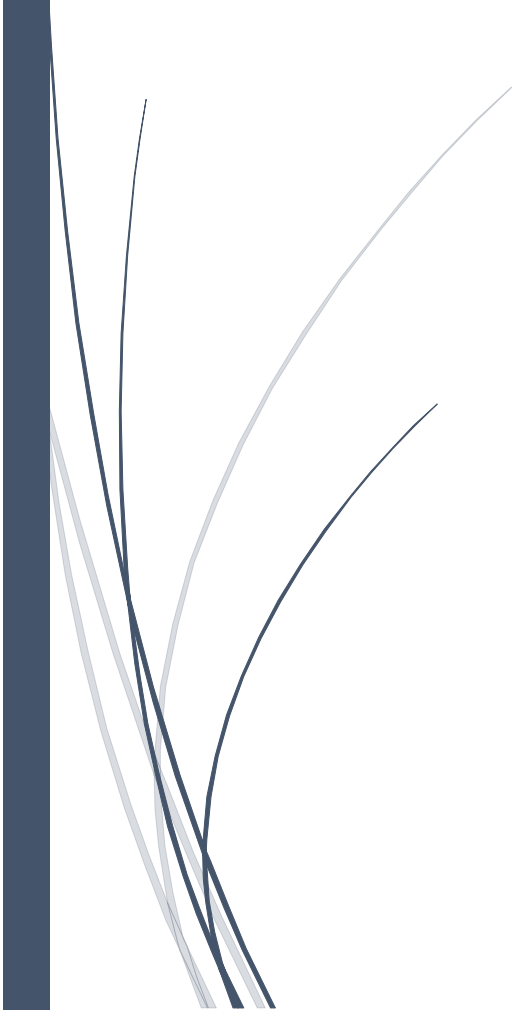
10/06/2021

Rapport

SwissCulture



PRO-A-07

HEIG-VD

Table des matières

1 Introduction

Le domaine de la culture est très impacté par la crise sanitaire, en grande partie à cause de la fermeture des lieux publics lors du de la crise sanitaire causée par le Covid19.

Le projet SwissCulture (https://www.swissculture.tk) vise à permettre aux institutions culturelles de mettre en avant leur contenu et leurs expositions de manière numérisée, en publiant des images et illustrations accompagnées d'une description sous forme de "visites virtuelles".

Le projet est une application web permettant aux utilisateurs professionnels de partager du contenu afin de faire découvrir leurs activités culturelles et potentiellement susciter de l'intérêt chez les visiteurs.

L'utilisation du site est gratuite pour tout le monde, mais la publication de contenu est réservée aux institutions qui doivent être vérifiées manuellement pas les administrateurs de l'application.

1.1 Description de l'application

Les utilisateurs de SwissCulture (https://www.swissculture.tk) peuvent naviguer facilement sur le site afin d'y trouver des visites virtuelles publiées par diverses institutions culturelles. Ces utilisateurs peuvent également facilement se connecter grâce à leur compte Facebook, afin de pouvoir, dans le futur (certaines fonctions ne sont pas encore disponibles), ajouter des institutions dans ses favoris, suivre des visites selon des catégories, aimer des visites. Les inscriptions des utilisateurs ne sont pas vérifiées. Ainsi l'application est rapide à utiliser.

Les institutions culturelles, doivent quanta elle remplir un formulaire d'inscriptions afin que les administrateurs du site puissent les vérifier manuellement. Ceci permet de vérifier qu'il s'agit d'une inscription sérieuse afin d'éviter que des contenus non qualitatifs soient publier.

Ces institutions peuvent uploader des photos pour créer des visites. Lors de ces upload SwissCulture propose d'y ajouter un filigrane en forme de texte ou de cachet afin de les aider à protéger leur photo.

Il est ensuite possible de créer des visites virtuelles en gérant les images publiées et en y ajoutant leurs descriptions, auteurs et autres informations si nécessaire.

Exemple : Le château de Chillon présente des photos du château, de l'intérieur, des murailles, etc. L'institution crée également un fil conducteur permettant de visiter une partie du château pour donner envie aux visiteurs potentiels

1.2 Fonctions implémentées et non implémentées

Les fonctionnalités primaires (must have) du cahier des charge ont pu être implémentées.

Il est notable que les fonctions suivantes sont également disponibles :

- Ajout optionnelle d'un filigrane sur les photos. Cependant ceci est fait lors de l'upload de l'image, et non lors de la gestion des visites.
- Connexion via Facebook
- Création et management de visites virtuelles.

Au niveau de la sécurité, les protections contre les attaques XSS et SQL ont été implémentées. Les gestions des Tokens Facebook est sécurisée également.

Par manque de temps, les fonctions suivantes son manquantes (Fonctions qui peuvent facilement s'implémenter dans le futur) :

- Affichage des tendances du mois (Top des institutions et visites)
- Affichage des institutions favorites si le compte utilisateur en a.
- Affichage des catégories suivies si le compte utilisateur en a.
- Recherche et filtrage de visites par canton, ville, catégorie
- Informations de contact vers les créateurs de l'application

Il manque une implémentation qui est malheureusement impossible à mettre en œuvre. Le site devait lire le flux des pages Facebook de ses institutions afin d'afficher leurs événements sur le page de profils. Facebook n'autorise pas la lecture de ses pages. Il pourrait être implémenter dans le futur une gestion d'événement propre à l'application.

1.3 Changement du cahier des charges

Originellement, le cahier des charges prévoyait de faire la partie backend en node.js. Du temps a été perdu en début de projet dessus car il a été compris après la réalisation du cahier des charges que l'hébergeur (Infomaniak) n'accepte pas cette implémentation et le backend a dû être changé en PHP.

1.4 Evolutivité

L'application a été conçue pour la faire évoluer facilement. Même si les fonctions secondaires n'ont pas pu être toutes implémentées, elle ne prendrait pas trop de temps à être concrétisées.

Il a aussi été fait en sorte, dans la base de données, que les médias publiés par les institutions ne soient pas simplement des photos. Si l'application est étendue, il serait possible d'ajouter des médias sous forme de gif, vidéos ou audios.

Le format utilisé a été créé pour des besoins nationaux, cependant, avec peu de changements il est possible de la convertir en une application internationale.

1.5 Apprentissage

Les leçons suivantes ont été apprises :

- Se renseigner sur les exigences des hébergeurs.
- L'apprentissage de nouvelle technologie prends beaucoup de temps.
- L'estimation du temps peut être très compliquée.
- Faire des hiérarchies des fonctions les plus importantes au moins importantes aides à gérer les priorités.
- Si possible, utiliser des technologies connues par la plupart des collègues.
- Prendre le temp de communiquer pour bien se coordonner aide à faire le travail efficacement.

Point positif

- La communication de l'équipe a bien fonctionné
- La segmentation des tâches (principales, secondaires, bonus) a été bien gérée.
- Beaucoup de travail au début du projet a permis de ne pas être pris au dépourvus et d'être proactif.

1.6 Potentiel financier de l'application

Comme décris dans le cahier des charges, le financement pourrait être assuré par ces exemples :

- Avec un partenariat public-privée, par exemple avec l'office de la Culture (confédération). On peut aussi imaginer un financement de la part des cantons, voire de certaines communes très touristiques.

Ce point peut être négocié avec la fédération suisse du tourisme.

Site : https://www.stv-fst.ch/fr

- Plusieurs fonctionnalités permettant de financer l'application :
  - Un compte premium(payant) destiné aux institutions qui leur permettraient d'organiser des événements depuis l'application. Par exemple :
    - Pouvoir organiser des vernissages pour les galeries d'arts.
    - Faire appel au mécénat.
  - Les institutions pourraient mettre en place des visites virtuelles payantes et exclusives sur lesquelles on prendrait une commission.
  - Qu'une institution puisse payer pour se mettre elle-même ou une de ses visites en avant sur la page d'accueil de l'application.

2 Organisation du code

2.1 Arborescence et contenu

L'organisation du code se définit par une partie backend en PHP et une partie Angular (TypeScript) en front-end.

2.2 Backend

Cette partie écrite en PHP a pour objectif principale de faire la communication entre le front-end et la base de données. Plus en détail, il y les dossiers suivants :

- api

  Ce dossier contient les fichiers où arrivent les requêtes depuis le front-end

Dossier	Description
authentification	La connexion / Les Logins.
home	Appelé notamment par la page d'acceuil. Obtenir les visites d'une catégorie Obtenir une catégorie
mediaManager	Gère les upload et les watermarks
profile	Concerne le profil de l'utilisateur.
visit	Concerne les visites

- media
  - Stock les images uploadées des visites de profil ou le cachet du watermark.
- modele
  - Gère en backend les user, profile et getVisite
- utils
  - Gestion de l'authentification
  - Librairie php
  - Le fichier composer.json contient toutes les dépendances

2.3 Frontend

La partie Frontend est la partie visible du site web. Celle-ci communique entre l'utilisateur et le backend.

Cette partie a été fait grâce à Angular 11. Comme wikipedia le décrit, Angular est un framework côté client, open source, basé sur TypeScript, et co-dirigé par l'équipe du projet « Angular » à Google et par une communauté de particuliers et de sociétés.

Il permet la création d'applications Web et plus particulièrement de ce qu'on appelle des « Single Page Applications » : des applications web accessibles via une page web unique qui permet de fluidifier l'expérience utilisateur et d'éviter les chargements de pages à chaque nouvelle action.

Le Framework est basé sur une architecture du type MVC et permet donc de séparer les données, le visuel et les actions pour une meilleure gestion des responsabilités.

Dans src/app se trouvent différents “components” qui sont pour la plupart décrits par le dossier conception technique. Les components, sont des décorateurs qui marque une classe en tant que composant d’Angular et fournit des métadonnées de configuration qui déterminent comment le composant doit être traité, instancié et utilisé au moment de l'exécution.

Source : https://fr.wikipedia.org/wiki/Angular

L'arborescence est la suivante :

- e2e (s’occupe de la connexion lors du serve)
- node_module (ce sont les imports des modules utilisés dans le projet)
- src/app (Les components réalisés pour le projet)
    - accueil
    - authentification
    - category-visit
    - list-institution
    - list-visit
    - media-manager
    - media-manager-dialog
    - profile
    - shared
    - upload-img
    - visit
- Dans src, il y a des fichiers de setting et de gestion de path pour le site.

3 Package de test/installation

3.1 Instructions d'installation

3.1.1 Sur le site web officiel

L'application SwissCulture est un site web hébergé sur Infomaniak. Il est donc simplement possible de se connecter sur https://swissculture.tk. Le lien amène sur la page d'accueil du site.

- Les photos mises sur le site dans les visites sont nos propres photos
- La photo de la page d'accueil ne vient pas de nous mais nous avons le droit de l'utiliser. Elle provient d'ici : https://besthqwallpapers.com/cities/lausanne-cityscapes-summer-swiss-cities-lake-geneva-93557

3.1.2 Instruction de test

Il est possible de se connecter à l'application avec votre propre compte facebook. Néanmoins, votre compte sera un compte prive et vous ne pouvez pas devenir une institution par vous-même car cela nécessite une validation dans la base de données de notre part.

Pour faciliter les tests, les utilisateurs facebook suivant ont été configurés

Adresse email	Mot de passe	Type de compte
[redacted]	[redacted]	Institution
[redacted]	[redacted]	Institution
[redacted]	[redacted]	Prive

3.1.3 Tourner l'application en local

Le code source ainsi que la base de données sont mis à disposition. Ceci permet de pouvoir tourner l'application en local. Le code source est disponible sur le git de l'application https://github.com/david-pellissier/PRO-A-07.

Afin de pouvoir tourner l'application en locale, il faut pouvoir faire tourner Angular. Pour ce faire, il est nécessaire d'avoir Node.js et npm package manager. Les instructions pour télécharger ces composants se trouvent au début de cette page : https://angular.io/guide/setup-local.

Pour que l'application SwissCulture puisse communiquer avec la base de données, il a été testé deux infrastructures backend différentes :

3.1.3.1 Docker:

Les sources sont disponibles à cette adresse : https://github.com/david-pellissier/PRO-A-07/tree/main/Docker

Infrastructure

Quatre containers sont utilisés pour l'environnement de développement

- web (apache) : Port 80. Serveur Backend lié au dossier "Backend".

- composer : installe toutes les dépendances PHP. S'arrête automatiquement lorsqu'il a fini ses tâches
- db (mysql) : Port 3306. Base de données. Les scripts situés dans "docker/dump" sont importés au démarrage
    - adminer : Port 8080. Client web pour la gestion de la base de données
- login (user:password): admin:test ou root:test

Fichiers de configuration

Les configurations ci-dessous doivent être utilisées :

- Backend/db.ini

```
host=db
port=3306
database=swiss_culture_db
user=admin
password=test
```

- Backend/constante.php

```
define("SERVER_URL", "/var/www/html/Backend");
define("SERVER_URL_MEDIA_FRONTEND", "http://localhost/media");
```

- SwissCulture/src/app/app.component.ts

```
  static SERVER_URL_ROOT = 'http://localhost';
  static SERVER_BACKEND = AppComponent.SERVER_URL_ROOT + '/Backend';
```

Préparation à l'exécution

Cette procédure n'est à effectuer qu'à la première installation ou lorsqu'il y a eu des changements dans le docker-compose ou dans les scripts de la BDD

1. Copier les scripts de "data/scripts" dans "docker/dump"
2. Build les images

`docker-compose build` ou `docker compose build`

Selon votre version de docker (resp. Toolbox ou Desktop).

Démarrage des containers

Tout simplement '*docker-compose up*' ou '*docker compose up*' selon votre version.
Pour les arrêter, Ctrl+C suffit.

3.1.3.2 Wamp ou autres services locaux

Il faut adapter les fichiers de configuration selon l'installation. Les fichiers de configuration sont décrits sont décrits sur le readme.md du projet https://github.com/david-pellissier/PRO-A-07.

3.1.3.3 Lancer en localhost

Il ne reste plus qu'à lancer l'application en tapant les commandes npm install –force et ng serve –open dans le dossier SwissCulture. Cela lancera le site en localhost.

4 Information pour support

Pour contacter le support technique de SwissCulture, il suffit de taguer l'équipe dans le canal A-07 du groupe PRO-2021 sur Teams.

5 Description des composants

5.1 Accueil

Ce composant contient la page d'accueil de l'application Web. En plus d'un bandeau visuel dans le haut de la page, cette page permet sous forme de vignettes :

- Les 3 institutions les plus récentes
- Les 3 visites les plus récentes
- La totalité des catégories de visites disponibles.

Lors du clic sur la vignette d'une institution ou visite, l'utilisateur est redirigé sur la page de l'institution ou de la visite correspondante. Un bouton juste en dessus des vignettes permet également d'aller sur le composant *List-institution* ou *List-visit* qui permettent d'afficher la totalité des institutions ou visites disponibles.

Les vignettes des catégories renvoient sur le composant *category-visit* qui affiche la liste de toutes les visites disponibles pour la catégorie donnée.

5.2 Authentification

5.2.1 Facebook

Ce composant offre l'interface utilisateur en front-end permettant de se connecter avec le compte facebook.

5.2.2 Security-service

Ce composant contient un certain nombre de service permettant d'authentifier l'utilisateur sur le site, de protéger les route Angular (guard) ainsi que d'ajouter le bearer token à chaque requête de l'utilisateur connecté.

5.3 List-institution

Ce composant est accessible depuis les vignettes des institutions de la page d'accueil. Il permet d'afficher la liste de toutes les institutions disponibles, un clic sur une institution renvoie sur la page de l'institution en question.

5.4 List-visit

Ce composant est accessible depuis les vignettes des visites de la page d'accueil. Il permet d'afficher la liste de toutes les visites disponibles, un clic sur une visite renvoie sur la page de la visite en question.

5.5 Category-visit

Ce composant est accessible depuis les vignettes des catégories de la page d'accueil. Il permet d'afficher la liste de toutes les visites disponibles pour une catégorie donnée, un clic sur une visite renvoie sur la page de la visite en question.

5.6 Upload-img

Ce composant permet l'upload d'une ou plusieurs images depuis le frontend vers le backend. Les images sont uploadées dans le dossier correspondant à l'institution qui effectue l'upload. Seuls certains formats de fichiers sont autorisés, une double vérification côté frontend et backend permet de garantir que seuls ces formats seront stockés en backend.

Ce composant est utilisé dans le *media-manager* ainsi que dans la gestion d'une visite.

5.7 Media-manager

Ce composant fournis une interface pour la gestion d'images. Il permet d'une part d'uploader des images avec le composant *upload-img*, et d'une autre part d'afficher les images uploadées d'une institution. Lors du clic sur une image, le composant *media-manager-dialog* est appelé afin d'afficher un pop-up permettant la gestion d'une image en particulier.

5.8 Media-manager-dialog

Ce composant est appelé lors du clic sur une image dans le *media-manager*. Il ouvre un pop-up permettant de gérer une image en particulier :

- Récupération et modification des informations d'une image (Titre, auteur, description)
- Suppression d'une image

5.9 Profile

5.9.1 Private

Page de profil d'un utilisateur privé ou institution. Pour les utilisateurs privés, deux composants sont utilisés : un pour la suppression de compte et l'autre pour "devenir une institution".

Pour les institutions, affiche toutes ses informations et inclut le composant *InstitutionPrivate*.

5.9.2 InstitutionPrivate

Composant servant à modifier le profil privé d'une institution et gérer ses visites.

5.9.3 InstitutionPublic

Affichage du profil public d'une institution (si validée) avec la liste de ses visites.

5.10 Visit

Composant fait pour afficher les visites en elle-même, soit une image avec une description, un auteur, un titre et permettre de naviguer d'une image à l'autre en suivant le fil de la visite.

5.10.1 Visite-image-dialog

Gère l'affichage de la fenêtre d'information concernant chaque image d'une visite.

5.10.2 Visit-creation

Composant gérant la création d'une visite avec un titre, une description et une catégorie.

5.10.3 Visit-list-instit

Composant permettant de lister toutes les visites d'une institution y compris les visites non visibles par le publique. Cela permet d'accéder à la modification de ces dites visites.

5.10.4 Visit-list-public

Composant pour lister les visites publiques d'une institution. Soit uniquement les visites qu'une institution a indiquées comme publique/visible.

5.10.5 Visit-modif

Composant gérant la modification des visites et permettant d'accéder à l'ajout d'images dans une visite. Il permet aussi de visualiser les images déjà dans la visite et de les supprimer de la visite si besoin.

5.10.5.1 Visit-modif-dialog

Composant gérant l'ajout des images à une visite. Il gère la multi sélection si l'ordre dans la visite n'a pas d'importance.

6 Suivis Qualité

Ce document a pour but de présenter le suivis qualité tout au long du projet. Il décrit les méthodes mises en place afin d'assurer la qualité de la réalisation du projet.

6.1 Rôle

Le responsable qualité effectue une vérification du cahier des charges, il s'assure également que des protocoles de tests soient appliqués pour chaque composant. Pour finir, il vérifie que la gestion des erreurs et du support entre les membres du groupe soit assurée.

6.2 Gestion des tests

Afin de garantir le bon fonctionnement de l'application, une validation de chaque composant est effectuée à travers un protocole de tests.

Quand un composant est terminé, son auteur s'assure de son bon fonctionnement en établissant un protocole de tests couvrant tous les points nécessaires à la validation du composant. Il applique ensuite ce protocole de tests au composant en validant chaque test établis.

Une fois le protocole de tests validé, le composant est considéré comme fonctionnel et peut alors être intégré au projet courant.

6.3 Intégration du code au projet courant

Afin de gérer le code, la solution *Git* est utilisée ainsi qu'un repos sur *Github* (https://github.com/) pour le partage de code.

Chaque composant ou fonctionnalité est développée sur une branche spécifique qui lui est propre. Une fois la fonctionnalité implémentée et validée à l'aide du protocole de tests, une *Pull Request* est effectuée par l'auteur.

Un autre membre du groupe va alors tester la fonctionnalité puis, dans le cas où aucune erreur n'est constatée, va *merge* la *Pull Request* dans la branche principale du projet courant (branche *main*). Si au contraire des erreurs sont constatées durant le test, l'auteur de la fonctionnalité corrige alors les erreurs, puis un autre membre du groupe va tester à nouveau. Cette opération se répète tant que la fonctionnalité n'est pas pleinement fonctionnelle et ne peut donner lieu à un merge de la *Pull Request*.

Cette méthode permet d'effectuer 2 tests d'une fonctionnalité, d'abord par le protocole de tests mis en place par l'auteur, puis par un membre du groupe n'ayant pas pris part à son implémentation.

Deuxièmement, une branche principale est utilisée pour le projet courant et permet d'avoir une version du code n'intégrant que des fonctionnalités testées et validées.

Pour finir, la séparation de chaque fonctionnalité en une branche distincte permet une gestion simplifiée et précise de chaque partie du projet.

6.4 Gestion du support et des erreurs

Afin de gérer les erreurs ou problèmes lors du projet, un support est mis en place sur un canal de l'application de VoIP *Discord* ainsi que sur un groupe de l'application de messagerie *Telegram*.

Ces 2 supports permettent une communication en tout temps entre les membres du groupe, cela permet de discuter de problèmes rencontrés et de s'entraider afin de trouver une solution.

6.5 Normes de codage

6.5.1 Langue

Les Commentaire en français.
La BDD est en français
Les noms des variables, fonctions, fichiers sont en anglais

6.5.2 Frontend TSLint

En frontend on utilise l'outil TSLINT. Celui-ci impose des normes de codage en TypeScript, ce qui permet d'uniformiser le code

Lien du package npm : https://www.npmjs.com/package/tslint

7 Mentions légales

Selon la Loi fédérale sur la protection des données (RS 235.1)

Inscription en tant que privé

L'inscription est facultative, hormis pour les comptes professionnels.

Les informations demandées lors de l'inscription ne sont utilisées que dans un but d'authentification de l'utilisateur.

La création du compte se fait automatiquement à la première connexion sur le site lorsque l'utilisateur se connecte avec son compte Facebook. L'utilisateur obtient un compte privé.

Il n'est pas nécessaire de se connecter au site pour voir les visites.

Inscription en tant qu'institution

Il est nécessaire d'être connecté en tant que compte privé pour pouvoir faire une demande afin de devenir une institution.

Les informations demandées lors de l'inscription ne sont utilisées que dans un but d'authentification de l'utilisateur et de s'assurer de son identité. Les administrateurs se réservent le droit de contacter par email l'utilisateur afin de s'assurer qu'il représente bien une institution.

Compte

Les données Facebook récupérées sont les suivantes :

- *Email*

A chaque connexion, l'adresse email du compte Facebook de l'utilisateur est récupérée.

Son adresse email peut être utilisée pour contacter l'utilisateur, elle n'est cependant pas utilisée pour envoyer des notifications.

- *Identifiant Facebook*

Son identifiant Facebook est également récupéré et stocké dans la base de données afin d'identifier l'utilisateur. Cette information est privée et n'est pas diffusée.

- *Nom du profil*

Le nom du profil Facebook est récupéré et stocké lors de la création du compte. Pour les institutions, il sera visible sur le site, public, et celui-ci peut être modifié. Pour les utilisateurs privés, le nom de profil n'est affiché qu'à lui-même et ne peut être modifié directement sur le site.

- *Autres informations*

Les autres informations de profil Facebook récupérées en frontend, par exemple l'image de profil, ne sont ni utilisées ni stockées dans la base de données. Elles ne sont pas transmises en backend et le serveur de SwissCulture ne fait pas de requête à l'API Facebook pour les récupérer.

Les informations présentent dans un compte ne sont pas transmises à des tiers et ne sont utilisées qu'à des buts d'authentification.

L'utilisateur a accès en tout temps à ses données.

Suppression d'un compte

L'utilisateur privé peut en tout temps décider de supprimer son compte, aucune donnée ne sera gardée après suppression d'un compte.

L'utilisateur institution doit en faire la demande par email, aucune donnée ne sera gardée après suppression d'un compte.

Traitement des données personnelles

Toute donnée relative à l'utilisateur n'est aucunement transmise à des tiers.

Toute donnée relative à l'utilisateur est stockée en Suisse, aucune donnée n'est transmise au-delà du territoire Suisse.

L'utilisateur privé a accès en tout temps à ses données. Il ne peut pas directement les modifier sur le site mais il peut en faire la demande à notre adresse de contact.

A l'exception du profil public, toute donnée relative à l'utilisateur n'est pas visible sur le site.

Les informations contenues dans le profil public sont visibles par toute personne accédant au site.

Pour les utilisateurs privés, ceux-ci n'ont pas de profil public.

Les informations suivantes sont contenues dans le profil public :

- *Nom*
- *Domaine*
- *Catégorie*
- *Image de profil*
- *Description*
- *Adresse*

L'utilisateur garantit l'exactitude des informations qu'il fournit.

Sécurité des données

L'intégrité des données relatives à l'utilisateur est garantie par des mécanismes de sécurité.

Les données relatives à l'utilisateur sont protégées contre une utilisation abusive ou un vol par des mécanismes de sécurité.

Propriété intellectuelle

Tout contenu publié par une institution est protégé contre une utilisation non autorisée.

Les utilisateurs (connecté ou non) n'ont pas le droit d'utiliser, modifier, publier et reprendre à leur nom le contenu des visites, que ça soit à but commercial ou non.

L'institution se doit de garantir que le contenu utilisé respecte le droit de propriété intellectuelle.

Le contenu publié sur le site par une institution reste la propriété de l'institution.

Publication des données

L'institution se doit de garantir que toute donnée publiée n'enfreint aucunement le code pénal et civil Suisse, sans quoi le contenu cité sera retiré.

Adresse de contact

admin@swissculture.tk

8 Conclusion

Ce projet nous a permis de mettre en œuvre les principes appris durant le cours de GEN sur la gestion de projet, en utilisant Github et en mettant en place un processus Agile. Nous avons développé nos compétences en organisation et en travail d'équipe.

Il nous a également permis de découvrir le framework Angular ainsi que les principes d'échanges des données entre le Backend et le front-end.

Nous avons pu aussi découvrir de nombreuses API et librairie notamment :

- Pour implémenter l'authentification avec facebook
- L'ajout de la watermark sur une image
- Affichage des visites
- Le design avec la librairie Material