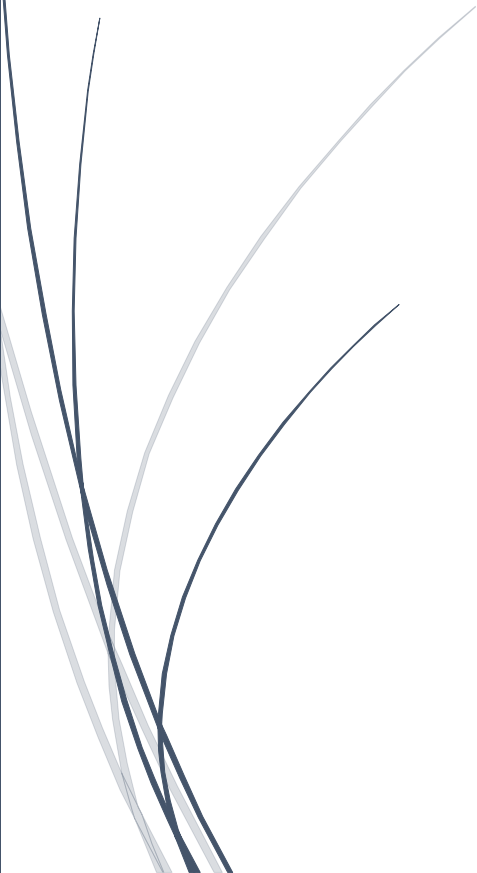




10/06/2021

Conception technique

SwissCulture



PRO-A-07
HEIG-VD

Conception technique

Ce document présente et explique les différents points cruciaux de la conception technique.

Table des matières

1	Documentation Infrastructure	3
1.1	Préparation côté Frontend	3
1.2	Préparation côté Backend	4
1.3	Préparation Base de données de production	4
1.4	Annexes	5
2	Modélisation	5
2.1	Diagramme de Contexte	5
2.2	Diagramme des conteneurs	6
2.3	Structure du serveur web	7
2.4	Base de données	7
2.4.1	Table	7
2.5	Schéma UML	10
3	API Facebook	11
3.1	FAQ	11
3.2	Configuration	12
3.2.1	Backend	12
3.3	Front-end :	12
3.4	Authentification	12
3.5	Procédure	12
3.6	Json Web Authentification	15
3.6.1	Génération des clés	15
3.7	Implémentation	15
3.7.1	Backend	15
3.8	Front-end	16
3.9	Sécurité	16
3.9.1	Vol de token	16
3.9.2	Méthode de vol	16

3.9.3	Faible XSS	17
3.9.4	XSRF	17
4	Événement Facebook sur les visites de SwissCulture.	18
4.1	Introduction.....	18
4.2	API graph	18
4.3	L'explorateur de l'API	18
4.4	Permissions	19
4.5	Nos pages	19
4.6	La communauté ne semble pas avoir de solutions	19
4.7	Alternatives ?	20
4.8	Conclusion	20
5	Watermark	20
5.1	Upload du cachet pour faire le watermark.....	20
5.2	Upload de l'image pour les visites avec le watermark approprié.	20
6	Checklist de sécurité	21

1 Documentation Infrastructure

Cette documentation explique la mise en production de l'application web *Swissculture* fonctionnant sous *Angular* ainsi que du Backend en PHP chez un hébergeur web.

L'hébergeur web utilisé ici est *Infomaniak* (<https://www.infomaniak.com/fr>) et leur solution d'hébergement *Web & Mail* (<https://www.infomaniak.com/fr/hebergement/web-et-mail/hebergement-web-et-mail>).

1.1 Préparation côté Frontend

1. Tout d'abord, il faut modifier le fichier **app.component.ts** dans le FrontEnd pour adapter à l'hébergeur web :
 - **app.component.ts** : Mettre l'URL du serveur (<https://www.swissculture.tk>) pour la constante `SERVER_URL_ROOT`.
 - **app.component.ts** : Changer la constante `API_FACEBOOK_ID` pour l'ID Facebook de l'API.
2. Effectuer la commande suivante pour la mise en production du code *Angular* :

L'option `--prod` permet d'optimiser les fichiers pour rendre le site plus rapide.

```
ng build --prod
```

Cette commande génère à la racine du projet *Angular* un dossier de production appelé : **dist**.

A noté : Si une erreur de type "*exceeded maximum budget*" survient, il faut modifier la valeur "*maximumError*" et "*maximumWarning*" dans le fichier *angular.json* du frontend en indiquant une valeur supérieure à la limite mentionnée dans le message d'erreur.

3. On met ensuite la totalité des fichiers qui se trouvent dans le dossier **dist** sur le serveur web dans le dossier **/sites/swissculture.tk**.

Afin de mettre des fichiers sur notre hébergeur web, il est possible soit d'utiliser le *Web FTP* fournis par *Infomaniak*, soit de configurer un compte FTP et d'utiliser notre propre client (*Filezilla*, *Cyberduck*, ...).

4. On doit ensuite mettre à jours les routes dans le **.htaccess** à la racine du site chez l'hébergeur.

```
RewriteEngine On
# If an existing asset or directory is requested go to it as it is
RewriteCond %{DOCUMENT_ROOT}%{REQUEST_URI} -f [OR]
RewriteCond %{DOCUMENT_ROOT}%{REQUEST_URI} -d
RewriteRule ^ - [L]
```

```
# If the requested resource doesn't exist, use index.html
RewriteRule ^ /index.html
```

1.2 Préparation côté Backend

1. Il faut modifier certains fichiers dans le backend pour que la liaison à la BDD fonctionne :

- **Backend/zone_protected/facebook.ini** : Modifier la variable *app_id* par l'ID Facebook de l'API.
- **Backend/constante.php** : Modifier les 2 variables statiques *SERVER_URL* et *SERVER_URL_MEDIA_FRONTEND* par le chemin physique du site chez l'hébergeur et le chemin vers le dossier *media* du backend.

```
define("SERVER_URL",
"/home/clients/5ca026c8911102419f284ec13f2e22e0/sites/swissculture.tk/Backend");

define("SERVER_URL_MEDIA_FRONTEND", "https://www.swissculture.tk/Backend/media");
```

- **Backend/zone_protected/db.ini** : Modifier les informations de connexion propre à hébergeur choisi pour la BDD.

```
host=rk4ha.myd.infomaniak.com
port=3306
database=rk4ha_swiss_culture_db
user=myUserName
password=myPassword
```

2. On met ensuite la totalité des fichiers qui se trouvent dans le dossier **Backend** sur le serveur web dans le dossier **/sites/swissculture.tk**.

Afin de mettre des fichiers sur notre hébergeur web, il est possible soit d'utiliser le *Web FTP* fournis par *Infomaniak*, soit de configurer un compte FTP et d'utiliser notre propre client (*Filezilla*, *Cyberduck*, ...).

1.3 Préparation Base de données de production

1. Il est nécessaire d'importer les 2 scripts SQL dans la base de données de production :

- **swissculture_script_01_structure** : Contient la structure de la base de données.
- **swissculture_script_02_data** : Contient les données pour remplir la base de données.

A noter : Le préfixe *rk4ha* devant le nom de la base de données est propre à *Infomaniak* et ne peut être changé, il faut donc modifier le nom de la BDD et de l'utilisateur dans les scripts en ajoutant ce préfixe.

1.4 Annexes

Déployer un site en production (liens utiles) :

<https://devtobecurious.com/comment-livrer-en-production-notre-application-angular/>

<https://angular.io/guide/deployment#fallback>

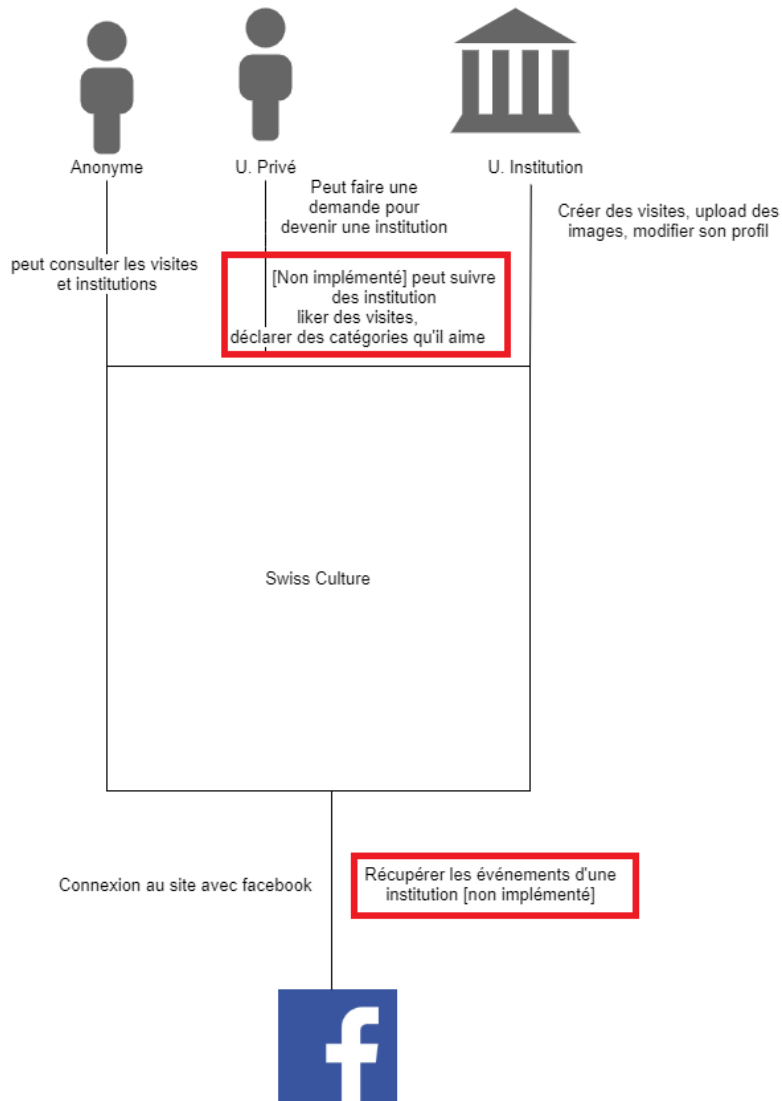
Configurer le .htaccess pour Apache (lien utile) :

<https://ngmilk.rocks/2015/03/09/angularjs-html5-mode-or-pretty-urls-on-apache-using-htaccess/>

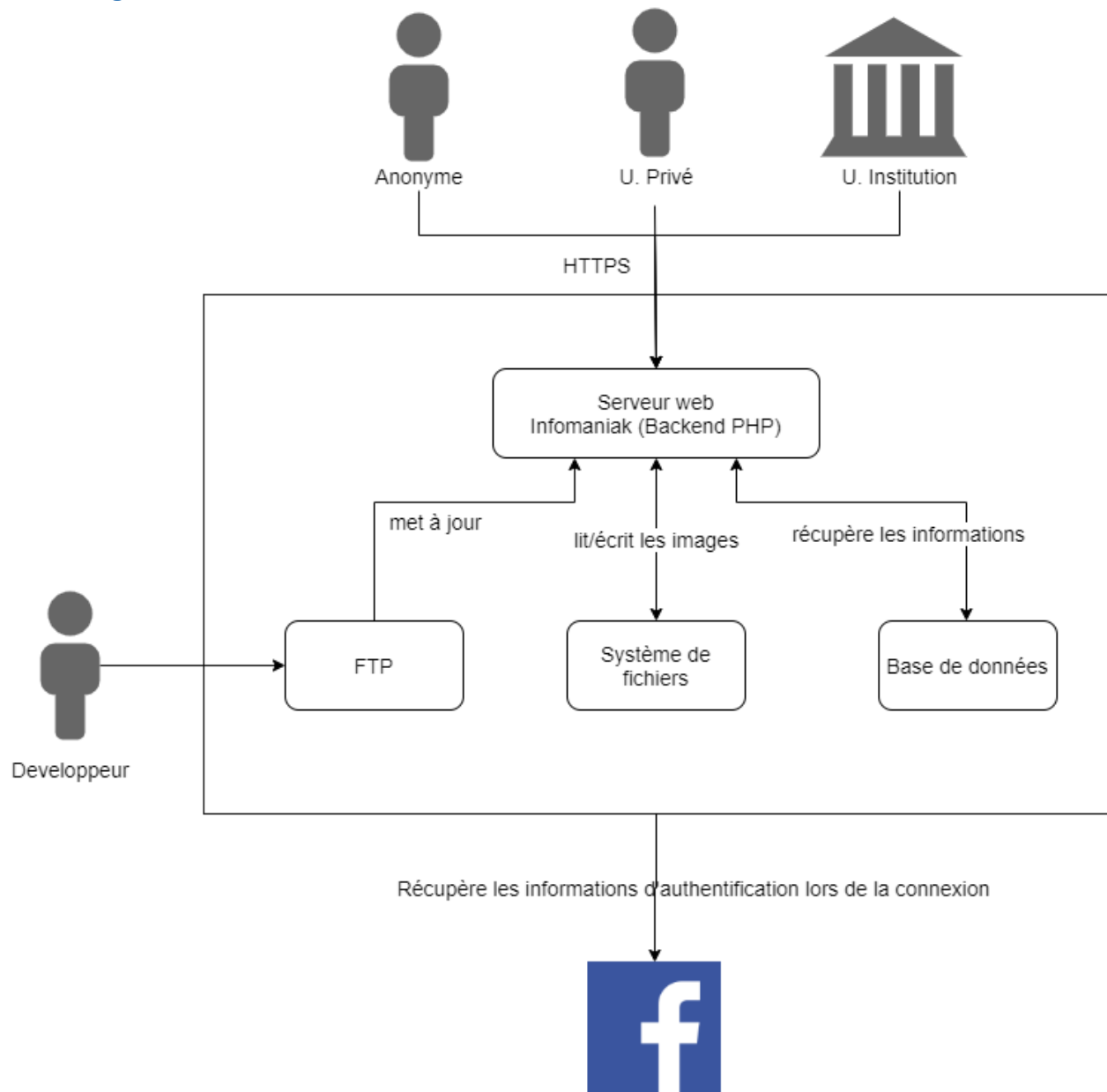
2 Modélisation

2.1 Diagramme de Contexte

En rouge, les fonctionnalités non implémentées

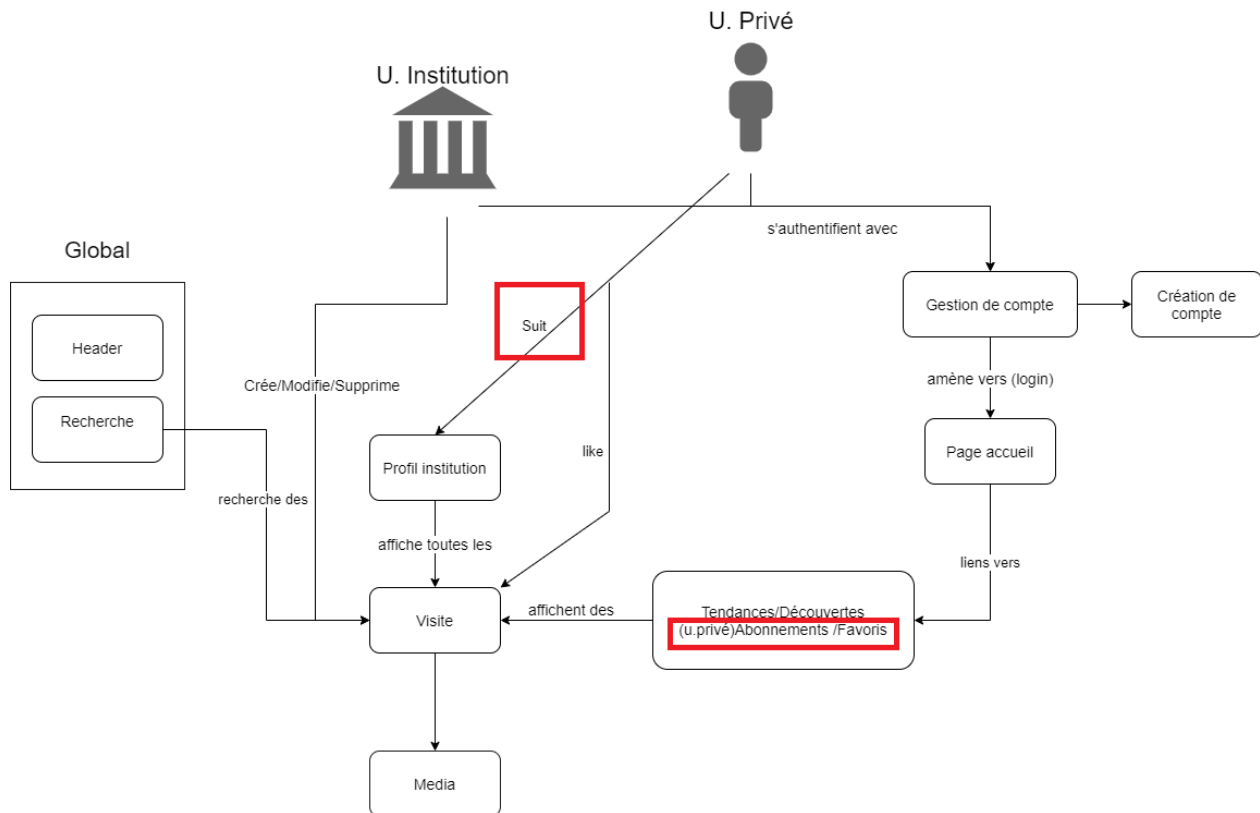


2.2 Diagramme des conteneurs



2.3 Structure du serveur web

En rouge il s'agit des fonctionnalités non implémentées actuellement.



2.4 Base de données

2.4.1 Table

Catégorie :

Cette table contient les catégories auxquelles appartiennent les visites.

Visite

Une visite à

- Un titre,
- Une description,
- Une date de création.
- Elle a également 1 booléen pour indiquer si la visite est publique ou non.

Media

- Un media à un **nom de fichier** et un nom

Le nom de fichier consistera en un hash à partir du contenu de l'image (du fichier). Une institution ne peut pas uploader 2 fois la même image.

On est parti du principe qu'une taille de 255 serait suffisante.

Compte

Un compte a :

- Un mot de passe (taille 255, à adapté selon la fonction de hash utilisé)
 - Null pour l'instant car on utilise uniquement le login facebook pour l'authentification
- Un email, celui-ci doit être unique
- Nom de profil, celui-ci sera affiché sur le profil publique de l'organisation. Celui-ci doit aussi être unique
- La date de la dernière connexion pour les notifications
- Un booléen estActivé qui indique si l'email a été vérifié [pas utilisé car on n'utilise seulement le login facebook]
- idFacebook : id facebook correspondant à son compte

email : La RFC5321 sur le protocole SMTP indique que la taille maximale d'un chemin est de 256 octets, en incluant la ponctuation et les séparateurs. La taille maximale d'une adresse email est alors de 254 car c'est le protocole SMTP qui rajoute les 2 derniers caractères.

Source 1 : <https://mydnic.be/post/longueur-dun-varchar-pour-un-champ-email>

Source 2 : <https://tools.ietf.org/html/rfc5321>

Institution

Une institution a :

- Nom d'établissement
- Un filigrane, voir implémentation watermark pour les détails
- Description de l'établissement
- Un booléen est validé, indiquant si le compte a été manuellement validé
- Sous forme d'association, une institution a une ville

Institution Compte (pas implémenté)

Des comptes peuvent être liés à des institutions et ont ainsi des droits d'administration, par exemple : l'ajout de visite.

Domaine

Une institution peut avoir un domaine : Art, Sport, Ville, Monument, Musée, etc.

Privé

Un privé possède une date de connexion et des associations lui permettant de Liker des visites, suivre des catégories, etc.

Ville

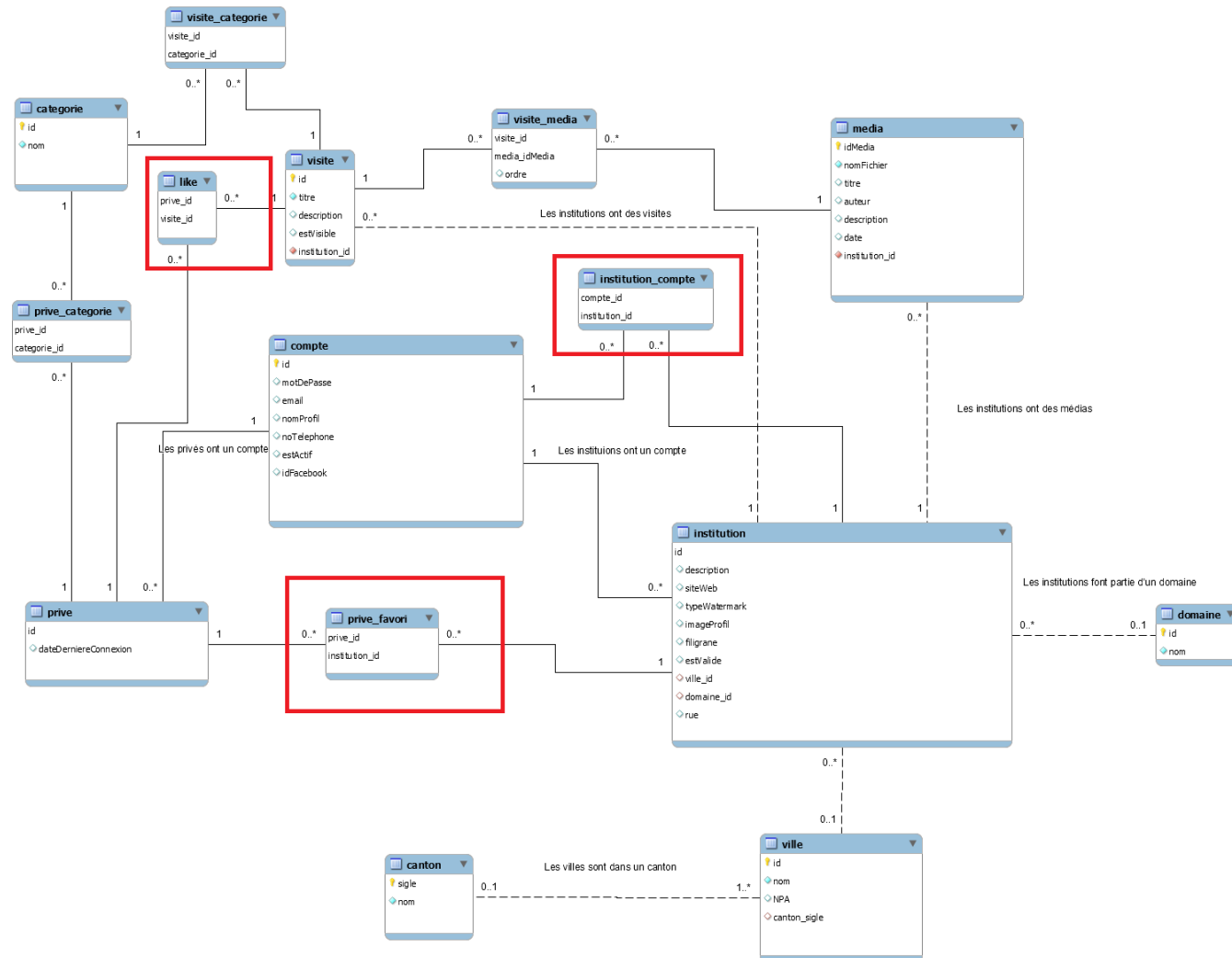
Une Ville a un nom et un NPA. Il peut y avoir plusieurs entrées pour un même nom de ville si celle-ci a plusieurs NPA (comme Lausanne)

Canton

Un canton possède un nom. Sa clé est son sigle/abréviation.

2.5 Schéma UML

Les parties en rouge correspondent à des fonctionnalités non implémentées.



3 API Facebook

3.1 FAQ

1) Est-ce que Facebook récolte des données de l'utilisateur sur le site ?

Non, une fois l'utilisateur est authentifié, c'est notre propre système d'authentification qui prend la main. L'utilisateur est authentifié sur le site avec un token que nous créons. Il n'y a par conséquent plus d'appel à l'API Facebook une fois les crédeniels validés à travers le jeton d'authentification renvoyé par Facebook.

Les seules données que Facebook peut potentiellement récupérer correspondent au moment où l'utilisateur utilise l'API Facebook lors de son authentification, par exemple le navigateur utilisé ou la date

2) Est-ce que Facebook vérifie l'usage que nous faisons de l'API ?

Chaque année, un administrateur de l'API (chez nous, SwissCulture) doit certifier que les données récoltées sont conformes aux conditions générales de Facebook. Cette certification consiste en de multiples cases à cocher et ne requière pas l'approbation directe d'un administrateur de Facebook. Pour résumé, les deux points principaux sont :

- De garantir la confidentialité des données récoltées
- D'en faire un usage éthique. Par exemple, il est interdit d'utiliser les données récoltées pour faire de la discrimination

A l'heure actuelle, nous stockons dans notre base de données :

- L'ID Facebook de l'utilisateur
- Son adresse email s'il en a une et qu'elle a été validée
- Son nom de profil

L'adresse email est récoltée pour permettre dans le futur d'envoyer des notifications. Cette fonctionnalité n'est pas implémentée, nous ne demandons pas à l'utilisateur de nous autoriser à lui envoyer des notifications. En cas d'implémentation d'un service de notification, il faudra obtenir le consentement de l'utilisateur.

Sources des conditions :

- <https://developers.facebook.com/devpolicy>
- <https://developers.facebook.com/terms>

3) Quel usage faisons-nous des données récoltées ?

Les usages sont décrits dans les mentions légales.

3.2 Configuration

3.2.1 Backend

Fichier de configuration : zone_protected/facebook.ini

Librairie php : <https://github.com/facebookarchive/php-graph-sdk/tree/master>

Celle-ci est la librairie officielle recommandées par facebook. Elle a néanmoins le défaut de ne pas supporter PHP 8.

3.3 Front-end :

Fichiers : app.component.ts et app.module.ts

Module utilisé : angularx-social-login

Lien : <https://www.npmjs.com/package/angularx-social-login>

Une nouvelle entrée a été ajoutée dans providers dans app.module.ts pour le module facebook. Pour ajouter d'autres login, comme google par exemple, il faudrait le déclarer ici de la même façon

```
providers: [  
  {  
    provide: 'SocialAuthServiceConfig',  
    useValue: {  
      autoLogin: false,  
      providers: [  
        {  
          id: FacebookLoginProvider.PROVIDER_ID,  
          provider: new FacebookLoginProvider(AppComponent.API_FACEBOOK_ID)  
        }  
      ]  
    } as SocialAuthServiceConfig,  
  },  
],
```

3.4 Authentification

Cette partie décrit le processus d'authentification dans son ensemble

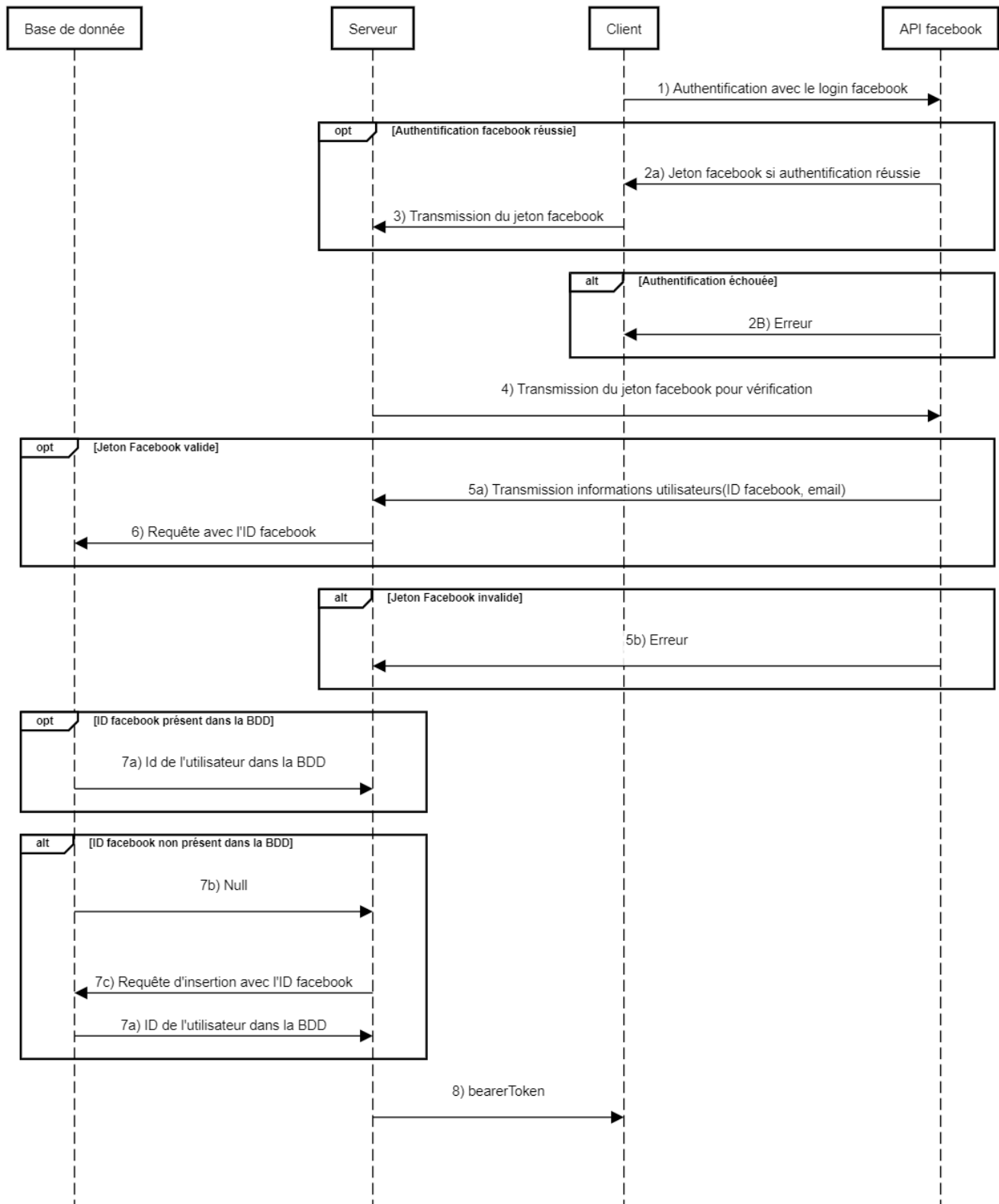
3.5 Procédure

- 1) L'utilisateur entre ses données d'authentification Facebook et le client envoie une demande à l'API Facebook
- 2) Si les credentials de l'utilisateur sont valides, l'API renvoie le jeton Facebook d'authentification ayant une durée de vie limitée
- 3) Notre application frontend envoie le jeton Facebook au backend
- 4) Le backend fait une requête à l'API Facebook avec le jeton.
- 5) L'API Facebook renvoie l'id de l'utilisateur Facebook si le jeton est valide
- 6) Le serveur fait une requête à la BDD avec l'id de l'utilisateur Facebook
- 7) a) Si l'utilisateur est déjà présent, la base de données renvoie l'id de l'utilisateur correspondant à l'utilisateur

- b) Null si aucun id correspondant à l'id facebook de l'utilisateur n'est présent
 - c) Le Backend fait une requête d'insertion à l'utilisateur pour l'insérer dans la base de données.
- 8) Avec l'id de l'utilisateur et son nom de profil, le Backend crée un JSON Web Token (jwt) avec sa clé privée et l'envoie au client

A chaque requête au Backend depuis le client, le JSON Web Token est attaché au header et vérifié par le backend. Si le client modifie le token, alors celui-ci ne sera plus valide.

Authentification



3.6 Json Web Authentication

Les informations permettant d'identifier l'utilisateur sont stockés dans un token au format JSON. Ce token est chiffré avec la clé privée du serveur. Il est transmis au client après authentification et stocké dans le local storage de son navigateur web. Le jeton est ensuite transmis dans le header à chaque requête du client au backend.

Le serveur va alors vérifier le token et récupérer les informations contenues à l'aide de sa clé privée.

3.6.1 Génération des clés

Les clés pour les token sont générées avec openssl et font 2048 bits.

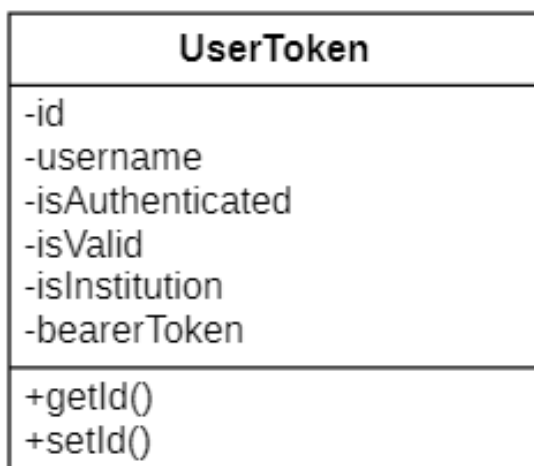
```
openssl genrsa -out private.pem 2048  
openssl rsa -in private.pem -pubout > public.pem
```

Nous utilisons du chiffrement asymétrique RSA car c'était plus simple à générer avec openssl, néanmoins nous l'utilisons pour faire du chiffrement asymétrique et la clé publique n'est jamais diffusée.

3.7 Implémentation

3.7.1 Backend

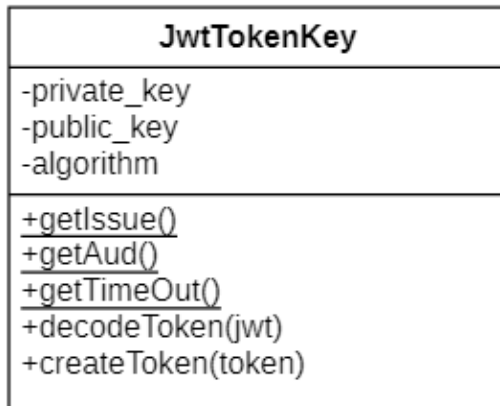
3.7.1.1 Class *UserToken*



En backend, l'utilisateur est représenté par la classe UserToken. Celle-ci possède 2 attributs : son ID au sein de la base de données et son token.

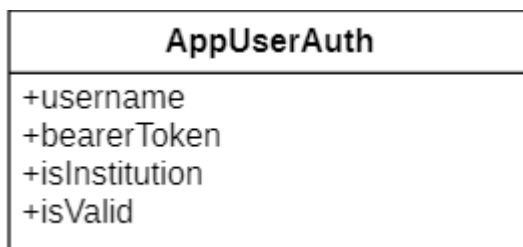
Les fonctions nécessitant que l'utilisateur soit connecté doit vérifier ses propriétés de l'objet.

3.7.1.2 Class JwtTokenKey



La classe JwtTokenKey représente un token d'authentification que nous créons. Il contient la clé privée et public qu'il utilise pour chiffrer et déchiffrer le token.

3.8 Front-end



La classe AppUserAuth représente un utilisateur en front-end. Cette classe est utilisée pour l'affichage des composant et comme 1ère barrière de sécurité. Néanmoins, c'est le bearer token qui fait foi en backend pour authentifier un utilisateur.

3.9 Sécurité

Cette partie traite de la sécurité de l'authentification.

3.9.1 Vol de token

Si un attaquant parvient à récupérer le token d'un utilisateur, il peut dès lors l'utiliser pour s'authentifier sur le site tant que la période de validité du token n'est pas terminée.

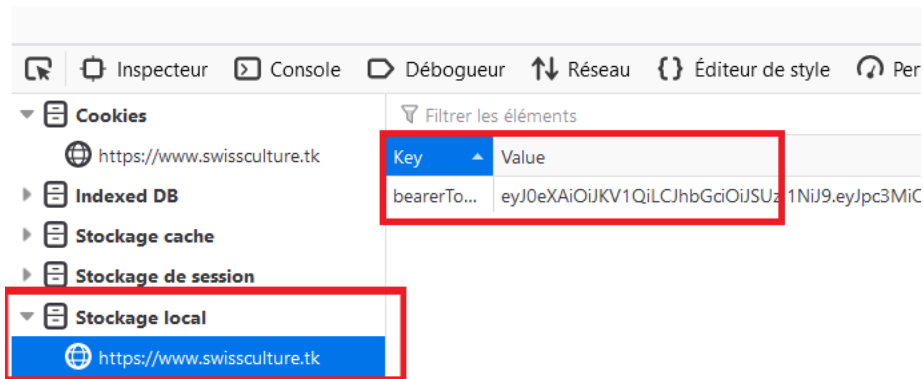
3.9.2 Méthode de vol

Avec un site compromis

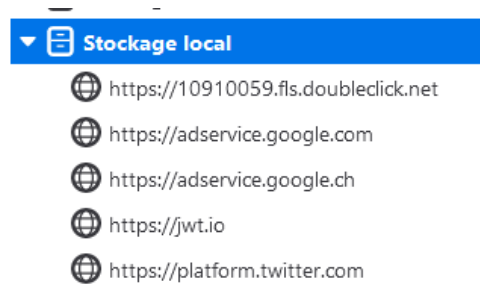
Il n'est pas possible de voler le token avec un site compromis sur lequel l'utilisateur se connecterait après son authentification. Le token est stocké dans le local storage de son navigateur web et celui-ci n'est accessible que sur le site web visité.

Exemple avec firefox :

L'utilisateur est actuellement sur la page du site et on peut voir que son token se situe dans son local storage



Tout en étant connecté au site, si je change pour aller sur le site <https://jwt.io/introduction>, on peut voir que swissculture n'apparaît pas. Il n'y a pas d'informations accessibles. Ainsi si au lieu de jwt.io on aurait un site contrôlé par l'attaquant, il ne pourrait pas l'utiliser pour voler le token.



3.9.3 Faille XSS

Un attaquant pourrait utiliser une faille XSS présente au sein du site afin de récupérer le token et l'envoyer par exemple à son adresse email.

Par conséquent, nous avons émis des vérifications et contrôle pour éviter ce genre de faille. De plus, le framework Angular possède également des protections en front-end contre le XSS

Lien de la documentation : <https://angular.io/guide/security#preventing-cross-site-scripting-xss>

3.9.4 XSRF

Une faille xsrf ne serait pas possible car le token est ajouté au header. Il faut alors le token pour espérer mener à bien l'attaque.

- Si on ne met que l'adresse url, alors le token n'est pas transmis avec le header ce qui fait que la requête n'est pas valide.
- Si l'attaquant crée son propre token, il ne sera pas valide
- L'attaquant n'a pas moyen de récupérer le token avec les principaux moyens d'attaques connus(xss). Nous ne sommes bien sûr pas à l'abri d'une faille dans le navigateur ou que

l'attaquant ait pris le contrôle à distance de l'ordinateur de l'utilisateur. Néanmoins ces cas-là sont indépendant de nous et on ne peut malheureusement pas y remédier.

4 Événement Facebook sur les visites de SwissCulture.

4.1 Introduction

Ce rapport met en avant une observation très décevante pour le projet SwissCulture et l'intention d'y ajouter des événements Facebook sur les visites proposées par les institutions. Pour résumer, Facebook, laisse lire les flux de pages publics seulement pour les partenaires marketings de Facebook ou si la page appartient au créateur de l'application qui veut exploiter les pages.

Les paragraphes qui suivent résument ce qui a été exploré :

4.2 API graph

La première proposition, a été de suivre le tutoriel suivant : <https://www.lije-creative.com/recuperation-contenu-page-groupe-facebook-api-graph-curl/>. Ce lien propose d'utiliser l'URL et le graphe API de Facebook.

Malheureusement, l'exemple effectué entièrement en PHP date, au plus tard, de 2012. Facebook a changé son API et ses conditions d'utilisation. Apparemment, après les procès de Mark Zuckerberg et les événements de Cambridge Analytica, Facebook a durci l'accès aux données.

De plus, ce tutoriel explique l'intégration d'événements pour un groupe et non une page publique. Les institutions qui nous intéressent utilisent les pages et non les groupes.

4.3 L'explorateur de l'API

Afin de tester l'API graph, Facebook met à disposition l'explorateur de ce même API qui permet de faire des requêtes avant d'implémenter ces dernières sur une application :

<https://developers.facebook.com/tools/explorer?method=GET&path=HEIGVD%2Fevents&version=v10.0>

Cet outil est fort intéressant. Cependant lors des tests, les tentatives de requêtes pour des événements sur des pages telles que HEIGVD/events mènent à l'erreurs suivante :

"(#100) Pages Public Content Access requires either app secret proof or an app token"

4.4 Permissions

Facebook demande actuellement de faire des requêtes de permissions afin de pouvoir accéder à des fonctionnalités et contrôler que les applications utilisant ses produits et ses API le fasse de façon appropriée. La page web suivante qui contient texte et vidéo <https://developers.facebook.com/docs/app-review> explique cela en détail. Le processus est lent, manuel et il faut justifier nos utilisations.

Les applications ont un mode développement qui donne accès à la plupart des fonctionnalités sans la permission de Facebook. Cependant, ce mode ne donne également pas accès au contenu public d'une page comme le paragraphe de cette page le mentionne https://developers.facebook.com/docs/apps/features-reference#reference-PAGES_ACCESS :

Pour les apps en mode développement, toutes les fonctionnalités hormis l'accès au contenu public de la Page et l'accès aux métadonnées publiques de Page sont activées pour tout utilisateur possédant un rôle dans l'app ou un rôle dans une entreprise ayant revendiqué l'app.

De plus la page https://developers.facebook.com/docs/public_feed/ indique :

L'accès à l'API Public Feed est limité à quelques éditeurs de médias et son utilisation requiert l'approbation préalable de la part de Facebook. Vous ne pouvez pas demander à utiliser cette API pour le moment.

Des contrats doivent être signés: <https://developers.facebook.com/docs/apps/features-reference/page-public-metadata-access> et <https://developers.facebook.com/docs/apps/features-reference/page-public-content-access>

Ce qui laisse comprendre que l'application SwissCulture a probablement aucune chance d'aboutir à ce genre de demande.

4.5 Nos pages

Les liens suivants montrent comment le propriétaire d'une application peut prendre des informations sur des pages qu'ils ont créés eux-mêmes. Mais pas par des tiers. [pages_read_engagement - Graph API](#) et [pages_read_user_content - Graph API](#).

4.6 La communauté ne semble pas avoir de solutions

Des utilisateurs de stackoverflow ont également essayés ce que nous voulions faire <https://stackoverflow.com/questions/49825680/how-can-i-get-the-events-for-a-facebook-page>. Le seul qui a trouvé une solution, a vu son projet refusé par Facebook.

4.7 Alternatives ?

Il existe des solutions non officielles et très floues sur la légalité (scraping). Pas de solution viable n'a été trouvée au 6 avril 2021, 04:05. Pas de recherche supplémentaire n'a été effectuée au-delà de cette date.

4.8 Conclusion

Il est préférable d'implémenter nous même les événements ou de simplement laisser l'institution donner le lien de ses événements Facebook. Après avoir effectué quelques recherches, aucun site proposant la fonctionnalité que SwissCulture veut offrir n'a été trouvé.

5 Watermark

Le watermark est la fonction permettant d'ajouter un filigrane lors de l'upload (component upload-img). Pour le cachet en forme d'image, il est conseillé de prendre un PNG d'un format d'environ 300x300 pixels. Le cachet est redimensionné selon la largeur de de l'image de visite. La taille du cachet est proportionnelle à une largeur d'une image de visite de 1080 pixels divisé par deux.

5.1 Upload du cachet pour faire le watermark.

Dans le code html, afin de faire l'upload du cachet du watermark, il faut appeler le component comme cela :

```
<app-upload-img isWatermark="true"></app-upload-img>
```

Ceci uploadera le cachet utilisé pour les watermark dans le chemin *Backend/media/<idInstitution>/wm*. De plus, cela modifie le champ « typeWatermark » de la table institution à 1 et insère le nom de l'image dans le champ filigrane (v. id 3).

	id	description	siteWeb	typeWatermark	imageProfil	filigrane	estValide	ville_id	domaine_id
<input type="checkbox"/>	1	Institution par défaut	https://fr.wikipedia.org/wiki/Wikip%C3%A9dia:Accueil_principal	0	NULL	It is a filigrane	1	1	2
<input type="checkbox"/>	3	Michael Test	lol ch	1	NULL	9c8f1a240667e9045a2741c2c16d892.jpg	1	2330	1

5.2 Upload de l'image pour les visites avec le watermark approprié.

Dans le code html, afin de faire l'upload d'une image pour les visites, il faut appeler le component comme cela :

```
<app-upload-img isWatermark="false"></app-upload-img>
```

Ceci uploadera l'image avec le bon watermark selon le type défini dans la base de données dans le chemin *Backend/media/<idInstitution>/visites*.

Et insère l'image dans la table média de la base de données.

	idMedia	nomFichier	titre	auteur	description	date	institution_id
<input type="checkbox"/>	1	ac30a098b236b77a69e3b32fe4638eb5.jpg	NULL	NULL	NULL	2021-05-12 13:49:14	3

Le type de water mark est défini dans le champ typeWatermark de la table institution (cf. capture d'écran plus haut).

Voici les différentes possibilités :

- **typeWatermark = 0**
Le cachet de l'image est le texte qui se trouve dans le champ filigrane.
- **typeWatermark = 1**
Le cachet est l'image dans le dossier *Backend/media/<idInstitution>/wm* de l'institution dont le nom est inscrit dans filigrane.
- **typerWatermark** est différent de 0 et 1
Aucun cachet n'est appliqué sur l'image de visite.

6 Checklist de sécurité

Sources :

- <https://pragmaticwebsecurity.com/files/cheatsheets/angularowasptop10.pdf>
- <https://snyk.io/blog/angular-security-best-practices/>

Cette checklist a été utilisée pour vérifier la sécurité de l'application

Description	Raison
Dépendances vulnérables	Peuvent amener des vulnérabilités
Session protégée par chiffrement	Afin d'éviter d'exposer des données sur l'utilisateur en clair
Vérification de l'intégrité des données envoyées	Peut amener à des erreurs de backend
Bindings safe ([href], [src], [style])	Pour profiter des mesures anti XSS de Angular
Pas de templates générés avec un input utilisateur	Car les templates générés dynamiquement ne sont pas protégés des faille XSS
Vérif de l'autorisation sur l'API dès que nécessaire	Pour éviter des accès non désirables
Fichiers backend non exposés	N'exposer que les fichiers nécessaires au frontend
Préparation des requêtes PDO	Afin d'éviter les injections SQL
Backend : retour après erreur	Il faut éviter d'afficher les exceptions PHP dans les réponses au Frontend car elles pourraient donner des informations sur la structure du code.