

SYMFONY 7

Crear un CRUD, Crear un Registro,
Autenticación, Integrar Bootstrap

CREAR UN CRUD

Con symfony podemos crear un CRUD automáticamente para manejar cada entidad que lo requiera.

➤ El comando es : **php bin/console make:crud**

Al ejecutarse el comando se iniciará un mensaje simple que te pedirá el nombre de la entidad que estás intentando usar para crear el CRUD.

➤ **php bin/console make:crud Nombre_entidad**

A continuación vamos a generar el crud de la entidad comentarios:

```
C:\wamp64\www\miAppSymfony>php bin/console make:crud Comentario
```

```
Choose a name for your controller class (e.g. ComentarioController) [ComentarioController]:  
>
```

```
Do you want to generate PHPUnit tests? [Experimental] (yes/no) [no]:  
>
```

```
created: src/Controller/ComentarioController.php  
created: src/Form/ComentarioType.php  
created: templates/comentario/_delete_form.html.twig  
created: templates/comentario/_form.html.twig  
created: templates/comentario/edit.html.twig  
created: templates/comentario/index.html.twig  
created: templates/comentario/new.html.twig  
created: templates/comentario/show.html.twig
```

Success!

```
Next: Check your new CRUD by going to /comentario/
```

```
C:\wamp64\www\miAppSymfony>
```

CREAR UN CRUD

Como podemos observar el comando nos ha creado el formulario, para introducir datos, el controlador para orquestar cada acción y las plantillas que nos permite manejar cada operación.

Todo esto se realiza en los formatos básico y estándar, luego se debe personalizar e ir mejorando el código básico generado.

Several white lines of varying lengths and angles are drawn in the bottom right corner of the slide, creating a modern, abstract graphic element.

CREAR UN CRUD

El generador CRUD registrará automáticamente una ruta con el nombre de la entidad que usó, un controlador que manejará los conceptos básicos del crud, como el listado principal, los formularios para editar, mostrar y eliminar automáticamente para ti.

Para ejecutarlo sería:

localhost/nombre-proyecto/public/index.php/Nombre_entidad

Donde puedes interactuar con cualquier vista que desees, por ejemplo, en la vista de índice debería ver todos los registros de la tabla, para la que has hecho el CRUD, de tu base de datos, y también puedes editar cada registro y actualizar sus valores, registrar una nueva persona o eliminarlo de la base de datos.



localhost/miAppSymfony/public/index.php/comentario

Comentario index

Id	Comentarios	Fecha	actions
2	adiossss	2024-12-12	show edit
3	hola y adios	2024-12-09	show edit
4	adios	2024-12-13	show edit


[Create new](#)


Para más información:

INSERTAR, ACTUALIZAR Y BORRAR

Create new Comentario

Comentarios

Fecha 

Usuario 


Posts 


[back to list](#)


Para más información:

Edit Comentario

Comentarios

Fecha 

Usuario 

Posts 

[back to list](#)

Para más información:

Comentario

Id	2
Comentarios	adiossss
Fecha	2024-12-12

[back to list](#) [edit](#)

Para más información:

CRUD DE LA ENTIDAD POST

Vamos a crear también el de Post, para ver como se pasan ahora las foreign key de cada tabla.

Several thin, white, parallel diagonal lines are positioned in the bottom right corner of the slide, extending from the right edge towards the center.


```
C:\wamp64\www\miAppSymfony>php bin/console make:crud Post
```

```
Choose a name for your controller class (e.g. PostController) [PostController]:  
>
```

```
Do you want to generate PHPUnit tests? [Experimental] (yes/no) [no]:  
>
```

```
created: src/Controller/PostController.php  
created: src/Form/PostType.php  
created: templates/post/_delete_form.html.twig  
created: templates/post/_form.html.twig  
created: templates/post/edit.html.twig  
created: templates/post/index.html.twig  
created: templates/post/new.html.twig  
created: templates/post/show.html.twig
```

Success!

```
Next: Check your new CRUD by going to /post/
```

Post index

Id	Título	Likes	Fecha	Contenido	actions
1	Reflexiones	3	2024-12-09	La meditación oriental	show edit
2	El debate	345	2024-12-03	Opiniones sobre el calentamiento climático	show edit
3	Como redactar un articulo??	234	2024-11-25	Reglas básicas de escritura divulgativa	show edit


[Create new](#)

Para más información:


Create new Post

Titulo

Likes

Fecha 

Contenido

Usuario 

[back to list](#)

Para más información:

Post

Id	1
Titulo	Reflexiones
Likes	3
Fecha	2024-12-09
Contenido	La meditación oriental


[back to list](#) [edit](#)

Para más información:


Edit Post

Titulo

Likes

Fecha 

Contenido

Usuario 

[back to list](#)

Para más información:

CRUD

Si vemos ahora la tabla comentarios ya se han rellenado la lista de los post que hemos insertado.

Create new Comentario

Comentarios

Fecha 

Usuario 

Posts 

[back to list](#)

Para más información:

Create new Comentario

Comentarios

Fecha 

Usuario 

Posts 

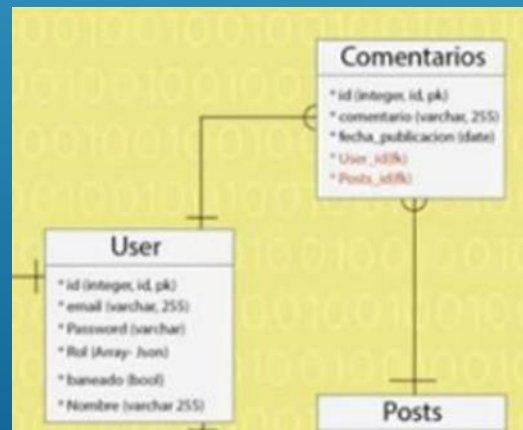
[back](#)

Para formación:

ERRORES FRECUENTE AL EJECUTAR ESTOS C.R.U.D.:

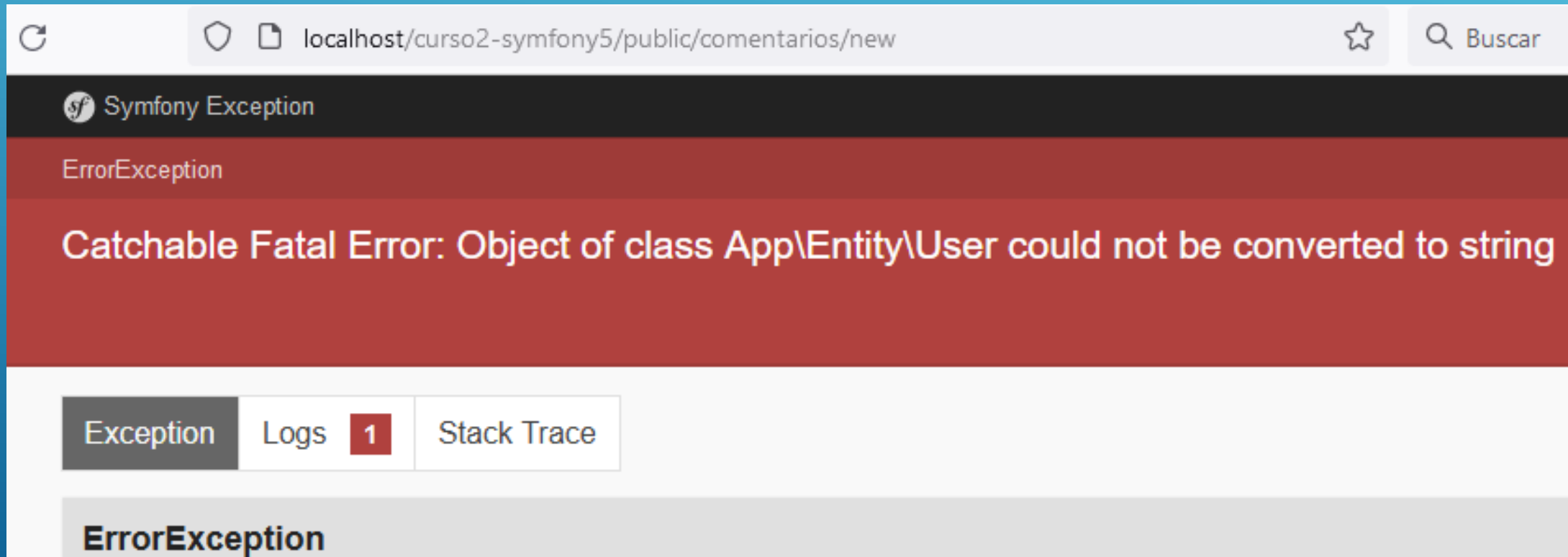
Contexto del error:

- Tenemos 2 tablas con una relación ManyToOne:
- Cuando necesitamos registrar o actualizar un nuevo registro en una tabla que hereda la foreign key, es necesario proporcionar la lista de datos de esa fk en la tabla que hereda esos valores.
- En nuestro ejemplo sería en la tabla comentarios, ya que debo poder elegir de entre todos los usuarios a quien le asigno el comentario.



ERRORES FRECUENTE AL EJECUTAR ESTOS C.R.U.D.:

Al ejecutar el crud tal cual lo genera el framework nos daría el siguiente error:



ERROR AL EJECUTAR ESTOS C.R.U.D.:

Solucion:

➤La única razón por la que aparece este error es porque no hay un método mágico `__toString` en la entidad Maestra(Usuarios en nuestro ejemplo), para mostrar en el campo de selección de la tabla detalle sus valores (Comentarios en nuestro ejemplo).

- Cuando se procesa el formulario, Symfony no sabe qué debería mostrarse en el campo de selección que causa el error, por lo que en este caso, como el select (id del usuario) debería permitir al usuario seleccionar uno de los múltiples datos registrados en el base de datos.
- Para ello la lista se debe imprimir como una cadena, para ello debemos devolver la propiedad del nombre del campo de la entidad que devuelve su valor en el método mágico.
- Por lo que solo hay que añadir este método en la entidad que genera la tabla maestra.

```

private $id;

/**
 * @ORM\Column(type="string", length=180, unique=true)
 */
private $email;

/**
 * @ORM\Column(type="json")
 */
private $roles = [];

/**
 * @var string The hashed password
 * @ORM\Column(type="string")
 */
private $password;

/**
 * @ORM\Column(type="boolean", nullable=true)
 */
private $banned;

/**
 * @ORM\Column(type="string", length=255)
 */
private $nombre;

/**
 * @ORM\OneToMany(targetEntity=Comentarios::class, mappedBy="usuario")
 */
private $comentarios;

```

```

private $email;

/**
 * @ORM\Column(type="json")
 */
private $roles = [];

/**
 * @var string The hashed password
 * @ORM\Column(type="string")
 */
private $password;

/**
 * @ORM\Column(type="boolean", nullable=true)
 */
private $banned;

/**
 * @ORM\Column(type="string", length=255)
 */
private $nombre;

public function __toString()
{
    return $this->nombre;
}

/**
 * @ORM\OneToMany(targetEntity=Comentarios::class, mappedBy="usuario")
 */
private $comentarios;

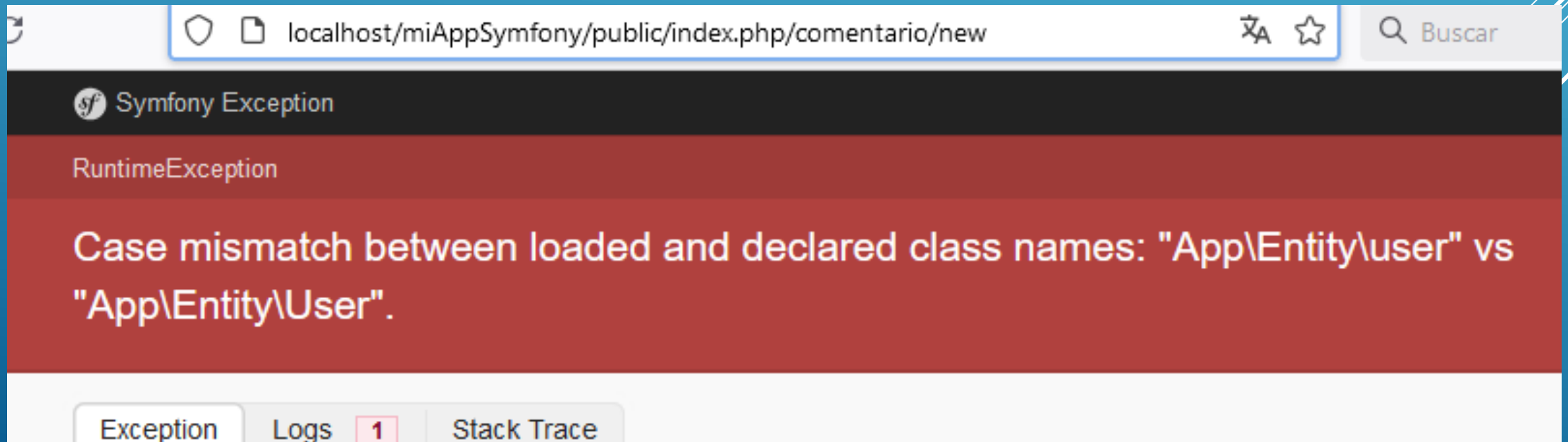
```

- De esta forma estoy permitiendo que el formulario representa una lista de los usuarios sin ninguna excepción.

ERRORES FRECUENTE AL EJECUTAR ESTOS C.R.U.D.:

El siguiente error es porque al crear la entidad se ha puesto el nombre de la entidad en minúsculas y el framework pone la primera letra en mayúsculas.

Solucion: Editar la entidad y ponerlo con la primera letra en mayúsculas sin añadir más campos.



```
C:\wamp64\www\miAppSymfony>php bin/console make:entity
```

```
Class name of the entity to create or update (e.g. AgreeableChef):
```

```
> Comentario
```

```
Your entity already exists! So let's add some new fields!
```

```
New property name (press <return> to stop adding fields):
```

```
>
```

Success!

```
Next: When you're ready, create a migration with php bin/console make:migration
```

REGISTRO DE LOGIN.

En el tema 4 de generación de entidades creamos la entidad User con el fin de realizar un formulario de autenticación, la orden de generación de esta entidad , realizaba cambios en el fichero de configuración security.yaml, actualizando tanto el providers, donde se le indica la ubicación de la entidad y el campo por el que se va a realizar el logueo para cargar usuarios desde una base de datos utilizando Doctrine, y también informa de la forma de hasear la clave de acceso ;

```
security:
  # https://symfony.com/doc/current/security.html#registering-the-user-hashing-passwords
password_hashers:
  Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterface: 'auto'
  # https://symfony.com/doc/current/security.html#loading-the-user-the-user-provider
providers:
  # used to reload user from session & other features (e.g. switch_user)
  app_user_provider:
    entity:
      class: App\Entity\User
      property: email
```

```
when@test:
  security:
    password_hashers:
      # By default, password hashers are resource intensive and take time. This is
      # important to generate secure password hashes. In tests however, secure hashes
      # are not important, waste resources and increase test times. The following
      # reduces the work factor to the lowest possible values.
      Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterface:
        algorithm: auto
        cost: 4 # Lowest possible value for bcrypt
        time_cost: 3 # Lowest possible value for argon
        memory_cost: 10 # Lowest possible value for argon
```

REGISTRO DE LOGIN.

Ahora que Symfony tiene la información de cómo vamos a codificar las contraseñas, y vamos a utilizar el servicio `UserPasswordHasherInterface` para codificar las claves antes de guardar tus usuarios en la base de datos.

Todo esto lo vamos a hacer en el controlador de registro de usuarios , que nos lo proporcionará el comando `make:registration-form` que configurará el controlador de registro y agregará funciones como la verificación de la dirección de correo electrónico utilizando `SymfonyCastsVerifyEmailBundle`

Instalamos la verificación de correo y generamos el formulario de registro

- `composer require symfonycasts/verify-email-bundle`
- `php bin/console make:registration-form`

```
C:\wamp64\www\miAppSymfony2>composer require symfonycasts/verify-email-bundle
./composer.json has been updated
Running composer update symfonycasts/verify-email-bundle
Loading composer repositories with package information
Updating dependencies
Lock file operations: 1 install, 0 updates, 0 removals
  - Locking symfonycasts/verify-email-bundle (v1.17.3)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 1 install, 0 updates, 0 removals
  - Downloading symfonycasts/verify-email-bundle (v1.17.3)
  - Installing symfonycasts/verify-email-bundle (v1.17.3): Extracting archive
Generating autoload files
113 packages you are using are looking for funding.
Use the `composer fund` command to find out more!

Symfony operations: 1 recipe (e4df826bef78683d5148276ca65e673c)
  - Configuring symfonycasts/verify-email-bundle (>=v1.17.3): From auto-generated recipe
Executing script cache:clear [OK]
Executing script assets:install public [OK]
Executing script importmap:install [OK]
```

What's next?

PHP BIN/CONSOLE MAKE:REGISTRATION-FORM

Nos va a Preguntar:

- Si queremos validación de usuario único: si
- Si queremos mandar un correo al usuario cuando se autentifique(como no vamos a configurar el envío de correo podríamos poner que no, pero descomentando la parte de MAILER_DSN=null://null del fichero .env aunque no mandemos un email no daría error, y así dejamos el controlador preparado el código.): si
- Nos va a preguntar por una dirección de correo para posibles envíos: symfony7@curso.com
- Qué "nombre" debe asociarse con esa dirección de correo electrónico: curso symfony7
- Si queremos ingresar nada mas autenticarnos: si
- Ahora debemos elegir la ruta donde queremos enviar al usuario registrado: he puesto 16 se puede luego cambiar , lo ideal sería mandarlo a un dashboard.

```
C:\wamp64\www\miAppSymfony2>php bin/console make:registration-form
```

```
Creating a registration form for App\Entity\User
```

```
Do you want to add a #[UniqueEntity] validation attribute to your User class to make sure duplicate accounts aren't
>
```

```
Do you want to send an email to verify the user's email address after registration? (yes/no) [yes]:
>
```

```
By default, users are required to be authenticated when they click the verification link that is emailed to them.
This prevents the user from registering on their laptop, then clicking the link on their phone, without
having to log in. To allow multi device email verification, we can embed a user id in the verification link.
```

```
Would you like to include the user id in the verification link to allow anonymous email verification? (yes/no) [no]:
> yes
```

```
What email address will be used to send registration confirmations? (e.g. mailer@your-domain.com):
> symfony7@curso.com
```

```
What "name" should be associated with that email address? (e.g. Acme Mail Bot):
> curso symfony7
```

```
Do you want to automatically authenticate the user after registration? (yes/no) [yes]:
>
```

```
! [NOTE] No authenticators found - so your user won't be automatically authenticated after registering.
```


What route should the user be redirected to after registration?:

```
[0 ] _preview_error
[1 ] _wdt_stylesheet
[2 ] _wdt
[3 ] _profiler_home
[4 ] _profiler_search
[5 ] _profiler_search_bar
[6 ] _profiler_phpinfo
[7 ] _profiler_xdebug
[8 ] _profiler_font
[9 ] _profiler_search_results
[10] _profiler_open_file
[11] _profiler
[12] _profiler_router
[13] _profiler_exception
[14] _profiler_exception_css
[15] app_test
[16] app_hola
> 16
```

Do you want to generate PHPUnit tests? [Experimental] (yes/no) [no]:
>

```
updated: src/Entity/User.php
updated: src/Entity/User.php
created: src/Security/EmailVerifier.php
created: templates/registration/confirmation_email.html.twig
created: src/Form/RegistrationFormType.php
created: src/Controller/RegistrationController.php
created: templates/registration/register.html.twig
```

Success!

Register

Email

Password

Agree terms ☐

Register

REGISTRO DE LOGIN.

Me permite asignar al usuario una contraseña hasheada, a partir de la que introduzco en el formulario. Esta se va a configurar a través de `UserPasswordHasherInterface`, el cual nos proporciona la función `hashPassword` a la que le pasamos como parámetro el usuario y la contraseña.

La contraseña la coge del formulario, que es donde la hemos introducido, y a continuación inserta los datos, manda el email, que como no está configurado lo vamos a comentar hasta que lo configuremos.

Register

Email

Password

Agree terms ☒

PDOException > Exception > InvalidFieldNameException

HTTP 500

An exception occurred while executing a query: SQLSTATE[42S22]: Column not found: 1054
Unknown column 't0.is_verified' in 'field list'

Exceptions 3

Logs 1

Stack Traces 3

Doctrine\DBAL\Exception

ERRORES Y SOLUCIONES:

El error quedá nos dice que is verify no se encuentra, esa es la columna del check del formulario, para solucionarlo o se quita del formulario el check, o se añade esa columna a la tabla de user .

Si usamos la primera opción debe ser porque esa columna no sea un campo de la entidad User, y entonces hay que ir al formulario y a la vista y comentamos el check.

```

class RegistrationFormType extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array $options): void
    {
        $builder
            ->add('email')
            /* ->add('agreeTerms', CheckboxType::class, [
                'mapped' => false,
                'constraints' => [
                    new IsTrue([
                        'message' => 'You should agree to our terms.',
                    ]),
                ],
            ]) */
            ->add('plainPassword', PasswordType::class, [
                // ...
            ])
    }
}

```

```

{{ form_start(registrationForm) }}
    {{ form_row(registrationForm.email) }}
    {{ form_row(registrationForm.plainPassword, {
        label: 'Password'
    }) }}
    {# {{ form_row(registrationForm.agreeTerms) }}#}

    <button type="submit" class="btn">Register</button>
{{ form_end(registrationForm) }}
{& endblock &}

```

ERRORES Y SOLUCIONES:

Pero en este caso también se ha generado en la entidad User por tanto es mejor añadirlo a la bbdd.

```
/**
 * @var Collection<int, Profesion>
 */
#[ORM\OneToMany(targetEntity: Profesion::class, mappedBy=
private Collection $profesions;

#[ORM\Column]
private bool $isVerified = false;

public function __construct()
```

ERRORES Y SOLUCIONES:

Para conseguirlo como es un cambio que se ha producido al hacer el registro, solo hay que actualizar con

- `Php bin/console doctrine:schema:update --force`

```
C:\wamp64\www\miAppSymfony2>php bin/console doctrine:schema:update --force
Updating database schema...
```

```
1 query was executed
```

```
[OK] Database schema updated successfully!
```

```
SELECT * FROM `user`
```

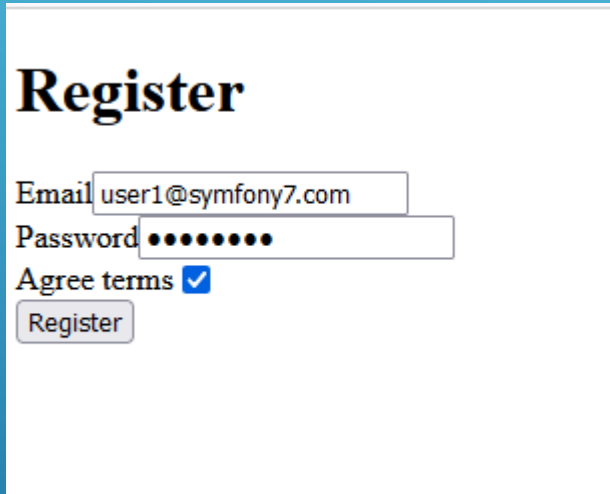
☐ Perfilando [[Editar en línea](#)] [[Editar](#)] [[Explicar SQL](#)] [[Crear código](#)]

id	email	roles	password	baneado	nombre	is_verified
----	-------	-------	----------	---------	--------	-------------

Operaciones sobre los resultados de la consulta

ERRORES Y SOLUCIONES:

Por tanto volvemos a ejecutar



The screenshot shows a web registration form with the following elements:

- Register**: The title of the form.
- Email**: A text input field containing the value `user1@symfony7.com`.
- Password**: A text input field with masked characters (dots).
- Agree terms**: A checkbox that is checked, indicated by a blue checkmark.
- Register**: A button to submit the form.

PDOException > Exception > NotNullConstraintViolationException

HTTP 500 Internal

An exception occurred while executing a query: SQLSTATE[23000]: Integrity constraint violation: 1048 Column 'nombre' cannot be null

Exception

ERRORES Y SOLUCIONES:

Ahora nos dice que la propiedad nombre que añadimos cuando hicimos la entidad user, no puede ser nulo, como en el formulario no existe, debo añadirla para poder introducir en el registro.

```
$builder
->add('email')
->add('agreeTerms', CheckboxType::class, [
    'mapped' => false,
    'constraints' => [
        new IsTrue([
            'message' => 'You should agree to our terms.',
        ]),
    ],
])
->add('plainPassword', PasswordType::class, [
    // instead of being set onto the object directly,
    // this is read and encoded in the controller
    'mapped' => false,
    'attr' => ['autocomplete' => 'new-password'],
    'constraints' => [
        new NotBlank([
            'message' => 'Please enter a password',
        ]),
        new Length([
```



```
$builder
->add('email')
->add('nombre')
->add('agreeTerms', CheckboxType::class, [
    'mapped' => false,
    'constraints' => [
        new IsTrue([
            'message' => 'You should agree
    ],
])
->add('plainPassword', PasswordType::class,
    // instead of being set onto the object
    // this is read and encoded in the cont
    'mapped' => false,
    'attr' => ['autocomplete' => 'new-passw
    'constraints' => [
        new NotBlank([
            'message' => 'Please enter a pa
        ]),
        new Length([
```

ERRORES Y SOLUCIONES:

Y también hay que añadirlo a la parte de la vista:

```
{{ form_errors(registrationForm) }}

{{ form_start(registrationForm) }}
{{ form_row(registrationForm.email) }}
{{ form_row(registrationForm.plainPassword, {
    label: 'Password'
}) }}
{{ form_row(registrationForm.agreeTerms) }}

<button type="submit" class="btn">Register</button>
{{ form_end(registrationForm) }}
```



```
{{ form_start(registrationForm) }}
{{ form_row(registrationForm.email) }}
{{ form_row(registrationForm.nombre) }}
{{ form_row(registrationForm.plainPassword, {
    label: 'Password'
}) }}
{{ form_row(registrationForm.agreeTerms) }}

<button type="submit" class="btn">Register</button>
{{ form_end(registrationForm) }}
```

Ahora si ejecutamos debería estar ya el nombre para poder introducirlo:

Register

Email

Nombre

Password

Agree terms ☒



ERRORES Y SOLUCIONES:

Como podemos ver en la BBDD se han insertado los usuarios y con las contraseñas haseadas:

id	email	roles	password	baneado	nombre	is_ve
1	user1@symfony7.com	[]	\$2y\$13\$0PIUPnpNpv0B9q.UsZd8U.nqCxcUbEQVZHfFAgdHSi...	NULL	Ana Antunez Sanz	
2	user2@symfony7.com	[]	\$2y\$13\$7vKYapvaTCs3rn4hmP/a4.g/bThy5zP1VrNiLt64BEt...	NULL	Ana Antunez Sanz	
3	user3@symfony7.com	[]	\$2y\$13\$YBoRU6RP/oGCi5oiwV.Y4OiKbzTAiFJdPpw/q84vEg0...	NULL	Ana Antunez Sanz	

AUTENTICACIÓN (LOGIN-LOGOUT)

Ahora vamos a proceder a la autenticación, para crear todo lo necesario y agregar un formulario de inicio de sesión en su aplicación ejecutaremos:

- `php bin/console make:security:form-login`
- pregunta por un nombre para el controlador de login: `FormLoginController`.
- Pregnta Si genera una ruta logout: `yes`

Y ya estaría configurado, además de la vista y el controlado ha actualizado el fichero `security.yaml` y que ha añadido la ruta logout .

```
C:\wamp64\www\miAppSymfony2>php bin/console make:security:form-login
```

```
Choose a name for the controller class (e.g. SecurityController) [SecurityController]:  
> FormLoginController
```

```
Do you want to generate a '/logout' URL? (yes/no) [yes]:  
> yes
```

```
Do you want to generate PHPUnit tests? [Experimental] (yes/no) [no]:  
>
```

```
created: src/Controller/FormLoginController.php  
created: templates/formlogin/login.html.twig  
updated: config/packages/security.yaml
```

Success!

```
Next: Review and adapt the login template: formlogin/login.html.twig to suit your needs.
```

```

security:
# https://symfony.com/doc/current/security.html#registering-the-user-hashing-passwords
password_hashers:
    Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterface: 'auto'
# https://symfony.com/doc/current/security.html#loading-the-user-the-user-provider
providers:
    # used to reload user from session & other features (e.g. switch_user)
    app_user_provider:
        entity:
            class: App\Entity\User
            property: email
firewalls:
    dev:
        pattern: ^/(_(profiler|wdt)|css|images|js)/
        security: false
    main:
        lazy: true
        provider: app_user_provider

        # activate different ways to authenticate
        # https://symfony.com/doc/current/security.html#the-firewall

        # https://symfony.com/doc/current/security/impersonating user.html
        # switch_user: true

# Easy way to control access for large sections of your site
# Note: Only the 'first' access control that matches will be used
access_control:
    # - { path: ^/admin, roles: ROLE_ADMIN }
    # - { path: ^/profile, roles: ROLE_USER }

when@test:
    security:
        password_hashers:
            # By default, password hashers are resource intensive and take time. Th
            # important to generate secure password hashes. In tests however, secur
            # are not important, waste resources and increase test times. The follo
            # reduces the work factor to the lowest possible values.
            Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterface
                algorithm: auto
                cost: 4 # Lowest possible value for bcrypt

```

```

# https://symfony.com/doc/current/security.html#registering-the-user-hashing-passwords
password_hashers:
    Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterface: 'auto'
# https://symfony.com/doc/current/security.html#loading-the-user-the-user-provider
providers:
    # used to reload user from session & other features (e.g. switch_user)
    app_user_provider:
        entity:
            class: App\Entity\User
            property: email
firewalls:
    dev:
        pattern: ^/(_(profiler|wdt)|css|images|js)/
        security: false
    main:
        lazy: true
        provider: app_user_provider
        form_login:
            login_path: app_login
            check_path: app_login
            enable_csrf: true
        logout:
            path: app_logout
            # where to redirect after logout
            # target: app_any_route

        # activate different ways to authenticate
        # https://symfony.com/doc/current/security.html#the-firewall

        # https://symfony.com/doc/current/security/impersonating user.html
        # switch_user: true

# Easy way to control access for large sections of your site
# Note: Only the 'first' access control that matches will be used
access_control:
    # - { path: ^/admin, roles: ROLE_ADMIN }

```

El controlador tiene las
Rutas login y logout:

```
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Attribute\Route;
use Symfony\Component\Security\Http\Authentication\AuthenticationUtils;

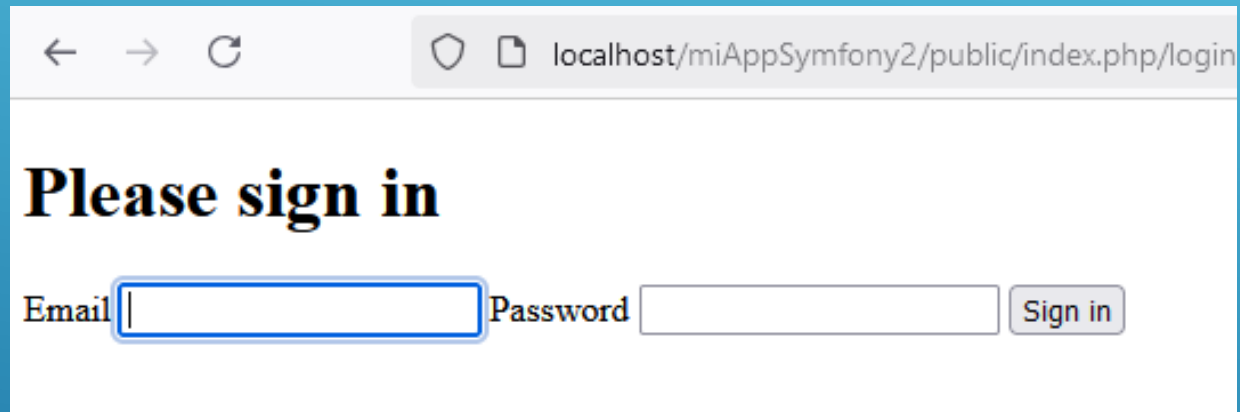
class FormLoginController extends AbstractController
{
    #[Route(path: '/login', name: 'app_login')]
    public function login(AuthenticationUtils $authenticationUtils): Response
    {
        // get the login error if there is one
        $error = $authenticationUtils->getLastAuthenticationError();

        // last username entered by the user
        $lastUsername = $authenticationUtils->getLastUsername();

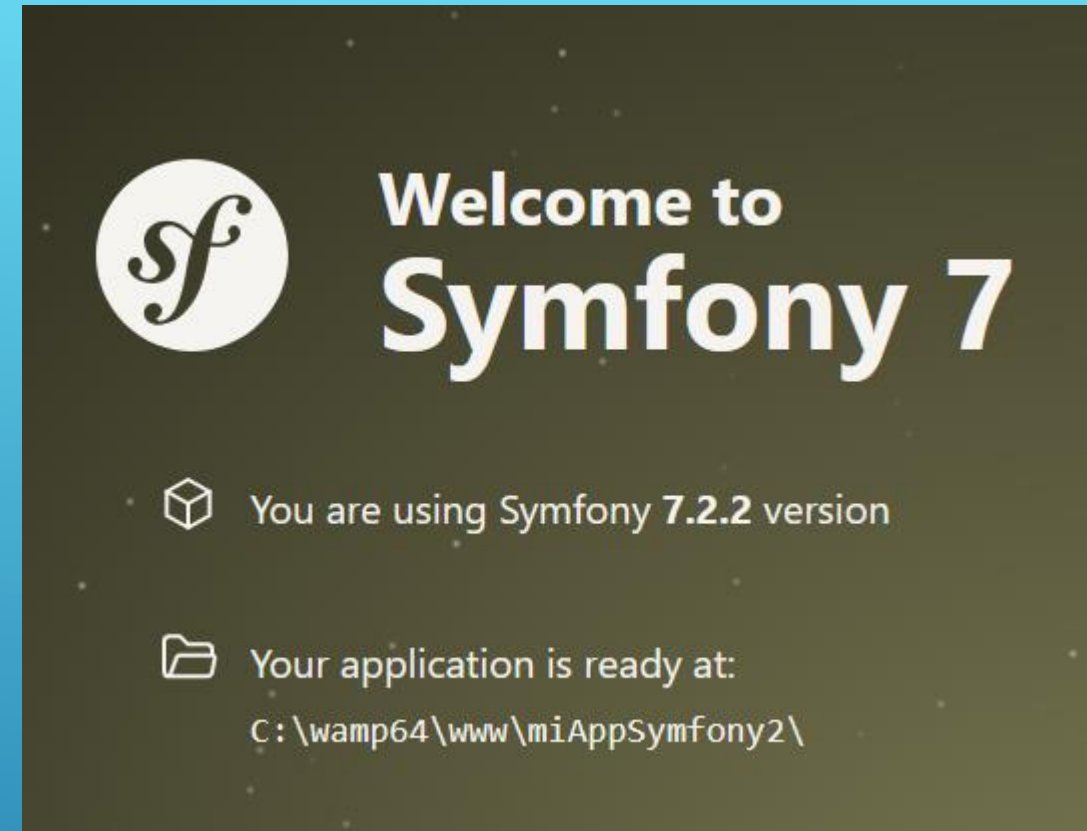
        return $this->render('formlogin/login.html.twig', [
            'last_username' => $lastUsername,
            'error' => $error,
        ]);
    }

    #[Route(path: '/logout', name: 'app_logout')]
    public function logout(): void
    {
        throw new \LogicException('This method can be blank - it will be interpreted as empty by the router');
    }
}
```

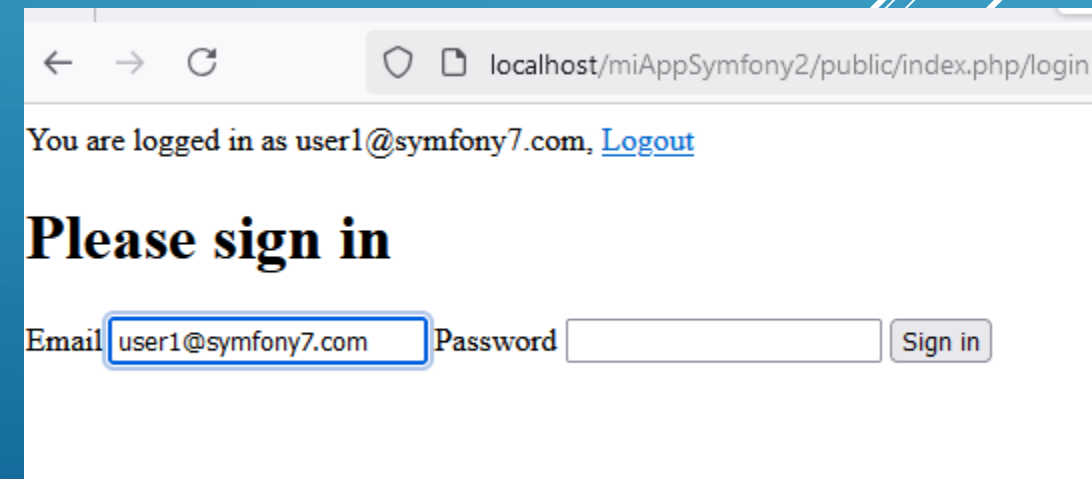

Si vamos a login nos mostrará un Formulario para loguearnos, y nos manda a la pg de bienvenida.



A screenshot of a web browser window showing a login form. The address bar displays 'localhost/miAppSymfony2/public/index.php/login'. The form has the heading 'Please sign in' and contains two input fields labeled 'Email' and 'Password', followed by a 'Sign in' button.




Y si volvemos a la ruta login, como ya Estoy autenticado, me da la opción de logout



A screenshot of the login form in a web browser, showing the user is logged in. The address bar displays 'localhost/miAppSymfony2/public/index.php/login'. At the top, it says 'You are logged in as user1@symfony7.com, [Logout](#)'. Below this is the 'Please sign in' form with the 'Email' field pre-filled with 'user1@symfony7.com' and a 'Sign in' button.


AUTENTICACIÓN (LOGIN-LOGOUT)

Se encarga de validarlo , por tanto si introduzco un usuario erróneo lo indica para corregirlo.



Please sign in

Email Password



Invalid credentials.

Please sign in

Email Password

Una vez logueado correctamente en la barra inferior se puede verificar:

You are logged in as user2@symfony7.com, [Logout](#)

Please sign in

Email Password

Logged in as	user2@symfony7.com
Authenticated	Yes
Roles	ROLE_USER
Inherited Roles	none
Token class	<u>UsernamePasswordToken</u>
Firewall name	main
Actions	Logout

INTEGRAR BOOTSTRAP

Para añadir Bootstrap, editamos el fichero templates/base.html.twig y en etiqueta head copio y pego la hoja de estilos en el <link> de la etiqueta <head>, antes de todas las demás hojas de estilos para cargar nuestro CSS:

```
{% block stylesheets %}  
    <link  
    href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"  
    rel="stylesheet" integrity="sha384-  
    EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC"  
    crossorigin="anonymous"/>  
{% endblock %}
```

INTEGRAR BOOTSTRAP

El siguiente paso es configurar la aplicación Symfony para que utilice los estilos de Bootstrap 5 al mostrar formularios. Si desea aplicarlos a todos los formularios, defina esta configuración:

```
# config/packages/twig.yaml
twig:
    form_themes: ['bootstrap_5_layout.html.twig']
```

Y con esto tendríamos un crud responsivo

```

<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>{% block title %}Welcome!{% endblock %}</title>

    {% block stylesheets %}

      <!--<link href="{{ asset('css/estilos.css') }}" rel="stylesheet" type="text/css"/>-->
      <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC"
crossorigin="anonymous"/>
    {% endblock %}

```

```

twig:
  #file_name_pattern: '*.twig'
  form_themes: ['bootstrap_5_layout.html.twig']

when@test:
  twig:
    strict_variables: true

```