Task Report: Migration from FileZilla to SFTPGo Server

Uysm Web Development Team

January 20, 2026

1 Overview

The objective of this task is to implement the `SftpGoService` by fulfilling the requirements of the `IFileTransferService` interface. This transition moves our system from FileZilla to SFTPGo, utilizing the SFTPGo REST API for management actions. This approach ensures better security, scalability, and asynchronous execution through our background job client.

2 System Architecture

The diagram below illustrates the flow from the Portal UI through the Proxy and API layers, eventually triggering the background job that executes the `SftpGoService` logic.

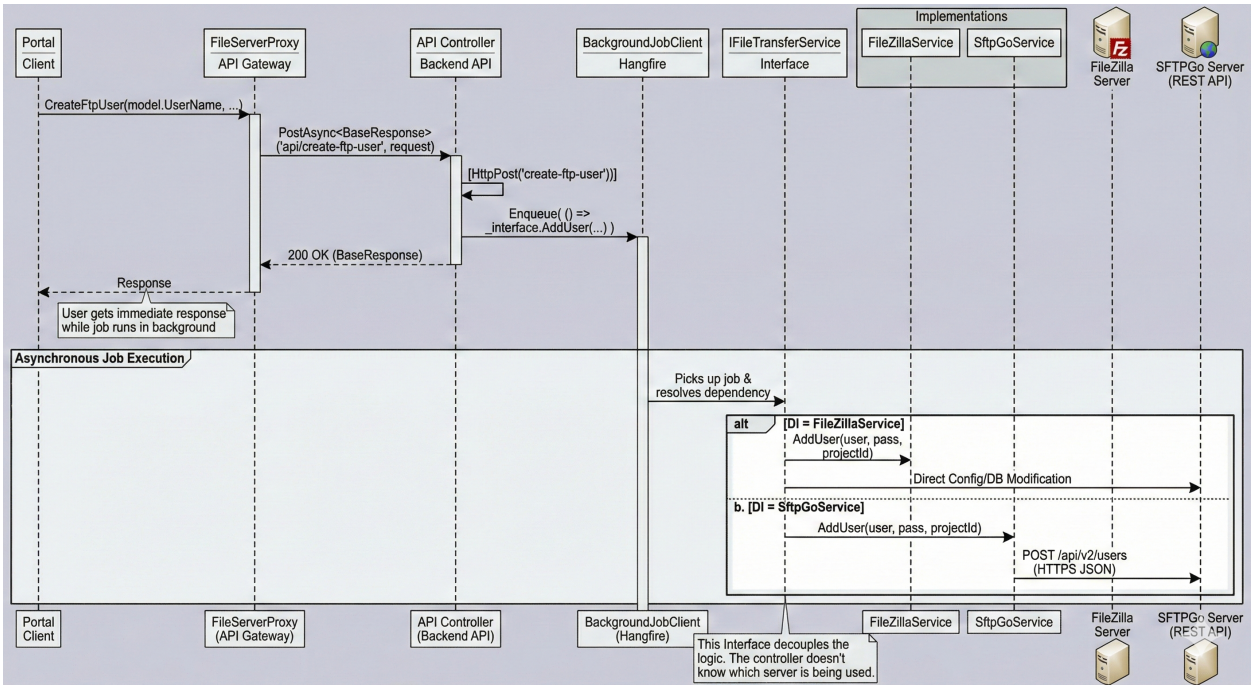

Figure 1: FlowDiagram

3 Interface Implementation Guidelines

The `SftpGoService` must implement the following methods using the SFTPGo REST API. All calls should be authenticated via the API Key or JWT token provided in the server configuration.

3.1 User Management

- **Local Environment:** Use the provided `docker-compose.yml` to spin up a local SFTPGo instance. You will use this as your target for REST API calls during development.
- **Service Configuration:** Implement the `SftpGoService` using `IOptions` or a configuration object. Do not hardcode the API URL or Keys.
- **Mocking Dependencies:** Since the `BackgroundJobClient` (Hangfire) is part of the main project, you should focus solely on the logic inside the `AddUser`, `DeleteUser`, etc., methods.
- **Deliverables:** Just develop on dotnet project we have sent to you. We will integrate your service to Uysm's web project.
- `Task AddUser(string username, string password, long projectId)`
  Logic: Send a `POST` request to `/api/v2/users`.
  - Set the `home_dir` based on the `projectId`.
  - Ensure the `password` is passed securely.
  - Define default permissions (e.g., list, download, upload).
- `void UpdateUser(string username, string password)`
  Logic: Send a `PUT` request to `/api/v2/users/{username}`. Update only the password field while maintaining existing directory permissions.
- `void DeleteUser(string username)`
  Logic: Send a `DELETE` request to `/api/v2/users/{username}`. Note: Confirm if the physical directory should be retained or deleted based on project policy.
- `void ActivateUsers(List<string> usernames)` / `DeactivateUsers(...)`
  Logic: Iterate through the list and update the `status` field (1 for active, 0 for disabled) via the `PUT` endpoint.

3.2 Folder and Permission Management

- `Task CreateFolder(long folderId)`
  Logic: Use the `/api/v2/folders` endpoint to define a virtual folder or ensure the physical path exists on the SFTPGo storage backend.
- `Task UpdateFolderPermissions(long folderId)`
  Logic: Update the `filesystem` or `virtual_folders` mapping for the users associated with this folder ID to reflect permission changes.
- `Task MoveFolderToRemovedFolders(long folderId)`
  Logic: Physically move the directory on the storage layer or update the user's root path to a "Recycle Bin" directory using the API.

3.3 Supplied Reference Implementation

A reference implementation and a test console application have been provided in the repository.

- **Authentication:** The service implements a `RefreshTokenAsync` method that handles the Basic-to-Bearer (JWT) handshake automatically.
- **Configuration:** Ensure you rename the `appsettings.json.template` to `appsettings.json` and update it with your local Docker credentials before running the test.
- **Cleanup:** The `DeleteUserAsync` method includes an optional `deleteData` flag. Use this cautiously as it triggers physical file deletion on the host volume.

4 Docker Deployment & FTP Bootstrapping

The SFTPGo server runs within a Docker container. Special attention must be paid to the FTP configuration, as it requires a specific bootstrap process to handle TLS certificates.

4.1 Docker Compose Configuration

The following configuration defines the environment. Note that port **8080** (Internal) is mapped to **9090** (Host) for the Web Admin and REST API.

Listing 1: docker-compose.yml snippet

```
services:
  sftpgo:
    image: drakkan/sftpgo:latest
    container_name: uysm-sftp-server
    user: "0:0"
    environment:
      - SFTPGO_HTTP__BINDINGS__0__PORT=8080
      - SFTPGO_HTTP__BINDINGS__0__ENABLE_REST_API=true
      - SFTPGO_FTPD__BINDINGS__0__PORT=2122
      - SFTPGO_FTPD__BINDINGS__0__TLS_MODE=1
      - SFTPGO_FTPD__PASSIVE_PORT_RANGE__START=50000
      - SFTPGO_FTPD__PASSIVE_PORT_RANGE__END=50100
    ports:
      - "9090:8080"
      - "2022:2022"
      - "2122:2122"
      - "50000-50100:50000-50100"
    volumes:
      - c:/sftp-data:/srv/sftpgo
```

4.2 The 2-Step FTP Activation Guide

To successfully activate FTP with TLS (FTPS), the following steps must be followed to avoid container startup failures due to missing certificates:

1. **Initial Run (Plain FTP):** Set `SFTPGO_FTPD__BINDINGS__0__TLS_MODE=0` in the environment variables and run `docker-compose up`. This allows the server to start without checking for certificate files.

2. **Certificate Generation:** While the server is running, generate or place your `ftps_cert.pem` and `ftps_key.pem` files into the mapped volume directory (`c:/sftp-data`).

3. **Enable TLS:** Once the certificates are present, change `TLS_MODE` back to `1` and restart the container. The server will now bind to the FTP port using the encrypted layer.

4.3 Create Admin User

You need to create user at the beginning in order to test your methods and you should connect this server via appsettings.json file.

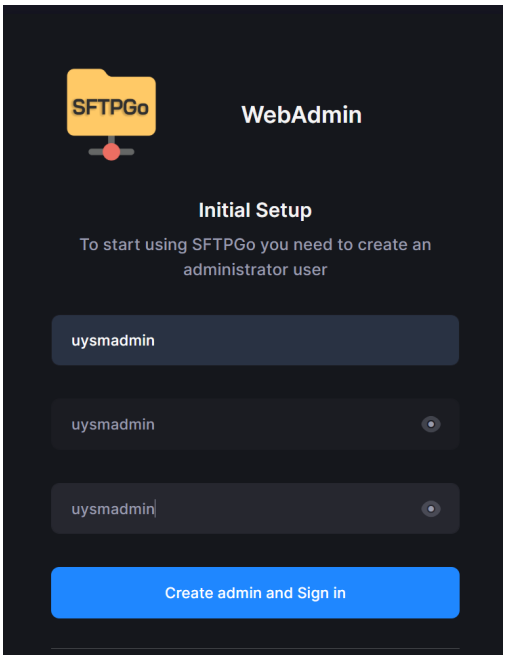

Figure 2: AdminUser

The example appsettings.json file already created you may change 9090 port as 8080.

```
    {
    "SFTPGoSettings": {
    "BaseUrl": "http://localhost:9090/api/v2",
    "AdminUsername": "uysmadmin",
    "AdminPassword": "uysmadmin",
    "InternalDataRoot": "/srv/sftpgo"
    }
}
```

4.4 Network Troubleshooting

If you cannot see the UI on port **9090**, ensure that the host firewall allows the passive port range (**50000-50100**). Without these ports, the FTP client will connect but fail to retrieve directory listings.

5 Crucial Technical Notes

1. **No Direct Connection Exposure:** Do not implement a method that returns a raw connection object. All logic must be contained within the service.
2. **Asynchronous Handshake:** Since the controller enqueues these tasks via Hangfire, ensure the `SftpGoService` handles its own internal `HttpClient` lifecycle (ideally using `IHttpClientFactory`).
3. **Error Handling:** The API will return specific HTTP status codes (e.g., 400 for user already exists). These must be caught and logged without breaking the background job.