МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Кафедра «Математическое обеспечение и применение ЭВМ»

Курсовая работа
по дисциплине
«Программирование в сетях»
на тему «CGI-приложение для удаленной работы с сервером FTP»

ПГУ 09.03.04 – 08КП201.14 ПЗ

Направление подготовки - 09.03.04 Программная инженерия

Выполнил студент: _____ Левин М.В.
Группа: 20ВП1

Руководитель:
д.т.н., профессор _____ Кольчугина Е.А.

Работа защищена с оценкой _____
Преподаватели _____
_____
Дата защиты _____

2024

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Кафедра “Математическое обеспечение и применение ЭВМ”

“УТВЕРЖДАЮ”
Зав. кафедрой _____
“_____”_____20___ г.

ЗАДАНИЕ

на курсовое проектирование по курсу
” Программирование в сетях”

Студенту Левину Максиму Вячеславовичу Группа 20ВП1
Тема проекта CGI-приложение для удаленной работы с сервером FTP

Исходные данные (технические требования) на проектирование

1. Реализация осуществляется на IBM-совместимых ПЭВМ, операционная система семейств Windows либо Linux (с преимущественной ориентацией на линию Red Hat)
2. Разработать интерактивное CGI-приложение на языке Perl, которое:

-регистрирует пользователя на FTP-сервере;
-получает и выводит на экран содержимое текущего каталога сервера;

-осуществляет смену каталогов по запросу пользователя;
- получает файлы по запросу пользователя;
-выводит содержимое файлов на экран.
3. Адрес сервера, имя и пароль пользователя, наименования каталогов и файлов вводятся пользователем интерактивно с помощью форм.

Объем работы по курсу

1. Расчетная часть

1. Анализ требований
2. Проектирование
3. Кодирование
4. Отладка и тестирование

2. Графическая часть

Отсутствует

3. Экспериментальная часть

1. Подготовка набора тестовых данных
2. Отладка и тестирование программы на ЭВМ

Срок выполнения проекта по разделам

1. Анализ требований	к
2. Проектирование	к
3. Кодирование	к
4. Тестирование	к
5. Оформление пояснительной записки	к

Дата выдачи задания “____” _____
Дата защиты проекта “____” _____

Руководитель ___Кольчугина Е.А.____
Задание получил “____” _____
Студент _____Левин М.В._____

Реферат

Пояснительная записка содержит 54 листа, 35 рисунков, 3 таблицы, 6 использованных источника, 2 приложения.

ПРОГРАММИРОВАНИЕ В СЕТЯХ, COMMON GATEWAY INTERFACE, CGI, FILE TRANSFER PROTOCOL, FTP СЕРВЕР, FTP КЛИЕНТ, PERL, APACHE, HTML ДОКУМЕНТ.

Целью курсового проектирования является разработка CGI приложения для работы с удаленным FTP сервером

Разработка проводилась на языке программирования Perl в редакторе кода Visual Studio Code. В качестве веб-сервера был выбран Apache

Разработка проведена с использованием операционной системы Windows 10. Осуществлено функциональное тестирование разработанного приложения, которое показало корректность его работы

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном информационном обществе удаленная работа с серверами является неотъемлемой частью повседневной деятельности специалистов в области информационных технологий. CGI (Common Gateway Interface) - это стандарт, который позволяет веб-серверу запускать программы на стороне сервера для обработки запросов от клиентов.

Цель данной курсовой работы заключается в разработке CGI-приложения, которое обеспечивает возможность удаленного управления файлами на сервере FTP через веб-интерфейс. Для достижения поставленной цели необходимо решить следующие подзадачи:

- сформулировать и проанализировать требования, по результатам которых составить диаграмму вариантов использования;
- спроектировать структуры данных;
- спроектировать алгоритмы и составить следующие диаграммы: диаграмма деятельности, диаграмма последовательности, диаграмма коммуникации, диаграмма состояний;
- спроектировать пользовательский интерфейс;
- выполнить программную реализацию и составить диаграмму развертывания;
- протестировать разработанное приложение.

Разработка приложения будет производиться в операционной системе Windows 10 в редакторе кода Visual Studio Code с использованием следующих технологий:

- HTML и CSS (создание пользовательского интерфейса);
- язык программирования JavaScript (добавление интерактивности);
- язык программирования Perl 5 (создание CGI скриптов);

1. Постановка задачи и анализ требований

1.1. Постановка задачи

Разрабатываемое CGI-приложение будет представлять из себя FTP клиент - компьютерную программу для упрощенного доступа к FTP серверу. Оно должно предоставлять пользователю такие возможности как авторизация, просмотр содержимого в определенной директории, просмотр содержимого выбранного файла, загрузка и удаление файлов с FTP сервера.

Приложение будет состоять из списка CGI скриптов, которые работают только на веб-сервере, поэтому его необходимо установить на компьютер. В качестве него будет использоваться **Apache**, так как у него отличная производительность для динамического контента, который будет генерироваться CGI скриптами разрабатываемого приложения.

Так как CGI скрипты будут запускаться только во время запросов, то необходимо где-то хранить следующие данные: адрес FTP сервера, логин и пароль пользователя -, для того, чтобы не приходилось их указывать в каждом запросе. Для этого будет использоваться компактная встраиваемая СУБД – **SQLite**.

1.2. Анализ программного обеспечения аналогичного назначения

1.2.1. FTP клиент FileZilla

FileZilla - кроссплатформенный мультиязычный клиент, простой в использовании, поддерживающий такие базовые протоколы, как FTP, SFTP, FTPS и другие, располагающий к себе удобным интерфейсом с возможностью перетаскивания, поддержкой вкладок, сравнением директорий, синхронизацией и удаленным поиском [1]. Программа регулярно обновляется, что говорит об активном статусе ее разработки. Интерфейс приложения «FileZilla» представлен на рисунке 1.

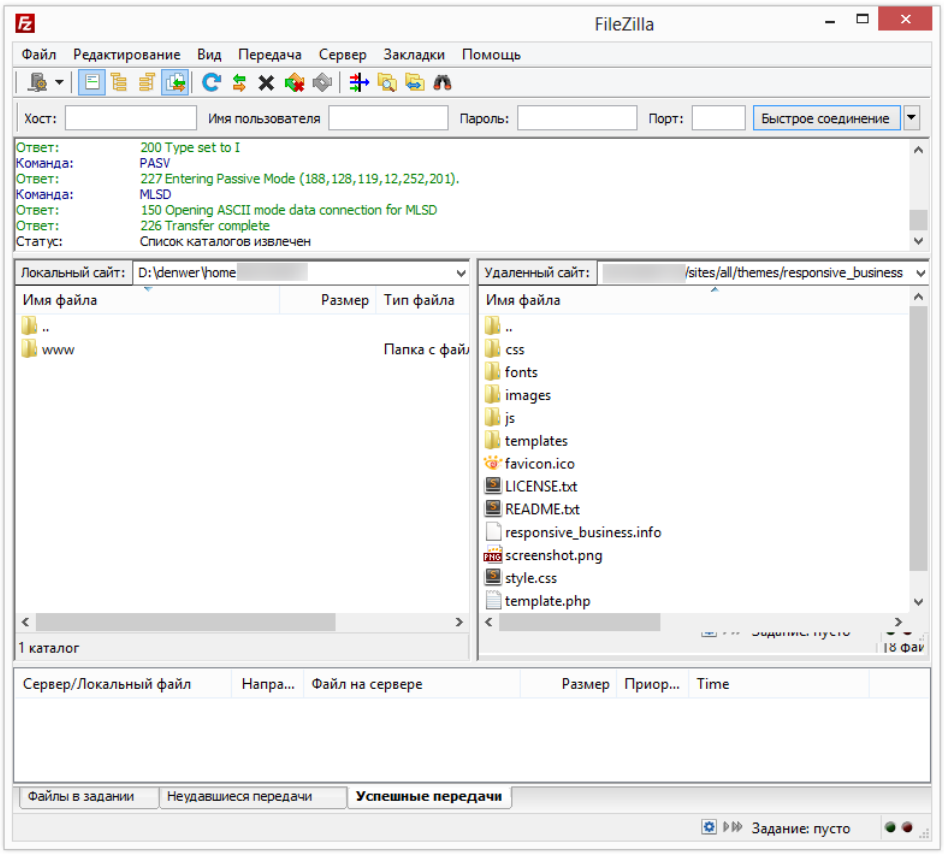

Рисунок 1 - Интерфейс FileZilla

Среди достоинств этого приложения можно выделить простой и удобный двухпанельный интерфейс, кроссплатформенность и наличие документация на русском языке, а также для протокола FTP дополнительно доступно шифрование. Недостатками этого приложения являются отсутствие поддержки командной строки и отсутствие шифрование паролей.

1.2.2. FTP клиент FTPRush

FTPRush - бесплатный кроссплатформенный FTP-клиент с поддержкой облачных хранилищ Google Drive, DropBox, OneDrive и Amazon S3, а также подключений по протоколу WebDAV. Интерфейс приложения «WinSCP» представлен на рисунке 2.

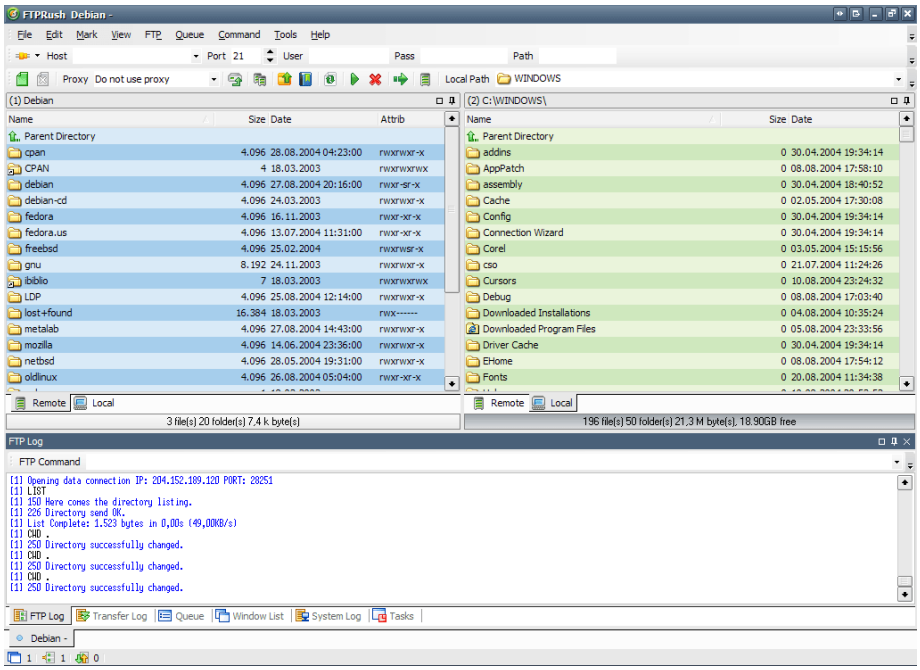

Рисунок 2 – Интерфейс FTPRush

Среди достоинств этого приложения можно выделить гибкий, настраиваемый двухпанельный интерфейс, кроссплатформенность, SSL/TLS/SFTP-шифрование, Z-компрессию при передаче. Также из примечательных особенностей можно отметить такую функцию, как FXP - перенос файлов с одного сайта на другой напрямую, не закачивая их на свой компьютер [2]. К недостаткам этого приложения можно отнести некачественный перевод интерфейса.

1.2.3. FTP клиент WinSCP

WinSCP - свободный графический клиент протоколов FTP, SFTP и SCP, предназначенный для Windows. Обеспечивает защищённое копирование файлов между компьютером и серверами. Интерфейс приложения «WinSCP» представлен на рисунке 3.

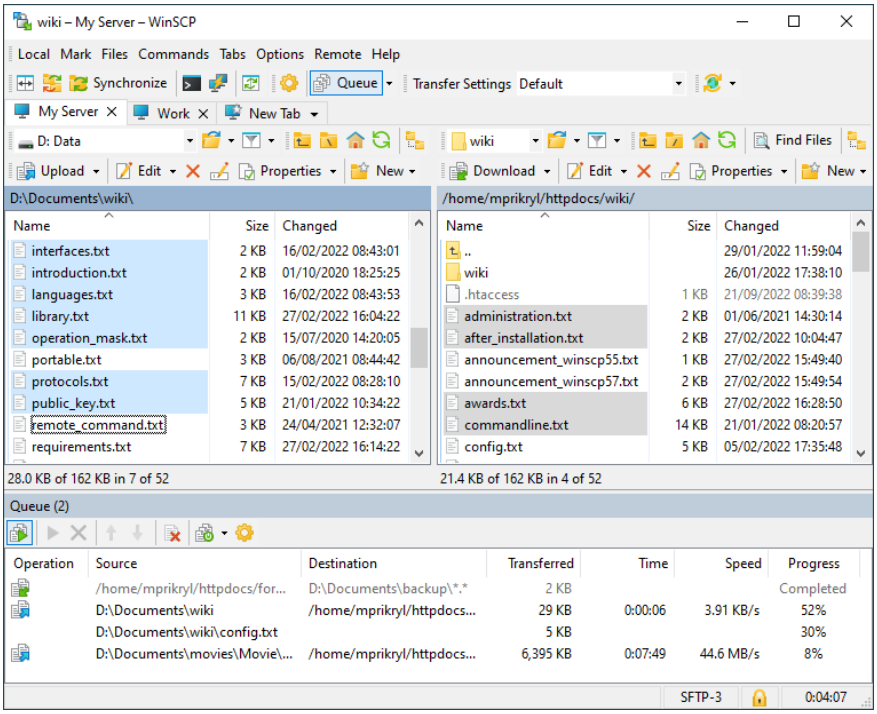

Рисунок 3 - Интерфейс WinSCP

К достоинствам приложения можно отнести возможность интеграции с популярной утилитой PuTTY, работу с протоколом как SSH-1, так и SSH-2, поддержку разных типов авторизации, мультиязычный интерфейс, поддержка скриптов для автоматизации передачи данных и планирование заданий в Windows через командную строку [3]. К недостаткам этого приложения можно отнести сложность использования у неопытных пользователей.

1.2.4. FTP клиент Cyberduck

Cyberduck - популярный инструмент с открытым исходным кодом для работы с облачными хранилищами и поддерживает следующие облачные службы и сетевые протоколы: WebDav, FTP, SFTP, Windows Azure Blob Storage, Backblaze B2 Coud Storage, Google Cloud Storage, Amazon S3, Dropbox, Google Drive, Microsoft OneDrive, Microsoft SharePoint [4]. Интерфейс приложения «Cyberduck» представлен на рисунке 4.

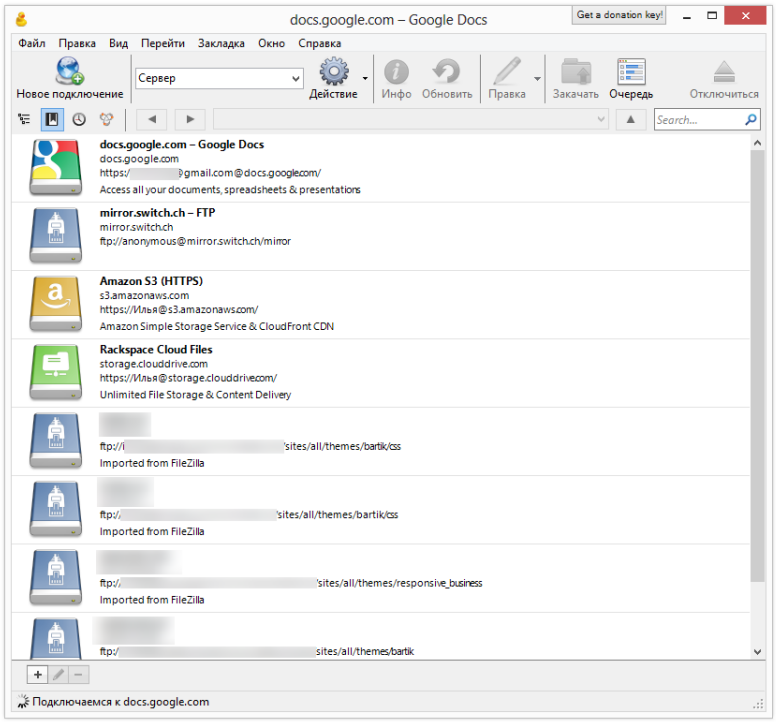

Рисунок 4 – Интерфейс Cyberduck

К достоинствам приложения можно отнести создание архивов, редактирование файлов в режиме реального времени, открытие одновременно нескольких соединений, ограничение скорости загрузки и скачивания. К недостаткам можно отнести неудобный файловый менеджер. Несмотря на поддержку FTP, Cyberduck сгодится лишь для закачки или скачивания файлов, просмотра документов и других простых операций. То есть предусмотрен самый базовый функционал с заявленными протоколами и сервисами.

1.2.5. Сравнение FTP клиентов

Проведем сравнительный анализ рассмотренных выше программных средств и разрабатываемого приложения. Результаты сравнения FTP клиентов представлены в таблице 1.

Таблица 1 - Сравнение программных средств аналогичного назначения

Критерий \ Приложение	FileZilla	FTPRush	WinScp	Cyberduck	Собственное приложение
Удобство интерфейса	3	3	2	1	3
Гибкая настройка интерфейса	0	1	1	0	0
Отсутствие функциональной избыточности	0	0	0	1	1

Продолжение таблицы 1

Бесплатный	1	1	1	1	1
Написание сценариев и автоматизация задач	0	0	1	0	0
Сжатия данных	0	1	1	0	0
Локализация	1	0	1	1	1
Безопасность и сохранность файлов	1	2	2	2	2
Поддержка ОС Windows	1	1	1	1	1
Поддержка UNIX подобных ОС	1	1	0	0	1
Поддержка ОС MacOS	1	1	0	1	1
ИТОГО	9	11	10	8	11

Среди всех продуктов-аналогов большее количество баллов набрал FTPRush и приложение, разрабатываемое в данной курсовой работе. Если выбирать наиболее подходящий FTP клиент для опытных пользователей, то лучшим вариантом для ОС Windows станет WinSCP, но если есть потребность в кроссплатформенности, то стоит выбрать FTPRush. Разрабатываемое приложение отлично подойдет для неопытных пользователей, которым требуется кроссплатформенность и минимально необходимые функции по работе с удаленным FTP сервером.

1.3. Формулировка и анализ требований к разрабатываемому программному обеспечению

К разрабатываемому приложению были выделены следующие функциональные требования:

- авторизация на FTP сервере;
- отключение от FTP сервера;
- просмотр содержимого выбранной директории;
- просмотр содержимого выбранного файла;
- переход в другую директорию;
- создание новой директории;
- скачивание файлов;
- загрузка файлов;

- удаление файлов.

В соответствии с выявленными требованиями была разработана диаграмма вариантов использования UML, представленная на рисунке 5 [5].

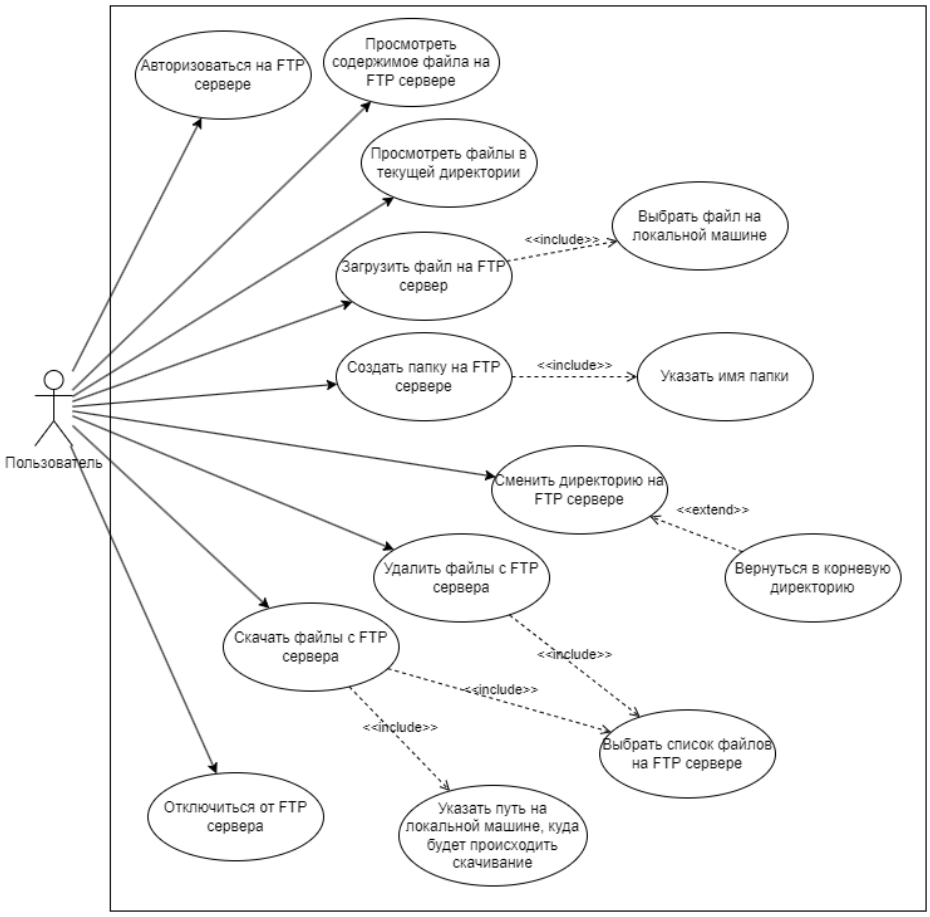

Рисунок 5 - Диаграмма вариантов использования

Сценарий варианта использования «Просмотреть содержимое файла на FTP сервере» представлен на рисунке 6.

Просмотреть содержимое файла на FTP сервере
Краткое описание: Вывод содержимого файла на экран
Действующие лица: Пользователь
Предусловие: Пользователь должен быть авторизован, а также должен быть выбран файл
Основной поток: 1. Пользователь переходит в нужную директорию 2. Выбирает файл и нажимает два раза по его названию 3. В соответствующий CGI скрипт передается путь до выбранного файла 4. CGI скрипт возвращает текст внутри выбранного файла 5. Полученный текст выводится на экран
Постусловие: На экран будет выведено содержимое файла

Рисунок 6 - Сценарий варианта использования «Просмотреть содержимое файла на FTP сервере»

Сценарий варианта использования «Авторизоваться на FTP сервере» представлен на рисунке 7.

Авторизоваться на FTP сервере
Краткое описание: Подключиться к FTP серверу для дальнейшей работы
Действующие лица: Пользователь
Предусловие: Пользователь должен быть не авторизован
Основной поток: 1. Пользователь вводит адрес FTP сервера, логин и пароль 2. Отправляется запрос на соответствующий CGI скрипт 3. CGI скрипт авторизует пользователя и возвращает положительный ответ (200 HTTP код)
Альтернативные потоки: 3.а. CGI скрипт не авторизует пользователя и возвращает ошибку, из-за невозможности создать соединение
Постусловие: Пользователь авторизован

Рисунок 7 - Сценарий варианта использования «Авторизоваться на FTP сервере»

Сценарий варианта использования «Скачать файлы с FTP сервера» представлен на рисунке 8.

Скачать файлы с FTP сервера
Краткое описание: скачать файлы с удаленного FTP сервера на локальную машину
Действующие лица: Пользователь
Предусловие: Пользователь должен быть авторизован
Основной поток: 1. Пользователь выбирает файлы, который он хочет скачать 2. Пользователь вводит путь на локальной машине 3. В соответствующий CGI скрипт передается массив путей выбранных файлов, и путь на локальной машине 4. CGI скрипт скачивает файлы
Альтернативные потоки: 4.а. CGI скрипт не скачивает файлы, а возвращает ошибку по причине отсутствия указанного пути на локальной машине
Постусловие: Файлы скачаны с удаленного FTP сервера

Рисунок 8 - Сценарий варианта использования «Скачать файлы с FTP сервера»

В результате анализа функциональных требований были выделены варианты использования программы. Для основных вариантов использования были представлены их спецификации, которые позволяют отобразить последовательность действий пользователя и системы.

2. Проектирование

2.1. Проектирование структур данных

Учетные данные пользователей будут храниться в базе данных (БД) SQLite в таблице «users», которая содержит 3 поля: идентификатор (id), логин (login) и пароль (password). Поле идентификатор имеет целочисленный тип данных и является первичным ключом. Поля логин и пароль имеют символьный тип данных, длину не более 30 символов и являются обязательными. На рисунке 9 показана схема базы данных.

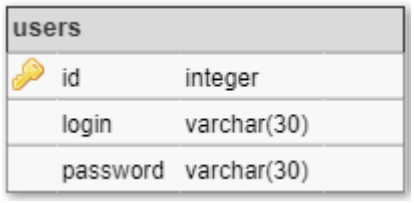

Рисунок 9 – Схема базы данных

После успешной авторизации пользователя на FTP сервере, приложение создает новую запись в БД или извлекает уже существующую и возвращает JSON объект с идентификатором пользователя. Полученный идентификатор, а также адрес FTP сервера сохраняются в куках браузера. Эти куки будут посылаться при каждом запросе.

В CGI скрипты данные передаются в формате JSON. Ответ от CGI скрипта передается либо так же в формате JSON, либо в формате HTML документа.

2.2. Проектирование алгоритмов

В разрабатываемом приложении можно выделить несколько состояний, в котором оно может находится. Все состояния и переходы между ними отображены на диаграмме состояний (рис. 10).

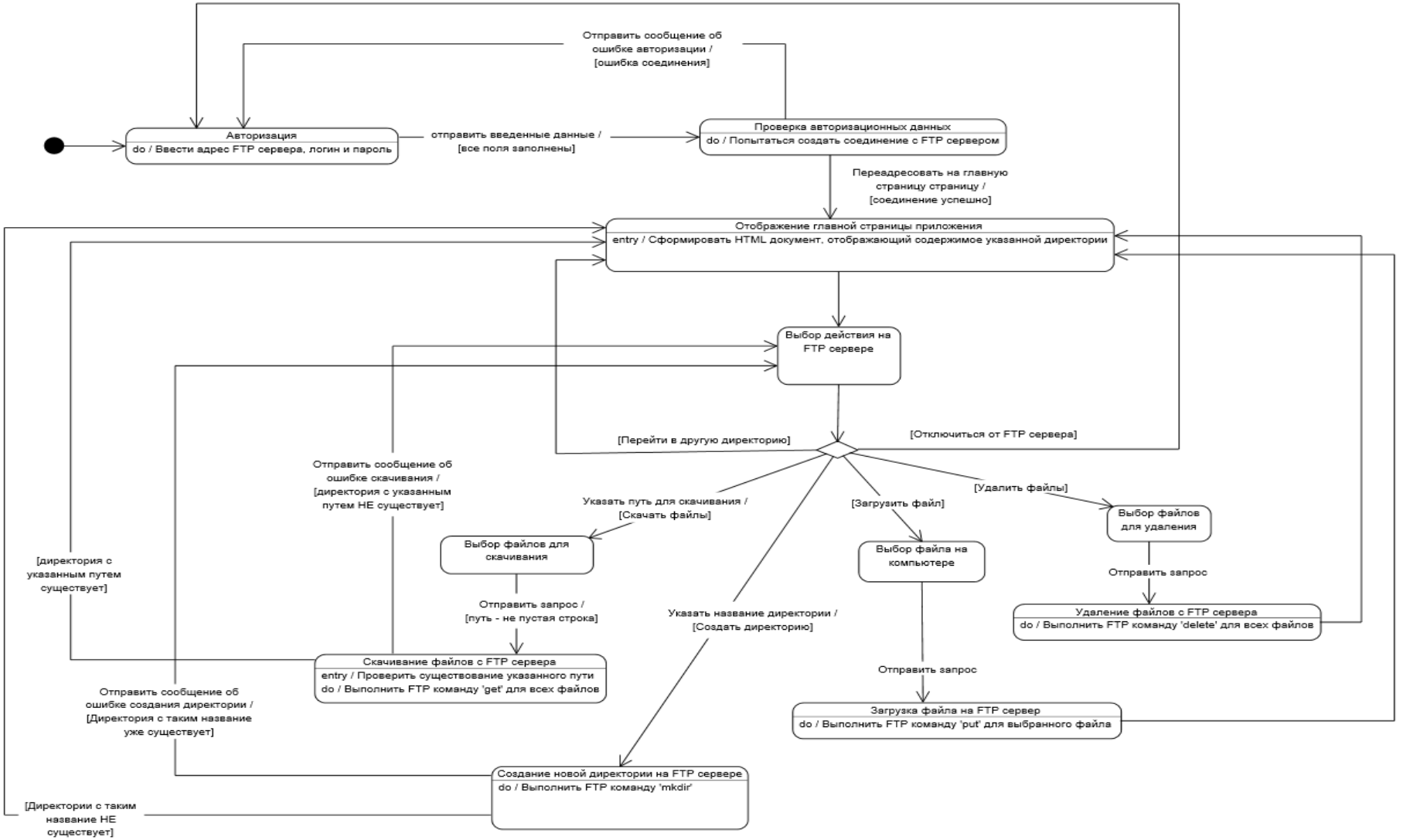

Рисунок 10 – Диаграмма состояний

В самом начале необходимо авторизоваться: ввести адрес FTP сервера, логин и пароль пользователя. Если все поля заполнены, то производится попытка создать соединение. Если произошла ошибка, то уведомляем об этом пользователя, иначе переадресуем его на главную страницу. Сразу после переадресации формируется HTML документ, отображающий содержимое корневой директории. Дальше пользователь выбирает действие, которое он хочет совершить на FTP сервере. При переходе в другую директорию, главная страница перезагружается, и отображается обновленный HTML документ, отображающий содержимое указанной директории. При отключении от FTP сервера, пользователя переадресует на форму авторизации. При выполнении таких операции как скачивание, загрузка и удаление файлов, а также создание новой директории, будет выполнен переход в одно из соответствующих состояний («выбор файлов для скачивания», «выбор файла на компьютере», «выбор файлов для удаления», «создание новой директории на FTP сервере»).

Для более подробного описания процесса авторизации была создана диаграмма последовательности (рис. 11).

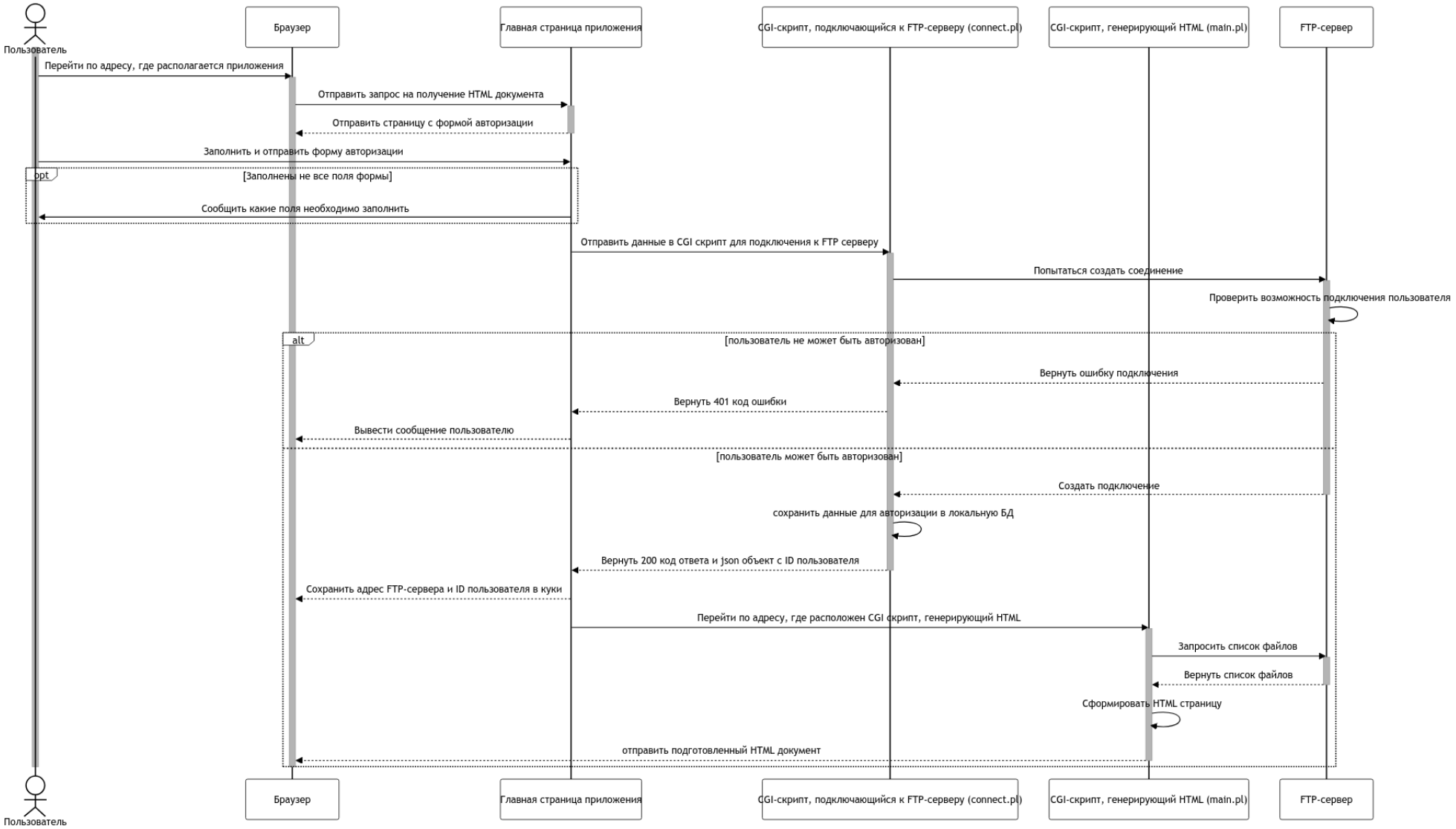

Рисунок 11 – Диаграмма последовательности

На диаграмме присутствует участник пользователь и следующие объекты:

- браузер;
- главная страница приложения;
- CGI скрипт, подключающийся к FTP серверу (connect.pl);
- CGI скрипт, генерирующий HTML (main.pl);
- FTP сервер.

В начале пользователь при помощи браузера переходит по ссылке, где расположено приложение. После он получает форму авторизации, которую заполняет. Если все поля формы заполнены, то данные отсылаются в CGI скрипт для создания подключения к FTP серверу (connect.pl). Если подключение не удалось, то скрипт возвращает ошибку авторизации, и пользователю отображается ошибка. Если подключение успешно, то скрипт сохраняет учетные данные в БД, и возвращает JSON объект с идентификатором пользователя. Полученный идентификатор и адрес FTP сервера сохраняются в куки браузера. Дальше происходит переход по адресу, где расположен CGI скрипт, генерирующий HTML (main.pl). Данный скрипт получает список файлов, содержащихся в корневой директории FTP сервера, и формирует HTML документ, после чего отсылает его пользователю.

Чтобы показать общее устройство разрабатываемой программы, структурно-функциональные взаимосвязи между модулями была создана диаграмма коммуникации (рис. 12).

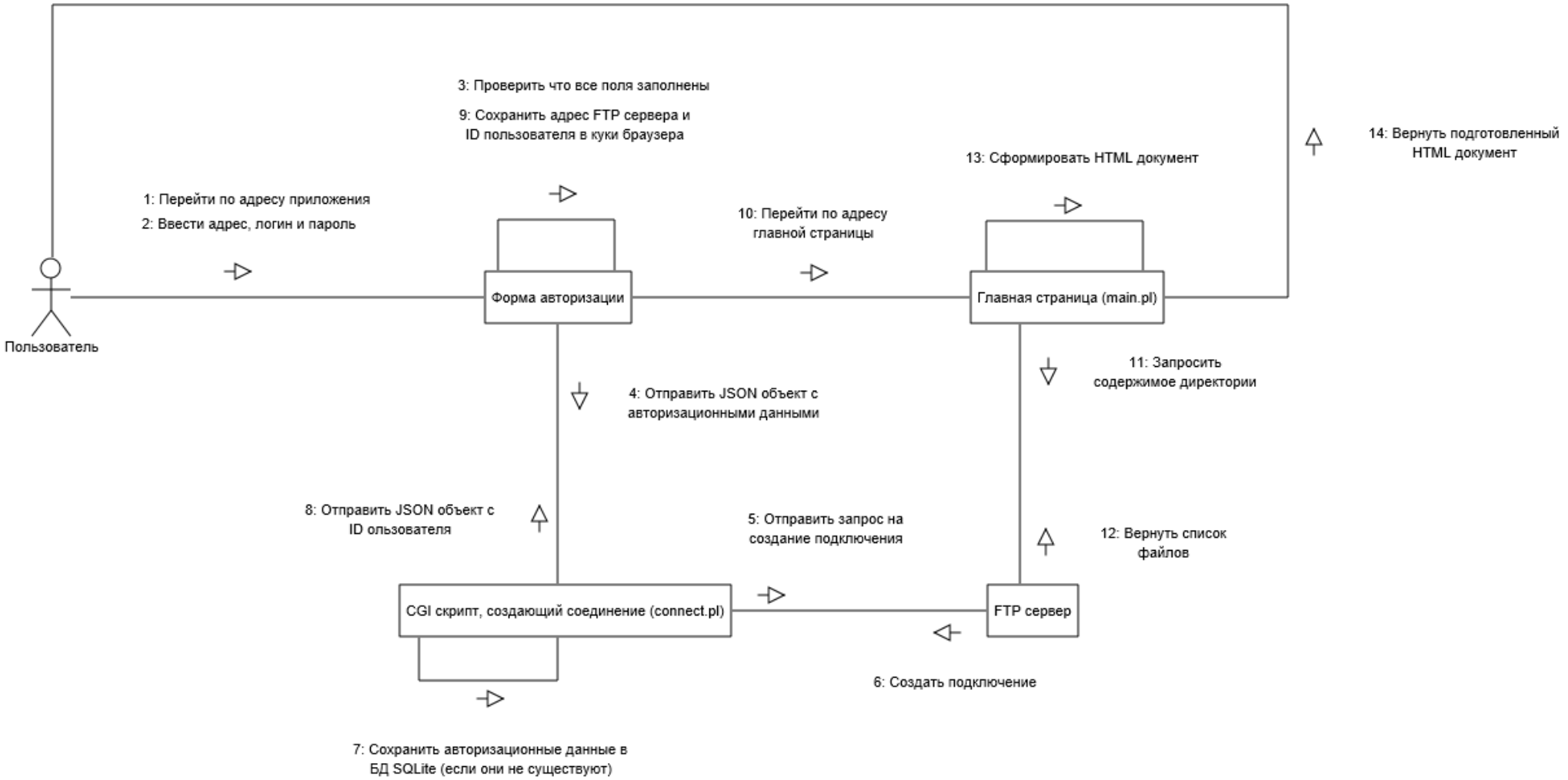

Рисунок 12 – Диаграмма коммуникации

Диаграмма коммуникации показывает во многом ту же информацию, что и диаграмма последовательности. Диаграмма коммуникаций нагляднее показывает, с какими элементами взаимодействует каждый элемент, а диаграмма последовательности яснее показывает в каком порядке происходят взаимодействия.

Для процесса формирования HTML документа, в котором отображается содержимое директории на FTP сервере, была разработана диаграмма деятельности (рис. 13).

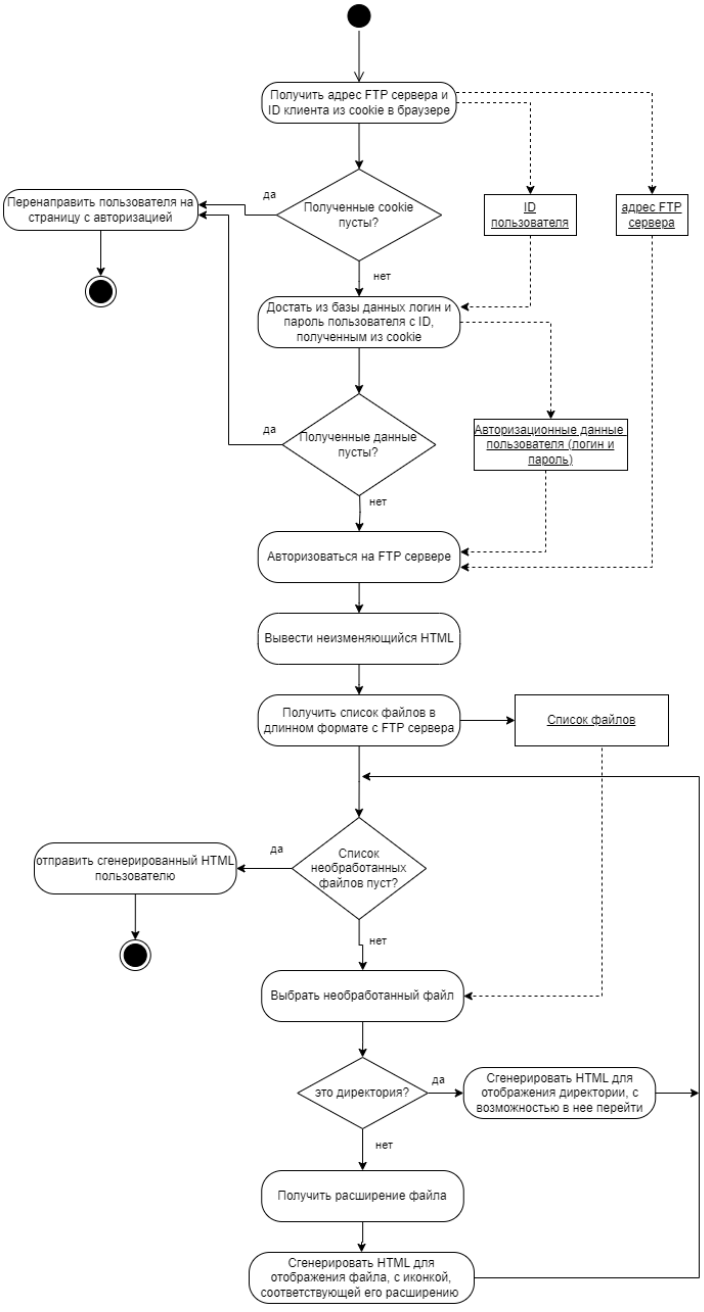

Рисунок 13 – Диаграмма деятельности

В начале проверяется прошел ли пользователь авторизацию. Извлекается адрес FTP сервера и идентификатор пользователя из кук браузера. Если полученные данные пусты, то завершаем выполнение, иначе получаем учетные данные пользователя из БД по идентификатору. Если данные отсутствуют в БД, то завершаем выполнение, иначе создаем подключение к FTP серверу. Дальше получаем список файлов и для каждого из них формируем соответствующий HTML. И наконец возвращаем сформированный документ пользователю.

2.3. Проектирование пользовательского интерфейса

Приложение будет состоять из двух HTML страниц. Первая будет представлять собой форму авторизации на FTP сервере (рис. 14), на которой содержатся следующие элементы:

1 – поле для ввода адреса FTP сервера;

2 – поле для ввода логина пользователя;

3 – поле для ввода пароля пользователя;

4 – кнопка «Подключиться».

В случае неверно введенных данных, пользователя необходимо об этом уведомить (рис. 15).

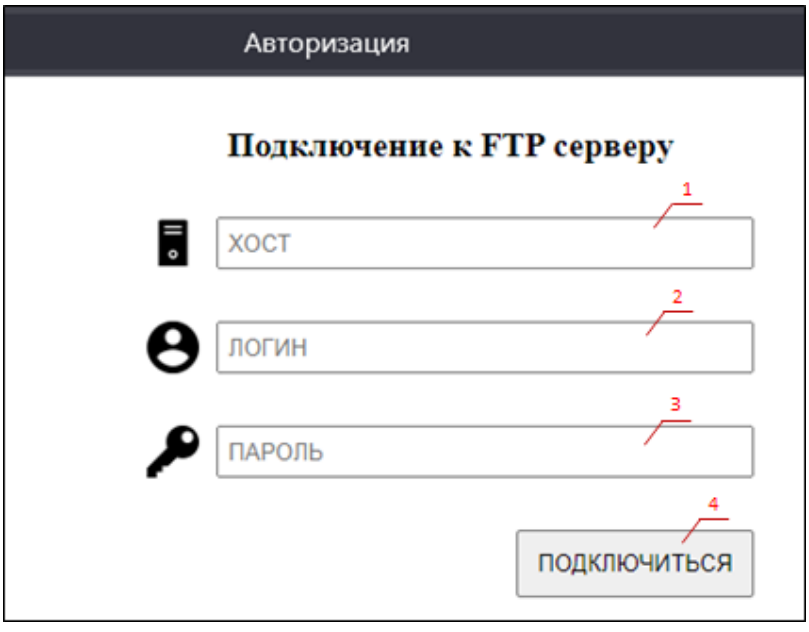

Рисунок 14 – Форма авторизации

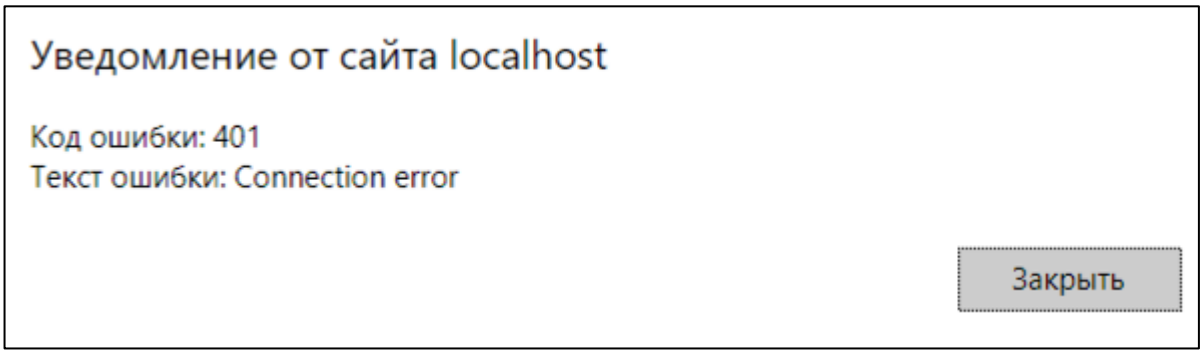

Рисунок 15 – Сообщение о неверно введенных данных

Вторая HTML страница будет динамически генерироваться при помощи CGI скрипта. Она представляет из себя интерфейс FTP клиента (рис. 16), на котором содержатся следующие элементы:

1 - адрес FTP сервера;

2 - логин пользователя;

3 – кнопка «Отключиться»;

4 - кнопка «Скачивание»;

5 - кнопка «Загрузка»;

6 - кнопка «Удаление»;

7 - кнопка «Создать директорию»;

8 - кнопка «Назад»;

9 - кнопка «Корневая директория»;

10 – поле с текущем путем;

11 – поле для вывода содержимого директории;

12 – поле для вывода размера файла;

13 – поле для вывода содержимого файла;

14 – поле, содержащее название выбранного файла.

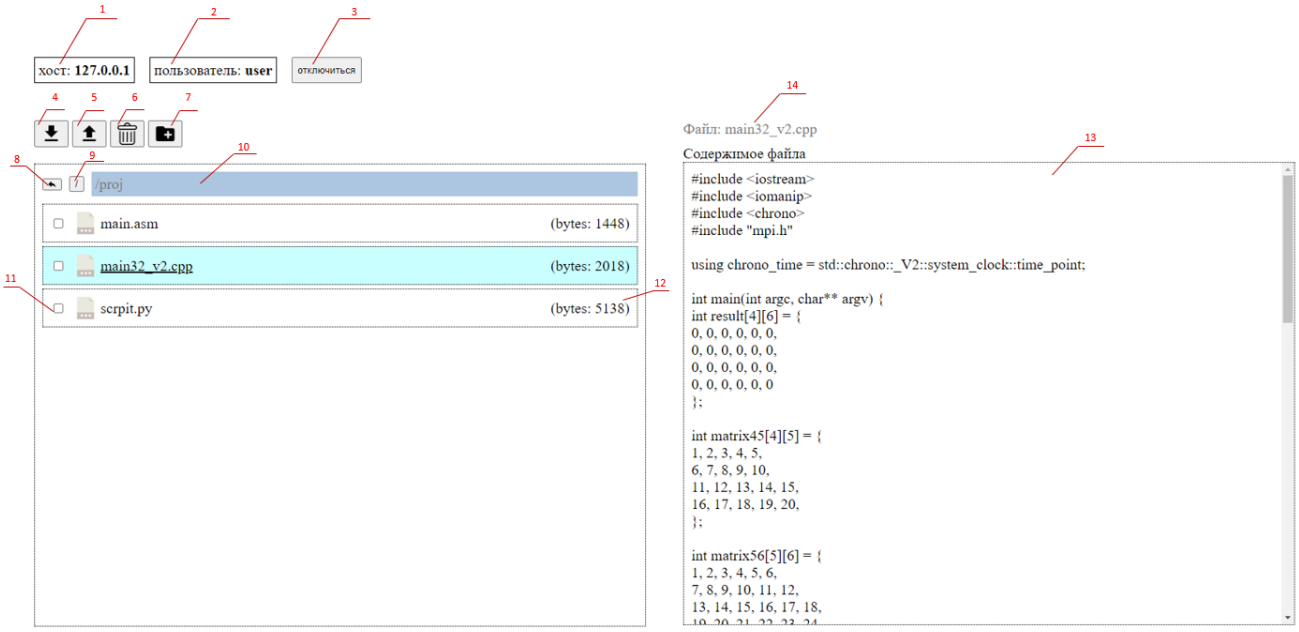

Рисунок 16 – Основная страница для работы с FTP сервером

При заходе в приложение пользователю будет отображена форма авторизации. Если авторизация пройдет успешно, то отобразится основная страница для работы с FTP сервером. Если пользователь захочет отключиться от сервера, то ему вновь отобразиться форма авторизации.

3. Кодирование

В результате выполнения поставленной задачи было разработано CGI-приложение для удаленной работы с сервером FTP, код которого приведен в приложении А. Описания разработанных программных единиц приведено в таблице 2.

Таблица 2 - Описания разработанных программных единиц

Название	**Тип программной единицы**	**Описание**
Common.pm	Perl пакет	Содержит функцию по созданию соединения с FTP сервером
DBUtil.pm	Perl пакет	Содержит функции по работе с sqlite базой данных
connect.pl	CGI скрипт	Авторизует пользователя на FTP сервере и сохраняет его в базе данных
main.pl	CGI скрипт	Генерирует HTML страницу, на которой располагается содержимое выбранной директории на FTP сервере
get.pl	CGI скрипт	Скачивает файлы с FTP сервера на локальную машину
get_content.pl	CGI скрипт	Возвращает содержимое файла с FTP сервера
put.pl	CGI скрипт	Загружает файл на FTP сервер
delete.pl	CGI скрипт	Удаляет файлы с FTP сервера
mkdir.pl	CGI скрипт	Создает директорию на FTP сервере
index.html	HTML документ	Содержит форму авторизации на FTP сервер
index.css	CSS	Содержит стили для формы авторизации
index.js	JS	Содержит функцию, которая отправляет введенные данные для подключения к FTP серверу в соответствующий CGI скрипт (connect.pl)
main.css	CSS	Содержит стили для основной страницы
main.js	JS	Содержит функции, которые отправляют запросы на выполнение операций на FTP сервер в соответствующие CGI скрипты (get.pl, put.pl, …)

Диаграмма развертывания для созданного приложения приведена на рисунке 17.

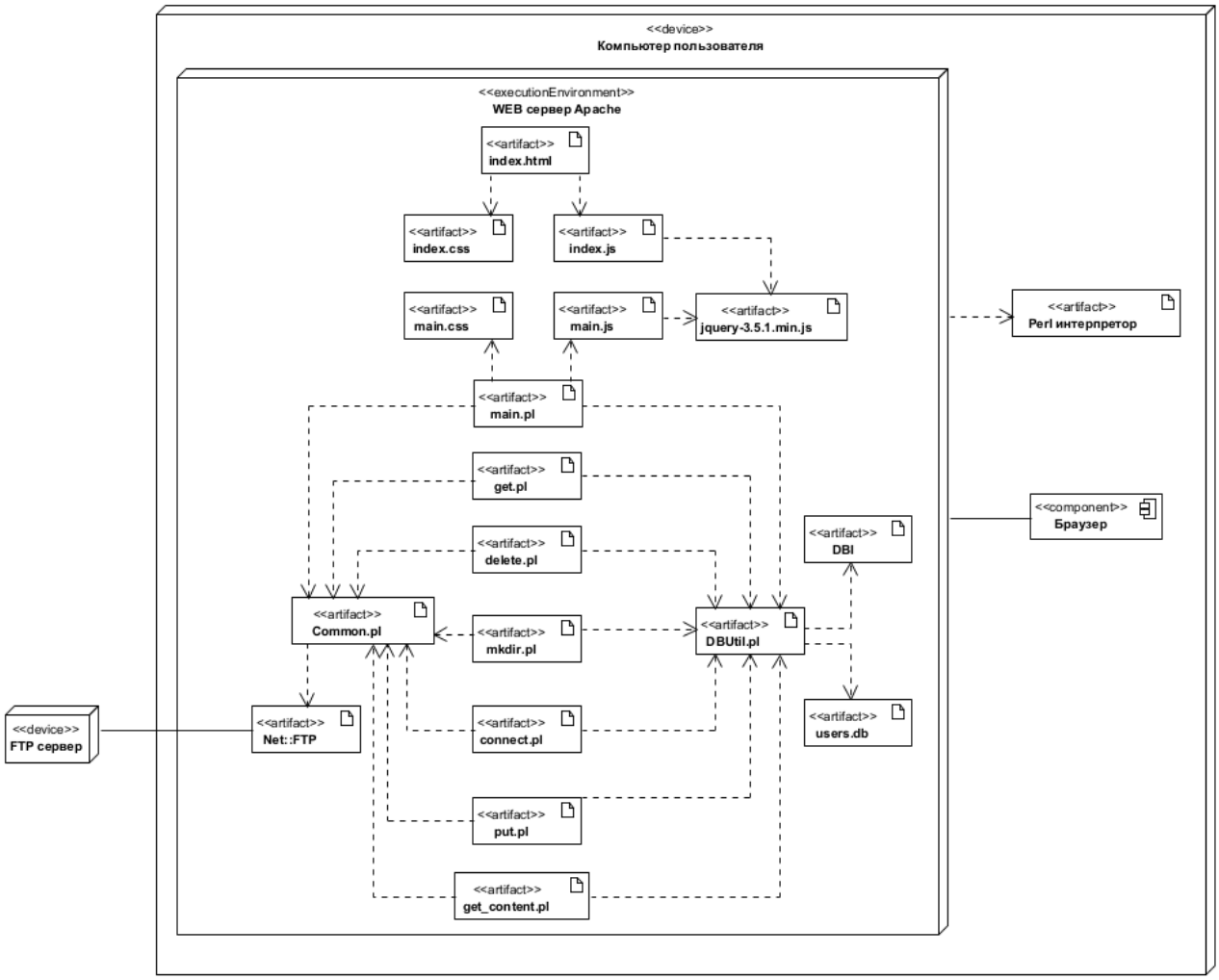

Рисунок 17 – Диаграмма развертывания

Для запуска приложения на компьютере должно быть уставлено следующее ПО: веб-сервер Apache, любой браузер и интерпретатор для языка программирования Perl. Исходный код необходимо разместить туда же, куда был установлен веб-сервер, в отдельную папку.

4. Отладка и тестирование

Функциональное тестирование - процесс обеспечения качества (QA) в рамках цикла разработки программного обеспечения, необходимый для проверки реализуемости функциональных требований, согласно спецификации тестируемого программного обеспечения. Оно проводится для оценки соответствия системы или компонента заданным функциональным требованиям. Функциональное тестирование проводится по принципу черного ящика, в связи с чем функциональность ПО можно протестировать, не зная принципа его внутренней работы. Это снижает требования к тестировщикам в части знания языков программирования или конкретных аспектов реализации программного обеспечения [6].

При решении поставленной задачи будет проведено функциональное тестирование, так как оно наиболее наглядно отображает корректность выполнения функций приложения, соответствующих функциональным требованиям.

В процессе разработки программы было проведено функциональное тестирование. Результаты тестирования представлены в таблице 3.

Таблица 3 - Результаты тестирования

Состав теста	**Ожидаемый результат**	**Наблюдаемый результат**	**Результат теста**
На форме авторизации оставить все поля пустыми и отправить запрос на подключение	Пользователя получит уведомление о том, имеются незаполненные поля	Появляется сообщение, что поля пусты (рис. Б.1)	Тест пройден
Указать на форме авторизации неверные данные для подключения	Ошибка подключения	Появляется сообщение, что возникла ошибка подключения (рис. Б.2)	Тест пройден
Перейти в другую директорию	На экран выведется содержимое директории, в которую был совершен переход	Содержимое текущей директории изменилось на содержимое другой директории (рис. Б.3)	Тест пройден

Продолжение таблицы 3

Загрузить файл на FTP сервер	На FTP сервере должен появиться новый файл	На FTP сервере появился новый файл (рис. Б.4 - Б.5)	Тест пройден
Удалить файл с FTP сервера	С FTP сервера должен удалиться файл	С FTP сервера удалился файл (рис. Б.6 - Б.7)	Тест пройден
Вывести содержимое файла, находящегося на FTP сервере	На экране выведется текст, содержащийся в выбранном файле	Содержимое файла вывелось на экран (рис. Б.8)	Тест пройден
Скачать файлы с FTP сервера	Выбранные файлы будут скачаны на локальную машину по указанному пути	Файлы скачались по указанному пути (рис. Б.9 - Б.11)	Тест пройден
При скачивании файлов указать несуществующую директорию	Файлы не будут скачаны, пользователь получит уведомление	Появляется сообщение о том, что указанная директория не существует (рис. Б.12 - Б.13)	Тест пройден
Создать директорию	На FTP сервере должна создаться новая директория	На FTP сервере создалась новая директория (рис. Б.14 - Б.15)	Тест пройден
Создать директорию с названием, которое используется другой директорией	Директория не должна пересоздаваться. Пользователь должен получить уведомление	Директория не пересоздалась и на экране появилось уведомление, сообщающее, что такая директорию уже существует (рис. Б16)	Тест пройден
Отключиться от сервера	Пользователя должен быть переадресован на форму авторизации	Пользователя переадресован на форму авторизации (рис. Б.17)	Тест пройден
Перейти по адресу, на котором располагается CGI скрипт, генерирующий HTML, минуя авторизацию	Переход должен быть недоступен, и должна вернуться ошибка авторизации	Страница недоступна. Возвращается ошибка с кодом 401 (Unauthorized) (рис. Б.18)	Тест пройден

В ходе выполнения тестирования несовпадения ожидаемого и наблюдаемого результата не выявлены. Следовательно, можно сделать вывод, что приложение работает корректно.

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной курсовой работы было разработано CGI-приложение, предназначенное для удаленной работы с сервером FTP. Реализация данного приложения позволяет пользователям управлять файлами на удаленном FTP-сервере через веб-интерфейс.

Выделены наиболее популярные в настоящее время FTP клиенты, для которых был проведен сравнительный анализ и выявлены их достоинства и недостатки.

Были сформулированы и проанализированы требования к разрабатываемому приложению. Создана диаграмма вариантов использования, описывающая основные функции.

На этапе проектирования был определен пользовательский интерфейс и были разработаны диаграммы, описывающие логику работы приложения:

- диаграмма состояний;
- диаграмма последовательности;
- диаграмма коммуникации;
- диаграмма деятельности.

На этапе кодирования была выполнена программная реализация приложения, а также создана диаграмма развертывания.

Произведено функциональное тестирование разработанного программного обеспечения. Все тесты пройдены успешно, ошибок не выявлено.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Информация о программе FileZilla [Электронный ресурс]. URL: https://www.filezilla.ru/ (дата обращения 28.03.2024).
2. Информация о программе FTPRush [Электронный ресурс]. URL: https://www.wftpserver.com/ftprush.htm (дата обращения 28.03.2024).
3. Информация о программе WinSCP [Электронный ресурс]. URL: https://winscp.net/eng/docs/lang:ru (дата обращения 28.03.2024).
4. Информация о программе Cyberduck [Электронный ресурс]. URL: https://cyberduck.io/ (дата обращения 28.03.2024).
5. Арлоу, Д. UML 2 и Унифицированный процесс. Практический объектно-ориентирован¬ный анализ и проектирование, 2-е издание / Д. Арлоу, А. Нейштадт – СПб: Символ – Плюс, 2007. – 624 с.
6. Лайза Криспин, Джанет Грегори. Гибкое тестирование: практическое руководство для тестировщиков ПО и гибких команд = Agile Testing: A Practical Guide for Testers and Agile Teams. — М.: «Вильямс», 2010. — 464 с. — (Addison-Wesley Signature Series).

ПРИЛОЖЕНИЕ А

(Исходный код приложения)

Листинг А.1 – Файл **./index.html**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="index.css">
    <script src="lib/jquery-3.5.1.min.js" defer></script>
    <script src="index.js" defer></script>
    <title>Авторизация</title>
</head>
<body>
    <div class="main">
        <div class="text">Подключение к FTP серверу</div>
        <div class="list_fields">
            <div class="field">
                <img src="icons/host.png" alt="host" height="30" width="30">
                <input type="text" placeholder="ХОСТ" id="input_host">
            </div>
            <div class="field">
                <img src="icons/user.png" alt="user" height="30" width="30">
                <input type="text" placeholder="ЛОГИН" id="input_user">
            </div>
            <div class="field">
                <img src="icons/password.png" alt="password" height="30"
width="30">
                <input type="password" placeholder="ПАРОЛЬ"
id="input_password">
            </div>
        </div>
        <div class="buttons">
            <button id="btn_connection">ПОДКЛЮЧИТЬСЯ</button>
        </div>
    </div>
</body>
</html>
```

Листинг А.2 – Файл **./index.css**

```
.main {
    width: 350px;
    padding: 20px;
    margin: 0 auto;
}

.text {
    margin-bottom: 30px;
    text-align: center;
    font-size: 20px;
    font-weight: bold;
}

.list_fields {
    margin-bottom: 30px;
}

.field {
    display: flex;
    margin-bottom: 30px;
}
```

```
.field > input {
    width: 100%;
    padding: 5px;
}

.field > img {
    margin-right: 10px;
}

.buttons {
    text-align: right;
}

.buttons button {
    padding: 10px;
}
```

Листинг А.3 – Файл **./index.js**

```
$(document).ready(function() {
    $("#btn_connection").click(function() {
        let host = $("#input_host").val();
        let user = $("#input_user").val();
        let password = $("#input_password").val();

        document.cookie = `host=${host}`;

        let postData = JSON.stringify({
            host: host,
            user: user,
            password: password,
        });

        $("#btn_connection").prop('disabled', true);

        $.ajax({
            url: "cgi-bin/connect.pl",
            method: "POST",
            headers: { "Content-Type": "text/plain" },
            data: postData,
            complete: () => $("#btn_connection").prop('disabled', false),
            error: (e) => alert(`Код ошибки: ${e.status}\nТекст ошибки:
${e.statusText}`),
            success: function(data) {
                let obj = JSON.parse(data);
                let userId = obj["id"];
                let username = obj["username"];
                document.cookie = `userId=${userId}`;
                document.cookie = `username=${username}`;
                window.location.href = "cgi-bin/main.pl";
            }
        })
    });
})
```

Листинг А.4 – Файл **./main.css**

```
html {
    font-size: 20px;
```

```
}

.head {
    display: flex;
    margin-bottom: 50px;
}

.host, .user {
    margin-right: 20px;
    border: 1px solid black;
    padding: 5px;
}

.buttons {
    margin-bottom: 20px;
}

.container {
    display: flex;
    gap: 50px;
}

.view_file_name {
    color: gray;
    margin-bottom: 10px;
}

.files_tree, .view_file_content {
    border: 1px dotted black;
    padding: 10px;
    overflow-y: auto;
    overflow-x: hidden;
    height: 600px;
    width: 800px;
}

.it_folder {
    font-weight: bold;
    user-select: none;
}

.it_folder:hover, .file_name:hover {
    text-decoration: underline;
}

.file:hover {
    cursor: pointer;
    background: rgba(202, 255, 255, 1);
}

.pwd {
    display: flex;
    align-items: center;
    margin-bottom: 10px;
}

.pwd > button {
    margin-right: 10px;
}

.pwd > div {
    width: 100%;
    color: gray;
```

```
    overflow-x: auto;
    padding: 5px;
    background-color: rgb(172, 198, 224);
}

.file {
    display: flex;
    align-items: center;
    margin-bottom: 5px;
    border: 1px dotted black;
    padding: 10px;
}

.file_size {
    margin-left: auto;
}

.file > input {
    margin-right: 15px;
}

.file > img {
    margin-right: 5px;
}
```

Листинг А.5 – Файл **./main.js**

```
const BASE_URL = "/perl_ftp";

$(document).ready(function() {
    let currentPath = $(".pwd > div").html();
    let prev_view_file;

    if (currentPath == "/") {
        currentPath = "";
    }

    // Обработчик для двойного нажатия на папку
    const divsFolders = document.querySelectorAll(".it_folder");
    for (const div of divsFolders) {
        div.addEventListener("dblclick", (e) => {
            let folderName = div.innerHTML;
            let path = `${currentPath}/${folderName}`;
            update(path);
        })
    }

    const files = document.querySelectorAll(".file > div:not(.it_folder)")
    for (let i = 0; i < files.length; i += 2) {
        const fileDiv = files[i];
        const sizeDiv = files[i+1];

        const myRe = /bytes: (\d+)/g;
        const myArray = myRe.exec(sizeDiv.innerHTML);
        const size = myArray[1] - '';

        fileDiv.addEventListener("dblclick", (e) => {
            let path = `${currentPath}/${fileDiv.innerHTML}`;
            if (path != prev_view_file) {
                prev_view_file = path;
                if (size < 10000) {
                    $.ajax({
```

```
                        url: "get_content.pl",
                        method: "POST",
                        headers: { "Content-Type": "text/plain" },
                        data: JSON.stringify({path: path}),
                        success: (res) => {
                            $(".view_file_name").html(`Файл:
${fileDiv.innerHTML}`);
                            $(".view_file_content").html(res);
                        }
                    });
                } else {
                    $(".view_file_name").html(`Файл: ${fileDiv.innerHTML}`);
                    const message = `<h2>Слишком большой файл
${sizeDiv.innerHTML}<br>Макс. кол-во байтов = 10000<h2>`;
                    $(".view_file_content").html(message);
                }
            }
        })
    }

    // Обработчик для поднятия наверх в файловой системе
    $("#btn_back").click(function() {
        let i = currentPath.lastIndexOf("/");
        const backPath = currentPath.substring(0, i);
        update(backPath);
    });

    $("#btn_root").click(function() {
        update("");
    });

    // Обработчик для скачивания файлов
    $("#btn_download").click(function() {
        const localPath = prompt("Путь на локальной машине");
        const { files, folders } = getSelectedItems();
        if (files.length != 0 || folders.length != 0) {
            const postData = {
                path: currentPath,
                localPath: localPath,
                files: files,
                folders: folders
            };
            $.ajax({
                url: "get.pl",
                method: "POST",
                headers: { "Content-Type": "text/plain" },
                data: JSON.stringify(postData),
                error: function() {
                    alert("Указанный путь не существует");
                },
                success: () => {
                    document.querySelectorAll(`input[type='checkbox']:checked
`)
                        .forEach(cb => cb.checked = false);
                    alert("Файлы скачаны");
                }
            });
        }
    });

    // Обработчик для удаления файлов
    $("#btn_delete").click(function() {
        const { files, folders } = getSelectedItems();
```

```
    if (files.length != 0 || folders.length != 0) {
        const postData = {
            path: currentPath,
            files: files,
            folders: folders
        };
        $.ajax({
            url: "delete.pl",
            method: "POST",
            headers: { "Content-Type": "text/plain" },
            data: JSON.stringify(postData),
            success: () => update(currentPath)
        });
    }
})

// Обработчики для загрузки файла
$("#btn_upload").click(() => $("#uploadForm_input_file").click());
$("#uploadForm_input_file").change(() => $("#uploadForm").submit());
$("#uploadForm").submit(e => {
    e.preventDefault();
    let formData = new FormData();
    formData.append("path", currentPath);
    formData.append("file", $("#uploadForm_input_file")[0].files[0]);
    $.ajax({
        url: "put.pl",
        method: "POST",
        processData: false,
        contentType: false,
        data: formData,
        success: () => update(currentPath)
    });
});

// Обработчики для создания папки
$("#btn_mkdir").click(function() {
    let folderName = prompt("Название папки");
    if (folderName == null || folderName == undefined) {
        return;
    }

    folderName = folderName.trim();
    if (folderName == "" || folderName.startsWith(".")) {
        alert("Некорректное имя для папки");
        return;
    }

    let fullPath = `${currentPath}/${folderName}`;

    $.ajax({
        url: "mkdir.pl",
        method: "POST",
        headers: { "Content-Type": "text/plain; charset=utf-8" },
        data: JSON.stringify({ path: fullPath }),
        success: () => update(currentPath)
    })
})

// Обработчики для отключения от ftp
$("#btn_disconnect").click(function() {
    deleteAllCookies();
    document.location.href = BASE_URL;
})
```

```
})

function update(path) {
    $("#updateForm_input_path").val(path);
    $("#updateForm").submit();
}

function getSelectedItems() {
    const files = [];
    const folders = [];
    const checkboxes =
document.querySelectorAll(`input[type='checkbox']:checked`);
    for (const checkbox of checkboxes) {
        const divFile = checkbox.parentNode;
        const nameFile = divFile.querySelector("div").innerHTML;
        if (checkbox.className == "cb_file") {
            files.push(nameFile);
        } else if (checkbox.className == "cb_folder") {
            folders.push(nameFile);
        }
    }

    return { files: files, folders: folders };
}

function deleteAllCookies() {
    let cookies = document.cookie.split("; ");
    for (let c = 0; c < cookies.length; c++) {
        let d = window.location.hostname.split(".");
        while (d.length > 0) {
            let cookieBase =
                encodeURIComponent(cookies[c].split(";")[0].split("=")[0])
                + '=; expires=Thu, 01-Jan-1970 00:00:01 GMT; domain='
                + d.join('.') + ' ;path=';

            let p = location.pathname.split('/');
            document.cookie = cookieBase + '/';
            while (p.length > 0) {
                document.cookie = cookieBase + p.join('/');
                p.pop();
            };
            d.shift();
        }
    }
}
```

Листинг А.6 – Файл **./cgi-bin/main.pl**

```
#!"C:\Dwimperl\perl\bin\perl.exe"
use CGI qw(:all);
use Encode qw(encode decode);
use CGI::Cookie;
use Net::FTP;
use JSON;
use Common;
use DBUtil;

sub getExt {
    my ($fileName) = @_;
    my @res = ($fileName =~ m/(.*)(\.)(.+)$/);
    return undef unless (@res);
```

```
    return @res[2];
}

my $cgi = CGI->new();

my %icons = (
    rar => "rar.png",
    zip => "zip.png",
    txt => "txt.png",
    docx => "docx.png",
    xlsx => "excel.png",
    jpg => "jpg.png",
    png => "png.png"
);

my %cookies = CGI::Cookie->fetch;
if (!defined($cookies{userId}) || !defined($cookies{host})) {
    print $cgi->header(
       -type=>'text/plain',
       -status=> '401 Coockies error'
    );
    exit;
}
my $host = $cookies{host}->value;
my $userId = $cookies{userId}->value;
my $username = $cookies{username}->value;

my ($id, $login, $password) = DBUtil::getUserById($userId);
my $ftp = Common::connectToFTPServer($host, $login, $password);
unless ($ftp) {
    print $cgi->header(
        -type=>'text/plain',
        -status=> '401 Connection error'
    );
    exit;
}

my $path = $cgi->param('path');

print $cgi->header(
    -type => 'text/html',
    -charset => 'utf-8',,
);

print $cgi->start_html({
    -title => 'ftp server',
    -head=>Link(
        {
            -rel=>'stylesheet',
            -type=>'text/css',
            -href=>'../main.css'
        }),
    -script => [
        {
            -type => 'text/javascript',
            -src => '../lib/jquery-3.5.1.min.js'
        },
        {
            -type => 'text/javascript',
            -src => '../main.js'
        }
    ],
});
```

```
print<<HTML;
<div class="head">
    <div class="host">хост: <span style="font-weight:
bold;">$host</span></div>
    <div class="user">пользователь: <span style="font-weight:
bold;">$login</span></div>
    <button id="btn_disconnect">отключиться</button>
</div>
<div class="main">
    <form action="main.pl" method="post" id="updateForm" hidden>
        <input type="text" id="updateForm_input_path" name="path">
    </form>
    <form action="put.pl" method="post" id="uploadForm" hidden
enctype="multipart/form-data">
        <input type="text" id="uploadForm_input_path" name="path">
        <input type="file" id="uploadForm_input_file" name="file">
    </form>
    <form action="mkdir.pl" method="post" id="mkdirForm" hidden>
        <input type="text" id="mkdirForm_input_path" name="path">
        <input type="text" id="mkdirForm_input_folderName" name="folderName">
    </form>
    <div class="container">
        <div class="files_manage">
            <div class="buttons">
                <button id="btn_download"><img src="../icons/download.png"
alt="download" height="30" width="30"></button>
                <button id="btn_upload"><img src="../icons/upload.png"
alt="upload" height="30" width="30"></button>
                <button id="btn_delete"><img src="../icons/delete.png"
alt="upload" height="30" width="30"></button>
                <button id="btn_mkdir"><img src="../icons/mkdir.png"
alt="mkdir" height="30" width="30"></button>
            </div>
            <div class="files_tree">
HTML

    my $backBtnDisabled = "disabled" unless $path;
    my $beginPath = $path;
    $beginPath = "/" unless $path;

print<<HTML;
                <div class='pwd'>
                    <button id='btn_back' $backBtnDisabled><img
src='../icons/back.png' alt='back' height='10' width='10'></button>
                    <button id='btn_root' $backBtnDisabled>/</button>
                    <div>$beginPath</div>
                </div>
HTML

    foreach my $f ($ftp->dir($path)) {
        $f =~ s/ +/ /gi;
        my @parts = split(/ /, $f);
        my $perm = @parts[0];
        my $name = join(" ", @parts[8..(scalar(@parts) - 1)]);
        my $size = @parts[4];
        my $ext = getExt($name);

        my $icon = $icons{$ext};
        $icon = "unknown_filetype.png" unless $icon;

        if ($perm =~ /^d/) {
            print "<div class='file'>";
```

```
            print "    <input type='checkbox' name='cb_folder'
class='cb_folder'>";
            print "    <img src='../icons/folder.png' alt='txt' height='30'
width='30'>";
            print "    <div class='it_folder'>$name</div>";
            print "</div>";
        } else {
            print "<div class='file'>";
            print "    <input type='checkbox' name='cb_file'
class='cb_file'>";
            print "    <img src='../icons/$icon' alt='txt' height='30'
width='30'>";
            print "    <div class='file_name'>$name</div>";
            print "    <div class='file_size'> (bytes: $size)</div>";
            print "</div>";
        }
    }

print<<HTML;
            </div>
        </div>
        <div class="view">
            <div class="view_file_name">Файл: </div>
            <caption>Содержимое файла</caption>
            <div class="view_file_content"></div>
        </div>
    </div>
</div>
HTML

print $cgi->end_html;

$ftp->quit;
```

Листинг А.7 – Файл **./cgi-bin/connect.pl**

```
#!"C:\Dwimperl\perl\bin\perl.exe"
use CGI qw(:all);
use Net::FTP;
use JSON;
use DBUtil;
use Common;

my $cgi = CGI->new();

my %data = %{JSON::decode_json($cgi->param('POSTDATA'))};

my $ftp = Common::connectToFTPServer($data{host}, $data{user},
$data{password});
unless ($ftp) {
    print $cgi->header(
        -type=>'text/plain',
        -status=> '401 Connection error'
    );
    exit;
}

my $db = DBUtil::create_connection();
my $query = "SELECT * FROM user WHERE login = '$data{user}' AND password =
'$data{password}'";
my $sth = $db->prepare($query);
```

```
$sth->execute();

my @existedUser = $sth->fetchrow_array();
my ($userId, $username);
unless(@existedUser) {
    my $query = "INSERT INTO user (login, password) VALUES ('$data{user}',
'$data{password}')";
    my $sth = $db->prepare($query);
    $sth->execute();

    $query = "SELECT last_insert_rowid()";
    $sth = $db->prepare($query);
    $sth->execute();

    my @lastId = $sth->fetchrow_array();
    ($userId, $username) = ($lastId[0], $data{user});
} else {
    ($userId, $username) = ($existedUser[0], $existedUser[1]);
}

print header('text/html','200 Success connect');
print JSON::encode_json({id => $userId, username => $username});
$ftp->quit;
```

Листинг А.8 – Файл **./cgi-bin/get.pl**

```
#!"C:\Dwimperl\perl\bin\perl.exe"
use FindBin qw( $RealBin );
use Encode;
use CGI qw(:all);
use CGI::Cookie;
use Net::FTP;
use JSON;
use Common;
use DBUtil;

my $cgi = CGI->new();

# достаем необходимые куки
my %cookies = CGI::Cookie->fetch;
my $userId = $cookies{userId}->value;
my $host = $cookies{host}->value;

my ($id, $login, $password) = DBUtil::getUserById($userId);

my $ftp = Common::connectToFTPServer($host, $login, $password);

my $postData = $cgi->param('POSTDATA');
exit unless $postData;
my %data = %{JSON::decode_json($postData)};
my $path = $data{path};
my $localPath = $data{localPath};
my @files = @{$data{files}};
my @folders = @{$data{folders}};

unless (-d $localPath) {
    print $cgi->header(
       -type=>'text/plain',
       -status=> '400 folder doesn\'t exist'
    );
    exit;
```

```
}

print $cgi->header(
    -type => 'text/html',
    -charset => 'utf-8',
);

$ftp->binary;
foreach my $file (@files) {
    my $remotePath = "$path/$file";
    my $localPath = "$localPath/$file";
    $ftp->get(encode_utf8($remotePath), $localPath);
}

foreach my $folder (@folders) {
    my $remotePath = "$path/$folder";
    my $localPath = "$localPath/$folder";

    mkdir($localPath);
    chdir($localPath);
    get_recursive(encode_utf8($remotePath));
    chdir($RealBin);
}

$ftp->quit;

sub get_recursive {
    my ($folder) = @_;
    foreach my $f ($ftp->dir($folder)) {
        $f =~ s/ +/ /gi;
        my @parts = split(/ /, $f);
        my $perm = @parts[0];
        my $name = join(" ", @parts[8..(scalar(@parts) - 1)]);
        if ($perm =~ /^d/) {
            mkdir($name);
            chdir($name);
            get_recursive("$folder/$name");
            chdir("..");
        } else {
            $ftp->get("$folder/$name", $name);
        }
    }
}
```

Листинг А.9 – Файл **./cgi-bin/put.pl**

```
#!"C:\Dwimperl\perl\bin\perl.exe"
use File::Basename;
use Encode;
use CGI qw(:all);
use CGI::Cookie;
use Net::FTP;
use JSON;
use Common;
use DBUtil;

my $cgi = CGI->new();

print $cgi->header(
    -type => 'text/html',
    -charset => 'utf-8',
```

```
);

# достаем необходимые куки
my %cookies = CGI::Cookie->fetch;
my $userId = $cookies{userId}->value;
my $host = $cookies{host}->value;

my ($id, $login, $password) = DBUtil::getUserById($userId);

my $ftp = Common::connectToFTPServer($host, $login, $password);

my $path = $cgi->param('path');
my $file = $cgi->param('file');
my $remotePath = "$path/$file";

$ftp->binary;
$ftp->put(\*$file, $remotePath) or print "error\n$!", $ftp->message;
$ftp->quit;
```

Листинг А.10 – Файл **./cgi-bin/mkdir.pl**

```
#!"C:\Dwimperl\perl\bin\perl.exe"
use CGI qw(:all);
use Encode;
use CGI::Cookie;
use Net::FTP;
use JSON;
use Common;
use DBUtil;

my $cgi = CGI->new();

print $cgi->header(
    -type => 'text/html',
    -charset => 'utf-8'
);

# достаем необходимые куки
my %cookies = CGI::Cookie->fetch;
my $host = $cookies{host}->value;
my $userId = $cookies{userId}->value;

my ($id, $login, $password) = DBUtil::getUserById($userId);

my $ftp = Common::connectToFTPServer($host, $login, $password);

my $postData = $cgi->param('POSTDATA');
exit unless $postData;
my %data = %{JSON::decode_json($postData)};
my $path = $data{path};

$ftp->mkdir(encode("utf8", $path));

$ftp->quit;
```

Листинг А.11 – Файл **./cgi-bin/get_content.pl**

```
#!"C:\Dwimperl\perl\bin\perl.exe"
use CGI qw(:all);
use HTML::Entities;
```

```
use Encode qw(encode decode);
use CGI::Cookie;
use Net::FTP;
use JSON;
use Common;
use DBUtil;

my $cgi = CGI->new();
my %cookies = CGI::Cookie->fetch;
my $host = $cookies{host}->value;
my $userId = $cookies{userId}->value;
my $username = $cookies{username}->value;

my ($id, $login, $password) = DBUtil::getUserById($userId);
my $ftp = Common::connectToFTPServer($host, $login, $password);

my $postData = $cgi->param('POSTDATA');
exit unless $postData;
my %data = %{JSON::decode_json($postData)};
my $path = $data{path};

my ($remote_file_content, $remote_file_handle);
open($remote_file_handle, '>', \$remote_file_content);

$ftp->get($path, $remote_file_handle) or die "get failed ", $ftp->message;

print $cgi->header(
    -type => 'text/html',
    -charset => 'utf-8',,
);

$ftp->quit;

$remote_file_content =~ s{>}{&gt;}g;
$remote_file_content =~ s{<}{&lt;}g;
$remote_file_content =~ s{\n}{<br>}g;
print $remote_file_content;
```

Листинг А.12 – Файл **./cgi-bin/delete.pl**

```
#!"C:\Dwimperl\perl\bin\perl.exe"
use CGI qw(:all);
use Encode;
use CGI::Cookie;
use Net::FTP;
use JSON;
use Common;
use DBUtil;

my $cgi = CGI->new();

print $cgi->header(
    -type => 'text/html',
    -charset => 'utf-8',
);

# достаем необходимые куки
my %cookies = CGI::Cookie->fetch;
my $userId = $cookies{userId}->value;
my $host = $cookies{host}->value;
```

```
my ($id, $login, $password) = DBUtil::getUserById($userId);

my $ftp = Common::connectToFTPServer($host, $login, $password);

my $postData = $cgi->param('POSTDATA');
exit unless $postData;
my %data = %{JSON::decode_json($postData)};
my $path = $data{path};
my @files = @{$data{files}};
my @folders = @{$data{folders}};

foreach my $file (@files) {
    my $remotePath = "$path/$file";
    $ftp->delete(encode_utf8($remotePath));
}

foreach my $folder (@folders) {
    my $remotePath = "$path/$folder";
    delete_recursive(encode_utf8($remotePath));
    $ftp->rmdir(encode_utf8($remotePath));
}

$ftp->quit;

sub delete_recursive {
    my ($folder) = @_;
    foreach my $f ($ftp->dir($folder)) {
        $f =~ s/ +/ /gi;
        my @parts = split(/ /, $f);
        my $perm = @parts[0];
        my $name = join(" ", @parts[8..(scalar(@parts) - 1)]);
        if ($perm =~ /^d/) {
            delete_recursive("$folder/$name");
            $ftp->rmdir("$folder/$name");
        } else {
            $ftp->delete("$folder/$name");
        }
    }
}
```

Листинг А.13 – Файл **./cgi-bin/DBUtil.pm**

```
#!"C:\Dwimperl\perl\bin\perl.exe"
package DBUtil;
use DBI;

sub create_connection{
    my $db = DBI->connect("DBI:SQLite:../users.db","","");
    $db->{sqlite_unicode} = 1;
    return $db;
}
sub getUserById {
    my ($userId) = @_;
    my $db = create_connection();
    my $query = "SELECT * FROM user WHERE id = '$userId'";
    my $sth = $db->prepare($query);
    $sth->execute();
    my @user = $sth->fetchrow_array();
    return @user;
}
1;
```

Листинг А.14 – Файл **./cgi-bin/Common.pm**

```
#!"C:\Dwimperl\perl\bin\perl.exe"
package Common;
use Net::FTP;
use Encode qw(encode decode);

sub connectToFTPServer {
    my ($host, $login, $password) = @_;

    my $ftp = Net::FTP->new($host, Timeout => 2);
    return undef unless $ftp;

    my $successLogin = $ftp->login($login, $password);
    return undef unless $successLogin;

    return $ftp;
}

1;
```

ПРИЛОЖЕНИЕ Б

(Скриншоты тестирования)

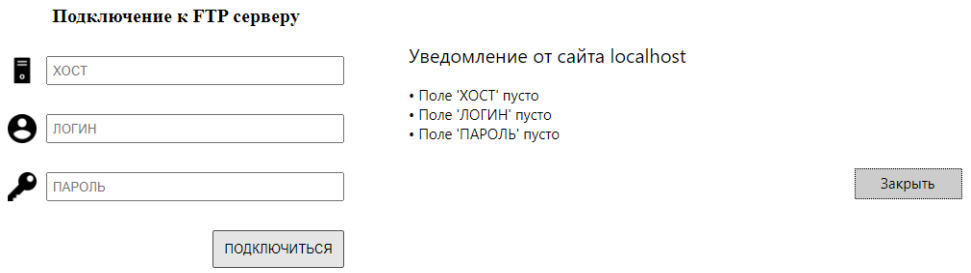

Рисунок Б.1 – Не заполнены поля формы авторизации

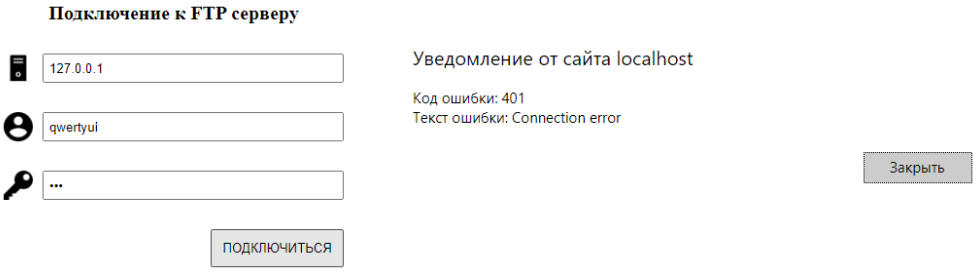

Рисунок Б.2 – Неверно указаны данные для подключения

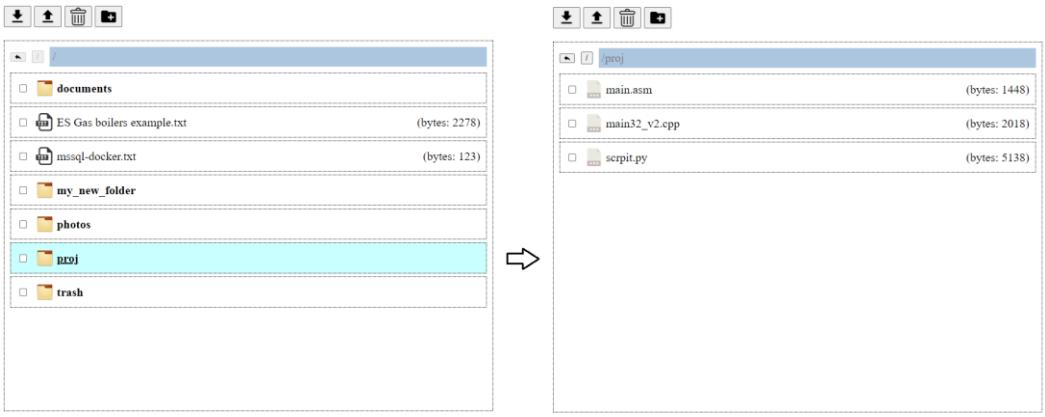

Рисунок Б.3 – Изменение директории

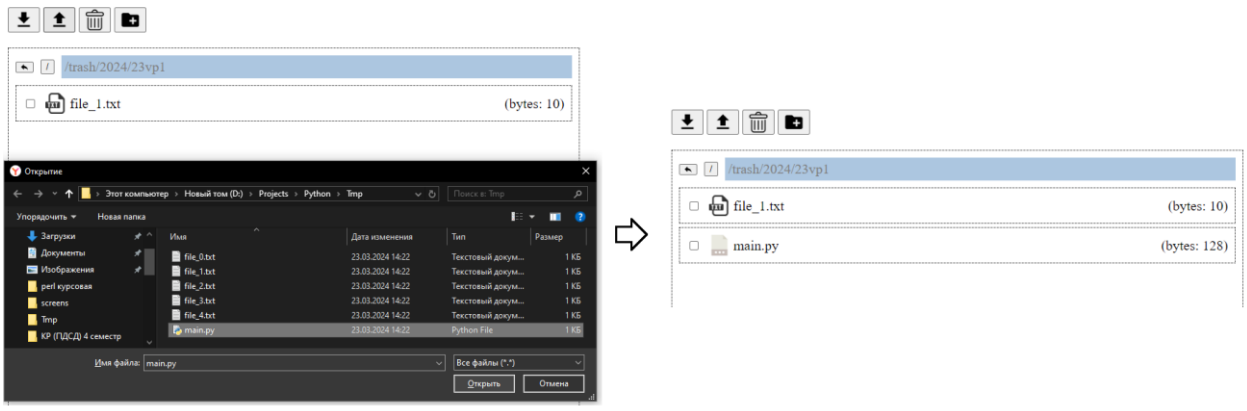

Рисунок Б.4 – Успешная загрузка файла

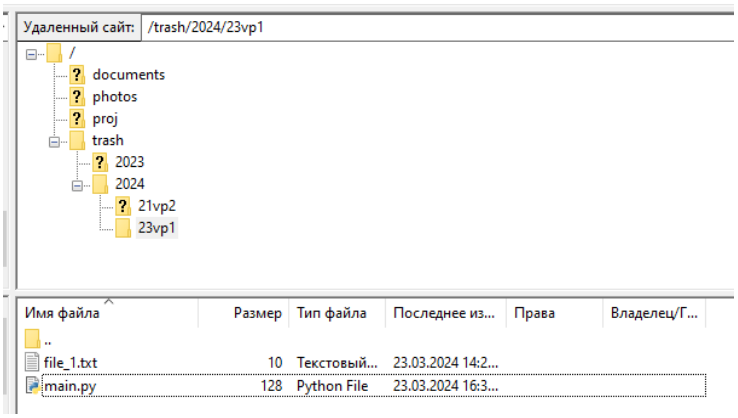

Рисунок Б.5 – Проверка успешности загрузки файла при помощи FileZilla

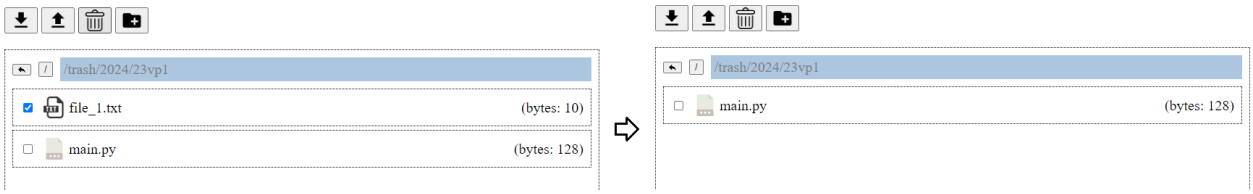

Рисунок Б.6 – Успешное удаление файла

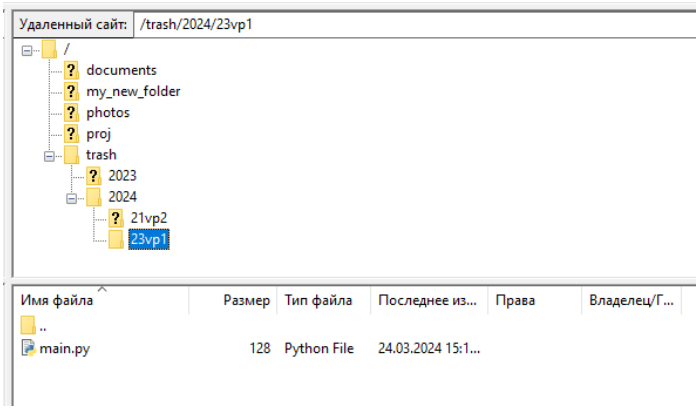

Рисунок Б.7 – Проверка успешности удаления файла при помощи FileZilla

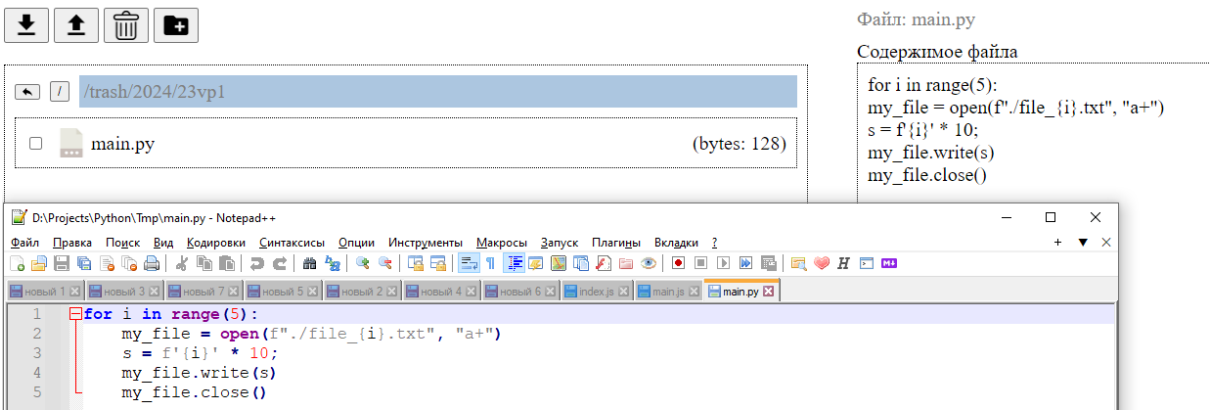

Рисунок Б.8 – Вывод содержимого файла на FTP сервере

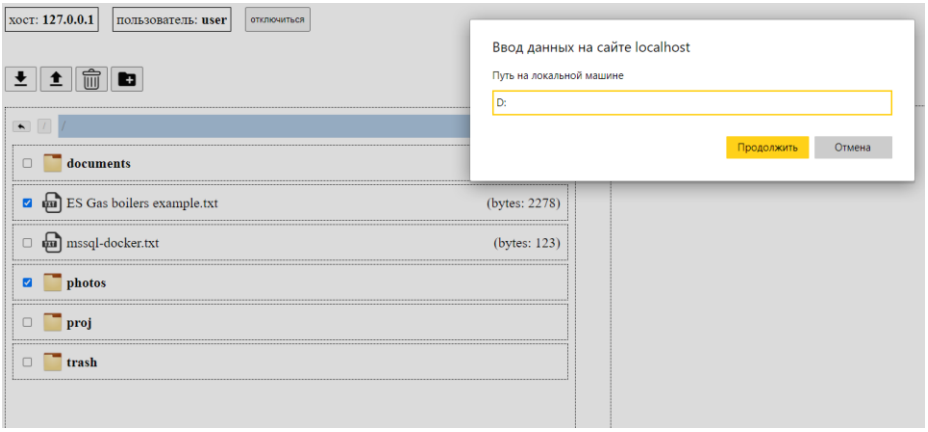

Рисунок Б.9 Выбор файлов для скачивания и указание пути

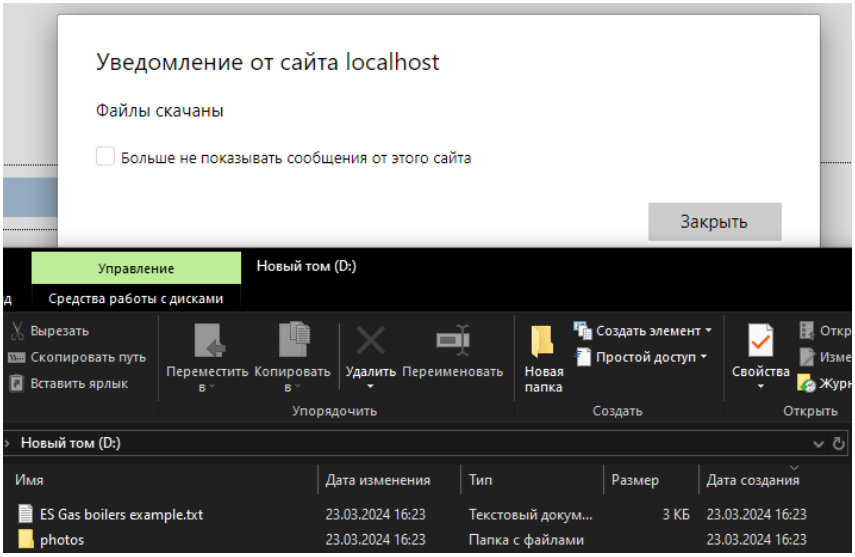

Рисунок Б.10 Выбранные файлы успешно скачались на локальную машину

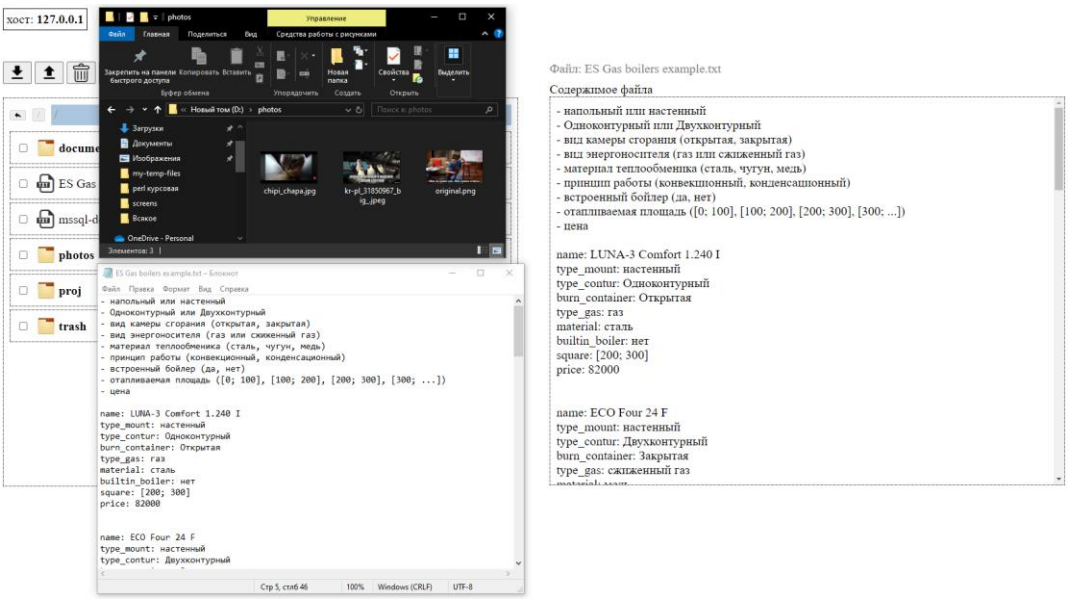

Рисунок Б.11 Проверка скаченных файлов на соответствие содержимого

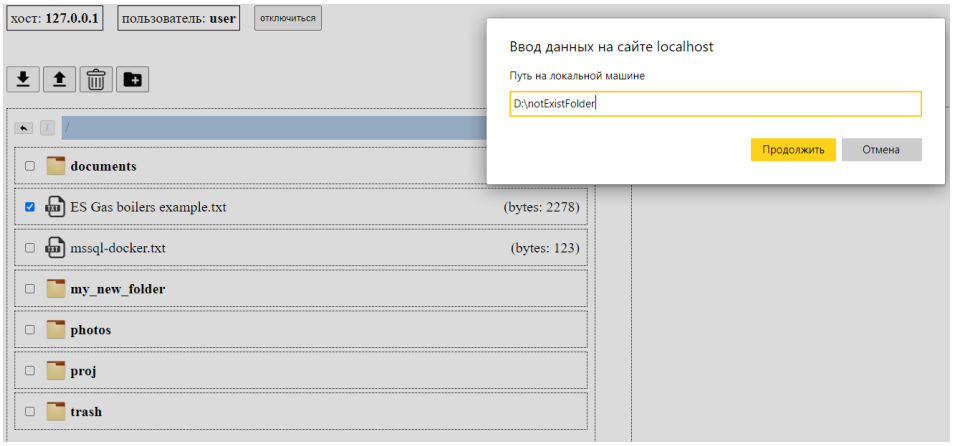

Рисунок Б.12 – Попытка скачать файл в несуществующую директорию

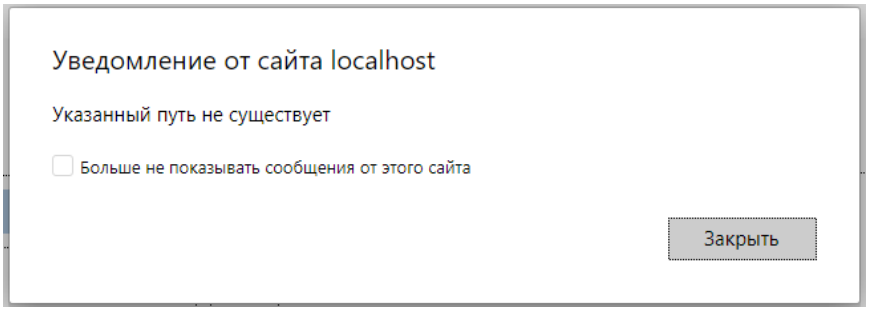

Рисунок Б.13 – Сообщение о том, что указанная директория не существует

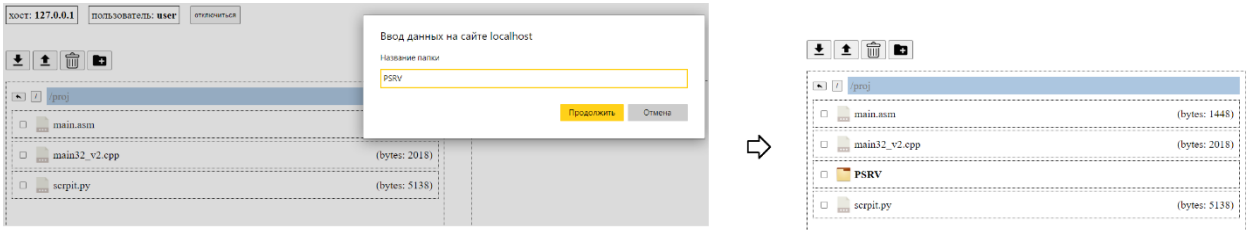

Рисунок Б.14 – Успешное создание директории

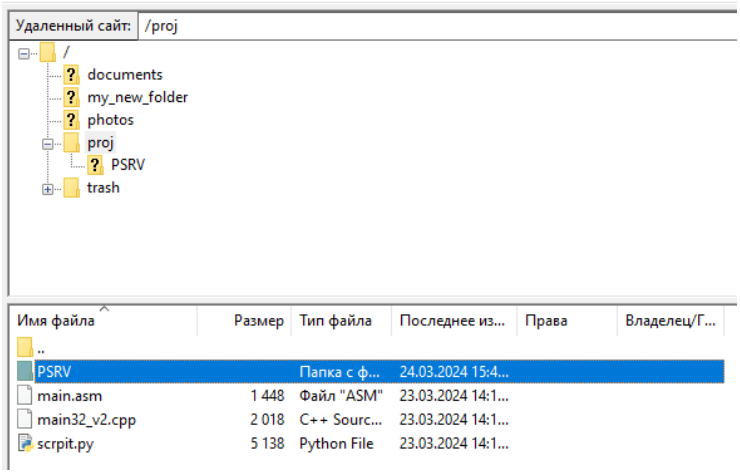

Рисунок Б.15 – Проверка успешности создания директории

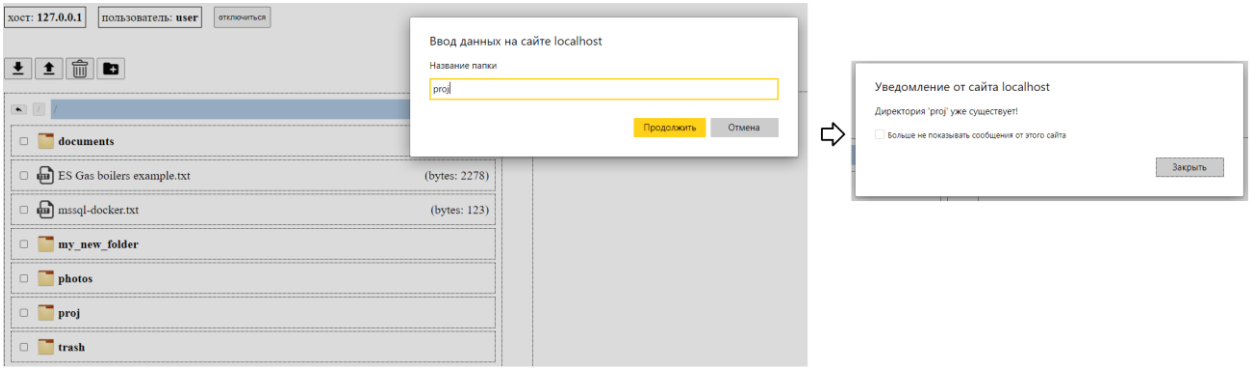

Рисунок Б.16 – Попытка создать директорию с названием, которое уже занято уже существующей директорией

Рисунок Б.17 – Переадресация на форму авторизации после отключения от FTP сервера

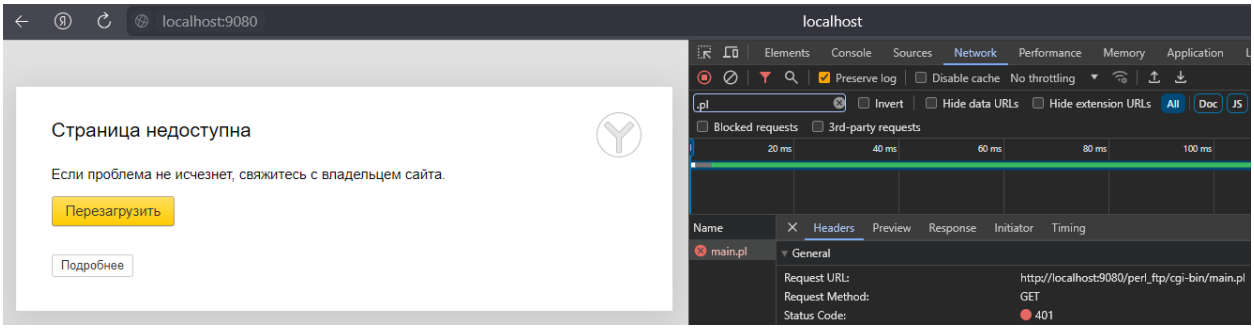

Рисунок Б.18 – Попытка получить доступ у неавторизированного пользователя