TPI - Chronoview

Documentation Technique

Zoé Cugni

19/06/2017

1. Table des matières

2. Introduction

2.1. Nature du projet

Chronoview est un site web permettant de visualiser des événements ayant eu lieu pendant une période déterminée. Ceux-ci sont positionnés (sous forme de points) sur des lignes chronologiques représentants différentes catégories d'événements.

2.2. Motivations

J'ai choisi de faire ce projet car je trouve compliqué de comparer ce qu'il s'est passé à une même époque entre différents domaines ou pays. Nous avons souvent tendance à étudier l'histoire d'un lieu sans la mettre en relation avec celle d'autres endroits, à part bien sûr si ceux-ci sont concernés.

Le but de ce projet est donc de répondre à ce manque et de permettre, en mettant à disposition une interface pratique, de facilement visualiser ce qui se passait par exemple en Asie pendant la Guerres de Cents, qui étaient les contemporains de Martin Luther King ou quelles grandes découvertes dans divers domaines ont été faites en même temps.

2.3. Analyse concurrentielle

Pour réaliser ce site, je me suis en partie inspirée d' Histography, un site présentant tous les événements de Wikipedia sous forme de points alignés les uns à côtés des autres de façon à former une ligne chronologique. Quand nous passons la souris sur ces points, une boite d'information apparaît et nous affiche le contenu de la page Wikipedia correspondante.

Ce site est impressionnant, mais il ne permet pas de comparer de nombreuses catégories entre elles, uniquement deux. De plus, les données étant prise de Wikipedia, nous ne pouvons pas rajouter nos propres événements avec des descriptions personnalisées.

J'ai donc repris un certain nombre d'idées de ce site que j'ai modifiés à ma sauce et j'ai rajouté les miennes pour arriver à un résultat similaires mais avec des fonctionnalités et un design différents.

3. Analyse fonctionnelle

3.1. Description globale des fonctionnalités

3.1.1. Pour un utilisateur

- Sélectionner une ou plusieurs catégories à visualiser (jusqu'à 8).
- Déplacer les barres du slider pour agrandir ou rétrécir la période à afficher.
- Visualiser les événements sous forme de points représentés sur une ligne chronologique en fonction des catégories et de la période déterminées.
- Passer sa souris sur les points et rester un petit moment dessus pour voir apparaître :

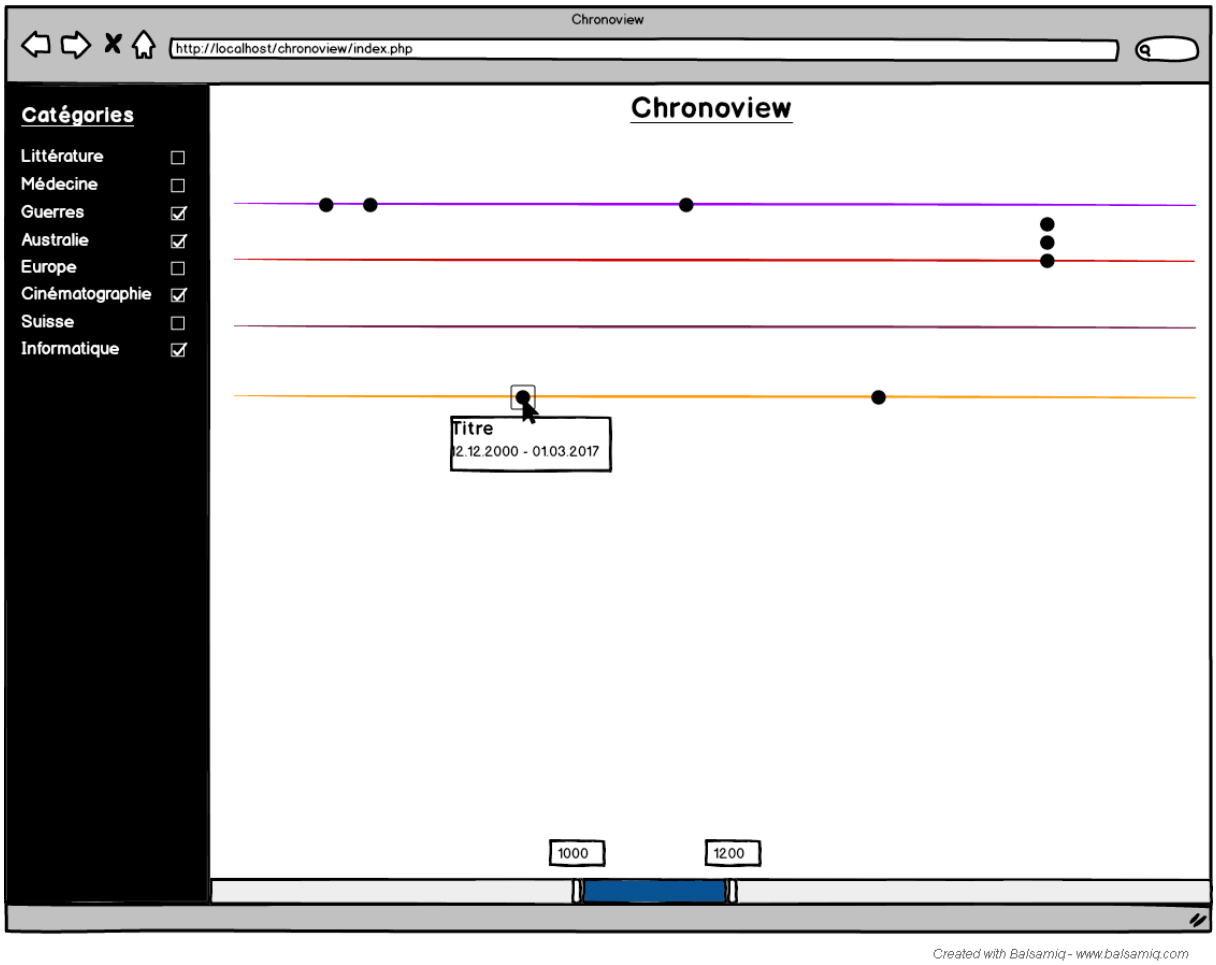
  - Une petite boite affichant le titre et les dates (début/fin) de l'événement.
  - Un second point et une ligne reliant les deux représentants la période pendant laquelle s'est déroulé l'événement (si elle est assez grande).
- Cliquer sur un point pour voir la totalité de ces informations affichée dans une grande boite (titre, date, description, image).

3.1.2. Pour un administrateur

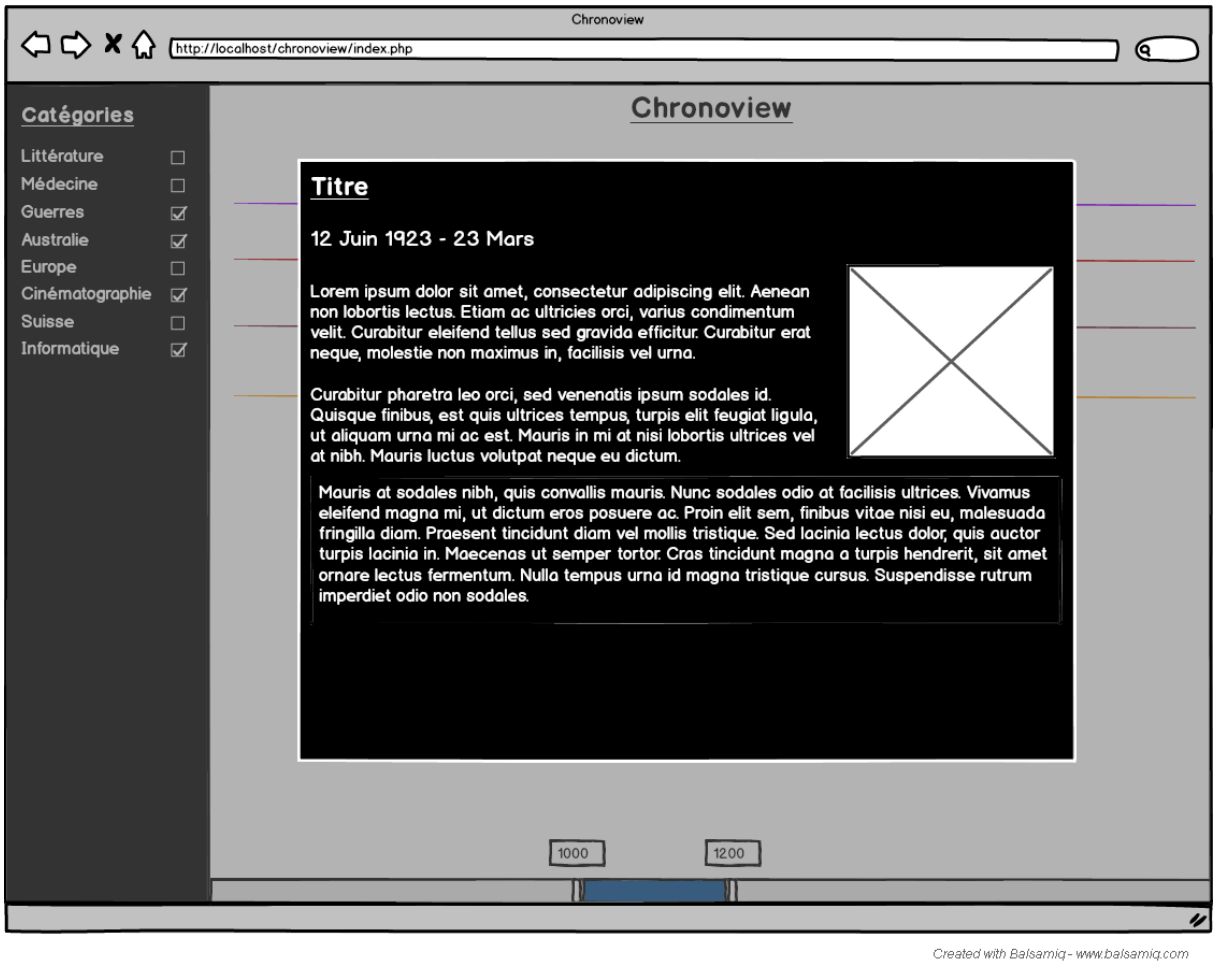
- Tout ce que peut faire l'utilisateur.
- Ajouter des catégories d'événements.
- Modifier/supprimer des catégories existantes.
- Ajouter des événements.
- Modifier/supprimer des événements existants.
- Relier des catégories à des événements.

3.2. Interface (maquette)

- Vue utilisateur avec un point en hover :



- Vue utilisateur avec lightbox pour un élément :



4. Analyse organique

4.1. Technologies / Langages utilisés

- **Base de données :** Mysql
- **Langage côté serveur :** PHP
- **Langages côté client :**
  - Javascript, ainsi que ces librairies :
    - Jquery : pour faciliter plusieurs actions
    - NoUiSlider : permet de créer le slider du bas
    - WNumb : pour formater les nombres du slider
    - Color-animation : pour faire une animation changeant la couleur
  - HTML 5
  - Styles :
    - CSS 3
    - Sass (pour faciliter l'écriture des styles, compilés avec koala)

4.2. Base de données

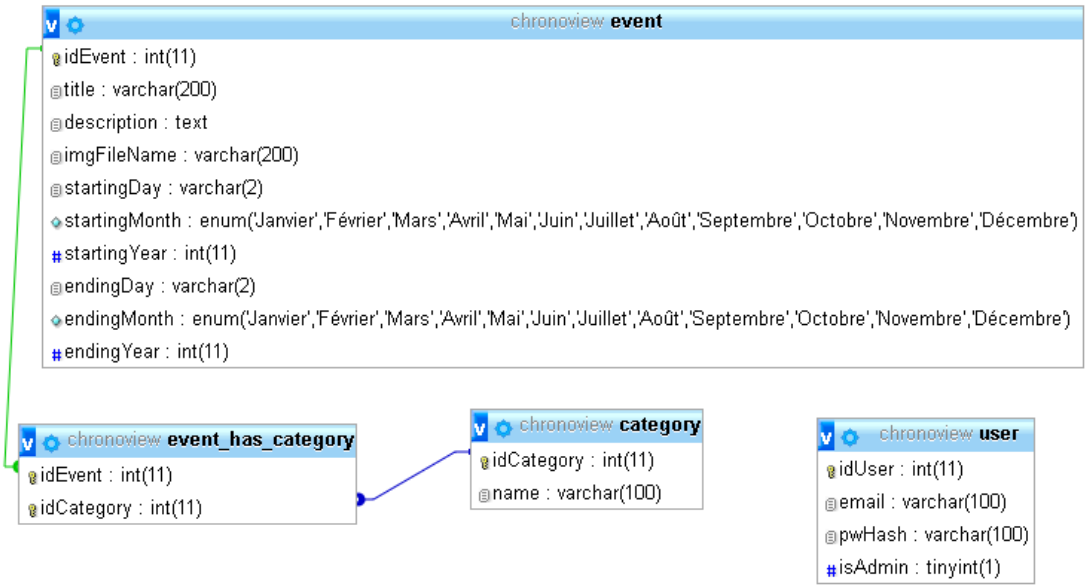


Figure 1 - Modèle de la base de données

Je vais revenir points par points sur chacune de ces tables.

4.2.1. Table Event

Cette table contient tous les événements. Ils ont chacun une date de début (*starting day/month/year*) et une de fin (*ending day/month/year*), si l'événement n'a duré qu'un jour, elles seront identiques.

Les dates sont séparées en jour, mois, année car il arrive souvent que nous ignorons le jour exact d'un événement, ou même son mois. Seule l'année est donc obligatoire, et c'est elle qui sera utilisée pour placer les points.

Seul *idEvent* (auto-incrémenté), *title*, *description* et les années sont obligatoires, tout le reste peut être *null*.

Les mois sont de types *enum*, seule une valeur contenue dans leur énumération est donc acceptée. Cela permet de s'assurer que les noms des mois seront toujours écrits de la même façon. Une autre option aurait été d'utiliser un nombre, mais je trouve qu'un mois écrit en toutes lettres rends la date plus agréable à lire.

4.2.2. Table Category

Cette table contient toutes les catégories d'événements, elles ont chacune un nom et une couleur en hexadécimal.

4.2.3. Table Event_has_category

Table de liaison entre les tables *Event* et *Category*, elle ne contient donc que leur deux clés primaires.

4.2.4. Table User

Table contenant toutes les informations permettant aux utilisateurs de se connecter. Pour l'instant, cette table n'est utilisée que pour des administrateurs mais elle peut être facilement modifiée si les fonctionnalités du site se complexifient.

Les mots de passe sont conservés sous leur forme hachée.

4.2.5. Fonction de connexion

Nom : MyPdo()

Emplacement : functions/dao.php (1ère fonction)

But : Crée et retourne un objet PDO (pouvant être utilisé dans les requêtes mySQL) en suivant le parton de conception du singleton.

Patron de conception singleton : L'idée est de créer une instance de classe s'il n'en existe pas encore, sinon de renvoyer une référence sur l'instance existante.

Retour : un objet PDO

Implémentation avec la combinaison php / mysql :

- Déclarer le pdo en static
- Si le pdo est null essayé de créer une nouvelle instance
- Penser à rendre la connexion persistante pour qu'elle non plus ne soit pas tout le temps ré-établie
- En cas d'échec récupérer l'erreur, tuer l'application et l'afficher
- Retourner l'objet pdp

4.2.6. Requêtes :

Les requêtes que j'ai utilisées sont toutes basiques. Pour chacune d'elle, j'ai mis en commentaire leur description, paramètres et variable de retour dans le code. Je ne les détaillerais donc pas à nouveau ici car elles ne sont pas très intéressantes.

4.3. Structure des fichiers

Mon code est répartit en plusieurs fichiers et dossiers que je vais détailler ici :

- *js :* contient tous les fichiers javascripts de l'application.
  - *noUiSlider.js :* libraire externe permettant d'utiliser le slider du bas de la page.
  - *wNumb.js :* librairie externe utilisée par noUiSlider pour formater les nombres affichés en-dessus des deux poignées du slider.
  - *Timeline.js :* les fonctions permettant de récupérer les événements, d'afficher les lignes et de gérer le slider.
  - *Lightbox.js :* les fonctions gérant les 3 lightbox (affichage, ajout, modification) ainsi que l'affichage de la durée et de la petite boite de résumé.
- *functions :* contient toutes les fonctions php du site

    - *dao.php :* toutes les fonctions d'accès (requête) à la base de données.
    - *dbCoConfig.php :* constantes de connexion à la base de données.
    - *interlocutor :* tous les fichiers appelés par d'autres fichiers, ils font souvent le lien entre le serveur et le client.
        - *event.php :* renvoi les événements pour une période et une catégorie donnée ainsi que l'id et la couleur de ladite catégorie.
        - *rangeYear.php :* renvoi la période du premier élément de la base au dernier.
        - *getSession.*php : renvoi un booléen indiquant si l'administrateur est connecté.
        - *insertEvent.*php : ajoute un événement dans la base.
        - *loginCheck.*php : vérifie que les informations données lors de la connexion soient correcte.
        - *updateEvent.*php : mets à jour un événement.
        - *deleteCategory.*php : supprime une catégorie
        - *deleteEvent.*php : supprime un événement
        - *getCategories.*php : récupère les catégories
        - *insertCategory.*php : insère une catégorie
        - *insertEvent.*php : insère un événement
- *database :* contient les scripts de création de la base (un pour la structure, un pour les données).
- *Css :* Contient toutes les feuilles de style du site.
    - *Nouislider.css :* style par défaut de nouislider.
    - *style.scss :* Fichier Sass contenant tous les styles du site.
    - *style.css :* Fichier css généré par Sass.
    - *style.css.map :* Fichier reliant le fichier Sass et son fichier CSS, généré par Sass.
- *Img :* contient toutes les images du site, y compris celles misent en ligne par les utilisateurs.
- *Doc :*
    - Documentation technique
    - Manuel utilisateur
    - Diagramme d'activités : les diagrammes utilisés dans cette documentation en format xml et png
    - Maquettes : Les maquettes utilisées dans cette documentation en format bmpr et png
    - Screenshot : les screenshots utilisés dans le manuel utilisateur
- *Index.php* : la seule page du site, définis la structure de base de la page et deux des trois lightbox (celle d'ajout et de modification), qui sont cachées au lancement. Le reste du contenu est généré en javascript.

4.4. Diagramme d'activités globale du programme

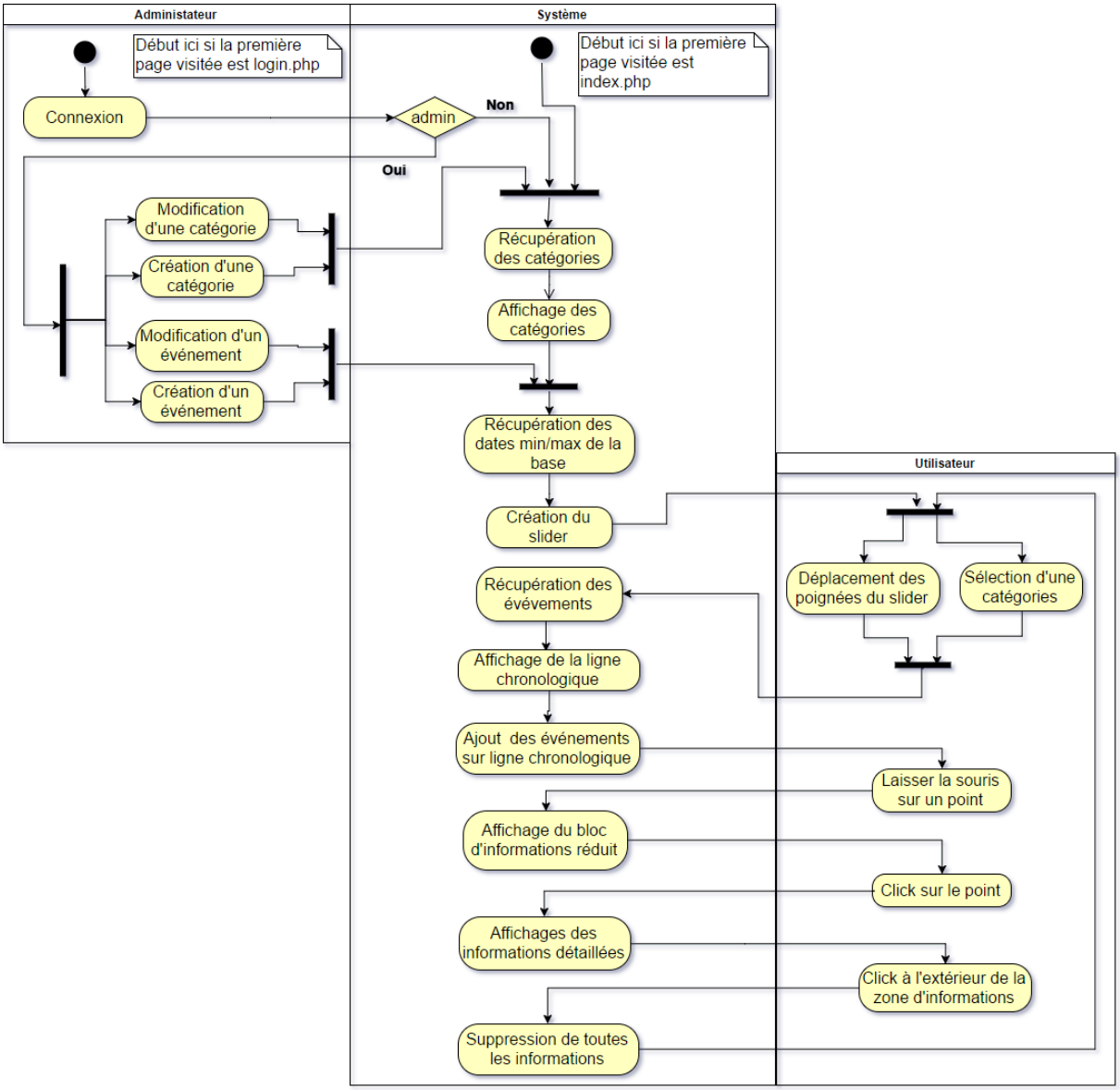


Figure 2 Diagramme d'activité globale

Ce diagramme d'activités représente le fonctionnement général du programme et ne détaille aucune des fonctions. Je reviendrais sur chaque point individuellement dans la suite de ce document.

Il me faut tout de même préciser qu'à tout moment un utilisateur peut décider de se connecter, auquel cas le processus retournera au stade de connexion. De même, un administrateur peut toujours se comporter comme un utilisateur.

4.5. Création du Slider

4.5.1. Terme utilisé

N'ayant pas réussi à trouver de terme équivalent français satisfaisant, j'utilise le terme anglais. Le slider est la barre horizontale en bas de la page permettant de sélectionner la période que les lignes chronologiques auront à représenter. La totalité

du slider représente quant à elle la période entre le plus vieil et le plus récent événement contenu dans la base de données.

Les deux petites barres verticales déplaçables sont elles appelées « poignées ».

4.5.2. Fonction addSlider()

La création du slider se fait dans la fonction addSlider car il doit parfois être mis à jour. Cette fonction traite les données reçues et appelle ensuite la fonction de noUiSlider, décrite plus bas.

4.5.3. NoUiSlider

Pour réaliser ce slider, j'ai utilisé une librairie externe, noUiSlider. J'aurais aussi pu utiliser le slider de Jquery Ui, mais l'avantage de celui-ci est qu'il est beaucoup plus léger et je l'avais déjà utilisé par le passé, il m'a donc semblé être un bon choix.

Pour l'utiliser, mis à part mettre des liens vers ces fichiers, il faut aussi l'appeler sur des balises « » avec les options de notre choix. J'utilise pour ma part celles-ci :

- Range { 'min' : nombre, 'max' : nombre } - Les valeurs minimums et maximums représentées par le slider.
- Start [nombre, nombre] - L'emplacement au chargement du slider des deux poignées.
- Connect : true – Mets dans une autre couleur la zone entre les deux poignées.
- Tooltips [formatage, formatage] - Affiche les nombres en dessus des deux poignées dans le format demandé.
- Behaviour : 'none' - Le slider propose par défaut plusieurs possibilités pour déplacer les poignées (cliquer devant, déplacer les deux en même en gardant l'écart, etc.) mais je préfère conserver uniquement la plus simple : le déplacement d'une poignée à la fois.

4.5.4. Formatage - WNumb

Le formatage des nombres ce fait lui avec une autre librairie externe, wNumb (c'est ce que conseil les développeurs de noUiSlider).

Pour ma part, je veux juste cacher les nombres après la virgule. Pour cela, le format ce définis simplement ainsi : wNumb({ decimals: 0 }).

4.5.5. Fonction - $(document).ready()

Le slider est créé dès que le document est prêt (dans la fonction ready() du document, qui est gérée dans js/timeline.js).

Celle-ci fait d'abord un appel ajax pour récupérer les années minimum et maximum stockées dans la base. L'appel est synchrone car la page ne peut pas fonctionner sans le slider.

Si l'appel a réussi, elle appelle la fonction addSlider qui appelle la fonction.create() de noUiSlider avec les bonnes options. Pour les positions initiales des poignées, elle les mets à la moitié du slider +/- 10% de sa taille.

Après cela, elle calcule les variables globales sliderValues et nbPixelPerYear. Ces deux variables sont utilisées dans différentes fonctions mais de dépendent que du positionnement des poignées du slider, raison pour laquelle j'évite de les recalculer à chaque fois. SliderValues est un tableau contenant la valeur représentée par chacune des poignées. NbPixelPerYear détermine combien de pixel représente une seule année, elle est calculée en divisant la largeur de la zone accordée aux lignes chronologique par le nombre d'années que ces lignes sont censée représentées.

Elle attache ensuite une fonction à l'événement du slider détectant quand l'utilisateur a fini de déplacer une poignée. Je reviens sur cette fonction au point suivant.

Finalement, bien que cela ne soit pas lié au slider, elle s'occupe aussi d'attacher un certain nombre de fonction aux événements de différents éléments de la page. Tels que les mouseenter/mouseleave de la petite boite d'information des événements et de la liste des catégories, le submit du formulaire insérant des événements et le click cachant les lignbox. Je reviendrais sur tous ceux-ci au moment venu.

Elle vérifie aussi si l'administrateur est connecté et ajoute le bouton de création d'événement si c'est le cas.

Certaines de ces fonctions sont attachées de façon déléguées car les éléments concernés n'existent pas encore et ne sont rajouter à la page que plus tard.

4.5.6. Evénement du slider

NoUiSlider met différents événements à disposition pour son slider. Comme dit précédemment, c'est celui qui détecte la fin d'un déplacement de poignée, « end », que j'utilise.

Quand il se produit, je vérifie d'abord si des catégories sont sélectionnées. Si tel est le cas, je vide la zone des lignes chronologiques et j'appelle les fonctions récupérant les événements concernés et affichant les lignes chronologiques. Ces fonctions sont décrites plus bas.

De plus, je vérifie aussi si l'administrateur est connecté, si tel est le cas, j'affiche le signe « + » permettant d'ajouter un événement.

4.6. Liste des catégories

4.6.1. Création (fonction displayCategories())

La liste des catégories est elle-aussi créée en javascript afin de pouvoir la mettre à jour sans recharger la page au moment venu.

La fonction displayCategories est appelée à l'événement ready() du document ainsi que quand une catégorie est modifiée, ajoutée ou supprimée.

Celle-ci fait un appel ajax récupérant les catégories, puis, s'il a été fructueux, vide les 3 zones de catégories et les remplies à nouveau avec les données mise à jour. (Les trois zones étant les lightbox de modification/ajout ainsi que la barre latérale).

Si l'administrateur est connecté, elle ajoute aussi après chaque élément les symboles de modification et de suppression. De plus, elle met au début de la liste un élément intitulé 'Nouvelle catégorie' qui permet l'ajout d'une catégorie quand il sera cliqué, mais je reviendrais là-dessus plus tard.

4.6.2. Sélection (selectCategory)

J'avais initialement prévu d'utiliser un « select multiple » pour afficher la liste mais il s'est avéré qu'il n'est pas possible de modifier l'affichage des options comme bon nous semble. Je me suis donc rabattue sur des éléments de liste, pour lesquels je simule le même comportement qu'un « select multiple ».

Ainsi, lors d'un click, la fonction selectCategory vérifie s'il s'agit d'une sélection ou désélection. Dans le premier cas, elle ajoute l'id de la catégorie à un tableau (checkedCatIdArray), appelle la fonction affichant la ligne chronologique de cette catégorie et change le style.

Dans le second cas, elle supprime l'id du tableau et remet l'option à son style initiale. Pour cela, étant donné que mes style de sélection et de hover sont les même, il me faut momentanément surcharger le style du hover pour qu'il ne fasse rien. En effet, la souris étant sur un élément quand on le désélectionne, sans surcharge, nous ne verrions pas de différence. Cette surcharge est annulée dès que la souris quitte cette option.

4.6.3. Apparition/disparition

La liste n'apparait que quand on passe la souris sur son titre afin de ne pas surcharger la page d'information. Tous ces petits événements font partis de ceux attachés lors de la fonction ready() du document.

4.7. Lines chronologiques

4.7.1. Deux fonctions

L'affichage des lignes chronologiques est divisé en deux fonctions, getEvent() qui récupère les bons événements et displayTimeLine() qui les affiches sur des lignes chronologiques, ces deux fonctions se trouvent dans js/timeline.js

4.7.2. GetEvent(idCategory)

GetEvent(idCategory) récupère d'abord les valeurs indiquées par le slider puis fait un appel ajax demandant la liste des événements (avec leurs informations, dont toutes les catégories auxquelles ils sont liées) pour cette période et la catégorie donnée en paramètre.

La page répondant à l'appel ajax ajoute aussi au début du tableau des événements reçus l'id et la couleur de la catégorie choisie.

Si elle reçoit une réponse, cette fonction envoi ces données à displayTimeline.

L'appel se fait de façon synchrone pour le cas où plusieurs catégories sont sélectionnées et qu'une poignée du slider est déplacée. En effet, cette fonction serait ainsi appelée plusieurs fois de suite pour chaque catégorie et si l'appel ajax n'est pas synchrone elles ne seront sûrement pas remises dans le même ordre sur la page.

4.7.3. DisplayTimeline(eventData)

J'avais initialement prévu de laisser l'utilisateur choisir la couleur des lignes chronologiques mais j'ai finalement changé d'avis car en ayant le contrôle je peux m'assurer que les lignes/points restent lisibles et que le tout va plus ou moins bien ensemble.

Le reste est décrit dans le diagramme d'activités ci-dessous :

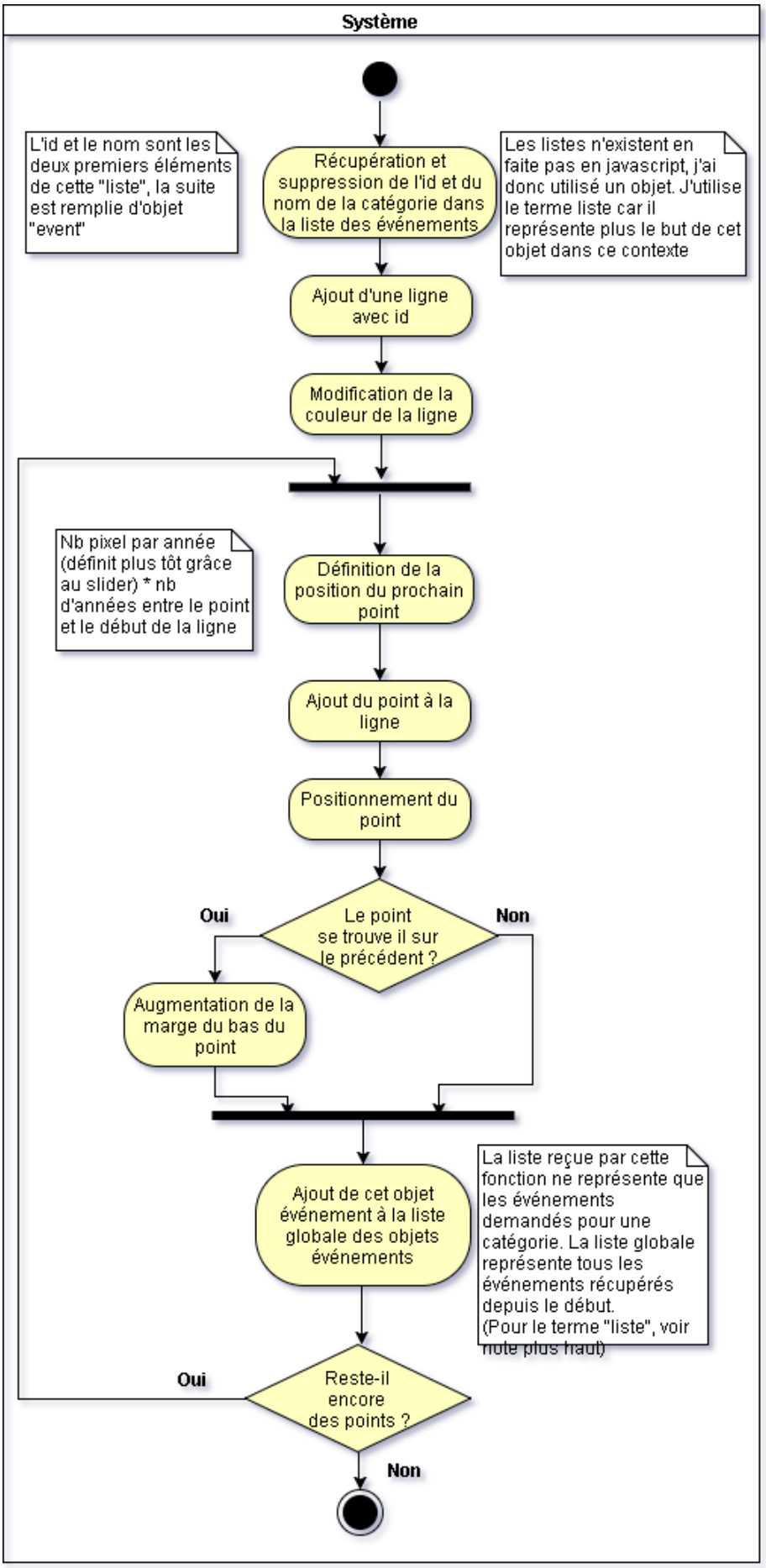

Système
L'id et le nom sont les deux premiers éléments de cette "liste", la suite est remplie d'objet "event"
Récupération et suppression de l'id et du nom de la catégorie dans la liste des événements
Les listes n'existent en faite pas en javascript, j'ai donc utilisé un objet. J'utilise le terme liste car il représente plus le but de cet objet dans ce contexte
Ajout d'une ligne avec id
Modification de la couleur de la ligne
Nb pixel par année (définit plus tôt grâce au slider) * nb d'années entre le point et le début de la ligne
Définition de la position du prochain point
Ajout du point à la ligne
Positionnement du point
Oui
Le point se trouve il sur le précédent ?
Non
Augmentation de la marge du bas du point
Ajout de cet objet événement à la liste globale des objets événements
La liste reçue par cette fonction ne représente que les événements demandés pour une catégorie. La liste globale représente tous les événements récupérés depuis le début.
(Pour le terme "liste", voir note plus haut)
Oui
Reste-il encore des points ?
Non

Figure 3 Diagramme d'activité displayTimeline()

4.8. Informations sur les événements

4.8.1. Trois types d'informations

Je donne des informations sur les événements que représentent les points de trois manières différentes :

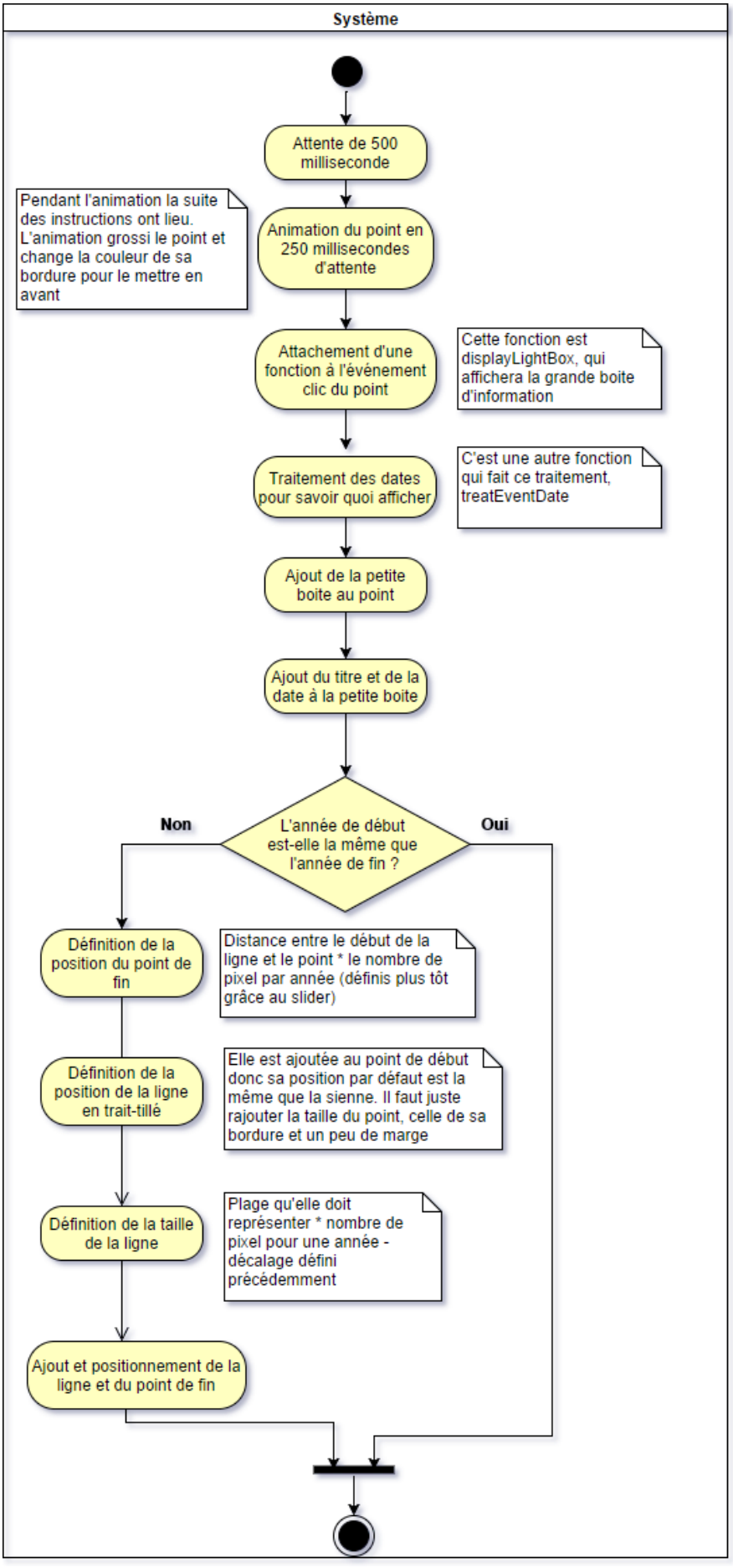
- Une mini boite d'informations donnant juste le nom de l'événement et sa date
- Une ligne en traits-tillés et un second point pour montrer où s'arrête un événement ayant cours sur plusieurs années.
- Une grande boite d'informations affichant le titre, les dates, la description et une image s'il y en a une.

Les deux premiers sont affichés ensemble, le dernier seul.

4.8.2. Mini boite & Durée (fonction displayMiniInfo()):

Cette fonction est attachée à l'événement mouseenter des points dans la fonction ready() du document (comme mentionné précédemment).

Cette fonction lance une animation, affiche la durée et la petite boite d'information. Elle fonctionne ainsi :


Système
Attente de 500 milliseconde
Pendant l'animation la suite des instructions ont lieu. L'animation grossi le point et change la couleur de sa bordure pour le mettre en avant
Animation du point en 250 millisecondes d'attente
Attachement d'une fonction à l'événement clic du point
Cette fonction est displayLightBox, qui affichera la grande boite d'information
Traitement des dates pour savoir quoi afficher
C'est une autre fonction qui fait ce traitement, treatEventDate
Ajout de la petite boite au point
Ajout du titre et de la date à la petite boite
L'année de début est-elle la même que l'année de fin ?
Non
Oui
Définition de la position du point de fin
Distance entre le début de la ligne et le point * le nombre de pixel par année (définis plus tôt grâce au slider)
Définition de la position de la ligne en trait-tillé
Elle est ajoutée au point de début donc sa position par défaut est la même que la sienne. Il faut juste rajouter la taille du point, celle de sa bordure et un peu de marge
Définition de la taille de la ligne
Plage qu'elle doit représenter * nombre de pixel pour une année - décalage défini précédemment
Ajout et positionnement de la ligne et du point de fin

4.8.3. Lightbox (fonction displayLightbox())

Si l'utilisateur clic sur le point une fois que la petite boîte d'informations est affichée, une lightbox apparaîtra avec les informations détaillées de l'événement (titre, dates, description, image).

Comme pour le slider, j'utilise le terme anglais pour parler de la lightbox, faute de bons équivalents. Ce terme, pour un site, fait référence au processus suivant (avec pour exemple une image) :

Une image de taille moyenne est affichée parmi d'autres sur un site. Quand l'utilisateur clic dessus, tout le reste de la page est assombri, l'image est agrandie et mise, centrée, au premier plan. Cela permet de la mettre en avant.

Quand l'utilisateur clic ailleurs ou sur une croix, l'image reprends sa taille et sa position initiale et le reste de la page reprends sa couleur initiale.

Je fais de même avec la fonction displayLightbox. La seule subtilité est la suivante : il n'est pas possible, du moins pas simplement, de mettre une couleur d'arrière-plan avec une opacité moindre en css. Je rajoute donc d'abord une « » de la bonne couleur et opacité puis je rajoute par-dessus la lightbox.

Quand l'utilisateur clic en dehors de la lightbox, celle-ci disparait.

4.8.4. Mouseleave (removeMiniInfo)

Une autre fonction attachée à un événement pendant la fonction ready() du document est removeMiniInfo(). Celle-ci s'enclenche quand la souris quitte le point. Elle annule le timeout (le temps d'attente avant le lancement de la fonction displayMiniInfo) si celle-ci n'est pas encore lancée.

Si elle l'a été, cette fonction supprime la petite boite ainsi que le point de fin et la seconde ligne (s'ils sont présents) et anime le point dans l'autre sens (vers son état originel).

4.8.5. Traitement des dates (treatEventDate)

Les deux fonctions affichant les petites et grandes boites appellent la fonction treatEventDate pour traiter les dates.

Celle-ci reçoit les jours/mois/années de la date de début et de la date de fin. Elle vérifie ensuite si les jours et mois ont bel et bien une valeur.

Puis elle forme une chaine de caractère pour les deux dates ainsi : Jour Mois Année. Si un élément est manquant, elle ne le mets pas dans la chaine.

En revoit un tableau contenant les deux chaines.

4.9. Modification d'un événement

4.9.1. Solutions envisagées

J'aurais pu aborder la modification de 2 façons, soit en faisant une page à part affichant la liste de tous les événements ainsi qu'un lien emmenant vers un formulaire permettant de les modifier. Soit en intégrant la modification dans l'affichage en le transformant en un formulaire pré-rempli.

J'ai choisi la seconde option car le nombre d'événements existants sera sûrement très grand après une utilisation réelle du site. Il ne me semblait donc pas très pratique de devoir les trier pour pouvoir modifier le bon. Une option de recherche aurait bien sûr aidé, mais ne me semblait pas régler entièrement le problème.

L'autre raison de ce choix et que mis à part pour la connexion, que je n'ai pas réussi à joliment intégrer au site, tout se passe sur une seule et même page. Il me semblait donc étrange d'en rajouter d'autre alors qu'il était possible de continuer dans cette lignée.

L'autre choix que j'ai dû faire concernait la façon d'abordé cette solution. Je pouvais soit conserver la même lightbox et la modifier directement, soit afficher une lightbox préalablement construire la remplir des bonnes informations au moment venu.

J'avais commencé par suivre la première méthodologie, car elle me semblait plus logique, mais remplir toute une page de formulaire en javascript devient vite laborieux. Ce n'est pas forcément compliqué, mais beaucoup de lignes sont nécessaire pour pas grand-chose et je trouvais que cela rendais mon code moins lisible.

J'ai donc décidé d'utiliser l'autre option, et de remplir puis afficher une lightbox préconstruite au moment voulu.

4.9.2. Modification (startUpdate())

Si l'administrateur est connecté, un bouton « Modifié » apparaît sur la lightbox. Quand il est cliqué, celui-ci appelle la fonction startUpdate.

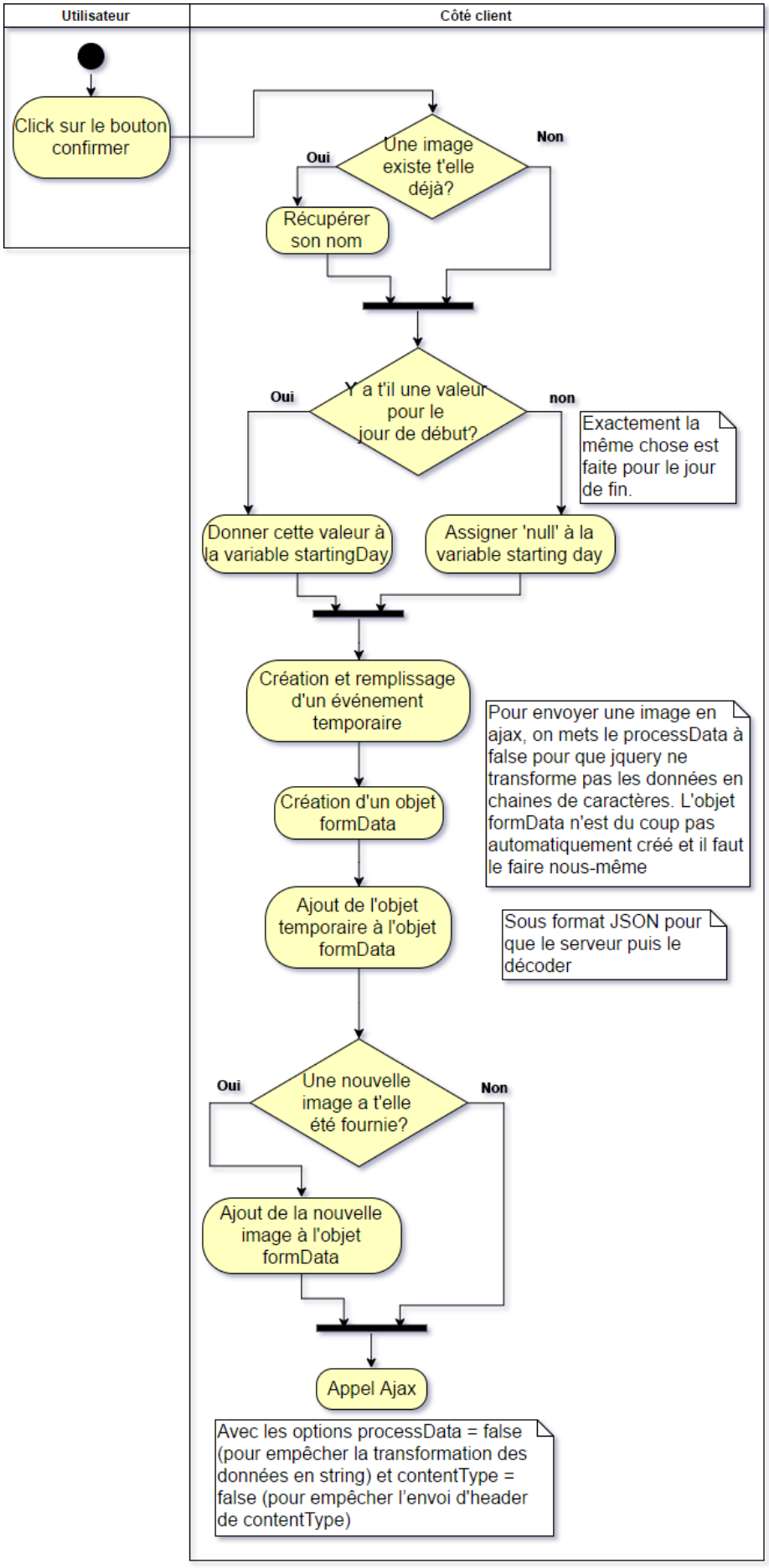
Elle récupère l'id de l'évément détaillé depuis la classe de la lightbox, cache cellec-i, affiche celle de modification et la remplie en récupérant les informations de cet événement stockées dans la liste globale remplie précédemment.

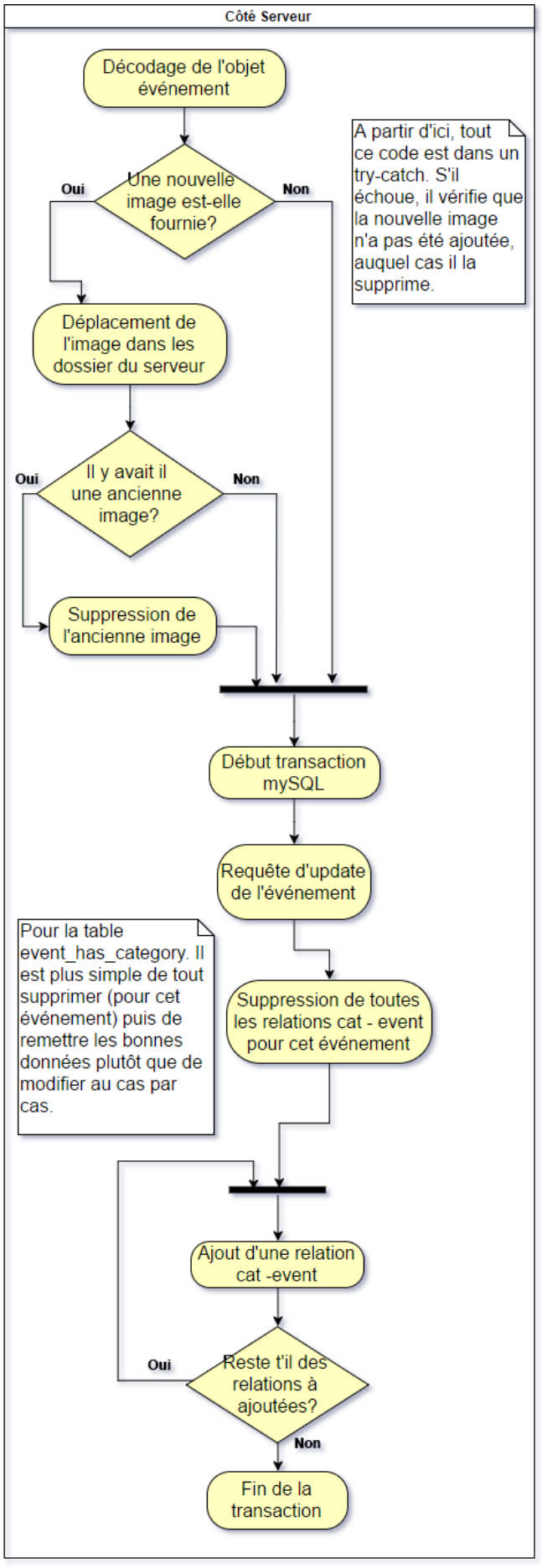
Elle remplace aussi le bouton « modifier » par deux autres, « Confirmer » et « Annuler ».

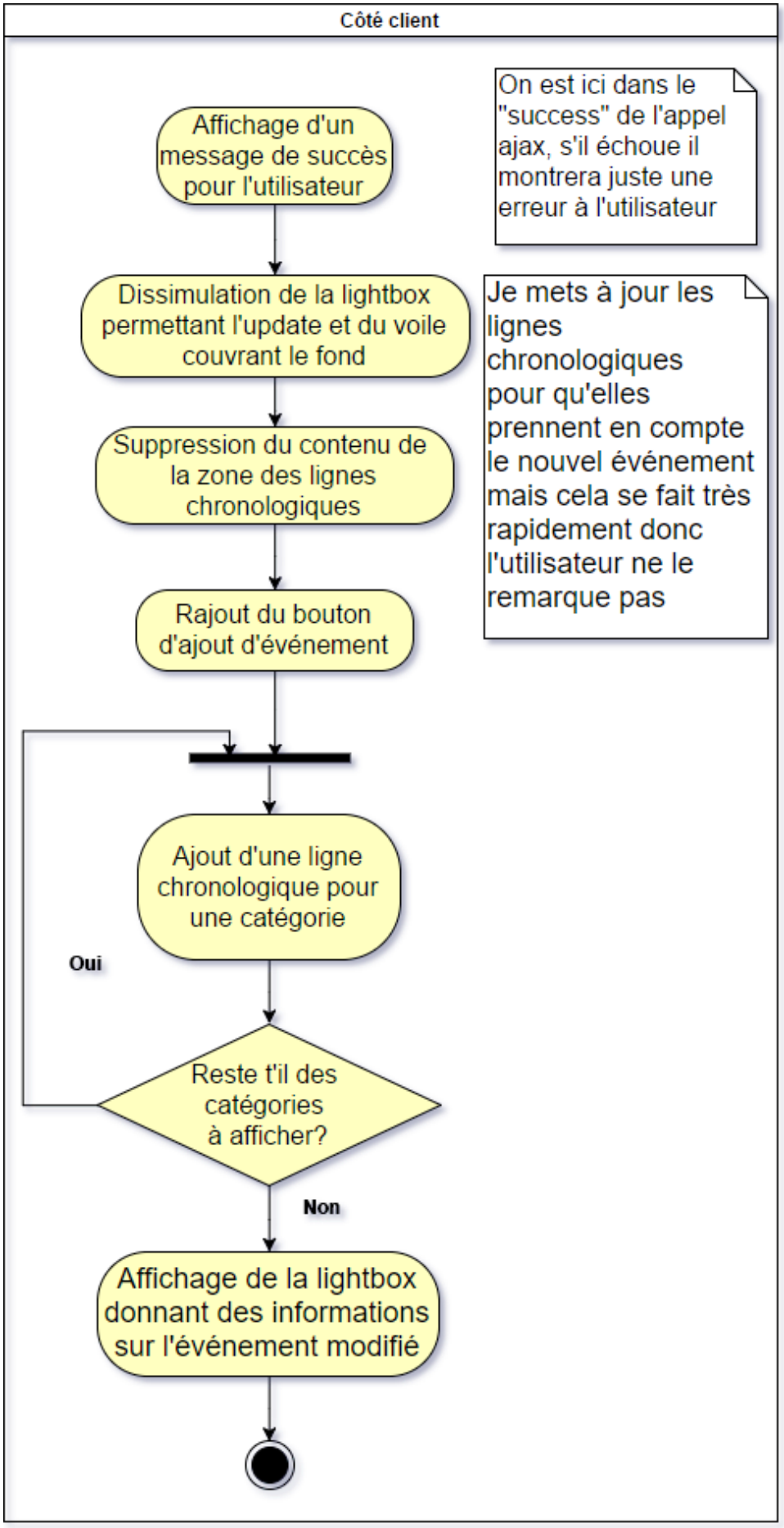
4.9.3. Confirmer les changements (saveUpdate())

J'ai dû séparer le processus en trois images différentes pour qu'elles restent lisibles.

La première montre le code côté client avant l'appel ajax, la seconde ce que fait la page répondant à cet appel et la dernière le code en cas de succès de l'appel.


Utilisateur
Côté client
Click sur le bouton confirmer
Une image existe t'elle déjà?
Oui
Non
Récupérer son nom
Y a t'il une valeur pour le jour de début?
Oui
non
Exactement la même chose est faite pour le jour de fin.
Donner cette valeur à la variable startingDay
Assigner 'null' à la variable starting day
Création et remplissage d'un événement temporaire
Pour envoyer une image en ajax, on mets le processData à false pour que jquery ne transforme pas les données en chaines de caractères. L'objet formData n'est du coup pas automatiquement créé et il faut le faire nous-même
Création d'un objet formData
Ajout de l'objet temporaire à l'objet formData
Sous format JSON pour que le serveur puis le décoder
Une nouvelle image a t'elle été fournie?
Oui
Non
Ajout de la nouvelle image à l'objet formData
Appel Ajax
Avec les options processData = false (pour empêcher la transformation des données en string) et contentType = false (pour empêcher l'envoi d'header de contentType)


Côté Serveur
Décodage de l'objet événement
Une nouvelle image est-elle fournie?
Oui
Non
A partir d'ici, tout ce code est dans un try-catch. S'il échoue, il vérifie que la nouvelle image n'a pas été ajoutée, auquel cas il la supprime.
Déplacement de l'image dans les dossier du serveur
Il y avait il une ancienne image?
Oui
Non
Suppression de l'ancienne image
Début transaction mySQL
Requête d'update de l'événement
Pour la table event_has_category. Il est plus simple de tout supprimer (pour cet événement) puis de remettre les bonnes données plutôt que de modifier au cas par cas.
Suppression de toutes les relations cat - event pour cet événement
Ajout d'une relation cat -event
Reste t'il des relations à ajoutées?
Oui
Non
Fin de la transaction

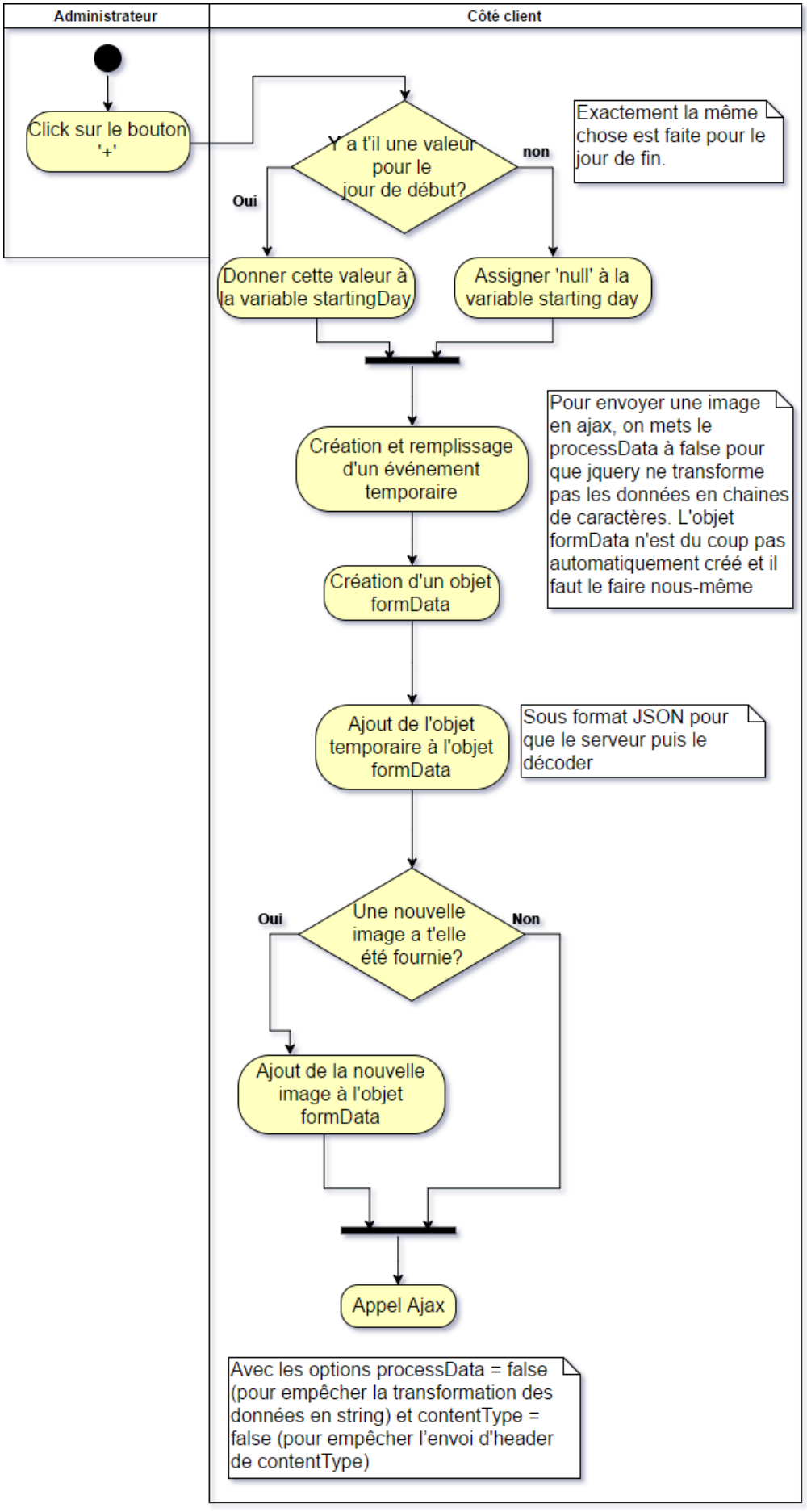

Côté client
On est ici dans le "success" de l'appel ajax, s'il échoue il montrera juste une erreur à l'utilisateur
Affichage d'un message de succès pour l'utilisateur
Dissimulation de la lightbox permettant l'update et du voile couvrant le fond
Je mets à jour les lignes chronologiques pour qu'elles prennent en compte le nouvel événement mais cela se fait très rapidement donc l'utilisateur ne le remarque pas
Suppression du contenu de la zone des lignes chronologiques
Rajout du bouton d'ajout d'événement
Ajout d'une ligne chronologique pour une catégorie
Oui
Reste t'il des catégories à afficher?
Non
Affichage de la lightbox donnant des informations sur l'événement modifié

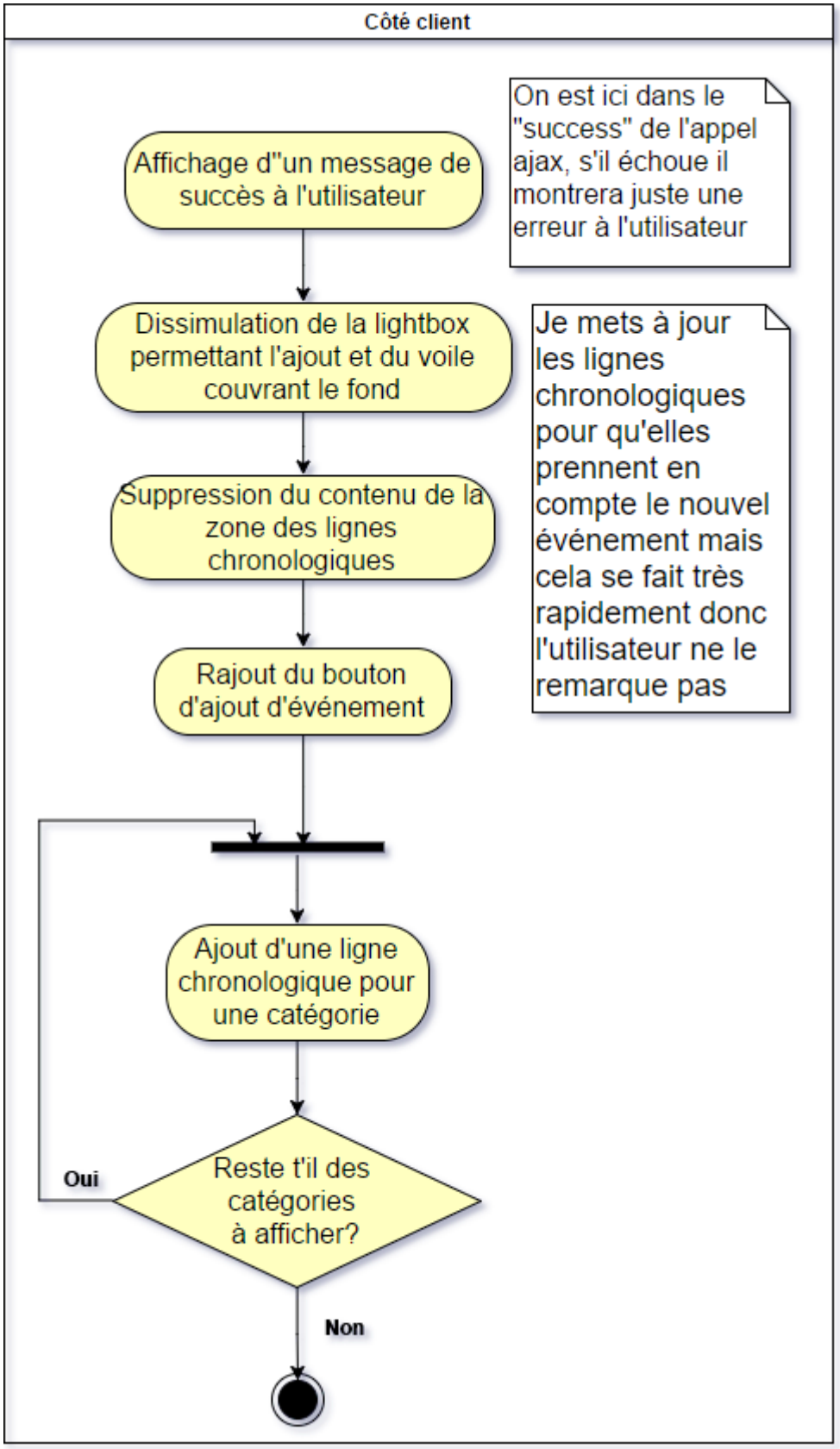
4.9.4. Annulation (cancelUpdate())

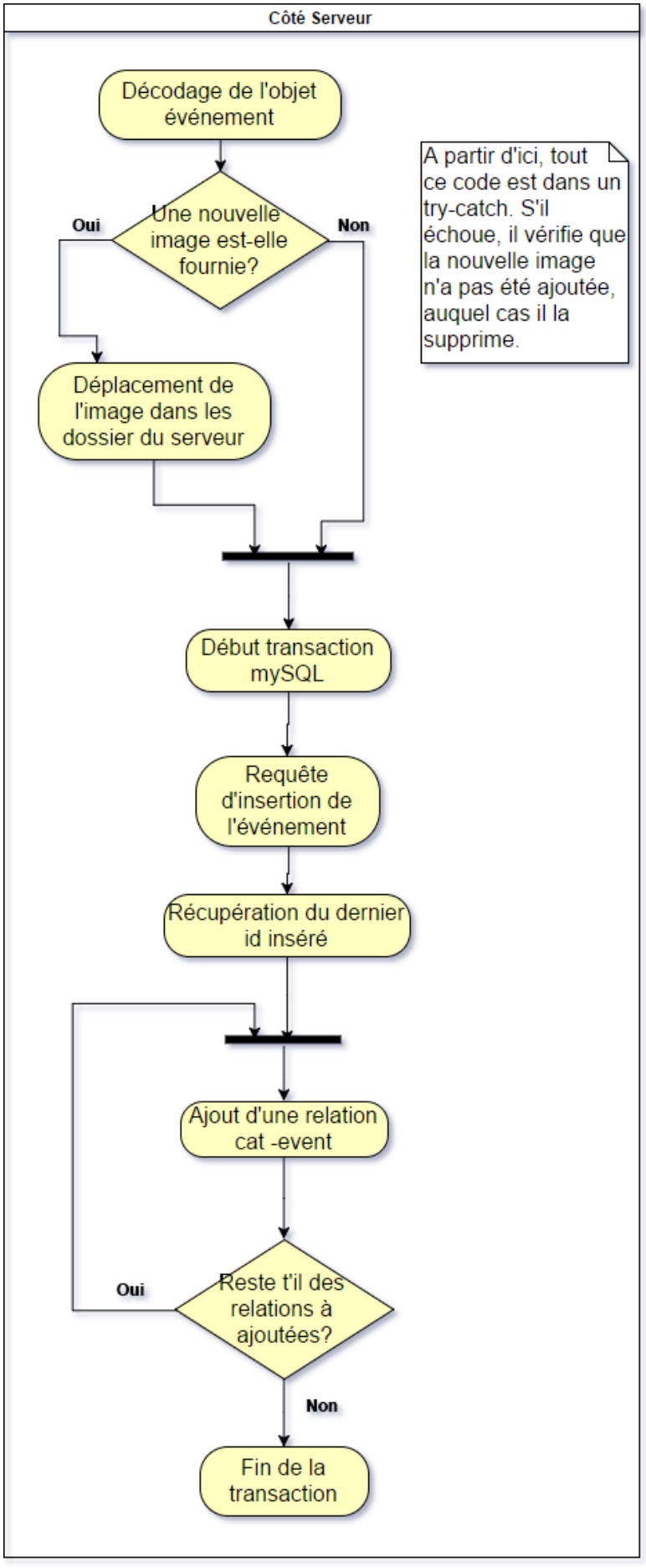
Un clic sur le bouton d'annulation cache la lightbox d'update et recharge celle montrant les informations.

4.10. Insertion d'événements (insertData)

Comme pour la lightbox de modification, celle d'insertion est construite préalablement puis cachée jusqu'au moment venu. Elle apparaît quand l'administrateur appuie sur le bouton « + » de la page. La fonction insertData fonctionne d'une manière similaire à celle de modification mais avec quelques conditions en moins. Je mets tout de même diagramme d'activité en cas de besoin. Il est lui aussi divisé en 3 parties pour des raisons de lisibilités. Celles-ci sont le côté client avant l'appel ajax, la page répondant à cet appel et le côté client si l'appel fut réussi.


Administrateur
Côté client
Click sur le bouton '+'
Y a t'il une valeur pour le jour de début?
Oui
non
Exactement la même chose est faite pour le jour de fin.
Donner cette valeur à la variable startingDay
Assigner 'null' à la variable starting day
Création et remplissage d'un événement temporaire
Pour envoyer une image en ajax, on mets le processData à false pour que jquery ne transforme pas les données en chaines de caractères. L'objet formData n'est du coup pas automatiquement créé et il faut le faire nous-même
Création d'un objet formData
Ajout de l'objet temporaire à l'objet formData
Sous format JSON pour que le serveur puis le décoder
Une nouvelle image a t'elle été fournie?
Oui
Non
Ajout de la nouvelle image à l'objet formData
Appel Ajax
Avec les options processData = false (pour empêcher la transformation des données en string) et contentType = false (pour empêcher l'envoi d'header de contentType)


Côté client
On est ici dans le "success" de l'appel ajax, s'il échoue il montrera juste une erreur à l'utilisateur
Affichage d''un message de succès à l'utilisateur
Dissimulation de la lightbox permettant l'ajout et du voile couvrant le fond
Je mets à jour les lignes chronologiques pour qu'elles prennent en compte le nouvel événement mais cela se fait très rapidement donc l'utilisateur ne le remarque pas
Suppression du contenu de la zone des lignes chronologiques
Rajout du bouton d'ajout d'événement
Ajout d'une ligne chronologique pour une catégorie
Reste t'il des catégories à afficher?
Oui
Non


Côté Serveur
Décodage de l'objet événement
Une nouvelle image est-elle fournie?
Oui
Non
A partir d'ici, tout ce code est dans un try-catch. S'il échoue, il vérifie que la nouvelle image n'a pas été ajoutée, auquel cas il la supprime.
Déplacement de l'image dans les dossier du serveur
Début transaction mySQL
Requête d'insertion de l'événement
Récupération du dernier id inséré
Ajout d'une relation cat -event
Reste t'il des relations à ajoutées?
Oui
Non
Fin de la transaction

4.11. Suppression d'un événement

En plus du bouton de modification, la lightbox d'affichage contient aussi un bouton de suppression. Quand il est cliqué, l'événement est supprimé et les lignes chronologiques mise à jour.

4.12. Ajout / Modification / Suppression d'une catégorie

Pour ne pas avoir à faire une page en plus juste pour les catégories, j'ai essayé d'intégrer leur ajout/modification/suppression directement dans la page, mais je crains que le résultat final ne soit trop chargé.

Si l'administrateur est connecté, displayCategories(), comme dit plus haut, rajoute ce qu'il faut pour ces trois actions.

Toutes les étapes que je vais décrire ci-dessous ont lieu dans des fonctions attachées aux événements des éléments de la liste pendant la fonction ready() du document.

4.12.1. Ajout

L'ajout se passe ainsi : si l'administrateur clic sur le premier élément 'Nouvelle catégorie', le texte est récupéré, effacé et remplacé par un champ textuel vide. Un symbole en forme de V est aussi ajouté à côté pour confirmer l'ajout.

Si la souris quitte le champ sans que l'ajout ait été confirmé, il n'est pas pris en compte et le champ avec son symbole disparaissent, remplacé par le texte 'Nouvelle Catégorie'.

Si par contre l'administrateur clic sur le nouveau symbole, un appel ajax est fait pour ajouter la catégorie dans la base de données. Les trois listes (dans la lightbox de modification, dans celle d'ajout et celle sur la page principale) sont ensuite mise à jour.

4.12.2. Modification

Quand l'administrateur clic sur le symbole de modification, le texte de l'élément est récupéré, effacé et remplacé par un champ textuel pré-rempli avec sa valeur.

Dès que la souris de l'utilisateur quitte le champ, la modification est exécutée, elle ne peut pas être annulée. J'ai choisi de m'y prendre ainsi car je ne pouvais pas encore rajouter une autre image, cela aurait surchargé la page. Je suis partie du principe que même une modification accidentelle n'est pas bien grave car il ne s'agit que du nom de la catégorie. Elle peut être aisément corrigée.

Les 3 listes de catégories sont comme précédemment mise à jour après la modification.

4.12.3. Suppression

Quand l'administrateur clic sur le bouton de suppression, la catégorie est supprimée et les 3 listes mise à jour

5. Connexion

La connexion de l'administrateur se fait sur la page login.php. Y est présent un formulaire de connexion avec nom et email.

Si les données sont valides, la variable de session 'connect' est mise à 'true' et l'administrateur redirigé vers la page principale. Je suis ici partie du principe que le nom de l'administrateur n'était pas important et que donc une simple variable 'connect' était suffisante.

Si les données ne sont pas valides, l'utilisateur est renvoyé sur la page de connexion avec une erreur.

Je stock la version hashée du mot de passe dans la base de donnée et je rajoute un sel au mot de passe de l'utilisateur avant de le hashé/vérifié.

6. Plan et rapport de test :

N° du test	Description du test	Résultat attendu	Date	Testeur	Statut
1	Aller sur index.php avec des événements dans la base de données	La page doit être chargée avec les bons styles, la liste des catégories existante mais cachée et le slider représentant la période du plus vieil au plus récent événement. Finalement, les deux poignées doivent se trouver autour du milieu du slider.	19.06.2017	Moi	OK
2	Passer la souris sur le titre « Catégories »	La liste des catégories doit apparaître en dessous	19.06.2017	Moi	OK
3	Passer la souris sur une des catégories	La couleur de fond et de police de cette catégorie doit changer	19.06.2017	Moi	OK
4	Cliquer sur une des catégories	Ses couleurs de fond et de police doivent changer, une ligne chronologique avec le nom de la catégorie sur la droite doit apparaître. S'il y a des événements pour cette catégorie/période, ils doivent aussi apparaître aux bons endroits	19.06.2017	Moi	Ok
5	[génération droite]	Si plusieurs points apparaissent au même endroit horizontalement, ils doivent se mettre les uns sur les autres verticalement.	19.06.2017	Moi	OK
6	Déplacement d'une des poignées du slider	La valeur de la poignée doit changer et la zone entre les deux poignées grandir/rétrécir	19.06.2017	Moi	OK
7	Essayer de faire passer une poignée par-dessus l'autre	C'est impossible, la poignée doit s'arrêter sur l'autre	19.06.2017	Moi	OK
8	Arrêter de déplacer une poignée	Les événements des lignes chronologiques affichées doivent se déplacer et éventuellement disparaître tandis que d'autre peuvent apparaître s'ils font partit de	19.06.2017	Moi	OK

		la période à afficher.			
9	Cliquer sur une catégorie quand elle est d'une autre couleur	Elle doit retourner à sa couleur initiale et sa ligne chronologique doit disparaître	19.06.2017	Moi	OK
10	Sortir la souris de la liste des catégories	La liste doit disparaitre	19.06.2017	Moi	OK
11	Laisser sa souris sur un point (0.5 secondes)	Un petit bloc avec son titre et ces dates doit apparaître (en-dessous, au milieu). De plus, le point grossi et sa bordure change de couleur	19.06.2017	Moi	OK
12	Laisser sa souris sur un point (0.5 secondes)	Si le bloc a une date de fin et de début différentes, les deux doivent apparaître, sinon juste celle de début	19.06.2017	Moi	OK
13	Laisser sa souris sur un point (0.5 secondes)	Si le point est trop près du bord, le bloc doit être décalé de façon à le voir dans son entièreté	19.06.2017	Moi	Ok côté gauche, pas côté droit
14	Laisser sa souris sur un point (0.5 secondes)	Si les dates de début/fin sont différentes un second point d'une autre couleur doit apparaître pour représenter la date de fin. De plus, une ligne en trait-tillé doit relier les deux	19.06.2017	Moi	OK
15	Enlever la souris du point	Le petit bloc, le second point et la ligne en trait-tillé doivent disparaitre. Le point doit retourner à son stade initial.	19.06.2017	Moi	OK
16	Cliquer sur un point quand il est dans son état initial.	Rien ne doit se produire	19.06.2017	Moi	OK
17	Cliquer sur le point après être resté au minimum 0,5 secondes dessus	Le fond doit être couvert d'un voile gris et une lightbox contenant le titre, les dates, la description et une image de l'événement si elle existe doivent apparaître.	19.06.2017	Moi	OK
20	Cliquer sur le voile gris	La lightbox d'affichage doit disparaitre	19.06.2017	Moi	OK
21	[lightbox]	Si les dates de début/fin sont différentes, les deux doivent apparaître, sinon uniquement celle de début	19.06.2017	Moi	OK
22	Connexion sur la page login.php avec les mauvais identifiant	Un message en rouge doit apparaître sous le formulaire de connexion	19.06.2017	Moi	OK
23	Connexion sur la page login.php avec les bons identifiants	Redirection sur la page index.php, en plus des autres éléments, un signe « + » doit apparaître	19.06.2017	Moi	OK
24	Clic sur le signe « + »	Un voile gris doit couvrir la page et une lightbox contenant un formulaire d'ajout d'événement doit apparaître	19.06.2017	Moi	OK
25	Validation du formulaire d'ajout sans avoir rempli tous champs	Les champs titre, description, année de fin, année de début et catégories doivent empêcher la validation	19.06.2027	Moi	OK
26	Validation du formulaire avec une année de fin plus	Une erreur doit apparaître et la validation bloquée.	19.06.2017	Moi	OK

	vielle qu'une année de début				
27	Validation du formulaire avec tous les champs obligatoires remplis	L'événement est créé, la lightbox d'ajout et le voile cachés et les lignes chronologiques rechargées pour prendre en compte l'événement	19.06.2017	Moi	OK
28	Clic sur un point en tant qu'administrateur après avoir laissé au moins 0.5 seconde la souris dessus	La lightbox d'affichage doit apparaître comme avant mais avec en plus un bouton « Modifier »	19.06.2017	Moi	OK
29	Clic sur le bouton « modifier »	La lightbox d'affichage doit disparaitre et celle de modification apparaître remplie de champs de formulaire contenant les valeurs de l'événement. Elle contient aussi un bouton « Confirmer » et « Annuler »	19.06.2017	Moi	OK
30	Clic sur le bouton « Modifier »	Qu'ils soit remplis ou non, les 3 champs pour les deux dates doivent être présents	19.06.2017	Moi	OK
31	Clic sur le bouton « Confirmer » sans les champs titre, description, année de début, année de fin et catégories remplis.	Apparition d'un message d'erreur	19.06.2017	Moi	OK
32	Clic sur le voile avec la lightbox d'ajout apparente	La lightbox et le voile disparaisse	19.06.2017	Moi	OK
33	Clic sur le bouton « Confirmer » avec les champs obligatoires apparent	La lightbox d'ajout disparait et celle d'affichage apparaît avec les données mise à jour. La zone des lignes chronologiques est mise à jour pour prendre en compte la modification.	19.06.2017	Moi	OK
34	Clic sur le bouton « Confirmer » avec la date de fin plus vielle que la date de début	Apparition d'un message d'erreur	19.06.2017	Moi	OK
35	Clic sur le bouton « Annuler »	La lightbox d'ajout disparait et celle d'affichage apparaît avec les valeurs non-modifiées	19.06.2017	Moi	OK
36	Retour sur la page login.php en étant connecté	Un bouton déconnexion apparait	19.06.2017	Moi	OK
37	Clic sur le bouton « déconnexion »	Redirection sur la page index.php, elle est chargée comme avant, sans le signe « + »	19.06.2017	Moi	OK
38	Ajout d'un événement avec une date de fin plus grande que la plus grande date de la bdd	L'événement doit être ajouté et le slider re-créé (il apparaît donc dans sa position initiale)	19.06.2017	Moi	OK
39	Passer sa souris sur le titre de la liste des catégories en tant qu'admin	La liste des catégories doit apparaître avec 'Nouvelle catégorie' comme premier élément et des symboles d'édition/suppression à côté des autres catégories	19.06.2017	Moi	Ok
40	Cliquer sur 'Nouvelle	Le texte doit être remplacé par un	19.06.2017	Moi	Ok

	Catégorie'	champ textuel vide et un symbole de validation doit apparaître			
41	Cliquer sur le symbole de validation avec du test dans le champ	La catégorie doit être ajoutée et l'élément revenir à son état de base (texte 'Nouvelle Catégorie')	19.06.2017	Moi	Ok
42	Cliquer sur un symbole d'édition de catégorie	Le texte doit être remplacé par un champ textuel rempli avec le texte	19.06.2017	Moi	OK
43	Enlever la souris de l'élément quand on a rempli le champ de modification	Le texte de l'élément doit être remplacé avec la valeur du champ (et le champ disparaître)	19.06.2017	Moi	OK
44	Cliquer sur le symbole de suppression d'une des catégories	La catégorie est supprimée	19.06.2017	Moi	Ok

7. Ressources utilisées

Tout au long de ce travail, je me suis aidée de différents sites pour vérifier les syntaxes ou l'utilisation de certaines fonctions. Voici la liste complète :

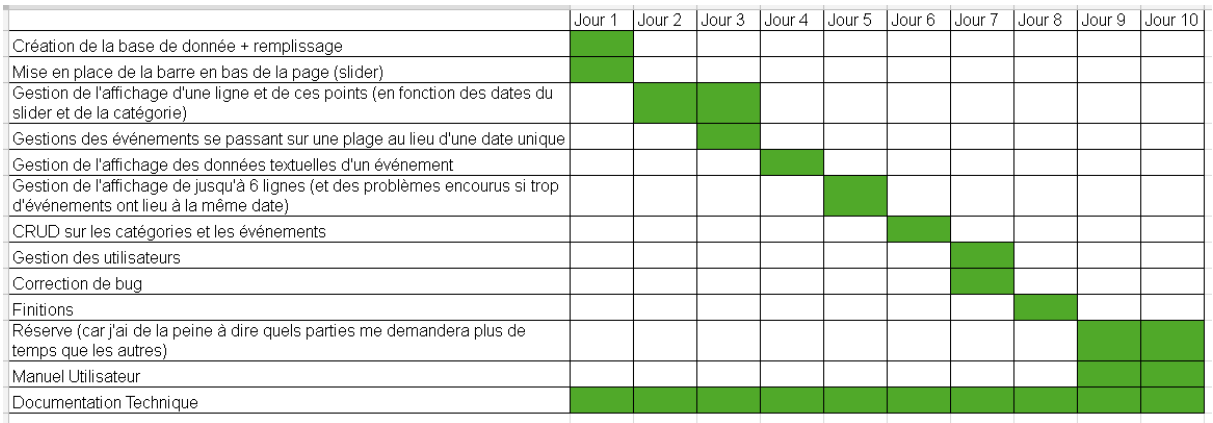
- Php.net
- Api.jquery.com
- Stack Overflow
- W3C Schools
- Mozilla Developer Network
- Sass doc
- noUiSlider doc
- wNumb doc

Je n'ai jamais pris de code tel quel mais je cherchais plutôt différents exemples desquels je tirais une utilisation personnel. Même si je l'avais voulu, je ne suis de toute façon jamais tombée exactement sur le bout de code qu'il me fallait.

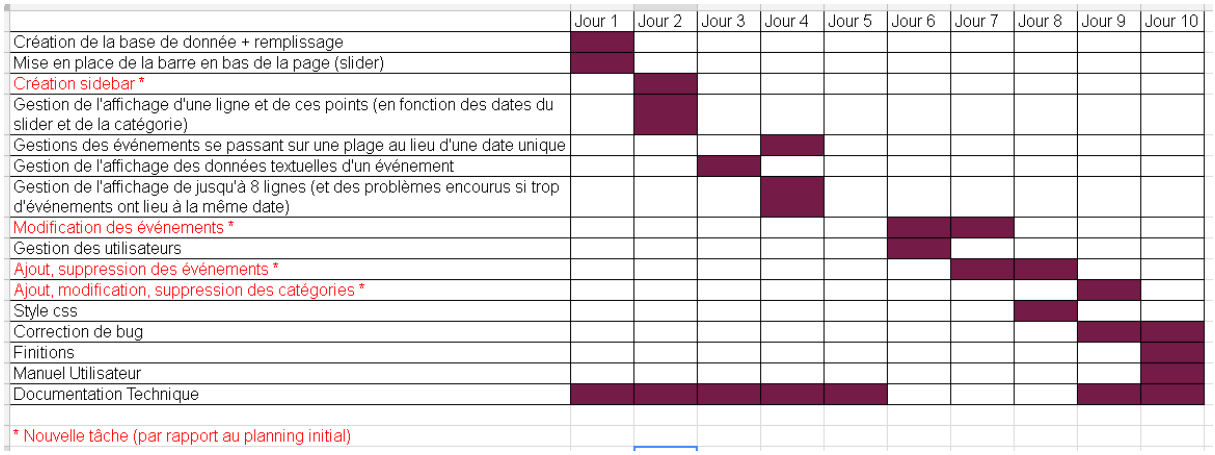
8. Conclusion

8.1. Plannings

Planning initiale :



Déroulement :



	Jour 1	Jour 2	Jour 3	Jour 4	Jour 5	Jour 6	Jour 7	Jour 8	Jour 9	Jour 10
Création de la base de donnée + remplissage										
Mise en place de la barre en bas de la page (slider)										
Création sidebar *										
Gestion de l'affichage d'une ligne et de ces points (en fonction des dates du slider et de la catégorie)										
Gestions des événements se passant sur une plage au lieu d'une date unique										
Gestion de l'affichage des données textuelles d'un événement										
Gestion de l'affichage de jusqu'à 8 lignes (et des problèmes encourus si trop d'événements ont lieu à la même date)										
Modification des événements *										
Gestion des utilisateurs										
Ajout, suppression des événements *										
Ajout, modification, suppression des catégories *										
Style css										
Correction de bug										
Finitions										
Manuel Utilisateur										
Documentation Technique										
* Nouvelle tâche (par rapport au planning initial)										

Comme on peut le voir, les deux sont assez différents.

Premièrement, j'ai dû diviser des étapes en plusieurs parties, principalement le CRUD.

J'ai aussi abordé des étapes dans un ordre différent, notamment pour la gestion de la plage de durée et l'affichage des données textuelles car le premier dépendait en fait du second.

8.2. Problèmes rencontrés

8.2.1. Une seule page

Rétrospectivement, je ne pense pas que ce fut une bonne idée de toute faire sur une page. Cela aurait très bien été sans le CRUD, mais en devant rajouter toutes ces actions, et en plus de cela pour deux éléments (les catégories et les événements), ça faisait beaucoup trop au même endroit.

En soit, je pense que mon site est viable et que si un designer m'aidais à l'amélioré visuellement, il pourrait être intuitif et utilisable, mais dans l'état ce n'est pas vraiment le cas.

Sans compter que gérer tout ainsi m'a pris beaucoup de temps, car même si ce n'est pas long à faire, utiliser du javascript pour interagir de différentes manières en fonctions des actions de l'utilisateur est assez long à coder.

8.2.2. Appel ajax avec un fichier

Un autre élément qui m'a posé problème fut l'appel ajax avec une image. J'ignorais qu'il était différent de celui sans image, et bien qu'il ne soit pas forcément compliqué, j'ai mis du temps à trouver la bonne syntaxe.

8.2.3. Documentation

Je ne suis pas convaincue par l'idée d'écrire la documentation en même temps que je code, car j'ai passé mon temps à modifier ce que j'avais déjà écrit.

C’est une des raisons pour lesquels je n’ai finalement pas autant de diagramme d’activités que ce que j’aurais voulu : le programme que j’utilise pour les faire ne rend pas leur modification aisée et j’ai perdu un temps fou à les retoucher au fur et à mesure que j’améliorais mon code.

Dans mon cas, je pense qu’il aurait été plus utile de réserver les deux derniers jours entièrement à la documentation et de faire que coder jusqu’à là.

8.2.4. Améliorations possible

- Design : le du site pourrait être plus propre et être rendue plus intuitif
- Code : je pense que le code pourrait être re-factoriser et améliorer encore une fois
- Fonctionnalité : Fusionner des catégories sur une même ligne au lieu de les comparer, ce serait intéressant dans d’autres scénarios d’utilisation.
- Fonctionnalité : permettre à l’administrateur de taper un texte avec titre, sous-titres, liste, images, etc. Bref un texte complet et non juste des paragraphes.
- Philosophie : au lieu d’uniquement permettre aux administrateurs d’ajouter des événements, se serait intéressant de permettre aux utilisateurs de chacun avoir leurs propres événement/catégories.

8.3. Avis personnel

Je pense avoir réussi à atteindre mon objectif mais j’aurais pu faire mieux, le site n’est pas exempt de défaut et la documentation n’est pas aussi détaillée que ce que j’aurais voulu. J’aurais préféré avec plus de diagramme d’activité car étant plus complet ils font plus honneur à mon travail que de simples phrases.

Je pense qu’avec un jour de plus, cela aurait été parfait, j’aurais eu le temps de bien complété mon rapport de tests (je ne doute pas que des tests plus spécifiques que ceux que j’ai eu le temps de faire échouerait), de compléter ma documentation et de la relire tranquillement.

Mais en tout et pour tout, je suis tout de même contente de mon travail.