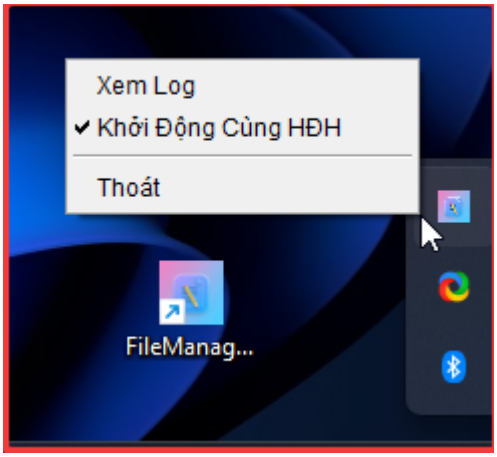
Mục lục

- Mục lục
- 1. Hướng dẫn chạy tool sửa file
- 2. Hướng dẫn chạy dịch vụ export file từ template, và dịch vụ convert file docx sang pdf
- 3. Hướng dẫn chạy service api để test (cung cấp các api upload / download file / xác thực)
- 4. Hướng dẫn chạy app react để test (cung cấp giao diện để test sửa file / convert file / export file)
- 5. Tài liệu mô tả chi tiết về tool sửa file
    - 5.1. Các tính năng của tool sửa file
    - 5.2. Quy trình sửa file
    - 5.3. Teckstack và các thư viện / công nghệ sử dụng
    - 5.4. Hướng dẫn debug local
    - 5.5. Hướng dẫn đóng gói
        - Cập nhật thông tin ứng dụng trước khi đóng gói
            - 1. Cập nhật tên ứng dụng và phiên bản
            - 2. Cập nhật icon ứng dụng
            - 3. Cập nhật thông tin hiển thị trong System Tray
        - Tiến hành đóng gói
    - 5.6. Cấu trúc dự án FileManagerClient
- 6. Tài liệu mô tả chi tiết về dịch vụ export file và convert file docx sang pdf
    - 6.1. Techstack và các thư viện / công nghệ sử dụng
    - 6.2. Tài liệu mô tả chi tiết về dịch vụ export file
    - 6.3. Tài liệu mô tả chi tiết về dịch vụ convert file docx sang pdf
    - 6.4. Hướng dẫn debug local
    - 6.5. Hướng dẫn đóng gói thành docker
    - 6.6. Cấu trúc dự án DocumentEditorService

1. Hướng dẫn chạy tool sửa file

- Cài đặt file FileManagerClient-1.0.exe, sau đó bấm vào biểu tượng shortcut FileManagerClient trên màn hình để khởi chạy.
- Nếu là chạy lần đầu, sẽ có một màn hình chào mừng và tùy chọn khởi động cùng hệ điều hành. Nhấn nút "Bắt đầu sử dụng để bắt đầu".



2. Hướng dẫn chạy dịch vụ export file từ template, và dịch vụ convert file docx sang pdf

- Sử dụng docker: `docker run -dp 81:8080 t2nh/document-editor-service-win:0.12`
- Khi chạy thành công API sẽ hoạt động ở địa chỉ http://localhost:81 (nếu chạy local)
- Có thể truy cập địa chỉ http://localhost:81/api-docs để xem swagger define các API đang có trong dịch vụ

3. Hướng dẫn chạy service api để test (cung cấp các api upload / download file / xác thực)

- cd FileManager
- chạy `mvn clean install chạy`mvn spring-boot:run`
- API sẽ hoạt động ở địa chỉ http://localhost:8080

4. Hướng dẫn chạy app react để test (cung cấp giao diện để test sửa file / convert file / export file)

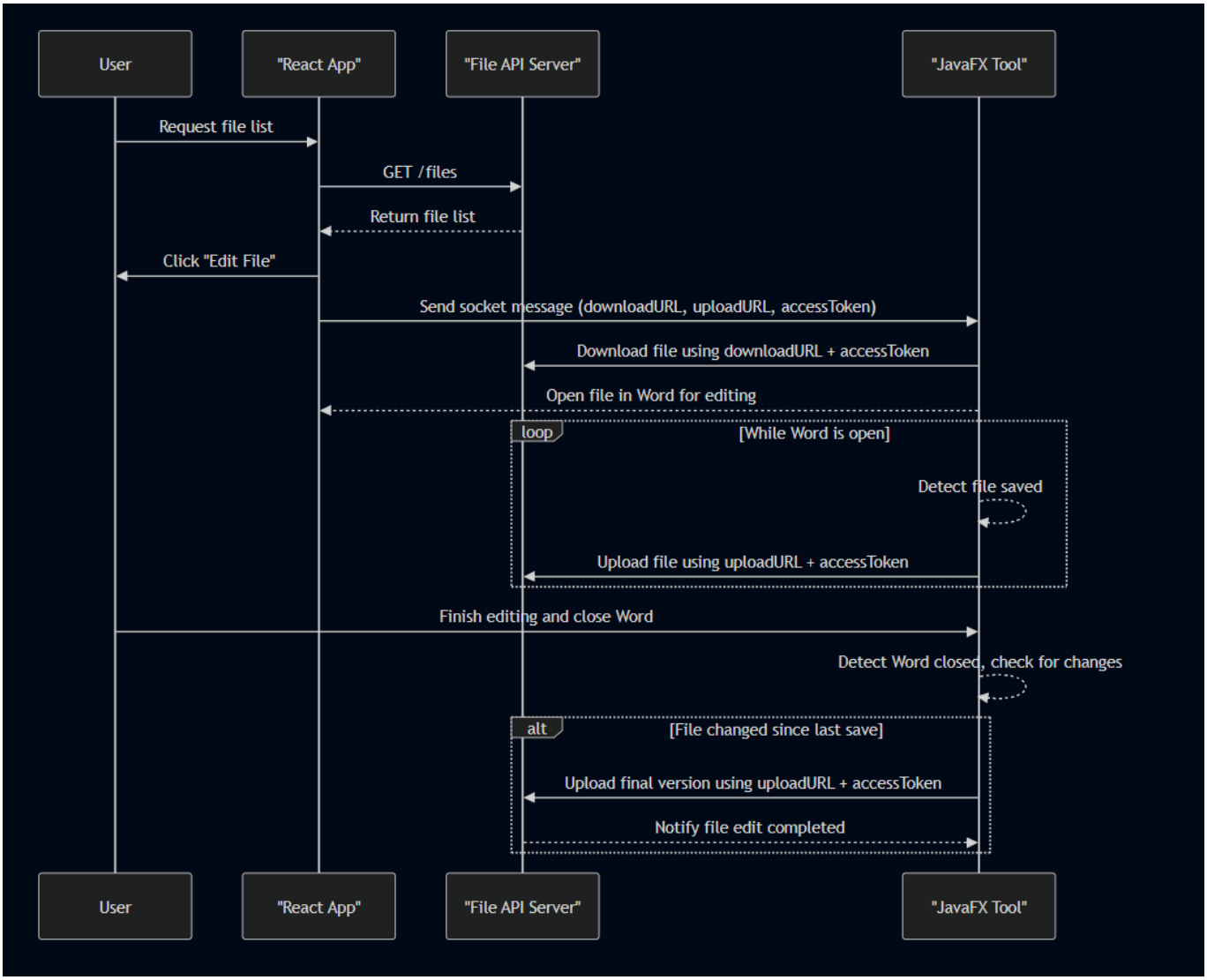
- cd document-editor-web
- npm install
- npm run dev
- app sẽ hoạt động ở địa chỉ http://localhost:5173/
- đăng nhập với tài khoản admin / mật khẩu admin

5. Tài liệu mô tả chi tiết về tool sửa file

5.1. Các tính năng của tool sửa file

1. **Tích hợp System Tray**: Ứng dụng chạy trong system tray, cho phép người dùng dễ dàng truy cập và quản lý.
2. **Tự động khởi động cùng hệ điều hành**: Tùy chọn cho phép ứng dụng tự động khởi động khi máy tính khởi động.
3. **Giao tiếp thời gian thực**: Sử dụng WebSocket/SockJS để giao tiếp với server theo thời gian thực. (Socket server lắng nghe ở địa chỉ http://localhost:61614)
4. **Tải và mở file từ URL**: Khả năng tải file từ URL và mở bằng Microsoft Word.
5. **Theo dõi thay đổi file**: Tự động phát hiện khi file được lưu hoặc đóng.
6. **Tự động upload file sau khi chỉnh sửa**: Tự động tải lên file đã chỉnh sửa lên server.
7. **Thông báo trạng thái**: Hiển thị thông báo qua System Tray và Toast Notification về trạng thái xử lý file.
8. **Xem log**: Cửa sổ xem log giúp theo dõi hoạt động của ứng dụng.
9. **Xác thực token**: Quản lý token xác thực để bảo mật khi tải và upload file.
10. **Kiểm tra thay đổi nội dung**: Sử dụng hash để xác định xem file có thay đổi nội dung không trước khi upload.

5.2. Quy trình sửa file



Quy trình sửa file

STT	Bước	Mô tả chi tiết
1	Người dùng nhấn nút "Sửa File"	Người dùng chọn file cần sửa và nhấn nút "Sửa File" trên giao diện web
2	Web App gửi yêu cầu sửa file	Web App gọi phương thức fileService.editFile(fileId) để gửi yêu cầu sửa file thông qua dịch vụ STOMP
3	Chuẩn bị URL download và upload	Web App tạo URL download và URL upload cho file, kèm theo token xác thực
4	Gửi yêu cầu qua STOMP	Web App gửi yêu cầu sửa file qua STOMP kèm theo URL download, URL upload và token xác thực
5	DocumentEditorService nhận yêu cầu	DocumentEditorService nhận yêu cầu sửa file từ STOMP
6	Tải file từ URL download	DocumentEditorService tải file từ URL download được cung cấp
7	Lưu file tạm thời	DocumentEditorService lưu file vào thư mục tạm thời trên máy chủ

STT	Bước	Mô tả chi tiết
8	Mở file bằng Microsoft Word	DocumentEditorService yêu cầu FileManagerClient mở file bằng Microsoft Word
9	FileManagerClient mở file	FileManagerClient nhận yêu cầu và mở file bằng Microsoft Word trên máy người dùng
10	Người dùng chỉnh sửa file	Người dùng thực hiện các thay đổi trên file trong Microsoft Word
11	Người dùng lưu và đóng file	Người dùng lưu các thay đổi và đóng file
12	FileManagerClient phát hiện file đã đóng	FileManagerClient theo dõi và phát hiện khi file đã được đóng
13	Tải file đã chỉnh sửa lên server	FileManagerClient tải file đã chỉnh sửa lên server thông qua URL upload
14	Cập nhật file trong hệ thống	Hệ thống cập nhật file trong cơ sở dữ liệu với phiên bản mới
15	Thông báo hoàn thành	Hệ thống thông báo cho người dùng rằng quá trình sửa file đã hoàn thành thành công

5.3. Teckstack và các thư viện / công nghệ sử dụng

- **Phiên bản Java**: JDK 17+
- **JavaFX SDK**: Version 21
- **Spring Boot**: Version 3.2.5
- **Các thư viện chính**:
  - Maven (quản lý dự án và build)
  - Spring Boot (framework chính)
  - Spring WebSocket (hỗ trợ WebSocket)
  - Gson (xử lý JSON)
  - JavaFX (giao diện người dùng)
  - SockJS (giao tiếp WebSocket với fallback)
  - STOMP (giao thức messaging)
  - Logback (ghi log)
  - ThreadPoolTaskScheduler (quản lý tác vụ)
- **Cơ chế giao tiếp**:
  - WebSocket với SockJS
  - STOMP protocol
- **Công nghệ khác**:
  - System Tray API (hiển thị icon trong khay hệ thống)
  - Windows/macOS Toast Notifications (thông báo hệ thống)

5.4. Hướng dẫn debug local

Để build dự án, chạy lệnh:

```
mvn clean package
```

Để chạy ứng dụng, sử dụng lệnh:

```
mvn javafx:run
```

Hoặc sau khi đóng gói:

```
java -jar target/FileManagerClient-1.0-SNAPSHOT.jar
```

5.5. Hướng dẫn đóng gói

Cập nhật thông tin ứng dụng trước khi đóng gói

1. Cập nhật tên ứng dụng và phiên bản

Để thay đổi tên ứng dụng và phiên bản, mở file `create-simple-exe.ps1` và chỉnh sửa các biến sau:

```
# Configuration
$appName = "FileManagerClient"  # Thay đổi tên ứng dụng tại đây
```

Để thay đổi phiên bản, mở file `pom.xml` và cập nhật thẻ version:

```
<groupId>com.filemanager</groupId>
<artifactId>FileManagerClient</artifactId>
<version>1.0-SNAPSHOT</version>  <!-- Thay đổi phiên bản tại đây -->
```

2. Cập nhật icon ứng dụng

Ứng dụng sử dụng hai file icon:

- File PNG: `src/main/resources/icon/app_icon.png`
- File ICO: `src/main/resources/icons/app_icon.ico`

Để thay đổi icon:

1. Chuẩn bị file icon mới ở cả hai định dạng PNG và ICO
    - File PNG nên có kích thước 256x256 pixels
    - File ICO nên chứa nhiều kích thước (16x16, 32x32, 48x48, 256x256)
2. Thay thế các file hiện có bằng file mới với cùng tên
3. Nếu muốn sử dụng tên file khác, cập nhật đường dẫn trong các file sau:

- `create-simple-exe.ps1`: Biến `$iconPng` và `$iconIco`
- `SystemTrayManager.java`: Hằng số `TRAY_ICON_PATH`
- `App.java`: Đường dẫn trong phương thức `setApplicationIcon()`

3. Cập nhật thông tin hiển thị trong System Tray

Để thay đổi tooltip và các thông tin hiển thị trong System Tray, mở file `src/main/resources/config.properties` và chỉnh sửa:

```
# System Tray Configuration
tray.icon.tooltip=File Manager Client  # Thay đổi tooltip tại đây
```

Tiến hành đóng gói

Sau khi đã cập nhật các thông tin cần thiết, chạy lệnh sau để đóng gói ứng dụng:

```
.\create-simple-exe.ps1
```

Quá trình đóng gói sẽ:

1. Build ứng dụng với Maven
2. Tạo file EXE installer sử dụng jpackage
3. Tạo shortcut trên màn hình và menu Start
4. Tạo script chạy tự động sau khi cài đặt

Sau khi hoàn thành, file installer sẽ được tạo với tên `FileManagerClient-[version].exe` trong thư mục hiện tại.

5.6. Cấu trúc dự án FileManagerClient

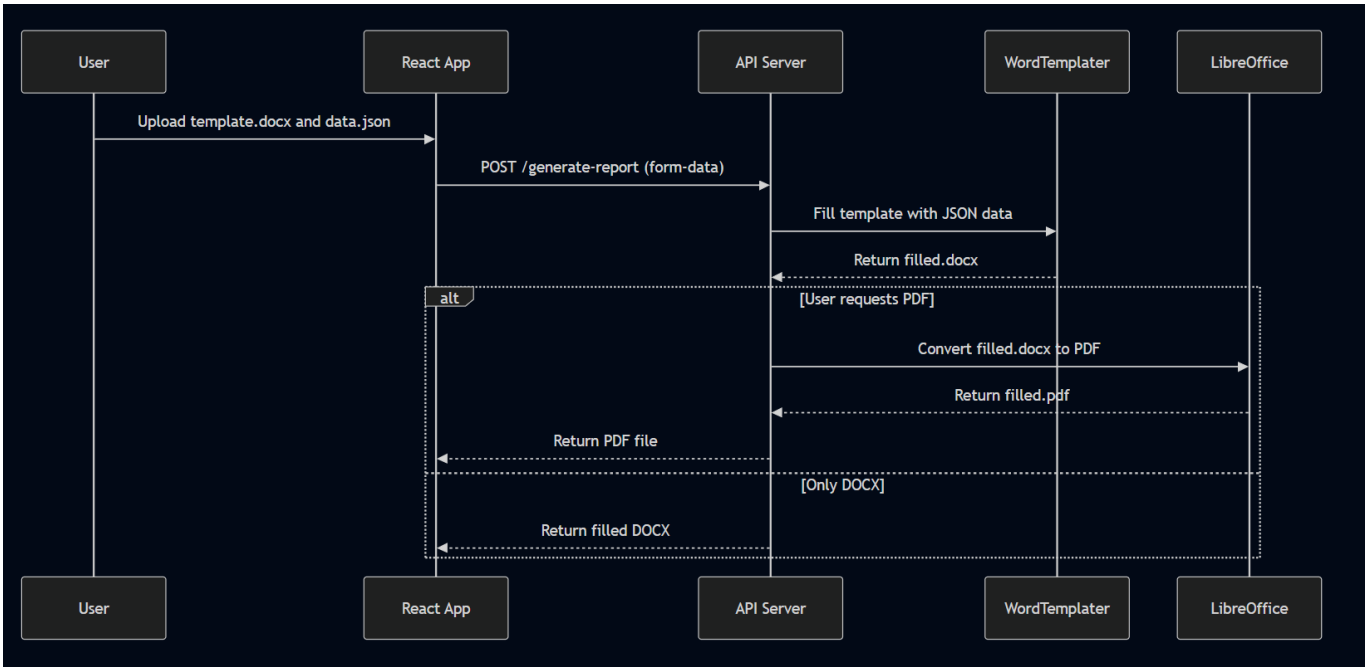
Dưới đây là cấu trúc thư mục và file chính của dự án FileManagerClient:

Thư mục/File	Mô tả
`src/main/java/com/filemanager/client/`	Thư mục chứa mã nguồn Java chính của ứng dụng
`src/main/java/com/filemanager/client/App.java`	Lớp chính khởi động ứng dụng JavaFX
`src/main/java/com/filemanager/client/SimpleApp.java`	Lớp khởi động đơn giản cho ứng dụng
`src/main/java/com/filemanager/client/service/`	Các dịch vụ của ứng dụng (thông báo, xử lý sự kiện)

Thư mục/File	Mô tả
src/main/java/com/filemanager/client/ui/	Các thành phần giao diện người dùng (LogWindow, WelcomeWindow)
src/main/java/com/filemanager/client/util/	Các tiện ích và công cụ hỗ trợ
src/main/java/com/filemanager/client/util/WordFileManager.java	Quản lý các hoạt động liên quan đến file Word (tải, mở, theo dõi, upload)
src/main/java/com/filemanager/client/util/SystemTrayManager.java	Quản lý hiển thị và tương tác với System Tray
src/main/java/com/filemanager/client/util/StompBrokerManager.java	Quản lý kết nối WebSocket/STOMP với server
src/main/java/com/filemanager/client/util/AuthTokenManager.java	Quản lý token xác thực cho các yêu cầu API
src/main/resources/	Thư mục chứa tài nguyên của ứng dụng
src/main/resources/application.properties	Cấu hình ứng dụng Spring Boot
src/main/resources/config.properties	Cấu hình riêng của ứng dụng
src/main/resources/logback.xml	Cấu hình ghi log
src/main/resources/icon/	Thư mục chứa các icon của ứng dụng
src/main/resources/fxml/	Thư mục chứa các file FXML định nghĩa giao diện
src/test/	Thư mục chứa mã nguồn kiểm thử
pom.xml	File cấu hình Maven, định nghĩa các dependency và cấu hình build

Thư mục/File	Mô tả
`create-simple-exe.ps1`	Script PowerShell để đóng gói ứng dụng thành file EXE
`run.bat`	Script batch để chạy ứng dụng với hỗ trợ System Tray
`run.ps1`	Script PowerShell để chạy ứng dụng với hỗ trợ System Tray
`run_tests.bat`	Script batch để chạy các bài kiểm thử
`SYSTEM_TRAY_README.md`	Tài liệu hướng dẫn về chức năng System Tray

6. Tài liệu mô tả chi tiết về dịch vụ export file và convert file docx sang pdf



Quy trình export file và convert file docx sang pdf

STT	Bước	Mô tả chi tiết
1	Người dùng tải lên template.docx và data.json	Người dùng tải lên file template Word và dữ liệu JSON thông qua giao diện React App
2	React App gửi yêu cầu tạo báo cáo	React App gửi yêu cầu POST /generate-report kèm theo form-data chứa template và dữ liệu JSON đến API Server

STT	Bước	Mô tả chi tiết
3	API Server gửi yêu cầu đến WordTemplater	API Server chuyển tiếp template và dữ liệu JSON đến dịch vụ WordTemplater để điền dữ liệu vào template
4	WordTemplater điền dữ liệu vào template	WordTemplater xử lý template, thay thế các placeholder bằng dữ liệu từ JSON và tạo ra file filled.docx
5	WordTemplater trả về file filled.docx	WordTemplater trả về file đã điền dữ liệu cho API Server
6	API Server trả về file filled.docx	API Server trả về file đã điền dữ liệu cho React App
7	[Nếu người dùng yêu cầu PDF] API Server gửi yêu cầu chuyển đổi	Nếu người dùng yêu cầu định dạng PDF, API Server gửi file filled.docx đến LibreOffice để chuyển đổi
8	LibreOffice chuyển đổi filled.docx sang PDF	LibreOffice thực hiện chuyển đổi file Word sang định dạng PDF
9	LibreOffice trả về file filled.pdf	LibreOffice trả về file PDF đã chuyển đổi cho API Server
10	API Server trả về file filled.pdf	API Server trả về file PDF cho React App
11	React App trả về file cho người dùng	React App hiển thị file kết quả (DOCX hoặc PDF) cho người dùng tải xuống

6.1. Techstack và các thư viện / công nghệ sử dụng

- **Framework**: ASP.NET Core 9.0
- **Ngôn ngữ**: C# (.NET 9.0)
- **Các thư viện chính**:
  - **DocumentFormat.OpenXml** (2.20.0): Xử lý file Word DOCX
  - **WordTemplater** (2.1.5): Thư viện mail merge để điền dữ liệu vào template Word
  - **Serilog.AspNetCore** (8.0.1): Ghi log
  - **Swashbuckle.AspNetCore** (6.5.0): Tạo Swagger API documentation
  - **Microsoft.AspNetCore.Authentication.JwtBearer** (9.0.0): Xác thực JWT
- **Công cụ chuyển đổi PDF**:
  - **LibreOffice**: Chuyển đổi file DOCX sang PDF (mặc định)
  - **Docx4j**: Chuyển đổi file DOCX sang PDF (tùy chọn thay thế)
- **Cơ chế xác thực**:
  - JWT Bearer Token
  - API Key

6.2. Tài liệu mô tả chi tiết về dịch vụ export file

API Endpoint

- **URL**: `/api/document/export-file-from-template`
- **Method**: POST
- **Content-Type**: multipart/form-data
- **Authentication**: JWT Bearer Token hoặc API Key (header `X-API-Key`)

Tham số

Tham số	Loại	Mô tả	Bắt buộc	Giá trị mặc định
file	IFormFile	File template Word (.docx)	Có	-
data	string	Dữ liệu JSON để điền vào template	Có	-
outputExtension	string	Định dạng đầu ra (docx hoặc pdf)	Không	docx
outputType	string	Kiểu đầu ra (file hoặc base64)	Không	file
inputType	string	Kiểu đầu vào (file hoặc base64)	Không	file

Kết quả trả về

- Nếu `outputType=file`:
  - **Content-Type**: application/vnd.openxmlformats-officedocument.wordprocessingml.document (docx) hoặc application/pdf (pdf)
  - **Body**: File kết quả dưới dạng binary
- Nếu `outputType=base64`:
  - **Content-Type**: application/json
  - **Body**: JSON object chứa chuỗi base64 của file kết quả

```
{
  "base64": "UEsDBBQABgAIAAAAIQA..."
}
```

Ví dụ Request

```
POST /api/document/export-file-from-template?outputExtension=pdf&outputType=file
HTTP/1.1
Host: localhost:81
X-API-Key: document-editor-fixed-api-key-2024-production
Content-Type: multipart/form-data; boundary=----WebKitFormBoundary7MA4YWxkTrZu0gW

------WebKitFormBoundary7MA4YWxkTrZu0gW
Content-Disposition: form-data; name="file"; filename="template.docx"
Content-Type: application/vnd.openxmlformats-
officedocument.wordprocessingml.document

(binary data)
------WebKitFormBoundary7MA4YWxkTrZu0gW
Content-Disposition: form-data; name="data"

{
  "company": "ACME Corporation",
  "date": "2025-05-09",
```

```
  "customer": {
    "name": "John Doe",
    "address": "123 Main St, Anytown, USA"
  },
  "items": [
    {
      "name": "Product A",
      "quantity": 2,
      "price": 100
    },
    {
      "name": "Product B",
      "quantity": 1,
      "price": 200
    }
  ]
}
------WebKitFormBoundary7MA4YWxkTrZu0gW--
```

6.3. Tài liệu mô tả chi tiết về dịch vụ convert file docx sang pdf

API Endpoint

- **URL**: `/api/document/convert-word-to-pdf`
- **Method**: POST
- **Content-Type**: multipart/form-data
- **Authentication**: JWT Bearer Token hoặc API Key (header `X-API-Key`)

Tham số

Tham số	Loại	Mô tả	Bắt buộc	Giá trị mặc định
file	IFormFile	File Word (.docx) cần chuyển đổi	Có	-
inputType	string	Kiểu đầu vào (file hoặc base64)	Không	file
outputType	string	Kiểu đầu ra (file hoặc base64)	Không	file

Kết quả trả về

- Nếu `outputType=file`:
  - **Content-Type**: application/pdf
  - **Body**: File PDF kết quả dưới dạng binary
- Nếu `outputType=base64`:
  - **Content-Type**: application/json
  - **Body**: JSON object chứa chuỗi base64 của file PDF

```
{
  "base64": "JVBERi0xLjUNCiW1tbW1..."
}
```

Ví dụ Request

```
POST /api/document/convert-word-to-pdf?outputType=file HTTP/1.1
Host: localhost:81
X-API-Key: document-editor-fixed-api-key-2024-production
Content-Type: multipart/form-data; boundary=----WebKitFormBoundary7MA4YWxkTrZu0gW

------WebKitFormBoundary7MA4YWxkTrZu0gW
Content-Disposition: form-data; name="file"; filename="document.docx"
Content-Type: application/vnd.openxmlformats-
officedocument.wordprocessingml.document

(binary data)
------WebKitFormBoundary7MA4YWxkTrZu0gW--
```

6.4. Hướng dẫn debug local

Để debug dịch vụ DocumentEditorService trên môi trường local, thực hiện các bước sau:

1. **Cài đặt các yêu cầu tiên quyết**:
    - .NET 9.0 SDK
    - LibreOffice (cho chuyển đổi PDF)
2. **Khôi phục các package**:

```
cd DocumentEditorService/src/DocumentEditor.Api
dotnet restore
```

3. **Chạy ứng dụng trong chế độ development**:

```
dotnet run --environment Development
```

4. **Chạy ứng dụng với cổng tùy chỉnh**:

```
dotnet run --urls=http://localhost:5052
```

5. **Xem Swagger API Documentation**:

- Truy cập http://localhost:5052/api-docs trong trình duyệt

6. **Debug với Visual Studio Code**:
   - Mở thư mục DocumentEditorService trong VS Code
   - Cài đặt extension C# Dev Kit
   - Nhấn F5 để bắt đầu debug

7. **Xem logs**:
   - Logs được lưu trong thư mục `DocumentEditorService/src/DocumentEditor.Api/logs`

6.5. Hướng dẫn đóng gói thành docker

Chuẩn bị

1. **Cài đặt Docker Desktop** (nếu chưa có):
   - Tải và cài đặt từ Docker Desktop

2. **Đảm bảo Docker đang chạy**:

```
docker --version
```

Đóng gói ứng dụng

1. **Build Docker image**:

```
cd DocumentEditorService
docker build -t document-editor-service:latest .
```

2. **Kiểm tra image đã tạo**:

```
docker images
```

3. **Chạy container từ image**:

```
docker run -d -p 81:8080 --name document-editor-service document-editor-
service:latest
```

4. **Kiểm tra container đang chạy**:

```
docker ps
```

5. **Xem logs của container**:

```
docker logs document-editor-service
```

Sử dụng image có sẵn

Để sử dụng image đã được đóng gói sẵn:

```
docker run -dp 81:8080 t2nh/document-editor-service-win:0.12
```

Sau khi chạy, dịch vụ sẽ hoạt động tại địa chỉ http://localhost:81 và Swagger API documentation có thể truy cập tại http://localhost:81/api-docs.

6.6. Cấu trúc dự án DocumentEditorService

Dưới đây là cấu trúc thư mục và file chính của dự án DocumentEditorService:

Thư mục/File	Mô tả
`src/`	Thư mục chứa mã nguồn chính của dự án
`src/DocumentEditor.Api/`	Dự án API chính, chứa controllers và cấu hình
`src/DocumentEditor.Api/Controllers/`	Các controllers xử lý các yêu cầu HTTP
`src/DocumentEditor.Api/Controllers/DocumentController.cs`	Controller xử lý các API liên quan đến tài liệu (export, convert)
`src/DocumentEditor.Api/Hubs/`	Các SignalR hubs cho giao tiếp thời gian thực
`src/DocumentEditor.Api/Middleware/`	Các middleware xử lý yêu cầu HTTP
`src/DocumentEditor.Api/Program.cs`	Điểm khởi đầu của ứng dụng, cấu hình dịch vụ
`src/DocumentEditor.Api/appsettings.json`	Cấu hình ứng dụng chính
`src/DocumentEditor.Api/appsettings.Development.json`	Cấu hình cho môi trường phát triển
`src/DocumentEditor.Api/appsettings.Production.json`	Cấu hình cho môi trường sản xuất

Thư mục/File	Mô tả
src/DocumentEditor.Application/	Dự án chứa logic nghiệp vụ và interfaces
src/DocumentEditor.Application/Interfaces/	Các interfaces định nghĩa các dịch vụ
src/DocumentEditor.Application/Interfaces/IDocumentService.cs	Interface định nghĩa dịch vụ xử lý tài liệu
src/DocumentEditor.Application/Services/	Các lớp triển khai logic nghiệp vụ
src/DocumentEditor.Application/Services/DocumentService.cs	Dịch vụ xử lý tài liệu (mail merge, chuyển đổi)
src/DocumentEditor.Infrastructure/	Dự án chứa các triển khai cụ thể (xử lý file, PDF)
src/DocumentEditor.Infrastructure/Services/	Các dịch vụ cơ sở hạ tầng
src/DocumentEditor.Infrastructure/FileStorage/	Các lớp xử lý lưu trữ file
src/DocumentEditor.Infrastructure/PdfConversion/	Các lớp xử lý chuyển đổi PDF
src/DocumentEditor.Infrastructure/WordProcessing/	Các lớp xử lý tài liệu Word
tests/	Thư mục chứa các dự án kiểm thử
tests/DocumentEditor.Application.Tests/	Kiểm thử cho dự án Application
tests/DocumentEditor.Infrastructure.Tests/	Kiểm thử cho dự án Infrastructure
Dockerfile	File cấu hình Docker chính
Dockerfile.win	File cấu hình Docker cho Windows
Dockerfile.mac	File cấu hình Docker cho MacOS
DocumentEditor.sln	File solution của Visual Studio
postman/	Thư mục chứa các tài liệu kiểm thử API với Postman