API Horse - Gestion des fichiers

Explication Du Code **API REST en Delphi avec le framework Horse**

Ce code est une **API REST en Delphi avec le framework Horse**, qui permet d'uploader et de récupérer des fichiers depuis un serveur. Il utilise **Horse.OctetStream** pour gérer les flux binaires (fichiers) et stocke les fichiers dans un répertoire local.

Explication du Code

Constantes et Sécurité

```
const
  VALID_FILE_TYPES: array[0..5] of string = ('.png', '.jpg',
'.svg','.pdf','.doc','.ppt');
  AUTH_TOKEN = 'Bearer 123456789';
  LOG_FILE = 'logs/api.log';
```

- **`VALID_FILE_TYPES`** : Liste des types de fichiers acceptés.
- **`AUTH_TOKEN`** : Clé d'authentification (devrait être remplacée par un vrai mécanisme sécurisé).
- **`LOG_FILE`** : Fichier où seront enregistrés les logs des actions effectuées.

Gestion des Logs

```
procedure LogMessage(Msg: string);
var
  LogFile: TextFile;
begin
  AssignFile(LogFile, LOG_FILE);
  if FileExists(LOG_FILE) then Append(LogFile) else
Rewrite(LogFile);
  Writeln(LogFile, FormatDateTime('YYYY-MM-DD HH:NN:SS', Now) + '
- ' + Msg);
  CloseFile(LogFile);
end;
```

Cette fonction écrit des logs dans **logs/api.log** pour garder une trace des événements de l'API.

Vérification de l'authentification

```
function IsAuthorized(Req: THorseRequest): Boolean;
begin
  Result := Req.Headers['Authorization'] = AUTH_TOKEN;
end;
```

- Vérifie si le client envoie le bon **token d’authentification** dans l’en-tête de la requête HTTP.

Validation du type de fichier

```
function IsValidFileType(const FileName: string): Boolean;
var
  Ext: string;
  I: Integer;
begin
  Result := False;
  Ext := LowerCase(ExtractFileExt(FileName));
  for I := Low(VALID_FILE_TYPES) to High(VALID_FILE_TYPES) do
    if Ext = VALID_FILE_TYPES[I] then Exit(True);
end;
```

- Vérifie si l’extension du fichier demandé ou uploadé correspond aux **types autorisés**.

Endpoints de l’API

L’API expose **deux routes** pour gérer le téléchargement et l’upload de fichiers.

---

1. Télécharger un fichier (GET `/stream/:fichier`)

Fonction : `GetStream`

```
procedure GetStream(Req: THorseRequest; Res: THorseResponse);
var
  LStream: TFileStream;
  FilePath, FileName: string;
begin
  if not IsAuthorized(Req) then
  begin
    Res.Status(THTTPStatus.Unauthorized).Send('401 Unauthorized');
    LogMessage('Unauthorized access attempt.');
    Exit;
  end;

  FileName := Req.Params['fichier'];
  FilePath := ExtractFilePath(ExtractFilePath(ParamStr(0))) +
'ImageBarber/Images/' + FileName;

  if not IsValidFileType(FileName) then
  begin
    Res.Status(THTTPStatus.BadRequest).Send('400 Bad Request -
Invalid file type');
    LogMessage('Invalid file type access: ' + FileName);
    Exit;
```

```
  end;

  if not FileExists(FilePath) then
  begin
    Res.Status(THTTPStatus.NotFound).Send('404 Not Found');
    Exit;
  end;

  try
    LStream := TFileStream.Create(FilePath, fmOpenRead);
    Res.Send<TStream>(LStream).ContentType('application/octet-
stream');
    LogMessage('File served: ' + FilePath);
  except
    on E: Exception do
    begin
      Res.Status(THTTPStatus.InternalServerError).Send('500
Internal Server Error');
    end;
  end;
end;
```

Ce que fait cette fonction

1. Vérifie si la requête est **autorisée**.
2. Vérifie si le fichier demandé **existe** et si son type est **valide**.
3. Renvoie le fichier sous forme de **flux binaire (octet-stream)**.

---

2. Uploader un fichier (POST `/stream/:fichier`)

Fonction : `PostStream`

```
procedure PostStream(Req: THorseRequest; Res: THorseResponse);
var
  FileName, FilePath: string;
begin
  if not IsAuthorized(Req) then
  begin
    Res.Status(THTTPStatus.Unauthorized).Send('401 Unauthorized');
    LogMessage('Unauthorized upload attempt.');
    Exit;
  end;

  FileName := Req.Params['fichier'];
  if not IsValidFileType(FileName) then
  begin
    Res.Status(THTTPStatus.BadRequest).Send('400 Bad Request -
Invalid file type');
    Exit;
  end;
```

```
  if Req.Body.IsEmpty then
  begin
    Res.Status(THTTPStatus.BadRequest).Send('400 Bad Request - No
file uploaded');
    Exit;
  end;

  FilePath := ExtractFilePath(ExtractFilePath(ParamStr(0))) +
'ImageBarber/Images/' + FileName;

  try
    Req.Body<TBytesStream>.SaveToFile(FilePath);
    Res.Status(THTTPStatus.Created).Send('201 Created');
    writeln('File uploaded: ' + FilePath);
  except
    on E: Exception do
    begin
      Res.Status(THTTPStatus.InternalServerError).Send('500
Internal Server Error');
    end;
  end;
end;
```

Ce que fait cette fonction

1. Vérifie si la requête est **autorisée**.
2. Vérifie si le type de fichier est **valide**.
3. Vérifie si un fichier a bien été **envoyé** dans le corps de la requête.
4. Sauvegarde le fichier dans le dossier **`ImageBarber/Images/`**.

Démarrage du serveur

```
procedure running (Horse:THorse);
begin
   writeln('Running Server : '+Thorse.Port.ToString + 'Host :
'+Horse.Host);
end;

begin
  THorse.Use(OctetStream);

THorseOctetStreamConfig.GetInstance.AcceptContentType.Add('applica
tion/octet-stream');

  THorse.Get('/stream/:fichier', GetStream);
  THorse.Post('/stream/:fichier', PostStream);

  LogMessage('API started on port 9001.');
  THorse.Listen(9001,@running);
end.
```

Ce que fait cette partie

- Initialise le serveur avec **Horse**.
- Active la gestion des **flux binaires** (`OctetStream`).
- Déclare les **routes** (`GET` et `POST`).
- **Lance le serveur** sur le **port 9001**.

Comment tester cette API ?

Lancer le serveur

Compilez et exécutez le programme, et assurez-vous qu'il écoute sur le **port 9001**.

Télécharger un fichier

Commande `cURL`

```
curl -H "Authorization: Bearer 123456789" -o image.jpg
http://localhost:9001/stream/image.jpg
```

- Récupère **image.jpg** du serveur.

Uploader un fichier

Commande `cURL`

```
curl -X POST -H "Authorization: Bearer 123456789" --data-binary
"@image.jpg" http://localhost:9001/stream/image.jpg
```

- Envoie **image.jpg** vers le serveur.

Résumé

API en Delphi avec Horse
Upload & téléchargement de fichiers
Sécurisation avec un token d’authentification
Gestion des logs
Filtrage des types de fichiers acceptés
Facilement testable avec `cURL`

Tu veux ajouter d'autres fonctionnalités, comme la suppression de fichiers ou une base de données ?

Utilisation Par Front-End -Developper

Cette API REST développée en **Delphi avec Horse** permet d'**uploader** et de **télécharger** des fichiers dans un répertoire serveur. Elle prend en charge plusieurs types de fichiers et est protégée par un système d'authentification basique via un token.

Base URL

```
http://localhost:9001
```

Authentification

L'API utilise un **token Bearer** pour sécuriser les requêtes.

Chaque requête doit inclure l'en-tête suivant :

```
Authorization: Bearer 123456789
```

> Remplacez `123456789` par un vrai système de gestion des tokens en production.

---

1. Uploader un fichier

➤ Requête

```
POST /stream/:fichier
```

Paramètre	Type	Description
`fichier`	`string`	Nom du fichier à uploader

➤ Corps de la requête

Le fichier doit être envoyé sous forme de **flux binaire** (`application/octet-stream`).

➤ Exemple avec cURL

```
curl -X POST \
     -H "Authorization: Bearer 123456789" \
     --data-binary "@image.jpg" \
     http://localhost:9001/stream/image.jpg
```

➤ Réponses possibles

Code HTTP	Description
`201 Created`	Fichier uploadé avec succès
`400 Bad Request`	Aucun fichier envoyé ou type de fichier non valide
`401 Unauthorized`	Authentification invalide
`500 Internal Server Error`	Erreur lors de l'enregistrement

---

2. Télécharger un fichier

➤ **Requête**

```
GET /stream/:fichier
```

Paramètre	Type	Description
`fichier`	`string`	Nom du fichier à télécharger

➤ **Exemple avec cURL**

```
curl -H "Authorization: Bearer 123456789" \
     -o image.jpg \
     http://localhost:9001/stream/image.jpg
```

➤ **Réponses possibles**

Code HTTP	Description
`200 OK`	Retourne le fichier sous forme de flux binaire
`400 Bad Request`	Type de fichier non autorisé
`401 Unauthorized`	Authentification invalide
`404 Not Found`	Fichier non trouvé
`500 Internal Server Error`	Erreur lors de la récupération

Formats de fichiers supportés

L'API accepte uniquement les fichiers suivants :

- `.png`
- `.jpg`
- `.svg`
- `.pdf`
- `.doc`
- `.ppt`

Exploitation avec JavaScript (Axios)

Uploader un fichier

```
const uploadFile = async (file) => {
    const formData = new FormData();
    formData.append('file', file);

    const response = await fetch(`http://localhost:9001/stream/$
{file.name}`, {
        method: 'POST',
        headers: {
            'Authorization': 'Bearer 123456789'
        },
```

```
        body: file
    });

    if (response.status === 201) {
        console.log('Fichier uploadé avec succès !');
    } else {
        console.error('Erreur lors de l'upload', await
response.text());
    }
};
```

Télécharger un fichier

```
const downloadFile = async (fileName) => {
    const response = await fetch(`http://localhost:9001/stream/$
{fileName}`, {
        headers: {
            'Authorization': 'Bearer 123456789'
        }
    });

    if (response.status === 200) {
        const blob = await response.blob();
        const link = document.createElement('a');
        link.href = window.URL.createObjectURL(blob);
        link.download = fileName;
        link.click();
    } else {
        console.error('Erreur lors du téléchargement', await
response.text());
    }
};
```

Démarrer le serveur

Exécutez l'API sur le port **9001** :

```
./horseimageapi
```

Vous pouvez maintenant exploiter cette API depuis votre application Front-End !