

Ingénierie Inversée & Intégration de Template

7

OBJECTIFS

- Création de la base de données à partir du script SQL
- Génération des Entités à partir d'une base de données existante
- Génération des Getter et Setter
- Génération des CRUDs : Formulaires, Contrôleurs avec leurs méthodes, Routes, Vues Twig, ...
- Test et validation du code généré
- Intégration du Template AdminLTE : Bootstrap 4 et responsive

SOMMAIRE

TP 7 :	43
I. Etude de cas	44
II. Génération des entités	45
III. Génération des CRUDs.....	48
IV. Intégration de Template	50

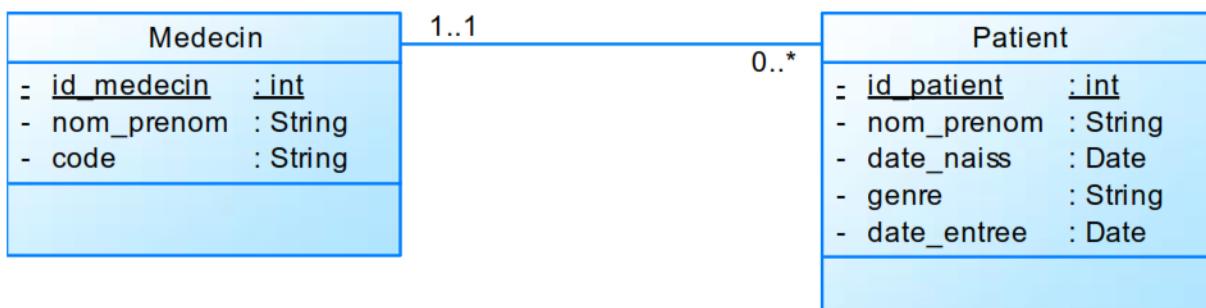
Ingénierie Inversée & Intégration de Template

I. Etude de cas

On se propose dans cette étude de cas de faire la gestion des rendez-vous des patients Covid19 dans un hôpital. Pour cela nous proposons de créer une application web qui gère les calendriers des médecins. Cette application va être développée avec Symfony 6 et les données vont être enregistrées dans une base de données MySQL.

Ci-dessous, le diagramme de classe de notre projet. Les entités qui composent notre diagramme de classe sont : Medecin et Patient.

La relation entre les entités de notre projet est la suivante :



Un Medecin est responsable de plusieurs Patients.

Un Patient est suivi par un et un seul Medecin.

D'où le MPD (Modèle Physique de Données) suivant :



pk : Primary Key

fk : Foreign Key

Le script SQL permettant la création de la BD est le suivant :

```
/*=====
/* Table : Medecin
/*=====
create table Medecin
(
    id_medecin      int not null auto_increment comment '',
    nom_prenom      varchar(254) not null comment '',
    code            varchar(254) comment '',
    primary key (id_medecin)
);

/*=====
/* Table : Patient
/*=====
create table Patient
(
    id_patient      int not null auto_increment comment '',
    medecin_id      int not null comment '',
    nom_prenom      varchar(254) not null comment '',
    date_naiss      date comment '',
    genre           varchar(20) comment '',
    date_entree     date comment '',
    primary key (id_patient)
);

alter table Patient add constraint FK_PATIENT_APP01_MEDECIN
    foreign key (medecin_id) references Medecin (id_medecin)
    on delete restrict
    on update restrict;
```

II. Génération des Entités

Créer une base de données intitulé "covid19" sous le SGBD MySQL en utilisant PhpMyAdmin.

Aussi, en utilisant PhpMyAdmin et le script SQL ci-dessus, créer le schéma de la BD (les tables et les relations entre les différentes tables) "covid19" que vous avez déjà créé.

Créer un nouveau projet Symfony :

- Ouvrez une fenêtre « invite de commande »,
- Puis placez-vous sur C:\xampp\htdocs
- Puis tapez

```
C:\xampp\htdocs> symfony new gestion_hopital --webapp --version="6.4.*"
```

Lancer le serveur intégré Symfony

- a. Ouvrez une fenêtre « invite de commande »,
- b. Puis placez-vous sur C:\xampp\htdocs\gestion_hopital
- c. Puis tapez :

C:\xampp\htdocs\gestion_hopital > **Symfony server:start**

Ouvrir votre projet dans l'IDE de votre choix (PhpStorm / VS Code).

Ouvrir le fichier **.env** et effectuer les opérations suivantes sur le "DATABASE_URL" :

- a. Désactiver le driver "PostgreSQL" en ajoutant le symbole "#" en début de ligne,
- b. Activer le driver "MySQL" en supprimant le symbole "#" en début de ligne
- c. Puis tapez le nom de la BD, le nom utilisateur "**root**" ainsi que son mot de passe "**null**" et l'adresse ip (**127.0.0.1**) et le port (**3306**) de votre SGBD comme ci-dessous :

```
DATABASE_URL="mysql://root:@127.0.0.1:3306/covid19?serverVersion=10.4.32-MariaDB&charset=utf8mb4"
```

- d. Dans le terminal de votre IDE ou dans le terminal CMD tapez la commande suivante :

```
> php bin/console doctrine:mapping:import "App\Entity" annotation --path=src/Entity
```

Surprise :

La commande **doctrine:mapping:import** a été supprimé de l'ORM Doctrine 2/3

Preuve : https://symfony.com/doc/current/doctrine/reverse_engineering.html

Comment Faire Alors ?

- a. Visitez l'adresse suivante : <https://github.com/siburuxue/doctrine-helper>
- b. Ce projet remplace la commande "Deprecated" "doctrine:mapping:import" de Doctrine2/3 et il est compatible avec Symfony 6/7
- c. Lancer le terminal de votre IDE
- d. Puis tapez la commande suivante pour créer les entités **Medecin** et **Patient**.

```
> php bin/console doctrine:mapping:import
```

Après avoir exécuter la commande de génération des entités, constater bien que dans votre projet et précisément dans le dossier "**Entity**", les deux fichiers suivants ont été créé automatiquement : "**Medecin.php**" et "**Patient.php**" contenant tous les attributs ainsi que les Getters et les Setters.

Aussi, constater bien que dans votre projet et précisément dans le dossier "Repository", les deux fichiers suivants ont été crée automatiquement : "**MedecinRepository.php**" et "**PatientRepository.php**".

III. Génération des Relations entre Entités

Constater bien que dans les deux fichiers "**Medecin.php**" et "**Patient.php**" il manque la relation entre les deux entités (OneToMany ou OneToOne ou ManyToOne ou ManyToMany). D'où nous allons les créer en utilisant le Bundle Maker de Symfony.

Correction du fichier "**Patient.php**" :

```
<?php

namespace App\Entity;

use App\Repository\PatientRepository;
use Doctrine\DBAL\Types\Types;
use Doctrine\ORM\Mapping as ORM;

#[ORM\Table(name: 'patient')]
#[ORM\Index(name: 'FK_PATIENT_APP01_MEDECIN', columns: ['medecin_id'])] ❌
#[ORM\Entity(repositoryClass: PatientRepository::class)]
class Patient
{
    #[ORM\Column(name: "id_patient")]
    #[ORM\Id]
    #[ORM\GeneratedValue(strategy: "IDENTITY")]
    private ?int $idPatient = null;

    #[ORM\Column(name: "medecin_id")] ❌
    private ?int $medecinId = null; ❌

    #[ORM\Column(name: "nom_prenom", length: 254)]
    private ?string $nomPrenom = null;

    #[ORM\Column(name: "date_naiss", type: Types::DATE_MUTABLE, nullable: true)]
    private ?\DateTimeInterface $dateNaiss = null;
```

```
#[ORM\Column(name: "genre", length: 20, nullable: true)]
private ?string $genre = 'NULL';

#[ORM\Column(name: "date_entree", type: Types::DATE_MUTABLE, nullable:true)]
private ?\DateTimeInterface $dateEntree = null;

public function getIdPatient(): ?int
{
    return $this->idPatient;
}

public function getMedecinId(): ?int 
{
    return $this->medecinId;
}

public function setMedecinId(int $medecinId): static 
{
    $this->medecinId = $medecinId;

    return $this;
}

...
}
```

- a. Ouvrez le fichier "**Patient.php**",
- b. Puis supprimer les lignes contenant le symbole 
- c. Supprimez aussi les méthodes "**getMedecinId()**" et "**setMedecinId(...)**"

Nous allons maintenant mettre à jour l'entité *Medecin* :

php bin/console make:entity Medecin

Nous allons ajouter la propriété "**patients**" à la classe "**Medecin**" :

- patients : relation/OneToMany

Les deux entités "**Medecin**" et "**Patient**" ont été mises à jour et la relation entre les deux entités a été ajouté avec succès.

```
<?php

namespace App\Entity;

use App\Repository\MedecinRepository;
use Doctrine\Common\Collections\ArrayCollection;
use Doctrine\Common\Collections\Collection;
use Doctrine\DBAL\Types\Types;
use Doctrine\ORM\Mapping as ORM;

#[ORM\Table(name: 'medecin')]
#[ORM\Entity(repositoryClass: MedecinRepository::class)]
class Medecin
{
    #[ORM\Column(name: "id_medecin")]
    #[ORM\Id]
    #[ORM\GeneratedValue(strategy: "IDENTITY")]
    private ?int $idMedecin = null;

    #[ORM\Column(name: "nom_prenom", length: 254)]
    private ?string $nomPrenom = null;

    #[ORM\Column(name: "code", length: 254, nullable: true)]
    private ?string $code = 'NULL';

    /**
     * @var Collection<int, Patient>
     */
    #[ORM\OneToMany(targetEntity: Patient::class, mappedBy: 'medecin')]
    private Collection $patients;

    public function __construct()
    {
        $this->patients = new ArrayCollection();
    }

    public function getIdMedecin(): ?int
    {
        return $this->idMedecin;
    }

    public function getNomPrenom(): ?string
    {
        return $this->nomPrenom;
    }

    public function setNomPrenom(string $nomPrenom): static
    {
        $this->nomPrenom = $nomPrenom;
    }
}
```

```
        return $this;
    }

    public function getCode(): ?string
    {
        return $this->code;
    }

    public function setCode(?string $code): static
    {
        $this->code = $code;

        return $this;
    }

    /**
     * @return Collection<int, Patient>
     */
    public function getPatients(): Collection
    {
        return $this->patients;
    }

    public function addPatient(Patient $patient): static
    {
        if (!$this->patients->contains($patient)) {
            $this->patients->add($patient);
            $patient->setMedecin($this);
        }

        return $this;
    }

    public function removePatient(Patient $patient): static
    {
        if ($this->patients->removeElement($patient)) {
            // set the owning side to null (unless already changed)
            if ($patient->getMedecin() === $this) {
                $patient->setMedecin(null);
            }
        }

        return $this;
    }
}
```

La propriété ***medecin*** sera ajoutée à l'entité ***Patient*** avec ses ***getter*** et ***setter*** :

```
##[ORM\ManyToMany(targetEntity: Medecin::class, inverseBy: 'patients')]
#[ORM\JoinColumn(name:'medecin_id',referencedColumnName:'id_medecin',nullable:false)]
private ?Medecin $medecin = null;

public function getMedecin(): ?Medecin
{
    return $this->medecin;
}

public function setMedecin(?Medecin $medecin): static
{
    $this->medecin = $medecin;

    return $this;
}
```

Voilà tout est effectué avec succès.

Maintenant nous allons regénérer la base de données à partir des entités de notre projet.

- Ouvrir PhpMyAdmin et supprimer la base de données "**Covid19**",
- Puis, supprimer les fichiers de migrations qui se trouve dans le dossier "migrations" de votre projet
- Créer la base de données en tapant dans le terminal :

php bin/console doctrine:database:create

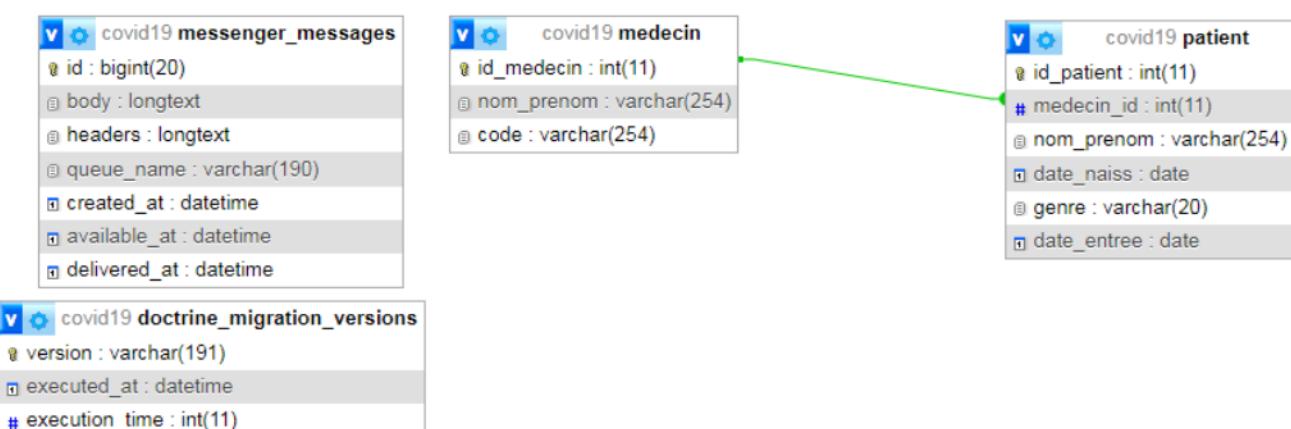
- Créer les tables Medecin et Patient qui correspondent aux entités Medecin et Patient, en tapant les deux commandes suivantes :

php bin/console make:migration

- puis

php bin/console doctrine:migrations:migrate

- Vérifier le schéma de la base de données **Covid19** avec PhpMyAdmin



IV. Génération des CRUDs

Et finalement, on va mettre la cerise sur le gâteau. Avec une simple commande, on va créer les CRUD (Create, Read, Update et Delete) des entités Medecin et Patient. Les vues Twig, les routes, les contrôleurs ainsi que toutes leurs méthodes d'ajout, de suppression, de modification, de listing, de recherche, ... seront créés automatiquement.

Tout d'abord tapez la commande suivante dans votre terminal :

```
C:\xampp\htdocs\gestion_hopital> php bin/console make:crud Medecin
```

Après l'exécution de la commande vous deviez avoir le message suivant :

```
created: src/Controller/MedecinController.php  
created: src/Form/MedecinType.php  
created: templates/medecin/_delete_form.html.twig  
created: templates/medecin/_form.html.twig  
created: templates/medecin/edit.html.twig  
created: templates/medecin/index.html.twig  
created: templates/medecin/new.html.twig  
created: templates/medecin/show.html.twig  
  
Success!
```

Ensuite, répéter la même commande avec comme entité Patient :

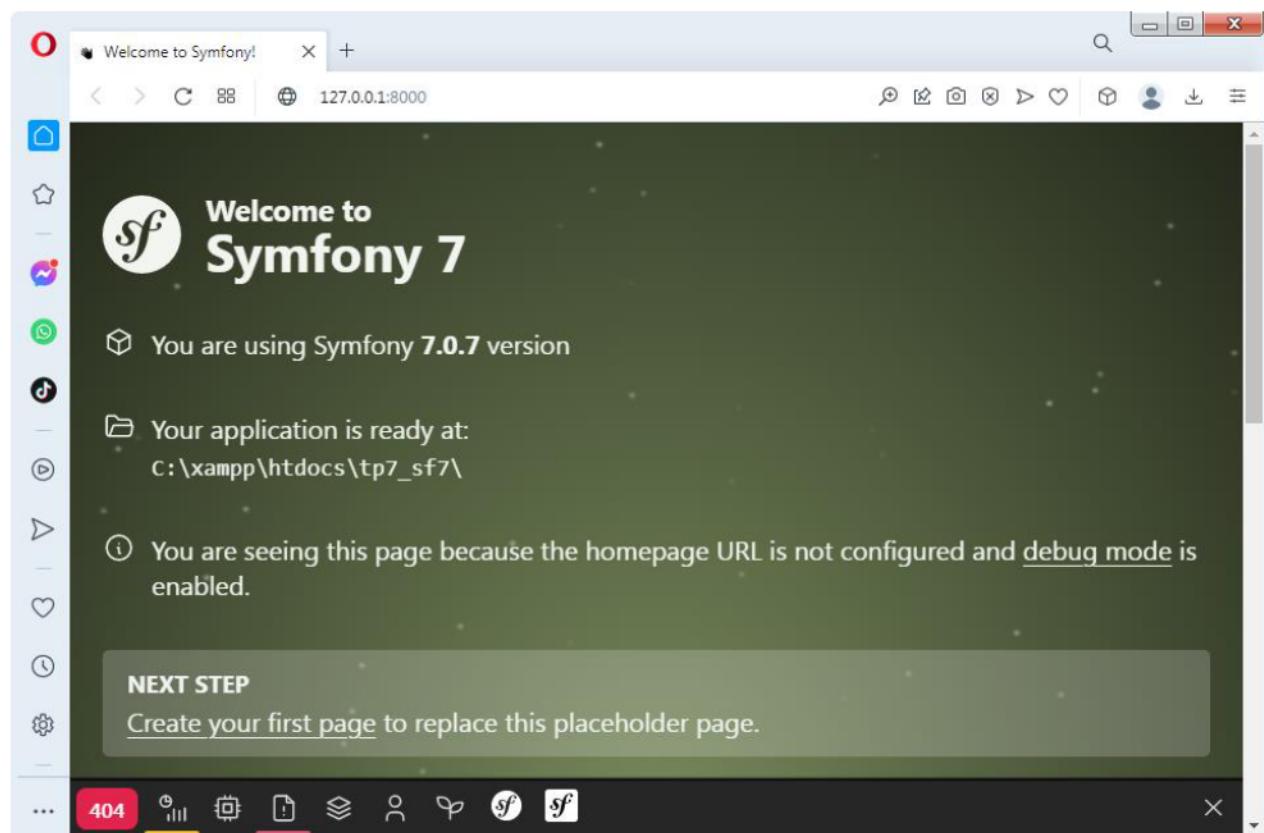
```
C:\xampp\htdocs\gestion_hopital> php bin/console make:crud Patient
```

Après l'exécution de la commande vous deviez avoir le message suivant :

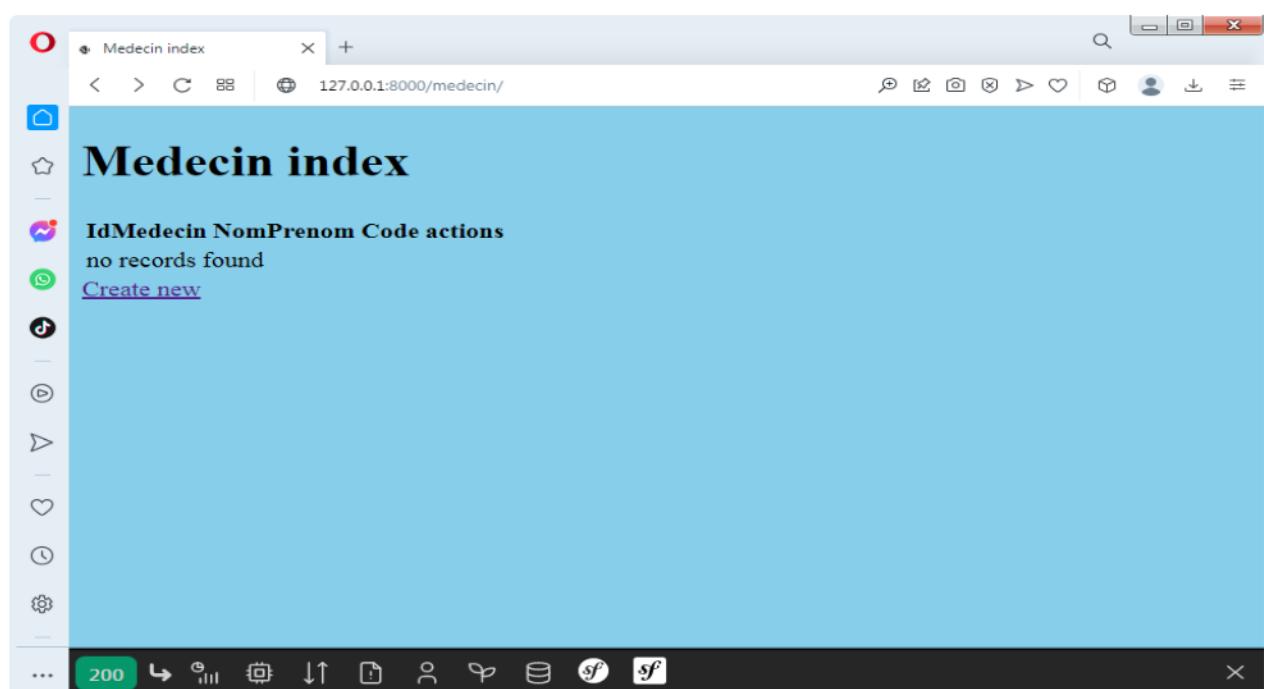
```
created: src/Controller/PatientController.php  
created: src/Form/PatientType.php  
created: templates/patient/_delete_form.html.twig  
created: templates/patient/_form.html.twig  
created: templates/patient/edit.html.twig  
created: templates/patient/index.html.twig  
created: templates/patient/new.html.twig  
created: templates/patient/show.html.twig  
  
Success!
```

Tester votre projet

Ouvrez l'URL <http://127.0.0.1:8000/> dans le navigateur de votre choix :



Visiter l'URL <http://127.0.0.1:8000/medecin/> dans votre navigateur et essayer de créer un nouveau médecin :



Cliquer sur le lien "**Create new**" est saisissez les données du nouveau médecin

New Medecin

Nom prenom: GHAZOUANI Ramzi
Code: 123456

Save

back to list

200

Cliquer sur le bouton "**Save**" pour enregistrer et retourner à la liste des médecins

ID	Medecin	NomPrenom	Code	actions
1	GHAZOUANI Ramzi	123456	show edit	Create new

200

Constater bien que le nouveau médecin a été ajouté avec succès.

Faites le test des liens "**Show**" et "**Edit**".

N'oubliez pas d'effectuer les mêmes tests pour l'entité "**Patient**" !

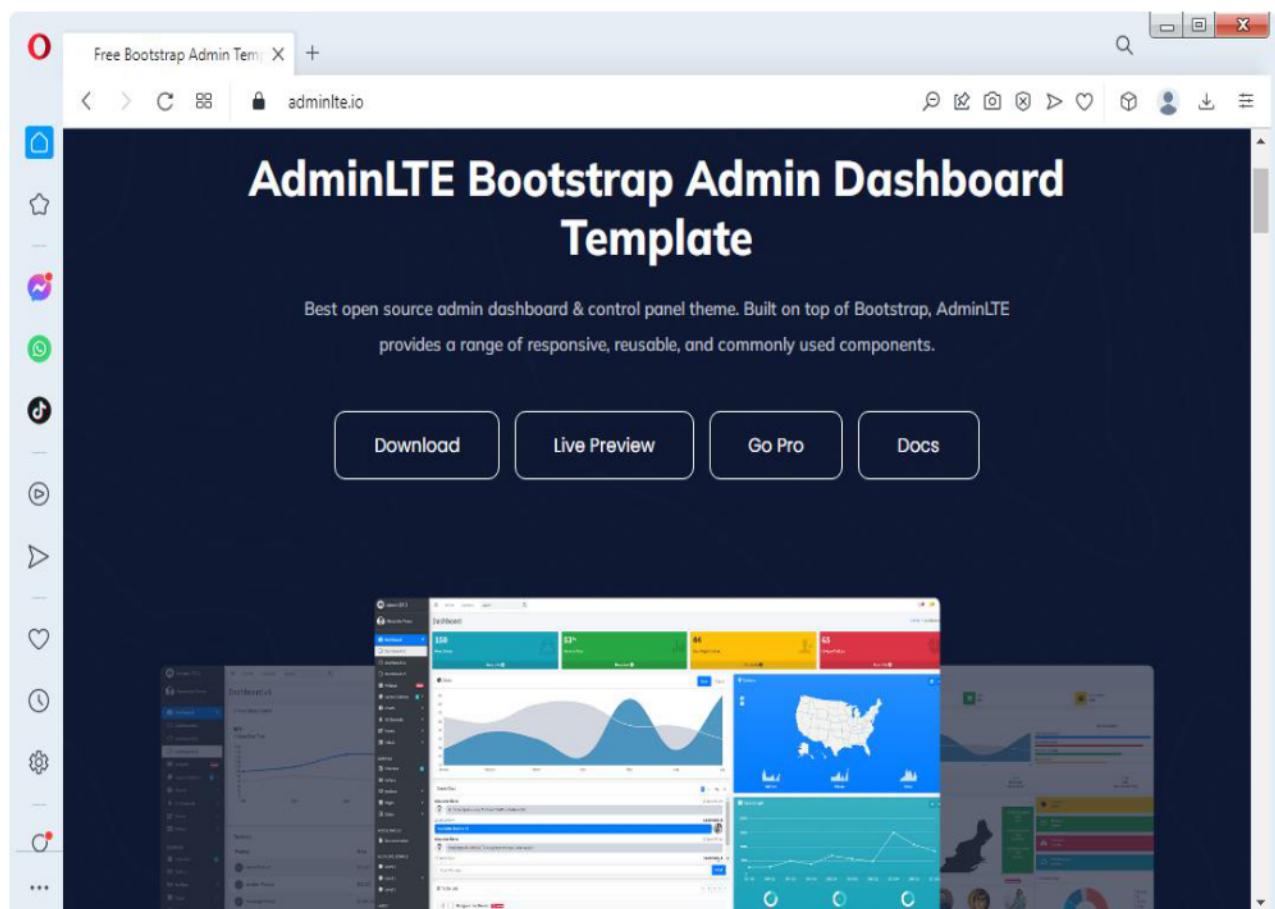
V. Intégration du Template AdminLTE

Dans cette partie on va essayer d'améliorer les vues TWIG de notre projet en intégrant le Template Bootstrap 4 AdminLTE.

Mais c'est très fastidieux de modifier des centaines de pages une par une.

Heureusement, dans Symfony 6 il est possible d'intégrer Bootstrap dans toutes nos vues Twig sans toucher à une vue.

Pour commencer, visiter l'URL <https://adminlte.io/> avec votre navigateur et télécharger le fichier "**Source Code**" qui est le fichier **zip** du Template AdminLTE v3.2.0



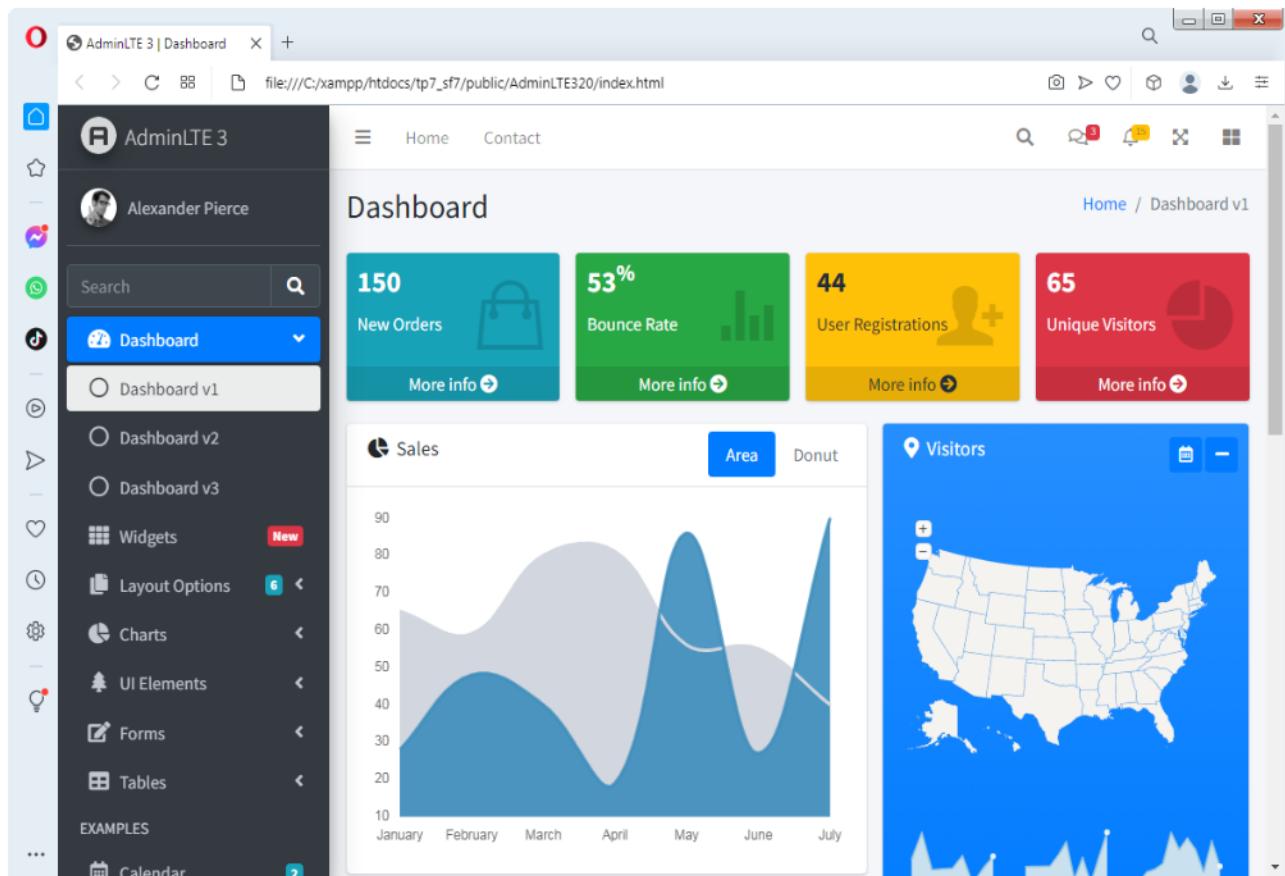
Une fois téléchargé, faites l'extraction du fichier **zip** dans le dossier "public" de votre projet et renommer-le en "**adminlte320**".

Ensuite, ajouter les lignes de code ci-dessous dans le fichier : `config/packages/twig.yaml`

YAML XML PHP

```
1 # config/packages/twig.yaml
2 twig:
3     form_themes: ['bootstrap_4_layout.html.twig']
```

Ouvrir le fichier "`\gestion_hopital\public\adminlte320\index.html`" dans votre navigateur.



C'est ce tableau de bord qu'on va intégrer dans notre projet.

Faites une copie de sauvegarde du fichier "`\gestion_hopital\templates\base.html.twig`" pour que vous puissiez le récupérer au cas où vous avez un problème.

Maintenant copier tout le contenu du fichier "index.html" à la fin du fichier "base.html.twig". Tout ce qui suit, va se passer dans le fichier "base.html.twig".

Etant que nous sommes dans le fichier "base.html.twig".

- Supprimer le contenu de la classe "container-fluid" pour qu'elle soit ainsi

```
<!-- Main content -->
<section class="content">
    <div class="container-fluid">

        </div><!-- /.container-fluid -->
    </section>
    <!-- /.content -->
```

- Couper le "**block body**" du début du fichier et coller-le dans la classe "container-fluid" pour qu'elle soit ainsi :

```
<!-- Main content -->
<section class="content">
    <div class="container-fluid">
        {% block body %}{% endblock %}
    </div><!-- /.container-fluid -->
</section>
<!-- /.content -->
```

- Couper le "**block javascripts**" du début du fichier et coller-le à la fin du fichier juste avant "**</body>**" pour qu'il soit ainsi :

```
<!-- AdminLTE App -->
<script src="dist/js/adminlte.js"></script>
<!-- AdminLTE for demo purposes -->
<script src="dist/js/demo.js"></script>
<!-- AdminLTE dashboard demo (This is only for demo purposes) -->
<script src="dist/js/pages/dashboard.js"></script>
{% block javascripts %}
    {% block importmap %}{{ importmap('app') }}{% endblock %}
{% endblock %}
</body>
</html>
```

- Couper le "**block stylesheets**" du début du fichier et coller-le à la fin de l'entête du fichier juste avant "**</head>**" pour qu'il soit ainsi :

```
<link rel="icon" href="data:image/svg+xml,
    <svg xmlns=%22http://www.w3.org/2000/svg%22 viewBox=%220 0 128 128%22>
        <text y=%221.2em%22 font-size=%2296%22>●</text>
        <text y=%221.3em%22 x=%220.2em%22 font-size=%2276%22 fill=%22%23fff%22>
            Sf
        </text>
    </svg>">
```

```
{% block stylesheets %}  
  {% endblock %}  
</head>  
<body class="hold-transition sidebar-mini layout-fixed">
```

- e. Couper le "**block title**" du début du fichier et coller-le dans l'entête du fichier juste entre les balise "<title></title>" pour qu'il soit ainsi :

```
<meta charset="utf-8">  
<meta name="viewport" content="width=device-width, initial-scale=1">  
<title>{% block title %}Gestion Hopital{% endblock %}</title>
```

- f. Supprimer le code HTML inutile qui se trouve au début du fichier "base.html.twig".
g. Aller vers la fin du fichier, couper tous les scripts javascripts et placez-les dans le "**block javascripts**" pour qu'il soit ainsi :

```
{% block javascripts %}  
  {% block importmap %}{{ importmap('app') }}{% endblock %}  
  <!-- jQuery -->  
  <script src="plugins/jquery/jquery.min.js"></script>  
  <!-- jQuery UI 1.11.4 -->  
  <script src="plugins/jquery-ui/jquery-ui.min.js"></script>  
  <!-- Resolve conflict in jQuery UI tooltip with Bootstrap tooltip -->  
  <script>  
    $.widget.bridge('uibutton', $.ui.button)  
  </script>  
  <!-- Bootstrap 4 -->  
  <script src="plugins/bootstrap/js/bootstrap.bundle.min.js"></script>  
  <!-- ChartJS -->  
  <script src="plugins/chart.js/Chart.min.js"></script>  
  <!-- Sparkline -->  
  <script src="plugins/sparklines/sparkline.js"></script>  
  <!-- JQVMap -->  
  <script src="plugins/jqvmap/jquery.vmap.min.js"></script>  
  <script src="plugins/jqvmap/maps/jquery.vmap.usa.js"></script>  
  <!-- jQuery Knob Chart -->  
  <script src="plugins/jquery-knob/jquery.knob.min.js"></script>  
  <!-- daterangepicker -->  
  <script src="plugins/moment/moment.min.js"></script>  
  <script src="plugins/daterangepicker/daterangepicker.js"></script>  
  <!-- Tempusdominus Bootstrap 4 -->  
  <script  
    src="plugins/tempusdominus-bootstrap-4/js/tempusdominus-bootstrap-4.min.js">  
  </script>  
  <!-- Summernote -->  
  <script src="plugins/summernote/summernote-bs4.min.js"></script>
```

```
<!-- overlayScrollbars -->
<script src="plugins/overlayScrollbars/js/jquery.overlayScrollbars.min.js">
</script>
<!-- AdminLTE App -->
<script src="dist/js/adminlte.js"></script>
<!-- AdminLTE for demo purposes -->
<script src="dist/js/demo.js"></script>
<!-- AdminLTE dashboard demo (This is only for demo purposes) -->
<script src="dist/js/pages/dashboard.js"></script>
{% endblock %}
```

- h. Rappelez-vous que nous avons fait l'extraction du fichier "**zip**" du template **AdminLTE** dans le dossier "**public\AdminLTE320**" de notre projet. D'où, on devrait modifier les chemins des **src="..."** de tous les fichiers **javascripts** du "**block javascripts**" pour qu'ils pointent dans le dossier "**public\AdminLTE320**".

La solution est simple et identique à ce que nous allons faire pour le script de "**jQuery**"

- Tout d'abord on a le code suivant de "**jQuery**"

```
<!-- jQuery -->
<script src="plugins/jquery/jquery.min.js"></script>
```

- Nous allons ajouter le code suivant dans toutes les **src="..."**

```
<!-- jQuery -->
<script
    src="{{ asset('AdminLTE320/') }}plugins/jquery/jquery.min.js">
</script>
```

- En fin, on corrige le code dans toutes les **src="..."** pour qu'il soit ainsi :

```
<!-- jQuery -->
<script
    src="{{ asset('AdminLTE320/plugins/jquery/jquery.min.js') }}">
</script>
```

- N'oubliez pas de refaire tout ce travail à tous les chemins des **src="..."** des fichiers **javascripts** du "**block javascripts**".

- i. Aller vers la fin de l'entête du fichier juste avant "**</head>**".

- Couper toutes les feuilles de styles et collez-les dans le "**block stylesheets**"
- Refaire tout le travail qui a été fait aux chemins des **src="..."** des fichiers **javascripts** du "**block javascripts**" aux chemins des **href="..."** des feuilles de styles du "**block stylesheets**".

- Attention : ne pas modifier les chemins `href="..."` des feuilles de styles dont le `href="..."` commence par "`http://...`" ou "`https://...`".
- j. Testez votre application !
- Visiter l'URL <http://127.0.0.1:8000/medecin/> dans votre navigateur et essayer de créer un nouveau médecin.
- Constatez que Bootstrap ainsi que le template AdminLTE ont bien été intégré dans toutes les pages de votre application web.

Medecin index

IdMedecin	NomPrenom	Code	actions
1	GHAZOUANI Ramzi	123456	show edit

Create new

New Medecin

Dashboard

Create new Medecin

Nom prenom

Ben Flen Flena

Code

54321

Save

back to list

Copyright © 2014-2021 AdminLTE.io. All rights reserved.

Version 3.2.0

IdMedecin	NomPrenom	Code	actions
1	GHAZOUANI Ramzi	123456	show edit
2	Ben Flen Flena	54321	show edit

Visiter l'URL <http://127.0.0.1:8000/patient/> dans votre navigateur et essayer de créer un nouveau patient.

IdPatient	NomPrenom	DateNaiss	Genre	DateEntree	actions
no records found					

The screenshot shows the 'Create new Patient' form in AdminLTE v1. The left sidebar has 'Dashboard v1' selected. The main form fields are:

- Nom prenom:** Ben Yaghlen Flen
- Date naiss:** 01/01/2000
- Genre:** Homme
- Date entree:** 13/05/2024
- Medecin:** A dropdown menu showing '1 | GHAZOUANI Ramzi' (selected), 'Veuillez choisir un médecin', and '2 | Ben Flen Flena'. A 'BACK TO LIST' link is at the bottom.

At the bottom, it says 'Copyright © 2014-2021 AdminLTE.io. All rights reserved.' and 'Version 3.2.0'.

The screenshot shows the 'Patient index' page in AdminLTE v1. The left sidebar has 'Dashboard v1' selected. The main area displays a table of patient data:

IdPatient	NomPrenom	DateNaiss	Genre	DateEntree	actions
1	Ben Yaghlen Flen	2000-01-01	Homme	2024-05-13	show edit

A 'Create new' button is located below the table. At the bottom, it says 'Copyright © 2014-2021 AdminLTE.io. All rights reserved.' and 'Version 3.2.0'.

