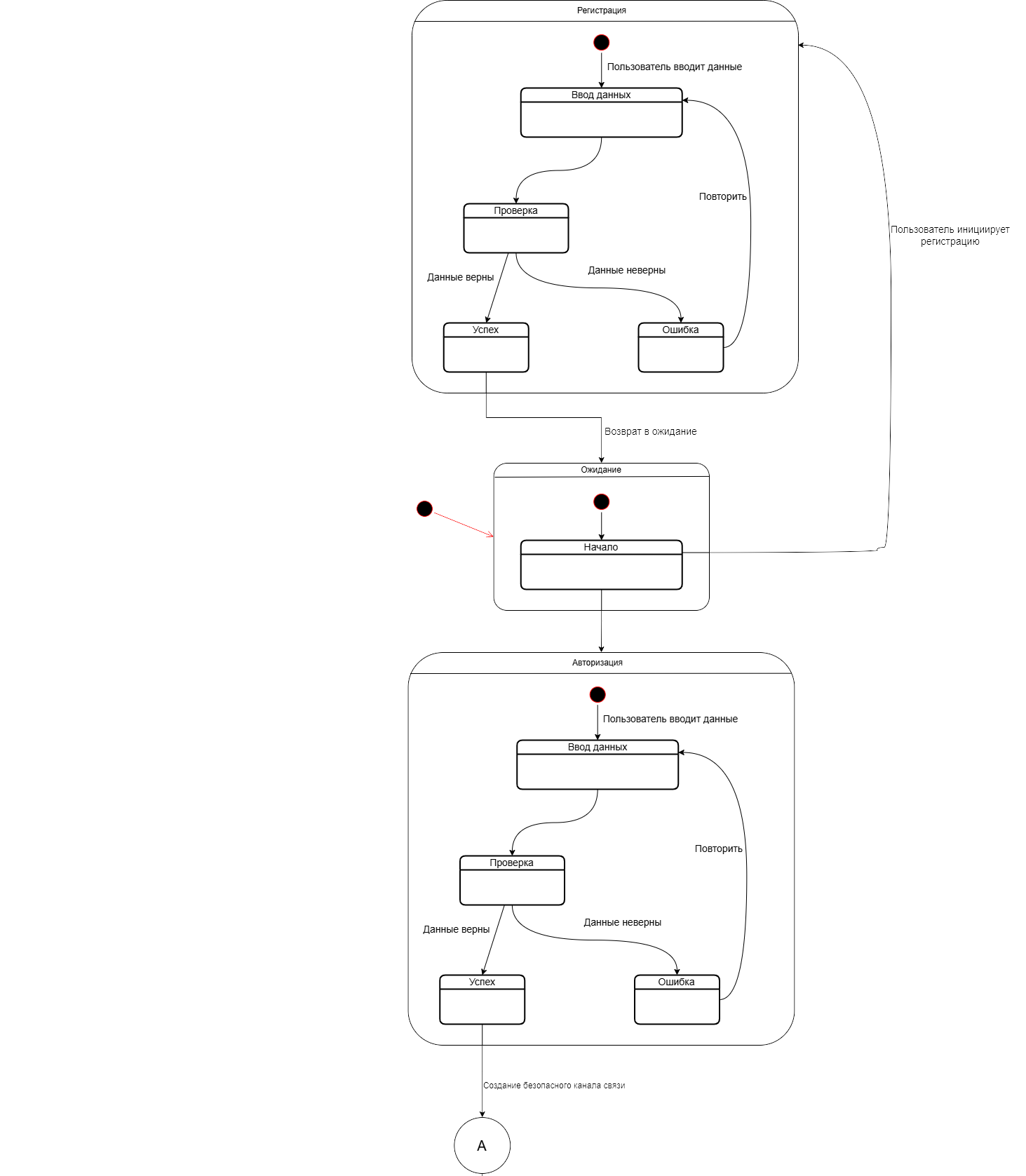


Рисунок 8 - UML-диаграмма состояний клиента часть 1

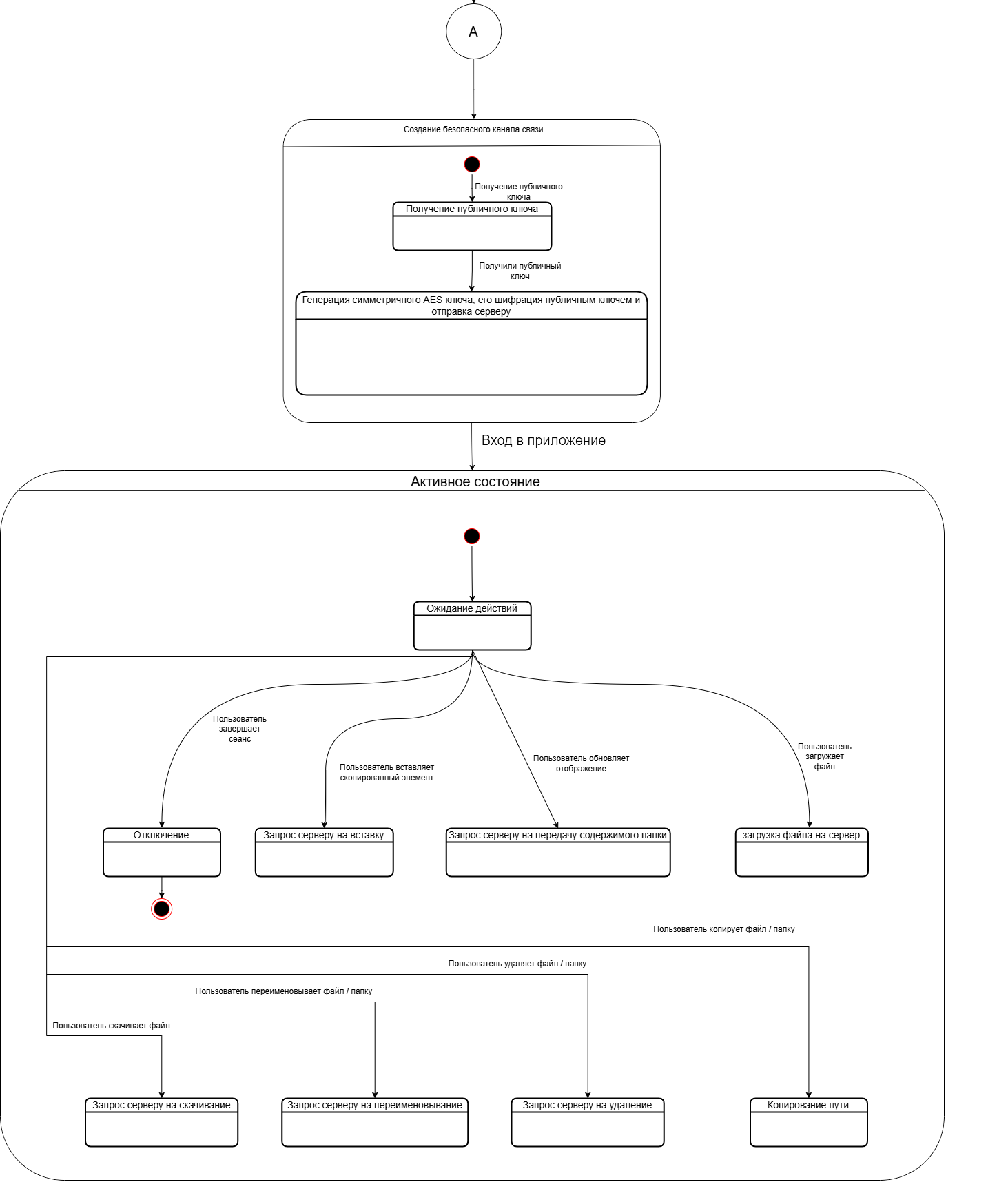


Рисунок 9 - UML-диаграмма состояний клиента часть 2

## **Приложение Б.5**

UML-диаграмма состояний сервера

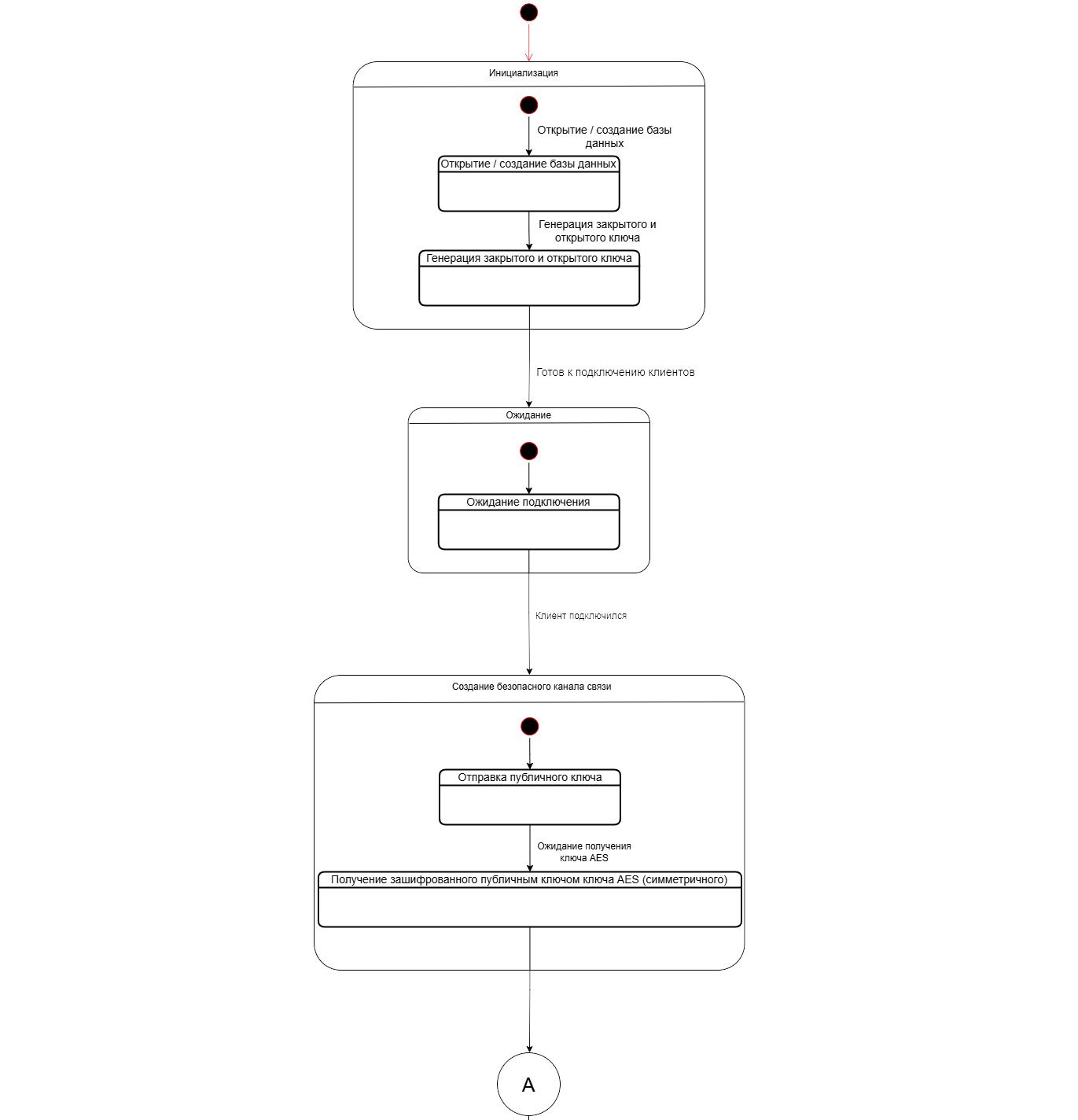


Рисунок 10 - UML-диаграмма состояний сервера часть 1

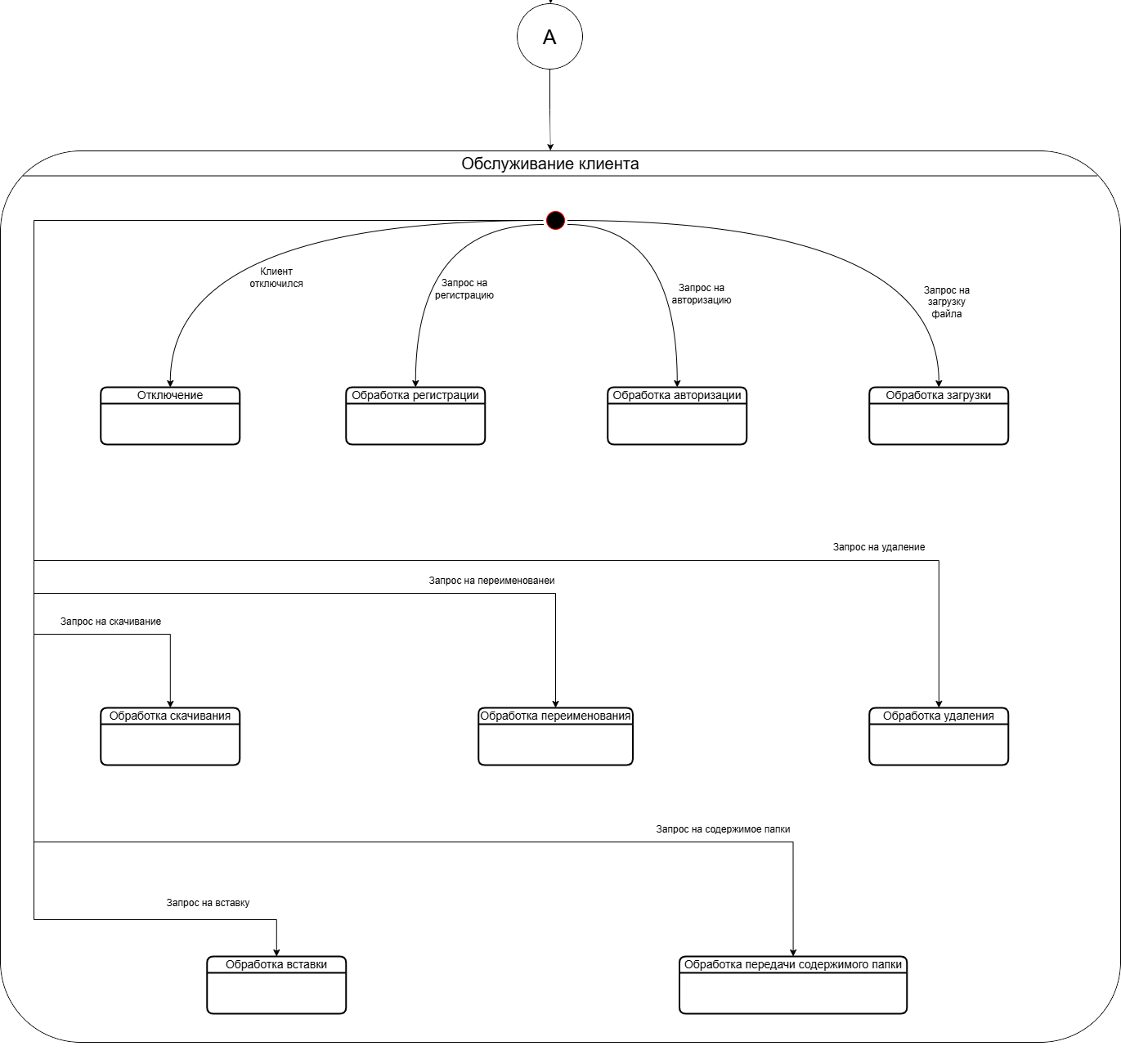


Рисунок 11 - UML-диаграмма состояний сервера часть 2

## **Приложение Б.6**

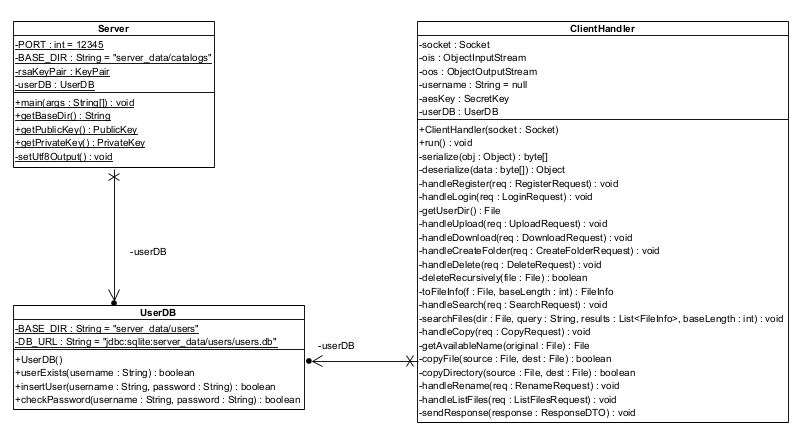
UML-диаграмма классов сервера

Рисунок 12 - UML-диаграмма классов сервера

## **Приложение Б.7**

UML-диаграмма классов клиента

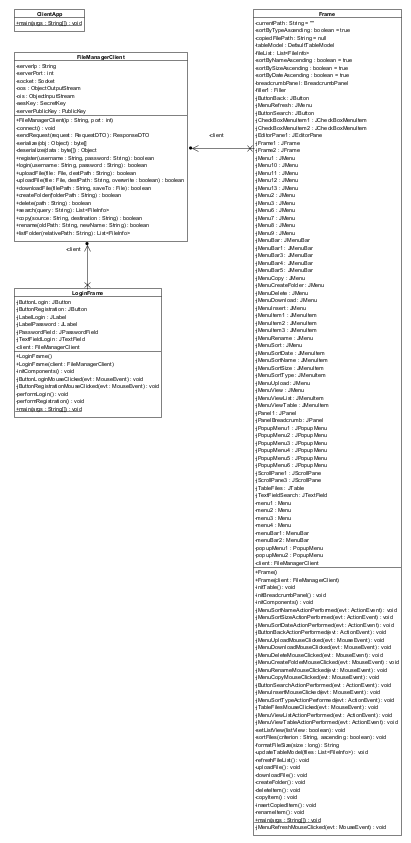


Рисунок 13 - UML-диаграмма классов клиента