Je m’appelle Laury et je suis actuellement en reconversion professionnelle chez Elan formation dans le but d’obtenir le titre de développeur web et web mobile.

Origine

Mon projet fait suite à la découverte d’expositions éphémères de dinosaures et de discussions sur l’Histoire avec une ancienne collègue. Regards de guerre est une association fictive qui gère un local comprenant six salles, chacune pouvant accueillir un artiste différent, et cela, autour du thème général de la guerre.

Benchmarks

Avant tout, j’ai réalisé du benchmark sur ces expositions mais également sur différents sites de musées ou encore des centres d’interprétation.

Détails des fonctionnalités

Le site met en avant les futures expositions et permet la réservation de tickets en ligne. Un blog permet aux utilisateurs de donner leur avis sur les expositions.

Le back office permet de gérer la gestion des stocks, les utilisateurs mais surtout la gestion dynamique des pages d’exposition. En effet, l’administrateur remplira un formulaire afin de renseigner toutes les informations concernant une exposition. Une fois validée la nouvelle page d’exposition viendra s’ajouter au site existant.

**Expositions**

Fonctionnalités :

Gestion des expositions

Technos :

Twig : pour afficher les vues dynamiquement en récupérant les données via Doctrine

Doctrine : pour structurer la relation entre les expositions les artistes et les tickets via la bdd

**Gestion et réservation des tickets**

Fonctionnalités :

Interface permettant de voir les tickets disponibles à la réservation

Ajout au panier

Validation de commande

Gestion des stocks

Envoi d’une confirmation après réservation

Interface user pour consulter l‘historique des commandes

Technos :

Symfony (controller + service) : gestion de l’ajout au panier et des commandes

Doctrine : stocker les commandes en bdd et associer les tickets aux users

**Blog et interactions users**

Fonctionnalités :

Consultation/rédaction et modification des avis

Modération des commentaires

Technos :

Twig : pour l’affichage

Doctrine : stocker les comments et gestion de la relations users/expos

**Back office / Tableau de bord**

Fonctionnalités :

Ajout/modif/suppression d’expos

Gestion des users activation/suppr compte

Gestion des billets vendus et historique des commandes

Technos :

Forms symfony : gestion formulaires d’expos

Doctrine : interactions avec la bdd

Security : gestion des roles

Gestion de projet

Pour une organisation flexible, j’ai opté pour une gestion de projet par le biais de trello ce qui m’a permis de suivre les étapes prévues mais également d’ajuster quand cela était nécessaire.

Méthode MoSCoW : pour identifier l’ordre de priorité des tâches

Méthode kanban : pour situer l’étape à laquelle se situe une tâche

MVP : site fonctionnel avec des expositions en ligne, une gestion de tickets basique et un back office

Maquette et choix ux/ui

UX

Une navigation claire et intuitive avec un panier visible à tout moment.

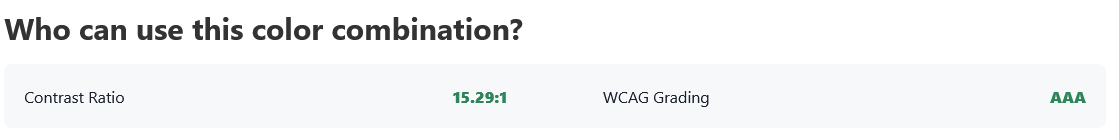
Du responsive afin que l’expérience utilisateur perdure

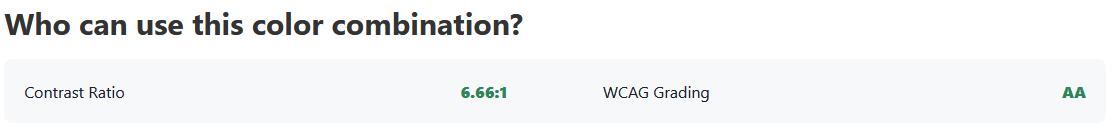
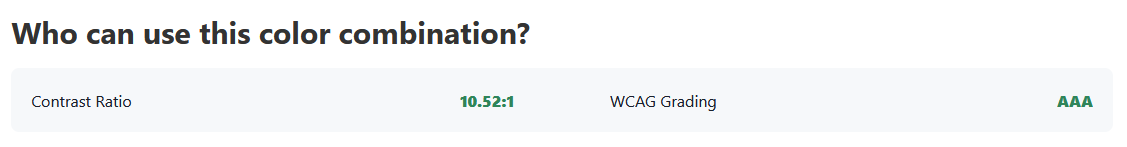
UI

Respect des contrastes, taille de police ajustables et navigation compatible avec les lecteurs d’écran.

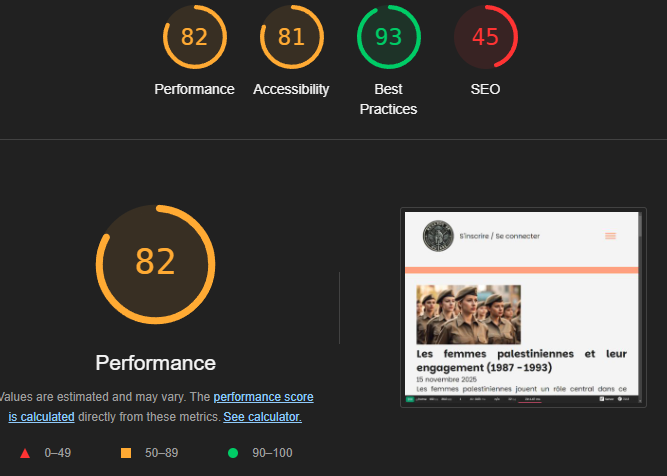
**Whocanuse** : pour vérifier le contraste :

Backgroundcolor+color

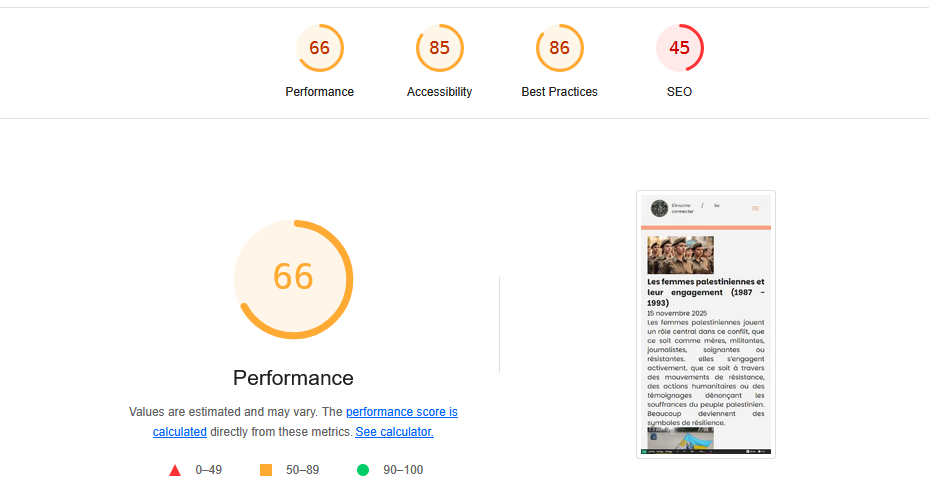


Button les principales couleurs  
  
  


**Lighthouse** :



**Unlighthouse :**

****

SEO

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

Balises méta définies (title-meta description)

Balises alt complètes sur toutes les images

URLs optimisées

Chargement des pages grâce à la compression des images

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

Base de données

Les expositions sont les éléments centraux du projet. De là partent les relations vers les artistes, l’achat de ticket, les commandes et bien sûr les utilisateurs.

Architecture logicielle

* Page d’expo

Le site est travaillé en version MVC

* View : L’utilisateur voit le détail d’une exposition affiché avec le moteur de template twig qui lui a été envoyé par le controller
* Controller : Le controller va interroger le modele afin de récupérer les informations demandées par Doctrine de l’entité et de ses relations afin de préparer les éléments nécessaires pour la vue
* Model : Le modèle enregistre la logique de récupération des données via Doctrine il interagit avec les repository pour accéder à la base de données

Sécurité

**Faille xss**

Twig échappe automatiquement les données {{ maVariable }}

Lorsque je créerais mes formulaires pour le back office je devrais utiliser les filter sanitization (ex que l’email correspond a un email) pour les input et htmlspecialchars pour vérifier qu’il n’y ait pas de code malveillant

**Faille csrf**

Symfony utilise un token unique inclus à chaque formulaire {{ form\_start(registrationForm) }}

Lorsque je créerais mes formulaires pour le back office je devrais mettre en place la génération d’un token dans ma fonction

**Injection sql**

Symfony utilise Doctrine pour la bdd et prépare les requêtes par le biais de requête paramétrées

-> lors de la création des dql : utilisation du queryBuilder / setParameter

**Attaque par force brute et par dictionnaire :**

Prise de l’empreinte numérique du mot de passe et ajout d’une regex

Je mettrais une limitation de tentative de connexion

**Injection dans le code source**

Mise en place de Noopener / No referrer dans les liens href (avec target \_blank)

Angle spécifique du projet

Le site respecte les principes d’affordance. En effet, conçu en mobile-first est pensé pour que la navigation reste fluide, intuitive et accessible par tous. Combiner à des mesures de sécurité les utilisateurs peuvent facilement explorer les expos et trouver les informations nécessaires en un clin d’œil.