
Projet d'Informatique Individuel
Rendu final

Projet : SoniPic



Tuteur : Maxime Poret
Année 2023 - 2024

Table des matières

| | |
|--|-----------|
| 1. Introduction..... | 2 |
| 1.1. La sonification d'images..... | 2 |
| 1.2. Accessibilité pour les personnes non/malvoyantes..... | 2 |
| 1.3. But du projet..... | 3 |
| 2. Gestion de projet..... | 4 |
| 2.1. Planning prévisionnel initial..... | 4 |
| 2.2. Planning final..... | 5 |
| 2.3. Planning effectif..... | 6 |
| 3. Réalisation..... | 7 |
| 3.1. L'extension chrome..... | 7 |
| 3.2. Le serveur Flask..... | 9 |
| 3.3. Le mode abstrait..... | 12 |
| 3.4. Le mode concret..... | 13 |
| 3.5. Le site web vitrine..... | 16 |
| 3.6. L'hébergement..... | 17 |
| 3.7. Résumé des outils utilisés..... | 18 |
| 4. Bilan..... | 19 |
| 4.1. Perspectives..... | 19 |
| 4.2. Bilan personnel..... | 19 |
| Annexes..... | 20 |
| Annexe 1 : Exemples d'images sonifiées selon le mode abstrait..... | 20 |
| Annexe 2 : Exemples d'images sonifiées selon le mode concret..... | 25 |

1. Introduction

1.1. La sonification d'images

La sonification est **le processus de transformation et de représentation des données au moyen de sons audio**. Cette discipline est équivalente à la visualisation des données, mais dans ce cas, la représentation est matérialisée de manière sonore. Son objectif est de transmettre des informations objectives par le son, qui peut varier en tempo, en amplitude, en rythme, en volume, en fréquence ou en hauteur pour mettre en évidence les changements dans les données.

Ainsi, la sonification d'images consiste à **convertir des informations visuelles**, telles que des photos, des graphiques ou des diagrammes, **en sons**.

1.2. Accessibilité pour les personnes non/malvoyantes

Ces dernières années, l'accès à l'information pour les personnes atteintes de déficiences visuelles a été facilité par l'arrivée de la transcription audio d'un grand nombre d'éléments textuels sur Internet. Cependant, il n'existe toujours pas de bonnes solutions pour la perception et la compréhension des images. On peut se demander si la sonification d'image pourrait permettre de mieux apprécier et traiter les images sur internet. Plusieurs études ont été menées sur la question.

Il faut savoir que près d'un tiers du cerveau humain est dédié à la vision, ainsi l'absence de vision entraîne des **différences de développement dans les zones auditives et visuelles des personnes malvoyantes** par rapport aux personnes voyantes. Par exemple, le cortex visuel des personnes aveugles de naissance est activé au cours de tâches non visuelles impliquant par exemple le langage ou la mémoire.

Les aveugles précoces et les aveugles de naissance font preuve d'une **meilleure discrimination auditive, en particulier pour reconnaître les transitions rapides de hauteur et de timbre**. Cependant, les aveugles tardifs sont moins performants dans ces tâches. Les personnes atteintes de cécité partielle présentent des réponses neuronales plus faibles aux stimuli auditifs simples, mais une activité accrue dans les tâches complexes de discrimination auditive. On peut prendre l'exemple des lecteurs d'écrans : pour maximiser l'efficacité, de nombreuses personnes malvoyantes règlent la vitesse de parole de leur appareil à un niveau très élevé et n'ont aucun souci de compréhension.

Au vu de ces spécificités, il semble que la sonification d'images est un outil qui peut être adapté et par conséquent utile pour les personnes atteintes de déficiences visuelles sur Internet.

1.3. But du projet

C'est de ce constat du manque d'accessibilité des images sur Internet que l'idée de ce projet m'est venue. Le projet SoniPic, a pour principal objectif de développer **une extension chrome permettant de sonifier les images**, c'est-à-dire de transformer les images en sons. Une extension chrome est une extension de navigateur qui est développé pour le navigateur Google Chrome, et qui permet d'ajouter des fonctionnalités au navigateur web.

L'extension SoniPic comprend **2 modes de sonification** :

- **Abstrait** : Traduire les formes dans l'image en sons. *Ex : Traduire le fait qu'il y a une ligne et qu'elle monte.*
- **Concret** : Entendre les objets présents dans l'image. *Ex : Entendre les rugissements d'un lion s'il y a un lion.*

2. Gestion de projet

2.1. Planning prévisionnel initial

Vous pouvez trouver le planning prévisionnel en version simplifiée et en version non simplifiée dans le lien suivant : [plannings prévisionnels initiaux](#).

| Tâche | 08/01 | 15/01 | 22/01 | 29/01 | 05/02 | 12/02 | 19/02 | 26/02 | 04/03 | 11/03 | 18/03 | 25/03 | 01/04 |
|--|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| Lancement du projet | | | | | | | | | | | | | |
| Choix du sujet et Cahier des Charges | | | | | | | | | | | | | |
| 1 ^e sprint | | | | | | | | | | | | | |
| Prise en main des outils et technologies | | | | | | | | | | | | | |
| Maquettage de l'interface | | | | | | | | | | | | | |
| Détection des images, et récupération d'une image sélectionnée | | | | | | | | | | | | | |
| Déterminer les paramètres de sonification | | | | | | | | | | | | | |
| Développement d'un script de sonification | | | | | | | | | | | | | |
| Développement de l'interface de l'extension | | | | | | | | | | | | | |
| Rendu livrable intermédiaire | | | | | | | | | | | | | |
| 2 ^e sprint | | | | | | | | | | | | | |
| Ajout des retranscriptions audio sur l'interface | | | | | | | | | | | | | |
| Ajout des options enregistrer et partager | | | | | | | | | | | | | |
| Segmentation de l'image et extraction des caractéristiques | | | | | | | | | | | | | |
| L'utilisateur peut modifier les paramètres de sonification | | | | | | | | | | | | | |
| Rédaction des livrables | | | | | | | | | | | | | |
| Préparation de la soutenance finale | | | | | | | | | | | | | |

Figure 1 : Planning prévisionnel initial simplifié

Le projet a été découpé en **deux sprints** finissant par un point et un rendu d'avancement de projets chacun. Le premier sprint se focalise sur le développement de l'extension chrome et la mise en place du mode abstrait, tandis que le deuxième sprint concerne davantage le mode concret.

2.2. Planning final

Vous pouvez trouver le planning prévisionnel final en version simplifiée et en version non simplifiée dans le lien suivant : [plannings prévisionnels finaux](#).

| Tâche | 08/01 | 15/01 | 22/01 | 29/01 | 05/02 | 12/02 | 19/02 | 26/02 | 04/03 | 11/03 | 18/03 | 25/03 | 01/04 |
|--|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| Lancement du projet | | | | | | | | | | | | | |
| Choix du sujet et Cahier des Charges | ■ | | | | | | | | | | | | |
| 1e sprint | | | | | | | | | | | | | |
| Prise en main des outils et technologies | ■ | | | | | | | | | | | | |
| Maquettage de l'interface | ■ | ■ | | | | | | | | | | | |
| Détection des images, et récupération d'une image sélectionnée | | ■ | | | | | | | | | | | |
| Déterminer les paramètres de sonification | | ■ | | | | | | | | | | | |
| Développement d'un serveur flask | | | ■ | ■ | | | | | | | | | |
| Développement d'un script de sonification | | | ■ | ■ | | | | | | | | | |
| Développement de l'interface de l'extension | | | | ■ | ■ | | | | | | | | |
| Rendu livrable intermédiaire | | | | | | ■ | ■ | ■ | | | | | ■ |
| 2e sprint | | | | | | | | | | | | | |
| Ajout des options enregistrer et partager | | | | | | ■ | ■ | | | | | | |
| Segmentation de l'image, et création d'une banque de sons | | | | | | ■ | ■ | ■ | | | | | |
| Page vitrine de l'extension | | | | | | | | | | ■ | ■ | | |
| L'utilisateur peut modifier les paramètres de sonification | | | | | | | | | | ■ | ■ | | |
| Hébergement de l'extension, du serveur et du site web | | | | | | | | | ■ | ■ | | | |
| Rédaction des livrables | | | | | | | | | | ■ | ■ | | ■ |
| Préparation de la soutenance finale | | | | | | | | | | ■ | ■ | | ■ |

Figure 2 : Planning prévisionnel final simplifié

Par rapport au planning de départ, quelques modifications ont été apportées :

- **Ajout de “Développement d'un serveur flask”** : le traitement des images étant une opération lourde pour le navigateur, j'ai décidé de faire un serveur backend pour accélérer ce processus.
- **Suppression de “Ajout des retranscriptions audio sur l'interface”** : j'ai considéré que cette fonctionnalité n'était pas nécessaire, et que la retirer me ferait gagner du temps pour d'autres fonctionnalités.
- **Ajout de “Page vitrine de l'extension” (BONUS)** : page web permettant de télécharger l'extension chrome, et qui permet de sonifier des images qu'on importe depuis son pc.
- **Ajout de “Hébergement du projet”** : l'hébergement d'un service est un point clé d'un projet informatique et qui nécessite du temps pour être mis en place.

2.3. Planning effectif

Vous pouvez trouver le planning effectif en version simplifiée et en version non simplifiée dans le lien suivant : [plannings effectifs](#).

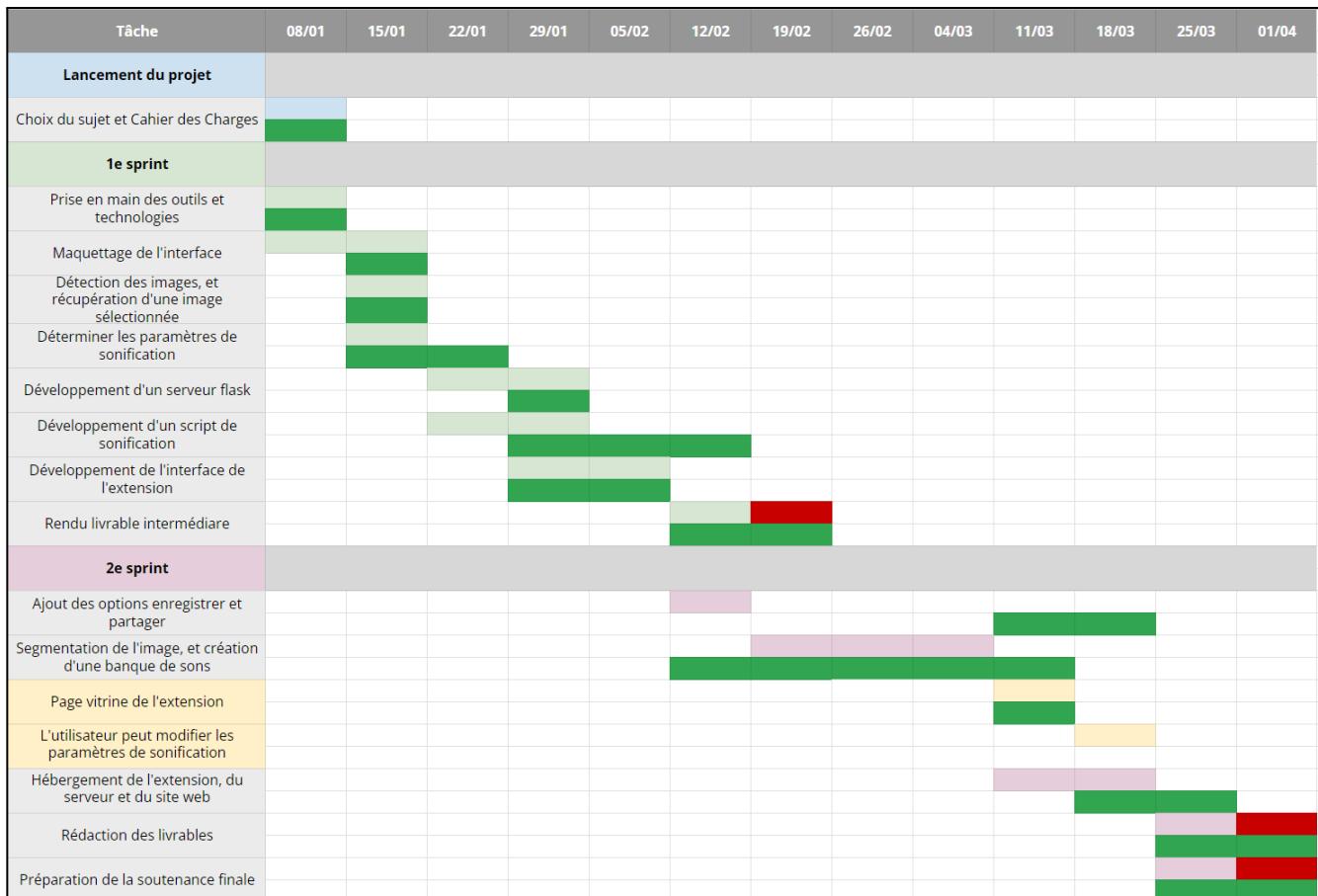


Figure 3 : Planning effectif simplifié - [En vert foncé : temps effectif]

Par rapport au planning prévisionnel, le temps alloué pour les tâches du premier sprint a été plutôt bien respecté, et j'ai pu finir ce premier sprint sans avoir de retard sur les fonctionnalités de celui-ci.

En ce qui concerne le deuxième sprint, j'ai commencé par travailler sur la tâche du développement du mode concret avant la tâche d'ajout des options "enregistrer" et "partager". De plus, cette tâche m'aura pris deux semaines de plus que prévu, entre autres à cause de problèmes de librairies. D'ailleurs, l'option de partage n'a finalement pas été effectuée, mais le fait de pouvoir enregistrer l'audio généré a été gardé, car je considère cette fonctionnalité comme importante. Enfin, la tâche bonus concernant le développement d'une page vitrine a été réalisée.

3. Réalisation

3.1. L'extension chrome

Spécifications techniques

Pour développer la partie extension chrome, j'ai utilisé de l'HTML, du CSS et du JavaScript sans framework. L'extension ne fonctionne que sur le navigateur Google Chrome et les navigateurs se basant sur Chromium (Brave, Edge...). En effet, il y a des différences dans le développement d'extensions entre Chrome et les autres navigateurs.

Une extension chrome se base sur trois contextes du navigateur. Un contexte référence une zone du navigateur que l'on peut modifier avec des scripts. Les scripts ne peuvent pas modifier directement les zones qui sont en dehors de leurs contextes. Les trois contextes d'une extension sont les suivants :

- **Popup** : l'interface popup de l'extension qui s'ouvre en cliquant sur l'icône
- **Page** : le site web qui est en train d'être consulté
- **Background** : fonctionne en arrière-plan du navigateur

Dans le cas de ce projet, j'ai utilisé les trois contextes d'une extension chrome. Tout d'abord, j'ai réalisé l'interface de l'extension dans le popup.

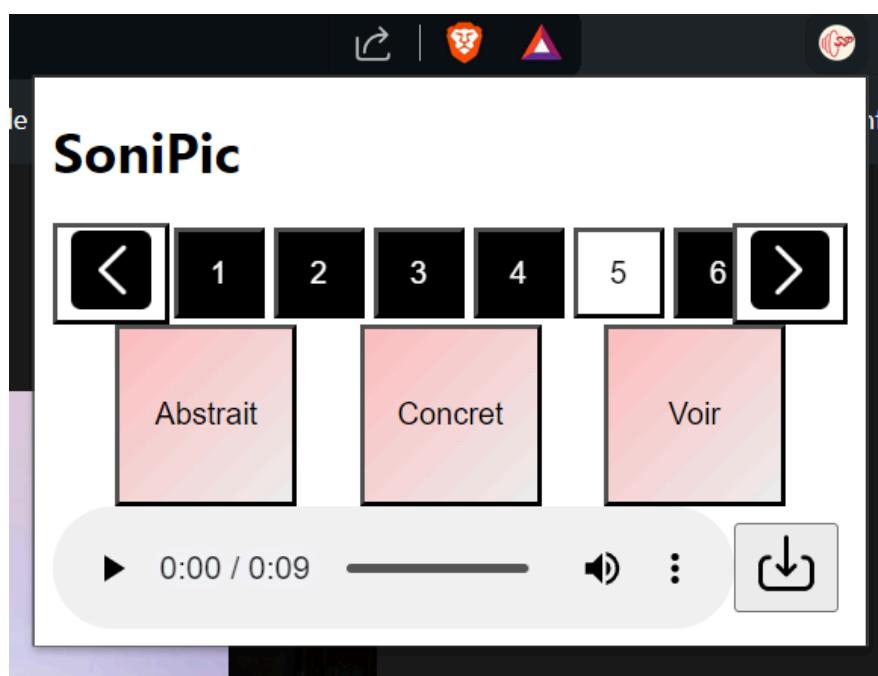


Figure 4 : Popup de l'extension chrome SoniPic

L'interface contient la possibilité de sélectionner une image, de pouvoir choisir un des deux modes de sonification, d'écouter l'audio généré, et de le télécharger sur son ordinateur.

Ensuite, j'utilise le contexte de la page pour afficher l'image sélectionnée après avoir cliqué sur le bouton "Voir" de l'interface. Pour faire cela, le popup envoie un message au script contenu dans le contexte de la page pour lui dire d'afficher l'image.

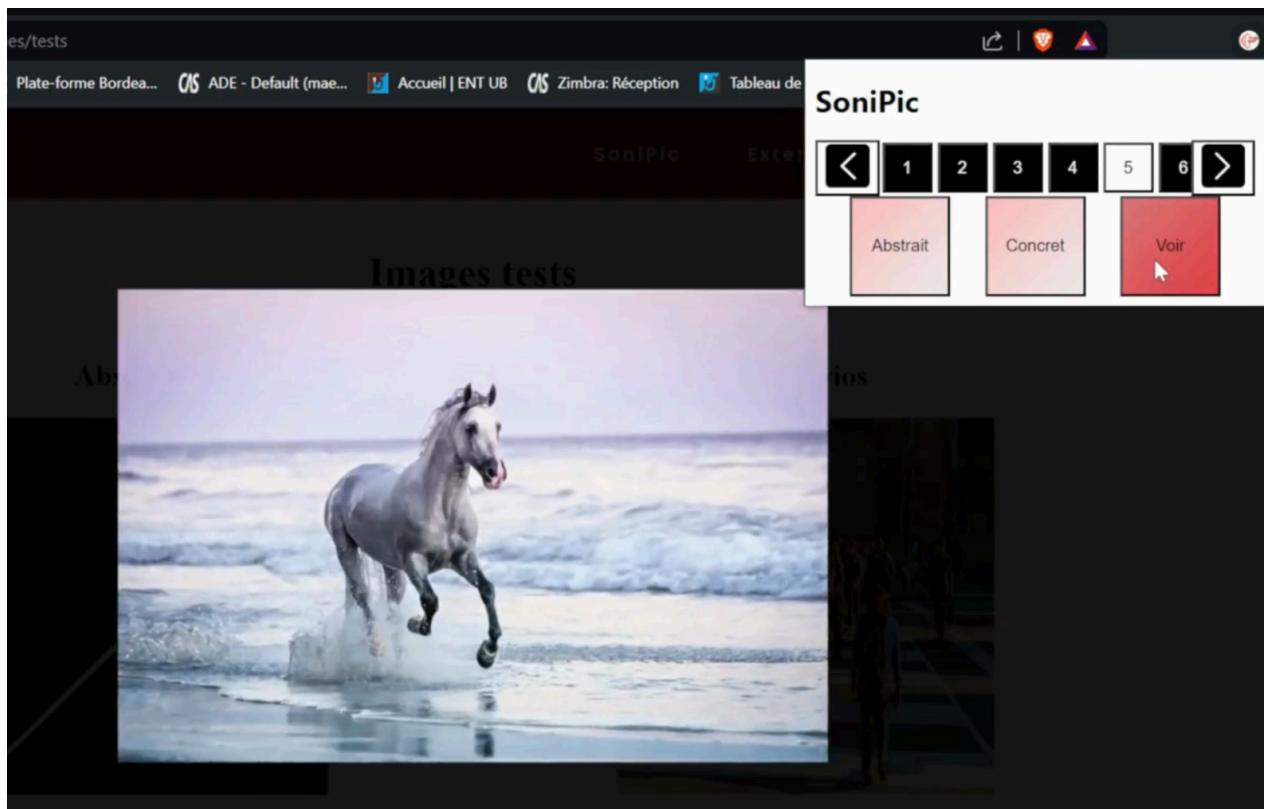


Figure 5 : Affichage d'une image sur la page

Enfin, j'ai réalisé un script en arrière-plan du navigateur pour récupérer tous les URLs des images présentes sur la page au chargement de celle-ci. Le script de background communique avec le popup pour lui faire part du nombre d'images contenues et les URLs de celles-ci.

Architecture du code

Le code de l'application est organisé de manière à pouvoir se repérer selon le contexte dans lequel interviennent les scripts. Ainsi, j'ai un dossier par contexte :

- background : contient le script qui fonctionne en arrière-plan du navigateur
- page : contient les scripts modifiant le site web
- popup : contient les fichiers de structure et les scripts du popup

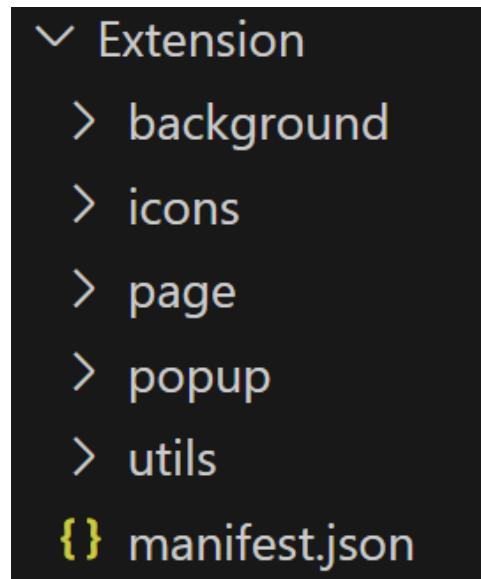


Figure 6 : Architecture des fichiers de l'extension

3.2. Le serveur Flask

Spécifications techniques

Au tout début du projet, j'avais pour idée de tout réaliser sur le navigateur, y compris le traitement de l'image. Cependant, je me suis vite rendu compte que cela n'allait pas être possible, du fait que, par exemple, il n'y avait pas forcément des librairies de segmentation d'images en JavaScript adapté à mon projet.

Par conséquent, j'ai décidé de créer un serveur backend en Python avec Flask pour faire la partie transformation des images en sons. Flask est un micro-framework Python pour le développement web, qui est léger et facile à configurer et à utiliser.

Sur ce serveur, j'ai développé **une API**, interface permettant la **communication entre le client et le serveur**, contenant trois requêtes :

- **POST-SOUND/** : Reçoit l'URL d'une image, récupère l'image et la transforme en fichiers audios
- **GET-SOUND/nom-fichier** : Retourne le fichier audio demandé
- **DELETE-SOUND/nom-fichier** : Supprime le fichier audio demandé

La communication entre l'extension chrome (le client) et le serveur fonctionne de cette manière :

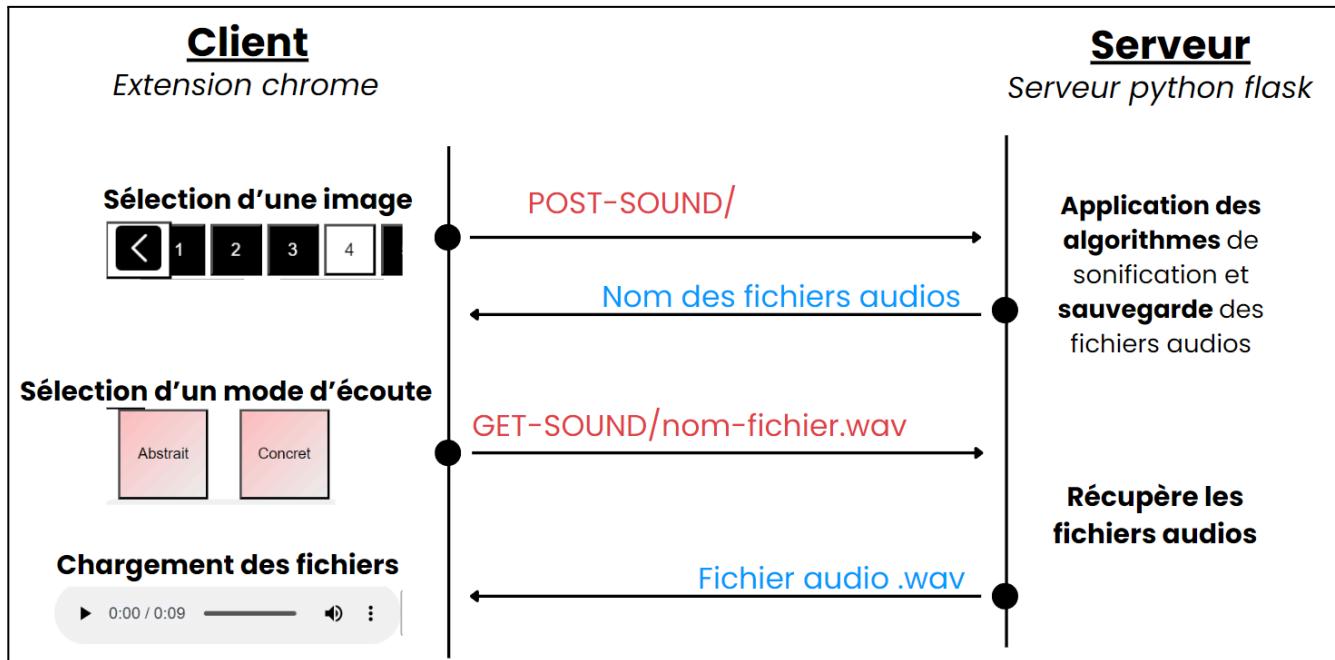


Figure 7 : Séquence des événements entre l'extension et le serveur

Lorsqu'on sélectionne une image, le client envoie une requête POST-SOUND au serveur, ce dernier applique les algorithmes de sonification, génère les fichiers audio, et renvoie les noms des fichiers au client. Il est intéressant de remarquer que le serveur ne renvoie pas directement les fichiers audio, mais juste les noms des fichiers. Au moment de la sélection d'une image, l'utilisateur n'a pas encore pris la décision d'écouter l'un des deux modes de sonification. Ainsi, cela permet de gagner du temps côté client, en laissant le serveur générer les fichiers audio avant même que l'utilisateur les demande.

Les fichiers audios sont générés et sauvegardés du côté serveur. Les noms des fichiers audios correspondent à l'URL de l'image pour permettre d'identifier les fichiers. Ainsi, si une image a déjà été sonifiée, le serveur n'a pas besoin d'appliquer de nouveau les algorithmes de sonification sur l'image. Cela permet de gagner du temps du côté serveur.

La requête pour supprimer des fichiers a été implémentée, mais n'est pas encore utilisée dans le processus du fonctionnement du projet. L'idée derrière cette fonctionnalité est de pouvoir éviter le surplus de fichiers audios sauvegardés du côté du serveur. Le but serait de ne garder les fichiers qu'un certain temps, puis de les supprimer.

Architecture du code

Le code côté serveur est organisé comme ceci :

- detectron2 : package de la librairie de segmentation d'image
- generatedSounds : contient les audios générés
- sounds : banque de sons
- imgToConcrete.py : script de sonification du mode concret
- imgToAbstract.py : script de sonification du mode abstrait
- utils.py : contient des fonctions communes

```
▽ Server
  > __pycache__
  > detectron2
  > generatedSounds
  > node_modules
  > sounds
  ⚡ env.py
  ⚡ imgToAbstract.py
  ⚡ imgToConcrete.py
  ⚡ main.py
  {} package-lock.json
  {} package.json
  ⚡ utils.py
```

Figure 8 : Architecture des fichiers du serveur

3.3. Le mode abstrait

Spécifications techniques

Pour réaliser l'algorithme de sonification du **mode abstrait**, j'ai implémenté un algorithme de traitement de l'image et du signal. Le but est de pouvoir faire une lecture de l'image de gauche à droite, donc de lire les colonnes de l'image les unes à la suite des autres. Toutes les images sont transformées en nuances de gris et redimensionnées en 400x400 pixels maximum avant d'être transformées en son.

L'algorithme que j'ai implémenté associe à chaque ligne de l'image une **fréquence entre 70 Hz et 1200 Hz**. Les lignes en haut de l'image sont associées aux fréquences les plus hautes et les lignes en bas aux fréquences les plus basses. De plus, l'algorithme associe à chaque valeur de pixel une intensité sonore entre 0 et 1. **Les pixels noirs auront 0 d'amplitude et les pixels blancs 1 d'amplitude.**

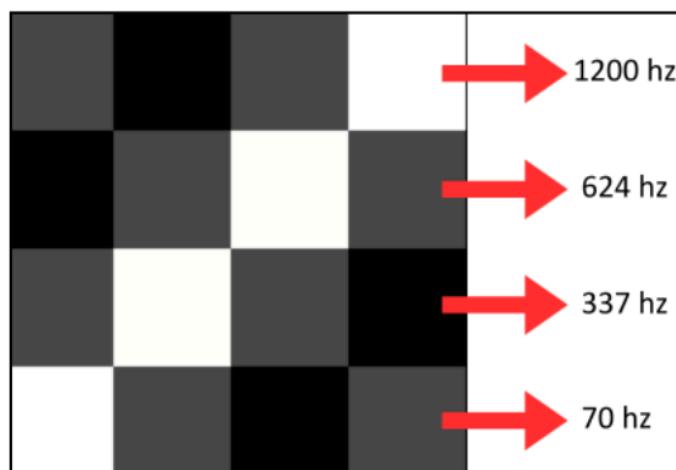


Figure 9 : Répartition des fréquences entre les lignes de l'image



Figure 10 : Association des amplitudes selon l'intensité d'un pixel

Ainsi, l'algorithme pour chaque ligne engendre un signal. Puis, il somme tous les signaux précédemment obtenus pour n'en avoir plus qu'un seul. Chaque colonne d'une image correspond à 0.025 seconde de l'audio, donc l'audio généré de l'image a une durée de 10 secondes.

Vous pouvez retrouver en annexe ([cf. annexe 1](#)) des exemples de spectrogrammes obtenus pour différentes images, et des liens contenant les fichiers audio des sons obtenus.

3.4. Le mode concret

Spécifications techniques

Pour réaliser l'algorithme de sonification du mode concret, j'ai dû utiliser un algorithme de segmentation d'image panoptique. La segmentation d'image vise à diviser tous les pixels d'une image en segments significatifs en fonction de certaines caractéristiques visuelles sur l'image, il existe trois types de segmentation. **La segmentation sémantique** classifie les pixels de l'image comme appartenant ou non à une certaine catégorie. **La segmentation d'instances**, en revanche, permet d'avoir plus d'informations sur l'image, en délimitant la forme exacte de chaque instance d'objet distincte. **La segmentation panoptique** est une segmentation globale qui détermine à la fois la classification sémantique de tous les pixels et différencient chaque instance d'objet. Dans le but de représenter l'ensemble des composantes d'une image, la segmentation panoptique me paraissait la plus appropriée pour ce projet.

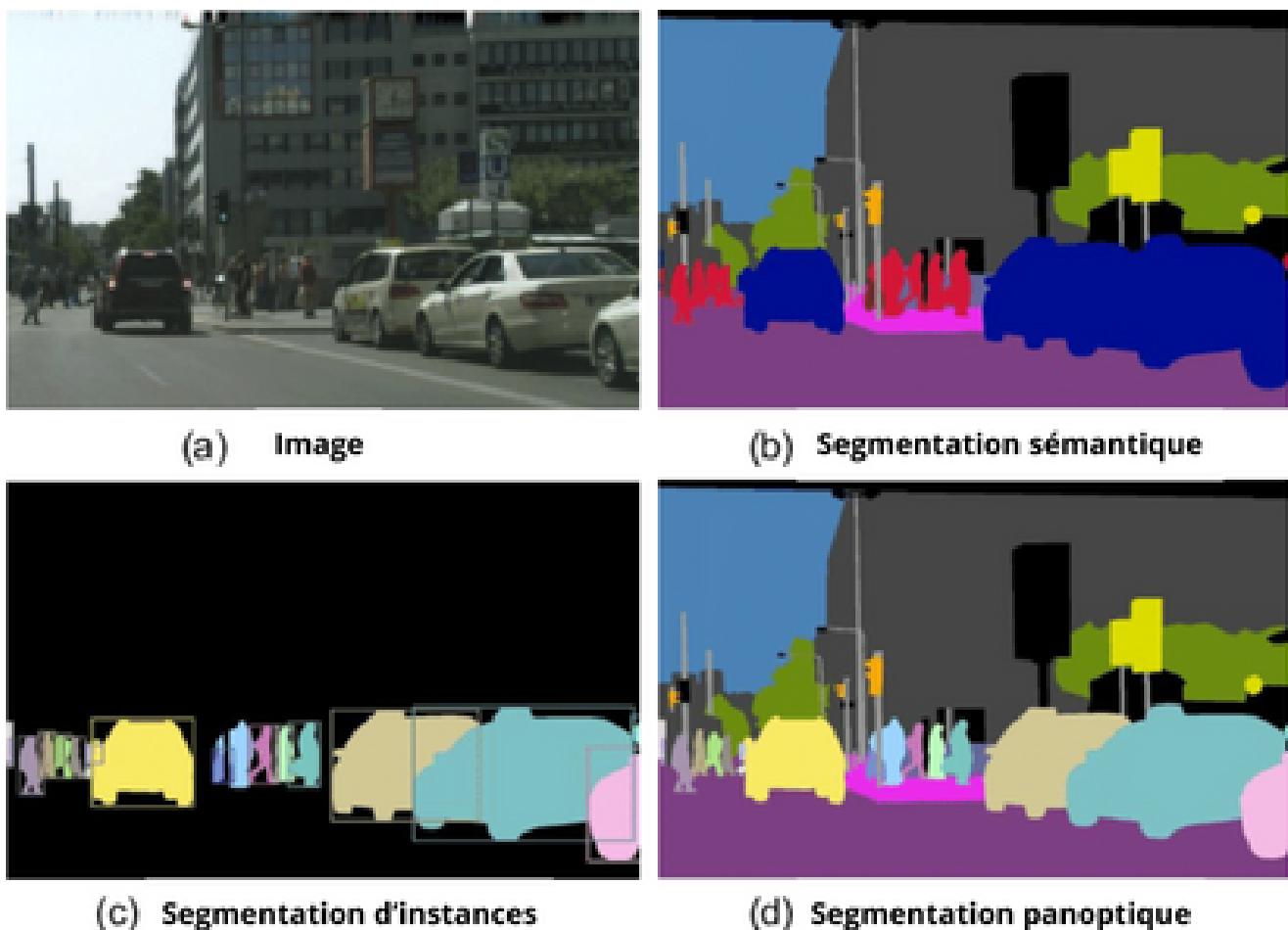


Figure 11 : Les trois types de segmentation d'images

Pour réaliser cela, j'ai utilisé la librairie Facebook de segmentation d'images Detectron2, et le modèle pré-entraîné MS COCO qui permet de faire de la segmentation panoptique. Ce modèle classifie les

objets en deux catégories. La catégorie “Stuff”, contenant 91 objets de décors et d’arrière-plan, et la catégorie “Things” contenant 80 objets d’instances.

Pour générer un son “concret” des images, j’ai **mis en place une banque de sons**, qui contient un grand nombre de fichiers audios tels que des bruits de voitures, d’animaux ou d’humains.

La première étape de mon algorithme consiste à récupérer les labels prédits par le modèle de segmentation.



- “Labels :
- Person
 - Person
 - Car
 - Bicycle
 - Building
 - Person
 - Pavement
 - ... ”

Figure 12 : Prédiction segmentation d’images avec MS COCO

La deuxième étape sélectionne les labels les plus fiables, c'est-à-dire ceux avec un score de prédiction > 0.9.

Enfin, la troisième étape consiste à mixer les fichiers audios de ma banque de sons selon les labels obtenus. Ainsi, si le mot “Person” et “Car” se trouve dans la liste des labels sélectionnés, alors mon algorithme combine les fichiers audios correspondants aux deux labels pour obtenir un son d’ambiance représentatif de l’image.

Ensuite, pour conserver une certaine cohérence de l’audio généré par rapport à l’image, on met en théorie autant de fichiers sons qu’il y a de labels correspondants sur l’image. Exemple, s’il y a 10 labels camions, il y aura en théorie 10 fichiers sons “camions” dans le fichier audio final. Mais cela peut entraîner une trop grande répétition. Pour éviter cela, j’ai décidé qu’un même label peut être mis 4 fois au maximum dans le fichier audio final. Si jamais le label avec l’occurrence maximale dans la liste des labels apparaît plus de 4 fois, on diminue proportionnellement le nombre d’occurrences des labels afin que l’occurrence maximale soit 4. Les nouvelles occurrences sont arrondies supérieurement.

Étape 1

- Trouver le label avec le plus grand nombre d'occurrences et garder ce nombre

Max : 10

| Labels | Occurrences |
|----------|-------------|
| Person | 10 |
| Car | 7 |
| Building | 2 |
| Sea | 1 |
| Tree | 3 |

Étape 2

- Repérer tous les labels avec un nombre d'occurrences > 4
- Mettre le nombre d'occurrences de ces labels à 4

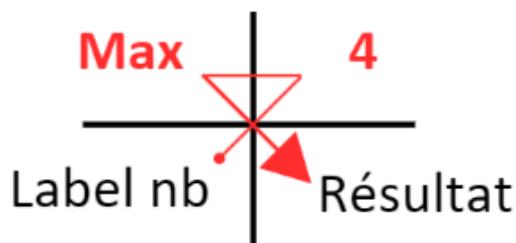
| Labels | Occurrences |
|----------|-------------|
| Person | 4 |
| Car | 4 |
| Building | 2 |
| Sea | 1 |
| Tree | 3 |

| Labels | Occurrences |
|----------|-------------|
| Person | 4 |
| Car | 2 |
| Building | 1 |
| Sea | 1 |
| Tree | 2 |

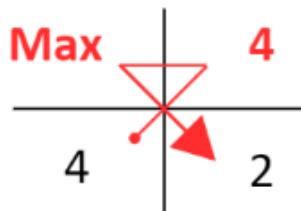
Étape 3

- Tous les nombres d'occurrences vont être modifiée selon le nombre d'occurrences maximal de l'étape 1
- Le label qui correspond au nombre maximal d'occurrences reste à 4

On effectue un produit en croix :



Exemple avec le label "Car"



Le résultat est arrondi au supérieur !

Figure 13 : Processus du traitement du nombre d'occurrences des labels

Enfin, le problème du mode concret est qu'il ne permet pas de sonifier toutes les images. Si certaines images sont abstraites ou si les composantes de l'image n'ont pas de bruits associés tels que des objets inanimés, alors l'extension indique qu'il n'est pas possible d'entendre l'image en mode concret.

Vous pouvez retrouver en annexe ([cf. annexe 2](#)) des exemples de fichiers audios obtenus après la sonification d'image en mode concret.

3.5. Le site web vitrine

Spécifications techniques

Le site web dit vitrine a été réalisé en HTML, CSS et JavaScript pure. L'objectif du site web est de pouvoir sonifier des images importées de l'ordinateur. En plus, un onglet dirige vers la page de téléchargement de l'extension chrome, et une des pages contient des images pour tester l'extension.

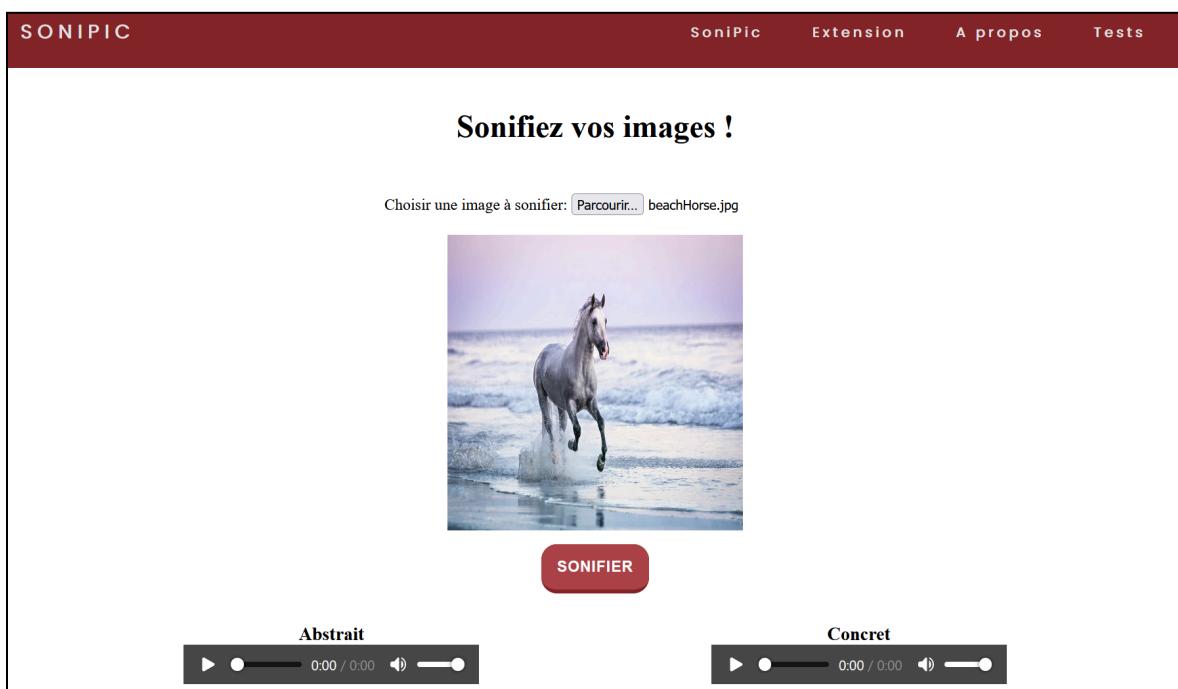


Figure 14 : Page principale du site web SoniPic

Architecture du code

Le code du site web est organisé de la manière suivante :

- index.html : page racine du site
- pages : contient toutes les pages HTML
- script : contient tous les scripts JavaScripts
- style : contient tous les fichiers de style CSS
- images : contient les images de tests

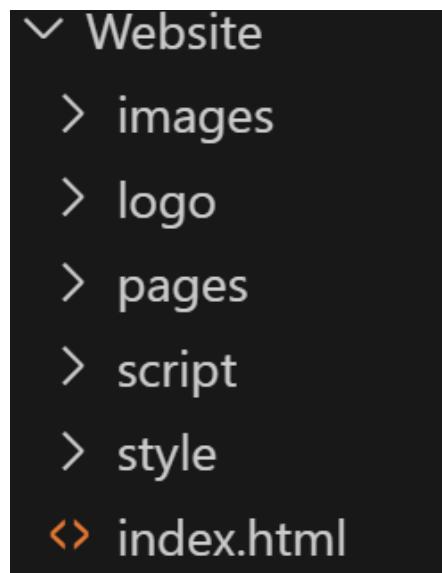


Figure 15 : Architecture des fichiers du serveur

3.6. L'hébergement

Spécifications techniques

Pour finir, la dernière tâche a été d'héberger le projet, ainsi, il a fallu chercher comment faire pour chacune des composantes du projet.

Le site Chrome WebStore Dev m'a permis d'héberger l'extension chrome, sachant que c'est la seule manière d'héberger une extension sous chrome . Cependant, l'extension chrome est encore en cours de validation et ne peut donc pas être téléchargée.

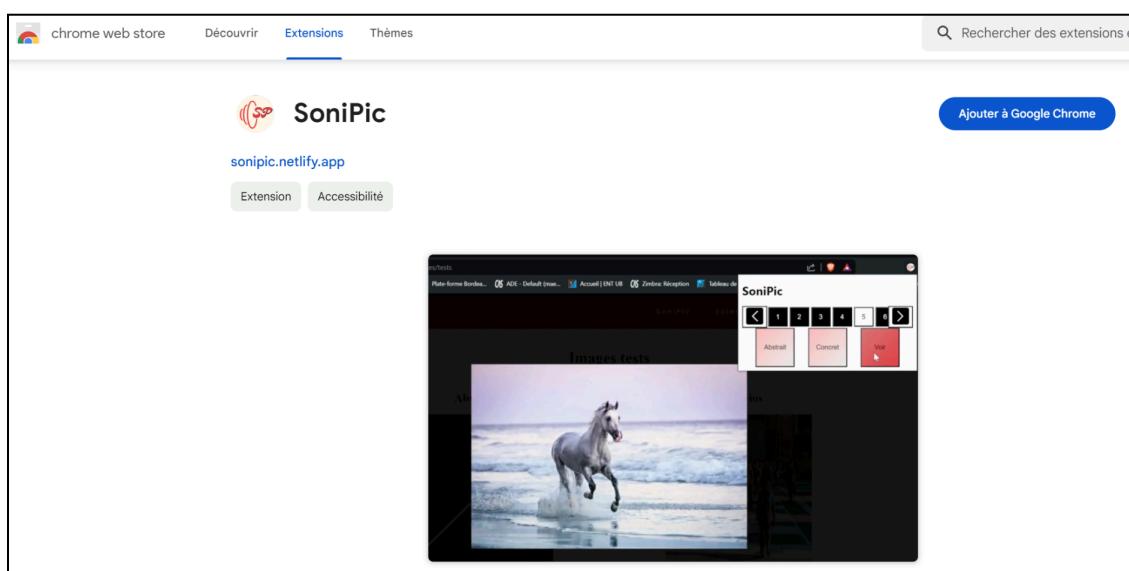


Figure 16 : Fiche Play Store de l'extension

Pour le serveur Python, j'ai utilisé PythonAnywhere pour héberger mon code Python. PythonAnywhere donne accès gratuitement à un serveur basique sur lequel il est possible d'héberger une application Flask.



Figure 17 : Hébergeur du serveur

Enfin, pour le site web vitrine, j'ai utilisé Netlify pour l'hébergement. Netlify est une solution facile à utiliser pour à la fois créer, déployer et héberger un site statique tout en pouvant avoir un nom de domaine semi-personnalisé, c'est-à-dire dans mon cas qui contient le mot "SoniPic". Voici le lien du site web : <https://sonipic.netlify.app/>



Figure 18 : Hébergeur du site web

3.7. Résumé des outils utilisés

En résumé, pour le développement, j'ai utilisé des langages de programmation tels que **JavaScript** et **Python**. J'ai utilisé **VSCode** pour coder et **Github** pour versionner le code. Le micro-framework Flask m'a permis de créer un serveur Backend. La librairie Facebook **Detectron 2** m'a servi pour faire de la segmentation d'image. Le logiciel **Postman** m'a été utile pour tester mes requêtes API. Enfin, j'ai hébergé le projet avec **Chrome WebStore Dev**, **PythonAnywhere** et **Netlify**.

4. Bilan

4.1. Perspectives

Actuellement, l'algorithme de sonification utilisé pour le mode abstrait est basique et ne prend pas en compte toutes les caractéristiques d'une image. Par exemple, l'algorithme pourrait tenir compte des couleurs ou des intentions de mouvements représentés dans l'image.

Ensuite, pour le mode concret, il serait intéressant de ne plus utiliser une banque de sons, mais de faire appel à une intelligence artificielle pour générer les sons. Pour pouvoir couvrir le plus de bruits possible et ainsi pouvoir s'adapter à tous types d'images, il faudrait passer énormément de temps à rassembler des fichiers audios dans le cas d'une banque de sons. Utiliser une IA déjà existante offrirait un meilleur rapport effort/résultat.

En conséquence du point précédent, entre autres, les deux modes de sonification des images actuels ne sont pas suffisants pour traiter tous les types d'images. Il serait bien d'ajouter un mode qui permet d'entendre la description audio d'une image sans dépendre de l'alternative textuelle issue du site web.

Enfin, l'interface de l'extension chrome n'a pas été développée dans le but d'être splendide, mais d'être fonctionnelle. De ce fait, l'interface n'est pas très claire et mériterait d'être améliorée. Par exemple, pouvoir ouvrir l'extension en cliquant sur une image du site serait une fonctionnalité utile pour savoir sur quelle image on applique la sonification.

4.2. Bilan personnel

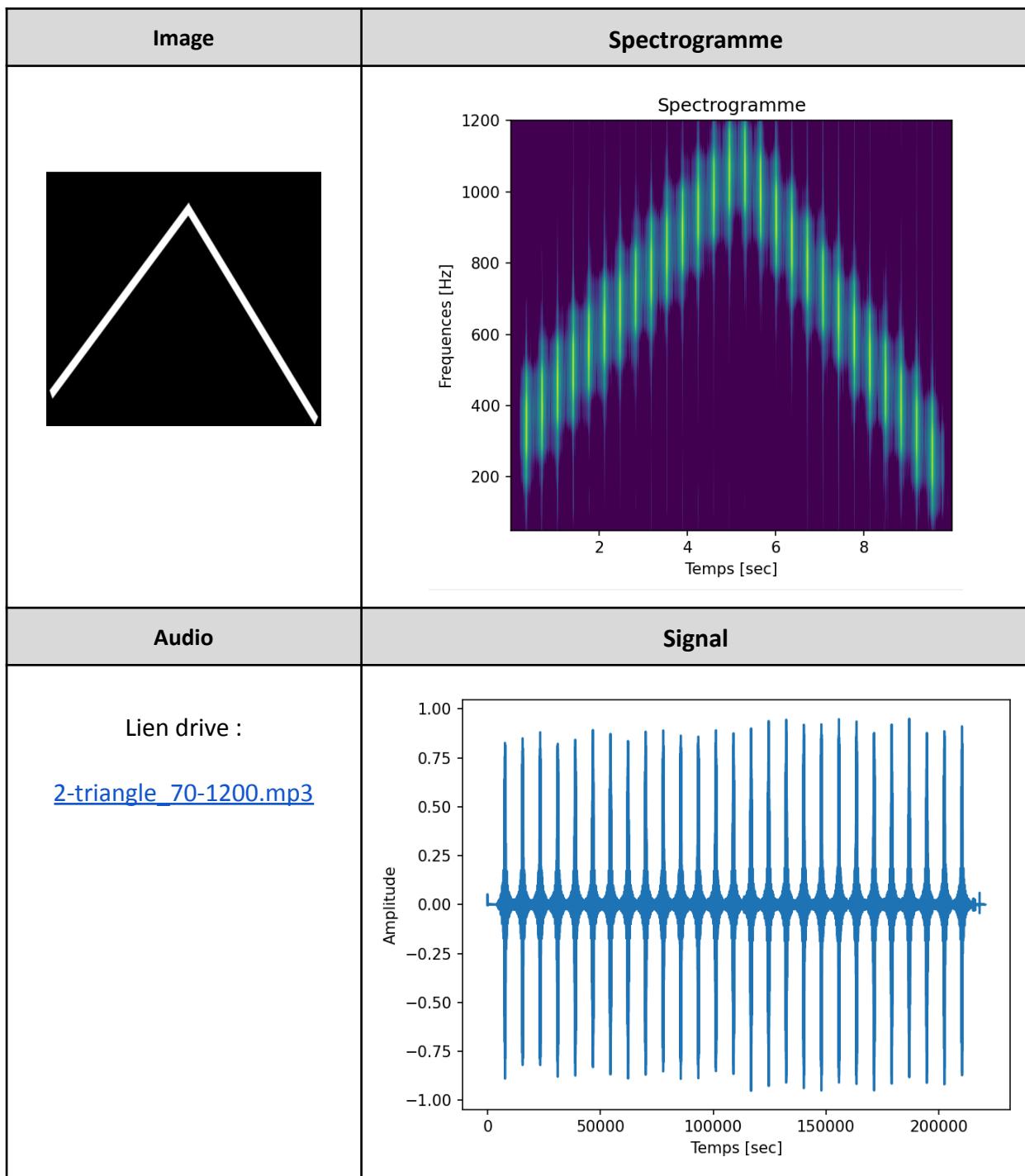
Ce projet m'a permis de **gagner en compétences dans plusieurs domaines** tout en utilisant des langages avec lesquels j'avais déjà expérimenté (JavaScript et Python). Ainsi, développer une extension chrome m'a permis de **découvrir tout un aspect du développement web** que je n'avais jamais traité auparavant. De plus, j'ai pu en **apprendre plus sur la segmentation d'image** et j'ai pu **mettre en pratique mes connaissances en traitement de l'image et du signal**.

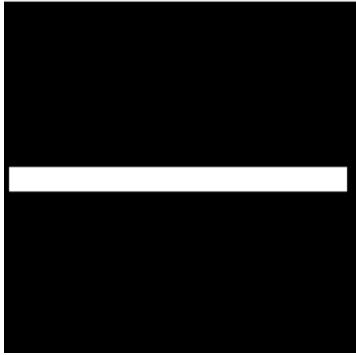
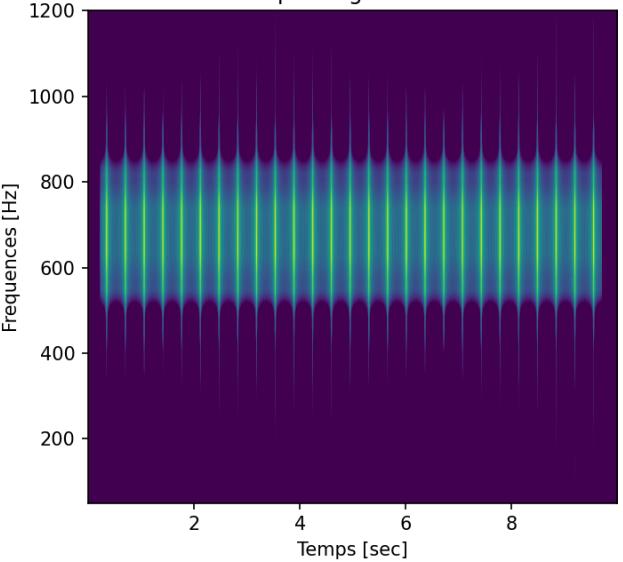
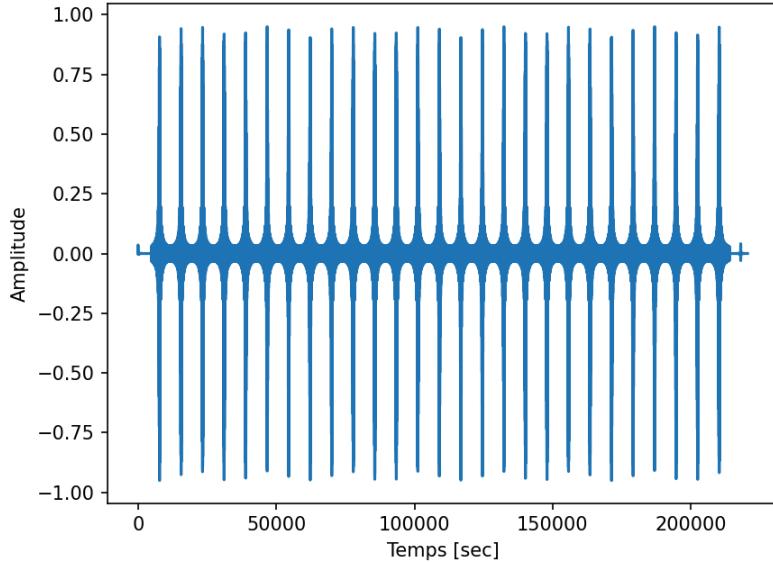
Je suis heureuse d'avoir pu réaliser ce projet, même si j'ai le sentiment de ne pas avoir pu pousser plus loin mes fonctionnalités, comme je l'imaginais au début. Le fait de ne pas connaître les librairies m'a fait perdre du temps pendant la phase de développement, mais ce sont des connaissances gagnées pour plus tard. Pour finir, j'ai pour ambition de continuer le projet pour l'améliorer.

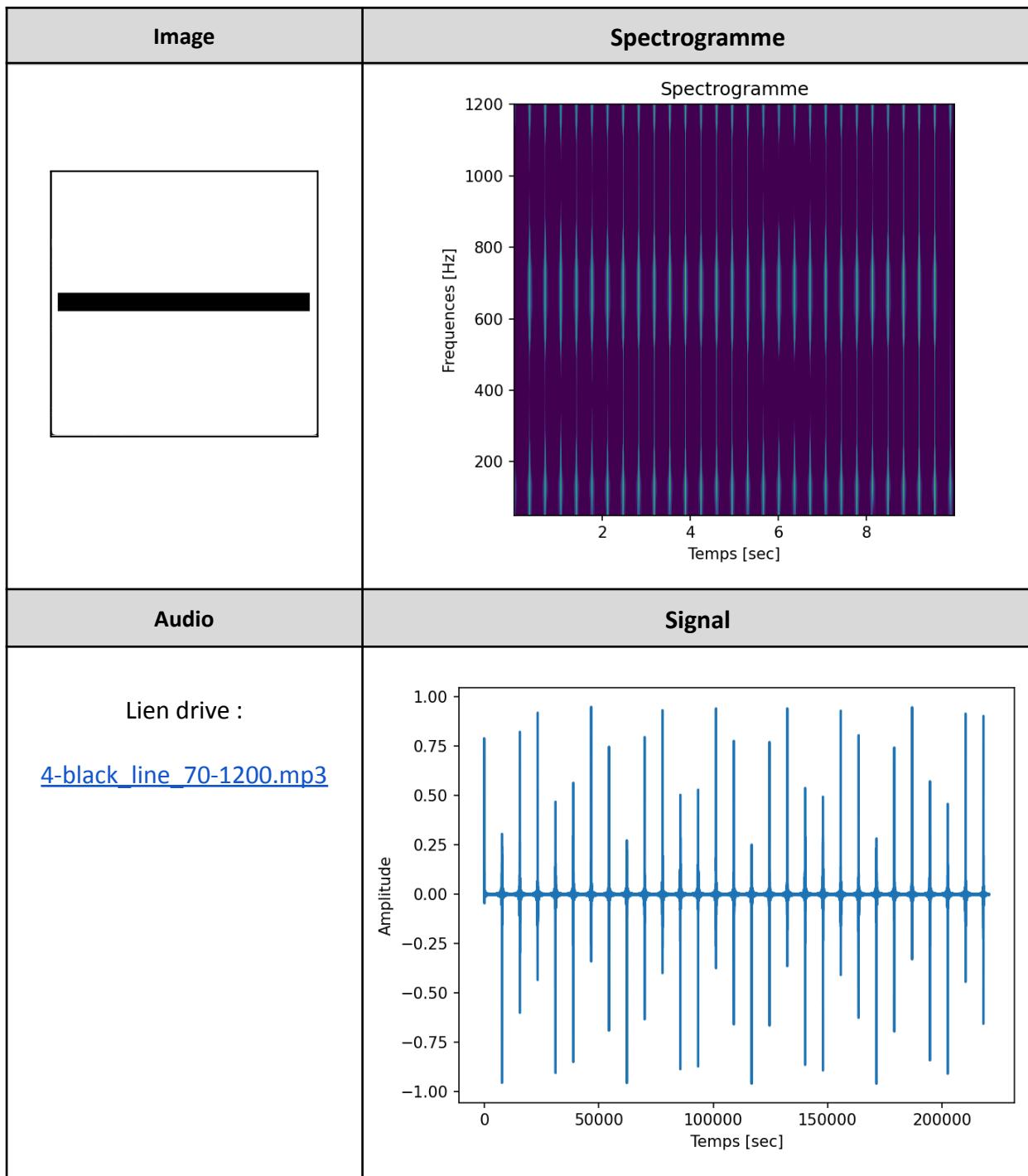
Annexes

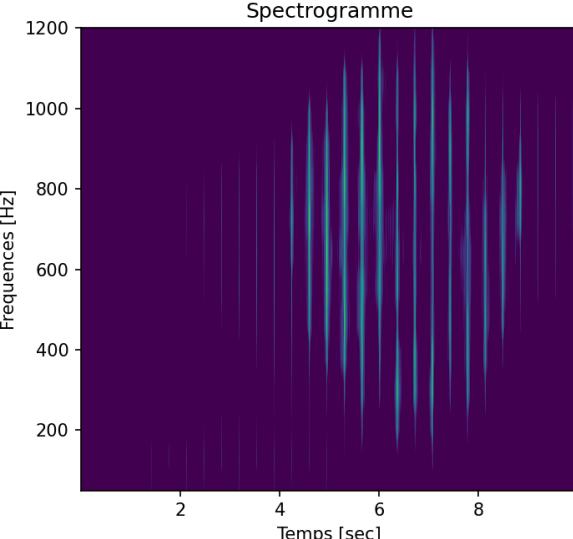
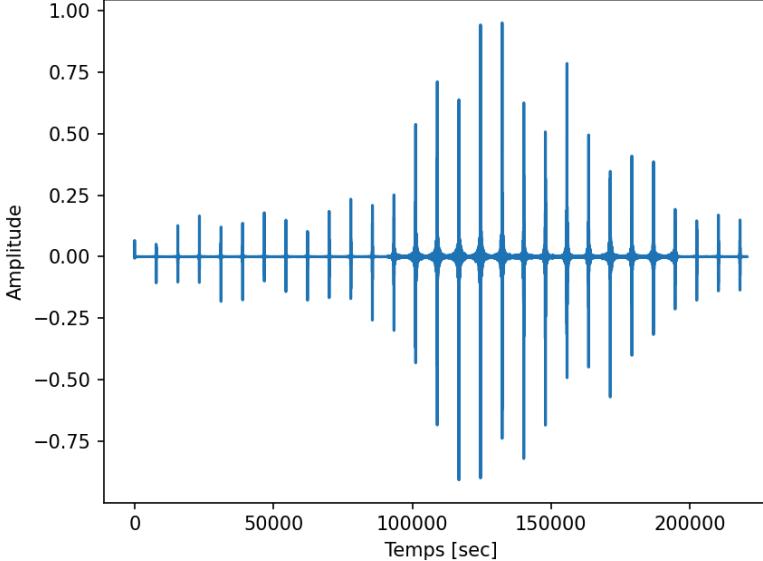
Annexe 1 : Exemples d'images sonifiées selon le mode abstrait

| Image | Spectrogramme |
|--|----------------------|
| | <p>Spectrogramme</p> |
| Audio | Signal |
| <p>Lien drive :</p> <p><u>1-lineUp_70-1200.mp3</u></p> | <p>Signal</p> |



| Image | Spectrogramme |
|---|--|
|  | <p>Spectrogramme</p>  |
| Audio | Signal |
| Lien drive : <u>3-white_line_70-1200.mp3</u> |  |



| Image | Spectrogramme |
|---|---|
|  | <p>Spectrogramme</p>  |
| Audio | Signal |
| Lien drive : 5-marguerite_70-1200.mp3 |  |

Annexe 2 : Exemples d'images sonifiées selon le mode concret

| Image | Audio |
|---|--|
|  | Lien drive : <u>6-beach_horse.wav</u> |

| Image | Audio |
|--|--|
|  | Lien drive : <u>7-amsterdam.wav</u> |

| Image | Audio |
|---|---|
|  | Lien drive : <u>8-car_road.wav</u> |