Jour 2 : TP2
API de Transcription (Whisper.NET)

Parcours : Assistant vocal multimodal (.NET + Python + React)
Durée estimée : 1h15 – 1h45

Objectif pédagogique

Construire une **Minimal API .NET** qui :

1. Reçoit un **fichier audio** (multipart/form-data),
2. Convertit en **WAV 16 kHz mono** si nécessaire,
3. Transcrit avec **Whisper.NET**,
4. Retourne un **JSON** : transcription + segments + métadonnées.

Pré-requis

- .NET 8 ou 9 installé
- Modèle Whisper au format **ggml** (ex. ggml-small.bin)
- OS Windows (NAudio). Sur Linux/Mac, préférer **ffmpeg** pour la conversion.

Packages NuGet

```
dotnet add package Whisper.net --version 1.5.0
dotnet add package NAudio --version 2.2.1
```

Spécification d’API (contrat)

POST /api/transcription

- **Body (form-data)** : file = <audio> (.wav, .mp3, .m4a, .webm…)
- **Réponse (200 OK)** :

```
{
  "success": true,
  "language": "fr",
  "transcript": "Bonjour, ...",
  "segmentCount": 5,
  "segments": [
    { "start": 0.0, "end": 1.2, "text": "Bonjour" },
    { "start": 1.2, "end": 3.4, "text": "..." }
  ]
}
```

- **Erreurs** :
  - 400 → fichier manquant
  - 413 → fichier trop volumineux
  - 415 → type non supporté
  - 500 → modèle non chargé / erreur interne

Étapes à réaliser

Étape 1 — Créer la Minimal API

dotnet new web -n TranscriptionApi

Étape 2 — Initialiser Whisper.NET (chargement unique)

- Placer le modèle ggml-*.bin dans ./models/
- Vérifier l'existence et lever une erreur claire si absent

Étape 3 — Endpoint **POST /api/transcription**

- Lire multipart/form-data
- **Contrôles** : taille max (ex. 50 MB), Content-Type audio
- Sauvegarder en **temp** → convertir en **WAV 16k mono PCM** si besoin
- Lancer la **transcription segmentée** (streaming)
- Concaténer les segments → transcript
- Nettoyer les fichiers temporaires

Étape 4 — Conversion audio (Windows)

- Utiliser **NAudio** (MediaFoundationReader + MediaFoundationResampler)
- **Alternatif Linux/Mac** : appeler ffmpeg (CLI) pour resampler en 16k mono

Extraits de code à intégrer

1) Initialisation Whisper

```
using Whisper.net;
var modelPath = Path.Combine(app.Environment.ContentRootPath, "models", "ggml-
small.bin");
if (!File.Exists(modelPath))
    throw new FileNotFoundException("Whisper model not found at " + modelPath);
var whisperFactory = WhisperFactory.FromPath(modelPath);
```

2) Endpoint principal

```
app.MapPost("/api/transcription", async (HttpContext context, CancellationToken ct) =>
{
    if (whisperFactory is null)
        return Results.Json(new { success = false, error = "Model not loaded" }, statusCode: 500);

    var form = await context.Request.ReadFormAsync(ct);
    var file = form.Files.GetFile("file");
    if (file is null || file.Length == 0)
        return Results.Json(new { success = false, error = "No file uploaded" }, statusCode: 400);

    if (file.Length > 50 * 1024 * 1024)
        return Results.Json(new { success = false, error = "File too large (max 50MB)" },
statusCode: 413);

    var contentType = (file.ContentType ?? "").ToLowerInvariant();
    if (!contentType.StartsWith("audio/") && !contentType.Contains("octet-stream"))
        return Results.Json(new { success = false, error = $"Unsupported content-type:
{contentType}" }, statusCode: 415);

    var tempIn = Path.GetTempFileName();
    var tempWav = tempIn;
    var semaphore = new SemaphoreSlim(1,1);
    const string LANGUAGE = "fr";

    await semaphore.WaitAsync(ct);
    try
    {
        await using (var fs = File.Create(tempIn))
            await file.CopyToAsync(fs, ct);

        // Convertir en WAV 16k mono si nécessaire
        tempWav = ConvertToWav16kMono(tempIn);
```

```
        var segments = new List<object>();
        var sb = new System.Text.StringBuilder();

        using var processor = whisperFactory.CreateBuilder()
            .WithLanguage(LANGUAGE)
            .WithPrompt("Transcription en français. Ponctuation automatique.")
            .Build();

        await using var audio = File.OpenRead(tempWav);
        await foreach (var s in processor.ProcessAsync(audio, ct))
        {
            var text = s.Text?.Trim();
            if (!string.IsNullOrEmpty(text))
            {
                segments.Add(new { start = s.Start.TotalSeconds, end = s.End.TotalSeconds, text });
                sb.Append(text).Append(' ');
            }
        }

        return Results.Json(new {
            success = true,
            language = LANGUAGE,
            transcript = sb.ToString().Trim(),
            segmentCount = segments.Count,
            segments
        });
    }
    catch (Exception ex)
    {
        return Results.Json(new { success = false, error = ex.Message }, statusCode: 500);
    }
    finally
    {
        await Task.Delay(60);
        semaphore.Release();
        try { if (File.Exists(tempIn)) File.Delete(tempIn); } catch {}
        try { if (tempWav != tempIn && File.Exists(tempWav)) File.Delete(tempWav); } catch {}
    }
});
```

3) Conversion WAV 16k mono (Windows/NAudio)

```
using NAudio.Wave;

static string ConvertToWav16kMono(string inputPath)
{
    // Si déjà WAV/PCM 16k mono, vous pouvez court-circuiter (omitted pour concision)
    var outPath = Path.Combine(Path.GetTempPath(), $"stt_{Guid.NewGuid():N}.wav");
    using var reader = new MediaFoundationReader(inputPath);
    var target = new WaveFormat(16000, 16, 1); // 16kHz, 16-bit, mono
```

```
    using var resampler = new MediaFoundationResampler(reader, target) { ResamplerQuality =
60 };
    WaveFileWriter.CreateWaveFile(outPath, resampler);
    return outPath;
}
```

Linux/Mac (option) : utiliser un wrapper ffmpeg (process) :

```
ffmpeg -i input -ar 16000 -ac 1 -acodec pcm_s16le output.wav
```