Upload de Arquivos:

1. Incluir no application.properties o endereço do diretório onde os arquivos serão armazenados;

2. Criar um Controller com os recursos de manipulação de arquivos que serão disponibilizados (upload, delete, etc);

3. Criar um Service para manipular os arquivos;

4. Criar um VO para transitar informações sobre o arquivo manipulado;

1. Incluir no application.properties o endereço do diretório onde os arquivos serão armazenados;

```
## Diretório para armazenamento dos arquivos de foto
arquivos.imagem = D:/_workspace/residencia_software/arquivos_imagem
```

2. Criar um Controller com os recursos de manipulação de arquivos que serão disponibilizados (upload, delete, etc);

```java
@RestController
@RequestMapping("/arquivos")
public class ArquivosController {

    @Autowired
    ArquivosService arquivosService;

    @PostMapping("/uploadFile")
    public ArquivosVO uploadFile(@RequestParam("file") MultipartFile file) {
        String fileName = arquivosService.storeFile(file);

        String fileDownloadUri = ServletUriComponentsBuilder.fromCurrentContextPath()
                .path("/foto/downloadFile/")
                .path(fileName)
                .toUriString();

        return new ArquivosVO(fileName, fileDownloadUri,
                file.getContentType(), file.getSize());
    }

    @PostMapping("/deleteFile")
    public boolean deleteFile(@RequestParam("file") String file) {
        return arquivosService.deleteFile(file);
    }
}
```

3. Criar um Service para manipular os arquivos;

```java
@Service
public class ArquivosService {

    private Path fileStorageLocation;

    @Value("${arquivos.imagem}")
    private String dirArquivosImagem;

    private void createDirectory() {
        this.fileStorageLocation = Paths.get(dirArquivosImagem)
                .toAbsolutePath().normalize();
        try {
            Files.createDirectories(this.fileStorageLocation);
        } catch (Exception ex) {
            throw new ArquivosException("Não foi possível criar o diretório para armazenar o arquivo.", ex);
        }
    }

    public String storeFile(MultipartFile file) {
        createDirectory();

        // Limpeza no nome do arquivo
        String fileName = StringUtils.cleanPath(file.getOriginalFilename());

        try {
            if(fileName.contains("..")) {
                throw new ArquivosException("Nome de arquivo inválido! " + fileName);
            }
            // Copia/Sobrescrita do arquivo na pasta de destino
            Path targetLocation = this.fileStorageLocation.resolve(fileName);
            Files.copy(file.getInputStream(), targetLocation, StandardCopyOption.REPLACE_EXISTING);

            return fileName;
        } catch (IOException ex) {
            throw new ArquivosException("Ocorreu um erro e não foi possível armazenar o arquivo " + fileName, ex);
        }
    }

    public boolean deleteFile(String file) {
        String fileName = StringUtils.cleanPath(file);

        try {
            if(fileName.contains("..")) {
                throw new ArquivosException("Desculpe, o nome do arquivo contém uma sequência de caminho inválida! " + fileName);
            }
            Path targetLocation = this.fileStorageLocation.resolve(fileName);

            return Files.deleteIfExists(targetLocation);

        } catch (IOException ex) {
            throw new ArquivosException("O arquivo " + fileName + " não foi encontrado no servidor!", ex);
        }
    }
}
```

4. Criar um VO para transitar informações sobre o arquivo manipulado;

```java
public class ArquivosVO {
    private String fileName;
    private String fileDownloadUri;
    private String fileType;
    private long size;

    public ArquivosVO(String fileName, String fileDownloadUri, String fileType, long size) {
        this.fileName = fileName;
        this.fileDownloadUri = fileDownloadUri;
        this.fileType = fileType;
        this.size = size;
    }

    public String getFileName() {
        return fileName;
    }

    public void setFileName(String fileName) {
        this.fileName = fileName;
    }

    public String getFileDownloadUri() {
        return fileDownloadUri;
    }

    public void setFileDownloadUri(String fileDownloadUri) {
        this.fileDownloadUri = fileDownloadUri;
    }

    public String getFileType() {
        return fileType;
    }

    public void setFileType(String fileType) {
        this.fileType = fileType;
    }

    public long getSize() {
        return size;
    }

    public void setSize(long size) {
        this.size = size;
    }

}
```

Testando o Upload de Arquivos:

No Postman, selecionar o método POST. Na aba “Body” selecionar “form-data”. Incluir em “Key” o nome do parâmetro definido no Controller e em “File” alterar o tipo para “File”. Por fim, selecionar o arquivo a ser enviado.