Rapport Maya Sprint 3 Mission 1

“Authentification à double facteurs”

Léandro VEYRENC-CORDERO - 06/11/2025

Index :

Rapport Maya Sprint 3 Mission 1

“Authentification à double facteurs”

Léandro VEYRENC-CORDERO - 06/11/2025

Mise en place de la solution :

Pour commencer à mettre en place l’authentification à double facteurs sur notre application Maya, certaines dépendances étaient requises comme 2fa pour gérer l’authentification et une extension 2fa-google-authenticator pour l’utiliser avec Google Authenticator et pour finir endroid/qr-code pour générer le qr code afin d’ajouter le code dans Google Authenticator plus facilement. Pour les installer, il nous suffit d'exécuter la commande *composer require <nom du module>*.

Pour continuer, j’ai mis à jour le firewall de l’application afin qu’elle utilise le bundle de l’authentification à double facteurs.

```
    two_factor:
        auth_form_path: 2fa_login           # Path or route name of the two-factor form
        check_path: 2fa_login_check         # Path or route name of the two-factor code check
        post_only: true                     # If the check_path should accept the code only as a POST request
        default_target_path: app_accueil    # Where to redirect by default after successful authentication
        auth_code_parameter_name: _auth_code  # Name of the parameter for the two-factor authentication code
                                              # (supports symfony/property-access notation for nested values)

        # Some two-factor providers need to be "prepared", usually a code is generated and sent to the user.
```

Ensuite, j’ai configuré le bundle pour qu’il utilise google authenticator en lui indiquant le nom du serveur, le nom à afficher dans l’application d’authentification, le nombre de chiffre du code, la template à utiliser et le temps de cache du code.

```
scheb_two_factor:
    google:
        enabled: true                   # If Google Authenticator should be enabled, default false
        server_name: maya.com           # Server name used in QR code
        issuer: La ferme Maya           # Issuer name used in QR code
        digits: 6                       # Number of digits in authentication code
        leeway: 0                       # Acceptable time drift in seconds, must be less or equal than 30 seconds
        template: security/2fa_form.html.twig   # Template used to render the authentication form
```

```
    security_tokens:
        - Symfony\Component\Security\Core\Authentication\Token\UsernamePasswordToken
        - Symfony\Component\Security\Http\Authenticator\Token\PostAuthenticationToken

    # A list of IP addresses or netmasks, which will not trigger two-factor authentication.
    # Supports IPv4, IPv6 and IP subnet masks.
    # à décommenter en mode développement pour ne pas passer par l'authentification 2fa
    # ip_whitelist:
    #     - 127.0.0.1  # One IPv4

    # By default the time to cache a 2FA-code is 60 seconds. After that the 2FA code should be
    # invalidated by the 2FA provider anyhow. Should your 2FA provider allow valid codes for
    # more than that, you should increase this amount!
    code_reuse_cache_duration: 60
```

Rapport Maya Sprint 3 Mission 1

“Authentification à double facteurs”

Léandro VEYRENC-CORDERO - 06/11/2025

Après avoir configuré l’authentification, il faut qu’elle soit incluse dans la connexion et la création des utilisateurs.

Pour ce faire, nous allons modifier l’entité donc l’objet correspondant à l’utilisateur. J’ai d’abord ajouter la nouvelle colonne pour la clé secrète puis ajouter les fonctions correspondantes : isGoogleAuthenticatorEnabled, getGoogleAuthenticatorUsername, getGoogleAuthenticatorSecret, setGoogleAuthenticatorSecret.
Pour que la base de données soit ensuite à jour, on exécute une migration.
Suite à cette modification de la base de données, une nouvelle colonne google_authenticator_secret est présente et mise à NULL pour tous les utilisateurs déjà inscrit.

Lors de la création d’un utilisateur, un code secret doit être généré.
Pour générer ce code, on va agir dans le controlleur de l’entité User plus particulièrement sur la méthode new en ajoutant l’interface dans les paramètres de la méthode et en ajoutant la génération et attribution du code secret à l’utilisateur.

```php
// génération clé secrète pour Google Authenticator
$secret = $googleAuthenticatorInterface->generateSecret();
$user->setGoogleAuthenticatorSecret($secret);
```

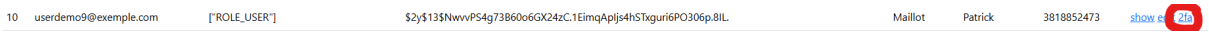
En testant, on voit bien que en créant un nouvel utilisateur, on voit bien que son code à été généré.

20	userdemo8@exemple.com	[]	$2y$13$NtEYAIP0X9BppK.pdP6TXOcD1bA2ECWaAJ0ukolY5FA...	Morel	Philippine	3863649730	*NULL*
21	userdemo9@exemple.com	[]	$2y$13$ZK2jgcObT0pu1BIpKrY1L.P4HI4djbR3YsYd3nAItkX...	Perrin	Auguste	3468954106	*NULL*
22	essai2fa@exemple.com	[]	$2y$13$vt6EEW3APT7WonWoCqSj2.nm8vMwowX9J9ArmyIBSX4...	Schmidt	Isabelle	0615852630	5PVQGXB7YIUMQCOKOVNZE

Ensuite, pour garder nos utilisateurs aléatoirement créé et mis à jour, nous allons modifier les Fixtures (données de test) en leur donnant la clé secrète générée précédemment.
Pour ce faire, on viens juste donner la clé secrète à l’entité avec la méthode setGoogleAuthenticatorSecret.

```php
// pour les tests, on utilise la même clé secrète pour le service Google Authenticator
// en production, chaque user a une clé personnalisée
$user->setGoogleAuthenticatorSecret('[redacted]');
```

Pour continuer, on va mettre en place le visuel du qr-code.
On va donc premièrement créer le bouton pour acceder au qr-code en rajoutant simplement : **2fa**
dans la vue des utilisateurs.



En ajoutant ce bouton, on a besoin de la route app_user_2fa qu’on va créer dans le controlleur des utilisateurs

Rapport Maya Sprint 3 Mission 1

“Authentification à double facteurs”

Léandro VEYRENC-CORDERO - 06/11/2025

```php
#[Route('/{id}/2fa', name: 'app_user_2fa', methods: ['GET'])]
public function user2fa(User $user): Response
{
    return $this->render('user/2fa.html.twig', [
        'user' => $user,
    ]);
}
```

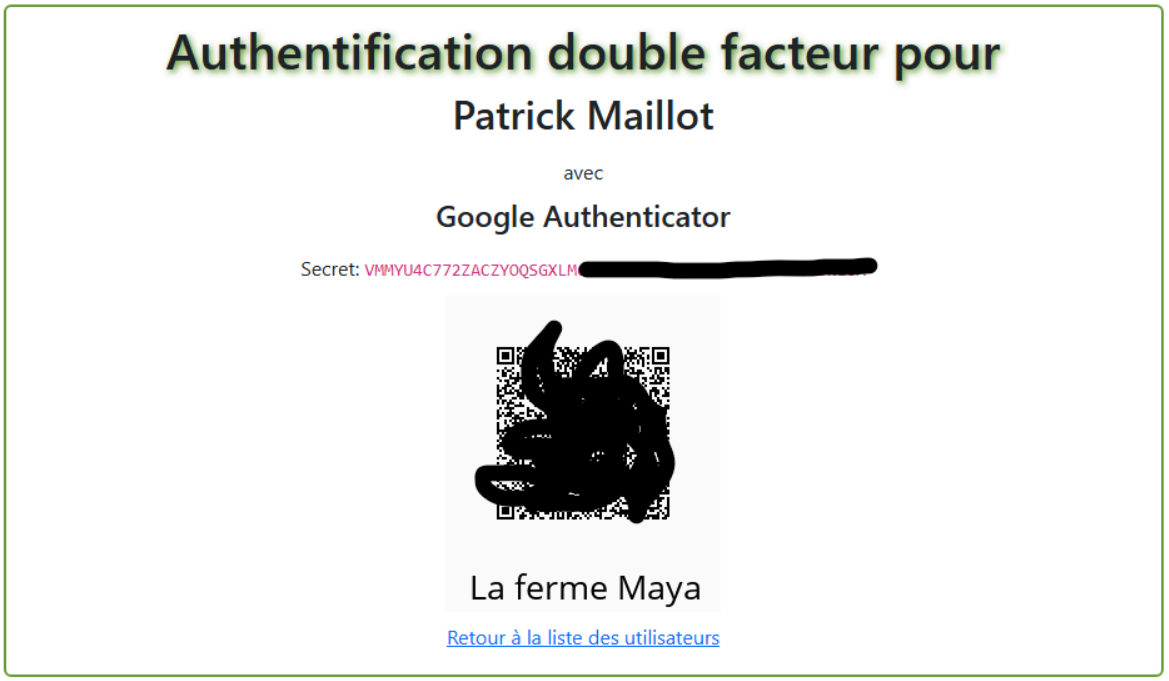
Cette route va nous afficher la vue correspondant au fichier 2fa.html.twig en correspondance avec un utilisateur mis en paramètre.
Voici la vue 2fa.html.twig qui affiche le nom et prenom de l’utilisateur, sa clé secrète ainsi que son qr-code :

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

{% block body %}

    <div class="col-md-6 contenu-blanc mx-auto text-center">
        <h1>Authentification double facteur pour</h1>
        <h2>{{ user.prenom ~ ' ' ~ user.nom}}</h2>
        <div>avec</div>
        <h4>Google Authenticator</h4>
        <div>Secret:
            <code>{{ user.googleAuthenticatorSecret }}</code>
        </div>
        <div><img src="{{ path("qr_code_ga", {'id': user.id }) }}"/></div>

        <a href="{{ path('app_user_index') }}">Retour à la liste des utilisateurs</a>
    </div>
{% endblock %}
```

Rapport Maya Sprint 3 Mission 1

“Authentification à double facteurs”

Léandro VEYRENC-CORDERO - 06/11/2025



Cette vue appelle la route qr_code_ga, elle est donc à créer dans un nouveau controlleur nommé QrCodeController.php

Cette route va nous afficher le QR code en fonction de l’utilisateur et donc de sa clé secrète.

Ensuite, pour que l’authentification à deux facteurs marche correctement, je dois modifier le controlleur de securité en ajoutant si on arrive à récupérer un utilisateur, ce qui veut dire qu’il a correctement saisi ses identifiants, on le redirige vers la route 2fa_login.

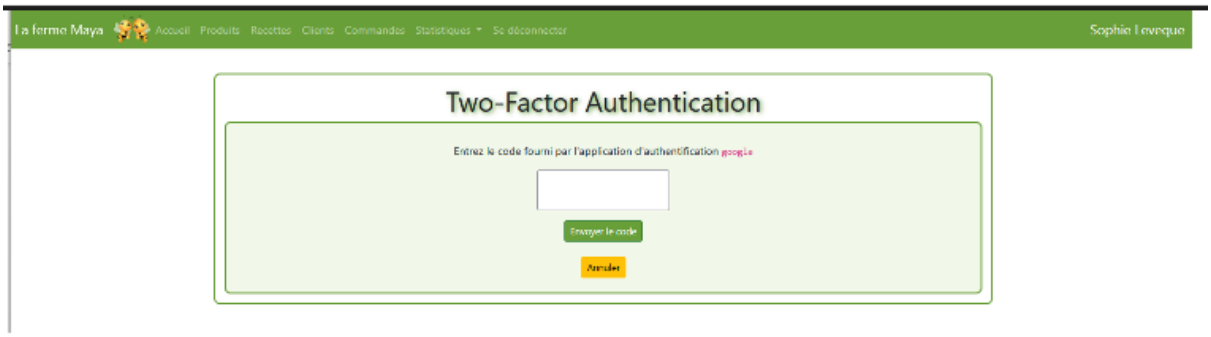
```
        // si après la première étape (login et mdp) nous avons un user
        // alors redirection vers la page de saisie du code d'authentification de Google
Authenticator
        if ($this->getUser()) {
            // dans security.yaml, nous avons indiqué la route 2fa_login
            // dans scheb_2fa.yaml, nous avons indiqué le template:
security/2fa_form.html.twig
            return $this->redirectToRoute('2fa_login');
        }
```

Rapport Maya Sprint 3 Mission 1

“Authentification à double facteurs”

Léandro VEYRENC-CORDERO - 06/11/2025

Pour qu’on puisse saisir le code donné par l’application d’authentification, il faut une page. Cette page récupère les données définis dans le security.yaml



Grâce à cette authentification, dans la navbar on vient pas seulement vérifier si un utilisateur est dans une variable mais si il est totalement connecté donc identifiant + 2fa avec : **{% if is_granted('IS_AUTHENTICATED_FULLY') %}**

Rapport Maya Sprint 3 Mission 1

“Authentification à double facteurs”

Léandro VEYRENC-CORDERO - 06/11/2025

Description du fonctionnement technique

Lorsqu’un utilisateur accède à notre application web Symfony, il n’est pas encore authentifié. Par conséquent, la première vue affichée est la page de connexion *security/login.html.twig*. L’utilisateur saisit alors ses identifiants (nom d’utilisateur et mot de passe) et clique sur le bouton de connexion.
À ce stade, Symfony valide les informations d’authentification via le système de sécurité configuré dans security.yaml.
Si les identifiants sont corrects, l’utilisateur est partiellement connecté : il est reconnu par l’application, mais doit encore passer la seconde étape d’authentification à deux facteurs (2FA).
L’utilisateur est ensuite redirigé vers la route *2fa_login*, qui correspond à la vue *security/2fa.html.twig*.
Cette page affiche un formulaire demandant à l’utilisateur de saisir le code de vérification généré par l’application Google Authenticator.
Le code saisi est ensuite envoyé et traité par les contrôleurs fournis par le package *scheb/2fa-bundle*, situé dans *vendor/scheb*.
Le bundle compare le code entré avec celui généré dynamiquement à partir de la clé secrète unique associée à l’utilisateur, stockée dans la base de données.

- Si le code correspond à celui attendu (dans la fenêtre de temps autorisée), la vérification est validée, et l’utilisateur est considéré comme entièrement connecté. Il est alors redirigé vers la page d’accueil ou toute autre route définie comme page de succès *default_target_path*.
- Si le code est incorrect, la page se recharge avec un message d’erreur, invitant l’utilisateur à réessayer.

Le processus repose donc sur le système de sécurité de Symfony (Security), la configuration du SchebTwoFactorBundle (scheb/2fa-bundle) et la génération de codes compatibles avec Google Authenticator.