

1. Lista secuencial de comandos vía terminal utilizados, una breve descripción de la utilidad que tienen y los componentes que afectan la ejecución de los mismos.

- a. `composer create-project symfony/skeleton proyectosym`
- b. `cd proyectosym`
- c. `composer require webapp`
- d. `symfony server:start`
- e. `composer require laminas/laminas-code`
- f. `composer require maker`
- g. `symfony console make:controller IndexController`
- h. `composer require "twig"`
- i. `composer require symfony/orm-pack`
- j. `composer require --dev symfony/maker-bundle`

Hasta aquí instalamos los paquetes necesarios para utilizar herencia y poder crear componentes mediante comandos

- k. `symfony console doctrine:database:create`
- l. `symfony console make:entity`
- m. `php bin/console make:migration`
- n. `: php bin/console doctrine:migrations:migrate`
- o. `php bin/console make:crud`

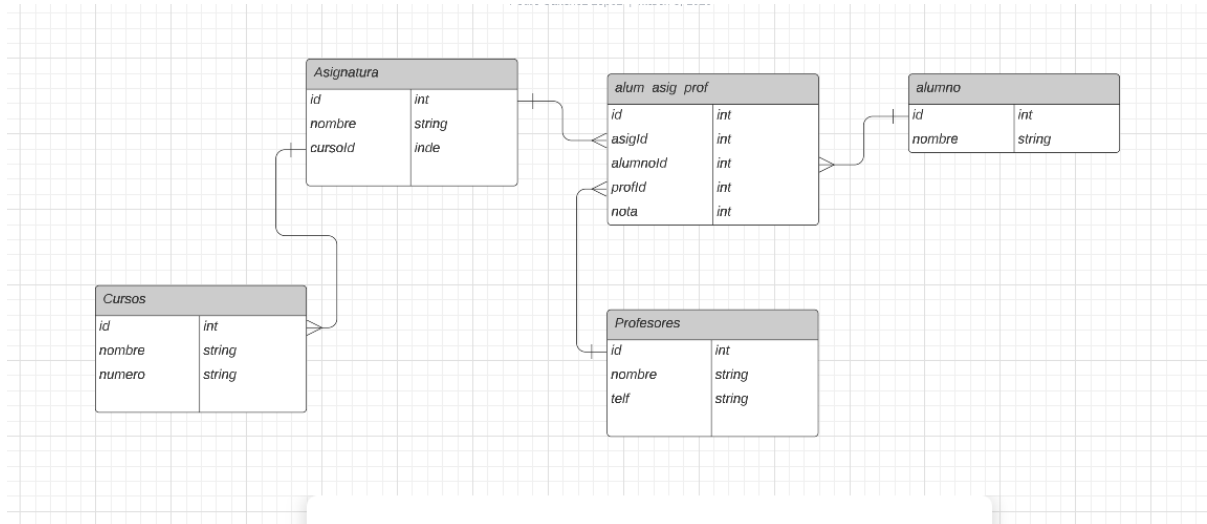
Creamos la base de datos, las entidades, las migraciones y enviamos la migración a la base de datos

Para cada creación de entidad se despliega un asistente que nos facilita la escritura del código al hacerlo por nosotros, preguntándonos por los nombres y los tipos de los atributos de la tablas

- p. `composer require symfony/security-bundle`
- q. `php bin/console make:user`
- r. `php bin/console make:migration`
- s. `php bin/console doctrine:migrations:migrate`
- t. `composer require symfony/mailer`
- u. `php bin/console make:controller Login`
- v. `php bin/console make:controller Registration`

Instalamos el bundle para implementar la seguridad, migramos la table usuarios, creada y creamos los controladores para el login y el registro

Como modelo de base de datos hemos usado el siguiente:



Podemos ver cómo alum_asig_prof relaciona a Asignaturas, Alumnos y Profesores y cómo Cursos es el contenedor de Asignaturas

Y en lo que respecta a la lógica y diferentes vistas podemos ver las siguientes imagenes

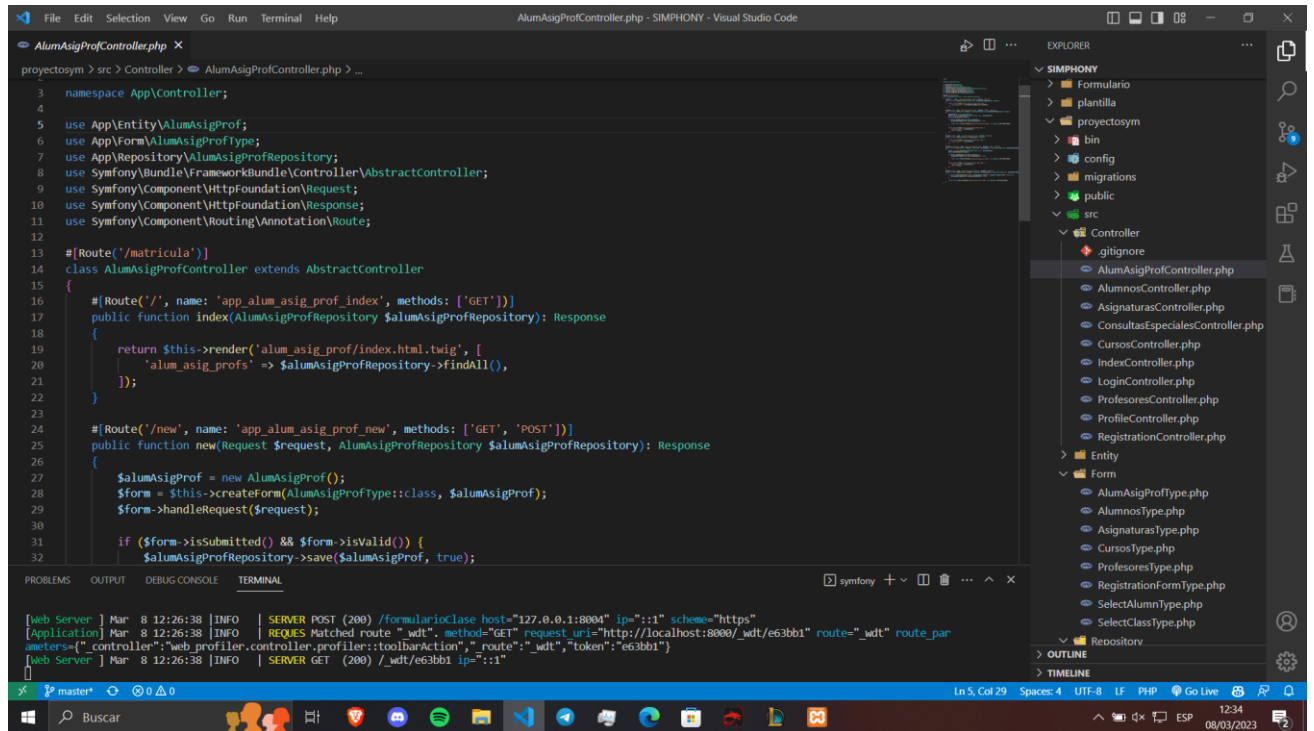
```
security.yaml
projectosym > config > packages > security.yaml
8 app_user_provider:
9   entity:
10     class: App\Entity\User
11     property: email
12 firewall:
13   dev:
14     pattern: ^/(_profiler|_wdt|_css|_images|_js)/
15     security: false
16   main:
17     lazy: true
18     # provider that you set earlier inside providers
19     provider: app_user_provider
20     form_login:
21       # "app_login" is the name of the route created previously
22       login_path: app_login
23       check_path: app_login
24       enable_csrf: true
25     login_throttling:
26       max_attempts: 4 # per minute ...
27       # interval: '15 minutes' # ... or in a custom period
28     logout:
29       path: app_logout
30       target: app_login
31       # where to redirect after logout
32     # activate different ways to authenticate
33     # https://symfony.com/doc/current/security.html#firewalls-authentication
34     # https://symfony.com/doc/current/security/impersonating_user.html
35     # switch_user: true
36 # Easy way to control access for large sections of your site
37 # Note: Only the "first" access control that matches will be used
38 access_control:
39   - { path: ^/index, roles: ROLE_USER }
40   - { path: ^/alumnos, roles: ROLE_USER }
41   - { path: ^/alumnos/, roles: ROLE_ADMIN }
```

El explorador de archivos a la derecha muestra la estructura del proyecto:

- projectosym
- bin
- config
- packages
 - cache.yaml
 - debug.yaml
 - doctrine_migrations.yaml
 - framework.yaml
 - mailer.yaml
 - messenger.yaml
 - notifier.yaml
 - routing.yaml
 - security.yaml
 - sensio_framework_extra.yaml
 - translation.yaml
 - twig.yaml
 - validator.yaml
 - web_profiler.yaml
- routes
 - bundles.php
 - preload.php
 - routes.yaml
 - services.yaml
- migrations
- public
- src
 - Controller
 - AlumAsigProfController.php
 - AlumnosController.php
 - AsignaturasController.php
 - ConsultasEspecialesController.php
 - CursosController.php
 - IndexController.php

Juan Carlos Ramírez Guerrero

En el archivo de security.yaml añadimos que vamos a usar login, logout así como restricciones de paths para diferentes tipos de usuarios

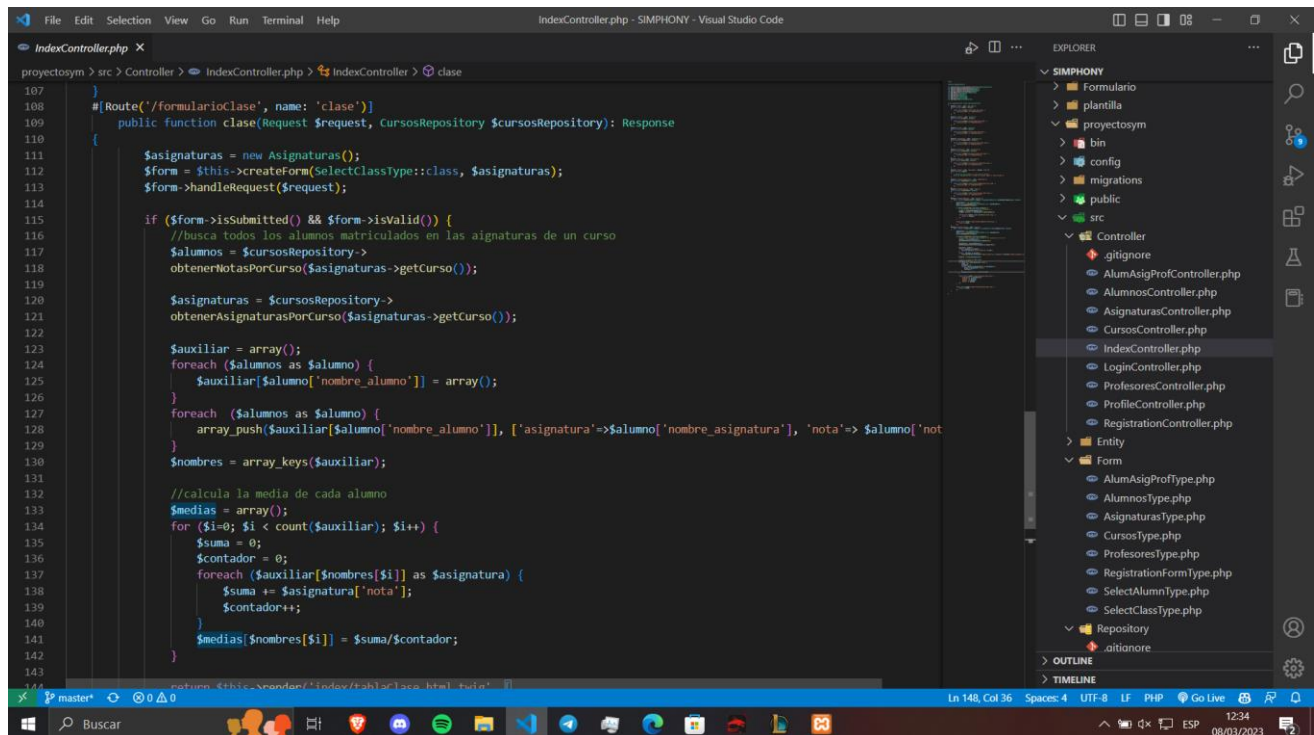


```
AlumAsigProfController.php - SIMPHONY - Visual Studio Code

projectosym > src > Controller > AlumAsigProfController.php > ...

3 namespace App\Controller;
4
5 use App\Entity\AlumAsigProf;
6 use App\Form\AlumAsigProfType;
7 use App\Repository\AlumAsigProfRepository;
8 use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
9 use Symfony\Component\HttpFoundation\Request;
10 use Symfony\Component\HttpFoundation\Response;
11 use Symfony\Component\Routing\Annotation\Route;
12
13 #[Route('/matricula')]
14 class AlumAsigProfController extends AbstractController
15 {
16     #[Route('/', name: 'app_alum_asig_prof_index', methods: ['GET'])]
17     public function index(AlumAsigProfRepository $alumAsigProfRepository): Response
18     {
19         return $this->render('alum_asig_prof/index.html.twig', [
20             'alum_asig_profs' => $alumAsigProfRepository->findAll(),
21         ]);
22     }
23
24     #[Route('/new', name: 'app_alum_asig_prof_new', methods: ['GET', 'POST'])]
25     public function new(Request $request, AlumAsigProfRepository $alumAsigProfRepository): Response
26     {
27         $alumAsigProf = new AlumAsigProf();
28         $form = $this->createForm(AlumAsigProfType::class, $alumAsigProf);
29         $form->handleRequest($request);
30
31         if ($form->isSubmitted() && $form->isValid()) {
32             $alumAsigProfRepository->save($alumAsigProf, true);
33         }
34     }
35 }
```

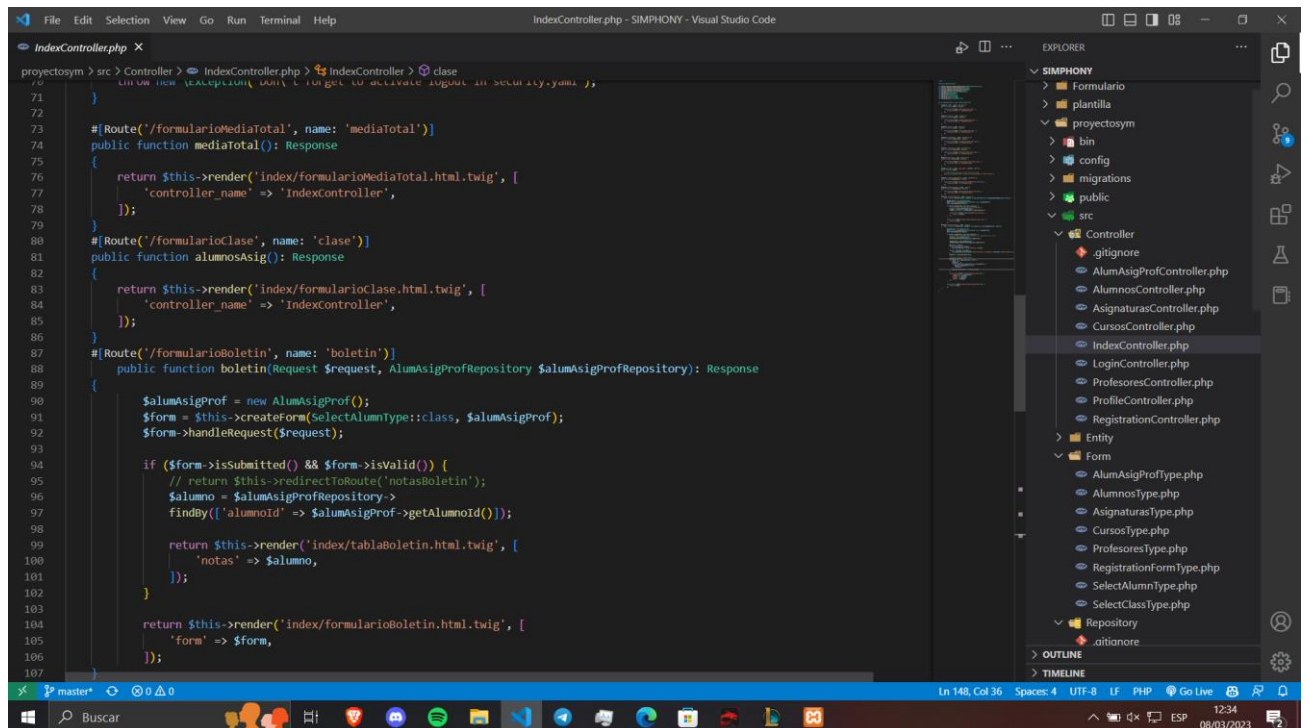
Creamos diferentes controladores para manejar los cruds de creación de cada entidad



```
IndexController.php - SIMPHONY - Visual Studio Code

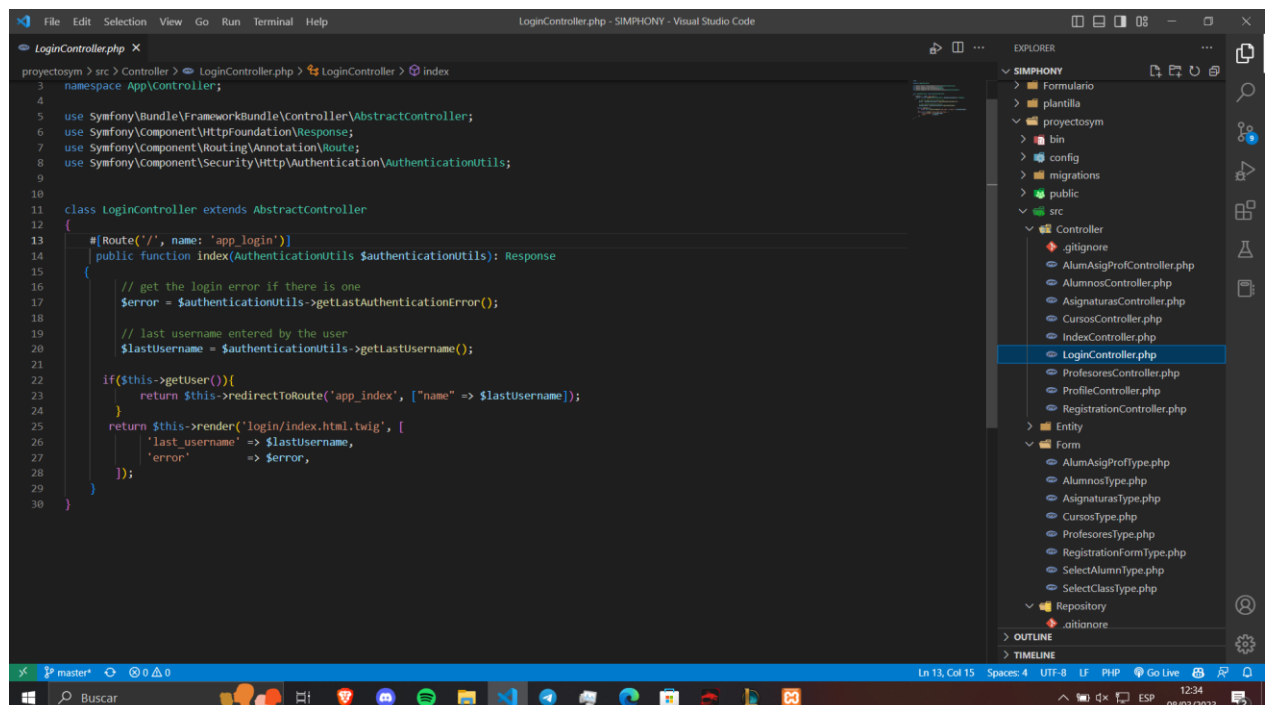
projectosym > src > Controller > IndexController.php > class

107
108 #[Route('/formularioClass', name: 'class')]
109 public function class(Request $request, cursosRepository $cursosRepository): Response
110 {
111     $asignaturas = new Asignaturas();
112     $form = $this->createForm(SelectClassType::class, $asignaturas);
113     $form->handleRequest($request);
114
115     if ($form->isSubmitted() && $form->isValid()) {
116         //busca todos los alumnos matriculados en las asignaturas de un curso
117         $alumnos = $cursosRepository->
118             obtenerNotasPorCurso($asignaturas->getCurso());
119
120         $asignaturas = $cursosRepository->
121             obtenerAsignaturasPorCurso($asignaturas->getCurso());
122
123         $auxiliar = array();
124         foreach ($alumnos as $alumno) {
125             $auxiliar[$alumno['nombre_alumno']] = array();
126         }
127         foreach ($alumnos as $alumno) {
128             array_push($auxiliar[$alumno['nombre_alumno']], ['asignatura'=>$alumno['nombre_asignatura'], 'nota'=> $alumno['nota']]);
129         }
130         $nombres = array_keys($auxiliar);
131
132         //calcula la media de cada alumno
133         $medias = array();
134         for ($i=0; $i < count($auxiliar); $i++) {
135             $suma = 0;
136             $contador = 0;
137             foreach ($auxiliar[$nombres[$i]] as $asignatura) {
138                 $suma += $asignatura['nota'];
139                 $contador++;
140             }
141             $medias[$nombres[$i]] = $suma/$contador;
142         }
143
144         return $this->render('index/formularioClass.html.twig', [
145             'medias' => $medias,
146             'nombres' => $nombres,
147         ]);
148     }
149 }
```



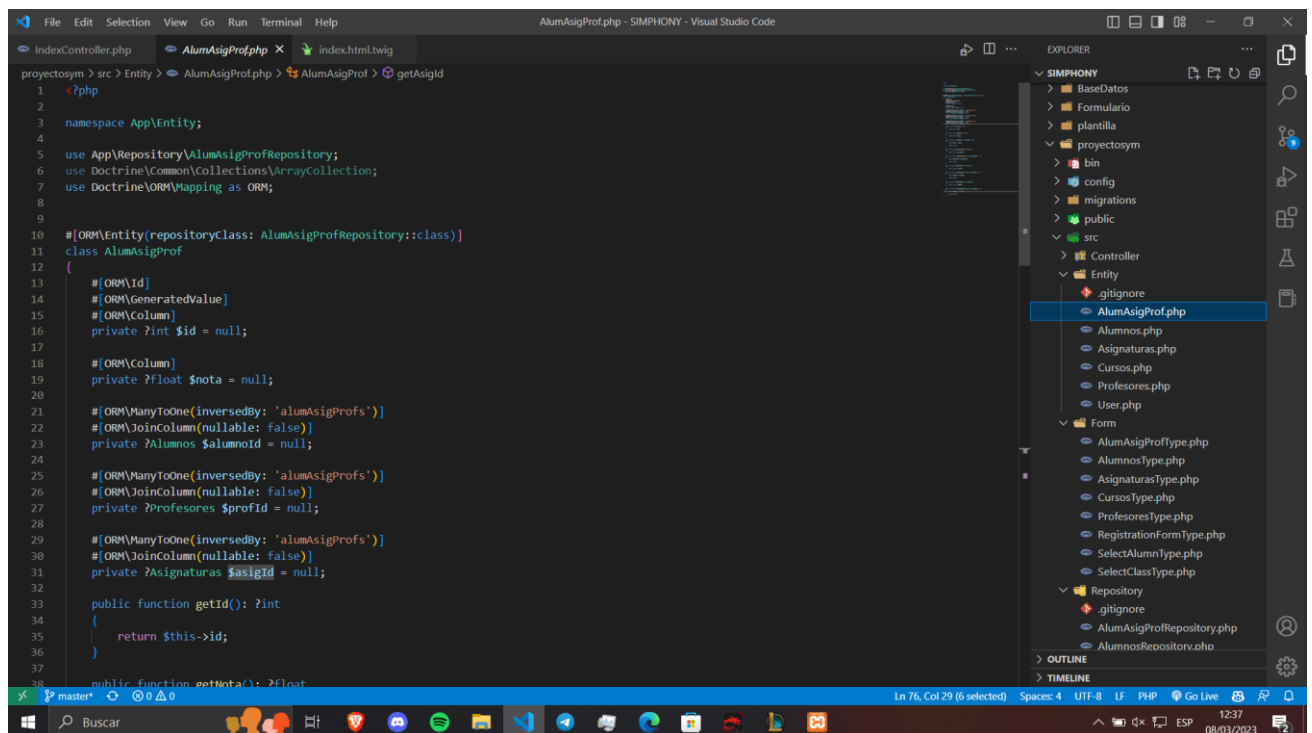
```
IndexController.php
projectosym > src > Controller > IndexController.php > class
70
71
72
73 #[Route('/formularioMediaTotal', name: 'mediaTotal')]
74 public function mediaTotal(): Response
75 {
76     return $this->render('index/formularioMediaTotal.html.twig', [
77         'controller_name' => 'IndexController',
78     ]);
79 }
80 #[Route('/formularioClase', name: 'clase')]
81 public function alumnosAsig(): Response
82 {
83     return $this->render('index/formularioClase.html.twig', [
84         'controller_name' => 'IndexController',
85     ]);
86 }
87 #[Route('/formularioBoletin', name: 'boletin')]
88 public function boletin(Request $request, AlumAsigProfRepository $alumAsigProfRepository): Response
89 {
90     $alumAsigProf = new AlumAsigProf();
91     $form = $this->createForm(SelectAlumnType::class, $alumAsigProf);
92     $form->handleRequest($request);
93
94     if ($form->isSubmitted() && $form->isValid()) {
95         // return $this->redirectToRoute('notasBoletin');
96         $alumno = $alumAsigProfRepository->
97             findBy(['alumnoId' => $alumAsigProf->getAlumnoId()]);
98
99         return $this->render('index/tablaBoletin.html.twig', [
100             'notas' => $alumno,
101         ]);
102     }
103
104     return $this->render('index/formularioBoletin.html.twig', [
105         'form' => $form,
106     ]);
107 }
```

También tenemos un controlador IndexController.php para movernos por la página, y que a la vez nos sirve para tomar los datos de las consultas personalizadas y ofrecérselas a las vistas con las tablas de formularioClase y formularioBoletin

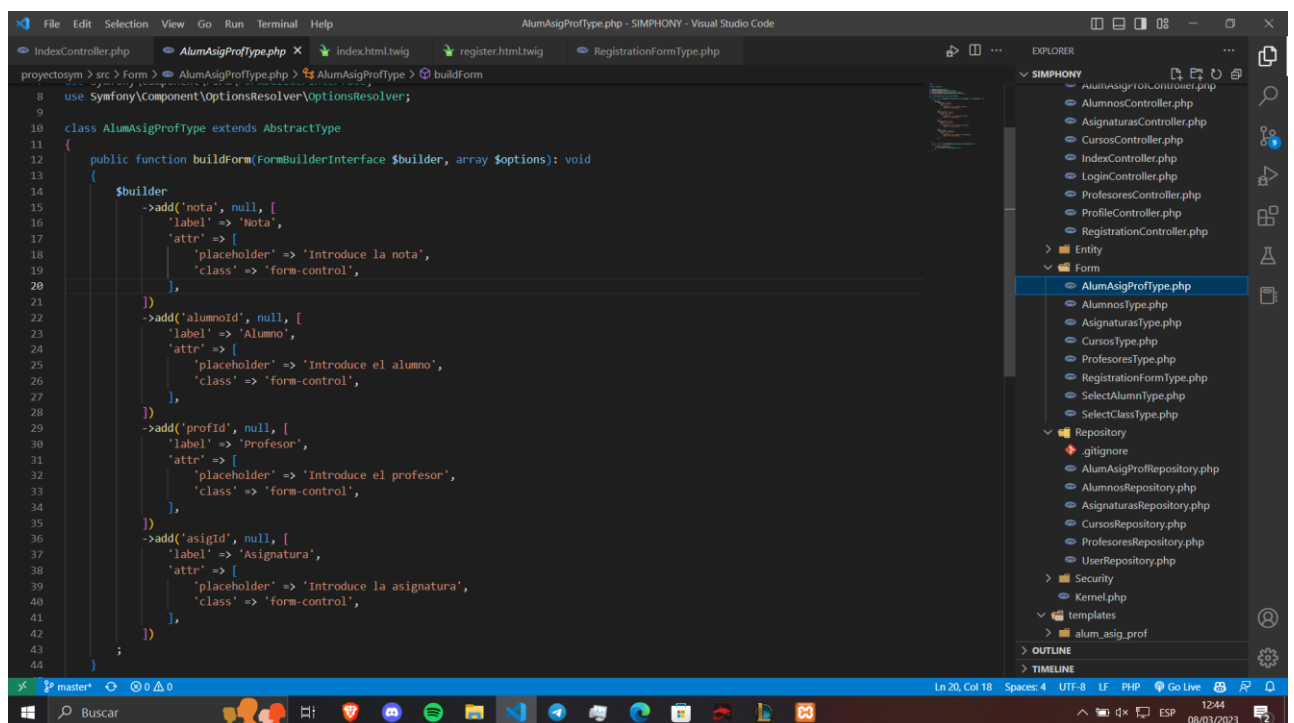


```
LoginController.php
projectosym > src > Controller > LoginController.php > index
3 namespace App\Controller;
4
5 use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
6 use Symfony\Component\HttpFoundation\Response;
7 use Symfony\Component\Routing\Annotation\Route;
8 use Symfony\Component\Security\Http\Authentication\AuthenticationUtils;
9
10
11 class LoginController extends AbstractController
12 {
13     #[Route('/', name: 'app_login')]
14     public function index(AuthenticationUtils $authenticationUtils): Response
15     {
16         // get the login error if there is one
17         $error = $authenticationUtils->getLastAuthenticationError();
18
19         // last username entered by the user
20         $lastUsername = $authenticationUtils->getLastUsername();
21
22         if($this->getUser()){
23             return $this->redirectToRoute('app_index', ["name" => $lastUsername]);
24         }
25         return $this->render('login/index.html.twig', [
26             'last_username' => $lastUsername,
27             'error' => $error,
28         ]);
29     }
30 }
```

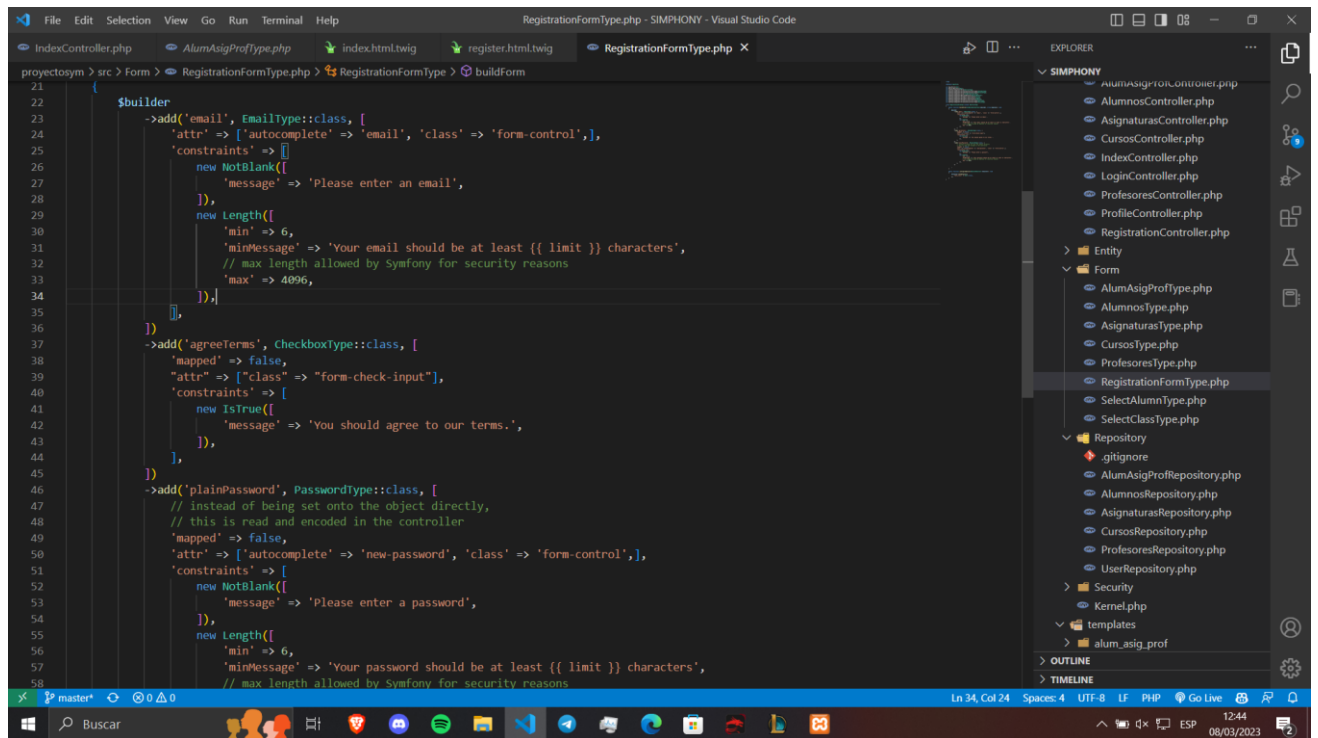
En el controlador de login comprobamos que estamos logeados antes de permitirnos acceder al resto de la página web



Para cada tabla de la base de datos tenemos una entidad asociada y representada en las migraciones, además, como se puede ver, presentamos una entidad `User`, que nos permite realizar los registros de usuario

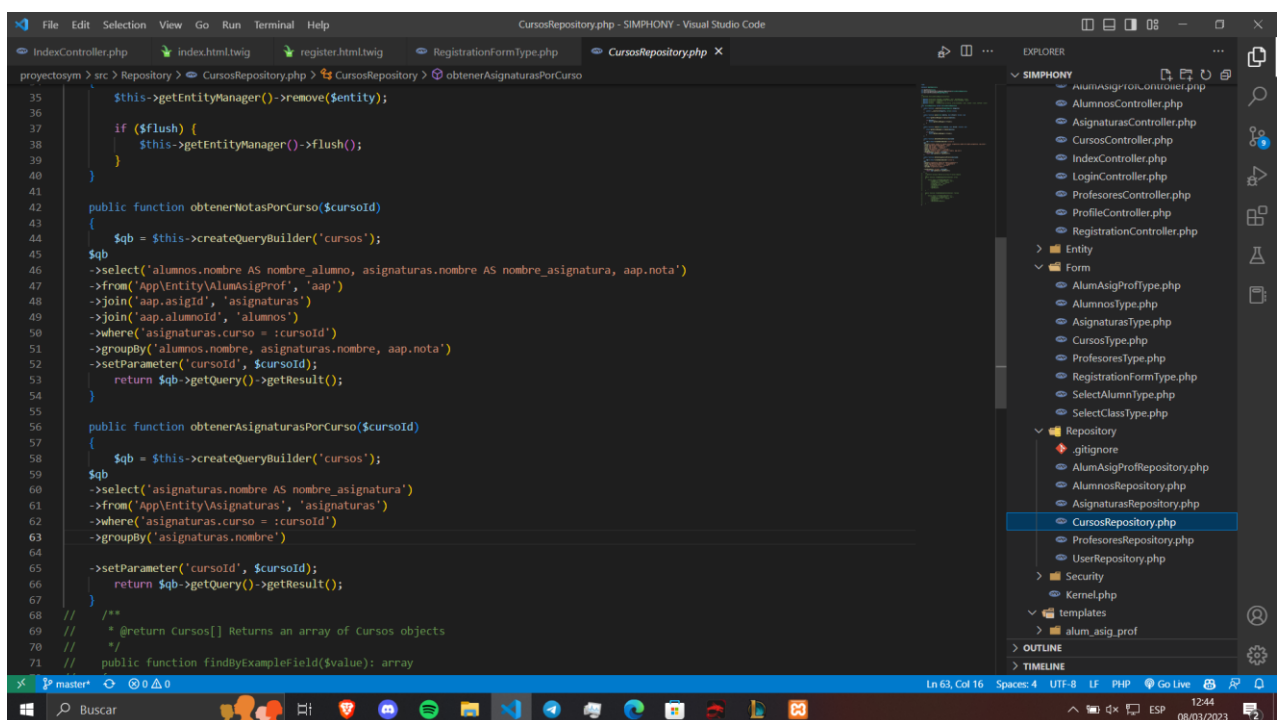


Los formularios se enlazan a las entidades y permiten mostrar los datos de las tablas, de tal forma que solo nos permita seleccionar valores ya existentes en la base de datos, a la hora de crear inserciones dependientes de estos



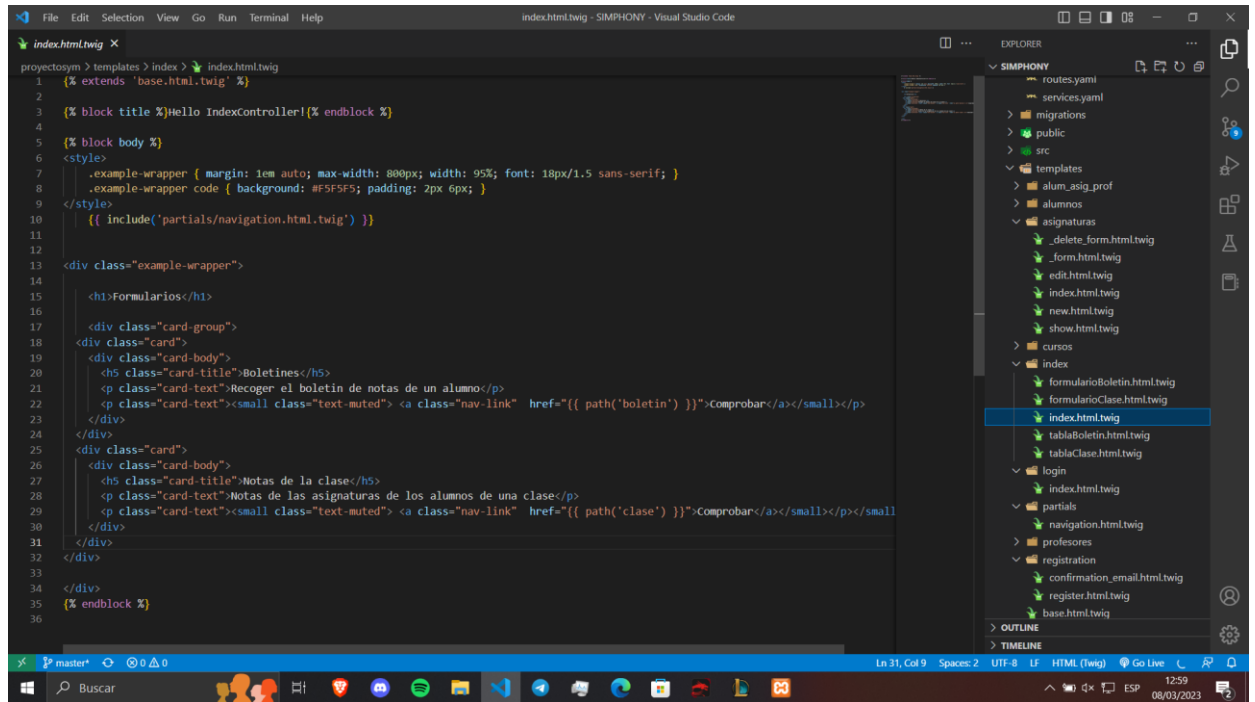
```
21 {
22     $builder
23     ->add('email', EmailType::class, [
24         'attr' => ['autocomplete' => 'email', 'class' => 'form-control'],
25         'constraints' => [
26             new NotBlank([
27                 'message' => 'Please enter an email',
28             ]),
29             new Length([
30                 'min' => 6,
31                 'minMessage' => 'Your email should be at least {{ limit }} characters',
32                 // max length allowed by Symfony for security reasons
33                 'max' => 4096,
34             ]),
35         ],
36     );
37     ->add('agreeTerms', CheckboxType::class, [
38         'mapped' => false,
39         'attr' => ['class' => 'form-check-input'],
40         'constraints' => [
41             new IsTrue([
42                 'message' => 'You should agree to our terms.',
43             ]),
44         ],
45     );
46     ->add('plainPassword', PasswordType::class, [
47         // instead of being set onto the object directly,
48         // this is read and encoded in the controller
49         'mapped' => false,
50         'attr' => ['autocomplete' => 'new-password', 'class' => 'form-control'],
51         'constraints' => [
52             new NotBlank([
53                 'message' => 'Please enter a password',
54             ]),
55             new Length([
56                 'min' => 6,
57                 'minMessage' => 'Your password should be at least {{ limit }} characters',
58                 // max length allowed by Symfony for security reasons
59                 'max' => 4096,
60             ]),
61         ],
62     );
63 }
```

Se puede ver que el registro cuenta con diferentes tipos de validaciones que aseguran un mayor nivel de seguridad



```
35 $this->getEntityManager()->remove($entity);
36
37 if ($flush) {
38     $this->getEntityManager()->flush();
39 }
40
41
42 public function obtenerNotasPorCurso($cursoId)
43 {
44     $qb = $this->createQueryBuilder('cursos');
45
46     ->select('alumnos.nombre AS nombre_alumno, asignaturas.nombre AS nombre_asignatura, aap.nota')
47     ->from('App\Entity\AlumAsigProf', 'aap')
48     ->join('aap.asigId', 'asignaturas')
49     ->join('aap.alumnoId', 'alumnos')
50     ->where('asignaturas.curso = :cursoId')
51     ->groupBy('alumnos.nombre, asignaturas.nombre, aap.nota')
52     ->setParameter('cursoId', $cursoId);
53     return $qb->getQuery()->getResult();
54 }
55
56 public function obtenerAsignaturasPorCurso($cursoId)
57 {
58     $qb = $this->createQueryBuilder('cursos');
59
60     ->select('asignaturas.nombre AS nombre_asignatura')
61     ->from('App\Entity\Asignaturas', 'asignaturas')
62     ->where('asignaturas.curso = :cursoId')
63     ->groupBy('asignaturas.nombre')
64
65     ->setParameter('cursoId', $cursoId);
66     return $qb->getQuery()->getResult();
67 }
68
69 /**
70  * @return Cursos[] Returns an array of Cursos objects
71  */
72 public function findByExampleField($value): array
```

En lo que respecta a consultas, en cada repositorio asociado a una entidad podemos diferentes métodos básicos, sin embargo, se han implementado varios métodos que permiten, entre otras cosas, obtener las notas individuales de cada alumno y por asignatura



```
1 {% extends 'base.html.twig' %}
2
3 {% block title %}Hello IndexController!{% endblock %}
4
5 {% block body %}
6 <style>
7   .example-wrapper { margin: 1em auto; max-width: 800px; width: 95%; font: 18px/1.5 sans-serif; }
8   .example-wrapper code { background: #f5f5f5; padding: 2px 6px; }
9 </style>
10 {{ include('partials/navigation.html.twig') }}
11
12
13 <div class="example-wrapper">
14
15   <h1>Formularios</h1>
16
17   <div class="card-group">
18     <div class="card">
19       <div class="card-body">
20         <h5 class="card-title">Boletines</h5>
21         <p class="card-text">Recoger el boletin de notas de un alumno</p>
22         <p class="card-text"><small class="text-muted"> <a class="nav-link" href="{{ path('boletin') }}">Comprobar</a></small></p>
23       </div>
24     </div>
25     <div class="card">
26       <div class="card-body">
27         <h5 class="card-title">Notas de la clase</h5>
28         <p class="card-text">Notas de las asignaturas de los alumnos de una clase</p>
29         <p class="card-text"><small class="text-muted"> <a class="nav-link" href="{{ path('clase') }}">Comprobar</a></small></p></small>
30       </div>
31     </div>
32   </div>
33
34 </div>
35 {% endblock %}
36
```

En las vistas se puede ver cómo se ha usado herencia para ahorrar en código y facilitar la legibilidad del contenido, tanto a la hora de implementar la cabecera como los formularios en aquellas vistas en las que se requieran