

Передача файлов в HTTP-запросах в контексте автоматизации обучения программированию

Болтовский Гавриил Александрович

Приамурский государственный университет им. Шолом-Алейхема

Студент

Аннотация

В данной статье рассматривается реализация алгоритма передачи архивированных файлов, закодированных в Base64, между клиентом и сервером через HTTP-запросы в JSON формате. Для реализации алгоритма был выбран язык программирования Python. Представленная реализация интегрирована в API-сервис для запуска программного кода на различных языках программирования ExecEngine, что позволило проверить работоспособность описанного алгоритма. Также была продемонстрирована работа обновленного сервиса на примере обучающей системы программированию.

Ключевые слова: Python, API, FastAPI, обучение программированию, система обучения программированию

File transfer in HTTP requests in the context of programming education automation

Boltovsky Gavriil Alexandrovich

Sholom-Aleichem Priamursky State University

Student

Abstract

This article discusses the implementation of an algorithm for transferring archived files encoded in Base64 between a client and a server via HTTP requests in JSON format. The Python programming language was chosen to implement the algorithm. The presented implementation was integrated into the ExecEngine API service for running program code in various programming languages, which made it possible to verify the functionality of the described algorithm. The updated service was also demonstrated using the example of a programming training system.

Keywords: Python, API, FastAPI, programming education, programming education system

1. Введение

1.1 Актуальность

В настоящее время системы для автоматизации обучения программированию занимают особое место в сфере преподавания. Такие системы сложны в реализации и включают в себя большое количество

функциональных возможностей. Сам процесс обучения программированию охватывает обширный спектр тем, связанных с различными функциями и возможностями языков программирования.

Одной из наиболее важных тем на начальных этапах изучения программирования является изучение работы с текстовыми файлами. Данная тема позволяет начинающим программистам понять принцип работы с файловой системой на языке программирования, узнать режимы работы с файлами и их отличия. Без корректного механизма обработки файлов на стороне сервера предоставить возможность реализации задач с использованием файловых потоков невозможно, поэтому тема передачи файлов в HTTP-запросах является актуальной, особенно в контексте автоматизации обучения программированию.

1.2 Обзор исследований

А.А. Ахтырский и Т.В. Волобуева в своей работе продемонстрировали принципы работы с JSON файлами на примере авторской программы, предназначенной для создания нового файла в формате JSON [1]. При создании файла имеется возможность указать наименования файла и директорию сохранения.

Ю.А. Зотова и И.Д. Котилевец в своей работе рассмотрели технологию создания информационной системы на основе REST API, а также обеспечивающую возможность изоляции базы данных CouchDB [2].

В.И. Гресь описал принципы работы шлюза API, его функциональные возможности и преимущества использования в микросервисной архитектуре [3]. Также изучены различные подходы к реализации шлюза API и примеры его использования в реальных проектах.

М.Э. Джалалов провёл исследование реализации шаблонов проектирования для управления API в микросервисах, включая их влияние на безопасность и производительность, а также проанализировал текущие тенденции и будущее развитие в данной области [4].

1.3 Цель исследования

Цель исследования – разработать модель передачи файлов между клиентом и сервером через HTTP-запросы в JSON формате.

2 Материалы и методы

Для решения поставленных задач выбран язык программирования Python. Для запуска программного кода в веб-ориентированных средах выбрана система ExecEngine пред релизной версии, а для тестирования результата предложена доработка сайта по обучению программированию.

3 Результаты и обсуждения

В настоящее время передача файлов через HTTP-запросы считается распространённой практикой, но она сопряжена с рядом потенциальных проблем и ограничений. При загрузке файла он обычно полностью читается в

память сервера (особенно при использовании нестриминговых подходов), что в свою очередь ведёт к высокой нагрузке на сервер и исчерпанию оперативной памяти.

Стоит отметить, что для передачи файлов через POST-запросы обычно используется multipart/form-data, что требует парсинга сложной структуры на сервере. Но такой подход может повлечь за собой ошибки при парсинге, несовместимость между клиентами и серверами или трудности при тестировании.

Способ решения возникающих проблем может разниться в зависимости от контекста ситуации. Можно отметить следующие способы решения проблемы передачи файлов:

1. Использование прямой загрузки в объектное хранилище.
2. Реализация чанковой загрузки (разделение файла на части).
3. Дополнительная настройка сервера: увеличение лимитов на передачу данных между клиентом и сервером или использование стриминга.
4. Отделение файлового хранилища от сервиса.

При этом можно выделить ещё один способ, который хоть и нельзя считать наилучшим в общем смысле, но способен решить поставленную задачу передачи файлов. Особенно описанный в данной статье способ может быть полезен для API-сервисов, передающих запросы между клиентом и сервером в JSON формате. Опишем алгоритм работы решения следующим образом (см. рис. 1):

1. Алгоритм получает данные от клиента в JSON формате, среди которых есть строка, хранящая в себе закодированный ZIP-архив с файлами, которые нужно обработать на стороне сервера.
2. Сервис обработки запроса декодирует строку и проводит распаковку архива в определённую для этой задачи директорию.
3. При необходимости сервер собирает информацию в архив, архивирует её и возвращает в закодированном формате.

Такой подход не предусматривает предварительную валидацию файлов, так как требуется распаковка архива. В таком случае стоит понимать, что в систему могут поступать вирусы и вредоносные скрипты, что может повлечь за собой отрицательные последствия. Поэтому использовать такой подход стоит либо в крайнем случае, когда multipart недоступен, либо существует способ изоляции процесса распаковки архива и чтения полученных со стороны клиента файлов.

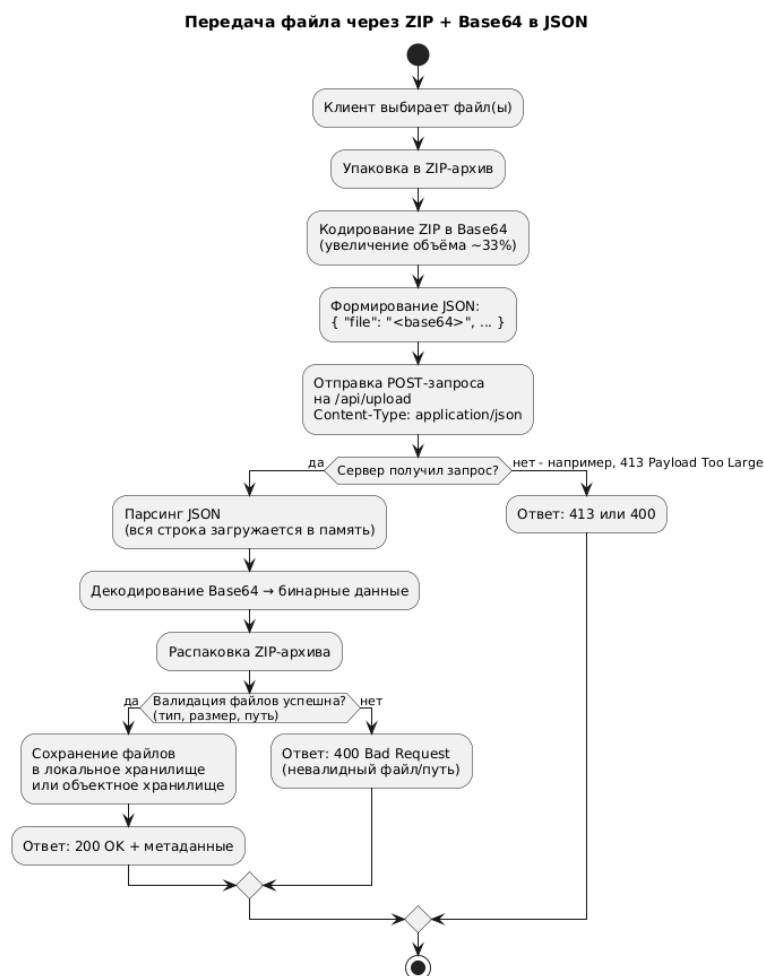


Рисунок 1. UML-диаграмма, описывающая алгоритм передачи файлов через ZIP

Сразу стоит отметить достоинства и недостатки. Плюсами такого подхода можно считать упрощённую структуру запроса, совместимость с классическим JSON форматом (особенно если API строго требует JSON и не поддерживает multipart, это может быть вынужденным решением) и возможность группировки нескольких файлов в ZIP-архив. Сам архив становится строкой в JSON, что может быть удобно для некоторых клиентов (например, мобильных приложений или систем без поддержки multipart/form-data).

Но есть и недостатки: очевидной проблемой становится увеличение объёма запроса, так как один ZIP-архив может представлять собой достаточно длинную строку, которая может не поместиться в предоставленные ограничения. Сам ZIP может сжать данные, но не всегда (например, медиафайлы формата JPEG, MP4 уже сжаты и почти не сожмутся дополнительно), а наличие его в запросе на сервер может в целом увеличить размер JSON данных, что также отрицательно скажется на эффективности работы сервера.

Но такой подход имеет место быть при передаче небольших наборов файлов, а также в микросервисах с контролируемой средой и малыми объёмами данных.

В 2023-м году на базе Приамурского государственного университета им. Шолом-Алейхема была запущена информационная система, основным назначением которой является предоставление инструментов для организации обучения студентов в веб-ориентированной среде (см. рис. 2). На 2026-й год в системе прошло обучение больше 100 студентов по различным дисциплинам, связанных с программированием («Основы алгоритмизации и программирования» на втором курсе по направлению 09.02.07 Информационные системы и программирование, «Технологии программирования» на первом курсе по направлению 09.02.03 Информационные системы и технологии, а также студенты других направлений), загружено более 6 тысяч решений к задачам практического характера.

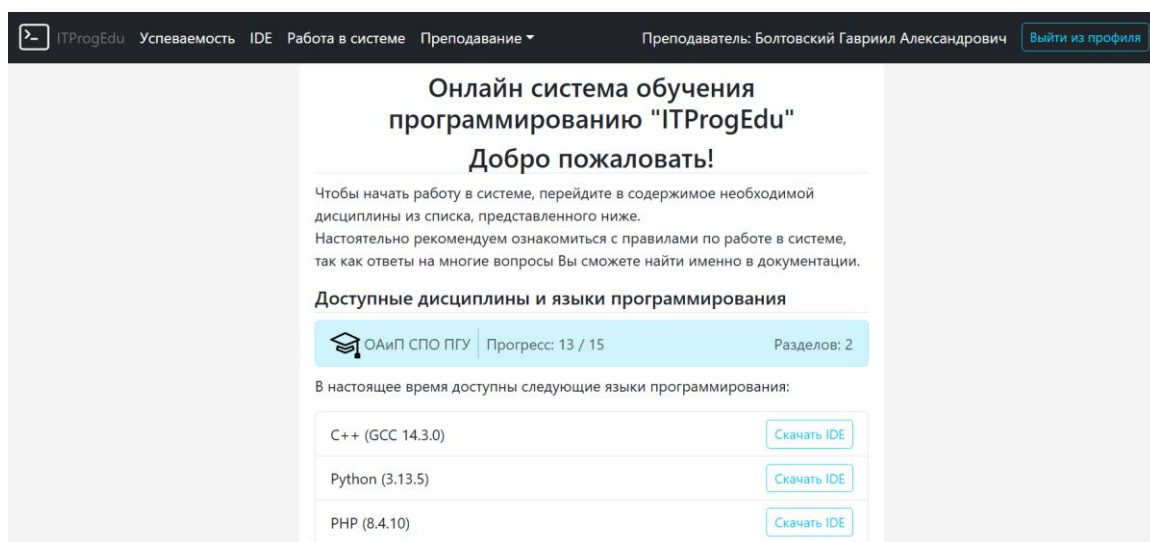


Рисунок 2. Интерфейс информационной системы автоматизации обучения программированию

В информационной системе студенты изучают лекционные материалы и решают задачи на предоставленных языках программирования. Результатом практической работы становится решение для загруженной преподавателем задачи (см. рис. 3). Реализована поддержка работы с 20-ю языками программирования.

Задачи включают в себя заранее заготовленные наборы тестовых данных, включающих входные и выходные данные. Реализована поддержка работы со стандартным и файловым потоками данных.

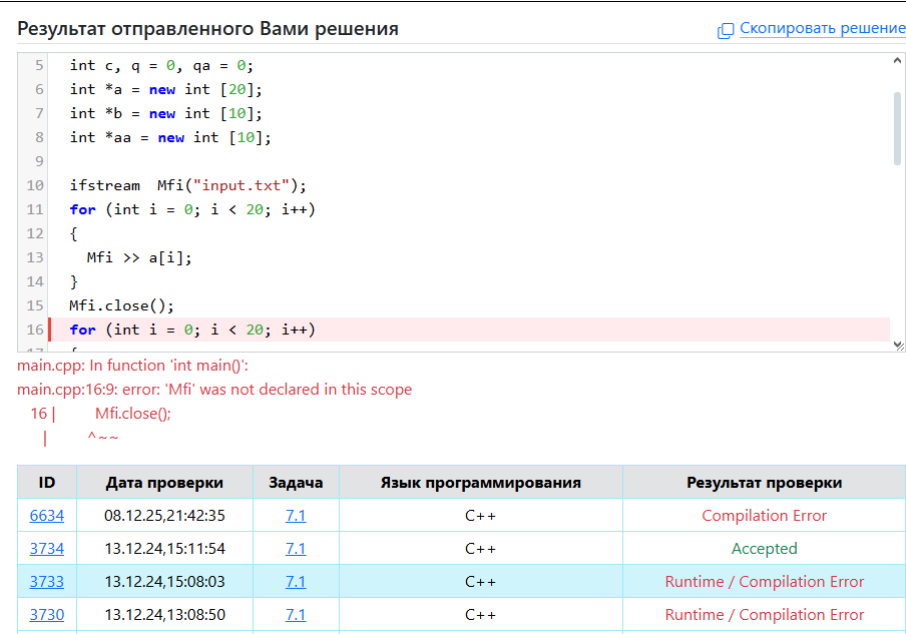


Рисунок 3. Интерфейс просмотра отправленного решения к задаче

Для автоматизации процесса проверки отправленных студентами решений был разработан отдельный API-сервис ExecEngine, предоставляющий необходимый инструментарий для загрузки программного кода для дальнейшего запуска и проверки его работоспособности по правилам АСМ. API-сервис обеспечивает бесперебойную работу информационной системы, а также обеспечивает безопасность работы основного сайта за счёт возможности изоляции процесса запуска программного кода. API-сервис был внедрён в конце 2024-го года и позволил повысить эффективность работы сайта (см. рис. 4), а также открыл больше возможностей для преподавателей в системе.

Лог проверки решения			
№ теста	Результат	Время	Память
1	Accepted	1.005 с.	52152 Кб.
2	Accepted	0.999 с.	52276 Кб.
3	Accepted	0.988 с.	52388 Кб.
4	Accepted	0.993 с.	52308 Кб.
5	Accepted	1.007 с.	52140 Кб.

Лог проверки решения			
№ теста	Результат	Время	Память
1	Accepted	0.032 с.	9836 Кб.
2	Accepted	0.031 с.	9756 Кб.
3	Accepted	0.032 с.	9832 Кб.
4	Accepted	0.035 с.	9888 Кб.
5	Accepted	0.026 с.	9944 Кб.

Рисунок 4. Результаты проверки решения студента до и после внедрения ExecEngine

Неотъемлемой частью обучения программированию является работа со сторонними файлами различных форматов: например, в основах алгоритмизации часто изучают работу с текстовыми файлами, а в анализе данных работу с табличными форматами. В таком случае важно предоставить преподавателям возможность загрузки файлов вместе с остальными данными о задачах (название, условие, тестовые данные и др.).

ExeсEngine написан на Python и разделён на несколько основных компонентов: модуль компиляторов, менеджер очереди, обработчики API-запросов, модуль работы с БД, сервис обработки пользовательского кода, конфигурационные файлы и др.

В пререлизной версии API-сервиса отсутствовала возможность отправки файлов вместе с программным кодом, а сам сервис передаёт данные только в JSON формате и поддерживается тип контента application/json (то есть multipart не предусмотрен). Для решения такой проблемы было предложено внести следующую доработку: добавить возможность передачи файлов в JSON формате через кодирование ZIP архива, что позволит обеспечить лёгкое взаимодействие с сервисом на различных языках программирования.

В ExeсEngine работает дополнительный модуль изоляции процессов isolate, который обеспечивает безопасный запуск программного кода в отдельных песочницах, которые после завершения работы программы удаляются сервисом. Алгоритм работы с isolate сводится к следующим шагам:

1. Инициализировать песочницу путём создания отдельной директории с последующей загрузкой программного кода и входных данных.
2. Запустить программный код (а также предварительно его скомпилировать) и передать в него входные данные.
3. Зафиксировать результат работы программы путём чтения специального файла meta.txt.
4. Очистить и удалить директорию после завершения работы.

В первый пункт был добавлен алгоритм декодирования Base64 строки из атрибута additional_files, в котором пользователь может передать ZIP архив с файлами. После успешного декодирования isolate распаковывает архив и сохраняет файлы в песочнице:

```
if submission.additional_files:
    archive_path = f"{work_dir}/box/archive.zip"
    with open(archive_path, 'wb') as f_unzip:
        f_unzip.write(base64.b64decode(submission.additional_files))

    unzip_cmd = (
        f"isolate --cg --box-id={box_id} "
        f"--time=5 "
        f"--fsize={settings.MAX_FILE_SIZE} "
        f"--cg-mem={settings.MAX_MEMORY_LIMIT} "
        f"--stderr-to-stdout "
        f"--run -- /usr/bin/unzip -n -qq archive.zip"
    )
```

```
proc = await asyncio.create_subprocess_shell(
    unzip_cmd,
    stdout = asyncio.subprocess.PIPE,
    stderr = asyncio.subprocess.PIPE
)
stdout_unzip, stderr_unzip = await proc.communicate()
print(stdout_unzip, stderr_unzip)
```

Таким образом пользователи получили возможность отправлять файлы вместе с программным кодом, а разработчикам, использующим API в реализации своих информационных систем, нет необходимости вносить изменения в работу сервиса самостоятельно.

Для проверки было реализовано обновление сайта обучения программирования на базе Приамурского государственного университета им. Шолом-Алейхема. Сайт написан на PHP, поэтому синтаксис работы с ZIP архивами отличается. Основной задачей на стороне ИС является архивирование файлов с последующим кодированием в Base64. Для этого был написан соответствующий метод (пути к файлам скрыты):

```
function encode_zip($taskId) {
    $additional_files_path = "/additional_files/";

    if (!is_dir($additional_files_path)) {
        return null;
    }

    $zip_filename = "archive.zip";
    $zip = new ZipArchive();
    if ($zip->open($zip_filename, ZipArchive::CREATE) !==
TRUE) {
        return null;
    }
    $files = new RecursiveIteratorIterator(
        new
RecursiveDirectoryIterator($additional_files_path),
        RecursiveIteratorIterator::LEAVES_ONLY
    );

    $file_count = 0;
    foreach ($files as $file) {
        if ($file->isFile()) {
            $file_count++;
        }
    }

    if ($file_count === 0) {
        return null;
    }

    foreach ($files as $name => $file) {
```

```

        $relative_path = substr($name,
        strlen($additional_files_path));

        if ($file->isDir()) {
            $zip->addEmptyDir($relative_path);
        } else {
            $zip->addFile($name, $relative_path);
        }
    }
    $zip->close();

    $zip_data = file_get_contents($zip_filename);
    unlink($zip_filename);
    return base64_encode($zip_data);
}

```

Также на сайт была добавлена возможность загрузки файлов к задачам. Для их отображения был задействован дополнительный модуль JSTree, позволяющий показать файловую структуру в древовидном формате (см. рис. 5).

Ограничение по времени: 5

Ограничение по памяти: 64

Сохранить

Дополнительные файлы

- file5.txt
- file2.txt
- file3.txt
- file1.txt
- file4.txt

Тесты к задаче

Рисунок 5. Отображение дополнительных файлов, загруженных к задаче

Студентам системы был предложен ряд задач на работу с файлами с различными условиями: в некоторых требовалось прочитать файл по названию, а в других был закрыт доступ к стандартному потоку данных, чтобы обучающиеся работали только с файловыми потоками.

4 Выводы

В результате проверки работоспособности системы был выявлен ряд недочётов, что позволило исправить ошибки до выпуска первой релизной версии API-сервиса. Например, одной из ошибок стала невозможность распаковки файлов в директорию, так как в функцию сервис передавал стандартное значение идентификатора песочницы (равняется -1) вместо корректного.

Результат проверки ([в отдельном окне](#))

```
1 f = open("prog.in", 'r')
2 n = int(f.read())
3
4 g = open("prog.out", 'w')
5 with open("data.txt", 'r') as ff:
6     if n <= 0:
7         pass
8     else:
9         count = 1
10        for line in ff:
11            g.write(line)
12            count +=1
13            if count>=n+1:
14                break
15 g.close()
```

ID отправки: 6630

Уникальный идентификатор решения (UUID): 457c8c8c-7e01-4e56-97f9-fd231cf6de94

Результат проверки: Принято

Accepted

Изменить вердикт

Повторная проверка

Удалить решение

Рисунок 6. Пример решения задачи студентом на закрепление знаний работы с файлами на Python

Библиографический список

1. Ахтырский А.А., Волобуева Т.В. Создание и обработка файлов в формате JSON // Информационные технологии в строительных, социальных и экономических системах. 2024. № 1-2(31-32). С. 128-133.
2. Зотова Ю.А., Котилевец И.Д. Разработка архитектуры rest api для взаимодействия с сервисами приложения // Информационные технологии и математическое моделирование систем 2018: труды международной научно-технической конференции, Одинцово, 19–21 ноября 2018 года. Одинцово: Федеральное государственное бюджетное учреждение науки Центр информационных технологий в проектировании Российской академии наук. 2018. С. 61-65.
3. Гресь В.И. Создание шлюза API: маршрутизации и фильтрации в контексте микросервисной архитектуры // Молодежь третьего тысячелетия : Сборник научных статей XLVIII региональной студенческой научно-практической конференции. В 2-х частях, Омск, 01 апреля – 15 2024 года. Омск: Омский государственный университет им. Ф.М. Достоевского. 2024. С. 256-258.
4. Джалалов М.Э. Применение шаблонов проектирования для управления API в микросервисной архитектуре // Экономика и качество систем связи. 2024. № 1(31). С. 128-136.