Учреждение образования

«Белорусский государственный технологический университет»

**Лабораторная работа №6**

Выполнил:

Студент 3 курса 4 группы ФИТ

Сосновец Мария Игоревна

2024 г.

**Лабораторная 6. Протокол WebDAV**

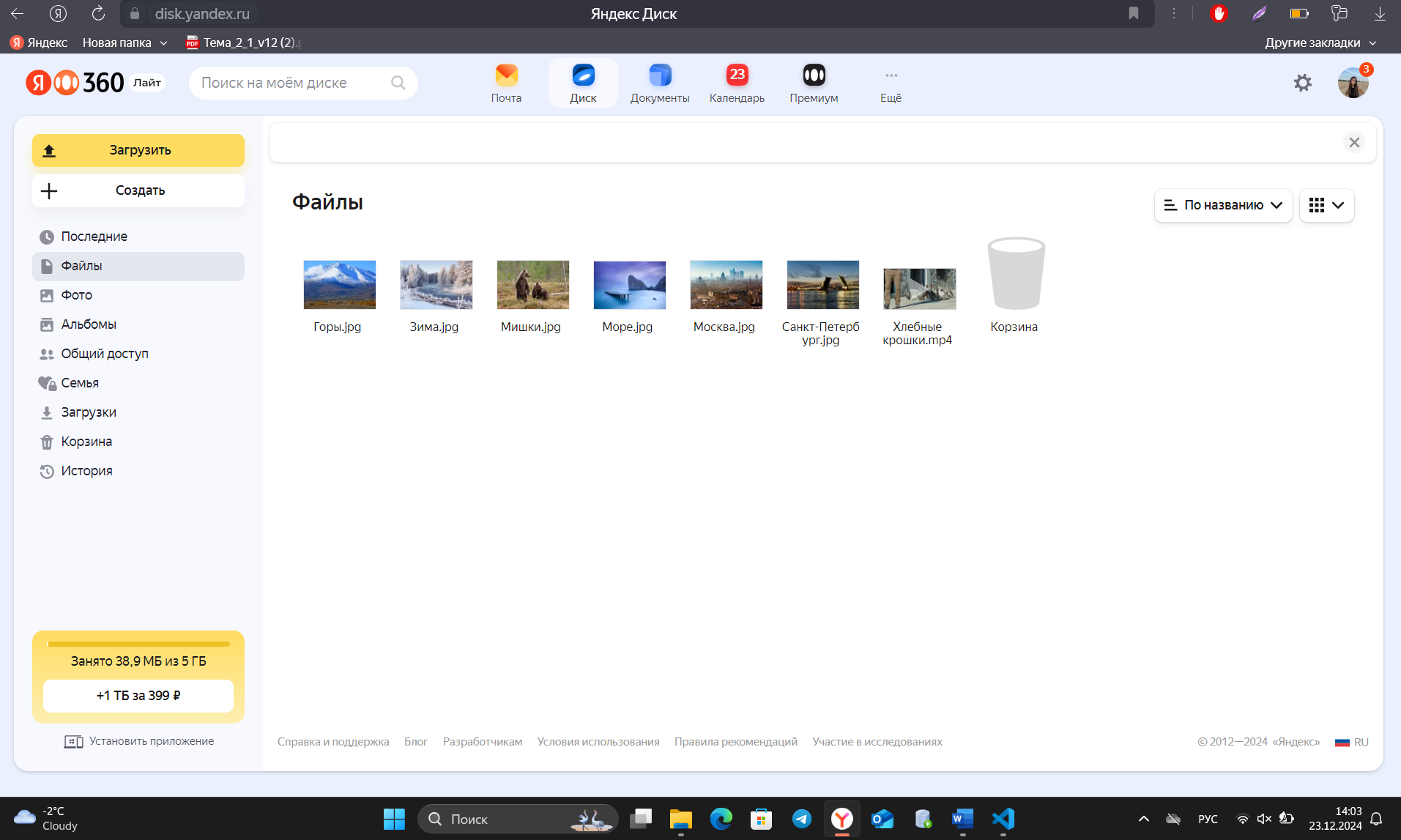
1. **Выберите любое облачное файловое хранилище, поддерживающее WebDAV-протокол.**

Было выбрано Яндекс.Диск в качестве облачного файлового хранилища, поддерживающего WebDAV-протокол, по нескольким причинам. Во-первых, Яндекс.Диск предлагает удобный интерфейс и простоту в использовании, что делает его доступным для пользователей любого уровня. Во-вторых, он обеспечивает надежную безопасность данных, включая шифрование и возможность двухфакторной аутентификации. В-третьих, интеграция с другими сервисами Яндекса позволяет легко обмениваться файлами и работать с ними в команде. Кроме того, Яндекс.Диск предоставляет достаточно бесплатного места для хранения, что является дополнительным преимуществом.

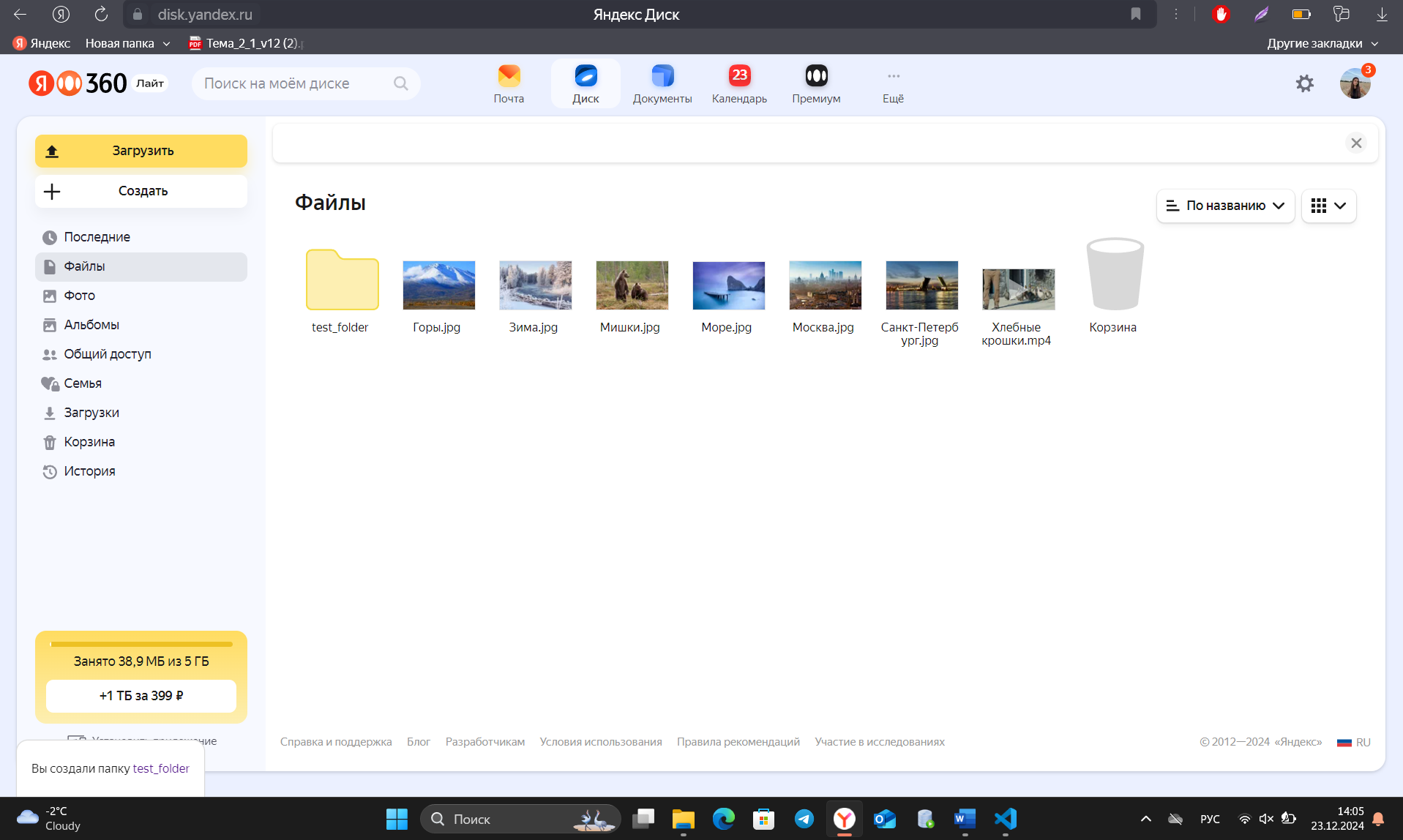
1. **Разработайте проиложение (произвольная реализация), выполняющее:**

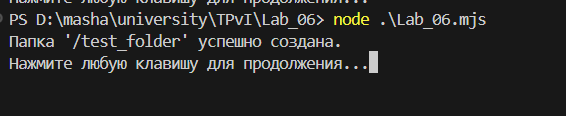
* **создание папки в файловом хранилище;**

|  |
| --- |
| //создание папки  async function createFolder(folderPath) {    try {      await client.createDirectory(folderPath);      console.log(`Папка '${folderPath}' успешно создана.`);    } catch (error) {      console.error(`Ошибка при создании папки: ${error}`);    }  } |

Яндекс.Диск до создания папки 

Яндекс.Диск после создания папки

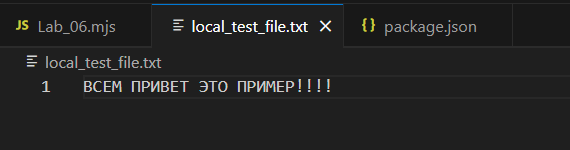
****

****

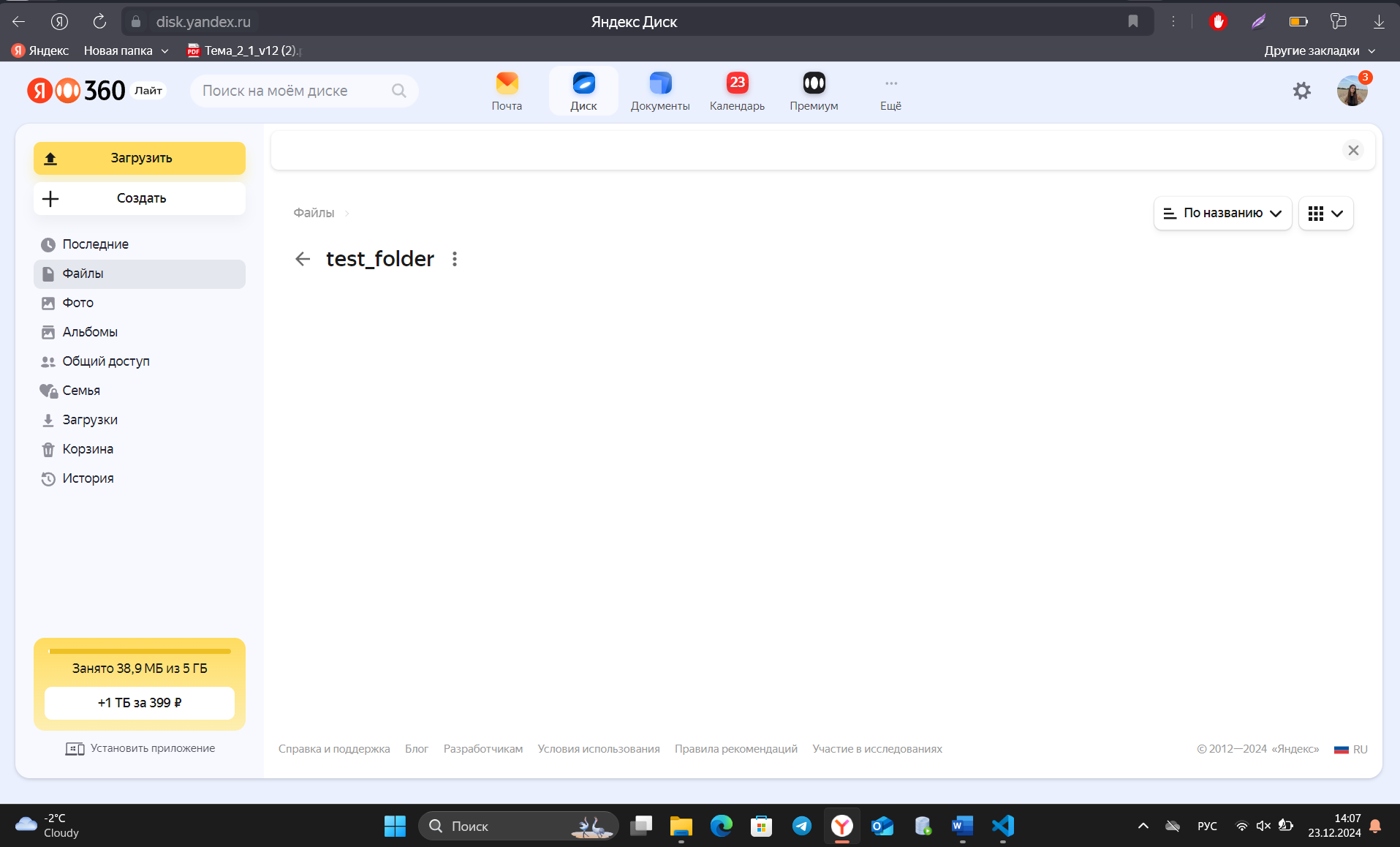
* **выгрузку (upload) файла в файловом хранилище;**

|  |
| --- |
| // Загрузка файла  async function uploadFile(localFilePath, remoteFilePath) {    try {      await client.putFileContents(remoteFilePath, fs.createReadStream(localFilePath));      console.log(`Файл '${localFilePath}' загружен на сервер как '${remoteFilePath}'.`);    } catch (error) {      console.error(`Ошибка при загрузке файла: ${error}`);    }  } |

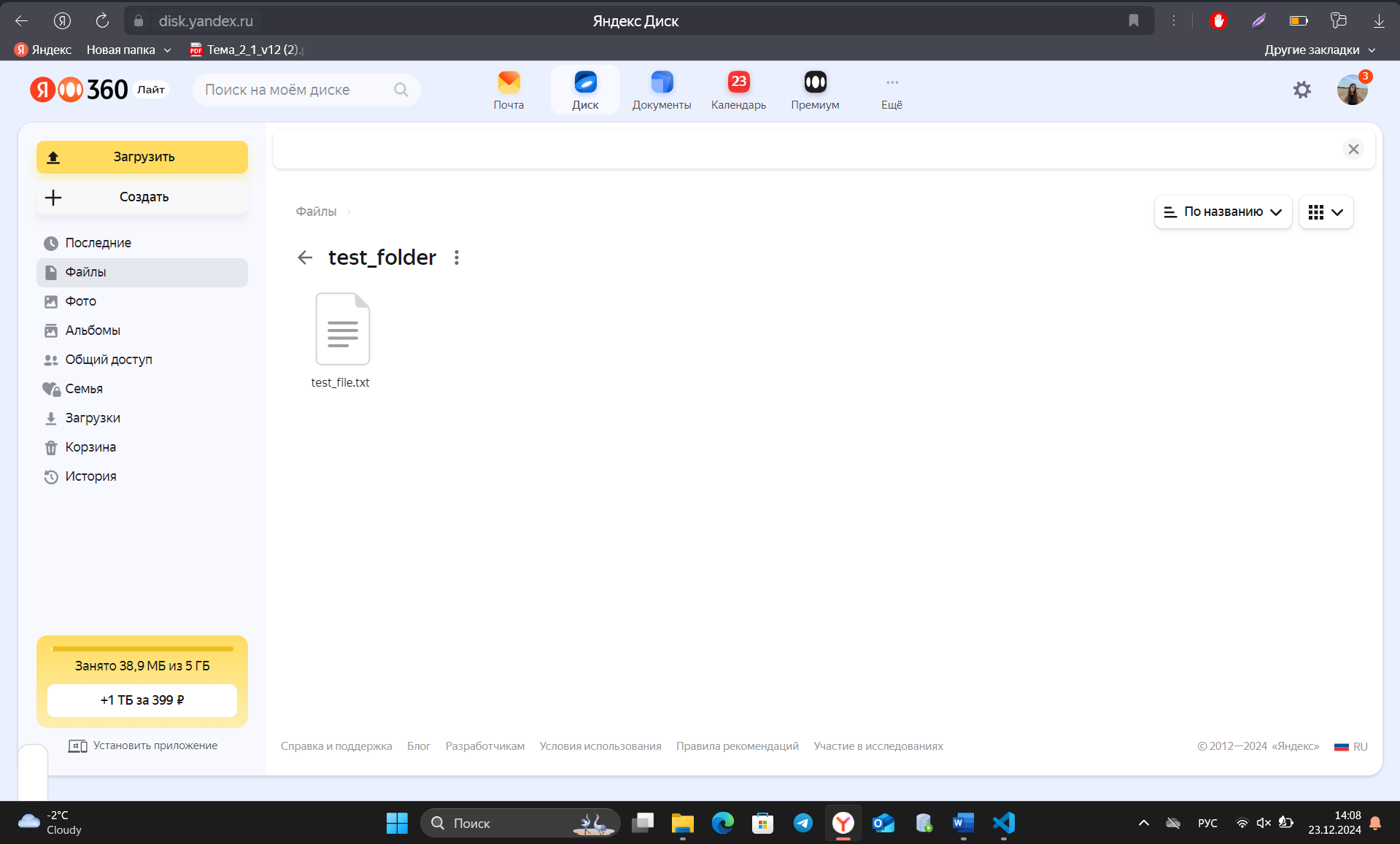
Файл, который будет загружен на сервер как '/test\_folder/test\_file.txt'.

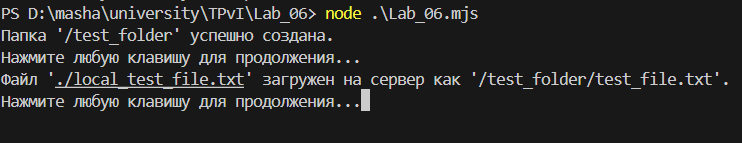
****

Папка до загрузки



Папка после загрузки файла

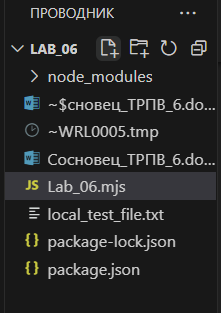




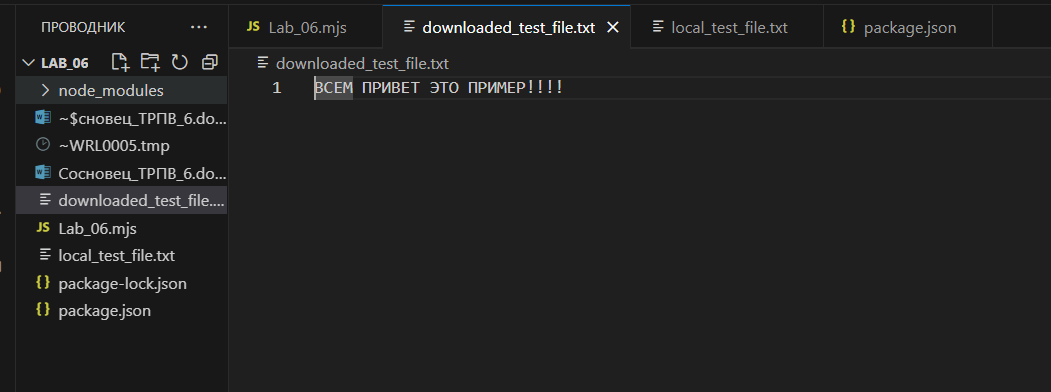
* **загрузку (download) файла из файлового хранилища;**

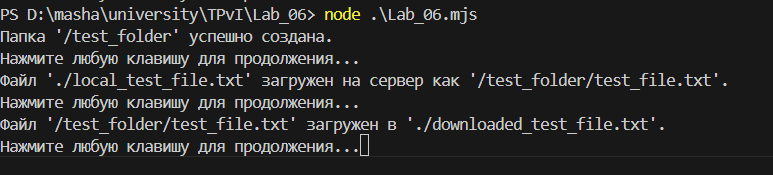
|  |
| --- |
| // Скачивание файла  async function downloadFile(remoteFilePath, localFilePath) {    try {      const fileContent = await client.getFileContents(remoteFilePath, { format: 'binary' });      fs.writeFileSync(localFilePath, fileContent);      console.log(`Файл '${remoteFilePath}' загружен в '${localFilePath}'.`);    } catch (error) {      console.error(`Ошибка при скачивании файла: ${error}`);    }  } |

До загрузки файла



После загрузки файла

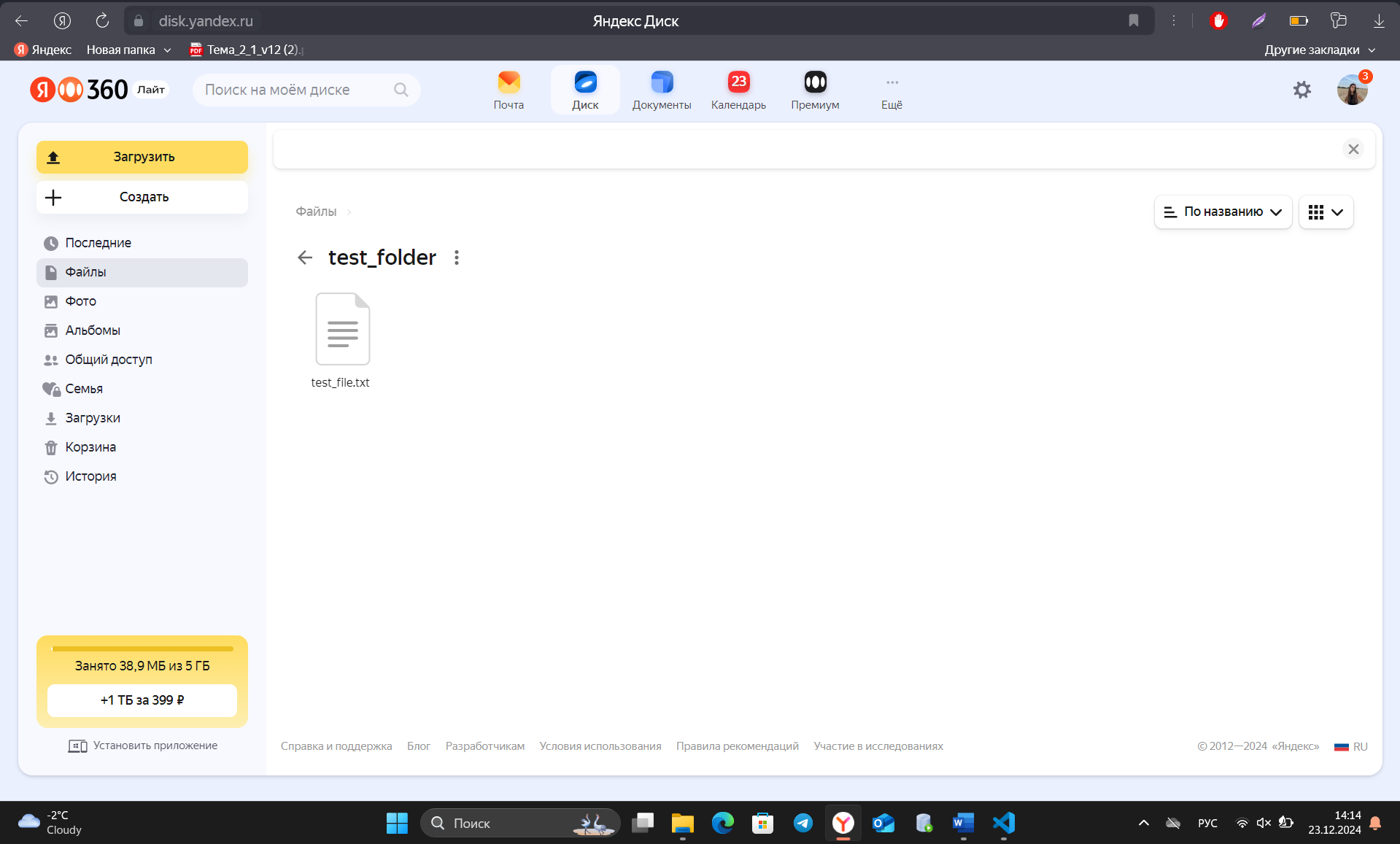




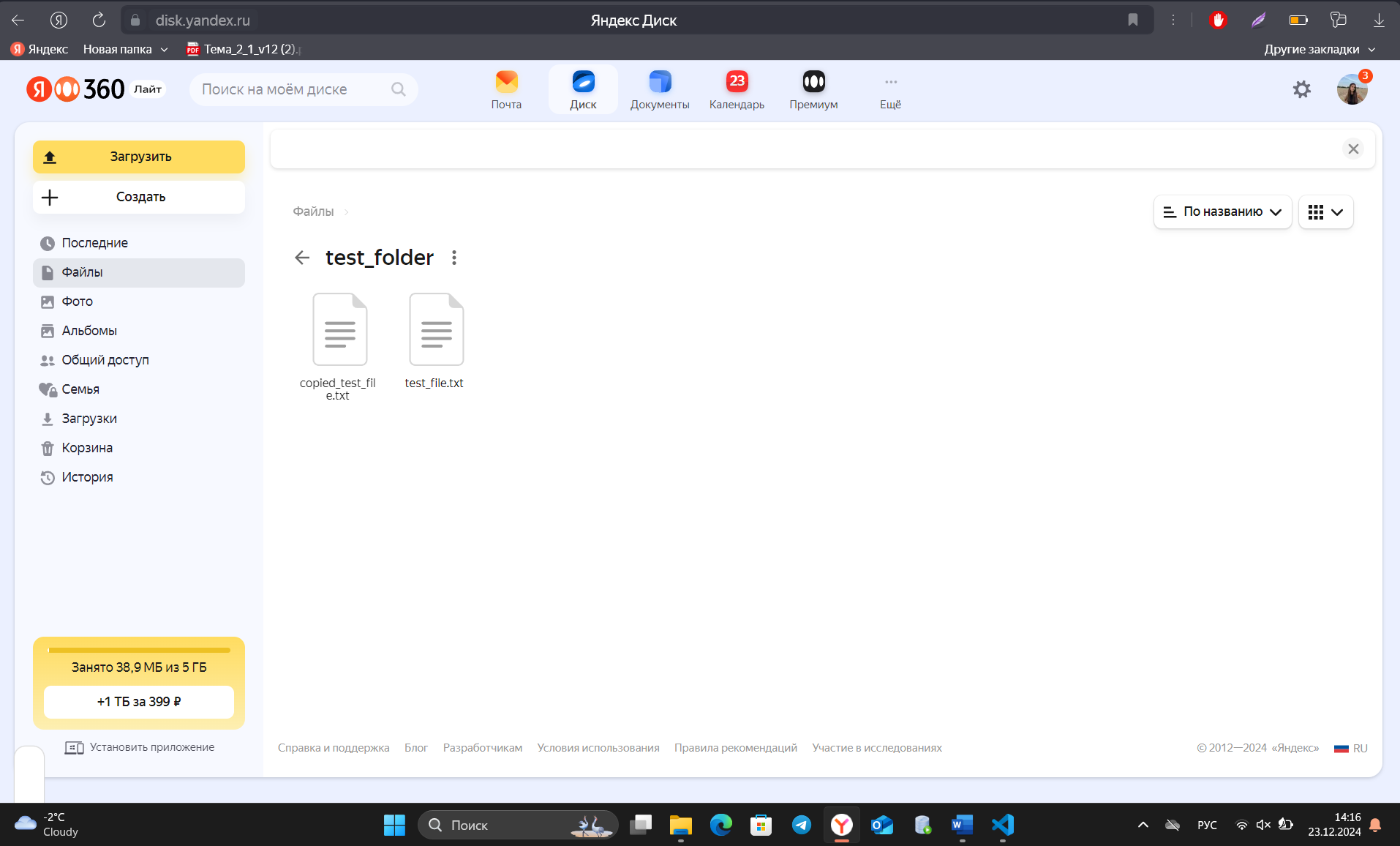
* **копирование файла из папки в папку файлового хранилища;**

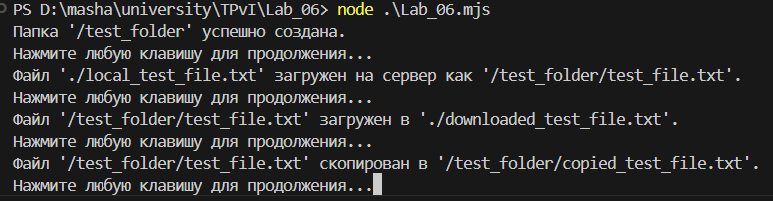
|  |
| --- |
| // Копирование файла  async function copyFile(sourceFilePath, destinationFilePath) {    try {      const fileContent = await client.getFileContents(sourceFilePath, { format: 'binary' });      await client.putFileContents(destinationFilePath, fileContent);      console.log(`Файл '${sourceFilePath}' скопирован в '${destinationFilePath}'.`);    } catch (error) {      console.error(`Ошибка при копировании файла: ${error}`);    }  } |

До копирования файла из папки в папку файлового хранилища



После копирования из папки в папку файлового хранилища

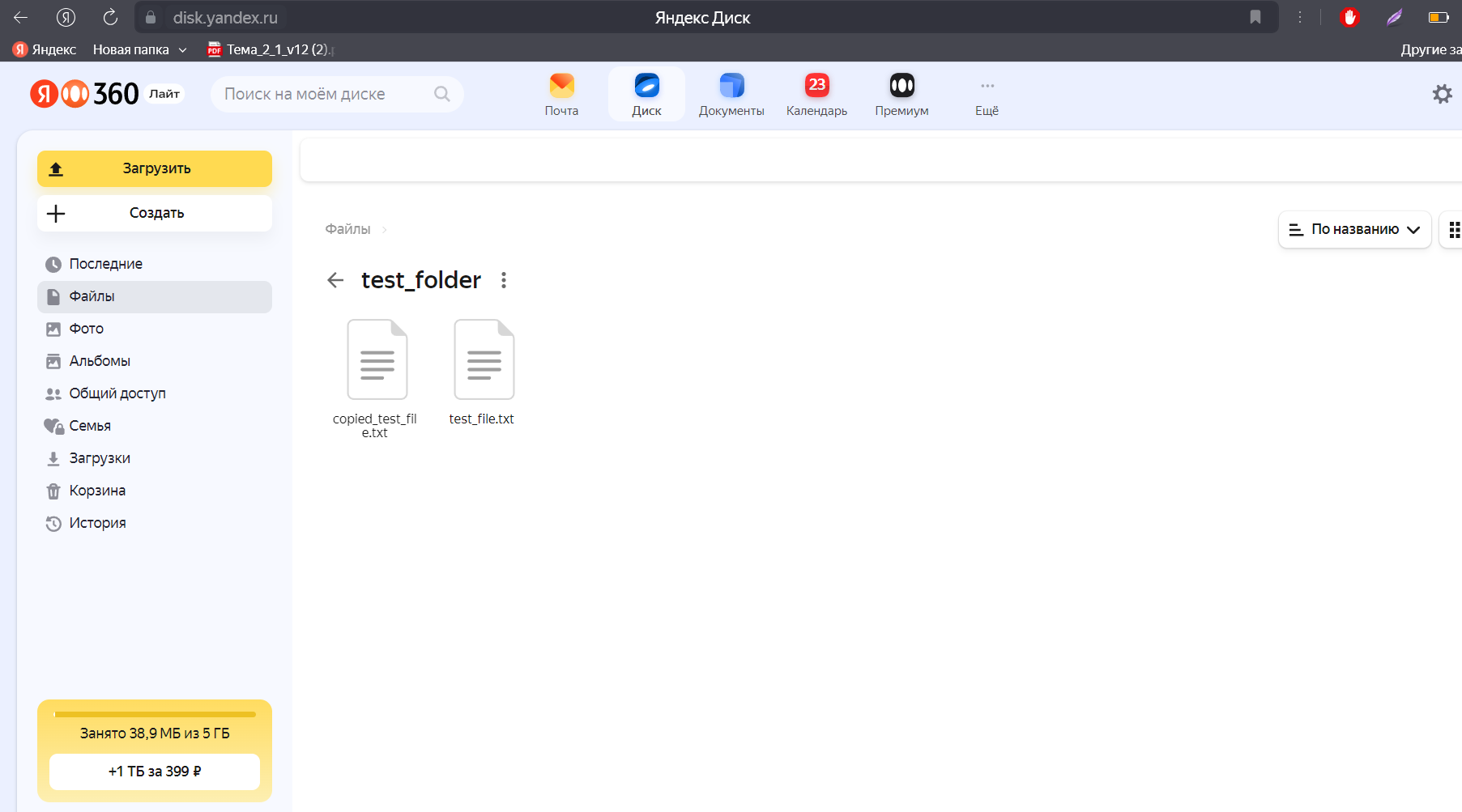




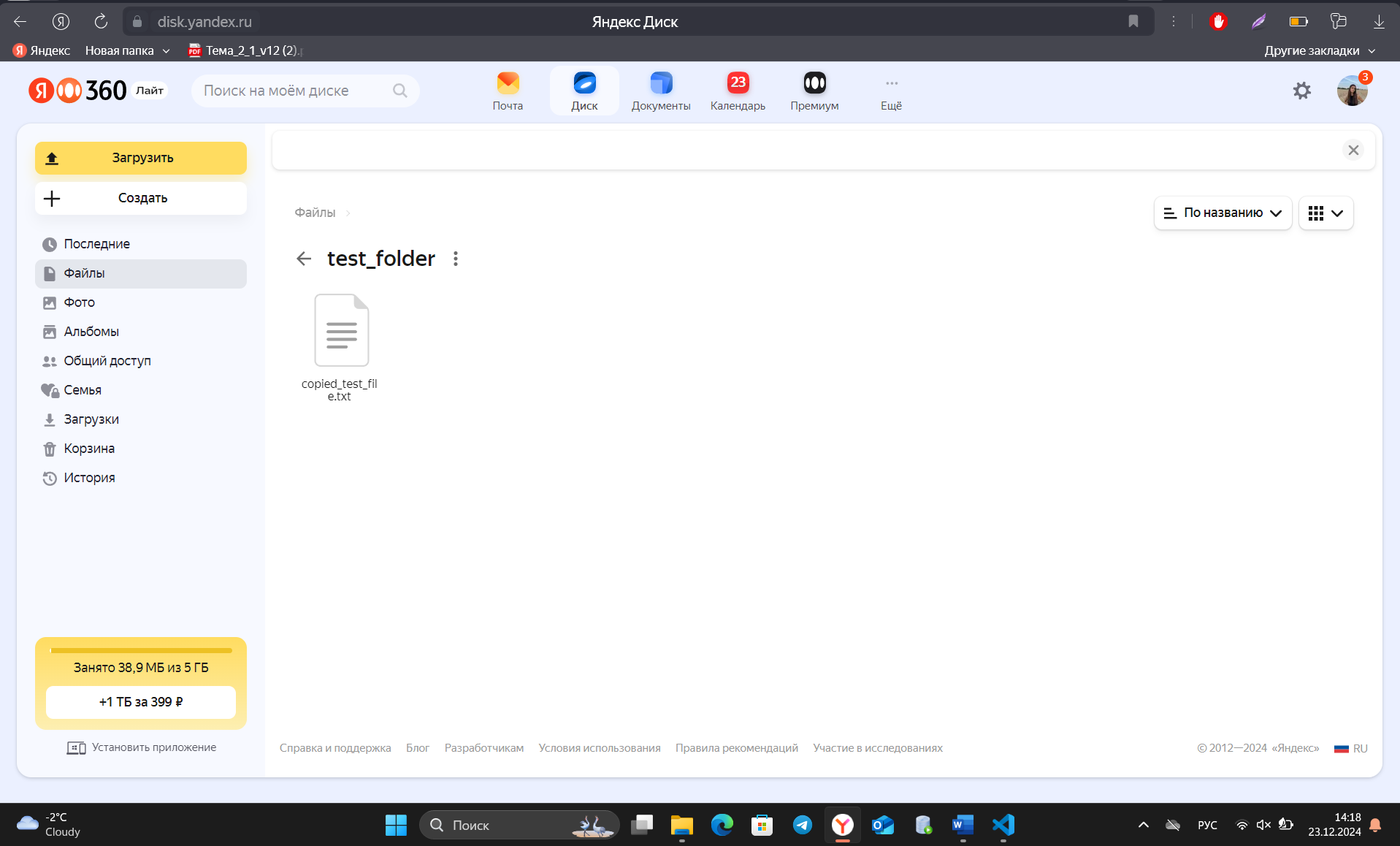
* **удаление файла в файловом хранилище;**

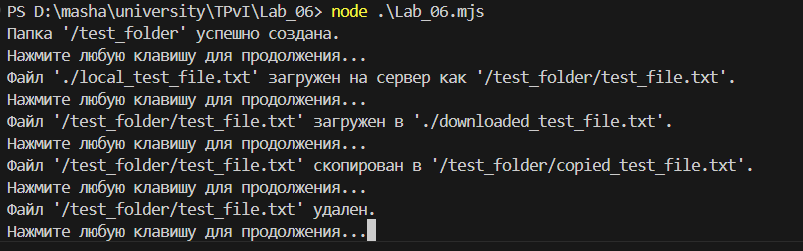
|  |
| --- |
| // Удаление файла  async function deleteFile(filePath) {    try {      await client.deleteFile(filePath);      console.log(`Файл '${filePath}' удален.`);    } catch (error) {      console.error(`Ошибка при удалении файла: ${error}`);    }  } |

До удаления файла в файловом хранилище



После удаления файла в файловом хранилище

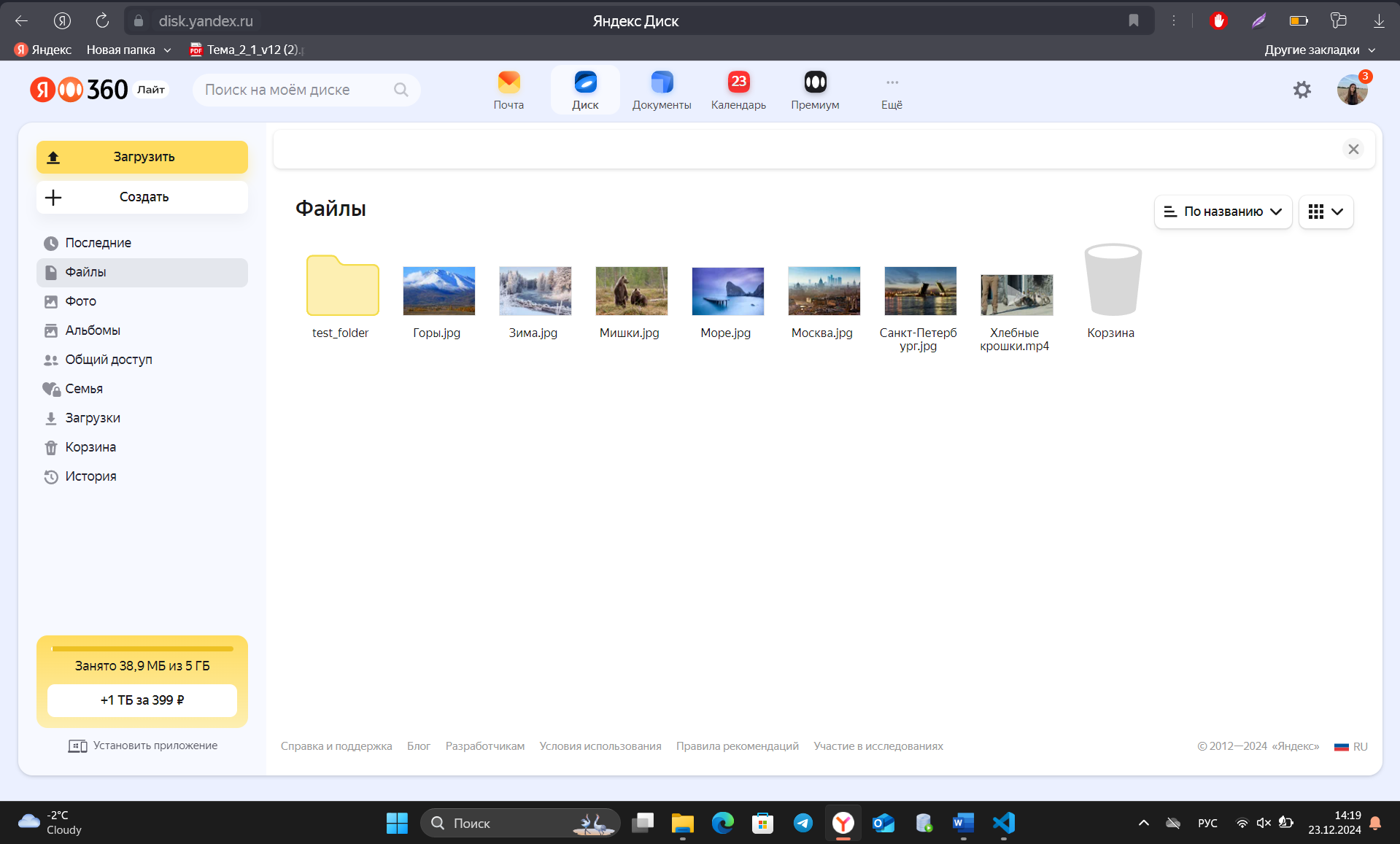




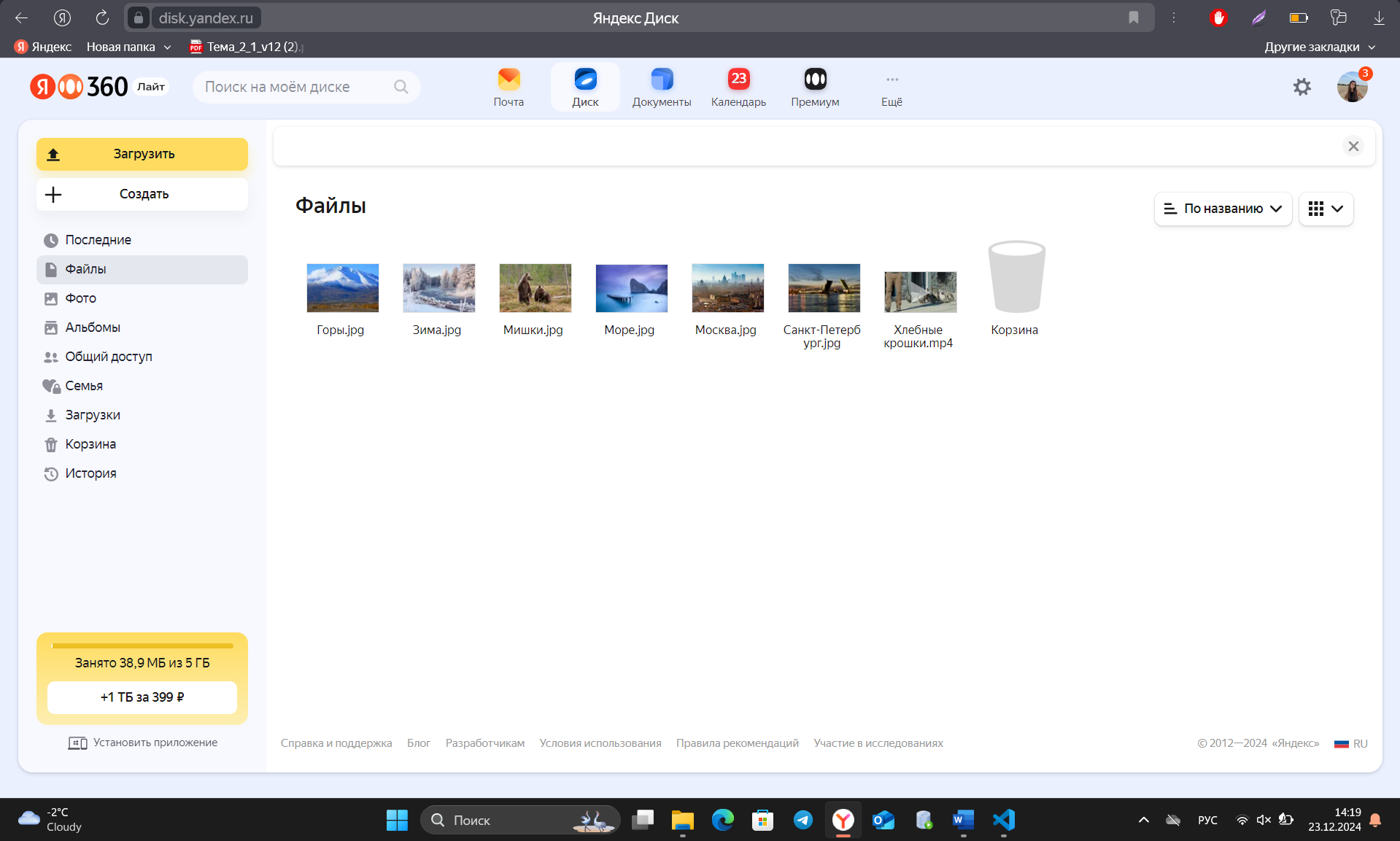
* **удаление папки в файловом хранилище.**

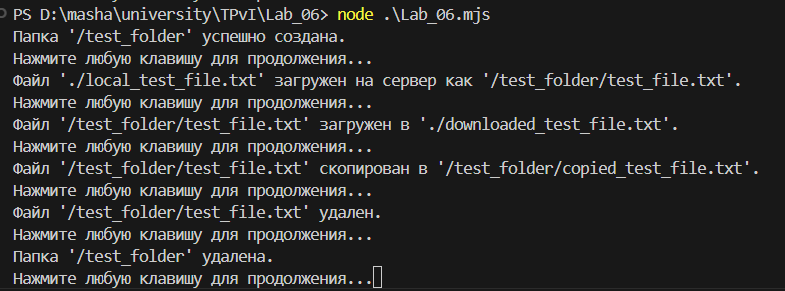
|  |
| --- |
| // Удаление папки  async function deleteFolder(folderPath) {    try {      const files = await client.getDirectoryContents(folderPath);      for (const file of files) {        await client.deleteFile(file.filename);      }      await client.deleteFile(folderPath);  // Удаляем саму папку      console.log(`Папка '${folderPath}' удалена.`);    } catch (error) {      console.error(`Ошибка при удалении папки: ${error}`);    }  } |

До удаления папки в файлом хранилище



После удаления папки в файлом хранилище





**Вывод:**

Основные преимущества WebDAV:

1. Удобство использования: WebDAV позволяет выполнять операции с файлами и папками, такие как создание, удаление и редактирование, используя стандартные HTTP-запросы, что делает его доступным для различных приложений.
2. Интеграция с облачными сервисами: Яндекс.Диск, поддерживающий WebDAV, предоставляет возможность управлять файлами через привычные инструменты, такие как файловые менеджеры и специализированные приложения.
3. Кроссплатформенность: WebDAV работает на различных операционных системах, что позволяет пользователям получать доступ к своим данным с разных устройств без необходимости установки дополнительных программ.
4. Безопасность: Яндекс.Диск обеспечивает надежную защиту данных, используя шифрование и другие методы безопасности, что делает работу с файлами безопасной.

**Приложение**

|  |
| --- |
| import { createClient } from 'webdav';  import fs from 'fs';  import readline from 'readline';  // Создаем интерфейс для чтения ввода с клавиатуры  const rl = readline.createInterface({    input: process.stdin,    output: process.stdout  });  // Настройки для WebDAV на Яндекс.Диске  const client = createClient(    'https://webdav.yandex.ru',  // URL WebDAV сервера    {      username: 'sosnovetsm@yandex.com',  // Логин Яндекс      password: 'vchuecoqohdsttmm',  // Пароль от Яндекс    }  );  // Функция для ожидания нажатия клавиши  function waitForKeyPress() {    return new Promise((resolve) => {      rl.question('Нажмите любую клавишу для продолжения...', () => {        resolve();      });    });  }  // Создание папки  async function createFolder(folderPath) {    try {      await client.createDirectory(folderPath);      console.log(`Папка '${folderPath}' успешно создана.`);    } catch (error) {      console.error(`Ошибка при создании папки: ${error}`);    }  }  // Загрузка файла  async function uploadFile(localFilePath, remoteFilePath) {    try {      await client.putFileContents(remoteFilePath, fs.createReadStream(localFilePath));      console.log(`Файл '${localFilePath}' загружен на сервер как '${remoteFilePath}'.`);    } catch (error) {      console.error(`Ошибка при загрузке файла: ${error}`);    }  }  // Скачивание файла  async function downloadFile(remoteFilePath, localFilePath) {    try {      const fileContent = await client.getFileContents(remoteFilePath, { format: 'binary' });      fs.writeFileSync(localFilePath, fileContent);      console.log(`Файл '${remoteFilePath}' загружен в '${localFilePath}'.`);    } catch (error) {      console.error(`Ошибка при скачивании файла: ${error}`);    }  }  // Копирование файла  async function copyFile(sourceFilePath, destinationFilePath) {    try {      const fileContent = await client.getFileContents(sourceFilePath, { format: 'binary' });      await client.putFileContents(destinationFilePath, fileContent);      console.log(`Файл '${sourceFilePath}' скопирован в '${destinationFilePath}'.`);    } catch (error) {      console.error(`Ошибка при копировании файла: ${error}`);    }  }  // Удаление папки  async function deleteFolder(folderPath) {    try {      const files = await client.getDirectoryContents(folderPath);      for (const file of files) {        await client.deleteFile(file.filename);      }      await client.deleteFile(folderPath);  // Удаляем саму папку      console.log(`Папка '${folderPath}' удалена.`);    } catch (error) {      console.error(`Ошибка при удалении папки: ${error}`);    }  }  // Основная функция  (async () => {    const remoteFolder = '/test\_folder';    const remoteFile = '/test\_folder/test\_file.txt';    const copiedFilePath = '/test\_folder/copied\_test\_file.txt';    const localFilePath = './local\_test\_file.txt';    // Создание папки    await createFolder(remoteFolder);    await waitForKeyPress();    // Загрузка файла    await uploadFile(localFilePath, remoteFile);    await waitForKeyPress();    // Скачивание файла    await downloadFile(remoteFile, './downloaded\_test\_file.txt');    await waitForKeyPress();    // Копирование файла    await copyFile(remoteFile, copiedFilePath);    await waitForKeyPress();    // Удаление файла    await deleteFile(remoteFile);    await waitForKeyPress();    // Удаление папки    await deleteFolder(remoteFolder);    await waitForKeyPress();    rl.close();  // Закрываем интерфейс чтения  })(); |