



Rapport du projet Symfony

APPLICATION WEB – VISITEUR D'UN SALON

Réalisé par : Marwa MRABET, Charlys RETITA FERAUD, Said AGHZOU, Imen GOMRI



Introduction :

L'objectif est de réaliser une application Web permettant à une école sur un salon de saisir les informations des visiteurs.

Les informations attendues sont :

Civilité (obligatoire)
Nom (obligatoire)
Prénom (obligatoire)
Mail (obligatoire)
Code Postal (obligatoire)
Téléphone (optionnel)
Classe actuelle (Obligatoire)
Commentaire (Champ long)

Les attendus sont les suivants :

- Il faut être authentifié pour pouvoir saisir (les informations sont saisies par les membres de l'école).
- Il doit y avoir la possibilité de consulter les informations pour un salon donné .
- Un export des données doit être possible depuis le BackOffice.

Authentification :

Pour résoudre ce problème, nous avons mis en place une base de donnée pour enregistrer les informations des administrateurs dans le but d'avoir un identifiant (dans notre cas soit un username ou un email) et un mot de passe (password) pour se connecter à la page principale de notre application.

1-Enregistrement :

Pour s'enregistrer, nous avons créé un formulaire qui a comme paramètres (Username, Email, Password et Confirm_Password) et ils ont le type de chaîne de caractère.

La fonction EnregistrerAction() permet de générer ce formulaire et récupérer puis vérifier les données saisies par l'utilisateur.

L'Algorithme de cette fonction :

1. Création d'un objet \$admin qui récupère l'Entité Admin qui contient ses paramètres.
2. Création d'un FormBuilder(notre formulaire) avec le service factory et l'ajout des champs de cette Entité à ce formulaire.
3. Redirection de la variable qui contient le formulaire(dans notre cas \$form) à la page register.html.twig qui permet son affichage.
4. Si les champs sont validés, alors on stocke Username, Email et Password dans la base de donnée de l'Entité User (cette dernière permet de vérifier les informations saisies par l'administrateur lors de la connexion). Ensuite, un message s'affichera qui indique à l'utilisateur que les données sont bien enregistrées et il pourra se connecter sans l'enregistrement.
5. Sinon, on reste à l'étape 4.

```

/**
 * @Route("/enregistrer")
 */
public function EnregistrerAction(Request $request)
{
    $admin = new Admin();
    // On crée le FormBuilder grâce au service form factory

    $formBuilder = $this->get('form.factory')->createBuilder(FormType::class,$admin);

    // On ajoute les champs de l'entité que l'on veut à notre formulaire

    $formBuilder
        ->add('Us', TextType::class)
        ->add('Email', EmailType::class)
        ->add('Password', PasswordType::class)
        ->add('ConPassword', PasswordType::class)
        ->add('Enregistrer', SubmitType::class)
        ;

    // À partir du FormBuilder, on génère le formulaire

    $form = $formBuilder->getForm();
    $form->handleRequest($request);

    // Reste de la méthode qu'on avait déjà écrit
    if ($form->isSubmitted() )

```

Suite de la fonction EnregisterAction() :

```
$u = $form->getData();

if($u->getPassword() == $u->getConPassword()){
    // Création de l'entité
    $user = new User();

    $user->setUsername($u->getUs());
    $user->setPassword($u->getPassword());
    $user->setEmail($u->getEmail());

    $request->getSession()->getFlashBag()->add('notice','Annonce bien enregistrée.');
```

// On récupère l'EntityManager

```
$db = $this
->getDoctrine()
->getManager()
;
// Étape 1 : On « persiste » l'entité
$db->persist($user);
// Étape 2 : On « flush » tout ce qui a été persisté avant
$db->flush();

    echo "Vous êtes bien enregistré, vous pouvez se connecter maintenant";
}
}

return $this->render('@decouverte/Default/register.html.twig',
array('form' => $form->createView(),));
```

La partie qui permet l'affichage du formulaire sur le fichier register.html.twig

```
<div class="container">
    {{ form_start(form, {'attr': {'novalidate': 'novalidate'}})}}

    <fieldset>
        <legend>Enregistrement</legend>
        <div class="form_group">
            <label>
                Username
            </label>
            {{ form_widget(form.Username) }}
            {{ form_errors(form.Username) }}
        </div>
        <div class="form_group">
            {{ form_label(form.Email) }}
            {{ form_widget(form.Email) }}
            {{ form_errors(form.Email) }}
        </div>
        <div class="form_group">
            {{ form_label(form.Password) }}
            {{ form_widget(form.Password) }}
            {{ form_errors(form.Password) }}
        </div>
        <div class="form_group">
            <label>
                Confirm Password
            </label>
            {{ form_widget(form.ConfirmPassword) }}
            {{ form_errors(form.ConfirmPassword) }}
        </div>
    </div>
    <hr>
```

The screenshot shows a web browser window with the title 'Formulaire'. The address bar displays '127.0.0.1:8000/enregistrer'. The page content is a registration form with the following fields and buttons:

- Enregistrement** (Section Header)
- Username** (Text input field)
- Email** (Text input field)
- Password** (Text input field)
- Confirm_Password** (Text input field)
- Enregistrer** (Submit button)
- Se Connecter** (Link/button)

2-Authentification :

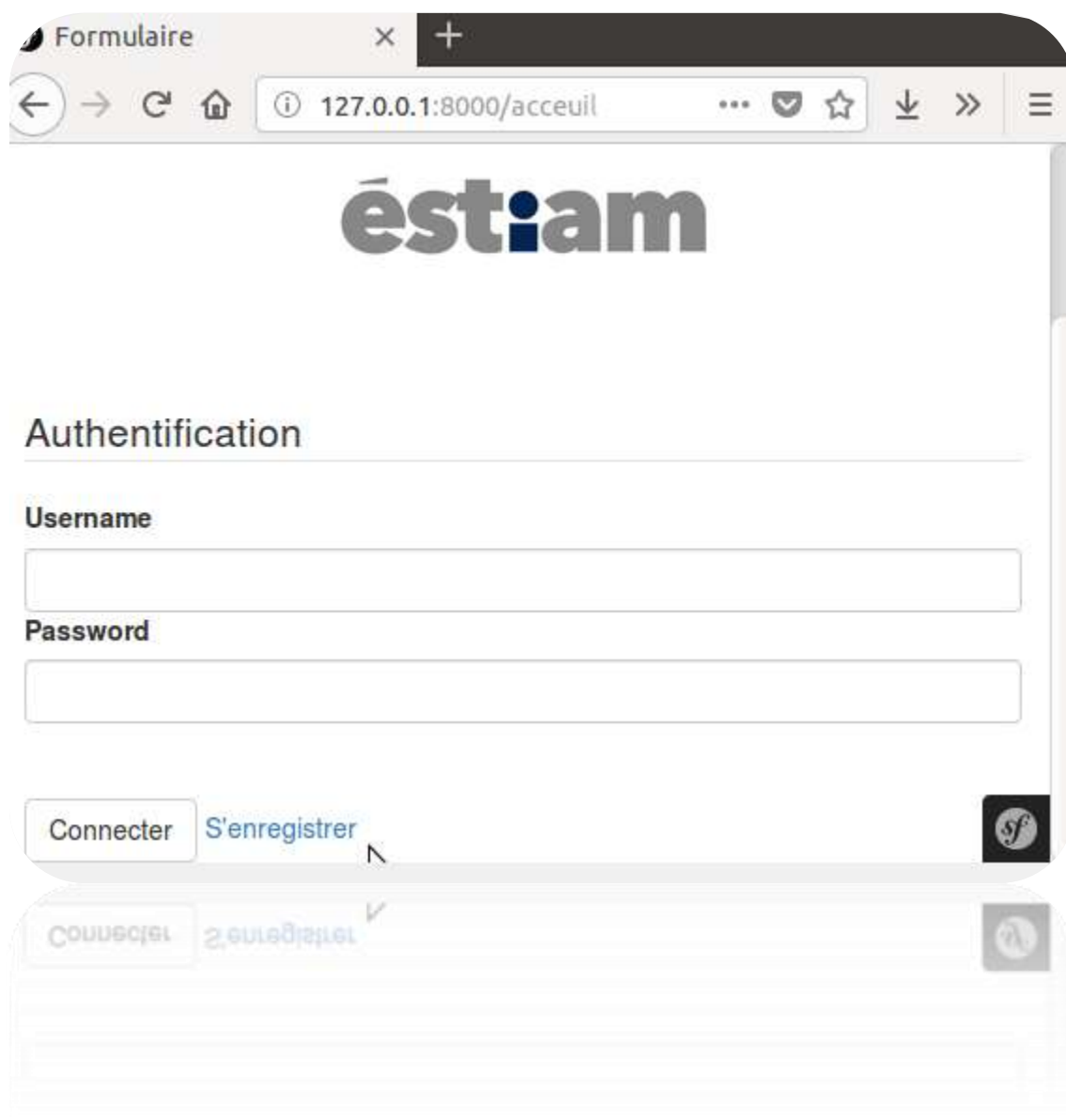
LoginAction() est la fonction qui permet à l'administrateur de se connecter à l'application pour saisir ou exporter les données des visiteurs du Salon.

L'Algorithme de cette fonction :

1. Création d'un objet User().
2. Création du formulaire avec FormBuilder.
3. La redirection du résultat dans la page accueil.html.twig.
4. Si le formulaire est soumis :
 - Récupération des données saisies.
 - Chargement des données de la base pour les comparer avec les données saisies.
 - Si les données saisies sont correctes, on redirige vers la page principale (index.html.twig).
5. Sinon, on reste sur la page de connexion.

Cette fonction est accessible par la route (/accueil) avec name= «accueil ».

La page de connexion :



The image shows a web browser window with a single tab titled 'Formulaire'. The address bar displays '127.0.0.1:8000/acceuil'. The page features the 'estiam' logo at the top. Below the logo, the section is titled 'Authentification'. It contains two input fields: 'Username' and 'Password'. At the bottom of the form, there are two buttons: 'Connecter' and 'S'enregistrer'. A small circular logo with the letters 'sf' is located in the bottom right corner of the page. The browser's navigation bar includes back, forward, refresh, and home icons, as well as status icons for security, bookmarks, and downloads.

Formulaire

127.0.0.1:8000/acceuil

estiam

Authentification

Username

Password

Connecter [S'enregistrer](#)

Connecter [S'enregistrer](#)

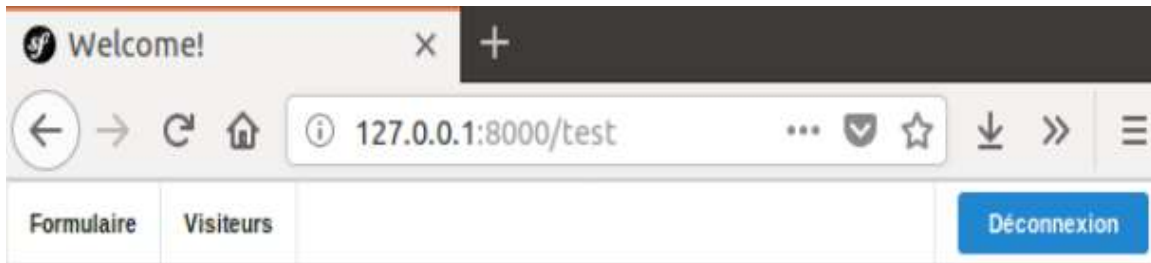
Le fichier accueil.html.twig :

```
<div class="container">
  {{ form_start(form, {'attr': {'novalidate': 'novalidate'}})}}

  <fieldset>
    <legend>Authentification</legend>
    <div class="form_group">
      {{ form_label(form.Username) }}
      {{ form_widget(form.Username) }}
      {{ form_errors(form.Username) }}
    </div>
    <div class="form_group">
      {{ form_label(form.Password) }}
      {{ form_widget(form.Password) }}
      {{ form_errors(form.Password) }}
    </div>
    <br>
    <br>
    <div class="form_group">
      {{ form_label(form.Connector) }}
      {{ form_widget(form.Connector) }}
      <a href="/enregistrer">S'enregistrer</a>
    </div>
  </fieldset>
  {{ form_end(form)}}
</div>
</body>
</html>
```

```
<\/html>
<\/body>
<\/html>
  {{ form_end(form)}}
<\/html>
```


La page principale :



Cette page a pour le but d'ajouter les informations des visiteurs dans le menu **Formulaire**, les visualiser et les exporter dans le menu **Visiteurs**.

Ce fichier est «index.html.twig» et il est accessible par la route /test. Il contient un extends d'un fichier de base «base.html.twig» qui représente le Header(la barre de menu de la page d'accueil). Ensuite, le lien de déconnexion permet de sortir de cette page pour pouvoir se reconnecter. Et l'image se situe dans le dossier « image » du répertoire « web » de ce projet.

1-Le menu **Formulaire** :

Il est généré par la fonction FormAction().

L'algorithme de cette fonction :

1. Création de l'objet Visiteur() qui vient de l'entité Visiteur qui contient aussi les variables suivants : civilité, nom, prénom, mail, code, telephone, code et commentaire.

2. Création du formulaire avec FormBuilder :

```
$formBuilder
    ->add('civilite', TextType::class)
    ->add('prenom', TextType::class)
    ->add('nom', TextType::class)
    ->add('mail', EmailType::class)
    ->add('code', IntegerType::class)
    ->add('telephone', TelType::class)
    ->add('classe', TextType::class)
    ->add('commentaire', TextareaType::class)
    ;
```

3. Pour afficher ce contenu, on redirige vers la page form.html.twig.

```
return $this->render('@decouverte/Default/form.html.twig',
    array('form' => $form->createView(),));
```

4. Si le formulaire est soumis, on récupère le contenu de tous les champs, puis on les affectent à un variable \$visiteur qui représente l'Entité Visiteur dans le but de les stocker dans la base de donnée avec persist(\$visiteur) après qu'on accède à cette base avec getDoctrine() et getManager().

```
// On récupère l'EntityManager
$em = $this->getDoctrine()->getManager();

// Étape 1 : On « persiste » l'entité
$em->persist($visiteur);

// Étape 2 : On « flush » tout ce qui a été persisté avant
$em->flush();

// Puis on redirige vers la page 'ajout'

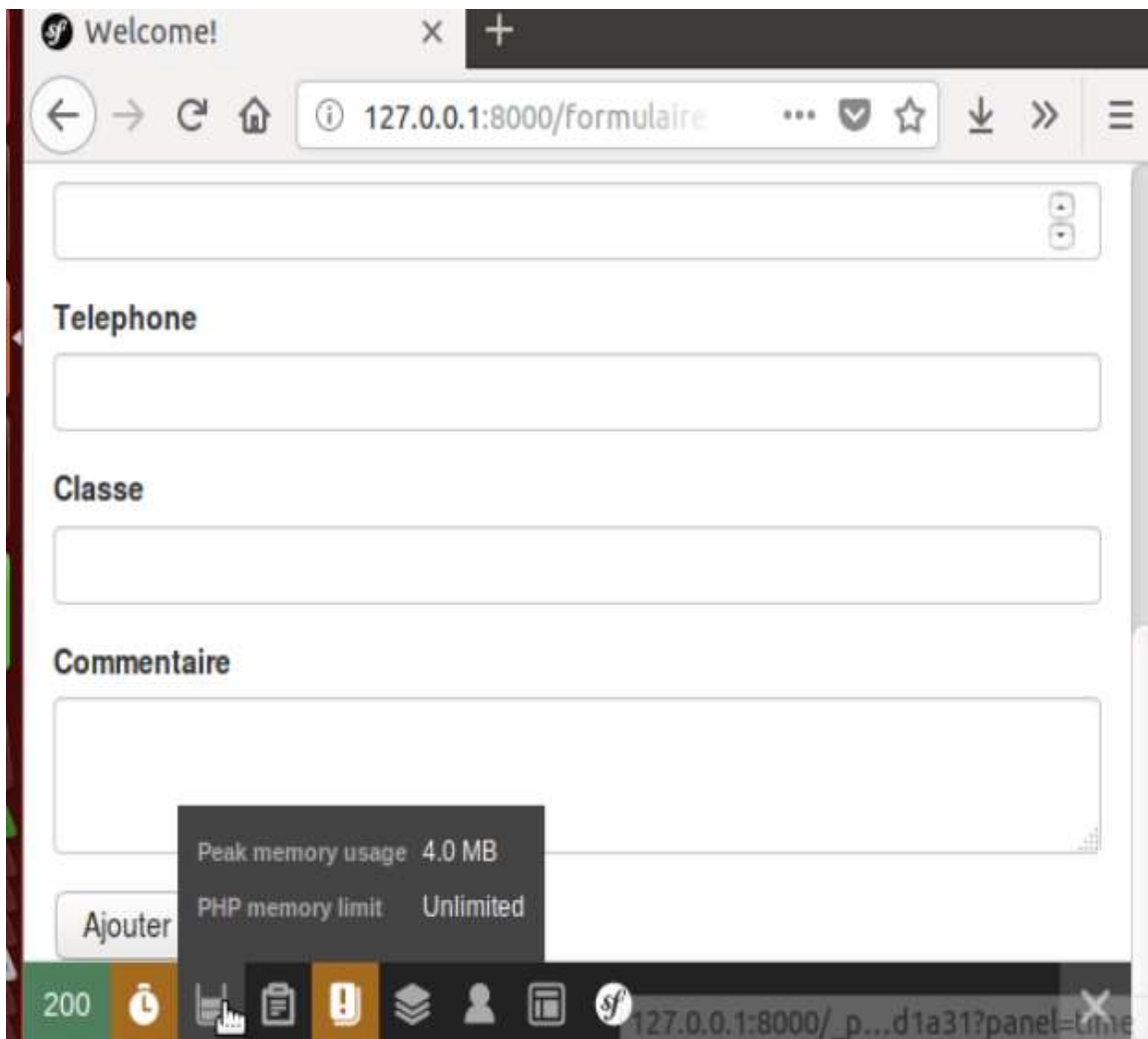
return $this->redirectToRoute('ajout');
```

5. Puis, on redirige vers la page ajout.html.twig qui a comme name= « ajout » pour indiquer que le visiteur a été bien ajouté.

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

{% block body %}
{{ 'Vous avez bien ajouté un nouveau visiteur !!!!' }}
{% endblock %}
```

6. Si ce n'est pas validé, on reste toujours dans la même page.



2-Le menu **Visiteurs** :

C'est là où se trouve l'affichage des données des visiteurs pour les exporter dans un fichiers CSV.

La fonction `AfficherAction()` permet l'affichage de ces données et elle est accessible par la route `/route`:

L'Algorithme utilisée :

On récupère tous les données de la base dans avec `Repository->findAll()` que l'on affecte à la variable `$visiteur`, ce dernier est un tableau que l'on passe à la page `export.html.twig` avec une autre variable `'visiteur'`.

```

/**
 * @Route("/export")
 */

public function AfficherAction(){

    $repository = $this
    ->getDoctrine()
    ->getManager()
    ->getRepository('decouverteBundle:Blog\Visiteur');

    $visiteurs = $repository->findAll();
    return $this->render('@decouverte/Default/export.html.twig',['visiteurs'=>$visiteurs]);
}

```

Le contenu du fichier export.html.twig :

```

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

{% block title %}
Visiteurs
{% endblock %}

{% block body %}

    <form action="csv">

        <table>
            <tr>
                <th>Id</th>
                <th>Civilite</th>
                <th>Prenom</th>
                <th>Nom</th>
                <th>Mail</th>
                <th>Code</th>
                <th>Telephone</th>
                <th>Classe</th>
                <th>Commentaire</th>
            </tr>
            {% for v in visiteurs %}
                <tr>
                    <td>{{ v.Id }}</td>
                    <td>{{ v.Civilite }}</td>
                    <td>{{ v.Prenom }}</td>
                    <td>{{ v.Nom }}</td>
                    <td>{{ v.Mail }}</td>

```

Exemple d'affichage des informations des visiteurs :

Id	Civilite	Prenom	Nom	Mail	Code	Telephone	Classe	Commentaire
1	Mr	ayoub	aghzou	aghzou@hotmail.fr	0	754267815	A	test
2	Mdm	Marwa	Mrabet	marwa.mrabet@estiam.com	2	987854326	B	test
3	Mr	Charlys	test	charlys@estiam.com	2	987854326	B	test
4	Mr	Charlys	test	charlys@estiam.com	2	987854326	B	test
5	Mr	said	aghzou	saghzou96@gmail.com	4	754267815	EMMB	4ème année

Cache Calls: 25

Lorsqu'on clique sur le bouton exporter, les données seront exportées dans un fichier CSV qui sera téléchargé automatiquement. C'est la fonction `GenerateCsvAction()` qui permet de faire cette instruction.

L'Algorithme utilisé :

1. Récupération de toutes les données de l'entité Visiteur.
2. Ouverture d'un Stream Output pour dans le fichier export.csv.
3. Mise en place de l'entête [id, civilite, nom, prenom, mail, code, telephone, classe, commentaire] dans ce fichier avec `fputcsv()`.
4. La mise en place des données récupérées aussi avec `fputcsv()` après l'entête.
5. Ce flux sera redirigé vers l'utilisateur pour pouvoir télécharger et visualiser le fichier.

```

* @Route("/csv")
*/
public function GenerateCsvAction()
{
    $container = $this->container;
    $response = new StreamedResponse(function() use($container) {

        $repository = $this
            ->getDoctrine()
            ->getManager()
            ->getRepository('decouverteBundle:Blog\Visiteur');

        $results = $repository->findAll();
        $handle = fopen('php://output', 'w');

        fputcsv($handle,['id', 'civilite', 'prenom', 'nom', 'mail', 'code', 'telephone', 'classe',
            'commentaire']);
        foreach ($results as $result) {

            $id = $result->getId();
            $civilite = $result->getCivilite();
            $prenom = $result->getPrenom();
            $nom = $result->getNom();
            $mail = $result->getMail();
            $code = $result->getCode();
            $telephone = $result->getTelephone();
            $classe = $result->getClasse();
            $commentaire = $result->getCommentaire();

            $array = [$id, $civilite, $prenom, $nom, $mail, $code, $telephone, $classe, $commentaire];

            fputcsv($handle, $array);
        }
        fclose($handle);
    });
}

```

```

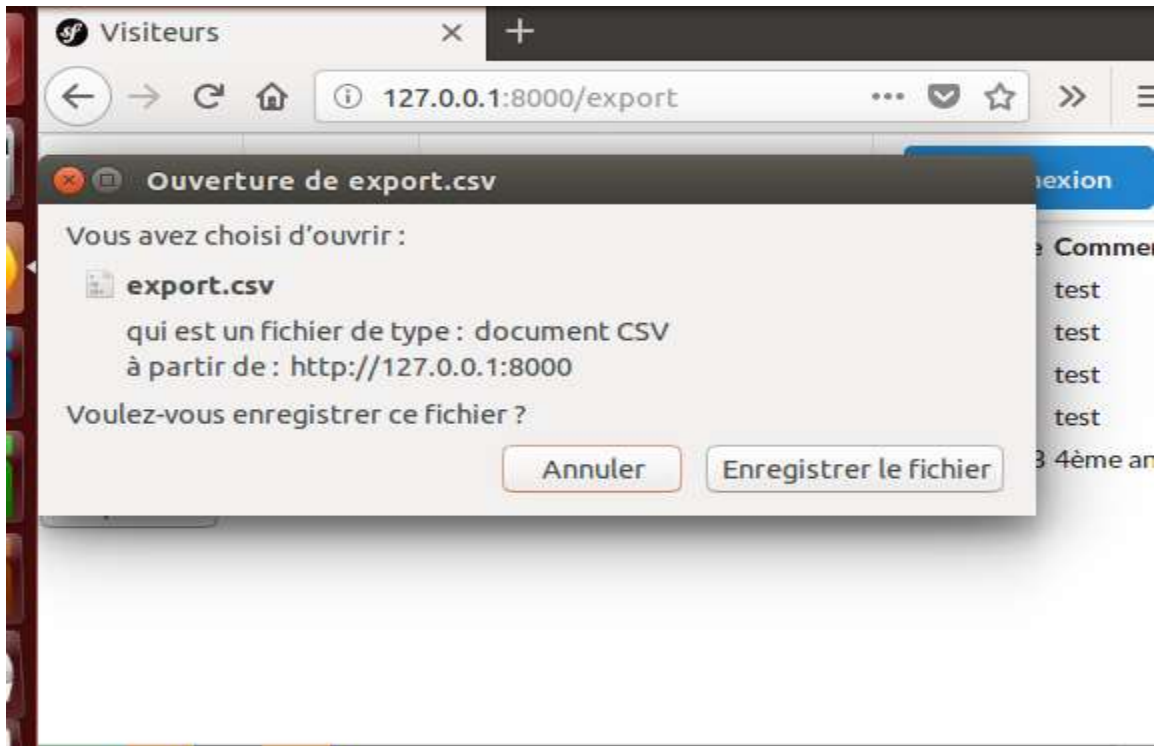
$response->headers->set('Content-Type', 'application/octet-stream');
$response->headers->set('Content-Disposition', 'attachment; filename="export.csv"');
$response->sendContent();
$response->sendHeaders();

return $response;

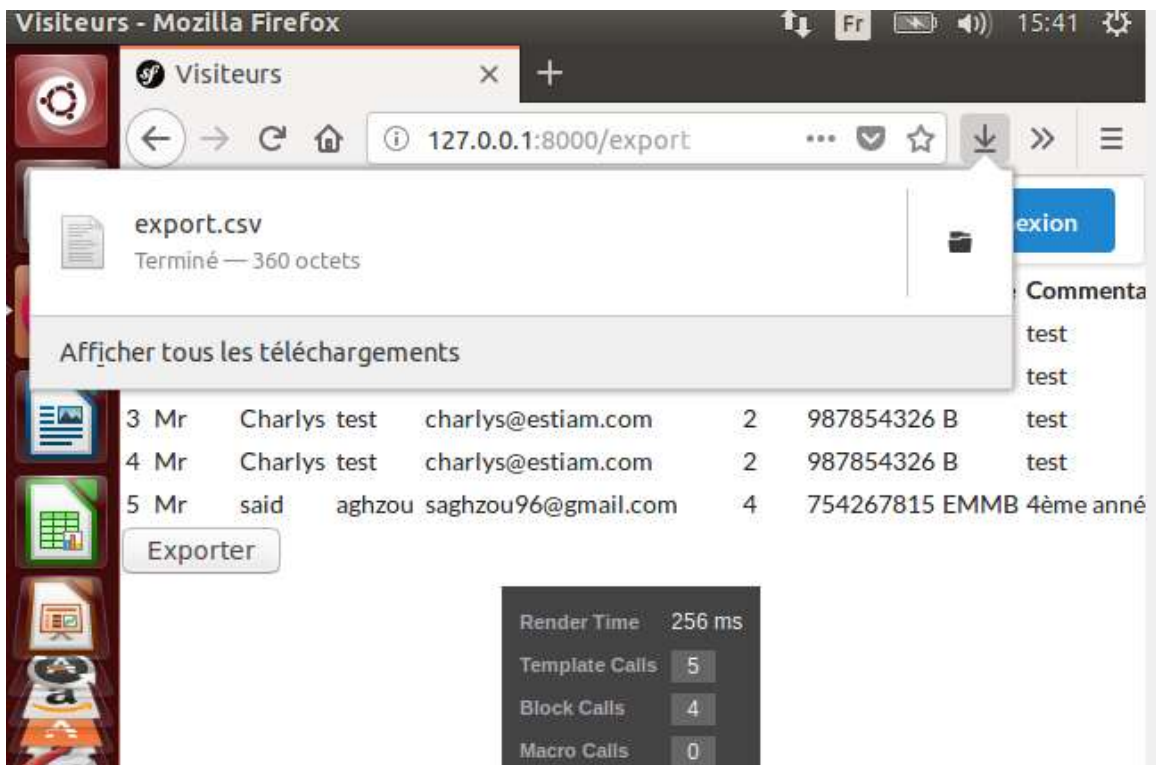
```

Les étapes qui se déclenchent pour avoir le fichier :

1. Enregistrer le fichier



2. Aller sur le menu téléchargement :



3. Cliquer sur le fichier export.csv

	A	B	C	D	E	F	G	H	I
1	id	civilite	prenom	nom	mail	code	telephone	classe	com
2	1	Mr	ayoub	aghzou	aghzou@hotmail.fr	0	754267815	A	test
3	2	Mdm	Marwa	Mrabet	marwa.mrabet@estiam.com	2	987854326	B	test
4	3	Mr	Charlys	test	charlys@estiam.com	2	987854326	B	test
5	4	Mr	Charlys	test	charlys@estiam.com	2	987854326	B	test
6	5	Mr	said	aghzou	saghzou96@gmail.com	4	754267815	EMMB	4ème
7									
8									
9									
10									
11									
12									
13									
14									
15									
16									
17									

La structure de la base de données :

Nous avons utilisé MySQL, nous travaillons sur la base 'symfony' qui est générée automatiquement avec 'php bin/console doctrine:create:database'.

A l'intérieur de cette base, on trouve les entités qui sont représentées par des tableaux.

Pour y accéder, il faut suivre les étapes suivantes sur le terminal :

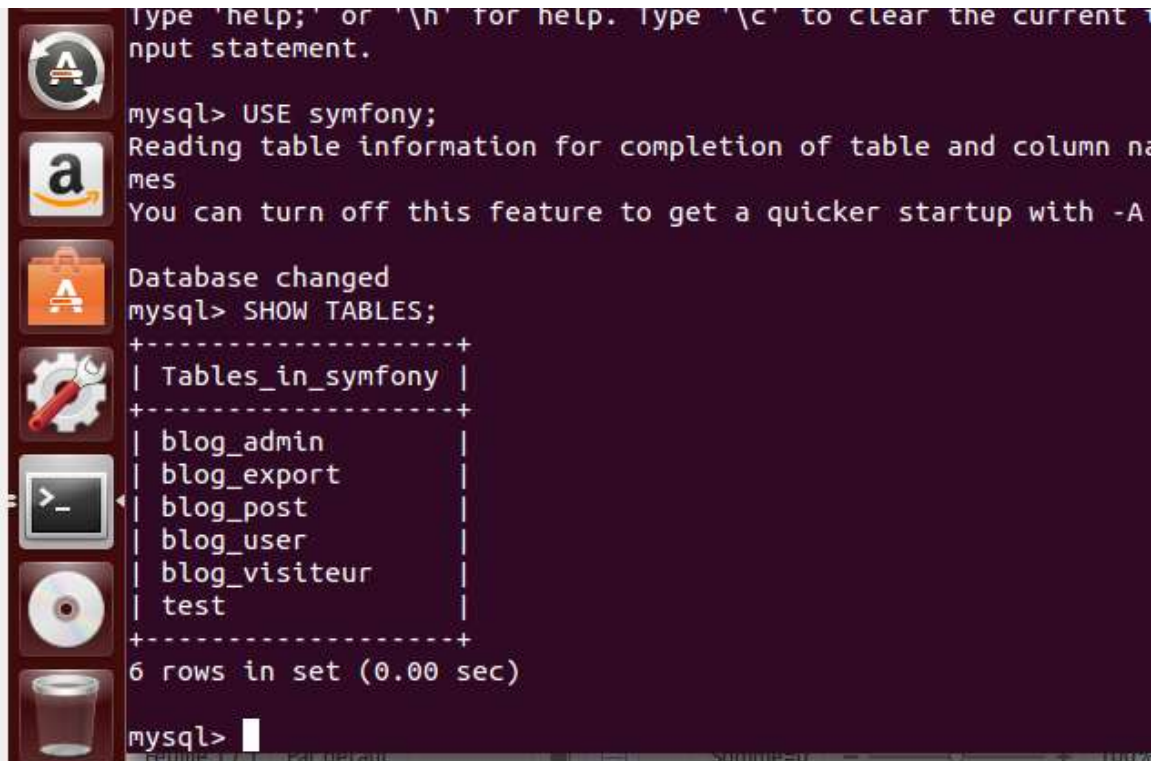
- mysql -u root -p
- USE symfony;
- SHOW TABLES;

Les tableaux sont représentés par blog_ 'nom de l'entité en minuscule' automatiquement avec : 'php bin/console doctrine : generate : entity'.

Pour sélectionner le contenu d'une table on utilise : select * FROM blog_ 'nom de l'entité en minuscule'.

Pour mettre à jour les données de la base, on utilise 'php bin/console doctrine:schema:update --force'.

Notre base et les tableaux :

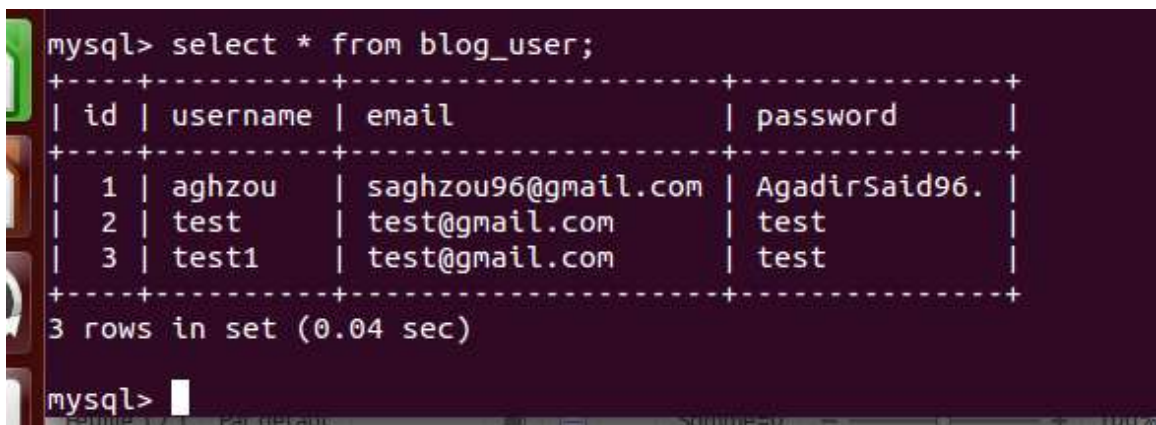


```
mysql> USE symfony;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> SHOW TABLES;
+-----+
| Tables_in_symfony |
+-----+
| blog_admin         |
| blog_export        |
| blog_post          |
| blog_user          |
| blog_visiteur      |
| test               |
+-----+
6 rows in set (0.00 sec)

mysql>
```

Contenu de la table blog_user:



```
mysql> select * from blog_user;
+----+-----+-----+-----+
| id | username | email                | password |
+----+-----+-----+-----+
| 1  | aghzou  | saghzou96@gmail.com | AgadirSaid96. |
| 2  | test    | test@gmail.com       | test     |
| 3  | test1   | test@gmail.com       | test     |
+----+-----+-----+-----+
3 rows in set (0.04 sec)

mysql>
```

Contenu de la table blog_visiteur:

```
mysql> select * from blog_visiteur;
```

id	civilite	commentaire	nom	prenom	mail
		code	telephone	classe	
1	Mr	test	aghzou	ayoub	aghzou@hotmail.fr
2	Mdm	test	Mrabet	Marwa	marwa.mrabet@estiam.com
3	Mr	test	test	Charlys	charlys@estiam.com
4	Mr	test	test	Charlys	charlys@estiam.com
5	Mr	4ème année	aghzou	said	saghzou96@gmail.com

Conclusion :

Ce projet nous a donné l'opportunité de découvrir le Framework Symfony. Et nous avons appris l'utilisation de différentes parties de ce Framework et une autre façon de développer les application Web avec plus d'efficacité et dans un temps réduit.