

Rapport de Projet : Visualisation d'images

Rapport final

Ben ammar Iliass

Professeur :SALEH IMAD

Doctorant : MEKOUAR YUCEF



Table des matières

Préambule	2
Les technologies.....	3
L'upload.....	4
Base de données MYSQL.....	5
Google Maps API	6
Courbe du temps.....	7
Interface Finale.....	8
Conclusion	9

Préambule

Ce projet, réalisé dans le cadre du cours d'Hypermédia dirigé par le professeur Imad Saleh, a pour objectif de créer une application web permettant de visualiser des images à travers une carte Google et une timeline interactive. Cela permet de visualiser les images en fonction de leur emplacement géographique lorsque les métadonnées de localisation sont disponibles, ou en utilisant la timeline lorsque ces métadonnées ne sont pas disponibles. Pour cela, nous avons utilisé les technologies PHP, HTML, CSS et JavaScript, ainsi que l'API Google Maps et la librairie vis.js. Les images et leurs métadonnées sont stockées dans une base de données. En tant qu'étudiant BEN AMMAR Iliass, j'ai participé activement à la conception et à la réalisation de cette application.

Les technologies

J'ai choisi d'utiliser les technologies PHP, HTML, CSS et JavaScript pour ce projet car elles sont idéales pour la création d'applications web. PHP est un langage de programmation côté serveur qui permet de générer des pages web dynamiques en fonction des données stockées dans une base de données. HTML, CSS et JavaScript sont des langages côté client qui permettent de créer une interface utilisateur conviviale et interactive.



Google Maps

J'ai utilisé l'API Google Maps pour visualiser les images sur une carte car cette API permet de facilement intégrer des cartes dans une application web et de les personnaliser en fonction des besoins. Elle permet également de géolocaliser les images en utilisant les métadonnées de localisation.



J'ai utilisé la librairie vis.js pour la création d'une timeline interactive car elle permet de créer des visualisations de données temporelles de manière intuitive et personnalisable. Cela permet de visualiser les images en fonction de leur date de prise de vue lorsque les métadonnées de localisation ne sont pas disponibles.



Enfin, j'ai stocké les images et les métadonnées dans une base de données MySQL pour pouvoir les récupérer et les afficher facilement dans l'application. Cela permet également de sauvegarder les images et les métadonnées de manière permanente et de les rendre accessibles à tout moment.

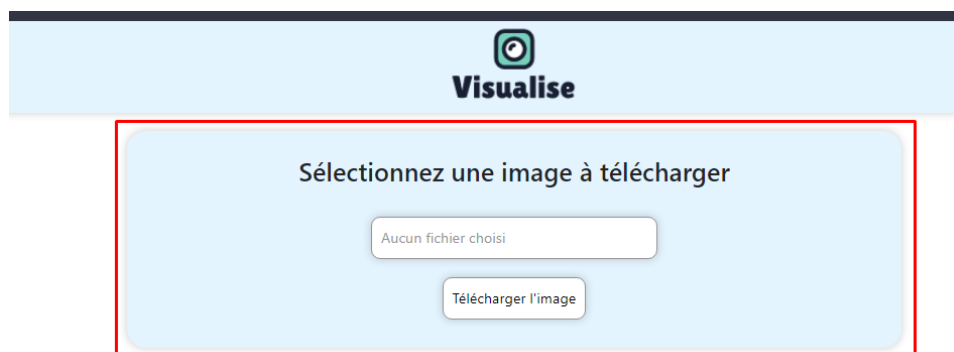
En somme, j'ai choisi ces technologies pour leur pertinence et leur facilité d'utilisation pour réaliser un projet de visualisation d'images à travers une carte Google et une timeline interactive, et pour leur capacité à gérer les données de manière efficace.

L'upload

Pour ce projet, j'ai développé une fonction d'upload qui permet aux utilisateurs de télécharger des images sur le site web. Lorsque le formulaire est soumis, cette fonction récupère les données des images téléchargées, ainsi que les métadonnées associées à ces images. Les métadonnées incluent des informations telles que le modèle de l'appareil photo, la marque, la taille, la date de prise de vue, ainsi que les coordonnées GPS de l'emplacement où l'image a été prise. Pour lire ces métadonnées, j'ai utilisé la librairie exif.

J'ai également utilisé une fonction gps2Num pour convertir les données GPS en un format lisible, ce qui permet de stocker les coordonnées GPS en degrés décimaux. Enfin, avant de stocker les images et les métadonnées dans la base de données, j'ai vérifié si l'image existait déjà dans la base de données pour éviter les doublons.

En utilisant cette fonction, les utilisateurs peuvent télécharger des images sur le site web, et ces images sont automatiquement stockées dans la base de données avec leurs métadonnées associées. Cela permet de rendre les images facilement accessibles pour l'affichage sur la carte ou la timeline.



The screenshot shows a web interface with a light blue header bar. In the center of the header is a circular logo with a camera icon and the word "Visualise" below it. Below the header is a light blue rounded rectangle containing the text "Sélectionnez une image à télécharger". Underneath this text is a white input field with the placeholder text "Aucun fichier choisi". Below the input field is a white button with the text "Télécharger l'image". A red rectangular border highlights the entire content area below the header.

Base de données MYSQL

La base de données MySQL utilisée pour ce projet comporte plusieurs colonnes pour stocker les informations sur les images téléchargées. La colonne "id" est une clé primaire qui permet d'identifier de manière unique chaque image dans la base de données. La colonne "image" utilise le format "mediumblob" pour stocker les images. Ce format est utilisé pour stocker des données binaires volumineuses, comme des images, car il permet de stocker des fichiers jusqu'à 16777215 bytes.

Les colonnes "name", "camera_model", "brand", "weight", "size" stockent respectivement le nom, le modèle de l'appareil photo, la marque, le poids et la taille de l'image. Les colonnes "created_at" et "gps_position_lat" et "gps_position_long" stockent respectivement la date de création de l'image, les coordonnées GPS de latitude et de longitude.

En utilisant cette architecture de base de données, nous pouvons facilement stocker les images téléchargées avec leurs métadonnées associées, telles que le nom, le modèle de l'appareil photo, la marque, le poids, la date de création, la taille et les coordonnées GPS. Cela permet de rendre les images facilement accessibles pour l'affichage sur la carte ou la timeline.

Options

				id	image	name	camera_model	brand	weight	created_at	size	gps_position_lat	gps_position_long			
<input type="checkbox"/>		Éditer		Copier		Supprimer	169	[BLOB - 131,2 kio]	CNQQ9645.JPG	Inconnu	Inconnu	134373	2022-12-15 18:48:56	1600 x 1200	0	0
<input type="checkbox"/>		Éditer		Copier		Supprimer	170	[BLOB - 157,9 kio]	DSCN0010.jpg	COOLPIX P6000	NIKON	161713	2008-10-22 16:28:39	480 x 640	43.4674483333	11.8851266667
<input type="checkbox"/>		Éditer		Copier		Supprimer	171	[BLOB - 155,4 kio]	DSCN0012.jpg	COOLPIX P6000	NIKON	159137	2008-10-22 16:29:49	480 x 640	43.4671566667	11.885395
<input type="checkbox"/>		Éditer		Copier		Supprimer	172	[BLOB - 146,6 kio]	DSCN0029.jpg	COOLPIX P6000	NIKON	150085	2008-10-22 16:46:53	480 x 640	43.4682433333	11.8801716666
<input type="checkbox"/>		Éditer		Copier		Supprimer	173	[BLOB - 153,9 kio]	DSCN0038.jpg	COOLPIX P6000	NIKON	157569	2008-10-22 16:52:15	480 x 640	43.467255	11.8792133333
<input type="checkbox"/>		Éditer		Copier		Supprimer	174	[BLOB - 153,7 kio]	DSCN0021.jpg	COOLPIX P6000	NIKON	157382	2008-10-22 16:38:20	480 x 640	43.4670816667	11.8845383333
<input type="checkbox"/>		Éditer		Copier		Supprimer	175	[BLOB - 129,5 kio]	ACSD5154.JPG	Inconnu	Inconnu	132565	2022-12-26 23:16:55	1600 x 1200	0	0

Google Maps API

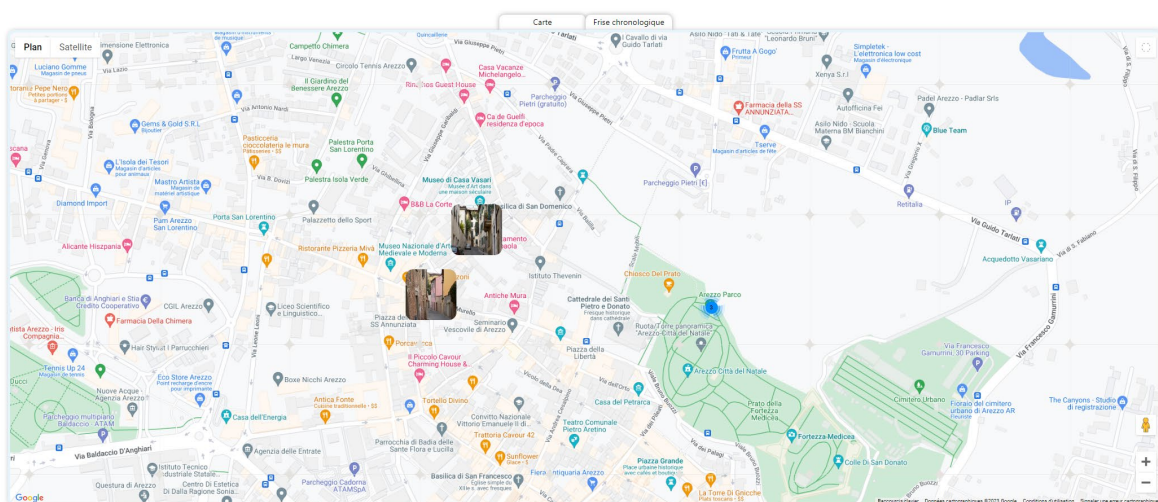
Dans ce projet, j'ai utilisé l'API Google Maps pour afficher les images téléchargées sur une carte. Pour utiliser cette API, j'ai dû créer un compte sur le site de développement Google Cloud Platform et obtenir une clé d'API valide.

Cette clé d'API doit être utilisée dans toutes les requêtes effectuées à l'API, et permet à Google de suivre l'utilisation de l'API et de facturer en cas d'utilisation excessive. Il est important de noter que l'utilisation de l'API Google Maps nécessite également l'inclusion d'un fichier JavaScript spécifique sur la page web, ce qui permet d'utiliser les fonctions de l'API.

Pour garantir la sécurité de l'application, j'ai utilisé un token qui est généré à chaque requête pour assurer que seul mon application peut utiliser l'API, ainsi pour éviter les abus et les utilisations non autorisées de l'API.

Enfin, pour rendre l'affichage des images sur la carte plus lisible et clair, j'ai utilisé un script de clustering pour regrouper les images qui sont trop proches les unes des autres, plutôt que d'afficher chaque image individuellement. Cela permet de garder une bonne lisibilité de la carte même si il y a un grand nombre d'images à afficher, en consolidant les images proches en un seul marqueur de cluster. Les utilisateurs peuvent alors zoomer sur la carte pour voir les images individuelles ou cliquer sur un cluster pour les afficher en vignettes.

Cette fonctionnalité est très utile pour éviter l'encombrement visuel sur la carte, en particulier lorsqu'il y a de grand nombres d'images à afficher. Cela permet également d'optimiser les performances de l'application en limitant le nombre d'éléments affichés sur la carte à un moment donné. En utilisant ce script de clustering, j'ai pu offrir une meilleure expérience utilisateur en permettant une visualisation claire et organisée des images sur la carte.

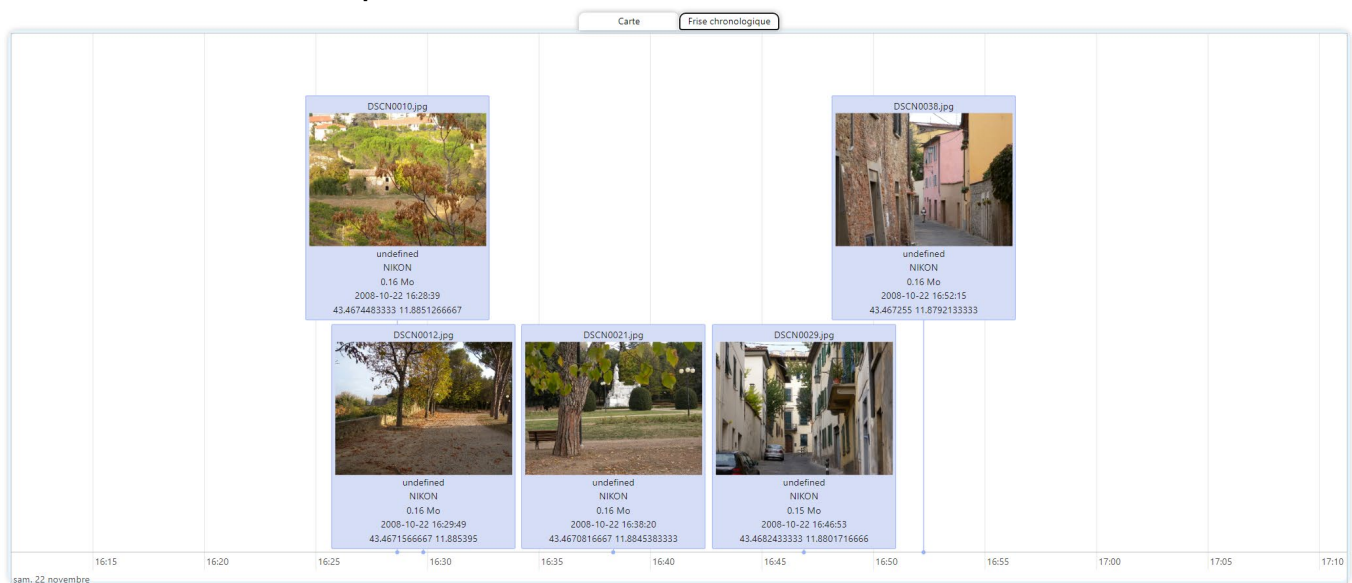


Courbe du temps

Dans ce projet, j'ai utilisé un script de timeline pour afficher les images téléchargées en fonction de leur date de création. Pour cela, j'ai utilisé la librairie vis.js qui permet de créer des timeline de manière simple. J'ai créé un script qui permet de récupérer les informations des images de la base de données, comme la date de création, et de les afficher sur la timeline en utilisant les fonctions de vis.js.

Pour utiliser la librairie vis.js, j'ai inclus le fichier JavaScript correspondant dans ma page web, et j'ai créé un élément HTML pour contenir la timeline. Ensuite, j'ai utilisé le script pour récupérer les informations des images de la base de données et les ajouter à la timeline en utilisant les fonctions de vis.js. J'ai également utilisé les options de vis.js pour définir une taille maximale pour les items de la timeline et pour spécifier les dates de début et de fin pour la période de temps à afficher sur la timeline.

Il est important de noter que dans le script, j'ai utilisé une fonction d'interval pour attendre que la liste des images soit remplie avant de lancer la timeline, ainsi on évite d'avoir des erreurs dans le cas où la liste est vide. J'ai également utilisé la fonction de zoom pour permettre à l'utilisateur de zoomer sur la timeline pour une meilleure lisibilité.

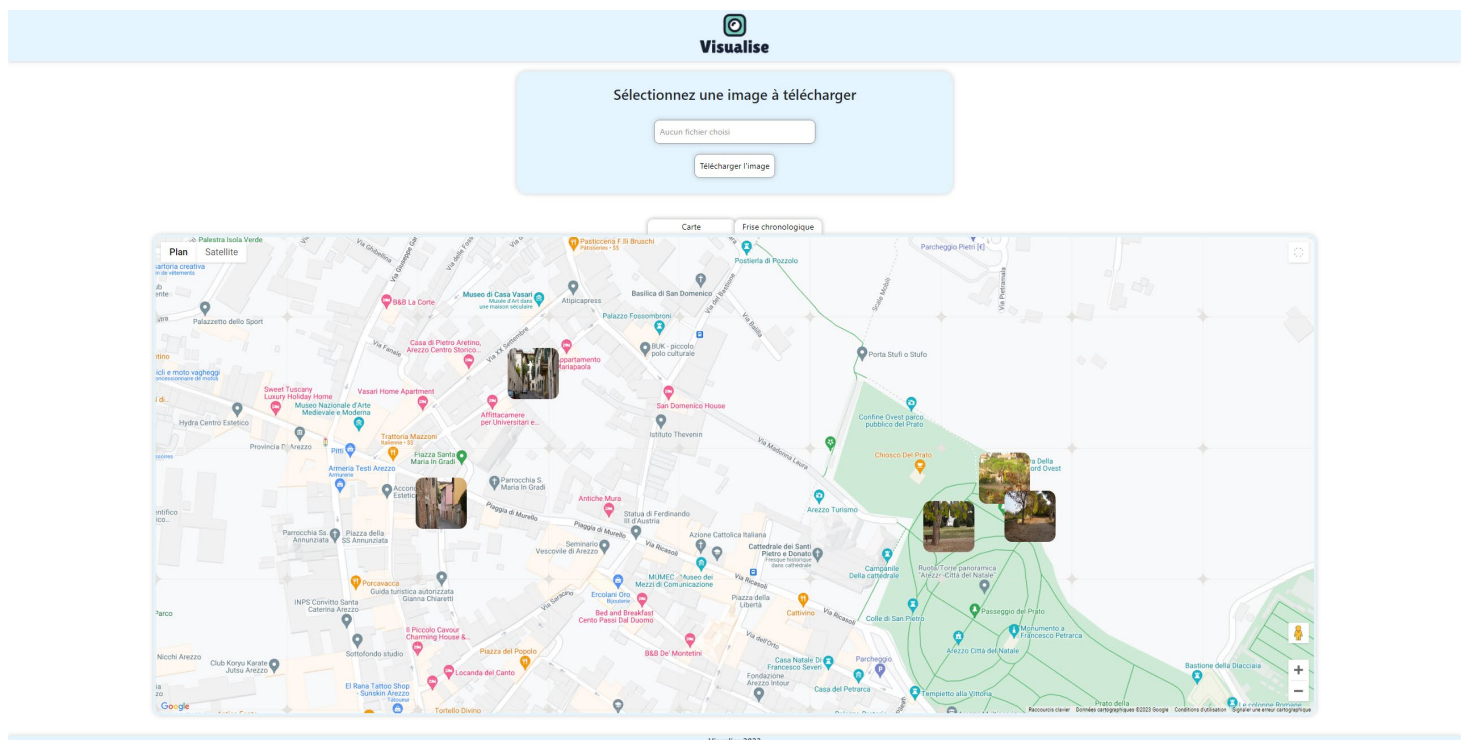


Interface Finale

J'ai créé l'interface pour l'application en utilisant principalement du HTML, CSS et JavaScript. J'ai commencé par ajouter un logo en en-tête de la page pour donner une identité visuelle à l'application. Ensuite, j'ai créé un formulaire d'upload pour permettre à l'utilisateur de télécharger des images sur le serveur. Ce formulaire utilise une méthode d'envoi "post" pour envoyer les données à un script PHP qui gère l'insertion des images dans la base de données.

Pour permettre à l'utilisateur de naviguer entre les différentes vues de l'application, j'ai ajouté deux boutons dans le bas de la page. Le premier bouton permet d'afficher la carte sur laquelle les images téléchargées sont affichées, tandis que le second bouton permet d'afficher une frise chronologique qui présente les images dans l'ordre chronologique de leur création. L'utilisateur peut basculer entre ces deux vues en cliquant sur les boutons correspondants. Pour cela j'ai utilisé du javascript pour changer le display de ces deux éléments, j'ai utilisé la fonction "display: none" pour cacher l'un des éléments et "display: block" pour l'afficher.

Pour finir, j'ai ajouté un footer à mon interface pour indiquer le nom du projet et la date du projet.



Conclusion

En conclusion, ce projet a été un exercice passionnant qui m'a permis de mettre en pratique les compétences acquises au cours de mes études en développement web. En utilisant les technologies PHP, HTML, CSS et JavaScript, ainsi que l'API Google Maps et la librairie vis.js, j'ai pu créer une application web qui permet de visualiser des images à travers une carte Google et une timeline interactive.

La fonctionnalité d'upload a également été un défi intéressant à relever, car elle impliquait la gestion des données des images téléchargées, ainsi que la lecture et le stockage des métadonnées associées. En utilisant la librairie exif, j'ai pu récupérer les informations telles que le modèle de l'appareil photo, la marque, la taille, la date de prise de vue, ainsi que les coordonnées GPS de l'emplacement où l'image a été prise.

En utilisant ces fonctionnalités, j'ai pu créer une application qui permet aux utilisateurs de visualiser les images en fonction de leur emplacement géographique lorsque les métadonnées de localisation sont disponibles, ou en utilisant la timeline lorsque ces métadonnées ne sont pas disponibles. Cela permet de créer une expérience utilisateur interactive et personnalisée qui facilite la navigation et la découverte des images.

Enfin, j'ai également intégré un système de boutons pour basculer entre la carte et la timeline, ainsi qu'un logo et un footer pour une présentation soignée de l'interface.

En somme, ce projet a été une expérience enrichissante qui m'a permis de mettre en pratique mes compétences en développement web, et de créer une application web fonctionnelle et interactive qui facilite la visualisation des images en fonction de leur emplacement géographique et de leur date de prise de vue. J'espère pouvoir continuer à explorer de nouveaux défis dans ce domaine passionnant de l'informatique.