SEBASTIAN PEREZ ET JESSICA CHAN

Générateur d'art dynamique

Projet Synthèse en techniques de l'informatique

DOCUMENT DE CONCEPTION

TABLE DES MATIÈRES

CONCEPTION

Les pages

Les pages du site web seront une page de connexion, une page d'inscription, une page principale pour les animations (page d'accueil), une page pour les partages des utilisateurs, une page du profil de l'usager courant et une page à propos.

Maquettes des pages

Une référence de l'allure du site web en termes d'emplacement des divers éléments que comportent les pages et la navigation. Les maquettes ont été conçues à l'aide de l'application Adobe XD.

Liens vers le prototype Adobe XD et les maquettes

Pour avoir un meilleur aperçu des maquettes du site web, on vous invite à suivre les liens suivants qui constituent le prototype du site web. Des commentaires y sont aussi disponibles pour aider à la compréhension de la navigation du site.

- Prototype interactive avec Adobe
- Maquettes des pages avec Adobe

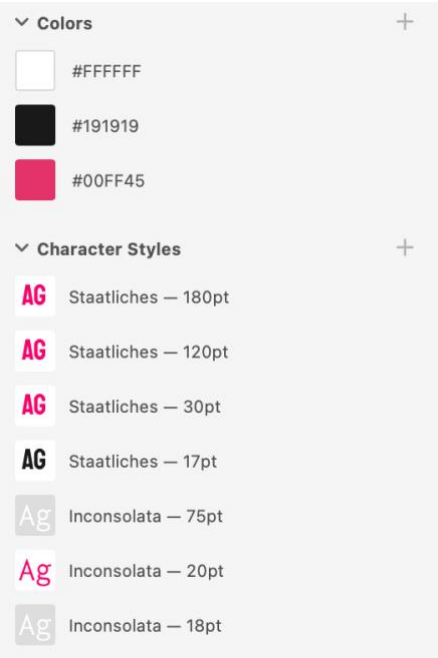
Couleurs et typographies

Famille de polices (Google fonts)

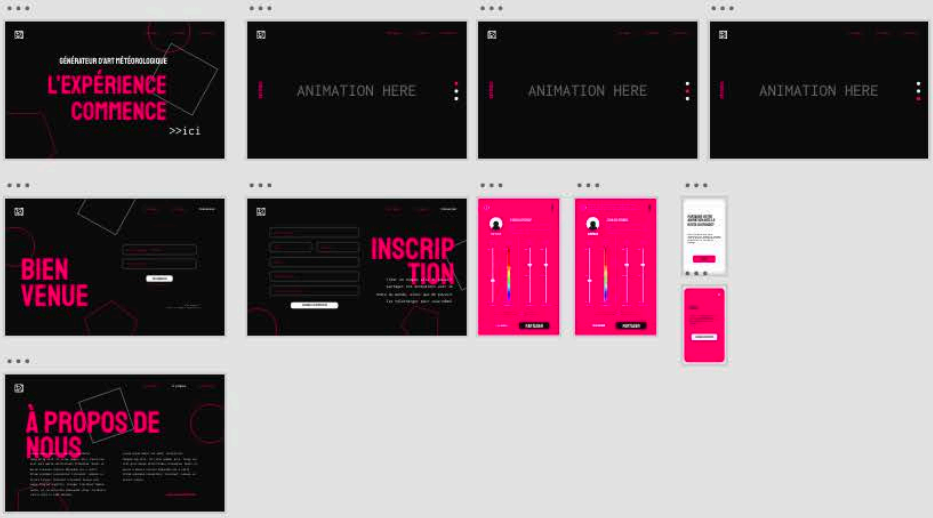
- Inconsolata Regular
- Staatliches Regular et Bold

Couleurs

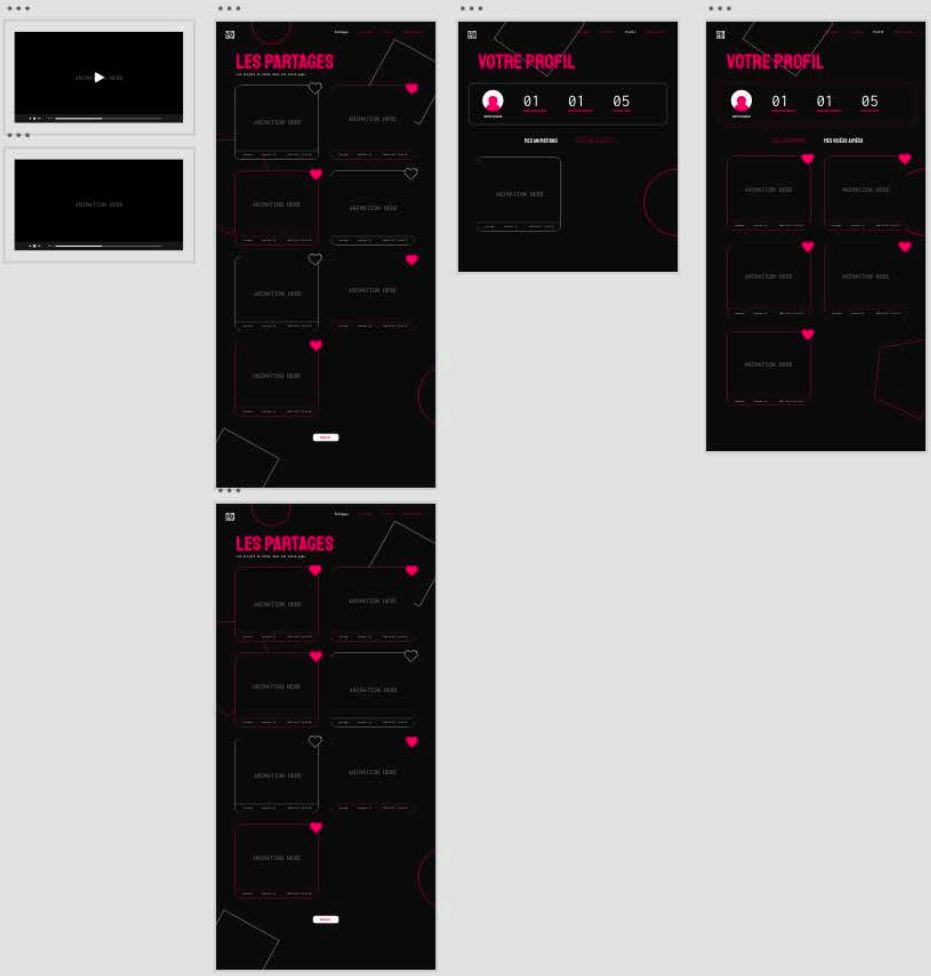
- Blanc
- Noir #191919
- Rose #00FF45



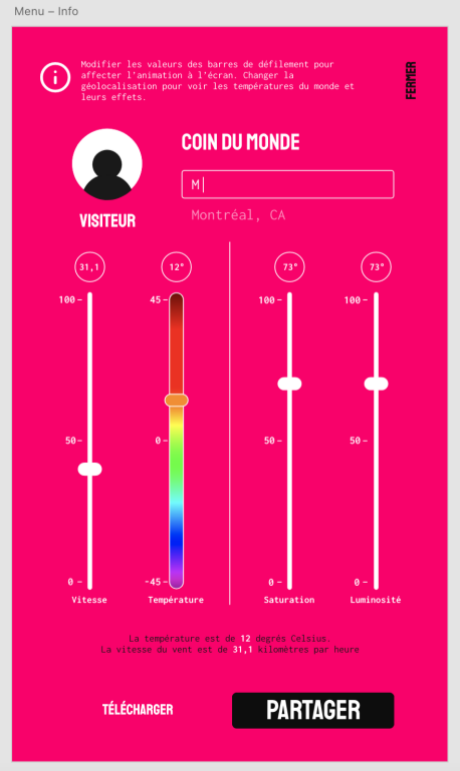
NON-MEMBER ACCESS



MEMBER ONLY ACCESS



Scrollbars



Les *scrollbars* des paramètres de la page d'animation seront des *widgets* personnalisés. Les scrollbars seront associés à certains aspects de la météo dépendamment des variables définies pour l'usager. Le nom de la barre de défilement correspondra au nom de la variable qu'elle modifie (lire section Algorithmes).

SVG et Javascript

Pour créer des barres de défilement propre à nous, nous utiliserons des fichiers SVG pour créer les composantes visuelles des scrollbars et nous y intégrerons du code Javascript afin de pouvoir y contrôler des événements.

Color picker

À partir de la barre de défilement de couleur, l'usager pourra choisir une couleur principale de l'animation. Ensuite, dans la classe de générateur d'algorithme, trois autres teintes dérivant de cette couleur seront créées modifiant le code Hex de la couleur sélectionnée. Ainsi, nos animations seront monochromatiques.

HSB

Nous utiliserons le HSB pour modifier les valeurs associées à la couleur principale dont le degré de la couleur, le pourcentage de saturation et le pourcentage de luminosité. Ces paramètres sont tous modifiables à partir des barres de défilement.

Interface Utilisateur

Mécanisme de saisie

Affichage d'un texte

Nous aurons une page à propos où nous voulons décrire de manière simple le fonctionnement du site web et les idées derrière étant donné que ce projet nous servira entre autres de pièce de portfolio pour entrer dans notre programme universitaire choisi.

Nombre

La page de profil de l'utilisateur contiendra une section de statistique où nous afficherons des nombres correspondants au nombre d'animation générés, au nombre d'animation aimées et au nombre d'animation partagées. De plus, nous aurons des nombres affichés au-dessus des barres de défilement du menu de paramètres modifiables sur la page d'animation. Ces nombres devront être formaté étant donné qu'ils représenteront des nombres décimaux.

Éléments différents :

- Logo format SVG retrouvable sur le coin en haut à gauche de la plupart des pages
- Icon information (i) dans le menu setting de la page animation
- Icon cœur sur le coin des cartes représentant si une animation a été aimé ou non par l'usager courant
- Photo de profil de l'utilisateur

Contenu multimédia

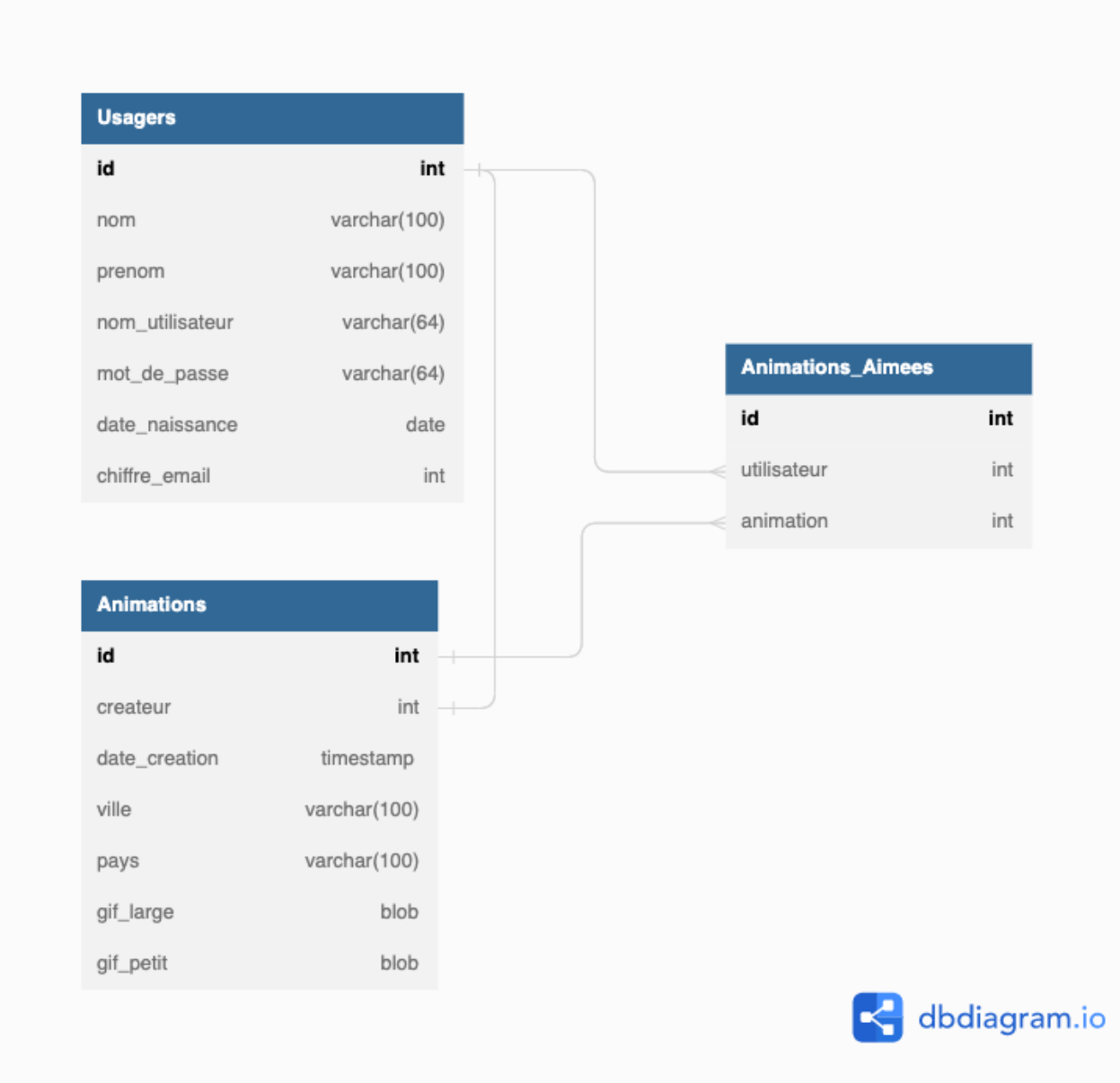
Dans la page des partages et dans la page du profil utilisateurs, les usagers pourront rejouer des animations précédemment sauvegardées sous forme de GIF. Ces GIF joueront automatiquement au chargement de la page et proviennent de la base de données où nous stockerons tous les contenus visuels associés aux utilisateurs.

Expression régulière

Nous validerons l'adresse courriel de l'usager avec les expressions régulières. De plus, nous utiliserons les trois premières lettres et les lettres 1, 3 et 5 après le '@' du courriel pour générer un chiffre qui détermine l'ordre des variables de l'usager

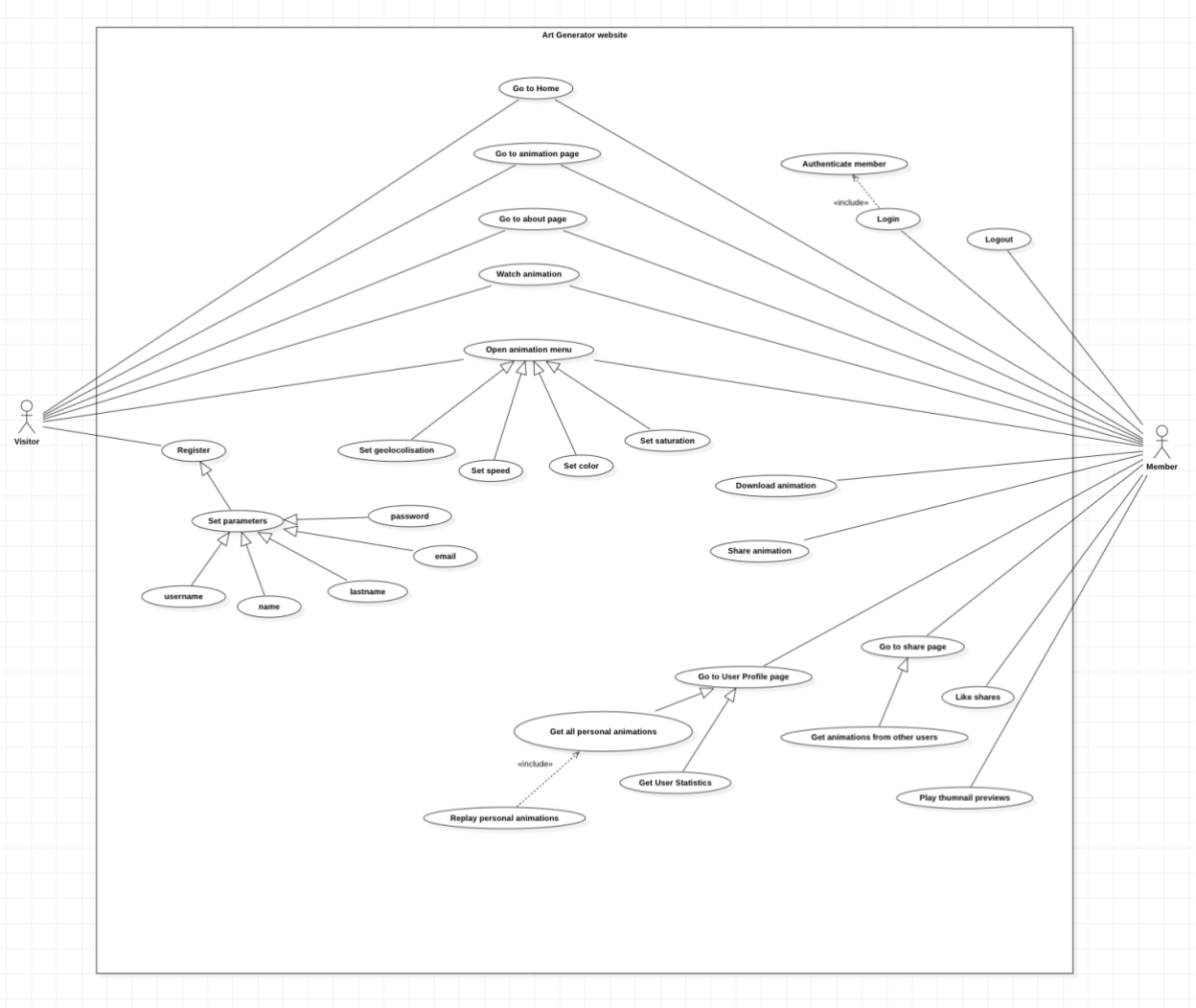
Base de données

Nous utiliserons PostgreSQL avec lequel nous aurons trois tables rationnelles : Usagers, Animations et Animatons_Aimees. La table Usagers sera principalement utilisée pour l'authentification de l'usager à la connexion. La date de naissance et le chiffre généré avec l'adresse courriel seront utilisés dans les algorithmes par la suite. La table Animations contiendra des animations partagées. Les données de cette table seront utilisées dans la page des partages pour afficher les animations partagées. Cette table contiendra deux GIF animés, le petit fichier sera un *thumbnail* et le fichier large sera le média à montrer une fois un partage sélectionné. Des informations relatives à l'animation (créateur, heure et lieu de création etc.) seront également disponibles dans cette table. La table Animations_Aimees sera une table de liaisons entre les Usagers et les Animations afin de garder en mémoire les animations qu'un utilisateur a aimé. Nous nous servirons de cette table pour afficher un cœur vide ou plein pour chaque carte de la page de partage afin que l'usager puisse voir s'il a aimé ou non une animation d'un autre usager.


Usagers
id int
nom varchar(100)
prenom varchar(100)
nom_utilisateur varchar(64)
mot_de_passe varchar(64)
date_naissance date
chiffre_email int
Animations_Aimees
id int
utilisateur int
animation int
Animations
id int
createur int
date_creation timestamp
ville varchar(100)
pays varchar(100)
gif_large blob
gif_petit blob
dbdiagram.io

UML

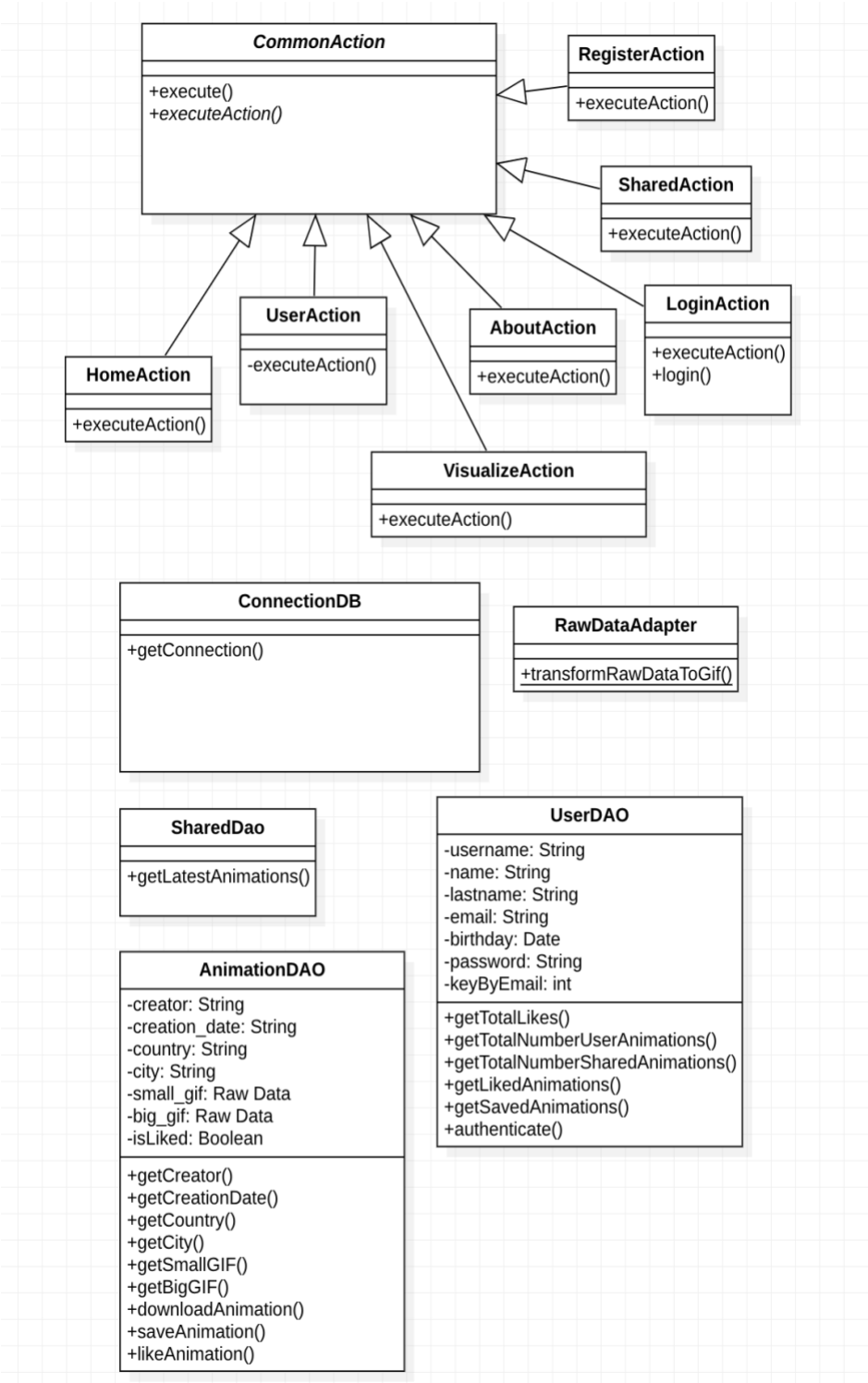
Diagramme des cas d'usage



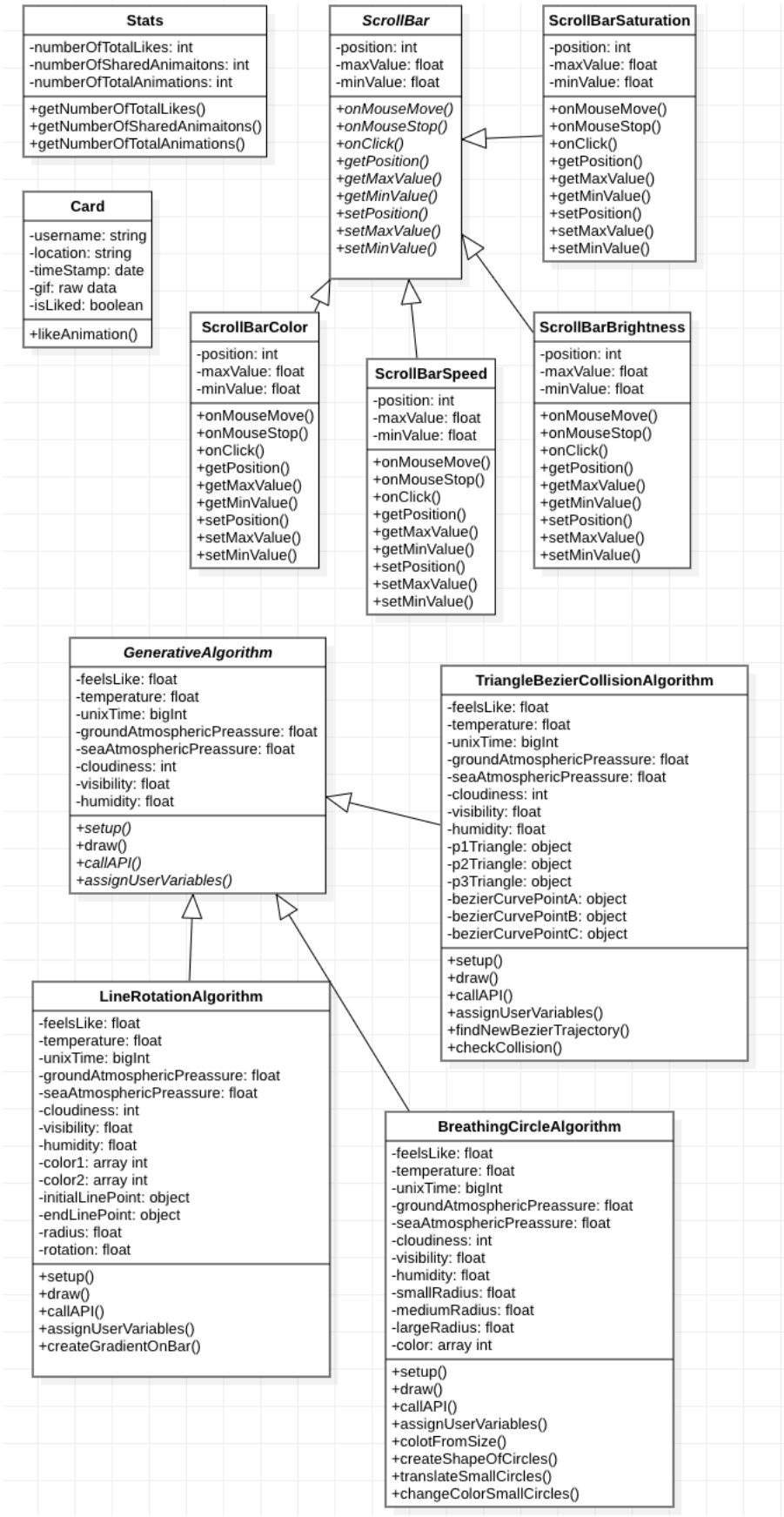
On démontre à partir du diagramme de cas d'usages les diverses fonctionnalités du site web auquel l'usager aura accès en tant que visiteur et en tant que membre. Les membres auront des droits qu'un visiteur ne possédera pas. Le visiteur devra s'enregistrer en créant un compte afin de devenir membre et pouvoir avoir le droit de faire des choses comme télécharger une animation, partager une animation, visionner la page des partages et la page de profil personnel, etc.

Diagramme de classes

PHP



Javascript


Stats
-numberOfTotalLikes: int
-numberOfSharedAnimaitons: int
-numberOfTotalAnimations: int
+getNumberOfTotalLikes()
+getNumberOfSharedAnimaitons()
+getNumberOfTotalAnimations()
ScrollBar
-position: int
-maxValue: float
-minValue: float
+onMouseMove()
+onMouseStop()
+onClick()
+getPosition()
+getMaxValue()
+getMinValue()
+setPosition()
+setMaxValue()
+setMinValue()
ScrollBarSaturation
-position: int
-maxValue: float
-minValue: float
+onMouseMove()
+onMouseStop()
+onClick()
+getPosition()
+getMaxValue()
+getMinValue()
+setPosition()
+setMaxValue()
+setMinValue()
Card
-username: string
-location: string
-timeStamp: date
-gif: raw data
-isLiked: boolean
+likeAnimation()
ScrollBarColor
-position: int
-maxValue: float
-minValue: float
+onMouseMove()
+onMouseStop()
+onClick()
+getPosition()
+getMaxValue()
+getMinValue()
+setPosition()
+setMaxValue()
+setMinValue()
ScrollBarSpeed
-position: int
-maxValue: float
-minValue: float
+onMouseMove()
+onMouseStop()
+onClick()
+getPosition()
+getMaxValue()
+getMinValue()
+setPosition()
+setMaxValue()
+setMinValue()
ScrollBarBrightness
-position: int
-maxValue: float
-minValue: float
+onMouseMove()
+onMouseStop()
+onClick()
+getPosition()
+getMaxValue()
+getMinValue()
+setPosition()
+setMaxValue()
+setMinValue()
GenerativeAlgorithm
-feelsLike: float
-temperature: float
-unixTime: bigInt
-groundAtmosphericPreassure: float
-seaAtmosphericPreassure: float
-cloudiness: int
-visibility: float
-humidity: float
+setup()
+draw()
+callAPI()
+assignUserVariables()
TriangleBezierCollisionAlgorithm
-feelsLike: float
-temperature: float
-unixTime: bigInt
-groundAtmosphericPreassure: float
-seaAtmosphericPreassure: float
-cloudiness: int
-visibility: float
-humidity: float
-p1Triangle: object
-p2Triangle: object
-p3Triangle: object
-bezierCurvePointA: object
-bezierCurvePointB: object
-bezierCurvePointC: object
+setup()
+draw()
+callAPI()
+assignUserVariables()
+findNewBezierTrajectory()
+checkCollision()
LineRotationAlgorithm
-feelsLike: float
-temperature: float
-unixTime: bigInt
-groundAtmosphericPreassure: float
-seaAtmosphericPreassure: float
-cloudiness: int
-visibility: float
-humidity: float
-color1: array int
-color2: array int
-initialLinePoint: object
-endLinePoint: object
-radius: float
-rotation: float
+setup()
+draw()
+callAPI()
+assignUserVariables()
+createGradientOnBar()
BreathingCircleAlgorithm
-feelsLike: float
-temperature: float
-unixTime: bigInt
-groundAtmosphericPreassure: float
-seaAtmosphericPreassure: float
-cloudiness: int
-visibility: float
-humidity: float
-smallRadius: float
-mediumRadius: float
-largeRadius: float
-color: array int
+setup()
+draw()
+callAPI()
+assignUserVariables()
+colotFromSize()
+createShapeOfCircles()
+translateSmallCircles()
+changeColorSmallCircles()

Patrons de conception

Les trois designs pattern utilisés sont le Template method, le Strategy et le DAO.

Template method

Pour implémenter ce patron de conception, nous utiliserons la structure de CommonAction vue dans le cadre du cours 420-C55-IN Web III. Cette structure nous permettra de partager aux classes enfants du CommonAction des fonctionnalités et des variables qui leur sont communes. Ainsi, nous allons pouvoir contrôler le niveau de visibilité des pages en utilisant les constantes définies dans le CommonAction. Ensuite, nous restreindrons les droits de l'utilisateur selon leurs permissions. Par exemple, un visiteur ne peut pas se diriger sur une page interne nécessitant une connexion même en entrant l'adresse URL.

Strategy

Nous utiliserons ce patron de conception pour limiter la duplication de code dans les différents algorithmes de génération d'animation. Seuls les éléments qui sont uniques à chaque algorithme devraient être codés en supplément au code commun. Nous coderons intégralement ce patron de conception.

DAO

Nous utiliserons le DAO afin de pouvoir séparer la logique des entités comme le UserDAO qui regroupe les informations et les fonctions propres à l'objet. Cela permettra entre autres de limiter les endroits dans le code où on aura accès à la base de données. Nous aurons surtout besoin du DAO pour afficher les GIF qui seront partagés par tous les utilisateurs, ainsi que les GIF qui seront télécharger sur le poste local de l'usager.

Structure de données

Les structures de données que nous utiliserons sont le array, le dictionnaire, la pile (stack) et la file (queue).

Array

L'array sera implémenté dans le contexte suivant : Depuis la base de données, nous récupérons les données de la table des animations qui seront assignées aux variables du constructeur d'un objet Carte. Chaque animation de la base de données sera représentée par un objet Carte en Javascript qui nous servira à afficher ultérieurement les animations partagées sur la page des partages. Pour s'y faire, chaque Carte sera ajoutée à une liste. Nous itérerons à travers cette liste pour afficher dans l'ordre de la base de données les GIF animés et les données textuelles accompagnant l'animation dans une balise DIV en passant par le DOM avec Javascript. Cette structure de données est pertinente car comparativement à une liste chaînée par exemple, il est plus efficace dans un array d'ajouter à la fin de la liste ce qui correspond à ce dont on a besoin dans le projet : ajouter des objets pour les transporter d'une page à l'autre. L'array a également une complexité algorithmique O(1) en accès aux données dû à l'index ce qui le rend extrêmement efficace pour utiliser les données qui se retrouvent à l 'intérieur.

Dictionnaire

Le dictionnaire sera présent à travers la fonction compact() de PHP. La fonction compact() utilise les arguments passés en paramètres comme clé avec laquelle sera associée les valeurs de la variable. On la retrouve dans les pages actions pour passer de l'information entre le backend et le frontend, ainsi que de pouvoir accéder aux propriétés des objets dans divers niveaux de profondeur dans le code. De plus, nous

utiliserons des variables de sessions afin de pouvoir garder des informations à propos de l'usager lors d'une connexion ou d'une visite au site web. Ces variables seront aussi stockées sous forme de dictionnaire comme dans cet exemple :

```
SESSION["Username"] = "Jessica123";
```

L'utilisation du dictionnaire est pertinente dans notre projet, parce que, dans les cas mentionnés plus haut, nous aurons besoin d'accéder à nos objets rapidement sans devoir itérer sur des listes. Les dictionnaires nous permettrons de récupérer les valeurs que nous avons besoin avec une clé correspondant au nom de la variable passé à la fonction compact par une complexité algorithmique O(1)

File (Queue) circulaire

Cette structure de données sera utilisée pour calculer la direction des courbes de Bézier du cercle à l'écran dans l'algorithme 3. Cet algorithme utilise une courbe de Bézier à trois points: un point initial, un point milieu et un point final.

Exemple :

```
Point initial = p1 // Point milieu = p2 // Point final = p3
```

Le comportement du cercle est que son point final devient son nouveau point initial et nous modifions le nouveau point milieu et le nouveau point final pour que le cercle puisse continuer son trajet. Donc, si l'on continue avec l'exemple précédent, une fois le premier trajet complété, p3 devient le point initial, nous modifions p2 et p1 afin d'éviter la création de nouveaux objets et nous itérons sur les 3 points en changeant leurs attributs pour permettre un mouvement fluide et transparent pour l'utilisateur. Cette structure de données est pertinente, car les points seront manipulés dans l'ordre premier arrivé premier sorti communément appelé *First In First Out* en anglais sauf qu'au lieu

d'effacer l'élément ou le sortir de la file, il restera en place et nous lui affections d'autres valeurs pour garder une performance optimale. Cette structure de données sera codée entièrement par nous et elle sera le plus modulaire possible afin qu'elle puisse itérer sur n éléments sans une perte de performance.

Pile (stack)

Une pile sera utilisée pour modifier la luminosité (en HSB) du triangle dans la troisième animation. La luminosité maximale du triangle va être déterminée par l'usager à l'aide d'une scroll bar. Une fois le maximum défini, la luminosité maximale sera divisée par quatre et elle oscille entre ces quatre fractions.

Par exemple, si la luminosité maximale est de 50, un stack avec [12.5, 25, 37.5,50] sera fait. Ainsi la brillance du triangle suivra cet ordre en premier et l'ordre inverse lorsqu'on enlèvera les éléments du stack. Cette structure de données est pertinente car nous suivons alors le mouvement du stack First In Last Out propre au Stack et elle crée l'effet cyclique désiré pour la brillance du triangle.

Algorithmes

D'abord, la date de naissance sera demandée lors de la création d'un compte et sera utilisée pour déterminer l'ensemble des variables qui seront utilisés dans chaque algorithme. Pour ce faire on utilisera le temps Unix positif, si la personne est née après le 1er janvier 1970, et négatif, si elle est née avant. Nous ajouterons aussi le chiffre calculé plus haut par l'expression régulière qui vérifie le courriel de l'usager et nous utiliserons des modulos sur le résultat pour déterminer quelle variable de chaque ensemble sera utilisée dans les algorithmes. Ainsi, même si deux personnes génèrent une animation au même moment dans la même ville, les animations différentes.

Légende des variables

Pour faciliter la compréhension, la variable choisie dans un algorithme sera une des celles listés ci-dessous en fonction de la date de naissance de l'usager. Chaque variable correspondra à une des options énumérées ci-dessous. Une fois que les variables sont déterminées pour un usager, elles sont inchangeables.

Variable 1 : température ressentie en matinée / pendant la journée / en soirée / pendant la nuit

Variable 2 : température en matinée / pendant la journée / en soirée / pendant la nuit

Variable 3 : temps Unix du lever du soleil / coucher de soleil / de minuit du même jour / et de midi du même jour

Variable 4 : Pression atmosphérique au niveau du sol en matinée / pendant la journée / en soirée / pendant la nuit

Variable 5 : Pression atmosphérique au niveau de la mer en matinée / pendant la journée / en soirée / pendant la nuit

Variable 6 : Nébulosité en matinée / pendant la journée / en soirée / pendant la nuit

Variable 7 : Visibilité moyenne en matinée / pendant la journée / en soirée / pendant la nuit

Variable 8 : humidité en matinée / pendant la journée / en soirée / pendant la nuit

Premier algorithme d'animation

La première animation consiste d'une petite ligne sur laquelle on affecte une première rotation. La vitesse de cette rotation sera déterminée en fonction de la variable 1. Cette ligne, tout en tournant sur elle-même, suivra un trajet d'ellipses. Les dimensions de

l'ellipse seront déterminées en fonction de la variable 2 et de la variable 3. Ensuite, nous allons lui faire effectuer une rotation autour d'un axe central. La vitesse de cette rotation sera déterminée par la variable 4 et la variable 5. Les deux extrémités de la petite barre seront de couleurs différentes et forment un effet de gradient au milieu de la forme. Ces deux couleurs seront déterminées par la variable 6 et la variable 7. La couleur principale de l'animation sera déterminée par la variable 8.

Deuxième algorithme d'animation

La deuxième animation consiste à créer une forme circulaire avec de petits cercles. Les cercles subiront une translation sur l'axe des x et des y en fonctions de la variable 1 et la variable 2. Le but serait d'émuler un mouvement fluide qui sera appliqué dans l'entièreté de la forme circulaire, comme une respiration ou un mouvement de nage. La taille des cercles à l'intérieur de la forme changera en fonction de la variable 3 et la variable 4. La vitesse de cette transformation sera calculée en fonction de la variable 5. Dans l'esprit de garder quelque chose de fluide, elle sera appliquée à une partie des cercles à l'intérieur de la forme circulaire et progressivement au restant. La couleur des cercles dépendra de leur taille. Les dimensions de l'ensemble contenant tous les petits cercles dépendent de la variable 6 et la variable 7. La couleur principale de l'animation sera déterminée par la variable 8.

Troisième algorithme d'animation

La troisième animation consiste d'un triangle qui se promène dans l'écran. La taille du triangle varie entre un certain intervalle et elle oscille à l'intérieur de ce dernier. Cet intervalle sera déterminé en fonction de la variable 1 et la variable 2. La variable 3 détermine la vitesse de cette oscillation. Une rotation axée sur le centre de la fenêtre de visualisation sera appliquée sur le triangle en fonction de la variable 4 et la variable 5.

Pendant cela, un cercle sera à l'écran. Ce dernier se déplace en effectuant des courbes de Bézier à trois points. Le point de début et le point de fin vont être dans les extrêmes de la fenêtre de visualisation pour donner l'impression de rebondir sur les contours. La position du point intermédiaire sera déterminée par la variable 6 et la variable 7. Chaque fois que le cercle touchera une extrémité du cadre de visualisation, une nouvelle courbe sera calculée pour garder un mouvement fluide et pour faire en sorte que le cercle se promène partout dans l'écran. Dans le cas d'une collision entre le cercle et le triangle, les deux entités rebondissent dans des directions opposées. La couleur principale de l'animation sera déterminée par la variable 8.

Mathématiques

Équation de la courbe de Bézier à trois points dans laquelle « t » représente la position du point dans la courbe 0 étant le début et 1 étant la fin :

$$x = (1-t)^2x_1 + 2(1-t)tx_2 + t^2x_3$$

$$y = (1-t)^2y_1 + 2(1-t)ty_2 + t^2y_3$$

Représentation de l'équation sous forme de code :

```
pointInital.x = Math.pow((1-t),2) * p1.x + 2 * (1-t) * t * p2.x + Math.pow(t,2) * p3.x

pointInital.y = Math.pow((1-t),2) * p1.y + 2 * (1-t) * t * p2.y + Math.pow(t,2) * p3.y
```

Une bonne partie des rotations circulaires qui ne sont pas axée sur le référentiel seront effectués avec les fonctions trigonométriques sin et cos. Le rayon de cette rotation circulaire sera calculé avec les mêmes fonctions auxquelles on ajoutera 1 car de base ces fonctions retournent des valeurs entre -1 et 1 et si on ajoute 1 aux extrêmes elles retourneront entre 0 et 2. Ensuite on divise par 2 et on obtient un intervalle de 0 à 1.

Ensuite on peut le multiplier par n et on obtient un intervalle entre 0 et n. Ainsi on peut paramétrer le rayon de la rotation circulaire. Dans l'exemple suivant, n = 250.

```
let radius  = (sin(frameCount / 100 * TWO_PI) +1 ) / 2 * 250
```

Intégration avec le cours de veille technologique

La technologie choisie dans le cadre du cours 420-C66-IN est React. Cette librairie Javascript est un atout important pour ceux qui souhaitent suivre un cheminement en développement Web, renforçant ainsi notre motivation de l'apprendre et de l'intégrer au projet. De plus, il est pertinent d'utiliser React, car la librairie permettra entre autres de rendre le temps de rafraîchissement de notre site web plus rapide en utilisant un DOM virtuel. Celui-ci ne rafraîchit que les éléments de la page qui sont changeants plutôt que de réafficher tout le contenu de la page. Plus encore, React permettra de créer du code réutilisable, ce qui nous permettra de créer plus de modularité dans le développement Frontend de notre projet.

MÉDIAGRAPHIE

1. Christian Heilmann. (4 oct. 2011). *A Quick Look Into The Math Of Animations With JavaScript*. SmashingMagazine. https://www.smashingmagazine.com/2011/10/quick-look-math-animations-javascript/
2. Varun Vachhar, Coding Tech. (8 déc. 2018). *Mathematics of Animation*. YouTube. https://www.youtube.com/watch?v=KbxJWx-ue0U
3. Chris Gannon. (9 mars 2016). *How To Create an Interactive SVG Splat Animation*. ChrisGannon. https://chrisgannon.wordpress.com/category/javascript/
4. Matt Rossman. (17 déc. 2020). *Recreating a Dave Whyte Animation in React-Three-Fiber*. Codrops. https://tympanus.net/codrops/2020/12/17/recreating-a-dave-whyte-animation-in-react-three-fiber/
5. Dave Whyte. (2019). *gifs by dave >:)*. Bees & Bombs. https://beesandbombs.tumblr.com/
6. German Cocca. (16 mai 2022). *Data Structures in Javascript - With Code Examples*. freeCodeCamp. https://www.freecodecamp.org/news/data-structures-in-javascript-with-examples/
7. The Modern Javascript Tutorial. (26 juin 2022). *Bezier curve*. Javascript.info. https://javascript.info/bezier-curve
8. React. (2022). *React - A Javascript library for building user interfaces*. React. https://reactjs.org/.
9. The PHP Group. (2022). *Compact*. PHP. https://www.php.net/manual/en/function.compact.php.