

UNIVERSITÉ CÔTE D'AZUR

MIAGE - MASTER 1



TER - Générateur de Questions

TIMMY DAUMAS, JEAN-BAPTISTE LOGNON

TER - Monsieur Renevier, Monsieur Buffa, Madame Cabrio, Monsieur Malapert.

Sommaire

Sommaire	2
Introduction	3
Présentation du sujet	3
Présentation du groupe	3
Etat de l'art	4
Technologies	4
NodeJS & Express	4
JQuery	5
MongoDB	5
Mongoose	5
HTML, CSS, SASS & Twig	5
Travail effectué	6
Presentation du resultat	6
Présentation du Code	9
Site	9
API	13
Base de données	24
Gestion de Projet	26
Conclusion	28
Perspectives et réflexions personnelles	29
Timmy Daumas	29
Jean-Baptiste Lognon	30
Annexe	31
Guide utilisateur	31
Accueil	31
Artistes	31
Génération à partir d'un artiste précis	32
Les questions à date, album et chanson	32
Le "Tout en un !"	34
Génération de X questions à partir d'une série artiste aléatoire	36

Introduction

Présentation du sujet

Notre sujet concerne le développement et déploiement d'une API de génération de questions pour alimenter un jeu de Quizz. Lors de la matière Suivi de Projet, notre équipe (de 5) avait décidé de développer en méthode Scrum un jeu de Quizz.

Finalement, Monsieur Renevier a proposé d'en faire un projet de TER en séparant le projet en 2 parties. 3 personnes développant le jeu en lui-même et 2 personnes s'occupant de générer des questions pour alimenter ce jeu.

Afin de générer ces questions, nous devons utiliser l'API Wasabi <https://wasabi.i3s.unice.fr/>.

Cette API est mise en ligne sur un vps et accessible par le site <https://ter.timmydaumas.fr>

Présentation du groupe

Timmy Daumas

A réalisé tout ce qui touche à la partie "Base de données" de la conception à l'implémentation. Il a également effectué le déploiement sur un vps, apporté une aide à la réalisation globale des fonctions de génération de questions et s'est penché à l'étude des données fournies par Wasabi.

Jean-Baptiste Lognon

A réalisé tout ce qui touche à la partie frontend de l'API des maquettes à l'implémentation du code qui forme le site final. De plus, il a aidé à la réalisation des fonctions de génération de questions et à l'étude des données fournies par Wasabi.

Etat de l'art

Technologies

NodeJS & Express

Pour la réalisation du projet, il a été décidé de partir sur une application en NodeJS Express (framework) qui va nous permettre de réaliser divers comportement comme des requêtes vers la base de données afin de récupérer toutes les informations à traiter et à afficher. De plus, cela nous permet de créer et d'utiliser des "routes" dans notre application.

Il est important de noter qu'en express, il existe plusieurs grandes façon de récupérer des informations envoyées par le client, nous en verrons ici trois afin de faciliter la compréhension des routes lors de la partie d'explication du code:

- Les query parameters, c'est le plus classique, il s'agit des paramètres passé après la route sous la forme `?parameterName=parameterValue&foo=bar`. Afin de récupérer ceci, Express nous donne accès à `req.query`, un objet contenant tous les query parameters de la requête. ici `req.query.parameterName` vaudra "parameterValue" et `req.query.foo` vaudra bar.
- Les route parameters, correspondant à des données insérées directement dans la route. Par exemple `foo.com/article/85667/price` qui aurait été écrit dans le code express comme `/article/:articleId/price` avec `articleId` le route parameter. Express nous propose alors `req.params`, un objet contenant toutes les routes parameters. Ici, `req.params.articleId` vaudra 85667.
- Le body de la requête, qui peut être assez intéressant car il peut être sous la forme d'un json contenant les données nécessaires au traitement de la requête. Mais d'après les spécifications du protocole HTTP, il ne faut pas utiliser le body pour influencer le résultat d'une requête GET, ce qui fait que cette méthode n'est pas très utilisée dans cette application.

JQuery

La première librairie est celle de JQuery qui va nous être très utile par la suite: "jQuery est une bibliothèque JavaScript rapide, petite et riche en fonctionnalités. Elle simplifie considérablement la traversée et la manipulation des documents HTML, la gestion des événements, l'animation et Ajax grâce à une API facile à utiliser qui fonctionne sur une multitude de navigateurs." (<https://jquery.com/>).

MongoDB

Nous avons décidé d'utiliser MongoDB pour notre projet. MongoDB est une base de données NOSQL qui utilise du bson pour stocker les objets. Ainsi, il n'y a presque pas de conversion à faire pour passer de la base de données MongoDB à un objet Javascript ce qui permet d'aller plus vite. C'est pour ces raisons que nous avons choisi d'utiliser une base de données MongoDB.

Mongoose

Afin d'utiliser MongoDB, nous avons décidé d'utiliser la librairie Mongoose qui est une surcouche de la librairie MongoDB. Elle permet de définir des Schémas et ainsi faire des vérifications sur les objets que l'on essaye d'enregistrer dans la base. De plus, il permet de plus facilement faire des requêtes sur la base de données et fournit beaucoup de fonctions permettant de gérer MongoDB.

HTML, CSS, SASS & Twig

En ce qui concerne notre frontend nous sommes partie sur du HTML et CSS via l'utilisation du framework Twig et de SASS.

Twig est un framework permettant de gérer les vues de notre application. Grâce à Twig, il est plus facile de réaliser de multiples pages ou de ne pas réécrire certaines parties de notre application. Pour l'instant, ce framework n'a pas été très utilisé dans notre projet, mais il est présent afin de faciliter une mise au propre futur de celui-ci. SASS est une extension du CSS de base, que nous avons utilisé avec des fichiers .scss. Ces fichiers permettent de réaliser des règles css plus facilement et plus rapidement. Cependant, au final, les fichiers scss n'ont pas été autant utilisés que nous le voulions et nous avons fini par utiliser des fichiers css de base.

Travail effectué

Presentation du resultat

Afin de permettre une facilité d'utilisation de notre API, nous avons mis en place une interface graphique (un site web) depuis lequel l'utilisateur pourra générer une série de questions utilisables dans le site du jeu de quizz.

Nous avons mis en place sur le site, 4 grand domaine de question:

- La génération de questions à date (ex: date de naissance (ou de mort) d'un artiste ?).
- La génération de questions à propos d'album (ex: quel album a ete realisee par X artiste ?).
- La génération de questions à chanson (ex: quelle chanson a été réalisée par X artiste ?).
- Le "Tout en un !" qui réalise une série de questions sur toutes les catégories précédemment citées.

De plus pour chaque domaine de question, vous aurez au choix de réaliser les question pour un artiste précis, ou bien de réaliser une série de X questions à partir d'artiste aléatoire (paramètre de filtrage pour les artiste possible).

Artistes

Question a Date	Question a Albums	Question a Chansons	Tout en un !
(exemple de resultat ci-dessous)	(exemple de resultat ci-dessous)	(exemple de resultat ci-dessous)	(exemple de resultat ci-dessous)
<pre>{ "question": "Quel est la date de naissance de Michael Jackson ?", "proposition": ["1957-10-28", "1958-08-29", "1963-02-15", "1961-06-02"], "index-reponse": 1 }</pre>	<pre>{ "question": "Quelle album a été réalisé par \Amelia Curran\ ?", "proposition": ["Storm In Your Brains", "Ambrosia", "War Brides"], "index-reponse": 3 }</pre>	<pre>{ "question": "Quelle chanson a été chanté par \Bill Nelson\ ?", "proposition": ["Przebojowa Kolekcja", "Boys Life", "Northern Dream", "Broken Bones And Bloody Kisses"], "index-reponse": 2 }</pre>	<pre>[{ "question": "Quel est la date de naissance de Michael Jackson ?", "proposition": ["1962-04-21", "1958-08-29", "1957-03-06", "1953-05-31"], "index-reponse": 1 }, { "question": "Quelle album a été réalisé par \Michael Jackson\ ?", "proposition": ["Przebojowa Kolekcja", "Got To Be There", "Boys Life", "Broken Bones And Bloody Kisses"], "index-reponse": 1 }, { "question": "Quelle album a été réalisé par \Michael Jackson\ ?", "proposition": ["Altar: Stop The Silence", "Alright", "Unicorn", "Got To Be There"], "index-reponse": 3 }]</pre>
<p>Retourne une question, composer d'une reponse juste et de 3 autres fausses, sur la date de naissance d'un artiste precis. Ou bien une serie de X questions/reponse sur des artistes pris aleatoirement.</p>	<p>Retourne une question, composer d'une reponse juste et de 3 autres fausses, sur les albums d'un artiste precis. Ou bien une serie de X questions/reponse sur des artistes pris aleatoirement.</p>	<p>Retourne une question, composer d'une reponse juste et de 3 autres fausses, sur les morceaux d'un artiste precis. Ou bien une serie de X questions/reponse sur des artistes pris aleatoirement.</p>	<p>Retourne une question, composer d'une reponse juste et de 3 autres fausses, sur la date de naissance, les albums et les morceaux d'un artiste precis. Ou bien une serie de X questions/reponse sur des artistes pris aleatoirement.</p>
<div>Artiste precis</div> <div>Serie aleatoire</div>	<div>Artiste precis</div> <div>Serie aleatoire</div>	<div>Artiste precis</div> <div>Serie aleatoire</div>	<div>Artiste precis</div> <div>Serie aleatoire</div>

(Les domaine de question)

De plus, en ce qui concerne la génération d'une série de questions, nous avons mis en place quelques options de filtrage permettant de définir le type d'artiste que nous souhaitons utiliser dans la génération. Nous avons pour ce faire créer les filtres suivants:

- Date de naissance minimum: afin de n'avoir que les artistes nés après une certaine date.
- Date de naissance maximum: afin de n'avoir que les artistes nés avant une certaine date.
- Nombre de Deezer fan minimum: Afin de selection un "niveau" de popularité, en effet plus le nombre de fan sera élevé plus l'artiste sera considéré comme "connu" et donc les question genre seront plus "abordable" pour le grand publique.

Date minimum l'artiste (Optionel)	Date maximum l'artiste (Optionel)
Nombre de minimum de Deezer fan par artiste (Optionel)	
Nombre de questions	Générer

(champs de renseignement des option)

Enfin les questions ainsi générées seront retournées à l'utilisateur sous un format JSON directement téléchargeable (via un bouton) et utilisable par le jeu de quiz qui n'aura qu'à placer le dit fichier dans le bon répertoire de son application. En effet le fichier téléchargez ce nommera "question.json" et aura un format ayant été décidé en coopération avec l'équipe du jeu de quiz afin d'être utilisable de manière simple et efficace sans que cela ne fasse intervenir de modification dans le code du jeu de quiz.

```
[{"question": "Quel est la date de naissance de David Frank ?", "proposition": ["1950-02-16", "1952-11-13", "1956-03-09", "1951-07-30"], "indexreponse": 1}, {"question": "Quel est la date de naissance de Axel Stordahl & His Orchestra ?", "proposition": ["1915-03-04", "1913-08-08", "1910-05-07", "1912-05-02"], "indexreponse": 1}, {"question": "Quel est la date de naissance de Ariana Grande ?", "proposition": ["1995-08-05", "1994-04-08", "1994-04-20", "1993-06-26"], "indexreponse": 3}, {"question": "Quel est la date de naissance de Anthony Santos ?", "proposition": ["1978-06-14", "1978-01-26", "1978-07-01", "1981-07-21"], "indexreponse": 3}, {"question": "Quel est la date de naissance de Lorne Balfe ?", "proposition": ["1976-01-31", "1971-06-16", "1976-02-23", "1977-11-05"], "indexreponse": 2}, {"question": "Quel est la date de naissance de Ricky Martin ?", "proposition": ["1969-07-25", "1972-12-27", "1976-09-01", "1971-12-24"], "indexreponse": 3}, {"question": "Quel est la date de naissance de Martin Garrix ?", "proposition": ["1994-10-24", "2000-03-10", "1996-05-15", "1991-07-22"], "indexreponse": 2}, {"question": "Quelle album a été réalisé par \"Fergie\" ?", "proposition": ["La Neve, Il Cielo, L'immenso", "I Colori Del Mio Universo", "Live 2007: Il Concerto", "The Dutchess"], "indexreponse": 3}, {"question": "Quelle album a été réalisé par \"Iggy Azalea\" ?", "proposition": ["Ignorant Art", "Ahmad", "For Lovers, Dreamers & Me", "She"], "indexreponse": 0}, {"question": "Quelle chanson a été réalisée par \"Kygo\" ?", "proposition": ["Firestone", "Ma Porte De Shed", "Anecdote", "Sapolin Numéro 148"], "indexreponse": 0}]
```

 Download

(JSON créer et téléchargeable par l'utilisateur)

Format du JSON (la valeur de champ en gras dépend de la question générée):

```
[
  {
    "question": "la question",
    "proposition": [
      "proposition-1",
      "proposition-2",
      "proposition-3",
      "proposition-4",
    ],
    "indexreponse": index
  }, { ... }
]
```

Vous retrouverez donc dans ce JSON, 3 champs principaux:

- Question: qui correspond à l'intitulé de la question qui sera posée à l'utilisateur "Quand est née Ariana Grande ?".
- Proposition: qui est un tableau de 4 éléments dont:
 - 3 proposition fausse
 - 1 proposition juste
- Indexreponse: qui correspond à l'index de la réponse juste dans le tableau de sa proposition.

Présentation du Code

Site

Notre site est assez basique. Il est composé d'une très grande page HTML qui définit plusieurs div que l'ont fait apparaître et disparaître pour simuler les différentes sections de notre application.

Il utilise du javascript avec la librairie JQuery et des requêtes ajax afin d'aller chercher sur l'API les ressources nécessaires à l'utilisation de notre application. Au final, le javascript est décomposé en plusieurs fichiers js:

- grid.js qui permet d'afficher ou de cacher les différentes parties de notre application en modifiant les propriétés display des <div> (display 'block' ou display 'none') en fonction de ce que l'on souhaite afficher.

```
document.getElementById('allArtistePrecis').addEventListener('click', function (event) {
    event.preventDefault();
    document.getElementById('artistesDiv').style.display = 'none';
    document.getElementById('dateArtistePrecisDiv').style.display = 'none';
    document.getElementById('dateArtisteAleatoireDiv').style.display = 'none';
    document.getElementById('albumArtistePrecisDiv').style.display = 'none';
    document.getElementById('albumArtisteAleatoireDiv').style.display = 'none';
    document.getElementById('chansonArtistePrecisDiv').style.display = 'none';
    document.getElementById('chansonArtisteAleatoireDiv').style.display = 'none';
    document.getElementById('allArtistePrecisDiv').style.display = 'block';
    document.getElementById('allArtisteAleatoireDiv').style.display = 'none';
    // document.getElementById('backToArtistDivFromPrecis').style.display = 'none';
    // document.getElementById('backToArtistDivFromAleatoire').style.display = 'block';
});
```

(Exemple d'un bouton changeant la div pour accéder à la rubrique "Artiste Précis" de la catégorie "Tout en un")

- artistes.js (qui interagit directement avec l'API de Wasabi) et artistsdb.js (qui utilise notre propre base de données) qui permettent de faire les requêtes sur l'API en récupérant les données des formulaires et en les envoyant avec des requêtes AJAX. De plus, c'est ici que nous effectuons les changements HTML/CSS permettant d'afficher le résultat ainsi obtenu sur la page de l'utilisateur.

```
function generateAllQuestion (nbQuestion) {
  document.querySelector('#allArtisteAleatoireDiv textarea').innerHTML = '';
  document.querySelector('#allArtisteAleatoireDiv p').innerHTML = 'Loading...';
  document.querySelector('#genererXQuestionsAll').disabled = true;

  let myURL = `/db/questions/gen10/allRandom?nbQuestion=${nbQuestion}`;

  const popularity = document.getElementById('popularityAll').value;
  if (popularity !== '') {
    myURL += `&popularity=${popularity}`;
  }

  const minDate = document.getElementById('dateMinAll').value;
  if (minDate !== '') {
    myURL += `&minDate=${minDate}`;
  }

  const maxDate = document.getElementById('dateMaxAll').value;
  if (maxDate !== '') {
    myURL += `&maxDate=${maxDate}`;
  }

  $.ajax({
    url: myURL,
    data: {},
    success: (res) => {
      document.querySelector('#allArtisteAleatoireDiv p').innerHTML = '';
      document.getElementsByClassName('downloadDivFromAleatoire')[3].style.display = 'block';
      document.getElementsByClassName('backToArtistDivFromAleatoire')[3].style.display = 'block';
      const textArea = document.querySelector('#allArtisteAleatoireDiv textarea');

      textArea.innerHTML = JSON.stringify(res);
      document.querySelector('#genererXQuestionsAll').disabled = false;
    },
    error: (error) => {
      document.querySelector('#allArtisteAleatoireDiv p').innerHTML = '';
      const textArea = document.querySelector('#allArtisteAleatoireDiv textarea');

      textArea.innerHTML = (error && error.error) ? error.error : JSON.stringify(error);
    }
  });
}
```

(Exemple de fonction récupérant les données d'un formulaire pour l'envoyer à l'API puis l'afficher à l'utilisateur)

- navbar.js, qui gère toute la partie navigation entre les grandes parties de l'application, mais qui du coup ne sert qu'à passer de l'accueil à la partie artiste et inversement.

Le site est décomposé en deux grandes parties:

- La partie Wasabi, permettant de générer des questions en utilisant l'API de Wasabi directement, ce qui a l'avantage d'être à jour avec les données de Wasabi, mais le désavantage d'être lent / bloquant.
- La partie Base de Données, accessible sur la route /db qui permet de générer des questions en utilisant notre propre Base de Données préalablement rempli par un script utilisant l'API Wasabi. Cette deuxième partie a l'avantage d'être beaucoup plus rapide dans la réalisation des requêtes.

API

L'API de notre application est la partie qui permet de générer des questions en utilisant le site Wasabi. Pour ceci, nous offrons donc 12 routes:

- Une route (/fake/date) permettant de générer une fausse date avec une vraie, elle est utilisée dans notre génération de question par date avec un artiste précis.

The screenshot shows a Postman interface for a GET request to `ter.timmydaumas.fr/api/fake/date?date=1994-09-20`. The 'Query Params' section contains a table with the following data:

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	date	1994-09-20			
	Key	Value	Description		

The 'Body' tab is selected, showing the response in JSON format:

```
1 {
2   "date": "1994-09-20",
3   "fakeDates": [
4     "1993-03-05",
5     "1999-09-22",
6     "1998-05-10"
7   ]
8 }
```

The status bar indicates a 200 OK response with a 25 ms response time and 317 B of data.

(Requête Postman avec son résultat pour la route /api/fake/date)

- Une route (/fake/chanson) permettant de trouver trois chansons pour pouvoir les utiliser comme réponses fausses lors d'une question de type chanson, également utilisée dans notre génération de question sur des chansons pour un artiste précis,

The screenshot shows a Postman interface for a GET request to the URL `ter.timmydaumas.fr/api/fake/chanson`. The request is configured with the method 'GET' and the URL. Below the URL bar, there are tabs for 'Params', 'Authorization', 'Headers (6)', 'Body', 'Pre-request Script', 'Tests', and 'Settings'. The 'Params' tab is active, showing a table with columns 'KEY', 'VALUE', and 'DESCRIPTION'. The table has one row with 'Key' and 'Value'.

Below the 'Params' tab, there is a 'Query Params' section. The response is displayed in the 'Body' tab, showing a JSON object with the key 'fakeChansons' and a value of an array containing three strings: 'Old Tone, New Song', 'Lucky', and 'Flowers In The Hat'. The status bar at the top right indicates a 200 OK response, a time of 8.17 s, and a size of 311 B. The 'Save Response' button is visible.

KEY	VALUE	DESCRIPTION
Key	Value	Description

```

1  {
2    "fakeChansons": [
3      "Old Tone, New Song",
4      "Lucky",
5      "Flowers In The Hat"
6    ]
7  }

```

(Requête Postman avec son résultat pour la route /api/fake/chanson)

- Une route (/fake/album) permettant de trouver trois albums pour pouvoir les utiliser avec l'album voulu et en faire une question dans une partie "Artiste précis" de la génération de questions sur albums notre application,

The screenshot shows a Postman interface for a GET request to the URL `ter.timmydaumas.fr/api/fake/album`. The request is configured with the method GET and the URL. Below the request configuration, there are tabs for Params, Authorization, Headers (6), Body, Pre-request Script, Tests, and Settings. The Params tab is active, showing a table with columns KEY, VALUE, and DESCRIPTION. The table contains one row with the key 'Key' and the value 'Value'. The DESCRIPTION column is empty. To the right of the table is a 'Bulk Edit' button.

Below the Params tab, there is a 'Query Params' section. The response tab is active, showing the response status as 200 OK, with a response time of 7.38 s and a response size of 311 B. The response body is displayed in a 'Pretty' format, showing a JSON object with a key 'fakeChansons' and a value of an array containing three strings: 'Hearts', 'Between A Rock And A Hard Place', and 'Melt'.

KEY	VALUE	DESCRIPTION
Key	Value	Description

```

1  {
2    "fakeChansons": [
3      "Hearts",
4      "Between A Rock And A Hard Place",
5      "Melt"
6    ]
7  }

```

(Requête Postman avec son résultat pour la route /api/fake/album)

- Une route (/group/:name/dates) permettant de trouver les dates de naissance et de mort d'un artiste à l'aide de son nom, ce qui est utilisé dans la partie "Artiste Précis" de la génération de question à date de notre application,

```
// Pour la generation d'une date sur un artiste
router.get('/group/:name/dates', async function (req, res, next) {
  const artist = await Artist.findOne({ name: req.params.name });

  if (!artist) {
    return res.status(404).json({ error: 'Artist not found' });
  }

  if (!artist.deathDate && req.query.typeDate === 'death') {
    return res.status(404).json({ error: 'This artist is alive !' });
  }

  res.status(200).json({ birthDate: artist.birthDate, endDate: artist.deathDate, ended: artist.deathDate !== null });
});
```

(Morceau de code pour la route /api/group/:name/dates)

The screenshot shows a Postman interface with a GET request to the URL 'ter.timmydaumas.fr/api/group/Ariana Grande/dates'. The 'Send' button is visible. Below the request, the 'Query Params' section is empty. The 'Body' tab is selected, showing a JSON response with the following structure:

```
{
  "birthDate": "1993-06-26",
  "endDate": "",
  "ended": false
}
```

The response status is 200 OK, with a response time of 118 ms and a size of 296 B. The 'Save Response' button is also visible.

(Requête Postman avec son résultat pour la route /api/group/:name/dates)

- Une route (/group/:name/chanson) permettant de récupérer une chanson d'un artiste précis en donnant son nom que nous utilisons dans les générations de questions d'un artiste précis sur ses chansons,

GET ▼ ter.timmydaumas.fr/api/group/Ariana Grande/chanson Send ▼

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
	Key	Value	Description		

Body Cookies Headers (7) Test Results 200 OK 3.13 s 272 B Save Response ▼

Pretty Raw Preview Visualize JSON ▼ ≡

```

1  {
2    "song": "Snow In California"
3  }

```

(Requête Postman avec son résultat pour la route /api/group/:name/chanson)

- Une route (/group/:name/album) permettant de récupérer un album d'un artiste précis en donnant son nom, nous l'utilisons également dans les parties "Artiste précis" de notre application,

GET ▼ ter.timmydaumas.fr/api/group/Ariana Grande/album Send ▼

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
	Key	Value	Description		

Body Cookies Headers (7) Test Results 200 OK 630 ms 265 B Save Response ▼

Pretty Raw Preview Visualize JSON ▼ ≡

```

1  {
2    "song": "Yours Truly"
3  }

```

(Requête Postman avec son résultat pour la route /api/group/:name/album)

- Une route (/group/:name/all) pour générer une question de chaque type sur un artiste précis,

```

1  {
2    {
3      "question": "Quel est la date de naissance de Ariana Grande ?",
4      "proposition": [
5        "1993-11-06",
6        "1998-02-02",
7        "1993-06-26",
8        "1991-11-05"
9      ],
10     "index-reponse": 2
11   },
12   {
13     "question": "Quelle album a été réalisé par \"Ariana Grande\" ?",
14     "proposition": [
15       "Year In A Red Room",
16       "A Touch Of Medieval Darkness",
17       "Yours Truly",
18       "Read Music/Speak Spanish"
19     ],
20     "index-reponse": 2
21   },
22   {
23     "question": "Quelle chanson a été réalisé par \"Ariana Grande\" ?",
24     "proposition": [
25       "Frozen Heart",
26       "On S'attache",
27       "Jacques A Dit",
28       "Yours Truly"
29     ],
30     "index-reponse": 3
31   }
32 }

```

(Requête Postman avec son résultat pour la route /api/group/:name/all)

- Une route (/questions/gen10/artistDates) permettant de générer X questions (bien que 10 soit la valeur par défaut) sur les dates d'artistes pris au hasard. Afin de choisir le nombre de questions, il faut utiliser un query parameter appelé nbQuestion.

GET ter.timmydaumas.fr/api/questions/gen10/artistDates?nbQuestion=10 Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	nbQuestion	10			
	Key	Value	Description		

Body Cookies Headers (7) Test Results 200 OK 6.45 s 1.68 KB Save Response

Pretty Raw Preview Visualize JSON

```

1  {
2    "question": "Quel est la date de naissance de Roy Clark ?",
3    "proposition": [
4      "1933-04-15",
5      "1931-11-05",
6      "1930-01-30",
7      "1928-04-23"
8    ],
9    "index-reponse": 0
10  },
11  {
12    "question": "Quel est la date de naissance de Tony Bennett ?",
13    "proposition": [
14      "1926-08-08",
15      "1926-08-08",
16      "1926-08-08",
17      "1926-08-08"
18    ],
19    "index-reponse": 1
20  }
21  ]

```

(Requête Postman avec son résultat pour la route /api/questions/gen10/artistDates)

- Une route (/questions/gen10/artistAlbums) permettant de permettant de générer X questions (bien que 10 soit la valeur par défaut) sur les albums d'artistes pris au hasard. Afin de choisir le nombre de questions, il faut utiliser un query parameter appelé nbQuestion.

ter.timmydaumas.fr/api/questions/gen10/artistAlbums?nbQuestion=10

GET ter.timmydaumas.fr/api/questions/gen10/artistAlbums?nbQuestion=10 Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	nbQuestion	10			
	Key	Value	Description		

Body Cookies Headers (7) Test Results 200 OK 5.46 s 1.98 KB Save Response

Pretty Raw Preview Visualize JSON

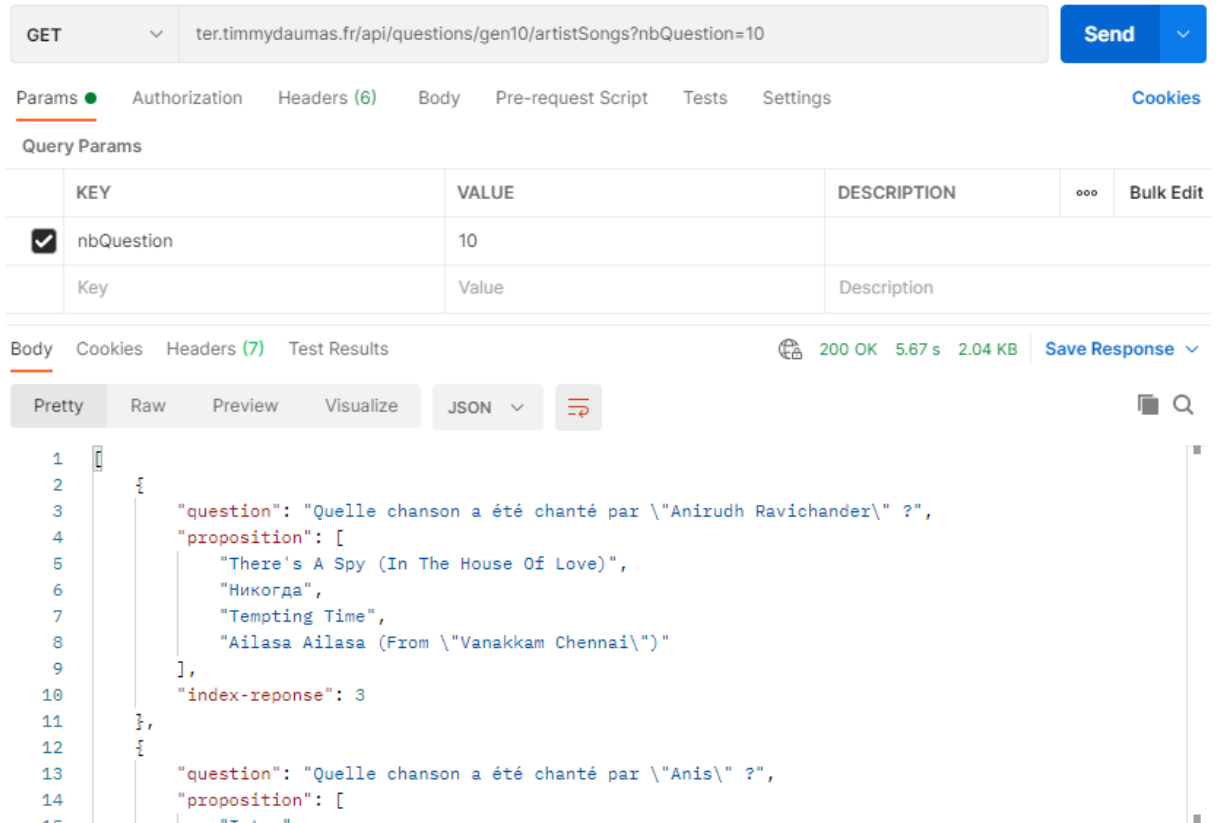
```

1  {
2    "question": "Quelle album a été réalisé par \"Bugle\" ?",
3    "proposition": [
4      "Buena Vista Social Club",
5      "Single",
6      "Nunca Fui Yo",
7      "Cardiac Arrest"
8    ],
9    "index-reponse": 1
10  },
11  {
12    "question": "Quelle album a été réalisé par \"Buika\" ?",
13    "proposition": [
14      "Buena Vista Social Club",
15      "Single",
16      "Nunca Fui Yo",
17      "Cardiac Arrest"
18    ],
19    "index-reponse": 1
20  }

```

(Requête Postman avec son résultat pour la route /api/questions/gen10/artistAlbums)

- Une route (/questions/gen10/artistSongs) permettant de permettant de générer X questions (bien que 10 soit la valeur par défaut) sur les chansons d'artistes pris au hasard. Afin de choisir le nombre de questions, il faut utiliser un query parameter appelé nbQuestion.



GET ▼ ter.timmydaumas.fr/api/questions/gen10/artistSongs?nbQuestion=10 Send ▼

Params ● Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	nbQuestion	10			
	Key	Value	Description		

Body Cookies Headers (7) Test Results 200 OK 5.67 s 2.04 KB Save Response ▼

Pretty Raw Preview Visualize JSON ▼ 🔍

```

1  {
2    {
3      "question": "Quelle chanson a été chanté par \"Anirudh Ravichander\" ?",
4      "proposition": [
5        "There's A Spy (In The House Of Love)",
6        "Никогда",
7        "Tempting Time",
8        "Ailasa Ailasa (From \"Vanakkam Chennai\")"
9      ],
10     "index-reponse": 3
11   },
12   {
13     "question": "Quelle chanson a été chanté par \"Anis\" ?",
14     "proposition": [
15       "Ailasa Ailasa (From \"Vanakkam Chennai\")",
16       "There's A Spy (In The House Of Love)",
17       "Никогда",
18       "Tempting Time"
19     ]
20   }
21 }

```

(Requête Postman avec son résultat pour la route /api/questions/gen10/artistSongs)

- Une route (/questions/gen10/allRandom) permettant de générer X questions (bien que 10 soit la valeur par défaut) sur les trois types de questions disponibles sur des artistes pris au hasard. Cette route est la plus intéressante car elle rassemble les trois précédentes. Afin de choisir le nombre de questions, il faut utiliser un query parameter appelé nbQuestion.

GET ▼ ter.timmydaumas.fr/api/questions/gen10/allRandom?nbQuestion=10 Send ▼

Params ● Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	nbQuestion	10			
	Key	Value	Description		

Body Cookies Headers (7) Test Results 200 OK 49.00 s 1.76 KB Save Response ▼

Pretty Raw Preview Visualize JSON ▼ ≡

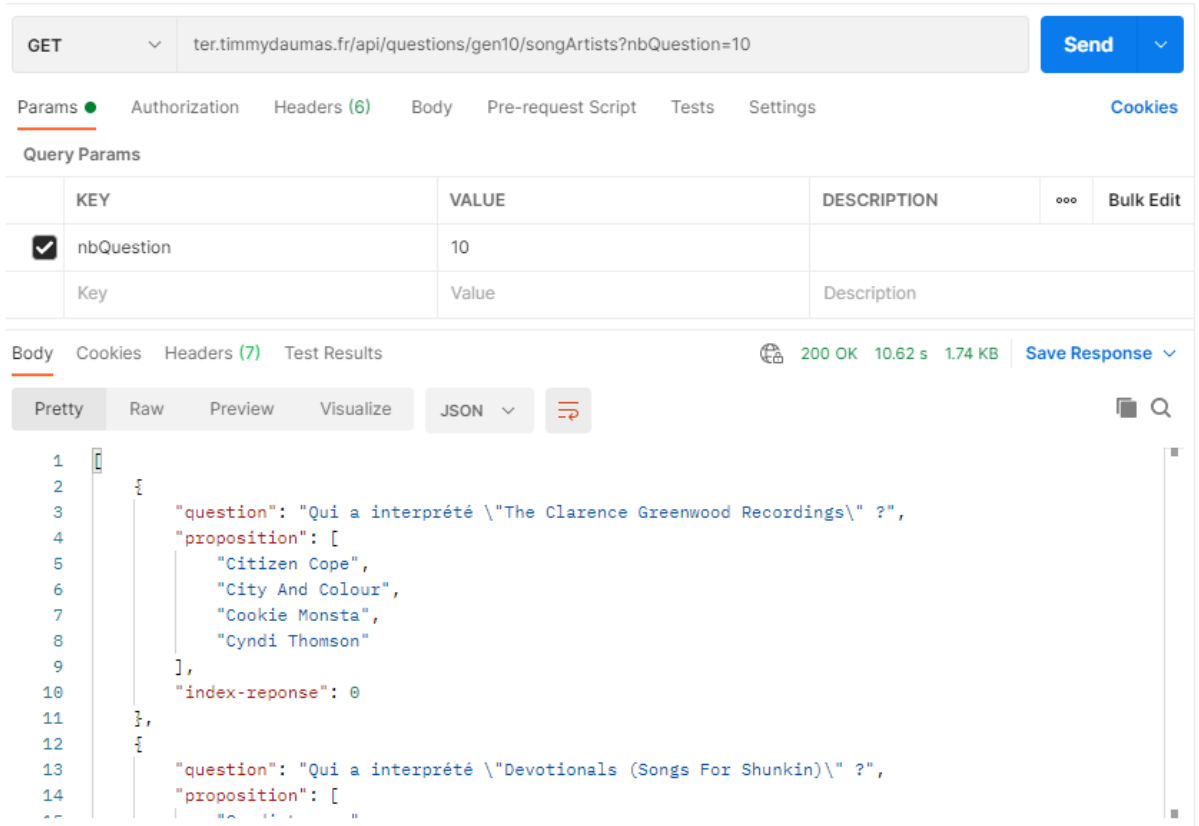
```

1  [
2    {
3      "question": "Quel est la date de naissance de Dominik Eulberg ?",
4      "proposition": [
5        "-3",
6        "1978",
7        "2",
8        "5"
9      ],
10     "index-reponse": 1
11   },
12   {
13     "question": "Quel est la date de naissance de Don Costa ?",
14     "proposition": [
15       "1898",
16       "1908",
17       "1918",
18       "1928",
19       "1938"
20     ]
21   }
22 ]

```

(Requête Postman avec son résultat pour la route /api/questions/gen10/allRandom)

- Une route (/questions/gen10/songArtists) qui n'est pas utilisé par le site, mais qui permet de générer des questions du type "Qui a interprété X chanson ?". Cependant, nous n'avons pas utilisé cette route pour le moment car nous nous sommes concentré sur les questions portant sur les artistes.



GET ter.timmydaumas.fr/api/questions/gen10/songArtists?nbQuestion=10 Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	nbQuestion	10			
	Key	Value	Description		

Body Cookies Headers (7) Test Results 200 OK 10.62 s 1.74 KB Save Response

Pretty Raw Preview Visualize JSON ≡

```

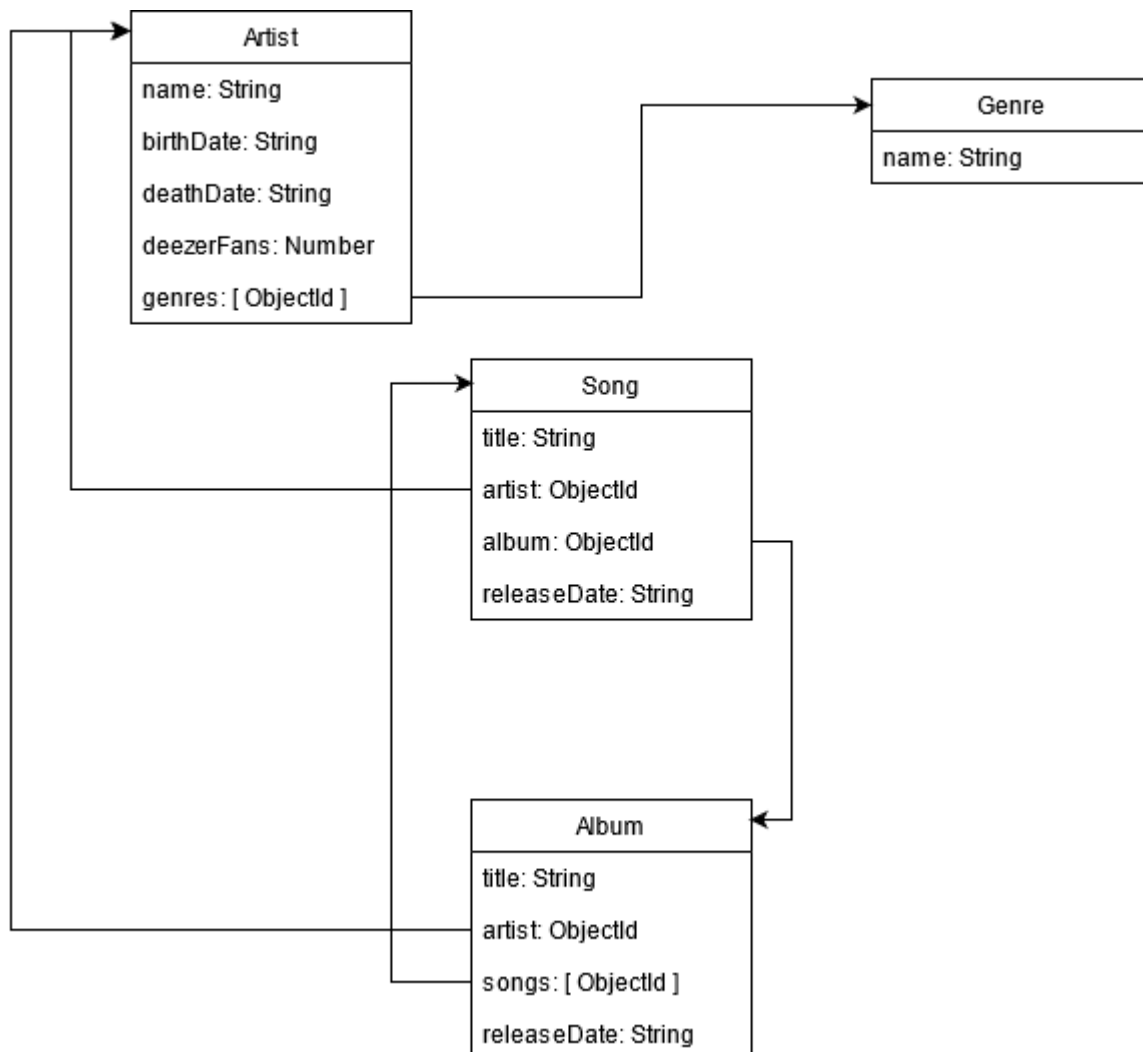
1  [
2    {
3      "question": "Qui a interprété \"The Clarence Greenwood Recordings\" ?",
4      "proposition": [
5        "Citizen Cope",
6        "City And Colour",
7        "Cookie Monsta",
8        "Cyndi Thomson"
9      ],
10     "index-reponse": 0
11   },
12   {
13     "question": "Qui a interprété \"Devotionals (Songs For Shunkin)\" ?",
14     "proposition": [
15       "Devotionals (Songs For Shunkin)"
16     ]
17   }
18 ]

```

(Requête Postman avec son résultat pour la route /api/questions/gen10/songArtists)

Base de données

La base de données utilise les schémas suivant:



(Schémas de la base de données)

Ici, ObjectId représente une référence sur un objet d'une autre table et [ObjectId] représente un tableau de références à des objets d'une autre table.

La base de données est remplie grâce au script populateDatabase.js qui peut être appelé en utilisant la commande "npm run populate".

```
MongoDB Connected
200 artists done out of 77491
400 artists done out of 77491
600 artists done out of 77491
800 artists done out of 77491
1000 artists done out of 77491
```

(Exemple du script en pleine exécution)

De plus, il existe un script “npm run test” afin de faire quelques vérifications sur la rapidité de différentes méthodes.

```
Array Some Length: 352302
Array IndexOf Length: 352302
BD length (should be the same): 352302
Calculating perfs on 1000 tests
BD mean time: 208.2196130399704 ms
Array Some mean time: 162.27064926052094 ms
Array IndexOf mean time: 0.4053449945449829 ms
```

(Résultats d'exécution de npm run test sur le vps)

En effet, nous ne voulions pas qu'il soit possible de rentrer plusieurs objets portant le même nom, car il arrive dans l'API Wasabi qu'un objet soit présent dans plusieurs endroits. Il a donc fallu faire des vérifications. Or, la méthode .find fournie par Mongoose est assez lente, ce qui demandait de laisser tourner le script une grande période de temps. Afin de pouvoir aller plus vite dans le remplissage de la base de données, il a été décidé de consommer plus de RAM. Les documents sont donc stockés dans des arrays enregistrant les noms de chaque objet, car la méthode .indexOf est très rapide, et la méthode .some, bien que plus lente, reste plus rapide que le .find de Mongoose. Cette implémentation possède l'avantage de rendre le script beaucoup plus rapide, mais a le désavantage que plus la base de données grandit, plus il faut de RAM pour stocker tous les objets. Sur mon VPS possédant uniquement 2GB de RAM, l'utilisation d'array devient impossible car le script npm remplit entièrement la mémoire du VPS et finit par se faire kill. Malheureusement, nous n'avons pour l'instant pas trouvé de solutions suffisamment rapides pour les utiliser autre que d'utiliser des array.

```
66800 artists done out of 77491
66800 artists done out of 77491
Killed
```

(Le processus se fait kill a cause du manque de RAM)

En ce qui concerne les routes de la base de données, elles sont les mêmes que celles de l'API. La différence vient du fait qu'elles sont plus faciles à manipuler et permettent donc de fournir des options supplémentaires. Il est donc possible de sélectionner le nombre de fans deezer minimum et maximum pour une requête, ou alors de définir une liste de genre musicaux que doivent respecter les artistes sur lesquels portent les questions.

Gestion de Projet

En ce qui concerne notre gestion de projet nous avons eu une approche se rapprochant de la méthodologie SCRUM. En effet, nous avons découpé nos phases de développement en SPRINT de 2 semaines (la plupart du temps).

De ce fait, pour chacun de nos sprint nous avons défini des user stories (qui correspondent à des tâches à réaliser que nous nous sommes auto attribuer).

Sprint Goal	En tant que	Acteur	Je veux	Afin de
Thème	En tant que	Utilisateur	Je veux générer des questions	Afin d'alimenter un quizz
Epic	En tant que	Utilisateur	Je veux générer une question via une page web	Afin d'alimenter un quizz
Faire les User Stories et mettre en place un squelette des pages principales du site	En tant que	Utilisateur	Je veux accéder à un site web	Afin d'utiliser l'application
	En tant que	Utilisateur	Je veux avoir accès à un formulaire	Afin de savoir quelles sont les informations nécessaires à la génération
	En tant que	Utilisateur	Je veux voir à quoi ressemble les pages du futur site web	Afin de savoir ce qu'il est possible de faire avec l'application
Commencer à récupérer des réponses à nos requêtes auprès de la base de donnée Wasabi afin de créer des questions simples	En tant que	Utilisateur	Je veux récupérer une information simple (date de naissance) sur un artiste	Afin d'en faire une question
	En tant que	Utilisateur	Je veux récupérer une information simple (date de sortie) sur un album	Afin d'en faire une question
	En tant que	Utilisateur	Je veux récupérer une information simple (date de sortie) sur une chanson	Afin d'en faire une question
	En tant que	Utilisateur	Je veux récupérer une information	Afin d'en faire une question
	En tant que	Utilisateur	Je veux créer des fausses dates à partir d'une vraie date	Afin d'en faire une question sans avoir à créer soit même des fausses dates
	En tant que	Utilisateur	Je veux récupérer un JSON de ma	Afin d'alimenter un quizz
	En tant que	Utilisateur	Je veux générer une question avec	Afin d'en faire un JSON

(user stories de nos 2 premiers sprints)

De plus pour chacune de nos user stories, nous avons associé une complexité, une estimation du temps de réalisation (en fonction de la complexité).

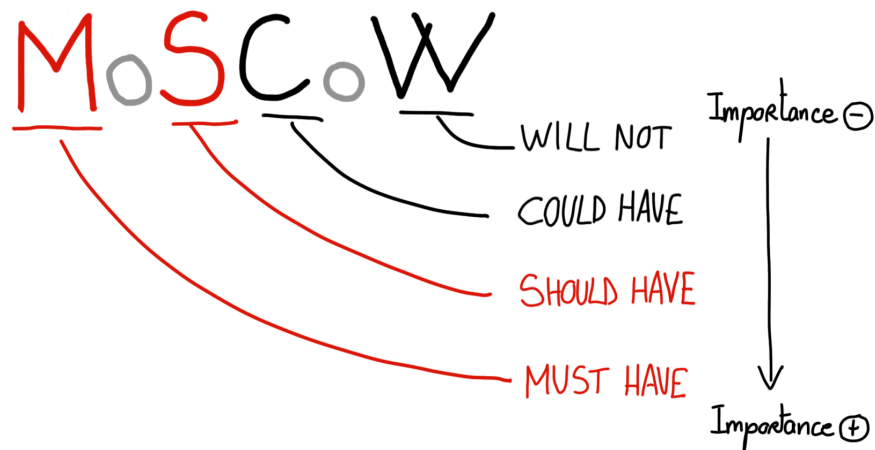
Complexité	Temps (h:m:s)
1	00:45:00
5	03:45:00
5	03:45:00
5	03:45:00
2	01:30:00
2	01:30:00
3	02:15:00
3	02:15:00
2	01:30:00
2	01:30:00

(complexité et estimation du temps de nos 2 premiers sprints)

Pour finir nous avons aussi décidé pour chacune de nos tâches d'attribuer un degré d'importance (suivant la méthode MoSCoW) permettant de définir leur criticité au sein du projet.

MoSCoW (en %)
Must ▼
Must ▼
Must ▼
Must ▼
Must ▼
Must ▼
Must ▼
Must ▼
Must ▼
Must ▼
Must ▼
Must ▼
Must ▼

(degré d'importance MoSCoW de nos 2 premiers sprints)



(schémas présentant les différents niveaux de la méthodologie MoSCoW)

Enfin nous avons réalisé toutes les deux semaine (pour nos fin de sprint) une réunion avec Monsieur Renevier afin de présenter l'avancée de notre projet, et de récolter des retours de ce derniers tel que:

- Les points à améliorer
- Les fonctionnalités à prioriser
- Ce qui va / ne va pas
- ...

Conclusion

En conclusion, le développement du projet d'application de génération de questions a été réalisé chaque semaine afin de fournir un moyen de générer des questions pour le jeu de Quiz correspondant à un autre TER. Le projet possède des forces, mais également des faiblesses qu'il faudra savoir corriger si le projet doit se poursuivre par la suite.

Le projet permet actuellement de générer des questions au format json demandé par l'autre groupe de TER (Une champ question pour la question, un champ proposition avec les 4 propositions de réponse et un champ nommé indexreponse représentant l'index de la bonne réponse dans le tableau des propositions).

Les questions pouvant être générées portent sur les artistes et permettent de générer 3 types de questions + le tout en un pour tout générer en même temps:

- Quel est la date de naissance de X ?

- Quelle chanson a été réalisée par X ?
- Quel album a été réalisé par X ?

Il est également possible de générer des questions de type “Qui a interprété Y ?”, une question sur des chansons, mais la route n’est pas utilisable directement sur le site et il faut donc réaliser la requête à la main.

Perspectives et réflexions personnelles

Timmy Daumas

Pour ma part, ce projet a été très intéressant. Dans un premier temps, ce projet a été un moyen de s’améliorer dans les différentes technologies que nous avons décidé d’utiliser, même si certaines ont été sous exploitées.

Dans un second temps, ce projet a été un moyen de se détendre après avoir dû s’occuper des autres projets qui ont reçu des difficultés lié à une partie de l’équipe choisie pour ceux-ci. Problèmes qui aurait donné le même résultat sans le découpage de l’équipe en 2 sous équipe. Jean-Baptiste étant quelqu’un qui travaille régulièrement et ayant un bon niveau, il a su réaliser toutes les tâches qu’il s’est attribué sans problème majeur. Il a toujours été possible d’avancer tout le long du projet sans se bloquer pendant des semaines.

Troisièmement, ce fût le premier projet où un client (M. Renevier ici) s’est autant impliqué, proposant des réunions toutes les semaines afin de suivre l’avancement, ce qui a été un facteur de motivation et d’aide non négligeable et qui a grandement participé à la réussite de ce projet (jusqu’à l’écriture du rapport qui a reçu un avis de M. Renevier sur une première version incomplète que nous avons réalisé).

Quatrièmement, je suis un peu déçu de ne pas avoir pu réaliser toutes les tâches de la manière dont je le désirais. Certaines parties sont sous-exploitées ou alors un peu redondantes. Twig a été vraiment très peu utilisé comparé à ses capacités, pareil pour SASS et cela donne un projet qui n’est pas optimal à ce niveau. Certaines parties sont un peu réécrite en dur plusieurs fois par exemple. Cependant, ces problèmes restent mineurs car le projet initial contient déjà les

technologies nécessaires au polissage du projet. Il ne faudrait donc pas longtemps à une équipe reprenant le projet pour venir corriger ces problèmes.

Enfin, l'IA, qui était un point que nous souhaitions aborder, a été finalement laissée de côté. Pas simplement à cause d'un manque de temps, mais surtout de par le fait que l'API Wasabi propose déjà dans ces champs toutes les informations nécessaires à la réalisation des questions et ne demandait jamais de vraiment s'intéresser à une IA. Il aurait été un peu inutile (et donc peu motivant) de venir créer une IA permettant de récupérer de manière un peu hasardeuse des informations étant déjà présentes dans des champs des réponses JSON.

Jean-Baptiste Lognon

En ce qui me concerne, j'ai dans la grande majorité apprécié réaliser ce projet. Tout d'abord faire partie de petite équipe (nous étions 2) a été très agréable car en effet cela a facilité la communication et la répartition du travail. De plus nous n'avions donc pas 2 personnes sur 5 à travailler. De ce fait, nous avons pu avoir une progression régulière dans notre projet afin d'obtenir un résultat satisfaisant. Le fait d'avoir un contact régulier avec le client nous a aussi beaucoup apporté car ce dernier nous indiquait les fonctionnalités sur lesquelles nous focaliser (car ce sont les plus intéressantes) et nous proposait des pistes en cas de blocage. Les seuls points que je regrette sur ce projet sont:

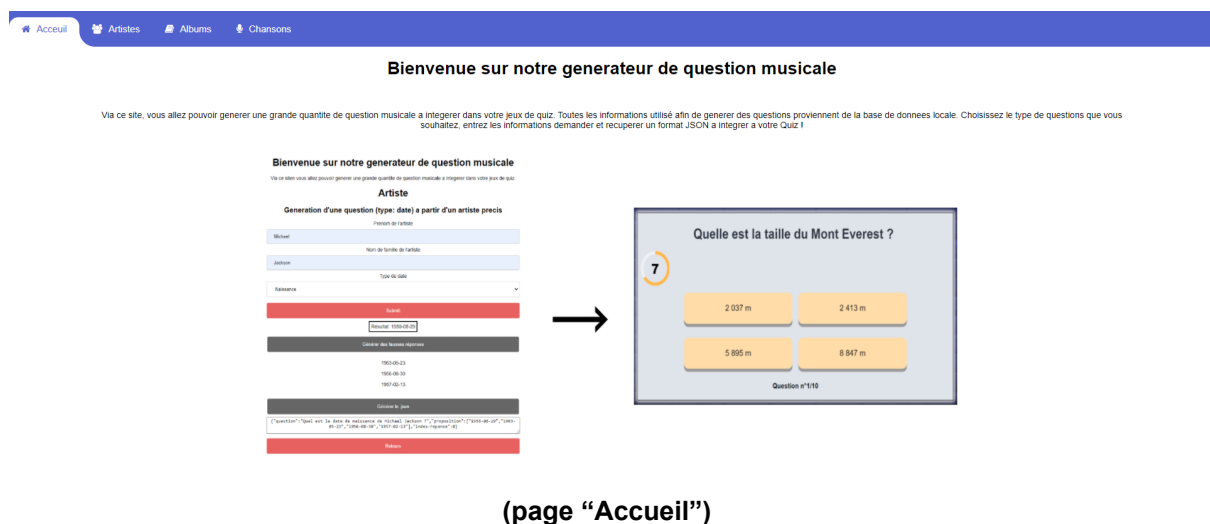
- Le fait de ne pas avoir eu le temps (et l'intérêt) de réaliser des questions à partir d'analyse du texte notamment car très peu d'entrées possédait du texte à analyser mais aussi car les informations étaient toutes trouvable dans des champs.
- Le fait d'avoir utilisé l'API de Wasabi qui à mon sens n'était pas très bien construite et aussi par le fait que le "thème" ne m'attire pas plus que ça.

Annexe

Guide utilisateur

Accueil

En lançant notre site, vous arriverez directement sur la page d'accueil ci-dessous. Sur cette dernière vous retrouverez un très bref résumé/explication de l'utilité de ce site et de son lien avec le projet du jeu de quizz. Et à partir de là, vous pourrez vous diriger vers la section suivante qui nous intéresse l'onglet "Artiste"



(page "Accueil")

Artistes

En cliquant sur l'onglet "Artiste" depuis l'Accueil vous arriverez donc sur la page suivante. Depuis cette page vous pourrez vous rendre    la g  n  ration de questions sur des artistes. Nous avons mis en place 4 grand domaine de question:

- La g  n  ration de questions    date (ex: date de naissance (ou de mort) d'un artiste ?).
- La g  n  ration de questions    propos d'album (ex: quel album a   te realisee par X artiste ?).

- La génération de questions à chanson (ex: quelle chanson a été réalisée par X artiste ?).
- Le “Tout en un !” qui réalise une série de questions sur toutes les catégories précédemment citées.

De plus pour chaque domaine de question, vous aurez au choix de réaliser les question pour un artiste précis, ou bien de réaliser une série de X questions à partir d’artiste aléatoire (paramètre de filtrage pour les artiste possible).

Accueil
Artistes
Albums
Chansons

Artistes

Question a Date

(exemple de resultat ci-dessous)

```
[{"question": "Quel est la date de naissance de Michael Jackson", "proposition": [{"date": "1957-10-28"}, {"date": "1958-08-28"}, {"date": "1961-05-15"}, {"date": "1961-06-02"}], "index-reponse": 1}]
```

Retourne une question, composer d'une réponse juste et de 3 autres fausses, sur la date de naissance d'un artiste précis. Ou bien une serie de X questions/reponse sur des artistes pris aléatoirement.

Artiste précis

Serie aleatoire

Question a Albums

(exemple de resultat ci-dessous)

```
[{"question": "Quel album a été réalisé par Amelia Curran", "proposition": [{"album": "Store In Your Brains"}, {"album": "Ambrosia"}, {"album": "War Brides"}], "index-reponse": 3}]
```

Retourne une question, composer d'une réponse juste et de 3 autres fausses, sur les albums d'un artiste précis. Ou bien une serie de X questions/reponse sur des artistes pris aléatoirement.

Artiste précis

Serie aleatoire

Question a Chansons

(exemple de resultat ci-dessous)

```
[{"question": "Quelle chanson a été chanté par Bill Nelson", "proposition": [{"chanson": "Przebojowa kolekcja"}, {"chanson": "Boys Life"}, {"chanson": "Northern Dream"}, {"chanson": "Broken Bones And Bloody Kisses"}], "index-reponse": 1}]
```

Retourne une question, composer d'une réponse juste et de 3 autres fausses, sur les morceaux d'un artiste précis. Ou bien une serie de X questions/reponse sur des artistes pris aléatoirement.

Artiste précis

Serie aleatoire

Tout en un !

(exemple de resultat ci-dessous)

```
[{"question": "Quel est la date de naissance de Michael Jackson", "proposition": [{"date": "1962-04-21"}, {"date": "1958-08-28"}, {"date": "1957-03-06"}, {"date": "1953-05-31"}], "index-reponse": 1}, {"question": "Quel album a été réalisé par Michael Jackson", "proposition": [{"album": "Przebojowa kolekcja"}, {"album": "Got To Be There"}, {"album": "Boys Life"}, {"album": "Broken Bones And Bloody Kisses"}], "index-reponse": 1}, {"question": "Quelle album a été réalisé par Michael Jackson", "proposition": [{"album": "Alter:Stop The Silence"}, {"album": "Alright"}, {"album": "Unicorn"}, {"album": "Got To Be There"}], "index-reponse": 3}]
```

Retourne une question, composer d'une réponse juste et de 3 autres fausses, sur la date de naissance, les albums et les morceaux d'un artiste précis. Ou bien une serie de X questions/reponse sur des artistes pris aléatoirement.

Artiste précis

Serie aleatoire

(page “Artiste”)

Génération à partir d’un artiste précis

Les questions à date, album et chanson

Continuons donc avec la génération de questions (a date, album ou chanson) à partir d’un artiste précis. Une fois sur cette page vous apercevrez un formulaire à remplir.

Artiste

Generation d'une question (type: date) a partir d'un artiste precis

Prenom de l'artiste

Nom de famille de l'artiste

Type de date

Naissance

Submit

Retours

(formulaire vide pour la génération (question à date) à partir d'un artiste précis)

Dans le formulaire ci-dessus, vous n'aurez qu'à saisir le prénom et le nom de l'artiste, mais aussi le type de date qui vous intéresse (naissance ou mort) puis valider.

A partir de la plusieurs comportement sont possible:

- L'artiste n'est pas trouver dans la base de donnée

Error: Artist not found

- L'artiste est trouvé mais ne possède pas la date qui nous intéresse

Error: This artist is alive !

- L'artiste est trouvé et possède la date qui nous intéresse.

Résultat: 1993-06-26

Une fois le résultat retourné et affiché à l'utilisateur, ce dernier va pouvoir cliquer sur une série de boutons permettant de générer des fausses réponses à partir de la "vrai" réponse, et de générer le JSON correspondant à la question souhaitée.

Artiste

Generation d'une question (type: date) a partir d'un artiste precis

Prenom de l'artiste

Ariana

Nom de famille de l'artiste

Grande

Type de date

Naissance

Submit

Résultat: 1993-06-26

Générer des fausses réponses

1993-03-12
1993-12-23
1994-07-09

Générer le .json

```
{"question": "Quel est la date de naissance de Ariana Grande ?", "proposition": ["1993-03-12", "1993-12-23", "1993-06-26", "1994-07-09"], "index-reponse": 2}
```

Retours

(formulaire remplis pour la génération (question à date) à partir d'un artiste précis)

Le "Tout en un !"

A présent passons à la présentation de la génération à partir d'un artiste précis pour le "Tout en un !" qui s'avère légèrement différent dans son utilisation. En arrivant sur cette page vous serez donc confronté à un formulaire que vous devrez remplir afin de continuer.

Artiste

Generation d'une question (type: date, album et chanson) a partir d'un artiste precis

Prenom de l'artiste

Nom de famille de l'artiste

Submit

Retours

(formulaire vide pour la génération à partir d'un artiste précis)

Dans ce formulaire vous devrez donc indiquer le prénom et le nom de l'artiste dont vous souhaitez générer les questions puis cliquer sur le bouton de validation. Une fois ceci fait vous pourrez voir apparaître sous le bouton de validation un champ textuel contenant le résultat au format json de la génération de question. Parmi les question générer vous retrouverez une question à date, une question sur un album et sur une chanson.

Artiste

Generation d'une question (type: date, album et chanson) a partir d'un artiste precis

Prenom de l'artiste

Nom de famille de l'artiste

Submit

```
[{"question": "Quel est la date de naissance de Ariana Grande ?", "proposition": ["1998-07-11", "1994-05-31", "1993-06-26", "1992-12-20"], "index-reponse": 2}, {"question": "Quelle album a été réalisé par \\\nAriana Grande\\\\\n", "proposition": ["Christmas Kisses", "Happy Soup", "The Negro Inside Me", "Christmas Kisses", "Stranger On The Sofa"], "index-reponse": 2}, {"question": "Quelle chanson a été réalisée par \\\nAriana Grande\\\\\n", "proposition": ["Adon Hashalom", "V'zocharta", "Hinay Yamim"], "index-reponse": 0}]
```

Retours

(formulaire remplis pour la génération à partir d'un artiste précis)

Génération de X questions à partir d'une série artiste aléatoire

En plus de pouvoir générer des question a partir d'un artiste précis, vous aurez aussi la possibilité de générer une série de X questions et ce à partir d'artiste pris aléatoirement. Pour ce faire, l'utilisateur va devoir remplir le formulaire ci-dessous qui comporte des parties "Optionnel" qui permettront de "filtrer" les artiste sélectionné selon des critères:

- Leur date de naissance minimal/maximal
- Leur nombre e fan sur Deezer (plus il y'a de fan, plus l'artiste est considéré comme "populaire" et donc "connue")

Artiste

Generation d'une serie de X questions (type: date, album et chanson) a partir d'artistes aleatoire

Date minimum l'artiste (Optionel)	Date maximum l'artiste (Optionel)
Nombre de minimum de Deezer fan par artiste (Optionel)	
Nombre de questions	Générer

(formulaire vide pour la génération à partir d'artistes aléatoires)

Une fois le formulaire rempli et validé alors vous pourrez apercevoir le résultat dans la textarea prévus à cet effet. De plus vous verrez apparaître un bouton

permettant de télécharger le résultat sous un fichier nommé “question.json” afin de directement l'intégrer au fichier de votre jeu de quizz.

Artiste

Generation d'une serie de X questions (type: date, album et chanson) a partir d'artistes aleatoire

1980	1985
1000000	
10	Générer

```
[{"question": "Quel est la date de naissance de Amy Winehouse ?", "proposition": ["1983-09-14", "1981-09-14", "1982-09-18", "1988-07-16"], "index-reponse": 0}, {"question": "Quel est la date de naissance de Justin Timberlake ?", "proposition": ["1983-08-08", "1981-01-31", "1977-03-06", "1984-11-06"], "index-reponse": 1}, {"question": "Quel est la date de naissance de Bobby Creekwater ?", "proposition": ["1982-11-20", "1982-08-31", "1980-11-19", "1978-12-04"], "index-reponse": 0}, {"question": "Quel est la date de naissance de Goldie Loc ?", "proposition": ["1976-02-10", "1983-05-12", "1980-01-16", "1976-01-04"], "index-reponse": 2}, {"question": "Quel est la date de naissance de La Fouine ?", "proposition": ["1981", "1982", "1983", "1978"], "index-reponse": 0}, {"question": "Quelle album a été réalisé par \\\\"Katy Perry\\\" ?", "proposition": ["Garçon D'Honneur", "Sus Primeras Canciones", "Ur So Gay", "Canciones"], "index-reponse": 2}, {"question": "Quelle album a été réalisé par \\\\"Beyoncé\\\" ?", "proposition": ["Adam Cohen", "Sentidos", "Abel", "Live At Wembley"], "index-reponse": 3}, {"question": "Quelle album a été réalisé par \\\\"Future\\\" ?", "proposition": ["Arrêt Facultatif", "Building 55", "Astronaut Status", "Plus Ça Change..."], "index-reponse": 2}, {"question": "Quelle album a été réalisé par \\\\"Avril Lavigne\\\" ?", "proposition": ["Ghost Music", "The Best Damn Thing", "The Power Of Music", "Land Of The Living"], "index-reponse": 1}, {"question": "Quelle chanson a été réalisée par \\\\"Britney Spears\\\" ?", "proposition": ["Los Pro", "Cartas Marcadas...", "Me Gusta", "3"], "index-reponse": 3}]
```

Download

Retours

(formulaire remplis pour la génération à partir d'artiste aléatoires)