Этап 1 – Сервер и клиент на сокетах. Обработка текстовых файлов.

1. Введение

Цель данного проекта состоит в разработке прототипа простого эмулятора файл-сервера. В рамках первого этапа будет реализована клиентская и серверная программа, обменивающиеся текстовыми файлами через сокеты.

2. Требования к продукту

2.1 Функциональные требования:

• Клиентская программа должна позволять пользователю вводить команды для создания, получения и удаления файлов на сервере.

• Серверная программа должна обрабатывать запросы клиентов и взаимодействовать с файлами на сервере.

• Для управления файлами использовать аналог упрощенных HTTP-запросов (GET, PUT, DELETE).

2.2 Нефункциональные требования:

• Клиентская и серверная программа должны быть написаны на языке программирования C#.

• Программы должны быть кроссплатформенными и работать на операционных системах Windows, Linux и macOS.

3. Описание данных

Данные передаются в текстовом формате через сокеты. Формат запросов и ответов описан в задании.

4. Описание процесса

• Клиентская программа запрашивает у пользователя действие (создание, получение, удаление файла).

• Клиент отправляет запрос на сервер.

• Сервер обрабатывает запрос и отвечает клиенту.

• Клиент получает ответ от сервера и печатает его в читаемом формате.

5. Требования к интерфейсу

• Интерфейс клиентской программы должен быть текстовым (CLI).

• Пользователю должно быть предложено ввести команду и, при необходимости, имя файла и его содержимое.

6. Требования к надежности

• Программа должна корректно обрабатывать все входные данные.

• Необходимо обеспечить надежное соединение между клиентом и сервером.

7. Требования к удобству использования

• Клиентская программа должна быть интуитивно понятной для пользователя.

• Серверная программа должна быть легко запускаться и настраиваться.

8. Требования к производительности

• Программа должна работать достаточно быстро для обработки запросов от нескольких клиентов одновременно.

9. Требования к технической поддержке

• Приложение должно быть документировано для облегчения поддержки и развития.

10. Дополнительные требования

• Программа должна завершать свою работу при получении команды "exit" от клиента.

11. План тестирования

• Провести тестирование функциональности каждой команды (GET, PUT, DELETE).

• Проверить обработку ошибок при вводе некорректных данных.

12. План внедрения

• Разработать инструкцию по установке и запуску клиентской и серверной программ.

• Подготовить документацию для пользователей.

13. Риски

• Отсутствие поддержки клиентской и серверной программ на разных операционных системах.

14. Приложения

Отсутствуют.

UML-диаграммы:

1. Диаграмма классов (Class Diagrams) - показывает классы клиентской и серверной программы, их атрибуты и методы.

2. Диаграмма последовательности (Sequence Diagrams) - отображает последовательность обмена сообщениями между клиентом и сервером при выполнении запроса.

3. Диаграмма состояний (State Diagrams) - показывает состояния клиентской и серверной программы и переходы между ними в ответ на действия пользователя или системы.

Этап 2 – Многопоточный сервер. Обработка произвольных файлов.

Этап 2 – Многопоточный сервер. Обработка произвольных файлов.

1. Введение

На втором этапе проекта требуется расширить функциональность сервера для обработки произвольных файлов и поддержки многопоточности.

2. Требования к продукту

2.1 Функциональные требования:

Серверная программа должна поддерживать многопоточность для обработки одновременных запросов от нескольких клиентов.

Клиентская программа должна позволять отправлять на сервер текстовые файлы и файлы изображений.

2.2 Нефункциональные требования:

Серверная программа должна быть способной обрабатывать большие объемы данных и эффективно использовать ресурсы системы.

Клиентская и серверная программа должны обеспечивать безопасность передачи данных.

3. Описание данных

В этой части проекта данные могут быть представлены в различных форматах в зависимости от типа файла: текстовые данные для текстовых файлов и бинарные данные для файлов изображений.

4. Описание процесса

Клиентская программа позволяет пользователю отправлять на сервер текстовые файлы и файлы изображений.

Серверная программа принимает файлы от клиентов, сохраняет их на сервере и выдает уникальные идентификаторы файлов.

При запросе на получение файла клиент указывает идентификатор файла или его имя, и сервер отправляет файл обратно клиенту.

5. Требования к интерфейсу

Клиентская программа должна предоставлять пользователю возможность выбора файла для отправки на сервер и указания имени файла для сохранения на сервере.

Серверная программа должна быть способна обрабатывать запросы от нескольких клиентов одновременно и безопасно сохранять файлы на сервере.

6. Требования к надежности

Программа должна корректно обрабатывать файлы различных форматов и размеров.

Сервер должен обеспечивать безопасное хранение и доступ к файлам клиентов.

7. Требования к удобству использования

Клиентская программа должна иметь интуитивно понятный интерфейс для отправки и получения файлов.

Серверная программа должна быть легко настраиваема и обеспечивать удобное администрирование файлов.

8. Требования к производительности

Сервер должен быть способен обрабатывать запросы от нескольких клиентов одновременно с высокой производительностью.

Программа должна эффективно использовать ресурсы системы для обработки и хранения файлов.

9. Требования к технической поддержке

Приложение должно быть документировано для обеспечения легкости поддержки и развития.

10. Дополнительные требования

Серверная программа должна сохранять связь между уникальными идентификаторами файлов и их именами для обеспечения доступа к файлам по их именам.

11. План тестирования

Провести тестирование функциональности отправки и получения файлов различных форматов.

Проверить многопоточность сервера и его способность обрабатывать запросы от нескольких клиентов одновременно.

12. План внедрения

Разработать инструкцию по установке и запуску серверной программы.

Подготовить документацию для пользователей.

13. Риски

Неправильная реализация многопоточности на сервере может привести к ошибкам и сбоям в работе программы.

14. Приложения

Отсутствуют.

UML-диаграммы:

Диаграмма классов (Class Diagrams) - показывает классы клиентской и серверной программы, их атрибуты и методы.

Диаграмма последовательности (Sequence Diagrams) - отображает последовательность обмена сообщениями между клиентом и сервером при отправке и получении файлов.

Диаграмма развертывания (Deployment Diagrams) - показывает физическое развертывание серверной программы на аппаратных узлах.