**Présentation personnelle**

Mon intérêt pour l’informatique a débuté dès mon adolescence, que ce soit en stage de première lors de la création d'un logiciel avec Access, ou encore à travers mes cours d'informatique en BTS d’assistante de gestion où j’ai appris la modélisation de tables de données, entre autres. Cependant, on m’a souvent dit que je pouvais, parfois, ne pas avoir assez de patience pour ce domaine. Manquant de confiance en moi, ces remarques m’ont fait prendre d’autres chemins. J’ai alors exploré d'autres univers tout aussi intéressants et variés. Ces expériences m'ont permis de développer une confiance en moi et une persévérance qui, aujourd'hui, me permettent de me reconvertir avec légitimité dans ce domaine. Pour cela, j’ai suivi une formation de découverte du numérique puis une formation où j’ai acquis les bases de la programmation ce qui m’a permis d’intégrer sereinement le cursus actuel d’Elan Formation.

**Cahier des charges**

Présentation du projet

Je me suis rendue compte que de nombreux conflits restent dans l'ombre, souvent parce qu'ils sont jugés peu historiques ou peu glorieux pour certains pays. Parfois, certains événements sont même passés sous silence dans les cours d'histoire, alors que certains combats méritent pourtant d'être connus. Bien que je ne sois pas spécialiste en Histoire, j'ai souhaité partager quelques-unes de mes recherches, même si mon site n'a pas vocation à voir le jour.

Regards de guerre est une association fictive dédiée à l’organisation d’expositions sur le thème de la guerre. Chaque exposition met en lumière un conflit particulier, en utilisant le pouvoir de l'art pour susciter l'intérêt et la réflexion.

Organisation du site

Header :

L'en-tête, situé en haut de chaque page, joue un rôle crucial dans l'expérience utilisateur et l'identité visuelle du site. Il doit être à la fois fonctionnel, esthétique et adapté à tous les appareils.

Logo et liens de connexion/inscription : Placés à gauche on les trouve aisément.

Menu de navigation : Clair et intuitif permet aux utilisateurs d'accéder aux pages principales du site (accueil, expositions, tickets, blog, panier).

Pages :

* Accueil : Permet de connaitre les 3 prochaines expositions grâce à un carrousel, de connaitre l’association (local, café et magasin), d’avoir accès à l’agenda de toutes les expositions, de présenter l’accessibilité au lieu d’exposition.
* Exposition : Permet de connaitre le détail de l’exposition (sujet, artistes/type d’art, salle où sera représenté l’artiste)
* Ticket : Permet de connaitre les tickets disponibles pour chaque exposition et de les réserver.
* Panier : Permet de réserver l’ensemble des tickets.
* Back-office : Permet aux administrateurs de gérer l’ensemble du site à un seul endroit et sans avoir besoin de passer par le développeur (gestion des utilisateurs, expositions, artistes, tickets, stock)
* Blog : Permet aux utilisateurs de partager leur expérience et leur ressenti en écrivant des commentaire sur la page de l’exposition.

Footer :

Le pied de page, situé en bas de chaque page, regroupe les informations importantes et les liens utiles. Il contribue à la crédibilité et à la transparence du site.

Formulaire d'inscription à la newsletter : Un formulaire incitera les visiteurs à s'abonner pour recevoir les actualités de l'association.

Liens vers les réseaux sociaux : Des icônes discrètes mènent aux pages des réseaux sociaux de l'association.

Informations de contact : Un lien permet d’envoyer directement un mail à un administrateur.

Liens vers les pages légales : Des liens mènent aux mentions légales et à la politique de confidentialité.

Copyright : La mention du type d’association sera affichée.

Champs d’action

Les champs d’action de l’utilisateur en tant que :

* **Visiteur** : Un visiteur est un utilisateur non connecté qui peut naviguer librement sur le site et consulter les informations mises à disposition.

Consultation des pages publiques :

* Accueil : Présentation du site, agenda des expositions, plan d'accès et coordonnées.
* Blog : Accès aux articles des expositions passées et lecture des commentaires associés.
* Tickets : Informations sur les billets disponibles et les conditions de réservation.

Ajout de tickets dans le panier en vue d'une réservation future.

* **Utilisateur connecté** : Un utilisateur connecté est un utilisateur authentifié qui dispose de fonctionnalités supplémentaires lui permettant d'interagir avec le site et de personnaliser son expérience.
* Effectuer une réservation :
* Finalisation d'un achat de tickets : Possibilité d’ajouter, modifier ou de supprimer des tickets dans son panier avant de finaliser sa commande. Confirmation effectuée par email incluant le détail de la commande.
* Paramétrer son compte :
* Consulter son historique de commande : Accéder à la liste des réservations passées et aux détails de chaque commande.
* Modifier ses informations : Mettre à jour ses coordonnées ou son mot de passe.
* Supprimer son compte : Le membre peut demander la suppression de son compte et de ses données personnelles.
* Participer aux discussions : Partage d'opinions et échanges avec la communauté.

Les champs d’action de l’utilisateur en tant que :

* **Root** : Il dispose des droits les plus élevés. Ses actions se limitent à la gestion des autres administrateurs et à la configuration globale du système.

Ses responsabilités incluent :

* Attribution et modification des rôles.
* Suppression de comptes administrateurs.
* Gestion des permissions d'accès.
* Surveillance de l'activité des administrateurs (logs d'actions).
* **Administrateur** :
* Utilisateur :
* Profil utilisateur : Consultation, modification et suppression (à la demande ou si nécessaire).
* Historique des commandes : Consulter les réservations effectuées et envoie de facture sur demande.
* Artistes :
* Renseignement des données des artistes en vue de la création de la page d’exposition.
* Expositions :
* Création des pages complètes des nouvelles expositions (titre, description, prix, artistes, salle, upload des images)
* Tickets :
* Gestion des tickets : Ajouter, modifier ou supprimer des tickets (type, prix, nombre).
* Gestion des stocks : Mise en place d’un stock d’alerte.

* Blog :
* Expositions : Vérifier que les expositions passées soient transférées dans l’espace blog.
* Commentaires : Modérer les commentaires postés par les utilisateurs sur les articles ou les expositions (modification ou suppression si nécessaire)

MVP

L'ensemble des fonctionnalités présentées constitue le Minimum Viable Product (MVP), visant à démontrer la faisabilité d'une plateforme de gestion d'expositions et de réservations de tickets. Ce projet répond aux besoins essentiels des utilisateurs tout en garantissant une expérience utilisateur fluide et une gestion optimisée des utilisateurs, des expositions et des communications. Ce MVP pose les bases d'une plateforme évolutive, permettant d'ajouter de nouvelles fonctionnalités en fonction des besoins futurs des utilisateurs et des exigences du marché.

**Liste des compétences couvertes par REAC1**

* Développer la partie front-end d’une application web ou web mobile sécurisée
* Installer et configurer son environnement de travail en fonction du projet web ou web mobile
* Maquetter des interfaces utilisateur web ou web mobile
* Réaliser des interfaces utilisateur statiques web ou web mobile
* Développer la partie dynamique des interfaces utilisateur web ou web mobile
* Développer la partie back-end d’une application web ou web mobile sécurisée
* Mettre en place une base de données relationnelle
* Développer des composants d’accès aux données SQL et NoSQL
* Développer des composants métier coté serveur
* Documenter le déploiement d’une application dynamique web ou web mobile

Gestion de projet

La gestion de projet moderne exige des outils flexibles et des méthodologies agiles. Dans ce contexte, Trello, la méthode MoSCoW et le Kanban, au sein d'une approche agile, constituent une combinaison puissante pour organiser, prioriser et suivre l'avancement du projet de manière efficace.

Méthode MoSCoW : Priorisation des exigences

La méthode MoSCoW est utilisée pour prioriser les exigences du projet en fonction de leur importance. Elle classe les exigences en quatre catégories :

Must have (Doit avoir) : Les exigences indispensables pour que le projet soit considéré comme un succès.

Should have (Devrait avoir) : Les exigences importantes, mais qui peuvent être reportées si nécessaire.

Could have (Pourrait avoir) : Les exigences souhaitables, mais qui ont une priorité plus faible.

Won't have (N'aura pas) : Les exigences qui ne seront pas incluses dans cette version du projet.

Cette méthode permet de se concentrer sur les fonctionnalités essentielles pour la réalisation d'un MVP viable.

Kanban : Un flux de travail continu

Le Kanban est une méthode de gestion de flux de travail qui vise à optimiser la productivité en limitant le travail en cours. Il repose sur un tableau visuel qui représente les différentes étapes du processus de travail ("À faire", "En cours", "Terminé"). Les tâches sont déplacées d'une étape à l'autre, ce qui permet de visualiser l'avancement du projet et d'identifier les obstacles. Le Kanban encourage l'amélioration continue en permettant à l'équipe de s'adapter aux changements et d'optimiser son flux de travail.

Trello : Un tableau de bord visuel

L'intégration de ces méthodes est facilitée par Trello. un tableau de bord centralisé offrant une vue d'ensemble claire de l'état d'avancement. Son interface intuitive, basée sur des cartes et des listes, permet de visualiser les tâches et les échéances. Chaque carte représente une tâche, classée selon la méthode MoSCoW, et peut être déplacée entre les listes représentant les étapes du projet (Kanban).

Méthodologie Agile

L'approche agile a permis de structurer mon travail et de maintenir une organisation rigoureuse, même en travaillant seul. Elle a favorisé l'adaptation aux changements et aux imprévus, la concentration sur les tâches prioritaires et la limitation de la dispersion. Les itérations courtes et les objectifs réalisables ont contribué à maintenir la motivation et à visualiser l'avancement du projet. Au quotidien, une liste de tâches était établie, réajustée en fin de journée ou servant de base à la planification du lendemain.

Intégration et avantages :

Cette combinaison d'outils et de méthodes a permis de :

Visualiser l'avancement du projet en temps réel.

Identifier et résoudre les problèmes rapidement.

S'adapter aux changements et aux imprévus.

Réaliser le MVP dans les délais impartis.

Liste des compétences couvertes par REAC1

* Développer la partie front-end d’une application web ou web mobile sécurisée

Installer et configurer son environnement de travail en fonction du projet web ou web mobile

Maquetter des interfaces utilisateur web ou web mobile

Réaliser des interfaces utilisateur statiques web ou web mobile

Développer la partie dynamique des interfaces utilisateur web ou web mobile

* Développer la partie back-end d’une application web ou web mobile sécurisée

Mettre en place une base de données relationnelle

Développer des composants d’accès aux données SQL et NoSQL

Développer des composants métier coté serveur

Documenter le déploiement d’une application dynamique web ou web mobile

**Langages**

Côté client :

* HTML (HyperText Markup Language) : Langage de balisage utilisé pour structurer et organiser le contenu des pages web. Il définit la signification et la structure des éléments.
* CSS (Cascading Style Sheets) : Langage de style utilisé pour définir la présentation et l'apparence des pages web. Il permet de contrôler la mise en page, les couleurs, les polices, etc.
* JS (JavaScript) : Langage de programmation permettant d'ajouter de l'interactivité et des fonctionnalités dynamiques aux pages web (carrousel, menu burger).

Côté serveur :

* PHP (Hypertext Preprocessor) : Langage de programmation principal qui gère la logique métier, la communication avec la base de données, traiter des formulaires, personnaliser les pages web et la génération de contenu dynamique.
* SQL (Structured Query Language) : Langage de requête utilisé pour interagir avec des bases de données relationnelles. Il permet de récupérer, de modifier et de gérer des données stockées dans des tables.

**Technologies :**

Symfony :

Framework PHP doté d'une architecture Modèle-Vue-Contrôleur (MVC) qui favorise une organisation claire du code, facilitant ainsi la séparation des responsabilités et la maintenance. Ses composants réutilisables (formulaires, routes, sécurité) et sa haute configurabilité permettent de l'adapter précisément aux besoins de chaque projet.

Doctrine :

Object Relationnal Mapping (ORM) qui établit une passerelle entre PHP et la base de données relationnelle MySQL. Son approche orientée objet permet de manipuler les données avec DQL, un langage de requête intuitif, évitant ainsi l'écriture de SQL complexe. Les migrations facilitent la gestion des schémas de base de données, assurant la cohérence et la traçabilité des changements.

Twig :

Moteur de templates de Symfony favorise la séparation entre la logique de présentation et la logique métier, améliorant ainsi la lisibilité et la maintenabilité du code. L'héritage de templates permet de créer des mises en page réutilisables, réduisant la duplication de code et assurant une cohérence visuelle. De plus, il intègre des fonctionnalités de sécurité robustes pour protéger les applications web contre les attaques courantes.

**Environnement**

VS Code (Visual Studio Code) :

VS Code est l'éditeur de code de choix pour ce projet. Sa légèreté, sa performance et sa vaste gamme d'extensions en font un outil indispensable.

Exemple d’extensions utilisées :

* Prettier : automatise le formattage du code
* PHP intelephense : autocomplétion et verification de refactoring
* Twig Pack : Templates réutilisables, Fonctionnalités avancées (héritage de templates, fonctions (ex : path())

Son terminal intégré permet d'exécuter les commandes Symfony et Composer directement depuis l'éditeur.

Laragon :

C’est un environnement de développement local portable, isolé, rapide et facile à utiliser pour Windows. Il simplifie considérablement la configuration d'un serveur web, d'une base de données.

HeidiSQL :

C’est un outil de gestion de base de données qui, grâce à son interface graphique intuitive et à ses fonctionnalités complètes, facilite grandement la manipulation et la gestion des données. Il permet de visualiser et de modifier les données, de créer et de modifier les tables, d'exécuter et de tester des requêtes SQL avant de les adapter pour Doctrine Query Language (DQL).

Github Desktop et GitHub :

Github desktop, grâce à son interface intuitive, facilite le suivi des modifications de code, la gestion des branches et la résolution des conflits.

GitHub, quant à lui, est la plateforme de contrôle de version et de collaboration utilisée pour ce projet et permet à mes formateurs de suivre l’avancement de mon projet en temps réel.

Composer :

Gestionnaire de dépendances qui simplifie l'installation et la mise à jour des bibliothèques et des composants nécessaires à une application PHP, assurant la cohérence des versions et évitant les conflits. Il utilise un fichier « composer.json » pour définir les dépendances du projet (doctrine, bundles (ex : la sécurité), packages (ex : les formulaires), bibliothèques (ex : Symfony mailer).

Bundle :

Ensemble structuré de fichiers PHP, de configurations, de routes, de contrôleurs, de templates et d'autres ressources qui fournissent une fonctionnalité spécifique. (exs : SecurityBundle, SymfonyCastsResetPasswordBundle, domPdf).

Looping :

Logiciel de modélisation conceptuelle des données (MCD) utilisé pour créer des diagrammes relationnels. Il permet de concevoir des bases de données en structurant les informations sous forme d’entités et de relations. L'outil facilite la création et l'édition de diagrammes avec des fonctionnalités intuitives. Looping aide à définir les clés primaires, étrangères, et les cardinalités entre les entités. Il génère automatiquement les scripts SQL pour la création des bases de données.

**RGPD**

Dans le cadre du respect du Règlement Général sur la Protection des Données (RGPD), j’ai implémenté un ensemble de mesures de sécurité rigoureuses. Ces mesures visent à protéger les données personnelles de nos utilisateurs à chaque étape de leur cycle de vie, depuis leur collecte initiale jusqu'à leur suppression définitive.

* Gestion des droits et des données personnelles
* Licéité, loyauté et transparence :

Chaque utilisateur est informé clairement des informations collectées et un consentement explicite est requis via une case à cocher lors de l’inscription. Une politique de confidentialité détaillée est accessible depuis la navigation du site, garantissant une communication transparente sur l’usage des données.

* Limitation des finalités :

Les données personnelles sont collectées uniquement pour des finalités déterminées, explicites et légitimes. Ces informations servent exclusivement à gérer les réservations de billets.

* Minimisation des données :

Seules les informations strictement nécessaires à la gestion des services sont collectées et conservées (email pour la connexion, pseudo pour les commentaires, nom et prénom pour commander). Ce principe de minimisation limite les risques en cas de compromission.

* Exactitude – droits d’accès/rectification :

Les utilisateurs peuvent consulter, modifier ou corriger leurs données personnelles via un espace dédié. En cas d’erreur ou de demande spécifique, ils ont également la possibilité de solliciter une rectification en nous contactant directement. Cette approche garantit que les informations traitées restent précises et à jour.

* Limitation de la conservation :

Les données personnelles sont conservées uniquement pendant la durée nécessaire aux finalités pour lesquelles elles sont collectées. Les informations liées aux réservations sont supprimées après 10 ans. Ces mesures garantissent le respect des obligations légales en matière de durée de conservation comptable.

* Droit à l'oubli :

Les utilisateurs peuvent supprimer leur compte et toutes les données associées. Une suppression définitive est effectuée sur demande explicite, garantissant la disparition totale des informations.

* Sécurisation des données utilisateurs
* Anonymisation des utilisateurs : Afin de préserver la confidentialité des informations personnelles, les emails et les pseudonymes sont générés de manière unique et anonymisée. Cette pratique garantit qu'aucune information sensible n'est directement exposée.

La gestion de l'inscription (Register)

Les informations minimales sont demandées : email + mot de passe

Le nom et prénom ainsi que le pseudo sont demandés mais sont facultatives en effet les premiers seront obligatoires pour une commande le troisième le sera pour la partie blog.

* Validation des données : Les données personnelles (telles que l'email ou le nom) sont validées à l'aide des contraintes Assert de Symfony. Cela permet de garantir la conformité et la validité des informations entrées par l'utilisateur.
* Protection contre les attaques courantes
* Protection Cross-Site Request Forgery (CSRF) :

Tous les formulaires manipulant des données sensibles sont protégés contre les attaques CSRF grâce à l'activation native de csrf\_protection dans Symfony.

Lors de la soumission du formulaire, un jeton CSRF (un identifiant unique) est envoyé avec la requête. Ce jeton est généralement inclus dans le formulaire sous forme d'un champ caché.

Lors de la réception de la requête sur le serveur, Symfony vérifie que le jeton soumis avec le formulaire correspond à celui généré pour cet utilisateur et cette session. Si le jeton est absent, expiré ou incorrect, Symfony rejette la requête comme étant potentiellement malveillante.

* Prévention des attaques Cross-Site Scripting (XSS) :

La protection contre les attaques XSS est assurée à la fois en entrée et en sortie. En entrée, les données des utilisateurs sont validées via les **FormType** de Symfony, avec des filtres de validation adaptés pour limiter les risques d'injection. En sortie, **Twig** s'occupe de l'échappement automatique des variables dynamiques, ce qui empêche l'injection de scripts malveillants dans les pages web. Cette double protection garantit que seules des données sûres et validées sont affichées aux utilisateurs. Sanitization+htmlspecialchars

* Protection contre les attaques de force brute et par dictionnaire :

Un système de limitation des tentatives de connexion est mis en place à l'aide de la fonctionnalité Login Throttling de Symfony. Cette fonctionnalité permet de bloquer temporairement l'accès après un certain nombre d'échecs de connexion consécutifs. Cette mesure empêche les attaques de force brute, où un attaquant tente de deviner un mot de passe par essais successifs.

De plus, les mots de passe des utilisateurs sont sécurisés grâce à une empreinte numérique qui est générée en utilisant un algorithme de hachage sécurisé comme bcrypt. Ce processus consiste à transformer le mot de passe en une valeur fixe de longueur déterminée, ce qui rend l'original impossible à retrouver, même si la base de données est compromise. Le processus de salage (salt) ajoute une valeur unique avant le hachage du mot de passe.

Enfin, afin de renforcer la sécurité des mots de passe, une validation par expression régulière (regex) est appliquée. La regex suivante impose des critères stricts pour les mots de passe :

'/^(?=.\*[A-Z])(?=.\*\d)(?=.\*[\W\_]).{12,}$/';

Le mot de passe doit contenir au moins :

(A-Z) : Une lettre majuscule.

(\d) : Un chiffre.

([\W\_]) : Un caractère spécial.

{12,} : Douze caractères minimum.

* Protection contre les Injection sql :

Symfony utilise Doctrine ORM pour gérer l'accès à la base de données. Pour éviter les attaques par injection SQL, qui visent à manipuler les requêtes SQL en injectant des instructions malveillantes, Doctrine prépare systématiquement les requêtes en utilisant des requêtes paramétrées. Ce mécanisme empêche l'insertion de code malveillant en séparant la logique de la requête et les données.

Lors de la création des DQL (Doctrine Query Language), le QueryBuilder est utilisé pour construire dynamiquement les requêtes SQL de manière sécurisée. La méthode setParameter() permet de lier les valeurs des paramètres à la requête sans les injecter directement dans la chaîne SQL. Cela garantit que les valeurs sont correctement échappées et ne risquent pas d'être interprétées comme des instructions SQL.

* Protection contre la faille Upload :

Elle se produit lorsqu'une application web permet à un utilisateur de télécharger des fichiers sans vérifier correctement leur type, contenu ou origine. Pour cela, une limite des fichiers autorisés à été mise en place :

$allowedMimeTypes = ['image/jpeg', 'image/webp'];

Qui est ensuite enregistré sous un certain format sécurisé :

// Enregistrer l'image principale avec le nom 00\_main\_image

$fileName = '00\_main\_image.' . $file->guessExtension(); // Crée le nom du fichier + extension

$fileUploader->upload($file, $uploadDirectory, $fileName);

* Protection contre l’injection dans le code source :

Lorsqu'un lien s'ouvre dans un nouvel onglet (target="\_blank") ouvre sur le risque que le site cible puisse accéder à la page source via l'objet window.opener du navigateur, ce qui pourrait permettre à un attaquant de modifier la page originale. Ainsi une mesure de sécurité importante pour éviter les attaques potentielles qui pourraient être exploitées par des sites externes et de mettre dans les liens « rel="noopener noreferrer" » :

noopener : empêche la nouvelle page ouverte d'accéder à la page d'origine via window.opener

noreferrer : noreferrer empêche le navigateur de transmettre des informations de référence HTTP (Referer) au site cible. Cela protège la confidentialité des utilisateurs en empêchant le site de destination de savoir d'où provient la requête.

* Sécurité des fichiers et des téléchargements :

Les fichiers téléchargés sont soumis à une validation stricte (à l'aide de la contrainte File de Symfony), limitant les types autorisés (par exemple : .jpg, .png, .pdf) et leur taille.

* Responsabilité

Droits d’accès : La gestion des droits d'accès est strictement encadrée par l'utilisation des rôles (ROLE\_USER, ROLE\_ADMIN, ROLE\_ROOT). Les contrôles sont effectués via la méthode isGranted()

La gestion des login et register (inscription) est essentielle pour offrir une expérience utilisateur complète et sécurisée. Ces fonctionnalités permettent aux utilisateurs de créer un compte (inscription) et de se connecter à une application (connexion). Symfony utilise SecurityBundle.

**SEO**

* Structure sémantique et hiérarchique du contenu respectée (header/footer-section-div)

Une structure sémantique et hiérarchique bien définie est cruciale pour une bonne indexation des pages par les moteurs de recherche et pour offrir une navigation claire aux utilisateurs. Une structure correcte aide à comprendre l’importance relative des éléments et à mieux organiser le contenu.

* Balise <header> : Elle doit inclure les éléments essentiels comme le logo, la navigation principale et les informations primaires relatives à la page. Cette section informe les moteurs de recherche et les utilisateurs sur le contenu principal du site.
* Balise <footer> : Située en bas de la page, cette balise contient généralement les informations de contact, les liens de navigation secondaires, les mentions légales et la politique de confidentialité. Elle aide les moteurs de recherche à comprendre la structure du site tout en étant utile pour les utilisateurs.
* Balises <section> : Utilisées pour structurer les différentes sections d'une page. Chaque section devrait contenir un sujet distinct.
* Balises <div> : Utilisées en complément des balises sémantiques comme <section> pour éviter une structure trop « plate » qui pourrait compliquer l'indexation par les moteurs de recherche.
* Balises méta définies

Les balises méta sont des éléments HTML qui fournissent des informations supplémentaires sur le contenu d'une page et doivent toujours être adaptées à chaque page pour garantir un maximum de pertinence.

* Balise <title> : Le titre de la page qui apparaît dans les résultats des moteurs de recherche. Il doit être concis, contenir des mots-clés pertinents et décrire le contenu de la page. En général, la longueur idéale est entre 50 et 60 caractères.
* Balise <meta description> : Elle permet de donner un aperçu du contenu de la page dans les résultats de recherche. Bien que la meta-description n’influence pas directement le classement, elle impacte le taux de clics, donc elle doit être attrayante et inclure des mots-clés.

Exemple : <meta name="description" content="Découvrez notre exposition dédiée à la guerre d'Algérie, à travers des photographies et des témoignages poignants.">

- Attributs Alt Complets sur Toutes les Images

Les balises alt (textes alternatifs) sont essentielles pour le SEO des images. Elles permettent aux moteurs de recherche de comprendre le contenu de l'image, ce qui est particulièrement important pour l'accessibilité et l'indexation du contenu visuel. Une balise alt complète décrit l'image de manière précise et naturelle.

Ils permettent :

Accessibilité : Les utilisateurs ayant des handicaps visuels utilisent des lecteurs d'écran qui lisent ces balises pour décrire les images.

SEO : Les moteurs de recherche ne "voient" pas les images, mais ils peuvent comprendre le texte des balises alt. Cela améliore l'indexation des images dans les résultats de recherche, notamment dans la Google Image Search.

* URLs optimisées

La structure des URLs joue un rôle fondamental pour améliorer la visibilité et l'accessibilité du site. Une URL optimisée doit être claire, descriptive et contenir des mots-clés pertinents afin de faciliter l’indexation par les moteurs de recherche et d’offrir une meilleure expérience utilisateur.

Pour garantir cette optimisation, j’ai mis en place un système de génération automatique de slugs en utilisant l’outil SluggerInterface de Symfony. Cette approche permet de transformer les titres dynamiques en slugs normalisés : les caractères spéciaux sont supprimés, les espaces sont remplacés par des tirets (-), et l’ensemble est converti en minuscules. Cela garantit une cohérence dans la structure des URLs et améliore leur lisibilité aussi bien pour les utilisateurs que pour les moteurs de recherche. Pour cela, j’ai veillé à éviter les chaînes de paramètres complexes, utiliser des mots-clés ciblés sans surcharge, et privilégier des URLs courtes et explicites.

Par exemple, une URL telle que /exposition/guerre-algerie est plus compréhensible et performante pour le SEO qu’une URL dynamique du type /page?id=123.

Cela permet d’assurer une meilleure indexation des pages, de favoriser leur partage et d’améliorer l'expérience utilisateur tout en garantissant une compatibilité optimale avec les moteurs de recherche.

* Chargement des pages grâce à la compression des images

La vitesse de chargement des pages est un facteur clé pour l'expérience utilisateur et le SEO. Des pages plus rapides sont favorisées par les moteurs de recherche et génèrent un meilleur taux de conversion. Les utilisateurs sont de plus en plus impatients, et un site lent risque de les faire fuir, entraînant ainsi un taux de rebond élevé (pourcentage de visiteur quittant un site après avoir visité une seule page). Une méthode importante pour améliorer la vitesse de votre site est la compression des images. Les formats WebP ou JPEG sont a privilégier.

* Interactivité légère pour ne pas alourdir le site et donc le seo

Une interactivité légère améliore l'expérience utilisateur sans nuire à la performance du site. Une interactivité trop complexe peut alourdir la page, ralentir son temps de chargement et affecter le SEO.

Exemples d’interactivité légère :

Utilisation de JavaScript de manière judicieuse : Les effets dynamiques ou les animations doivent être utilisés de manière mesurée pour ne pas surcharger le site.

Chargement paresseux (Lazy Loading) des images et des vidéos : Cela permet de ne charger que les éléments visibles à l’écran, ce qui améliore le temps de chargement initial de la page.

Minimisation des scripts : Réduire la taille des fichiers JavaScript et CSS en utilisant des outils de compression comme UglifyJS ou CSSNano.

**Maquettage**

**Modélisation des données**

* **Méthode Merise**

La modélisation des données est une étape essentielle pour organiser et gérer les informations de manière optimale. J’ai choisi d’utiliser la méthode Merise qui offre une approche structurée et rigoureuse pour la modélisation des données et des traitements, permettant de passer d'une vision métier à une implémentation technique.

* **Modèle Conceptuel de Données (MCD)**

Il représente une vue globale et indépendante des contraintes techniques. Ce modèle décrit les entités (objets ou concepts manipulés), les associations (relations entre ces entités) et leurs cardinalités (contraintes numériques sur ces relations). L’objectif est de structurer les informations en se concentrant sur leur signification métier.

* Les relations et cardinalités

Dans la méthode Merise, les cardinalités sont essentielles pour définir la nature des relations entre les entités. Elles précisent combien d’occurrences d’une entité peuvent être associées à une autre.

Dans ce projet, 8 entités ont été identifiées.

* User : Représente les utilisateurs de la plateforme
* ManyToMany à l’entité Order
* ManyToMany à l’entité Exhibition
* ManyToMany à l’entité Comment

Avec pour attributs : userName, userFirstname, userNickname, reasonNickname, userEmail, password et une collection de rôles

* Order : Représente les commandes effectuées par les utilisateurs pour l'achat de tickets.
* OneToOne à l’entité User
* OneToMany à l’entité Ticket
* OneToMany à l’entité Exhibition

Avec pour attributs : orderCreation et orderStatus

* Ticket : Représente les billets d'entrée pour les expositions.
* ManyToMany à l’entité Type
* ManyToMany à l’entité Exhibition
* ManyToMany à l’entité Order

Avec pour attributs : titleTicket, imageTicket, imageTicketAlt

* Type : Représente les types de ticket.
* OneToOne à l’entité Ticket

Avec pour attribut : titleType

* Exhibition : Représente les expositions organisées.
* OneToMany à l’entité User
* ManyToMany à l’entité Order
* ManyToMany à l’entité Ticket
* ManyToMany à l’entité Artist
* ManyToMany à l’entité Room
* ManyToMany à l’entité Comment

Avec pour attributs : mainImage , mainImageAlt, titleExhibit, subTitleExhibit, hookExhibit, dateWarBegin, dateWarEnd, dateExhibit, hourBegin, hourEnd, descriptionExhibit, stockMax et stockMin

* Artist : Représente les artistes.
* ManyToMany à l’entité Exhibition
* ManyToMany à l’entité Room

Avec pour attributs : artistName, artistFirstname, artistBirthDate, artistDeathDate, artistJob, artistBio

* Room : Représente les salles présentes.
* ManyToMany à l’entité Exhibition
* ManyToMany à l’entité Artist

Avec pour attributs : titleRoom

* Comment : Représente les commentaires rédigés par les utilisateurs.
* OneToOne à l’entité User
* OneToOne à l’entité Exhibition
* **Modèle Logique de Données (MLD)**

Il traduit le MCD en une structure directement exploitable par un système de gestion de base de données (SGBD). À ce niveau, les entités deviennent des tables, les associations se traduisent par des clés étrangères et les cardinalités définissent les types de relations. Les relations ayant un ManyToMany de part et d’autre deviennent une table associative. Dans Symfony, les tables associatives deviennent une entité à part entière.

La relation Exhibtion – Artist – Romm donnera naissance à la table Show et présentera des attributs complémentaires (artistPhoto, artistPhotoAlt, artistTextArt) qui n’existeront que dans cette nouvelle table.

La relation Exhibition – Ticket donnera naissance à la table ticketPricing et présentera des attributs complémentaires (unitPrice, quantity) qui n’existeront que dans cette nouvelle table.

La relation Exhibition – Order donnera naissance à la table orderDetail et présentera un attribut complémentaire (standardPrice) qui n’existera que dans cette nouvelle table.

**Maquettage**

Le maquettage permet de donner un visuel au mvp et permet de visualiser la structure, l'agencement et l'interaction des éléments avant de passer au développement. Figma, avec ses fonctionnalités intuitives, est un outil de choix pour cette phase.

* Version Mobile (Mobile-First) : J'ai commencé par concevoir la version mobile pour assurer une compatibilité optimale sur les petits écrans. Cette approche est essentielle car la majorité des utilisateurs accèdent aux sites via leur smartphone. Cette méthode présente plusieurs avantages :
* Priorisation du contenu essentiel : Les contraintes des petits écrans obligent à se concentrer sur les informations les plus importantes.
* Amélioration de l'expérience utilisateur (UX) : Un design optimisé pour le mobile offre une navigation fluide et intuitive.
* Optimisation du référencement (SEO) : Google privilégie les sites adaptés aux mobiles.
* Version Desktop : Après la validation de la version mobile, j'ai adapté les maquettes pour les résolutions plus larges en optimisant l'affichage sur ordinateur.

**UX - UI**

L'UX se concentre sur la facilité d'utilisation et l'efficacité, tandis que l'UI concerne l'aspect visuel et l'esthétique.

UX : Améliorer le Parcours Utilisateur

L'objectif principal de l'UX est de garantir une navigation intuitive et efficace. Pour cela, j'ai adopté plusieurs bonnes pratiques :

Navigation fluide : Un menu clair et un panier accessible sur toutes les pages.

Parcours simplifié : Réduction des clics nécessaires pour atteindre une information.

Cohérence visuelle : Utilisation de repères visuels (boutons, icônes) pour guider l'utilisateur.

UI : Créer une Interface Claire et Esthétique

L'interface utilisateur se concentre sur l'aspect visuel du site. J'ai adopté une présentation épurée avec des éléments visuels cohérents :

Typographie lisible :

* Utilisation de la police Cormorant Garamond pour son esthétique élégante a été privilégiée pour les titres et certains paragraphes peu importants.
* Utilisation de la police Parkinsans pour sa facilité de lecture a été privilégiée pour l’emble des textes.

Espacements équilibrés : Pour une lecture confortable et une hiérarchie claire.

Boutons et interactions : Des boutons bien contrastés pour encourager l'action.

* Whocanuse : contraste
* Lighthouse :
* Unlighthouse :

Benchmark :

Permet de comparer les pratiques et les caractéristiques de sites similaires afin de mieux comprendre les attentes des utilisateurs et adapter le projet en conséquence. Cela aide à définir des besoins tout en préservant les habitudes des utilisateurs. Par exemple, je me suis inspiré du site du Musée Beyeler pour la gestion des tickets, afin de faciliter l'expérience utilisateur tout en intégrant une logique simple et fluide. De plus, le design du site Festival Film Festival, qui a eu une mention honorable en 2023, sur le site awwwards, a été une référence pour l’aspect visuel et l’interactivité, afin de garantir un design moderne et attractif.

**Moodboard : L'Univers Visuel**

Le moodboard est un tableau d'inspiration visuelle regroupant des éléments clés pour définir l'identité du site.

* Inspirations Principales :
* Ambiance Rétro-Mémorielle : Des tons doux et des textures rappelant le passé.
* Minimalisme Moderne : Interface épurée, facilitant la lecture et l'interaction.
* Accessibilité et Clarté : Contraste suffisant et hiérarchisation de l'information pour une expérience optimale.

**Charte Graphique**

La charte graphique assure la cohérence visuelle et identitaire du site. Elle comprend :

Palette de Couleurs :

* Salmon Puff (#FFA07A et ses déclinaisons) : Symbolise la chaleur et la joie.
* Gris Anthracite (#1e1e1e) : Pour les textes, apportant contraste et lisibilité.
* Blanc légèrement grisé (#F5F5F5) : Favorise la clarté sans agresser l’œil.
* Marron (#944C43) : Pour mettre en valeur les titres et les icônes.

Logos et Icônes :

* Logo, bien qu’il ne soit pas totalement minimaliste, il met en avant, dès le premier regard, un œil, symbole essentiel du projet. À côté de celui-ci, on retrouve des éléments liés à la guerre, comme le heaume, les épées, et les plumes, qui illustrent à la fois la lutte et les pertes humaines. Les ailes, quant à elles, ajoutent une dimension de mémoire et de souffrance, évoquant les nombreuses vies perdues au cours du conflit.
* Icônes simples et reconnaissables pour faciliter la navigation.

Images et Visuels :

* Des photographies immersives pour illustrer les expositions. Toutes issues de freepik et donc libres de droit, à l’exception de l’image pour l’exposition sur l’Ukraine qui est une œuvre de Bansky.
* Un traitement visuel sobre et épuré pour rester en accord avec le thème mémoriel.

**Disponibilité des fonctionnalités**

L'affordance d'un utilisateur désigne les indices visuels qui indiquent comment interagir avec un élément de l'interface. Par exemple, un bouton ressemble à un bouton (forme, couleur et taille adaptées) et être clairement étiqueté, comme "Commander" ou "Ajouter", pour que l'utilisateur sache qu'il doit cliquer dessus.

Les icônes doivent être immédiatement compréhensibles grâce à des symboles universels, comme trois lignes pour un menu ou une icône de retour.

La navigation doit être fluide, avec des éléments comme des liens soulignés ou des menus déroulants qui sont familiers et facilement identifiables. Une bonne affordance rend l'interface intuitive, minimisant la nécessité d’explications supplémentaires et offrant une expérience utilisateur agréable et efficace.

L'accessibilité des fonctionnalités à tout moment et pour tous les utilisateurs est essentielle. Pour cela :

* Interface intuitive : L'interface utilisateur est simple et claire, avec des éléments visibles et compréhensibles. Cela inclut des boutons, des liens et des icônes qui indiquent clairement leurs fonctions.
* Retours visuels : Afin que l'utilisateur sache ce qui se passe quand il interagit avec l'interface (par ex, lors du clic sur un bouton ou lors de la soumission d'un formulaire) un hover est mis en place que ce soit en version mobile ou en destkop.
* Visibilité des informations importantes : Les informations clés (prix, date d’exposition, boutons, call to action) sont mis en évidences sur les pages adéquates.

**Accessibilité**

L’accessibilité est notamment régie par les règles déjà évoquées dans la partie SEO comme le respect de la structure sémantique, les balises méta définie ou encore les balises alt complètes pour les images ou encore le contraste des couleurs argué dans la partie UI. Mais également :

* Les formulaires accessibles : étiquette label correspondant a l’attribut for
* Alertes accessibles aux lecteurs d’écran
* Langage de la page établi à fr pour que le lecteur d’écran puisse s’adapter
* Title sur tous les boutons et les liens

Responsive

**Architecture de l’application**

Le site est travaillé en version MVC grâce au framework Symfony :

Exemple : Affichage de l’agenda des expositions

* View : La vue est la partie du site que l'utilisateur voit et avec laquelle il interagit. Elle est responsable de la présentation des données, c'est-à-dire de l'affichage de l'interface utilisateur.

La vue est principalement gérée par Twig, le moteur de templates de Symfony. Ce moteur permet de séparer la logique de présentation de la logique métier.

Exemple : Pour afficher le détail de l’agenda sur le fichier de template Twig, j’ai fait appel à une boucle foreach d'afficher les informations nécessaires (date et horaire de l’exposition, texte d’information etc) Ces informations sont transmises au template par le contrôleur.

* Controller : Le contrôleur est responsable de la gestion des requêtes HTTP. Il interroge le modèle pour récupérer les données nécessaires et les envoie à la vue pour qu'elles soient affichées à l'utilisateur.

Exemple : Le controller rassemble les informations récupérées depuis la base de données via le model puis les transmet à la vue pour qu'elles soient affichées.

* Model : Le model est responsable de la gestion des données et de la logique métier de l'application. Dans Symfony, cela inclut la gestion des entités (objets représentant des tables de la base de données) et l'interaction avec Doctrine pour récupérer ou manipuler les données.

Exemple : Suite à la demande du contrôleur, le repository effectue une requête DQL (Doctrine Query Language) afin de récupérer les expositions, et les renvoie au contrôleur pour qu'il les envoie à la vue.

Requetes HTTP

Couches d appli

**Programmation Orientée Objet (POO)**

C’est un paradigme de programmation qui permet de structurer le code autour de "l'objet", une entité regroupant des données et des comportements. Dans la POO, tout est un objet, et chaque objet appartient à une classe. Ce paradigme vise à rendre le code plus modulaire, réutilisable, et facile à maintenir.

Les principaux concepts de la POO sont :

* Objet et Classe

Classe : Une classe est un plan pour créer des objets. Elle définit les attributs (titre, date, text, etc) et les méthodes (getters et settters ainsi que par exemple un getter spécialisé dans le formatage de date) que les objets de cette classe auront.

Exemple : Exhibition

Objet : Un objet est une instance d'une classe. Il possède les propriétés définies dans la classe et peut exécuter les méthodes définies pour cette classe.

Exemple : $exhibition = new exhibition(‘titre’, ‘date’, ‘text’ etc) ;

* Encapsulation

L'encapsulation consiste à restreindre l'accès direct aux données d'un objet. Les attributs sont généralement privés et ne peuvent être modifiés ou accédés qu'à travers des méthodes publiques appelées getters (pour obtenir des valeurs) et setters (pour modifier des valeurs). Cela permet de protéger l'intégrité des données et d'ajouter des règles de validation.

* L'héritage

C’est un concept fondamental de la programmation orientée objet (POO) où une classe enfant (ou sous-classe) hérite des attributs et méthodes d'une classe parent (ou classe de base). Cela permet de réutiliser le code d'une classe existante et de le spécialiser dans des sous-classes spécifiques sans avoir à le réécrire.

Symfony fournit une classe parent AbstractController avec des methodes pratiques comme celles pour gérer les formulaires, l’accès aux services ou encore rendre les réponses http et les vues Twig (grâce aux méthodes render() ).

* Polymorphisme

Le polymorphisme permet à une même méthode de fonctionner sur des objets de types différents, tout en ayant un comportement propre à chaque type.

Exemple : Service d’email

**Fonctionnalité phare**

Dans le cadre de mon projet, la gestion des expositions représente une fonctionnalité centrale, car elle permet à l'administrateur de gérer le contenu complet des événements proposés aux visiteurs. Cette gestion inclut plusieurs aspects importants, allant de la création et l'organisation des expositions à leur affichage et leur interaction avec les utilisateurs. Je vais détailler ici la gestion des expositions et du show.

Création et gestion des expositions

Les expositions sont créées, modifiées et supprimées par les administrateurs via une interface dédiée sur le backend de l'application, accessible uniquement aux utilisateurs ayant les permissions adéquates (rôle administrateur). L'administrateur peut ajouter une exposition en renseignant des informations essentielles comme :

Le titre de l'exposition : Un champ de texte pour le nom de l'exposition.

La description : Un champ de texte riche pour permettre une présentation détaillée de l'événement.

Les dates :

* Date de l’exposition : Cette date est cruciale pour déterminer le jour où venir découvrir l’exposition.
* Date de début et fin de la guerre : Pour bien cibler une période.

Photo : 1 image principale pour l’exposition ainsi que la description (alt)

Les horaires et les stocks de tickets : Renseignés par défaut, ils peuvent également être modifié au besoin.

Ticket + prix ?

Les expositions peuvent également être modifiées ou supprimées par l'administrateur, ce qui permet de garder un contrôle sur leur visibilité en ligne.

Association avec les artistes

Chaque artiste dispose d'une fiche associée dans la base de données comprenant son nom, prénom, sa date de naissance et de décès s’il y a lieu, son métier et sa biographie personnelle. Cette fiche doit être obligatoirement renseignée en amont.

Une exposition peut être liée à plusieurs artistes. Une fois l'exposition créée, il est possible d'associer les artistes qui participent à l'événement associé à une salle.

Photo : 1 image principale pour l’exposition ainsi que la description (alt)

Parcours artistique : Permet de savoir ce qui a motivé l’artiste dans cette créativité.

Les salles : Afin de retrouver son artiste préféré une fois sur place

Ainsi il est pratique d’ajouter, de modifier ou de supprimer un artiste d’une exposition tout en restant sur la même page.

L'association entre les expositions et les artistes est réalisée via des relations dans la base de données, permet une gestion souple.

**Affichage dynamique des expositions**

Une fois les expositions créées et configurées, un bouton de publication permettra alors la mise en ligne. Ce qui permet de gérer de manière flexible la visibilité ou non de ces dernières.

Il est crucial que le site affiche ces événements de manière claire et dynamique. L’affichage des expositions sur le site repose sur une série de pages et de fonctionnalités :

Page d'accueil : La page d'accueil du site affiche l’agenda des expositions à venir avec une courte description. Les expositions sont triées par date. L'utilisateur peut rapidement voir les événements à venir et en cliquant sur le lien « Fiche détaillée » il aura accès au détail complet de l’exposition.

Page de détails de l'exposition : Chaque exposition dispose de sa propre page détaillant toutes les informations relatives à l'événement : description complète, photos, artistes, horaires, etc. Cette page permet aux visiteurs de mieux comprendre le thème de l’exposition et de commander des billets.

Grâce à un système flexible de création, gestion et affichage des expositions, les administrateurs peuvent organiser efficacement les événements, tandis que les utilisateurs peuvent facilement trouver, découvrir et s’inscrire aux expositions qui les intéressent.

Autre fonctionnalité phare ?

**Axes d’améliorations**

* Système de paiement
* Blog
* Accesibilité : balises aria