УДК 004

Создание FTP-клиента на C#

Ульянов Егор Андреевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Беликов Андрей Геннадьевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В данной статье рассматривается и описывается разработка FTP-клиента на языке программирования C#. Практическим результатом является разработанный клиент.
Ключевые слова: FTP-клиент, C#, Visual Studio

Creating an FTP client in C#

Ulianov Egor Andreevich
Sholom-Aleichem Priamursky State University
Student

Belikov Andrey Gennadievich
Sholom-Aleichem Priamursky State University
Student

Abstract
This article discusses and describes the development of an FTP client in the C# programming language. The practical result is a developed client.
Keywords: FTP client, C#, Visual Studio

FTP — протокол передачи файлов по сети, появившийся в 1971 году задолго до HTTP и даже до TCP/IP, благодаря чему является одним из старейших прикладных протоколов. Изначально FTP работал поверх протокола NCP, на сегодняшний день широко используется для распространения ПО и доступа к удалённым хостам.

FTP построен на архитектуре модели клиент-сервер и использует отдельные соединения для управления и передачи данных между клиентом и сервером. Пользователи FTP могут аутентифицировать себя с помощью открытого протокола входа в систему, обычно в форме имени пользователя и пароля, но могут подключаться анонимно, если сервер настроен на это.

Для безопасной передачи, которая защищает имя пользователя и пароль и шифрует содержимое, FTP часто защищается с помощью SSL/TLS (FTPS). Вместо этого иногда также используется протокол передачи файлов SSH (SFTP), но он технологически отличается.

Целью данной статьи является создание простого FTP-клиента на C# в среде разработки «Visual Studio» на языке программирования C#.

В своей работе Н. Н. Додобоев, О. И. Кукарцева, Я. А. Тынченко рассмотрели вопросы появления различных языков программирования (в частности C#), определения особенностей этих языков, а также составления основных видов и классификаций языков программирования[1]. З. С. Магомадова рассмотрела языки программирования высокого уровня, особенности, недостатки и сложности в изучении, а также описала несколько легких алгоритмов [2]. В своей работе Н.Г. Авдеев, С.А. Ткачёв, А.В. Борисов создали и описали клиент-серверного приложения на основе протокола FTP [3]. В статье G.R. Notess содержится информация об использовании протокола передачи файлов (FTP), интернет-механизма для копирования файлов [4].

Создаем проект «Windows Forms App» и называем его FTPClient см. рисунок 1.

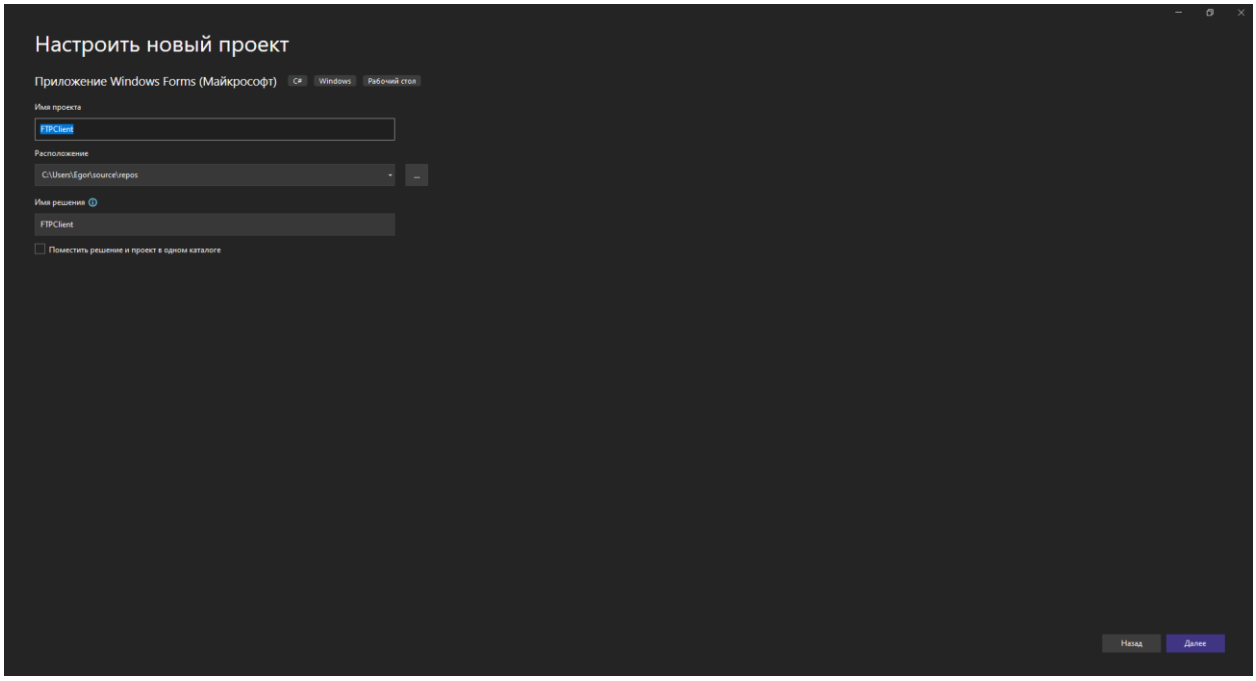

Рис. 1. Создание проекта

Далее необходимо добавить элементы в проект: поля ввода, маркеры, «ProgressBar», «BackgroundWorker» и кнопку загрузки. Для этого из окна «Панель элементов» перетаскиваем нужные элементы и располагаем как на рисунке 2.

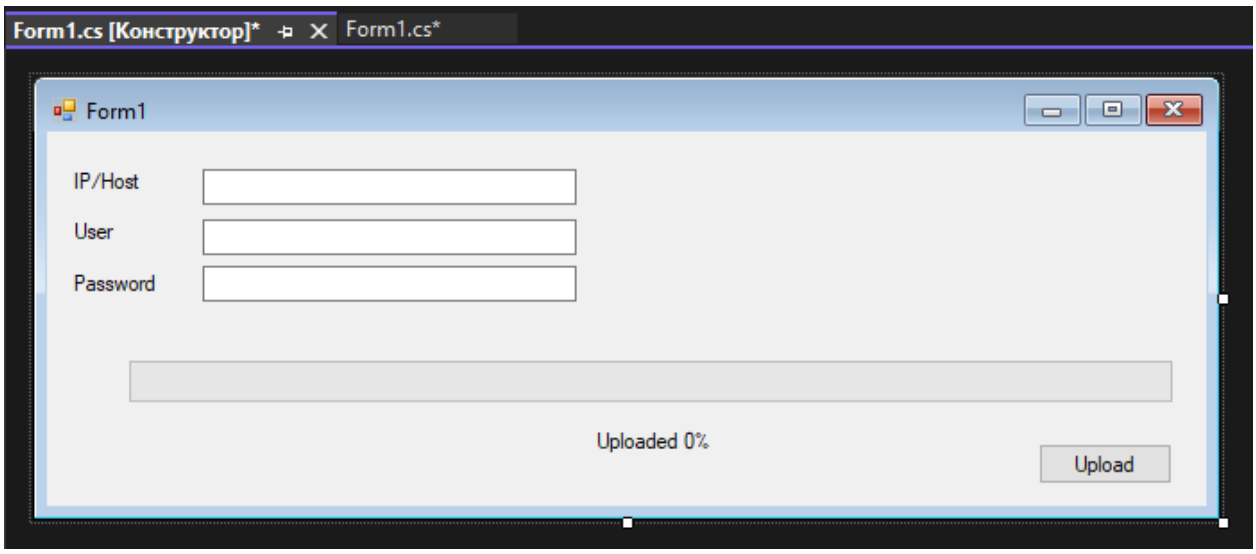

Рис. 2. Расположение элементов

В свойствах элементов меняем значение в пункте «Name» на «txtServe», «txtUser», «txtPass» соответственно см. рисунок 3-7.

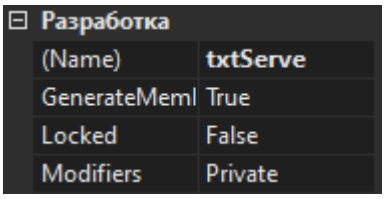

Рис. 3. Название элемента «TextBox»

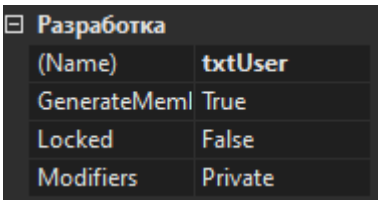

Рис. 4. Название элемента «TextBox»

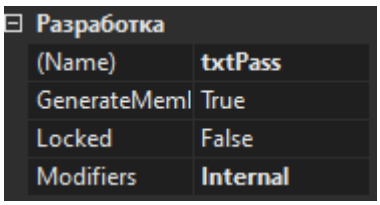

Рис. 5. Название элемента «TextBox»

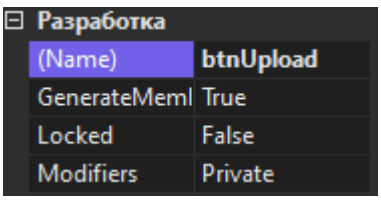

Рис. 6. Название элемента «Button»

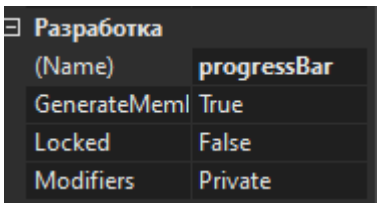

Рис. 7. Название элемента «ProgressBar»

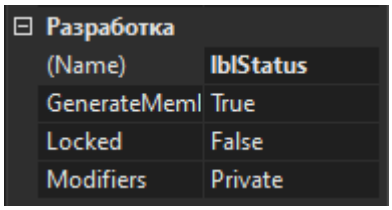

Рис. 8. Название элемента «Label»

Для отображения количество загруженного в процентах необходимо настроить элемент «BackgroundWorker» см. рисунок 9.

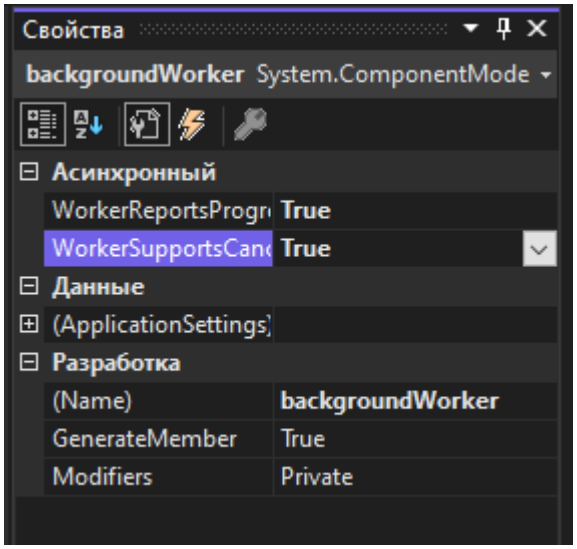

Рис. 8. Свойства элемента «BackgroundWorker»

Далее переходим во вкладку события, и кликаем два раза по каждому событию «DoWork», «ProgressChanged», «RunWorkerComplete» для создания событий в коде см. рисунок 10.

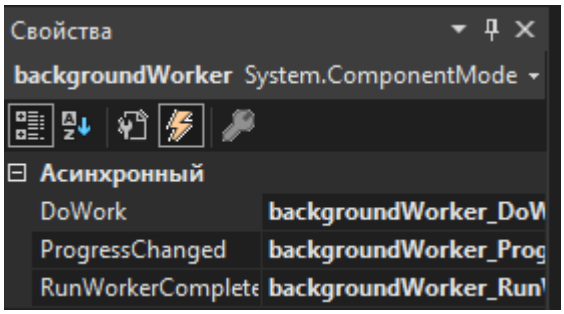

Рис. 10. Создание событий

Приступаем к написанию кода, для начала подключим необходимые библиотеки см. рисунок 11.

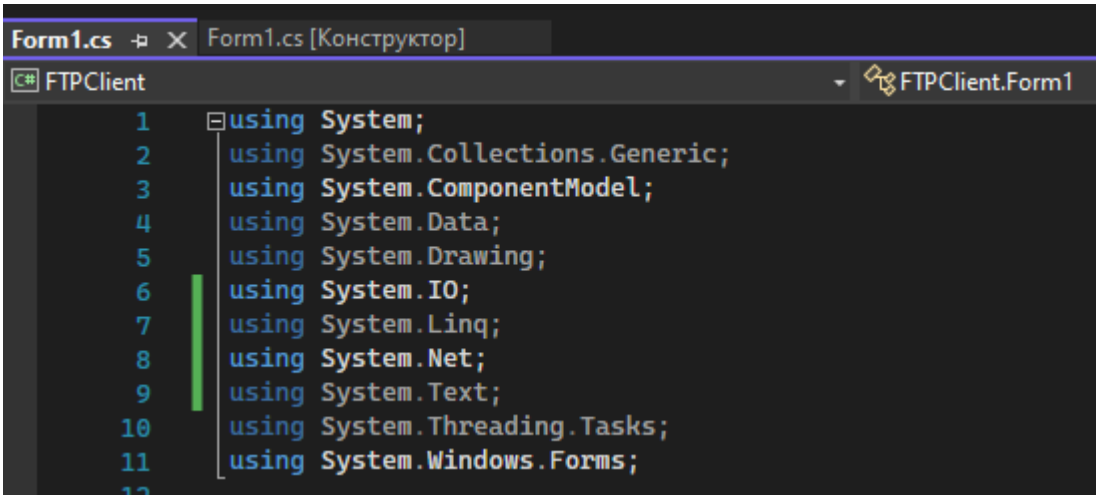

Рис. 11. Добавление библиотек

Далее напишем триггер нажатия на кнопку «Upload», для записи переменных с пользовательского текста в код см. рисунок 12.

```
Ссылок: 1
private void btnUpload_Click(object sender, EventArgs e)
{
    using(OpenFileDialog ofd = new OpenFileDialog() { Multiselect=false, ValidateNames=true, Filter= "All files|*.*" })
    {
        if(ofd.ShowDialog() == DialogResult.OK)
        {
            FileInfo fi=new FileInfo(ofd.FileName);
            _InputParameter.UserName = txtUser.Text;
            _InputParameter.Password = txtPass.Text;
            _InputParameter.Server = txtServe.Text;
            _InputParameter.FileName = fi.Name;
            _InputParameter.FullName = fi.FullName;
        }
    }
}
```

Рис. 12. Программирование кнопки «Upload»

Теперь напишем всплывающее сообщение при успешной загрузки файла на сервер см. рисунок 13.

```
Ссылок: 1
private void backgroundWorker_RunWorkerCompleted(object sender, RunWorkerCompletedEventArgs e)
{
    lblStatus.Text = "Upload complete";
}
```

Рис. 13. Программирование метки

Далее напишем метод обращение к FTP серверу см. рисунок 14-18.

```
Ссылок: 1
private void backgroundWorker_DoWork(object sender, DoWorkEventArgs e)
{
    string fileName = ((FtpSetting)e.Argument).FileName;
    string fullName = ((FtpSetting)e.Argument).FullName;
    string userName = ((FtpSetting)e.Argument).UserName;
    string password = ((FtpSetting)e.Argument).Password;
    string server = ((FtpSetting)e.Argument).Server;
    FtpWebRequest request = (FtpWebRequest)WebRequest.Create(new Uri(string.Format("{0}/{1}", server, fileName)));
    request.Method = WebRequestMethods.Ftp.UploadFile;
    request.Credentials = new NetworkCredential(userName, password);
    Stream ftpStream = request.GetRequestStream();
    FileStream fs = File.OpenRead(fullName);
    byte[] buffer = new byte[1024];
    double total = (double)fs.Length;
    double read = 0;
    int bytesRead = 0;
    do
    {
        if (!backgroundWorker.CancellationPending)
        {
            bytesRead = fs.Read(buffer, 0, 1024);
            ftpStream.Write(buffer, 0, bytesRead);
            read += (double)bytesRead;
            double percentage = read / total * 100;
            backgroundWorker.ReportProgress((int)percentage);
        }
    } while (bytesRead!=0);
    fs.Close();
    ftpStream.Close();
}
```

Рис. 14. Метод обращение к серверу

Struct поможет инкапсулировать данные и связанные функции см. рисунок 15.

```
Ссылок: 6
struct FtpSetting
{
    Ссылок: 2
    public string Server { get; set; }
    Ссылок: 2
    public string UserName { get; set; }
    Ссылок: 2
    public string Password { get; set; }
    Ссылок: 2
    public string FileName { get; set; }
    Ссылок: 2
    public string FullName { get; set; }
}
```

Рис. 15. Метод «Struct»

Теперь напишем метод всплывающее сообщение при успешной загрузки файла на сервер см. рисунок 16.

```
Ссылок: 1
private void backgroundWorker_ProgressChanged(object sender, ProgressChangedEventArgs e)
{
    lblStatus.Text = $"Uploaded{e.ProgressPercentage} %";
    progressBar.Value = e.ProgressPercentage;
    progressBar.Update();
}
```

Рис. 16. Метод всплывающего сообщения

Приступаем к тестированию программы см. рисунок 17-21.

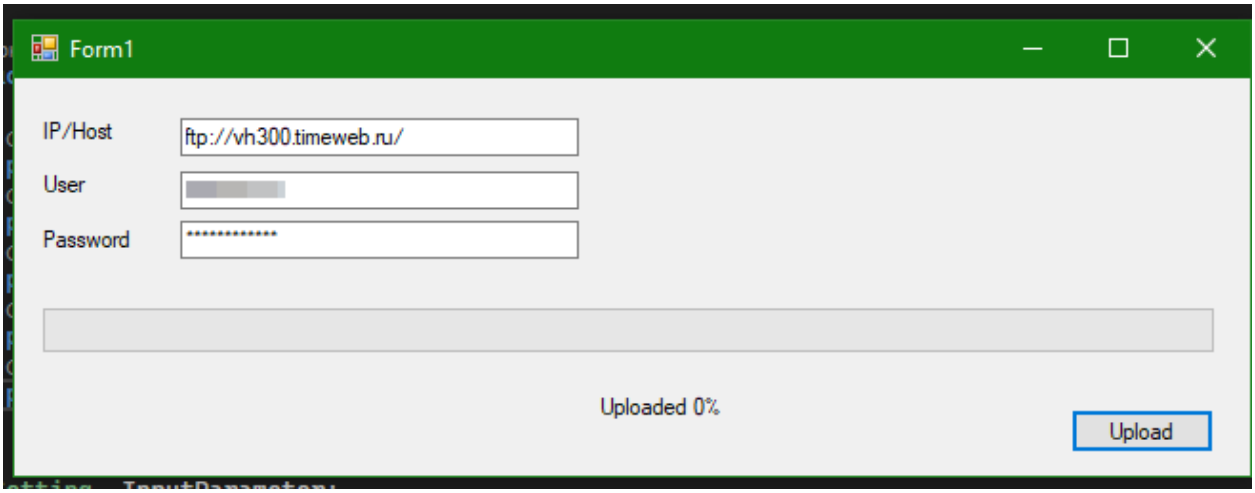

Рис. 17. Подключение к серверу

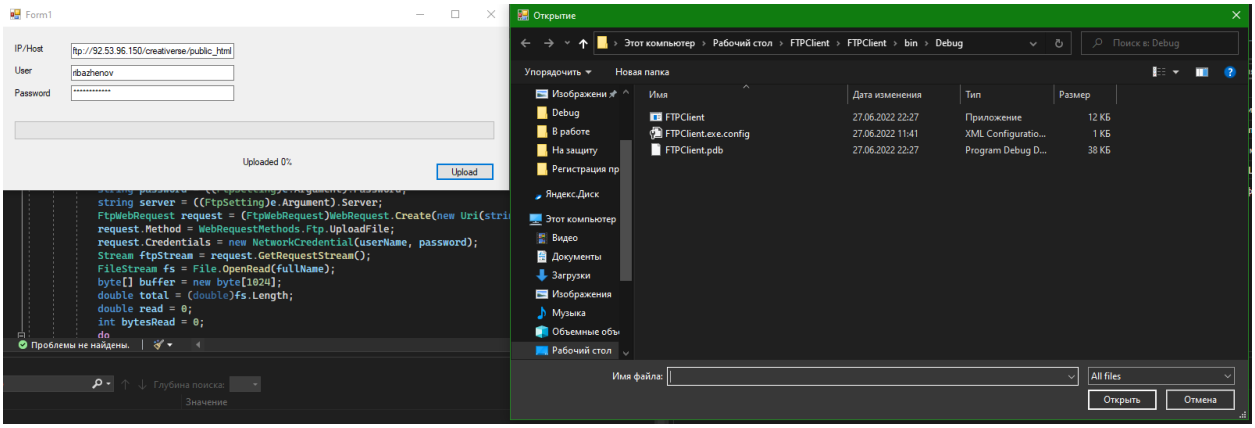

Рис. 18. Выбор файла для загрузки

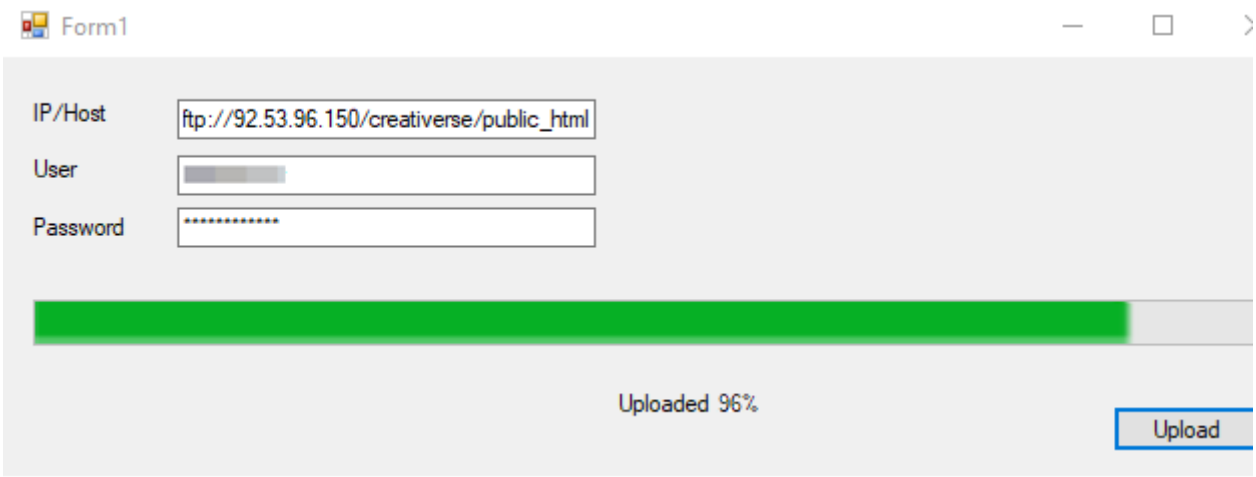

Рис. 19. Прогресс загрузки

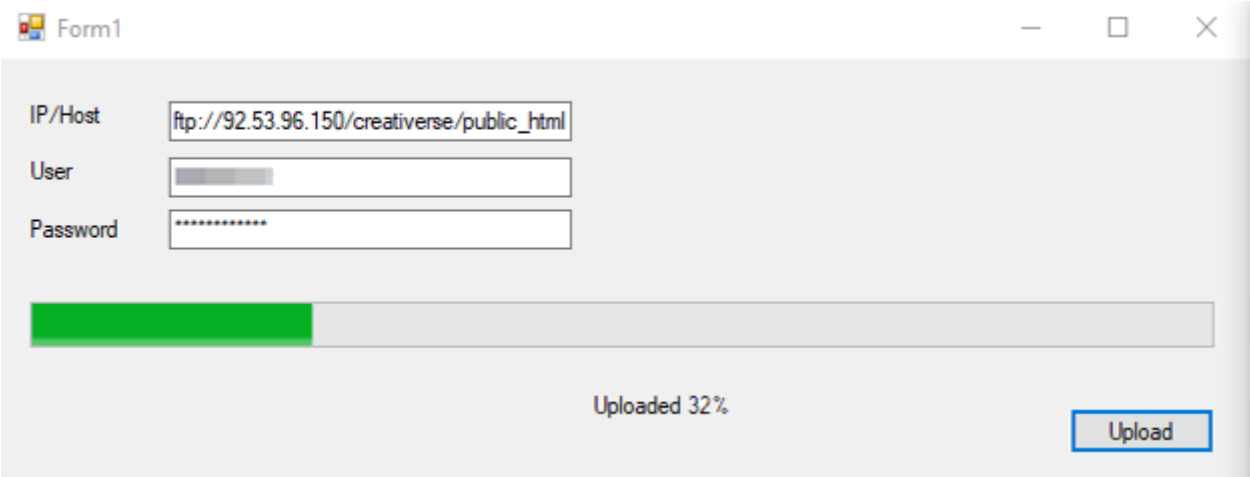

Рис. 20. Прогресс загрузки

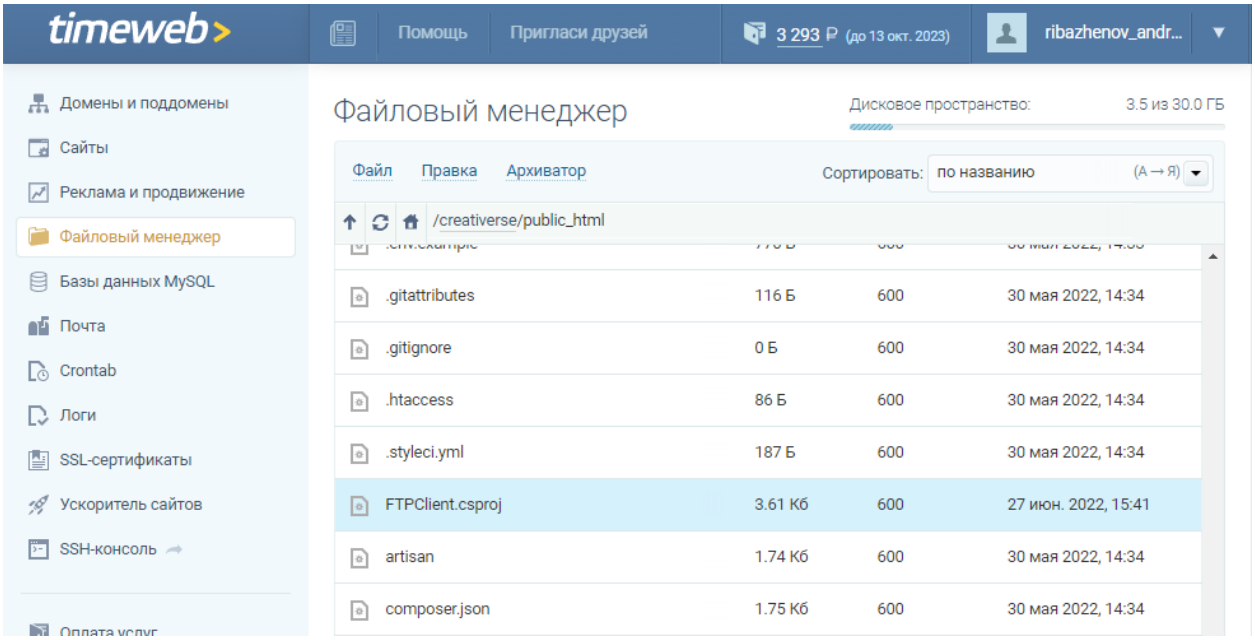

Рис. 21. Итог загрузки

Таким образом, был написан FTP-клиент в среде разработки «Visual Studio» на языке программирования C#.

Библиографический список

1. Додобоев Н. Н., Кукарцева О. И., Тынченко Я. А. Современные языки программирования // Современные технологии: актуальные вопросы, достижения и инновации. 2014. №5. С. 81-85.
2. Магомадова З. С. Языки программирования высокого уровня // Разработка и применение наукоёмких технологий в эпоху глобальных трансформаций. 2020. №8. С. 94-96.
3. Авдеев Н.Г., Ткачёв С.А., Борисов А.В. А. Разработка клиент-серверного приложения на основе протокола FTP // Молодежь и современные информационные технологии. 2016. С. 46-47.
4. Notess G.R. Learning to FTP //Online. 1994. №2. С. 79-82.