M1 INFO 2025-2026 / Projet Sampler Audio / Angular

FRENCH VERSION

Voici le travail attendu avec des parties obligatoires et des parties optionnelles.

- A FAIRE EN BINÔME. Indiquer qui a fait quoi dans le README
- DATE DE RENDU : XXX (mais d'ici 3 semaines environs, la date sera définitivement fixée ce lundi)
- MODALITÉS
    - Repo GitHub avec README détaillé, indiquez ce que vous avez fait avec de l'IA (indiquer les parties, les gros prompts, les outils utilisés).
    - Toute feature moins bien que vue en cours et généré par IA enlève des points.
    - Toute feature optionnelle non documentée enlève des points.
    - Si le projet est hébergé quelque part : bonus !

Je me réserve la possibilité de vous convoquer pour démos + questions et demandes d'explications.

FRONT-END (sampler)

1. Vous utiliserez un back-end NodeJS qui propose une API pour les presets, comme vu en cours. Bien entendu vous pouvez le personnaliser avec vos propres fonctionnalités customs.
2. Séparer GUI et moteur audio. La page HTML que vous proposerez pour tester votre sampler devra comporter un test en mode "headless" (sans GUI).
3. Le sampler devra proposer un menu de presets construit dynamiquement par envoi d'une requête sur le back-end via une requête fetch GET.
    a. Optionnel : proposer des catégories de presets (drums, pianos, etc. je vous avais donné un exemple en cours, il est sur le repository GitHub du cours)
4. Lorsqu'on choisit un preset, les sons se chargent et sont affectés aux différents pads. Le pad 0 qui correspond, si on a un kit de batterie, au son "kick" (grosse caisse) sera en bas à gauche, le son snare (caisse claire) juste à sa droite, etc.
5. Lors du chargement des presets on a des barres de progression animées (voir exemples dans le repository GitHub).
6. Une fois les sons chargés, si on clique sur un pad, ça joue le son et ça affiche sa forme d'ondes dans un canvas.
    a. Optionnel : mapper des touches du clavier de l'ordinateur sur des pads pour jouer.
    b. Optionnel : support MIDI, on veut pouvoir utiliser un contrôleur MIDI hardware branché en USB pour jouer les sons du sampler. Vous pouvez tester cela avec un "clavier midi virtuel", il en existe pour windows, sans doute pour Linux et pour

Mac (sur Mac j'utilise le logiciel gratuit MIDIkeys). Du code est fourni dans les exemples du cours pour cela.

7. Possibilité de "trimmer" les sons (choisir le temps de début et le temps de fin) comme vu en cours. Pour chaque son on doit pouvoir faire des réglages individuels.
8. OPTIONNEL (MAIS SOUHAITE) : possibilité d'enregistrer des sons **avec le micro**. Suivent des options plus ou moins compliquées:
    a. Possibilité d'affecter ce son à un pad et donc pour ce son visualiser sa forme d'onde dans le canvas et possibilité de le trimmer.
    b. Plus compliqué : analyser le son et détecter les silences ou les parties "de faible intensité / faible volume", découper le son et au lieu de l'affecter à un seul pad, il sera découpé en plusieurs parties et chaque partie affectée à un pad différent, en partant du pad 0 en bas à droite et en remplissant de bas en haut et de gauche à droite. Bien entendu, chaque son pourra être visualisé et trimmé. Voir cette vidéo pour comprendre les différentes manières d'enregistrer : https://www.youtube.com/watch?v=RnF1md-xxTc
9. OPTIONNEL: possibilité d'aller chercher des sons sur freesound.org , voir l'exemple de code proposé dans la séance 7. Vous devrez vous enregistrer comme développeur pour obtenir une clé d'API. Attention: le nombre de requêtes est limité par jour, en mode dev, ne lancez pas des requêtes tout le temps sinon ça va finir par bloquer.
    a. On veut pouvoir "previewer" les sons avant de les affecter à des pads.
    b. Comme pour tous les autres sons : visualisation et trims des sons.
10. OPTIONNEL (MAIS SOUHAITE): possibilité de sauvegarder un nouveau preset sur le serveur. Vous devrez vous inspirer de l'exemple Seance2/ExampleRESTEndpointCORRIGE pour voir comment gérer un envoi de fichier par HTTP POST. C'est une option assez complexe mais vraiment importante dans la fonctionnalité du sampler. Il faudra donner un nom et éventuellement une catégorie/type au preset. Les fichiers audio nouveau seront envoyés sur le serveur et un fichier .json sera créé sur le serveur.
11. OPTIONNEL : la GUI est sous la forme d'un Web Component (cfr cours sur les Web Components).
12. OPTIONNEL : le sujet est ouvert, vous avez la possibilité d'intégrer des features de votre choix
    a. Effets audio globaux ou par son (réglage du volume, pan gauche/droite, inverser le son, changer son pitch (via un changement de vitesse de lecture), autres…
    b. Possibilité d'enregistrer ce que l'on joue pour le rejouer, un peu en mode boîte à rythme (certains l'ont déjà fait dans la classe)
    c. Autres….

PARTIE BACK-END

- Là, tout est optionnel dans la mesure où je vous ai donné des back-ends fonctionnels pour les features de base (envoyer des presets). A vous de voir pour adapter le code ou le refaire pour qu'il corresponde aux features optionnelles que vous aurez éventuellement implémentées.
- Dans le cours Angular j'ai terminé en montrant un back-end connecté à une base de données MongoDB dans le cloud. Il manquait beaucoup d'élèves (vous aviez des examens le lendemain). Je ne vais pas rendre cette partie obligatoire.

    - OPTIONNEL : sauvegarder les presets (les objets JSON, pas les fichiers audio) dans une base de données cloud et adapter le back-end pour utiliser cette base de données.
    - OPTIONNEL : héberger le back-end sur render.com dans le cloud (voir fin du cours Angular) et vidéos que je vais vous préparer.

PARTIE ANGULAR

- A FAIRE OBLIGATOIREMENT: une appli front-end dans un projet séparé du sampler et du back-end, qui va afficher au minimum une page avec la liste des presets. Elle parlera au même back-end que celui utilisé par le sampler.
- Possibilité de lister les presets et de les renommer.
- OPTIONNEL : ajouter des fonctionnalités de gestion des presets :
    - suppression d'un preset,
    - création d'un nouveau preset depuis la page Angular (on donne son nom puis la liste des URLs des sons à ajouter, dans un premier temps: pas d'upload de fichiers audio)
    - Un peu mieux: possibilité d'uploader des fichiers de son lors de la création d'un nouveau preset
- Là aussi: le sujet est ouvert, possibilité de proposer des features à vous, de personnaliser le projet.

ENGLISH VERSION

M1 INFO 2025–2026 / Audio Sampler Project / Angular

Here is the expected work, with mandatory parts and optional parts.

TO BE DONE IN PAIRS. Indicate who did what in the README.
DEADLINE: XXX (but within about 3 weeks, the exact date will be fixed this Monday).

TERMS

GitHub repository with a detailed README. Indicate what you did with AI (specify the parts, the main prompts, the tools used).
Any feature that is worse than what was seen in class and generated by AI will lose points.
Any optional feature that is not documented will lose points.
If the project is hosted somewhere: bonus!

I reserve the right to call you in for demos, plus questions and requests for explanations.

FRONT-END (sampler)

You will use a NodeJS back-end that provides an API for presets, as seen in class. Of course, you may customize it with your own custom features.

Separate GUI and audio engine. The HTML page you provide to test your sampler must include a “headless” test mode (without GUI).

The sampler must offer a presets menu built dynamically by sending a request to the back-end via a fetch GET request.

Optional: provide preset categories (drums, pianos, etc. — I gave you an example in class, it is on the course GitHub repository).

When a preset is chosen, the sounds are loaded and assigned to the different pads. Pad 0, which corresponds (if we have a drum kit) to the “kick” sound (bass drum), will be at the bottom left, the snare sound just to its right, etc.

When loading presets, animated progress bars must be displayed (see examples in the GitHub repository).

Once the sounds are loaded, clicking on a pad plays the sound and displays its waveform in a canvas.

Optional: map computer keyboard keys to pads for playing.

Optional: MIDI support. We want to be able to use a hardware MIDI controller connected via USB to play the sampler sounds. You can test this with a “virtual MIDI keyboard”; such tools exist for Windows, probably for Linux and Mac as well (on Mac I use the free software MIDIkeys). Code is provided in the course examples for this.

Ability to “trim” sounds (choose start time and end time) as seen in class. For each sound, individual settings must be possible.

OPTIONAL (BUT RECOMMENDED): ability to record sounds with the microphone. Then come more or less complex options:

- Ability to assign this sound to a pad and therefore visualize its waveform in the canvas and trim it.
- More complex: analyze the sound and detect silences or “low intensity / low volume” parts, cut the sound, and instead of assigning it to a single pad, split it into several parts and assign each part to a different pad, starting from pad 0 at the bottom right and filling from bottom to top and left to right. Of course, each sound can be visualized and trimmed. See this video to understand the different ways of recording: https://www.youtube.com/watch?v=RnF1md-xxTc

OPTIONAL: ability to fetch sounds from freesound.org; see the example code from session 7. You will need to register as a developer to obtain an API key. Warning: the number of requests is limited per day in dev mode; do not send requests constantly or it will end up being blocked. We want to be able to “preview” sounds before assigning them to pads.
As with all other sounds: waveform visualization and trimming.

OPTIONAL (BUT RECOMMENDED): ability to save a new preset on the server. You should take inspiration from the example *Seance2/ExampleRESTEndpointCORRIGE* to see how to handle file uploads via HTTP POST. This is a fairly complex option but very important for sampler functionality. You will need to give a name and optionally a category/type to the preset. New audio files will be sent to the server and a .json file will be created on the server.

OPTIONAL: GUI in the form of a Web Component (see course on Web Components).

OPTIONAL: open topic — you may integrate features of your choice, for example:

- Global or per-sound audio effects (volume control, left/right pan, reverse sound, change pitch via playback speed change, others…).
- Ability to record what you play and replay it, somewhat like a drum machine (some already did this in class).
- Others…

BACK-END PART

Here, everything is optional, since I provided functional back-ends for the basic features (sending presets). It's up to you to adapt the code or redo it so that it matches the optional features you may implement.

In the Angular course, I ended by showing a back-end connected to a MongoDB cloud database. Many students were absent (you had exams the next day). I will not make this part mandatory.

OPTIONAL: save presets (JSON objects, not audio files) in a cloud database and adapt the back-end to use this database.
OPTIONAL: host the back-end on render.com in the cloud (see the end of the Angular course) and videos that I will prepare.

ANGULAR PART

MANDATORY: a front-end app in a separate project from the sampler and the back-end, which will display at minimum a page with the list of presets. It will communicate with the same back-end used by the sampler.

Ability to list presets and rename them.

OPTIONAL: add preset management features:

- delete a preset,

- create a new preset from the Angular page (give its name and then the list of sound URLs to add, at first: no audio file upload).

Better: ability to upload sound files when creating a new preset.

Here too: open topic — you may propose your own features and customize the project.