МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»

(БГТУ им. В.Г. Шухова)

ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И УПРАВЛЯЮЩИХ СИСТЕМ

КУРСОВОЙ ПРОЕКТ

по дисциплине: Компьютерные сети

тема: «Разработка FTP клиент-серверного приложения»

Автор работы _____ Пахомов Владислав Андреевич ПВ-223

(подпись)

Руководитель проекта _____ Федотов Евгений Александрович

(подпись)

Оценка ____

Белгород 2025 г.

Оглавление

1 Введение

Передача файлов на сегодняшний день является одним из самых важных сценариев использования компьютерной сети. Перечислим лишь несколько из сценариев:

- **Интернет**. Ежедневно пользователи интернета посещают миллиарды сайтов и загружают копии сайтов на портативные и стационарные устройства. В свою очередь сайты могут выполнять большое количество функций, как образовательных так и развлекательных.

- **Домашняя локальная сеть**. Ещё одним сценарием передачи файлов между компьютерами является формирование локальной сети. Например, в доме можно установить NAS - специализированный компьютер, который выполняет функции хранения файлов, разграничения ролей, резервного копирования. Пользователи локальной сети должны иметь доступ как загружать данные на домашний сервер, так и получать данные с домашнего сервера. Файлы могут иметь различный характер, например это может быть домашний фотоархив или видеозаписи, исполняемые файлы, текстовые документы.

- **Нейронные сети**. Ещё одним сценарием использования компьютерных сетей может являться организация работы в высоконагруженных системах. Например, в сегодняшние дни всё больше набирает популярность использование нейронных сетей. Для обучения нейронных сетей требуются большие вычислительные мощности. Но также для таких систем требуются и большое хранилище данных, дело в том что нейронные сети обучаются на большом количестве датасетов - наборов

данных, которые нужно где-то хранить. Для этого и есть компьютерные сети - они позволяют распределить нагрузку и создать сеткомьютеров, которая позволяет эффективно хранить данные и использовать их. В случае нейронных сетей также может появляться дополнительная нагрузка в виде интерпретации данных, так как нейронные сети не могут работать с сырыми данными напрямую, их необходимо закодировать в бинарную последовательность а также представить в памяти в соответствующем виде. И так как сегодня нейронные сети используются в очень большом спектре задач, хранилища должны уметь подстраиваться под выбранный формат данных.

Задача передачи файлов была актуальной и сохраняет свою актуальность по сегодняшний день. В связи с этим было разработано множество протоколов, которые позволяют эффективно организовать передачу файлов.

Цель работы заключается в исследовании популярных протоколов передачи данных, а также создания клиент-серверного приложения для управления файлами на удалённом компьютере на основе протокола FTP.

Задачи

- Исследовать известные протоколы передачи данных, их сильные и слабые стороны
- Разработать сервер
  - Выбрать подходящий стек технологий
  - Выполнить декомпозицию задач
  - Разработать архитектуру приложения
  - Разработать серверное приложение

- Разработать клиент
  - Выбрать подходящий стек технологий
  - Выполнить декомпозицию задач
  - Разработать архитектуру приложения
  - Разработать клиентское приложение
- Выполнить анализ функционала клиент-серверного взаимодействия

2 Анализ протоколов передачи данных

Как уже было сказано ранее, в компьютерных сетях передача файлов - очень популярная задача. В связи с этим существует множество протоколов, которые подходят для решения конкретных специфических задач. Перечислим некоторые из них.

2.1 FTP

FTP - один из самых старых и самых распространённых протоколов. За полвека протокол постоянно изменялся и расширялся под главенством Internet Engineering Task Force.

Впервые протокол FTP появился в 1971 году как один из механизмов передачи файлов в RFC-114 и применялся для обмена файлами в MIT.

В дальнейшем протокол дополнялся и дорабатывался, RFC-294 ввёл возможность изменения типа передаваемых данных. RFC-354 закрепил задачу протокола как протокол для эффективной и надёжной передачи данных между устройствами сети в сети ARPANET, в 1973 с появлением RFC-454 протокол закрепился официально.

Следующим большим скачком для протокола стал переход от основного используемого протокола для обмена данными между устройствами сети от NCP к TCP вместе с RFC-765.

На сегодняшний день актуальным описанием для протокола является RFC-959, это описание в основном вводит некоторые дополнительные команды для более удобной работы с каталогами, команду уникального сохранения файла и получение информации о системе.

В дальнейшем выходили только дополнения к текущему описанию. Так, RFC-2640 вводит кодировку UTF-8, которая используется многими современными компьютерами сегодня. Стандарт RFC-2228 вводит более широкие возможности для обеспечения безопасности при передаче файлов.

Положительные стороны

- Протокол FTP имеет очень широкую поддержку, имеет множество рабочих проверенных временем решений, большинство компьютеров имеют поддержку FTP, в том числе и на системном уровне, то есть нет необходимости устанавливать стороннее программное обеспечение для использования FTP.
- Очень хорошая совместимость со старыми компьютерами. Реализации FTP есть как для старых так и для современных компьютеров. Кроме того, протокол FTP поддерживает очень большое количество способов передачи файлов, что позволит запустить сервер на множестве старых систем, вроде компьтютеров IBM. Такие случаи редки сегодня, однако они позволят администрировать устаревшие системы, которые по сегодняшний день всё ещё являются частью множества бизнес процессов.
- Поддержка многих способов передачи файлов. Протокол может передавать и получать данные как постранично, так и в скомпрессированном виде.
- Относительная простота протокола для пользователя. Изначально FTP разрабатывался как консольное приложение, поэтому множество ответов имеют понятное описание, так же как и команды выражаются относительно просты в освоении и понимании.

Отрицательные стороны

- Большинство команд протокола при условии использования современных систем просто мусорные, вряд ли сегодня

найдётся устройство, которое будет использовать в качестве системной кодировки EBCDIC или обычный ASCII.

- Протокол

  не подходит для автоматизации. В сегодняшнем мире пользователю приятней всего использовать понятный и интуитивный графический интерфейс, однако как уже было сказано ранее, протокол заточен для использования в консоли, из-за чего возникает множество проблем. Например, команда вывода списка файлов в текущей директории платформоспецифична, что вызывает трудности в интерпретации для оформления графического интерфейса в зависимости от того, на какой машине запущен протокол.

- Протокол

  не предназначен для нагруженных систем. Протокол использует два соединения - одно для передачи данных, а второе для передачи команд (порт 21 и 20 соответственно). В нагруженных системах с большим количеством соединение это непозволительная роскошь.

- Безопасность. Протокол в текущем описании никак не шифрует данные, любой может подключиться к сети и прослушать пакеты, из-за чего существует угроза несанкционированного доступа к данным.

- Некое моральное устаревание протокола нивелируется новыми стандартами, дополняющими протокол. Однако заставить пересесть множество уже отлаженных серверов на старых стандартах на новые и неисследованные команды может быть экономически неоправданно и технологически сложно, в том числе и при оптимизации процессов.

Протокол FTP хорошо отлажен и проверен временем, он подходит для консольной работы на старых системах. Однако для построения новых систем рекомендуется использовать более современные протоколы.

2.2 SFTP

На смену протоколу FTP пришёл протокол SFTP, который решает большинство проблем протокола FTP.

Этот протокол был разработан The OpenSSH Project, его описание можно найти в расширении к описанию протокола SSH в RFC-4253 от 2006 года. В этом и первое основное отличие протокола FTP, SFTP работает поверх SSH, в то время как FTP работает на устаревшем протоколе Telnet.

SFTP не является дополнением FTP, хотя и берёт очень много из его функционала. Протокол SFTP нужно рассматривать как совершенно новый переработанный протокол для современных систем.

Протокол SFTP берёт очень много из соединения по SSH, его основной особенностью стала перекочевавшая из SFTP повышенная безопасность, большое количество доступных методов шифрования.

Положительные стороны

- Безопасность. Как уже было сказано ранее, протокол SFTP обеспечивает гораздо большую безопасность, чем FTP, позволяя передавать данные, например, с использованием ассиметричного шифрования.
- Протокол SFTP использует одно соединение, а не два, как в FTP. Это делает работу в высоконагруженных системах более удобной, нежели в FTP, так как для передачи используется только один порт.
- Расширенный функционал. SFTP дополняет возможности FTP, позволяя например получать атрибуты файлов.
- Протокол SFTP не создавался для работы в консоли, ответы сервера не заточены для под конкретную операционную систему, а также предоставляется гораздо большеее количество данных, что позволяет

легко распарсить полученные данные и представить их в графическом виде.

Отрицательные стороны

- Протокол SFTP гораздо сложнее запрограммировать. Расширенный функционал и требования к безопасности ведут к повышенной нагрузке при написании функционала, что может занять большее количество времени у программистов и следовательно большие расходы при реализации и отладке собственного SFTP сервера.
- Из-за
  повышенной сложности, большего количество шагов для подготовки к передаче и кодировании данных возникают дополнительная нагрузка как на клиент, так и на сервер, вследствие чего передача файлов может работать медленнее.

Несмотря на повышенную вычислительную нагрузку а также стоимость разработки, протокол SFTP - это настоящее и будущее для манипуляции файловой системой сервера. В первую очередь, этот протокол обеспечивает высокую безопасность, что позволяет работать с сервером в публичной сети. Протокол FTP пусть и проще в исполнении и быстрее, однако безопасно может использоваться в хорошо изолированных локальных сетях.

2.3 SCP

Протокол SCP был также разработан The OpenSSH Project.

В 2019 коду протокол был признан OpenSSH как устаревший, в последней версии SCP используется SFTP для передачи файлов.

Отличительной способностью SCP является его относительная простота. Он также работает на SSH, однако предлагает гораздо более узкий функционах

Положительные стороны

- Безопасность. Протокол SCP работает на основе SSH, а значит может использовать надёжное ассиметричное шифрование.
- Протокол SCP гораздо менее громоздок по сравнению с SFTP. Он содержит одну единственную команду - скопировать указанный файл или получить его. Из-за этого его гораздо легче запрограммировать, он гораздо меньше нагружает как сервер так и клиент. Узкий спектр задач протокола позволяет решать конкретную задачу более эффективно.

Отрицательные стороны

- Уменьшенный функционал протокола привёл к некоторым проблемам с безопасностью. Так, согласно CVE-2019-6111 SCP-клиент не проверяет пути полученных файлов, и поэтому вредоносный SCP-сервер может перезаписать файлы на клиенте. SFTP протокол может принимать только целые пути, в то время как SCP клиент может выбрать файл по определённому паттерну.

Протокол SCP подходит, если нужно быстро передать файл от одного компьютера на другой. На данный момент этот протокол признан небезопасным и при использовании SCP используется SFTP.

2.4 HTTP/HTTPS

Протокол HTTP, так же как и FTP, прожил очень долгую историю развития, однако в отличие от своего современника, сегодняшний интернет

без HTTP/HTTPS незаменим.

Именно при помощи HTTP сегодня происходит передача большинства файлов в интернете: загружаются миллионы HTML, CSS, JS, JSON, XML, медиафайлов.

HTTP/1.0 впервые был описан в 1997 году в RFC-2068. Он описывал формат передачи данных, заголовки, коды ответов. Также в HTTP/1.0 появилась важная особенность, позволяющая обмениваться данными - появилась возможность задавать MIME-тип ответа и размер ответа, что позволило передавать файлы как от клиента так и получать их от сервера. Также появился в дополнение к методу GET метод POST, который позволял редактировать состояние сервера.

HTTP/1.1 - самый известный на данный момент протокол. Он обеспечил необходимую безопасность при передаче данных, введя опциональное SSL/TLS ассиметричное кодирование данных (протокол HTTPS). Также новый стандарт добавил новые методы взаимодействия.

Положительные стороны

- Безопасность. Протокол поддерживает безопасное соединение с использованием шифрования, что позволяет безопасно передавать данные.
- Простота. В отличие от громоздкой настройки SFTP сервера, с HTTP сервером можно начинать работу без предварительной настройки.
- Большая гибкость. Обмен информации с использованием HTTP/HTTPS очень гибок и зависит исключительно от реализации веб-сервера.
- Широкая поддержка. На современном рынке существует большое количество браузеров, которые умеют эффективно и отлаженно выполнять HTTP запросы, обрабатывать их и использовать мощности

современных устройств максимально эффективно. Также существуют браузеры и для старых систем, которые также могут использовать как безопасный так и небезопасный вариант HTTP. Существует большое количество фрейморков и библиотек, позволяющих создать свой сервер (FastAPI, Flask, Django, Spring, Poem, Express.js)

- Простота в реализации. Веб-сервер гораздо проще написать, чем FTP или SFTP сервер, что позволяет бизнесу создавать бюджетные быстроразвивающиеся решения.

Отрицательные стороны

- Отсутствие сессии.
  Преимущество всех вышеперечисленных протоколов являлось то, что пользователь всегда находился в контексте сессии, например у сессии была рабочая директория и пользователь без дополнительных средств проверки. HTTP лишён такой возможности, из-за чего программистам приходится обеспечивать эту сессию вручную, например при помощи Session-Cookie, JWT-токена и др.

- Необходимость разрыва соединения.
  После успешного получения ответа от сервера соединение немедленно разрывается в HTTP. Открытие нового сокета для общения с сервером требует дополнительных затрат.

HTTP/HTTPS - очень гибкий, простой и, следовательно, дешёвый протокол. Так же как в компьютерных сетях победил не совсем оптимальный но дешёвый Ethernet, в качестве прикладного протокола для обмена файлов сегодня лидирует HTTP.

3 Разработка клиент-серверного приложения

3.1 Разработка сервера

Одним из преимуществ протокола HTTP является его гибкость, возможность создавать любое количество команд и модифицировать их поведение.

Текущим решениям FTP этого не хватает, пусть они и являются надёжными и отлаженными, однако не предоставляют возможности кастомизации, переопределения поведения.

Именно на эти концепции и был сделан акцент при разработке FTP сервера, были позаимствованы некоторые паттерны современных веб-серверов.

Для разделения слоёв приложения было выделено три слоя: CtrlFTP-Core, CtrlFTP-Dependencies и сам сервер. Рассмотрим все три слоя ниже

3.1.1 Выбор стека

В качестве языка программирования был выбран язык программирования Java. Java очень давно находится на рынке, кроме того фреймворк Spring для Java сегодня используется многими компаниями для создания высоконагруженных веб-серверов, то есть Java множество лет использовалась для создания серверов.

В качестве ORM для проекта был выбран Hibernate, так как он является основным инструментом на выбранном языке программирования для манипуляирования базы данных.

В качестве системы сборки был выбран Gradle, так как он хорошо себя зарекомендовал на рынке как система для сборки проектов на Java.

3.1.2 Слой Core

Этот слой позволяет выстроить базовую работу сервера, которая позволит серверу определять команды, которые он должен выполнять, а также зависимости в командах.

Команды

CtrlFTP-Core содержит классы, которые позволяют создать свой сервер. Каждая команда создаётся при помощи аннотации `@Command`, которая находится в пакете `rchat.info.ctrlftp.core.annotations.Command`. Например:

```
public class ServiceService {
    @Command(name = "NOOP")
    public static Response noop() {
        return new Response(ResponseTypes.COMMAND_OK, "Glad to work with ya, human 'fella!");
    }
}
```

Таким образом можно определить FTP-команду NOOP, которая возвращает код OK (200) вместе с сообщением `Glad to work with ya, human 'fella!`. Каждая такая команда должна быть статична, а также должна возвращать класс Response из пакета `rchat.info.ctrlftp.core.responses`

Работа с аннотациями в Java происходит при помощи Reflections API. FTP сервер открывает сокет и ждёт сообщения. Когда это сообщение приходит, формируется новая виртуальная нить, которая обрабатывает запросы пользователя. Когда пользователь отправил сообщение, сопровождающееся переносом строки, сервер просканирует все доступные ему команды и будет искать команду с соответствующим именем. В этом примере, если пользователь пришлёт команду NOOP, сервер проанализирует все классы-контролеры и их методы, найдёт статический метод с именем NOOP (нечувствительно к регистру) и вызовет этот метод.

Метод должен вернуть Response, который сервер должен перенаправить клиенту, выполнившему запрос.

Зависимости

Сейчас метод noop достаточно скучен, он не может получать параметр а также оригинальную команду от пользователя.

Для таких случаев CtrlFTP предоставляет систему Dependency Injection, рассмотрим её на примере.

Каждый метод может принимать следующие зависимости:

- `String` - команда в ”сыром”виде передаётся в метод
- `Session` - класс текущей сессии из пакета `rchat.info.ctrlftp.core`
- `Server` - класс текущего сервера из пакета `rchat.info.ctrlftp.core`
- `AbstractDependency` - зависимость, разработанная самим разработчиком, мы рассмотрим этот механизм позже

Зависимость внедряется в метод когда мы передаём параметр в метод в случае команд, например так:

```
public class ServiceService {
    @Command(name = "NOOP")
    public static Response noop(String command) {
        return new Response(ResponseTypes.COMMAND_OK, "Glad to work with ya, human 'fella!");
    }
}
```

Теперь мы можем получить доступ к ”сырой”команде, которую отправил пользователь на сервер.

Библиотека CtrlFTP также позволяет написать собственные зависимости, которые также могут включать в себя другие зависимости, взаимодействовать с ними. Создать свою зависимость можно при помощи `@AbstractDependency` и аннотации `@Dependency` из пакета `rchat.info.ctrlftp.core.dependencies`. Аннотация принимает в качестве аргумента уровень зависимости, на уровне

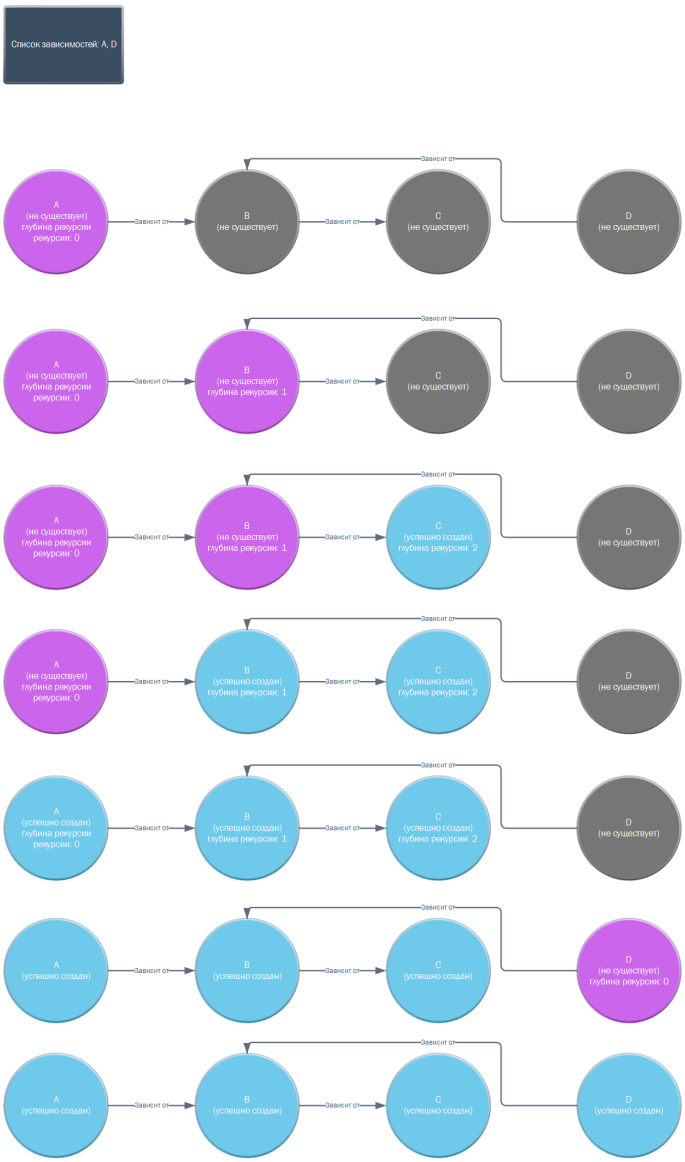
которого она будет существовать как объект. На данный момент доступно три уровня:

- `Global` - зависимость используется всё время существования сервера, уникальна для каждого сервера
- `Session` - зависимость используется всё время существования сессии, уникальна для каждой сессии
- `Command` - зависимость используется всё время существования команды, уникальна для каждой команды

Для создания зависимостей используется публичный конструктор, в котором, как и в команде, зависимости передаются в качестве параметров.

Есть и определённые ограничения, связанные с зависимостями разных уровней. Так, например, зависимость глобального уровня не может получить доступ к зависимости сессионного или командного уровня. В то время как зависимость командного уровня может обратиться к зависимости глобального уровня. Также нельзя ссылаться на саму себя, рекурсивные зависимости также недопустимы.

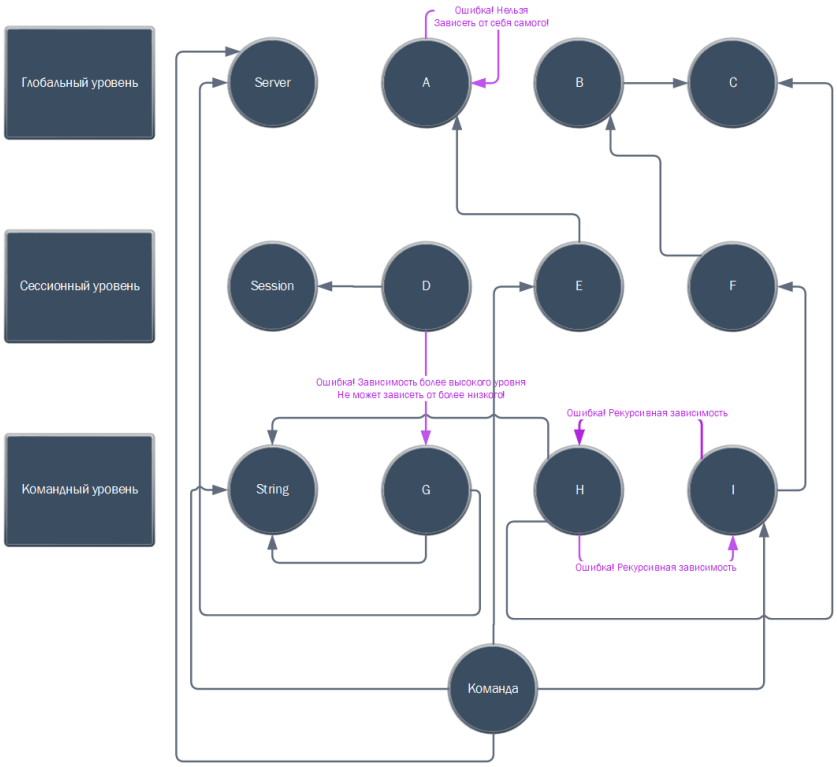
Текущие ограничения связаны также с работой Reflections API. Специальный класс `DependencyManager`, который существует на трёх уровнях - сессии, сервера и команды, выполняет функции внедрения необходимых зависимостей в команду и в другие зависимости. При поступлении списка аргументов менеджер зависимостей менеджер сначала проверяет, какой уровень у переданной зависимости. Если это зависимость его уровня, он проходит по списку всех своих зависимостей и ”запоминает”найденную зависимость. Иначе он пытается рекуррентно создать зависимость. То есть, берёт параметры из конструктора зависимости, снова вызывает метод, который ищет эти зависимости, конструирует объект-зависимость и сохраняет у себя, чтобы в дальнейшем использовать её.



Выбранный подход может привести к бесконечной рекурсии, поэтому метод отслеживает глубину своей рекурсии и выбрсывает исключение, если она слишком большая. Также при попытке сконструировать самого себя можно попасть в бесконечную рекурсию.

Если же зависимость находится уровнем выше, чем текущая зависимость, то менеджер зависимостей просит у родительского менеджера зависимостей найти эту зависимость. Если же запрашиваемая зависимость находится на нижнем уровне, метод выбросит исключение.

Менеджер зависимостей воспринимает класс строки как зависимость командного уровня, сессию как зависимость сессионного уровня и сервер как зависимость глобального уровня.



	String	Session	Server	Уровень команды	Уровень сессии	Уровень глобальный
Уровень команды	+	+	+	+	+	+
Уровень сессии	-	+	+	-	+	+
Уровень глобальный	-	-	+	-	-	+
Команда	+	+	+	+	+	+

Можно заметить, что сейчас наследование от AbstractDependency бесполезно - это пустой класс, ничего не делающий. Однако в будущем возможно он будет использован для пост- и предпроцессинга ввода/вывода.

Пример зависимости:

```
package rchat.info.ctrlftp.dependencies.deserializer;

/**
 * A dependency to parse single string after a command name
 */
```

```
public class SingleStringDeserializer extends BaseDeserializer<SingleStringDeserialized> {
    public SingleStringDeserializer(String command) {
        super(command);
    }

    @Override
    public SingleStringDeserialized deserialize(String command) {
        var firstSpaceIndex = command.indexOf(' ');

        return new SingleStringDeserialized(firstSpaceIndex == -1 ? "" :
↪   command.substring(firstSpaceIndex + 1));
    }
}
```

Эта зависимость принимает команду и оставляет только её аргумент, иногда очень полезная зависимость, если нужно извлечь только аргумент.

Запуск сервера

Сервер можно запустить при помощи конструктора, который принимает конфигурационные файлы

```
package rchat.info.ctrlftp.examplebasic;

import rchat.info.ctrlftp.core.Server;

import java.io.IOException;
import java.util.List;

public class Main {
    public static void main(String[] args) throws ClassNotFoundException, IOException {
        var u = new Server(List.of("dependencies.xml"));
        u.mainLoop();
    }
}
```

Путь к конфигурационному файлу - это имя ресурса Java. Он представляет из себя XML-файл, который декларирует контролеры (сервисы) и зависимости для сервера. Именно в указанных классах сервер будет искать методы-команды и зависимости. Он может выглядеть так

```
<?xml version="1.0" encoding="utf-8"?>
<config>
    <services>
        <service>path.to.class.with.commands.ServiceClass</service>
        ...
    </services>
    <dependencies>
        <dependency>path.to.dependency.class.DependencyClass</dependency>
        ...
    </dependencies>
</config>
```

Файл конфигурации содержит корень `<config>`, который содержит в себе список сервисов `<services>` и список зависимостей `<dependencies>`.

Реализация представлена в репозитории и *Приложении 1*.

3.1.3 Слой Dependencies

Пакет `rchat.info.ctrlftp.dependencies` предоставляет зависимости, которые могут быть полезны при разработке FTP сервера.

Аутентификация

Класс `BaseAuthenticationDependency` позволяет проверить, аутентифицирован ли вошедший пользователь и позволяет ему выйти. Абстракция не включает в себя ввод логина и пароля, так как разработчик может захотеть позволить анонимный доступ к файловой системе. Метод `authenticate` возвращает `AuthenticationResult`, который содержит информацию о том, аутентифицирован ли пользователь, информацию об аутентификации и ошибку, если она произошла при проверке пользователя.

Десериализатор

Класс-десериализатор предназначен для расшифровки пользовательского ввода, парсинга команд. Класс `BaseDeserializer` расшифровывает полученную команду. Реализация расшифровки предоставлена разработчику. Есть также подготовленный заранее `SingleStringDeserializer`, который предназначен для частых случаев расшифровки единственного аргумента после названия

команды.

Передача файлов

Передача файлов является одним из ключевых функционалов работы FTP сервера. Класс `AcceptTransferDependency` - это абстрактная зависимость, которая может помочь разработчику в реализации функционала для передачи/получения файлов. Сам класс позволяет установить пассивный/активный режим `setPassive`, порт для клиента `setClientAddress`. Можно также прервать текущую передачу при помощи `disconnect`, метод `accept` используется для принятия файлов, `send` - передачи файлов. Передача и приём выполняется асинхронно, то есть после вызова этих двух методов код не приостановится, в исходной функции-команде можно, например, вернуть сообщение с кодом 150. Методы приёма и передачи принимают функцию-колбек, которая вызывается при ошибке или успешном принятии/отправке файла. В качестве зависимости используется класс, который наследуется от `AcceptTransferDependency`.

Важно отметить, что `AcceptTransferDependency` занимается в основном установлением соединения с клиентом. Обработкой входных/выходных данных занимается другая зависимость, которая наследуется от `BasePipeDependency`.

Эта абстрактная зависимость содержит два метода `pipeClientInputToDataClass` и `pipeDataClassToClient`, которые сериализует и десериализует данные при отправке/получении. Реализация может, например использоваться для установления кодировки данных, режима сжатия данных, кодирования.

Реализация представлена в репозитории и *Приложении 2*.

3.1.4 Реализация сервера

С применением указанного функционала мы можем относительно легко подготовить сервер. Для данной лабораторной работы был реализован сервер для Linux ОС с аутентификацией через базу данных Postgres.

Полученный сервер, если введённый пользователь не существует, создаёт пользователя с захешированным паролем в базе данных. Если пароль подходит и логин есть, аутентифицирует пользователя, иначе - запрещает доступ пользователю.

Так как мы работаем с файловой системой, в реализации сервера была дополнительно реализована зависимость сессионного уровня `NavigationDependency`, которая позволяет сохраняет текущую позицию сессии (рабочую директорию), получает файлы, удаляет файлы и директории и другие полезные методы.

Реализация представлена в репозитории и *Приложении 3*.

3.2 Разработка клиента

Так как сервер разрабатывался совместимым с существующими FTP клиентами согласно RFC-959, прдеставленный клиент также совместим с серверами, которые поддерживают самый распространённый стандарт FTP RFC-959.

3.2.1 Выбор стека

Для разработки десктопного приложения была выбрана библиотека Qt 6 как одна из самых популярных и удобных библиотек для быстрой разработки графических интерфейсов.

В качестве языка программирования был выбран Python 3.12, он содержит библиотеку `PySide6`, которая подвязывается к библиотекам Qt 6 и позволяет быстро разрабатывать GUI.

Одна из совместимых клиентских FTP библиотек, которая позволяет отправлять кастомные команды на FTP сервер, встроенная в сам язык программирования Python 3, это `ftplib`.

3.2.2 Предварительная настройка среды

В проекте находится скрипт `setup.sh`. Он выполняет несколько задач.

Во-первых, скрипт устанавливает необходимые зависимости проекта, которые находятся в `requirments.txt`, которые нужны для запуска клиента.

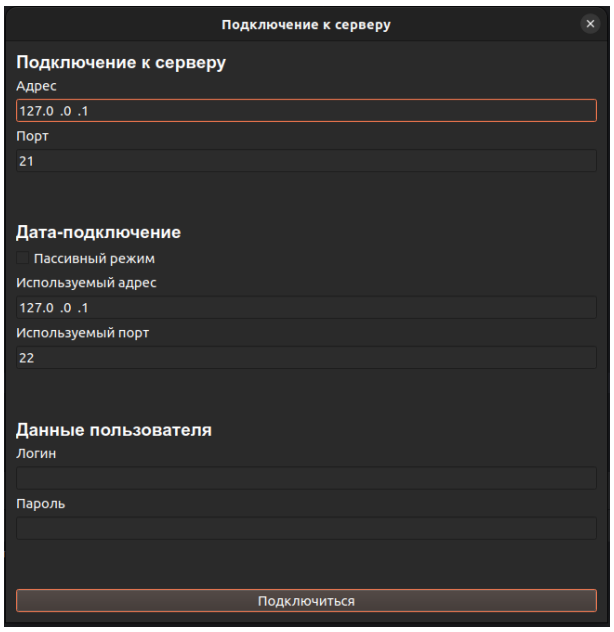
Во-вторых, необходимо скомпилировать файл ресурсный файл в бинарный файл. Файл `resources.qrc` содержит пути к файлам, после чего бинарный файл включается там где это необходимо. Скомпилированный бинарный файл содержит все необходимые вызовы оригинальных функций, которые загружают файлы в Qt.

В-третьих, в проекте используются .ui файлы, которые позволяют описать GUI в XML-формате. Их необходимо преобразовать в декларативный формат описания интерфейса.

3.2.3 Разработанные диалоги

ConnectDialog

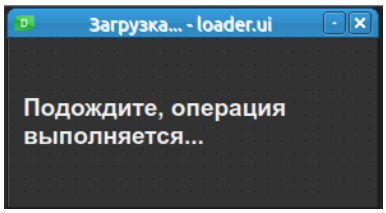
Перед подключением к клиенту необходимо сначала получить параметры пользователя. Эту задачу выполняет диалог `ConnectDialog`. В нём можно ввести IP-адрес и порт сервера, пассивный/активный режим для пользователя а также логин/пароль пользователя.



После получения данных можно подключаться к FTP серверу

LoaderDialog

Этот утилитный диалог принимает функцию-колбэк, который содержит ”тяжёлую”задачу, например передачу очень большого файла. Этот диалог будет особо полезен в случае FTP клиента.



ExplorerDialog

Проводник используется для отображения файлов и состояния текущей сессии. Диалог предлагает возможности изменения директории, скачивания и загрузки файлов, удаления папок/директорий, переименования файлов и папок, создания директории.

Класс проводника содержит несколько полезных функций:

- `update_list` получает текущий лист и директорию, обновляет таблицу.
- `change_directory` обновляет директорию. Принимает опциональный аргумент на изменение текущей директории. Если новое место не

указано, берёт текущую директорию из выше указанной строки. Иначе - переданный аргумент.

- `cell_double_clicked` метод, вызываемый при дабл клике по ячейке. Если выполнен дабл клик по файлу, вызывает метод сохранения файла `save_file`. Иначе - переходит в новую директорию при помощи `change_directory`.

- `button_download_clicked` метод, вызываемый по нажатию на кнопки загрузки. Вызывает `save_file`, если выбран файл в списке проводника.

- `save_file` вызывает системный диалог для сохранения файла, выполняет запрос на файл на сервер и сохраняет в выбранный файл.

- `button_upload_clicked` вызывает системный диалог для открытия файла, выполняет запрос на сохранения файла на сервер.

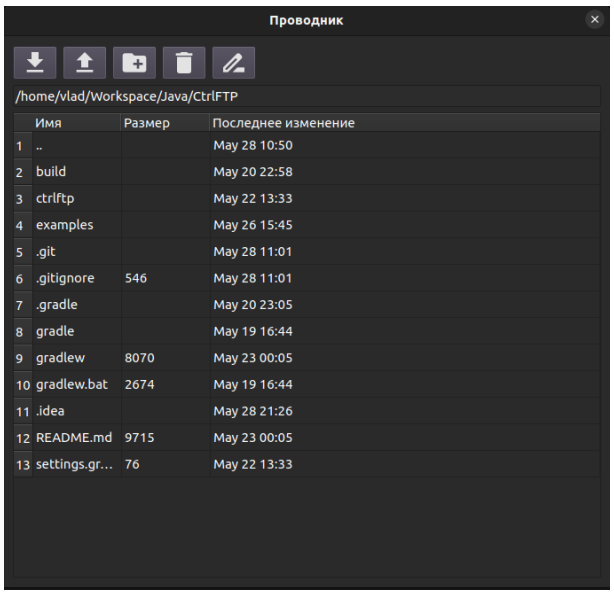
- `button_folder_created_clicked` вызывает диалог для создания папки и отправляет команду для создания папки на сервер.

- `button_delete_clicked` удаляет файл при помощи `delete_file`, иначе, если выбрана папка - `delete_folder`.

- `delete_file` отправляет на сервер команду удаления файла.

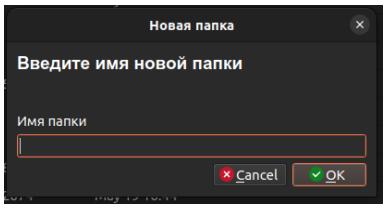
- `delete_folder` отправляет на сервер команду удаления папки.

- `button_rename_clicked` вызывает окно переименования, после чего выполняет команду переименования.



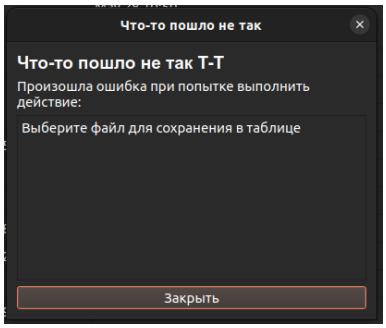
CreateDialog

Простой диалог с вводом для создания папки.



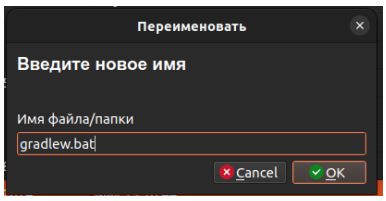
ErrorDialog

Утилитный диалог для отображения ошибки. Может иметь несколько целей как для отображения ошибок с сервера, так и ошибок клиента.



RenameDialog

Простой диалог с вводом для нового имени файла или папки. Принимает опциональное старое имя папки или файла.



Реализация представлена в репозитории и *Приложении 4*.

4 Вывод о проделанной работе

Протокол FTP, пусть и не является на сегодняшний день рекомендованным и безопасным вариантом передачи данных, всё ещё может использоваться в некоторых специфических сценариях, вроде работы с устаревшими системами а также для более быстрой передачи данных в локальных малонагруженных сетях.

Компьютерные сети развивались стремительно, особенно в конце XX века, и это можно заметить по ходу развития протокола передачи данных. Он дополнялся и расширялся, многие вводили предложения и экспериментировали, чтобы на сегодняшний день мы имели эффективный безопасный способ обмена данными, позволяющий нам выполнять множество задач. На сегодняшний день технологии передачи файлов замедлили своё развитие в сфере передачи файлов из файловой системы в файловую систему компьютеров. Для решения этой задачи уже был найден подходящий безопасный и рабочий протокол SFTP. Однако с решением одних задач приходят новые проблемы, и задача современных разработчиков - разрешение этих проблем. Это возможно с использованием лучшего от старых подходов и применения новых идей для решения новых задач.

Разработанная библиотека CtrlFTP позволяет создавать гибкий расширяемый FTP сервер, который может быть полезен для решения специфических задач бизнеса, вроде расширенной поддержки форматов и кодировок, файловой интерпретации нефайловых данных (записи в базе данных).

Получившийся клиент позволяет работать с RFC-959 совместимыми серверами, удалять и копировать файлы между компьютерами, создавать папки, переименовывать. Он также допускает работу в пассивном и активном режимах.

Ссылка на репозиторий с кодом: https://github.com/IAmProgrammist/CtrlFTP

5 Список источников и литературы

1. A File Transfer Protocol [Электронный ресурс]
   Режим доступа: https://datatracker.ietf.org/doc/html/rfc114
2. The Use of ’Set Data Type’ Transaction in File Transfer Protocol [Электронный ресурс]
   Режим доступа: https://datatracker.ietf.org/doc/html/rfc294
3. The File Transfer Protocol [Электронный ресурс]
   Режим доступа: https://datatracker.ietf.org/doc/html/rfc354
4. File Transfer Protocol [Электронный ресурс]
   Режим доступа: https://datatracker.ietf.org/doc/html/rfc454
5. File Transfer Protocol (FTP) [Электронный ресурс]
   Режим доступа: https://datatracker.ietf.org/doc/html/rfc959
6. Internationalization of the File Transfer Protocol [Электронный ресурс]
   Режим доступа: https://datatracker.ietf.org/doc/html/rfc2640
7. FTP Security Extensions [Электронный ресурс]
   Режим доступа: https://datatracker.ietf.org/doc/html/rfc2228
8. The Secure Shell (SSH) Transport Layer Protocol [Электронный ресурс]
   Режим доступа: https://datatracker.ietf.org/doc/html/rfc4253
9. Hypertext Transfer Protocol – HTTP/1.1 [Электронный ресурс]
   Режим доступа: https://datatracker.ietf.org/doc/html/rfc2068
10. Hypertext Transfer Protocol – HTTP/1.1 [Электронный ресурс]
    Режим доступа: https://datatracker.ietf.org/doc/html/rfc2616
11. Documentation | Adoptium [Электронный ресурс]
    Режим доступа: https://adoptium.net/docs/

12. Getting started with Hibernate ORM [Электронный ресурс] Режим доступа: https://hibernate.org/orm/documentation/getting-started/5.x/

13. Qt for Python
Режим доступа: https://doc.qt.io/qtforpython-6/index.html

14. ftplib — FTP protocol client
Режим доступа: https://docs.python.org/3/library/ftplib.html

15. SFTP vs. FTP: Understanding the Difference
Режим доступа: https://www.sharetru.com/blog/sftp-vs-ftp-understanding-the-difference

16. Telnet Protocol Specification
Режим доступа: https://datatracker.ietf.org/doc/html/rfc854

6 Приложения

6.1 Приложение 1. Исходный код библиотеки CtrlFTP Core

ctrlftp/ctrlftp-core/src/main/java/rchat/info/ctrlftp/core/annotations/Command.java

```java
package rchat.info.ctrlftp.core.annotations;

import java.lang.annotation.*;

/**
 * A method annotated with {@link Command} would be called when a command with a
 * corresponding {@link Command#name} is sent. The namecheck is case-insensitive.
 */
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.METHOD)
@Inherited
public @interface Command {
    /**
     * A command name
     */
    public String name();
}
```

ctrlftp/ctrlftp-core/src/main/java/rchat/info/ctrlftp/core/annotations/Dependency.java

```java
package rchat.info.ctrlftp.core.annotations;

import rchat.info.ctrlftp.core.dependencies.DependencyLevel;

import java.lang.annotation.*;

/**
 * A class annotated with {@link Dependency} will be able to be injected into
 * commands annotated by {@link Command} and dependency methods.
 * Level of dependency existence is passed using parameter {@link Dependency#level}
 */
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.TYPE)
@Inherited
public @interface Dependency {
    /**
     * Sets the level of dependency. Defaults to {@link DependencyLevel#SESSION}
     */
    public DependencyLevel level() default DependencyLevel.SESSION;
}
```

ctrlftp/ctrlftp-core/src/main/java/rchat/info/ctrlftp/core/dependencies/AbstractDependency.java

```java
package rchat.info.ctrlftp.core.dependencies;

public abstract class AbstractDependency {
}
```

ctrlftp/ctrlftp-core/src/main/java/rchat/info/ctrlftp/core/dependencies/DependencyLevel.java

```java
package rchat.info.ctrlftp.core.dependencies;

public enum DependencyLevel {
    /**
     * A dependency will be unique for server and will exist while server runs
     */
    GLOBAL,
    /**
     * A dependency will be unique for session and will exist while session exists
     */
    SESSION,
    /**
     * A dependency will be unique for command and will exist while command executes
     */
    COMMAND
}
```

ctrlftp/ctrlftp-core/src/main/java/rchat/info/ctrlftp/core/dependencies/DependencyManager.java

```java
package rchat.info.ctrlftp.core.dependencies;

import rchat.info.ctrlftp.core.Server;
import rchat.info.ctrlftp.core.Session;
import rchat.info.ctrlftp.core.annotations.Dependency;

import java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Parameter;
import java.util.ArrayList;
import java.util.List;

public class DependencyManager {
    private final DependencyLevel level;
    private DependencyManager parentResolver = null;
    private List<AbstractDependency> dependencies;
    private final String command;
    private final Server serverContext;
    private final Session sessionContext;
```

```
    /**
     * This constructor may be used for creating global level dependency resolver.
     * A parent is not specified for this level
     *
     * @param context a server context
     */
    public DependencyManager(Server context) {
        this.level = DependencyLevel.GLOBAL;
        this.dependencies = new ArrayList<>();
        this.command = null;
        this.serverContext = context;
        this.sessionContext = null;

        linkDependencies(context);
    }

    /**
     * This constructor is used for creating any intermediate-leveled dependency resolver.
     * Lookup {@link DependencyManager#DependencyManager(Server)}  DependencyResolver} to
     * create global-leveled dependency resolver and
     * {@link DependencyManager#DependencyManager(Server, String, DependencyManager)}
↪   DependencyResolver} for
     * creating command-leveled dependency resolver
     *
     * @param context        a server context
     * @param level          a level
     * @param parentResolver a parent-level resolver
     */
    public DependencyManager(Server context, Session sessionContext, DependencyLevel level,
↪   DependencyManager parentResolver) {
        this.level = level;
        this.dependencies = new ArrayList<>();
        this.parentResolver = parentResolver;
        this.command = null;
        this.serverContext = context;
        this.sessionContext = sessionContext;
        if (areParentAndChildLevelsNotCompatible()) {
            throw new RuntimeException(
                    "A parent dependency resolver shouldn't have upper level. " +
                            "Parent level: " + this.parentResolver.level.name() + " (" +
↪   this.parentResolver.level.ordinal() + ") " +
                            "current level: " + this.level.name() + " (" + this.level.ordinal() + ")");
        }

        linkDependencies(context);
    }

    /**
     * A dependency resolver could be created for a command scope. The difference for this resolver
```

```
     * is that command string should be passed and processed by dependencies.
     *
     * @param context        a server context
     * @param command        a raw command coming from client
     * @param parentResolver a parent-level resolver
     */
    public DependencyManager(Server context, Session sessionContext, String command, DependencyManager
↪   parentResolver) {
        this.level = DependencyLevel.COMMAND;
        this.dependencies = new ArrayList<>();
        this.parentResolver = parentResolver;
        this.command = command;
        this.serverContext = context;
        this.sessionContext = sessionContext;
        if (areParentAndChildLevelsNotCompatible()) {
            throw new RuntimeException(
                    "A parent dependency resolver shouldn't have upper level. " +
                            "Parent level: " + this.parentResolver.level.name() + " (" +
↪   this.parentResolver.level.ordinal() + ") " +
                            "current level: " + this.level.name() + " (" + this.level.ordinal() + ")");
        }

        linkDependencies(context);
    }

    /**
     * Checks if parent and current resolvers levels are compatible
     *
     * @return true if they are not compatible. Amogus tun dun dun dun dun dun dun
     * dududun. PUM BUM. Tun-dun-dun-dun-dun-dun-DUN! DU-du-du-DU-du-du-DUN!
     */
    private boolean areParentAndChildLevelsNotCompatible() {
        return this.parentResolver.level.ordinal() >= this.level.ordinal();
    }

    /**
     * Finds dependency classes with level that equals to dependency resolver level
     *
     * @param context a server context
     * @return a list of suitable classes
     */
    private List<Class<? extends AbstractDependency>> getDependenciesForCurrentResolver(Server context) {
        return context.getDependencyClasses().stream()
                .filter(aClass -> {
                    var annotation = aClass.getAnnotation(Dependency.class);
                    if (annotation == null) return false;

                    return annotation.level() == this.level;
                })
                .toList();
```

```
    }

    /**
     * Searches for dependencies in dependency manager and paren dependency managers
     *
     * @param dependencyClass a class to search for
     * @return dependency with fitting class
     */
    protected AbstractDependency getDependency(Class<?> dependencyClass) {
        for (var dependency : this.dependencies) {
            if (dependency.getClass().equals(dependencyClass)) {
                return dependency;
            }
        }

        return parentResolver != null ? parentResolver.getDependency(dependencyClass) : null;
    }

    /**
     * Finds objects for parameters
     *
     * @param parameterList a list of parameters
     * @return a list of objects: Strings and AbstractDependencies to be injected
     */
    public List<Object> getDependenciesForParameters(Parameter[] parameterList) {
        return getDependenciesForParameters(parameterList, 0);
    }

    /**
     * Finds objects for parameters
     *
     * @param parameterList a list of parameters
     * @param depth         a recursive depth
     * @return a list of objects: Strings and AbstractDependencies to be injected
     */
    private List<Object> getDependenciesForParameters(Parameter[] parameterList, int depth) {
        var constructorArgs = new ArrayList<Object>();

        for (var parameter : parameterList) {
            if (parameter.getType().equals(String.class)) {
                if (level == DependencyLevel.COMMAND) {
                    constructorArgs.add(command);
                } else {
                    throw new RuntimeException("A dependency " + parameter.getClass() + " with not
↪   command-level " +
                            level + " can't have a String as a parameter");
                }
            } else if (Server.class.isAssignableFrom(parameter.getType())) {
                constructorArgs.add(this.serverContext);
            } else if (Session.class.isAssignableFrom(parameter.getType())) {
```

```
                if (sessionContext != null) {
                    constructorArgs.add(this.sessionContext);
                } else {
                    throw new RuntimeException("A session context is not available now");
                }
            } else {
                var dependencyClass = parameter.getType();

                if (AbstractDependency.class.isAssignableFrom(dependencyClass)) {
                    var dependencyClassAnnotation = dependencyClass.getAnnotation(Dependency.class);

                    if (dependencyClassAnnotation == null) {
                        throw new RuntimeException("A @Dependency annotation is missing on " +
↪   dependencyClass);
                    } else if (dependencyClassAnnotation.level().ordinal() > level.ordinal()) {
                        throw new RuntimeException("You can't require upper-level dependencies from
↪   lower-level dependencies");
                    } else {
                        var dependencyObject = getDependency(dependencyClass);

                        if (dependencyObject == null && dependencyClassAnnotation.level().ordinal() ==
↪   level.ordinal()) {
                            this.dependencies.add((AbstractDependency)
↪   constructDependency(dependencyClass, depth + 1));

                            dependencyObject = getDependency(dependencyClass);

                            if (dependencyObject == null) {
                                throw new RuntimeException("Couldn't inject dependency");
                            }
                        }

                        constructorArgs.add(dependencyObject);
                    }
                } else {
                    throw new RuntimeException("A dependency should be inherited from a base class
↪   AbstractDependency and " +
                            "contain @Dependency annotation " + dependencyClass);
                }
            }
        }

        return constructorArgs;
    }

    /**
     * Constructs dependencies and searches for sub-dependencies
     *
     * @param classToInject a dependency class to inject
     * @param depth         a recursive depth
```

```
     * @return a constructed object
     */
    private Object constructDependency(Class<?> classToInject, int depth) {
        if (depth >= 512) {
            throw new RuntimeException("Injection depth exceeded limits (512 iterations), " +
                    "probably you have recursive dependencies");
        }

        if (!AbstractDependency.class.isAssignableFrom(classToInject)) {
            throw new RuntimeException("A dependency should be inherited from AbstractDependency " +
↪   classToInject);
        }

        if (classToInject.getConstructors().length != 1) {
            throw new RuntimeException("A dependency should contain single constructor which parameters
↪   should " +
                    "be inherited from AbstractDependency or be a String object for DependencyLevel
↪   command");
        }

        var constructor = classToInject.getConstructors()[0];

        try {
            return constructor.newInstance(getDependenciesForParameters(constructor.getParameters(),
↪   depth).toArray());
        } catch (InstantiationException | IllegalAccessException | InvocationTargetException e) {
            throw new RuntimeException(e);
        }
    }

    /**
     * Initializes dependencies and executes dependency injection for dependency
     *
     * @param context a server context
     */
    private void linkDependencies(Server context) {
        this.dependencies.clear();

        for (var dependency : getDependenciesForCurrentResolver(context)) {
            if (getDependency(dependency) == null) {
                this.dependencies.add((AbstractDependency) constructDependency(dependency, 0));
            }
        }
    }
}
```

ctrlftp/ctrlftp-core/src/main/java/rchat/info/ctrlftp/core/reflections/ClassesLoader.java

```java
package rchat.info.ctrlftp.core.reflections;

import org.w3c.dom.Document;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;
import rchat.info.ctrlftp.core.dependencies.AbstractDependency;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import java.io.IOException;
import java.io.InputStream;
import java.util.List;
import java.util.Set;

/**
 * A utility class which purpose is to load classes for services and dependencies
 */
public class ClassesLoader {
    public static void loadDependencies(Set<Class<? extends AbstractDependency>> dependencyClasses,
                                        NodeList nodes) throws ClassNotFoundException {
        for (int i = 0; i < nodes.getLength(); i++) {
            if (!nodes.item(i).getNodeName().equals("dependency"))
                continue;
            var loadedClass = Class.forName(nodes.item(i).getChildNodes().item(0).getNodeValue());
            if (!AbstractDependency.class.isAssignableFrom(loadedClass))
                throw new ClassNotFoundException("Class " + loadedClass + " should be inherited from
↪  AbstractDependency");

            dependencyClasses.add((Class<? extends AbstractDependency>) loadedClass);
        }
    }

    public static void loadServices(Set<Class<?>> serviceClasses,
                                    NodeList nodes) throws ClassNotFoundException {
        for (int i = 0; i < nodes.getLength(); i++) {
            if (!nodes.item(i).getNodeName().equals("service"))
                continue;
            var loadedClass = Class.forName(nodes.item(i).getChildNodes().item(0).getNodeValue());

            serviceClasses.add(loadedClass);
        }
    }

    /**
     * Utility class to load services and dependencies from config files
     * @param serviceClasses a link for a set of service classes
     * @param dependencyClasses a link for a set of dependency classes
     * @param configFiles array of config files
     */
```

```
    public static void load(Set<Class<?>> serviceClasses, Set<Class<? extends AbstractDependency>>
↪   dependencyClasses,
                            List<String> configFiles) {
        for (var configFilePath : configFiles) {
            try {
                InputStream in =
↪   ClassesLoader.class.getClassLoader().getResourceAsStream(configFilePath);
                DocumentBuilderFactory docBuilderFactory = DocumentBuilderFactory.newInstance();
                DocumentBuilder docBuilder = docBuilderFactory.newDocumentBuilder();
                Document doc = docBuilder.parse(in);

                var root = doc.getDocumentElement();
                var nodes = root.getChildNodes();
                for (int i = 0; i < nodes.getLength(); i++) {
                    if (nodes.item(i).getNodeName().equalsIgnoreCase("dependencies")) {
                        loadDependencies(dependencyClasses, nodes.item(i).getChildNodes());
                    } else if (nodes.item(i).getNodeName().equalsIgnoreCase("services")) {
                        loadServices(serviceClasses, nodes.item(i).getChildNodes());
                    }
                }
            } catch (ParserConfigurationException | IOException | SAXException | ClassNotFoundException
↪   e) {
                throw new RuntimeException(e);
            }
        }
    }
}
```

ctrlftp/ctrlftp-core/src/main/java/rchat/info/ctrlftp/core/reflections/MethodResolver.java

```
package rchat.info.ctrlftp.core.reflections;

import rchat.info.ctrlftp.core.Server;
import rchat.info.ctrlftp.core.annotations.Command;

import java.lang.reflect.Method;
import java.util.Optional;

/**
 * An utility class to find methods by command name
 */
public class MethodResolver {
    /**
     * A method for finding corresponding method to process command
     *
     * @param serverContext a server context
     * @param name a name of a command
     * @return method to be called
```

```
     */
    public static Optional<Method> findMethod(Server serverContext, String name) {
        var services = serverContext.getServiceClasses();

        for (var serviceClass : services) {
            for (var serviceMethod : serviceClass.getMethods()) {
                Command c = serviceMethod.getAnnotation(Command.class);

                if (c != null && name.equalsIgnoreCase(c.name())) return Optional.of(serviceMethod);
            }
        }

        return Optional.empty();
    }
}
```

ctrlftp/ctrlftp-core/src/main/java/rchat/info/ctrlftp/core/responses/Response.java

```
package rchat.info.ctrlftp.core.responses;

/**
 * A Response class to send back to user some data
 */
public class Response {
    private ResponseTypes type;
    private String message = "Message not provided";

    /**
     * A constructor of {@link Response}. You can use {@link Response#Response(ResponseTypes)}
     * to use default message for selected type
     *
     * @param type a return type
     * @param message a custom message
     */
    public Response(ResponseTypes type, String message) {
        this.type = type;
        this.message = message;
    }

    /**
     * A constructor of {@link Response}, a message is selected for specified
     * type automatically
     *
     * @param type a return type
     */
    public Response(ResponseTypes type) {
        this.type = type;
    }
```

```
    /**
     * A method to print by socket connection to user
     *
     * @return a server response in a form of string to by passed by Telnet
     */
    public StringBuilder serialize() {
        return new StringBuilder(
                type.code + (message != null ?
                        (" " + message.replaceAll("\r{0,1}\n", " "))
                        : "Message not provided")
                        + "\r\n");
    }
}
```

ctrlftp/ctrlftp-core/src/main/java/rchat/info/ctrlftp/core/responses/ResponseTypes.java

```
package rchat.info.ctrlftp.core.responses;

public enum ResponseTypes {
    RESTART_MARKER(110),
    SERVICE_READY_IN_N_MINUTES(120),
    DATA_CONNECTION_ALREADY_OPEN(125),
    ABOUT_TO_OPEN_CONNECTION(150),

    COMMAND_OK(200),
    NOT_IMPLEMENTED_SUPERFLUOUS(202),
    SYSTEM_STATUS(211),
    DIRECTORY_STATUS(212),
    FILE_STATUS(213),
    HELP_MESSAGE(214),
    SYSTEM_TYPE(215),
    SERVICE_READY_FOR_NEW_USER(220),
    SERVICE_CLOSING_CONTROL_CONNECTION(221),
    DATA_CONNECTION_OPEN(225),
    CLOSING_DATA_CONNECTION(226),
    ENTERING_PASSIVE_MODE(227),
    AUTH_SUCCESS(230),
    FILE_ACTION_OK(250),
    PATHNAME_CREATED(257),

    USERNAME_OK_NEED_PASS(331),
    NEED_ACCOUNT_TO_LOGIN(332),
    FILE_ACTION_PENDING_INFO(350),

    SERVICE_NOT_AVAILABLE(421),
    CANT_OPEN_DATA_CONNECTION(425),
    TRANSFER_ABORTED(426),
```

```
    FILE_BUSY(450),
    ERROR_PROCESSING_ABORTED(451),
    NOT_ENOUGH_STORAGE(452),

    SYNTAX_ERROR(500),
    BAD_PARAMETERS(501),
    NOT_IMPLEMENTED(502),
    BAD_SEQUENCE_OF_COMMANDS(503),
    COMMAND_NOT_IMPLEMENTED_FOR_PARAMETER(504),
    NOT_LOGGED_IN(530),
    NEED_ACCOUNT_TO_STORE_FILES(532),
    REQUESTED_ACTION_NOT_TAKEN(550),
    PAGE_TYPE_UNKNOWN(551),
    EXCEEDED_STORAGE_ALLOCATION(552),
    FILENAME_NOT_ALLOWED(553);

    public final int code;

    ResponseTypes(int code) {
        this.code = code;
    }
}
```

ctrlftp/ctrlftp-core/src/main/java/rchat/info/ctrlftp/core/Server.java

```
package rchat.info.ctrlftp.core;

import rchat.info.ctrlftp.core.dependencies.AbstractDependency;
import rchat.info.ctrlftp.core.dependencies.DependencyManager;
import rchat.info.ctrlftp.core.reflections.ClassesLoader;

import java.io.IOException;
import java.net.InetSocketAddress;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.HashSet;
import java.util.List;
import java.util.Set;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

/**
 * An FTP server
 */
public class Server {
    private ServerSocket serverSocket;
    private ExecutorService sessions;
    private DependencyManager dependencyManager;
```

```
    private Set<Class<?>> serviceClasses;
    private Set<Class<? extends AbstractDependency>> dependencyClasses;

    public Server(List<String> configs) {
        sessions = Executors.newVirtualThreadPerTaskExecutor();
        serviceClasses = new HashSet<>();
        dependencyClasses = new HashSet<>();
        ClassesLoader.load(serviceClasses, dependencyClasses, configs);
        dependencyManager = new DependencyManager(this);
    }

    public void mainLoop() throws IOException {
        serverSocket = new ServerSocket(10021);
        while (!serverSocket.isClosed()) {
            Socket client = serverSocket.accept();
            sessions.submit(new Session(this, client));
        }
    }

    public void shutdown() {
        sessions.shutdown();
    }

    public DependencyManager getDependencyManager() {
        return dependencyManager;
    }

    public Set<Class<?>> getServiceClasses() {
        return serviceClasses;
    }

    public Set<Class<? extends AbstractDependency>> getDependencyClasses() {
        return dependencyClasses;
    }
}
```

ctrlftp/ctrlftp-core/src/main/java/rchat/info/ctrlftp/core/Session.java

```
package rchat.info.ctrlftp.core;

import rchat.info.ctrlftp.core.dependencies.DependencyLevel;
import rchat.info.ctrlftp.core.dependencies.DependencyManager;
import rchat.info.ctrlftp.core.reflections.MethodResolver;
import rchat.info.ctrlftp.core.responses.Response;
import rchat.info.ctrlftp.core.responses.ResponseTypes;

import java.io.*;
import java.lang.reflect.InvocationTargetException;
```

```java
import java.net.Socket;
import java.net.SocketAddress;
import java.nio.charset.StandardCharsets;

/**
 * A class that stores current session
 */
public class Session implements Runnable {
    private Server serverContext;
    private Socket client;
    private StringBuilder inputBuffer;
    private DependencyManager dependencyManager;

    public Session(Server serverContext, Socket client) {
        this.serverContext = serverContext;
        this.client = client;
        this.inputBuffer = new StringBuilder();
        this.dependencyManager = new DependencyManager(serverContext,
                this,
                DependencyLevel.SESSION,
                serverContext.getDependencyManager());
    }

    /**
     * A method that finds required method,
     * injects dependencies, launches method and
     * sends back response
     *
     * @param command a raw command from a user
     */
    private Response launchMethod(String command) {
        try {
            var optionalMethod = MethodResolver.findMethod(this.serverContext, command.split(" ")[0]);
            if (optionalMethod.isEmpty()) {
                return new Response(ResponseTypes.NOT_IMPLEMENTED, "Method " + command + " not
↪  implemented");
            }
            var targetMethod = optionalMethod.get();

            DependencyManager local = new DependencyManager(
                    serverContext,
                    this,
                    command,
                    this.dependencyManager);
            var methodParameters = local.getDependenciesForParameters(targetMethod.getParameters());

            return (Response) targetMethod.invoke(null, methodParameters.toArray());
        } catch (Exception e) {
            e.printStackTrace();
            return new Response(ResponseTypes.REQUESTED_ACTION_NOT_TAKEN);
```

```
        }
    }

    /**
     * A method that gets result from method launching and sends it to a user
     *
     * @param command a raw command from a user
     */
    private void processCommand(String command) throws IOException {
        sendResponse(launchMethod(command));
    }

    public void sendResponse(Response response) throws IOException {
        if (client.isClosed() || response == null) return;

        BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(client.getOutputStream()));
        writer.write(response.serialize().toString());
        writer.flush();
    }

    /**
     * Disconnects from the current user
     */
    public void disconnect(Response reason) {
        try {
            sendResponse(reason);
            client.close();
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }

    /**
     * Gives user remote address
     * @return user remote address
     */
    public SocketAddress getRemoteSocketAddress() {
        if (client == null || !client.isConnected())
            return null;

        return client.getRemoteSocketAddress();
    }

    /**
     * A main method that buffers user input and distinguishes raw command lines
     */
    @Override
    public void run() {
        try {
            sendResponse(new Response(ResponseTypes.COMMAND_OK, "Connected succesfully"));
```

```
            BufferedReader r = new BufferedReader(new InputStreamReader(client.getInputStream(),
↪  StandardCharsets.UTF_8));
            char[] buffer = new char[1024];
            int read;
            while ((read = r.read(buffer)) != -1) {
                inputBuffer.append(new String(buffer, 0, read));

                int newLinePosition;
                while ((newLinePosition = inputBuffer.indexOf("\r\n")) != -1) {
                    processCommand(inputBuffer.substring(0, newLinePosition));
                    inputBuffer = new StringBuilder(inputBuffer.substring(newLinePosition + 2));
                }
            }
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
}
```

6.2 Приложение 2. Исходный код библиотеки CtrlFTP Dependencies

ctrlftp/ctrlftp-dependencies/src/main/java/rchat/info/ctrlftp/dependencies/authentication/AuthenticationResu

```
package rchat.info.ctrlftp.dependencies.authentication;

import rchat.info.ctrlftp.core.responses.Response;

/**
 * A class that describes authentication result (user info, cause of error and isAuthenticated flag)
 */
public record AuthenticationResult<UserInfo>(boolean isAuthenticated, Response cause, UserInfo authInfo)
↪  {
}
```

ctrlftp/ctrlftp-dependencies/src/main/java/rchat/info/ctrlftp/dependencies/authentication/BaseAuthenticationDependen

```
package rchat.info.ctrlftp.dependencies.authentication;

import rchat.info.ctrlftp.core.annotations.Dependency;
import rchat.info.ctrlftp.core.dependencies.AbstractDependency;
import rchat.info.ctrlftp.core.dependencies.DependencyLevel;
```

```java
/**
 * A dependency for authentication. Has a session level. You should specify UserInfo
 * type when overriding
 */
@Dependency(level = DependencyLevel.SESSION)
public abstract class BaseAuthenticationDependency<UserInfo> extends AbstractDependency {
    /**
     * Checks user authentication. Is something is wrong or missing, return object contains
     * cause of error
     * @return result of authentication
     */
    public abstract AuthenticationResult<UserInfo> authenticate();

    /**
     * Logouts from the accout
     */
    public abstract void logout();
}
```

ctrlftp/ctrlftp-dependencies/src/main/java/rchat/info/ctrlftp/dependencies/deserializer/BaseDeserializer.java

```java
package rchat.info.ctrlftp.dependencies.deserializer;

import rchat.info.ctrlftp.core.annotations.Dependency;
import rchat.info.ctrlftp.core.dependencies.AbstractDependency;
import rchat.info.ctrlftp.core.dependencies.DependencyLevel;

/**
 * A dependency which deserializes and parses command arguments
 * @param <DeserializeClass> a result deserialize class
 */
@Dependency(level = DependencyLevel.COMMAND)
public abstract class BaseDeserializer<DeserializeClass> extends AbstractDependency {
    private final DeserializeClass deserializeData;

    public BaseDeserializer(String command) {
        this.deserializeData = deserialize(command);
    }

    /**
     * A deserialize method that parses command and returns DeserializeClass
     * @param command a raw command
     * @return deserialized data
     */
    protected abstract DeserializeClass deserialize(String command);
```

```
    /**
     * A method that returns args from command
     * @return deserialized args from command
     */
    public DeserializeClass getDeserializeData() {
        return deserializeData;
    }
}
```

ctrlftp/ctrlftp-dependencies/src/main/java/rchat/info/ctrlftp/dependencies/deserializer/SingleStringDeserialized.jav

```
package rchat.info.ctrlftp.dependencies.deserializer;

public record SingleStringDeserialized(String arg) {}
```

ctrlftp/ctrlftp-dependencies/src/main/java/rchat/info/ctrlftp/dependencies/deserializer/SingleStringDeserializer.jav

```
package rchat.info.ctrlftp.dependencies.deserializer;

/**
 * A dependency to parse single string after a command name
 */
public class SingleStringDeserializer extends BaseDeserializer<SingleStringDeserialized> {
    public SingleStringDeserializer(String command) {
        super(command);
    }

    @Override
    public SingleStringDeserialized deserialize(String command) {
        var firstSpaceIndex = command.indexOf(' ');

        return new SingleStringDeserialized(firstSpaceIndex == -1 ? "" :
↪  command.substring(firstSpaceIndex + 1));
    }
}
```

ctrlftp/ctrlftp-dependencies/src/main/java/rchat/info/ctrlftp/dependencies/filetransfer/AcceptEvent.java

```java
package rchat.info.ctrlftp.dependencies.filetransfer;

/**
 * An interface to call when file is accepted by
 * file accept transfer dependency
 *
 * @param <DataClass> a program specific data class to store incoming data
 */
public interface AcceptEvent<DataClass> {
    void onAccept(DataClass tempFile);

    void onError(Exception e);
}
```

ctrlftp/ctrlftp-dependencies/src/main/java/rchat/info/ctrlftp/dependencies/filetransfer/AcceptTransferDependency.jav

```java
package rchat.info.ctrlftp.dependencies.filetransfer;

import rchat.info.ctrlftp.core.Session;
import rchat.info.ctrlftp.core.annotations.Dependency;
import rchat.info.ctrlftp.core.dependencies.AbstractDependency;
import rchat.info.ctrlftp.core.dependencies.DependencyLevel;

import java.io.*;
import java.net.InetAddress;
import java.net.ServerSocket;
import java.net.Socket;
import java.net.UnknownHostException;
import java.util.AbstractMap;

/**
 * Dependency to handle file transfering and accepting
 *
 * @param <Pipe> a piping class that transforms user input to readable data classes and backwards
 * @param <DataClass> a data class that stores user input and output
 */
@Dependency(level = DependencyLevel.SESSION)
public class AcceptTransferDependency<Pipe extends BasePipeDependency<DataClass>, DataClass> extends
↪  AbstractDependency {
    private final Session session;
    private AbstractMap.SimpleEntry<String, Integer> clientAddress = new AbstractMap.SimpleEntry<>("",
↪  20);
    private Socket clientSocket = null;
    private ServerSocket server = null;
    private Thread acceptTransferThread = null;
    private final Pipe pipeClass;
```

```
    private boolean isPassive = false;

    public AcceptTransferDependency(Session session, Pipe pipeClass) {
        this.session = session;
        this.pipeClass = pipeClass;
    }

    /**
     * Updates client address and port. Warning: stops current data connection if new client port is
↪   different
     *
     * @param address a new client address and port
     */
    public void setClientAddress(AbstractMap.SimpleEntry<String, Integer> address) throws IOException {
        if (!this.clientAddress.equals(address))
            disconnect();

        this.clientAddress = address;
    }

    /**
     * Sets mode passive or active. If current mode and passed mode are not equal, throws an error
     *
     * @param isPassive sets
     */
    public void setPassive(boolean isPassive) throws IOException {
        if (this.isPassive != isPassive) {
            try {
                disconnect();
            } catch (IOException _) {
            }

            if (isPassive) {
                this.server = new ServerSocket(0);
            } else {
                this.server = null;
            }

            this.isPassive = isPassive;
        }
    }

    public boolean isPassive() {
        return isPassive;
    }

    /**
     * Gets server info if it runs in passive mode
     */
    public AbstractMap.SimpleEntry<String, Integer> getServerInfo() throws UnknownHostException {
```

```
        if (this.isPassive && this.server != null && !this.server.isClosed()) {
            var remoteSocketAddress = session.getRemoteSocketAddress().toString();
            return new AbstractMap.SimpleEntry<>(remoteSocketAddress.substring(1,
↪   remoteSocketAddress.indexOf(":")),
                    this.server.getLocalPort());
        }

        return null;
    }

    /**
     * Closes current connection and interrupts sending/accepting thread
     *
     * @throws IOException
     */
    public void disconnect() throws IOException {
        if (this.acceptTransferThread != null && this.acceptTransferThread.isAlive()) {
            this.acceptTransferThread.interrupt();
        }
        if (clientSocket != null && !clientSocket.isClosed()) {
            clientSocket.close();
        }
    }

    /**
     * Connects this.clientSocket to a client data port in active mode and awaits for client
     * connection in passive mode
     *
     * @throws IOException
     */
    private void connect() throws IOException {
        if (clientSocket != null && !clientSocket.isClosed()) return;

        if (this.isPassive) {
            if (this.server == null) {
                this.server = new ServerSocket(0);
            }

            clientSocket = this.server.accept();
        } else {
            if (this.server != null && !this.server.isClosed()) {
                this.server.close();
                this.server = null;
            }

            clientSocket = new Socket(
                    this.clientAddress.getKey() == null ?
                            this.session.getRemoteSocketAddress().toString() :
                            this.clientAddress.getKey(),
                    this.clientAddress.getValue());
```

```
    }
}

/**
 * Makes server to accept a file asyncrhonously.
 *
 * @param events calls onAccept method if file accepted
 *               succesfully, or else onError
 */
public void accept(AcceptEvent<DataClass> events) {
    try {
        if (this.acceptTransferThread != null && this.acceptTransferThread.isAlive()) {
            throw new AcceptTransferIsBusy("Accept transfer is busy right now");
        }

        this.acceptTransferThread = Thread.ofVirtual().start(() -> {
            InputStream clientInput = null;
            try {
                connect();

                clientInput = this.clientSocket.getInputStream();
                var parsedData = this.pipeClass.pipeClientInputToDataClass(clientInput);
                events.onAccept(parsedData);
            } catch (Exception e) {
                events.onError(e);
            } finally {
                if (clientInput != null) {
                    try {
                        clientInput.close();
                    } catch (IOException _) {

                    }
                }
            }

            try {
                disconnect();
            } catch (IOException _) {
            }
        });
    } catch (Exception e) {
        events.onError(e);
    }
}

/**
 * Sends a data to a client
 *
 * @param data a path to a data
 * @param events a callback to call onError and onTransferred events
```

```
     */
    public void send(DataClass data, TransferEvent events) {
        try {
            if (this.acceptTransferThread != null && this.acceptTransferThread.isAlive()) {
                throw new AcceptTransferIsBusy("Accept transfer is busy right now");
            }

            this.acceptTransferThread = Thread.ofVirtual().start(() -> {
                OutputStream clientOutput = null;
                try {
                    connect();

                    clientOutput = this.clientSocket.getOutputStream();
                    this.pipeClass.pipeDataClassToClient(data, clientOutput);
                    events.onTransferred();
                } catch (Exception e) {
                    events.onError(e);
                } finally {
                    if (clientOutput != null) {
                        try {
                            clientOutput.close();
                        } catch (IOException _) {
                        }
                    }
                }

                try {
                    disconnect();
                } catch (IOException e) {
                    throw new RuntimeException(e);
                }
            });
        } catch (Exception e) {
            events.onError(e);
        }
    }
}
```

ctrlftp/ctrlftp-dependencies/src/main/java/rchat/info/ctrlftp/dependencies/filetransfer/AcceptTransferIsBusy.java

```
package rchat.info.ctrlftp.dependencies.filetransfer;

public class AcceptTransferIsBusy extends RuntimeException {
    public AcceptTransferIsBusy(String message) {
        super(message);
    }
```

```
}
```

ctrlftp/ctrlftp-dependencies/src/main/java/rchat/info/ctrlftp/dependencies/filetransfer/BasePipeDependency.java

```java
package rchat.info.ctrlftp.dependencies.filetransfer;

import rchat.info.ctrlftp.core.annotations.Dependency;
import rchat.info.ctrlftp.core.dependencies.AbstractDependency;
import rchat.info.ctrlftp.core.dependencies.DependencyLevel;

import java.io.InputStream;
import java.io.OutputStream;
/**
 * The purpose of this class is transform client data of type A
 * and transform it into system-specific data type and backwards
 */
@Dependency(level = DependencyLevel.SESSION)
public abstract class BasePipeDependency<DataClass> extends AbstractDependency {
    /**
     * Pipes client specific input to a implementation specific data class
     * @param client a client raw input
     * @return parsed DataClass
     */
    public abstract DataClass pipeClientInputToDataClass(InputStream client);

    /**
     * Pipes specific data structure to a raw user content
     * @param data a specific data structure
     * @param client a client raw output
     */
    public abstract void pipeDataClassToClient(DataClass data, OutputStream client);
}
```

ctrlftp/ctrlftp-dependencies/src/main/java/rchat/info/ctrlftp/dependencies/filetransfer/TransferEvent.java

```java
package rchat.info.ctrlftp.dependencies.filetransfer;

/**
 * An interface to call when file is transfered by
 * file accept transfer dependency
 */
public interface TransferEvent {
    /**
```

```
     * A callback
     */
    void onTransferred();

    void onError(Exception e);
}
```

6.3 Приложение 3. Исходный код FTP сервера

examples/example-dbauth/src/main/java/rchat/info/ctrlftp/dbauth/controllers/AuthenticationController.java

```
package rchat.info.ctrlftp.dbauth.controllers;

import rchat.info.ctrlftp.core.Session;
import rchat.info.ctrlftp.core.annotations.Command;
import rchat.info.ctrlftp.core.responses.Response;
import rchat.info.ctrlftp.core.responses.ResponseTypes;
import rchat.info.ctrlftp.dbauth.dependencies.AuthenticationDependency;
import rchat.info.ctrlftp.dbauth.dependencies.FileAcceptTransferDependency;
import rchat.info.ctrlftp.dependencies.deserializer.SingleStringDeserializer;

import java.io.IOException;

public class AuthenticationController {
    @Command(name = "USER")
    public static Response onUser(AuthenticationDependency auth, SingleStringDeserializer arg) {
        auth.setLogin(arg.getDeserializeData().arg());

        var authResponse = auth.authenticate();
        return authResponse.cause();
    }

    @Command(name = "PASS")
    public static Response onPassword(AuthenticationDependency auth, SingleStringDeserializer arg) {
        auth.setPassword(arg.getDeserializeData().arg());

        var authResponse = auth.authenticate();
        return authResponse.cause();
    }

    @Command(name = "REIN")
    public static Response onRein(AuthenticationDependency auth,
                                  FileAcceptTransferDependency fileAcceptTransferDependency) {
        try {
            fileAcceptTransferDependency.disconnect();
```

```
        } catch (IOException _) {
        }
        auth.logout();

        return new Response(ResponseTypes.COMMAND_OK, "Logout succesfully");
    }

    @Command(name = "QUIT")
    public static Response onQuit(AuthenticationDependency auth, Session systemSessionContext,
                                  FileAcceptTransferDependency fileAcceptTransferDependency) {
        try {
            fileAcceptTransferDependency.disconnect();
        } catch (IOException _) {
        }
        auth.logout();
        systemSessionContext.disconnect(new Response(ResponseTypes.COMMAND_OK, "Quitted"));

        return null;
    }
}
```

examples/example-dbauth/src/main/java/rchat/info/ctrlftp/dbauth/controllers/FileAcceptTransferController.java

```
package rchat.info.ctrlftp.dbauth.controllers;

import rchat.info.ctrlftp.core.annotations.Command;
import rchat.info.ctrlftp.core.responses.Response;
import rchat.info.ctrlftp.core.responses.ResponseTypes;
import rchat.info.ctrlftp.dbauth.dependencies.AuthenticationDependency;
import rchat.info.ctrlftp.dbauth.dependencies.deserializers.DataPortDeserializer;
import rchat.info.ctrlftp.dbauth.dependencies.FileAcceptTransferDependency;
import rchat.info.ctrlftp.dbauth.dependencies.FilePipeDependency;
import rchat.info.ctrlftp.dependencies.deserializer.SingleStringDeserializer;

import java.io.IOException;
import java.util.AbstractMap;

public class FileAcceptTransferController {
    @Command(name = "PORT")
    public static Response dataPort(
            DataPortDeserializer dataPort,
            FileAcceptTransferDependency fileAcceptTransferDependency,
            AuthenticationDependency auth) {
        var authResult = auth.authenticate();
        if (!authResult.isAuthenticated())
            return authResult.cause();
```

```
        if (dataPort.getDeserializeData() == null)
            return new Response(ResponseTypes.BAD_PARAMETERS, "Invalid args");

        try {
            fileAcceptTransferDependency.setClientAddress(new AbstractMap.SimpleEntry<>(
                    dataPort.getDeserializeData().ip(),
                    dataPort.getDeserializeData().port()
            ));
        } catch (IOException e) {
            return new Response(ResponseTypes.SERVICE_NOT_AVAILABLE, "Couldn't close current
↪   connection");
        }

        return new Response(ResponseTypes.COMMAND_OK);
    }

    @Command(name = "PASV")
    public static Response setPassive(FileAcceptTransferDependency fileAcceptTransferDependency,
                                      AuthenticationDependency auth) {
        var authResult = auth.authenticate();
        if (!authResult.isAuthenticated())
            return authResult.cause();

        try {
            fileAcceptTransferDependency.setPassive(true);
            var serverInfo = fileAcceptTransferDependency.getServerInfo();
            String[] ipBytes = serverInfo.getKey().split("\\.");

            return new Response(ResponseTypes.ENTERING_PASSIVE_MODE, String.format("Entering Passive Mode
↪   (%s,%s,%s,%s,%d,%d)",
                    ipBytes[0], ipBytes[1], ipBytes[2], ipBytes[3], serverInfo.getValue() >> 8,
                    serverInfo.getValue() & 0xFF));
        } catch (IOException e) {
            return new Response(ResponseTypes.SERVICE_NOT_AVAILABLE, "Couldn't make current server
↪   passive");
        }
    }

    @Command(name = "TYPE")
    public static Response setType(AuthenticationDependency auth,
                                   FilePipeDependency pipe,
                                   SingleStringDeserializer arg) {
        var authResult = auth.authenticate();
        if (!authResult.isAuthenticated())
            return authResult.cause();

        return pipe.setType(arg.getDeserializeData().arg());
    }
}
```

examples/example-dbauth/src/main/java/rchat/info/ctrlftp/dbauth/controllers/NavigationController.java

```
package rchat.info.ctrlftp.dbauth.controllers;

import rchat.info.ctrlftp.core.Session;
import rchat.info.ctrlftp.core.annotations.Command;
import rchat.info.ctrlftp.core.responses.Response;
import rchat.info.ctrlftp.core.responses.ResponseTypes;
import rchat.info.ctrlftp.dbauth.dependencies.FileAcceptTransferDependency;
import rchat.info.ctrlftp.dbauth.dependencies.FilePipeRecord;
import rchat.info.ctrlftp.dbauth.dependencies.NavigationDependency;
import rchat.info.ctrlftp.dependencies.deserializer.SingleStringDeserializer;
import rchat.info.ctrlftp.dependencies.filetransfer.TransferEvent;
import rchat.info.ctrlftp.dbauth.dependencies.AuthenticationDependency;

import java.io.IOException;
import java.nio.file.Paths;
import java.util.List;

public class NavigationController {
    @Command(name = "PWD")
    public static Response PWD(AuthenticationDependency auth,
                               NavigationDependency navigation) {
        var authResult = auth.authenticate();
        if (!authResult.isAuthenticated())
            return authResult.cause();

        return new Response(ResponseTypes.PATHNAME_CREATED, String.format("\"%s\" is current directory.",
                navigation.getCurrentFolder().toString()
                        .replace("\\", "/")));
    }

    @Command(name = "CWD")
    public static Response CWD(AuthenticationDependency auth,
                               NavigationDependency navigation,
                               SingleStringDeserializer arg) {
        var authResult = auth.authenticate();
        if (!authResult.isAuthenticated())
            return authResult.cause();

        try {
            navigation.changeWorkingDirectory(arg.getDeserializeData().arg());

            return new Response(ResponseTypes.PATHNAME_CREATED, String.format("\"%s\" is current
↪  directory.",
                    navigation.getCurrentFolder().toString()
                            .replace("\\", "/")));
        } catch (Exception e) {
```

```
            return new Response(ResponseTypes.BAD_PARAMETERS, "Bad filepath");
        }
    }

    @Command(name = "LIST")
    public static Response getList(AuthenticationDependency auth,
                                   FileAcceptTransferDependency transfer,
                                   NavigationDependency navigationDependency,
                                   SingleStringDeserializer args,
                                   Session session) {
        var authResult = auth.authenticate();
        if (!authResult.isAuthenticated())
            return authResult.cause();

        try {
            List<String> filesList;
            if (args.getDeserializeData().arg().isEmpty()) {
                filesList = navigationDependency.getFilesNames();
            } else {
                filesList =
↪   navigationDependency.getFilesNames(Paths.get(args.getDeserializeData().arg()));
            }

            String data = String.join("\r\n", filesList);
            transfer.send(new FilePipeRecord(null, data), new TransferEvent() {
                @Override
                public void onTransferred() {
                    try {
                        session.sendResponse(
                                new Response(ResponseTypes.COMMAND_OK, "A file sent succesfully")
                        );
                    } catch (IOException e) {
                        throw new RuntimeException(e);
                    }
                }

                @Override
                public void onError(Exception e) {
                    e.printStackTrace();
                    try {
                        session.sendResponse(
                                new Response(ResponseTypes.TRANSFER_ABORTED, "Failed to send a file")
                        );
                    } catch (IOException ex) {
                        throw new RuntimeException(ex);
                    }
                }
            });
        } catch (IOException e) {
            return new Response(ResponseTypes.FILENAME_NOT_ALLOWED, "Couldn't get folder contents");
```

```
        }

        return new Response(ResponseTypes.ABOUT_TO_OPEN_CONNECTION, "Ready to send data");
    }

    @Command(name = "NLST")
    public static Response getListShort(AuthenticationDependency auth,
                                        FileAcceptTransferDependency transfer,
                                        NavigationDependency navigationDependency,
                                        SingleStringDeserializer args,
                                        Session session) {
        var authResult = auth.authenticate();
        if (!authResult.isAuthenticated())
            return authResult.cause();

        try {
            List<String> filesList;
            if (args.getDeserializeData().arg().isEmpty()) {
                filesList = navigationDependency.getFilesNamesShort();
            } else {
                filesList =
↪   navigationDependency.getFilesNamesShort(Paths.get(args.getDeserializeData().arg()));
            }

            String data = String.join("\r\n", filesList);
            transfer.send(new FilePipeRecord(null, data), new TransferEvent() {
                @Override
                public void onTransferred() {
                    try {
                        session.sendResponse(
                                new Response(ResponseTypes.COMMAND_OK, "A file sent succesfully")
                        );
                    } catch (IOException e) {
                        throw new RuntimeException(e);
                    }
                }

                @Override
                public void onError(Exception e) {
                    e.printStackTrace();
                    try {
                        session.sendResponse(
                                new Response(ResponseTypes.TRANSFER_ABORTED, "Failed to send a file")
                        );
                    } catch (IOException ex) {
                        throw new RuntimeException(ex);
                    }
                }
            });
        } catch (IOException e) {
```

```
            return new Response(ResponseTypes.FILENAME_NOT_ALLOWED, "Couldn't get folder contents");
        }

        return new Response(ResponseTypes.ABOUT_TO_OPEN_CONNECTION, "Ready to send data");
    }
}
```

examples/example-dbauth/src/main/java/rchat/info/ctrlftp/dbauth/controllers/ServiceController.java

```
package rchat.info.ctrlftp.dbauth.controllers;

import rchat.info.ctrlftp.core.Session;
import rchat.info.ctrlftp.core.annotations.Command;
import rchat.info.ctrlftp.core.responses.Response;
import rchat.info.ctrlftp.core.responses.ResponseTypes;
import rchat.info.ctrlftp.dbauth.dependencies.AuthenticationDependency;
import rchat.info.ctrlftp.dbauth.dependencies.FileAcceptTransferDependency;
import rchat.info.ctrlftp.dbauth.dependencies.FilePipeRecord;
import rchat.info.ctrlftp.dbauth.dependencies.NavigationDependency;
import rchat.info.ctrlftp.dbauth.dependencies.ServiceDependency;
import rchat.info.ctrlftp.dependencies.deserializer.SingleStringDeserializer;
import rchat.info.ctrlftp.dependencies.filetransfer.AcceptEvent;
import rchat.info.ctrlftp.dependencies.filetransfer.TransferEvent;

import java.io.File;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.nio.file.StandardOpenOption;

public class ServiceController {
    @Command(name = "SYST")
    public static Response getSystemType(ServiceDependency serviceDependency) {
        var osName = serviceDependency.getSystemName();
        if (osName != null) {
            return new Response(ResponseTypes.SYSTEM_TYPE, osName);
        } else {
            return new Response(ResponseTypes.NOT_IMPLEMENTED, "Unable to recognize current system");
        }
    }

    @Command(name = "NOOP")
    public static Response noop() {
        return new Response(ResponseTypes.COMMAND_OK, "Glad to work with ya, human 'fella!");
    }

    @Command(name = "RETR")
    public static Response getFile(
```

```
            AuthenticationDependency auth,
            FileAcceptTransferDependency fileTransfer,
            NavigationDependency navigation,
            SingleStringDeserializer args,
            Session session
    ) {
        var authResult = auth.authenticate();
        if (!authResult.isAuthenticated())
            return authResult.cause();

        var file = navigation.getFile(args.getDeserializeData().arg());
        if (!file.isFile())
            return new Response(ResponseTypes.FILE_BUSY, "File doesnt exists at path");

        fileTransfer.send(new FilePipeRecord(file, null), new TransferEvent() {
            @Override
            public void onTransferred() {
                try {
                    session.sendResponse(new Response(ResponseTypes.FILE_ACTION_OK, "File sent
↪  successfully"));
                } catch (IOException e) {
                    throw new RuntimeException(e);
                }
            }

            @Override
            public void onError(Exception e) {
                try {
                    session.sendResponse(new Response(ResponseTypes.FILE_BUSY));
                } catch (IOException ex) {
                    throw new RuntimeException(ex);
                }
            }
        });

        return new Response(ResponseTypes.ABOUT_TO_OPEN_CONNECTION, "Ready to send data");
    }

    @Command(name = "STOR")
    public static Response storeFile(
            AuthenticationDependency auth,
            FileAcceptTransferDependency fileTransfer,
            NavigationDependency navigation,
            SingleStringDeserializer args,
            Session session
    ) {
        var authResult = auth.authenticate();
        if (!authResult.isAuthenticated())
            return authResult.cause();
```

```
        var file = navigation.getPathRelativeToCWD(args.getDeserializeData().arg());

        fileTransfer.accept(new AcceptEvent<FilePipeRecord>() {
            @Override
            public void onAccept(FilePipeRecord tempFile) {
                var oldFile = file.toFile();
                oldFile.delete();
                tempFile.iFile().renameTo(new File(oldFile.getAbsolutePath()));

                try {
                    session.sendResponse(new Response(ResponseTypes.FILE_ACTION_OK, "File moved
↪  successfully"));
                } catch (IOException e) {
                    throw new RuntimeException(e);
                }
            }

            @Override
            public void onError(Exception e) {
                try {
                    session.sendResponse(new Response(ResponseTypes.FILE_BUSY));
                } catch (IOException ex) {
                    throw new RuntimeException(ex);
                }
            }
        });

        return new Response(ResponseTypes.ABOUT_TO_OPEN_CONNECTION, "Ready to send data");
    }

    @Command(name = "STOU")
    public static Response storeUniqueFile(
            AuthenticationDependency auth,
            FileAcceptTransferDependency fileTransfer,
            NavigationDependency navigation,
            SingleStringDeserializer args,
            Session session
    ) {
        var authResult = auth.authenticate();
        if (!authResult.isAuthenticated())
            return authResult.cause();

        var file = navigation.getPathRelativeToCWD(args.getDeserializeData().arg());

        fileTransfer.accept(new AcceptEvent<FilePipeRecord>() {
            private File getUniqueFile(String folderName, String searchedFilename) {
                int num = 1;
                String extension = getExtension(searchedFilename);
                String filename = searchedFilename.substring(0, searchedFilename.lastIndexOf("."));
                File file = new File(folderName, searchedFilename);
```

```
            while (file.exists()) {
                searchedFilename = filename + "(" + (num++) + ")" + extension;
                file = new File(folderName, searchedFilename);
            }
            return file;
        }

        private String getExtension(String name) {
            return name.substring(name.lastIndexOf("."));
        }

        @Override
        public void onAccept(FilePipeRecord tempFile) {
            var oldFile = file.toFile();
            var uniqueFile = getUniqueFile(
                    oldFile.getParent(),
                    oldFile.getName()
            );
            tempFile.iFile().renameTo(uniqueFile);

            try {
                session.sendResponse(new Response(ResponseTypes.FILE_ACTION_OK, String.format("\"%s\"
↪  saved with this name",
                        uniqueFile.getName())));
            } catch (IOException e) {
                throw new RuntimeException(e);
            }
        }

        @Override
        public void onError(Exception e) {
            try {
                session.sendResponse(new Response(ResponseTypes.FILE_BUSY));
            } catch (IOException ex) {
                throw new RuntimeException(ex);
            }
        }
    });

    return new Response(ResponseTypes.ABOUT_TO_OPEN_CONNECTION, "Ready to send data");
}

@Command(name = "APPE")
public static Response appendFile(
        AuthenticationDependency auth,
        FileAcceptTransferDependency fileTransfer,
        NavigationDependency navigation,
        SingleStringDeserializer args,
        Session session
) {
```

```
        var authResult = auth.authenticate();
        if (!authResult.isAuthenticated())
            return authResult.cause();

        var file = navigation.getPathRelativeToCWD(args.getDeserializeData().arg());
        if (!file.toFile().isFile())
            return new Response(ResponseTypes.BAD_PARAMETERS, "Invalid filename");

        fileTransfer.accept(new AcceptEvent<FilePipeRecord>() {
            @Override
            public void onAccept(FilePipeRecord tempFile) {
                var oldFile = file.toFile();
                if (oldFile.exists()) {
                    tempFile.iFile().renameTo(new File(oldFile.getAbsolutePath()));
                } else {
                    try {
                        Files.write(
                                file,
↪  Files.readAllBytes(Paths.get(tempFile.iFile().getAbsoluteFile().toString())),
                                StandardOpenOption.APPEND
                        );
                    } catch (IOException e) {
                        try {
                            session.sendResponse(new Response(ResponseTypes.FILENAME_NOT_ALLOWED,
↪  "Couldn't append a file"));
                        } catch (IOException ex) {
                            throw new RuntimeException(ex);
                        }
                    }
                }

                try {
                    session.sendResponse(new Response(ResponseTypes.FILE_ACTION_OK, "File moved
↪  successfully"));
                } catch (IOException e) {
                    throw new RuntimeException(e);
                }
            }

            @Override
            public void onError(Exception e) {
                try {
                    session.sendResponse(new Response(ResponseTypes.FILE_BUSY));
                } catch (IOException ex) {
                    throw new RuntimeException(ex);
                }
            }
        });
```

```
        return new Response(ResponseTypes.ABOUT_TO_OPEN_CONNECTION, "Ready to send data");
    }

    @Command(name = "RNFR")
    public static Response renameFrom(
            AuthenticationDependency auth,
            NavigationDependency navigation,
            SingleStringDeserializer args
    ) {
        var authResult = auth.authenticate();
        if (!authResult.isAuthenticated())
            return authResult.cause();

        navigation.setRenameFrom(args.getDeserializeData().arg());

        return new Response(ResponseTypes.FILE_ACTION_PENDING_INFO, "Send rename to info");
    }

    @Command(name = "RNTO")
    public static Response renameTo(
            AuthenticationDependency auth,
            NavigationDependency navigation,
            SingleStringDeserializer args
    ) {
        var authResult = auth.authenticate();
        if (!authResult.isAuthenticated())
            return authResult.cause();

        return navigation.renameTo(args.getDeserializeData().arg());
    }

    @Command(name = "ABOR")
    public static Response abort(
            AuthenticationDependency auth,
            FileAcceptTransferDependency transfer) {
        var authResult = auth.authenticate();
        if (!authResult.isAuthenticated())
            return authResult.cause();

        try {
            transfer.disconnect();
        } catch (IOException e) {
            return new Response(ResponseTypes.REQUESTED_ACTION_NOT_TAKEN, "Unable to abort file
↪   transfer");
        }

        return new Response(ResponseTypes.CLOSING_DATA_CONNECTION, "Closing data connection");
    }

    @Command(name = "DELE")
```

```
public static Response delete(
        AuthenticationDependency auth,
        NavigationDependency navi,
        SingleStringDeserializer arg) {
    var authResult = auth.authenticate();
    if (!authResult.isAuthenticated())
        return authResult.cause();

    var file = navi.getFile(arg.getDeserializeData().arg());
    if (!file.exists() || !file.isFile() || !file.delete()) {
        return new Response(ResponseTypes.FILE_BUSY, "File doesn't exists");
    }

    return new Response(ResponseTypes.FILE_ACTION_OK, "File deleted successfully");
}

static void deleteDir(File file) {
    File[] contents = file.listFiles();
    if (contents != null) {
        for (File f : contents) {
            deleteDir(f);
        }
    }
    file.delete();
}

@Command(name = "RMD")
public static Response removeDirectory(
        AuthenticationDependency auth,
        NavigationDependency navi,
        SingleStringDeserializer arg) {
    var authResult = auth.authenticate();
    if (!authResult.isAuthenticated())
        return authResult.cause();

    var file = navi.getFile(arg.getDeserializeData().arg());
    if (!file.exists() || !file.isDirectory()) {
        return new Response(ResponseTypes.FILE_BUSY, "Folder doesn't exists");
    }

    deleteDir(file);

    return new Response(ResponseTypes.FILE_ACTION_OK, "Directory deleted successfully");
}

@Command(name = "MKD")
public static Response makeDirectory(
        AuthenticationDependency auth,
        NavigationDependency navi,
        SingleStringDeserializer arg) {
```

```
        var authResult = auth.authenticate();
        if (!authResult.isAuthenticated())
            return authResult.cause();

        var file = navi.getFile(arg.getDeserializeData().arg());
        if (!file.mkdirs()) {
            return new Response(ResponseTypes.FILE_BUSY, "Unable to create directory");
        }

        return new Response(ResponseTypes.FILE_ACTION_OK, "Directory created succesfully");
    }
}
```

examples/example-dbauth/src/main/java/rchat/info/ctrlftp/dbauth/dependencies/deserializers/DataPortDeserializer.java

```
package rchat.info.ctrlftp.dbauth.dependencies.deserializers;

import rchat.info.ctrlftp.core.annotations.Dependency;
import rchat.info.ctrlftp.core.dependencies.DependencyLevel;
import rchat.info.ctrlftp.dependencies.deserializer.BaseDeserializer;
import rchat.info.ctrlftp.dependencies.deserializer.SingleStringDeserializer;

/**
 * Deserializer for data port command
 */
@Dependency(level = DependencyLevel.COMMAND)
public class DataPortDeserializer extends BaseDeserializer<DataPortDeserializer.DataPort> {
    public DataPortDeserializer(SingleStringDeserializer deserializer) {
        super(deserializer.getDeserializeData().arg());
    }

    @Override
    public DataPort deserialize(String command) {
        String[] bytesArgs = command.split(",");
        if (bytesArgs.length == 6) {
            String ip = String.format("%s.%s.%s.%s",
                    bytesArgs[0], bytesArgs[1], bytesArgs[2], bytesArgs[3]);
            Integer port = Integer.parseInt(bytesArgs[4]) << 8 + Integer.parseInt(bytesArgs[5]);

            return new DataPort(ip, port);
        }
        return null;
    }

    public record DataPort(String ip, Integer port) {}
}
```

examples/example-dbauth/src/main/java/rchat/info/ctrlftp/dbauth/dependencies/AuthenticationDependency.java

```java
package rchat.info.ctrlftp.dbauth.dependencies;

import rchat.info.ctrlftp.core.responses.Response;
import rchat.info.ctrlftp.core.responses.ResponseTypes;
import rchat.info.ctrlftp.dbauth.dependencies.DatabaseDependency;
import rchat.info.ctrlftp.dbauth.entities.UserEntity;
import rchat.info.ctrlftp.dbauth.repositories.UserEntityRepository;
import rchat.info.ctrlftp.dependencies.authentication.AuthenticationResult;
import rchat.info.ctrlftp.dependencies.authentication.BaseAuthenticationDependency;

import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

public class AuthenticationDependency extends BaseAuthenticationDependency<UserEntity> {
    private String login = null;
    private String hashedPassword = null;
    private UserEntity found = null;
    private final MessageDigest messageDigest;
    private final UserEntityRepository userEntityRepository;

    public AuthenticationDependency(DatabaseDependency database) {
        this.userEntityRepository = new UserEntityRepository(database.getDatabase());
        try {
            this.messageDigest = MessageDigest.getInstance("SHA-256");
        } catch (NoSuchAlgorithmException e) {
            throw new RuntimeException(e);
        }
    }

    public void setLogin(String login) {
        this.login = login;
    }

    public void setPassword(String password) {
        messageDigest.update(password.getBytes());

        this.hashedPassword = new String(messageDigest.digest());
    }

    @Override
    public AuthenticationResult<UserEntity> authenticate() {
        if (found != null) {
            return new AuthenticationResult<>(
                    true,
                    new Response(ResponseTypes.AUTH_SUCCESS, String.format("You logged in successfully as
↪  '%s' (%d)",
```

```
                        found.getLogin(), found.getId())),
                found);
        }

        if (login == null) {
            return new AuthenticationResult<>(false,
                    new Response(ResponseTypes.NEED_ACCOUNT_TO_LOGIN, "Please, provide login"),
                    null);
        }

        if (hashedPassword == null) {
            return new AuthenticationResult<>(false,
                    new Response(ResponseTypes.USERNAME_OK_NEED_PASS, "Please, provide password"),
                    null);
        }

        UserEntity foundUser = userEntityRepository.findUserEntityByLoginAndPassword(this.login,
↪   this.hashedPassword);
        if (foundUser != null) {
            this.found = foundUser;
            this.login = null;
            this.hashedPassword = null;
            return new AuthenticationResult<>(
                    true,
                    new Response(ResponseTypes.AUTH_SUCCESS, String.format("You logged in successfully as
↪   '%s' (%d)",
                            found.getLogin(), found.getId())),
                    foundUser);
        }

        if (userEntityRepository.findUserEntityByLogin(this.login) != null) {
            return new AuthenticationResult<>(true,
                    new Response(ResponseTypes.NOT_LOGGED_IN, "Invalid password"),
                    null);
        }

        UserEntity newUser = new UserEntity();
        newUser.setLogin(this.login);
        newUser.setHashPassword(this.hashedPassword);
        userEntityRepository.addUser(newUser);

        this.found = newUser;
        this.login = null;
        this.hashedPassword = null;

        return new AuthenticationResult<>(
                true,
                new Response(ResponseTypes.AUTH_SUCCESS, String.format("You created account '%s' (%d)",
                        found.getLogin(), found.getId())),
                this.found);
```

```
    }

    @Override
    public void logout() {
        this.found = null;
        this.login = null;
        this.hashedPassword = null;
    }

    public Long getUserId() {
        if (this.found == null) return null;

        return this.found.getId();
    }
}
```

examples/example-dbauth/src/main/java/rchat/info/ctrlftp/dbauth/dependencies/DatabaseDependency.java

```
package rchat.info.ctrlftp.dbauth.dependencies;

import jakarta.persistence.EntityManager;
import jakarta.persistence.EntityManagerFactory;
import jakarta.persistence.Persistence;
import rchat.info.ctrlftp.core.annotations.Dependency;
import rchat.info.ctrlftp.core.dependencies.AbstractDependency;
import rchat.info.ctrlftp.core.dependencies.DependencyLevel;

import java.util.HashMap;
import java.util.Map;

@Dependency(level = DependencyLevel.GLOBAL)
public class DatabaseDependency extends AbstractDependency {
    private static final EntityManagerFactory emf;
    private EntityManager entityManager;

    static {
        Map<String, String> env = System.getenv();
        Map<String, Object> configOverrides = new HashMap<String, Object>();
        for (String envName : env.keySet()) {
            if (envName.contains("DB_URL")) {
                configOverrides.put("jakarta.persistence.jdbc.url", env.get(envName));
            } else if (envName.contains("DB_USER")) {
                configOverrides.put("jakarta.persistence.jdbc.user", env.get(envName));
            } else if (envName.contains("DB_PASSWORD")) {
                configOverrides.put("jakarta.persistence.jdbc.password", env.get(envName));
            }
        }
```

```
        emf = Persistence.createEntityManagerFactory("ctrlftp-example-s3", configOverrides);
    }

    public DatabaseDependency() {
        this.entityManager = emf.createEntityManager();
    }

    public EntityManager getDatabase() {
        return this.entityManager;
    }
}
```

examples/example-dbauth/src/main/java/rchat/info/ctrlftp/dbauth/dependencies/FileAcceptTransferDependency.java

```
package rchat.info.ctrlftp.dbauth.dependencies;

import rchat.info.ctrlftp.core.Session;
import rchat.info.ctrlftp.dependencies.filetransfer.AcceptTransferDependency;

public class FileAcceptTransferDependency extends AcceptTransferDependency<FilePipeDependency,
↪  FilePipeRecord> {
    public FileAcceptTransferDependency(Session session, FilePipeDependency pipeClass) {
        super(session, pipeClass);
    }
}
```

examples/example-dbauth/src/main/java/rchat/info/ctrlftp/dbauth/dependencies/FilePipeDependency.java

```
package rchat.info.ctrlftp.dbauth.dependencies;

import rchat.info.ctrlftp.core.responses.Response;
import rchat.info.ctrlftp.core.responses.ResponseTypes;
import rchat.info.ctrlftp.dependencies.filetransfer.BasePipeDependency;

import java.io.*;
import java.util.Arrays;

public class FilePipeDependency extends BasePipeDependency<FilePipeRecord> {
    public enum TransferTypes {
        ASCII("A"),
        EBCDIC("E"),
        IMAGE("I");

        public final String val;
```

```
        TransferTypes(String val) {
            this.val = val;
        }
    }

    private TransferTypes transferType = TransferTypes.ASCII;

    public Response setType(String type) {
        var parsedTransferType = Arrays.stream(TransferTypes.values())
                .filter(tranferType -> tranferType.val.equals(type))
                .findFirst();

        if (parsedTransferType.isEmpty()) {
            return new Response(ResponseTypes.BAD_PARAMETERS);
        }

        this.transferType = parsedTransferType.get();
        return new Response(ResponseTypes.COMMAND_OK, "Set up transfer type");
    }

    @Override
    public FilePipeRecord pipeClientInputToDataClass(InputStream client) {
        if (transferType == TransferTypes.IMAGE) {
            try {
                File tempFile = File.createTempFile(this.getClass().getName() + "-", ".temp");
                client.transferTo(new FileOutputStream(tempFile));

                return new FilePipeRecord(tempFile, null);
            } catch (IOException e) {
                throw new RuntimeException(e);
            }
        } else {
            try {
                ByteArrayOutputStream result = new ByteArrayOutputStream();
                byte[] buffer = new byte[1024];
                for (int length; (length = client.read(buffer)) != -1; ) {
                    result.write(buffer, 0, length);
                }

                return new FilePipeRecord(null, result.toString(transferType == TransferTypes.ASCII ?
↪   "ASCII" : "Cp1047"));
            } catch (IOException e) {
                throw new RuntimeException(e);
            }
        }
    }

    @Override
    public void pipeDataClassToClient(FilePipeRecord data, OutputStream client) {
        if (data.text() != null) {
```

```
            try (ByteArrayInputStream inputStream = new ByteArrayInputStream(data.text().getBytes(
                    transferType == TransferTypes.ASCII ? "ASCII" :
                            transferType == TransferTypes.EBCDIC ? "Cp1047" : "UTF-8"))) {
                inputStream.transferTo(client);
            } catch (IOException e) {
                throw new RuntimeException(e);
            }
        } else {
            try (FileInputStream inputStream = new FileInputStream(data.iFile())) {
                inputStream.transferTo(client);
            } catch (IOException e) {
                throw new RuntimeException(e);
            }
        }

        /*if (transferType == TransferTypes.IMAGE) {

        } else {
            try (ByteArrayInputStream inputStream = new
↪  ByteArrayInputStream(data.text().getBytes(transferType == TransferTypes.ASCII ? "ASCII" : "Cp1047")))
↪  {
                inputStream.transferTo(client);
            } catch (IOException e) {
                throw new RuntimeException(e);
            }
        }*/
    }
}
```

examples/example-dbauth/src/main/java/rchat/info/ctrlftp/dbauth/dependencies/FilePipeRecord.java

```
package rchat.info.ctrlftp.dbauth.dependencies;

import java.io.File;

public record FilePipeRecord(File iFile, String text) {
}
```

examples/example-dbauth/src/main/java/rchat/info/ctrlftp/dbauth/dependencies/NavigationDependency.java

```
package rchat.info.ctrlftp.dbauth.dependencies;

import rchat.info.ctrlftp.core.annotations.Dependency;
import rchat.info.ctrlftp.core.dependencies.AbstractDependency;
import rchat.info.ctrlftp.core.dependencies.DependencyLevel;
import rchat.info.ctrlftp.core.responses.Response;
```

```
import rchat.info.ctrlftp.core.responses.ResponseTypes;
import rchat.info.ctrlftp.dbauth.utils.OSValidator;

import java.io.BufferedReader;
import java.io.File;
import java.io.IOException;
import java.io.InputStreamReader;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.List;

@Dependency(level = DependencyLevel.SESSION)
public class NavigationDependency extends AbstractDependency {
    private Path currentFolder;
    private Path renameFrom;

    public NavigationDependency() {
        currentFolder = Paths.get("").toAbsolutePath();
    }

    public Path getCurrentFolder() {
        return currentFolder;
    }

    public Path getPathRelativeToCWD(String path) {
        Path resultPath;
        if (Paths.get(path).isAbsolute()) {
            resultPath = Paths.get(path).toAbsolutePath();
        } else {
            resultPath = Paths.get(currentFolder.toAbsolutePath() + "/" + path).toAbsolutePath();
        }

        return resultPath;
    }

    public void changeWorkingDirectory(String path) {
        var newFolder = getPathRelativeToCWD(path);
        if (!newFolder.toFile().isDirectory() || !newFolder.toFile().exists())
            throw new RuntimeException("Folder doesn't exists");
        this.currentFolder = newFolder;
    }

    public File getFile(String path) {
        return getPathRelativeToCWD(path).toFile();
    }

    private List<String> getFilesNames(Path path, boolean shortened) throws IOException {
        List<String> result = new ArrayList<>();
//        try (Stream<Path> paths = Files.walk(path, 1)) {
```

```
//            result = paths
//                    .filter(Files::isRegularFile)
//                    .map(Path::getFileName)
//                    .map(Path::toString)
//                    .toList();
//        }
//
//        return result;

        if (OSValidator.isWindows()) {
            // TODO: implement LS for windows
        } else {
            try {
                Process process = new ProcessBuilder()
                        .command("bash", "-c",
                                "export LC_ALL=en_US.UTF-8 ; " +
                                        (shortened ? "ls -1 -a" : "ls -la"))
                        .directory(path.toFile())
                        .start();

                BufferedReader stdInput = new BufferedReader(new
                        InputStreamReader(process.getInputStream()));

                String s;
                while ((s = stdInput.readLine()) != null) {
                    result.add(s);
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
        }

        return result;
    }



    public List<String> getFilesNames(Path path) throws IOException {
        return getFilesNames(path, false);
    }

    public List<String> getFilesNamesShort(Path path) throws IOException {
        return getFilesNames(path, true);
    }

    public List<String> getFilesNames() throws IOException {
        return getFilesNames(this.currentFolder);
    }

    public List<String> getFilesNamesShort() throws IOException {
```

```
        return getFilesNamesShort(this.currentFolder);
    }

    public void setRenameFrom(String path) {
        this.renameFrom = getPathRelativeToCWD(path);
    }

    public Response renameTo(String newPath) {
        if (this.renameFrom == null) {
            return new Response(ResponseTypes.FILENAME_NOT_ALLOWED, "You should setup rename from
↪  first");
        }

        var renameToPath = getPathRelativeToCWD(newPath);

        var succeeded = this.renameFrom.toFile().renameTo(renameToPath.toFile());

        if (!succeeded) {
            return new Response(ResponseTypes.FILENAME_NOT_ALLOWED, "Unabled to rename");
        }

        return new Response(ResponseTypes.FILE_ACTION_OK, "File renamed succesully");
    }
}
```

examples/example-dbauth/src/main/java/rchat/info/ctrlftp/dbauth/dependencies/ServiceDependency.java

```
package rchat.info.ctrlftp.dbauth.dependencies;

import rchat.info.ctrlftp.core.annotations.Dependency;
import rchat.info.ctrlftp.core.dependencies.AbstractDependency;
import rchat.info.ctrlftp.core.dependencies.DependencyLevel;
import rchat.info.ctrlftp.dbauth.utils.OSValidator;

@Dependency(level = DependencyLevel.GLOBAL)
public class ServiceDependency extends AbstractDependency {
    public String getSystemName() {
        if (OSValidator.isWindows()) {
            return "Windows_NT";
        } else {
            return "UNIX Type: L8";
        }
    }
}
```

examples/example-dbauth/src/main/java/rchat/info/ctrlftp/dbauth/dependencies/TransferTypeNotFound.java

```
package rchat.info.ctrlftp.dbauth.dependencies;

public class TransferTypeNotFound extends RuntimeException {
    public TransferTypeNotFound(String message) {
        super(message);
    }
}
```

examples/example-dbauth/src/main/java/rchat/info/ctrlftp/dbauth/entities/UserEntity.java

```
package rchat.info.ctrlftp.dbauth.entities;

import jakarta.persistence.*;

@Entity
@Table
public class UserEntity {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    @Column(nullable = false, unique = true)
    private String login;

    @Column(nullable = false)
    private String hashPassword;

    public Long getId() {
        return id;
    }

    public String getLogin() {
        return login;
    }

    public void setLogin(String login) {
        this.login = login;
    }

    public void setHashPassword(String hashPassword) {
        this.hashPassword = hashPassword;
    }
}
```

examples/example-dbauth/src/main/java/rchat/info/ctrlftp/dbauth/repositories/UserEntityRepository.java

```
package rchat.info.ctrlftp.dbauth.repositories;

import jakarta.persistence.EntityManager;
import jakarta.persistence.EntityTransaction;
import rchat.info.ctrlftp.dbauth.entities.UserEntity;

public class UserEntityRepository {
    private final EntityManager database;

    public UserEntityRepository(EntityManager database) {
        this.database = database;
    }

    public UserEntity findUserEntityByLoginAndPassword(String login, String hashedPassword) {
        return database.createQuery("""
                        select user
                        from UserEntity user
                        where user.login = :login and
                        user.hashPassword = :password
                        """, UserEntity.class)
                .setParameter("login", login)
                .setParameter("password", hashedPassword)
                .getSingleResultOrNull();
    }

    public UserEntity findUserEntityByLogin(String login) {
        return database.createQuery("""
                        select user
                        from UserEntity user
                        where user.login = :login
                        """, UserEntity.class)
                .setParameter("login", login)
                .getSingleResultOrNull();
    }

    public void addUser(UserEntity newUser) {
        EntityTransaction entityTransaction = this.database.getTransaction();
        entityTransaction.begin();
        this.database.persist(newUser);
        entityTransaction.commit();
    }
}
```

examples/example-dbauth/src/main/java/rchat/info/ctrlftp/dbauth/utils/OSValidator.java

```
package rchat.info.ctrlftp.dbauth.utils;

public class OSValidator {
```

```
    private static String OS = System.getProperty("os.name").toLowerCase();

    public static boolean isWindows() {
        return OS.contains("win");
    }

    public static boolean isMac() {
        return OS.contains("mac");
    }

    public static boolean isUnix() {
        return (OS.contains("nix") || OS.contains("nux") || OS.contains("aix"));
    }

    public static boolean isSolaris() {
        return OS.contains("sunos");
    }

    public static String getOS(){
        if (isWindows()) {
            return "win";
        } else if (isMac()) {
            return "osx";
        } else if (isUnix()) {
            return "uni";
        } else if (isSolaris()) {
            return "sol";
        } else {
            return "err";
        }
    }

}
```

examples/example-dbauth/src/main/java/rchat/info/ctrlftp/dbauth/utils/OSValidator.java

```
package rchat.info.ctrlftp.dbauth;

import rchat.info.ctrlftp.core.Server;

import java.io.IOException;
import java.util.List;

public class Main {
    public static void main(String[] args) throws IOException {
        Server s = new Server(List.of("dependencies.xml"));
        s.mainLoop();
```

```
    }
}
```

examples/example-dbauth/src/main/resources/dependencies.xml

```
<?xml version="1.0" encoding="utf-8"?>
<config>
    <services>
        <service>rchat.info.ctrlftp.dbauth.controllers.AuthenticationController</service>
        <service>rchat.info.ctrlftp.dbauth.controllers.FileAcceptTransferController</service>
        <service>rchat.info.ctrlftp.dbauth.controllers.NavigationController</service>
        <service>rchat.info.ctrlftp.dbauth.controllers.ServiceController</service>
    </services>
    <dependencies>
        <dependency>rchat.info.ctrlftp.dependencies.deserializer.SingleStringDeserializer</dependency>
        <dependency>rchat.info.ctrlftp.dbauth.dependencies.AuthenticationDependency</dependency>
        <dependency>rchat.info.ctrlftp.dbauth.dependencies.DatabaseDependency</dependency>
        <dependency>rchat.info.ctrlftp.dbauth.dependencies.FileAcceptTransferDependency</dependency>
        <dependency>rchat.info.ctrlftp.dbauth.dependencies.FilePipeDependency</dependency>
        <dependency>rchat.info.ctrlftp.dbauth.dependencies.NavigationDependency</dependency>
        <dependency>rchat.info.ctrlftp.dbauth.dependencies.ServiceDependency</dependency>

↪   <dependency>rchat.info.ctrlftp.dbauth.dependencies.deserializers.DataPortDeserializer</dependency>
    </dependencies>
</config>
```

examples/example-dbauth/src/main/resources/META-INF/persistence.xml

```
<persistence xmlns="http://xmlns.jcp.org/xml/ns/persistence"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
             http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd"
             version="2.1">

    <persistence-unit name="ctrlftp-example-s3" transaction-type="RESOURCE_LOCAL">
        <class>rchat.info.ctrlftp.dbauth.entities.UserEntity</class>
        <class>rchat.info.ctrlftp.dbauth.entities.FileEntity</class>
        <properties>
            <property name="jakarta.persistence.jdbc.driver" value="org.postgresql.Driver" /> <!-- DB
↪   Driver -->
            <property name="jakarta.persistence.jdbc.url" value="PLACE_IN_ENV_AS_DB_URL" /> <!-- BD Mane
↪   -->
            <property name="jakarta.persistence.jdbc.user" value="PLACE_IN_ENV_AS_DB_USER" /> <!-- DB
↪   User -->
            <property name="jakarta.persistence.jdbc.password" value="PLACE_IN_ENV_AS_DB_PASSWORD" />
↪   <!-- DB Password -->
```

```
            <property name="hibernate.hbm2ddl.auto" value="update" /> <!-- create / create-drop / update
↪  -->
            <!--property name="hibernate.show_sql" value="true" /--> <!-- Show SQL in console -->
            <!--property name="hibernate.format_sql" value="true" /--> <!-- Show SQL formatted -->
        </properties>

    </persistence-unit>

</persistence>
```

6.4 Приложение 4. Исходный код FTP клиента

examples/example-client/setup.sh

```
#!/bin/bash

PYTHON_DIR="${PYTHON_DIR:-.venv}"
UI_COMPILE_OUTPUT=./uicompiled

echo "Python dir: ${PYTHON_DIR}"

echo "Installing requirments"
${PYTHON_DIR}/bin/pip3 install -r ./requirments.txt

echo "Compiling .qrc file"
$(find ${PYTHON_DIR}/lib -wholename */PySide6/*/rcc) resources.qrc -g python -o resources_rc.py; \

echo "Compiling .ui files"
for file in $(find ui -type f); do \
  filename=$(basename ${file}); \
  $(find ${PYTHON_DIR}/lib -wholename */PySide6/*/uic) ${file} -g python -o
↪  ${UI_COMPILE_OUTPUT}/"${filename%.*}".py; \
done

echo "Setup completed! Now you can run main"
```

examples/example-client/resources.qrc

```
<!DOCTYPE RCC><RCC version="1.0">
<qresource>
    <file>assets/create.svg</file>
    <file>assets/delete.svg</file>
    <file>assets/download.svg</file>
    <file>assets/rename.svg</file>
```

```
    <file>assets/upload.svg</file>
</qresource>
</RCC>
```

examples/example-client/requirments.txt

```
PySide6==6.9.0
PySide6_Addons==6.9.0
PySide6_Essentials==6.9.0
shiboken6==6.9.0
```

examples/example-client/main.py

```
import sys
from ftplib import FTP

from PySide6.QtWidgets import QApplication

from dialogs.connectdialog import ConnectDialog
from dialogs.errordialog import ErrorDialog
from dialogs.explorerdialog import ExplorerDialog
from dialogs.loaderdialog import LoaderDialog

def show_error(error: str, parent=None):
    error_dialog = ErrorDialog(str(error), parent)
    error_dialog.exec()

def get_connection_options():
    connect_dialog = ConnectDialog()
    return_code = connect_dialog.exec()
    return connect_dialog.get_options() if return_code == 1 else None

def prepare_connection(connection_options) -> FTP:
    def inner_ftp_execute():
        ftp = FTP()
        ftp.connect(host=connection_options['server_address'],
        ↪  port=int(connection_options['server_port']))
        if not ftp.login(user=connection_options['user_login'],
        ↪  passwd=connection_options['user_password']).startswith(
                "2"):
            wait_dialog.close()
            raise ConnectionError("Некорректный логин, пароль или адрес")

        if connection_options["is_passive"]:
            ftp.set_pasv(True)
        else:
```

```
            ftp.sendport(connection_options["user_address"], int(connection_options["user_port"]))

        if not ftp.lastresp.startswith("2"):
            wait_dialog.close()
            raise ConnectionError("Некорректный логин, пароль или адрес")

        return ftp

    wait_dialog = LoaderDialog(inner_ftp_execute)
    wait_dialog.show()

    return wait_dialog.get_result()

def run_program():
    try:
        while True:
            connection_options = get_connection_options()

            if connection_options is None:
                break

            ftp = prepare_connection(connection_options)

            explorer_dialog = ExplorerDialog(ftp)
            explorer_dialog.exec()
    except Exception as e:
        show_error(str(e))
        raise e


if __name__ == "__main__":
    app = QApplication(sys.argv)
    run_program()
    sys.exit(app.exec())
```

examples/example-client/dialogs/connectdialog.py

```
from PySide6.QtGui import QIntValidator
from PySide6.QtWidgets import QDialog

from uicompiled.connect import Ui_Dialog as UiConnect

class ConnectDialog(QDialog):
    def __init__(self, parent=None) -> None:
        super().__init__(parent)
        self.ui = UiConnect()
        self.ui.setupUi(self)
        self.ui.buttonConnect.clicked.connect(self.selected)
```

```
        self.ui.inputIsPassive.checkStateChanged.connect(self.is_passive_selected)
        self.ui.inputServerPort.setValidator(QIntValidator(0, 65536))
        self.ui.inputDataPort.setValidator(QIntValidator(0, 65536))
        self.is_passive_selected()

    def get_options(self):
        return {
            "server_address": self.ui.inputServerAddress.text(),
            "server_port": self.ui.inputServerPort.text(),
            "is_passive": self.ui.inputIsPassive.isChecked(),
            "user_port": self.ui.inputDataPort.text(),
            "user_address": self.ui.inputDataAddress.text(),
            "user_login": self.ui.inputLogin.text(),
            "user_password": self.ui.inputPassword.text()
        }

    def is_passive_selected(self):
        is_disabled = self.ui.inputIsPassive.isChecked()
        self.ui.inputDataPort.setDisabled(is_disabled)
        self.ui.inputDataAddress.setDisabled(is_disabled)

    def selected(self):
        self.done(1)
```

examples/example-client/dialogs/createdialog.py

```
from PySide6.QtWidgets import QDialog

from uicompiled.create import Ui_Dialog as UiCreate

class CreateDialog(QDialog):
    def __init__(self, parent = None):
        super().__init__(parent)
        self.ui = UiCreate()
        self.ui.setupUi(self)
        self.ui.buttonBox.accepted.connect(self.accept)
        self.ui.buttonBox.rejected.connect(self.reject)

    def get_name(self):
        return self.ui.inputNewName.text()
```

examples/example-client/dialogs/errordialog.py

```
from PySide6.QtWidgets import QDialog

from uicompiled.error import Ui_Dialog as UiError
```

```
class ErrorDialog(QDialog):
    def __init__(self, textError: str, parent=None):
        super().__init__(parent)
        self.ui = UiError()
        self.ui.setupUi(self)
        self.ui.textEdit.setText(textError)
        self.ui.closeButton.clicked.connect(self.close)
```

examples/example-client/dialogs/explorerdialog.py

```
import os
import re
from ftplib import FTP
from os.path import abspath

from PySide6.QtWidgets import QDialog, QTableWidgetItem, QAbstractItemView, QFileDialog

from dialogs.createdialog import CreateDialog
from dialogs.errordialog import ErrorDialog
from dialogs.loaderdialog import LoaderDialog
from dialogs.renamedialog import RenameDialog
from uicompiled.explorer import Ui_Dialog as UiExplorer


def show_error(error: str, parent=None):
    error_dialog = ErrorDialog(str(error), parent)
    error_dialog.exec()


class ExplorerDialog(QDialog):
    def __init__(self, ftp: FTP, parent=None):
        super().__init__(parent)
        self.ui = UiExplorer()
        self.ui.setupUi(self)
        self.__ftp = ftp
        self.ui.inputCurrentDir.returnPressed.connect(self.change_directory)
        self.ui.contents.setEditTriggers(QAbstractItemView.EditTrigger.NoEditTriggers)
        self.ui.contents.cellDoubleClicked.connect(self.cell_double_clicked)
        self.ui.contents.setSelectionBehavior(QAbstractItemView.SelectionBehavior.SelectRows)
        self.ui.buttonDownload.clicked.connect(self.button_download_clicked)
        self.ui.buttonUpload.clicked.connect(self.button_upload_clicked)
        self.ui.buttonCreateDir.clicked.connect(self.button_folder_created_clicked)
        self.ui.buttonRemove.clicked.connect(self.button_delete_clicked)
        self.ui.buttonRename.clicked.connect(self.button_rename_clicked)
        self.__is_folder = []
        self.update_list()
```

```
    def update_list(self):
        self.ui.inputCurrentDir.setText(self.__ftp.pwd())

        current_index = -1

        def line_extractor(line: str):
            if line.startswith("total"):
                self.ui.contents.clearContents()
                self.ui.contents.setRowCount(0)
                self.__is_folder = []
                return
            nonlocal current_index

            if current_index == -1:
                current_index += 1
                return

            line_split = re.split(r'[ ]+', line)
            self.ui.contents.insertRow(self.ui.contents.rowCount())
            self.ui.contents.setItem(current_index, 0, QTableWidgetItem(line_split[-1]))
            self.ui.contents.setItem(current_index, 1,
                                     QTableWidgetItem("" if line_split[0].startswith("d") else
                                     ↪  line_split[4]))
            self.ui.contents.setItem(current_index, 2, QTableWidgetItem(" ".join(line_split[5:8])))
            self.__is_folder.append(line_split[0].startswith("d"))
            current_index += 1

        self.__ftp.retrlines("LIST", line_extractor)

    def change_directory(self, new_place: str = None):
        if new_place is not None:
            self.ui.inputCurrentDir.setText(new_place)

        self.ui.inputCurrentDir.setText(abspath(self.ui.inputCurrentDir.text()))

        try:
            self.__ftp.cwd(self.ui.inputCurrentDir.text())
        except Exception as e:
            show_error(f"Невозможно переключиться на директорию '{self.ui.inputCurrentDir.text()}'")

        self.update_list()

    def cell_double_clicked(self):
        selected_items = self.ui.contents.selectedItems()
        if len(selected_items) == 0:
            return

        selected_item = selected_items[0]

        if self.__is_folder[selected_item.row()]:
```

```
            self.change_directory(
                f"{self.ui.inputCurrentDir.text()}/{self.ui.contents.item(selected_item.row(),
                ↪  0).text()}")
        else:
            self.save_file(self.ui.contents.item(selected_item.row(), 0).text())

    def button_download_clicked(self):
        selected_items = self.ui.contents.selectedItems()
        if len(selected_items) == 0:
            show_error("Выберите файл для сохранения в таблице")
            return

        selected_item = selected_items[0]
        if self.__is_folder[selected_item.row()]:
            show_error("Невозможно сохранить папку, выберите файл")
            return

        self.save_file(self.ui.contents.item(selected_item.row(), 0).text())

    def save_file(self, file_name):
        (save_file_name, _) = QFileDialog.getSaveFileName(self,
                                                          "Сохранить файл",
                                                          os.getcwd() + "/" + file_name,
                                                          "Все файлы (*)"
                                                          )

        if save_file_name == "":
            return

        def inner_func():
            with open(save_file_name, "wb") as fp:
                self.__ftp.retrbinary(f"RETR {file_name}", fp.write)

        loader_dialog = LoaderDialog(inner_func)
        try:
            loader_dialog.show()
            loader_dialog.get_result()
        except Exception as e:
            show_error(str(e))
            loader_dialog.close()

    def button_upload_clicked(self):
        (upload_file_path, _) = QFileDialog.getOpenFileName(self,
                                                            "Загрузить файл",
                                                            os.getcwd(),
                                                            "*.*"
                                                            )

        if upload_file_path == "":
            return
```

```
        def inner_func():
            with open(upload_file_path, "rb") as fp:
                self.__ftp.storbinary(f"STOU {upload_file_path.split('/')[-1]}", fp)

        loader_dialog = LoaderDialog(inner_func)
        try:
            loader_dialog.show()
            loader_dialog.get_result()
        except Exception as e:
            show_error(str(e))
            loader_dialog.close()

        self.update_list()

    def button_folder_created_clicked(self):
        create_dir = CreateDialog(self)
        return_code = create_dir.exec()
        if return_code == QDialog.DialogCode.Rejected:
            return

        self.__ftp.mkd(create_dir.get_name())
        if not self.__ftp.lastresp.startswith("2"):
            show_error(f"Не удалось создать папку")

        self.update_list()

    def button_delete_clicked(self):
        selected_items = self.ui.contents.selectedItems()
        if len(selected_items) == 0:
            show_error("Выберите файл или папку для удаления в таблице")
            return

        selected_item = selected_items[0]
        if self.__is_folder[selected_item.row()]:
            self.delete_folder(self.ui.contents.item(selected_item.row(), 0).text())
        else:
            self.delete_file(self.ui.contents.item(selected_item.row(), 0).text())

    def delete_file(self, file_name: str):
        self.__ftp.delete(file_name)
        if not self.__ftp.lastresp.startswith("2"):
            show_error(f"Не удалось удалить файл {self.__ftp.lastresp}")

        self.update_list()

    def delete_folder(self, folder_name: str):
        self.__ftp.rmd(folder_name)
        if not self.__ftp.lastresp.startswith("2"):
            show_error(f"Не удалось удалить папку {self.__ftp.lastresp}")
```

```
        self.update_list()

    def button_rename_clicked(self):
        selected_items = self.ui.contents.selectedItems()
        if len(selected_items) == 0:
            show_error("Выберите файл или папку для переименования в таблице")
            return

        selected_item = selected_items[0]

        rename_dialog = RenameDialog(self.ui.contents.item(selected_item.row(), 0).text(), self)
        rename_dialog.exec()

        self.__ftp.rename(self.ui.contents.item(selected_item.row(), 0).text(),
                                    rename_dialog.get_name())
        if not self.__ftp.lastresp.startswith("2"):
            show_error(f"Не удалось переименовать {self.ui.contents.item(selected_item.row(), 0)} в
            ↪  {rename_dialog.get_name()}")

        self.update_list()
```

examples/example-client/dialogs/loaderdialog.py

```
from PySide6.QtWidgets import QDialog
from typing import Callable, Any

from uicompiled.loader import Ui_Dialog as UiLoader

class LoaderDialog(QDialog):
    def __init__(self, job: Callable[[], Any], parent = None):
        super().__init__(parent)
        self.ui = UiLoader()
        self.ui.setupUi(self)
        self.__job = job

    def get_result(self):
        result = self.__job()
        self.done(0)
        return result
```

examples/example-client/dialogs/renamedialog.py

```
from PySide6.QtWidgets import QDialog

from uicompiled.rename import Ui_Dialog as UiRename
```

```
class RenameDialog(QDialog):
    def __init__(self, old_name: str = None, parent = None):
        super().__init__()
        self.ui = UiRename()
        self.ui.setupUi(self)
        self.ui.buttonBox.accepted.connect(self.accept)
        self.ui.buttonBox.rejected.connect(self.reject)
        if old_name is not None:
            self.ui.inputNewName.setText(old_name)

    def get_name(self):
        return self.ui.inputNewName.text()
```

examples/example-client/ui/connect.ui

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>Dialog</class>
 <widget class="QDialog" name="Dialog">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>647</width>
    <height>626</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>Подключение к серверу</string>
  </property>
  <layout class="QVBoxLayout" name="verticalLayout_2">
   <item>
    <layout class="QVBoxLayout" name="verticalLayout">
     <item>
      <widget class="QLabel" name="label">
       <property name="styleSheet">
        <string notr="true">font: 700 14pt &amp;quot;Arial&amp;quot;;</string>
       </property>
       <property name="text">
        <string>Подключение к серверу</string>
       </property>
      </widget>
     </item>
     <item>
      <widget class="QLabel" name="label_2">
       <property name="text">
        <string>Адрес</string>
```

```
    </property>
   </widget>
  </item>
  <item>
   <widget class="QLineEdit" name="inputServerAddress">
    <property name="inputMask">
     <string>000.000.000.000</string>
    </property>
    <property name="text">
     <string>127.0.0.1</string>
    </property>
   </widget>
  </item>
  <item>
   <widget class="QLabel" name="label_3">
    <property name="text">
     <string>Порт</string>
    </property>
   </widget>
  </item>
  <item>
   <widget class="QLineEdit" name="inputServerPort">
    <property name="autoFillBackground">
     <bool>false</bool>
    </property>
    <property name="text">
     <string>21</string>
    </property>
    <property name="frame">
     <bool>true</bool>
    </property>
   </widget>
  </item>
 </layout>
</item>
<item>
 <spacer name="verticalSpacer">
  <property name="orientation">
   <enum>Qt::Vertical</enum>
  </property>
  <property name="sizeHint" stdset="0">
   <size>
    <width>20</width>
    <height>40</height>
   </size>
  </property>
 </spacer>
</item>
<item>
 <layout class="QVBoxLayout" name="verticalLayout_3">
```

```
<item>
 <widget class="QLabel" name="label_4">
  <property name="styleSheet">
   <string notr="true">font: 700 14pt &amp;quot;Arial&amp;quot;;</string>
  </property>
  <property name="text">
   <string>Дата-подключение</string>
  </property>
 </widget>
</item>
<item>
 <widget class="QCheckBox" name="inputIsPassive">
  <property name="text">
   <string>Пассивный режим</string>
  </property>
 </widget>
</item>
<item>
 <layout class="QVBoxLayout" name="dataConnectionInputs">
  <item>
   <widget class="QLabel" name="label_9">
    <property name="text">
     <string>Используемый адрес</string>
    </property>
   </widget>
  </item>
  <item>
   <widget class="QLineEdit" name="inputDataAddress">
    <property name="inputMask">
     <string>000.000.000.000</string>
    </property>
    <property name="text">
     <string>127.0.0.1</string>
    </property>
   </widget>
  </item>
  <item>
   <widget class="QLabel" name="label_6">
    <property name="text">
     <string>Используемый порт</string>
    </property>
   </widget>
  </item>
  <item>
   <widget class="QLineEdit" name="inputDataPort">
    <property name="text">
     <string>22</string>
    </property>
   </widget>
  </item>
```

```
   </layout>
  </item>
 </layout>
</item>
<item>
 <spacer name="verticalSpacer_2">
  <property name="orientation">
   <enum>Qt::Vertical</enum>
  </property>
  <property name="sizeHint" stdset="0">
   <size>
    <width>20</width>
    <height>40</height>
   </size>
  </property>
 </spacer>
</item>
<item>
 <layout class="QVBoxLayout" name="verticalLayout_4">
  <item>
   <widget class="QLabel" name="label_5">
    <property name="styleSheet">
     <string notr="true">font: 700 14pt &quot;Arial&quot;;</string>
    </property>
    <property name="text">
     <string>Данные пользователя</string>
    </property>
   </widget>
  </item>
  <item>
   <widget class="QLabel" name="label_7">
    <property name="text">
     <string>Логин</string>
    </property>
   </widget>
  </item>
  <item>
   <widget class="QLineEdit" name="inputLogin"/>
  </item>
  <item>
   <widget class="QLabel" name="label_8">
    <property name="text">
     <string>Пароль</string>
    </property>
   </widget>
  </item>
  <item>
   <widget class="QLineEdit" name="inputPassword">
    <property name="echoMode">
     <enum>QLineEdit::Password</enum>
```

```
      </property>
     </widget>
    </item>
   </layout>
  </item>
  <item>
   <spacer name="verticalSpacer_3">
    <property name="orientation">
     <enum>Qt::Vertical</enum>
    </property>
    <property name="sizeHint" stdset="0">
     <size>
      <width>20</width>
      <height>40</height>
     </size>
    </property>
   </spacer>
  </item>
  <item>
   <widget class="QPushButton" name="buttonConnect">
    <property name="text">
     <string>Подключиться</string>
    </property>
   </widget>
  </item>
 </layout>
</widget>
<resources/>
<connections/>
</ui>
```

examples/example-client/ui/create.ui

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>Dialog</class>
 <widget class="QDialog" name="Dialog">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>400</width>
    <height>166</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>Новая папка</string>
  </property>
```

```
  <layout class="QVBoxLayout" name="verticalLayout">
   <item>
    <widget class="QLabel" name="label">
     <property name="styleSheet">
      <string notr="true">font: 700 14pt &amp;quot;Arial&amp;quot;;</string>
     </property>
     <property name="text">
      <string>Введите имя новой папки</string>
     </property>
    </widget>
   </item>
   <item>
    <spacer name="verticalSpacer">
     <property name="orientation">
      <enum>Qt::Vertical</enum>
     </property>
     <property name="sizeHint" stdset="0">
      <size>
       <width>20</width>
       <height>32</height>
      </size>
     </property>
    </spacer>
   </item>
   <item>
    <widget class="QLabel" name="label_2">
     <property name="text">
      <string>Имя папки</string>
     </property>
    </widget>
   </item>
   <item>
    <widget class="QLineEdit" name="inputNewName"/>
   </item>
   <item>
    <widget class="QDialogButtonBox" name="buttonBox">
     <property name="standardButtons">
      <set>QDialogButtonBox::Cancel|QDialogButtonBox::Ok</set>
     </property>
    </widget>
   </item>
  </layout>
 </widget>
 <resources/>
 <connections/>
</ui>
```

examples/example-client/ui/error.ui

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>Dialog</class>
 <widget class="QDialog" name="Dialog">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>400</width>
    <height>300</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>Что-то пошло не так</string>
  </property>
  <layout class="QVBoxLayout" name="verticalLayout">
   <item>
    <widget class="QLabel" name="label">
     <property name="styleSheet">
      <string notr="true">font: 700 14pt &amp;quot;Arial&amp;quot;;</string>
     </property>
     <property name="text">
      <string>Что-то пошло не так T-T</string>
     </property>
    </widget>
   </item>
   <item>
    <widget class="QLabel" name="label_2">
     <property name="text">
      <string>Произошла ошибка при попытке выполнить действие:</string>
     </property>
     <property name="wordWrap">
      <bool>true</bool>
     </property>
    </widget>
   </item>
   <item>
    <widget class="QTextEdit" name="textEdit">
     <property name="readOnly">
      <bool>true</bool>
     </property>
    </widget>
   </item>
   <item>
    <widget class="QPushButton" name="closeButton">
     <property name="text">
      <string>Закрыть</string>
     </property>
    </widget>
   </item>
```

```
  </layout>
 </widget>
 <resources/>
 <connections/>
</ui>
```

examples/example-client/ui/explorer.ui

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>Dialog</class>
 <widget class="QDialog" name="Dialog">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>685</width>
    <height>622</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>Проводник</string>
  </property>
  <layout class="QVBoxLayout" name="verticalLayout">
   <item>
    <layout class="QHBoxLayout" name="horizontalLayout">
     <item>
      <widget class="QPushButton" name="buttonDownload">
       <property name="focusPolicy">
        <enum>Qt::NoFocus</enum>
       </property>
       <property name="toolTip">
        <string>Сохранить файл</string>
       </property>
       <property name="styleSheet">
        <string notr="true">background-color: rgb(61, 56, 70);</string>
       </property>
       <property name="text">
        <string/>
       </property>
       <property name="icon">
        <iconset resource="../resources.qrc">
         <normaloff>:/assets/download.svg</normaloff>:/assets/download.svg</iconset>
       </property>
       <property name="iconSize">
        <size>
         <width>32</width>
         <height>32</height>
```

```
   </size>
  </property>
 </widget>
</item>
<item>
 <widget class="QPushButton" name="buttonUpload">
  <property name="focusPolicy">
   <enum>Qt::NoFocus</enum>
  </property>
  <property name="toolTip">
   <string>Загрузить файл</string>
  </property>
  <property name="styleSheet">
   <string notr="true">background-color: rgb(61, 56, 70);</string>
  </property>
  <property name="text">
   <string/>
  </property>
  <property name="icon">
   <iconset resource="../resources.qrc">
    <normaloff>:/assets/upload.svg</normaloff>:/assets/upload.svg</iconset>
  </property>
  <property name="iconSize">
   <size>
    <width>32</width>
    <height>32</height>
   </size>
  </property>
 </widget>
</item>
<item>
 <widget class="QPushButton" name="buttonCreateDir">
  <property name="focusPolicy">
   <enum>Qt::NoFocus</enum>
  </property>
  <property name="toolTip">
   <string>Добавить папку</string>
  </property>
  <property name="styleSheet">
   <string notr="true">background-color: rgb(61, 56, 70);</string>
  </property>
  <property name="text">
   <string/>
  </property>
  <property name="icon">
   <iconset resource="../resources.qrc">
    <normaloff>:/assets/create.svg</normaloff>:/assets/create.svg</iconset>
  </property>
  <property name="iconSize">
   <size>
```

```
     <width>32</width>
     <height>32</height>
    </size>
   </property>
  </widget>
 </item>
 <item>
  <widget class="QPushButton" name="buttonRemove">
   <property name="focusPolicy">
    <enum>Qt::NoFocus</enum>
   </property>
   <property name="toolTip">
    <string>Удалить</string>
   </property>
   <property name="styleSheet">
    <string notr="true">background-color: rgb(61, 56, 70);</string>
   </property>
   <property name="text">
    <string/>
   </property>
   <property name="icon">
    <iconset resource="../resources.qrc">
     <normaloff>:/assets/delete.svg</normaloff>:/assets/delete.svg</iconset>
   </property>
   <property name="iconSize">
    <size>
     <width>32</width>
     <height>32</height>
    </size>
   </property>
  </widget>
 </item>
 <item>
  <widget class="QPushButton" name="buttonRename">
   <property name="focusPolicy">
    <enum>Qt::NoFocus</enum>
   </property>
   <property name="toolTip">
    <string>Переименовать</string>
   </property>
   <property name="styleSheet">
    <string notr="true">background-color: rgb(61, 56, 70);</string>
   </property>
   <property name="text">
    <string/>
   </property>
   <property name="icon">
    <iconset resource="../resources.qrc">
     <normaloff>:/assets/rename.svg</normaloff>:/assets/rename.svg</iconset>
   </property>
```

```
     <property name="iconSize">
      <size>
       <width>32</width>
       <height>32</height>
      </size>
     </property>
    </widget>
   </item>
   <item>
    <spacer name="horizontalSpacer">
     <property name="orientation">
      <enum>Qt::Horizontal</enum>
     </property>
     <property name="sizeHint" stdset="0">
      <size>
       <width>40</width>
       <height>20</height>
      </size>
     </property>
    </spacer>
   </item>
  </layout>
 </item>
 <item>
  <widget class="QLineEdit" name="inputCurrentDir"/>
 </item>
 <item>
  <widget class="QTableWidget" name="contents">
   <property name="columnCount">
    <number>3</number>
   </property>
   <attribute name="horizontalHeaderStretchLastSection">
    <bool>true</bool>
   </attribute>
   <column>
    <property name="text">
     <string>Имя</string>
    </property>
    <property name="textAlignment">
     <set>AlignLeading|AlignVCenter</set>
    </property>
   </column>
   <column>
    <property name="text">
     <string>Размер</string>
    </property>
    <property name="textAlignment">
     <set>AlignLeading|AlignVCenter</set>
    </property>
   </column>
```

```
     <column>
      <property name="text">
       <string>Последнее изменение</string>
      </property>
      <property name="textAlignment">
       <set>AlignLeading|AlignVCenter</set>
      </property>
     </column>
    </widget>
   </item>
  </layout>
 </widget>
 <resources>
  <include location="../resources.qrc"/>
 </resources>
 <connections/>
</ui>
```

examples/example-client/ui/loader.ui

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>Dialog</class>
 <widget class="QDialog" name="Dialog">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>284</width>
    <height>130</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>Загрузка...</string>
  </property>
  <layout class="QVBoxLayout" name="verticalLayout">
   <item>
    <widget class="QLabel" name="label">
     <property name="styleSheet">
      <string notr="true">font: 700 14pt &amp;quot;Arial&amp;quot;;</string>
     </property>
     <property name="text">
      <string>Подождите, операция выполняется...</string>
     </property>
     <property name="wordWrap">
      <bool>true</bool>
     </property>
    </widget>
```

```
   </item>
  </layout>
 </widget>
 <resources/>
 <connections/>
</ui>
```

examples/example-client/ui/rename.ui

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>Dialog</class>
 <widget class="QDialog" name="Dialog">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>400</width>
    <height>163</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>Переименовать</string>
  </property>
  <layout class="QVBoxLayout" name="verticalLayout">
   <item>
    <widget class="QLabel" name="label">
     <property name="styleSheet">
      <string notr="true">font: 700 14pt &amp;quot;Arial&amp;quot;;</string>
     </property>
     <property name="text">
      <string>Введите новое имя</string>
     </property>
    </widget>
   </item>
   <item>
    <spacer name="verticalSpacer">
     <property name="orientation">
      <enum>Qt::Vertical</enum>
     </property>
     <property name="sizeHint" stdset="0">
      <size>
       <width>20</width>
       <height>40</height>
      </size>
     </property>
    </spacer>
   </item>
```

```
   <item>
    <widget class="QLabel" name="label_2">
     <property name="text">
      <string>Имя файла/папки</string>
     </property>
    </widget>
   </item>
   <item>
    <widget class="QLineEdit" name="inputNewName"/>
   </item>
   <item>
    <widget class="QDialogButtonBox" name="buttonBox">
     <property name="standardButtons">
      <set>QDialogButtonBox::Cancel|QDialogButtonBox::Ok</set>
     </property>
    </widget>
   </item>
  </layout>
 </widget>
 <resources/>
 <connections/>
</ui>
```