

FORMATION DÉVELOPPEUR WEB ET WEB MOBILE

DOSSIER SNOWTRICKS



SESSION 2023

Maxime Lemée

Table des matières

Description projet :.....	3	Fixture.....	26
Contexte client	3	Fonctionnalités.....	27
Cahier des charges:.....	3	Page d' inscription.....	27
Objectif du projet:.....	4	Mot de passe oublié.....	30
Pré-Étude /.....	4	Slug / URL.....	31
Choix technique / Veille.....	5	Gestion des tricks.....	32
Choix technologique.....	5	Page de détail.....	32
Nommage.....	5	Espace de discussions.....	32
Données.....	6	Captcha.....	33
Planification du projet :	7	Pagination.....	33
Git.....	8	Carrousel.....	34
Front-end.....	10	Page de création.....	35
Maquettage du projet.....	10	Message flash.....	35
Création des diagramme UML.....	12	WYSIWYG.....	35
Diagramme de cas d'utilisation.....	12	Suppression.....	36
Diagramme d'activité.....	13	Gestions des droits.....	37
Diagramme de classe.....	14	EasyAdmin.....	37
SCSS.....	15	Référencement.....	38
Propriété Sticky.....	16	sitemap.xml / robot.txt.....	38
Bouton.....	16	SEO Quaker.....	39
Twig.....	17	LightHouse / Google.....	40
Responsive.....	19	Matomo Statistique web.....	41
Back-end.....	21	Sécurité.....	42
Création de l'architecture symfony.....	21	Nikto / Owasp zap.....	42
Installation.....	21	RGPD.....	43
Entité.....	21	CGU.....	43
Relation.....	22	Cookie.....	43
Doctrine.....	22	Compétences acquises.....	44
Controller.....	24	Remerciements.....	45
Profiler.....	25	Conclusion.....	46

Description projet :

Contexte client :

John Snow est un entrepreneur ambitieux, passionné de snowboard. Son objectif est la création d'un site collaboratif pour faire connaître ce sport auprès du grand public et aider à l'apprentissage des figures (tricks). Il souhaite capitaliser sur du contenu apporté par les internautes afin de développer un contenu riche et suscitant l'intérêt des utilisateurs du site. Par la suite, John souhaite développer un business de mise en relation avec les marques de snowboard grâce au trafic que le contenu aura généré.

Cahier des charges:

Après lecture du Cahier des charges ([voir annexe](#)), je visualise globalement ce qui est demandé.

Un site collaboratif (Figure 1) répertoriant des figures de snowboard, avec leur classement par catégorie et une gestion des utilisateurs connectés ou non, ce qui influencera l'expérience en leur permettant d'ajouter ou supprimer un commentaire sur la page, le détail d'une figure et de pouvoir créer/ modifier/ supprimer une figure.

- Un annuaire des figures de snowboard. Vous pouvez vous inspirer de la [liste des figures](#) sur Wikipédia. Contentez-vous d'intégrer 10 figures, le reste sera saisi par les internautes
- La gestion des figures (création, modification, consultation) ;
- Un espace de discussion commun à toutes les figures.

Figure 1: CDC Vue d'ensemble

Les informations suivantes doivent figurer sur la page :

- Nom de la figure
- Sa description
- Le groupe de la figure
- La ou les photos rattachées à la figure
- La ou les vidéos rattachées à la figure
- L'espace de discussion (plus de détails à la section suivante).

Figure 2: CDC Information Figure

- Le nom complet de l'auteur du message
- La photo de l'auteur du message
- La date de création du message
- Le contenu du message.

Dans cet espace de discussion, on peut voir la liste des messages postés par les membres, du plus récent au plus ancien. Ces messages doivent être paginés (10 par page).

Si l'utilisateur est authentifié, il peut voir un formulaire au-dessus de la liste avec un champs "message" qui est obligatoire et un champ captcha. L'utilisateur peut poster autant de messages qu'il le souhaite.

Figure 3: CDC Espace de discussions

Il est demandé que chaque figure contienne une liste d'informations (voir Figure 2) ainsi que certaines fonctionnalités comme des URL propres (lisibles par un humain) ainsi que l'espace de discussions qui devra contenir toute ces informations (voir Figure 3).

L'enregistrement de l'utilisateur se fera sur une page dédiée. La connexion de l'utilisateur se fera, également, sur une page dédiée avec la possibilité de réinitialiser son mot de passe en cas de perte.

Page d'inscription

La page d'inscription présente un formulaire qui demande :

- Le prénom
- Le nom
- Le nom d'utilisateur
- L'adresse email
- Le mot de passe
- Un champ de confirmation de mot de passe

Figure 4: Informations demandées lors d'une inscription

Objectif du projet:

Pré-Étude /

1. Après bonne réception du projet et lecture approfondie du cahier des charges, j'ai décidé dans un premier temps, afin de bien cerner le projet dans sa globalité, de créer dans un logiciel de gestion de projet, une première approche en listant les fonctionnalités demandées ainsi que la structure des pages.
2. Je continuerais sur la création des différents schémas de modélisation afin de structurer le squelette de l'application lors de sa création, c'est une étape très importante car toute la suite du projet sera dirigée par les choix faits à ce moment précis : le diagramme de classe pour la création des entités ainsi que leur relations, le diagramme d'activité pour la mise en place des différents flux et comment les mettre en place.
3. Ensuite, une fois tout cela validé, je concevrais le maquettage de l'application en privilégiant la méthode « Desktop First » c'est-à-dire concevoir le visuel pour l'application Desktop en premier lieu et si possible simuler la navigation d'un utilisateur.
4. Une fois le visuel confirmé, je commencerais le back-end, avec le framework Symfony, en portant une attention particulière à la création des entités et des relations entre elles, je résoudrais les divers bugs de navigation afin d'obtenir une première navigation propre, fluide et fonctionnelle.
5. Ensuite, je concevrais la partie front-end en m'attardant particulièrement sur le responsive pour les tablettes et les smartphones et validerais le bon affichage des données voulues et leur position sur les différentes pages.
6. Je rajouterais par la suite toutes les fonctionnalités voulues au cahier des charges et vérifierais leur bon fonctionnement.
7. Si le temps me le permettra, je rajouterais des options non imposées, comme un éditeur WYSIWYG ou un carrousel pour les photos des tricks ou une option pour gérer ses tricks favoris.
8. Une fois toutes ces étapes finalisées, je travaillerais sur le référencement de l'application.
9. Et pour finaliser, je réaliserais quelques tests sur les points critiques de l'application ainsi que tout un check-up de sécurité grâce à des outils dédiés.

Choix technique / Veille

Choix technologique

Devant un choix conséquent de solution et de mes connaissances limitées, j'ai beaucoup hésité sur lesquelles partir. J'ai commencé par poser les conditions du projet.

Côté back-end, le **framework Symfony nous a été imposé**, c'est la seule impérative de l'application.

Il faut choisir désormais la technologie pour le Front.

Nativement, Symfony utilise le générateur de template Twig que nous avons vu en cours , un outil bien documenté et avec une forte communauté donc techniquement un bon choix et j'ai également découvert le site ux.symfony.com qui propose une multitude d'outils front très intéressants pour l'optimisation de l'expérience utilisateur. Parmis ceux-là, je souhaiterais implémenté si le temps me le permet Image Cropper, qui permet de dimensionner une image lors de son upload par l'utilisateur, Vu.js également que l'on a vu en cours et qui permettra un affichage dynamique avec sa logique SFC (Single-File-Single) qui permet de regrouper dans un fichier le HTML, le javascript ainsi que le CSS.

Parmi les autres outils intéressants pour SnowTricks, on pourrait citer Stylized Dropzone qui rajouterai une 'drop zone' pour l'upload des fichiers des utilisateurs.

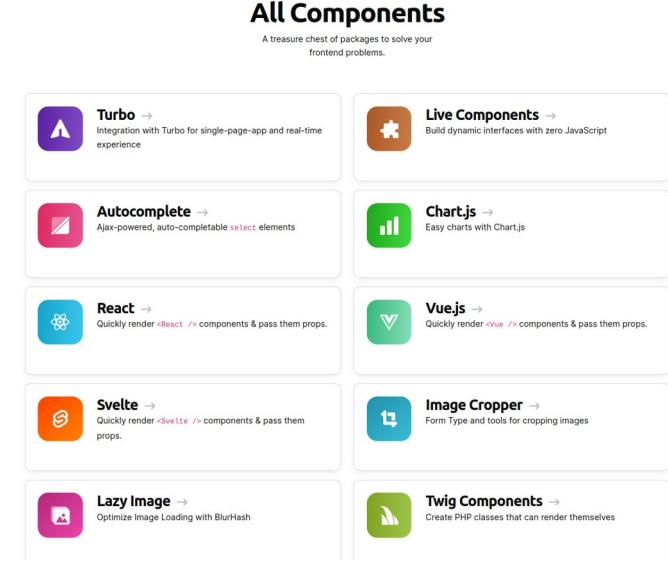
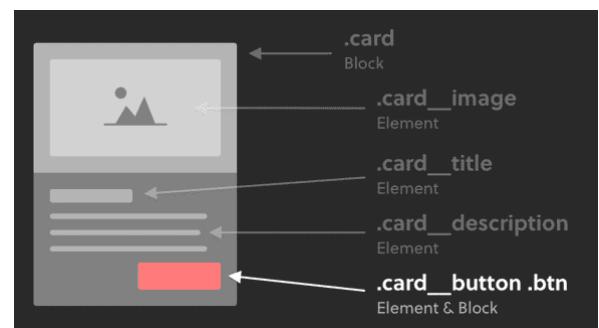


Figure 1: ux.symfony.com

Nommage

Afin de bien me retrouver dans le nommage de mes classes au sein de mon code HTML/CSS, j'ai décidé de suivre la convention BEM (Figure 3), qui est elle-même la synthèse de deux conventions déjà existantes : BEM (pour block Element Modifier) et ITCSS (Invertes Triangle CSS Architecture) (Figure 2). Cela me permettra de retrouver mes classes durant la mise en place de mon CSS.



Pour le nommage PHP et le JavaScript, ce sera du camelCase pour les Variables et les fonctions et du Pascal case pour les classes.

Pas forcément respecté actuellement au sein de mon développement, mais on peut également citer le nommage pour les données structurées, de la convention Schema.org que j'ai découvert un peu tardivement.

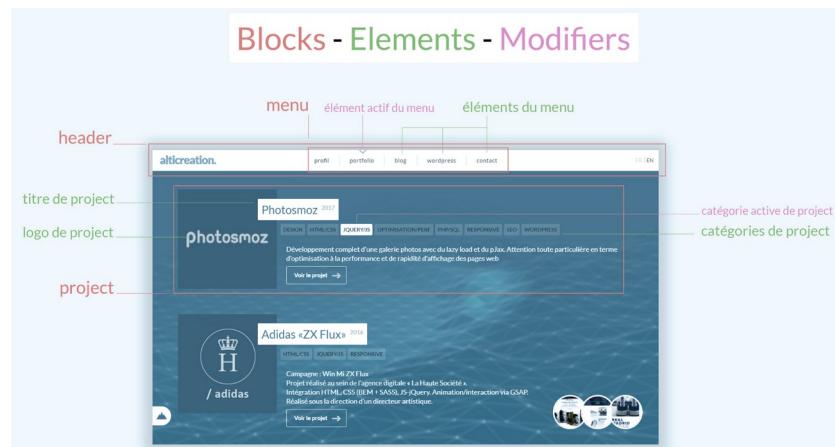


Figure 3: Logique BEM

Données

Dans l'ensemble de mon dossier de présentation, les données affichées lors des captures d'écrans seront liées aux fixtures (voir chapitre Fixture) que je générerais régulièrement au fil de mes tests et auront visuellement ou textuellement aucun lien avec le thème du site, mais les bonnes caractéristiques techniques comme la taille des images ou le nombre de caractères qui correspondront en tout point avec les données qui au final seront présentes sur le site et présentées aux internautes.

photo principal



Photo principale du troisième tableau de bord.

Description

/Debitis quisquam voluptas mollitia sint consequatur qui sed. Sunt ut iure eveniet fugit. Doloremque et animi et consectetur aut placeat.

Gallerie Photos



Photo secondaire du troisième tableau de bord.

Espace commentaire

Connectez-vous pour écrire un commentaire

Inscrivez-vous en cliquant ici

Nombre de commentaires : 29 commentaire(s)

Ipsam debitis quis magni praesentium nobis. Perferendis id consequatur ut quisquam est.

Publié le 2023 04 11

nvidal

Ut consequatur veniam expedita ipsum et. Enim nemo iusto doloremque iste dolorem dolor cupiditate. Qui ea maiores et dicta ipsam. Cumque quibusdam perspiciat quia.

Publié le 2022 10 22

fleury.william

Vidéos

<

2s

>

• • •

Figure 4: Données fictives

Planification du projet :

Afin de planifier mon projet, je vais utiliser l'outil de gestion de projet [OpenProject](#). Cela va me permettre de quantifier les actions et les fonctionnalités à réaliser durant toute la durée de ce projet .(Figure 5)



Fonctionnalités

Utilisateur connecté	Utilisateur non connecté	Administrateur optionnelle	Divers obligatoire	Tricks	Optionnelle	Bug à résoudre
#76 - Snow Trick TASK Création d'un compte utilisateur	#86 - Snow Trick TASK Proposition de se connecter ou de s'inscrire à tout moment	#88 - Snow Trick TASK Accès à une page dédié qui liste utilisateurs et figure avec tous les droits dessus	#102 - Snow Trick TASK Faire apparaître des messages flash pour diverses occasions	#100 - Snow Trick TASK Lors de l'ajout d'un trick, modal de confirmation	#99 - Snow Trick TASK gérage la page 404	#103 - Snow Trick TASK Suppression d'un utilisateur si actuellement connecté
#95 - Snow Trick TASK Vérification par mail	#85 - Snow Trick TASK Simple consultation des détails d'une figure	#87 - Snow Trick TASK Droit d'accepter/décliner ou proposer modification à une proposition de figure	#101 - Snow Trick TASK slug pour des URLs propres	#97 - Snow Trick TASK Mette en place un captcha	#98 - Snow Trick TASK Accès à la charte graphique du site	#89 - Snow Trick TASK Accès à un espace personnelle avec ces créations
#79 - Snow Trick TASK Pouvoir se déconnecter à tout moment	#84 - Snow Trick TASK Simple consultation de la page principale	#83 - Snow Trick TASK Soumettre une proposition de modification de n'importe quelle figure en passant par la validation de l'admin	#96 - Snow Trick TASK Pouvoir supprimer son compte définitivement			
#78 - Snow Trick TASK Pouvoir changer de mot de passe si perdu	#77 - Snow Trick TASK Crée une figure					

Figure 5: Tableau Fonctionnalités

Cet outil possède aussi une multitude d'options pouvant m'être utile durant toutes les étapes de la réalisation de mon parcours, comme son calendrier, son diagramme de Gantt, ses différents tableaux, son wiki et sa gestion du temps et optionnellement du coût du projet (Figure 6).

Snow Trick

Vue globale

OpenProject

Vue globale

DESCRIPTION DU PROJET

Développement du site

communautaire Snow Tricks

John Snow est un entrepreneur ambitieux passionné de snowboard. Son objectif est la création d'un site collaboratif pour faire connaître ce sport auprès du grand public et aider à l'apprentissage des figures (tricks). Il souhaite capitaliser sur le contenu apporté par les internautes afin de développer un contenu riche et susciter l'intérêt des utilisateurs du site. Par la suite, John souhaite développer un business de mise en relation avec les marques de snowboard grâce au trafic que le contenu aura générée.

CALENDRIER

lun. 22/05	mar. 23/05	mer. 24/05	jeu. 25/05	ven. 26/05	sam. 27/05	dim. 28/05

APERÇU DES LOTS DE TRAVAUX

Type

Ouverts 59
Clôture 12

DOCUMENTS

UML
07/04/2023 13:10

Cahier des charges
06/04/2023 08:05

ÉTAT DU PROJET

● SUR LA BONNE VOIE

MEMBRES

Member
● OpenProject Admin

+ Membre Afficher tous les membres

Figure 6: Vue globale Projet SnowTricks

Git

Dans la suite directe, j'ai créé mon dépôt git sur github(Figure 7) et cloné le projet sur mon ordinateur de travail. Je peux désormais travailler dans un environnement de travail bien organisé et sécurisé. J'ai aussitôt créé une branche develop parallèle à la branche Main (principale) sur laquelle j'effectuerais toutes les étapes mineures que je mergerais sur la branche main aux validations majeures.

J'essayerais de bien cibler mes commits effectués par action, cela structurera le projet et me permettra de voir les changements effectués à tel ou tel moment suivant la nomination du commit (exemple commit pour la prévisualisation de la photo principale du trick lors d'une création (Figure 9) ou le commit pour la mise en place du slug).

This screenshot shows a GitHub repository page for a project named 'Dwmm'. The 'Develop' branch is selected, showing 3 branches and 0 tags. The repository has 37 commits from user 'Maxlamenace53' over the last week. The commits are listed in a table with columns for file, message, and date. Key commits include 'Ajout gestion commentaire show trick', 'Modification pour suppression en cascade et oublie config pour slug', and 'Ajout Message flash lors de la création d'un trik'. The repository has 0 stars, 1 watching, and 0 forks. It also includes sections for Releases, Packages, Languages (PHP 98.8%, Twig 1.2%), and Suggested Workflows (Actions Importer, SLSA Generic generator).

This screenshot shows the commit history for the 'Develop' branch. The commits are organized by date: May 14, 2023; May 12, 2023; May 11, 2023; and May 10, 2023. Each commit is shown in a card with the author, message, date, and a copy/paste icon. For example, on May 14, there is a commit 'Ajout gestion commentaire show trick' from 'Maxlamenace53' committed last week. On May 11, there are several commits related to slug creation and refactoring. On May 10, there are commits for refactoring and adding a logo.

Figure 8: Exemple de commit branche Develop

Filter changed files

- assets
- app.js
- src/Form
- TrickType.php
- templates/trick
- _form.html.twig

```

16 00 -15,3 +15,19 @@ import './styles/style.scss';
15   import './bootstrap';
16
17 //registerVueControllerComponents(require.context('./vue/controllers', true, /\.vue$/));
18 +
19 +
20 + function readURL(input) { // nom de la fonction à appeler dans l'attribut onchange de l'input
21 +   if (input.files && input.files[0]) {
22 +     const reader = new FileReader();
23 +
24 +     reader.onload = function (e) {
25 +       document.getElementById('blah').setAttribute('src', e.target.result);
26 +     };
27 +
28 +     reader.readAsDataURL(input.files[0]);
29 +   }
30 +
31 +
32 + window.readURL = readURL;
33 +

```



```

25 00 -25,7 +25,9 @@ public function buildForm(FormBuilderInterface $builder, array $options): void
26   ->>> add('nameTrick', TextType::class)
27   ->>> add('description', TextareaType::class)
28   ->>> add('mainPhoto', FileType::class, [
29     'data_class' => null
30   ])
31   /* ->>> add('creationDate')
32   ->>> add('user', EntityType::class, [
33   */

```

Figure 9: Commit 'Prévisualisation de photoMain avant save lors creation new trick'

Front-end

Maquettage du projet

Pour maquetter mon application, j'ai utilisé l'outil Penpot, c'est une application semblable à Figma et adobe XD (que l'on a utilisé en cours mais payant) mais Open source et disponible soit en cloud soit en auto hébergement via docker.

J'ai conçu la page principale en premier (Figure 2) ainsi qu'une page de login en modal (Figure 1) à base de composants réutilisables comme le header, le footer ou la carte 'LastTrick', j'ai cherché la

palette de couleur adéquate ainsi que la police qui irait le mieux à l'ensemble.



Figure 1: Logo Penpot

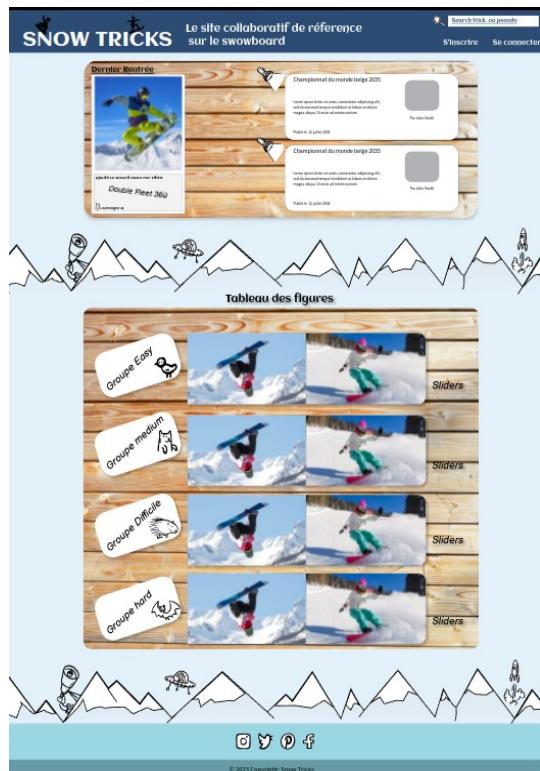


Figure 2: Maquettage Page d'accueil

Après cela, avec le logiciel [Krita](#), j'ai réalisé le logo principal du site (Illustration 1) que j'ai pu implanté en format SVG directement au sein du code HTML par la suite.



Pour les icônes du site, je me suis servis d'un bibliothèque Open sources [Figure 3: Logo Krita](#) s'appelant [Cocomaterial](#) ([Coco Material](#)) qui est sous licence CC0 1.0 Universal (CC0 1.0)([Voir annexe](#)) ce qui signifie qu'elle peut être utilisée gratuitement à des fins commerciales. Il en est de même pour l'arrière plan en bois des containers que j'ai prise dans une bibliothèque libre de droit d'image s'appelant [Pexels](#).

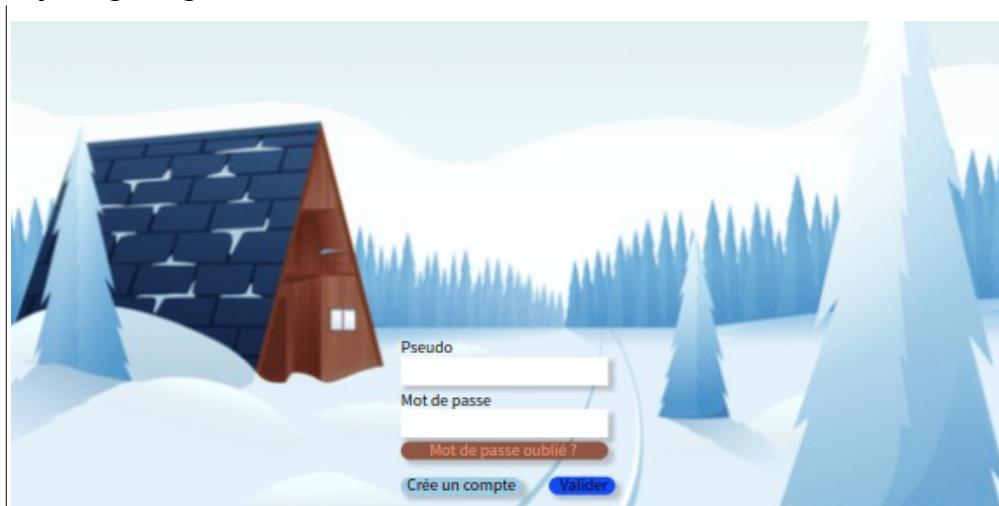


Illustration 2: Maquettage page de login

Pour la suite du projet, j'ai maquetté la page de détail d'un trick en y implantant tout ce qui est demandé au cahier des charges, c'est à dire le nom de la figure, sa description, le groupe, les photos et les vidéos rattachées ainsi que l'espace de discussions.

J'y ai ajouté un menu sidebar à gauche pour plus de lisibilité ainsi qu'une section en dessous indiquant qui a créé la fiche et son historique de modification.

Pour la page de création (Figure 5) ou de modification, j'ai pris exactement la même structure et je l'ai adaptée au besoin, avec la possibilité d'ajouter ou de supprimer des photos ou des vidéos à la volée.

J'ai par la suite simulé la navigation d'un utilisateur en utilisant le principe de prototype de Penpot (similaire Adode XD ou figma) (Annexe :Illustration 26)

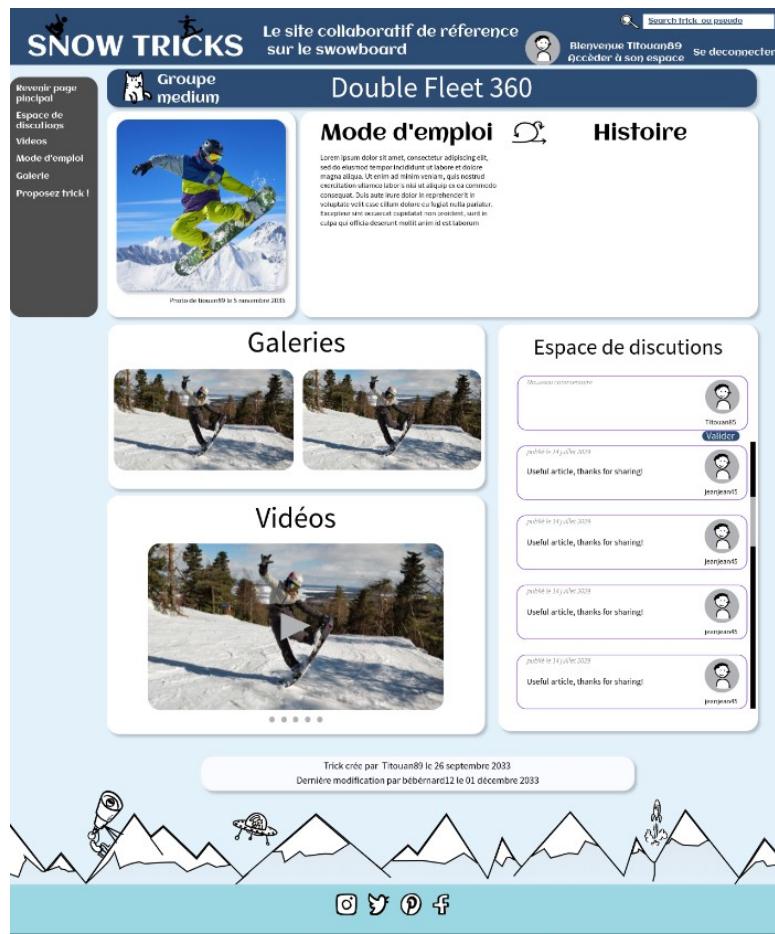


Figure 4: Page détail figure



Figure 5: Page de création/modification figure

On peut voir le résultat à travers cette vidéo (Annexe :Dessin 2)

J'ai totalement conscience que ma mise en page est assez abouti et vu le temps disponible, je n'arriverais pas à avoir véritablement la même apparence mais elle me permet d'imaginer globalement l'apparence du site et d'être clair sur les implantations des éléments.

Création des diagramme UML

Pour générer les différents diagrammes, j'ai utilisé le logiciel [Gaphor](#), logiciel Open-source. C'est un logiciel Open source permettant la génération de diagrammes divers (UML, SysML, RAAML, C4 Model).

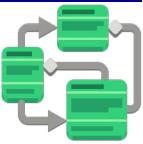


Illustration 3: Logo Gaphor

Diagramme de cas d'utilisation

J'ai partagé ce diagramme en 3 parties :

- Diagramme Login(Annexe :Figure 3)
- Diagramme Tricks(Annexe :Figure 6)
- Diagramme Commentaires(Figure 2)

Je vais parler ici du Diagramme de cas d'utilisation des tricks (Figure 6)

Avec à gauche, les différents acteurs et les flèches entre eux qui signifient les héritages, c'est à dire que l' administrateur peut réaliser certaines actions et également tout ce que peut faire un utilisateur connecté ou non connecté et un utilisateur connecté peut effectuer certaines actions en plus de celles de l'utilisateur non connecté.

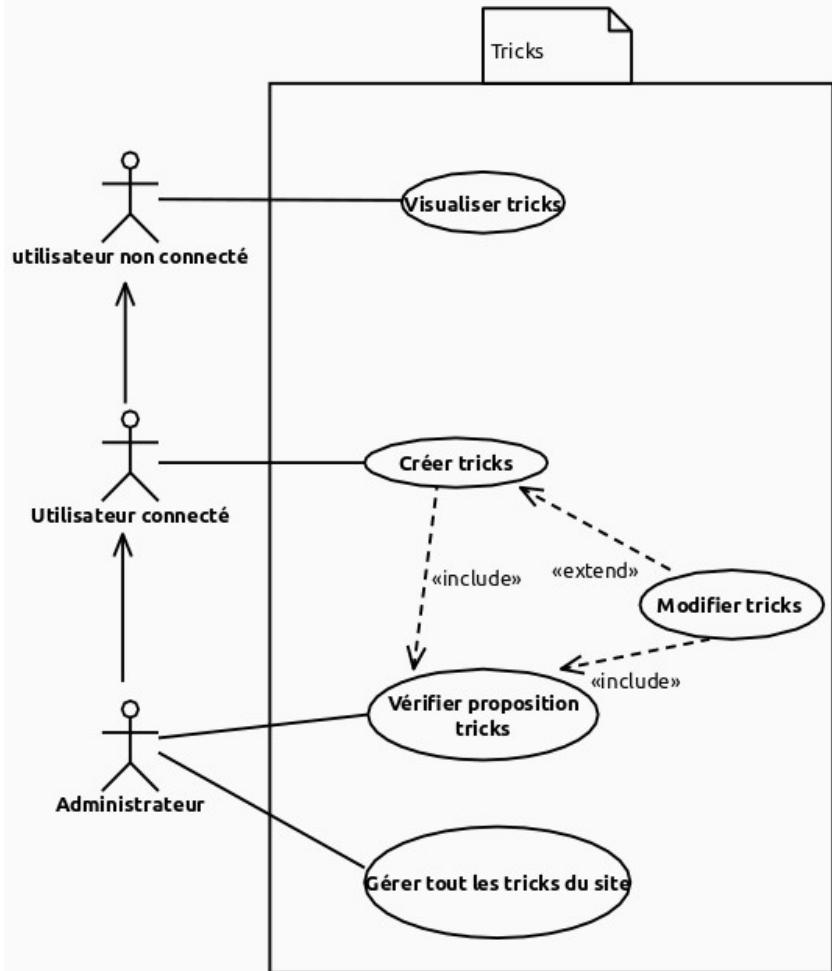


Figure 6: Diagramme cas utilisation Tricks

Concrètement un administrateur peut réaliser tout ce qui se trouve sur ce schéma, un utilisateur connecté peut visualiser et créer un trick et un utilisateur non connecté ne peut visualiser que les différents tricks .

Lors qu'un utilisateur connecté créé un trick ou le modifie, il devra attendre la validation de l'administrateur.

Diagramme d'activité

Pour le diagramme d'activité, j'ai créé 2 diagrammes distincts :

- Un diagramme pour la gestion des commentaires (Annexe Figure 4)
- Un diagramme pour la gestion des tricks (Figure 7)

Je vais vous décrire ce dernier qui va suivre les étapes d'un trick vu par un utilisateur.

Tout utilisateur connecté ou non peut visualiser les tricks et les commentaires liés, s'il veut créer ou modifier l'un d'entre eux, il devra être connecté, soit en se créant un compte, ou alors si problème de connexion, il aura la possibilité de réinitialiser son mot de passe.

Une fois connecté, il aura la possibilité de proposer, modifier ou de supprimer un Trick qui une fois fait, attendra l'approbation de l'administrateur pour la validation de la demande.

Si l'administrateur refuse, alors il enverra une notification de refus à l'utilisateur précisant le pourquoi du refus, si la demande est acceptée alors il recevra une notification d'acceptation et pourra voir en ligne le résultat de sa demande.

Cette gestion d'administration des changements et des ajouts n'est pas demandée au cahier des charges mais j'ai pensé que cela pourrait éviter les abus des utilisateurs et permettra une modération des données accessibles par la suite à toutes personnes naviguant sur le site.

Je n'ai, par contre, pas dans l'idée de mettre cette modération en place pour les commentaires.

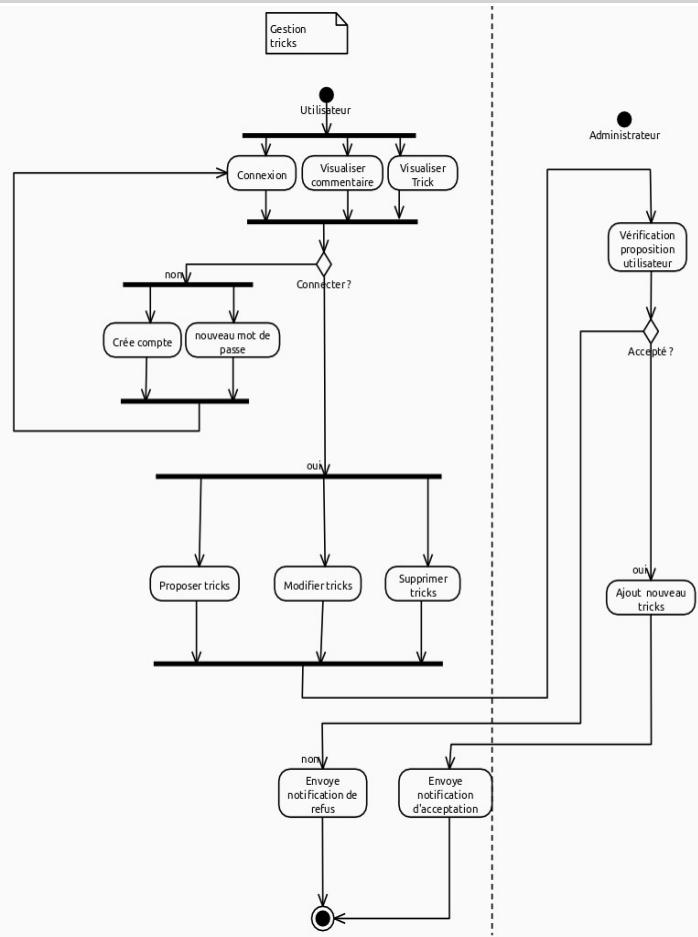


Figure 7: Diagramme D'activité 'gestion trick'

Diagramme de classe

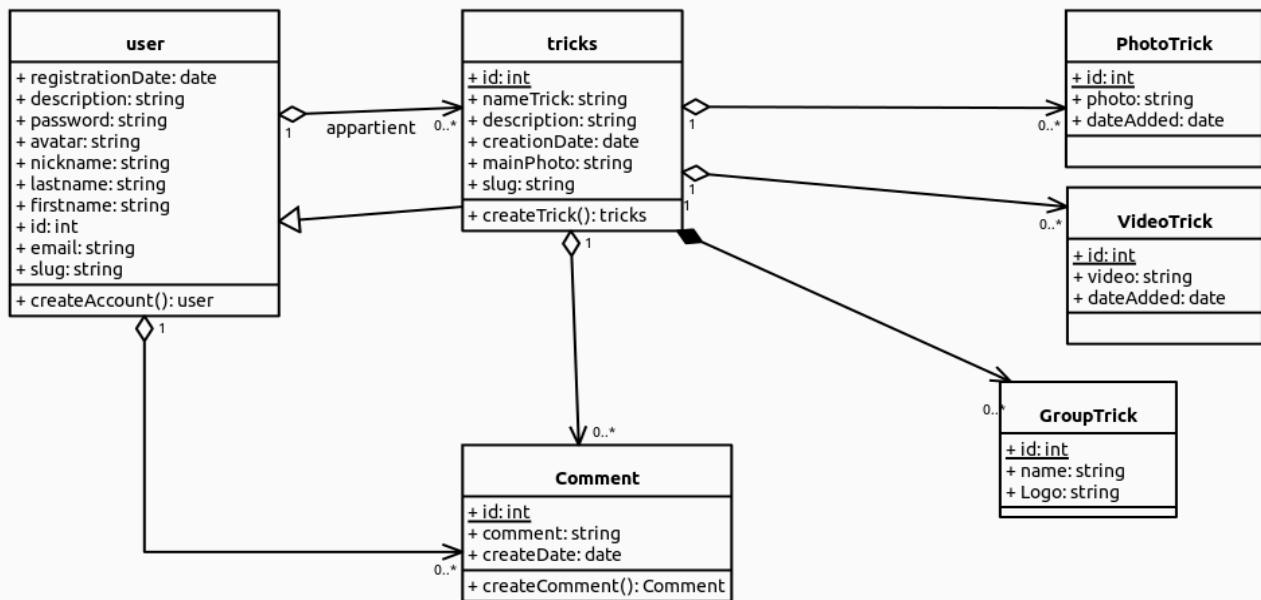


Figure 8: Diagramme de classe SnowTricks

Le diagramme de classe est primordial pour ce type de projet et se doit d'être le plus exact possible dans ses propriétés et surtout les relations entre eux, un framework comme Symfony grâce à sa console permet de générer des entités relatives à ses classes et de gérer les relations entre eux très rapidement et assez facilement (voir chapitre Entité) ce qui facilite grandement l'accès aux données lors la conception.

J'ai créé une classe User avec toutes les propriétés demandées, idem pour la classe Trick.

J'ai décidé de créer des classes à part pour les commentaires, les photos, les vidéos et les groupes car ce seront des relations one to many ou many to many

Exemple : Un trick n'aura qu'un nom (nameTrick) donc cela peut être une propriété propre à sa classe mais un Trick peut posséder plusieurs commentaires donc il fallait externaliser cela dans une classe à part, ce sera une relation many to one, un trick peut posséder plusieurs commentaires mais un commentaire n'appartient qu'à un trick. Idem pour la classe GroupTrick, qui sera une relation many to one avec la classe Trick, c'est à dire qu'un GroupTrick peut posséder plusieurs Tricks mais un Trick ne peut posséder qu'un GroupTrick.

SCSS

Pour la mise en forme, j'ai décidé d'utiliser le préprocesseur SCSS qui permet d'organiser plus facilement son code CSS au sein de plusieurs fichiers distincts et de profiter des possibilités de SCSS comme l'imbrication des sélecteurs et les variables (voir exemple chapitre Responsive).

Dans l'exemple Figure 9: Ex imbrication commentaire, J'ai le sélecteur parent `.trick-comment`, suivi de la sélection de son enfant possédant la classe `.btn` dans lequel j'ai mis un padding de 0 et une font-size de 15px, un autre enfant la balise `a` à qui j'ai mis la propriété `color:black` et une autre balise `H3` à laquelle j'ai également mis quelques propriétés propres à elle seule.

J'ai organisé mon code scss au sein de plusieurs dossiers possédant chacun

leurs fichiers propres (Figure 10: Organisation SCSS). Parmi ceux-ci, dans le dossier base, se trouve le fichier `_colors.scss` qui répertorie au sein de variable la chartre colométrique de l'application (Figure 11: `_colors.scss`). Dans le dossier components, les buttons et les divers composants comme les différentes cartes ou le code gérant la lightbox de la photo principale du trick. Dans le dossier layout, tout ce qui concerne l'affichage global de l'ensemble. Dans le dossier pages, les éléments propres à une page. Et dans sass-utils, tout ce qui est fonction, mixin et variables .

Afin que le code scss soit correctement chargé, je suis maintenant obligée de compiler le code à chaque démarrage soit avec la commande `npm run build`, qui fera une compilation unique soit avec `npm run watch` qui compilera le code scss en temps réel, plus gourmand en ressources mais très utiles lors du développement.

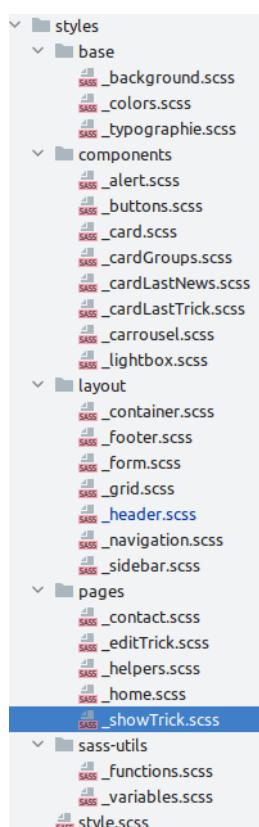


Figure 10:
Organisation SCSS

```
.trick-comment {  
    grid-area: comment;  
    align-items: center;  
    overflow-y: scroll;  
    width: 429px;  
    height: 800px;  
  
    .btn {  
        padding: 0;  
        font-size: 15px;  
    }  
  
    a {  
        color: black;  
    }  
  
    h3 {  
        text-align: center;  
        font-size: 36px;  
        margin-top: 5px;  
        margin-bottom: 5px;  
    }  
}
```

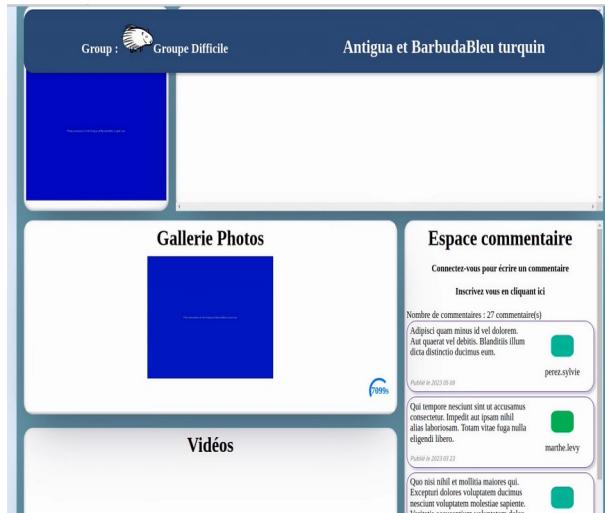
Figure 9: Ex
imbrication
commentaire

```
$color-primary: #0B2C3D;  
$color-background : #e3f1fb;  
$color-secondary: #FF5A3C;  
$color-success: #BADA55;  
$color-danger: #ff0000;
```

Figure 11: `_colors.scss`

Propriété Sticky

Durant la mise en place du css, j'ai trouvé de bon usage d'utiliser la propriété css de positionnement sticky sur le titre du trick afin que celui-ci soit toujours visible lors du défilement de la page (Dessin 1: Vidéos css sticky), l'élément suit son positionnement normal dans le flux mais reste à l'écran quand ce même élément quitte le viewport et avec la propriété top (Figure 12), je lui dis à partir de quelle position elle doit rester à l'écran.



Dessin 1: Vidéos css sticky

```
.trick-title {  
    grid-area: title;  
    position: sticky;  
    z-index: 20;  
    display: flex;  
    flex-wrap: wrap;  
    align-items: baseline;  
    justify-content: space-around;  
    top: 5px;  
    color: white;  
    border-radius: 20px;  
    background: rgba(45, 76, 115, 1);  
    box-shadow: 4px 4px 4px 0px rgba(0, 0, 0, 0.2);  
    justify-items: stretch;
```

Figure 12: css .trick-title

Bouton

Pour la mise en forme des boutons, j'ai créé une classe `.btn` avec différentes propriétés donc la propriété`:active` qui réagit au clique de la souris ou du doigt ce qui lui fait changer sa couleur ainsi que son ombrage.

La propriété `:hover`, lui réagit au passage de la souris et lui fait changer sa couleurs d'arrière plan (`background-color`).



Figure 14: btn état normal



Figure 15: btn état :hover



Figure 16: btn état :active

```
.btn{  
    display: inline-block;  
    padding: 10px 20px;  
    font-size: 20px;  
    cursor: pointer;  
    text-align: center;  
    text-decoration: none;  
    outline: none;  
    color: #fff;  
    background-color: colors.$color-primary;  
    border: none;  
    border-radius: 15px;  
    box-shadow: 0 9px #999;  
    &:active{  
        background-color: #00BAF0;  
        box-shadow: 0 3px #666;  
        transform: translateY(4px);  
    }  
    &:hover{  
        background-color: #00BAF0;  
    }  
}
```

Figure 13: scss .btn

Twig

Afin de développer les templates HTML, j'ai utilisé le moteur de template Twig.

Au fur et à mesure du développement, de la génération de templates par symfony ou de la création de ma part, tous ces fichiers se sont situés dans le dossier templates qui correspond à la partie vue de l'application (le MVC).

Leur dénomination en `_template.twig.html` signifie qu'il ne s'agit pas de page entière à afficher, ce sont soit des composants à appeler, des formulaires, soit des bouts de code avec une utilité propre.

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8"/>
    <meta http-equiv="Content-Type"/>
    <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
    <meta http-equiv="X-UA-Compatible" content="ie=edge"/>
    <meta lang="FR-fr" >
    <title>{% block title %}Bienvenue !{% endblock %}</title>
    <link rel="icon"
        href="data:image/svg+xml,<svg xmlns=%22http://www.w3.org/2000/svg%22></svg>">
    {% block stylesheets %}
        {{ encore_entry_link_tags('app') }}
    {% endblock %}

    {% block javascripts %}
        {{ encore_entry_script_tags('app') }}
    {% endblock %}
    <!-- Matomo -->
    <script...>
    <!-- End Matomo Code -->

</head>
<body>

    {% include 'layout/_header' %}
    <main class="container">
        {% block body %}{% endblock %}
    </main>

    {% include 'layout/_footer' %}
</body>
</html>
```

Figure 18: `base.html.twig`

La template de base de l'application est le fichier `base.html.twig`(Figure 18) composé de la balise `<head>` contenant toutes les méta données et le titre à afficher entre autres, elle est suivie du `<body>` dans lequel, j'ai inclus des éléments twig comme le `_header`(annexe Figure 12) et le `_footer`(annexe Figure 11)

Grâce à cette configuration, cette base ne change jamais quelque soit la page du site visitée et seule la partie entre ces 2 chevrons changera `{% block body %}{% endblock %}`.

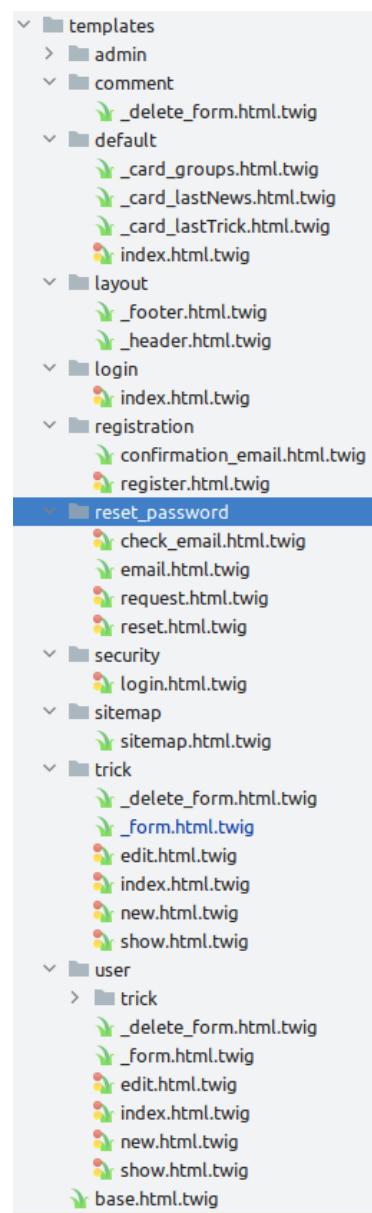


Figure 17: Dossier template

Je vais prendre l'exemple de la base d'accueil qui grâce à cette configuration permet d'alléger le code et ainsi de le rendre plus clair.

Dans celui ci (Figure 19), on peut voir que ce fichier est lié a `base.html.twig` avec

```
{% extends 'base.html.twig' %}
```

Un titre unique à cette page entre les balises

```
{% block title %}Accueil Snowtricks!{% endblock %}
```

L'Extends permet d'appeler les balises

```
{% block body %}
```

```
{% endblock %}
```

présent dans `base.twig.html` et d'y inclure le code voulu.

Pour cette page, j'y ai inclus des composants avec les balises

```
{% include 'default/_card_lastTrick.html.twig' %}
```

```
{% extends 'base.html.twig' %}

{% block title %}Accueil Snowtricks!{% endblock %}

{% block body %}

<section class="container__homepage--top">
    {% include 'default/_card_lastTrick' %}
    {% include 'default/_card_lastNews' %}

</section>

<section class="container__homepage--bottom">
    {% include 'default/_card_groups' %}
</section>
```

Figure 19: (page d'accueil) `index.html.twig`

```
<footer class="article article__footer--lastTrick">
    <h4 class="article article__user">Ajouté le {{ LastTrick.creationDate.format('Y m d') }}
        par {{ LastTrick.user.nickname }}</h4>
    <h2 class="article article__nameTrick">{{ LastTrick.nameTrick }}</h2>
    <span class="article article__Group">{{ LastTrick.groupTrick.name }}</span>
</footer>
```

Figure 20: footer de `_card_last_trick.html.twig`

Une autre propriété des template twig est l'appel à des fonctions via `{%...%}` et des variables entre `{...}`. Dans l'exemple (Figure 20), je fais appel à un objet `$lastTrick` envoyé par le contrôleur et récupère des éléments de celui ci comme `{{ LastTrick.creationDate.format('d/m/Y') }}` et `{{ LastTrick.user.nickname }}` ou `{{ LastTrick.nameTrick }}`.

```
<div class="content trick-comment">
    <h3>Espace commentaire</h3>
    <div class="spaceComment ">
        {% if app.user %}
            <div>
                {{ form_start(form) }}
                {{ form_end(form) }}
            </div>
        {% else %}
            <div class="spaceComment__content--subscribe">
                <a href="{{ path('/login') }}"><h4>Connectez-vous pour écrire un commentaire</h4></a>
                <a href="{{ path('/register') }}"><h4>Inscrivez vous en cliquant ici</h4></a>
            </div>
        {% endif %}
```

Figure 21: Condition twig commentaire

Dans l'exemple (Figure 21), je vérifie si un user est connecté `{% if app.user %}` et affiche un résultat en conséquence ou un autre avec un `{% else %}` en n'oubliant pas de finir cette conditions avec un `{% endif %}`.

Responsive

Avec l'aide des Media Queries, j'ai pu mettre en place le css correspondant aux affichages Desktop, tablette et smartphone. Les valeurs sont stocker en tant que variable dans le fichier variables.scss (Figure 22) et modifiable sur tout l'ensemble du code scss simplement en changeant les valeurs dans ce fichier.

Pour gérer l'affichage des containers (un terme souvent utilisé pour parler d'une balise html contenant un ensemble de contenu) dans le fichier _container.scss, j'ai utilisé les variables nommées plus haut afin que leurs affichages suivent la taille de l'écran, comme dans l'exemple de la Page d'accueil (Figure 23: Polypane Page d'accueil)

- Sans média Queries, la largeur (width) du container serait égale à la variable \$breakpoint-desktop
- Arrivé à la variable \$breakpoint-desktop, la largeur passe à la variable \$breakpoint-tablet
- Arrivé à la variable \$breakpoint-tablet, la largeur passe à la variable \$breakpoint-smartphone
- Enfin arrivé au \$breakpoint-smartphone, la largeur passe à 100 % de l'écran

```
$breakpoint-smartphone: 390px;  
$breakpoint-tablet: 950px;  
$breakpoint-desktop: 1280px;
```

Figure 22: variables.scss

```
@use '../sass-utils/variables';  
  
.container {  
    width: variables.$breakpoint-desktop;  
    margin: 0 auto;  
    background-color: #689ca6;  
}  
  
@media screen and (max-width: variables.$breakpoint-desktop) {  
    .container {  
        width: variables.$breakpoint-tablet;  
    }  
}  
  
@media screen and (max-width: variables.$breakpoint-tablet) {  
    .container {  
        width: variables.$breakpoint-smartphone;  
    }  
}  
  
@media screen and (max-width: variables.$breakpoint-smartphone) {  
    .container {  
        width: 100%;  
    }  
}
```

Illustration 4: _container.scss

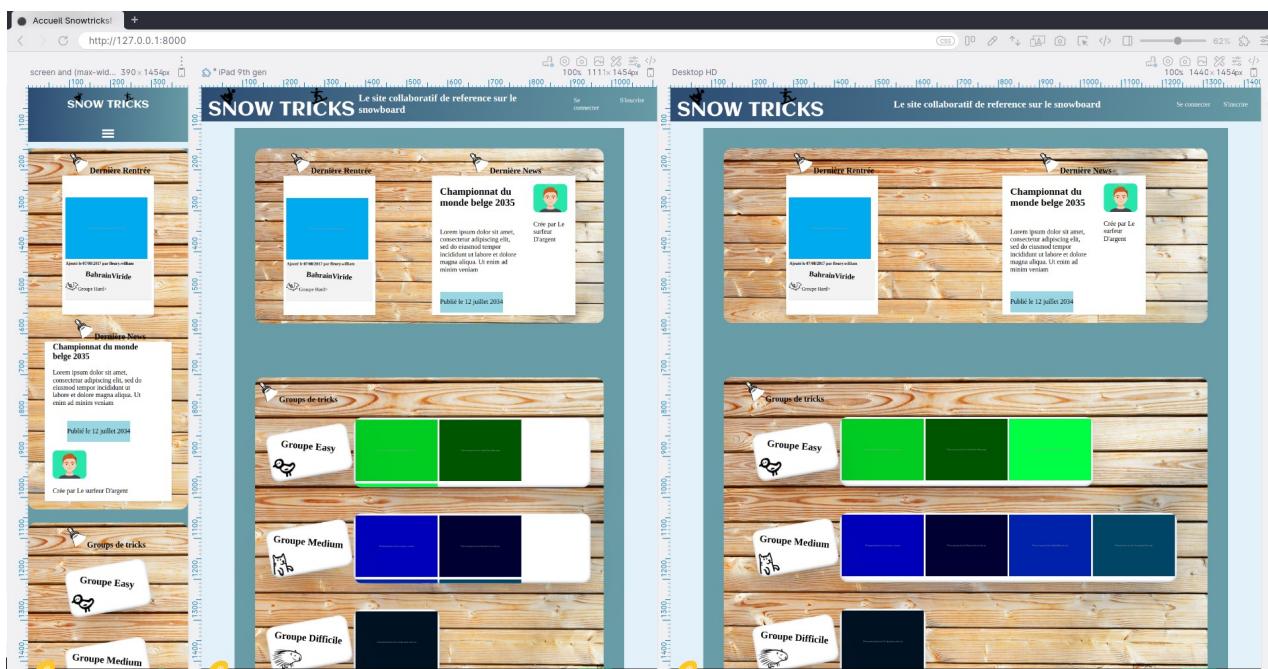


Figure 23: Polypane Page d'accueil

Un autre exemple est la page de détails d'un trick(Figure 24).

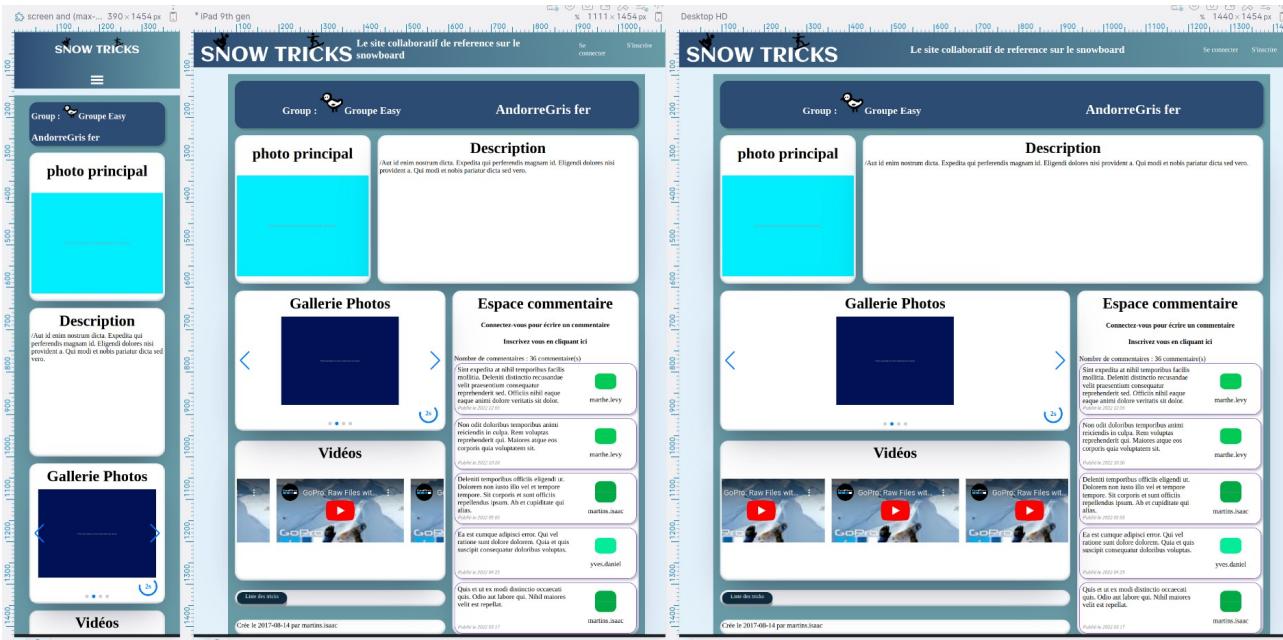


Figure 24: Polypane page détail trick

On peut remarquer qu'avec l'aide du gestionnaire layout css GRID (Figure 25), j'ai changé l'organisation de la page, en donnant, par exemple à la photo principale et au commentaire une hauteur et une largeur fixe, et seulement une hauteur fixe pour les autres éléments, ce qui donne des largeurs automatisées suivant la largeur du viewport (zone d'affichage du navigateur).

Pour l'affichage en mode smartphone (Figure 26), j'ai changé l'organisation avec l'aide de la propriété Grid template et nommant chaque conteneur avec Grid area .

```
@media screen and (max-width: variables.$breakpoint-tablet) {
  .container-show-trick {
    justify-items: center;
    grid-template-columns: minmax(auto, 250px) minmax(250px, 1fr);
    grid-template:
      "title title"
      "photo photo"
      "descr descr"
      "gallery gallery"
      "videos videos"
      "comment comment"
      "button button"
      "footer footer"
  }
}
```

Figure 26: Grid template mode tablet

```
.container-show-trick {
  display: grid;
  justify-items: stretch;
  padding: 1rem;
  row-gap: 1rem;
  column-gap: 1rem;
  grid-template:
    "title title title"
    "photo descr descr"
    "gallery gallery comment"
    "videos videos comment"
    "comment comment"
    "button button comment"
    "footer footer comment"
}
```

Figure 25: Grid area mode desktop

Back-end

Création de l'architecture symfony

Installation

Après avoir versionné mon dossier de projet (Voir chapitre Git), J'ai ouvert un terminal bash, je me suis situé dans ce même dossier et j'ai tapé cette commande :

```
composer create-project symfony/skeleton:"6.2.*" snowtricks --webapp
```

L'option webapp permet d'installer, en même temps, tout un ensemble de packages parmi ceux les plus nécessaires à une application web (Doctrine, form, mailer, maker-bundle, notifier, profiler, twig...etc).

Une fois cela fait, j'ai lancé la commande

```
php -S localhost:8000 -t public/ afin de visualiser si l'installation s'est bien déroulée dans un navigateur. je me retrouve avec un environnement de travail prêt et structuré prêt à être commité pour la 1ère fois en effectuant dans le terminal un git commit -m 'first commit create skeleton with webapp'
```

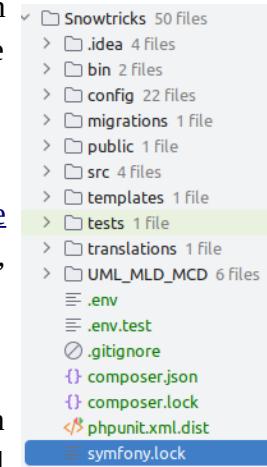


Figure 1: Création projet

Entité

Ma première action a été de créer les différentes entités à l'aide du package maker-bundle de symfony (Comment, GroupTrick, PhotoTrick, Trick, User, VideoTrick)

Par exemple pour l'entité Trick, j'ai rentré dans la console :
php bin/console make:entity Trick et au fur et à mesure, elle me demandait le nom de la propriété que je voulais lui apporter ainsi que son type (String, Date, float, booléen..), sa longueur (si string) et si la valeur pouvait être nulle, jusqu'à la proposition d'une nouvelle propriété et comme cela ainsi de suite.

Symfony a ainsi généré la classe correspondante (voir annexe Entity Trick) dans le dossier entity ainsi que le repository (voir annexe TrickRepository), c'est à dire le pont avec la base de données, dans le dossier repository.

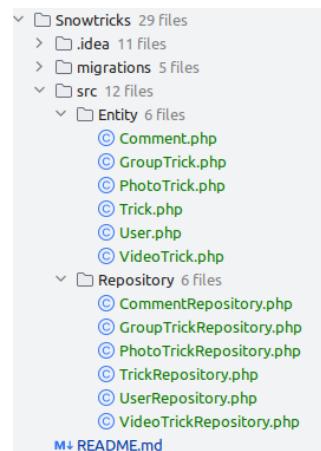


Figure 2: Création des Entités

Relation

Une fois toutes les entités générées, j'ai contextualisé leurs relations en retapant la même commande que précédemment pour leur rajouté une propriété mais cette fois de type Relationships/Associations (Figure 3) et étape après étape, j'ai lié les entités entre elles en faisant très attention à l'orthographe et surtout au singulier et au pluriel des mots .

Mon IDE PhpStorm à un outil qui permet de représenter visuellement les entités du projet sous la forme d'une diagramme, cela m'a permis de réaliser une dernière vérification avant la génération de la base de données qui est l'étape suivante (Figure 4). Je peux ainsi vérifier que par exemple un **User** possède des **comments** et d'un **Comment** possède un **user** , un **Trick** possède des **photoTricks** et une **PhotoTrick** possède un **Trick**. La première lettre majuscule ou minuscule et le pluriel /singulier de chaque mot est très important.

Les relations peuvent être vérifiées et validées.

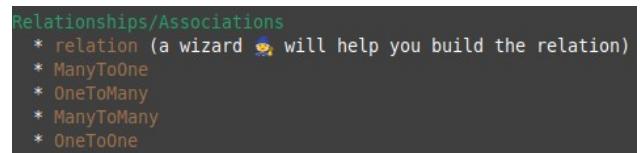


Figure 3: Choix de relation maker symfony

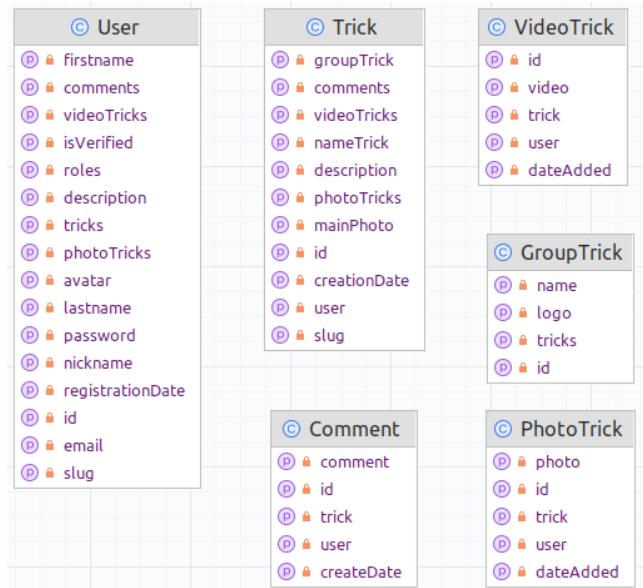


Figure 4: Diagramme Entities via PhpStorm

Doctrine

Une fois les relations et les entités validées, il reste à les créer en base de données.

Durant les cours, nous avons créé des bases de donnée MySQL, je suis donc partis sur ce système. Pour cela, il me fallait configurer symfony pour qu'il utilise ce système, cette option se fait au niveau du fichier `.env` situé à la base de projet. Il y a plusieurs lignes à décommenter au fonction de ce que l'on recherche(Figure 5), on peut voir que symfony a déjà prévu 3 lignes déjà écrites pour sqlite, mysql et postgresql, des bases de données que l'on retrouvent très souvent dans les projets actuels.

```
# DATABASE_URL="sqlite:///%kernel.project_dir%/var/data.db"
DATABASE_URL="mysql://max:@127.0.0.1:3306/snowtricks?serverVersion=8&charset=utf8mb4"
# DATABASE_URL="postgresql://app:!ChangeMe!@127.0.0.1:5432/app?serverVersion=15&charset=utf8"
```

Figure 5: Connection à la Database symfony

Je personnalise cette commande avec la bonne URL, les bons identifiants, et le nom de la base que je désire.

Dorénavant, je vais pouvoir grâce à doctrine créer, modifier, mettre à jour ou supprimer cette base de données avec des commandes à rentrer dans la console de symfony, concrètement dans l'ordre :

1. Je crée ma base de données avec cette commande `php bin/console doctrine:database:create`
2. Je crée le fichier de migration avec `php bin/console make:migration` (annexe Exemple de fichier de migration Doctrine), l'ensemble du script SQL nécessaire à l'application.
3. J'effectue la migration vers la base de données avec `php bin/console doctrine:migrations:migrate`

```
maxime@MaxVictus:~/Documents/dwmm/Projet snowTrick/Snowtricks (Develop)$ php bin/console make:migration

Success!

Next: Review the new migration "migrations/Version20230420093552.php"
Then: Run the migration with php bin/console doctrine:migrations:migrate
See https://symfony.com/doc/current/bundles/DoctrineMigrationsBundle/index.html
maxime@MaxVictus:~/Documents/dwmm/Projet snowTrick/Snowtricks (Develop)$ php bin/console d:m:m

WARNING! You are about to execute a migration in database "snowtricks" that could result in schema changes and data loss. Are you sure you wish to continue? (yes/no) [yes]: yes

[notice] Migrating up to DoctrineMigrations\Version20230420093552
[notice] finished in 20.3ms, used 20M memory, 1 migrations executed, 1 sql queries
[OK] Successfully migrated to version : DoctrineMigrations\Version20230420093552
```

Figure 6: Doctrine migration

Après ceci, je peux aller vérifier dans la base ce qu'a donné cette migration grâce à des outils comme PhpMyAdmin, DBeaver(Figure 7), MySQLWorkbench qui viennent se connecter sur les bases de données et qui possèdent une liste d'outil de visualisation.

On peut ainsi vérifier si la migration s'est bien déroulée en la comparant avec le précédent diagramme des entités (Figure 4).

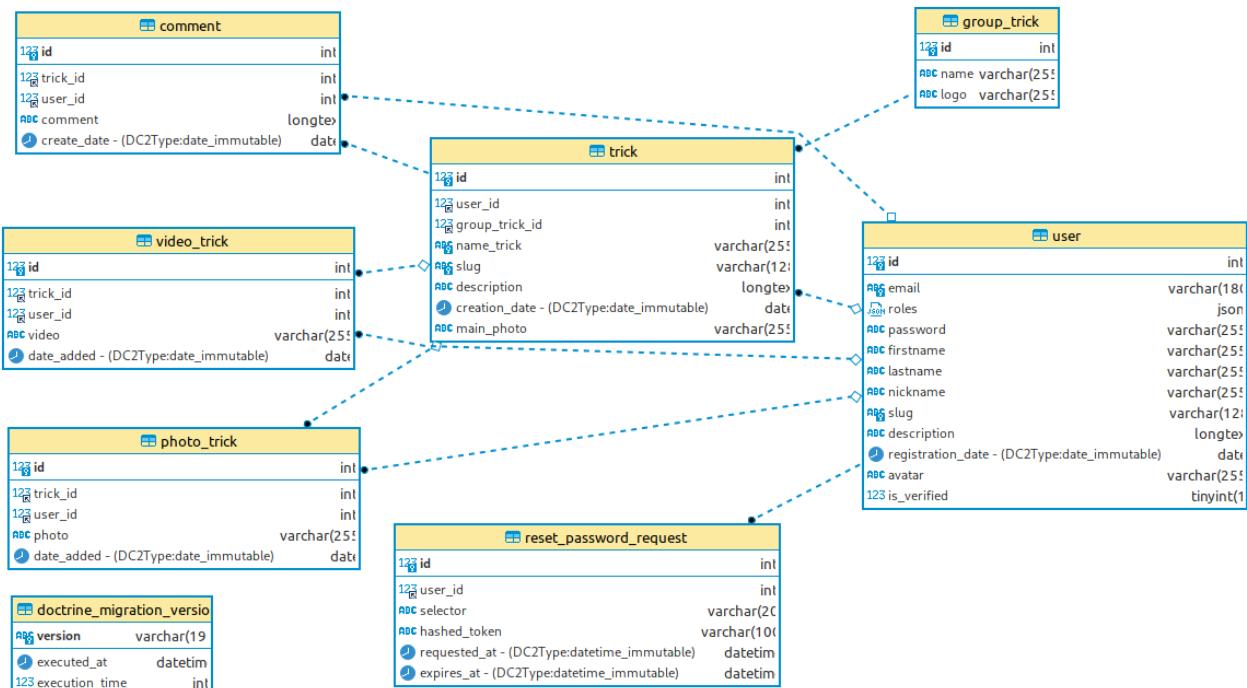


Figure 7 : Diagramme BDD snowtricks

Controller

Maintenant que la structure des données est faite, il faut générer les routes et cela va se faire avec une commande aux multiples utilités, la commande crud.

Celle-ci va générer à partir d'une entité toute une série de fichiers donc le controller lié à la classe, le formulaire qui permettra la création, la modification et la suppression ainsi que les templates Twig pour l'affichage dans un navigateur, c'est une commande très utile qui accélère grandement le processus et qui permet

d'avoir une hiérarchisation des routes de l'entité *Figure 8: php bin/console make:crud*

New>Show/Edit/Delete certe basique mais répondant dans la plus grande majorité des cas au besoins voulus.

Comme exemple, je vais parler de l'entité User (*Figure 8*) et de Trick que j'ai générée avec la commande *php bin/console make:crud*, celle-ci m'a généré et rangé les fichiers liés aux entités aux endroits correspondant et a effectué les liens nécessaires entre tous ces fichiers.

- Le controller avec les routes (Annexe Figure 8)
- Le formulaire lié à l'entité permettant l'ajout ou la modification(Figure 10)
- Le fichier de test correspondant (expérimental)
- Les différents templates :
 - Index(Figure 11) / Edit / new / Show
 - _delete et _form, sont 2 templates appelé sur les autres templates et non destiné à être affiché tel quel.

```
class TrickType extends AbstractType
{
    no usages
    public function buildForm(FormBuilderInterface $builder, array $options): void
    {
        $builder
            ->add('nameTrick')
            ->add('description')
            ->add('creationDate')
            ->add('user')
            ->add('groupTrick')
    }

    no usages
    public function configureOptions(OptionsResolver $resolver): void
    {
        $resolver->setDefaults([
            'data_class' => Trick::class,
        ]);
    }
}
```

Figure 10: Formulaire trick crud

```
maxime@MaxVictus:~/Documents/dwww/Projet snowTrick/Snowtricks [Develop]$ php bin/console make:crud

The class name of the entity to create CRUD (e.g. OrangePopsicle):
> User

Choose a name for your controller class (e.g. UserController) [UserController]:
>

Do you want to generate tests for the controller?. [Experimental] (yes/no) [no]:
> yes

created: src/Controller/UserController.php
created: src/Form/UserType.php
created: templates/_delete_form.html.twig
created: templates/_form.html.twig
created: templates/_edit.html.twig
created: templates/_index.html.twig
created: templates/_new.html.twig
created: templates/_show.html.twig
created: tests/Controller/UserControllerTest.php

Success!
```

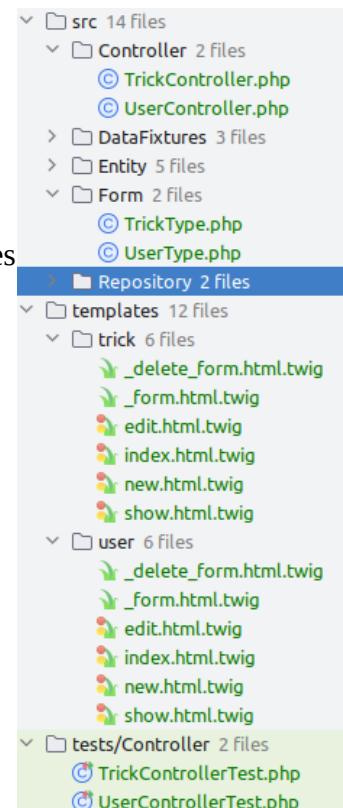


Figure 9: Crud User et Trick

```

{% extends 'base.html.twig' %}

{% block title %}tricks index{% endblock %}

{% block body %}
    <h1>tricks index</h1>

    <table class="table">
        <thead>
            <tr>
                <th>Id</th>
                <th>NameTrick</th>
                <th>Description</th>
                <th>CreationDate</th>
                <th>actions</th>
            </tr>
        </thead>
        <tbody>
            {% for trick in tricks %}
                <tr>
                    <td>{{ trick.id }}</td>
                    <td>{{ trick.nameTrick }}</td>
                    <td>{{ trick.description }}</td>
                    <td>{{ trick.creationDate ? trick.creationDate|date('Y-m-d') : '' }}</td>
                    <td>
                        <a href="{{ path('app_trick_show', {'id': trick.id}) }}">show</a>
                        <a href="{{ path('app_trick_edit', {'id': trick.id}) }}">edit</a>
                    </td>
                </tr>
            {% else %}
                <tr>
                    <td colspan="5" no records found</td>
                </tr>
            {% endif %}
        </tbody>
    </table>

```

Figure 11: index.twig.html Entité Trick

```

$builder
    ->add( child: 'nameTrick', type: TextType::class)
    ->add( child: 'description', type: TextareaType::class)
    ->add( child: 'mainPhoto', type: FileType::class,
        'data_class' => null,
        'attr'=>['onchange'=>'readURL(this);']
    )
]

```

Figure 12: Personnalisation des champs du formulaire Trick

Après quelques ajustements au niveau du formulaire en indiquant bien le style de form voulu à chaque propriété (Figure 12),

On arrive à avoir un affichage dans un navigateur fonctionnel, j'ai fais par la suite des essais d'ajout de trick, de modification et de suppression pour valider la bonne marche de l'ensemble, j'ai fais de même pour l'entité User.

Profiler

Pour nous aider durant la conception de l'application, Symfony nous fournit une aide avec leur profiler, c'est une console de débug graphique qui possède, même réduit, une foule d'informations comme l'identité de la personne connecté, le nombre de requêtes SQL qui réclame l'ouverture d'une page, le temps de chargement, le chemin actuel de l'URL et d'autres informations.

Figure 13

Une fois totalement ouvert, l'outil nous permet l'accès à tout un autre ensemble d'informations comme une timeline sur les différents événements qui surviennent lors du chargement de la page, les logs, le routing avec toute les routes disponibles et leur URL, la gestion du cache, toutes les informations concernant twig comme le chargement des éléments twig, et toutes les requêtes SQL en détails nécessaires à la page et également d'autres parties comme la translation, la sécurité, le système de notification de symfony.

Figure 14: Profiler symfony defaultController::index

Quand la page ne peut pas s'afficher, il exprime assez clairement le pourquoi et nous propose souvent une solution ou une partie de solution afin de nous aider.

Fixture

Les Fixtures me permettent de remplir le site de données fictives, fausses mais utiles pour maîtriser l'affichage des éléments. Grâce aux boucles, on peut en créer un grand nombre et voir ainsi comment notre application réagit en terme de performance et d'affichage.

Pour le besoin de site, j'en ai créé pour chaque entité (Figure 15). Pour la *Figure 15: Liste des fixtures*

J'ai créé dans un premier temps les Users (Illustration 5) en créant un admin et 4 users, je leur ai attribué toutes les propriétés qu'un user doit posséder.

J'ai fais de même pour les tricks (Illustration 6) à la différence que j'ai utilisé une boucle pour en générer le nombre voulu et grâce à l'injection de dépendances aux autres Fixtures, j'ai pu planter les Fixtures groupe ainsi que les Users et attribuer à chaque boucle un groupe ainsi qu'un user au hasard grâce à la fonction arrayRand de Php. J'ai fais de même pour les autres entités, j'ai ensuite pu les générer avec la console en tapant *php bin/console doctrine:fixtures:load*.

© CommentFixtures.php
© GroupFixtures.php
© PhotoTrickFixtures.php
© TrickFixtures.php
© UserFixtures.php
© VideoTrickFixtures.php

```
public function __construct(UserPasswordHasherInterface $hasher)
{
    $this->faker = Factory::create(['locale' => 'fr_FR']);
    $this->hasher = $hasher;
}

± Maxime Lemée
public function load(ObjectManager $manager): void
{
    $admin = new User();
    $admin->setFirstname('admin')->setLastname('admin')
        ->setNickname($this->faker->userName)
        ->setEmail('admin@mail.com')
        ->setPassword($this->hasher->hashPassword($admin, plainPassword: 'password'))
        ->setAvatar($this->faker->image(['dir' => 'public/uploads', width: 640, height: 480, category: null, fullPath: false]))
        ->setRegistrationDate(new \DateTimeImmutable($this->faker->date()))
        ->setDescription($this->faker->paragraph)->setIsVerified(isVerified: true)
        ->setRoles(['ROLE_ADMIN']);
    $manager->persist($admin);

    $user1 = new User();
    $user1->setFirstname($this->faker->firstName)->setLastname($this->faker->lastName)
        ->setNickname($this->faker->userName)
        ->setEmail('max@mail.com')
        ->setPassword($this->hasher->hashPassword($admin, plainPassword: 'password'))
        ->setAvatar($this->faker->image(['dir' => 'public/uploads', width: 150, height: 150, category: null, fullPath: false]))
        ->setRegistrationDate(new \DateTimeImmutable($this->faker->date()))
        ->setDescription($this->faker->paragraph)->setIsVerified(isVerified: true)
        ->setRoles(['ROLE_USER']);
    $manager->persist($user1);
    $this->addReference('user1', $user1);
}
```

Illustration 5: Fixture User

```
public function load(ObjectManager $manager): void
{
    $groupeEasy = $this->getReference('groupe-easy');
    $groupeMedium = $this->getReference('groupe-medium');
    $groupeDifficile = $this->getReference('groupe-difficile');
    $groupeHard = $this->getReference('groupe-hard');
    $groupArray = [$groupeEasy, $groupeMedium, $groupeDifficile, $groupeHard];

    $user1 = $this->getReference('user1');
    $user2 = $this->getReference('user2');
    $user3 = $this->getReference('user3');
    $user4 = $this->getReference('user4');
    $userArray = [$user1, $user2, $user3, $user4];

    for ($i = 1; $i <= 10; $i++) {
        $groupArrayRand = array_rand($groupArray, num: 1);
        $userArrayRand = array_rand($userArray, num: 1);
        $trick = new Trick();
        $trick->setNameTrick(nameTrick: $this->faker->country . $this->faker->colorName)
            ->setDescription($this->faker->text)
            ->setCreationDate(new \DateTimeImmutable($this->faker->date()))
            ->setGroupTrick($groupArray[$groupArrayRand])
            ->setMainPhoto($this->faker->image(['dir' => 'public/uploads', width: 640, height: 480, category: null, fullPath: false]))
            ->setUser($userArray[$userArrayRand]);
        $manager->persist($trick);
        $this->addReference('trick' . $i, $trick);
    }

    $manager->flush();
}

no usages ± Maxime Lemée
public function getDependencies(): array
{
    return [GroupFixtures::class, UserFixtures::class];
}
```

Illustration 6: Fixture Trick

Fonctionnalités

Page d' inscription

Pour permettre l'inscription d'un nouvel utilisateur, grâce à la commande `php bin/console make:registration-form` (Figure 1), symfony m'a généré tout le nécessaire comme le formulaire d'enregistrement, le template associé ainsi que le controller(Figure 5), et tout cela avec la sécurité nécessaire. J'ai par la suite programmer la mise en forme sur l'Illustration 7, j'y ai implanté un éditeur WYSIWYG (What You See Is what You Get) qui remplace le TextArea classique par un éditeur plus riche, je l'ai implanté via Composer, activé dans les paramètres de l'application et ensuite implanté au niveau de formulaire à l'endroit désiré (Figure 2).

```
maxime@MaxVictus:~/Documents/dwm/Projet snowTrick/Snowtricks (Develop)$ php bin/console make:registration-form
Creating a registration form for App\Entity\User
Do you want to add a @UniqueEntity validation annotation on your User class to make sure duplicate accounts aren't created? (yes/no) [yes]:
>

Do you want to send an email to verify the user's email address after registration? (yes/no) [yes]:
>

[WARNING] We're missing some important components. Don't forget to install these after you're finished.
composer require symfonycasts/verify-email-bundle

By default, users are required to be authenticated when they click the verification link that is emailed to them.
This prevents the user from registering on their laptop, then clicking the link on their phone, without
having to log in. To allow multi device email verification, we can embed a user id in the verification link.

Would you like to include the user id in the verification link to allow anonymous email verification? (yes/no) [no]:
>

What email address will be used to send registration confirmations? (e.g. mailer@your-domain.com):
> no-reply@snowtrick.com

What "name" should be associated with that email address? (e.g. Acme Mail Bot):
> No Reply

Do you want to automatically authenticate the user after registration? (yes/no) [yes]:
>

updated: src/Entity/User.php
updated: src/Entity/User.php
created: src/Security/EmailVerifier.php
created: templates/registration/confirmation_email.html.twig
created: src/Form/RegistrationFormType.php
created: src/Controller/RegistrationController.php
created: templates/registration/register.html.twig
```

Figure 1: `make:registration-form`

Durant l'installation du registration-form, Symfony m'a proposé de passer par leur service mailer afin de valider l'adresse mail, ce que j'ai fait, de ce fait j'ai installé via docker Maildev et précisé dans le fichier .env de symfony la configuration suivante

`MAILER_DSN=smtp://127.0.0.1:1025` ce qui permet à maildev d'intercepter tous les mails sortant envoyés par l'application.

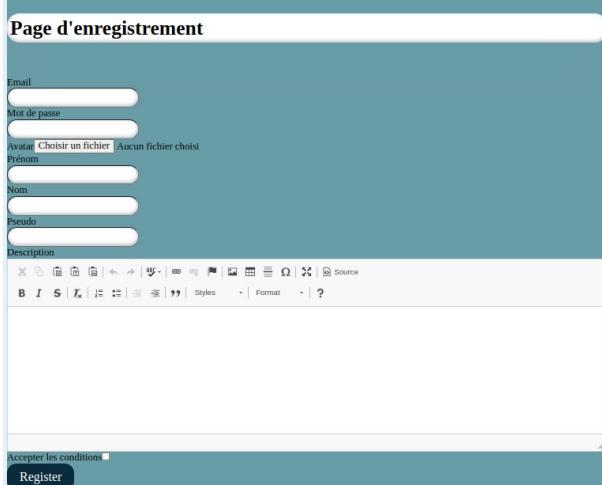


Illustration 7: template registration Form

```
->add( child: 'description', type: CKEditorType::class, [
    'label' => 'Description',
    'attr' => [
        'class' => 'content'
    ]
])
```

Figure 2: Type `CkEditor` dans un form

L'utilisateur remplit les champs, reçoit les alertes s'il ne remplit pas les bonnes conditions, conditions que l'on donne soit directement dans les entités au niveau des propriétés avec des Assert (Figure 3),

```
#[Assert\Length(
    min: 2,
    max: 200,
    minMessage: 'Le commentaire est trop court, rallonge un peu !',
    maxMessage: 'Oooh trop long !',
)]
#[ORM\Column(type: Types::TEXT)]
private ?string $comment = '';
```

Figure 3: Exemple d'assert pour un commentaire

soit dans les formulaires en donnant des contraintes en paramètres (Figure 4). Une fois toutes les conditions remplies, l'utilisateur valide, à ce moment là, symfony effectue dans le controller associé toute une série d'action donc un hachage du password et une entrée en base de données avec un persist des données suivies d'un push via doctrine, par la suite, il envoie le mail de vérification avec le template associé à l'adresse de l'utilisateur, celui-ci confirme sa réception en cliquant sur un lien qui le renvoie sur la page *verify/email*(Figure 6)

```
#[Route('/register', name: 'app_register')]
public function register(Request $request, UserPasswordEncoderInterface $userPasswordHasher, UserAuthenticator $userAuthenticator, EntityManagerInterface $entityManager)
{
    $user = new User();
    $form = $this->createForm(RegistrationFormType::class, $user);
    $form->handleRequest($request);

    if ($form->isSubmitted() && $form->isValid()) {
        // encode the plain password
        $user->setPassword(
            $userPasswordHasher->hashPassword(
                $user,
                $form->get('plainPassword')->getData()
            )
        );
    }

    $entityManager->persist($user);
    $entityManager->flush();

    // generate a signed url and email it to the user
    $this->emailVerifier->sendEmailConfirmation(verifyEmailRouteName: 'app_verify_email', $user,
        (new TemplatedEmail())
            ->from(new Address(address: 'no-reply@snowtrick.com', name: 'No Reply'))
            ->to($user->getEmail())
            ->subject('Confirme ton Email SNOWTRICKS')
            ->htmlTemplate(template: 'registration/confirmation_email.html.twig')
    );
    // do anything else you need here, like send an email

    return $userAuthenticator->authenticateUser(
        $user,
        $authenticator,
        $request
    );
}

return $this->render('registration/register', [
    'registrationForm' => $form->createView(),
]);
}
```

Figure 5: RegistrationController

```
->add( child: 'plainPassword', type: PasswordType::class, [
    // instead of being set onto the object directly,
    // this is read and encoded in the controller
    'mapped' => false,
    'attr' => ['autocomplete' => 'nouveau-mot-de-passe'],
    'label' => 'Mot de passe',
    'attr' => [
        'class' => 'content'
    ],
    'constraints' => [
        new NotBlank([
            'message' => 'Veuillez rentrez votre mot de passe'
        ]),
        new Length([
            'min' => 6,
            'minMessage' => 'Your password should be at least {{ limit }} characters',
            // max length allowed by Symfony for security reasons
            'max' => 4096,
        ]),
    ],
),
]
```

Figure 4: Form Registration password

qui vérifie ainsi si la signature de l'URL correspond bien, si c'est le cas, un message flash apparaît pour indiquer que le mail a bien été vérifié, Symfony finalise en cochant l'user à l'état *is_verified* en base de données)

	id	is_verified	roles	email
1	166	1	["ROLE_ADMIN"]	admin@mail.com
2	167	1	["ROLE_USER"]	max@mail.com

Figure 7: Colonne *is_verified*

Page de connexion

Maintenant que l'utilisateur possède un compte et que celui là est vérifié, il lui faut une page d'inscription afin de se connecter.

Pour cela, symfony à une commande via son maker_bundle qui est la suivante *php bin/console make:controller Login*.

```
#[Route('/verify/email', name: 'app_verify_email')]
public function verifyUserEmail(Request $request, TranslatorInterface $translator): Response
{
    $this->denyAccessUnlessGranted(attribute: 'IS_AUTHENTICATED_FULLY');

    // validate email confirmation link, sets User::isVerified=true and persists
    try {
        $this->emailVerifier->handleEmailConfirmation($request, $this->getUser());
    } catch (VerifyEmailExceptionInterface $exception) {
        $this->addFlash(type: 'verify_email_error', $translator->trans($exception->getReason(), [], 'VerifyEmailBundle'));

        return $this->redirectToRoute(route: '/register');
    }

    // @TODO Change the redirect on success and handle or remove the flash message in your templates
    $this->addFlash(type: 'success', message: 'Your email address has been verified.');

    return $this->redirectToRoute(route: '/');
}
```

Figure 6: RegistrationController /verify/mail

```

class LoginController extends AbstractController
{
    // Maxime Lemée
    #[Route('/login', name: 'app_login')]
    public function index(): Response
    {
        return $this->render('login/index', [
            'controller_name' => 'LoginController',
        ]);
    }
}

```

Figure 8: LoginController



Figure 9: Composant de connections



Figure 10: Header de l'application logout

Une fois la bonne connexion effectuée, le menu change afin d'afficher le pseudo de la personne connectée ainsi que les différentes pages auxquelles il peut désormais avoir accès (Figure 11).



Figure 11: Header de l'application logged

Pour la sécurité au niveau des accès, Symfony possède plusieurs outils soit au niveau de la route (Figure 13) avec la fonction `#[IsGranted('ROLE_USER'])` qui vérifie le rôle de l'user soit dans le fichier de config `security.yaml` en décommenttant les lignes suivantes .

```

# Easy way to control access for large sections of your site
# Note: Only the *first* access control that matches will be used
access_control:
    - { path: ^/admin, roles: ROLE_ADMIN }
    - { path: ^/profile, roles: ROLE_USER }

```

Figure 12: security.yaml

```

#[Route('/trick')]
class TrickController extends AbstractController
{
    // Maxime Lemée
    #[Route('/', name: 'app_trick_index', methods: ['GET'])]
    public function index(TrickRepository $trickRepository): Response
    {
        return $this->render('trick/index', [
            'tricks' => $trickRepository->findAll(),
        ]);
    }

    // Maxime Lemée *
    #[IsGranted('ROLE_USER', message: 'Connectez-vous pour créer !Telle est la devise de SnowTricks')]
    #[Route('/new', name: 'app_trick_new', methods: ['GET', 'POST'])]
    public function new(Request $request, TrickRepository $trickRepository, FileUploader $fileUplo
    {
        $trick = new Trick();
        $form = $this->createForm(TrickType::class, $trick);
        $form->handleRequest($request);
    }
}

```

Figure 13: Fonction Is_granted

Mot de passe oublié

Dans le cas où l'utilisateur a oublié son mot de passe ou s'il souhaite le changer pour des raisons de sécurité, Symfony possède un bundle dédié .

En entrant dans le maker, la commande `php bin/console make:reset-password`, Symfony m'a généré tout le nécessaire pour que l'utilisateur puisse réinitialiser son mot de passe(Figure 14).

Symfony a créé un nouveau controller (voir annexe Figure 9etFigure 10) assez conséquent possédant toute une série de route dont

```
maxime@MaxVictus:~/Documents/dwm/Projet snowTrick/Snowtricks (Develop)$ php bin/console make:reset-password
Let's make a password reset feature!
=====
Implementing reset password for App\Entity\User
-----
- ResetPasswordController -
-----
A named route is used for redirecting after a successful reset. Even a route that does not exist yet can be used here.
What route should users be redirected to after their password has been successfully reset? [app_home]:
>

- Email -
-----
These are used to generate the email code. Don't worry, you can change them in the code later!
What email address will be used to send reset confirmations? e.g. mailer@your-domain.com:
> no-reply@mail.com

What "name" should be associated with that email address? e.g. "Acme Mail Bot":
> Snowtricks

created: src/Controller/ResetPasswordController.php
created: src/Entity/ResetPasswordRequest.php
updated: src/Entity/ResetPasswordRequest.php
created: src/Repository/ResetPasswordRequestRepository.php
updated: src/Repository/ResetPasswordRequestRepository.php
updated: config/packages/reset_password.yaml
created: src/Form/ResetPasswordRequestFormType.php
created: src/Form/ChangePasswordFormType.php
created: templates/reset_password/check_email.html.twig
created: templates/reset_password/email.html.twig
created: templates/reset_password/request.html.twig
```

Figure 14: `make:reset-password`

- `#[Route('/reset-password')]`, pour l'implantation graphique de la demande (Figure 15)
- `#[Route('/', name: 'app_forgot_password_request')]`

Cette route nous mène vers un template qui nous demande notre mail celui-ci est alors envoyé et si le mail est bien celui de l'user, alors il peut alors le valider pour indiquer qu'il s'agit bien de lui, et pour qu'il soit ainsi redirigé vers la prochaine route .

- `#[Route('/check-email', name: 'app_check_email')]`

Cette route génère un faux token si le user n'existe pas ou si quelqu'un tombe sur cette page directement en rentrant une URL.

- `#[Route('/reset/{token}', name: 'app_reset_password')]`

En cas de bon token, le user est dirigé vers un template de réinitialisation avec les champs à remplir et est averti du bon déroulement par un message flash.

Le mot de passe est mis à jour en base de données et la session nettoyée de son token.

```
{% extends 'base.html.twig' %}

{% block title %}Réinitialisé son mot de passe{% endblock %}

{% block body %}
    <h1>Réinitialisé son mot de passe</h1>

    {{ form_start(resetForm) }}
        {{ form_row(resetForm.plainPassword) }}
        <button class="btn btn-primary">Réinitialisé mot de passe</button>
    {{ form_end(resetForm) }}
{% endblock %}
```

Figure 15: Template pour la demande de réinitialisation

Slug / URL

Les slugs permettent d'obtenir des URL compréhensibles par n'importe quels internautes, c'est une fonction demandée au cahier des charges et maintenant indispensable à tous les sites et applications modernes.

Pour les mettre en place, j'ai utilisé une extension de Doctrine s'appelant [Gedmo](#), celle-ci à plusieurs utilités dont la génération de slug automatique. Pour cela, j'ai installé via composer le package nécessaire avec composer require gedmo/doctrine-extensions, j'ai créé le fichier de configuration (Figure 16) et profité pour activer une option **timestampable** qui permet, lors de la création d'un trick ou d'un user, d'automatiquement enregistrer la date du moment en base de données.

```
# Read the documentation: https://  
# See the official DoctrineExtensions  
stof_doctrine_extensions:  
    default_locale: fr_FR  
    orm:  
        default:  
            sluggable: true  
            timestampable: true
```

Figure 16:
StofDoctrineExtensions.yml

Ensuite, dans les entités Trick et User, que je veux Slugifiés, j'ai créé une propriété Slug que j'ai inscrit en base de données avec les commandes **make:migration** et **doctrine:migration:migrate**, la colonne slug a été créé suite à cela (Figure 20).

```
#[ORM\Column(length: 255, unique: true)]  
private ?string $nameTrick = null;  
  
1 usage  
#[ORM\Column(length: 128, unique: true)]  
#[Gedmo\Slug(fields: ['nameTrick'])]  
private $slug;
```

Figure 17: Slug Entité Trick

La prochaine étape est de préciser à gedmo à quelle propriété de l'entité elle doit se référer pour créer le slug (Figure 17 ,Figure 21)

Une fois cela fait, il faut désormais modifier les routes au niveau des controllers(Figure 18) et ainsi utiliser le slug comme identifiant de route, on obtient ainsi des URL compréhensibles.

```
Maxime Lemée *  
#[Route('/{slug}', name: 'app_trick_show', methods: ['GET', 'POST'])]
```

Figure 18: Slug app_trick_show

```
#[ORM\Column(length: 255)]  
private ?string $nickname = null;
```

```
1 usage  
#[ORM\Column(length: 128, unique: true)]  
#[Gedmo\Slug(fields: ['nickname'])]  
private $slug;
```

Figure 19: Slug app_user_edit

Figure 21: Slug Entité User

id	name_trick	slug	user_id
194	BeninBleu outremer	beninbleu-outremer	169
201	MarocBleu canard	marocbleu-canard	167
202	uytreza	uytreza	167
192	Cayman (îles)Basané	cayman-iles-basane	170
195	Svalbard et Jan Mayen (îles)Violet	svalbard-et-jan-mayen-iles-violet	167
196	LiechtensteinGris perle	liechtensteingris-perle	169
198	TongaJaune canari	tongajaune-canari	168
200	Norfolk (îles)Bleu marine	norfolk-iles-bleu-marine	167
199	Chine (Rép. pop.)Cramoisi	chine-rep-pop-cramoisi	170
193	Vatican (État du)Écrù	vatican-etat-du-ecru	167
197	BurundiBisque	burundibisque	170

Figure 20: Table trick base de donnée

Gestion des tricks

Page de détail

Espace de discutions

Comme demandé dans le cahier des charges, j'ai implanté un espace de discussions dans la page de détail d'un trick. Chaque trick à ses messages propres. Tous les utilisateurs peuvent les lire mais ces mêmes utilisateurs doivent être connectés pour commenter.

C'est pourquoi, j'ai inclus dans le code une condition if (Figure 22) qui affiche un formulaire pour un nouveau commentaire en cas où l'user serait connecté (Illustration 9) ou soit des liens qui l'invitent soit à se connecter soit à s'inscrire (Illustration 8).

Pour le nouveau commentaire, j'ai créé le

formulaire lié à l'entité Comment.php (Figure 23) à l'aide du make bundle symfony en tapant dans

Nombre de commentaires : 27 commentaire(s)
Adipisci quam minus id vel dolorem.
Aut quaerat vel debitis. Blanditiis illum
dicta distinctio ducimus eum.
Publié le 2023
Qui tempore nesciunt sint ut accusamus
consectetur. Impedit aut ipsum nihil
alias laboriosam. Totam vitae fuga nulla
eligendi libero.
Publié le 2023
Quo nisi nihil et mollitia maiores qui.
Excepturi dolores voluptatem ducimus
nesciunt voluptatem molestiae sapiente.
Veritatis accusantium voluptatem dolor
sit sit numquam velit et.
Publié le 2023
Voluptas qui debitis rerum qui beatae
minima. Mollitia dolore est ex ab eum.
Voluptatem non libero dolorem.
Publié le 2022
Vei molestias veniam nihil nihil
voluptate consequatur. Corporis vel ut
ducimus exercitationem officia et.
Harum dolorum non omnis mollitia
molestiae. Ut est dolores tenetur.
Publié le 2022

Comment
Votre commentaire
b5798
Soumettre
Nombre de commentaires : 27 commentaire(s)
Adipisci quam minus id vel dolorem.
Aut quaerat vel debitis. Blanditiis illum
dicta distinctio ducimus eum.
Publié le 2023
Qui tempore nesciunt sint ut accusamus
consectetur. Impedit aut ipsum nihil
alias laboriosam. Totam vitae fuga nulla
eligendi libero.
Publié le 2023
Quo nisi nihil et mollitia maiores qui.
Excepturi dolores voluptatem ducimus
nesciunt voluptatem molestiae sapiente.
Veritatis accusantium voluptatem dolor
sit sit numquam velit et.
Publié le 2023
Voluptas qui debitis rerum qui beatae
minima. Mollitia dolore est ex ab eum.
Voluptatem non libero dolorem.
Publié le 2022
Vei molestias veniam nihil nihil
voluptate consequatur. Corporis vel ut
ducimus exercitationem officia et.
Harum dolorum non omnis mollitia
molestiae. Ut est dolores tenetur.
Publié le 2022

Illustration 9: Espace de discutions connecté

Illustration 8: Espace de discutions non connecté

```
<div class="content trick-comment">
    <h3>Espace commentaire</h3>
    <div class="spaceComment">
        {% if app.user %}
            <div>
                {{ form_start(form) }}
                {{ form_end(form) }}
            </div>
        {% else %}
            <div class="spaceComment__content--subscribe">
                <a href="{{ path('app_login') }}><h4>Connectez-vous pour écrire un commentaire</h4></a>
                <a href="{{ path('app_register') }}><h4>Inscrivez vous en cliquant ici</h4></a>
            </div>
        {% endif %}
    </div>
```

Figure 22: Condition if commentaire

le terminal `php bin/console make:form` et je l'ai envoyé sur cette page via le contrôleur (Illustration 27)

```
class CommentType extends AbstractType
{
    no usages  ± Maxime Lemée
    public function buildForm(FormBuilderInterface $builder, array $options): void
    {
        $builder
            ->add( child: 'comment', type: TextareaType::class,
                'attr'=>[
                    'classe'=>'spaceComment__content',
                    'onload'=>'clean_textarea()',
                    'placeholder'=>'Votre commentaire',
                    'label'=> false,
                ]
            )
            ->add( child: 'captcha', type: CaptchaType::class,
                'disabled' => true,
                'label' => false,
                'invalid_message' => 'Mauvais captcha',
            )
            ->add( child: 'soumettre', type: SubmitType::class)
    }
}
```

Figure 23: Formulaire commentaire

Captcha

L'application réclamait un captcha et après en avoir fait la recherche, j'ai choisi le bundle Gregwar's que j'ai installé via `composer require gregwar/captcha-bundle`

Je l'ai par la suite implanté dans le formulaire en le rajoutant avec un `add` (*Illustration 11*) et j'en est profité pour le personnaliser légèrement comme le message d'alerte en lui attribuant des options.



Illustration 10: Captcha

```
->add('captcha', CaptchaType::class,[  
    'disabled' => true,  
    'label' => false,  
    'invalid_message' => 'Mauvais captcha',  
]);
```

Illustration 11: Captcha form
Comment.php

Pagination

Pour la pagination, j'ai utilisé le bundle symfony KnpPaginator avec la commande `composer require knplabs/knp-paginator-bundle`.

J'ai ensuite créé, dans le contrôleur la requête et l'implantation de la pagination. Pour la requête, écrit en DQL(doctrine), elle va chercher à chaque page seulement les 10 commentaires suivants ou précédents afin d'optimiser les accès à la base de données, une fois la Query (requête) faite, j'ai créé le système de pagination et y ai inclus cette même Query.

```
$dql = "SELECT c FROM App\Entity\Comment c WHERE c.trick = ".$trick->getId()." ORDER BY c.createDate DESC";  
$query = $manager->createQuery($dql);  
  
$pagination = $paginator->paginate(  
    $query,  
    $request->query->getInt( key: 'page', default: 1), /*Nombre de page*/  
    10 /*Limite par page*/  
);  
  
return $this->render( view: 'trick/show', [  
    'trick' => $trick,  
    'form' => $form,  
    'pagination'=>$pagination  
]);
```

Illustration 12: Pagination dans contrôleur Show trick

Je l'ai ensuite intégré dans le template twig, en précisant le nombre de commentaire avec `{{ pagination.getTotalItemCount }}` j'ai ensuite crée la boucle if avec `{% for comment in pagination %}` et finis par le nombre de page avec `{{ knp_pagination_render(pagination) }}`.

Nombre de commentaires : 28 commentaire(s)

Publié le 2016

<< < 1 2 3 > >>

```
Nombre de commentaires : {{ pagination.getTotalItemCount }} commentaire(s)  
{% for comment in pagination %}  
    <div id="one-comment" class="spaceComment__content">  
        <div class="spaceComment__left">  
            <div class="spaceComment__date">  
                <p>Publié le {{ comment.createDate.format('Y') }}</p>  
            </div>  
            {{ comment.comment }}  
        </div>  
        <div class="spaceComment__right">  
              
            <span>{{ comment.user.nickname }}</span>  
            {% if comment.user == app.user %}  
                {{ include('comment/_delete_form.html.twig') }}  
            {% endif %}  
        </div>  
    </div>  
{% endfor %}  
<div class="navigation">  
    {{ knp_pagination_render(pagination) }}  
</div>
```

Illustration 13: template pour pagination

Carrousel

Pour la mise en place d'un carrousel, je me suis appuyé sur la librairie JavaScript **swiper.js** que j'ai importé au sein du fichier app.css via les 2 imports suivants, import Swiper from 'swiper/bundle'; et import 'swiper/css/bundle';

En suivant la documentation, j'ai instancié une nouvelle variable et y ai implanté des options comme le délai d'affichage (3000ms), la pagination cliquable, l'espace entre les images à afficher .

```
const progressCircle :Element = document.querySelector( selectors: ".autoplay-progress svg");
const progressContent :Element = document.querySelector( selectors: ".autoplay-progress span");

const swiper = new Swiper(".mySwiper", {
    spaceBetween: 30,
    centeredSlides: true,
    autoplay: {
        delay: 3000,
        disableOnInteraction: false
    },
    pagination: {
        el: ".swiper-pagination",
        clickable: true
    },
    navigation: {
        nextEl: ".swiper-button-next",
        prevEl: ".swiper-button-prev"
    },
    on: {
        autoplayTimeLeft(s, time, progress) :void {
            progressCircle.style.setProperty( name: "--progress", value: 1 - progress);
            progressContent.textContent = `${Math.ceil( x: time / 1000)}s`;
        }
    }
});
```

Figure 24: Code javascrip pour carrousel swiper

J'ai ensuite développé le code css correspondant (voir annexe Figure 13) avec toutes les différentes parties comme la position du compteur (autoplay-progress) ou la taille de l'image (.swiper-slide img)



Figure 25: Carrousel Swiper

Page de création

Message flash

A chaque action importante, comme l'ajout d'un commentaire, sa suppression, la création d'un trick, sa modification ou sa suppression, j'affiche un message flash précisant à l'internaute la bonne marche de l'opération.



Figure 26: Message flash création d'un trick

C'est une option disponible nativement dans symfony en appelant l'instance `$this->addFlash(type, message)` comme dans l'exemple (Figure 28)

```
<div class="flash__content">
    {% for label, messages in app.flashes %}
        {% for message in messages %}
            <div class="alert alert-{{ label }}>
                {{ message }}
            </div>
        {% endfor %}
    {% endfor %}
</div>
```

Figure 27: Implantation message flash template twig

Le message est ensuite à planter au sein du template twig à l'endroit voulu comme ceci (Figure 27)

Un message flash pour indiquer le bon déroulement d'une action ressemblera à cela (Figure 26).

```
$trickRepository->save($trick, flush: true);

$this->addFlash( type: 'success', message: 'Ton trick '.$trick->getNameTrick().' a bien été créée !');

return $this->redirectToRoute( route: '/trick/', [], status: Response::HTTP_SEE_OTHER);
```

Figure 28: Contrôleur new trick addFlash

WYSIWYG

Lors de la création d'un nouveau trick ou de sa modification, j'ai implanté au sein du formulaire lié à l'entité Trick, un éditeur de texte plus riche que la balise Textarea par défaut de symfony, qui permettra à l'utilisateur une plus grande liberté de création lors de la rédaction de la description du Trick.

Pour ce faire, j'ai utilisé CkEditor qui est un WYSIWYG bien implanté au sein de symfony et implantable dans les formulaires très simplement après son installation.



Figure 29: Éditeur de texte Description Trick

Suppression

Lors de la génération du CRUD sur l'entité Trick, Symfony a généré le nécessaire pour créer, afficher, modifier et supprimer une instance de cette entité.

Pour la suppression d'un trick, l'utilisateur connecté pourra via un lien (Figure 30) supprimer un trick `{{ include('trick/_delete_form.html.twig') }}`

```
<div class="content trick-button">
    <a class="btn" href="{{ path('/trick/') }}"> Liste des tricks</a>
    {% if app.user %}
        <a class="btn" href="{{ path('/trick/{slug}/edit', {'slug': trick.slug}) }}> <span><svg ...></span> Modifier</a>
        <span>{{ include('trick/_delete_form.html.twig') }}</span>
    {% endif %}
```

Figure 30: Twig include route delete

Celui ci va, via un include, inclure à la page en cours le morceau de code contenu dans `_delete_form.html.twig` (Figure 31).

```
<form method="post" action="{{ path('app_trick_delete', {'slug': trick.slug}) }}" onsubmit="return confirm('Are you sure you want to delete this item?');">
    <input type="hidden" name="_token" value="{{ csrf_token('delete' ~ trick.id) }}>
    <svg xmlns:xlink="http://www.w3.org/1999/xlink" width="15" xmlns="http://www.w3.org/2000/svg" height="15" id="Screenshot-59e14f1a-7712-80dc-8002-7ca709">
</form>
```

Figure 31: `_delete_form.html.twig`

C'est à dire un formulaire qui une fois soumis avec un message de confirmation, enverra un token unique conçue avec l'id du trick vers la route `app_trick_delete` (Figure 32).

```
#[Route('/{slug}', name: 'app_trick_delete', methods: ['POST'])]
public function delete(Request $request, Trick $trick, TrickRepository $trickRepository): Response
{
    if ($this->isCsrfTokenValid( id: 'delete' . $trick->getId(), $request->request->get( key: '_token')) {
        foreach ($trick->getPhotoTricks() as $photoTrick)

            unlink($this->getParameter( name: ('uploads_path') . '/' . $photoTrick->getPhoto()));

        $trickRepository->remove($trick, flush: true);
    }

    return $this->redirectToRoute( route: '/trick/', [], status: Response::HTTP_SEE_OTHER);
}
```

Figure 32: Route `app_trick_delete`

Cette route vérifie si le token est valide en vérifiant également si l'id du trick dans l'URL correspondant à l'id présent dans le token lors de sa création, si c'est le cas, alors on supprime les photos liées à l'entité en question dans une boucle foreach en récupérant cette id.

Et enfin, on supprime le trick en question avec la méthode `$trickRepository->remove($trick, true);`

Pour ensuite être redirigé vers la route `/trick/`.

Gestions des droits

Au sein de l'application, un utilisateur connecté peut l'être en tant que **USER** ou **ADMIN**, ces différents rôles lui octroient chacun des droits différents.

Comme convenu dans le cahier des charges, un utilisateur connecté en rôle **USER** (vérifié lors de son inscription (Chapitre Page d'inscription) peut créer ou modifier un trick ainsi que le commenter, il peut également supprimer ses propres commentaires (Espace de discussions)

Pour la partie administration, j'ai utilisé le bundle de symfony EasyAdmin, qui permet la génération d'une administration assez rapidement et personnalisable. Ainsi un utilisateur ayant le rôle **ADMIN** pourra gérer l'ensemble des données du site (utilisateurs, trick, photo, commentaire) via ce dashboard dédié.

EasyAdmin

J'ai installé EasyAdmin via la commande *symfony composer req "admin:^4"* afin d'installer les dépendances nécessaires suivi d'un *php bin/console make:admin:dashboard* pour générer le controller dédié à l'administration. (Illustration 31: DashboardController).

Automatiquement, il va attribuer la route /admin pour son dashboard sur laquelle il affichera cette page (Illustration 15). Après sa configuration, c'est à dire l'ajout des entités que l'on souhaite gérer

```
public function configureFields(string $pageName): iterable
{
    return [
        TextField::new('propertyName: 'nameTrick'),
        DateTimeField::new('propertyName: 'creationDate'),
        TextEditorField::new('propertyName: 'description'),
        ImageField::new('propertyName: 'mainPhoto')
            ->setUploadDir('uploadDirPath: 'public/' . $this->getParameter('name: 'uploads_dir'))
            ->setBasePath($this->getParameter('name: 'uploads_dir')),
        CollectionField::new('propertyName: 'Comments'),
        CollectionField::new('propertyName: 'PhotoTricks'),
        CollectionField::new('propertyName: 'VideoTricks')
    ];
}
```

Illustration 14: Configuration trick Admin

par ce biais en rentrant la commande *php bin/console make:admin:crud* et certaines options d'affichage donc la configuration des champs (Illustration 14).

L'administrateur arrivera sur un template ressemblant à cela (Illustration 16) où il aura accès à toutes les informations. On peut remarquer toutes les entités rattachées au dashboard dans le menu de gauche.

En rattachant les entités via des crud, EasyAdmin nous permet désormais de les créer, modifier ou les supprimer au besoin directement via cette interface.

Welcome to EasyAdmin 4

You have successfully installed EasyAdmin in your application.

[Read EasyAdmin Docs](#)
Learn how to customize EasyAdmin to fit your needs

[Watch EasyAdmin Course on SymfonyCasts](#)
More than 30 videos to learn how to build a powerful admin area

[Sponsor EasyAdmin](#)
Fund the development of new features. One-time or regular donations

[Star EasyAdmin on GitHub](#)
Help us promote EasyAdmin to reach new developers

Snowtricks Administration						
Trick						
	Name Trick	Creation Date	Description	Main Photo	Comments	Photo Tricks
<input type="checkbox"/>	MonacoOr (couleur)	Jan 20, 1976, 12:00:00 AM	View content		22	1
<input type="checkbox"/>	ChypreZinolín	Jun 27, 2002, 12:00:00 AM	View content		32	1
<input type="checkbox"/>	Cayman (les)Lapis-lazuli	Aug 13, 1983, 12:00:00 AM	View content		30	3
<input type="checkbox"/>	TaiwanOlive	Dec 18, 2020, 12:00:00 AM	View content		42	2
<input type="checkbox"/>	Territoire britannique de l'océan IndienVert Véronèse	Jul 21, 1996, 12:00:00 AM	View content		30	3
<input type="checkbox"/>	EthiopieVioide	Feb 21, 1982, 12:00:00 AM	View content		34	1
<input type="checkbox"/>	NamibieAubergine	Jul 23, 2001, 12:00:00 AM	View content		36	3
<input type="checkbox"/>	MozambiqueTerre d'ombre	Aug 24, 1997, 12:00:00 AM	View content		26	5
<input type="checkbox"/>	BahrainBleu roi	Sep 19, 1982, 12:00:00 AM	View content		22	...
<input type="checkbox"/>	TogoGris de Payne	Aug 27, 2006, 12:00:00 AM	View content		26	7

Illustration 15: Page d'accueil

EasyAdmin

Illustration 16: Dashboard EasyAdmin

Référencement

Afin d'améliorer le référencement naturel de l'application, ce qui est de nos jours primordial, j'ai décidé de créer une page dédié au sitemap et une autre au fichier robot.txt afin de préciser aux différents moteurs de recherche les pages à exclure lors de son indexation .

sitemap.xml / robot.txt

J'ai créé pour chaque fichier, une route dédiée avec chacun leur controller.

Pour le sitemap.xml (Figure 3), j'ai créé un controller (Figure 2) générant un sitemap à jour à chaque requête du robot d'indexation à l'aide d'une boucle sur *trickRepository* et à l'aide de la méthode *\$this -> generatorUrl*, celle-ci me permet de créer un tableau *\$urls[]* en y incluant les urls des différentes tricks existants lors de l'appel de la page /sitemap.xml . Ainsi le robot aura connaissance du plan du site avec la liste des principales urls à indexer.

Pour le fichier robot.txt, le controller plus simple renvoie vers le template twig robot.html.twig contenant le texte à afficher pour le robot (Figure 1), celui ci indique au robot google d'exclure de son indexation tous les chemins qui suivent /admin/ et qu'il peut inclure ceux qui suivent /trick/

Pour les 2 cas d'usages, il fallait bien faire attention que la route possède bien l'extension du fichier, sitemap.xml et robot.txt, c'est primordial pour que le robot de google puissent les reconnaître et ainsi les lire.

```
User-Agent: Googlebot
Disallow: /admin/
Allow: /trick/
```

Figure 1: /robot.txt

```
class SitemapController extends AbstractController
{
    // ...
}

2 usages
private $trickRepository;

// Maxime Lemée
public function __construct(TrickRepository $trickRepository)
{
    $this->trickRepository = $trickRepository;
}

// Maxime Lemée
#[Route('/sitemap.xml', name: 'sitemap', defaults: ['_format' => 'xml'])]
public function index(): Response
{
    $tricks = $this->trickRepository->findAll();

    $urls = [];
    foreach ($tricks as $trick){
        $urls[] = [
            'loc' =>$this->generateUrl(
                route: 'app_trick_show', ['slug'=>$trick->getSlug()],
                referenceType: UrlGeneratorInterface::ABSOLUTE_URL
            ),
            'lastmod'=>$trick->getCreationDate()->format( format: 'Y-m-d')
        ];
    }

    $response = new Response(
        $this->renderView( view: '/sitemap/sitemap.html.twig', ['urls' => $urls]),
        status: 200
    );
    $response->headers->set( key: 'Content-Type', values: 'text/xml');

    return $response;
}
```

Figure 2: SitemapController

```
This XML file does not appear to have any style information associated with it.

<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9" xmlns: http://www.sitemaps.org/schemas/sitemap/0.9/sitemap.xsd">
    <url>
        <loc>http://127.0.0.1:8000/trick/bahrainviride</loc>
        <lastmod>2017-08-07</lastmod>
    </url>
    <url>
        <loc>http://127.0.0.1:8000/trick/coree-du-nordisabelle</loc>
        <lastmod>1987-10-12</lastmod>
    </url>
    <url>
        <loc>http://127.0.0.1:8000/trick/laosazur</loc>
        <lastmod>2000-01-17</lastmod>
    </url>
    <url>
        <loc>http://127.0.0.1:8000/trick/croatiejaune-citron</loc>
        <lastmod>1972-09-18</lastmod>
    </url>
    <url>
        <loc>http://127.0.0.1:8000/trick/mauricecorail</loc>
        <lastmod>1992-11-16</lastmod>
    </url>
    <url>
        <loc>http://127.0.0.1:8000/trick/suedechamois</loc>
        <lastmod>1992-12-14</lastmod>
    </url>
    <url>
        <loc>http://127.0.0.1:8000/trick/peroujaune-citron</loc>
        <lastmod>1983-11-08</lastmod>
    </url>
    <url>
        <loc>http://127.0.0.1:8000/trick/nicaraguacuivre</loc>
        <lastmod>1981-04-07</lastmod>
    </url>
    <url>
        <loc>http://127.0.0.1:8000/trick/ukrainevermillon</loc>
        <lastmod>2011-01-08</lastmod>
    </url>
    <url>
        <loc>http://127.0.0.1:8000/trick/chiliblanc-casse</loc>
        <lastmod>1981-03-01</lastmod>
    </url>
</urlset>
```

Figure 3: /sitemap.xml

SEO Quaker

Pour s'assurer de sa bonne place dans le classements au sein des futures recherches des internautes, j'ai fais un Audit SEO via l'application Quaker et Lighthouse afin de voir les améliorations possibles.

Quaker m'indique qu'il manque des attributs ALT sur 11 images, ce qui ne permettra pas à google de connaître la description de ces images et de ce fait de ne pas pouvoir les indexer correctement.

Il m'indique également que le ratio texte/image n'est pas bon et que je devrais ajouter beaucoup plus de texte.

Je n'ai pas d'objet Open Graph ni de carte Twitter, ce qui aiderais à sa visibilité sur des sites de réseaux sociaux ou autres.

Il situe bien le fichier robot.txt et sitemap.xml

AUDIT SEO DE PAGE			
Analyse de la page		Validé : 11	Erreur : 4
URL	✓	15 caractères – optimal. 127.0.0.1:8000/	Astuces▼
Canonical	⚠	Aucune balise canonical n'est définie pour cette page.	Astuces▼
Titre	✓	19 caractères – optimal. Accueil Snowtricks!	Astuces▼
Description Méta	⚠	51 caractères – moyen. Solution optimale entre 160 et 300 caractères. Le site collaboratif de référence sur le Snowboard	Astuces▼
Mots clés Méta	ⓘ	0 caractères, 0 mots.	Astuces▼
En-têtes	⚠	Votre page implémente des en-têtes HTML. H1 (3), H2 (3), H3 (4), H4 (3), H5 (0), H6 (0) <H1>: Dernière Rentrée Voir les autres ▾	Astuces▼
Images	⚠	11 images sans attribut ALT.	Astuces▼
Ratio texte / HTML	⚠	9.83% – aïe ! Le ratio de texte par rapport au code HTML de votre site est inférieur à 15 %. Nous vous suggérons d'ajouter beaucoup plus de texte à votre site Web.	Astuces▼
Cadres	✓	Non détecté	Astuces▼
Flash	✓	Non détecté	Astuces▼
Microformats	⚠	Cette page n'utilise pas les microformats de balisage. Vous pouvez valider votre balisage avec l'Outil de test des données structurées de Google.	Astuces▼
Schema.org	⚠	Votre page n'utilise pas Schema.org le balisage. Vous pouvez valider votre balisage avec l'Outil de test des données structurées de Google.	Astuces▼
Open Graph	⚠	Votre page n'a pas d'objets Open Graph. Vous pouvez valider votre balisage avec le Débogueur d'objet Facebook.	Astuces▼
Carte Twitter	⚠	Votre page n'a pas de Cartes Twitter. Vous pouvez valider vos cartes avec l'outil de validation des Cartes Twitter.	Astuces▼

Figure 4: Audit SEO Quaker 01

Conformité du site			
Robots.txt	✓	Bien, il semble que votre site ait un fichier robots.txt. http://127.0.0.1:8000/robots.txt	Astuces▼
Plans des sites XML	✓	Bien ! Votre site web contient un plan du site XML. http://127.0.0.1:8000/sitemap.xml	Astuces▼
Langue	✓	Super ! Vous avez spécifié la langue de votre site Web. fr	Astuces▼
DocType	✓	Super ! Vous avez spécifié le doctype. HTML5	Astuces▼
Encodage	✓	Déclarer un codage de caractères / de langue renforcera considérablement votre référencement. Il empêche également de compliquer l'affichage de la page. UTF-8	Astuces▼
Google™ Analytics	⚠	Google™ Analytics ne surveille pas votre site Web. Nous vous conseillons de profiter de ce formidable outil.	Astuces▼
Favicon	✓	C'est une bonne chose d'avoir une favicon. Inline image	Astuces▼

Figure 5: Audit SEO Quaker 02

LightHouse / Google

J'ai par la suite effectué un test via l'outil de google Lighthouse, ce qui m'a amené ces résultats (Figure 6).

Cette outil teste l'application sur ses performances, son accessibilité, ses bonnes pratiques ainsi que son SEO.

Sur les performances (Figure 7), la note tout en étant correct me précise que l'image la plus grande met trop de temps à s'afficher (2,8s), il y a donc moyen d'optimiser cela en changeant son format et/ou en la redimensionnant correctement.

Sur les performances SEO, il m'indique des liens qui ne sont pas explorables, et effectivement, ce sont des pages pas encore existantes à l'heure actuelle.

Il m'indique également qu'il y a un problème avec le fichier robot.txt, ce qui bloque son indexation, d'où son importance.

Sur les performances d'accessibilité, il m'indique que certaines couleurs ne sont pas assez contrastés et que certains de mes liens n'ont pas de texte visible.



Figure 6: Résultat test SEO Lighthouse google

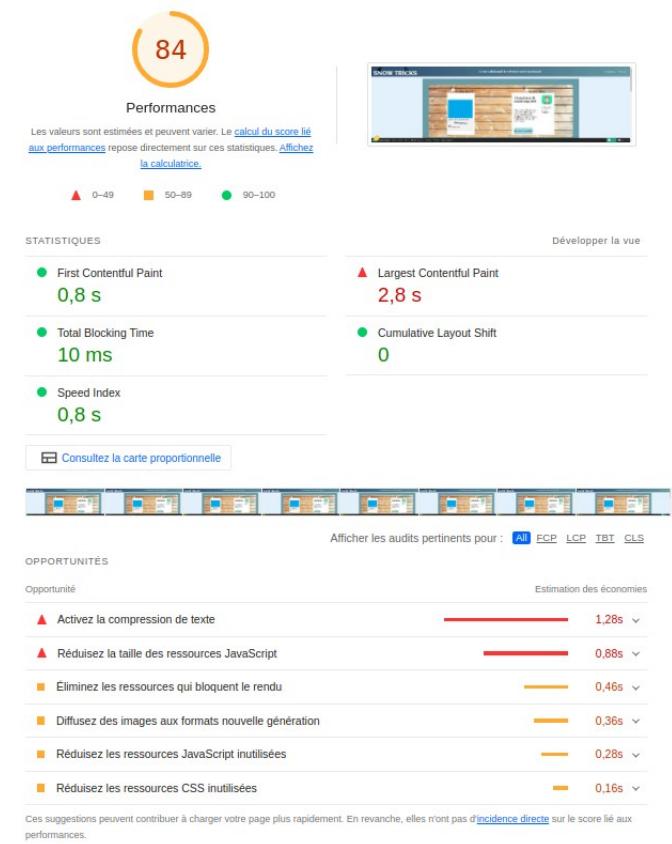


Figure 7: Résultat performances lighthouse

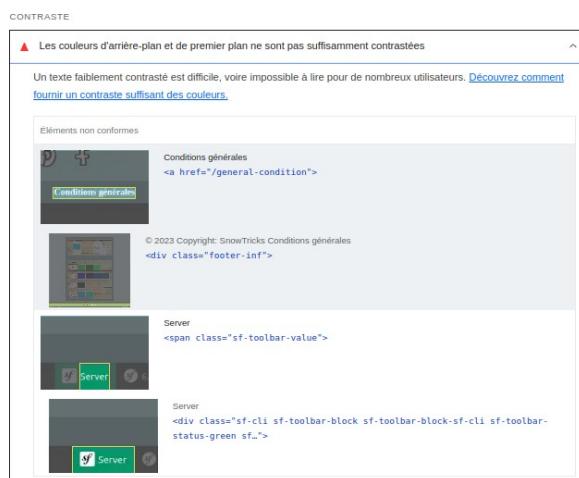


Figure 8: Résultat accessibilité Lighthouse

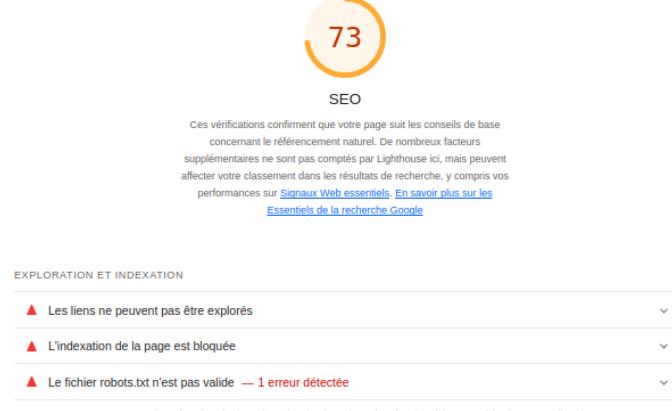


Figure 9: Résultat SEO lighthouse

Matomo Statistique web

Matomo, une alternative à Google Analytics, censé être plus respectueux de notre vie privée, est un outil servant à analyser le comportement des visiteurs, et le suivi du site.

Après l'avoir installé en local, dans le dossier html de mon serveur apache, j'ai configuré mon site

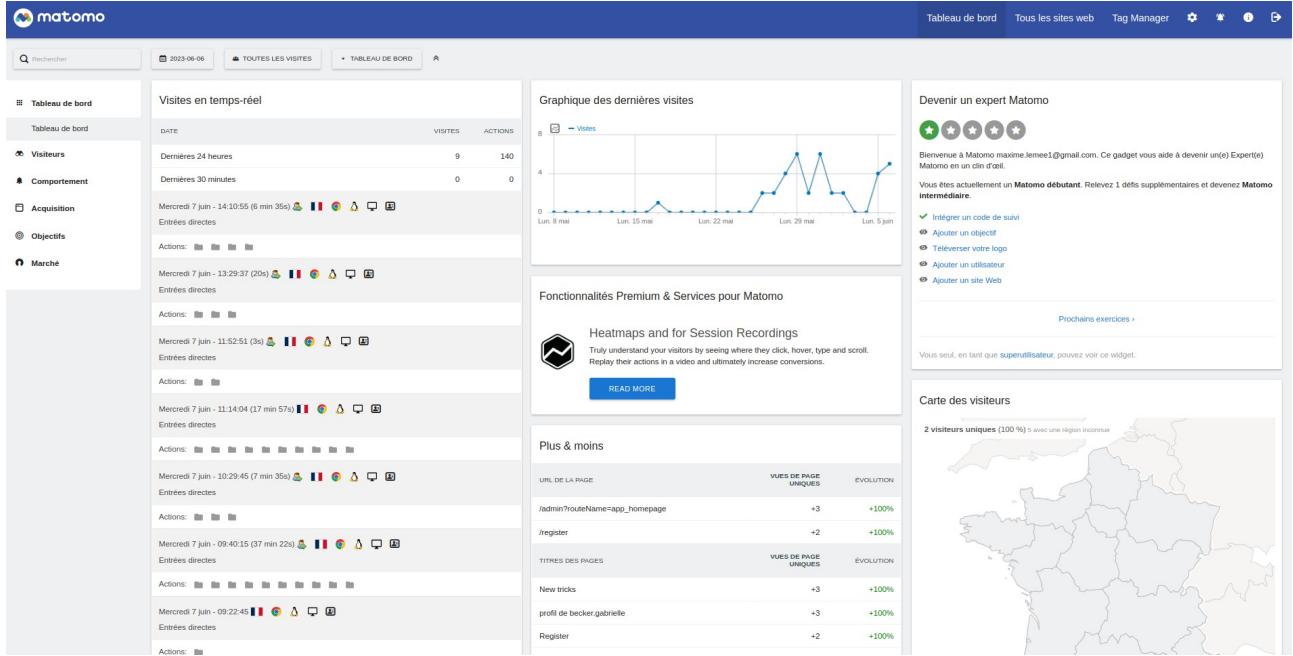


Illustration 17: Tableau de bord Matomo

en lui indiquant son adresse et y ai inclus un morceau de code fourni afin de le lié à Matomo (Illustration 18: Script matomo).

Grâce à ceci, désormais, j'ai accès au tableau de bord (Illustration 17: Tableau de bord Matomo), évidemment vide car je suis actuellement le seul à y naviguer mais qui sera très utile dans le cas d'une mise en ligne avec sa multitude d'outils de calcul et de suivis de navigation.

```
<script>
    var _paq = window._paq = window._paq || [];
    /* tracker methods like "setCustomDimension" should be called before "trackPageView" */
    _paq.push(['trackPageView']);
    _paq.push(['enableLinkTracking']);
    (function() {
        var u="//localhost/matomo/";
        _paq.push(['setTrackerUrl', u+'matomo.php']);
        _paq.push(['setSiteId', '1']);
        var d=document, g=d.createElement('script'), s=d.getElementsByTagName('script')[0];
        g.async=true; g.src=u+'matomo.js'; s.parentNode.insertBefore(g,s);
    })();
</script>
```

Illustration 18: Script matomo

Sécurité

Afin de gérer la sécurité de l'application, le framework Symfony possède tout une série d'outils permettant de rendre notre application moins sensible aux attaques, leur configuration se situe dans le fichier security.yaml (Illustration 30: security.yaml). Il contient tout ce qui concerne le hachage des mots de passes, les identifiants avec lesquels l'utilisateur pourra se connecter, le niveau du hachage (plus le flashage est fort et plus il demandera de la ressource).

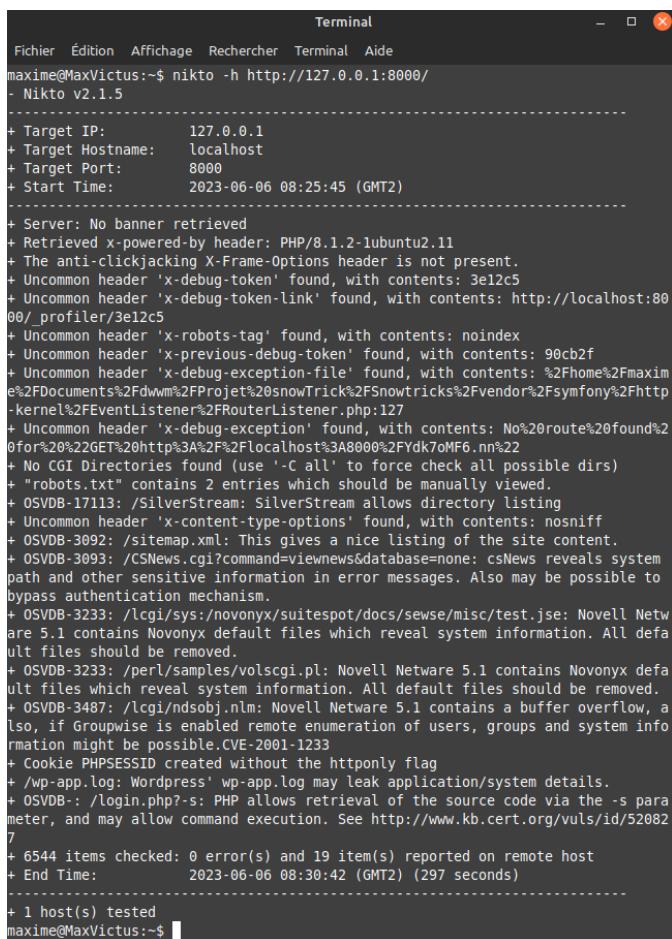
Durant la procédure de login, Symfony utilise des jetons CSRF via l'URL et possède également une protection contre les attaques XSS avec son option **sanitize_html** qui est activée par défaut lors de la création d'un formulaire.

```
<label for="inputEmail">Email</label>
<input type="email" value="{{ last_username }}" name="email" id="inputEmail" class="form-control"
    autocomplete="email" required autofocus>
<label for="inputPassword">Mot de passe</label>
<input type="password" name="password" id="inputPassword" class="form-control"
    autocomplete="current-password" required>
<input type="hidden" name="_csrf_token"
    value="{{ csrf_token('authenticate') }}"/>
```

Illustration 19: Formulaire de login

Symfony contient également une protection contre les injections SQL via le langage DQL de doctrine et ses requêtes préparées.

Nikto / Owasp zap



```
Fichier Édition Affichage Rechercher Terminal Aide
maxime@MaxVictus:~$ nikto -h http://127.0.0.1:8000/
- Nikto v2.1.5
-----
+ Target IP:      127.0.0.1
+ Target Hostname: localhost
+ Target Port:     8000
+ Start Time:    2023-06-06 08:25:45 (GMT2)
-----
+ Server: No banner retrieved
+ Retrieved x-powered-by header: PHP/8.1.2-1ubuntu2.11
+ The anti-clickjacking X-Frame-Options header is not present.
+ Uncommon header 'x-debug-token' found, with contents: 3e12c5
+ Uncommon header 'x-debug-token-link' found, with contents: http://localhost:8000/_profiler/3e12c5
+ Uncommon header 'x-robots-tag' found, with contents: noindex
+ Uncommon header 'x-previous-debug-token' found, with contents: 90cb2f
+ Uncommon header 'x-debug-exception-file' found, with contents: %2Fhome%2Fmaxime%2Fdocuments%2Fdwww%2FProject%20snowTrick%2FSnowtricks%2Fvendor%2Fsymfony%2Fhttp-kernel%2FEventListener%2FRouterListener.php:127
+ Uncommon header 'x-debug-exception' found, with contents: No%20route%20found%20for%20%22GET%20http%3A%2Flocalhost%3A8000%2FYdk7oMF6.nn%22
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ "robots.txt" contains 2 entries which should be manually viewed.
+ OSVDB-17113: /SilverStream: SilverStream allows directory listing
+ Uncommon header 'x-content-type-options' found, with contents: nosniff
+ OSVDB-3092: /sitemap.xml: This gives a nice listing of the site content.
+ OSVDB-3093: /CSNews.cgi?command=viewnews&database=none: CSNews reveals system path and other sensitive information in error messages. Also may be possible to bypass authentication mechanism.
+ OSVDB-3233: /lcgi/sys:/novonyx/sitespot/docs/sewse/misc/test.jse: Novell Netware 5.1 contains Novonyx default files which reveal system information. All default files should be removed.
+ OSVDB-3233: /perl/samples/volscgi.pl: Novell Netware 5.1 contains Novonyx default files which reveal system information. All default files should be removed.
+ OSVDB-3487: /lcgi/ndsoobj.nlm: Novell Netware 5.1 contains a buffer overflow, also, if Groupwise is enabled remote enumeration of users, groups and system information might be possible.CVE-2001-1233
+ Cookie PHPSESSID created without the httponly flag
+ /wp-app.log: Wordpress' wp-app.log may leak application/system details.
+ OSVDB : /login.php?-s: PHP allows retrieval of the source code via the -s parameter, and may allow command execution. See http://www.kb.cert.org/vuls/id/520827
+ 6544 items checked: 0 error(s) and 19 item(s) reported on remote host
+ End Time:        2023-06-06 08:30:42 (GMT2) (297 seconds)
-----
+ 1 host(s) tested
maxime@MaxVictus:~$
```

Illustration 20: Test sécurité Nikto

J'ai effectué un test de sécurité à l'aide de l'outil Nikto et il en ressort 0 erreur critique.

J'ai effectué un autre test avec Owasp Zap(Illustration 28: Owasp Zap) et il en ressort également aucune erreur majeure, éventuellement en priorité moyenne, [l'absence de jeton anti CRSE](#).

Ces jetons sont utilisés pour contrer les attaques cross-site request.

RGPD

CGU

J'ai créé dans l'application une route menant vers les conditions générales d'utilisation et l'utilisateur peut y avoir accès via le lien dans le footer .

J'ai rédigé ce texte à l'aide d'une IA, le texte y *Illustration 21: CGU* figurant (voir annexe *Illustration 29: CGU*) met bien l'accent sur les obligations de l'utilisateur à accepter, les CGU et la politique de confidentialité sur site.



Cookie

Afin de gérer les cookies, j'ai installé via un service externe, un outil de gestion des cookies .



Celui-ci se configure directement sur leur site (*Illustration 22*) et s'implante sur notre application en intégrant, dans le footer html, un script JavaScript (*Illustration 23*).

```
<script>
  window.axeptioSettings = {
    clientId: "64785273f85b082feda5798a",
    cookiesVersion: "snowtrick-fr",
  };

  (function(d,s) {
    var spt = d.getElementsByTagName(s)[0], e = d.createElement(s);
    e.async = true; e.src = "//static.axept.io/sdk.js";
    t.parentNode.insertBefore(e, t);
  })(document, "script");
</script>
```

Illustration 23: script axeptio

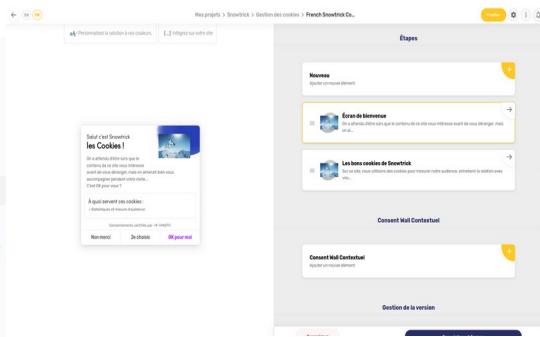


Illustration 22: Configuration cookie Axeptio

En ouvrant l'application, j'ai désormais une icône dédiée aux cookie utilisateurs (*Illustration 25*) et qui demande si besoin l'accord pour leur utilisation (*Illustration 24*).

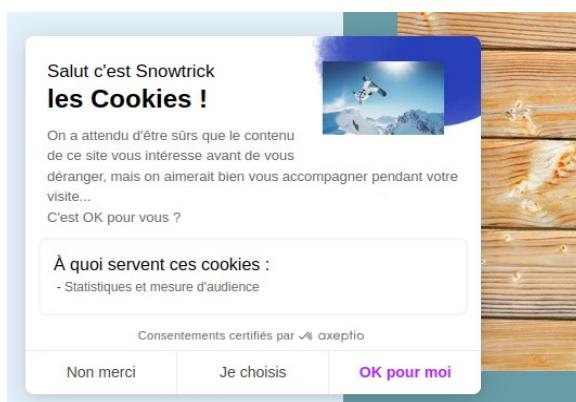


Illustration 24: Fenêtre cookie

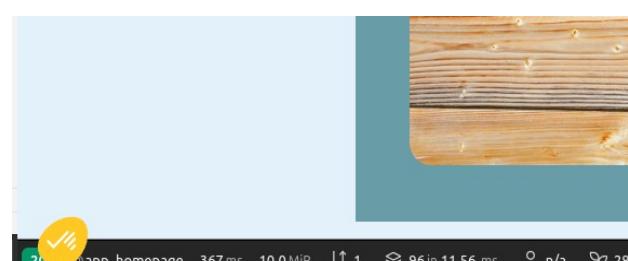


Illustration 25: Implantation axeptio

Compétences acquises

- **Gestion de projet**

- J'ai appris à lire, étudier un cahier des charges et se servir d'un logiciel de gestion de projet afin de structurer son développement à l'aide de tableau de tâches et de diagrammes de Gantt. Planification du projet :
- J'ai appris à effectuer une veille technologique et à faire les choix correspondants au mieux à chaque situation. Choix technique / Veille Nommage
- J'ai appris à utiliser Git à bon escient en tant que moyen de sauvegarde et en temps qu'aide durant le développement. Git

- **Front-end**

- J'ai appris à maquetter une application web via un logiciel tiers en lui incluant une chartre graphique et typographique suivant un cahier des charges. Maquettage du projet
- J'ai appris à réaliser une interface graphique statique avec HTML et CSS ou soit SCSS avec ses avantages. SCSS Twig
- J'ai appris à utiliser le responsive design afin de répondre à son utilisation sur desktop, tablette et smartphone Responsive
- J'ai appris à renforcer le référencement ainsi que l'accessibilité d'une application web. Référencement
- J'ai appris les obligations que réclame un site envers ses utilisateurs. RGPD

- **Back-end**

- J'ai appris à réaliser des diagrammes UML correspondant à la demande du cahier des charges. Création des diagramme UML
- J'ai appris à créer une base de données, et générer des requêtes en SQL et en DQL Doctrine
- J'ai appris à utiliser un framework back-end (symfony) et y inclure les packages et les bundles suivant les besoins Création de l'architecture symfony Fonctionnalités
- J'ai appris la gestion des droits dans une application symfony. Gestions des droits
- J'ai appris à tester la sécurité de mon application et de prendre les mesures nécessaires aux situations les plus courantes. Sécurité

Remerciements

Je remercie tous mes professeurs pour leurs partages de connaissances et d'expériences durant ces 6 mois et particulièrement mon professeur référent Maxime FLASQUIN qui m'a été d'une grande aide pour la réalisation de mes dossiers.

Je remercie également toute l'équipe de l'école d'Informatique Appliquée de Laval pour son accueil, son suivi, ses conseils et ses réponses durant mes moments de doutes.

Je remercie Transition Pro pour avoir accepté de financer cette formation importante pour moi.

Je remercie mes proches pour leurs aides et leurs motivations.

Conclusion

Durant ce développement, toutes les étapes ont été pour moi l'occasion d'approfondir mes connaissances actuelles et acquises durant les cours pour un projet concret et tout à fait déployable.

Durant ces 3 mois de conception, je me suis forcé à rester organisé à l'aide d'une ligne de tâches précises et ce, malgré mon manque d'expérience qui s'est naturellement fait sentir.

Il manque certaines fonctionnalités que j'aurais bien voulu mettre en place,

- La validation ou le refus par l'administrateur via EasyAdmin d'une nouvelle création ou d'une modification d'un trick.
- La personnalisation des pages d'erreurs 403 et 404 aux couleurs du site.
- L'accès à une charte graphique par l'administrateur et la possibilité de pouvoir changer le logo du site.
- Un espace personnel pour l'utilisateur plus aboutis avec la possibilité de modifier et de proposer des modifications à la volée.
- J'aurais également évité le rechargeage de la page entière lors de l'ajout d'un nouveau commentaire.

Ce sont des fonctionnalités que j'implanterais à la suite de ma formation afin de parfaire ce projet et d'approfondir mes connaissances.

Cette formation et ce projet m'ont ouvert les yeux sur le fait que le métier de développeur est d'une très grande richesse et demande des connaissances très variées et qu'il faut toujours être au fait des évolutions en maintenant une veille active.

C'est pourquoi, j'ai décidé de compléter l'acquisition des mes savoirs en poursuivant mon cursus de formation en effectuant une licence informatique en développement en alternance.

Annexe



CC0 1.0 universel (CC0 1.0)
Transfert dans le Domaine Public

Ceci est un résumé explicatif de la [Licence](#) (lire la version complète). Avertissement

Pas de droit d'auteur

 La personne qui a associé une œuvre à cet acte a **dédié** l'œuvre au domaine public en renonçant dans le monde entier à ses droits sur l'œuvre selon les lois sur le droit d'auteur, droit voisin et connexes, dans la mesure permise par la loi.



Vous pouvez copier, modifier, distribuer et représenter l'œuvre, même à des fins commerciales, sans avoir besoin de demander l'autorisation. Voir **d'autres informations** ci-dessous.

Autres informations

- Les brevets et droits de marque commerciale qui peuvent être détenus par autrui ne sont en aucune façon affectés par CC0, de même pour les droits que pourraient détenir d'autres personnes sur l'œuvre ou sur la façon dont elle est utilisée, comme [le droit à l'image ou à la vie privée](#).
- À moins d'une mention expresse contraire, la personne qui a identifié une œuvre à cette notice ne concède aucune garantie sur l'œuvre et décline toute responsabilité de toute utilisation de l'œuvre, dans la mesure permise par la loi.
- Quand vous utilisez ou citez l'œuvre, vous ne devez pas sous-entendre [le soutien](#) de l'auteur ou de la personne qui affirme.

Figure 1: Licence CC0 1.0 universel (CC0 1.0)



Illustration 26: Maquettage Prototype



Dessin 2:
Maquettage
vidéos
maquettage

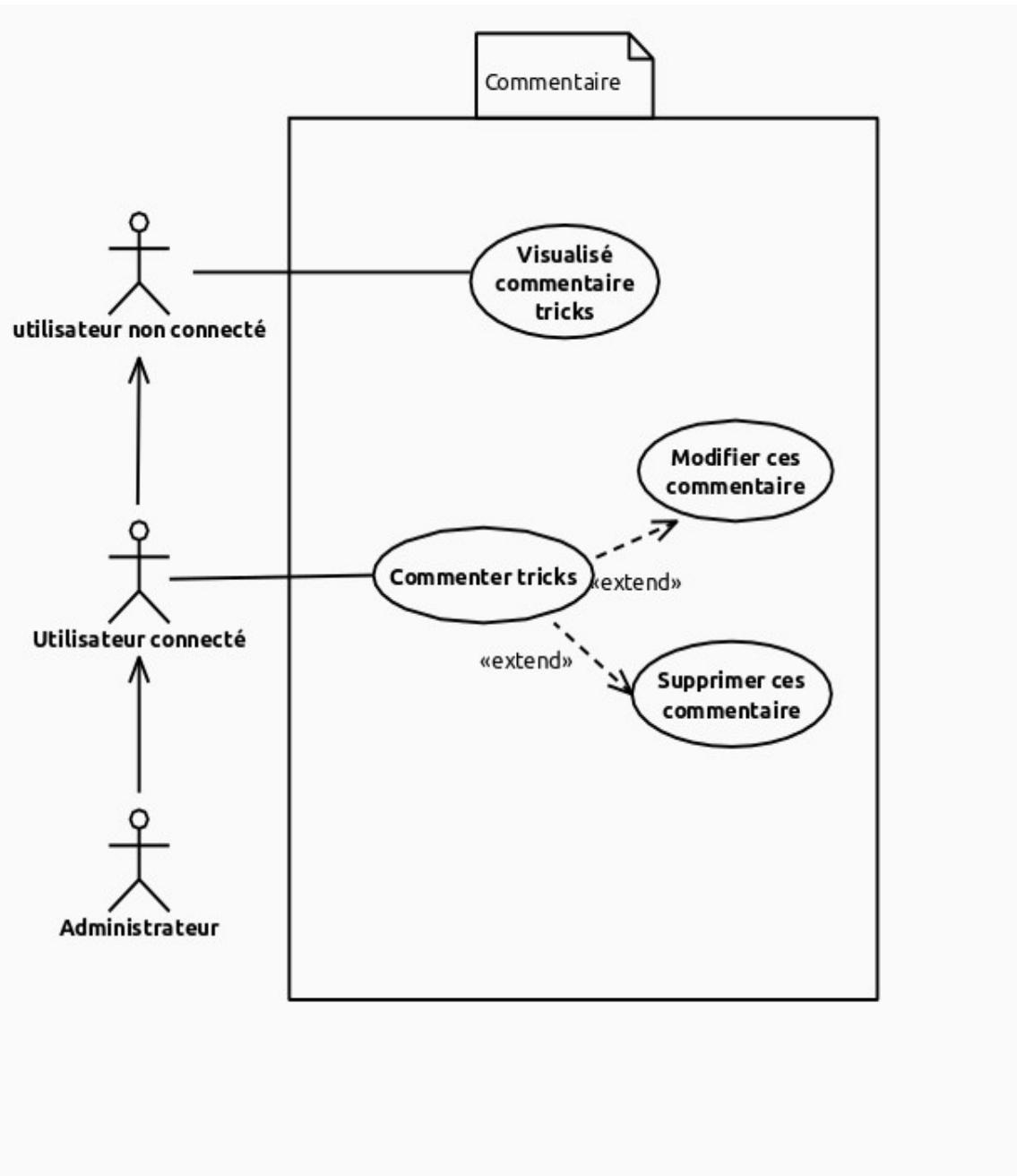


Figure 2: Diagramme cas utilisation Commentaire

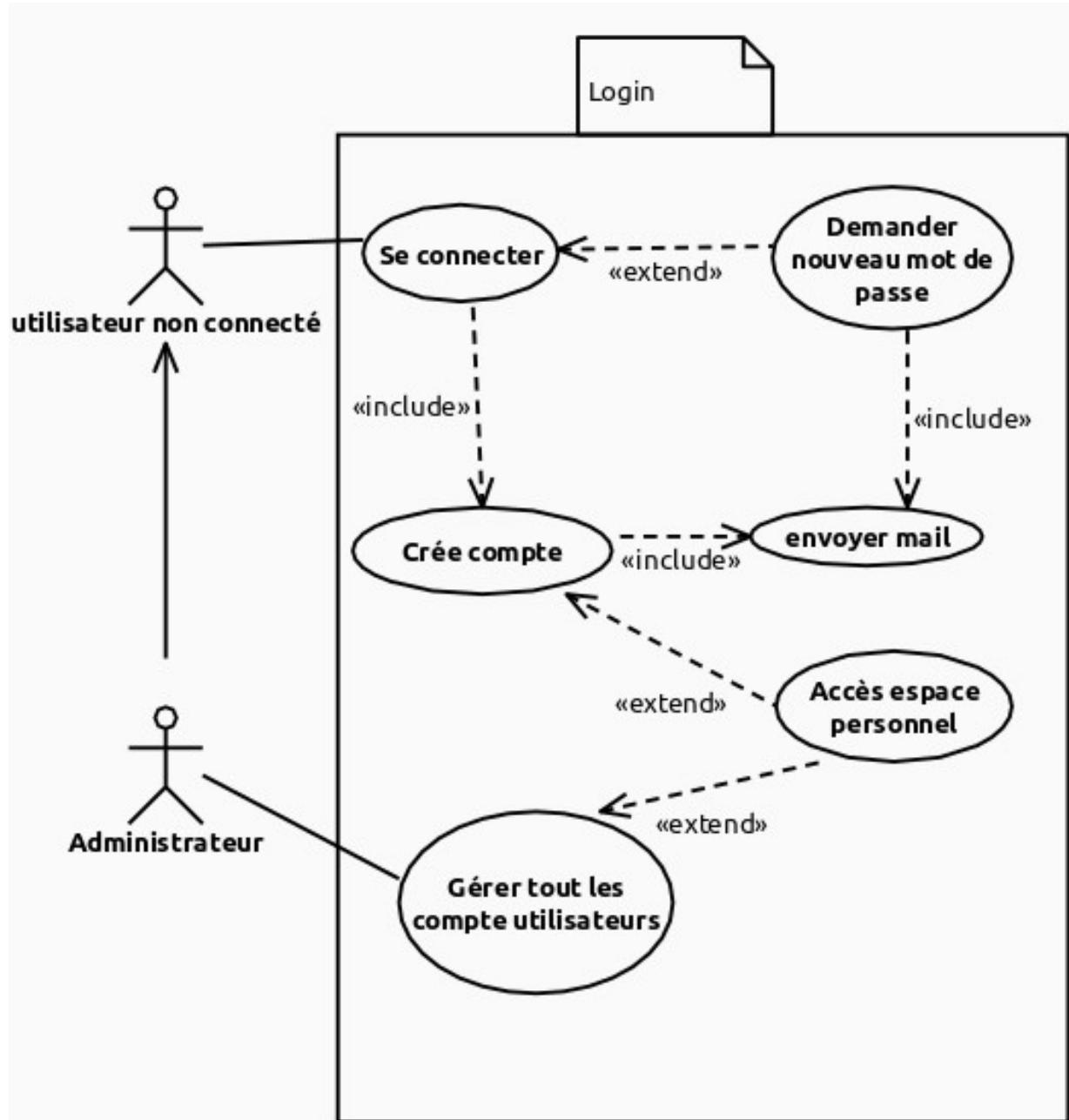
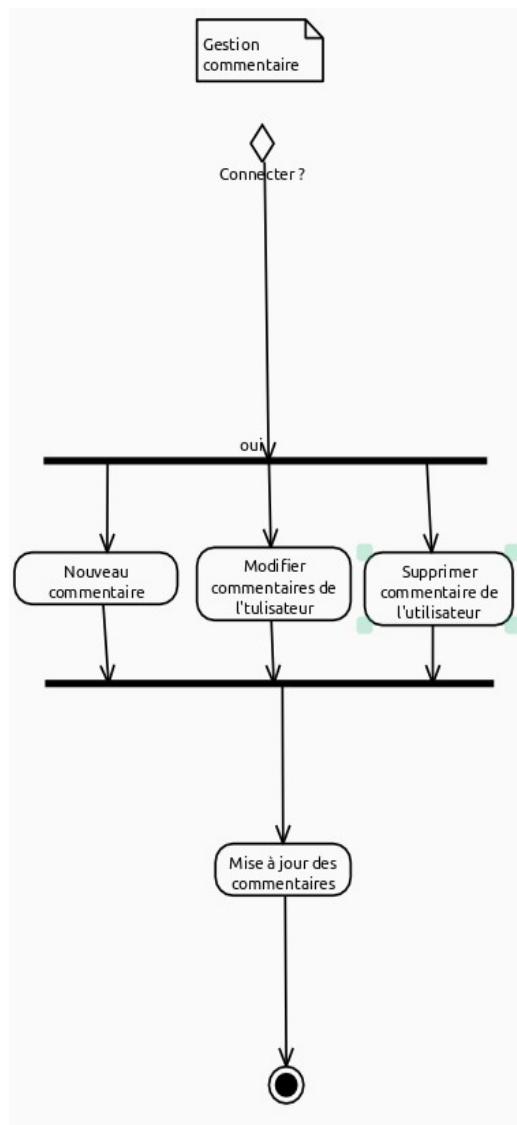


Figure 3: Diagramme cas utilisation Login



*Figure 4: Diagramme d'activité
'Commentaire'*

```

#[ORM\Entity(repositoryClass: TrickRepository::class)]
class Trick
{
    #[ORM\Id]
    #[ORM\GeneratedValue]
    #[ORM\Column]
    private ?int $id = null;

    #[ORM\Column(length: 255)]
    private ?string $nameTrick = null;

    #[ORM\Column(type: Types::TEXT)]
    private ?string $description = null;

    #[ORM\Column(type: Types::DATE_IMMUTABLE)]
    private ?\DateTimeImmutable $creationDate = null;

    #[ORM\ManyToOne(inversedBy: 'Trick')]
    #[ORM\JoinColumn(nullable: false)]
    private ?User $user = null;

    #[ORM\OneToMany(mappedBy: 'Trick', targetEntity: Comment::class)]
    private Collection $comments;

    #[ORM\ManyToOne(inversedBy: 'Trick')]
    #[ORM\JoinColumn(nullable: false)]
    private ?GroupTrick $groupTrick = null;

    #[ORM\OneToMany(mappedBy: 'trick', targetEntity: PhotoTrick::class)]
    private Collection $photoTricks;

    #[ORM\OneToMany(mappedBy: 'Trick', targetEntity: VideoTrick::class)]
    private Collection $videoTricks;

    /**
     * @return Collection
     */
    public function getComments(): Collection
    {
        return $this->comments;
    }

    /**
     * @return Collection
     */
    public function getPhotoTricks(): Collection
    {
        return $this->photoTricks;
    }

    /**
     * @return Collection
     */
    public function getVideoTricks(): Collection
    {
        return $this->videoTricks;
    }
}

```

Figure 5: Entity Trick

```

namespace App\Repository;

use App\Entity\Trick;
use Doctrine\Bundle\DoctrineBundle\Repository\ServiceEntityRepository;
use Doctrine\Persistence\ManagerRegistry;

/**
 * @extends ServiceEntityRepository<Trick>
 *
 * @method Trick|null find($id, $lockMode = null, $lockVersion = null)
 * @method Trick|null findOneBy(array $criteria, array $orderBy = null)
 * @method Trick[]    findAll()
 * @method Trick[]    findBy(array $criteria, array $orderBy = null, $limit = null, $offset = null)
 */

```

Maxime Lemée

```

class TrickRepository extends ServiceEntityRepository
{
    Maxime Lemée
    public function __construct(ManagerRegistry $registry)
    {
        parent::__construct($registry, Trick::class);
    }

```

Maxime Lemée

```

    public function save(Trick $entity, bool $flush = false): void
    {
        $this->getEntityManager()->persist($entity);

        if ($flush) {
            $this->getEntityManager()->flush();
        }
    }

```

Maxime Lemée

```

    public function remove(Trick $entity, bool $flush = false): void
    {
        $this->getEntityManager()->remove($entity);

        if ($flush) {
            $this->getEntityManager()->flush();
        }
    }
}

```

Figure 6: TrickRepository

```

<?php

declare(strict_types=1);

namespace DoctrineMigrations;

use ...

/**
 * Auto-generated Migration: Please modify to your needs!
 */

final class Version20230511090055 extends AbstractMigration
{

    public function getDescription(): string
    {
        return '';
    }

    public function up(Schema $schema): void
    {
        // this up() migration is auto-generated, please modify it to your needs
        $this->addSql( sql: 'CREATE TABLE comment (id INT AUTO_INCREMENT NOT NULL, trick_id INT NOT NULL, user_id INT DEFAULT NULL, comment LONGTEXT');
        $this->addSql( sql: 'CREATE TABLE group_trick (id INT AUTO_INCREMENT NOT NULL, name VARCHAR(255) NOT NULL, logo VARCHAR(255) NOT NULL, PRIMARY KEY (id)');
        $this->addSql( sql: 'CREATE TABLE photo_trick (id INT AUTO_INCREMENT NOT NULL, trick_id INT DEFAULT NULL, user_id INT NOT NULL, photo VARCHAR(255) NOT NULL, PRIMARY KEY (id)');
        $this->addSql( sql: 'CREATE TABLE reset_password_request (id INT AUTO_INCREMENT NOT NULL, user_id INT NOT NULL, selector VARCHAR(20) NOT NULL, hash VARCHAR(100) NOT NULL, created_at DATETIME NOT NULL, EXPIRED_AT DATETIME NOT NULL, EXPIRATION_AT DATETIME NOT NULL, ip_address VARCHAR(45) NOT NULL, user_agent VARCHAR(255) NOT NULL, type ENUM("reset", "change") NOT NULL, provider VARCHAR(255) NOT NULL, status ENUM("new", "verified", "expired", "used") NOT NULL, UNIQUE KEY (selector)');
        $this->addSql( sql: 'CREATE TABLE trick (id INT AUTO_INCREMENT NOT NULL, user_id INT DEFAULT NULL, group_trick_id INT NOT NULL, name_trick VARCHAR(255) NOT NULL, description TEXT, content TEXT, created_at DATETIME NOT NULL, updated_at DATETIME NOT NULL, PRIMARY KEY (id)');
        $this->addSql( sql: 'CREATE TABLE user (id INT AUTO_INCREMENT NOT NULL, email VARCHAR(180) NOT NULL, roles JSON NOT NULL, password VARCHAR(255) NOT NULL, salt VARCHAR(64) NOT NULL, PRIMARY KEY (id)');
        $this->addSql( sql: 'CREATE TABLE video_trick (id INT AUTO_INCREMENT NOT NULL, trick_id INT DEFAULT NULL, user_id INT NOT NULL, video VARCHAR(255) NOT NULL, PRIMARY KEY (id)');
        $this->addSql( sql: 'ALTER TABLE comment ADD CONSTRAINT FK_9474526CB281BE2E FOREIGN KEY (trick_id) REFERENCES trick (id)');
        $this->addSql( sql: 'ALTER TABLE comment ADD CONSTRAINT FK_9474526CA76ED395 FOREIGN KEY (user_id) REFERENCES user (id) ON DELETE SET NULL');
        $this->addSql( sql: 'ALTER TABLE photo_trick ADD CONSTRAINT FK_62A1AEA9A76ED395 FOREIGN KEY (trick_id) REFERENCES trick (id)');
        $this->addSql( sql: 'ALTER TABLE photo_trick ADD CONSTRAINT FK_62A1AEA9A76ED395 FOREIGN KEY (user_id) REFERENCES user (id)');
        $this->addSql( sql: 'ALTER TABLE reset_password_request ADD CONSTRAINT FK_7CE748AA76ED395 FOREIGN KEY (user_id) REFERENCES user (id)');
        $this->addSql( sql: 'ALTER TABLE trick ADD CONSTRAINT FK_D8F0A91E76ED395 FOREIGN KEY (user_id) REFERENCES user (id) ON DELETE SET NULL');
        $this->addSql( sql: 'ALTER TABLE trick ADD CONSTRAINT FK_D8F0A91EBB1F251 FOREIGN KEY (group_trick_id) REFERENCES group_trick (id)');
        $this->addSql( sql: 'ALTER TABLE video_trick ADD CONSTRAINT FK_5792A0BCB281BE2E FOREIGN KEY (trick_id) REFERENCES trick (id)');
        $this->addSql( sql: 'ALTER TABLE video_trick ADD CONSTRAINT FK_5792A0BCA76ED395 FOREIGN KEY (user_id) REFERENCES user (id)');
    }

    public function down(Schema $schema): void
    {
        // this down() migration is auto-generated, please modify it to your needs
        $this->addSql( sql: 'ALTER TABLE comment DROP FOREIGN KEY FK_9474526CB281BE2E');
        $this->addSql( sql: 'ALTER TABLE comment DROP FOREIGN KEY FK_9474526CA76ED395');
        $this->addSql( sql: 'ALTER TABLE photo_trick DROP FOREIGN KEY FK_62A1AEA9A76ED395');
        $this->addSql( sql: 'ALTER TABLE reset_password_request DROP FOREIGN KEY FK_7CE748AA76ED395');
        $this->addSql( sql: 'ALTER TABLE trick DROP FOREIGN KEY FK_D8F0A91E76ED395');
        $this->addSql( sql: 'ALTER TABLE trick DROP FOREIGN KEY FK_D8F0A91EBB1F251');
        $this->addSql( sql: 'ALTER TABLE video_trick DROP FOREIGN KEY FK_5792A0BCB281BE2E');
        $this->addSql( sql: 'ALTER TABLE video_trick DROP FOREIGN KEY FK_5792A0BCA76ED395');
    }
}

```

Figure 7: Exemple de fichier de migration Doctrine

```

#[Route('/tricks')]
class TrickController extends AbstractController
{
    #[Route('/', name: 'app_trick_index', methods: ['GET'])]
    public function index(TrickRepository $trickRepository): Response
    {
        return $this->render( view: 'trick/index', [
            'tricks' => $trickRepository->findAll(),
        ]);
    }

    #[Route('/new', name: 'app_trick_new', methods: ['GET', 'POST'])]
    public function new(Request $request, TrickRepository $trickRepository): Response
    {
        $trick = new Trick();
        $form = $this->createForm( type: TrickType::class, $trick);
        $form->handleRequest($request);

        if ($form->isSubmitted() && $form->isValid()) {
            $trickRepository->save($trick, flush: true);

            return $this->redirectToRoute( route: '/trick/', [], status: Response::HTTP_SEE_OTHER);
        }

        return $this->renderForm( view: 'trick/new', [
            'tricks' => $trick,
            'form' => $form,
        ]);
    }

    #[Route('/{id}', name: 'app_trick_show', methods: ['GET'])]
    public function show(Trick $trick): Response
    {
        return $this->render( view: 'trick/show', [
            'tricks' => $trick,
        ]);
    }

    #[Route('/{id}/edit', name: 'app_trick_edit', methods: ['GET', 'POST'])]
    public function edit(Request $request, Trick $trick, TrickRepository $trickRepository): Response
    {
        $form = $this->createForm( type: TrickType::class, $trick);
        $form->handleRequest($request);

        if ($form->isSubmitted() && $form->isValid()) {
            $trickRepository->save($trick, flush: true);

            return $this->redirectToRoute( route: '/trick/', [], status: Response::HTTP_SEE_OTHER);
        }
    }
}

```

Figure 8: Controller avec les différentes routes pour l'entité Trick

```

        return $this->render( view: 'reset_password/check_email', [
            'resetToken' => $resetToken,
        ]);
    }

    /**
     * Validates and process the reset URL that the user clicked in their email.
     */
    ↳ Maxime Lemée
#[Route('/reset/{token}', name: 'app_reset_password')]
public function reset(Request $request, UserPasswordHasherInterface $passwordHasher, TranslatorInterface $translator, string $token = null): Response
{
    if ($token) {
        // We store the token in session and remove it from the URL, to avoid the URL being
        // loaded in a browser and potentially leaking the token to 3rd party JavaScript.
        $this->storeTokenInSession($token);

        return $this->redirectToRoute( route: '/reset-password/reset/{token}');
    }

    $token = $this->getTokenFromSession();
    if (null === $token) {
        throw $this->createNotFoundException('No reset password token found in the URL or in the session.');
    }

    try {
        $user = $this->resetPasswordHelper->validateTokenAndFetchUser($token);
    } catch (ResetPasswordExceptionInterface $e) {
        $this->addFlash( type: 'reset_password_error', sprintf(
            format: '%s - %s',
            $translator->trans(ResetPasswordExceptionInterface::MESSAGE_PROBLEM_VALIDATE, [], 'ResetPasswordBundle'),
            $translator->trans($e->getReason(), [], 'ResetPasswordBundle')
        ));
    }

    return $this->redirectToRoute( route: '/reset-password');
}

// The token is valid; allow the user to change their password.
$form = $this->createForm( type: ChangePasswordFormType::class);
$form->handleRequest($request);

if ($form->isSubmitted() && $form->isValid()) {
    // A password reset token should be used only once. remove it.
}

```

Figure 9: ResetPasswordController01

```

        return $this->render( view: 'reset_password/check_email',
            [
                'resetToken' => $resetToken,
            ]
        );

    }

    /**
     * Validates and process the reset URL that the user clicked in their email.
     */
    ± Maxime Lemée
#[Route('/reset/{token}', name: 'app_reset_password')]
public function reset(Request $request, UserPasswordHasherInterface $passwordHasher, TranslatorInterface $translator, string $token = null): Response
{
    if ($token) {
        // We store the token in session and remove it from the URL, to avoid the URL being
        // loaded in a browser and potentially leaking the token to 3rd party JavaScript.
        $this->storeTokenInSession($token);

        return $this->redirectToRoute( route: '/reset-password/reset/{token}' );
    }

    $token = $this->getTokenFromSession();
    if (null === $token) {
        throw $this->createNotFoundException('No reset password token found in the URL or in the session.');
    }

    try {
        $user = $this->resetPasswordHelper->validateTokenAndFetchUser($token);
    } catch (ResetPasswordExceptionInterface $e) {
        $this->addFlash( type: 'reset_password_error', sprintf(
            format: '%s - %s',
            $translator->trans(ResetPasswordExceptionInterface::MESSAGE_PROBLEM_VALIDATE, [], 'ResetPasswordBundle'),
            $translator->trans($e->getReason(), [], 'ResetPasswordBundle')
        ));
    }

    return $this->redirectToRoute( route: '/reset-password' );
}

// The token is valid; allow the user to change their password.
$form = $this->createForm( type: ChangePasswordFormType::class );
$form->handleRequest($request);

if ($form->isSubmitted() && $form->isValid()) {
    // A password reset token should be used only once. Remove it.
}

```

Figure 10: ResetPasswordController02

```

<footer>
    ...
    <div class="footer_montagne">
        ...
    </div>
    <div class="footer-sup">
        ...
        <div class="icon">
            <a href="https://www.instagram.com" aria-label="instagram">
                <svg xmlns:xlink="http://www.w3.org/1999/xlink" width="35" xmlns="http://www.w3.org/2000/svg" ...>
            </a>
            <a href="www.twitter.com" aria-label="twitter">
                <svg xmlns:xlink="http://www.w3.org/1999/xlink" width="35" xmlns="http://www.w3.org/2000/svg" ...>
            </a>
            <a href="www.pinterest.com" aria-label="pinterest">
                <svg xmlns:xlink="http://www.w3.org/1999/xlink" width="35" xmlns="http://www.w3.org/2000/svg" ...>
            </a>
            <a href="www.facebook.com" aria-label="facebook">
                <svg xmlns:xlink="http://www.w3.org/1999/xlink" width="27" xmlns="http://www.w3.org/2000/svg" ...>
            </a>
        </div>
    </div>
    <div class="footer-inf">
        <h4>© 2023 Copyright: SnowTricks </h4>
        <h4><a href="">Conditions générales</a></h4>
    </div>
</footer>

```

Figure 11: *_footer.twig.html*

```

<header class="top-nav">
    ...
    <div>
        <a href="{{ path('/') }}><svg xmlns:xlink="http://www.w3.org/1999/xlink" width="359" xmlns="http://www.w3.org/2000/svg" ...>
    </a>
    </div>
    <h2>Le site collaboratif de référence sur le snowboard</h2>
    {% if app.user %}
        <div class="user-avatar">
            <h3>Bienvenue {{ app.user.nickname }}</h3>
        </div>
    {% endif %}
    <input id="menu-toggle" type="checkbox"/>
    <label class='menu-button-container' for="menu-toggle">
        <div class='menu-button'></div>
    </label>
    <ul class="menu">
        {% if app.user %}
            {% if not path('/') %}
                <li><a href="{{ path('/') }}>Page d'accueil</a></li>
            {% endif %}
            <li><a href="{{ path('/user/{slug}', {'slug':app.user.slug}) }}>Page profil</a></li>
            <li><a href="{{ path('/trick/new') }}>Créer un trick</a></li>
            <li><a href="{{ path('/logout') }}>Se déconnecter</a></li>
        {% else %}
            <li><a href="{{ path('/login') }}>Se connecter</a></li>
            <li><a href="{{ path('/register') }}>S'inscrire</a></li>
        {% endif %}
    </ul>
</header>

```

Figure 12: *_header.twig.html*

```

#[Route('/{slug}', name: 'app_trick_show', methods: ['GET', 'POST'])]
public function show( EntityManagerInterface $manager ,Request $request, Trick $trick, CommentRepository $commentRepository, PaginatorInterface
{
    $comment = new Comment();
    $form = $this->createForm( type: CommentType::class, $comment);
    $form->handleRequest($request);

    $comment->setUser($this->getUser() );
    $comment->setTrick($trick);

    if ($form->isSubmitted() && $form->isValid()) {
        //dd($comment);
        $commentRepository->save($comment, flush: true);
        unset($form);
        $form = $this->createForm( type: CommentType::class);
        $this->addFlash( type: 'success', message: 'Ton commentaire à bien été créé');
    }
}

$dql   = "SELECT c FROM App\Entity\Comment c WHERE c.trick = ".$trick->getId()." ORDER BY c.createDate DESC";
$query = $manager->createQuery($dql);

$pagination = $paginator->paginate(
    $query, /* query NOT result */
    $request->query->getInt( key: 'page', default: 1), /*page number*/
    10 /*limit per page*/
);

return $this->render( view: 'trick/show', [
    'trick' => $trick,
    'form' => $form,
    'pagination'=>$pagination
]);
}

```

Illustration 27: TrickController.php Route Show

The screenshot shows the Owasp Zap interface during a scan of a local proxy at port 8080. The left sidebar lists various alerts found, such as PII Disclosure, Content Security Policy (CSP) Header Not Set, and Missing Anti-clickjacking Header. The main pane provides a detailed view of one specific alert, likely related to sensitive data exposure, showing the raw HTTP response and its content.

Illustration 28: Owasp Zap

```

.autoplay-progress svg {
    --progress: 0;
    position: absolute;
    left: 0;
    top: 0;
    z-index: 10;
    width: 100%;
    height: 100%;
    stroke-width: 4px;
    stroke: var(--swiper-theme-color);
    fill: none;
    stroke-dashoffset: calc(125.6 * (1 - var(--progress)));
    stroke-dasharray: 125.6;
    transform: rotate(-45deg);
}

//position du compteur
.autoplay-progress {
    position: absolute;
    right: 16px;
    bottom: 16px;
    z-index: 999;
    width: 48px;
    height: 48px;
    display: flex;
    align-items: center;
    justify-content: center;
    font-weight: bold;
    color: var(--swiper-theme-color);
}

.swiper-slide img {
    display: block;
    height: 65%;
}

.swiper-slide {
    text-align: center;
    font-size: 18px;
    background: #fff;
}

```

Figure 13: Code css pour carrousel

Conditions Générales d'Utilisation (CGU) pour un Site Communautaire de Figures de Snowboard

Dernière mise à jour : 1er juin 2023

Les présentes **Conditions Générales d'Utilisation** (ci-après dénommées "CGU") régissent l'utilisation du site communautaire de figures de snowboard (ci-après dénommé "snowtricks"). Le Site permet aux utilisateurs de partager des figures de snowboard, de créer de nouvelles figures, d'écrire des commentaires et de s'inscrire pour accéder à un espace personnel. En utilisant le Site, vous acceptez les présentes CGU dans leur intégralité. Si vous n'acceptez pas ces CGU, veuillez ne pas utiliser le Site.

1. Acceptation des CGU

En utilisant le Site, vous déclarez avoir lu, compris et accepté les présentes CGU, ainsi que notre Politique de Confidentialité. Si vous êtes en désaccord avec l'une de ces conditions, veuillez ne pas utiliser le Site.

2. Utilisation du Site

- a. Inscription : Pour accéder à certaines fonctionnalités du Site et pour posséder un espace personnel, vous devez vous inscrire en fournissant des informations exactes, complètes et à jour. Vous êtes responsable de maintenir la confidentialité de vos informations de connexion et de restreindre l'accès à votre compte. Vous êtes entièrement responsable de toutes les activités effectuées sous votre compte.
- b. Contenu généré par les utilisateurs : Vous pouvez contribuer au Site en soumettant des figures de snowboard, des commentaires ou d'autres contenus (ci-après dénommés "Contenu généré par les utilisateurs"). En soumettant du Contenu généré par les utilisateurs, vous gardez le propriétaire de ce contenu ou avoir le droit de le publier. Vous accordez au Site une licence non exclusive, mondiale, perpétuelle, irrévocable, transférable et libre de redevances pour utiliser, reproduire, modifier, adapter, distribuer, afficher publiquement et créer des œuvres dérivées à partir de votre Contenu généré par les utilisateurs.
- c. Règles de conduite : Vous acceptez de ne pas utiliser le Site d'une manière qui pourrait nuire à d'autres utilisateurs ou porter atteinte à la sécurité ou au bon fonctionnement du Site. Vous vous engagez à respecter les lois en vigueur et à ne pas publier de contenu illégal, diffamatoire, offensant, discriminatoire, obscène ou pornographique.

3. Propriété intellectuelle

Le Site et son contenu (y compris, mais sans s'y limiter, les textes, les images, les vidéos, les logos et les marques de commerce) sont protégés par les lois sur la propriété intellectuelle. Vous vous engagez à respecter tous les droits de propriété intellectuelle relatifs au Site.

4. Limitation de responsabilité

- a. Utilisation du Site : Le Site est fourni "tel quel" et sans garantie d'aucune sorte. Nous ne garantissons pas que le Site sera exempt d'erreurs, de virus ou d'autres composants nuisibles. Vous utilisez le Site à vos propres risques.
- b. Contenu généré par les utilisateurs : Nous ne sommes pas responsables du Contenu généré par les utilisateurs.

Illustration 29: CGU

```

security:
    # https://symfony.com/doc/current/security.html#registering-the-user-hashing-passwords
    password_hashers:
        Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterface: 'auto'
    # https://symfony.com/doc/current/security.html#loading-the-user-the-user-provider
providers:
    # used to reload user from session & other features (e.g. switch_user)
    app_user_provider:
        entity:
            class: App\Entity\User
            property: email
firewalls:
    dev:
        pattern: ^/(_(profiler|wdt)|css|images|js)/
        security: false
    main:
        lazy: true
        provider: app_user_provider
        custom_authenticator: App\Security\LoginFormAuthenticatorAuthenticator
        logout:
            path: app_logout
            # where to redirect after logout
            # target: app_any_route

        # activate different ways to authenticate
        # https://symfony.com/doc/current/security.html#the-firewall

        # https://symfony.com/doc/current/security/impersonating\_user.html
        # switch_user: true

    # Easy way to control access for large sections of your site
    # Note: Only the *first* access control that matches will be used
access_control:
    - { path: ^/admin, roles: ROLE_ADMIN }
    - { path: ^/profile, roles: ROLE_USER }

when@test:
    security:
        password_hashers:
            # By default, password hashers are resource intensive and take time. This is
            # important to generate secure password hashes. In tests however, secure hashes
            # are not important, waste resources and increase test times. The following
            # reduces the work factor to the lowest possible values.
            Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterface:
                algorithm: auto
                cost: 4 # Lowest possible value for bcrypt
                time_cost: 3 # Lowest possible value for argon
                memory_cost: 10 # Lowest possible value for argon

```

Illustration 30: security.yaml

```

<?php

namespace App\Controller\Admin;

use ...

class DashboardController extends AbstractDashboardController
{
    #[Route('/admin', name: 'admin')]
    public function index(): Response
    {
        //return parent::index();

        //Option 1. You can make your dashboard redirect to some common page of your backend
        //
        // $adminUrlGenerator = $this->container->get(AdminUrlGenerator::class);
        // return $this->redirect($adminUrlGenerator->setController(OneOfYourCrudController::class)->generateUrl());

        // Option 2. You can make your dashboard redirect to different pages depending on the user
        //
        // if ('jane' === $this->getUser()->getUsername()) {
        //     return $this->redirect('...');
        // }

        // Option 3. You can render some custom template to display a proper dashboard with widgets, etc.
        // (tip: it's easier if your template extends from @EasyAdmin/page/content.html.twig)
        //
        return $this->render( view: 'admin/dashboard');
    }

    no usages
    public function configureDashboard(): Dashboard
    {
        return Dashboard::new()
            ->setTitle( title: 'Snowtricks Administration');
    }

    no usages
    public function configureMenuItems(): iterable
    {
        yield MenuItem::linkToDashboard( label: 'Dashboard', icon: 'fa fa-home');
        yield MenuItem::linkToRoute( label: 'Retour au site', icon: 'fa fa-home', routeName: 'app_homepage');
        yield MenuItem::linkToCrud( label: 'Trick', icon: 'fa fa-snowflake-o', entityFqn: Trick::class);
        yield MenuItem::linkToCrud( label: 'Utilisateur', icon: 'fa fa-user', entityFqn: User::class);
        yield MenuItem::linkToCrud( label: 'Commentaire', icon: 'fa fa-comment', entityFqn: Comment::class);
        yield MenuItem::linkToCrud( label: 'Groupe', icon: 'fa fa-object-group', entityFqn: GroupTrick::class);
    }
}

```

Illustration 31: DashboardController