Mohammed NABI

**Architecture de présentation :**

Durée 35 minutes.

Table des matières

[**1.1.1** **Introduction :** 3](#_Toc107825707)

[**1.2** **Présentation du ‘’Centre Guinot’’ :** 3](#_Toc107825708)

[**1.2.1** **Présentation de l’établissement ‘’Biogesoft’’ :** 3](#_Toc107825709)

[**1.3** **Projet et objectifs du stage de Guinot :** 3](#_Toc107825710)

[**1.4** **Projet de l’organisme de formation BIOGESOFT :** 4](#_Toc107825711)

[**Choix des technologies et environnement :** 4](#_Toc107825712)

[**2** **technique à adopter :** 5](#_Toc107825713)

[**2.1** **Analyse, cas d’utilisation :** 6](#_Toc107825714)

[**2.2** **Authentification :** 8](#_Toc107825715)

[**3** **Création d'un formulaire de connexion&d'une route de déconnexion :** 8](#_Toc107825716)

[*Conclusion et Remerciements* 10](#_Toc107825717)

[**4** **Extrait du code :** 11](#_Toc107825718)

[**4.1** **Enregistrez l'authentificateur de garde dans security.yaml :** 11](#_Toc107825719)

[**4.2** **Créer une vue de connexion :** 11](#_Toc107825720)

[**4.3** **#Créez le SecurityController.php :** 13](#_Toc107825721)

[**4.4** **#Création d'un authentificateur de formulaire de connexion :** 14](#_Toc107825722)

[**5** **Captures d’écran de mon application :** 17](#_Toc107825723)

[**5.1** **Base de donnée toute les tables :** 18](#_Toc107825724)

[**5.2** **Base de donnée tables messages :** 18](#_Toc107825725)

[**5.3** **Base de donnée tables Utilisateurs :** 19](#_Toc107825726)

[**5.4** **Capture site accueil :** 20](#_Toc107825727)

[**5.5** **Capture site liste formation :** 21](#_Toc107825728)

[**5.6** **Capture site enregistrement nouveau message :** 21](#_Toc107825729)

[**5.7** **Capture se connecter :** 22](#_Toc107825730)

### **Introduction :**

Bonjour,

je m'appelle NABI,

je suis en formation de développeur web et web mobile au centre Guinot.

## **Présentation du ‘’Centre Guinot’’ :**

L’Etablissement de Réadaptation Professionnelle ‘’Paul et Liliane Guinot’’ forme des personnes handicapées visuelles, en vue d’une intégration professionnelle en milieu ordinaire de travail, par le biais d’une formation professionnelle et d’un accompagnement social en milieu spécialisé.

### **Présentation de l’établissement ‘’Biogesoft’’ :**

Biogesoft est une école de Formation Professionnelle en Développement

et en Informatique de Gestion .

Le centre propose de nombreuses formations numériques pour créer, valoriser,

ou développer des projets personnel ou professionnel.

## **Projet et objectifs du stage de Guinot :**

Dans le cadre de ma deuxième et dernière année de formation de développeur web et web mobile,

Et pour des raisons administratives, et vue le contexte actuel,

j’ai réalisé la première partie de mon projet de stage au centre Guinot.

L’objectif a été de développer, une application web de messagerie privé

pour les stagiaires du centre de formation et de rééducation professionnelle .

qui ont fréquenté l’établissement durant ces dix dernières années.

Cette plateforme en ligne, offre un espace de rencontre, de souvenirs, d’apprentissage, d’échanges et de partages d’expériences professionnelles,

entre les anciens stagiaires qui ont fréquenté l’établissement et les nouveaux stagiaires qui le découvrent.

Une messagerie interne, pour que les stagiaires puissent échanger leur savoir

et renforcer leurs acquis en programmations.

L’application ‘STAGIAIRES-GUINOT’, établit en autre, un véritable lien générationnel, une continuité culturelle et éducative, riche en conseils, en expériences,

et principalement dans le domaine lié à l’accessibilité.

## **Projet de l’organisme de formation BIOGESOFT :**

Afin de compléter le projet de fin de stage, et clôturer mon cursus de développeur web et web mobile, La deuxième partie de mon projet

consistait à mettre en place un Système d’Inscription de Candidats dans différentes formations.

L’étape de développement a nécessité d’effectuer quelques manipulations

afin d’intégrer à mon projet en cours,

le module formation proposé par Biogesoft, qui consiste à proposer des cours en ligne, sur des langages de programmation.

Chaque formation débouche sur une liste de langages de programmation tels que le PHP, JAVA, PYTHON…

Le futur stagiaire doit choisir le langage de sa formation.

Chaque langage propose une liste de cours en ligne.

Une liste de cours en ligne est proposée avec des cases à cocher.

Une fois le stagiaire identifié et le choix validé, il peut accéder à la formation qu’il a choisie et entamé son premier cours en ligne.

Les cours sont sous différentes forma :

Cours en lignes,

Cours en fichiers PDF,

Cours en vidéo,

Cours en fichier Word…

# **Choix des technologies et environnement :**

L’application a été développée sous Symfony 4.4,

Symfony est un Framework libre qui utilise le langage PHP

avec une architecture MVC, une maintenance facile pour le site Web,

- La rapidité des temps de réponse,

- L’optimisation des URL permet d’avoir des URL distinctes pour chaque page

- Un système de configuration très facile avec langage de description YAML

- La documentation de Symfony est accessible

Bootstrap 5

- D’avoir des pages responsives.

1.3) HTML5 / TWIG

1.4) JavaScript28

1.5) PostgreSQL

« PostgreSQL est un système de gestion de base de données relationnelle orienté objet puissant qui est en open source.

- Php7.4

- Composer 2

- PostgreSQL

- Localhost

- Serveur Apache v2.4

- Visual Code Studio

# **technique à adopter :**

On aura deux entités principales.

• L’entité Utilisateurs, qui gère les utilisateurs connectés ;

• L’entité Messages, qui gère les messages des utilisateurs.

Dans l’entité Utilisateurs, on a deux profils actifs.

• Le profil Expéditeur, celui qui envoie le message ;

• Le profil Destinataire, celui qui reçoit le message ou la réponse.

• Un expéditeur peut envoyer un ou plusieurs messages->Relation(OneToMany) ;

• Un destinataire peut recevoir un ou plusieurs messages->Relation(OneToMany) ;

Chaque stagiaire possède une liste d'abonnement d’utilisateurs.

Chaque abonnement est lié à une file de messages émis par un autre utilisateur.

Côté Administrateur :

La gestion des listes d'abonnements s’effectue du côté Administrateur.

Côté client : Les messages sont visibles par tous les stagiaires.

Une fois l’utilisateur abonné, le filtrage s’effectue par rapport au choix de l'utilisateur qui reçoit et donc seuls les messages choisis, lui seront transmis.

En résumé, l’application "STAGIAIRES-GUINOT.COM’’ dispose d’un service de tchat public, accessible aux utilisateurs de Guinot.

La personne connectée, peut, si elle le désire, discuter, envoyer des documents, des photos, des vidéos…

Elle dispose également d’un service de tchat privé entre deux personnes maximum,

## **Analyse, cas d’utilisation :**

Tableau des opérations d’un cas d’utilisation.

Ce tableau détaille les cas Graphiques sans nom d’opération. Il sert de base à la construction de support d’interface graphique.

Acteur : Visiteur ;

Evènement Déclencheur : Néant ;

Intérêt : Accéder au Serveur pour s’authentifier et accéder à l’interface réservée au compte en question ;

Précondition : Serveur Actif ;

Post-Condition : Connexion Etablie ;

Scenario Nominal :

1. L’utilisateur saisit l’adresse IP du Serveur ;

2. L’utilisateur demande une connexion au Serveur ;

3. Le Serveur répond ;

Scenario Alternatif :

o L’adresse IP n’est pas renseignée, retour au cas 1 ;

o Le Serveur retourne un cas d’erreur, retour au cas 1 ;

o Le Serveur ne répond pas, retour au cas 1 ;

- Pour accéder à « Espace administrateur » de la « Messagerie STAGIAIRES-GUINOT», l'administrateur doit s'authentifier.

- Le formulaire d'authentification est disponible via l'adresse URL : «-STAGIAIRES-GUINOT.COM/admin ». Pour information, tous les utilisateurs ne possédant pas le statut administrateur seront automatiquement redirigés vers la page d’accueil.

Éléments composant le formulaire :

- Champ « Adresse email » (type="email" / champ obligatoire) ;

- Champ « Mot de passe » (type="password" / obligatoire) ;

- Bouton « Se connecter » (pour valider le formulaire).

Cas d'utilisation :

* Nom : S'authentifier (module « Authentification Administrateur ») ;
* Acteur(s) : Administrateur ;
* Description : L'utilisateur doit pouvoir s'authentifier et accéder à la messagerie «STAGIAIRES-GUINOT.COM» ;
* Préconditions : Avoir un compte utilisateur créé et posséder le rôle Admin ;
* Démarrage : L'utilisateur a demandé la page de connexion.

Scénario nominal :

1. Le système affiche le formulaire de connexion ;

2. L'utilisateur saisit son adresse e-mail et son mot de passe ;

3. Le système vérifie les données saisies ;

4. Le système affiche le tableau de bord du back-office Admin.

Scénarios alternatifs :

3a Le système indique que le compte n'existe pas ;

3b Le système redirige vers la page d’accueil car l’utilisateur ne possède pas un compte ayant le statut Admin.

Post conditions :

Scénario nominal : reconnaissance de l'utilisateur / accès à la plateforme = affichage des menus de gestion et du back-office Admin25.

## **Authentification :**

Création de l'entité User :

Php bin/console make:user User

enregistrer les utilisateurs en base de données avec Doctrine

On choisit `email`, qui servira de login.

hacher les mots de passe (chiffrés),

**Formulaire de création d'utilisateurs :**

Php bin/console make:registration-form

utilisateurs soient uniques sur le champ de login,

utilisateurs soient logués automatiquement après la création du compte

Maintenant que l'entité est créée, on peut faire une migration pour créer la table

en base de données :

bin/console make:migration

bin/console doctrine:migrations:migrate

Nous venons de créer notre premier compte utilisateur et nous l'avons rediriger vers la page d'accueil.

**Gérer le login d'utilisateurs :**

Maintenant que nos utilisateurs peuvent créer un compte, il va falloir les loguer :

php bin/console make:auth

gérer l'authentification par un formulaire

créer un service qui va gérer les méthodes d'authentification`LoginFormAuthenticator`,

créer un controller qui va gérer les routes d'authentification (/login

`SecurityController`

une route /logout est créée automatiquement

Dans `LoginFormAuthenticator.php`,

indiqué à l’utilisateur qui vient d'être logué la page vers laquelle il sera rediriger

# **Création d'un formulaire de connexion&d'une route de déconnexion :**

nous avons vu comment créer un formulaire d'inscription pour ajouter de nouveaux utilisateurs dans notre application.

les utilisateurs doivent se connecter à l'application s'ils ont déjà un compte sur notre application,

il aura donc une session et des informations d'identification pendant qu'il visite notre site Web.

Symfony 4 rend cela très facile à réaliser

**création des routes de connexion et de déconnexion :**

Dans un premier temps, nous devrons créer une route où l'utilisateur

accédera au formulaire de connexion.

L'action de connexion recevra comme premier argument via l'injection de dépendance

* AuthenticationUtilsinstance, un objet à partir duquel nous pouvons obtenir des informations

Tel que :

* la dernière erreur d'authentification,
* Le nom d'utilisateur fourni sur le formulaire.

**Création d'un authentificateur de formulaire de connexion :**

Ensuite, on crée la classe d'authentification

qui étend la classe de base AbstractFormLoginAuthenticator,

ce qui facilite l'authentification de connexion par formulaire.

Cette classe recevra dans le constructeur 4 composants clés requis dans ce module,

- Le gestionnaire d'entités (pour créer des requêtes),

- L'interface du routeur (pour créer des routes),

- Le gestionnaire de jetons CSRF (vérifier si le formulaire était valide)

- L'encodeur de mot de passe (pour vérifier si l'authentification est valide).

lorsque la méthode supports renvoie true,

L'authentificateur est exécuté (après avoir été enregistré

**Condition d'exécution :**

- La route actuelle est l'app\_login,

- La méthode est POST,

- Lorsque le formulaire a été soumis.

- rediriger l'utilisateur vers la page d’accueil de notre application,

en fonction de nos besoins,

au lieu de lever l'exception,

3. Enregistrez l'authentificateur de garde

L'authentificateur existe déjà, mais on doit l'enregistrer dans le pare-feu principal :

# app/config/packages/security.yaml

4. Créer une vue de connexion :

login.html.twig,

{# /app/templates/security/login.html.twig #}

# *Conclusion et Remerciements*

*Je tiens à remercier vivement, toutes les personnes Qui m’ont accueilli à Guinot,*

*Au temps qu’on a passé ensemble,*

*au partage de leur expertise au quotidien,*

*à leur confiance, leur collaboration et assistance,*

*qui m'ont permis d’acquérir des connaissances fondamentales*

*dans le domaine du développement web et web mobile,*

*à leurs conseils et recommandations, à l’élaboration et au bon déroulé de mon rapport de stage.*

*Remerciements à :*

*Aux membres du jury ici présents,*

*A Monsieur* ***Akli SAIDANI,*** *mon tuteur de stage et directeur des études BIOGESOFT,*

*A Mon maître de stage, Monsieur* ***Igor PALAKOT****,*

*A Monsieur* ***Gilles LEVY****, mon professeur d’informatique,*

*A l’ensemble du personnel du centre de formation* ***Guinot****,*

*Et en fin à ma famille et mes proches.*

*Ce stage m’a permis d’affiner certaines pistes pour bâtir mon Projet d’Orientation Professionnel et signe l’aboutissement de mon cursus de développeur web et*

*web mobile.*

# **Extrait du code :**

## **Enregistrez l'authentificateur de garde dans security.yaml :**

# app/config/packages/security.yaml

Sécurité:

pare-feu :

principale:

anonyme : vrai

gardien:

authentificateurs :

- App\Sécurité\LoginFormAuthenticator

# Un moyen simple de contrôler l'accès à de grandes sections de votre site

# Remarque : Seul le \*premier\* contrôle d'accès qui correspond sera utilisé

contrôle d'accès:

- { chemin : ^/login$, rôles : IS\_AUTHENTICATED\_ANONYMOUSLY }

Ce faisant, la classe sera enregistrée et réagira lorsque la méthode supports de la classe retournera true.

## **Créer une vue de connexion :**

login.html.twig,

{# /app/templates/security/login.html.twig #}

{% étend 'base.html.twig' %}

{% block title %}Connectez-vous !{% endblock %}

{% bloc corps %}

<form method="post">

{# S'il y a une erreur, affichez-la à l'utilisateur #}

{% si erreur %}

<div class="alert alert-danger">{{ error.messageKey|trans(error.messageData, 'security') }}</div>

{% fin si %}

<h1>Veuillez vous connecter</h1>

{# Entrée d'e-mail #}

<label for="inputEmail" class="sr-only">E-mail</label>

<input type="email" value="{{ last\_username }}" name="email" id="inputEmail" class="form-control" placeholder="Email" requis autofocus>

{# Saisie du mot de passe #}

<label for="inputPassword" class="sr-only">Mot de passe</label>

<input type="password" name="password" id="inputPassword" class="form-control" placeholder="Mot de passe" requis>

{# Entrée de jeton CSRF #}

<input type="hidden" name="\_csrf\_token" value="{{ csrf\_token('authenticate') }}" />

<bouton class="btn btn-lg btn-primary" type="soumettre">

S'identifier

</bouton>

</form>

{% endblock %}

## **#Créez le SecurityController.php :**

itinéraire de connexion : src/Controller/SecurityController.php

<?php

namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;

use Symfony\Component\HttpFoundation\Response;

use Symfony\Component\Routing\Annotation\Route;

use Symfony\Component\Security\Http\Authentication\AuthenticationUtils;

class SecurityController extends AbstractController

{

/\*\*

\* @Route("/login", name="app\_login")

\*/

public function login(AuthenticationUtils $authenticationUtils): Response

{

// Get the login error if there is one

// Obtenir l'erreur de connexion s'il y en a une

$error = $authenticationUtils->getLastAuthenticationError();

// Retrive the last email entered by the user

// Récupérer le dernier email saisi par l'utilisateur

$lastUsername = $authenticationUtils->getLastUsername();

return $this->render('security/login.html.twig', [

'last\_username' => $lastUsername,

'error' => $error

]);

}

}

## **#Création d'un authentificateur de formulaire de connexion :**

src/Security/LoginFormAuthenticator.php

<?php

namespace App\Security;

use App\Entity\User;

use Doctrine\ORM\EntityManagerInterface;

use Symfony\Component\HttpFoundation\RedirectResponse;

use Symfony\Component\HttpFoundation\Request;

use Symfony\Component\Routing\RouterInterface;

use Symfony\Component\Security\Core\Authentication\Token\TokenInterface;

use Symfony\Component\Security\Core\Encoder\UserPasswordEncoderInterface;

use Symfony\Component\Security\Core\Exception\CustomUserMessageAuthenticationException;

use Symfony\Component\Security\Core\Exception\InvalidCsrfTokenException;

use Symfony\Component\Security\Core\Security;

use Symfony\Component\Security\Core\User\UserInterface;

use Symfony\Component\Security\Core\User\UserProviderInterface;

use Symfony\Component\Security\Csrf\CsrfToken;

use Symfony\Component\Security\Csrf\CsrfTokenManagerInterface;

use Symfony\Component\Security\Guard\Authenticator\AbstractFormLoginAuthenticator;

use Symfony\Component\Security\Http\Util\TargetPathTrait;

class LoginFormAuthenticator extends AbstractFormLoginAuthenticator

{

use TargetPathTrait;

private $entityManager;

private $router;

private $csrfTokenManager;

private $passwordEncoder;

public function \_\_construct(EntityManagerInterface $entityManager, RouterInterface $router, CsrfTokenManagerInterface $csrfTokenManager, UserPasswordEncoderInterface $passwordEncoder)

{

$this->entityManager = $entityManager;

$this->router = $router;

$this->csrfTokenManager = $csrfTokenManager;

$this->passwordEncoder = $passwordEncoder;

}

public function supports(Request $request)

{

return 'app\_login' === $request->attributes->get('\_route')

&& $request->isMethod('POST');

}

public function getCredentials(Request $request)

{

$credentials = [

'email' => $request->request->get('email'),

'password' => $request->request->get('password'),

'csrf\_token' => $request->request->get('\_csrf\_token'),

];

$request->getSession()->set(

Security::LAST\_USERNAME,

$credentials['email']

);

return $credentials;

}

public function getUser($credentials, UserProviderInterface $userProvider)

{

$token = new CsrfToken('authenticate', $credentials['csrf\_token']);

if (!$this->csrfTokenManager->isTokenValid($token)) {

throw new InvalidCsrfTokenException();

}

$user = $this->entityManager->getRepository(User::class)->findOneBy(['email' => $credentials['email']]);

if (!$user) {

// fail authentication with a custom error

throw new CustomUserMessageAuthenticationException('Email could not be found.');

}

return $user;

}

public function checkCredentials($credentials, UserInterface $user)

{

return $this->passwordEncoder->isPasswordValid($user, $credentials['password']);

}

public function onAuthenticationSuccess(Request $request, TokenInterface $token, $providerKey)

{

if ($targetPath = $this->getTargetPath($request->getSession(), $providerKey)) {

return new RedirectResponse($targetPath);

}

// For example : return new RedirectResponse($this->router->generate('some\_route'));

throw new \Exception('TODO: provide a valid redirect inside '.\_\_FILE\_\_);

}

protected function getLoginUrl()

{

return $this->router->generate('app\_login');

}

}

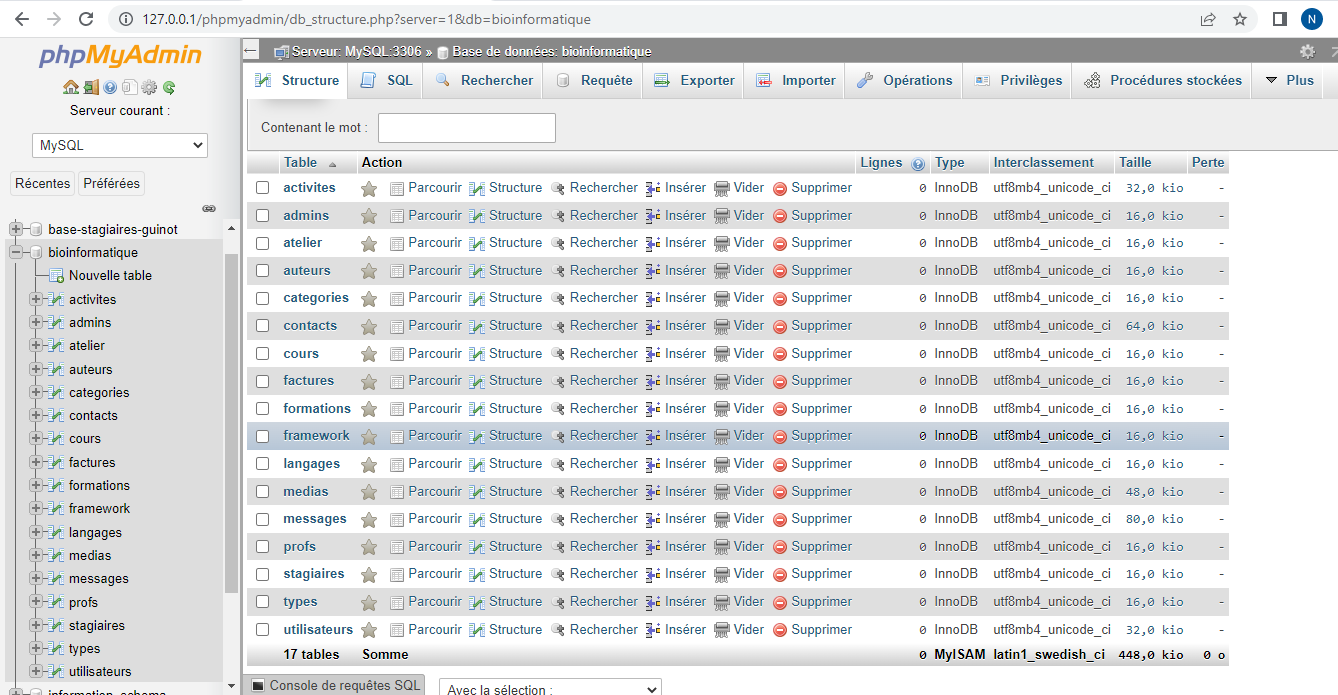
*Ce stage m’a permis d’affiner certaines pistes pour bâtir mon Projet d’Orientation Professionnel et signe l’aboutissement de mon cursus de développeur web et*

*web mobile.*

# **Captures d’écran de mon application :**

## **Base de donnée toute les tables :**

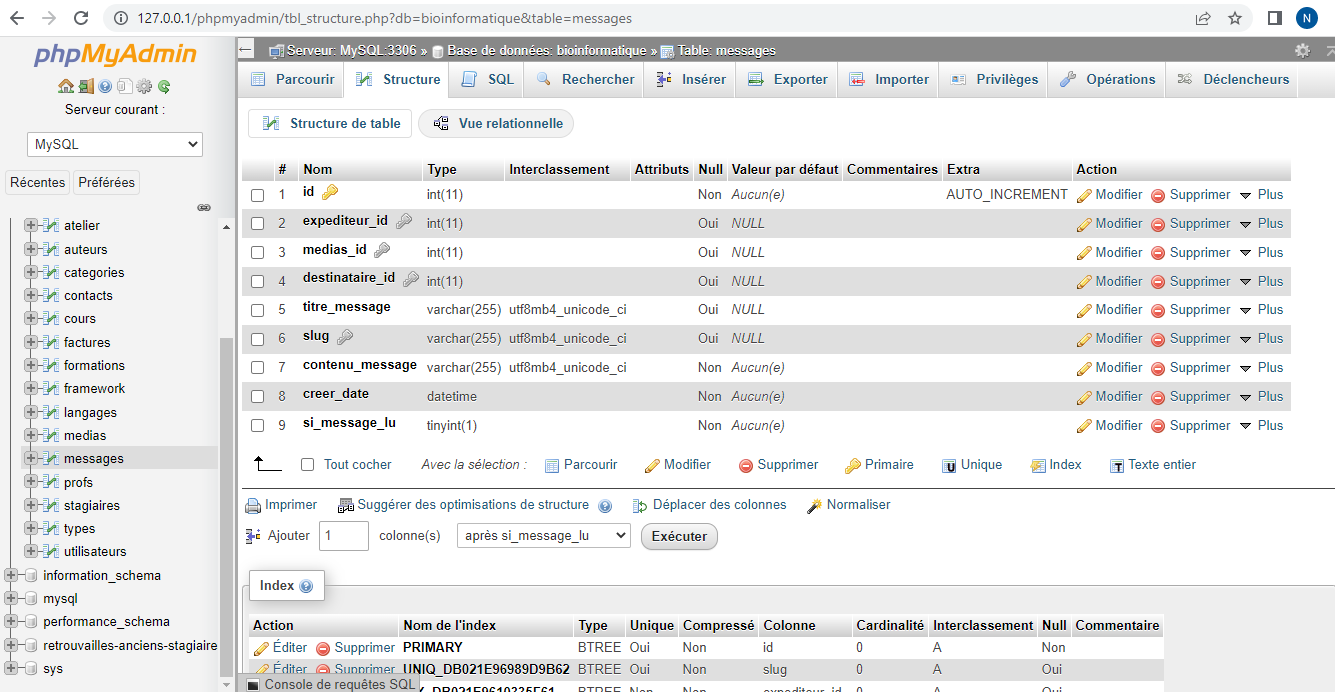
Base de donnée toute les tables



Texte à remplir

## **Base de donnée tables messages :**

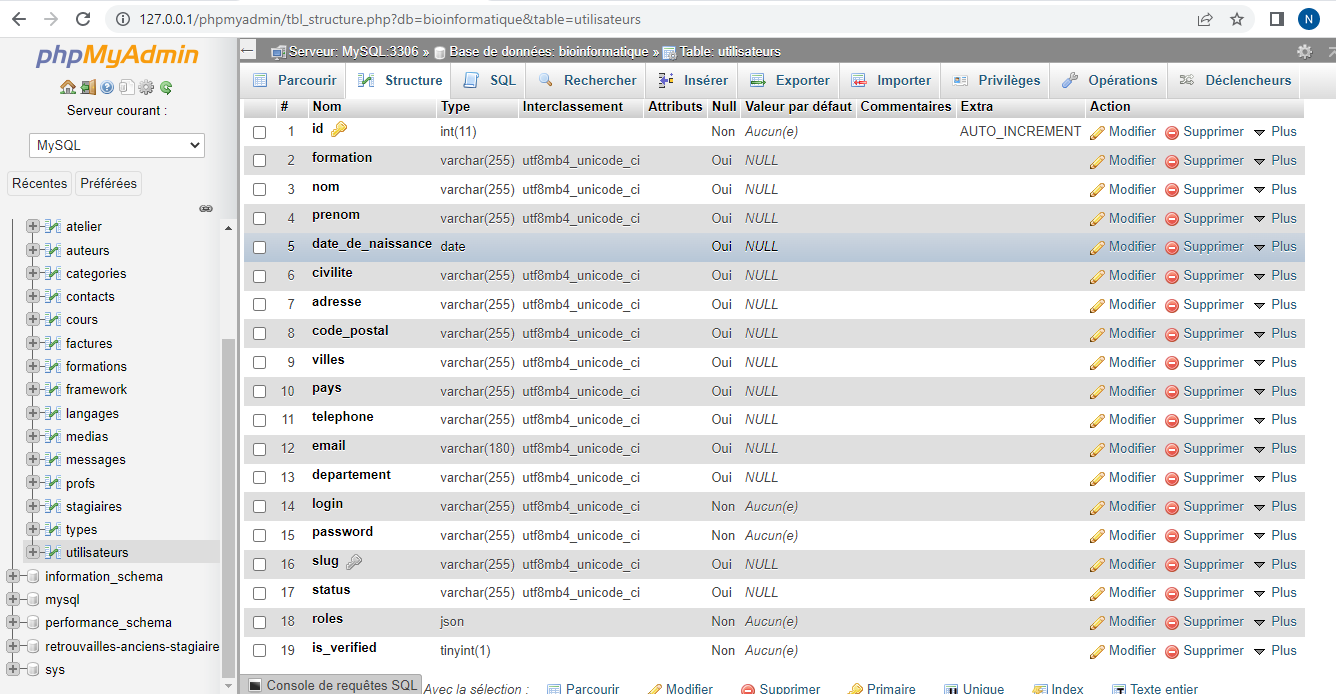
Base de donnée tables messages



Texte à remplir

## **Base de donnée tables Utilisateurs :**

Base de donnée tables Utilisateurs



Texte à remplir

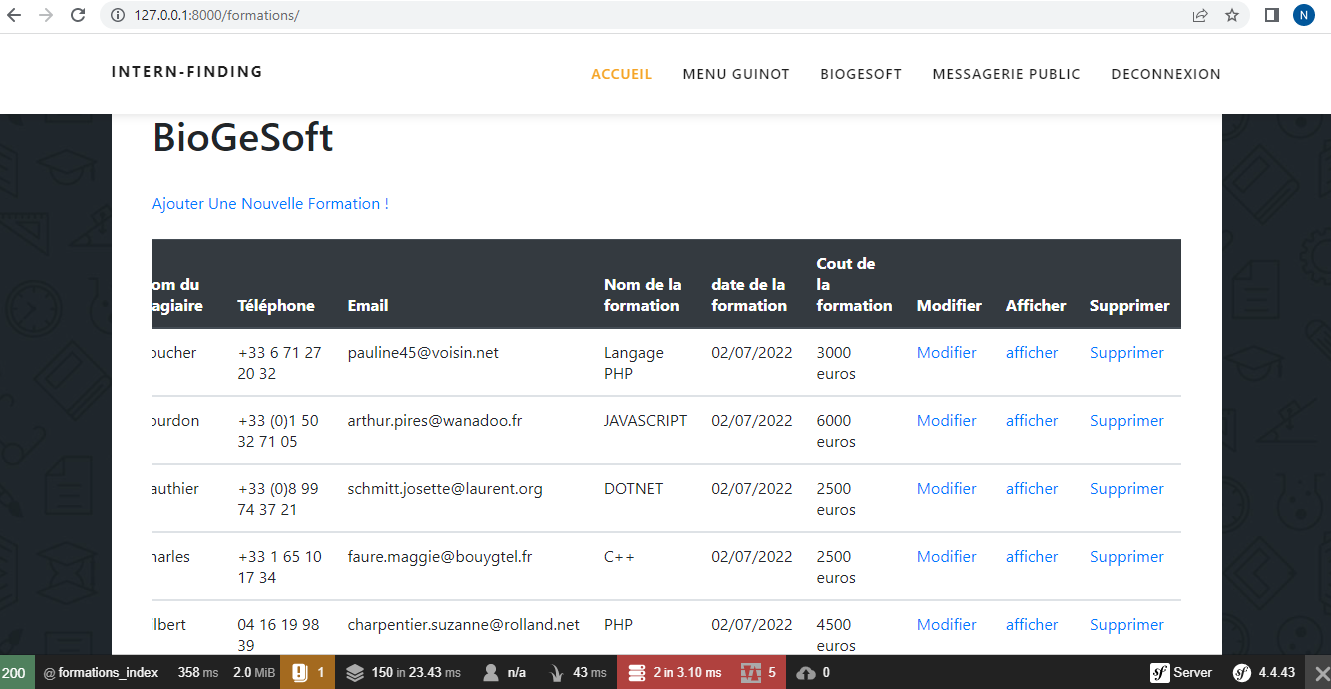
Capture du site web

## **Capture site accueil :**



## **Capture site liste formation :**

Capture site liste formation



## **Capture site enregistrement nouveau message :**

Capture site enregistrement nouveau message



## **Capture se connecter :**

Capture se connecter

